Wi-Fi сети: векторы атаки и Kali Linux

Режимы работы Wi-Fi адаптера

Для начала нужно перевести наш беспроводной адаптер (сетевую карту) в «хакерский режим» — **monitor mode**.

Дело в том, что каждый Wi-Fi адаптер, а точнее, антенна, на физическом уровне улавливает любые сигналы, пересылаемые устройствами в радиусе действия. Антенна не может «не принимать» посторонние пакеты. А вот драйвер может работать в трёх режимах (на самом деле 6, но это за рамками данной статьи):

1. **Client mode** (также **managed mode**) — пакеты, не предназначенные этому адаптеру — отбрасываются, а остальные передаются внутрь ОС как «полученные». В этом режиме повреждённые пакеты также отбрасываются. Нормальный режим работы «без свистелок».
2. **Monitor mode** (также **rfmon mode**) — драйвер не фильтрует пакеты и передаёт всё, что улавливает антенна, в ОС. Пакеты с неверной контрольной суммой *не* отбрасываются и их можно видеть, к примеру, в **Wireshark**.
3. **Promiscuous mode** («беспорядочный режим») — режим монитора «наполовину». Драйвер будет передавать в ОС пакеты, полученные в рамках сети, к которой мы сейчас подключены (associated), но в отличие от обычного режима не будут отбрасываться пакеты, предназначенные другим клиентам этой сети. Пакеты других сетей будут игнорироваться. Понятно, что это работает только когда вы можете успешно подключиться и авторизоваться в некоторой сети (открытой или нет). В отличии от монитора, этот режим поддерживается меньшим числом адаптеров. При работе в этом режиме, равно как и в режиме клиента, драйвер будет убирать из переданных в ОС пакетов низкоуровневые заголовки канала.

Нам нужен как раз режим монитора — в режиме клиента нам не интересно смотреть на пакеты, которые нам самим и предназначены, а для режима promiscious нам нужно сперва подключиться к некоторой сети. Если не считать повреждённых пакетов и наличия заголовков 802.11, режим монитора может то же и больше, что и беспорядочный режим, да и поддерживается львиной долей адаптеров. Единственная проблема — не все адаптеры могут одновременно передавать данные в режиме монитора, но у меня лично с этим проблем не было.

Открываем терминал и выполняем:

```
airmon-ng start wlan0
```

wlan0 — идентификатор устройства-адаптера. В *nix это **wlan** + порядковый номер (**wlan0, wlan1, wlan2** и т.д.). Номер адаптера можно узнать, выполнив **ifconfig** или **iwconfig** (эта выведет только устройства беспроводных адаптеров и специфичную для них информацию).

Если **airmon-ng** выведет сообщение `(monitor mode enabled on mon0)` — значит, адаптер успешно переведён в режим наблюдения.

Обратите внимание, что в командах, которые мы будем выполнять дальше, тоже указывается идентификатор адаптера, но виртуального — **mon0** (`mon1`, etc.), а не исходного — **wlan0**. **mon0** — адаптер, созданный **airmon-ng**, предназначенный для работы с функциями монитора.

Беспроводные каналы

Wi-Fi — технология передачи данных по радиоканалу, ~~то есть даром~~ с использованием того же механизма, что и в радиостанциях. И точно так же, как в радиостанциях, если передавать слишком много данных на одной частоте (обычно 2.4 ГГц), то нельзя будет разобрать, что к чему относится. Для этого существует разделение на каналы.

Всего каналов 13, хотя последний канал — тринадцатый — ~~несчастливый и~~ в некоторых странах запрещён, поэтому чаще всего используются 1-12. Однако 13-й можно включить, выбрав для беспроводного адаптера такую страну, как Боливия (BO) — или любую другую, где данная частота разрешена. Да, техническая возможность, как это часто бывает, есть в любом адаптере, но заблокирована по этическим предпочтениям.

Кроме этого есть и 14-й канал (японцы, как всегда, отличились) и частота 5 ГГц с ещё 23 каналами. Вообще, каналы 2.4 ГГц частично пересекаются, плюс их ширина может быть 20 и 40 МГц. Эта тема запутанная из-за наличия разных версий стандарта — кому интересно могут почитать в Википедии. Нам хватит и того, что уже сказано. Пока ещё 5 ГГц применяются реже, но описанные далее приёмы применимы и к этой частоте.

Некоторые используют 13-й канал для скрытия своих сетей, но этот способ здесь даже не рассматривается, так как достаточно перевести адаптер в нужный регион, как он сам их увидит. Например:

```
ifconfig wlan0 down
iw wlan0 reg set BO
ifconfig wlan0 up
iwconfig wlan0 channel 13
```

Любой беспроводной адаптер может принимать и передавать данные только на одном канале за раз. Однако текущий канал можно менять сколь угодно часто — для получения полной информации о беспроводных передачах вокруг **airodump-ng** (~~после рекламы~~ ниже) переключает канал несколько раз в секунду и ловят всё, что попадается. Это называется *channel hopping* («прыжки по каналам»). Это делают и системные программы — например, `NetworkManager` для GNOME, который показывает список беспроводных сетей в правом верхнем углу.

Если канал не зафиксирован, то некоторые пакеты могут быть потеряны — адаптер переключился на соседний канал, а в это время по предыдущему каналу прошёл новый пакет, но его антенна уже не уловила. Это критично, когда вы пытаетесь перехватить handshake, поэтому для отключения всех программ, которые используют беспроводной адаптер и не дают зафиксировать канал, используется эта команда:

```
airmon-ng check kill
```

Без **kill** будет выведен список всех подозрительных процессов, а с **kill** они будут завершены. После этого адаптер останется целиком в вашем распоряжении. Команду выше рекомендуется выполнять перед любыми действиями, кроме простого обзора сети, так как беспроводной адаптер — общий ресурс; несколько программ могут использовать его одновременно (например, можно ловить список сетей в **airodump-ng** и одновременно подбирать WPS в **reaver**), но любая из них может переключить канал, поэтому важно зафиксировать его и при её запуске (обычно параметр называется **-c** или —**channel**).

Antenna overclock

Кроме манипуляций с регионом **ifconfig** может попытаться заставить адаптер работать на большей мощности, чем он это делает по умолчанию. Результат сильно зависит от типа адаптера и региона, при долгом использовании может испортить устройство ~~и вообще поджарить вам нос~~ , так что используйте на свой страх и риск. В случае успеха антенна сможет улавливать более слабые сигналы.

```
ifconfig wlan0 down
# Обход региональных ограничений на максимальную мощность передатчика.
iw reg set BO
iwconfig wlan0 txpower 500mW
# Либо:
iwconfig wlan0 txpower 30
ifconfig wlan0 up
```

Обычная мощность — 15-20 дБм. При ошибке будет сообщение типа `Invalid argument`, однако его может и не быть — после выполнения проверьте значение *txpower* в **iwconfig**.

Выполняем:

```
airodump-ng mon0
```

airodump-ng — команда для сбора пакетов в радиоэфире. Она выводит в консоль две таблицы. В верхней выводятся найденные беспроводные сети, в нижней — клиенты, подключенные к ним, или не подключенные, но с активным беспроводным адаптером, транслирующем какие-то пакеты (например, о поиске сети с определённым именем).

Столбцы первой таблицы с доступными беспроводными сетями:

1. **BSSID** — уникальный MAC-адрес беспроводной сети. По аналогии с MAC-адресом сетевых карт, это 6 чисел в шестнадцатеричном формате, разделённых двоеточием, например: `AA:00:BB:12:34:56`. Он передаётся в большинство других команд.
2. **PWR** — уровень сигнала. Это отрицательное число; чем оно ближе к 0 — тем сигнал сильнее. Обычно для комфортной работы это число до -50, для видеосвязи — до -65, для VoIP — до -75. Значения ниже -85 и в особенности ниже -90 можно считать крайне слабым уровнем. Число зависит как от мощности передатчика, так и от коэффициента усиления антенны в вашем адаптере (внешние адаптеры имеют усиление 0-12 dB, внешние 1-2-метровые всенаправленные антенны — до 24 dB)
3. **Beacons** — число переданных этой точкой доступа «маячков» — пакетов, оповещающих находящихся рядом устройств о существовании этой беспроводной сети, уровне сигнала, её имени (BSSID/ESSID) и прочей информации. Используется для подключения. По умолчанию точки доступа обычно настроены на передачу маячков каждые 100 мсек (10 раз в секунду), но интервал можно увеличить до 1/сек. Отсутствие маячка не говорит об отсутствии беспроводной сети — в скрытом (hidden) режиме точка доступа не передаёт маячков, но к ней можно подключиться, если знать точное имя сети. О способе обнаружения таких сетей — ниже.
4. **#Data** — число пакетов с данными, которые пришли от этой точки доступа. Это может быть HTTP-трафик, ARP-запросы, запросы на авторизацию (handshake) и прочее. Если к сети не подключен ни один клиент или если он ничего не передаёт, то это значение не меняется и может быть 0.
5. **#/s** — число пакетов с данными в секунду. **#Data** делённое на время наблюдения за этой сетью.
6. **CH** — номер канала. Как уже было описано выше, весь доступный спектр Wi-Fi разделён на 14 каналов; точка доступа и, соответственно, клиенты, передают данные на определённом канале и этот столбец указывает, к какому каналу привязана эта точка доступа и её клиенты
7. **MB** — скорость передачи (ширина канала) в Мбит/с. Точка в конце обозначает, что точка доступа поддерживает короткую преамбулу (short preamble). Можно увидеть значения 11, 54, 54e. Нас это обычно мало волнует
8. **ENC** — тип беспроводной сети — OPN (открытая), WEP, WPA, WPA2. На основании этого параметра мы выбираем подходящую схему атаки.
9. **CIPHER** — тип шифрования данных после handshake. Может быть TKIP и CCMP (см. обзор в первой части).
10. **AUTH** — механизм аутентификации для передачи временного ключа. Может быть PSK (обыная авторизация по единому паролю для WPA(2)), MGT (WPA(2) Enterprise с отдельным сервером с ключами RADIUS), OPN (открытая).
11. **ESSID** — имя беспроводной сети. Именно его вы видите в «Диспетчере беспроводных сетей» в Windows и указываете в настройках точки доступа. Так как это пользовательское имя, то оно может не быть уникальным, и для всех внутренних

Операций используется BSSID (то есть MAC-адрес адаптера в точке доступа), а это — просто отображаемое название.

Иногда в некоторых столбцах можно увидеть числа **-1**, а в последнем столбце — `<length: 0>`. Это признаки беспроводной сети, которая не транслирует свои данные в открытом виде, а отвечает лишь когда клиент сделал явный запрос на подключение с указанием корректного ESSID и пароля. Кроме этого, точка доступа может вообще не транслировать маячки и станет активна только, когда к ней подключится клиент, знающий её имя.

Если оставить **airodump-ng** запущенным достаточно долгое время и если в этом промежутке к скрытой сети подключится новый клиент, то строка, соответствующую «скрытой» сети, будет автоматически раскрыта и там появится номер канала, ESSID и данные о защите. При этом может быть нужно зафиксировать канал, чтобы не упустить момент подключения (см. ниже).

Столбцы второй таблицы с беспроводными клиентами:

1. **BSSID** — MAC-адрес точки доступа, к которой подключен клиент (см. первую таблицу). Если указано **(not associated)** — клиент отключен от всех сетей, но адаптер работает (возможно, он ищет доступные сети).
2. **STATION** — MAC-адрес клиента. Когда-то эти адреса были вшиты в адаптер на фабрике и не могли быть изменены, но сегодня может быть настроен в подавляющем большинстве случаев. В Linux/Mac для этого есть штатные средства., в Windows с этим сложнее и поддержка зависит от драйвера. MAC-адрес точно так же, как и в проводных сетях, передаётся буквально в каждом пакете от этого клиента и это основная причина, почему фильтрация по MAC почти бесполезна.
3. **PWR** — уровень сигнала от клиента. Чем ближе к 0, тем клиент ближе/сигнал мощнее (см. первую таблицу).
4. **Rate** — когда **airodump-ng** запущен с фиксацией канала (см. ниже), этот столбец покажет частоту передачи пакетов с данными от точки доступа к клиенту (слева от дефиса) и от клиента обратно (справа).
5. **Lost** — число потерянных пакетов, которые наша система (не клиент) не зарегистрировала. Это легко вычислить, так как в передаваемых пакетах есть счётчик.
6. **Frames** или **Packets** — число пакетов с данными, которые мы уловили от этого клиента (см. **#Data** в первой таблице).
7. **Probe** — список ESSID-имён беспроводных сетей, к которым клиент пытался подключиться. Здесь могут быть перечислены совсем не те сети, которые вы видите вокруг, а те, к которым клиент подключался ранее, или же скрытые сети. На основе этого можно организовать атаку типа **Karma**, как упомянуто ранее.

Параметры запуска airodump-ng (можно комбинировать; более детально в следующих частях):

- `airodump-ng -c 3 mon0` — зафиксировать канал №3 — полный приём пакетов без потерь при переключении на другие каналы (из-за перекрывающихся частот в список могут попасть соседние каналы).
- `airodump-ng -w captures.pcap mon0` — записывать все принятые пакеты в файл `captures.pcap` — используется для offline-атаки на перебор пароля WEP/WPA (будет освещено в следующей части).
- `airodump-ng --essid "Имя сети" mon0` — фильтровать принимаемые (записываемые/отображаемые) пакеты по принадлежности к заданному имени сети. Обычно используется для уменьшения размера файла, так как на качество перехвата это никак не влияет.
- `airodump-ng --bssid 01:02:03:AA:AA:FF mon0` — фильтрация по MAC-адресу точки доступа (BSSID). Аналогично **--essid**.

Каждый подключенный клиент общается с базовой станцией по её BSSID — и это именно то, что мы видим в обеих таблицах **airodump-ng**. Мы можем «отключить» клиента от сети, после чего он должен будет подключиться вновь — и в этот момент **airodump-ng** перехватит рукопожатие со всеми идентификаторами и ключами.

Либо мы можем просто оставить ноутбук включенным на пару часов с **airodump-ng** на нужном канале и подождать. Кстати, при успешном «вскрытии» в правом верхнем углу появится сообщение вида `[ Decloak: 00:00:11:11:22:22 ]` с BSSID точки доступа, конспирацию которой мы раскрыли.

Отключение клиентов предусмотрено во всех беспроводных стандартах и делается с помощью **aireplay-ng** (все три уже знакомые нам утилиты — часть проекта Aircrack-ng, который содержит в своём наборе всевозможные инструменты для работы с тонкими материями):

```
aireplay-ng wlan0 --deauth 5 -a AP_BSSID -c CLIENT_BSSID
```

Внимание: эта команда — исключение и принимает идентификатор *реального* беспроводного адаптера, а не **mon0**, созданного с помощью **airmon-ng**.

Если при запуске появилась ошибка о незафиксированном канале и/или **airodump-ng** показывает в правом верхнем углу `[ fixed channel -1 ]` — значит, какая-то программа или сервис заставляет адаптер перескакивать с канала на канал (это может быть тот же **airodump-ng**) и их нужно закрыть с помощью `airmon-ng check kill`, как было описано в начале.

Команда выше имитирует ситуацию, когда точка доступа сообщает клиенту, что старый ключ недействительный (и что его следует обновить, повторив handshake, то есть передав пароль и имя сети заново). На адрес клиента отправляются сообщения якобы от точки доступа о том, что следует отключиться от сети и обновить данные сессии.

Так как этот тип пакетов не защищён шифрованием (то есть команда может быть выполнена ещё до аутентификации), то противодействовать ему невозможно по той простой причине, что невозможно установить «личность» того, кто это рассылает — MAC-адреса в пакетах поддельные и установлены в те значения, которые мы передали после **-a** и **-c**. Можно только отслеживать слишком частые отключения и принимать какие-то меры.

так, допустим во второй таблице **airodump-ng** мы видим следующую строчку:

```
BSSID              STATION            PWR   Rate    Lost    Frames  Probe
4F:B1:A4:05:5C:21  5B:23:15:00:C8:57  -54    0 - 1      0     1266  homenet,
XCom
```

Первый столбец — значение для **-a** (MAC базовой станции), второй — для **-c** (MAC клиента). Перезапустим **airodump-ng** и зафиксируем его на канале 5 (это канал нашей атакуемой станции), после чего выполним отключение клиента в соседнем окне терминала:

```
airodump-ng -c5 mon0
aireplay mon0 -0 5 -a 4F:B1:A4:05:5C:21 -c 5B:23:15:00:C8:57
```

Если сигнал достаточно сильный, число пакетов велико и клиент/точка доступа нас услышали — они отключатся друг от друга, затем передадут информацию о сети вновь при подключении и **airodump-ng** покажет их и строчку `[ Decloak ]`.

В самом деле, каждый клиент — подключенный к сети или нет — раскрывает свой MAC-адрес при передаче любого пакета. В таблице **airodump-ng** эти адреса видны в столбце **STATION**. Соответственно, как только мы увидели сеть, куда нам почему-то не попасть (Linux обычно рапортует об `Unspecified failure` на этапе подключения, а Windows долго думает, после чего сообщает о неизвестной ошибке), и тут же в списке видим связанного с этой сетью клиента — мы можем взять его MAC-адрес и поменять свой на него. Последствия могут быть разными ~~и может даже стукнуть~~. Если сделаем это при активном клиенте — в сети появится клон и клиента может «заглючить», так как он будет получать ответы на пакеты, которые он не отправлял. В итоге может отключиться, а может продолжать работать, предупредив о проблеме пользователя — или нет. Но вариант лучше — записать его MAC и подключится, когда он уйдёт.

В Linux изменить MAC своего адаптера можно следующим образом (работает как для проводных сетей, так и для беспроводных):

```
ifconfig wlan0 down
ifconfig wlan0 hw ether 00:11:22:AA:AA:AA
ifconfig wlan0 up
```

Предварительно нужно отключить любые **mon**-интерфейсы. Проверить, заработала ли подмена, можно вызвав `ifconfig wlan0` — в строке **Hwaddr** должен быть указанный выше MAC.

Кроме того, в *nix есть **macchanger** — с его помощью можно выставить себе случайный MAC. Если поставить его в *init.d*, то неприятель будет совершенно сбит с толку, так как при каждой загрузке наш MAC будет другим (работает для любых проводных и беспроводных адаптеров, как и **ifconfig**).

```
# Случайный MAC:
macchanger -r wlan0
# Определённый MAC:
macchanger -m 11:22:33:AA:BB:CC wlan0
# Показать MAC:
macchanger -s wlan0
```

Reaver

А вот теперь мы проверим, насколько добросовестно ваша точка доступа выполняет свои обязанности. Если у вас нет роутера с WPS — ~~всегда можно протестировать соседский~~ вам эта атака не грозит.

Для эксплуатации уязвимости в WPS у *Kali* есть несколько утилит, но на мой взгляд самая наглядная и гибкая — тот самый **reaver**. Синтаксис вызова:

```
reaver -i mon0 -c 5 -b AP_BSSID -va
```

Кроме этого мы можем указать:

- **-m OUR_MAC** — если вы меняли MAC-адрес своего адаптера (см. выше про MAC-фильтрацию), то укажите этот параметр с новым (не фабричным) MAC.
- **-e ESSID** — при атаке на скрытую сеть **reaver** должен знать её имя, а не только BSSID; если сеть отправляет маячки — ESSID будет автоматически получен из них, но для скрытых сетей вы должны передать его явно.
- **-p PIN** — если вы хотите вручную передать 8-мизначный код WPS, который нужно попробовать применить к этой сети (этот код можно прочитать и изменить в веб-интерфейсе устройства). Если он верный — утилита выведет сам пароль сети.
- **-vv** — для показа принятых/переданных пакетов, в том числе тех самых **M4** и **M6**. Полезно для отслеживания активности при плохом соединении.

reaver перебирает 11 000 комбинаций, пока не обнаружит, что точка доступа приняла одну из них. Скорость перебора сильно зависит от силы сигнала/расстояния до базовой станции и может колебаться от 3 до 30 в секунду. Обычно на одну сеть уходит до 10-15 часов.

Прервать **reaver** можно с помощью **Ctrl+C**, при этом он сохранит текущий прогресс для этой сети и при повторном запуске начнёт с прерванного PIN. Если вы работаете в live-режиме (read only), то данные сессии можно переписать с `/etc/reaver/` на постоянный носитель и при следующей загрузке записать их обратно.

В этой папке хранятся текстовые файлы с именами вида `AP_MAC.wpc` и списком всех номеров для перебора внутри (первая строка — номер строки с PIN, с которого начать перебор при повторном запуске).

Перечисление сетей

Напоследок несколько слов ~~о вечном~~ об альтернативах **reaver**.

reaver будет пытаться атаковать любую сеть, которую вы ему передадите в виде параметра BSSID, но она может не поддерживать WPS — в этом случае программа зависнет над `Waiting for beacon...` и через полминуты сообщит о том, что сеть не обнаружена.

Проверить, какие сети в округе поддерживают WPS, можно несколькими способами.

- Некоторые версии **airodump-ng** имеют отдельный столбец **WPS**, который заполнен для сетей, поддерживающих эту технологию (там даже указана модель роутера)
- **wash -i mon0** выводит удобно отформатированную таблицу всех WPS-enabled сетей, при этом продолжая работать, пока не будет явно закрыт через **Ctrl+C**. Очень полезен и тем, что сообщает, когда сеть поддерживает WPS, но возможность заблокирована (см. про rate limiting выше)
- **iwlist wlan0 scan** выводит множество информации по всем доступным в радиусе сетям, в том числе расширенные данные поддержки WPS с именами устройств
- **wifite** — автоматический сканер сетей и универсальный взломщик; однако на мой взгляд он слишком универсальный, поэтому сложно понять, что он делает и на какой стадии этот процесс находится. А вот вывод сетей у него очень удобный: показано наличие подключенных клиентов и поддержка WPS

Кроме того, есть очень любимый многими **kismet** (и аналог для Macintosh — **KisMAC**), но на мой взгляд он слишком навороченный. В обоих есть интеграция с GPS, что может быть полезно для тех, кто исследует сети на большой территории (wardriving).