TCP/UDP/ICMP, IP, Ethernet

Статья про основные сетевые протоколы. Самое важное.

TCP

Попробуем отправить какой-нибудь текст через nc. "Слушатель" - 192.168.101.103, "клиент" - 192.168.101.104.

```
# Устанавливаем соединение
# Syn client -> server (привет, хочу подключиться к тебе)
14:14:32.476957 IP 192.168.101.104.40188 > 192.168.101.103.1234: Flags [S], seq 3022494785,
win 64240, options [mss 1460,sackOK,TS val 1927177028 ecr 0,nop,wscale 7], length 0
# Syn ack server -> client (привет, готов соединиться)
14:14:32.476984 IP 192.168.101.103.1234 > 192.168.101.104.40188: Flags [S.], seq 3324441448,
ack 3022494786, win 65160, options [mss 1460,sackOK,TS val 3590889825 ecr
1927177028,nop,wscale 7], length 0
# ack client -> server (Я понял что ты готов соединиться)
14:14:32.476998 IP 192.168.101.104.40188 > 192.168.101.103.1234: Flags [.], ack 1, win 502,
options [nop,nop,TS val 1927177028 ecr 3590889825], length 0

# Отправляем данные
# Push data + ack client -> server (вот тебе данные, ответь что ты их принял)
14:15:24.138228 IP 192.168.101.104.40188 > 192.168.101.103.1234: Flags [P.], seq 1:3, ack 1,
win 502, options [nop,nop,TS val 1927228689 ecr 3590889825], length 2
# ack server -> client (да, я их принял)
14:15:24.138263 IP 192.168.101.103.1234 > 192.168.101.104.40188: Flags [.], ack 3, win 510,
options [nop,nop,TS val 3590941486 ecr 1927228689], length 0

# Закрываем соединение
# Final, ack client -> server (Был рад пообщаться, пока)
14:40:06.307980 IP 192.168.101.104.39352 > 192.168.101.103.1234: Flags [F.], seq 3, ack 1, win
502, options [nop,nop,TS val 1928710859 ecr 3592422397], length 0
# Final, ack server -> client (Да, давай, пока)
14:40:06.308087 IP 192.168.101.103.1234 > 192.168.101.104.39352: Flags [F.], seq 1, ack 4, win
```

```
510, options [nop,nop,TS val 3592423656 ecr 1928710859], length 0
# Ack client -> server (Услышал тебя. Конец)
14:40:06.308109 IP 192.168.101.104.39352 > 192.168.101.103.1234: Flags [.], ack 2, win 502,
options [nop,nop,TS val 1928710859 ecr 3592423656], length 0
```

Список флагов и их представление в tcpdump:

SYN	S	Флаг, который используется при запросе на соединение.
ACK	.	Используется при подтверждении пришедшего пакета.
FIN	F	Флаг устанавливается при нормальном закрытии соединения.
URGENT	U	Этот флаг нужен при передаче экстренных данных — например, при посылке Ctrl+C в telnet-соединении.
PUSH	P	Как правило, данный флаг устанавливается при передаче пользовательских данных.
RESET	R	Немедленный разрыв соединения.
Заполнитель	none	В случае, если в пакете отсутствует какой-либо флаг используется данный заполнитель.

Из чего состоит пакет:

- Порт src/dst
- Порядковый номер
- Номер подтверждения (ACK SN)
- Длина заголовка
- Флаги (см. таблицу выше)
- Размер окна
- Чексумма
- Уровень важности

UDP

Соединение не устанавливается и не закрывается! Мы просто кидаем данные. UDP не гарантирует, что пакет дойдет, но гарантирует, что пакет будет проверен, в пакете UDP предусмотрена чексумма.

```
# Запустили nc
# *Никакие пакеты никуда не полетели, открытия соединения не предусмотрено протоколом*

# Отправили текст
14:17:48.336527 IP 192.168.101.104.59808 > 192.168.101.103.1234: UDP, length 4

# Закрыли nc клиент
# *Опять же никаких пакетов, т.к. не предусмотрено закрытия соединения*
```

ICMP

Ping не подразумевает передачу данных. Он подразумевает передачу самих пакетов. Т.е. условно дошел-не дошел.

```
# Живой?
14:59:26.330559 IP 192.168.101.104 > 192.168.101.103: ICMP echo request, id 46223, seq 1,
length 64
# Да, живой
14:59:26.330593 IP 192.168.101.103 > 192.168.101.104: ICMP echo reply, id 46223, seq 1, length
64
```

Также можно указать размер пакета в утилите ping. Если мы укажем размер больше Ethernet фрейма (MTU), то будет уже 2 пакета:

```
# MTU = 1500
root@test2:/# ip l show eth0
2: eth0@if7: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP mode DEFAULT
group default qlen 1000
    link/ether 00:16:3e:83:51:d8 brd ff:ff:ff:ff:ff:ff link-netnsid 0

root@test2:/# ping -s 1400 192.168.101.103
PING 192.168.101.103 (192.168.101.103) 1400(1428) bytes of data.
1408 bytes from 192.168.101.103: icmp_seq=1 ttl=64 time=0.042 ms
1408 bytes from 192.168.101.103: icmp_seq=2 ttl=64 time=0.063 ms
1408 bytes from 192.168.101.103: icmp_seq=3 ttl=64 time=0.048 ms

15:01:13.453126 IP 192.168.101.104 > 192.168.101.103: ICMP echo request, id 29861, seq 3,
```

```
length 1480
15:01:13.453152 IP 192.168.101.104 > 192.168.101.103: ip-proto-1
15:01:13.453166 IP 192.168.101.103 > 192.168.101.104: ICMP echo reply, id 29861, seq 3, length
1480
15:01:13.453166 IP 192.168.101.103 > 192.168.101.104: ip-proto-1
```

Датаграмму можно посмотреть тут: https://ru.wikipedia.org/wiki/ICMP

Однако другие типы пакетов ICMP могут нести в себе какие-либо данные. Вот хитрая статья о том как сделать проксирование через ICMP.

IP (IPv4)

Дословно - Internet Protocol, межсетевой протокол. Если L4 (TCP/UDP) доставляет данные, то L3 (IP) объединяет сети и вцелом **организует сетевое пространство**. С его помощью можно понять, как попасть к тому или иному хосту, как маршрутизировать трафик. **IP не гарантирует надежности доставки пакетов**, т.е. они могут продублироваться, повредиться, быть получены не в том порядке, что их отправили или вообще не прийти. За это отвечают протоколы более высокого уровня, например TCP.

Важные компоненты IP пакеты:

- TTL - время жизни пакета. Это количество узлов, которое может пройти пакет. Он полезен для предотвращения зацикливания пакетов и чтобы пакет не искал нужный хост до бесконечности. При прохождении узла число уменьшается на единицу. Однако можно настроить маршрутизатор так, чтобы он не уменьшал TTL. Этой возможностью пользуются для обхода блокировки раздачи интернета через с мобильного телефона, так как некоторые операторы опираются на TTL при выявлении раздачи интернета. Также по этому принципу работае traceroute/tracepath - он отправляет пакеты каждый раз увеличивая TTL, пока не будет отвечать один и тот же хост. Когда приходит пакет с TTL = 0, то получивший его хост отбрасывает пакет обратно с сообщением `Time Exceeded` (ICMP тип 11 код 0).
- Длина пакета. Минимальное корректное значение для этого поля равно 20, максимальное — 65 535. В длине пакета учитывается заголовок и сами данные. Так как заголовок содержит 20 байт данных, то IP пакет **не может быть меньше 20 байт.**
- Протокол. TCP, UDP, ICMP.
- Контрольная сумма.

Ethernet (802.3)

Ethernet это довольно простой протокол. Состоит он из:

- MAC заголовок (14 байт): Отправитель, получатель, EtherType
- Данные (46 - 1500 байт): Например, IPv4 внутри
- CRC чексумма (4 байта)

Надо не забывать, что Ethernet **ходит между хостами и не выходит за их пределы**. Т.е. буквально в пределах одного кабеля. Когда пакет приходит на роутер, он уже создает другой пакет к следующему хосту. И **броадкаста в рамках Ethernet никакого нет**. Этот протокол работает на уровне сетевой карты/сетевого оборудования, ОС не участвует в формировании Ethernet пакета.

VLAN

> TODO

??????? ??????

Чаще всего используют 2 модели: OSI из 7 уровней и DOD (TCP/IP) из 4х уровней. А иногда из 5 уровней. А иногда вообще из 3х уровней.

OSI

Хоть эта модель не особо совместима с сегодняшними реалиями, она используется повсеместно и строго стандартизирована.

Чаще всего упоминают 4 уровня:

- L7 - Прикладной уровень. Это уровень самого протокола, например http, smtp, ftp
- L4 - Транспортный уровень. Это TCP, UDP.
- L3 - Сетевой уровень. Это адресация, т.е. IPv4, IPv6.
- L2 - Канальный уровень. 802.3 или Ethernet (всеми любимая "медь"), PON (Оптоволокно, которое тянут сегодня в квартиры), 802.11 (вайфай), PPPoE, PPP, MPLS. **Fun fact**: WinBox, приложение для настройки оборудования Mikrotik (а также мобильное приложение), умеет подключаться на уровне L2, т.е., например, по MAC адресу Ethernet порта. Поэтому если вы примените неправильные настройки адресации, то можно будет подключиться по MAC адресу.

DOD

Это не очень популярная и стандартизированная модель, однако она более простая и актуальная. В ней есть всего 4 уровня:

- Прикладной (HTTP, FTP, SMTP)
- Транспортный (TCP, UDP)
- Сетевой (IPv4, IPv6)
- Канальный (802.3, 802.11)

Вцелом то, зачастую все эти 4 уровня и используются и большего не надо, но если заглянуть вот сюда, то можно ужаснуться тому как модель DOD меняется от автора к автору. Хоть суть и одна, но везде они разные.

??????? ??????, ???????? ?????

Нам помогут 2 утилиты: tcpdump и ss. С помощью первой можно слушать трафик, с помощью второй смотреть соединения.

Краткий ликбез по tcpdump:

- `-n` : не резолвить адреса и домены, т.е. вместо yandex.ru:https будет 77.88.55.88:443. Это практически обязательная опция, так как резолвинг PTR записей адресов занимает неприлично много времени и практической пользы никакой, соотв-но трафик мы увидим с большой задержкой
- `-c` : количество IP пакетов (в штуках), которое мы хотим захватить
- `-i` : интерфейс, который мы хотим послушать. Можно указать any и мы будем слушать все существующие в системе интерфейсы
- `-v` : вербозность вывода. Можно указать -vv, -vvv, будет еще больше данных выводиться. При -v кроме Ethernet пакета будет выведен еще и TCP/UDP пакет.
- `-X` : будут выведены данные из пакета в ASCII и hex виде. Иногда может пригодиться, но более удобным будет параметр -A
- `-A` : будут выведены данные из пакетв в ASCII формате. Полезно для просмотра http/smtp/ftp или любого другого нешифрованного трафика.
- `-w file.dump` : Записать дамп в файл. Потом его можно будет открыть, например, в wireshark или в том же tcpdump
- `-r file.dump` : Прочитать дамп из файла. Можно также применять различные фильтры и параметры, как при просмотре трафика вживую.
- `-t, -tt, -ttt, -tttt, -ttttt` : разные форматы вывода времени. Лучше посмотри их в man'e

Вот пример трафика:

```
losted@pg-new:~$ sudo tcpdump -v -n  -i br0 port 1234
tcpdump: listening on br0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
# Открыли соединение
17:46:20.818195 IP (tos 0x0, ttl 64, id 12548, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.101.104.57202 > 192.168.101.103.1234: Flags [S], cksum 0x4c4f (incorrect ->
0x6a63), seq 524276269, win 64240, options [mss 1460,sackOK,TS val 1939885369 ecr 0,nop,wscale
7], length 0
17:46:20.818225 IP (tos 0x0, ttl 64, id 0, offset 0, flags [DF], proto TCP (6), length 60)
    192.168.101.103.1234 > 192.168.101.104.57202: Flags [S.], cksum 0x4c4f (incorrect ->
0xd5d5), seq 4126030036, ack 524276270, win 65160, options [mss 1460,sackOK,TS val 3603598166
ecr 1939885369,nop,wscale 7], length 0
17:46:20.818242 IP (tos 0x0, ttl 64, id 12549, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.101.104.57202 > 192.168.101.103.1234: Flags [.], cksum 0x4c47 (incorrect ->
0x0135), ack 1, win 502, options [nop,nop,TS val 1939885369 ecr 3603598166], length 0

# Отправили данные
17:46:23.889145 IP (tos 0x0, ttl 64, id 12550, offset 0, flags [DF], proto TCP (6), length 58)
    192.168.101.104.57202 > 192.168.101.103.1234: Flags [P.], cksum 0x4c4d (incorrect ->
0xb14b), seq 1:7, ack 1, win 502, options [nop,nop,TS val 1939888440 ecr 3603598166], length 6
17:46:23.889198 IP (tos 0x0, ttl 64, id 19860, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.101.103.1234 > 192.168.101.104.57202: Flags [.], cksum 0x4c47 (incorrect ->
0xe928), ack 7, win 510, options [nop,nop,TS val 3603601237 ecr 1939888440], length 0

# Закрыли соединение
17:46:28.637162 IP (tos 0x0, ttl 64, id 12551, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.101.104.57202 > 192.168.101.103.1234: Flags [F.], cksum 0x4c47 (incorrect ->
0xd6a3), seq 7, ack 1, win 502, options [nop,nop,TS val 1939893188 ecr 3603601237], length 0
17:46:28.637432 IP (tos 0x0, ttl 64, id 19861, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.101.103.1234 > 192.168.101.104.57202: Flags [F.], cksum 0x4c47 (incorrect ->
0xc40e), seq 1, ack 8, win 510, options [nop,nop,TS val 3603605985 ecr 1939893188], length 0
17:46:28.637448 IP (tos 0x0, ttl 64, id 12552, offset 0, flags [DF], proto TCP (6), length 52)
    192.168.101.104.57202 > 192.168.101.103.1234: Flags [.], cksum 0x4c47 (incorrect ->
0xc416), ack 2, win 502, options [nop,nop,TS val 1939893188 ecr 3603605985], length 0
```

На первой строке - IP, на второй строке (которая с табуляцией) - TCP. Зачастую глупые атаки это флуд однотипными пакетами. Например, у них может быть один и тот же **TTL, флаги, длина IP пакета, длина TCP пакета, длина окна**. Правила для iptables на основе этих

параметров можно легко сформировать.

Также можно посмотреть данные внутри и заблокировать через модуль `string` в iptables.

> TODO TCP Syn cookie

Revision #17
Created 2023-05-20 17:14:44 MSK by Ivan
Updated 2023-05-21 11:08:47 MSK by Ivan