Network side channels and introduction to DoS and DNS security basics; “network alchemy”
CSE 468 Fall 2022

Outline

- Review of port scanning, idle scans
- Examples of network side channels
  - SYN backlogs and DoS
  - DNS poisoning
- “Network alchemy”

TCP 3-way handshake (review)

- SYN: I'd like to open a connection with you, here's my initial sequence number (ISN)
- SYN/ACK: Okay, I acknowledge your ISN and here's mine
- I ACK your ISN

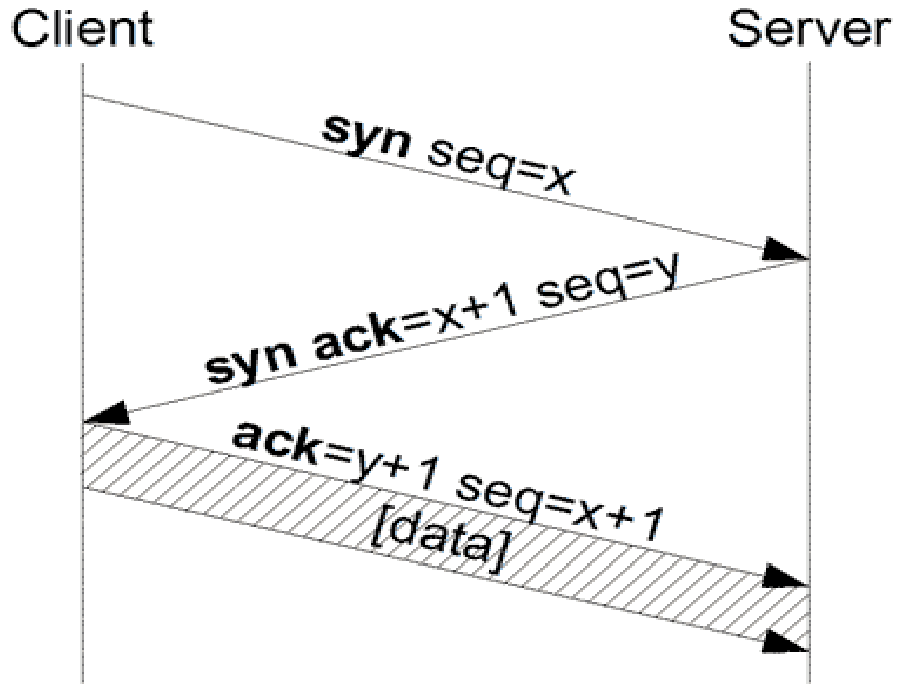


Image from Wikipedia

Open port == listening

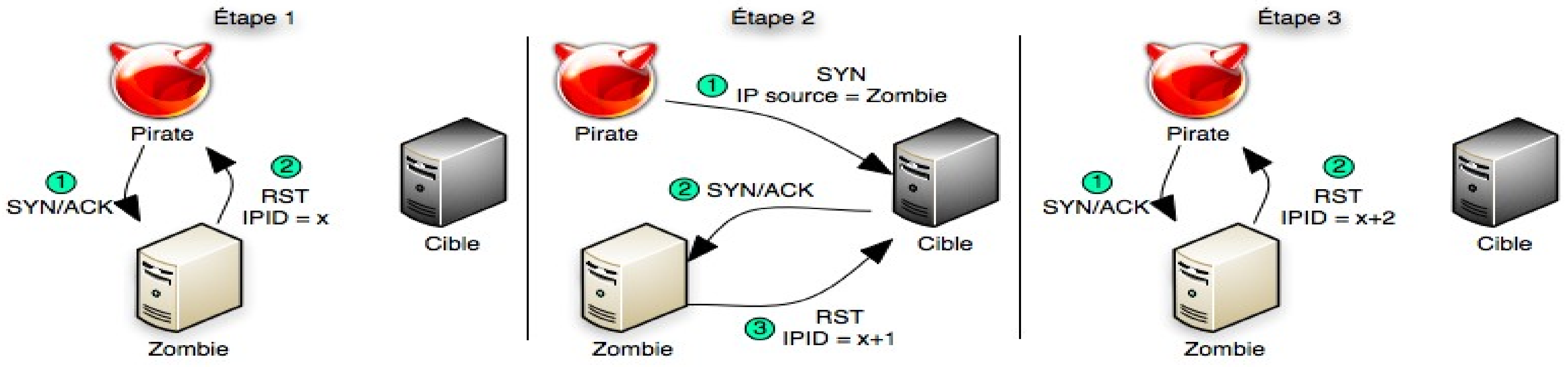
- If you send a SYN packet to port 80 (the HTTP port) on a remote host and that host replies with a SYN/ACK, then we say that port 80 on that machine is “open”
  - In this example, that probably means it's a web server
- If it responds with a RST, we say it's “closed”
- If there is evidence of filtering (no response or ICMP==Internet Control Message Protocol error), we say it's “filtered”
  - UDP is more complicated: open|filtered *vs.* closed

Things nmap can do

- Is a port open? Closed? Filtered?
  - Many ports on one machine is a “vertical scan”
- For a /24 network, which machines are up? Which machines have port 80 open?
  - One port for a range of machines is a “horizontal scan”
- OS detection (research on your own)
- Stealth, info about middleboxes, etc.

Idle scan

- Every IP packet sent has an IP identifier
  - In case it gets fragmented along the way
- Old and/or stupid machines use a globally incrementing IPID that is shared state for all destinations



Examples of network side channels

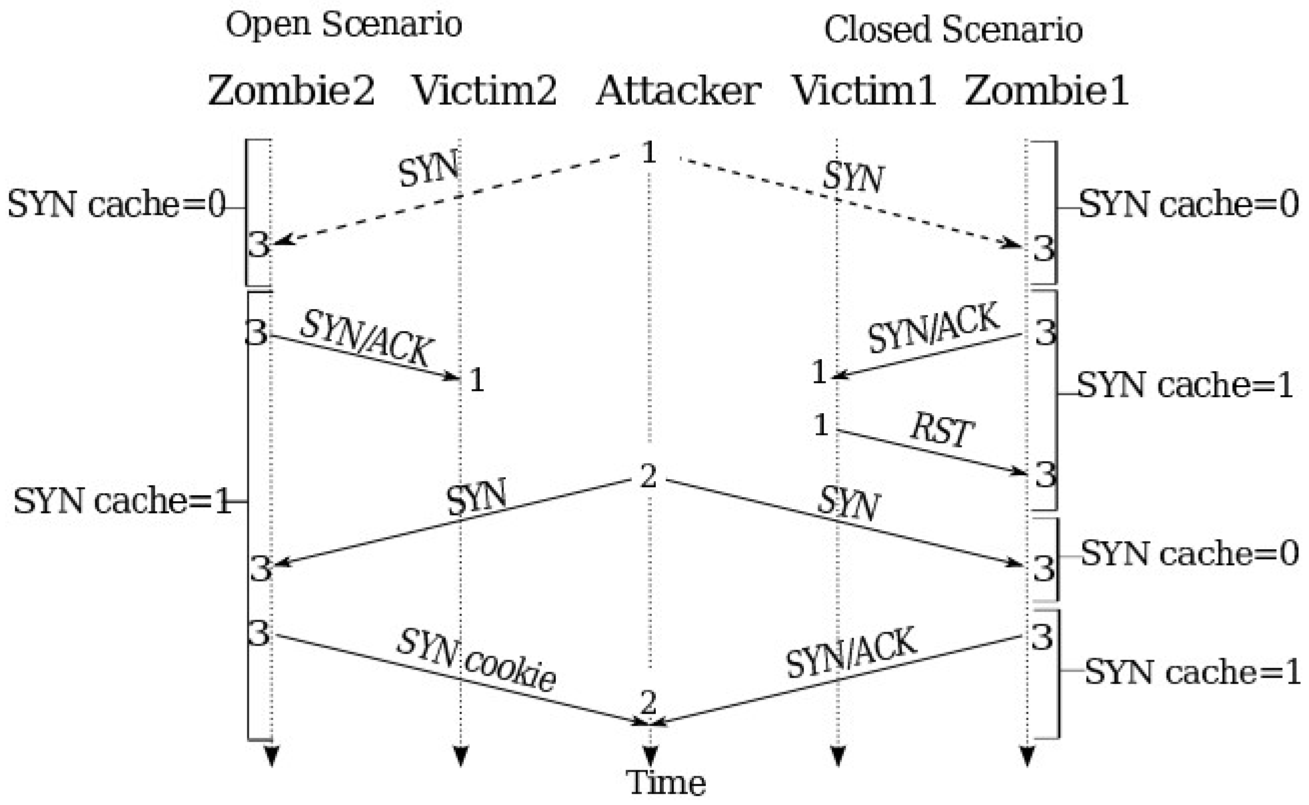
- DoS and SYN backlog basics
  - A side channel based on the SYN backlog
- Counting packets off-path (Jeff Knockel’s slides)
- DNS poisoning overview and attacking DNS (Travis Palmer’s slides)

DoS in general

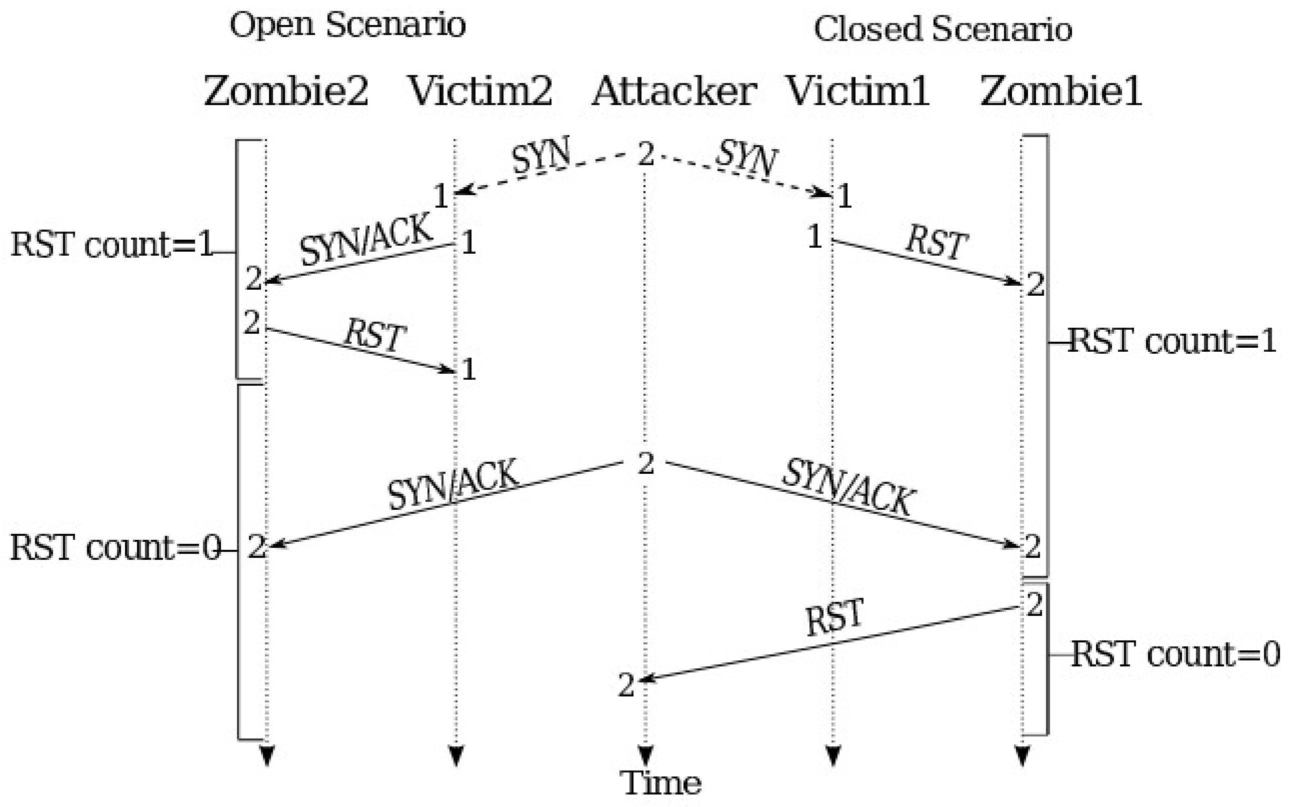
- Exhaust some kind of resource, *e.g.*:
  - Optimistic ACK to exhaust bandwidth
    - See https://homes.cs.washington.edu/~tom/pubs/CCR99.pdf
  - PING of death (*e.g.*, large PING) causes crash
  - Exhaust CPU in layer 7
  - More examples: http://www.isi.edu/~mirkovic/bench/attacks.html
  - SYN flood: Older hosts had either a fixed amount of half-open connections they could keep track of or no limitations at all, attack is to send lots of SYNs and never ACK or RST
    - Defenses: SYN backlog policies and SYN cookies

SYN cookies and SYN backlogs

- SYN cookies
  - Special kind of SYN/ACK
  - See https://cr.yp.to/syncookies.html
  - Can confirm ACK number and reconstruct the necessary state for a connection without having kept any state after sending the SYN cookie
- SYN backlog examples
  - Linux reserves ½, ¼, 1/8th, and so on for successively older SYNs, prunes 5 times a second
  - FreeBSD has 512 buckets of 30, you can't predict what bucket you fall into (in theory)



From… https://jedcrandall.github.io/usenix10.pdf



From… https://jedcrandall.github.io/usenix10.pdf

Jeff Knockel’s FOCI 2014 slides…

Travis Palmer’s DEFCON 27 slides...

Network alchemy

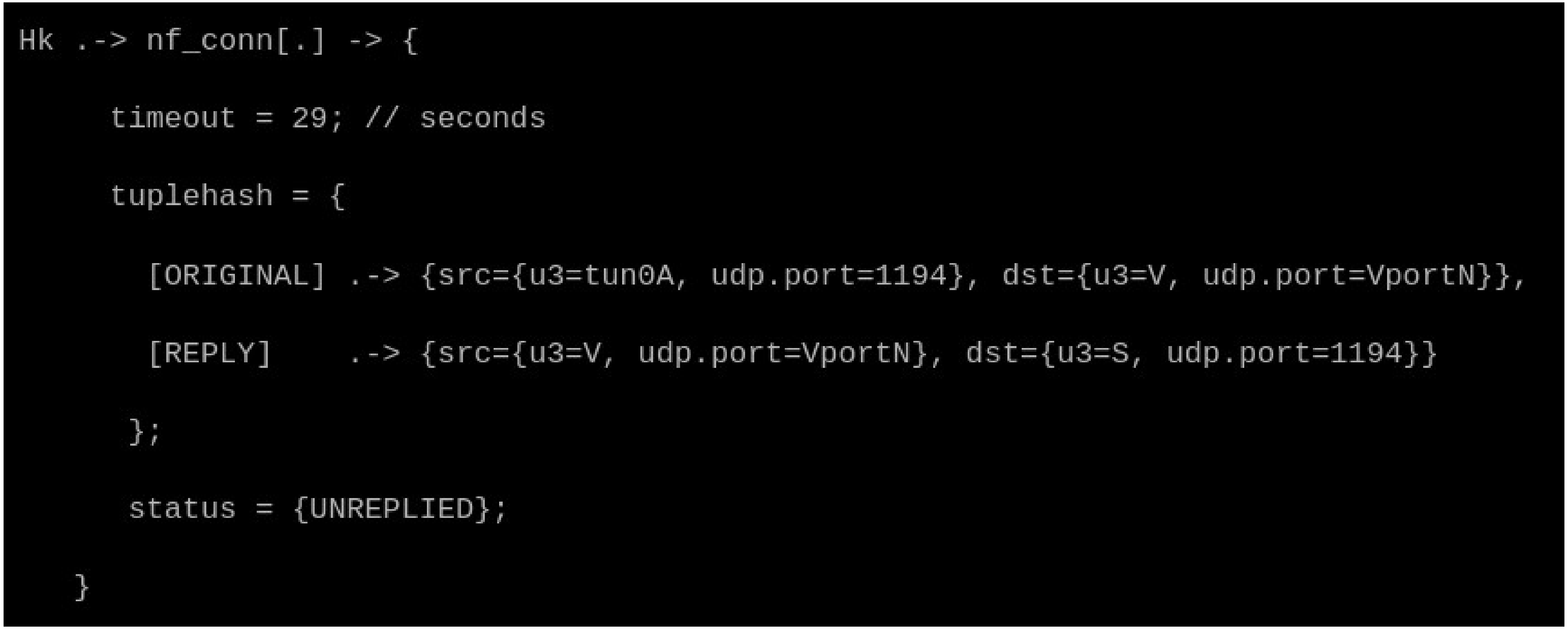
Just to give you some idea, you’ll explore this in the homework...

Attacker connects to VPN...

```
H1 .-> nf_conn[.] -> {

     timeout = 179; // seconds

     tuplehash = {

       [ORIGINAL] .-> {src={u3=A, udp.port=Aport}, dst={u3=S, udp.port=1194}},

       [REPLY]    .-> {src={u3=S, udp.port=1194}, dst={u3=A, udp.port=Aport}}

     };

     status = {ASSURED};

   },
```

Attacker sends UDP packets with source port 1194 and destination ports across the whole ephemeral space to the victim’s IP address...

Tens of thousands of these...



```
Hk .-> nf_conn[.] -> {
     timeout = 29; // seconds
     tuplehash = {
       [ORIGINAL] .-> {src={u3=tun0A, udp.port=1194}, dst={u3=V, udp.port=VportN}},
       [REPLY]    .-> {src={u3=V, udp.port=VportN}, dst={u3=S, udp.port=1194}}
      };
      status = {UNREPLIED};
   }
```

What happens when the victim tries to connect to the VPN? (This is half of homework 3).

References

- *NMAP NETWORK SCANNING*, by Gordon “Fyodor” Lyon
- Google “nmap”, “idle scan”, etc.
- Other references were linked to inline