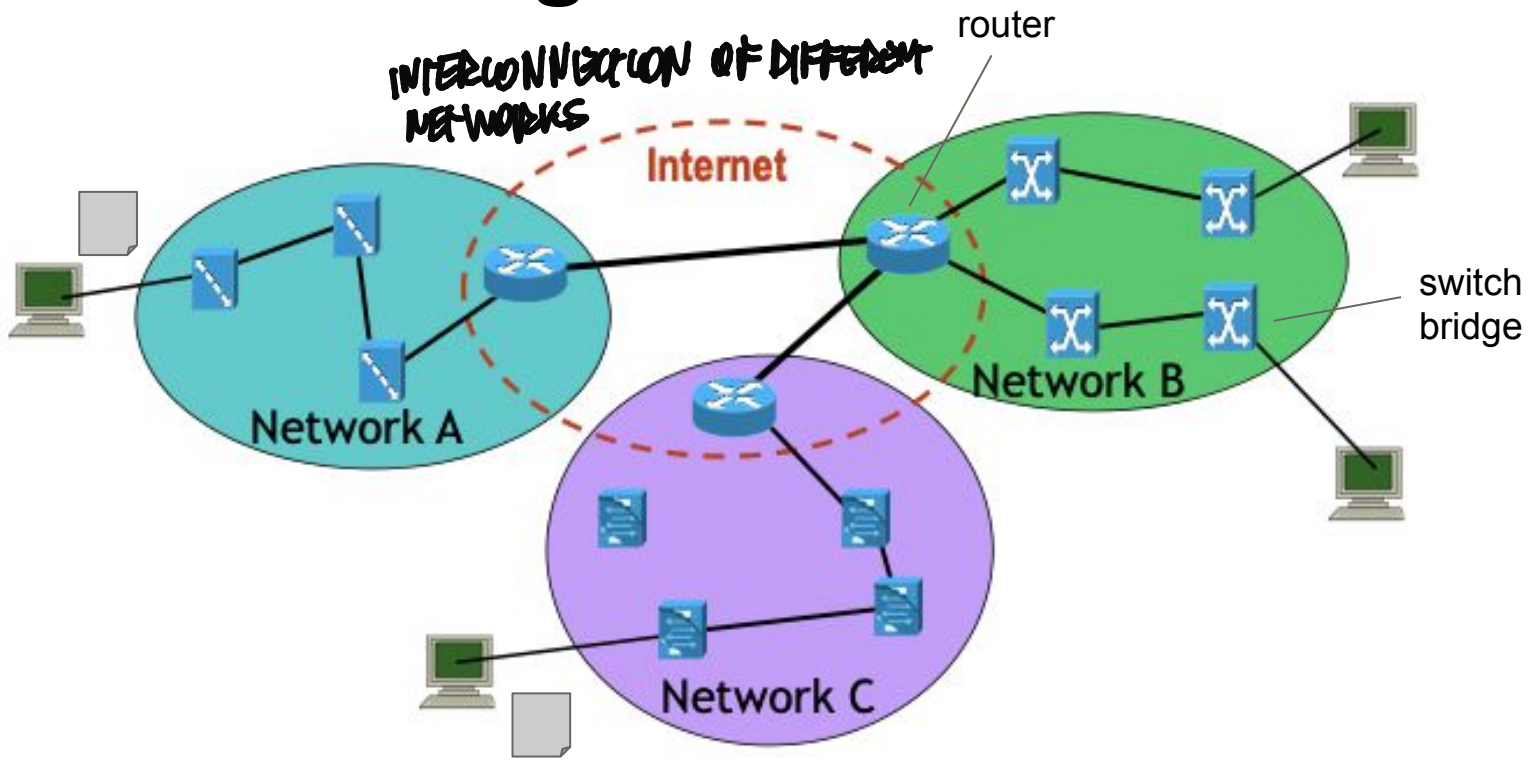
9. Network Protocol Attacks

Computer Security Courses @ POLIMI

RECALL

Internetworking



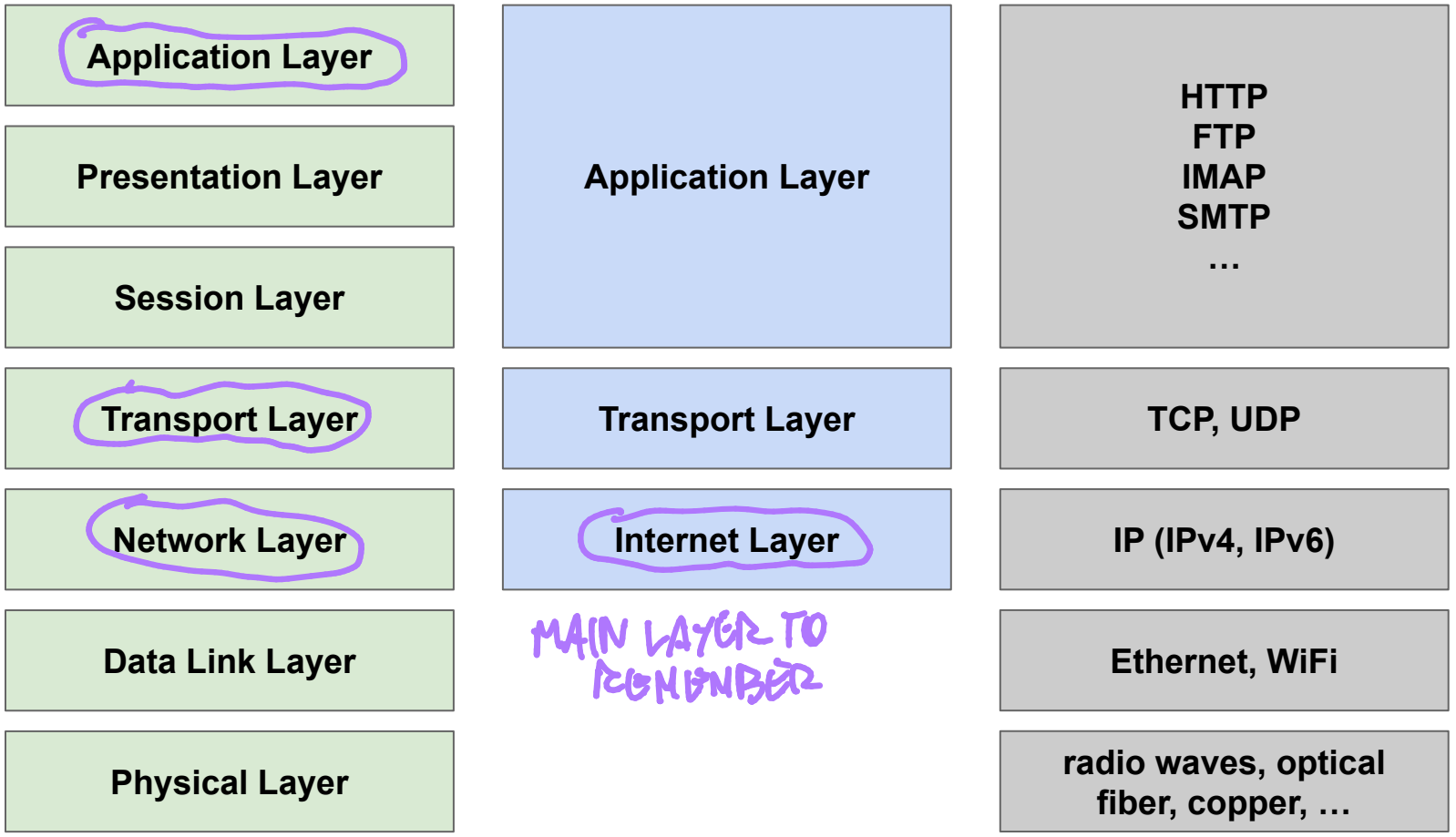
- Networks = different physical media, topologies, …
- Need a *layered* approach

RECALL

Layering and Protocols

2 IMPORTANT STANDARDS

EACH LAYER HAS ITS OWN PROTOCOL



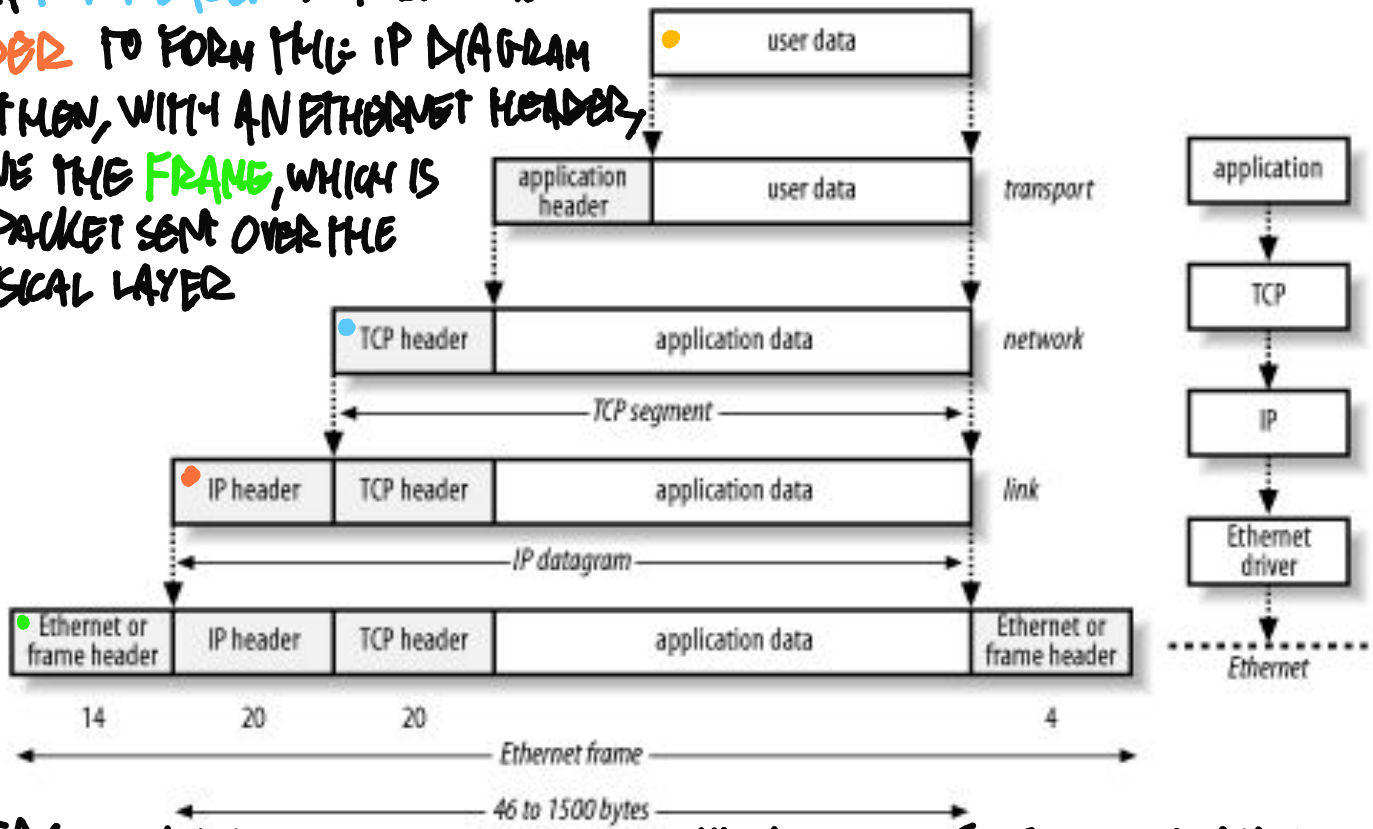
MAIN LAYER TO REMEMBER

EVERY SINGLE TIME WE WANT TO SEND SOMETHING ON THE NETWORK, GO THROUGH ALL THE LAYERS

RECALL

Packet Encapsulation

EVERY SINGLE TIME WE NEED TO SEND DATA, THIS WILL BE ENCAPSULATED FIRST FROM A TCP HEADER TO FORM THE TCP SEGMENT AND THEN WITH AN IP HEADER TO FORM THE IP DIAGRAM AND THEN, WITH AN ETHERNET HEADER, DEFINE THE FRAME, WHICH IS THE PACKET SENT OVER THE PHYSICAL LAYER



HEADERS GENERALLY CONTAIN ALL THE INFORMATION NEEDED TO INTERPRET THE RESPECTIVE PAYLOAD + ROUTING AND ADDRESSING INFORMATION

RECALL

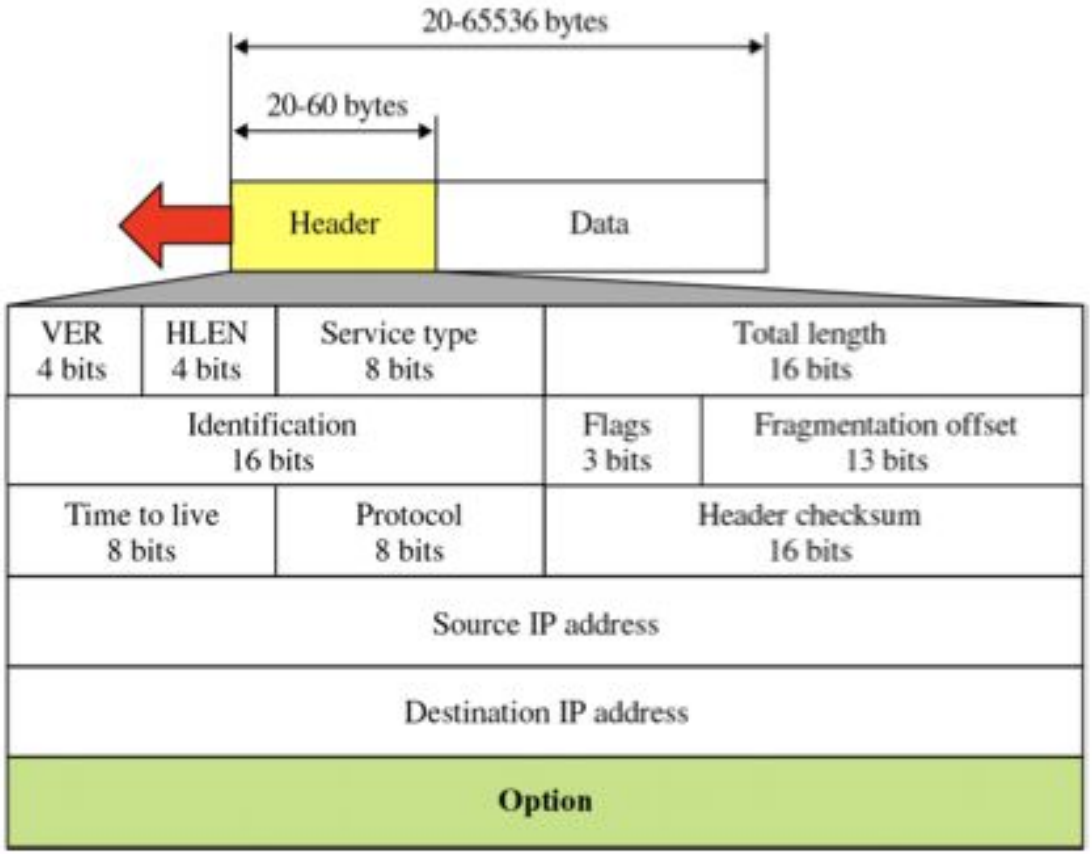
Addressing

WHICH MAY CHANGE BASED ON THE CONSIDERED LAYER

- Hosts are identified uniquely by **addresses**
  - ~= phone number or a (snail) mail address
- Each layer has its own addressing structure
  - Data link layer: **MAC address** (for Ethernet)
    - A globally unique address “burnt” in the NIC
    - The ARP protocol maps an IP address to a MAC addr
  - Internet layer: **IP address**
    - Identifies “globally” a network host
    - There can be private addresses (RFC1918 for IPv4)
  - Transport layer: **port**
    - Identifies a specific service on a host

RECALL

Example: IP Datagram

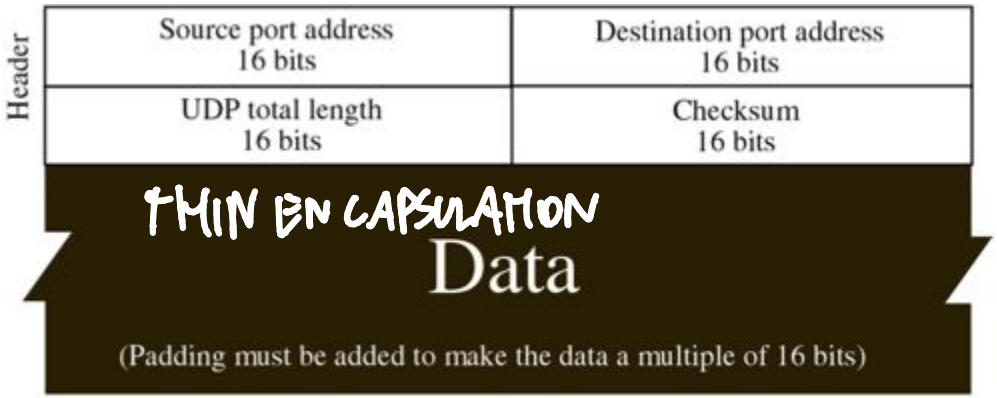


RECALL

Transport Protocols

UDP

- Connectionless
- A thin wrapper around an IP packet with a port number and not much else



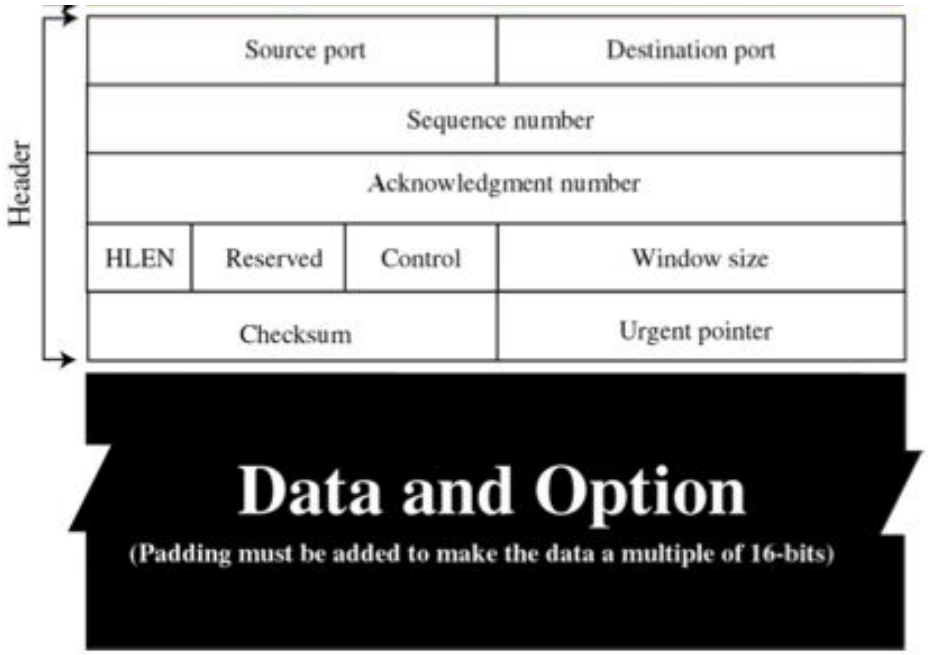
RECALL

Transport Protocols

TCP

- Connection-oriented
- Concept of **connection** ~> state (closed, open, established)
- Connections are set up with a three-way handshake

HEADER IS MUCH MORE COMPLEX



Taxonomy of Typical Attacks

Denial of Service (against availability):

- service unavailable to legitimate users

Sniffing (against confidentiality):

- abusive reading of network packets

Spoofing (against integrity and authenticity):

- forging network packets WITH DIFFERENT IDENTITIES
  —> FAKING A SOURCE

In the following we will present examples of attacks, not an exhaustive list

Examples of Denial of Service

THE RECEIVING SYSTEM IF THE PROTOCOL IMPLEMENTATION OF EXPLOITING A VULNERABILITY

- Killer Packets → ATTACKS IN WHICH YOU MAKE THE COMMUNICATION FAIL BY
- SYN flood
- Smurf, multiplication or amplification attacks
- Distributed DoS

CATEGORY OF ATTACKS THAT MAKES THE COMMUNICATION FAIL BY SATURATING THE RESOURCING MAKING THE SERVER UNAVAILABLE FOR LEGITIMATE USERS

FLOODING

USUALLY, THEY ARE NOT PRESENT ANYMORE

Killer Packets (1): Ping of Death

→ PAG. 83

Pathological ICMP echo request that exploit a memory error in the protocol implementation.

"gazillions of machines can be crashed by sending IP packets that exceed the maximum legal length (65535 octets)" ≃ BUFFER OVERFLOW

http://insecure.org/sploits/ping-o-death.html

- ping -l 65527 (Win), or ping -s 65527 (*NIX)

Killer Packets (2): Teardrop

FOCUS ON TCP PROTOCOL IN WHICH, IF WE HAVE A PAYLOAD GREATER THAN THE MAXIMUM TRANSFER UNIT, IT IS GOING TO BE SEGMENTED IN DIFFERENT PACKETS COMPOSED BY PART OF THE PAYLOAD AND AN OFFSET THAT SPECIFY THE POSITION OF THE PAYLOAD INSIDE THE ORIGINAL PAYLOAD

Exploit vulnerabilities in the TCP reassembly.

Fragmented packets with overlapping offsets.

=> BY MEANS OF THIS MECHANISM, A RECEIVER OF THE COMMUNICATION IS ABLE TO RECONSTRUCT A PAYLOAD BY PUTTING EACH SINGLE PACKET AND READING THE OFFSET

While reassembling, kernel can hang/crash.

BECAUSE OF THE PARTIAL OVERLAP

- 1997 (TCP level - basically every major OS was affected)
  - http://www.cert.org/historical/advisories/CA-1997-28.cfm
- 2009 (SMB level - Windows Vista)
  - http://g-laurent.blogspot.it/2009/09/windows-vista7-smb20-negotiate-protocol.html

Killer Packets (3): Land Attack

CREATE A TCP PACKET WITH SRC IP = DST IP AND, IF NOT CORRECTLY MANAGED, THE PACKET STARTS TO LOOP INSIDE THE SYSTEM UNTIL THE MEMORY GETS FILLED AND MACHINE CRASHES

A long time ago, in a Windows 95 far, far away, a packet with

- src IP == dst IP
- SYN flag set

could loop and lock up a TCP/IP stack.

IN GENERAL, NETWORK PROTOCOLS ARE NOT DESIGNED FOR SECURITY PERSPECTIVE. ONLY FOR FUNCTIONAL PERSPECTIVE

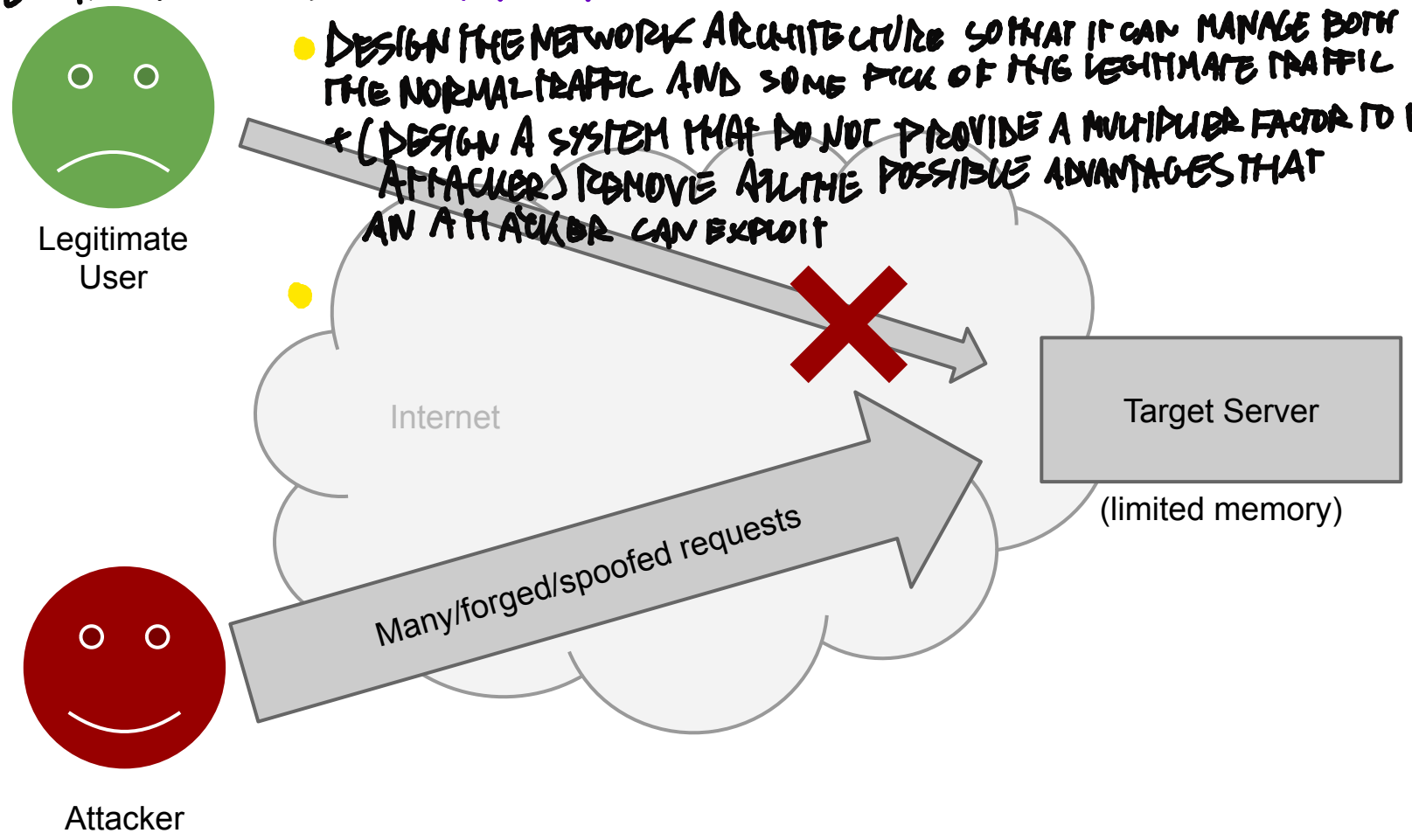
Back to the future, same happened with SP2 in Windows XP: “This thing is like Dracula: it just won't stay dead”

http://www.cert.org/historical/advisories/CA-1997-28.cfm

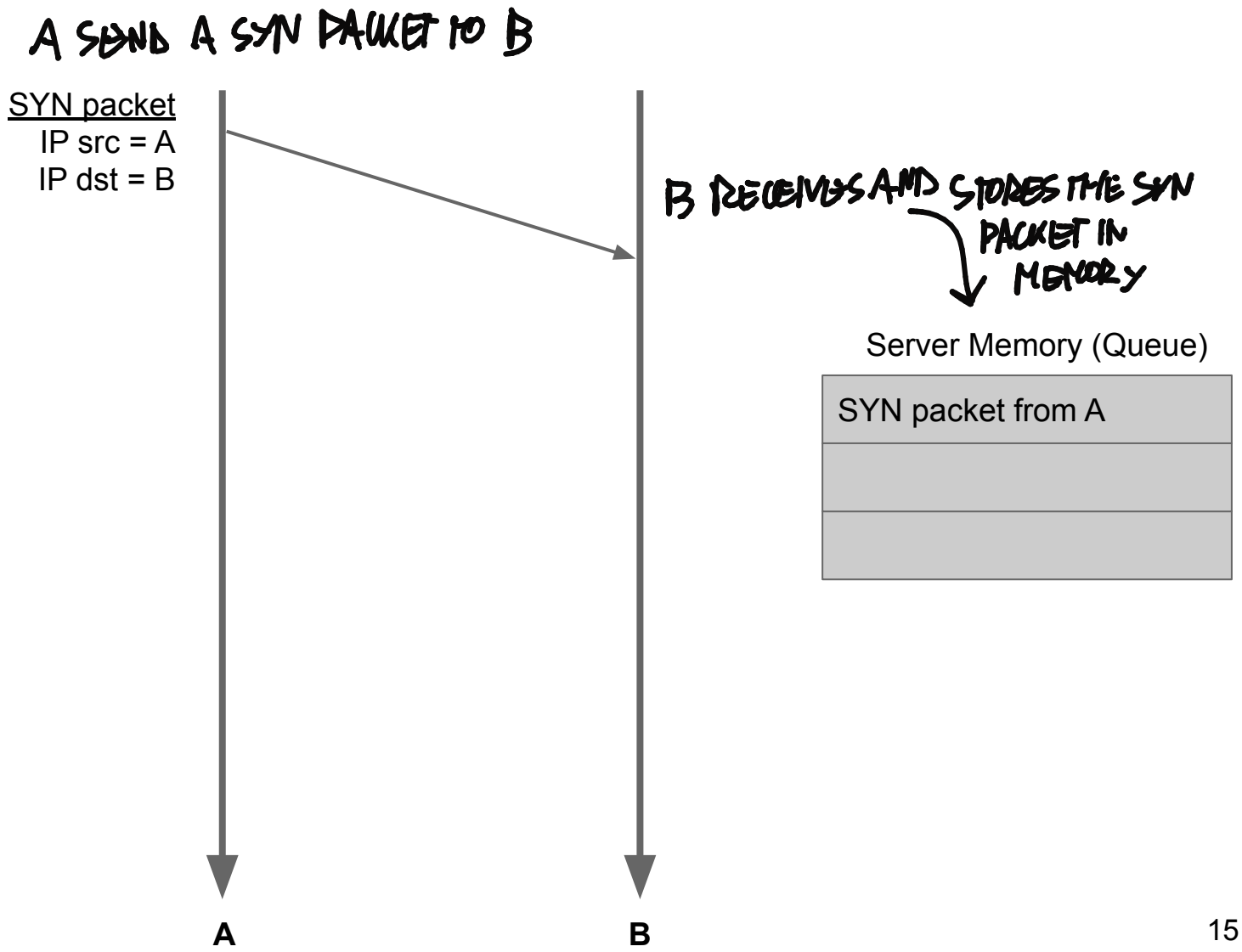
Denial of Service via Flooding

IMPORTANCE OF THIS ATTACK IS BASED ON THE FACT THAT IT CAN'T BE SOLVED, WE CAN ONLY MITIGATE IT. 2 POSSIBLE WAYS:

- DESIGN THE NETWORK ARCHITECTURE SO THAT IT CAN MANAGE BOTH THE NORMAL TRAFFIC AND SOME PICK OF THE LEGITIMATE TRAFFIC
- (DESIGN A SYSTEM THAT DO NOT PROVIDE A MULTIPLIER FACTOR TO THE ATTACKER) REMOVE ALL THE POSSIBLE ADVANTAGES THAT AN ATTACKER CAN EXPLOIT

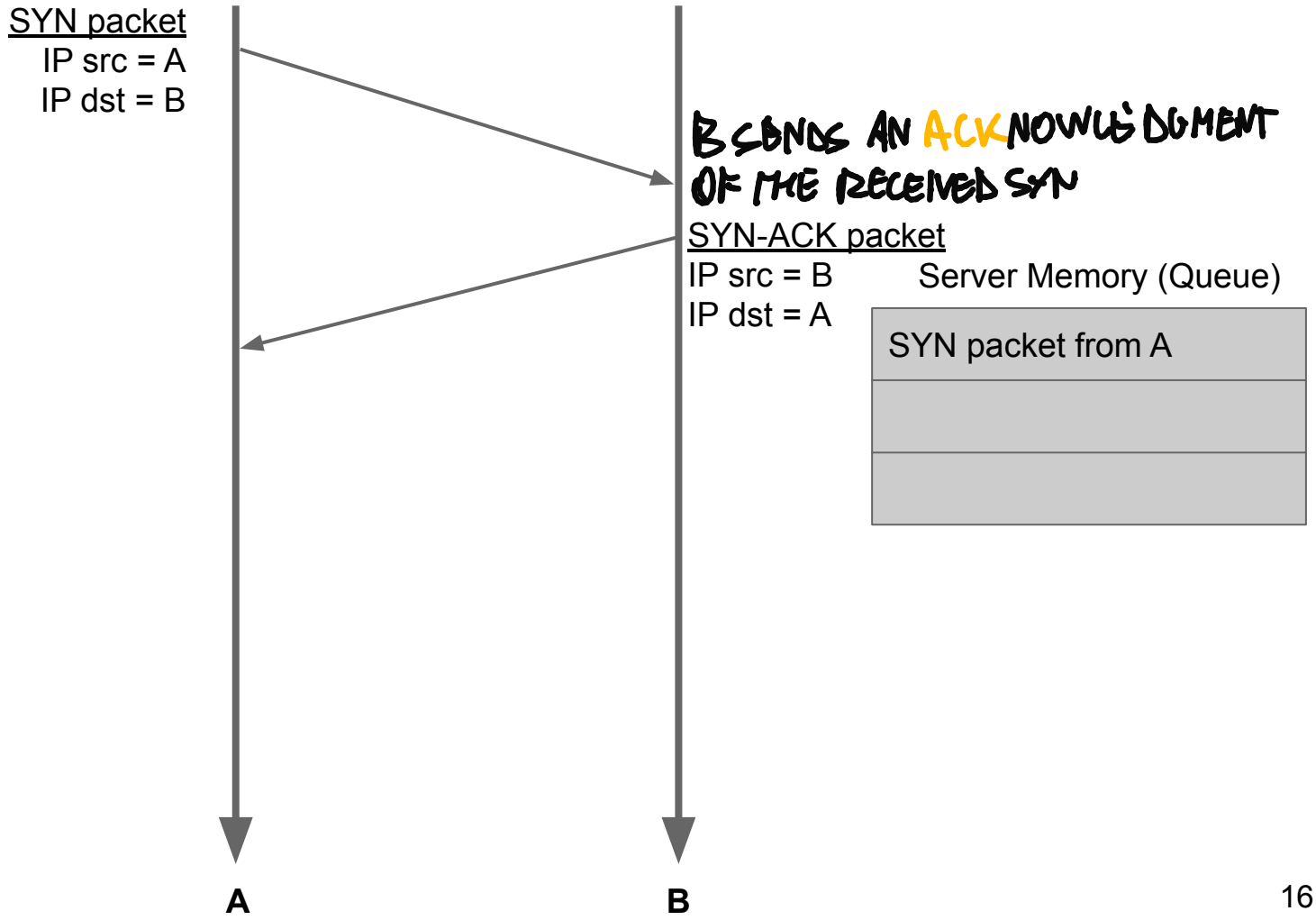


SYN Flood Attack: The TCP/IP Three Way Handshake



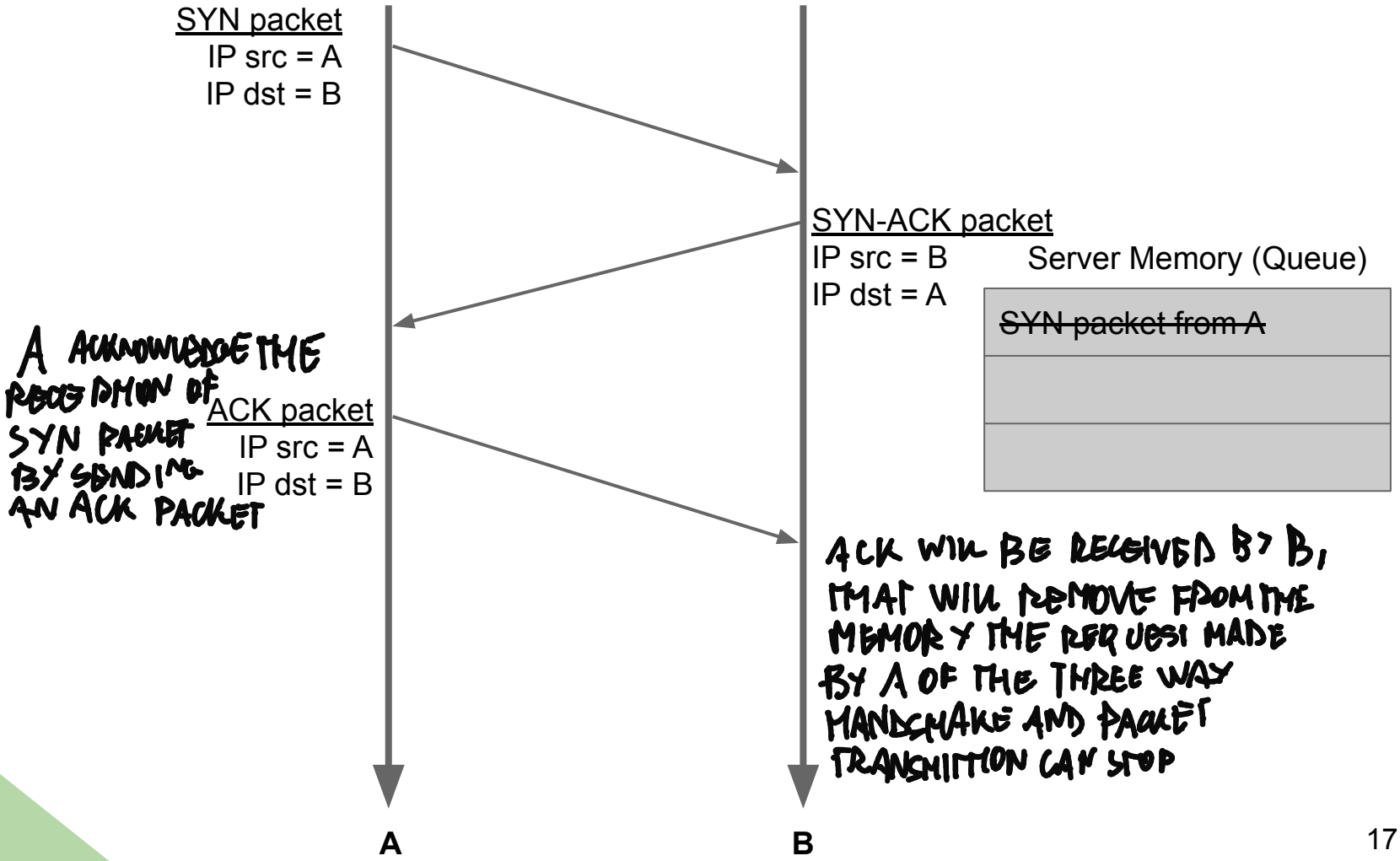
RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake



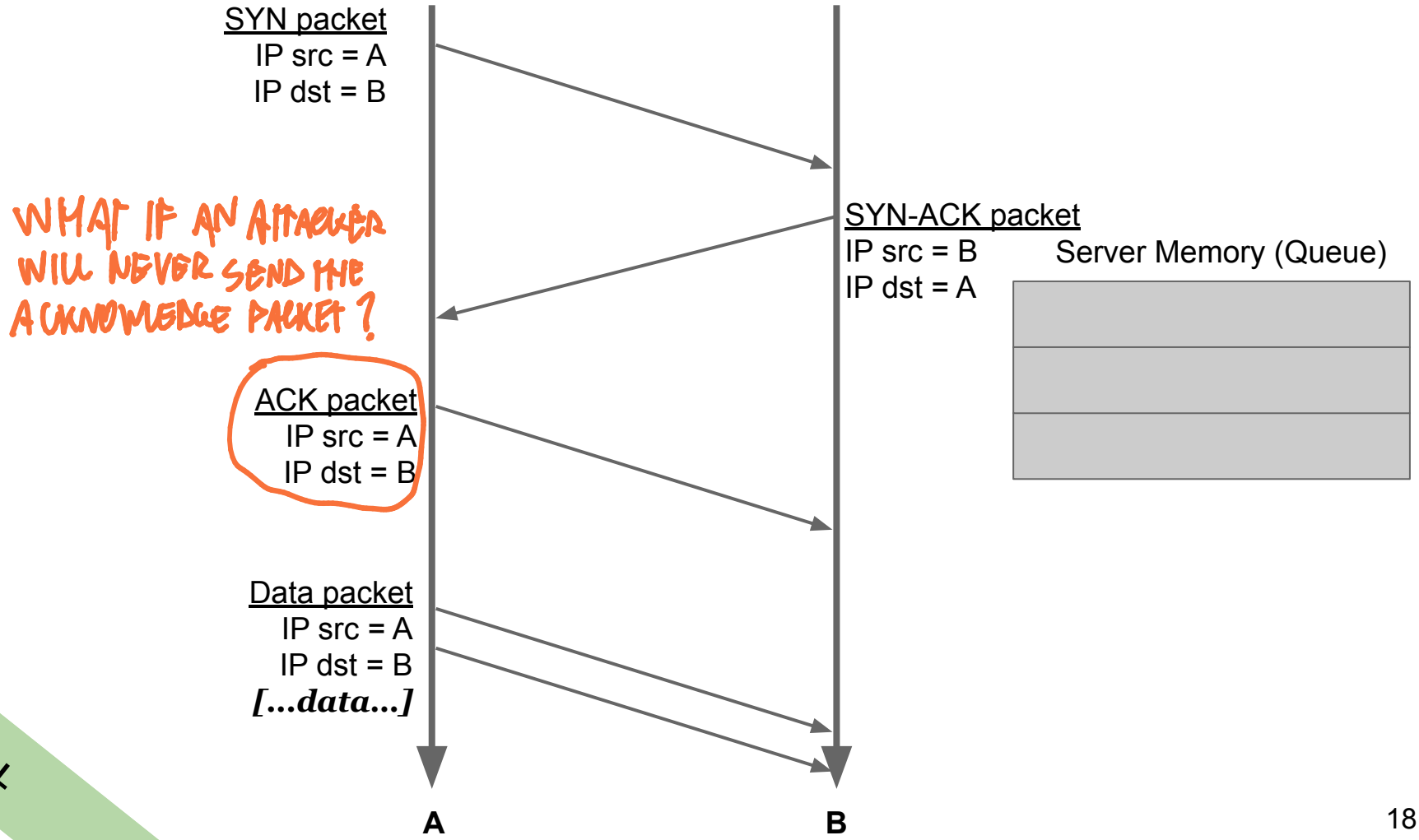
RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake



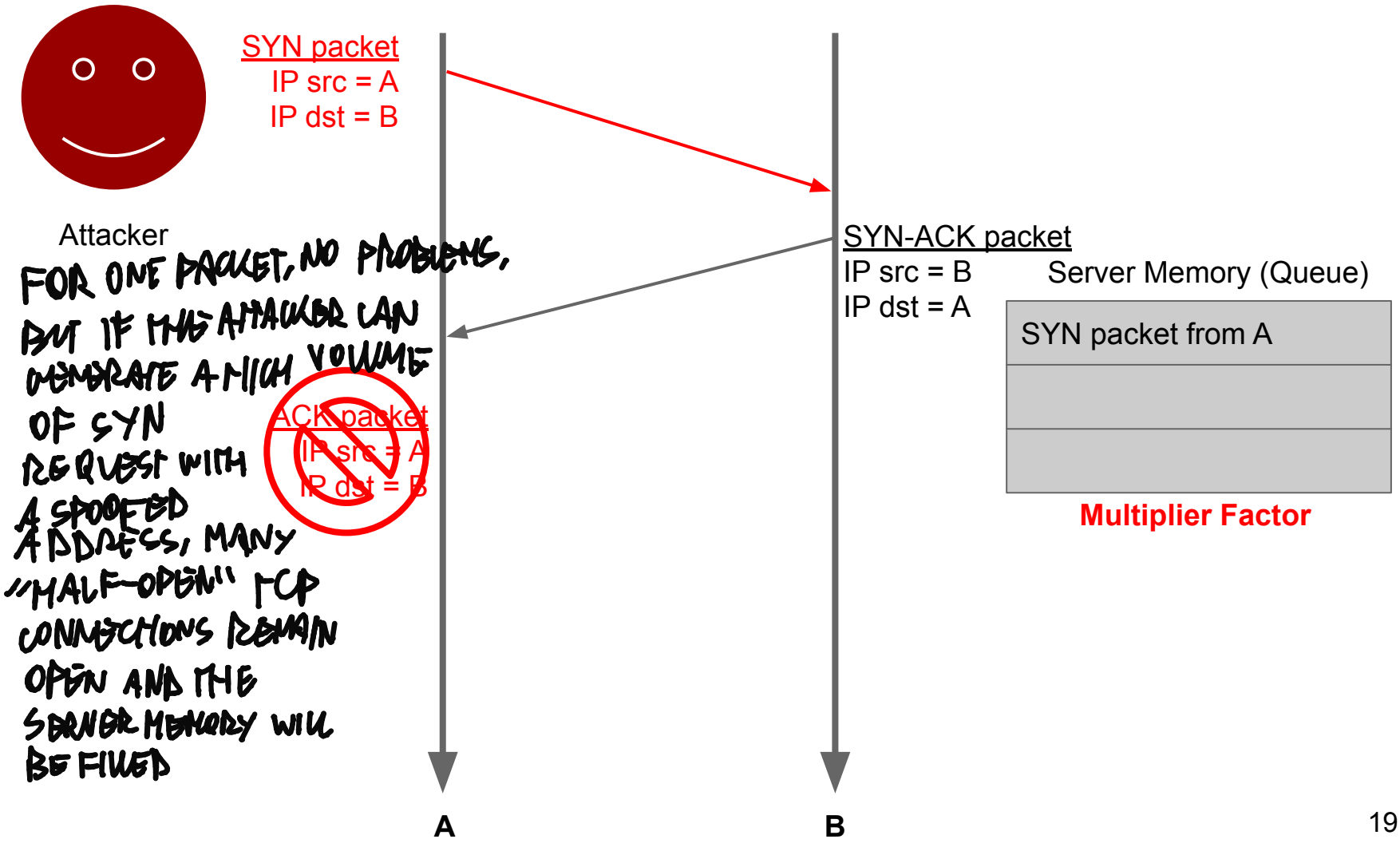
RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake



RECALL

SYN Flood Attack: Exploiting the three way handshake



SYN Flood Attacks

Attacker generates a high volume of SYN requests with **spoofed source address**.

Many half-open TCP/IP connections fill the queue.

SYN Flood Attack: Exploiting the three way handshake

Attacker

SYN packet
IP src = A
IP dst = B

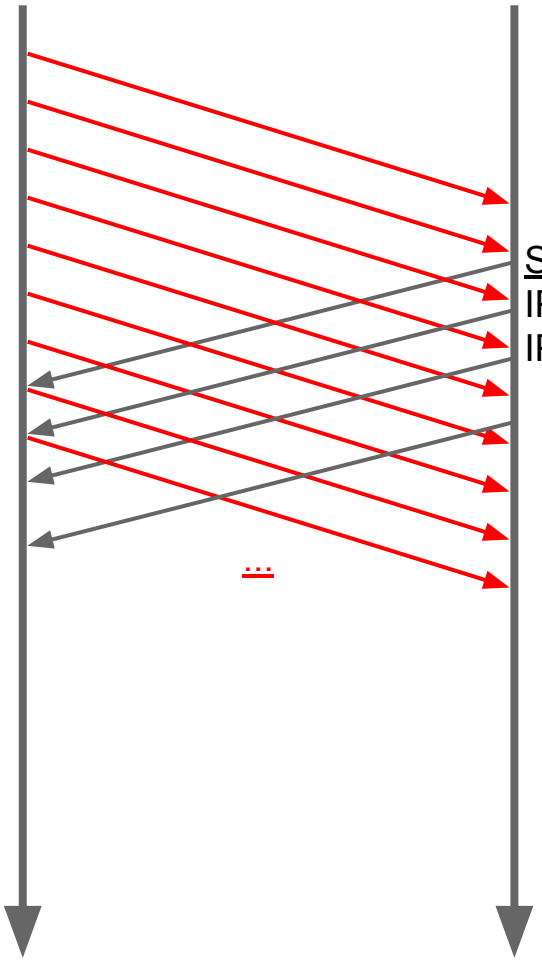
SYN packet
IP src = A''
IP dst = B

SYN packet
IP src = A'''
IP dst = B

SYN packet
IP src = A''''
IP dst = B

...

≠ IP ADDRESSES
A, A', A''...



Server Memory (Queue)

SYN packet from A
SYN packet from A'
SYN packet from A''

...

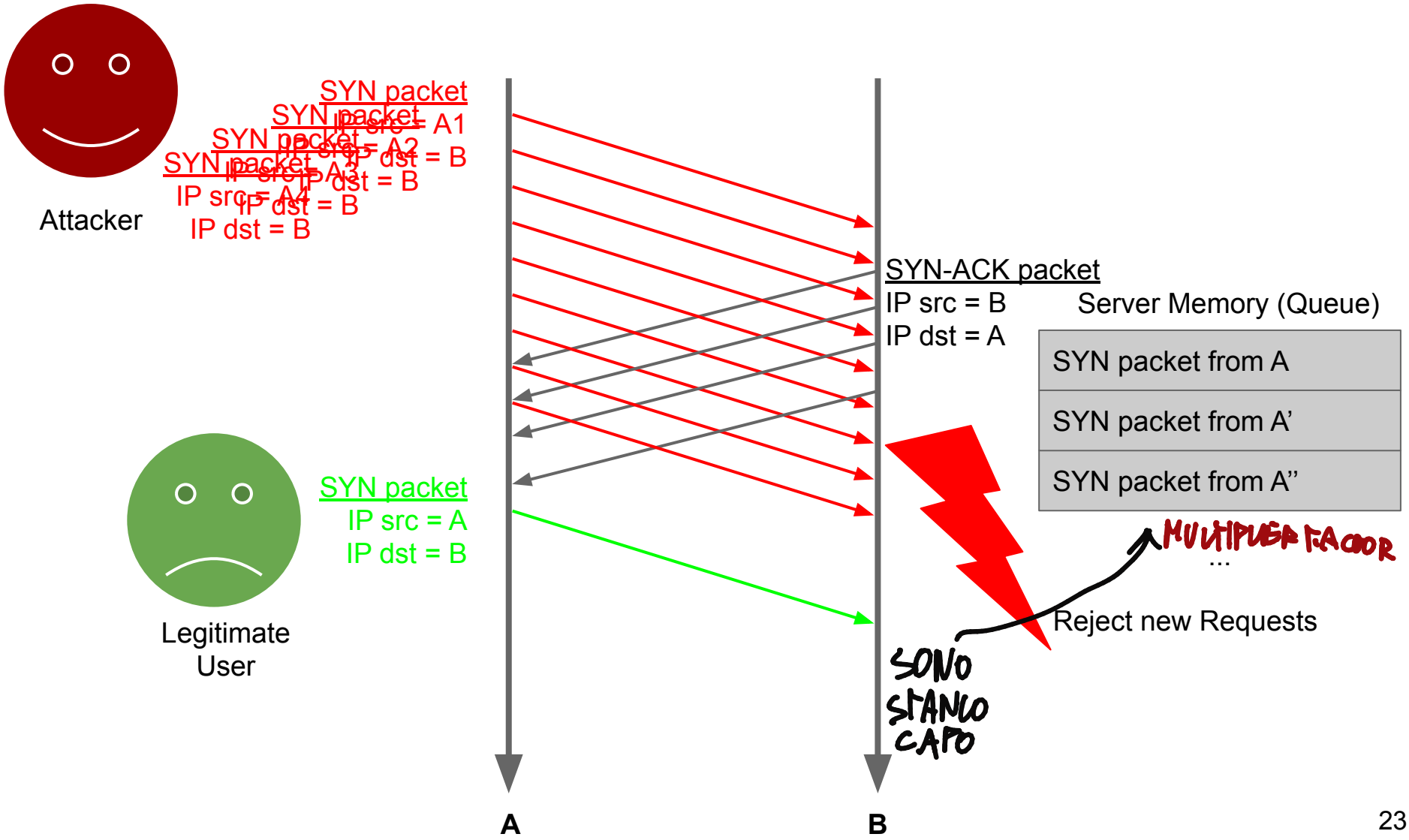
SYN Flood Attacks

Attacker generates a high volume of SYN requests with **spoofed source address**.

Many half-open TCP/IP connections fill the queue.

SYN requests from legitimate clients dropped.

SYN Flood Attack: Exploiting the three way handshake



SYN Flood Attacks

Attacker generates a high volume of SYN requests with **spoofed source address**.

Many half-open TCP/IP connections fill the queue.

SYN requests from legitimate clients dropped.

Mitigation: ???

SYN Flood Attacks

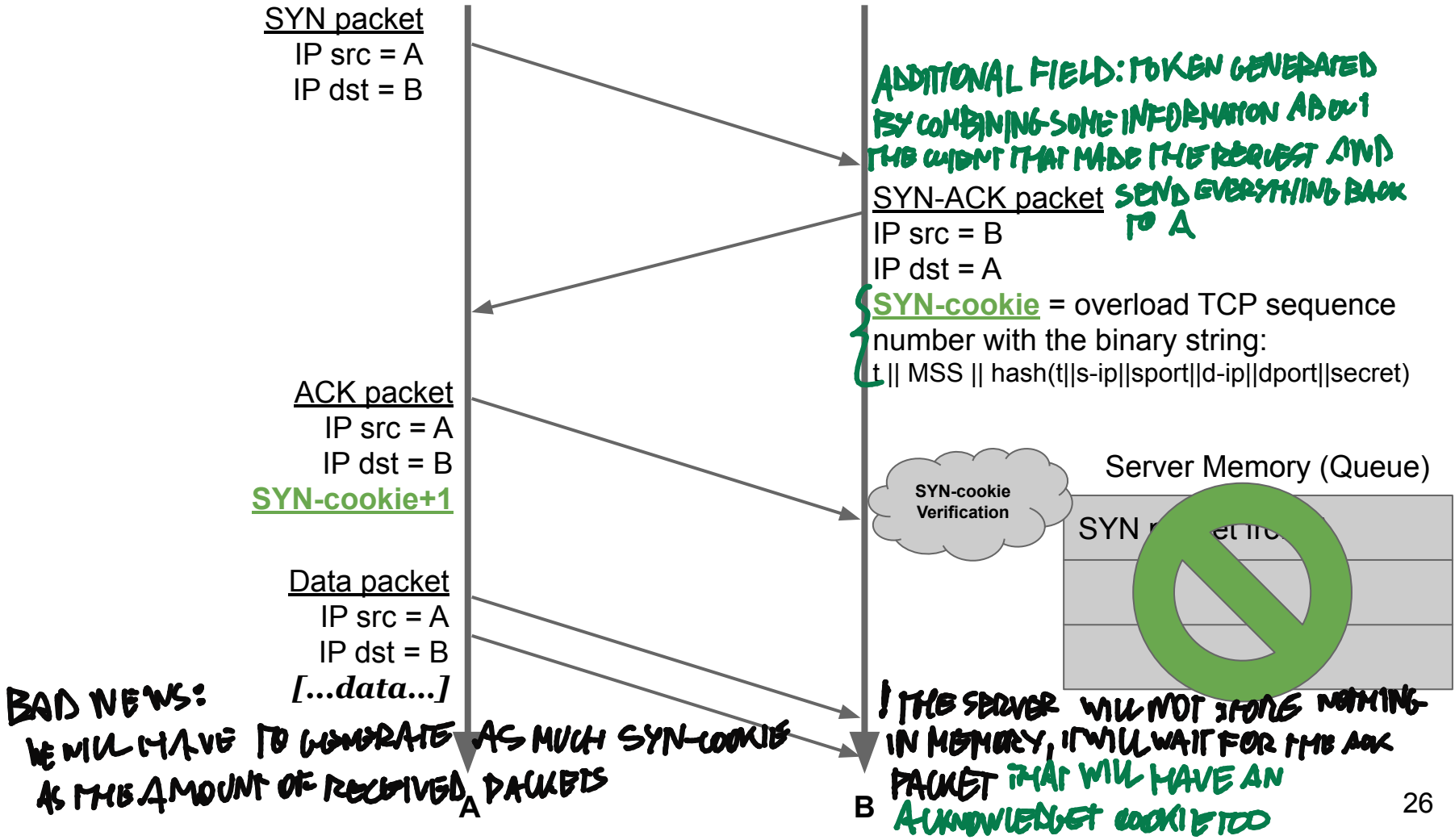
Attacker generates a high volume of SYN requests with **spoofed source address**.

Many half-open TCP/IP connections fill the queue.

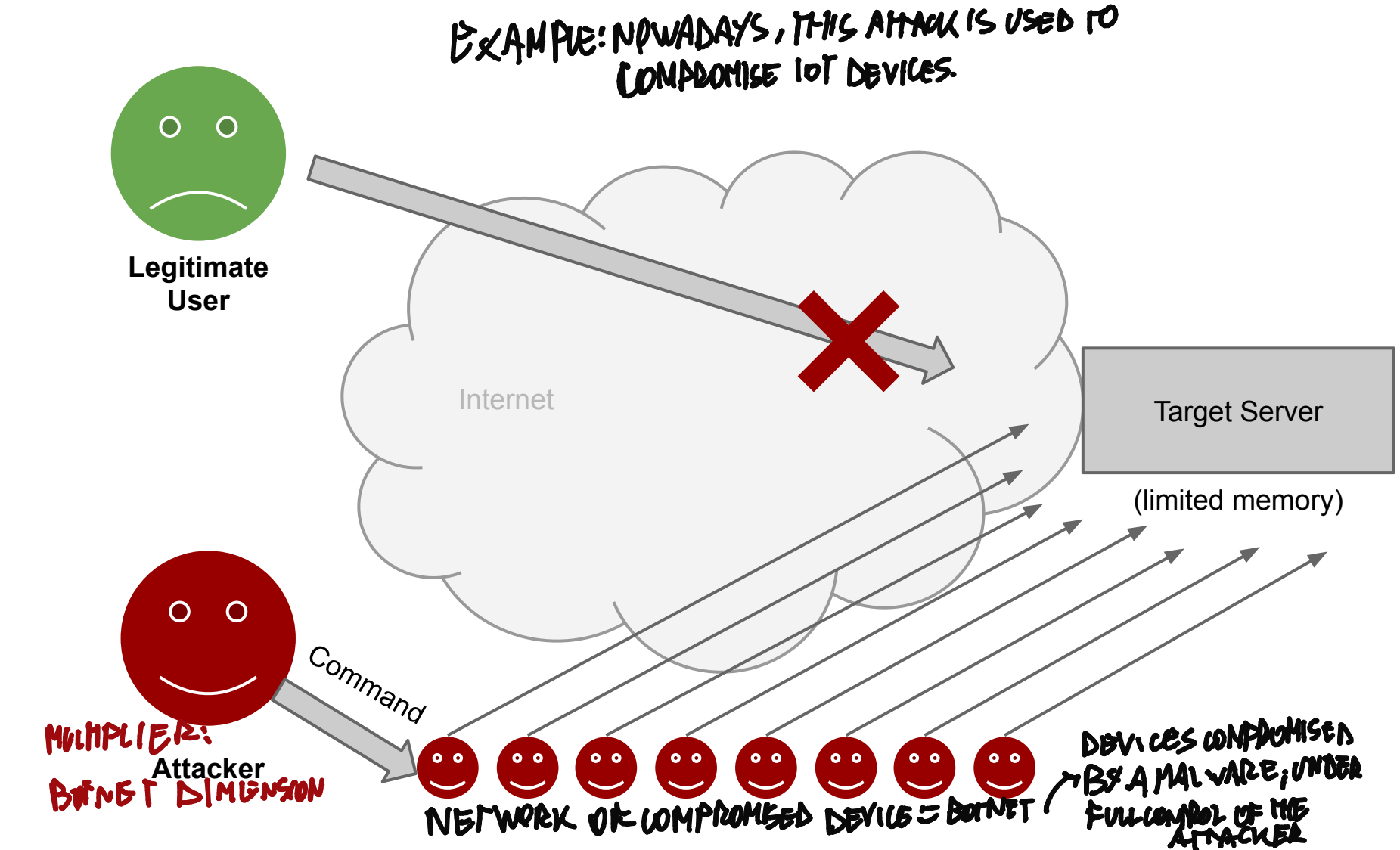
SYN requests from legitimate clients dropped.

Mitigation: SYN-cookies avoid this: reply with SYN+ACK but discard the half-open connection, and wait for a subsequent ACK http://cr.yp.to/syncookies.html

SYN Flood Attack: SYN cookies



Distributed DoS (DDoS)



The Botnet Case

Botnet: network of compromised computers, called *bots* (i.e., infected by malware).

C&C: dedicated command-and-control infrastructure so that the attacker (botmaster) can send commands to the bots.

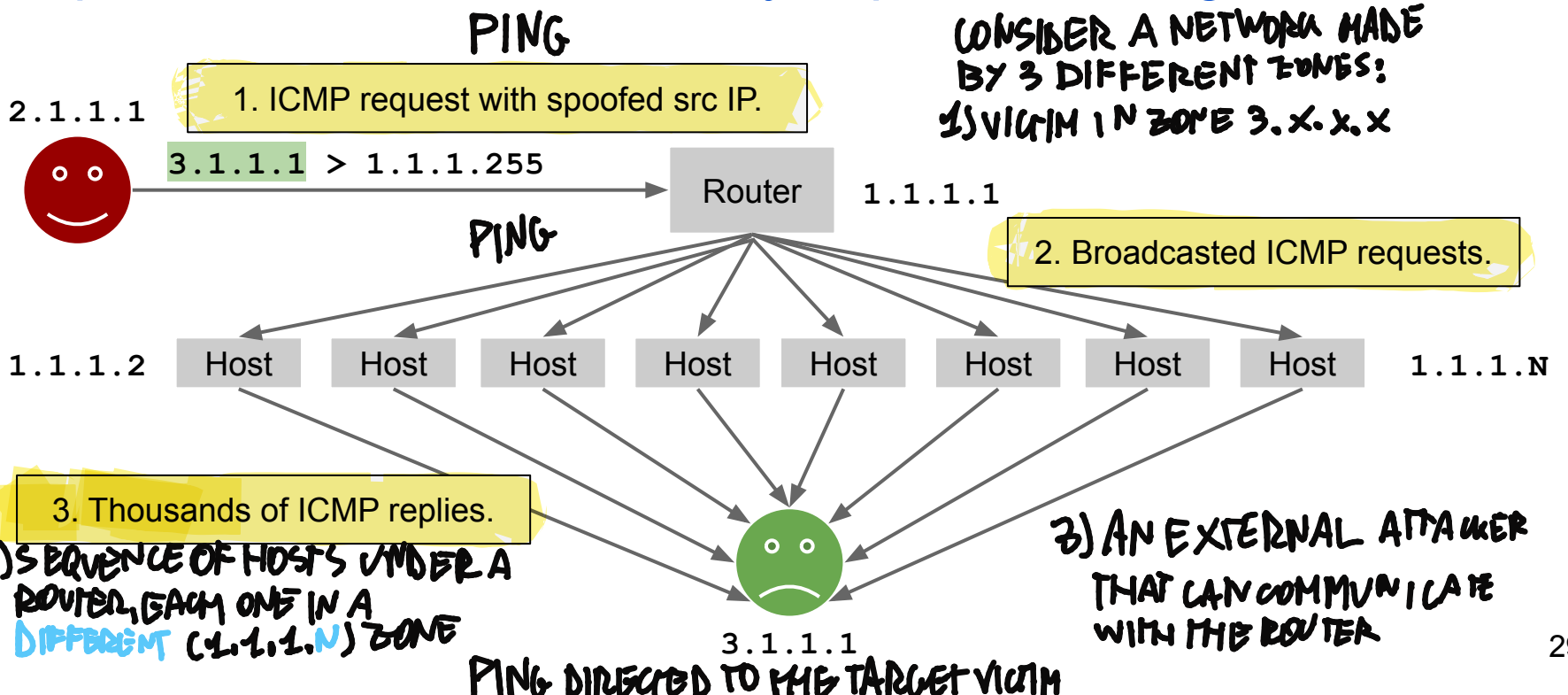
Various uses (e.g., spamming, phishing, info stealing), including DDoS-ing.

Distributed DoS (DDoS): Smurf

DDoS BY MEANS OF AN AMPLIFICATION ATTACK

The attacker sends ICMP packets with spoofed sender (victim) to a broadcast address.

http://www.hoobie.net/security/exploits/hacking/smurf.c



Amplification Hell

Bandwidth Amplification Factor

Protocol	BAF *all*	BAF 50%	BAF 10%	PAF *all*	Scenario
SNMP v2	6.3	8.6	11.3	1.00	*GetBulk* request
NTP	556.9	1083.2	4670.0	3.84	Request client statistics
DNS_{NS}	54.6	76.7	98.3	2.08	ANY lookup at author. NS
DNS_{OR}	28.7	41.2	64.1	1.32	ANY lookup at open resolv.
NetBios	3.8	4.5	4.9	1.00	Name resolution
SSDP	30.8	40.4	75.9	9.92	*SEARCH* request
CharGen	358.8	n/a	n/a	1.00	Character generation request
QOTD	140.3	n/a	n/a	1.00	Quote request
BitTorrent	3.8	5.3	10.3	1.58	File search
Kad	16.3	21.5	22.7	1.00	Peer list exchange
Quake 3	63.9	74.9	82.8	1.01	Server info exchange
Steam	5.5	6.9	14.7	1.12	Server info exchange
ZAv2	36.0	36.6	41.1	1.02	Peer list and cmd exchange
Sality	37.3	37.9	38.4	1.00	URL list exchange
Gameover	45.4	45.9	46.2	5.39	Peer and proxy exchange

http://www.christian-rossow.de/articles/Amplification_DDoS.php [paper and details]

Network-level Sniffing

Normally, a network interface card (NIC) intercepts and passes to the OS only the packets directed to that host's IP

IT IS POSSIBLE TO PUT THE NIC IN

Promiscuous mode: the NIC passess to the OS any packet read off of the wire → ALL THE PACKETS THAT ARE GOING TO BE TRANSMITED IN THE NETWORK

DSniff tool www.monkey.org/~dugsong/dsniff
ARP spoofing, MAC flooding, sniffing

More modern, complete and complex:
https://www.bettercap.org/

Sniffing vs time

- Originally: ethernet networks were on a shared media (BNC cable)
- RJ-45 cables changed shape but the medium was still shared because they ended in hubs
- **Hubs** broadcast traffic to every host in broadcast domain
- **Switches** selectively relay traffic to the wire corresponding to the correct NIC (Eth address based)
  - Performance, not security reasons

ADDRESS RESOLUTION PROTOCOL

POSSIBLE ONLY IF THE ATTACKER IS IN THE SAME NETWORK OF THE VICTIM, OTHERWISE YOU CAN'T RECEIVE BROADCAST MESSAGES

ARP Spoofing (& Cache Poisoning)

INFORMATIONS RELATED TO MAPPING IP/MAC ARE STORED IN THE ARP CACHE

The ARP maps 32-bits IPv4 addresses to 48-bits hardware, or MAC, addresses.

- ARP request `"where is 192.168.0.1?"` (WAIT FOR HIS MAC ADDRESS)
- ARP reply (ANYONE WHO KNOWS THE MAC CAN REPLY) `"192.168.0.1 is at b4:e9:b0:c9:81:03"`

! EVERY SINGLE TIME YOU CONNECT A NETWORK AND WANT TO SEND A PACKAGE TO ANOTHER DESTINATION, BESIDES KNOWING IP ADDRESS IT IS REQUIRED TO KNOW THE MAC ADDRESS

First come, first trusted! An attacker can forge replies easily: **lack of authentication**.

IN THE ARP CACHE, "FAKE" MAPPINGS CAN BE STORED

Each host **caches** the replies: try `arp -a`

ATTACKER GOALS CAN BE TO REDIRECT ALL THE TRAFFIC THAT WAS SUPPOSED TO GO TO ITS GO ABOUT MACHINE TO HIS ONE

On the Victim's Machine

```
C:\> arp -d 15.1.1.1              # clear the record for 15.1.1.1
C:\> ping -n 1 15.1.1.1           # try to reach 15.1.1.1
```

SINCE WE HAD CLEARED THE RECORDS, WE DON'T KNOW THE MAC

```
Pinging 15.1.1.1 with 32 bytes of data:
Reply from 15.1.1.1: bytes=32 time<10ms TTL=255

# under the hood, the ARP layer has resolved the MAC address
# that corresponds to 15.1.1.1

C:\> arp -a

Interface: 15.1.1.26 on Interface 2
  Internet Address      Physical Address      Type
  15.1.1.1              00-10-83-34-29-72     dynamic
  15.1.1.25             00-04-4e-f2-d8-01     dynamic
```

On the Attacker's Machine

Suppose an attacker is already in the network and "talk" it with a sequence of messages in broadcast that tells a different MAC address

Tell every host that 15.1.1.1 is at the attacker's NIC, which is 0:4:4e:f2:d8:01.

```
[d3v1lz@host]# ./arpspoof 15.1.1.1
0:4:43:f2:d8:1 ff:ff:ff:ff:ff:ff 0806 42: arp reply 15.1.1.1 is-at
0:4:4e:f2:d8:1
0:4:43:f2:d8:1 ff:ff:ff:ff:ff:ff 0806 42: arp reply 15.1.1.1 is-at
0:4:4e:f2:d8:1
0:4:43:f2:d8:1 ff:ff:ff:ff:ff:ff 0806 42: arp reply 15.1.1.1 is-at
0:4:4e:f2:d8:1
0:4:43:f2:d8:1 ff:ff:ff:ff:ff:ff 0806 42: arp reply 15.1.1.1 is-at
0:4:4e:f2:d8:1
```

When the user asks 15.1.1.1's MAC, he'll not have the correct one

...Back on the Victim's Machine

```
C:\> arp -a                 # the ARP cache has been poisoned

Interface: 15.1.1.26 on Interface 2
  Internet Address      Physical Address      Type
  15.1.1.1              00-04-4e-f2-d8-01     dynamic
  15.1.1.25             00-04-4e-f2-d8-01     dynamic
```

Possible Mitigations?

- CHECK IF $\exists IP_i, IP_j, i \neq j \mid MAC_i = MAC_j \Rightarrow$ ANOMALOUS ASSIGNMENT

...Back on the Victim's Machine

```
C:\> arp -a                 # the ARP cache has been poisoned

Interface: 15.1.1.26 on Interface 2
  Internet Address      Physical Address      Type
  15.1.1.1              00-04-4e-f2-d8-01     dynamic
  15.1.1.25             00-04-4e-f2-d8-01     dynamic
```

Possible Mitigations

- Check responses before trusting (if they conflict with existing addresses mappings)
- Add a SEQ/ID number in the request
- ...

CAM Table

- Switches use **CAM tables** to know (i.e., cache) which MAC addresses are on which ports
- Switches are just as vulnerable to ARP spoofing! WHAT SHOULD AN ATTACKER DO?

```
Switch#show mac address-table

          Mac Address Table
-------------------------------------------

Vlan    Mac Address       Type        Ports
----    -----------       --------    -----
  10    AAAA.AAAA.AAAA    DYNAMIC     Fa0/1
  20    BBBB.BBBB.BBBB    DYNAMIC     Fa0/2
  30    CCCC.CCCC.CCCC    STATIC      Fa0/3
```

Filling up a CAM Table

- Switches use **CAM tables** to know (i.e., cache) which MAC addresses are on which ports
- **Dsniff (macof)** can generate ~155k spoofed packets a minute: fills the CAM table in seconds (**MAC flooding**)
- CAM table **full:** cannot cache ARP replies and must forward everything to every port (like a hub does).

Mitigation: PORT Security (CISCO terminology)

Abusing the Spanning Tree Protocol

The STP (802.1d) avoids loops on switched networks by building a spanning tree (ST).

Switches decide how to build the ST by exchanging **BPDU** (bridge protocol data unit) **packets** to elect the root node.

BPDU packets are **not authenticated**, so, an attacker can change the shape of the tree for sniffing or ARP spoofing purposes.

BESIDES IP ADDRESSES, EVEN MAC CAN BE SPOOFED

IP Address Spoofing (UDP/ICMP)

The IP source address is **not authenticated**.

Changing it in **UDP or ICMP** packets is **easy**.

However, the attacker will not see the answers (e.g., he/she is on a different network), because they will be sent to the spoofed host (**blind spoofing**).

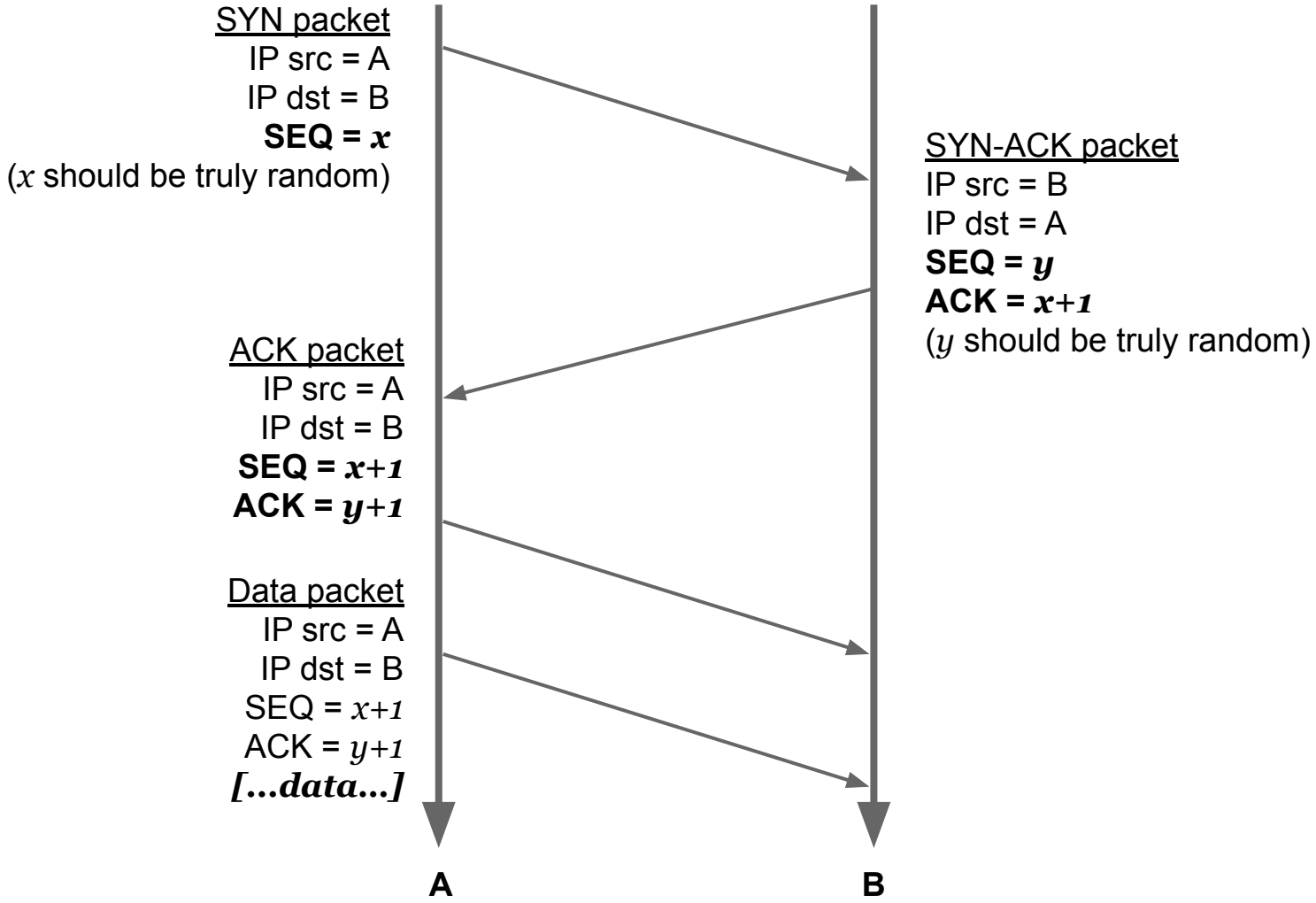
But if the attacker is on the same network, s(he) can sniff the rest, or use ARP spoofing.

For TCP it is not the same....

IP Address Spoofing (TCP): TCP Sequence Number Guessing

- TCP uses **sequence numbers** for reordering and acknowledging packets.
- A semi-random **Initial Sequence Number** (**ISN**) is chosen.

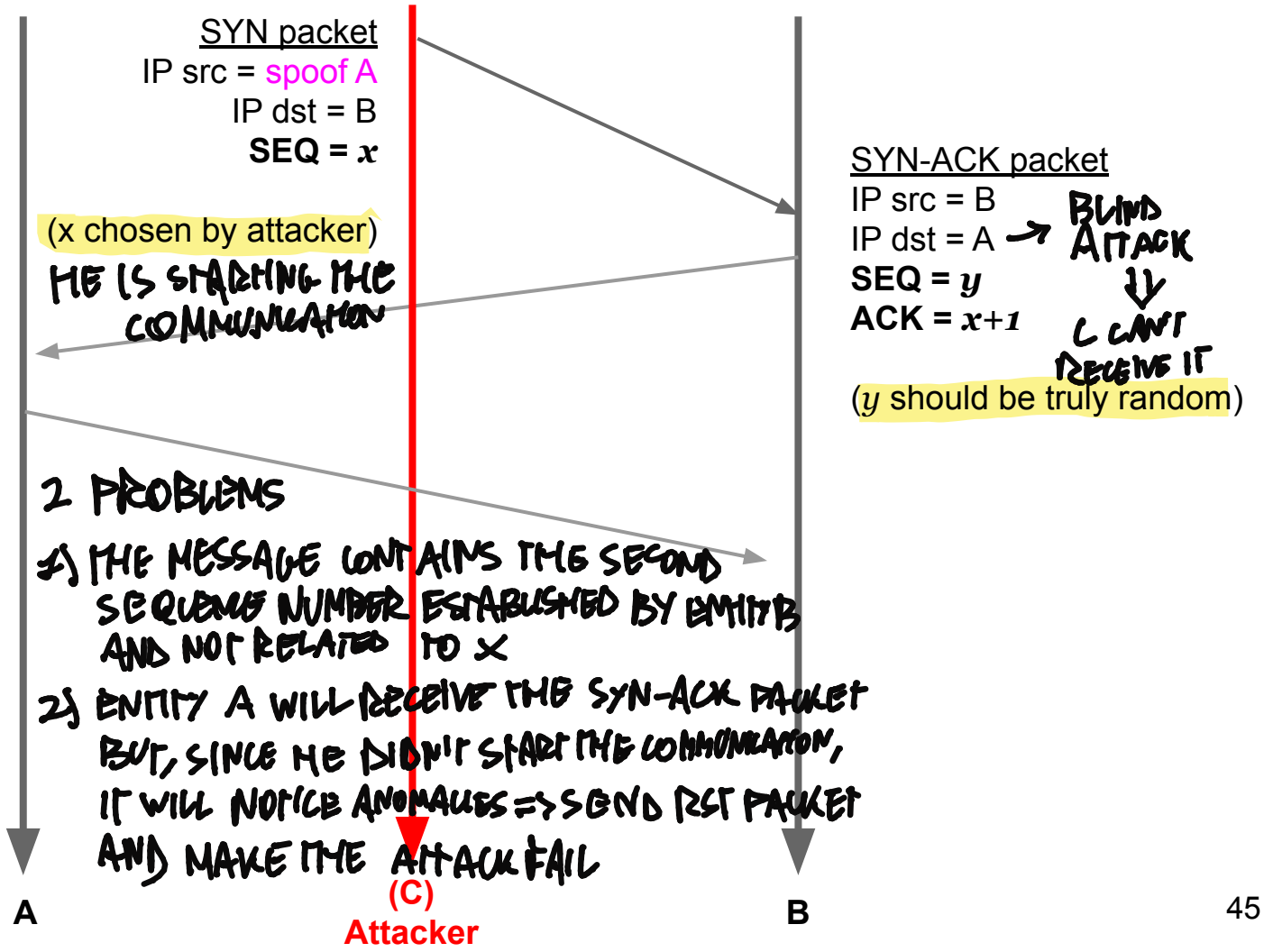
Recall the Three Way Handshake



TCP Sequence Number Guessing

- TCP uses **sequence numbers** for reordering and acknowledging packets.
- A semi-random Initial Sequence Number (ISN) is chosen.
- If a **blind spoofer** can predict the ISN, he can blindly complete the 3-way handshake without seeing the answers.
- However, the spoofed source should not receive the response packets, otherwise it might answer with a RST.

TCP/IP (Blind) Spoofing Attack: The RST packet Issue

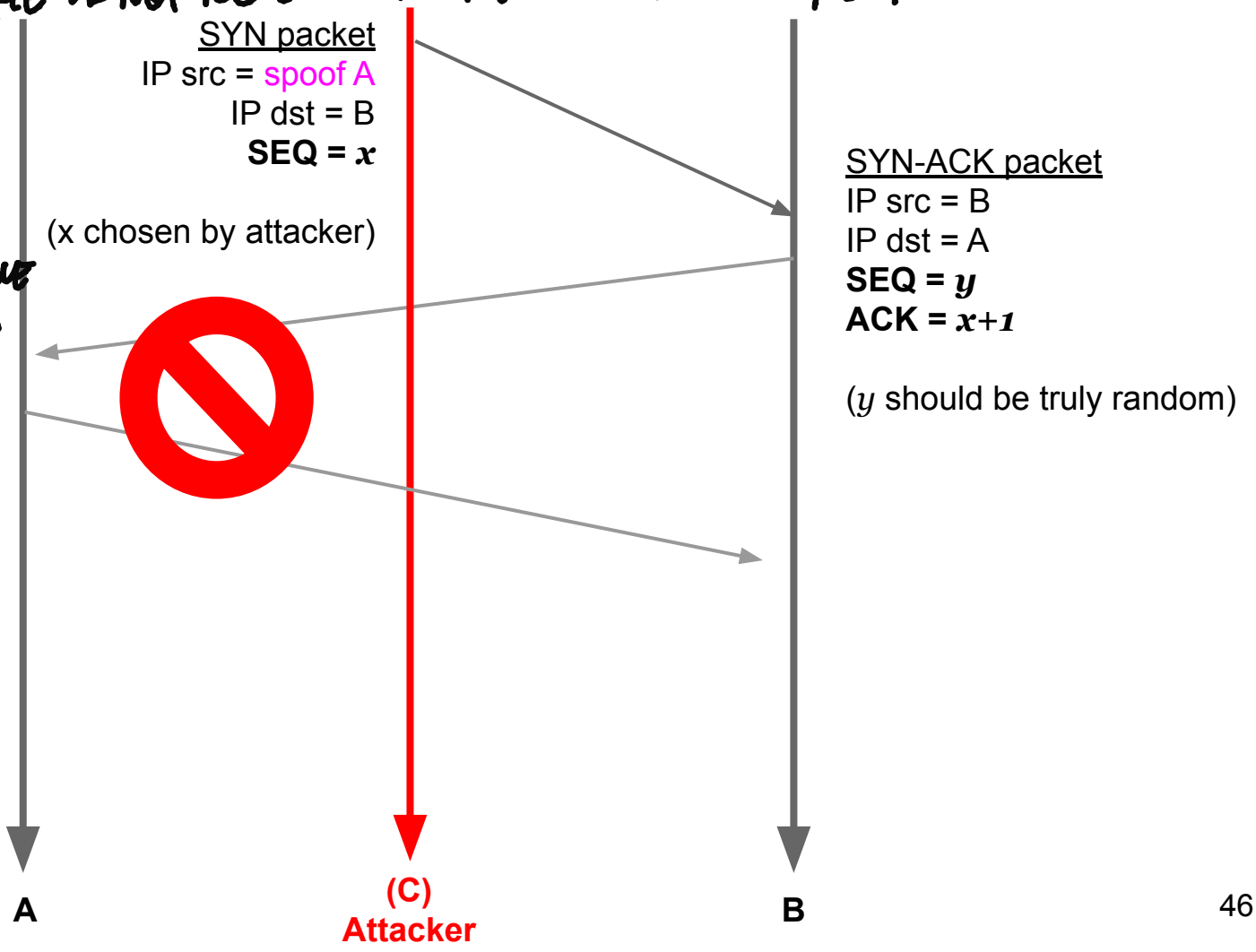


(if the spoofed machine receives a packet, it might reply with RST)

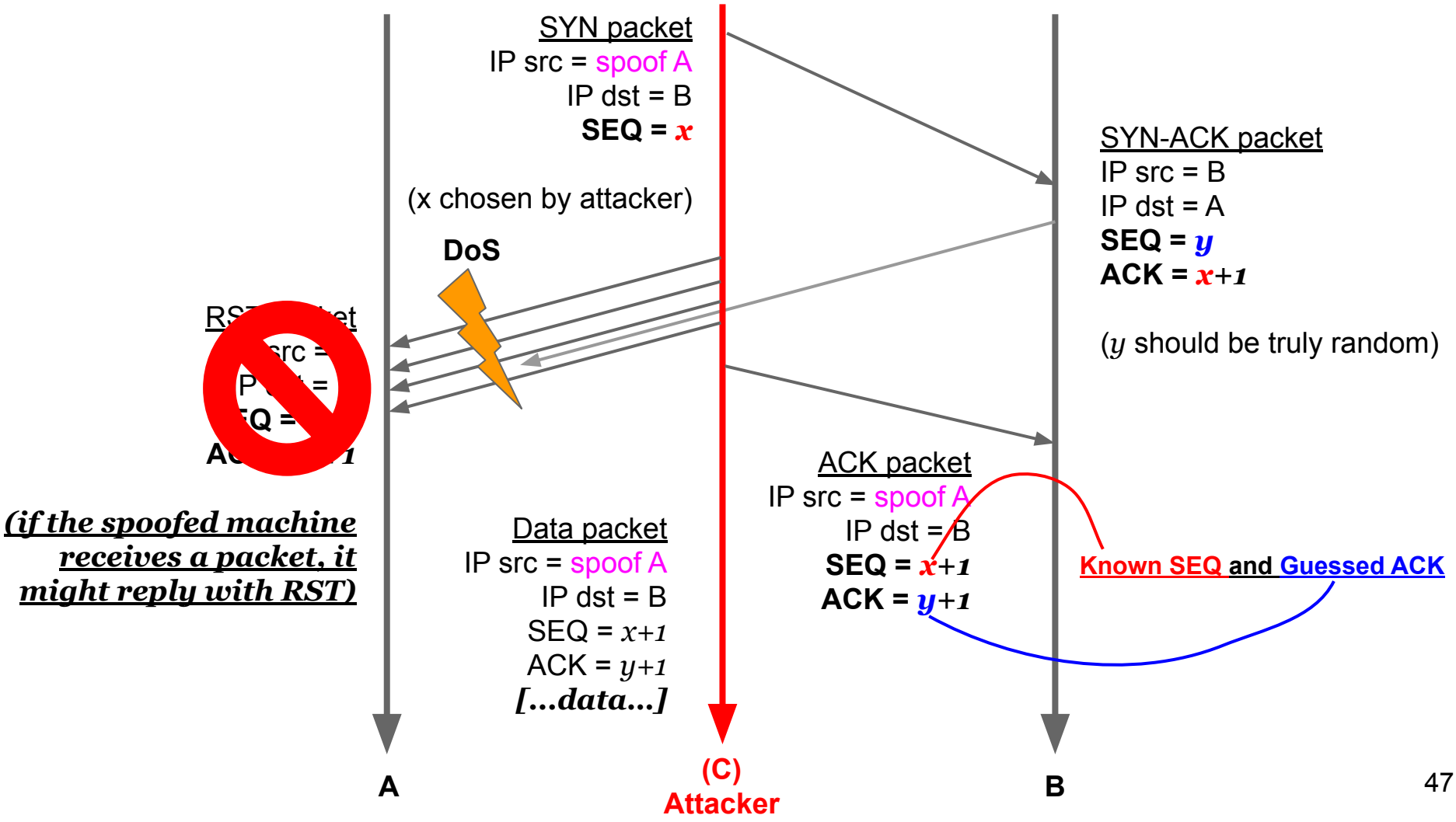
TCP/IP (Blind) Spoofing Attack: The RST packet Issue

TO AVOID THE ATTACK FAILURE, THE ATTACKER CAN PUT A DOS MAKING A UNAVAILABLE, SO HE'LL NOT RECEIVE THE MESSAGE. TO DO SO, C HAS TO GUESS Y.

THIS CAN HAPPEN IF THE "RANDOM GENERATOR" OF Y IS WEAK, VULNERABLE OR IT'S IMPLEMENTED IN SUCH A WAY THAT C CAN PREDICT THE NEXT Y



TCP/IP (Blind) Spoofing Attack: Sequence Number Guessing



How Random is Random?

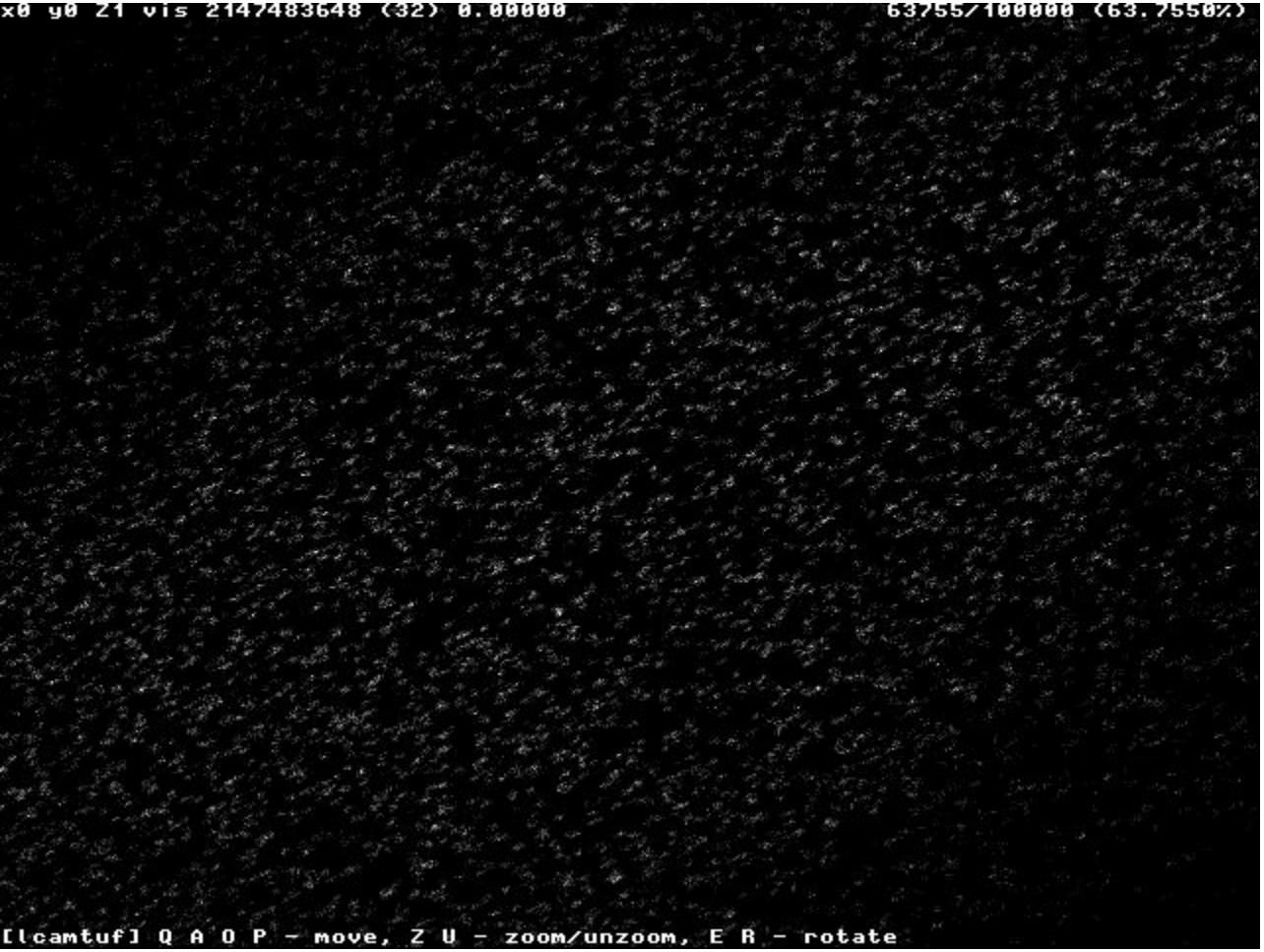
In 1995 Kevin Mitnick used a TCP/IP spoofing attack as the first step to break into Tsutomu Shimomura's machine.

Back then, TCP implementations used easily guessable ISNs, so Mitnick managed to send the right SYN-ACK-ACK to Shimomura's computer and hijack the connection.

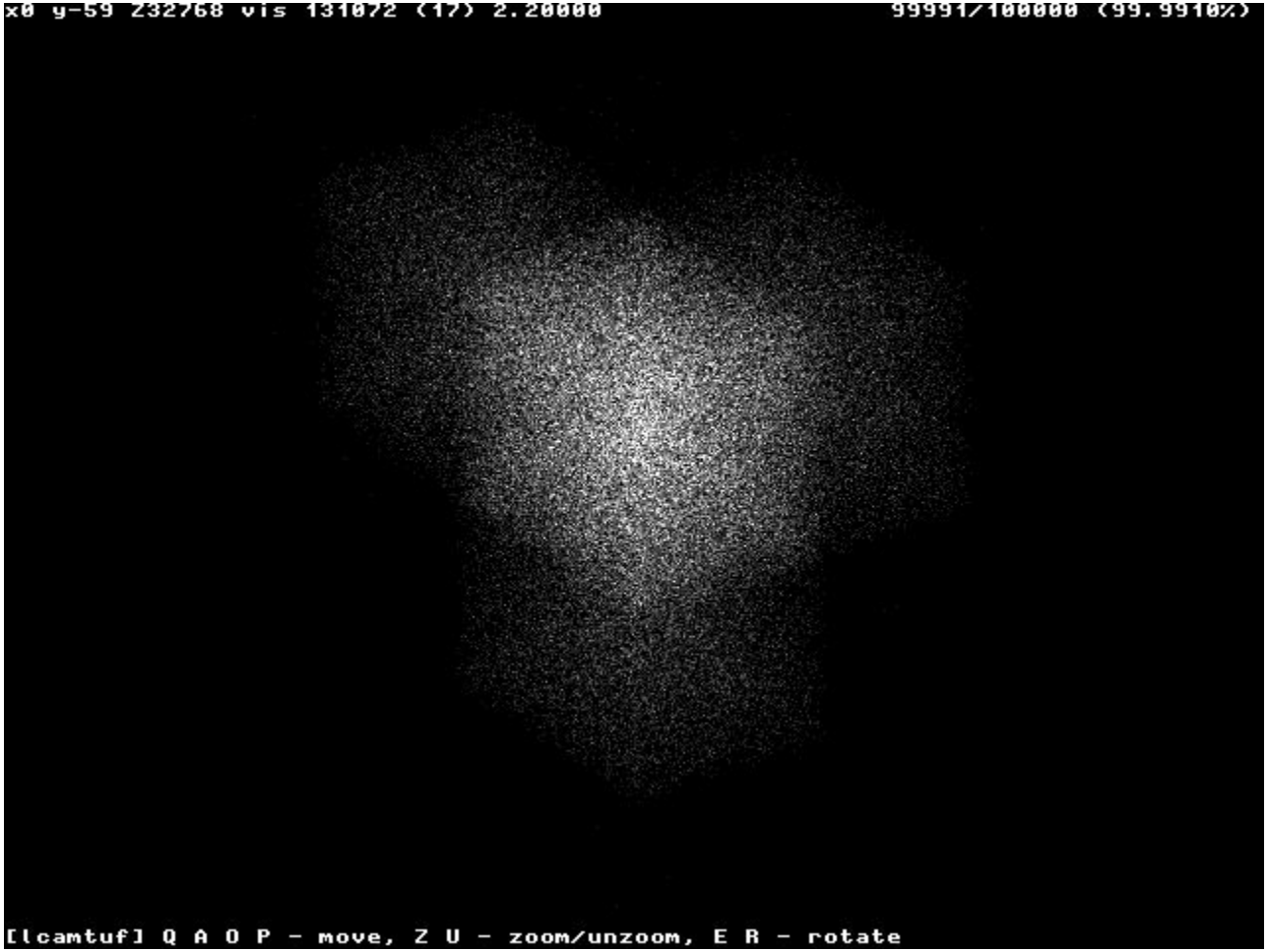
What changed since 1994?

http://lcamtuf.coredump.cx/newtcp/

Netware 6 SP3 ISN Distribution



Windows XP SP2 ISN Distribution



IRIX 6.5.15 ISN Distribution

Netware 6 ISN Distribution

*BSD family ISN Distribution

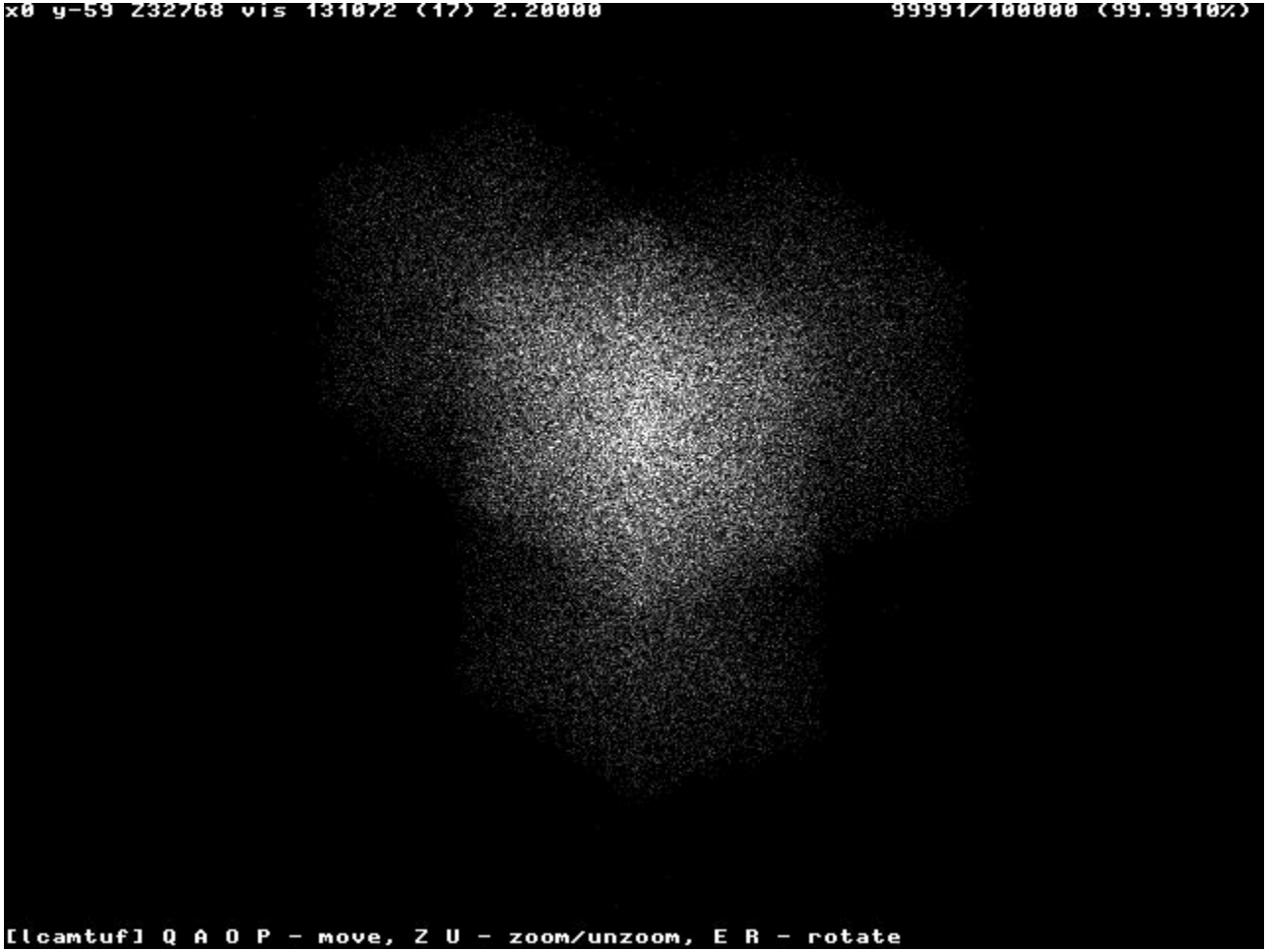
5°: IRIX 6.5.15 ISN Distribution

OS Name:	IRIX 6.5.15 tcpiss_md5=0
R1 radius:	0
Average R2:	93
Average N:	297
Average error:	0
Attack feasibility:	100.00%

4°: Netware 6 ISN Distribution

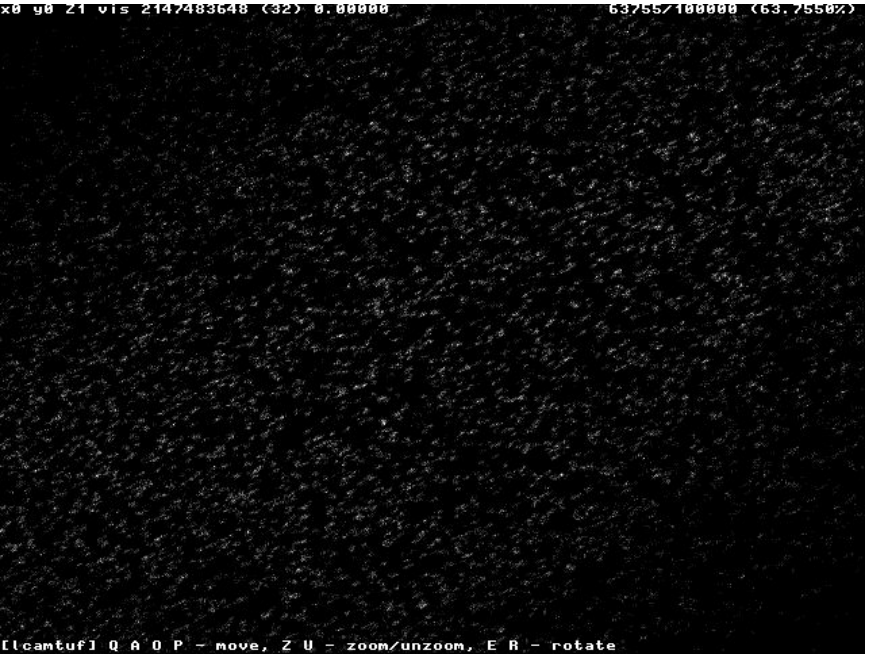
OS Name:	Netware 6
R1 radius:	10
Average R2:	2484
Average N:	11
Average error:	0
Attack feasibility:	90.00%

3°: Windows XP SP2 ISN Distribution



OS Name:	Windows XP
R1 radius:	10
Average R2:	251
Average N:	179
Average error:	279
Attack feasibility:	12.00%

1° - 2°: Netware 6 SP3 & *BSD ISN Distribution



OS Name:	Netware 6 (SP3)
R1 radius:	100000
Average R2:	999
Average N:	34
Average error:	n/a
Attack feasibility:	0.00%

OS Name:	FreeBSD 4.6
R1 radius:	1000000
Average R2:	101
Average N:	279
Average error:	n/a
Attack feasibility:	0.00%

TCP Session Hijacking

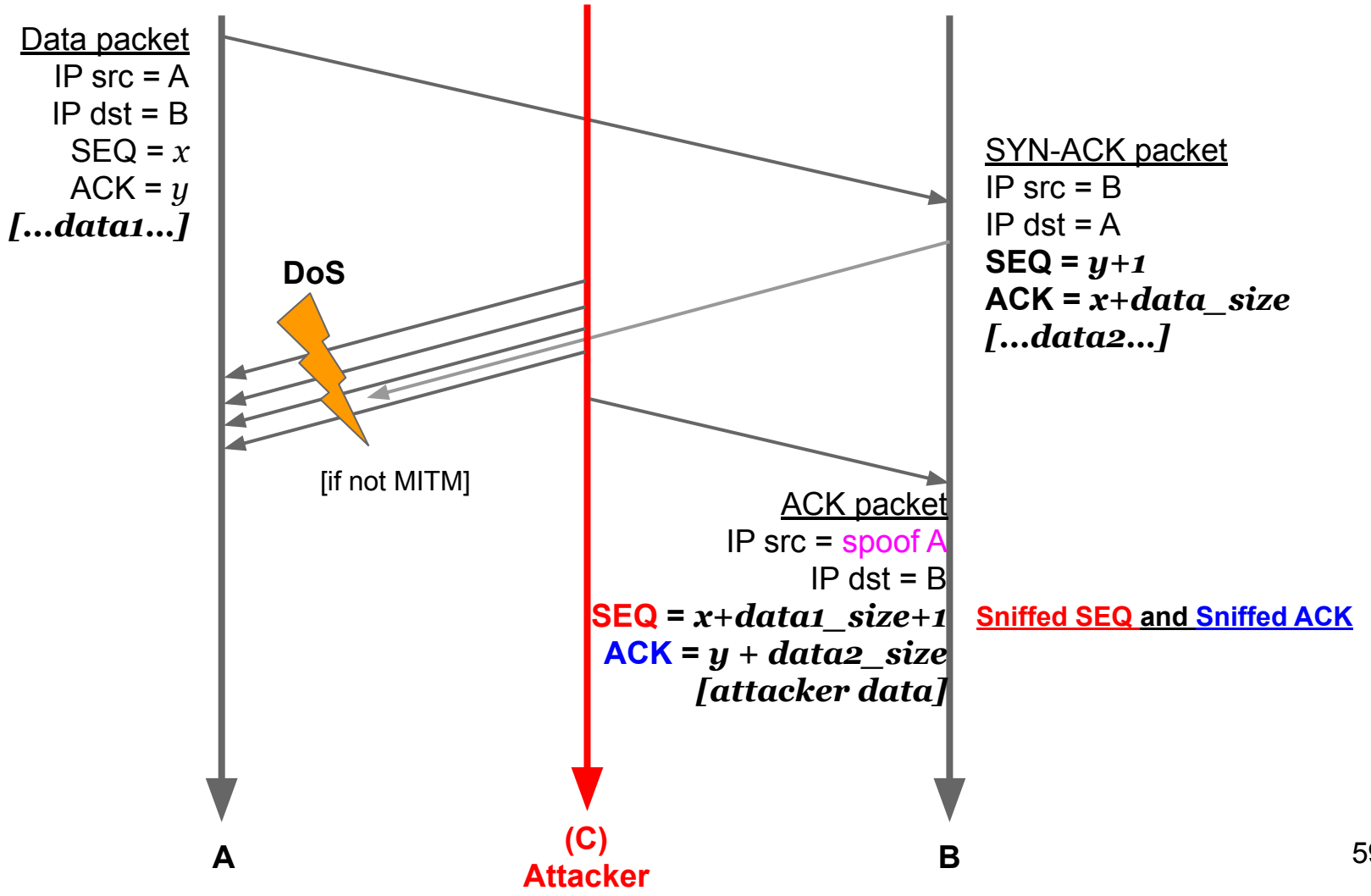
FULL SPOOFING → ATTACKER IN THE SAME VICTIM NETWORK

Taking over an active TCP session.

If the attacker (C) can sniff the packets:

1. **C** follows the conversation of **A** and **B** **recording** the sequence numbers.
2. **C** somehow **disrupts A**'s connection (e.g., SYN Flood): **A** sees only a “random” disruption of service.
3. **C** takes over the dialogue with **B** by **spoofing** **A** address and using a correct ISN. **B** suspects nothing.

TCP Session Hijacking Visualized



TCP Session Hijacking (2)

A lot of tools (e.g., hunt/dsniff) implement this attack automatically.

The attacker can avoid disrupting B's session and just inject things in the flow only if s(he) is a **man in the middle**

- It can control/resync all the traffic flowing through.

What's a **man in the middle**?

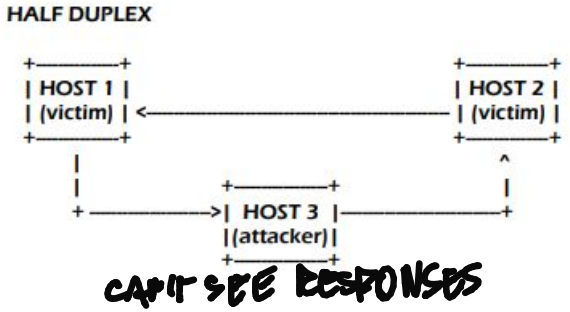
MITM: Man In The Middle

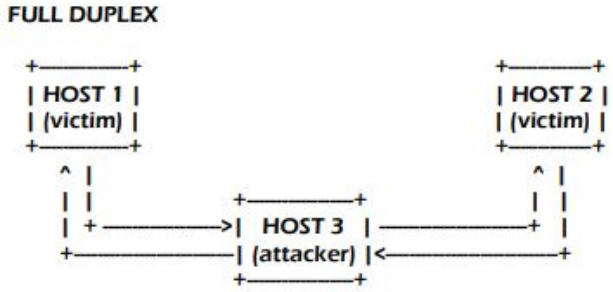
A broad category comprising all the attacks where an attacker can impersonate the server with respect to the client and vice-versa.

- physical or logical
- full- or half-duplex (blind)

4 TYPES

PHYSICAL CONTROL OF THE MACHINE IN WHICH A VICTIM RESIDES





What happens if the attacker is able to ARP-spoof the gateway of a LAN? :-)

RECALL

Addressing So Far

- MAC addresses for hardware
- IP addresses for Internet routing

Problems

- Humans are bad at remembering strings of numbers
- Need of a human-friendly naming system

RECALL

Requirements for Naming System

- As short as possible
- Easy to memorize (i.e., not arbitrary)
- Unique
- Customizable
- Reflect organizational structure (Hierarchy)
- Quickly translate to and from the existing, "computer-friendly" addressing systems
- Address specific resources/services

RECALL

Domain Names System (DNS) (1/2)

- Maps/Translates "domain names" to numerical IP addresses
  - You can type www.google.com into the browser, and the browser will know to go to 173.194.33.179
- But how might this be done?
  - Some sort of hash (not really practical)
  - A file of all of the mappings (not really practical)

RECALL

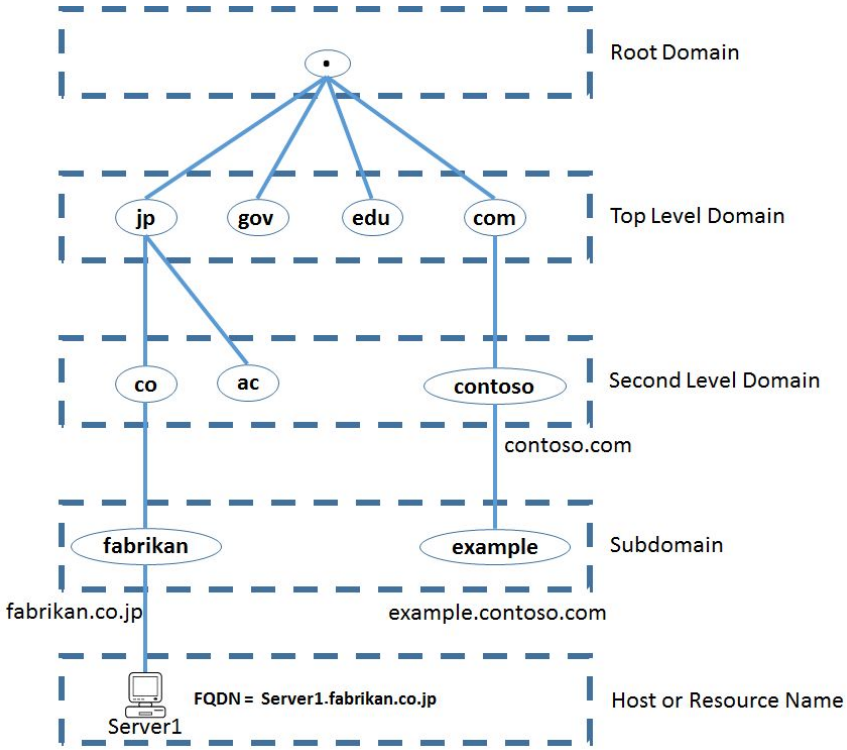
Domain Names System (2/2)

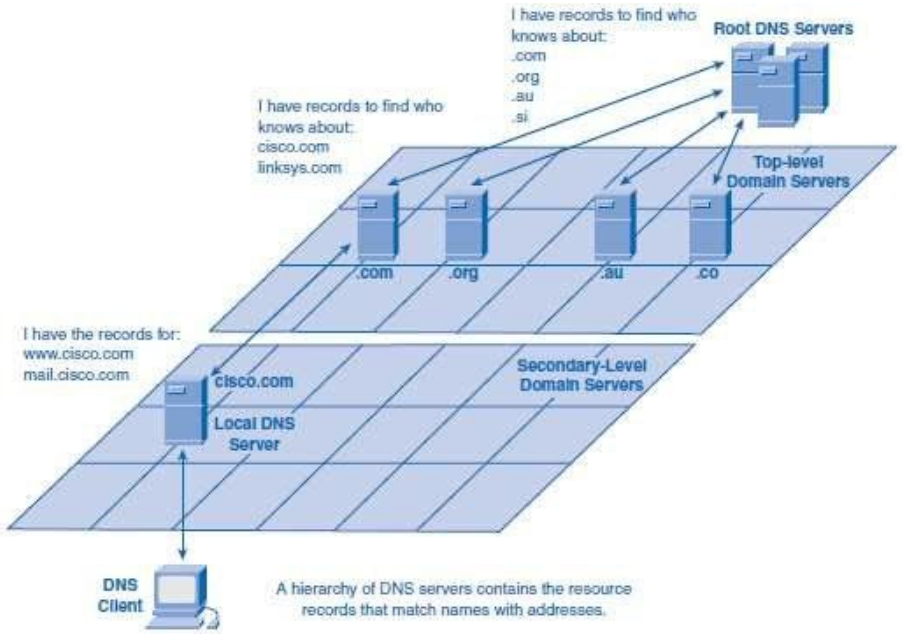
- **Distributed Database**
- **Hierarchy** of servers that provide the mappings
  - Each server keeps a small cache of the mappings
- Based on UDP (Port 53)
- Messages are not authenticated.
- When a domain name is used/requested and isn't in the local cache, the system queries a DNS server

RECALL

Hierarchical DNS Servers

A hierarchy of DNS servers that contains the resource records to match DN with IP

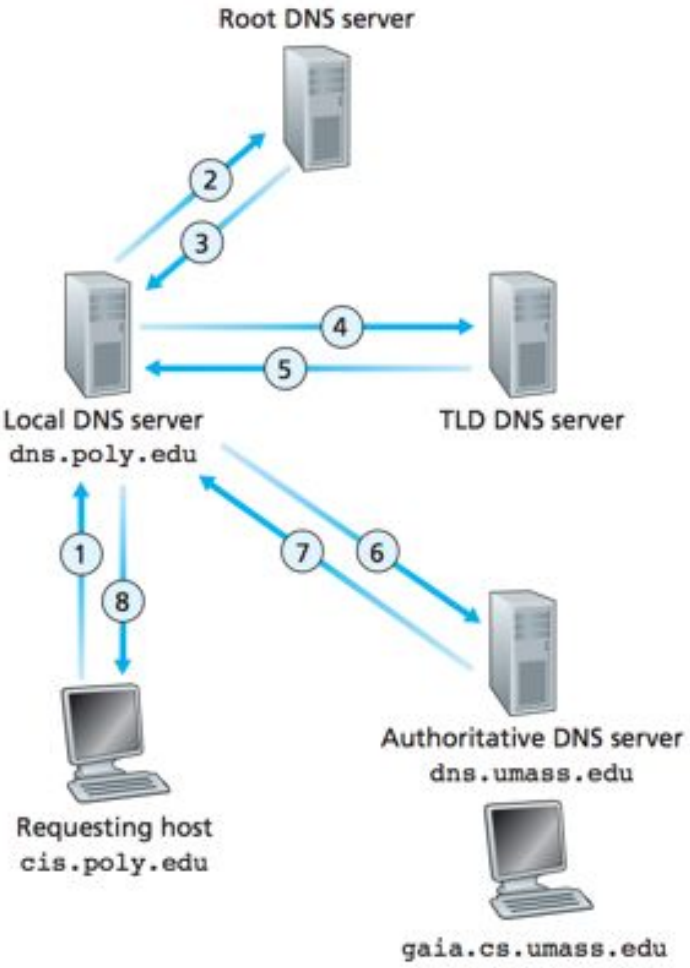




RECALL

Resolving a Domain Name (1/2)

- If I type sports.polimi.com, what happens?
  - Check /etc/hosts
  - Check DNS cache
  - Check local DNS server
  - Go through the hierarchy:
    - Ask . DNS root server
    - Ask .com TLD/SLD (Top/Second Level Domain) server
    - Ask the **Authoritative polimi.com's NS**
  - Send HTTP request to the IP address obtained

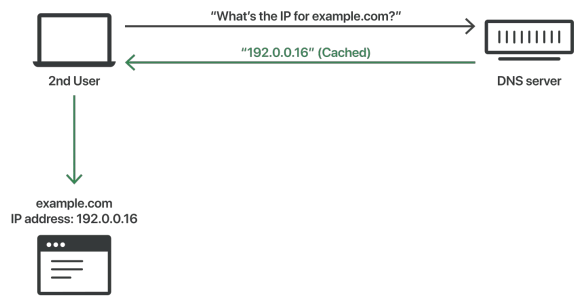


RECALL

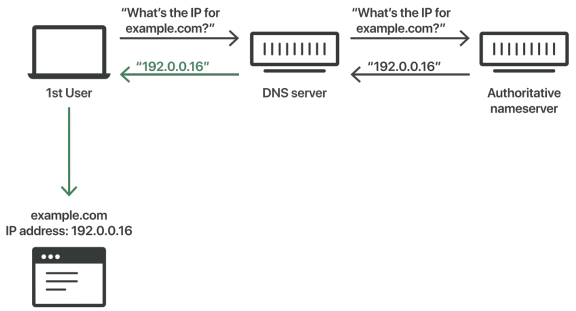
Resolving a Domain Name (2/2)

When a **non-authoritative** DNS server receives a request to resolve a domain name:

- if the answer is **cached**, it answers



- If **no answer** in cache:
  - **Recursive:** resolves the name on behalf of the client.



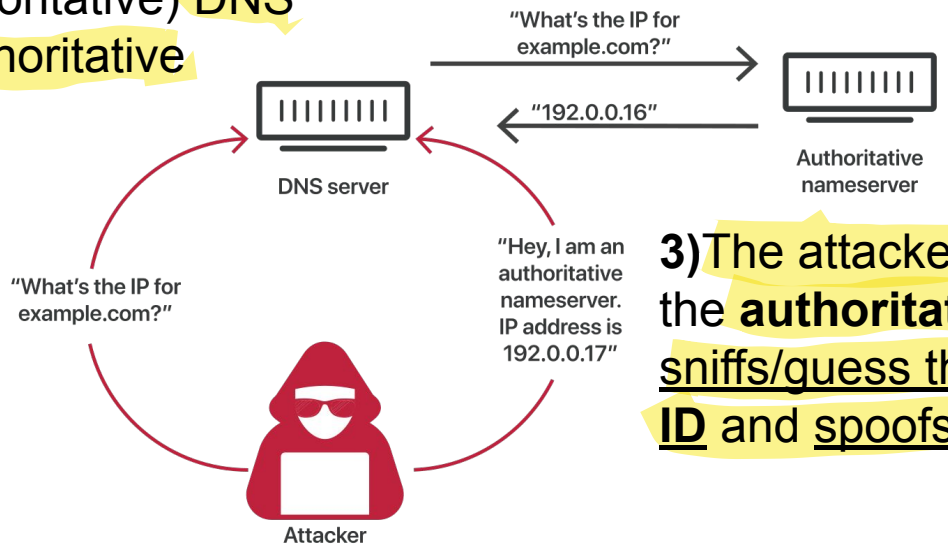
  - **Iterative:** gives the authoritative DNS address.

DNS (Cache) Poisoning Attack

Poison the cache of a non authoritative DNS server

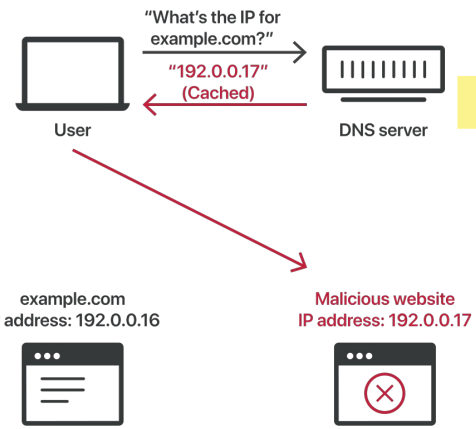
2) The victim (non authoritative) DNS server contacts the authoritative server.

1)The attacker makes a **recursive query** to the victim DNS server.



3)The attacker, **impersonating** the **authoritative** DNS server, sniffs/guess the the **DNS query ID** and spoofs the answer.

All clients that request to resolve the DN to the poisoned DNS server are redirected to the malicious website



4) The victim DNS server trusts and caches the malicious record **[POISONED].**

DNS (Cache) Poisoning Attack

1. The attacker makes a **recursive query** to the victim DNS server.
2. The victim (non authoritative) DNS server contacts the authoritative server.
3. The attacker, **impersonating** the **authoritative** DNS server, spoofs the answer (before the legitimate one).
4. The victim DNS server trusts and caches the malicious record **[POISONED]**.

In the spoofed answer we need to use the **ID of the DNS query** initiated by the victim DNS server (step 2.).

Guess? Bruteforce? (Kaminsky, 2008)

https://duo.com/blog/the-great-dns-vulnerability-of-2008-by-dan-kaminsky

Dynamic Host Configuration Protocol (DHCP) (SPOOFING)

RECALL

Protocol that dynamically assigns IP addresses (and network parameters) to each device in a network:

- It automatically assigns a new IP address when a computer is plugged into the network

It allows network administrators to supervise and distribute configuration parameters for network hosts from a central point:

- IP address
- Router
- Subnet Mask
- ...

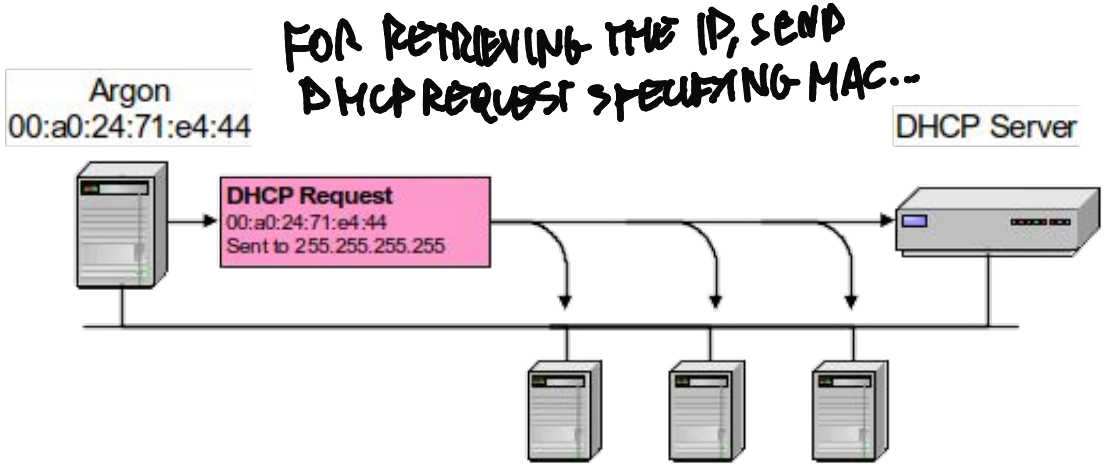
RECALL

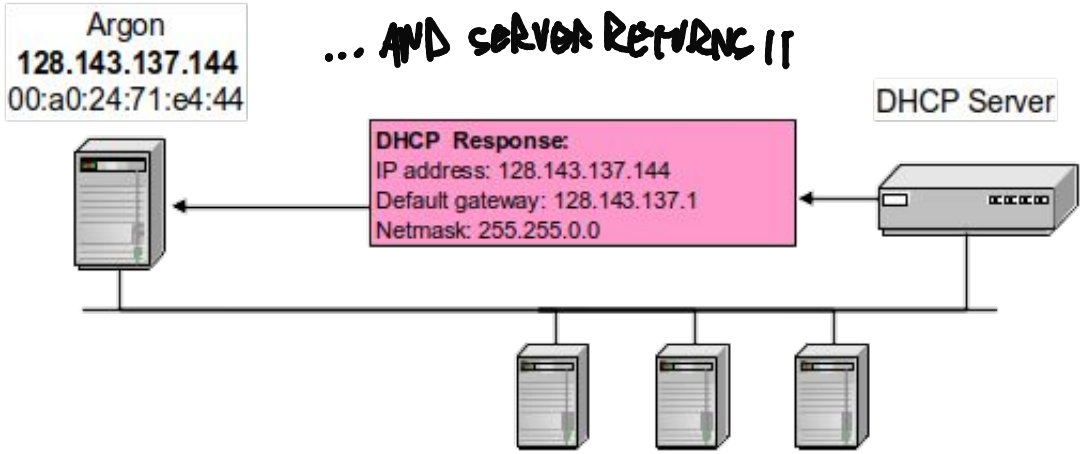
Limitations of DHCP

- Again. DHCP is **not authenticated**
  - Performance?
- Based on UDP.
- DHCP server must run continually
  - When DHCP server is unavailable, client is unable to access enterprises network.

RECALL

DHCP Interaction (Simplified)

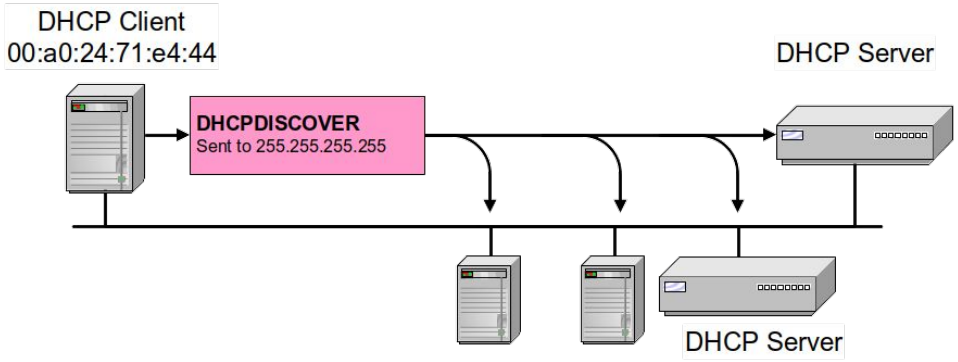




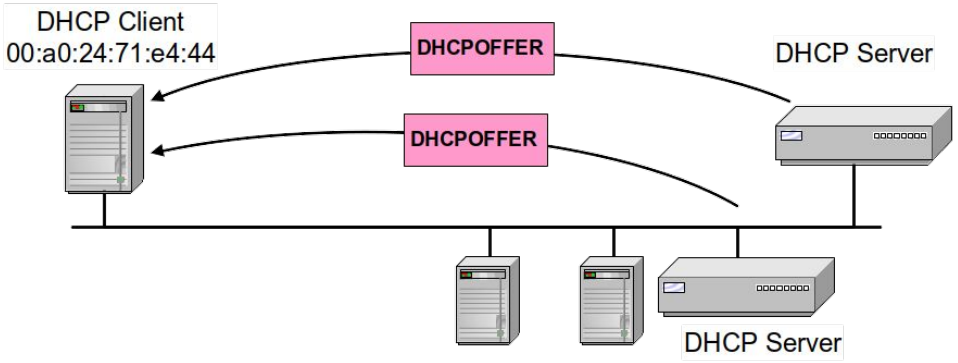
RECALL

DHCP Operation (1/3)

- **DCHP DISCOVER**



- **DCHP OFFER**



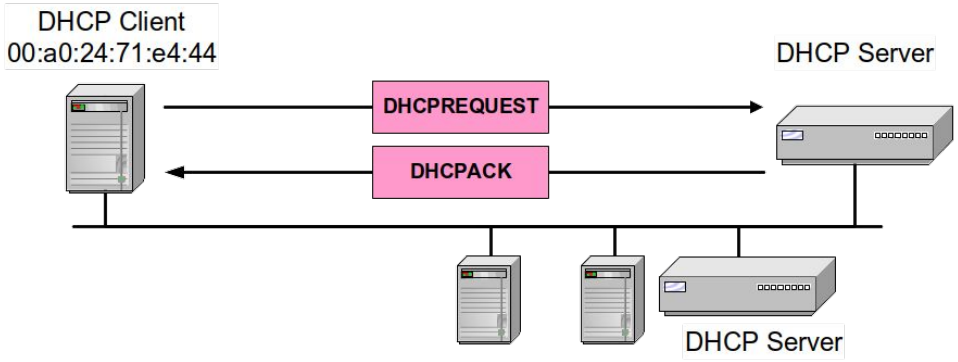
RECALL

DHCP Operation (2/3)

DCHP DISCOVER

- At this time, the DHCP client can start to use the IP address

If DHCP server sends **DHCP ACK**, then address is assigned.

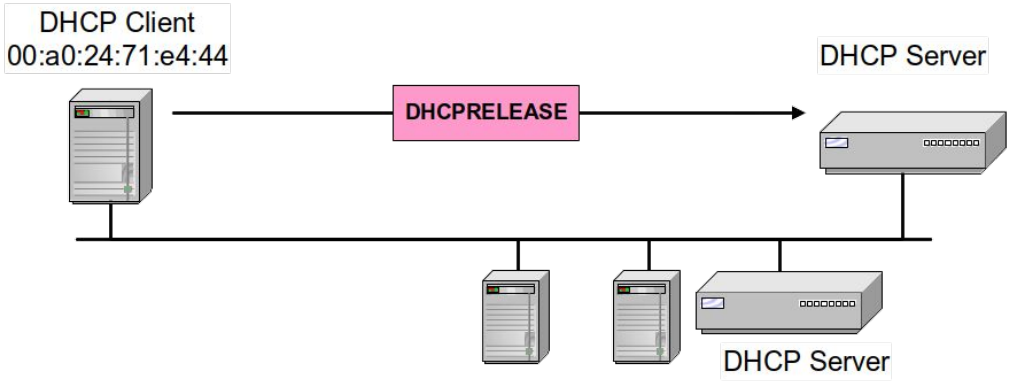


RECALL

DHCP Operation (3/3)

DCHP RELEASE

The DHCP client releases the IP address



DHCP Poisoning Attack

DHCP is an **unauthenticated protocol**

The attacker can intercept the “DHCP requests”, be the first to answer, and client will believe that answer.

With a (spoofed) “DHCP response”, the attacker can set:

- IP address,
- DNS addresses,
- default gateway of the victim client.

RECALL

Internet Control Message Protocol

ICMP is used to send debugging information and error reports between hosts, routers and other network devices at IP level.

ICMP messages can be:

- Requests
- Responses
- Error messages

RECALL

ICMP Messages

- Address mask request/reply:
  - used by diskless systems to obtain the network mask at boot time.
- Timestamp request/reply:
  - used to synchronize clocks.
- Source quench:
  - used to inform about traffic overloads.
- Parameter problem:
  - used to inform about errors in the IP datagram fields.
- **Echo request/reply:**
  - used to test connectivity (ping).
- Time exceeded:
  - used to report expired datagrams (TTL = 0).
- Redirect:
  - **used to inform hosts about better routes (gateways).**
- Destination unreachable:
  - used to inform a host of the impossibility to deliver traffic to a specific destination

BETWEEN THE DIFFERENT MESSAGES THAT CAN BE SENT

OF DIFFERENT DEVICES

RECALL

ICMP Echo Request/Reply

Used by the ping program (return to Ping of Death)

PAG. 11

ATTACK THAT WORKS AT IP LEVEL => MAXIMUM ALLOWED DIMENSION IS 64Kb => PING COMMAND WITH THAT DIMENSION CAN BE SENT
⇓
BUT MACHINE WERE DESIGNED TO MANAGE 64b
⇓
CRASH!

```
# ping 192.168.1.1
PING 192.168.1.1 (192.168.1.1) from 192.168.1.100 : 64 bytes of data.
64 bytes from 192.168.1.1: icmp_seq=0 ttl=64 time=1.049 msec
64 bytes from 192.168.1.1: icmp_seq=1 ttl=64 time=660 usec
64 bytes from 192.168.1.1: icmp_seq=2 ttl=64 time=597 usec
64 bytes from 192.168.1.1: icmp_seq=3 ttl=64 time=548 usec
64 bytes from 192.168.1.1: icmp_seq=4 ttl=64 time=601 usec
64 bytes from 192.168.1.1: icmp_seq=5 ttl=64 time=592 usec
64 bytes from 192.168.1.1: icmp_seq=6 ttl=64 time=547 usec

--- 192.168.1.1 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/mdev = 0.547/0.656/1.049/0.165 ms
```

RECALL

ICMP Messages

- Address mask request/reply:
  - used by diskless systems to obtain the network mask at boot time.
- Timestamp request/reply:
  - used to synchronize clocks.
- Source quench:
  - used to inform about traffic overloads.
- Parameter problem:
  - used to inform about errors in the IP datagram fields.
- Echo request/reply:
  - used to test connectivity (ping).
- Time exceeded:
  - used to report expired datagrams (TTL = 0).
- **Redirect:**
  - **used to inform hosts about better routes (gateways).**
- Destination unreachable:
  - used to inform a host of the impossibility to deliver traffic to a specific destination

FOR SPECIFIC DESTINATIONS

IN FACT, INTERNET STRUCTURE IS NOT STATIC. IT IS ORGANIZED IN A DISTRIBUTED SYSTEM IN WHICH EACH SINGLE ROUTE IS DYNAMICALLY COMPILED ON THE BASE OF THE SYSTEM'S AVAILABILITY

RECALL

Route Change Requests

- Routers (not hosts) are
  - Responsible for keeping routing information up-to-date.
  - Assumed to discover best routes for every destination.
- Hosts begin with minimal routing information and learn new routes from routers.
- A host may boot up knowing the address of only one router – but that may not be the best route.

ICMP Redirect

Tells an host that a **better route** exists for a given destination, and gives the **gateway** for that route.

When a router detects a host using a non-optimal route it:

- Sends an ICMP Redirect message to the host and forwards the message.
- The host is expected to then update its routing table.

ICMP Redirect Attack (1/2)

The attacker can **forge** a spoofed ICMP redirect packet to re-route traffic on specific routes or to a specific host that may be not a router at all.

The attack can be used to:

- Hijack traffic (elect his/her computer as the gateway).
- Perform a denial-of-service attack.

Weak authentication:

- An ICMP message includes the IP header and a portion of the payload (usually the first 8 bytes) of the original IP datagram.

ICMP Redirect Attack (2/2)

The attacker needs to intercept a packet in the "original" connection in order to forge the reply (i.e., must be in the same network).

Creates a (half-duplex) MITM situation.

Handling of ICMP redirect is OS dependent:

- Windows 9x accepted them adding a temporary host entry in routing tables.
- Linux: default off, configured by value in **`/proc/sys/net/ipv4/<int>/accept_redirects`**

Route Mangling

If the attacker can announce routes to a router, s(he) can play a number of magical tricks

- IGRP, RIP, OSPF: no/weak authentication
- EIGRP, BGP: authentication available but seldom used (see next slide).

http://www.blackhat.com/presentations/bh-europe-03/bh-europe-03-dugan.pdf

http://www.renesys.com/wp-content/uploads/2013/05/blackhat-09.pdf

BGP Hijacks in Late 2013

http://www.renesys.com/2013/11/mitm-internet-hijacking/

Conclusions

Certain DoS attacks exploit memory errors in the network stack implementations.

DoS is generally always feasible, given enough resources (i.e., the attacker can just rent a botnet for a few hours).

Network attacks can happen at different layers.

Attacks are made possible essentially by the lack of (strong) authentication in the protocols.

10. Secure Network Architectures

Computer Security Courses @ POLIMI

Firewalls

Firewall: network access control system that verifies all the packets flowing through it.

Its main **functions** are usually:

- IP packet filtering → "WHICH PACKETS CAN PASS? WHICH CAN'T?"
- Network address translation (NAT)

Must be the single enforcement point between a screened network and outside networks.

↓ CHECKPOINT FOR CHECKING THE TRAFFIC BETWEEN THE 2 POINTS (INTERNET) → USELESS IF THE CHECKPOINT CAN BE BYPASSED

A Side Note on Grammar

Bear in mind:

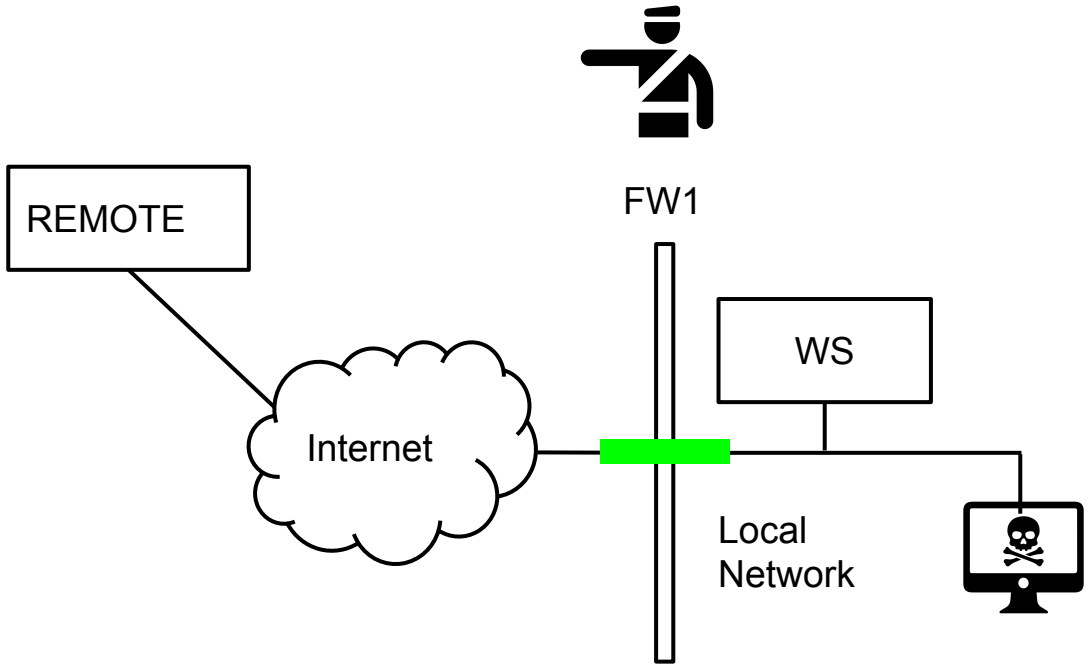
- "firewall" - correct
- “fire wall” - incorrect
- “un firewall” (IT) - correct
- “una firewall” (IT) - incorrect

A firewall (construction) is a *wall* designed to partition a building and stop fire spreading,

“Wall of fire” is a 4th level spell.

Firewalls are not Omnipotent

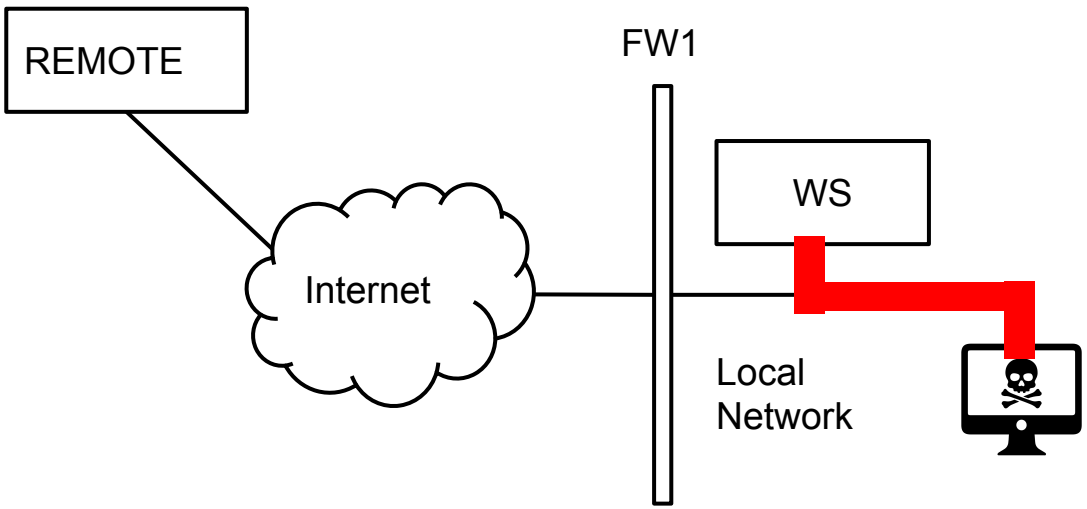
A firewall checks **all** the traffic flowing through it, and **only** the traffic flowing through it.



Firewalls are not Omnipotent

A firewall checks **all** the traffic flowing through it, and **only** the traffic flowing through it.

Powerless against **insider attacks** (unless the network is partitioned somehow).



Firewalls are not Omnipotent

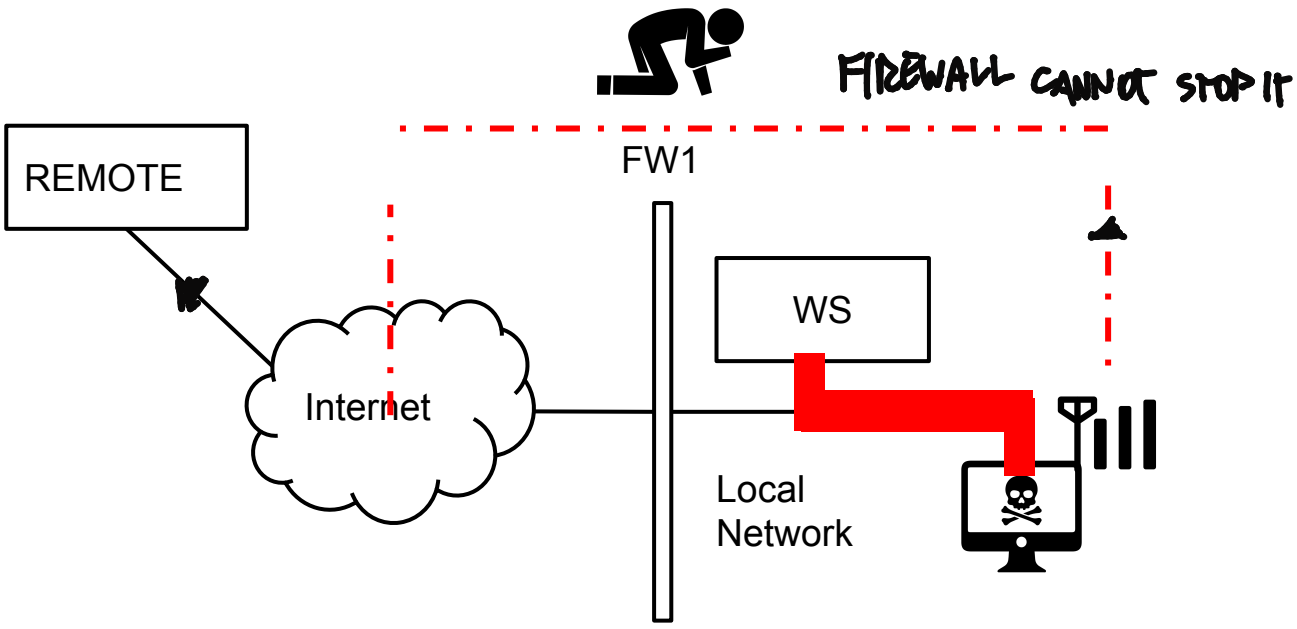
A firewall checks **all** the traffic flowing through it, and **only** the traffic flowing through it.

Powerless against **insider attacks** (unless the network is partitioned somehow).

In general, **powerless** against unchecked paths

- E.g. a modem or 5G connection of machine connected also to a LAN

Insider Attacks/Unchecked Paths



Firewalls are Computers

The firewall itself is a computer: it could have vulnerabilities and be violated.

Most of the times

- It's a single purpose machine, an embedded appliance with just a firmware.
- Usually offers few or no services

Smaller attack surface

Security Policies = Firewall Rules

A firewall is a stupid “bouncer at the door”

- Just applies rules
- Bad rules = no protection

Firewall **rules** are essentially the implementation of **higher-level security policies**.

- E.g. “I want no clients to be able to download email from external email servers!”

Policy must be built on a **default deny base**:

- I.e., "deny all, except ..."

FOLLOW THE SAME RULES FOR FILTERING—VALIDATING INPUTS

Firewall Taxonomy

We divide them depending on their **packet inspection capability** (from low to high layers).

WHICH KIND OF INFORMATION THEY CAN ANALYZE FROM A NETWORKING PERSPECTIVE

Network layer firewalls NETWORK AND TRANSPORT LAYER

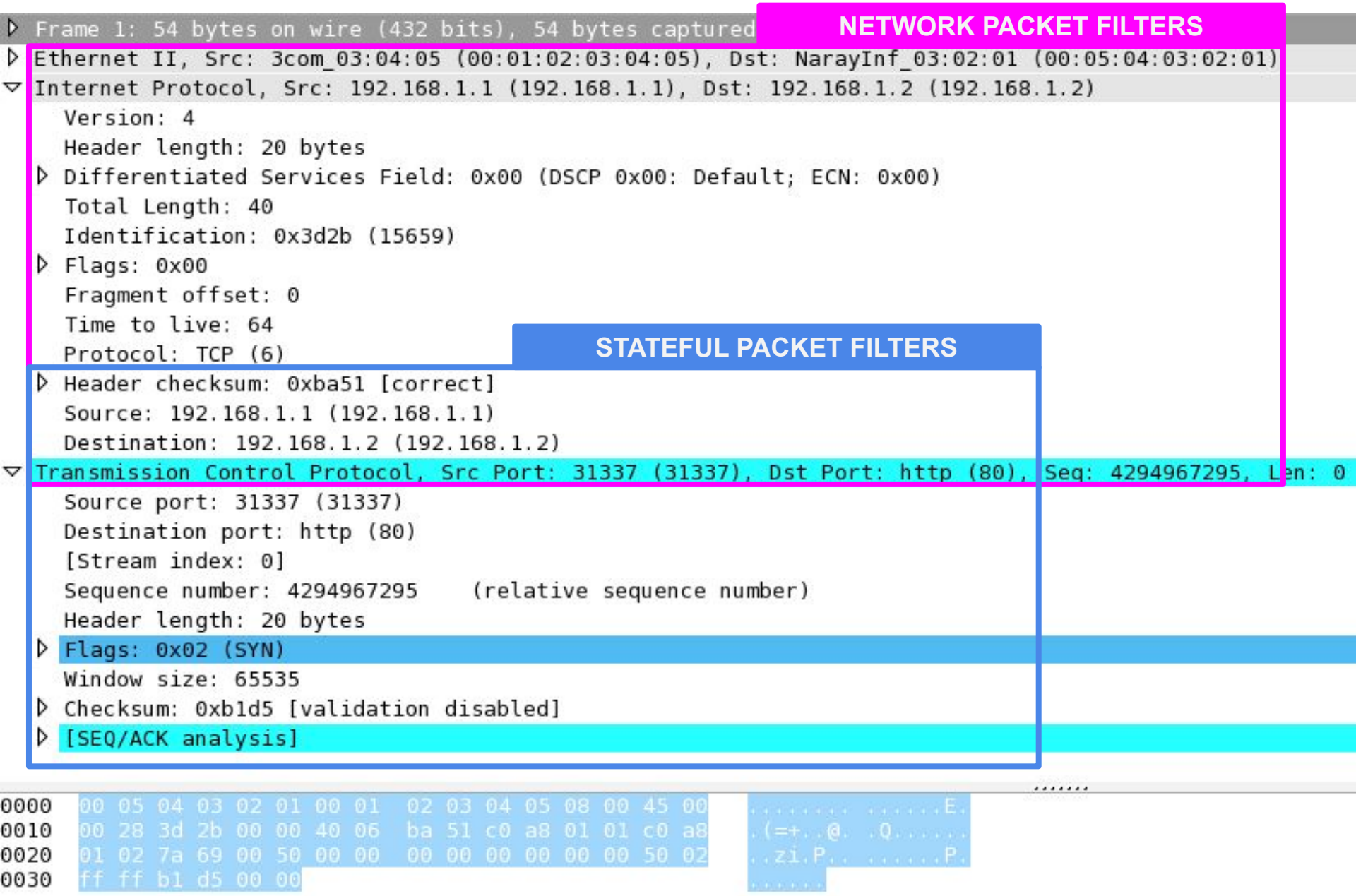
- Packet Filters firewall (network layer)
- Stateful Packet Filters (SPF) firewall (network and transport layer)

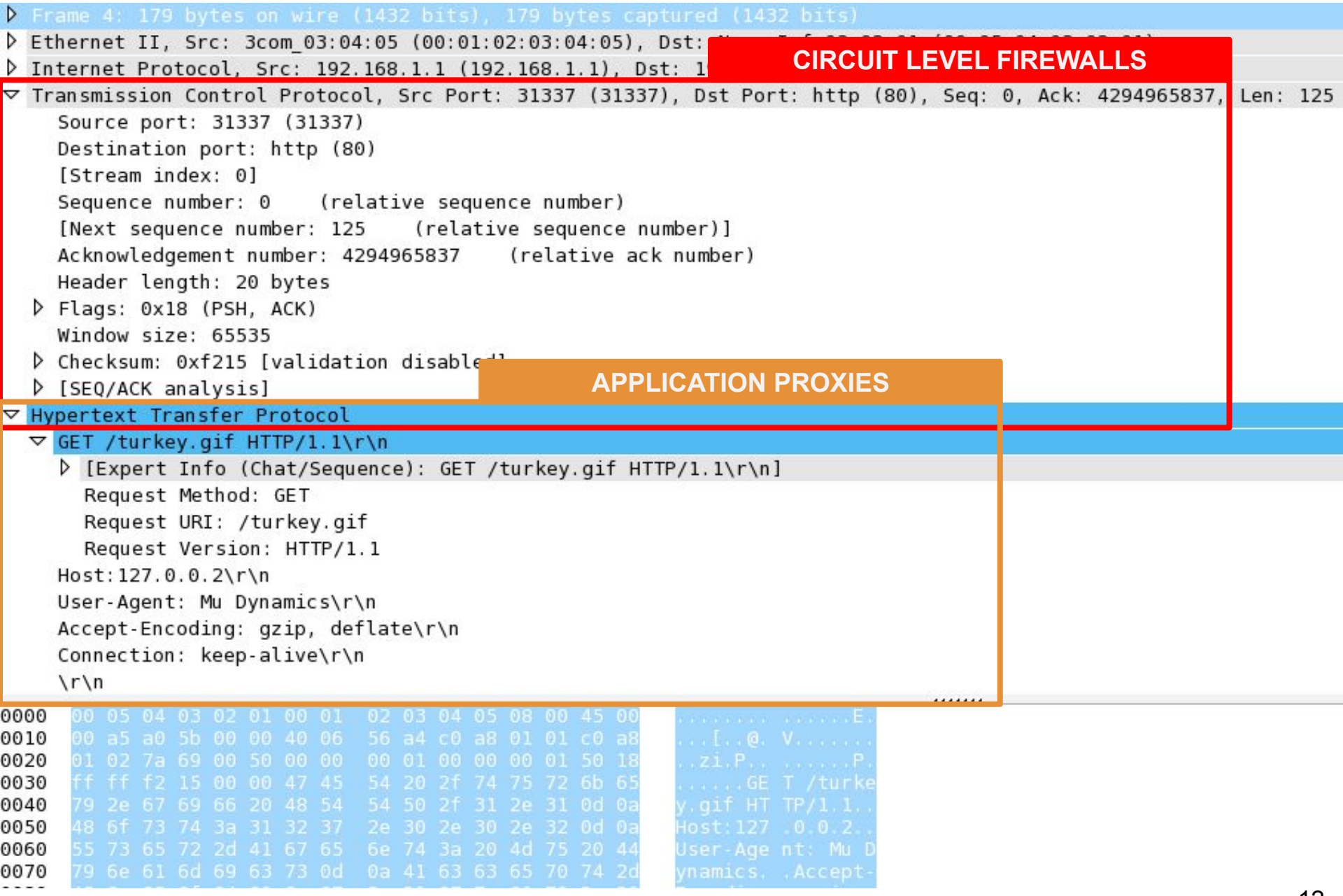
→ MODERN FIREWALLS

Application layer firewalls TRANSPORT AND APPLICATION LAYER

→ THEY DO NOT EXIST ANYMORE

- Circuit level firewalls (transport-application)
- Application proxies (application)


NETWORK PACKET FILTERS
STATEFUL PACKET FILTERS
Frame 1: 54 bytes on wire (432 bits), 54 bytes captured
Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst: NarayInf_03:02:01 (00:05:04:03:02:01)
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 192.168.1.2 (192.168.1.2)
Version: 4
Header length: 20 bytes
Differentiated Services Field: 0x00 (DSCP 0x00: Default; ECN: 0x00)
Total Length: 40
Identification: 0x3d2b (15659)
Flags: 0x00
Fragment offset: 0
Time to live: 64
Protocol: TCP (6)
Header checksum: 0xba51 [correct]
Source: 192.168.1.1 (192.168.1.1)
Destination: 192.168.1.2 (192.168.1.2)
Transmission Control Protocol, Src Port: 31337 (31337), Dst Port: http (80), Seq: 4294967295, Len: 0
Source port: 31337 (31337)
Destination port: http (80)
[Stream index: 0]
Sequence number: 4294967295 (relative sequence number)
Header length: 20 bytes
Flags: 0x02 (SYN)
Window size: 65535
Checksum: 0xb1d5 [validation disabled]
[SEQ/ACK analysis]
0000 00 05 04 03 02 01 00 01 02 03 04 05 08 00 45 00
0010 00 28 3d 2b 00 00 40 06 ba 51 c0 a8 01 01 c0 a8
0020 01 02 7a 69 00 50 00 00 00 00 00 00 00 00 50 02
0030 ff ff b1 d5 00 00


Frame 4: 179 bytes on wire (1432 bits), 179 bytes captured (1432 bits)
Ethernet II, Src: 3com_03:04:05 (00:01:02:03:04:05), Dst:
Internet Protocol, Src: 192.168.1.1 (192.168.1.1), Dst: 1
CIRCUIT LEVEL FIREWALLS
Transmission Control Protocol, Src Port: 31337 (31337), Dst Port: http (80), Seq: 0, Ack: 4294965837, Len: 125
Source port: 31337 (31337)
Destination port: http (80)
[Stream index: 0]
Sequence number: 0 (relative sequence number)
[Next sequence number: 125 (relative sequence number)]
Acknowledgement number: 4294965837 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 65535
Checksum: 0xf215 [validation disable
[SEQ/ACK analysis]
APPLICATION PROXIES
Hypertext Transfer Protocol
GET /turkey.gif HTTP/1.1\r\n
[Expert Info (Chat/Sequence): GET /turkey.gif HTTP/1.1\r\n]
Request Method: GET
Request URI: /turkey.gif
Request Version: HTTP/1.1
Host:127.0.0.2\r\n
User-Agent: Mu Dynamics\r\n
Accept-Encoding: gzip, deflate\r\n
Connection: keep-alive\r\n
\r\n
0000 00 05 04 03 02 01 00 01 02 03 04 05 08 00 45 00 ....E.
0010 00 a5 a0 5b 00 00 40 06 56 a4 c0 a8 01 01 c0 a8 ...[..@. V.....
0020 01 02 7a 69 00 50 00 00 00 01 00 00 00 01 50 18 ..zi.P.. .....P.
0030 ff ff f2 15 00 00 47 45 54 20 2f 74 75 72 6b 65 ..... GE T /turke
0040 79 2e 67 69 66 20 48 54 54 50 2f 31 2e 31 0d 0a y.gif HT TP/1.1..
0050 48 6f 73 74 3a 31 32 37 2e 30 2e 30 2e 32 0d 0a Host:127 .0.0.2..
0060 55 73 65 72 2d 41 67 65 6e 74 3a 20 4d 75 20 44 User-Age nt: Mu D
0070 79 6e 61 6d 69 63 73 0d 0a 41 63 63 65 70 74 2d ynamics. .Accept-

Packet Filters

EACH PACKET IS ANALYZED INDEPENDENTLY FROM THE OTHERS

Packet by packet processing.

Decodes the IP (and part of the TCP) header:

- SRC and DST IP
- SRC and DST port
- Protocol type
- IP options

Stateless

- Cannot track TCP connections.
- No (full) payload inspection.

} => UNABLE TO DETECT SNIFFING/SPOOFING ...

Mostly found on routers ("ACLs" on Cisco)

Expressiveness

Predicate only on what we can decode. E.g.:

1ST THING TO DO WHEN CONFIGURING A FIREWALL IS BLOCK ANYTHING

- Block any incoming packet ("default deny")
  - **`iptables -P INPUT DROP`** `# P = policy`
- Allow incoming packet if going to 10.0.0.1
  - **`iptables -A INPUT -d 10.0.0.1 -j ALLOW`**
- Block anything out except SMTP (port 25)
  - **`iptables -P OUTPUT DROP`**
  - **`iptables -A OUTPUT --dport 25 -j ALLOW`**

What happens with packets with spoofed IP?

Rule ~> Reaction

Regardless of the specific syntax, every network packet filter allows to express the following concept:

- if (packet matches certain condition)
  - do this, e.g.:
    - block
    - allow
    - log
    - (other actions, e.g., rate limit)

Stateful (or Dynamic) Packet Filters

Include network packet filters, plus:

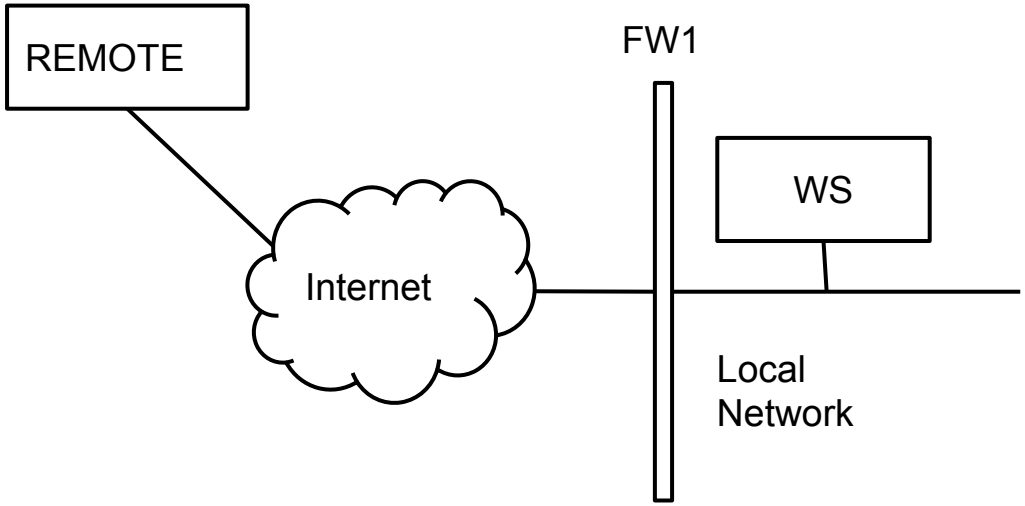
- keep track of the **TCP state machine**
  - after SYN, SYN-ACK must follow
- we can **track connections** without adding a response rule.
- Make deny rule safer

IF SOMETHING STRANGE IS NOTICED, BLOCK THE CONNECTION

↳SIMPLER WAY TO WRITE RULES ON THE FIREWALL (EX: NO NEED TO ADD RESPONSE RULES)

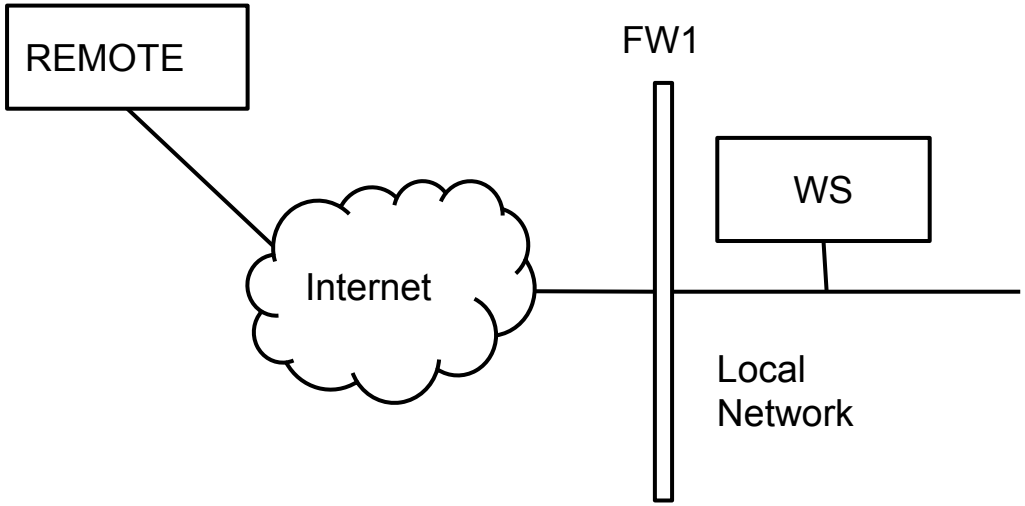
Example

Stateless Packet Filter



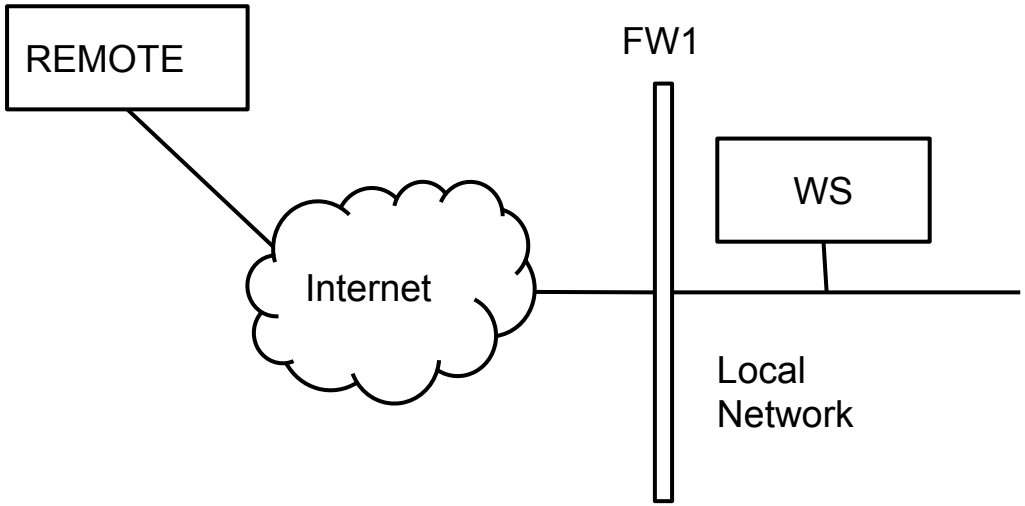
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	**10.0.0.1 (example)**	**ANY**	**zone 1 -> zone 2**	**192.168.0.2 (example)**	**443**	**ALLOW/ DENY**	**(example: the X server in zone 1 cannot contact the Y server)**

Stateless Packet Filter



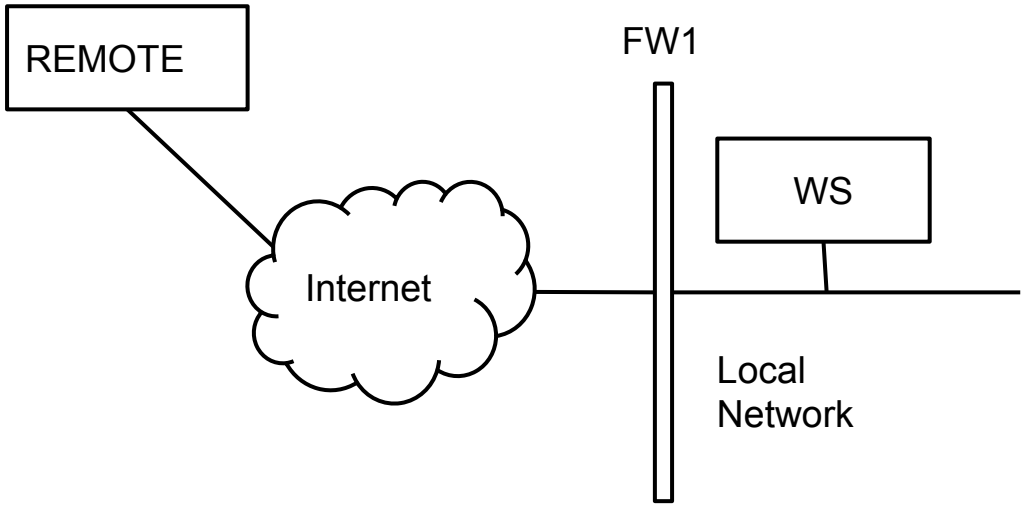
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	**10.0.0.1 (example)**	**ANY**	**zone 1 -> zone 2**	**192.168.0.2 (example)**	**443**	**ALLOW/ DENY**	**(example: the X server in zone 1 cannot contact the Y server)**
FW1	***ANY***	***ANY***	***WWW -> Local***	***ANY***	***ANY***	***DENY***	***Default deny on all firewalls***

Stateless Packet Filter



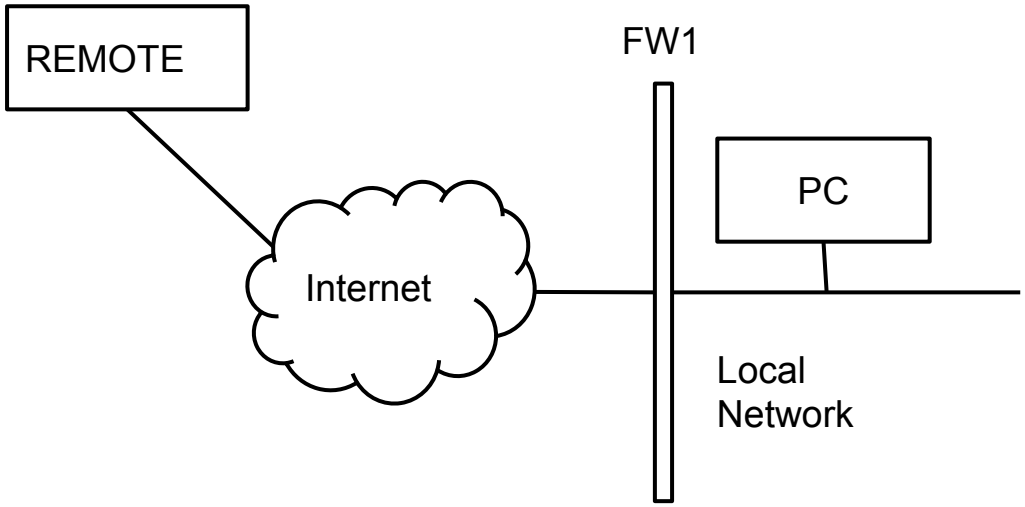
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	ALLOW/ DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*

Stateless Packet Filter



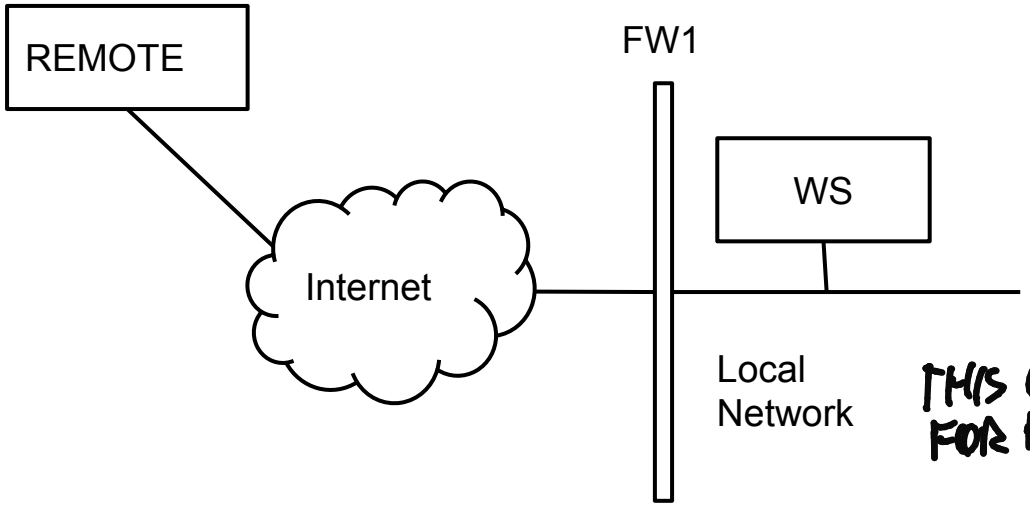
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	ALLOW/ DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*WWW -> Local*	*WS_IP*	*80*	*ALLOW*	*Allow incoming connection to the WS on PORT 80*

Stateless Packet Filter



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	ALLOW/ DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*PC_IP*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*ALLOW*	*Allow access to internet*
FW1	*ANY*	*ANY*	*WWW->Local*	*PC_IP*	*ANY*	*ALLOW*	*Allow access to internet*

STATEFUL Packet Filter

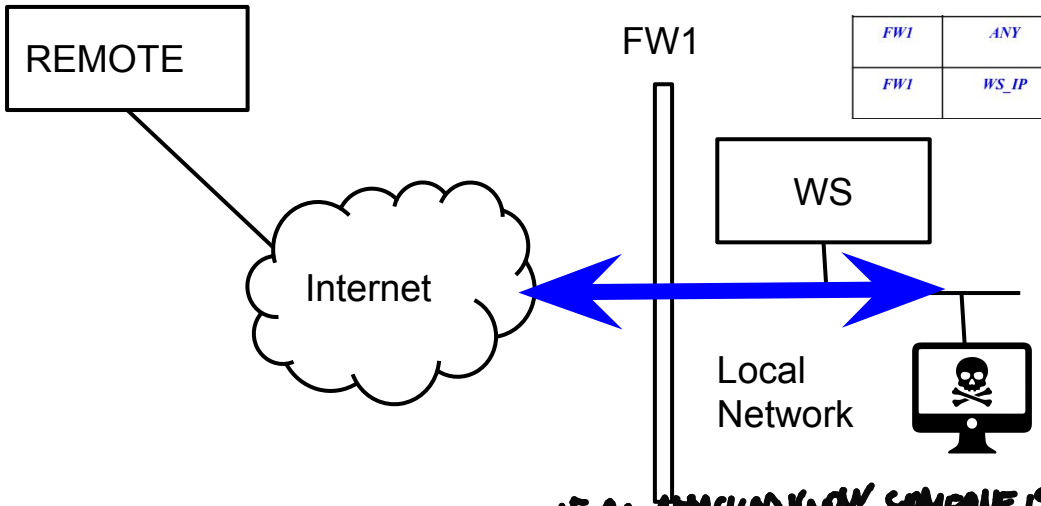


THIS CONFIGURATION ALLOWS ONLY FOR REQUESTS FROM THE EXTERNAL

Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	ALLOW/ DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ANY*	*ANY*	*WWW -> Local*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*Local -> WWW*	*ANY*	*ANY*	*DENY*	*Default deny on all firewalls*
FW1	*ANY*	*ANY*	*WWW -> Local*	*WS_IP*	*80*	*ALLOW*	*Allow incoming connection to the WS on PORT 80*
~~*FW1*~~	~~*WS_IP*~~	~~*80*~~	~~*Local -> WWW*~~	~~*ANY*~~	~~*ANY*~~	~~*ALLOW*~~	~~*Allow outgoing connection from the WS*~~

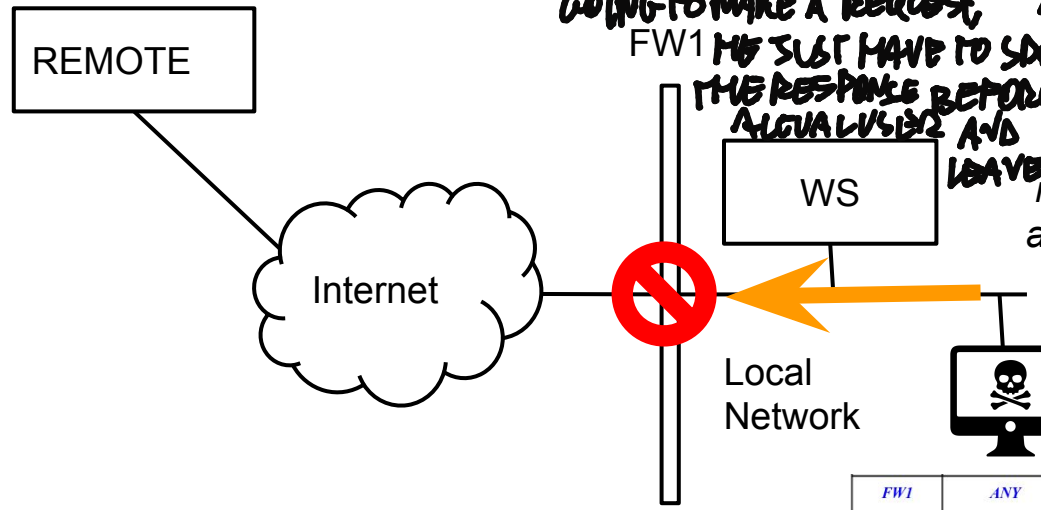
NO NEED TO ADD THE RESPONSE RULE

STATEFUL vs STATELESS Packet Filter



FW1	ANY	ANY	WWW -> Local	WS_IP	80	ALLOW	Allow incoming connection to the WS on PORT 80
FW1	WS_IP	80	Local -> WWW	ANY	ANY	ALLOW	Allow outgoing connection from the WS

Stateless



IF AN ATTACKER KNOW SOMEONE IS GOING TO MAKE A REQUEST, HE JUST HAVE TO SPOOF THE RESPONSE BEFORE THE LEGAL USER AND LEAVE

Only responses are allowed

RESPONSE ARE ALLOWED IF PREVIOUS REQUESTS HAVE BEEN PERFORMED

BUT

AVOID ATTACKER THAT SPOOF IP ADDRESSES FROM LEGITIMATE USERS, TO GO OUT WITHOUT HAVING A REQUEST BEFORE

Stateful

FW1	ANY	ANY	WWW -> Local	WS_IP	80	ALLOW	Allow incoming connection to the WS on PORT 80
~~FW1~~	~~WS_IP~~	~~80~~	~~Local -> WWW~~	~~ANY~~	~~ANY~~	~~ALLOW~~	~~Allow outgoing connection from the WS~~

Deny Rule Safer

Stateful (or Dynamic) Packet Filters

Include network packet filters, plus:

- keep track of the **TCP state machine**
  - after SYN, SYN-ACK must follow
- we can **track connections** without adding a response rule.
- Make deny rule safer

CONS: STATELESS PACKET FILTER → CHECK ONLY NUMBER OF PACKETS/SECOND TO MAKE STATEFUL ALSO HAS TO STORE CONNECTIONS INFORMATION IN MEMORY

Performance bounded on a **per-connection basis**, not on a **per-packet basis**.

- The **number of simultaneous connections** are just as important as packets per second.

Stateful (or Dynamic) Packet Filters

Include network packet filters, plus:

- keep track of the **TCP state machine**
  - after SYN, SYN-ACK must follow
- we can **track connections** without adding a response rule.
- Make deny rule safer

CONS:

Performance bounded on a **per-connection basis**, not on a **per-packet basis**.

- The **number of simultaneous connections** are just as important as packets per second.

Better Expressiveness and Pluses

Tracking (Logging and accounting on) connections.

Deeper content inspection:

WE KNOW THE RELATION BETWEEN THE VARIOUS PACKETS

- Reconstruct application-layer protocols
- Application-layer filtering, e.g. ActiveX objects in HTTP responses disallowed.

Network Address Translation (**NAT**) offered as embedded feature.

Packet defragmenting and reassembly (helps avoiding pathological fragmented packets, e.g., teardrop).

Session Handling

A **session** is an atomic, transport-layer exchange of application data between 2 hosts.

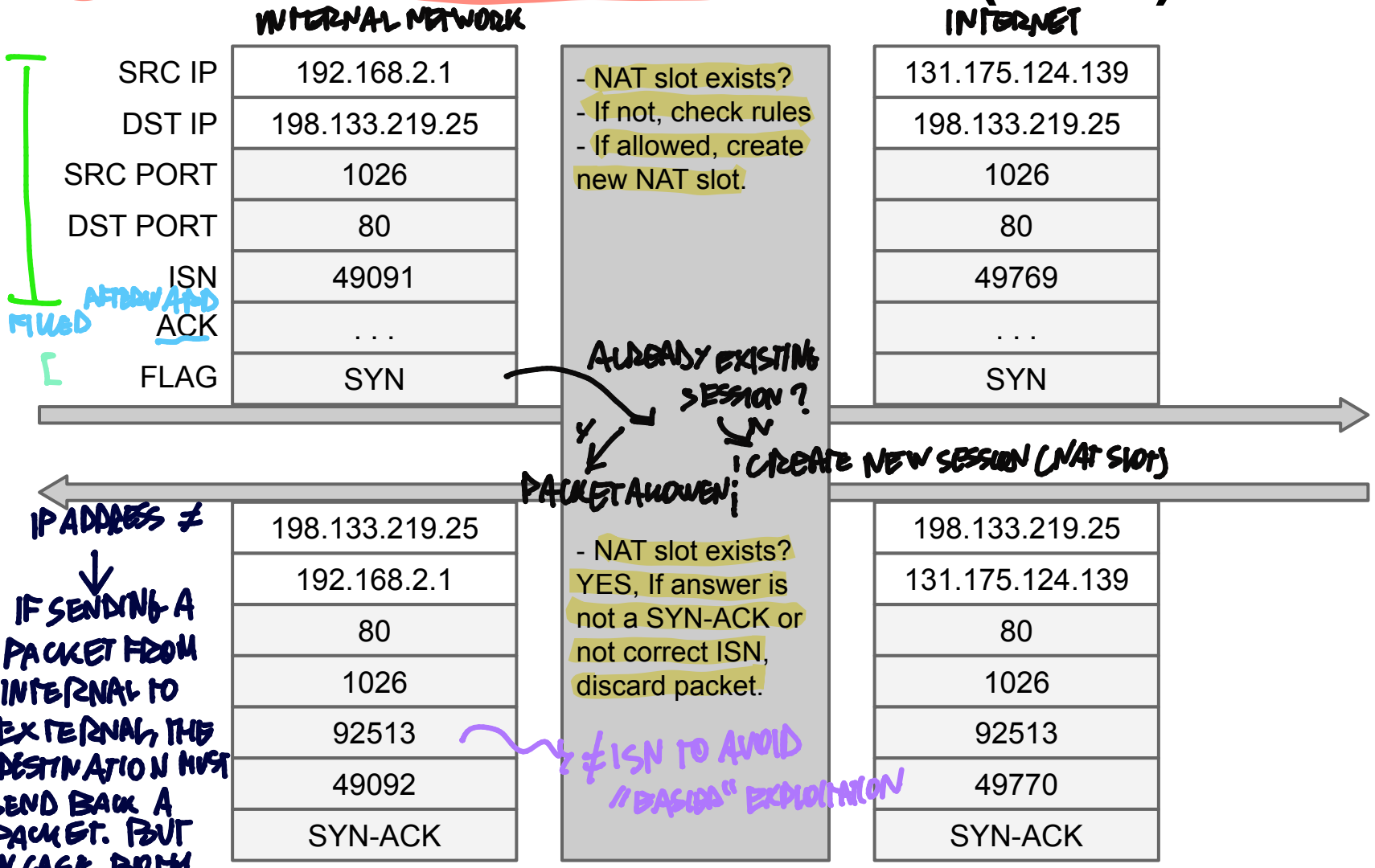
Main transport protocols:

- TCP (Transmission Control Protocol)
  - session =~ TCP connection
- UDP (User Datagram Protocol)
  - session = this concept does not exist

→ HANDLE SESSIONS IF AIMING TO PERFORM NAT

Session-handling is fundamental for NAT.

NAT Session Initialization (TCP)



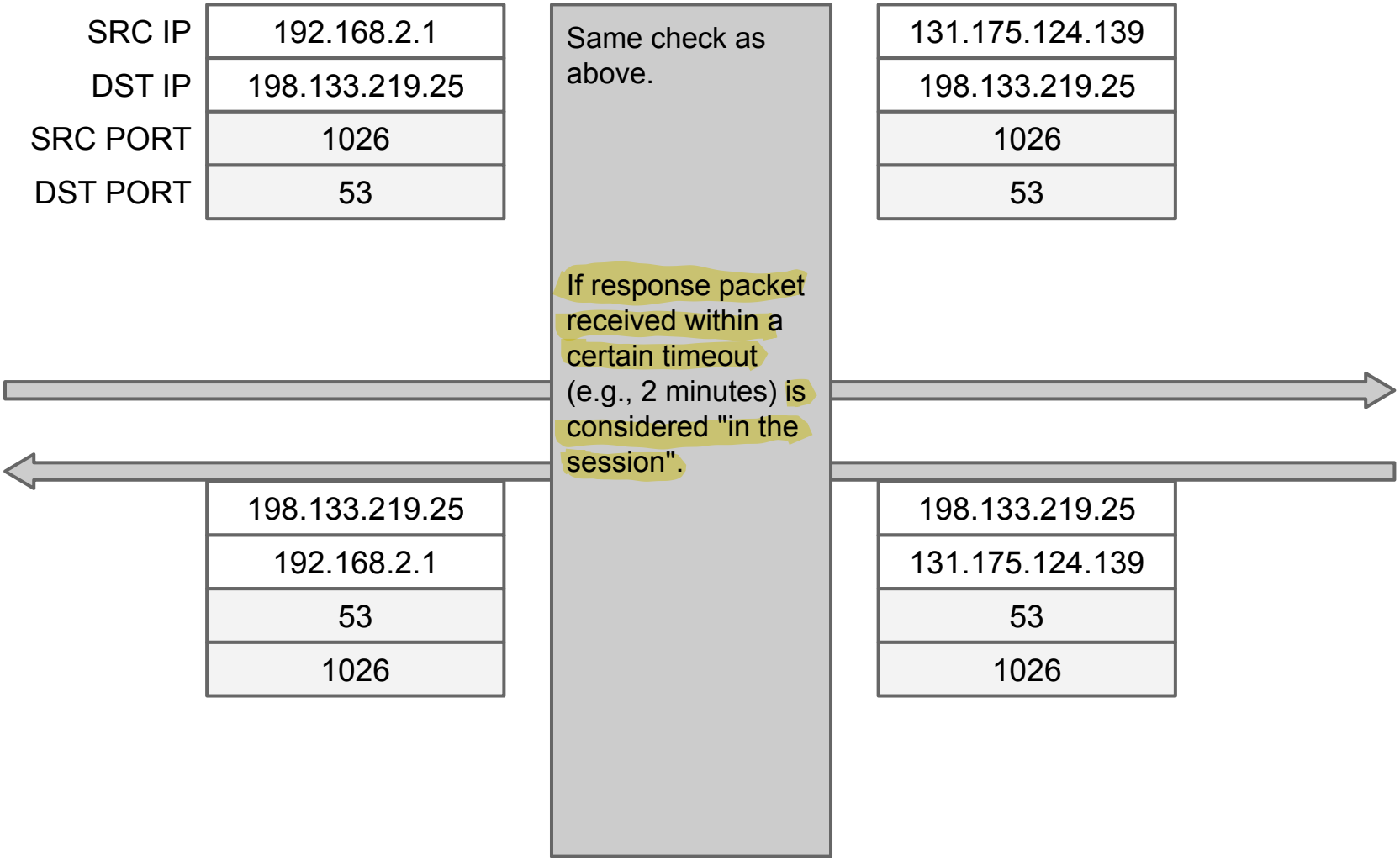
What about UDP

Widely used, so we can't dismiss it (e.g. DNS, VoIP H.323, video streaming).

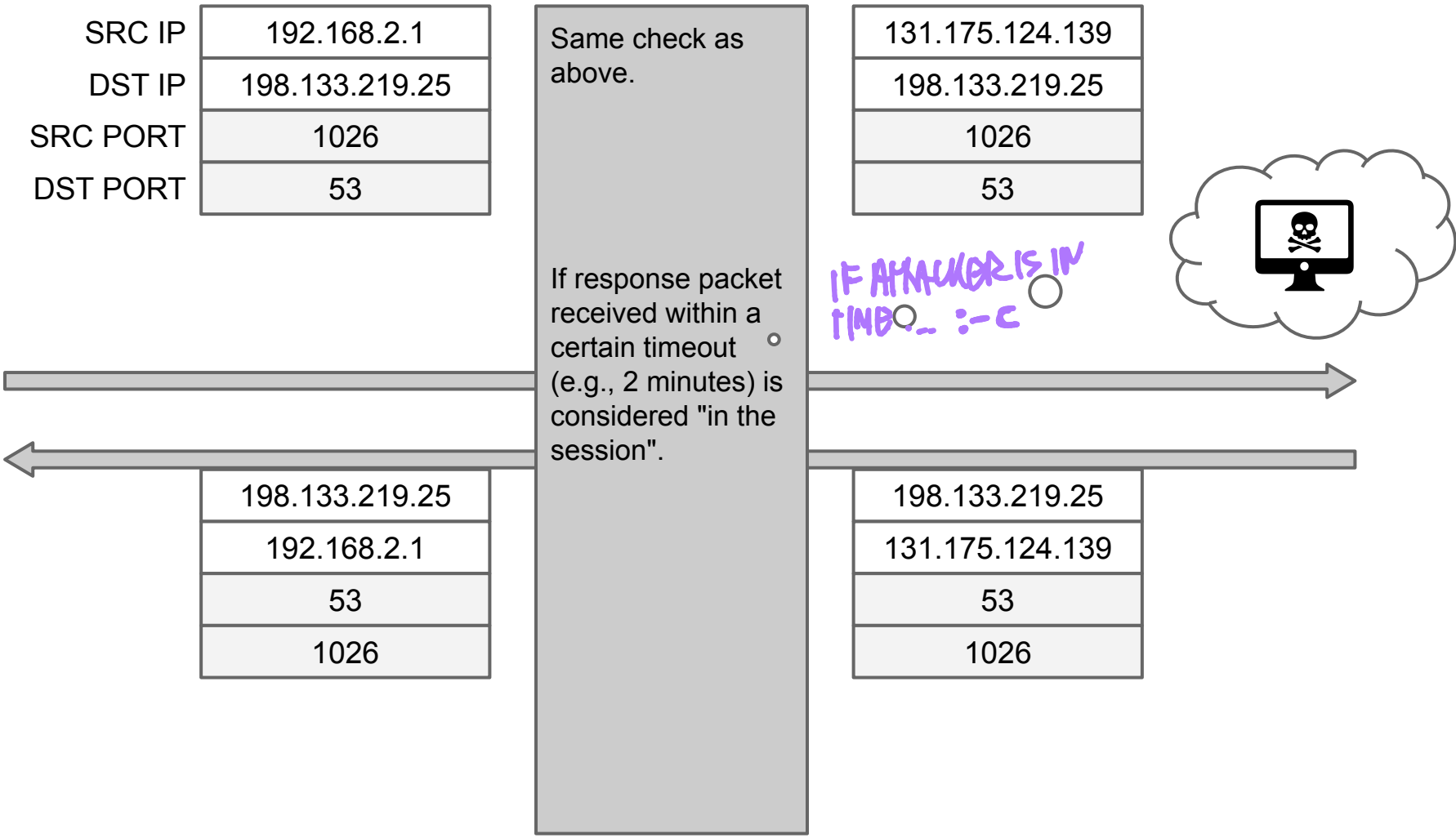
Difficult to secure and handle, because there are no connections.

Connectionless, but a “session” can be inferred for NAT and controls.

UDP and NAT


SRC IP
DST IP
SRC PORT
DST PORT
192.168.2.1
198.133.219.25
1026
53
Same check as above.
131.175.124.139
198.133.219.25
1026
53
If response packet received within a certain timeout (e.g., 2 minutes) is considered "in the session".
198.133.219.25
192.168.2.1
53
1026
198.133.219.25
131.175.124.139
53
1026
Stateful Firewall

UDP and NAT


SRC IP
DST IP
SRC PORT
DST PORT
192.168.2.1
198.133.219.25
1026
53
Same check as above.
If response packet received within a certain timeout (e.g., 2 minutes) is considered "in the session".
131.175.124.139
198.133.219.25
1026
53
IF ATTACKER IS IN TIME :-C
198.133.219.25
192.168.2.1
53
1026
198.133.219.25
131.175.124.139
53
1026
Stateful Firewall

Application-layer Inspection for NAT

DYNAMIC PROTOCOLS → INSTANTIATE NEW CONNECTION BASED ON INFORMATION SENT PREVIOUSLY BETWEEN CLIENT AND SERVER

Some protocols (e.g., DCC, RDT, instant messengers, file transfer) transmit network information data (e.g., port) at application layer.

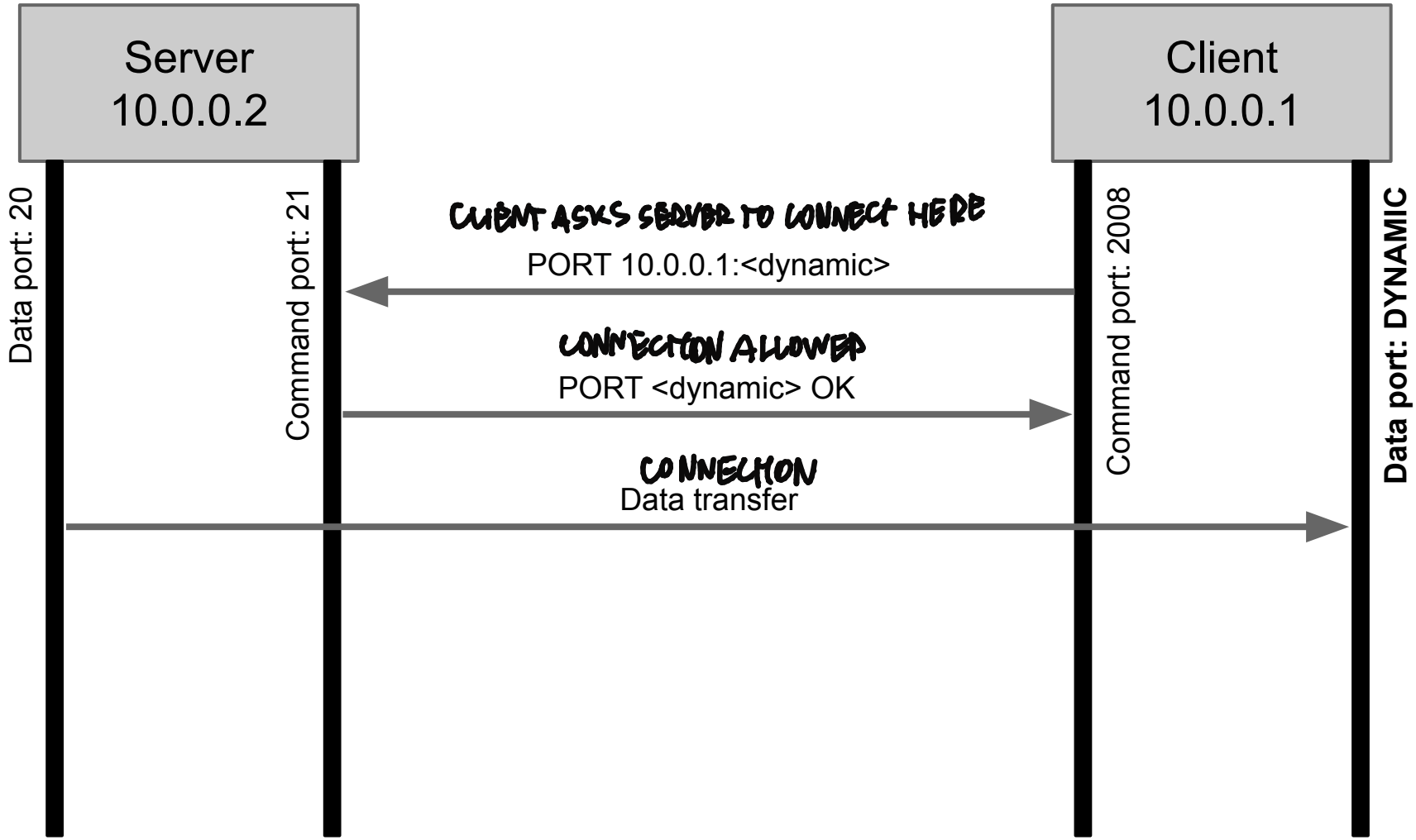
For instance, FTP uses dynamic connections:

- Allocated for file uploads, downloads, output of commands
- "PORT" application command

Stateful firewalls must take care of these.

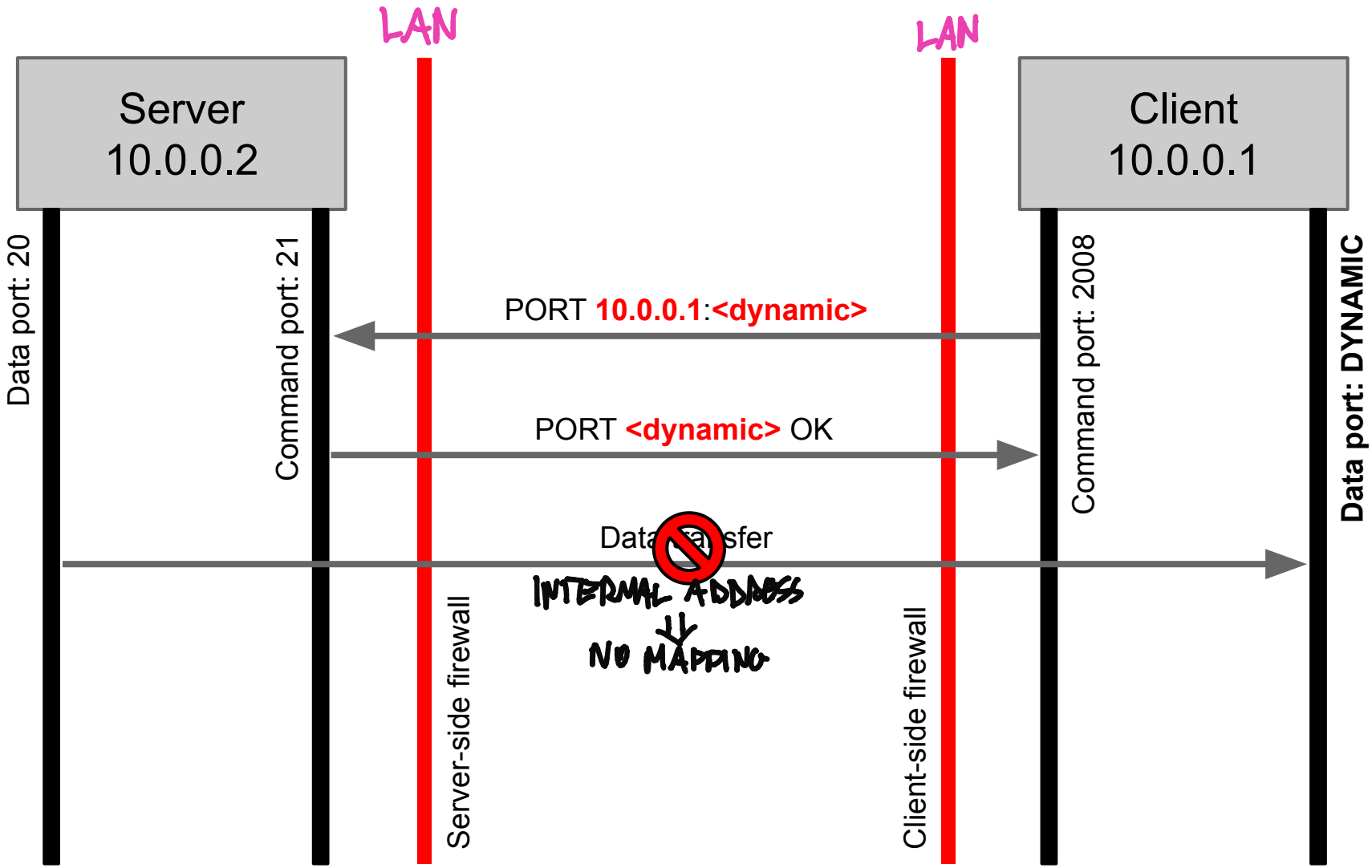
Example: FTP Standard Mode (NO FW)

! CLIENT AND SERVER AT THE SAME NETWORK

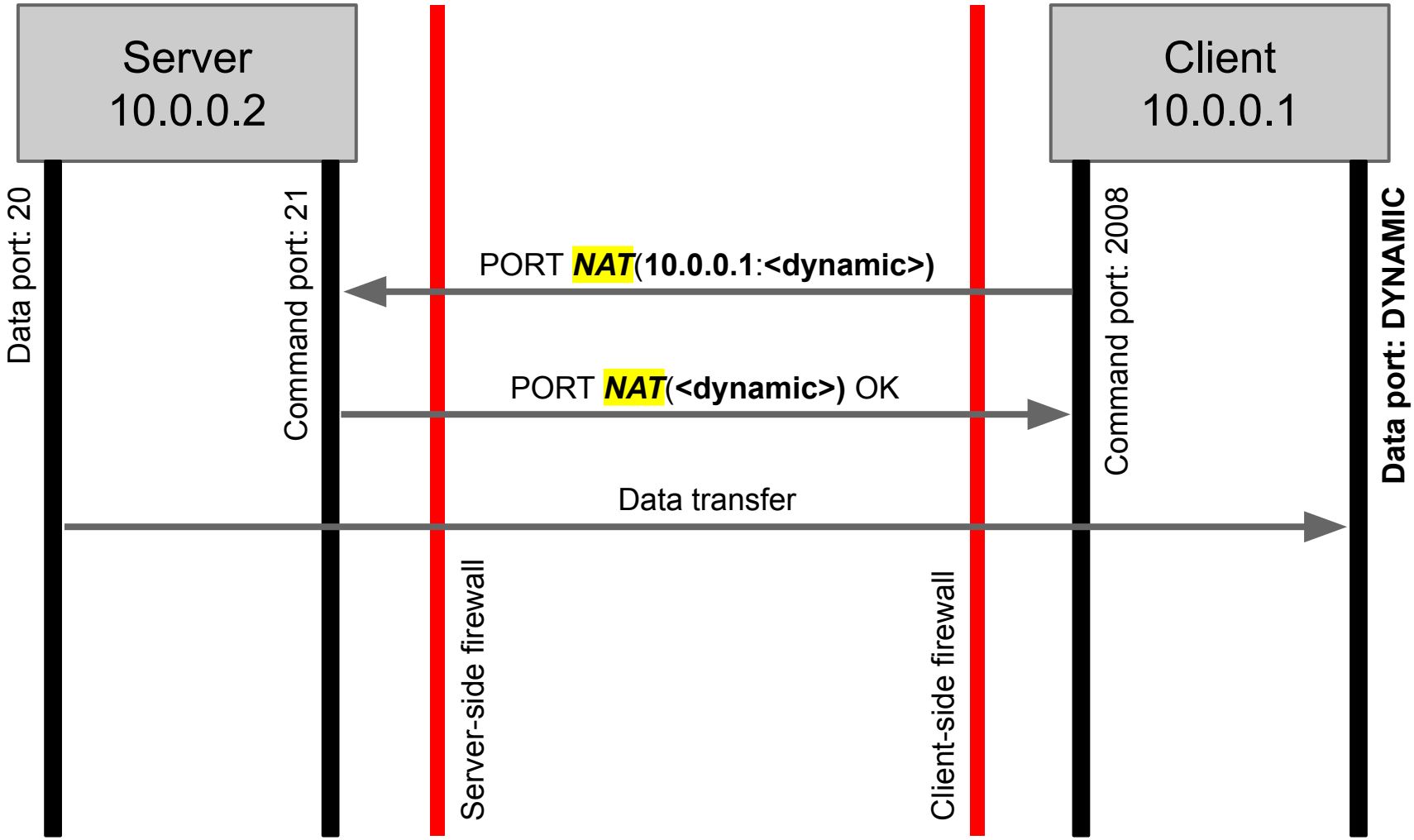


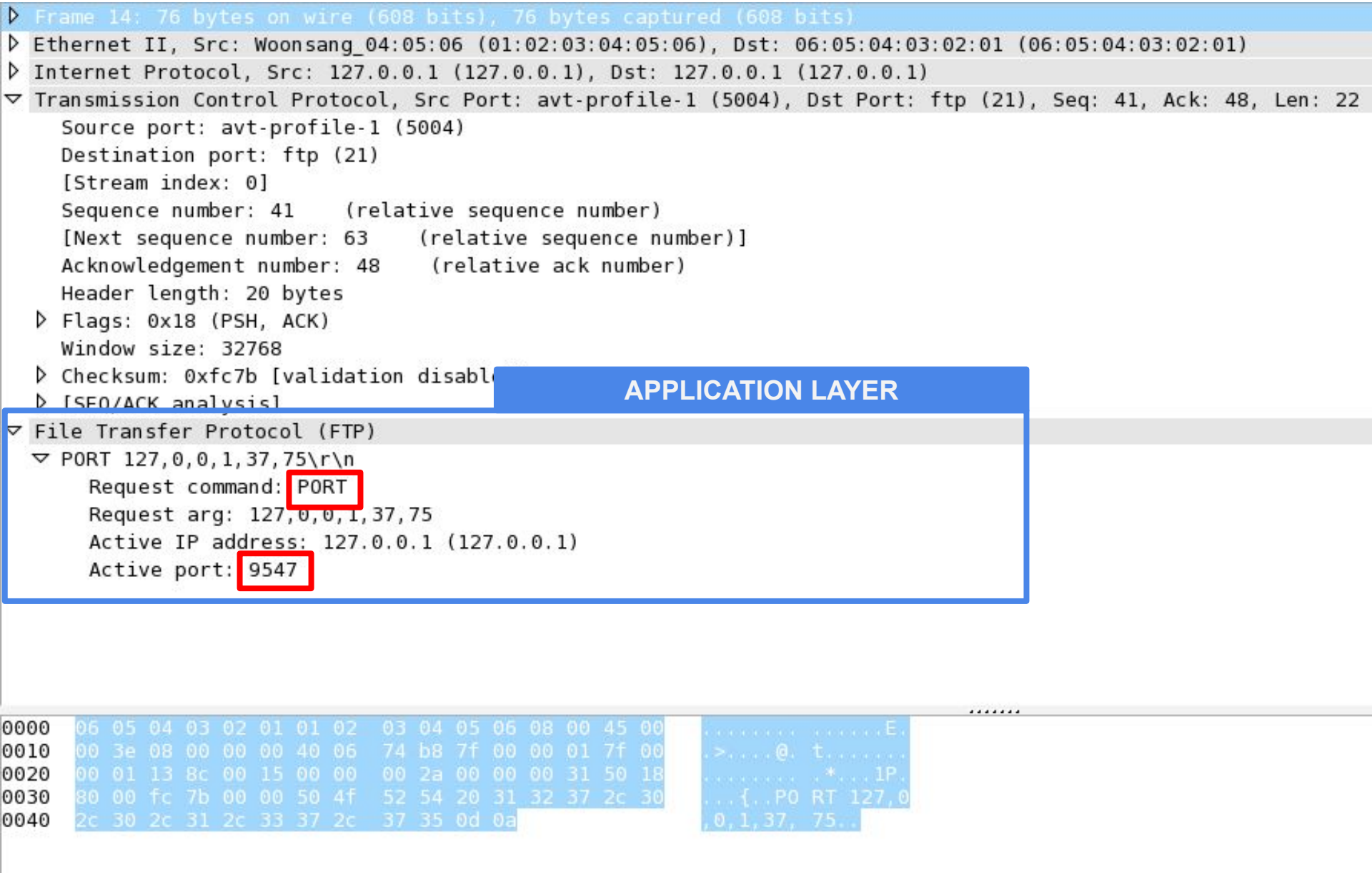
ADD 2 FIREWALLS

Example: FTP Standard Mode (1)



Example: FTP Standard Mode (1)




Frame 14: 76 bytes on wire (608 bits), 76 bytes captured (608 bits)
Ethernet II, Src: Woonsang_04:05:06 (01:02:03:04:05:06), Dst: 06:05:04:03:02:01 (06:05:04:03:02:01)
Internet Protocol, Src: 127.0.0.1 (127.0.0.1), Dst: 127.0.0.1 (127.0.0.1)
Transmission Control Protocol, Src Port: avt-profile-1 (5004), Dst Port: ftp (21), Seq: 41, Ack: 48, Len: 22
Source port: avt-profile-1 (5004)
Destination port: ftp (21)
[Stream index: 0]
Sequence number: 41 (relative sequence number)
[Next sequence number: 63 (relative sequence number)]
Acknowledgement number: 48 (relative ack number)
Header length: 20 bytes
Flags: 0x18 (PSH, ACK)
Window size: 32768
Checksum: 0xfc7b [validation disabl
[SEQ/ACK analysis]
APPLICATION LAYER
File Transfer Protocol (FTP)
PORT 127,0,0,1,37,75\r\n
Request command: PORT
Request arg: 127,0,0,1,37,75
Active IP address: 127.0.0.1 (127.0.0.1)
Active port: 9547
0000 06 05 04 03 02 01 01 02 03 04 05 06 08 00 45 00 ....E.
0010 00 3e 08 00 00 00 40 06 74 b8 7f 00 00 01 7f 00 .>....@. t.....
0020 00 01 13 8c 00 15 00 00 00 2a 00 00 00 31 50 18 ..... .*...1P.
0030 80 00 fc 7b 00 00 50 4f 52 54 20 31 32 37 2c 30 ...{..PO RT 127,0
0040 2c 30 2c 31 2c 33 37 2c 37 35 0d 0a ,0,1,37, 75..

Example: FTP Standard Mode (2)

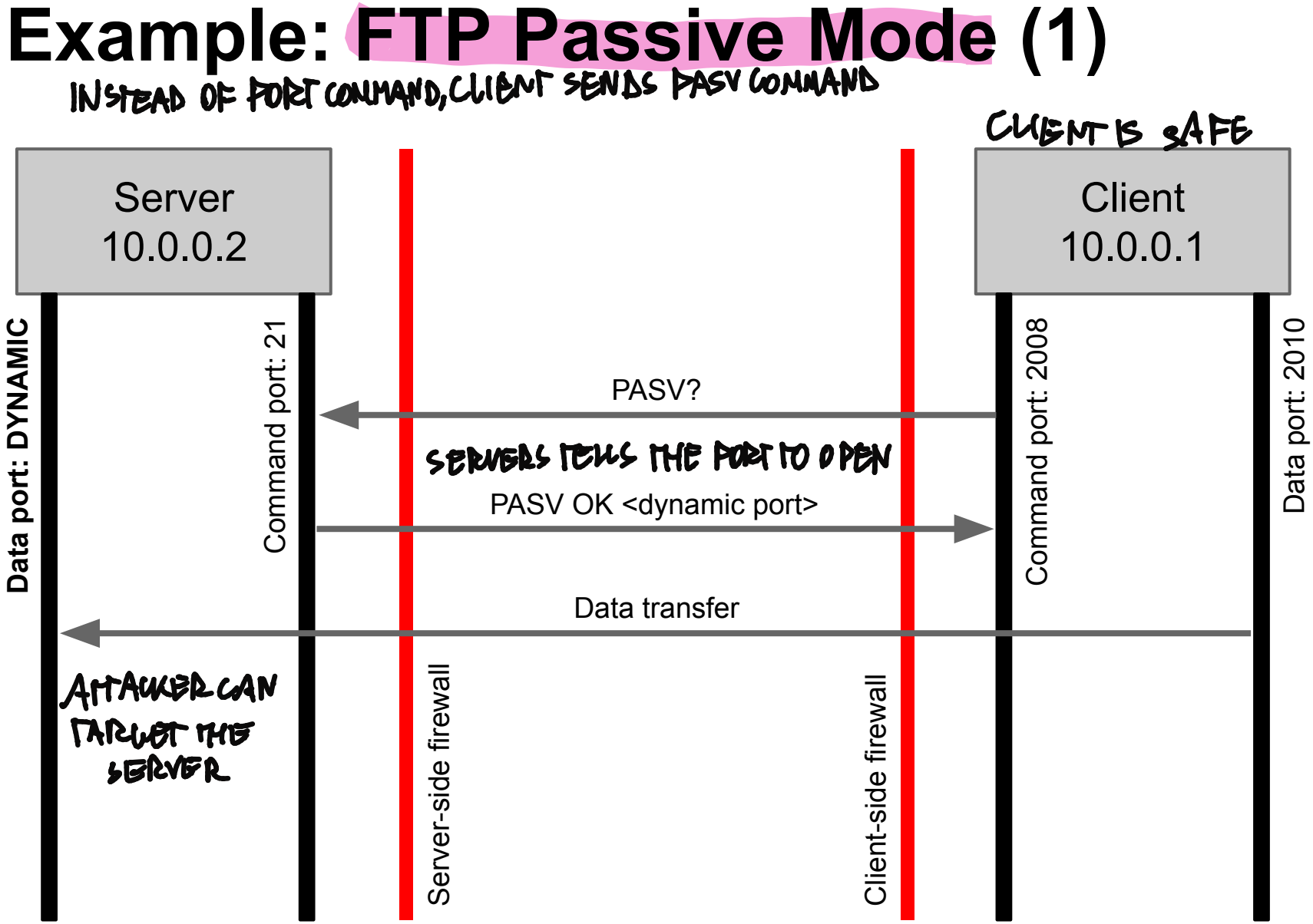
Client-side firewall:

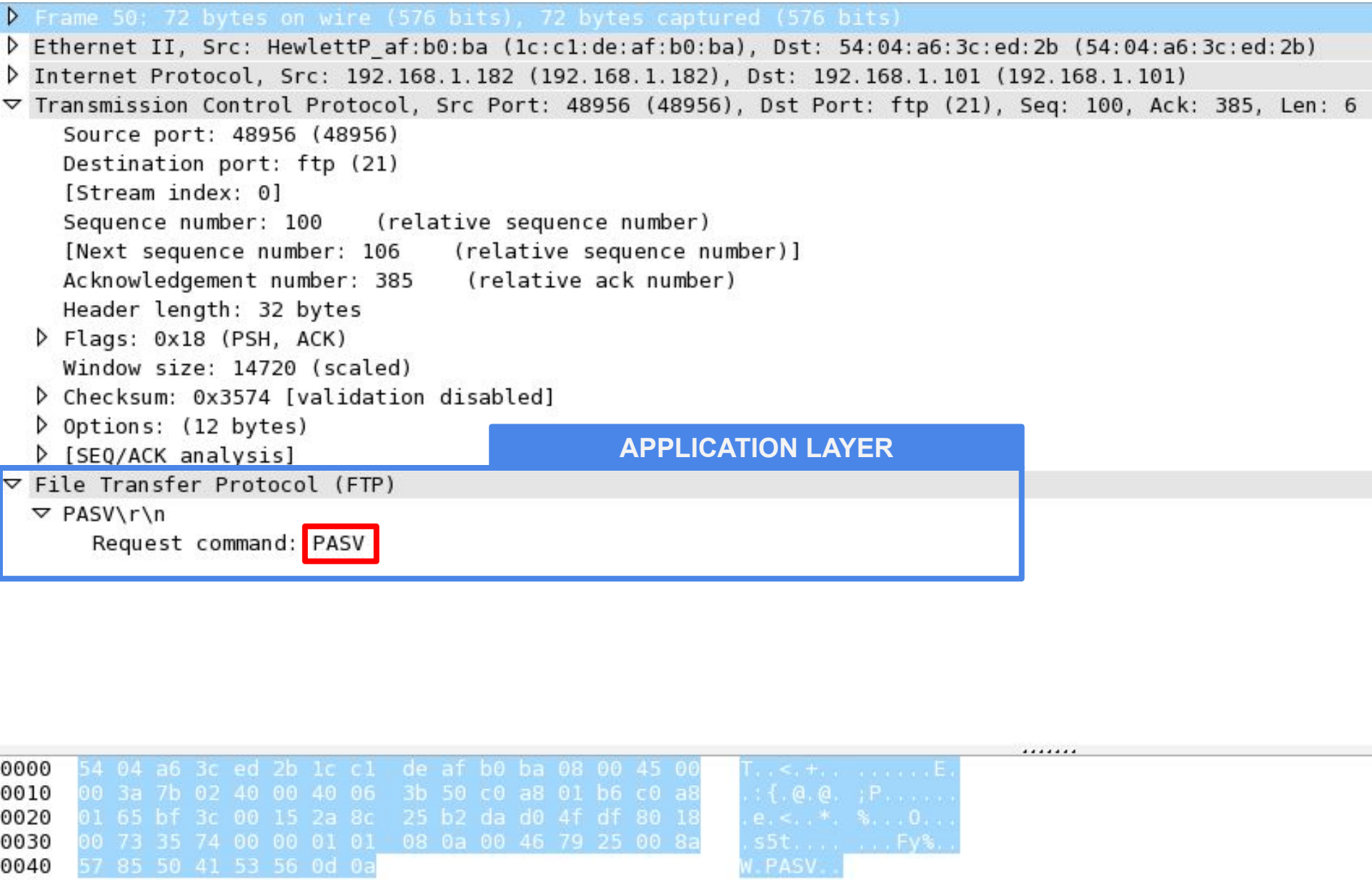
- must open port 21 outbound
- must dynamically open/close the (inbound) ports that the client specifies in the command.

Server-side firewall:

- must open port 21 inbound
- must dynamically open/close the (outbound) ports that the client specifies in the command.

Example: FTP Passive Mode (1)




Frame 50: 72 bytes on wire (576 bits), 72 bytes captured (576 bits)
Ethernet II, Src: HewlettP_af:b0:ba (1c:c1:de:af:b0:ba), Dst: 54:04:a6:3c:ed:2b (54:04:a6:3c:ed:2b)
Internet Protocol, Src: 192.168.1.182 (192.168.1.182), Dst: 192.168.1.101 (192.168.1.101)
Transmission Control Protocol, Src Port: 48956 (48956), Dst Port: ftp (21), Seq: 100, Ack: 385, Len: 6
Source port: 48956 (48956)
Destination port: ftp (21)
[Stream index: 0]
Sequence number: 100 (relative sequence number)
[Next sequence number: 106 (relative sequence number)]
Acknowledgement number: 385 (relative ack number)
Header length: 32 bytes
Flags: 0x18 (PSH, ACK)
Window size: 14720 (scaled)
Checksum: 0x3574 [validation disabled]
Options: (12 bytes)
[SEQ/ACK analysis]
APPLICATION LAYER
File Transfer Protocol (FTP)
PASV\r\n
Request command: PASV
0000 54 04 a6 3c ed 2b 1c c1 de af b0 ba 08 00 45 00
0010 00 3a 7b 02 40 00 40 06 3b 50 c0 a8 01 b6 c0 a8
0020 01 65 bf 3c 00 15 2a 8c 25 b2 da d0 4f df 80 18
0030 00 73 35 74 00 00 01 01 08 0a 00 46 79 25 00 8a
0040 57 85 50 41 53 56 0d 0a

Example: FTP Passive Mode (2)

Both channels are client-initiated.

Client-side firewall:

- must open port 21 outbound
- must dynamically open/close the (outbound) ports that the server specifies

Server-side firewall:

- must open port 21 inbound
- must dynamically open/close the (inbound) ports that the server specifies

Circuit Firewalls (Legacy)

Relay TCP connections.

Client connects to a specific TCP port on the firewall, which then connects to the address and port of the desired server (**not transparent!**).

Essentially, a TCP-level proxy.

Only historical example: SOCKS

Application Proxies

Same as circuit firewalls, but at application layer.

Almost never transparent to clients:

- May require modifications
- Each protocol needs its own proxy server

Inspect, validate, manipulate protocol application data (e.g., rewrite HTTP frames)

Functionalities & Additional Benefits

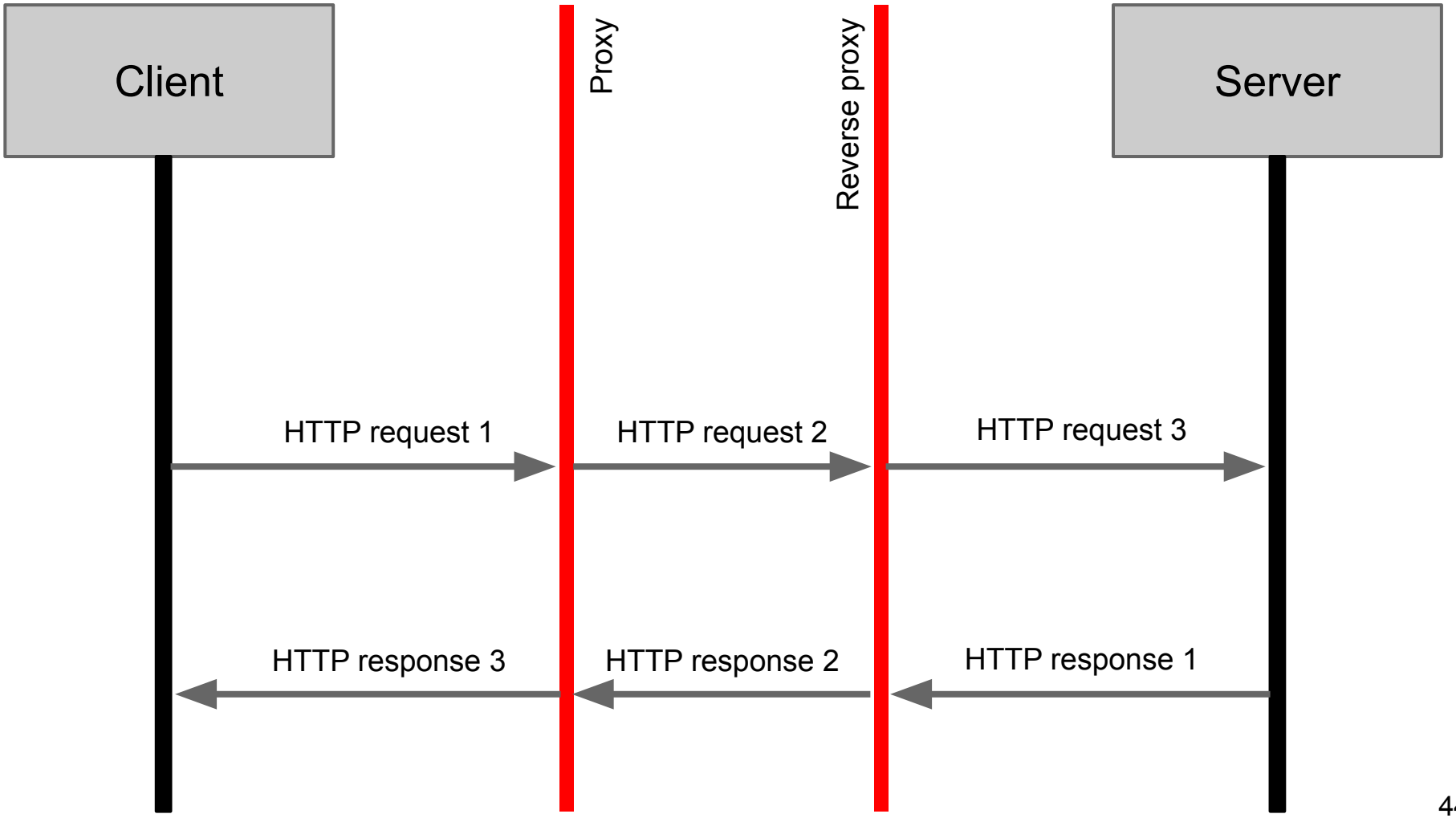
Can authenticate users, apply specific filtering policies, perform advanced logging, content filtering or scanning (e.g., anti-virus/spam).

Can be used to expose a subset of the protocol:

- to defend clients,
- to defend servers ("reverse proxy").

Usually implemented on COTS OSs.

Example: HTTP Proxy (1)



Example: HTTP Proxy (2) - client

Example: HTTP Proxy (3)

Example request that the client sent to the proxy.

The proxy could modify it, if needed.

```
GET / HTTP/1.1
Host: maggi.cc
Proxy-Connection: keep-alive
Cache-Control: no-cache
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8
Pragma: no-cache
User-Agent: Mozilla/5.0 (Macintosh; Intel Mac OS X 10_9_2) AppleWebKit/537.36 (KHTML,
like Gecko) Chrome/35.0.1916.99 Safari/537.36
DNT: 1
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en-US,en;q=0.8,it;q=0.6
Cookie: __utma=15436074.439502269.1326703042.1399588315.1400190112.331;
__utmb=15436074.5.10.1400190112; __utmc=15436074;
__utmz=15436074.1376676768.202.4.utmcsr=google|utmccn=(organic)|utmcmd=organic|utmctr=(n
ot%20provided)
```

Example: HTTP Proxy (4)

Example response that the proxy receives from the server.

The proxy could modify it, if needed.

```
HTTP/1.1 200 OK
Server: nginx
Date: Thu, 15 May 2014 21:48:09 GMT
Content-Type: text/html; charset=utf-8
Connection: keep-alive
Vary: Accept-Encoding
Content-Length: 19263

<!doctype html lang="en">

<!--
paulirish.com/2008/conditional-stylesheets-vs-css-hacks-answer-neither/
-->
```

Architectures for secure networks

Dual- or Multi-zone Architectures (1)

In most cases, the perimeter defense works on the assumption that what is “good” is inside, and what's outside should be kept outside if possible.

There are two counterexamples:

- Access to resources from remote (i.e., to a web server, to FTP, mail transfer).
- Access from remote users to the corporate network.

Dual- or Multi-zone Architectures (2)

Problem: if we mix externally accessible servers with internal clients, we lower the security of the internal network.

Solution: we allow external access to the accessible servers, but not to the internal network.

General idea: split the network by privileges levels. Firewalls to regulate access.

Dual- or Multi-zone Architectures (3)

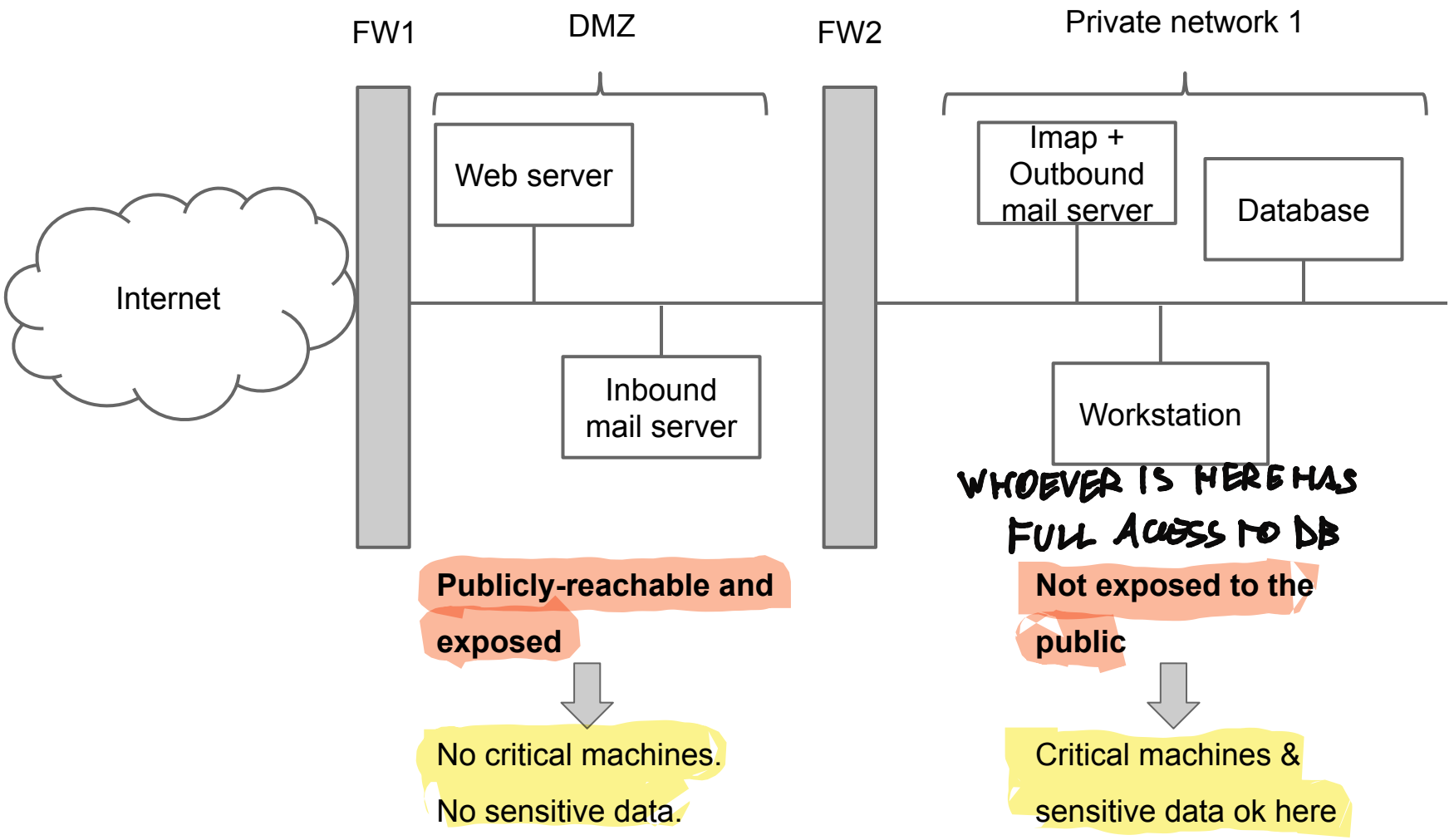
In practice, we create a semi-public zone called DMZ (demilitarized zone).

The DMZ will host public servers (web, FTP, public DNS server, intake SMTP).

On the DMZ no critical or irreplaceable data.

The DMZ is almost as risky as the Internet.

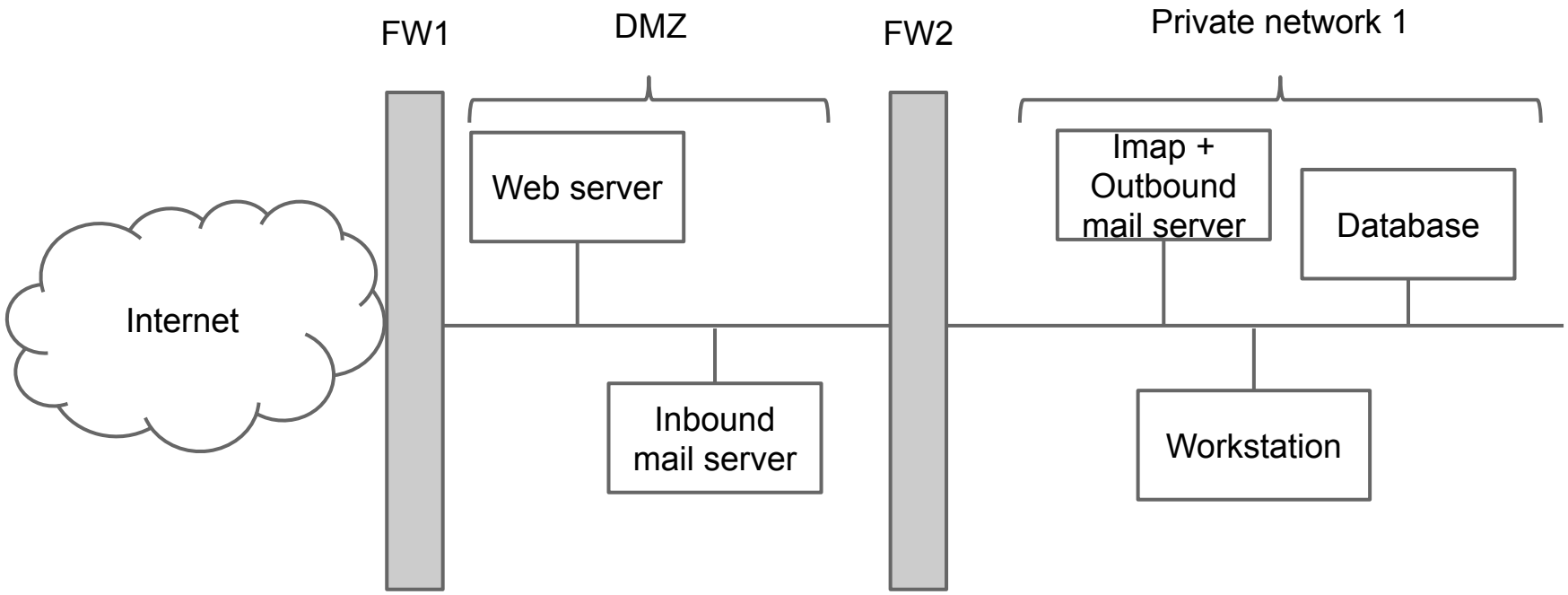
Example DMZ + Private Zone



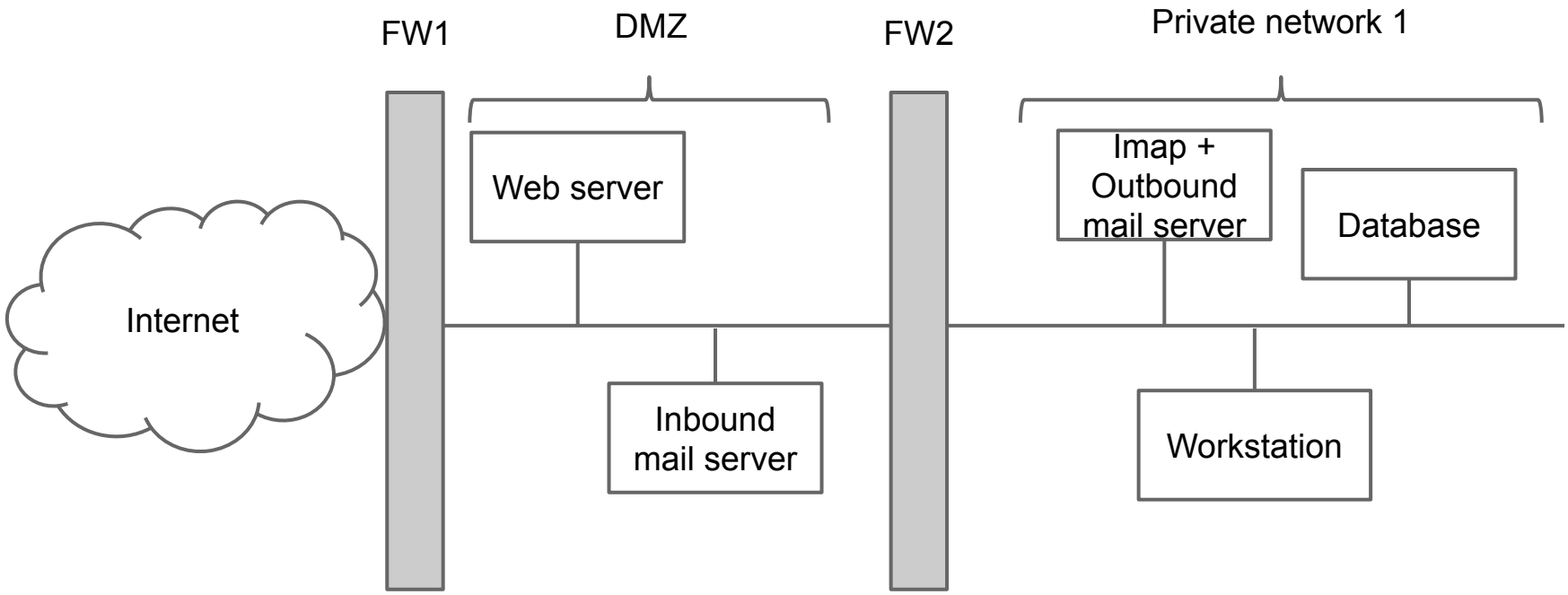
Exercise

Write the rules for the previous network layout.

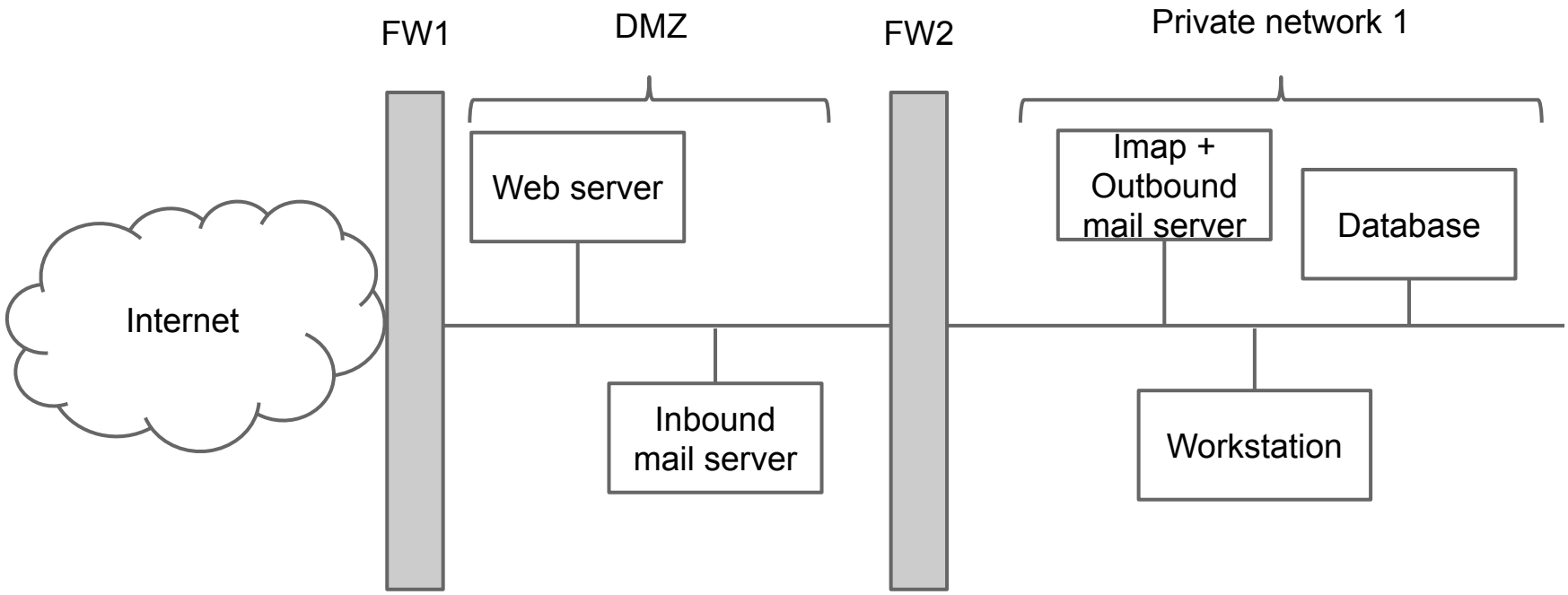
Implement it with a **single, multi-homed firewall** (i.e., a firewall with more than one port).



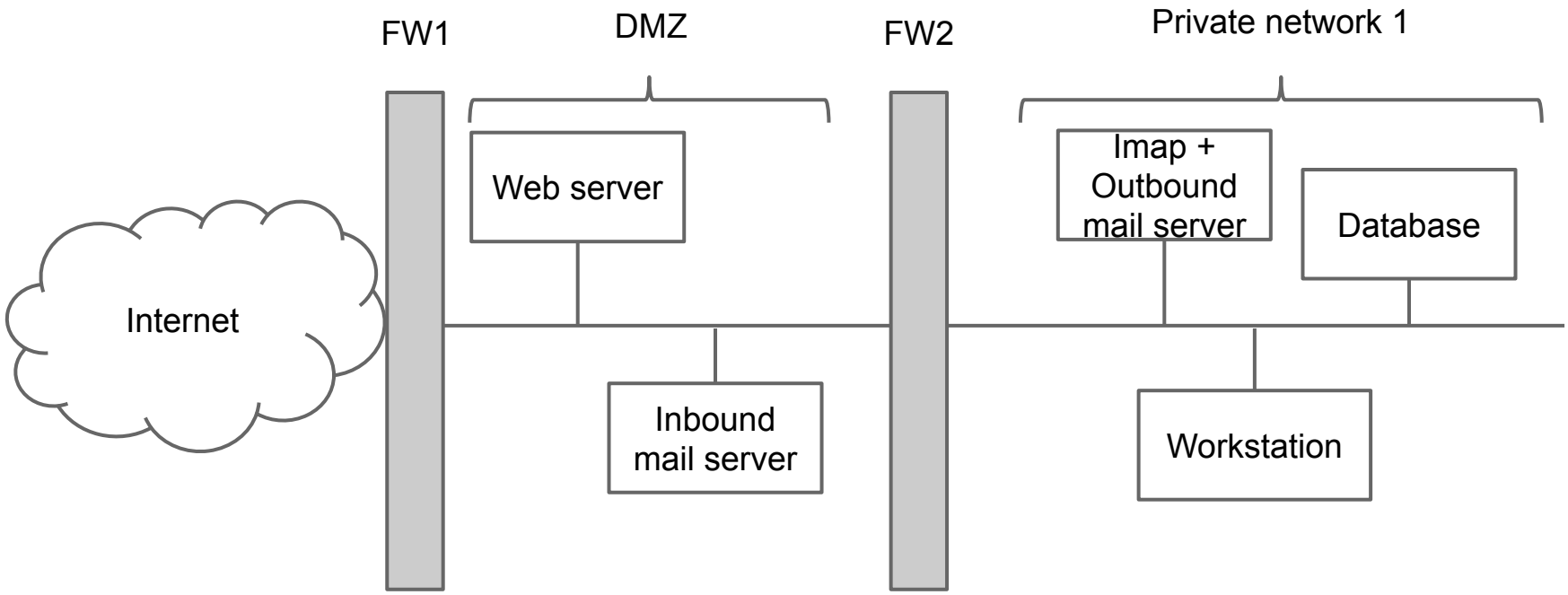
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)



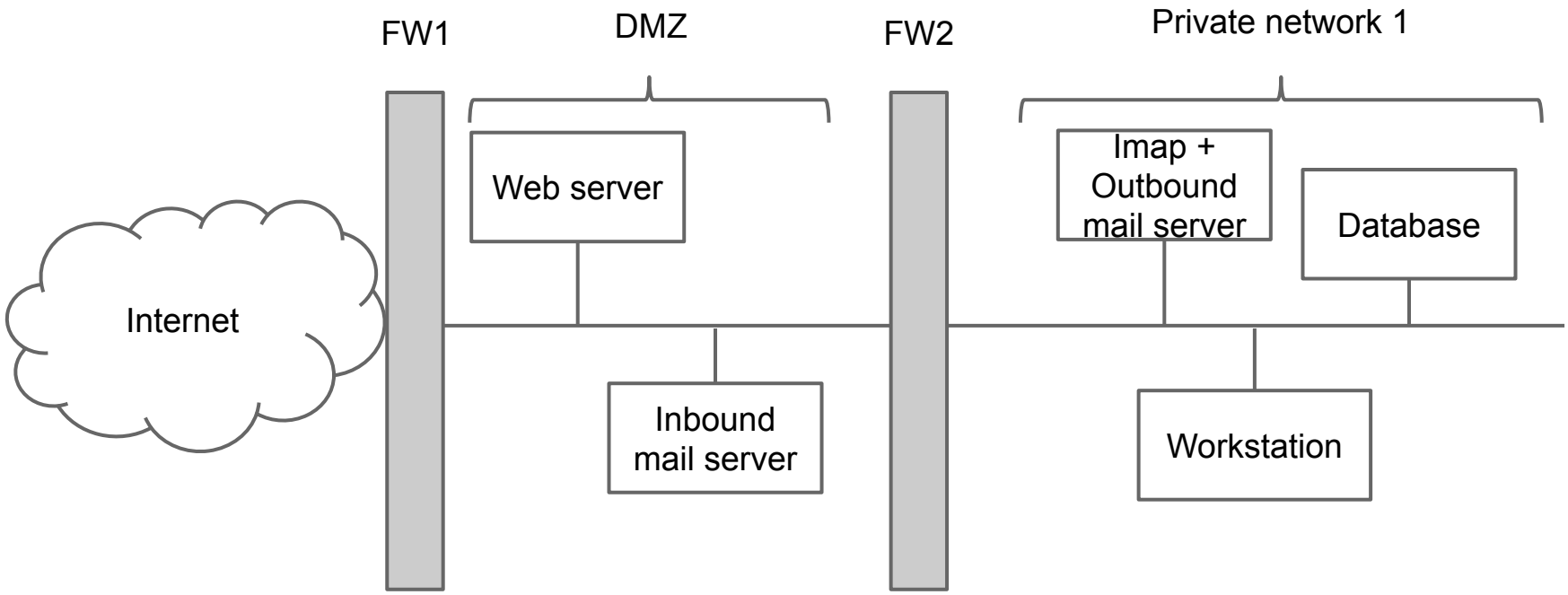
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*



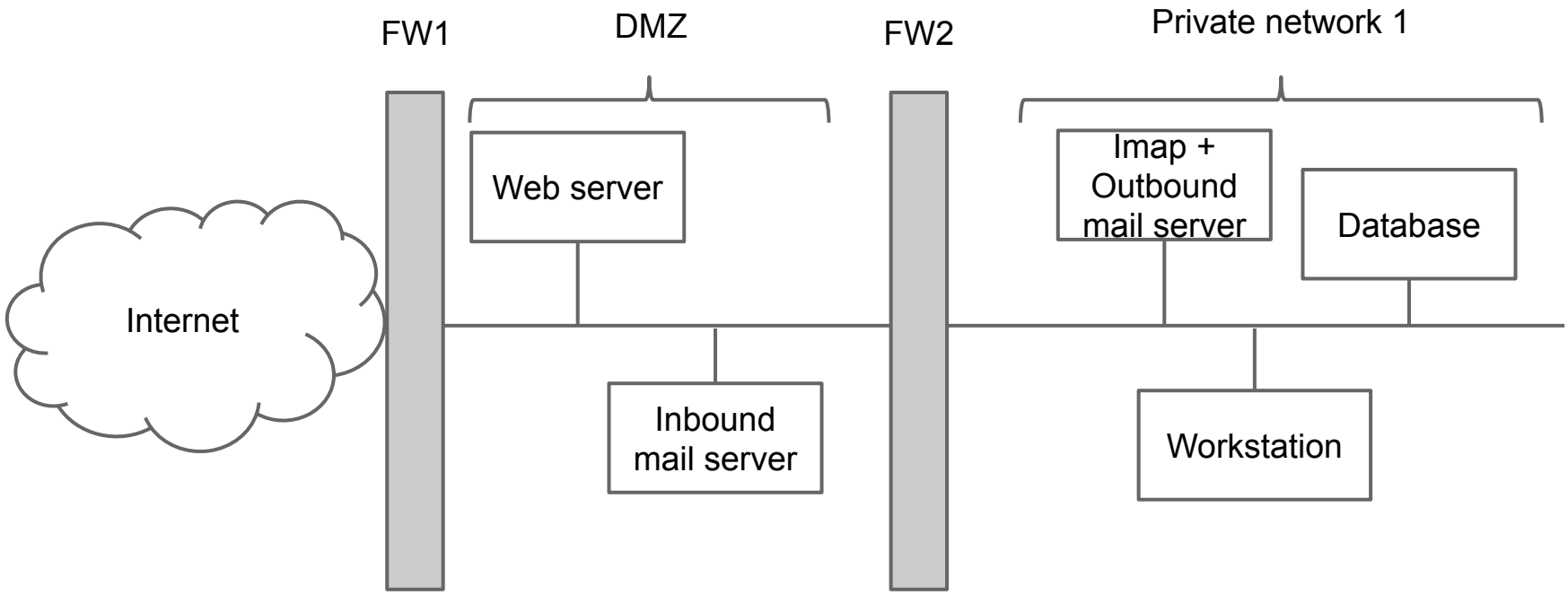
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*



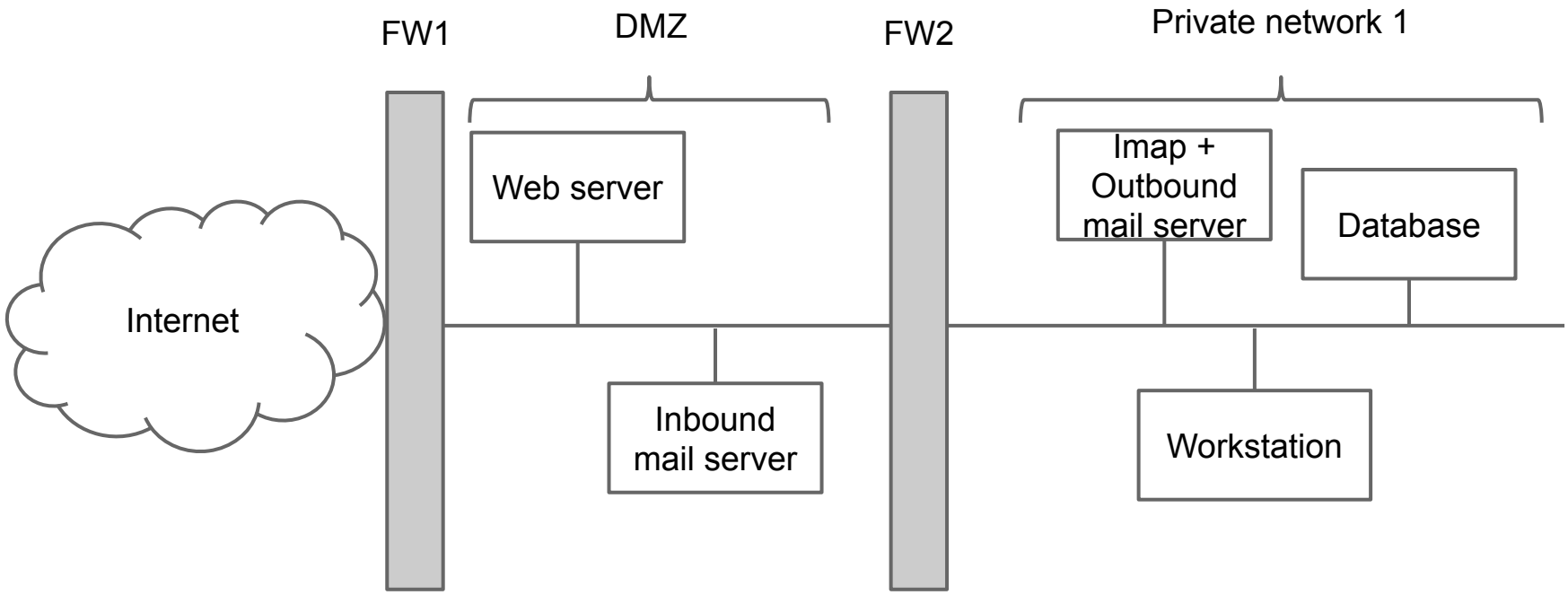
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*



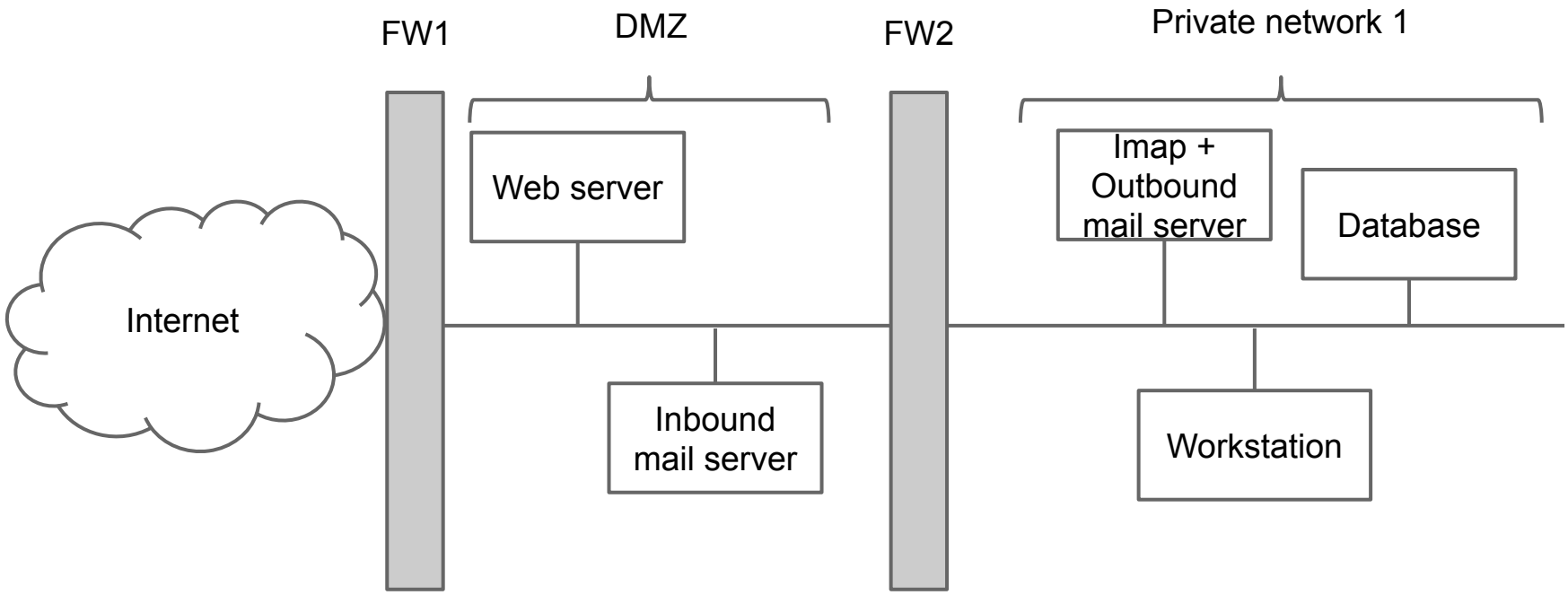
Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*143*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*587*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*
FW2	*Workstation_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW1	*Workstation_IP*	*ANY*	*DMZ → Internet*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*



Firewall	Src IP	Src PORT	Direction of the 1st packet	Dst IP	Dst PORT	Policy	Description
FW1 (example)	10.0.0.1 (example)	ANY	zone 1 -> zone 2	192.168.0.2 (example)	443	DENY	(example: the X server in zone 1 cannot contact the Y server)
FW1	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*WS_IP*	*443 (HTTPS)*	*ALLOW*	*The webserver is publicly reachable*
FW1	*ANY*	*ANY*	*Internet -> DMZ*	*SMTPIN_IP*	*25*	*ALLOW*	*The SMTP server is publicly reachable*
FW2	*ALL*	*ANY*	*ANY*	*ALL*	*ANY*	*DENY*	*Default deny*
FW2	*SMTPIN_IP*	*ANY*	*DMZ → Z1*	*IMAP_IP*	*587*	*ALLOW*	*SMTPIn relays the incoming e-mails to the POP3\IMAP server*
FW2	*Workstation_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW1	*Workstation_IP*	*ANY*	*DMZ → Internet*	*ANY*	*80, 443*	*ALLOW*	*The workstation connects to websites*
FW2	*SMTPOUT_IP*	*ANY*	*Z1 → DMZ*	*ANY*	*25*	*ALLOW*	*The application server sends email (relayed by the SMTPOut server)*
FW1	*SMTPOUT_IP*	*ANY*	*DMZ → Internet*	*ANY*	*25*	*ALLOW*	*The application server sends email (relayed by the SMTPOut server)*

Virtual Private Networks (VPNs)

Requirements:

- Remote employees need to work "as if" they were in the office, accessing resources on the private zone.
- Connecting remote sites without using dedicated lines.

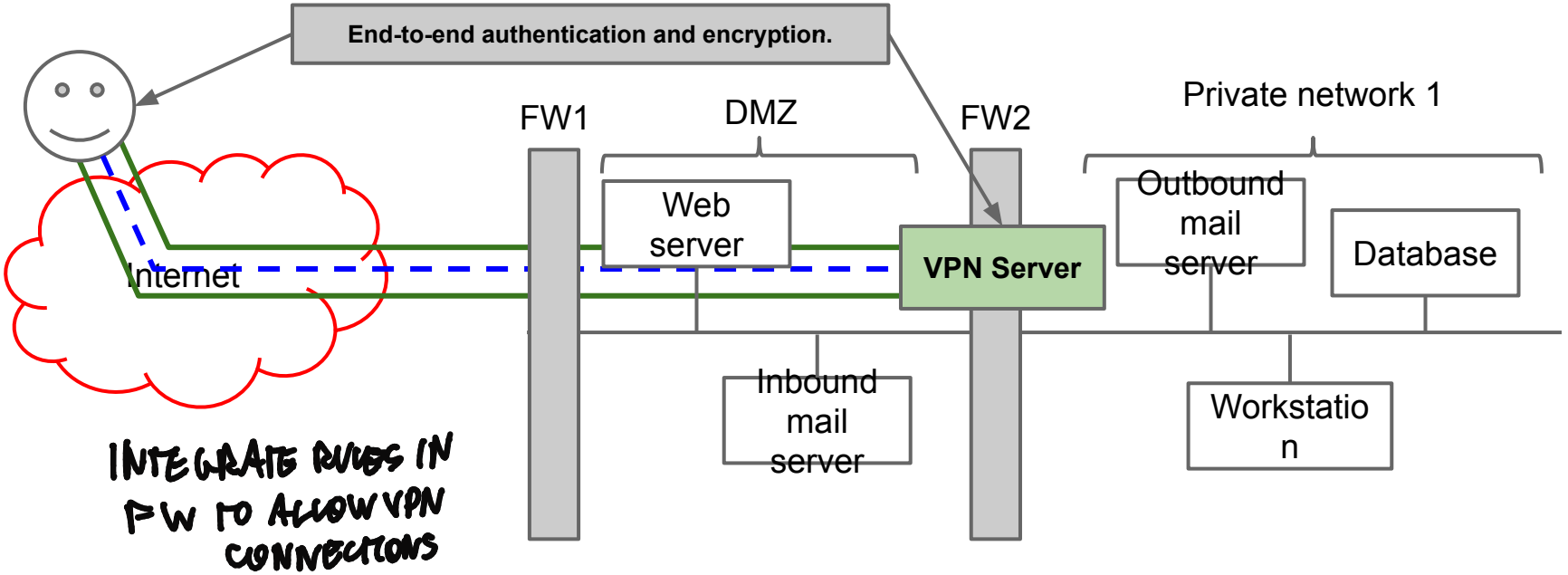
Which means:

Ensure CIA to data transmitted over a public network (i.e., the Internet).

VPNs: Basic Concept

Solution: use a VPN, an encrypted overlay connection over a (public) network.

Many different technologies, same basic idea.



Two VPN Modes: Full vs. Split

Full tunnelling

- Every packet goes through the tunnel.

Split tunnelling

- Traffic to the corporate network: in VPN;
- Traffic to the Internet: directly to ISP.

Two VPN Modes: Full vs. Split

Full tunnelling

- Every packet goes through the tunnel.
- Traffic multiplication, could be inefficient.
- Single point of control and application of all security policies as if the client were in the corporate network.

Split tunnelling

- Traffic to the corporate network: in VPN;
- Traffic to the Internet: directly to ISP.
- More efficient, less control.
  - Just similar to the case of the PC connected via 4G modem to the Internet.

VPN Technologies

PPTP (Point-to-point Tunnelling Protocol): proprietary Microsoft protocol, variant of PPP with authentication and confidentiality.

VPN over pure **TLS** (we will see the TLS protocol in detail), **SSH tunnel** , or **OpenVPN** (TLS based).

IPSEC

- Security extensions of IPv6, backported to IPv4
- Authentication and confidentiality at IP layer

Conclusions

Firewalls can enforce policies only on the traffic that they can inspect, and up to the layer that they can decode.

Firewalls can be used to implement multi-zone architectures.

VPNs solve the problem of creating a trusted network transport over an untrusted channel.

11. Network Security
The tale of SSL/TLS and SET

Computer Security Courses @ POLIMI

Issues of Communications Security

- Problems of remoteness
  - Trust factor between parties
  - Use of sensitive data
  - Atomicity of transaction
- Internet protocol problems
  - Authentication
  - Confidentiality
- Transparence and critical mass problem

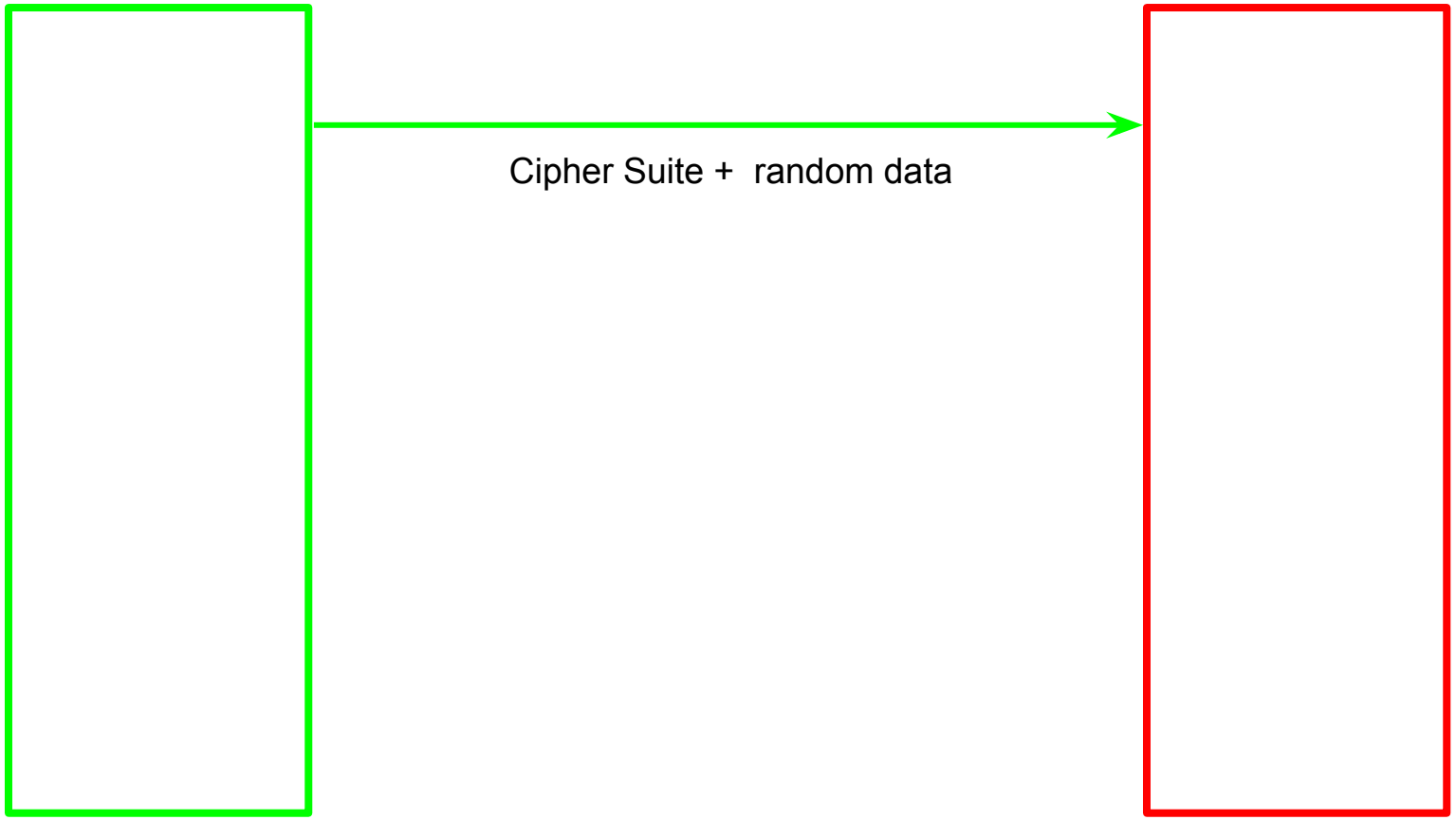
A Tale of Two Protocols

- Two valiant protocols fought against the darkness
- HTTP over SSL (Secure Socket Layer), or HTTPS
  - Communication confidentiality and integrity
  - Mutual authentication
  - No guarantees on data usage
  - No strict authentication of client (in practice)
- SET (Secure Electronic Transaction)
  - Guarantees on data usage and transaction security enforcement
  - Missing critical mass support

SSL -> TLS

- Originally designed by Netscape for securing web communication
  - de facto standard also for other protocols
  - IETF standardized TLS, which comes after version SSL v3, and is now at version 1.3.
  - All versions up to TLS 1.1 (included) are insecure
- TLS enforces:
  - Confidentiality and integrity of the communications
  - Server authentication
  - Client authentication (optionally)
- Uses both symmetric and asymmetric cryptography for performance reasons

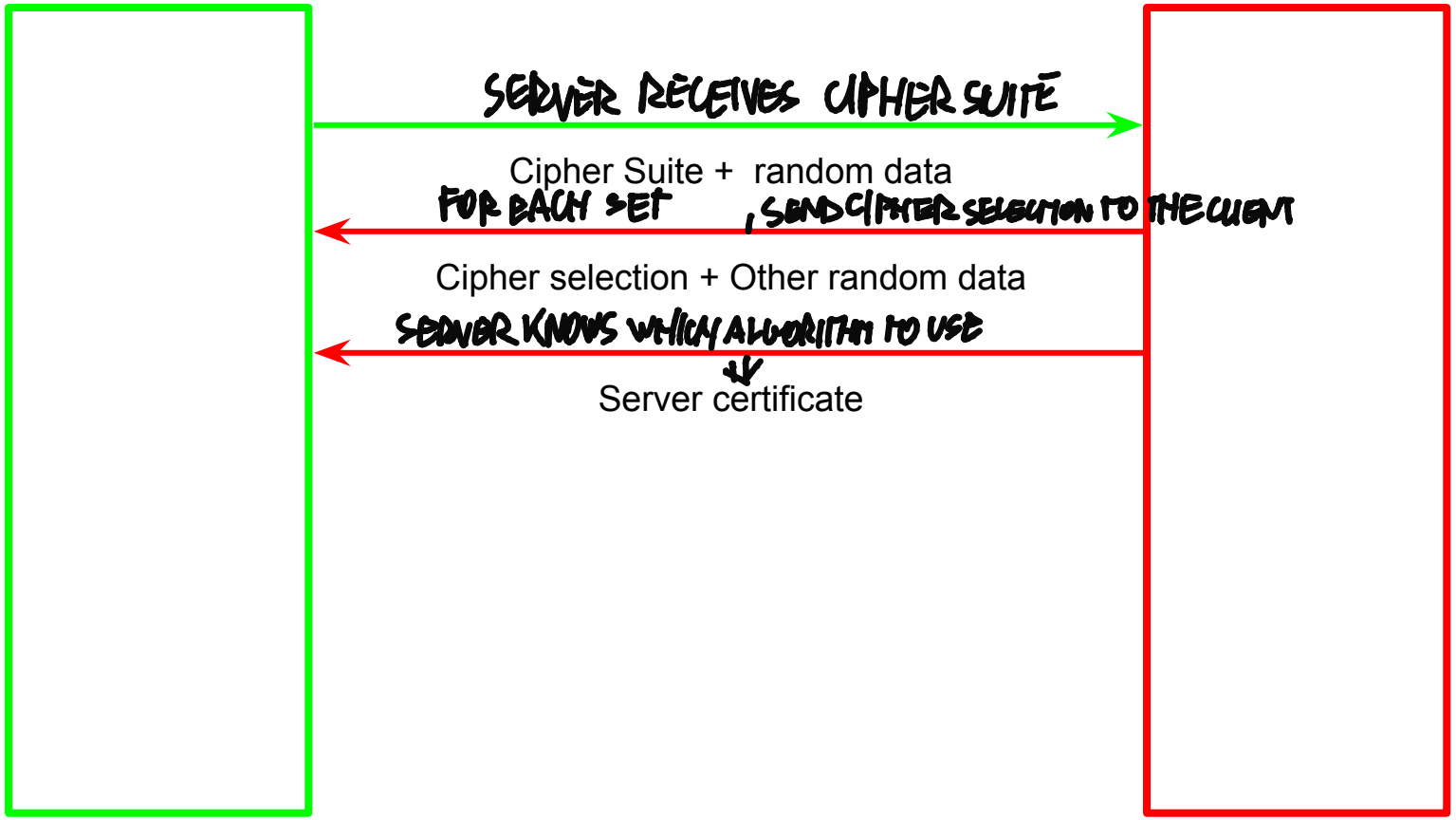
TLS Handshake Phases



Cipher Suite

- TLS designed to be flexible wrt to technical evolution
- Clients and servers may use different *suites* of algorithms for different functions
  - a key exchange/key encapsulation algorithm
  - a symmetric encryption algorithm
  - a digital signature algorithm
  - a hash function (for symm. key derivation)
- During handshake, cipher suites are compared to agree on shared algorithms in order of preference
- The standard mandates the implementation of a minimal cipher set

Server Authentication



Verification of Server Certificate

- Is the certificate in the validity period?
- Is the root CA trusted?
- Is the certificate valid?
- Is it revoked?
- **Is the *name* of the server in the certificate the same that I requested?**

Remember the implementation issues that we have learned a couple of months ago?

https://docs.google.com/presentation/d/1bFNs37n_zsp6f167U0GxaAkMekNi049hYrsc09NdFFw/edit#slide=id.g1826c6e

11. Network Se

File Edit View

8

Verification of Server Certificate

- Is the certificate in the validity period?
- Is the root CA trusted?
- Is the certificate valid?
- Is it revoked?
- Is the name of the server in the certificate the same that I requested?

Remember the implementation issues that we have learned a couple of months ago?

9

10

Secret Transmission

11

(Optional) Client Authentication

Equifax Secure Certificate Authority
GeoTrust Global CA
Google Internet Authority G2
*.google.com

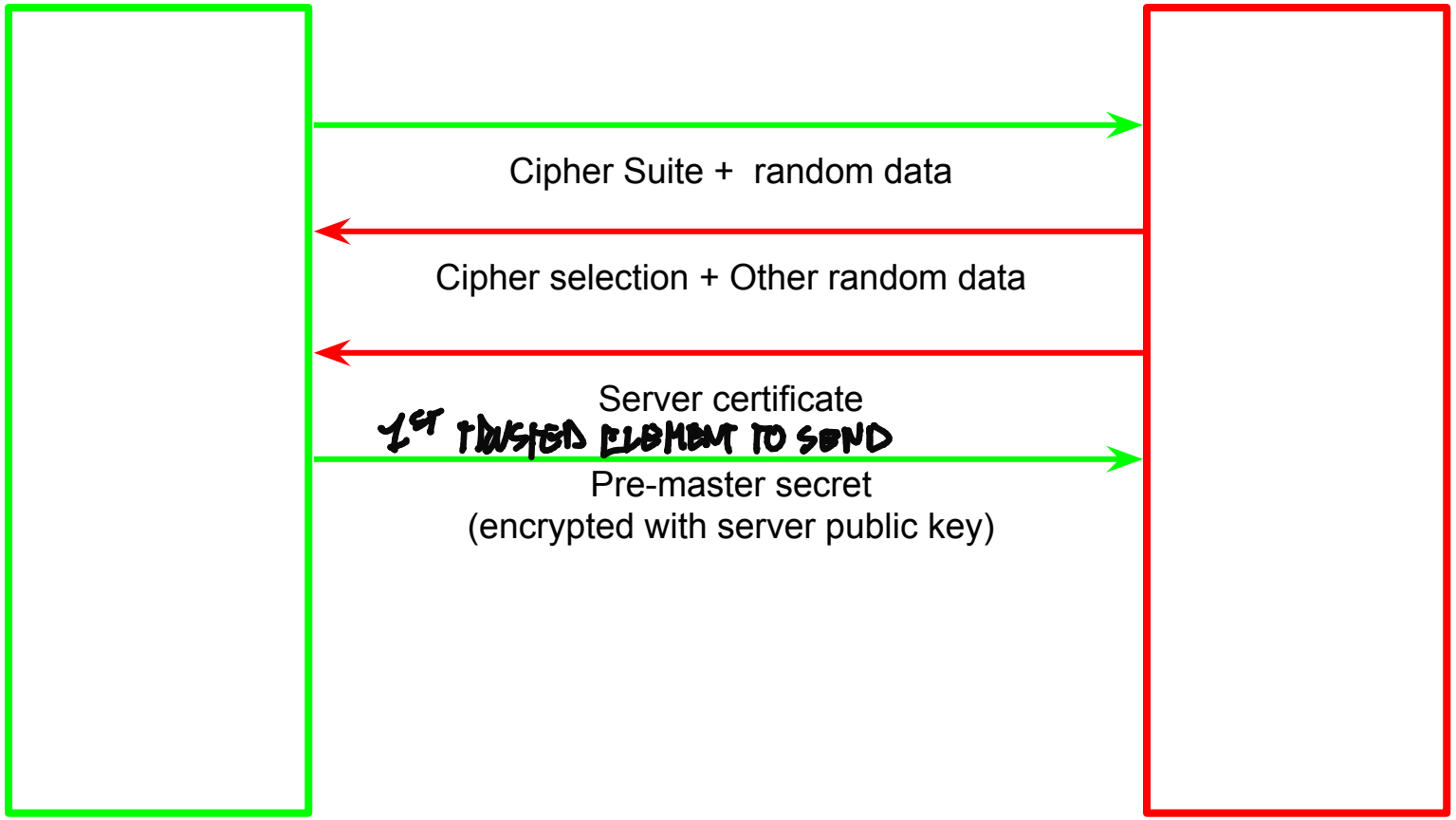
***.google.com**
Issued by: Google Internet Authority G2
Expires: Tuesday 5 August 2014 02 h 00 min 00 s Central European Summer Time
This certificate is valid

Details

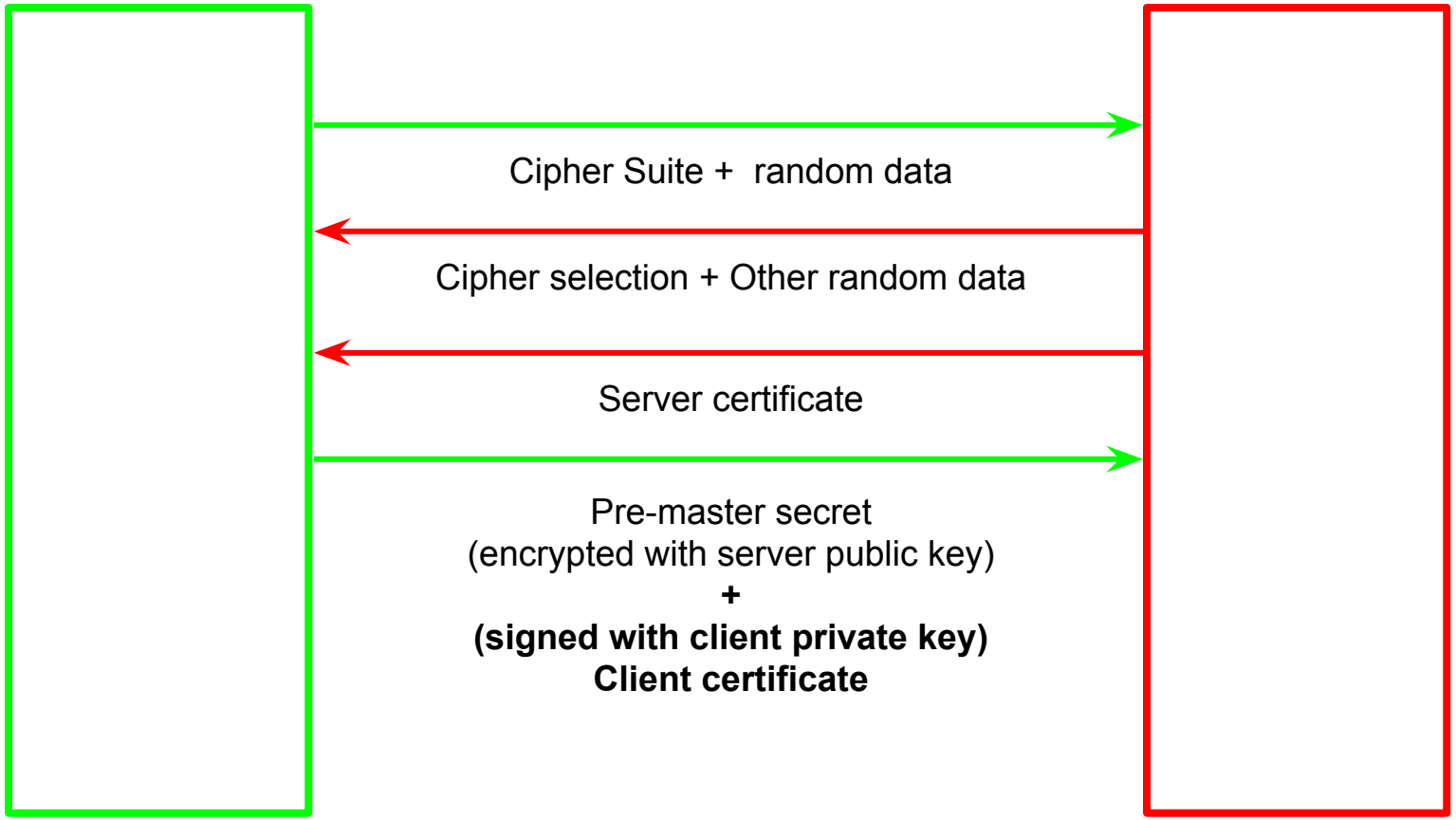
Subject Name	
Country	US
State/Province	California
Locality	Mountain View
Organization	Google Inc
Common Name	*.google.com
Issuer Name	
Country	US
Organization	Google Inc
Common Name	Google Internet Authority G2
Serial Number	1405391832758935466
Version	3
Signature Algorithm	SHA-1 with RSA Encryption (1.2.840.113549.1.1.5)
Parameters	none
Not Valid Before	Wednesday 7 May 2014 14 h 37 min 59 s Central European Summer Time

OK

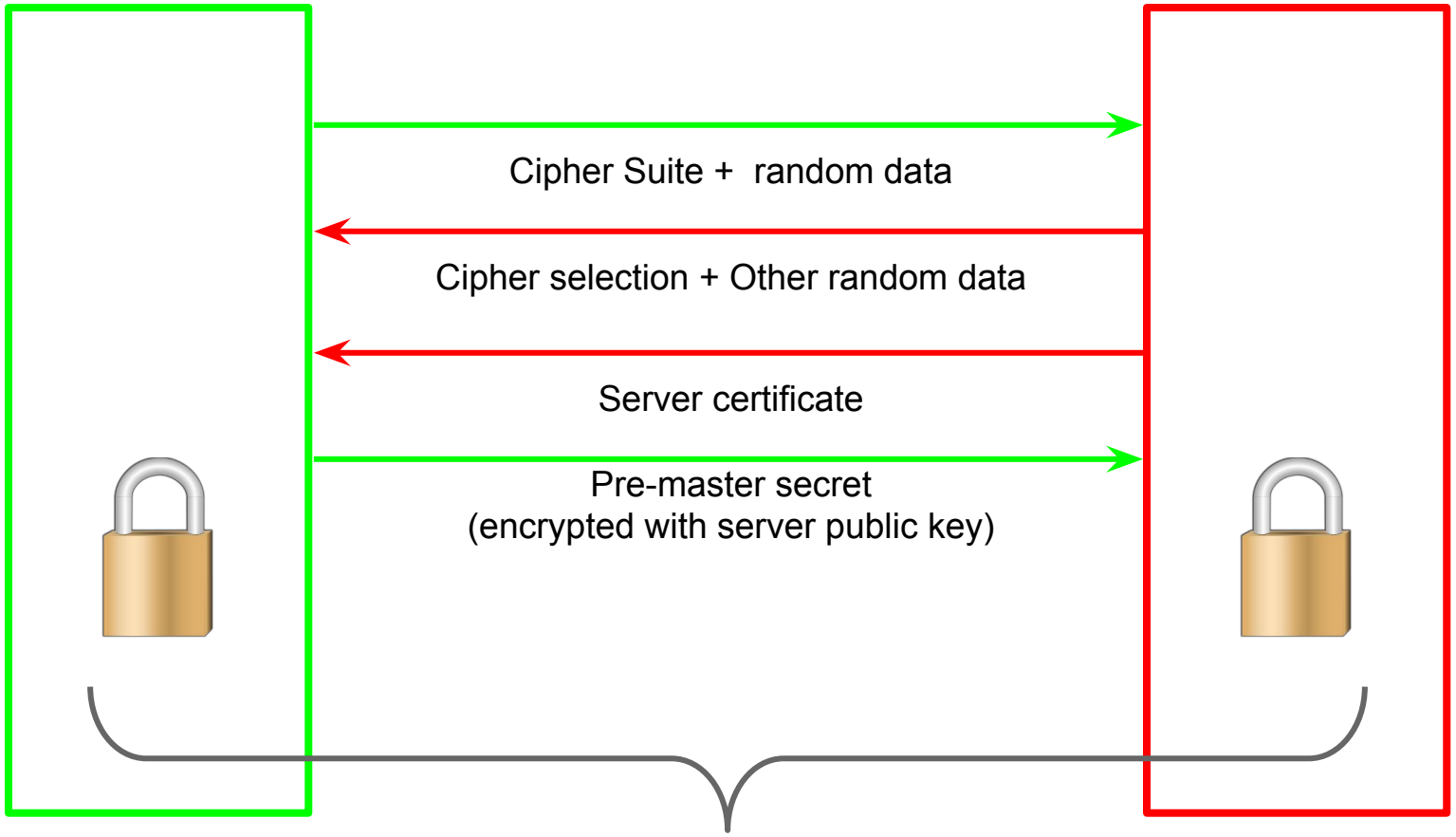
Secret Transmission



(Optional) Client Authentication

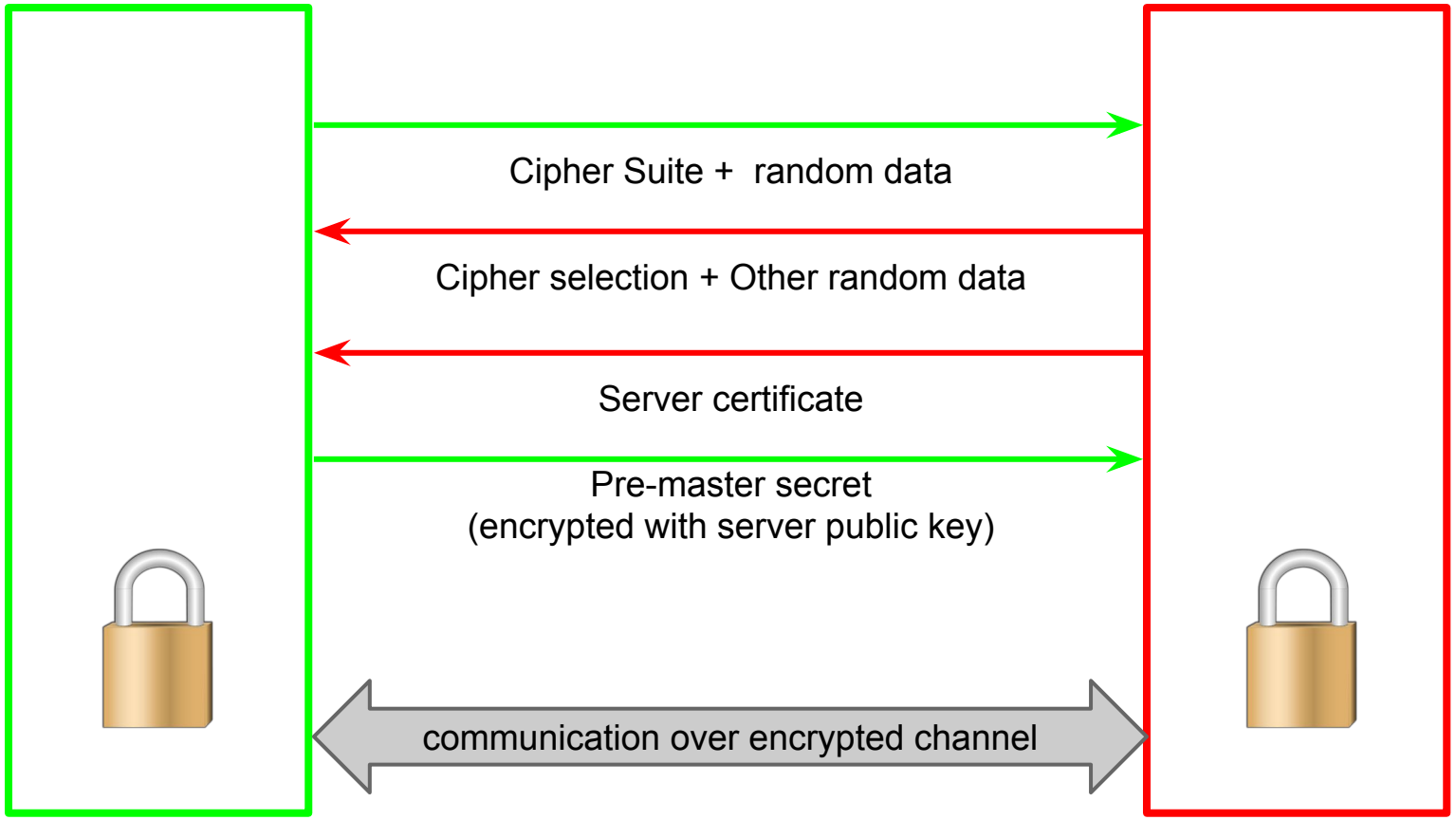


Secret Computation



Compute shared secret from pre-master secret, client random data and server random data

Encrypted Communication Phase



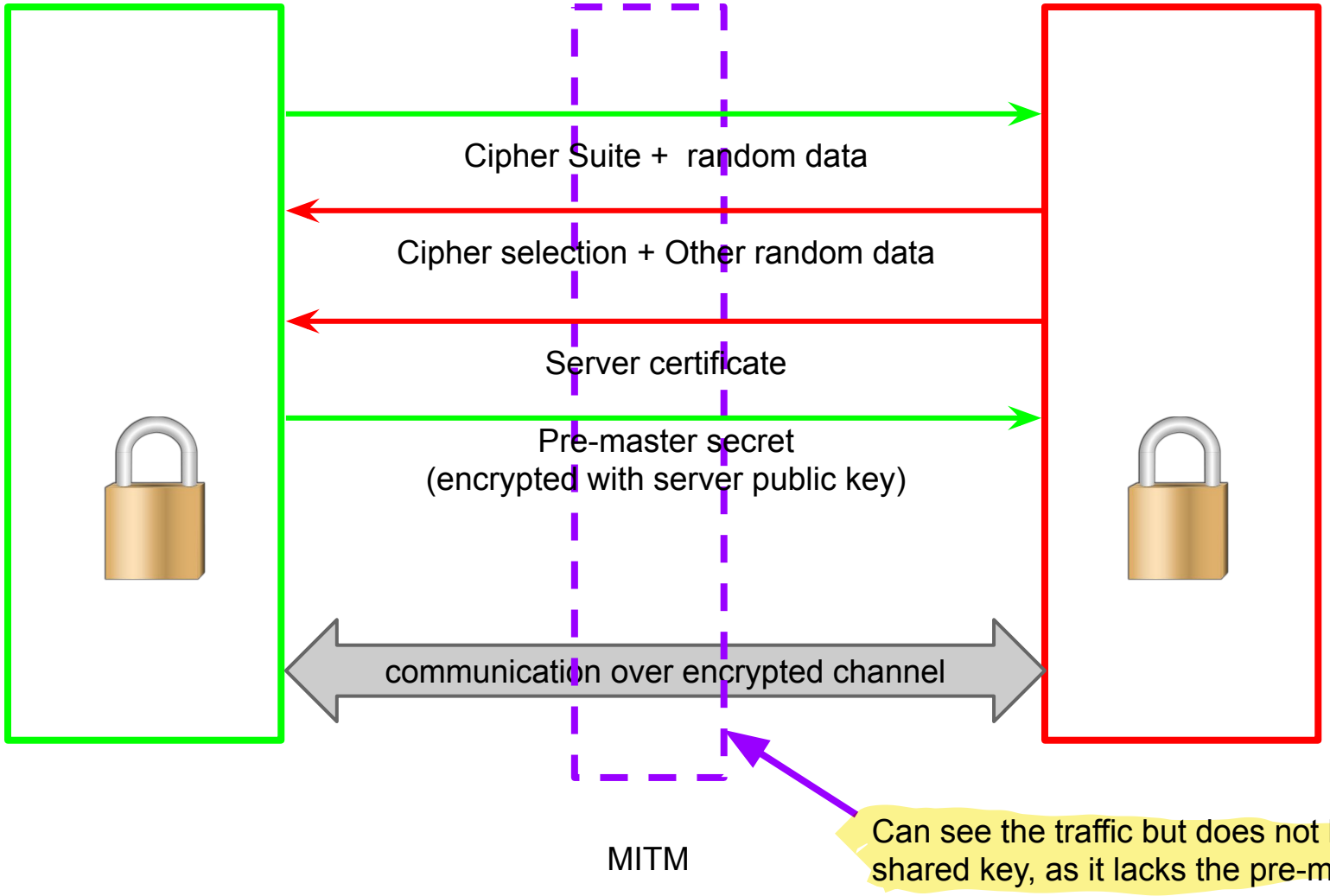
Is TLS Resistant to MITM?

What could the MITM do?

- Let the original certificate through

"OBSERVE THE CONVERSATION"

Cut out!

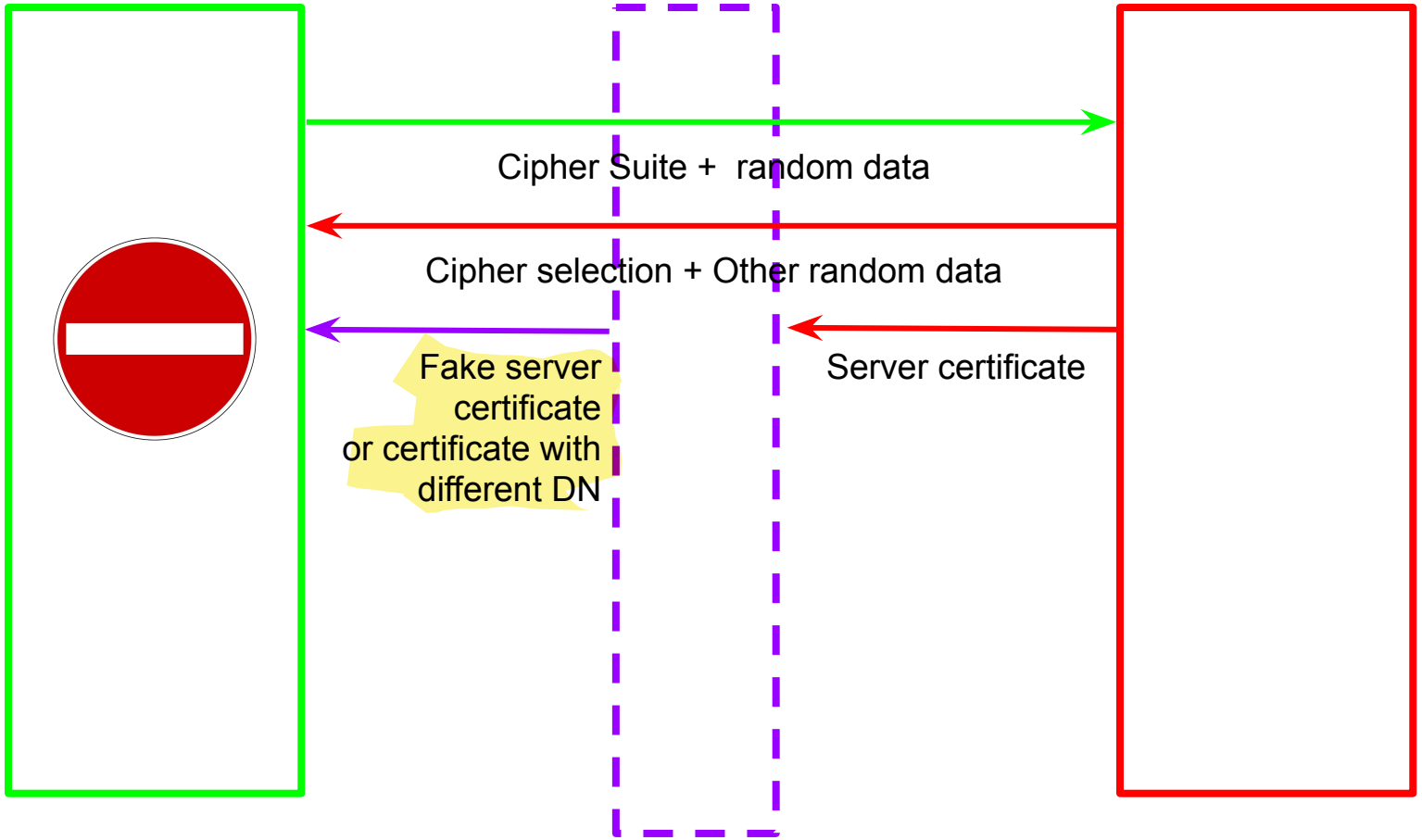

Cipher Suite + random data
Cipher selection + Other random data
Server certificate
Pre-master secret
(encrypted with server public key)
communication over encrypted channel
MITM
Can see the traffic but does not have the shared key, as it lacks the pre-master

Is TLS Resistant to MITM?

What could the MITM do?

- ~~Let the original certificate through~~
  - Needs to actually have the key on that cert!
- Send a fake certificate (i.e., signed by a non-trusted CA)
- Send a good certificate with a fake name

Rejected!



MITM

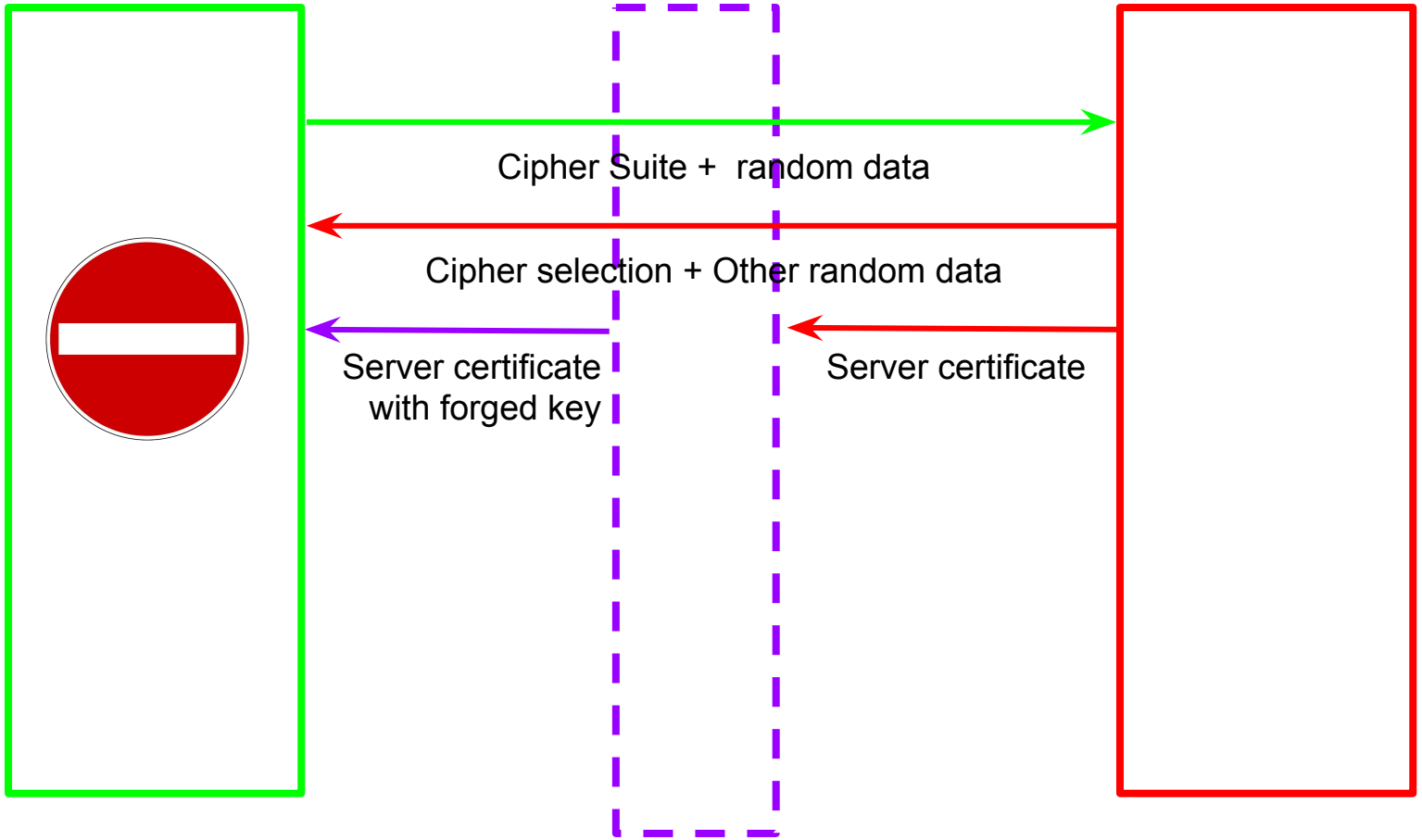
Is TLS resistant to MITM?

What could the MITM do?

- ~~Let the original certificate through~~
- ~~Send a fake certificate (i.e. signed by a non-trusted CA)~~
- ~~Send a good certificate with a fake name~~
- Send a good certificate but substitute the public key (invalidating the signature)

HASH WILL NOT CORRESPOND

Rejected!


Cipher Suite + random data
Cipher selection + Other random data
Server certificate with forged key
Server certificate
MITM

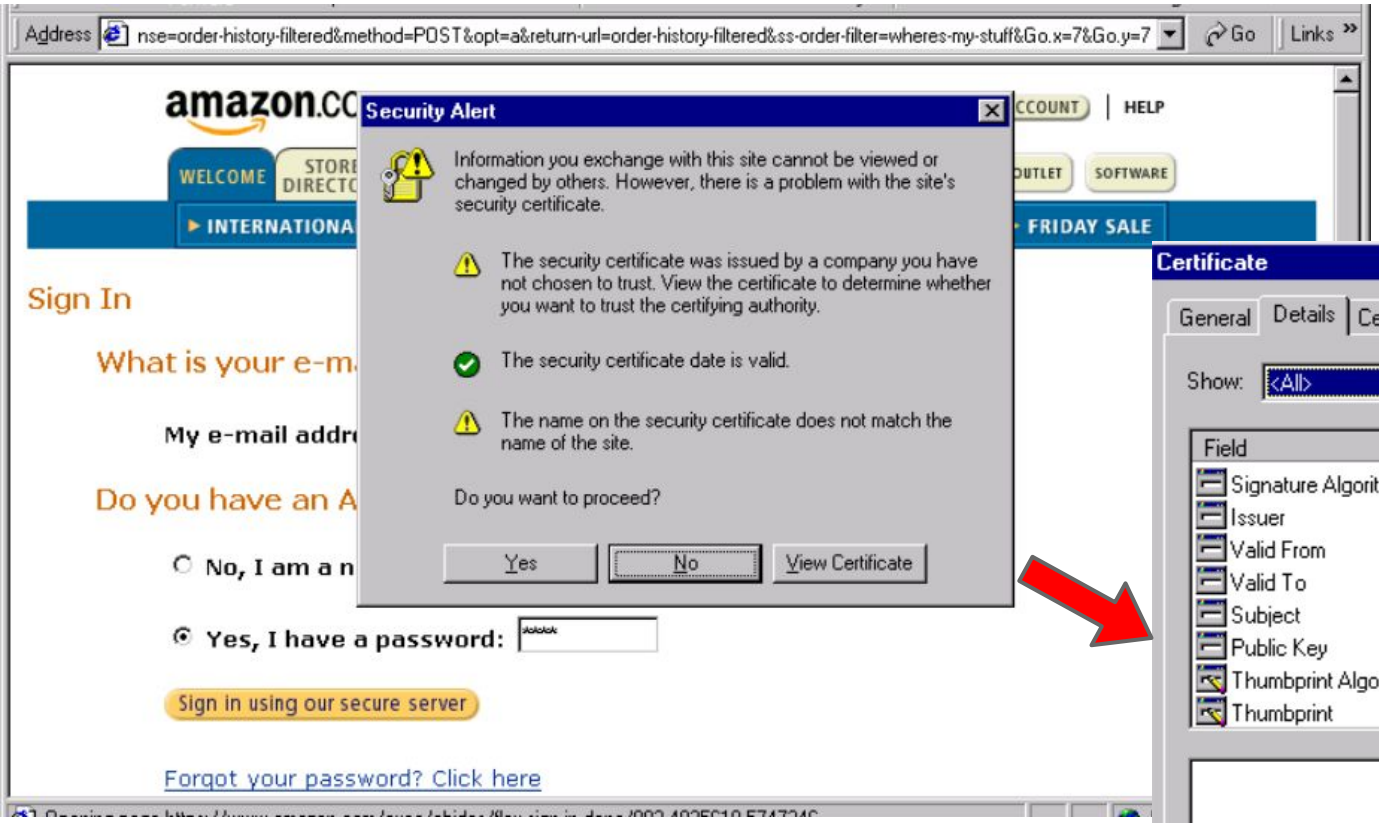
Is TLS resistant to MITM?

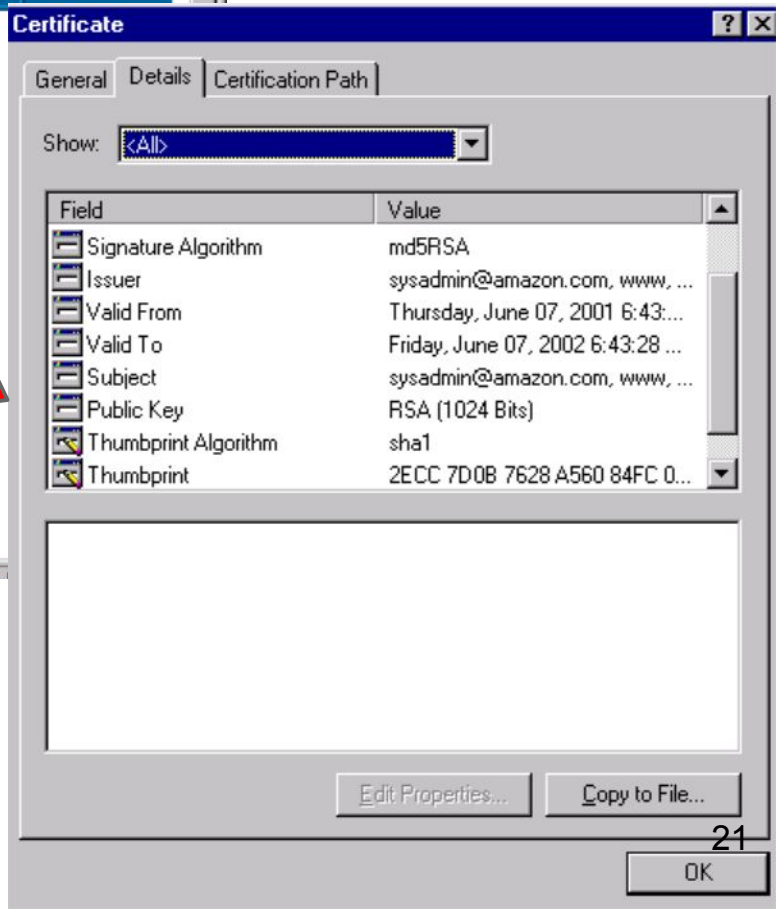
What could the MITM do?

- ~~Let the original certificate through~~
- ~~Send a fake certificate (i.e. signed by a non-trusted CA)~~
- ~~Send a good certificate with a fake name~~
- ~~Send a good certificate but substitute the public key (making it invalid)~~

Nothing: TLS is resistant to MITM by design!

Social Engineering = Fail



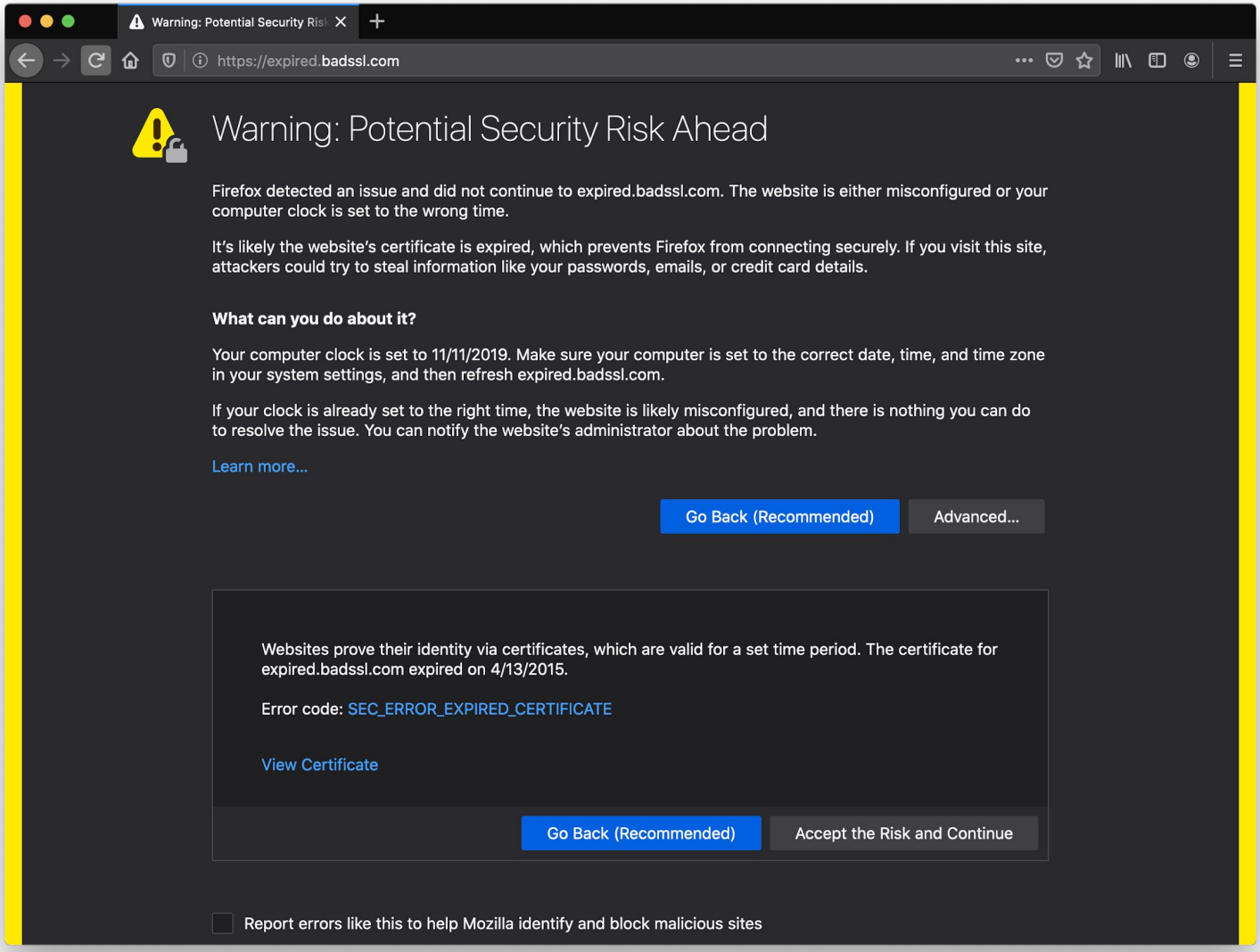


If the user clicks "yes" the assumptions of TLS are violated and attack is successful.

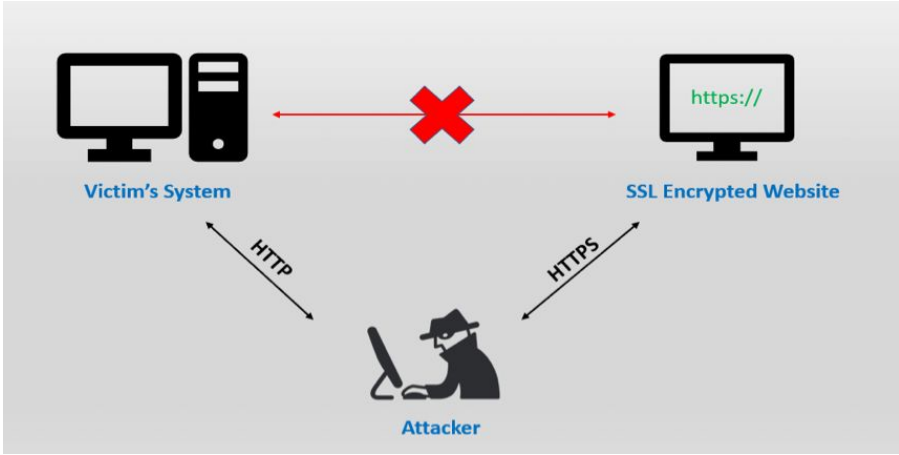
If he clicks no, he cannot buy his book.

Guess what's going to happen...

Security UI: Evolution



Security UI Pitfalls & SSL strip



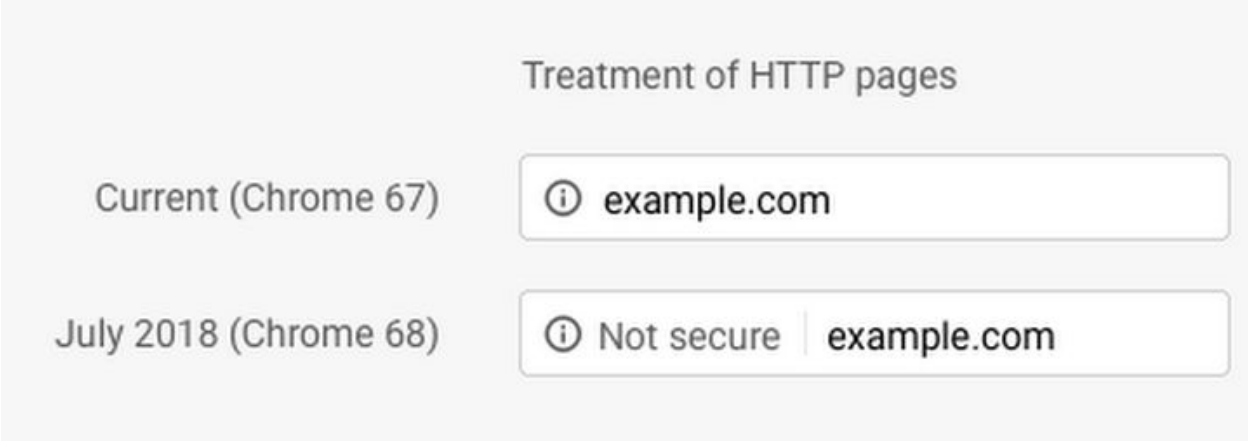
View source:

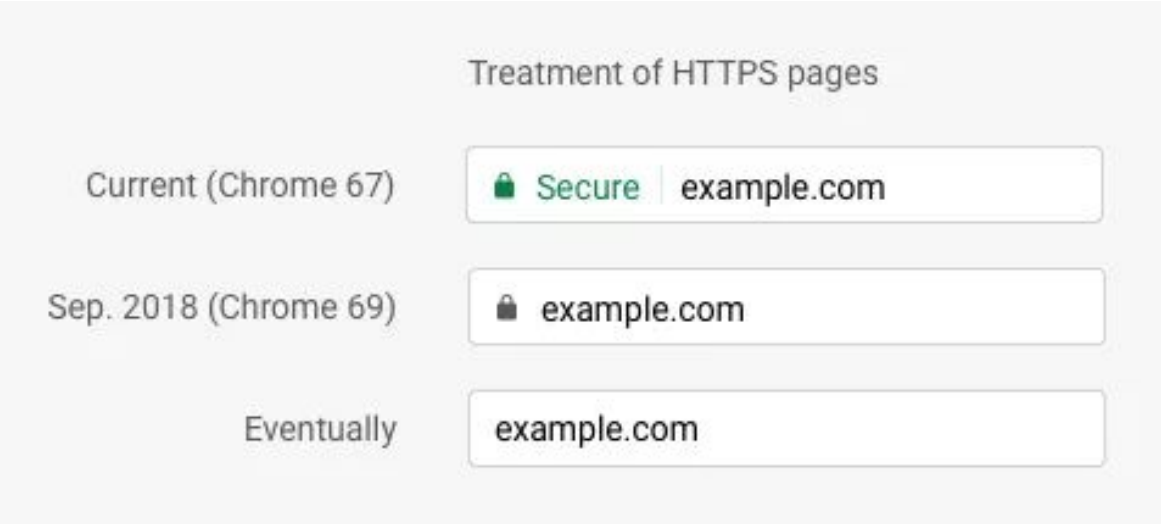
<form method="post"

action="**https**://onlineservices.wachovia.com/..."

Source: https://cs155.stanford.edu/

Security UI: Evolution





TLS: Pros and Cons

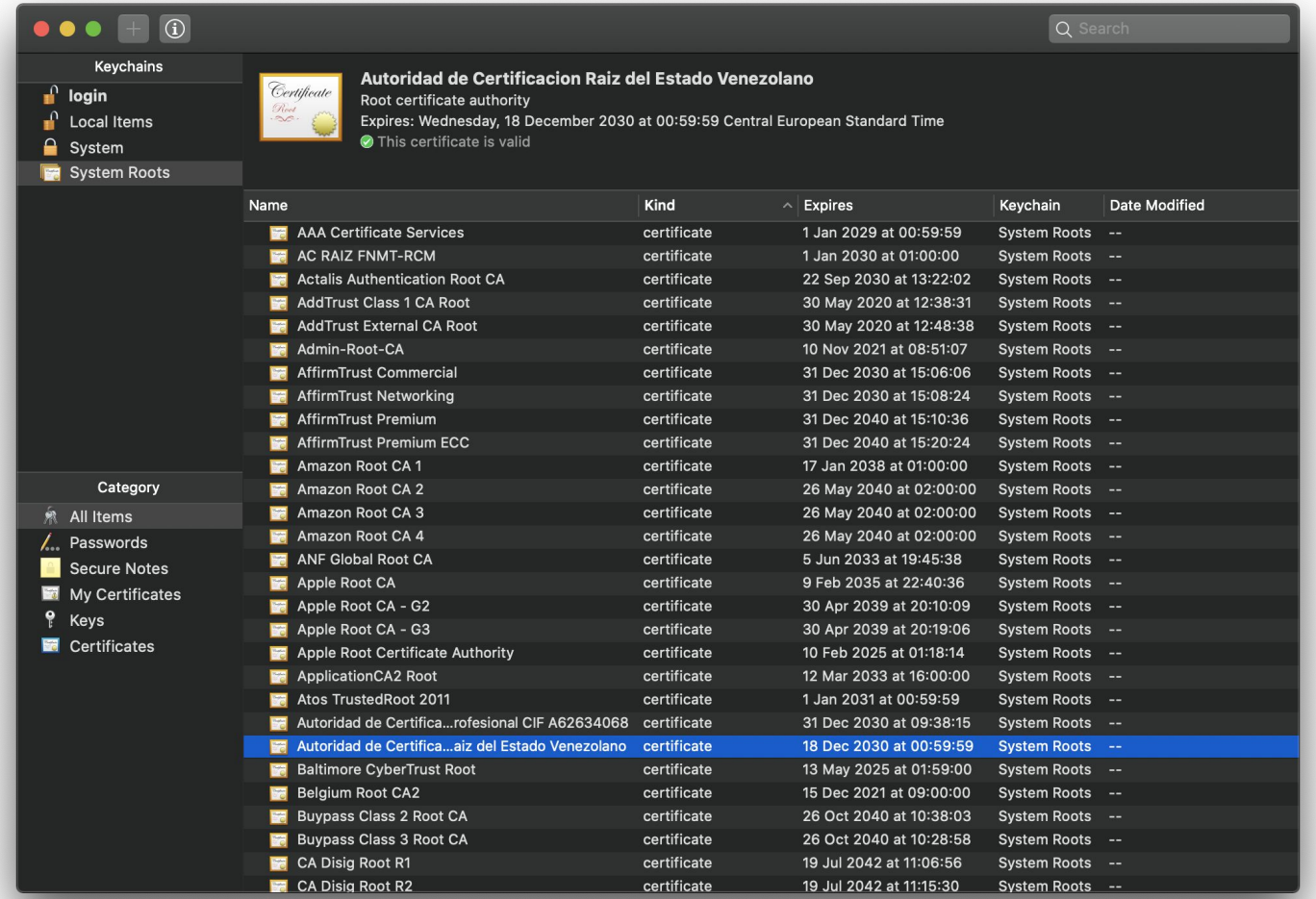
- Protects transmissions
  - Confidentiality
  - Integrity
- Ensures authentication
  - Of server
  - Of client (optionally)

- No protection before or after transmission
  - On server
  - On client (e.g., trojan)
  - By abuser (e.g., non-honest merchant)
- Relies on PKI
- Not foolproof

TLS cons: reliance on PKI

- Security guarantees of TLS depend on the security and trust of the root and intermediate CAs
- CAs can generate certs for any domain
  - They are responsible for domain\org validation
- In order to be included in browsers and OS root programs, CAs must abide to a set of requirements (CA/Browser Forum baseline requirements)
  - Dropping a non-compliant CA is a very difficult decision as it breaks websites

Reminder: CAs your OS trusts



Corollary: you need to trust the list of CA pre-installed in your computer.
It is not always so: see the 2015 Lenovo Superfish incident…

Pitfalls: A few CA Incidents

- **2011**: Diginotar as well as some Comodo resellers are compromised → rogue certs (at least 500 for Diginotar)
  - Diginotar is distrusted on all major platforms
- **2009 - 2015**: Symantec → test certificates for existing domains (e.g., Google)
  - Caught through CT logs
  - Outcome: CA gradually distrusted (2016-2018)
- **2012**: Trustwave issues a MITM certificate for a data loss prevention appliance
- **2012**: TurkTrust mistakenly gives two CA certificates to users
- **2016**: WoSign / StartSSL → various issues, including cert mis-issuance due to vulnerability in the domain verification.
  - Outcome: CA distrusted

Overcoming TLS/PKI limitations

HSTS (HTTP Strict Transport Security)

- HTTP header to tell the browser to always connect to that domain using HTTPS
- Browsers (e.g., Chrome) implement a HSTS preload list: some websites are simply never loaded over plain HTTP
- Defends against SSL stripping

Overcoming TLS/PKI limitations

HPKP (HTTP Public Key Pinning)

- HTTP header to tell the browser to "pin" a specific certificate or a specific CA
- Browser will refuse certificates from different CAs for that origin
- Defends against trust cert mis-issuance
- **Deprecated!** Can you think about why?

EXTREMELY HARD TO MANAGE (I.E., WEBSITE CHANGE PIN ⇓ UPDATE EVERYTHING)

Overcoming TLS/PKI limitations

Certificate Transparency

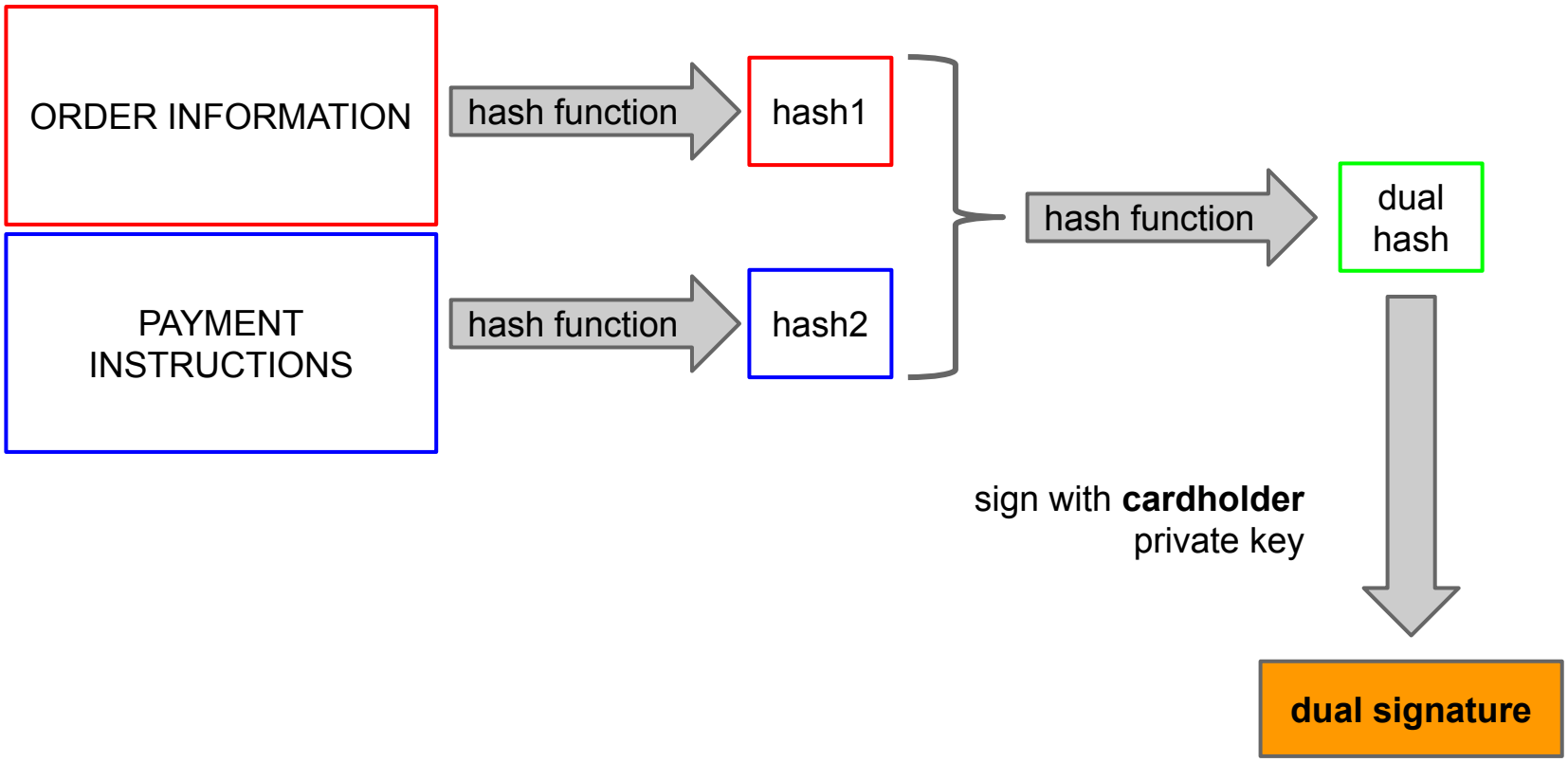
- CA submit the metadata of every issued certificate to a (independent, replicated) log
- Can be enforced by browsers
  - browsers refuse certs not logged in CA logs
- Defends against certificate mis-issuance: site owner can check \ be notified of certificates issued for the properties they manage
- You can look at the CT logs: https://crt.sh

Introducing SET

- Joint effort VISA+MasterCard consortium
- Protect *transactions*, not connections
- Approach
  - **Cardholder** sends
    - the order of goods to the **merchant** only
    - the payment data to the **payment gateway** only
  - Empower gateway to verify correspondence
- Uses the concept of a dual signature
  - A signature that joins together the two pieces of a message, directed to two distinct recipients

DUAL SIGNATURE (SEE NEXT PAGE)

Dual Signature Generation



Data Transmission

ORDER INFORMATION | hash2 | **dual sign.**

encrypt with public key of **merchant**

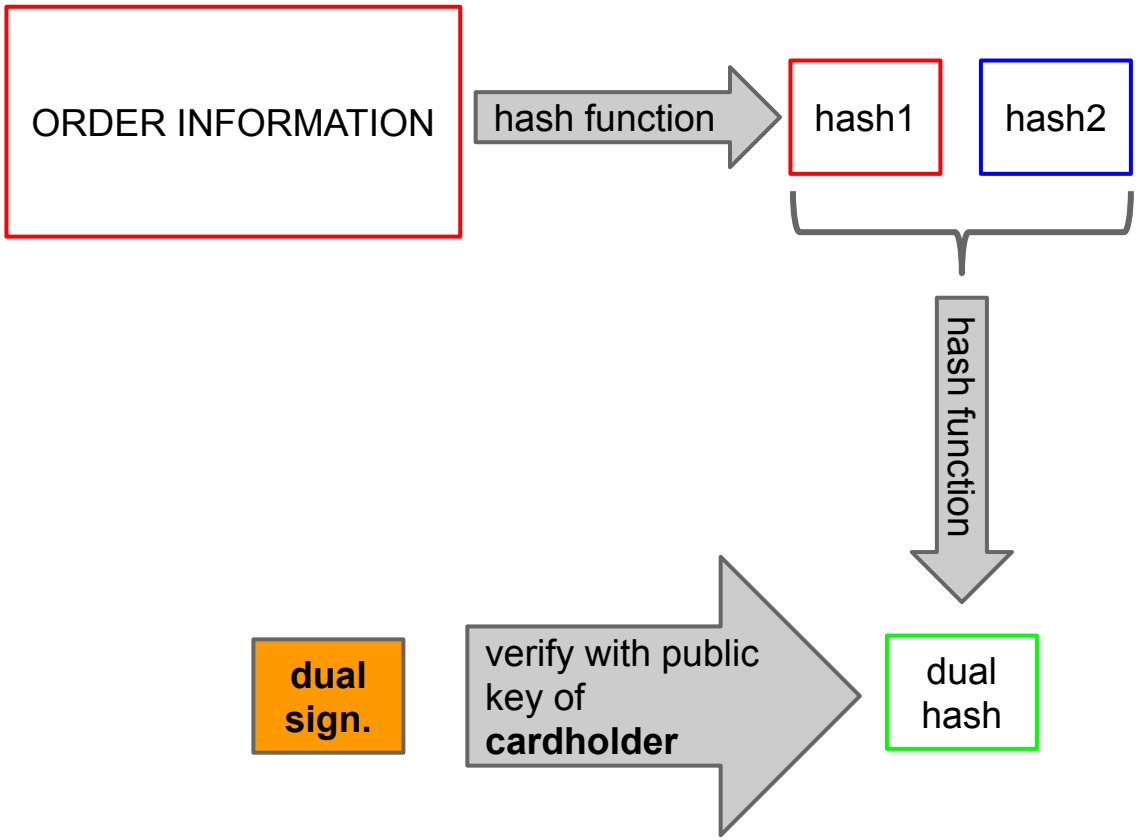
PAYMENT INSTRUCTIONS | hash1 | **dual sign.**

encrypt with public key of **payment gateway**

PAYMENT INSTRUCTIONS | hash1 | **dual sign.**

(still encrypted)

Verification (merchant side)



(the payment gateway side verification is the perfect dual)

Why did SET Fail?

- SET requires a **digital certificate** from:...
  - Merchant: OK, reasonable and feasible ... EVERYONE
  - Payment Gateway: OK, reasonable and feasible
  - **Cardholder: KO, does not scale!**
- Therefore, a pre-registration of the cardholder is needed! (won't buy that book)

- Non-transparent = less critical mass = failure
- Nowadays a simple redirect with a token to the website of the bank is commonly used
  - **Exercise:** think how this is implemented securely ;-)