Пассивные сетевые атаки

Пассивные сетевые атаки. Обнаружение хостов и сервисов. Определение ОС, nmap, zetmap.

[Пассивные сетевые атаки](#_obvybbll4bdj)

[Сканирование сети](#_bkbhetvvflhf)

[Обнаружение хостов](#_d0wq3pymk6va)

[arp-пинг](#_e72clm4h6gw0)

[ping-scan](#_bobpb6jzhn14)

[tcp-сканирование](#_jxhfdl7a6q88)

[Сканирование портов](#_tw7b0hfqhwfc)

[telnet](#_tiuzcfybyi34)

[TCP-connect](#_lvr3ixo7zp8t)

[TCP-SYN](#_aljb5o93fnhi)

[Менее точные методы](#_xcvr9fjeqgye)

[FIN-сканирование](#_ha1p95m5bdpy)

[Ack-сканирование](#_qn886ro1qmk4)

[Tap ident](#_epys6vu1ocxj)

[UDP-сканирование](#_s4pghoolc01j)

[Методы «невидимого» удаленного сканирования](#_4iywcntpks5r)

[numb host scan](#_1fmhc4f0q9zo)

[Сканирование сервисов](#_9vpg3u766uu)

[Еще примеры сканирования](#_ac6rcyttpceq)

[Уязвимость в настройках MySQL](#_2pu85ct9n7yc)

[SSL heartbleed](#_svri6og8ouog)

[Zmap (zetmap)](#_t2971lx1j4os)

Практическое [задание](#_3tfrjxxltv85)

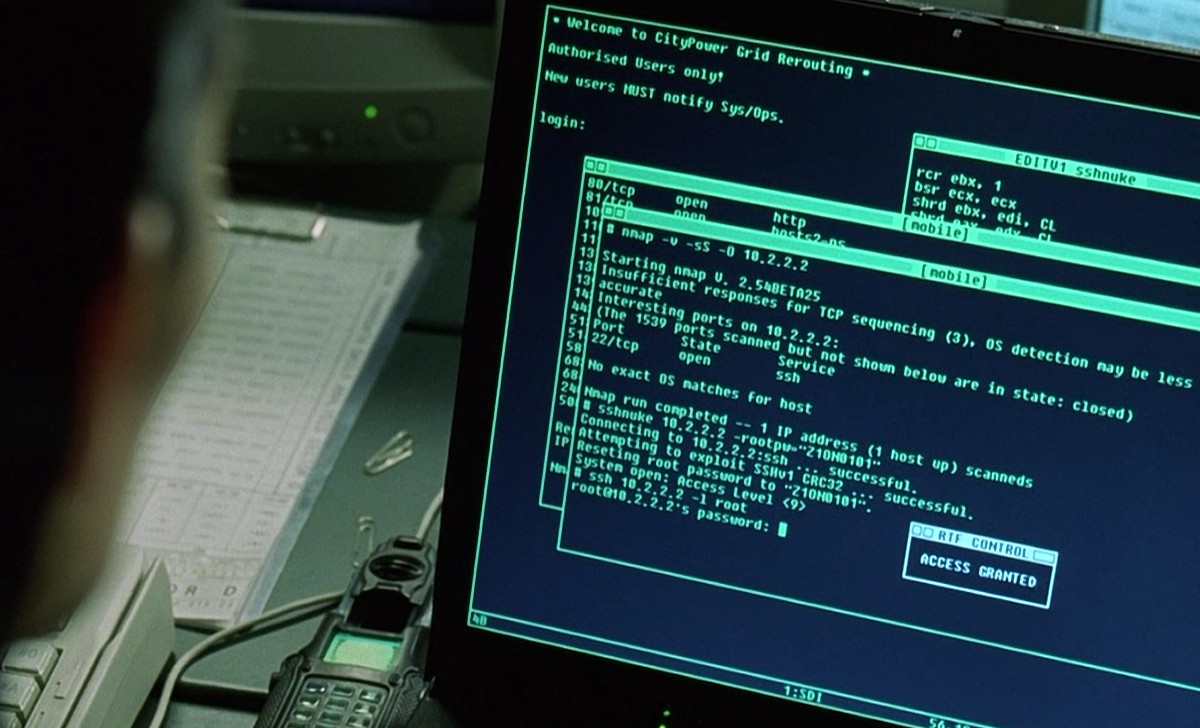
[Дополнительные материалы](#_jeeoh8oz8lfe)

# 

# Пассивные сетевые атаки

Задача пассивных сетевых атак — просканировать в сети хосты и сервисы, не привлекая внимания. В зависимости от выбранного способа в логах жертвы останутся следы активности. Они могут быть незначительными, а если использовать **numb host scan**, будут вести не на машину злоумышленника.

## Сканирование сети



Сканирование выявляет активные компьютеры сети.

Способы обнаружить хосты:

* ICMP;
* ARP;
* RST-сегменты TCP;
* Ответы на несуществующие DNS-запросы;
* Прослушивание трафика и т.д.

**nmap** — это утилита для сканирования, которая без параметров выводит справку.

|  |
| --- |
| nmap |

Запускаем **nmap** и **Wireshark**: они будут работать в паре, чтобы мы могли полностью понять, что происходит на канальном, сетевом и транспортном уровне при атаках.

|  |
| --- |
| Wireshark>capture filter> |

Как тестовый стенд для обучения можно использовать **scanme.nmap.org** — это узел, созданный разработчиками **nmap** специально для тестирования. Можно экспериментировать абсолютно легально.

В качестве компьютера-злоумышленника воспользуемся **Kali Linux**, а в роли жертв выступят две машины: с Windows 7 и DVL.

Формат вызова:

|  |
| --- |
| nmap (адрес хоста) |

Разные примеры вызова, с их помощью можно сканировать перечни и списки адресов:

|  |
| --- |
| nmap 192.168.66.1 192.168.66.2 192.168.66.3 |

|  |
| --- |
| nmap 192.168.66.1-192.168.66.3 |

Можно сканировать целиком всю сеть:

|  |
| --- |
| nmap 192.168.66.0/24 |

Указав ключ **nmap -sL**,перед сканированиемможно вывести список хостов, которые должны пройти эту процедуру.

|  |
| --- |
| nmap -sL 192.168.66.1-20 |

Обратите внимание в **Wireshark**, что **nmap** по умолчанию для каждого IP-адреса запрашивает доменное имя в системе DNS. Информация об атакующем хосте может остаться в логе DNS-сервера, обслуживающего PTR-записи сети. Обращение к PTR-записям DNS-сервера надо сразу отключить с помощью ключа **-n**.nn

Пример:

|  |
| --- |
| nmap -sL -n 192.168.66.1-20 |

Выведется только список IP-адресов для сканирования, без доменных имен.

Так как **nmap** сканирует долго и перебором, часто его используют с **zetmap**. С ним сканирование идет быстрее: генерируется список хостов, которые **nmap** принимает на вход и исследует.

## Обнаружение хостов

Популярный способ обнаружить хосты в сети — ICMP, но его можно отключить. Просканировать хосты надежно и бесшумно помогут ARP-запросы.

#### arp-пинг

Сканируем с помощью ARP:

|  |
| --- |
| nmap -PR 192.168.66.0/24 |

Фильтруем в Wireshark. Выделим только запросы, чтобы понять, что отправляет **nmap**:

|  |
| --- |
| arp.opcode==1 |

И ответы, чтобы понять, какие узлы есть в сети:

|  |
| --- |
| arp.orpcode==2 |

### ping-scan

Arp-сканирование действует только локально, если вы находитесь в том же сегменте сети. Для других сетей нужно использовать **ping scan**:

|  |
| --- |
| nmap -n -sP --send-ip 192.168.66.0/24 |

В локальной сети будет применяться и **arp**, и **icmp**.

Не используется в чистом виде, если на удаленных машинах сетевым фильтром разрешен **icmp**.

### tcp-сканирование

Для TCP-сканирования необходимо работать от **root**, так как для доступа к сырым сокетам (**raw socket**) требуются права суперпользователя.

При запуске с теми же ключами еще делаются запросы на 80 и 443. Ведь если **icmp** закрыт сетевым фильтром, сканер попытается выявить хосты по наиболее вероятно открытым портам. А самый распространенный сервис в интернете — это HTTP/HTTPS.

Если TCP-соединение установлено, то узел в сети, если нет, то скорее нет, чем да.

Можно просканировать общеизвестные TCP-порты. Если в сети есть узел, у которого ssh-сервис использует порт 27574, нам будет казаться, что он не в сети.

Но можно заставить **nmap** сканировать все порты:

|  |
| --- |
| nmap -Pn 192.168.66.0/24 |

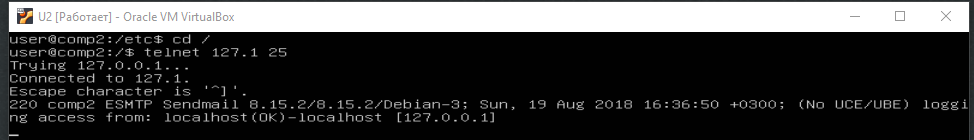
**Nmap** будет считать, что все порты в сети, и сканировать их.

## Сканирование портов

#### telnet

Это самый простой способ сканировать порты. Telnet-клиент похож на реализацию сырого tcp-клиента, хотя не идентичен ему, и содержит telnet команды. Позволяет подключиться на нужный порт, и если соединение удалось установить, значит сервис присутствует на сканируемом узле.

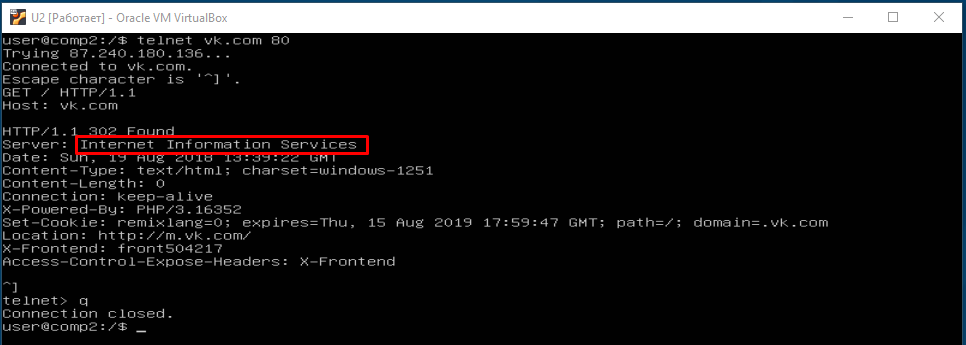
|  |
| --- |
| telnet 127.0.0.1 25 |



Многие сервисы ПО предоставляют информацию о своей версии и операционной системе. Это значительно упрощает жизнь злоумышленнику. А вот vk.com не раскрывает ни имя, ни версию веб-сервера.

Придется не только подключиться, но и сформировать простейший HTTP-запрос. HTTP-сервер не приветствует нас, в отличие от SMTP.

|  |
| --- |
| telnet vk.com 80 GET / HTTP/1.1 Host: vk.com |



Не будем всерьез считать, что vk.com работает на Internet Information Services. Это может быть nginx, varnish или apache.

Выходим из **telnet**:

|  |
| --- |
| ^] q |

Этот метод оставляет много следов, так что рассмотрим альтернативные.

### TCP-connect

|  |
| --- |
| nmap -n -sT 192.168.66.20 |

Посмотрим открытые порты. Для этого в Wireshark укажем следующее:

|  |
| --- |
| tcp.port==135 tcp.port==445 |

Если порт закрыт, в ответ мы получим **RST,ACK**.

**Nmap** использует статистику часто используемых портов. Как правило, в работу берутся 1000 наиболее популярных.

### TCP-SYN

Этот метод не регистрируется в журнале, потому что не посылает последний **ACK**:

|  |
| --- |
| nmap -n -sS 192.168.66.20 |

В Wireshark:

|  |
| --- |
| tcp.port==445 |

Видим **syn**, **syn.ack**, **rst**.

Метод требует привилегий: его не всегда получится использовать на машине, где **nmap** есть, а рутовых прав нет.

### Менее точные методы

#### FIN-сканирование

|  |
| --- |
| nmap -n -sF |

**Nmap** посылает не **syn**, а **fin**, и ждет реакцию определенной ОС. Стандарту **RFC** никто в полной мере не следует, и по тому, какие пункты RFC выполняются, мы можем узнать ОС. Так различают операционные системы по реакции на запросы. Такие способы позволяют определить ОС, когда сами сервисы не сообщают сведения о ПО или дезинформируют.

Windows XP на fin-сканирование не реагирует, а Linux некоторых версий откликается.

#### Ack-сканирование

Метод сканирования, где вместо **fin** — **ack**. Узнаем, открыт ли порт и версия ОС:

|  |
| --- |
| nmap -n -sA |

#### Tap ident

Устаревший и неактуальный метод, но изредка встречается. В Linux-машинах работала служба, которая сама рассказывала, какие порты активны, выдавая информацию о себе. Сейчас ее нет, но в старых версиях линукса можно найти.

## UDP-сканирование

Отличается от TCP, так как UDP — это протокол без состояния: пакет ушел и дошел ли — не известно. Но на UDP-порту тоже может быть сервер. И здесь нам снова помогают стандарты.

Если UDP-порт закрытый, **icmp** посылает icmp-сообщение: **port unreachable**.

Количество icmp в единицу времени ограничено: на Linux — 64 в секунду. Чем реже мы будем посылать запросы, тем более точных результатов сможем добиться. И по этой же причине этот метод сканирования может быть менее надежным:

|  |
| --- |
| nmap -n -sU 192.168.66.20 |

Выводит:

|  |
| --- |
| open|filtered |

То есть **nmap** не может достоверно определить, открыт порт или фильтруется, так как **ICMP unreachable** не пришел.

Чтобы повысить точность, можно использовать тайминги. Для этого предназначен ключ **-t**, который можно выставить как 1–5.

Еще примеры:

|  |
| --- |
| nmap -n -sS -p 80 192.168 |

|  |
| --- |
| nmap -n -sS -p 80,100 192.168 |

**TCP Syn** и **UDP** с указанием портов:

|  |
| --- |
| nmap -n -sS -sU -p T:80,1000m U:53 192.168.66.20 |

Это были методы открытого сканирования, которые оставляют следы.

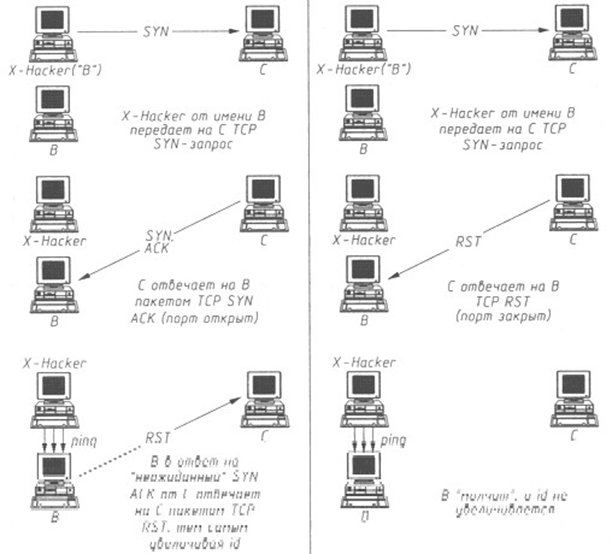
## Методы «невидимого» удаленного сканирования

Наиболее очевидные варианты — анонимизирующие прокси и тор. Но надо понимать, что **exit node** останется открытым, что позволяет отследить злоумышленника.

Есть и другие методы:

* **Dumb host scan**:
  + с каждым переданным пакетом значение ID в заголовке IP-пакета обычно увеличивается на 1;
  + хост отвечает на **TCP SYN** запрос **TCP | SYN | ACK**, если порт открыт, и **TCP RST**, если закрыт;
  + можно узнать количество пакетов, переданных хостом, по параметру ID в заголовке IP;
  + хост отвечает **TCP RST** на **TCP SYN | ACK** и ничего — на **RST**.
* Сканирование через proxy-сервер**.**

### numb host scan

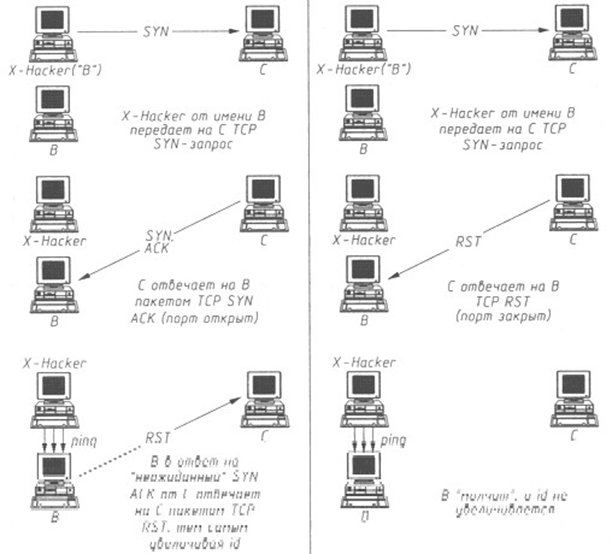


Сканирование с помощью немого хоста (**dumb host scan**) — это старая атака, но работает и сейчас. Зависит от того, какая машина выбрана в качестве немого хоста и как она формирует идентификацию IP-пакетов. Если вместо последовательного приращения используются случайные номера, провести такую атаку не получится.

Атака строится на четырех вещах:

* есть злоумышленник A, узел B — немой хост, и узел С, который хотим просканировать;
* злоумышленник посылает пакет на C от имени B;
* B получает **syn.ack** и высылает **РСТ**;
* мы пингуем B и можем увидеть разницу в ID пакета. Если был **РСТ** — значит +1.

Так как мы должны отследить ID более чем на единицу, нужен хост с минимальными сетевой активностью и потоком трафика. Иначе ID будет расти непредсказуемо и будет трудно понять. Поэтому атака и называется «немой хост».



Для работы понадобится вручную генерировать icmp-пакеты с помощью утилиты **hping3**:

|  |
| --- |
| hping3 192.168.66.20 -r |

В Wireshark увидим, что **identification** изменился на 1.

**TCP** отправляет с 0 порта.

Просканируем **DVL** — посмотрим сетевые интерфейсы и открытые порты:

|  |
| --- |
| ifconfig netstat -nltp |

Далее в качестве немого хоста рассмотрим Windows 7, а как сканируемый узел — DVL.

Формат такой:

hping3 [кого сканируем] -a [исходный адрес] -p [порт] -S (используем SYN-флаг).

|  |
| --- |
| hping3 192.168.88.1 -a 192.168.88.253 -p 3306 -S |

|  |
| --- |
| hping3 192.168.88.1 -a 192.168.88.253 -p 1000 -S |

Смотрим на результаты **hping3**.

Можно сделать ту же атаку с помощью **nmap**, используя 192.168.66.20:

|  |
| --- |
| nmap -Pn -sI 192.168.66.20 192.168.88.251 |

#### Сканирование сервисов

Пример:

|  |
| --- |
| nmap -n -sS -sV scanme.nmap.org |

В этом случае нужно подождать.

**Nmap** распознает сервисы по тому, как они отвечают на запросы: по экспресс-тестам или из заголовков, если сами сервисы дают сведения.

Если на 192.168.88.1 установлен **telnet**, он сам расскажет много интересного:

|  |
| --- |
| telnet 192.168.88.1 |

Такой подход позволяет сканировать ОС и версию ядра.

#### Еще примеры сканирования

##### Уязвимость в настройках MySQL

В **DVL**:

|  |
| --- |
| mysql GRANT ALL PRIVILEGES ON \*.\* TO 'root'@'%' IDENTIFIED BY '' WITH GRANT OPTION; exit |

В **Kali** запускаем **nmap** со скриптом на поиск пустого пароля:

|  |
| --- |
| nmap -sV --script=mysql-empty-password 192.198.66.21 map -p 3306 192.168.88.251 --script mysql-query --script-args='qoery-"select 1;",username-root' |

##### SSL heartbleed

Heartbleed — ошибка, которая существовала в SSL в 2011 году и была обнаружена в 2014. Сейчас версия openssl в основном обновлена, но Heartbleed еще можно найти.

Принцип работы:



*Иллюстрация принципа* [*из «Википедии»*](https://commons.wikimedia.org/w/index.php?curid=35480964)*. Автор Fenix Feather. File:Simplified Heartbleed explanation.svg, CC BY-SA 3.0*

Пример сканирования:

|  |
| --- |
| nmap -p 443 -sV --script=ssl-heartbleed 86.62.90.163 |

На данном ресурсе уязвимость уже закрыта.

## Zmap (zetmap)

Есть более быстрая утилита, чем nmap — **zmap** (произносился как «зетмап»). В отличие от последовательного **nmap**, **zmap** умеет работать параллельно. Не путать с **zenmap**, графической оберткой для **nmap**.

В **Kali** **zmap** не стоит. Устанавливается этой командой:

|  |
| --- |
| apt update&&apt install zmap |

Просканируем произвольные 10 000 хостов по протоколу FTP, результаты запишем в файл **result.csv.**

|  |
| --- |
| zmap --bandwidth=10M --target-port=21 --max-targets=10000 --output-file=result.csv |

|  |
| --- |
| zmap -B 10M -p 21 -o results\_cidr.csv 190.215.0.0/16 |

Если **nmap** делает это последовательно, то **zmap** выполняет **Syn SynAck Rst** многопоточно. Могут теряться пакеты, потому нужно использовать его 2–3 раза для надежности.

**Zmap** работает в паре с **nmap**: ищет список открытых портов, и перечень на вход отправляет **nmap**. Чтобы использовать его со скриптами, действуют так:

|  |
| --- |
| nmap -Pn -n -p 21 --script=ftp-anon -iL resulsts\_cidr.csv -oN anonymous\_ftp\_result.txt |

Результат:

|  |
| --- |
| cat anonymous\_ftp\_result.txt|grep -A 5 -B 5 allowed |

# Практическое задание

1. Выбрать подсеть с маской 24 и просканировать ее по **tcp** по всем портам (65535) с детектированием версий сервисов и ОС. Желательно использовать скрипты **NSE**. Можно взять четыре любых октета в глобальной сети.

Результаты могут быть и не впечатляющими. Не надо искать самую «интересную» сеть. Задача — попрактиковаться и сдать репорт.

# Дополнительные материалы

1. [http://nmap.org/.](http://nmap.org/)
2. <http://nmap.org/book/idlescan.html> (Idle scan).
3. <http://nmap.org/nsedoc/index.html> (scripts).
4. <http://www.shodanhq.com/> и [www.shodan.io](http://www.shodan.io).
5. <https://zmap.io/>.
6. <https://ru.wikipedia.org/wiki/Heartbleed>.
7. <https://events.yandex.ru/lib/talks/2334/>.
8. <https://events.yandex.ru/lib/talks/5531/>.