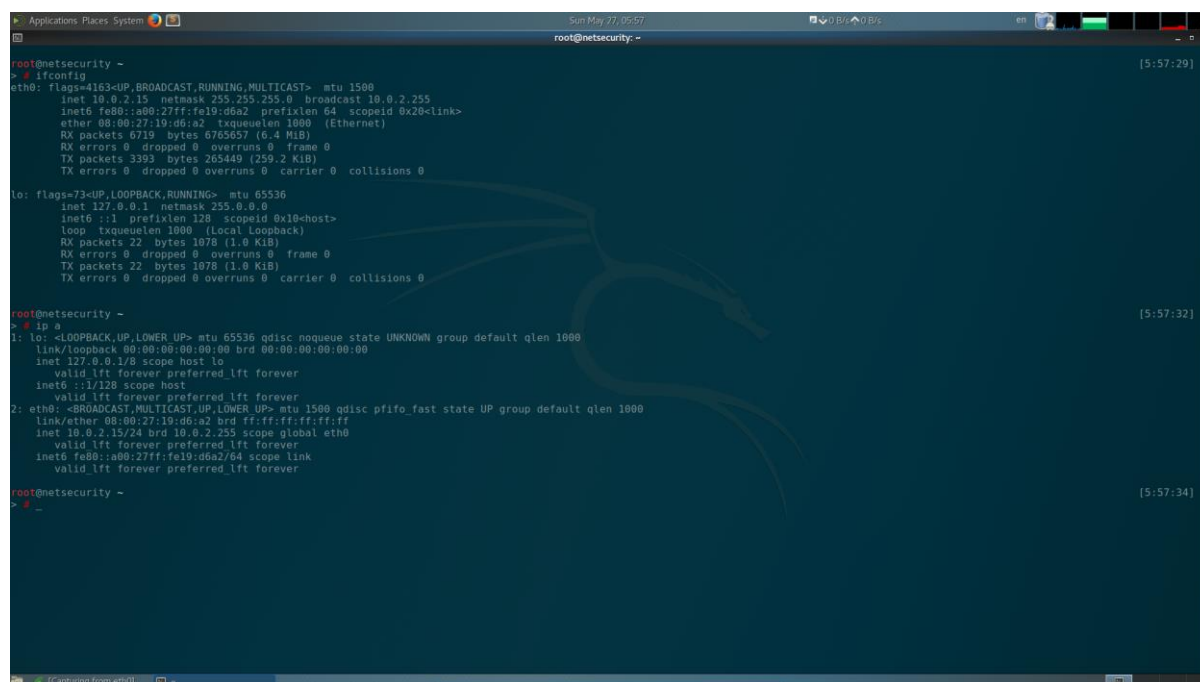


# Немного о сети и протоколах

Не будем мусолить историю появления интернета и т.д. Начнем сразу с протоколов и из чего все состоит и как взаимодействует. Для начала прошу ознакомиться со [стеком протоколов](#) и с [общими принципами устройства сети](#), [TCP](#), [UDP](#), [Сетевые пакеты](#).

На виртуальной машине уже настроен интернет, чтобы посмотреть список интерфейсов можно выполнить команду "ifconfig" или, актуальную сейчас, команду "ip a".



```
root@netsecurity ~
# ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.2.15 netmask 255.255.255.0 broadcast 10.0.2.255
    inet6 fe80::a00:27ff:fe19:d6a2 prefixlen 64 scopeid 0x20<link>
    ether 08:00:27:19:d6:a2 txqueuelen 1000 (Ethernet)
    RX packets 6719 bytes 6765857 (6.4 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 3393 bytes 265449 (259.2 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    inet6 ::1 prefixlen 128 scopeid 0x10<host>
    loop txqueuelen 1000 (local loopback)
    RX packets 22 bytes 1078 (1.0 KiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 22 bytes 1078 (1.0 KiB)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

root@netsecurity ~
# ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP group default qlen 1000
    link/ether 08:00:27:19:d6:a2 brd ff:ff:ff:ff:ff:ff
    inet 10.0.2.15/24 brd 10.0.2.255 scope global eth0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe19:d6a2/64 scope link
        valid_lft forever preferred_lft forever
```

Как видно у меня 2 сетевых интерфейса: lo и eth0. lo - loopback (обратная петля) адаптер, который есть всегда и везде, он нужен для внутренних нужд ОС и соответствует адресу 127.0.0.1 или localhost. Иначе говоря это локальная сеть твоего устройства. ethX, где X это номер адаптера, так как их может быть несколько, это Ethernet адаптер, т.е. тот в который втыкается кабель. Еще может быть wlan - wifi, tun - vpn и другие. Ну и еще они могут по разному называться, например в дебиан Ethernet адаптер называется enp. Отличить можно по "link/ether", а посмотреть именно wifi интерфейсы можно командой "iwconfig". Что еще тут интересного, включенный IPv4 и IPv6, это видно по строкам inet 10.0.2.15 и inet6 каляка-маляка. IPv6 принято считать проблемным и небезопасным, его также принято отключать (ввели чтобы решить проблему конечности IPv4). Отключается просто редактированием файла - /etc/sysctl.conf.

```
$ sudo nano /etc/sysctl.conf
```

И в конец файла добавь следующие строки

```
# IPv6 disabled
```

```
net.ipv6.conf.all.disable_ipv6 = 1
```

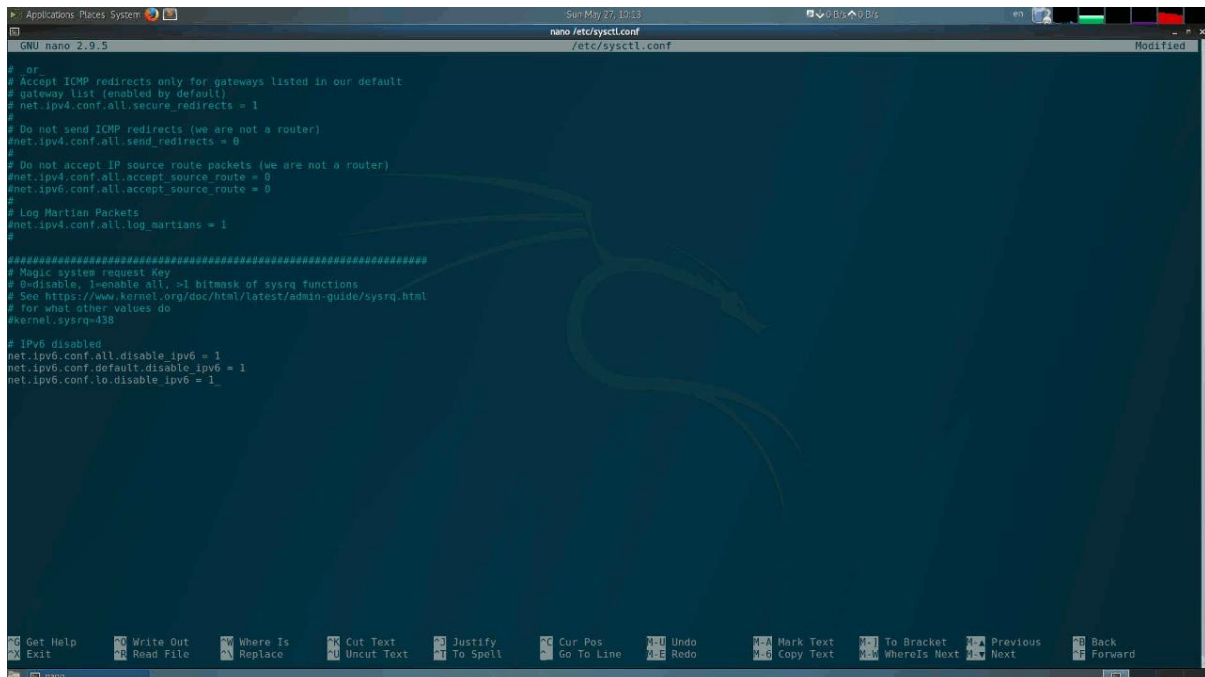
```
net.ipv6.conf.default.disable_ipv6 = 1
```

```
net.ipv6.conf.lo.disable_ipv6 = 1
```

Сохрани файл и закрой его. Перезапусти sysctl с помощью следующей команды

```
$ sudo sysctl -p
```

Снова проверь выходные данные, выдаваемые командой `ifconfig`, и на этот раз адреса `ipv6` не должно быть. Т.е. маска такая "`net.ipv6.conf.<ADAPTER>.disable_ipv6 = 1`"



```
GNU nano 2.9.5 nano /etc/sysctl.conf
# or
# Accept ICMP redirects only for gateways listed in our default
# gateway list (enabled by default)
# net.ipv4.conf.all.secure_redirects = 1
#
# Do not send ICMP redirects (we are not a router)
net.ipv4.conf.all.send_redirects = 0
#
# Do not accept IP source route packets (we are not a router)
net.ipv4.conf.all.accept_source_route = 0
net.ipv6.conf.all.accept_source_route = 0
#
# Log Martian Packets
net.ipv4.conf.all.log_martians = 1
#
#####
# Magic system request Key
# 0=disable, 1=enable all, >1 bitmask of sysrq functions
# See https://www.kernel.org/doc/html/latest/admin-guide/sysrq.html
# for what other values do
kernel.sysrq=438
#
# IPv6 disabled
net.ipv6.conf.all.disable_ipv6 = 1
net.ipv6.conf.default.disable_ipv6 = 1
net.ipv6.conf.lo.disable_ipv6 = 1
```

Мы подключены к интернету, а что дальше? А дальше кто то должен инициировать подключение куда то, например к сайту. Например нам нужен какой то сайт в интернете, мы должны знать либо его `ip` либо его доменное имя. DNS (Domain Name System) - Система Доменных Имен хранит в себе списки соответствия `IP` адресов их доменным именам. Когда ты открываешь браузер и пишешь доменное имя, первым делом браузер отправляет `udp` запрос на `dns`, дожидается ответа и если он поступил, то составляет `HTTP` запрос в котором указаны `IP` адрес запрашиваемого ресурса, доменное имя и путь по которому ты хочешь обратиться, также указывается длина запроса (Content Length), User-Agent браузера, Куки, список разрешенных алгоритмов сжатия (gzip, deflate, etc) и еще ряд других заголовков. Этот запрос уже получает сервер, обрабатывает и отдает ответ. Надо запомнить, что сейчас на одном сервере (по одному `IP`) может висеть далеко не один сайт, поэтому важно подключаться именно с указанием домена. Или например есть сайты, как блокчейн, которые не дадут ответа по `ип`, только по домену.

В первую очередь пользователь привлекает внимание генерацией большого объема трафика. Так как весь твой трафик (трафик это поток данных от тебя до целевых ресурсов и обратно) проходит через провайдера, он видит, что и куда ты отправляешь. Брутеры, парсеры, сканеры и другие подобные отправляют много запросов. Чтобы как то скрыть свое присутствие нужно пользоваться `впн`. Тебе уже дали `впн` и сидишь ты сейчас под тором. Наверняка под рукой есть другая машина без `впн` и прокси. Открой там консоль (в винде это `cmd.exe`, в маке тоже есть терминал) и выполни команду "`tracert ya.ru`". Утилита `tracert` покажет тебе через какие узлы сети пройдут пакеты при обращении к `ya.ru`, в данном случае (иначе говоря покажет маршрут

следования трафика). Наверняка первым узлом будет роутер, вторым непонятный IP (это IP коробки на чердаке) 3-им, а возможно даже и 4-м и 5-м будут тоже узлы провайдера и только потом будет узел yandex и потом сам ya.ru. Теперь попробуй тоже самое на своей боевой машине. В этом случае ты не должен увидеть хостов провайдера, если конечно ты все настроил правильно. Таким образом vpn помогает тебе скрыть сетевую активность от провайдера, кстати не только от провайдера, а еще и от других участников локальной сети, которые могут sniffать твой трафик. Почему этого не достаточно? Потому что хостер сервера арендованного под vpn, также ведет логи, при чем тут надо понимать, если логи не ведутся на самом сервере, они могут вестись в дата центре, поэтому одного vpn недостаточно. Тут уже можно пользоваться прокси, тором, ssh туннелями (распределил по надежности, и да, я считаю, что "ssh туннелирование трафика" не только звучит круче чем TOR, а еще и надежнее).

Перейдем к протоколам. Так как по сети передается поток байт и компам до фени что там файл, текст, музыка или картинки с котиками люди решили? что нужны какие то стандарты обмена информацией. И придумали они протоколы и RFC. RFC - это документы описывающие какие то стандарты или протоколы. Например [RFC 6797](#) описывающий стандарт HSTS. Поэтому если вам вдруг понадобилось знать как работает ssh протокол то лучшим вариантом будет ознакомиться с RFC. Ниже несколько протоколов о которых стыдно не знать:

## [FTP](#)

File Transfer Protocol - протокол передачи файлов. Работает FTP сервер на порту 21 по умолчанию и нужен, как понятно из названия, для обмена файлами между клиентом и сервером. Обычно для доступа требуется авторизация. Подключение производится FTP клиентом, мне нравится клиент FileZilla. А еще иногда можно просматривать ftp через веб, например <http://ftp.mozilla.org/pub/firefox/releases/> или вот так [ftp://\[user\]:\[password\]@\[host\]:\[port\]/url-path](ftp://[user]:[password]@[host]:[port]/url-path)

Wiki

## Безопасность

FTP не разрабатывался как защищенный (особенно по нынешним меркам) протокол и имеет многочисленные уязвимости в защите. В мае 1999 авторы [RFC 2577](#) свели уязвимости в следующий список проблем:

- Скрытые атаки (bounce attacks)
- Спуф-атаки (spoof attacks)
- Атаки методом грубой силы (brute force attacks)
- Перехват пакетов, sniffing (packet capture, sniffing)
- Защита имени пользователя
- Захват портов (port stealing)

FTP не может зашифровать свой трафик, все передачи — открытый текст, поэтому имена пользователей, пароли, команды и данные могут быть прочитаны кем угодно, способным перехватить пакет по сети. Эта проблема характерна для многих спецификаций Интернет-протокола (в их числе SMTP, Telnet, POP, IMAP), разработанных до создания таких механизмов шифрования, как TLS и SSL. Обычное решение этой проблемы — использовать «безопасные», TLS-защищенные версии уязвимых протоколов (FTPS для FTP, TelnetS для Telnet и т. д.) или же другой, более защищенный протокол, вроде SFTP/SCP, предоставляемого с большинством реализаций протокола Secure Shell.

## Telnet

Телнет позволяет передавать текст между клиентом и сервером. Обычно используется для доступа к интерфейсу командной строки. Протокол уязвим для любого вида атак, к которым уязвим TCP. По дефолту висит на 23-м порту.

## **Безопасность**

В протоколе не предусмотрено использование ни шифрования, ни проверки подлинности данных. Поэтому он уязвим для любого вида атак, к которым уязвим его транспорт, то есть протокол TCP. Для функциональности удалённого доступа к системе в настоящее время применяется сетевой протокол SSH (особенно его версия 2), при создании которого упор делался именно на вопросы безопасности. Так что следует иметь в виду, что сессия Telnet весьма беззащитна, если только не осуществляется в полностью контролируемой сети или с применением защиты на сетевом уровне (различные реализации виртуальных частных сетей). По причине ненадёжности от Telnet как средства управления операционными системами давно отказались.

## SSH

Secure Shell - безопасная оболочка. Этот протокол предоставляет защищенный доступ к командной строке сервера. SSH очень крутая штука, позволяющая не только выполнять текстовые команды на стороне сервера, а еще пробрасывать порты на удаленную машину, туннелировать трафик, который будет по умолчанию зашифрован (шифрование в протоколе ssh работает из коробки), передавать файлы (SCP протокол или SFTP), монтировать папки удаленной машины как виртуальные диски (для WebDAV) (протокол SSHFS) и даже пробрасывать удаленную сессию X сервера (для VNC или RDP). По умолчанию использует 22-ой порт.

## HTTP

Hiper Text Transfer Protocol - протокол передачи гипертекста. HTTP протокол позволяет нам серфить интернет, HTTPS протокол - это HTTP использующий шифрование SSL. используемые порты: 80 и 443, последний для ssl. SSL (Secure Socket Layer) - криптографический протокол, необходимый для защиты передаваемой по сети информации.