



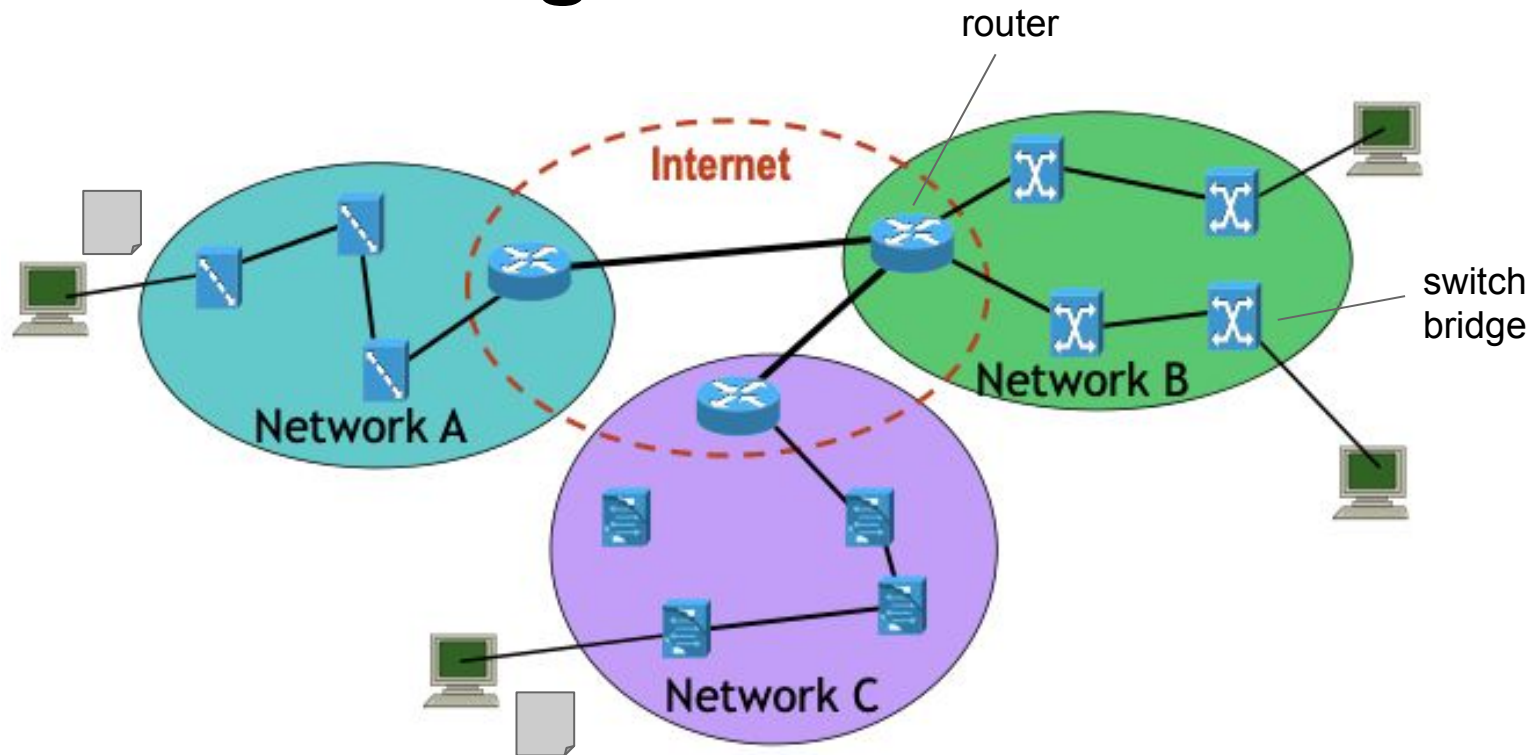
9. Network Protocol Attacks



Computer Security Courses @ POLIMI

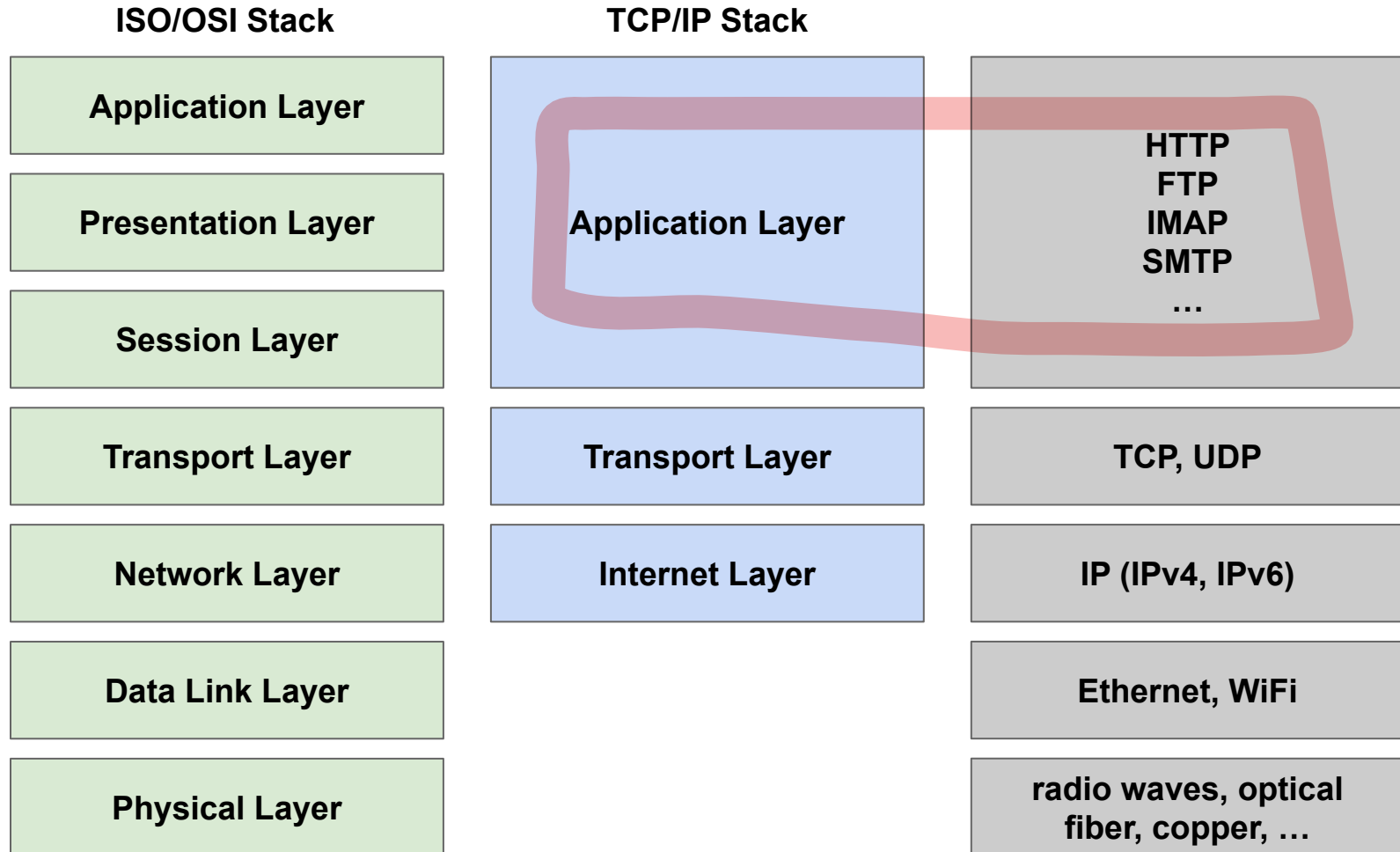


Internetworking

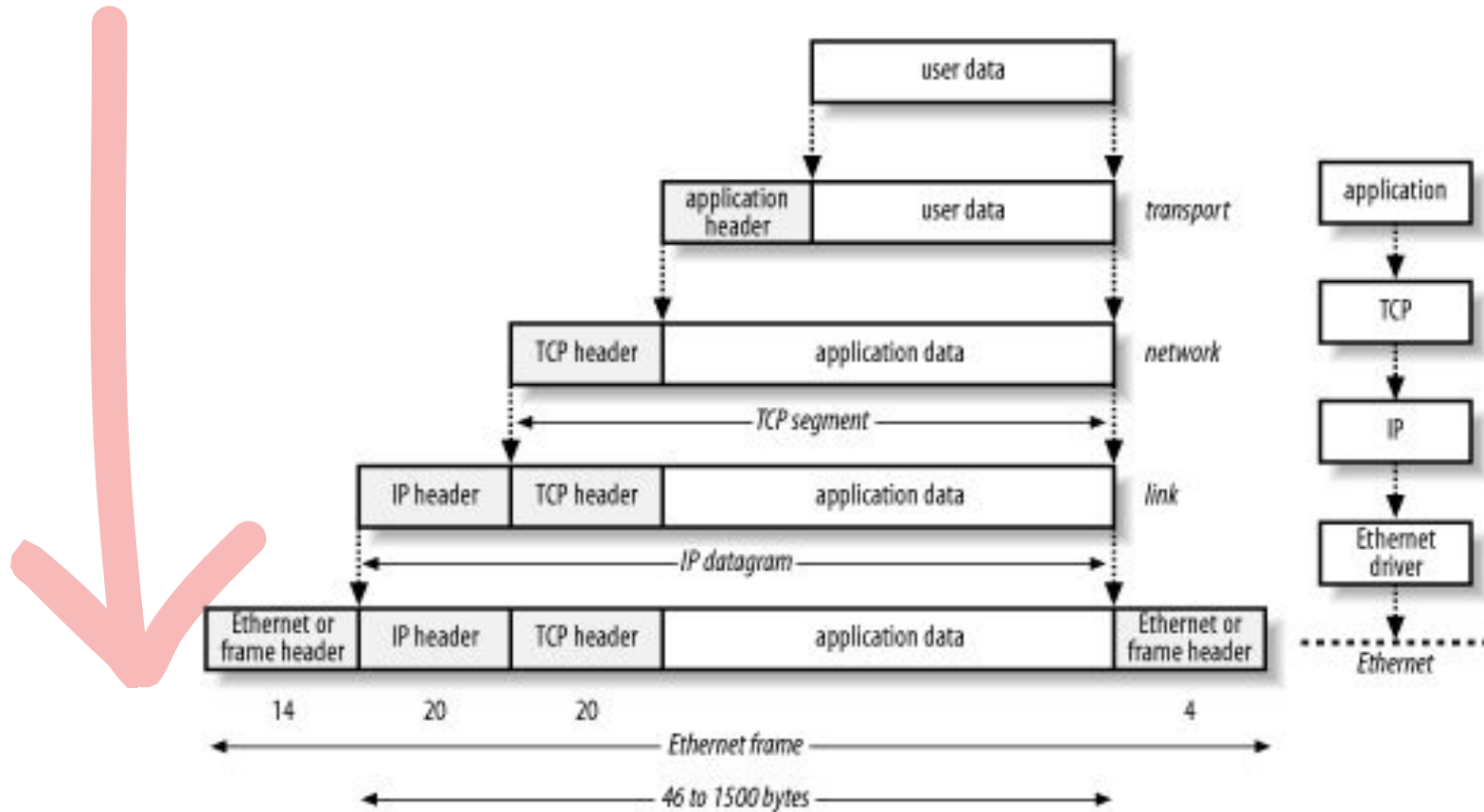


- Networks = different physical media, topologies, ...
- Need a *layered* approach

Layering and Protocols



Packet Encapsulation

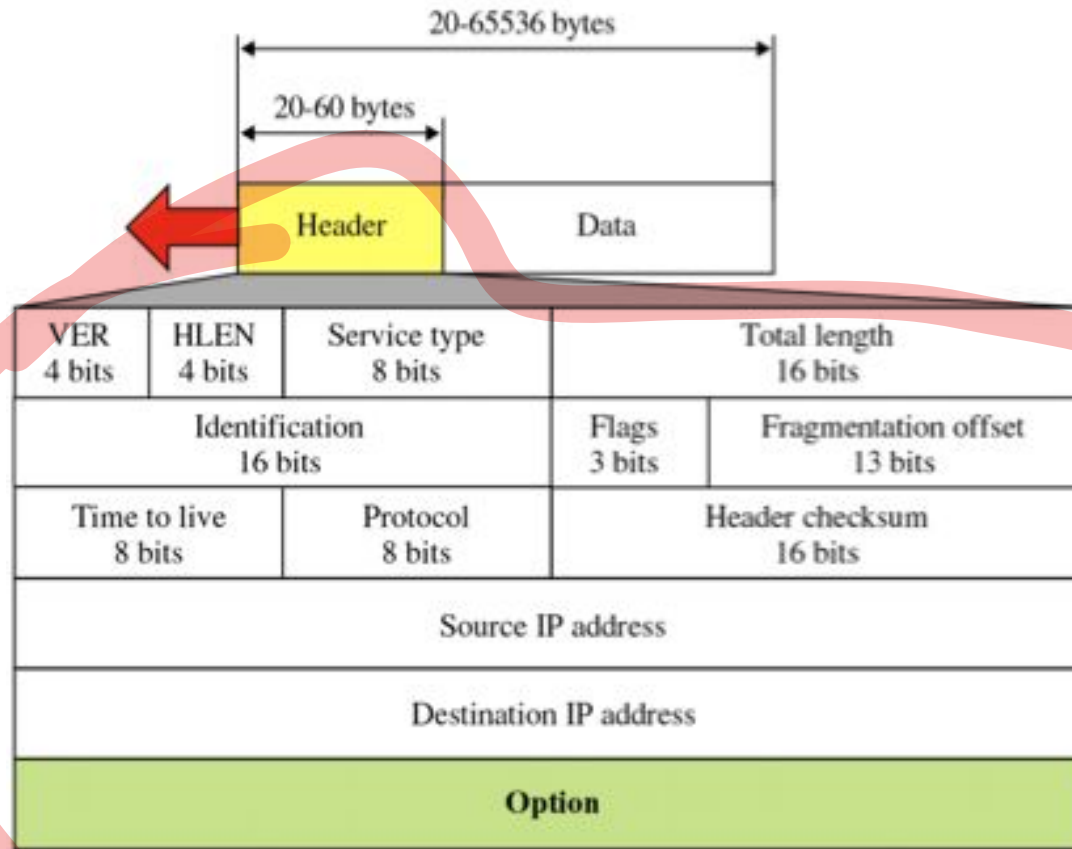


Addressing

- 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



Example: IP Datagram

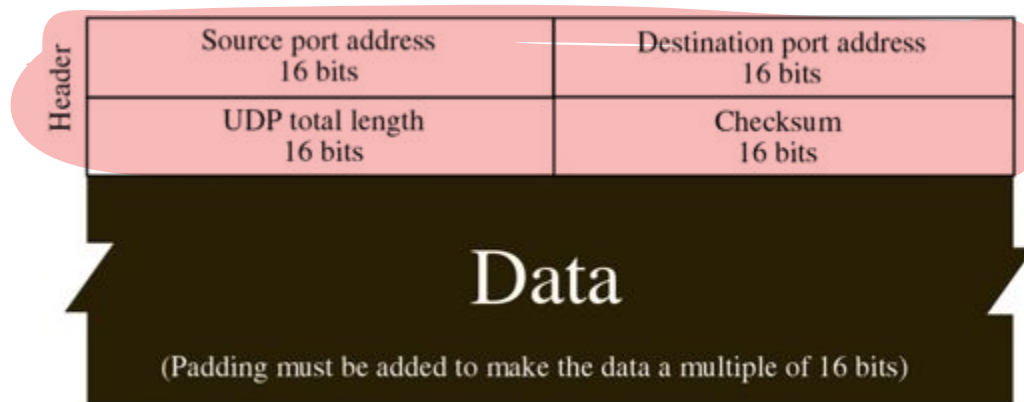


Transport Protocols



UDP

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

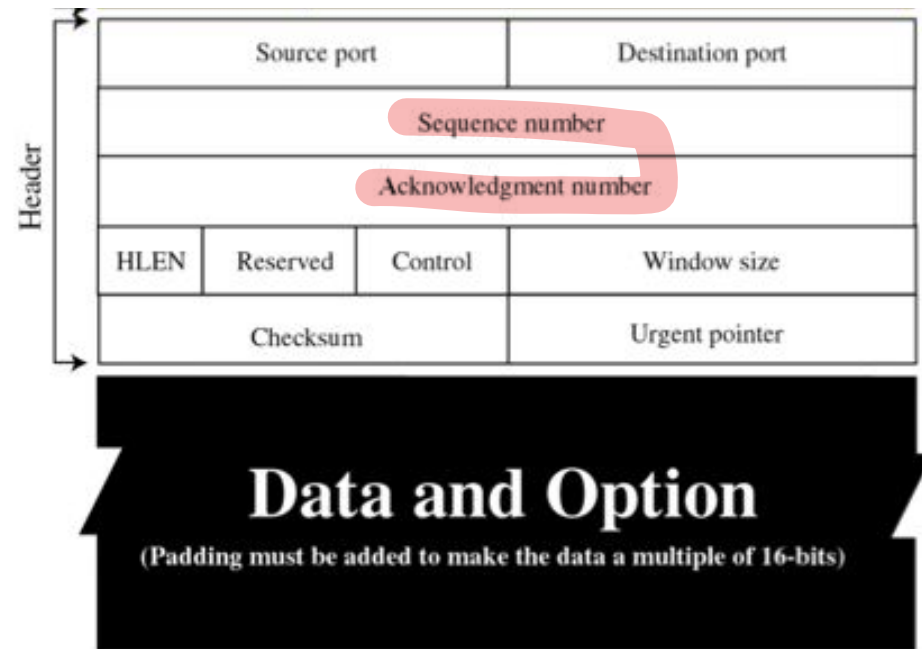


Transport Protocols



TCP

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



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

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

Examples of Denial of Service



- Killer Packets
- SYN flood
- Smurf, multiplication or amplification attacks
- Distributed DoS

Killer Packets (1): Ping of Death

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)"

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

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

Killer Packets (2): Teardrop

Exploit vulnerabilities in the TCP reassembly.

Fragmented packets with overlapping offsets.

While reassembling, kernel can hang/crash.

- 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

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

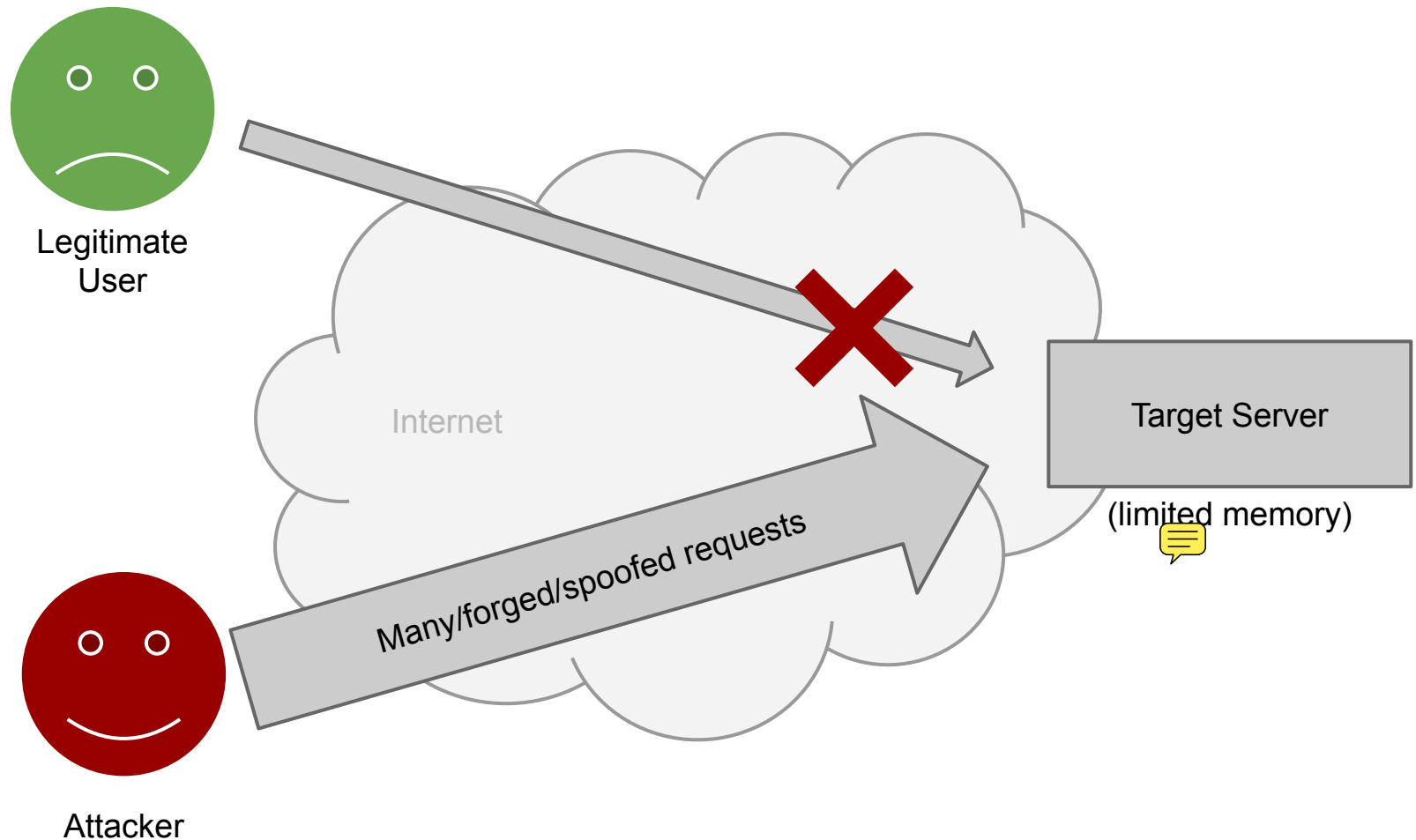
- src IP == dst IP
- SYN flag set

could loop and lock up a TCP/IP stack.

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>

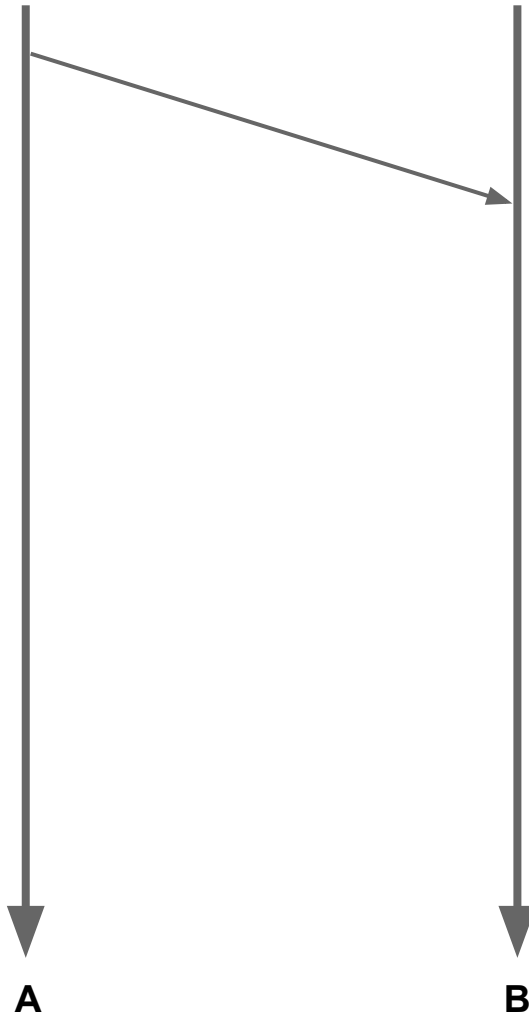
Denial of Service via Flooding



SYN Flood Attack: The TCP/IP Three Way Handshake

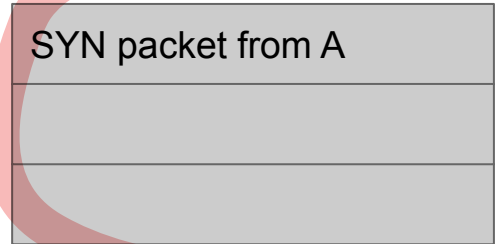


SYN packet
IP src = A
IP dst = B



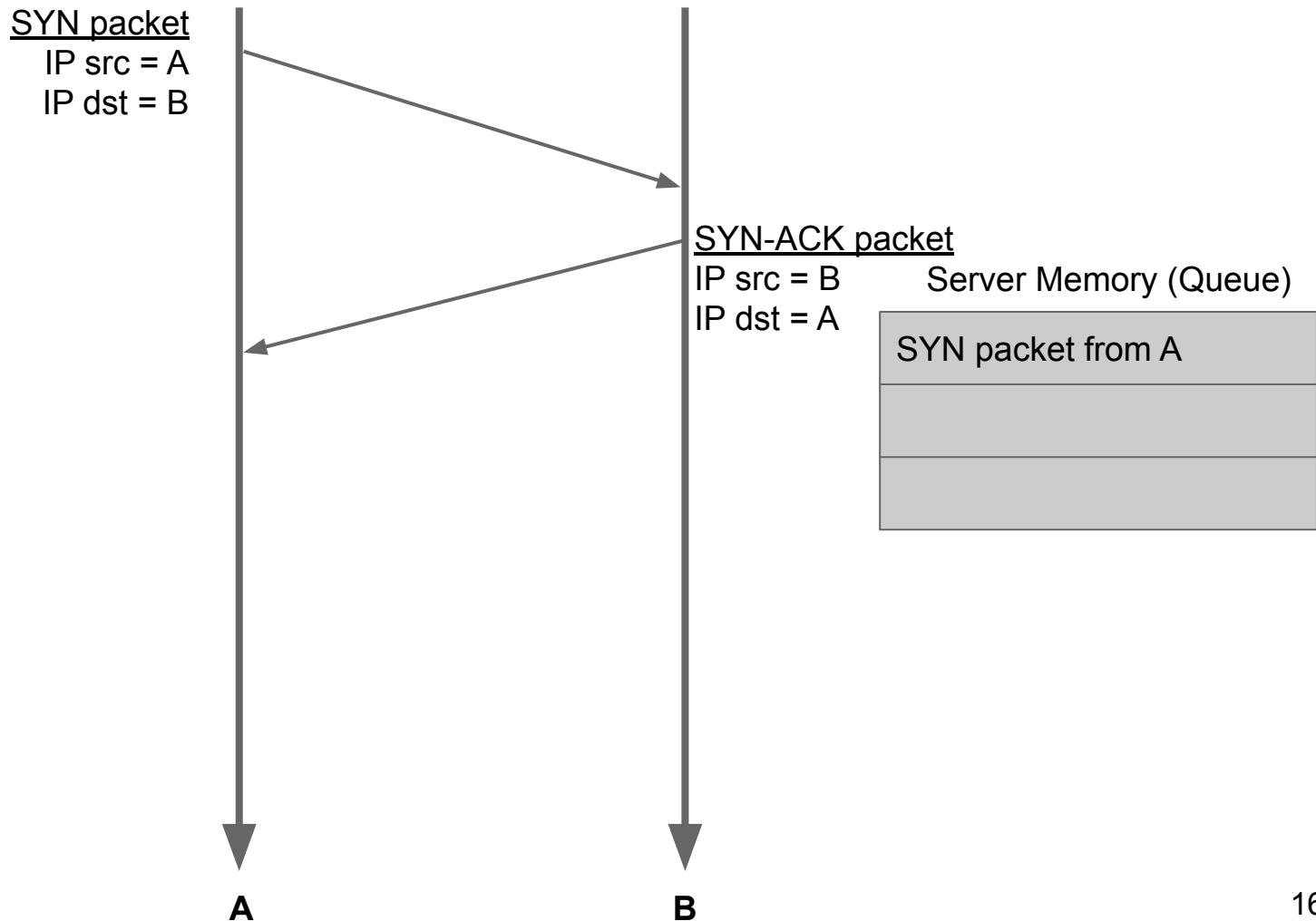
Server Memory (Queue)

SYN packet from A



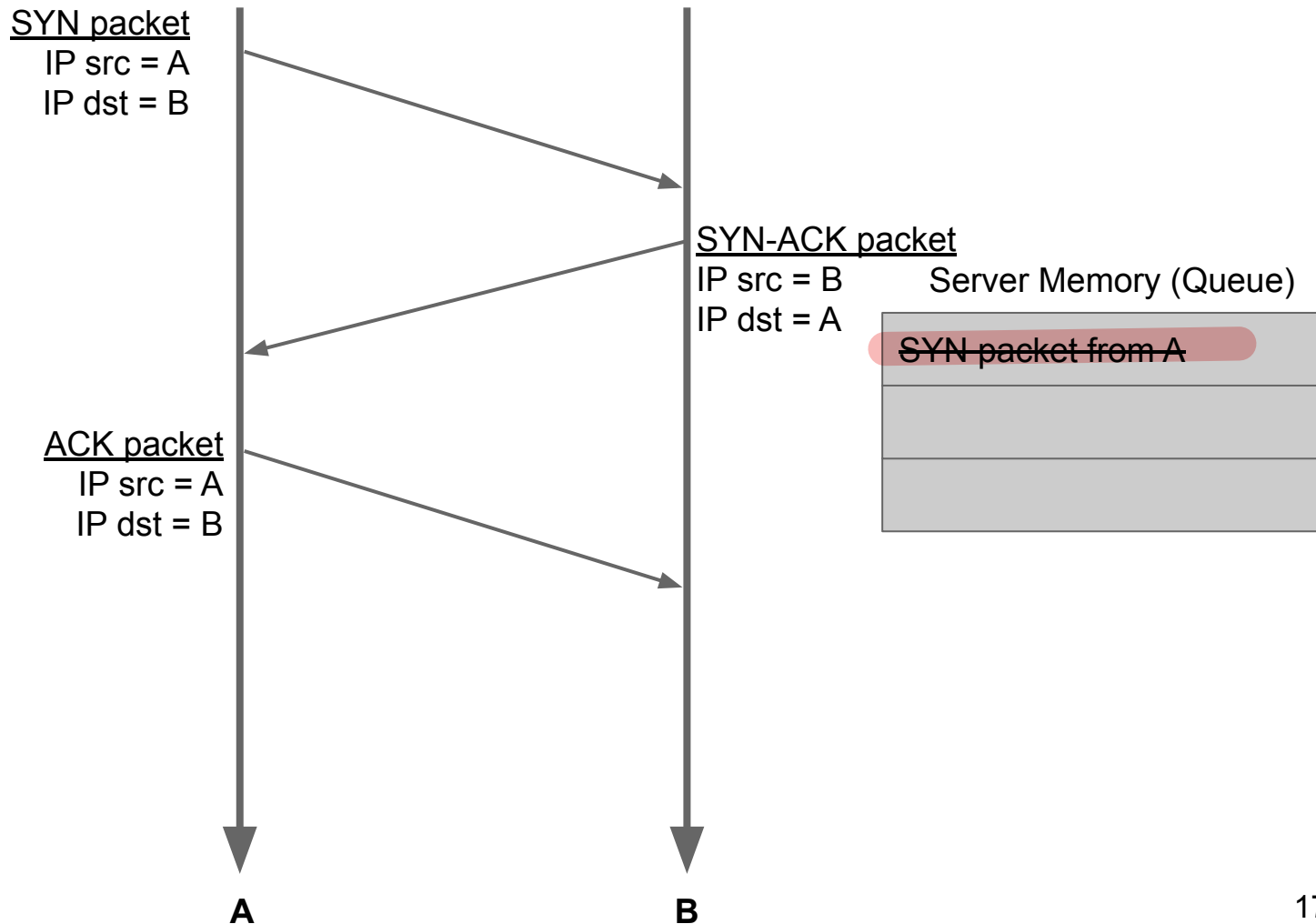
RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake



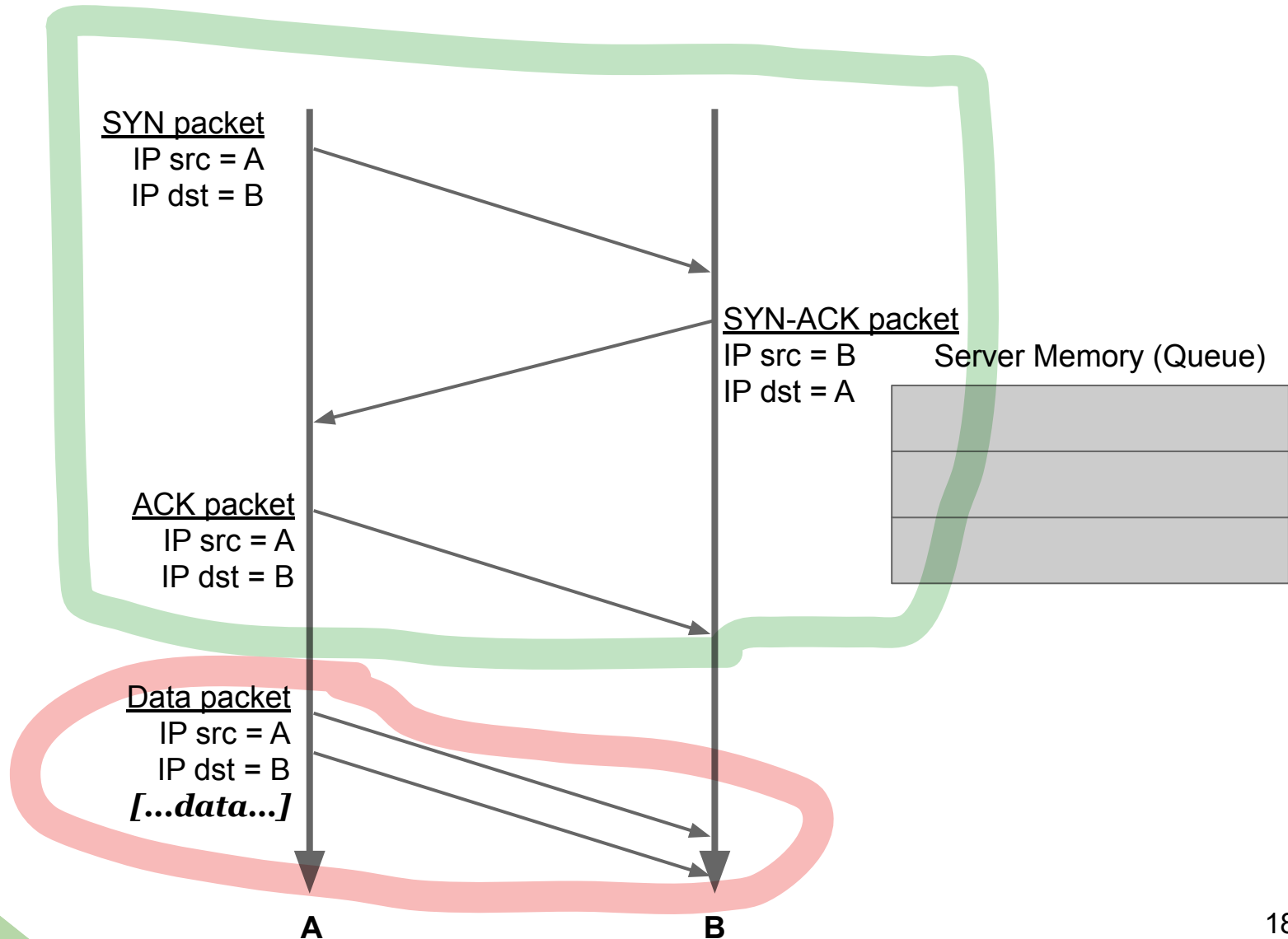
RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake

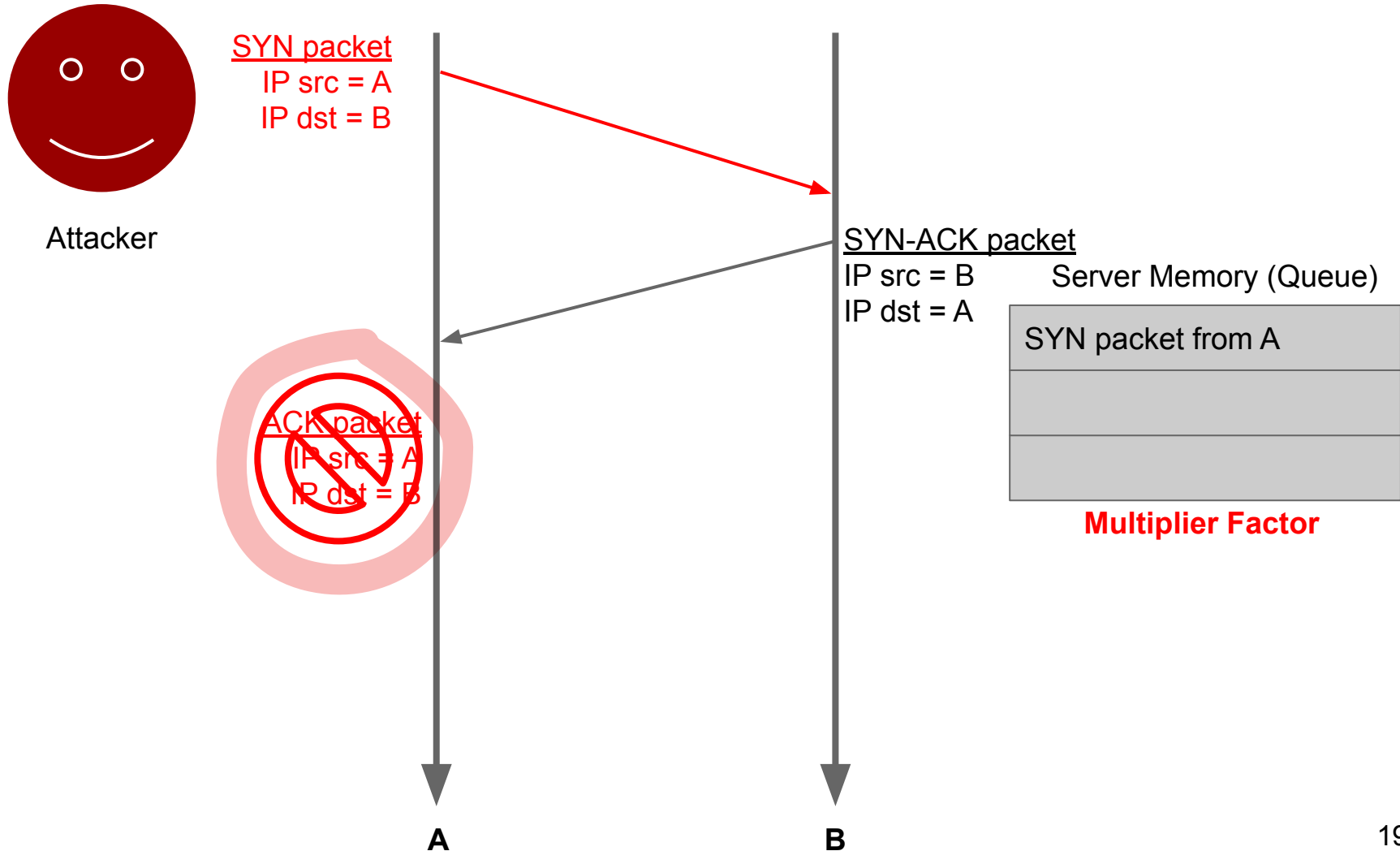


RECALL

SYN Flood Attack: The TCP/IP Three Way Handshake



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

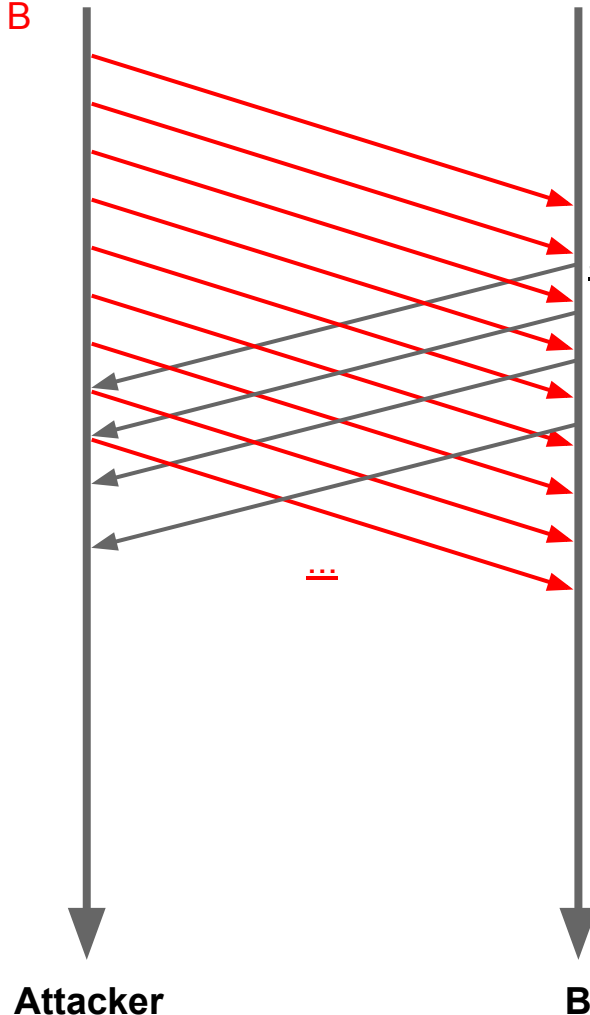
SYN packet
IP src = A
IP dst = B

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

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

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

...



SYN-ACK packet

IP src = B
IP dst = A

Server Memory (Queue)

SYN packet from A

SYN packet from A'

SYN packet from A''

...

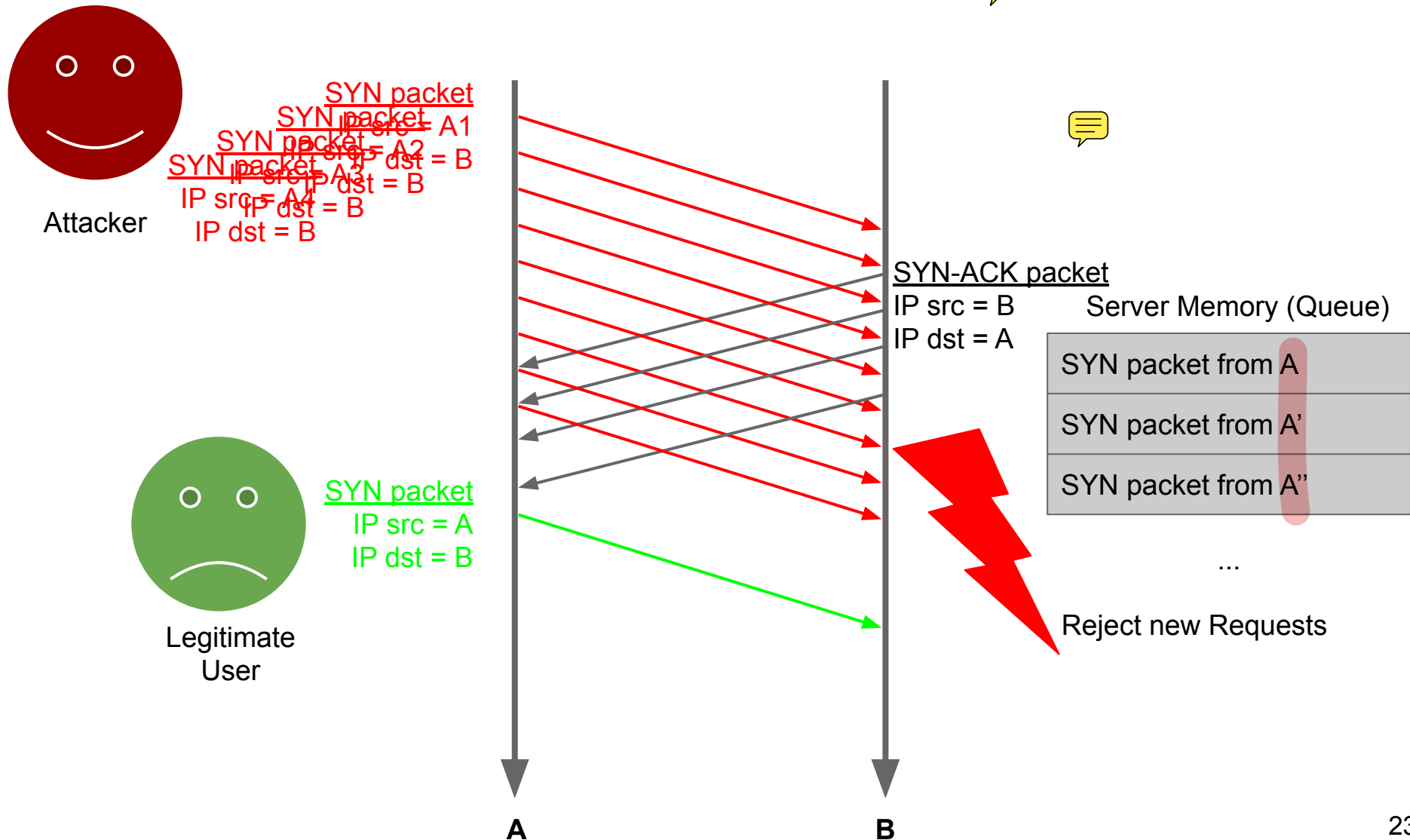
SYN Flood Attacks

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**.

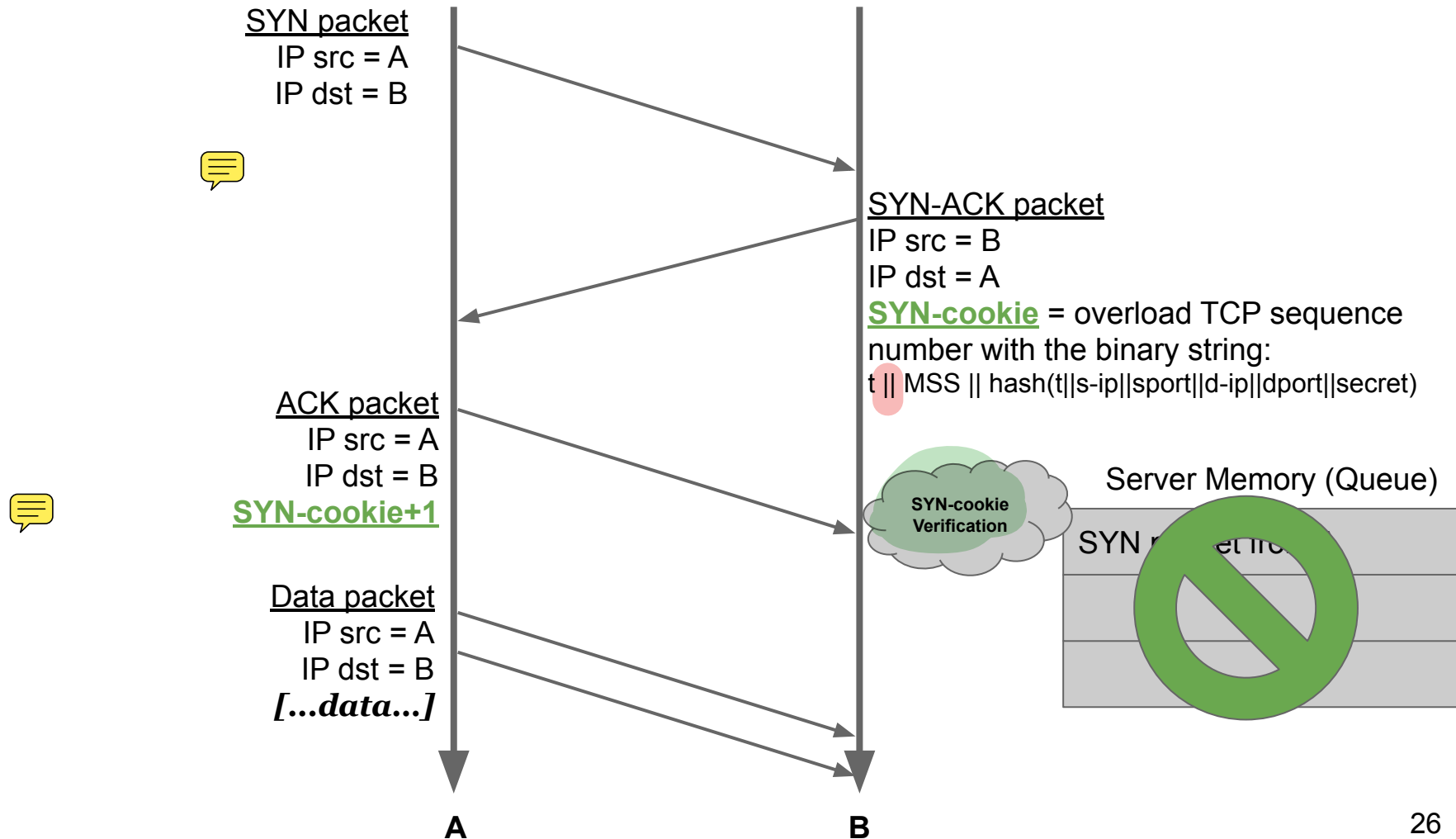
Many half-open TCP/IP connections fill the queue.

SYN requests from legitimate clients dropped.

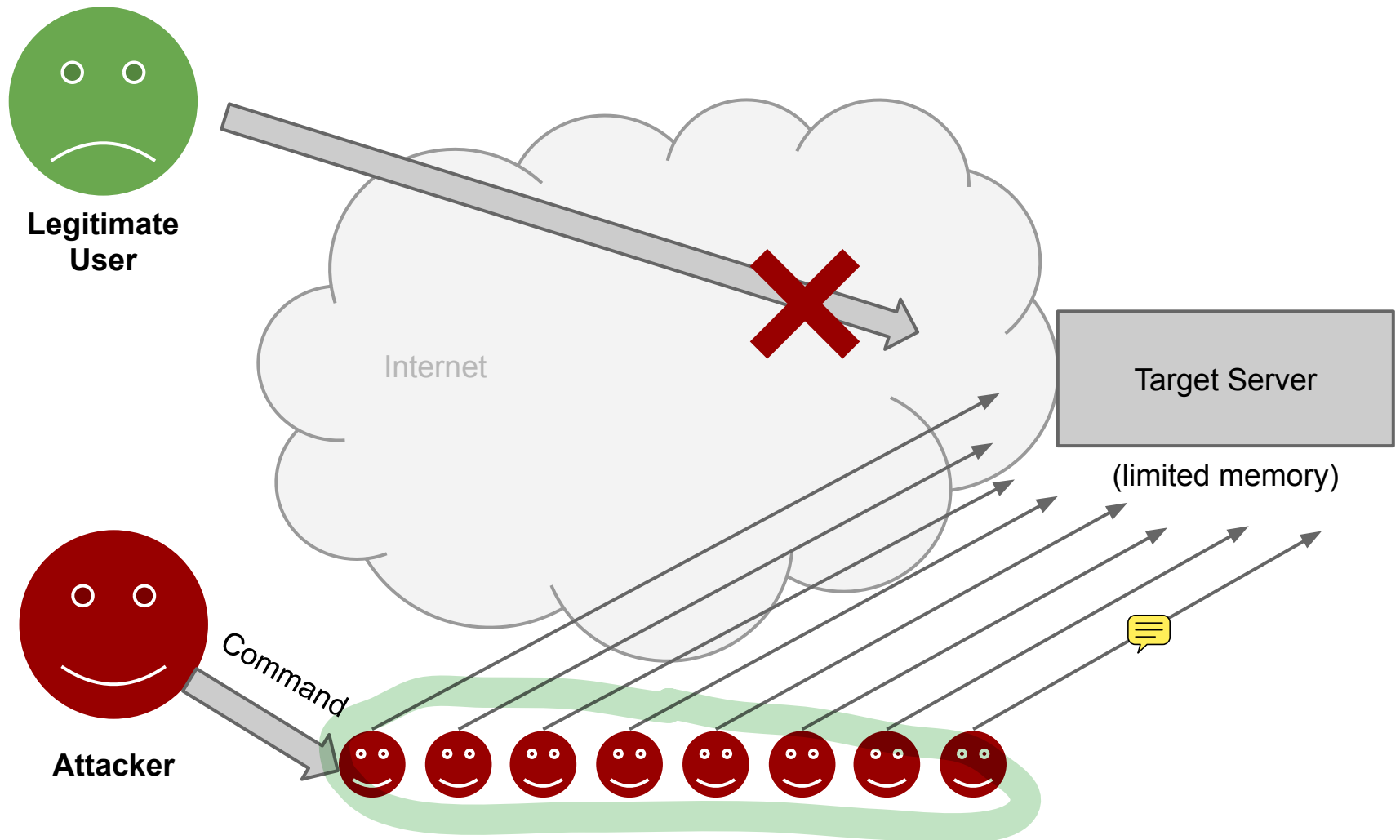


Mitigation: SYN-cookies avoid this: reply with SYN+ACK but discard the half-open connection, and wait for a subsequent ACK <http://cr.yp.to/syncookies.html>

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*.

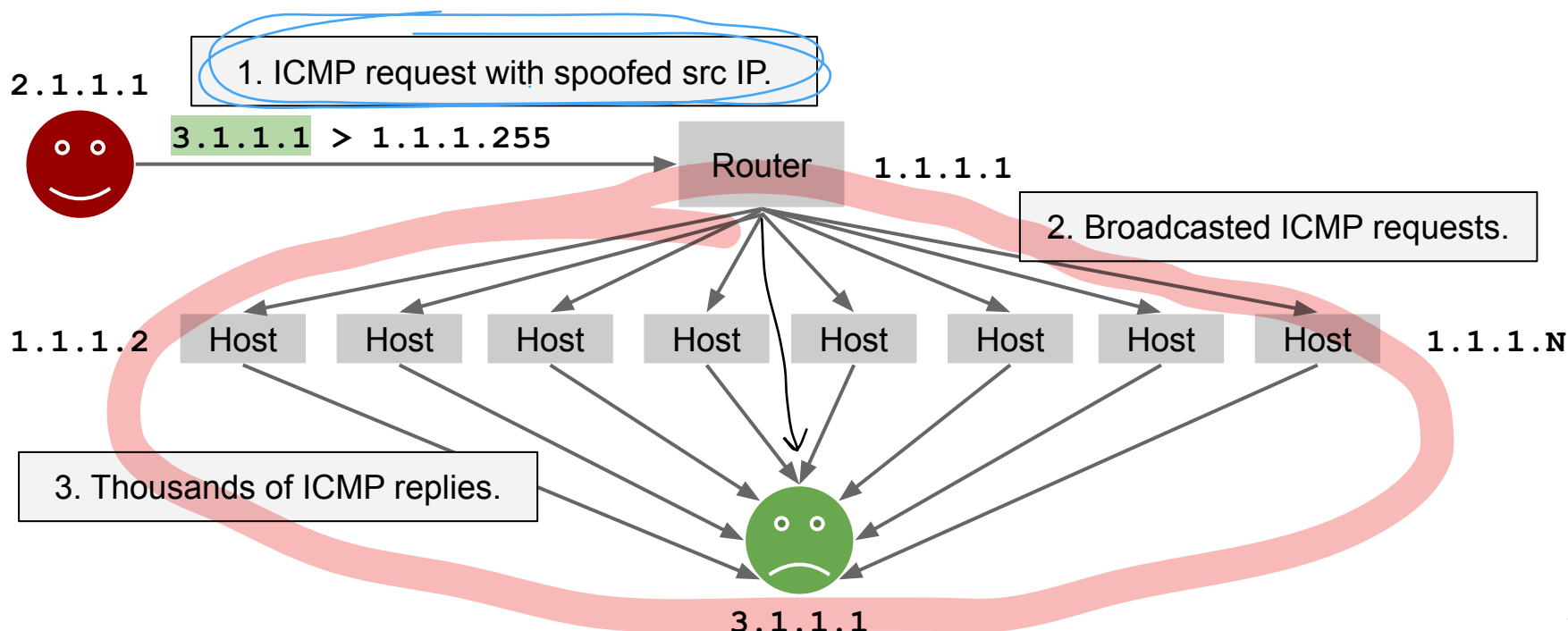
Various uses (e.g., spamming, phishing, info stealing), including DDoS-ing.



Distributed DoS (DDoS): Smurf

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	<i>all</i>	BAF		PAF	<i>all</i>	Scenario
		50%	10%			
SNMP v2	6.3	8.6	11.3	1.00		<i>GetBulk</i> 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		<i>SEARCH</i> 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
Salinity	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

Promiscuous mode: the NIC passess to the OS any packet read off of the wire

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



ARP Spoofing (& Cache Poisoning)



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

- ARP request `"where is 192.168.0.1?"`
- ARP reply `"192.168.0.1 is at b4:e9:b0:c9:81:03"`

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

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

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
```

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



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

```
[d3v11z@host]# ./arp spoof 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
```

...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?

...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!



```
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.



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**).

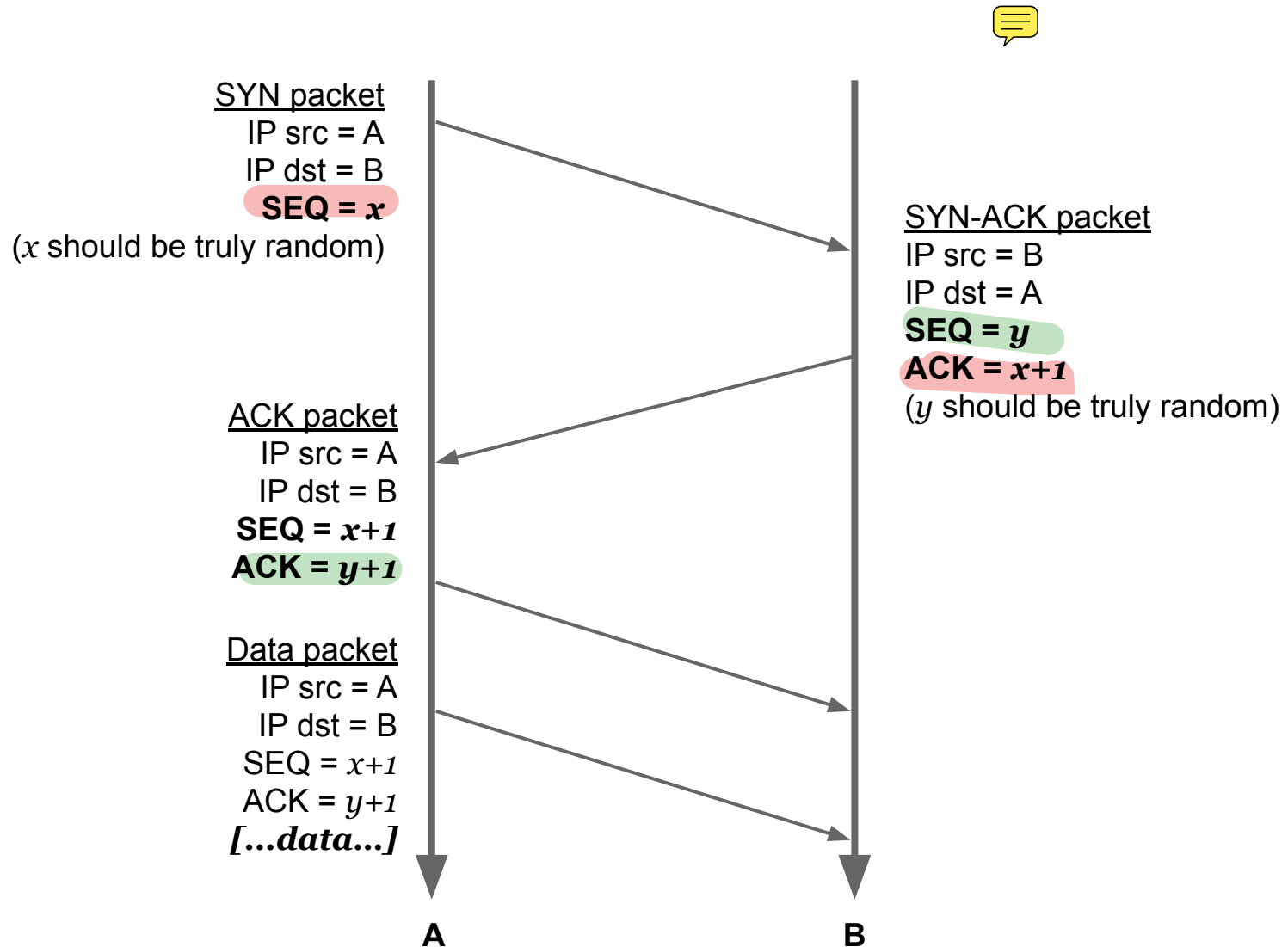
But if the attacker is on the same network, s(he) can **sniff the rest**, or use **ARP spoofing**.

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.

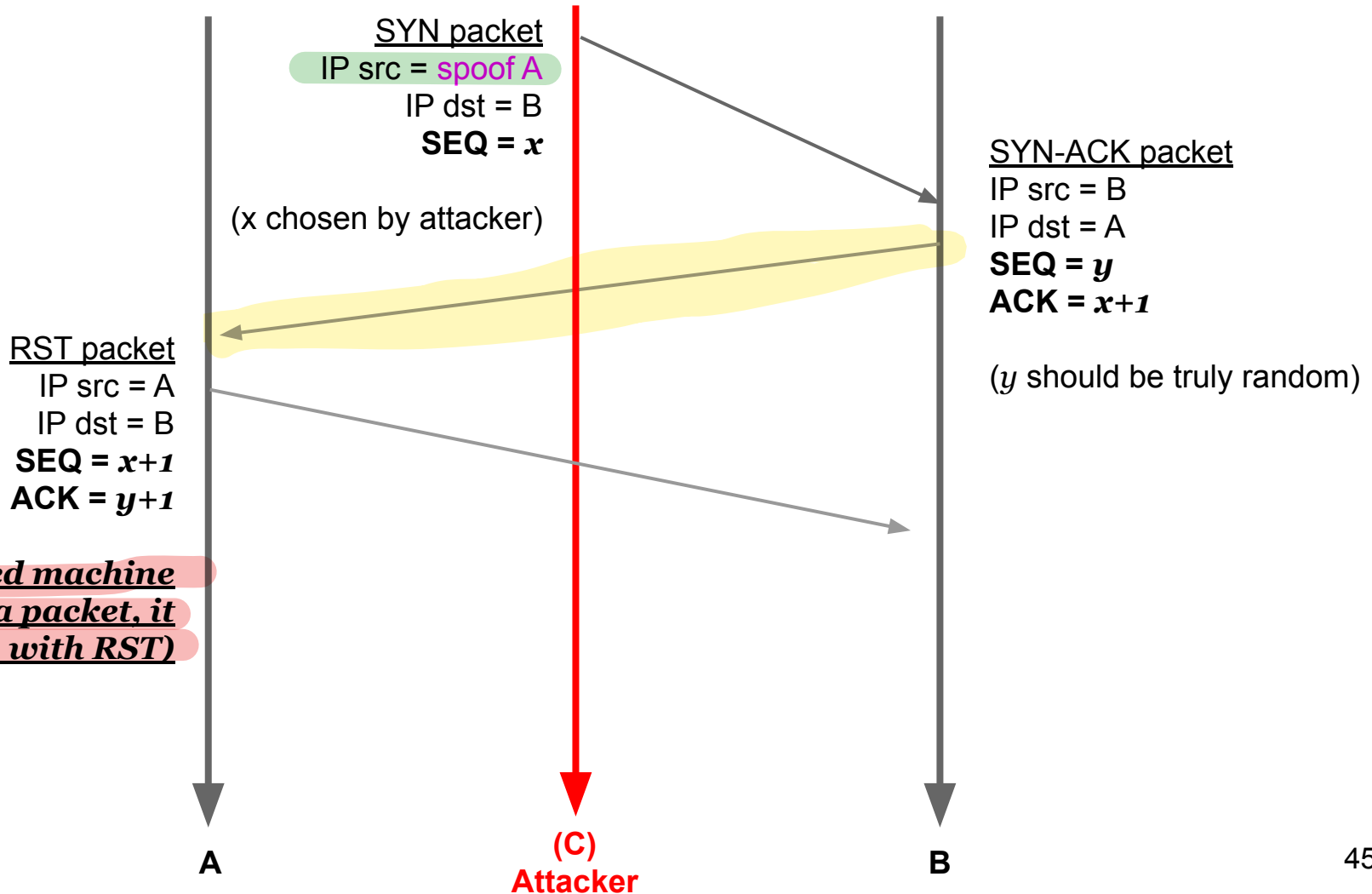
Recall the Three Way Handshake



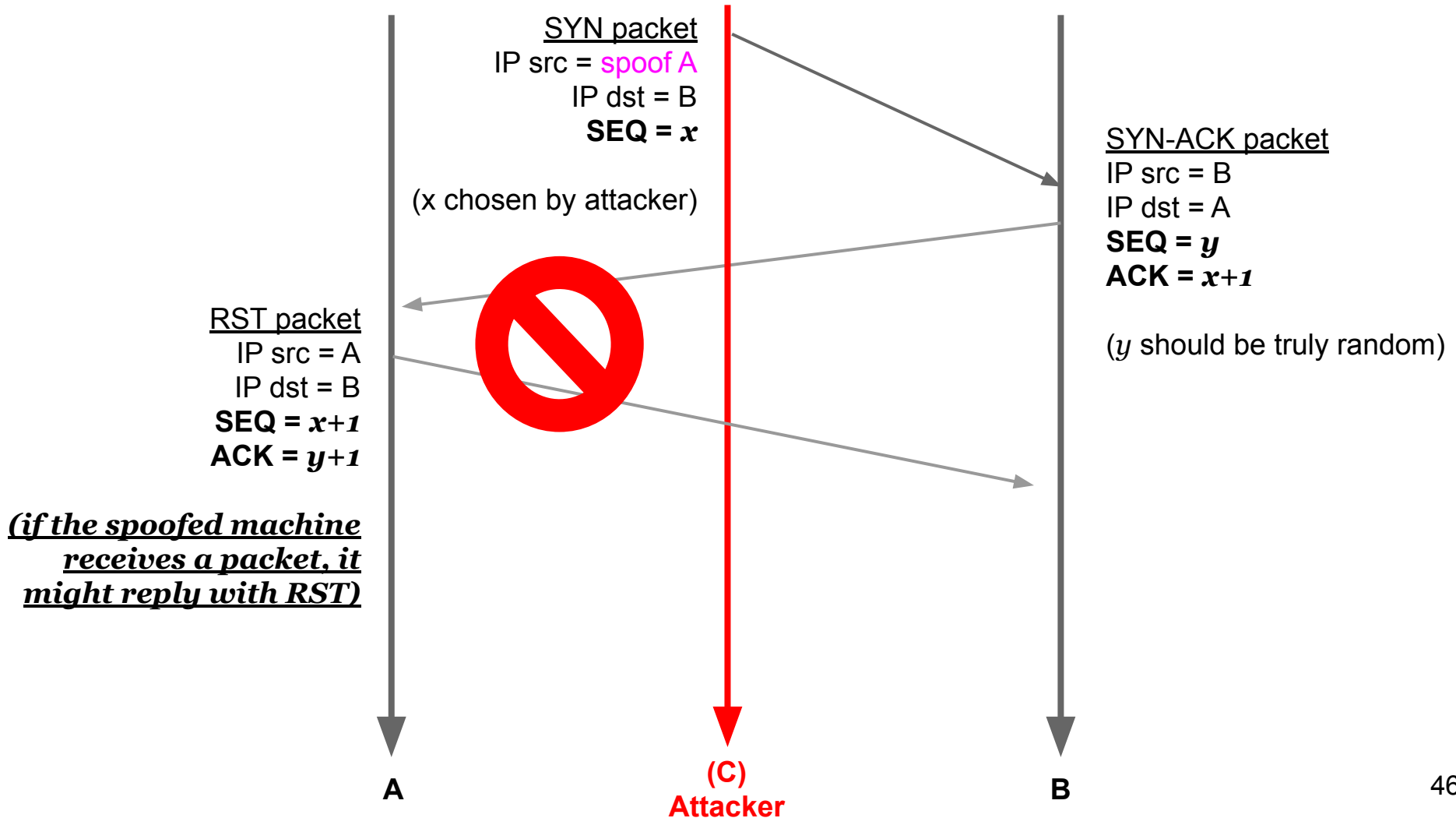
TCP Sequence Number Guessing

- 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.

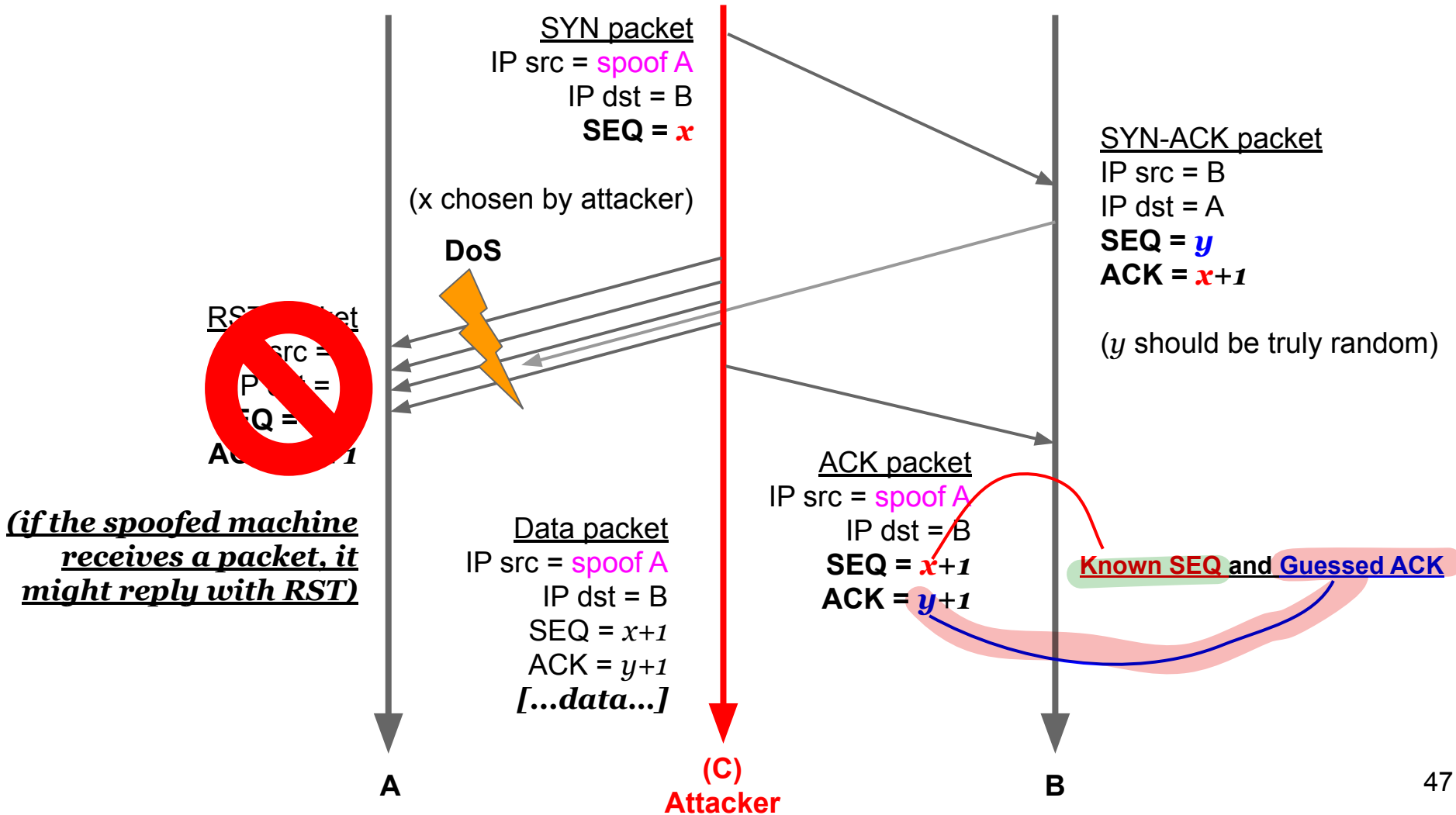
TCP/IP (Blind) Spoofing Attack: The RST packet Issue



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



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



How Random is Random?

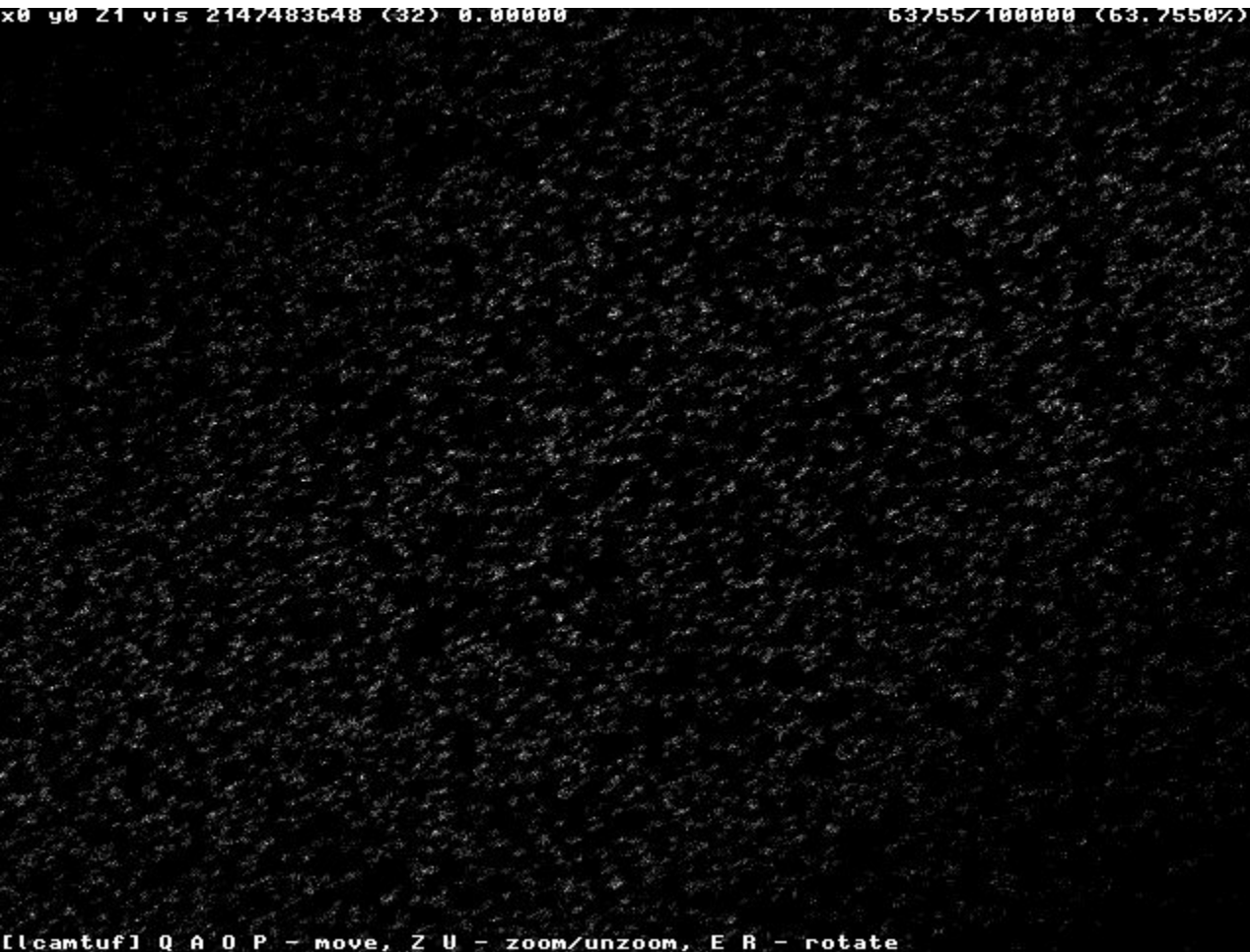
In 1995 Kevin Mitnick used a TCP/IP spoofing attack as the first step to break into Tsutomu Shimomura's machine.

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.

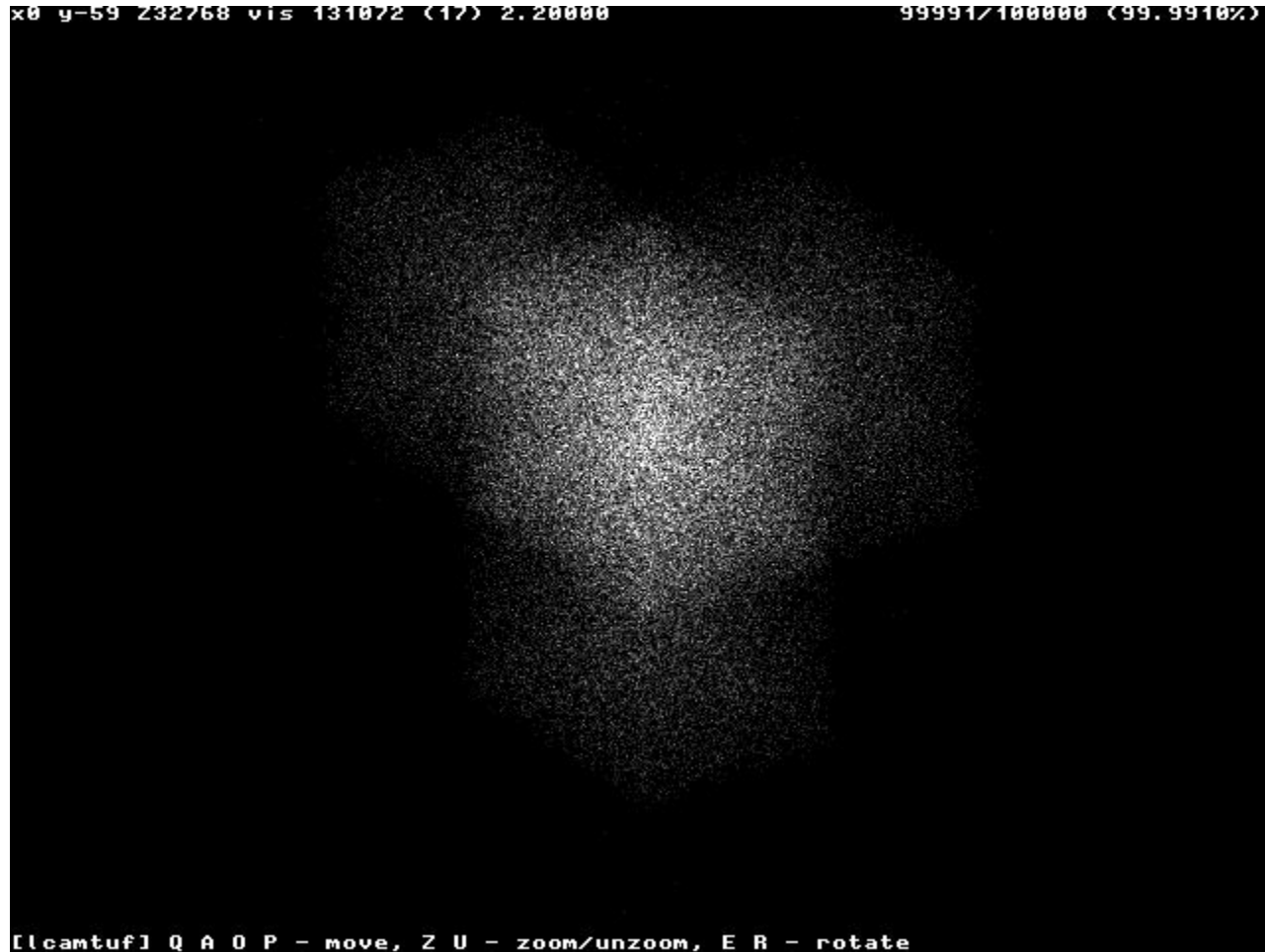
What changed since 1994?

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

Netware 6 SP3 ISN Distribution



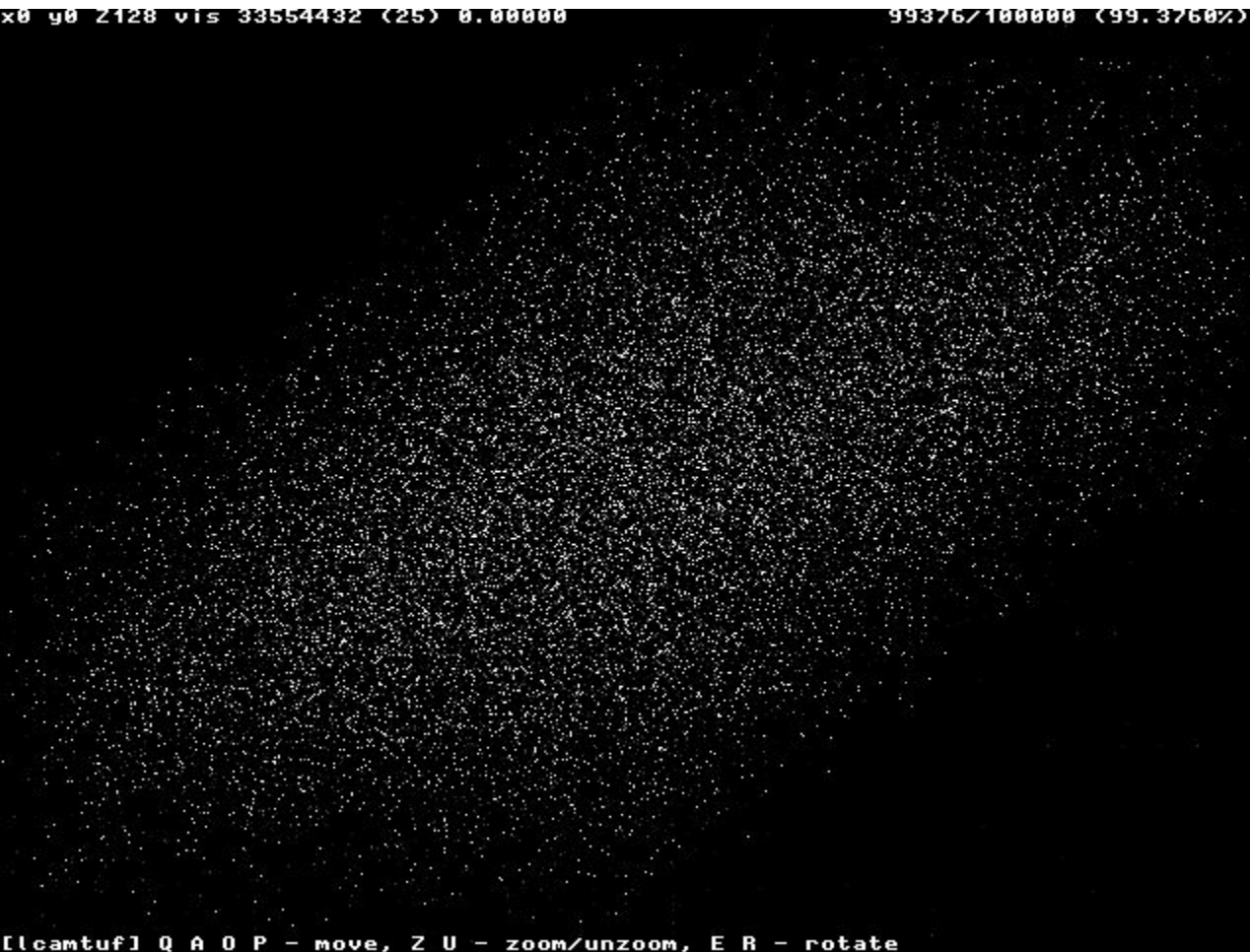
Windows XP SP2 ISN Distribution



IRIX 6.5.15 ISN Distribution



Netware 6 ISN Distribution



*BSD family ISN Distribution

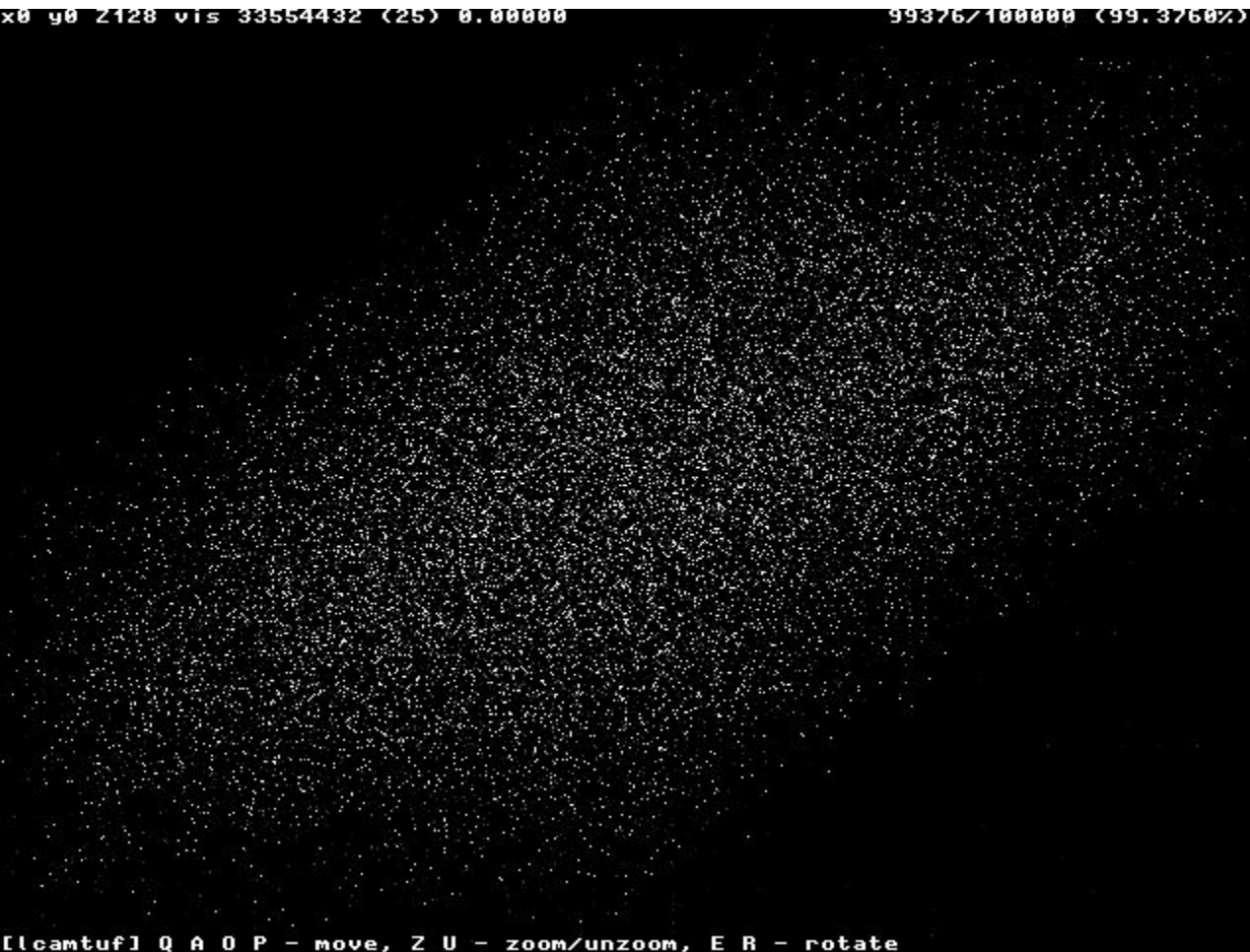


5°: IRIX 6.5.15 ISN Distribution



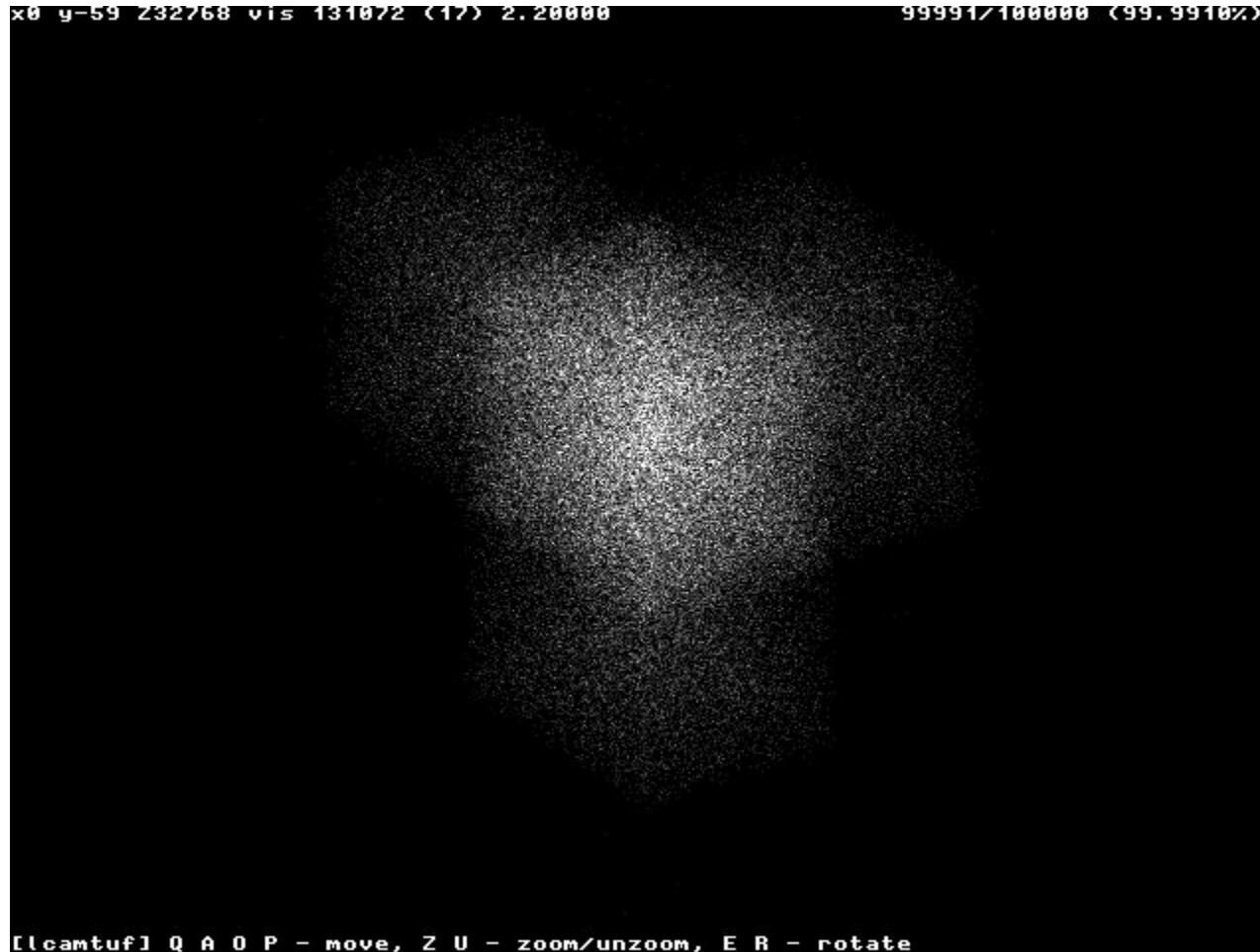
OS Name:	IRIX 6.5.15 tcpiss_md5=0
R1 radius:	0
Average R2:	93
Average N:	297
Average error:	0
Attack feasibility:	100.00%

4°: Netware 6 ISN Distribution



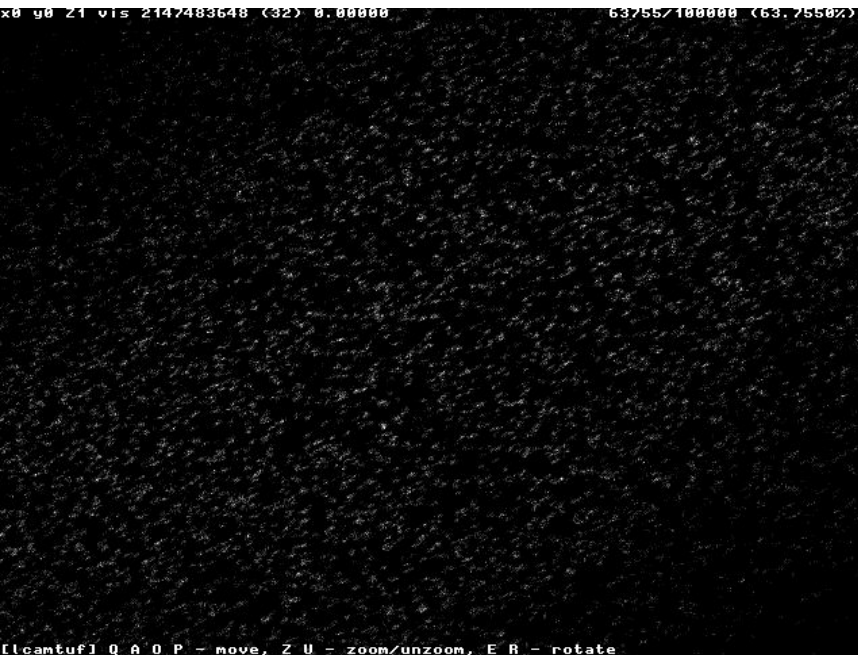
OS Name:	Netware 6
R1 radius:	10
Average R2:	2484
Average N:	11
Average error:	0
Attack feasibility:	90.00%

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%



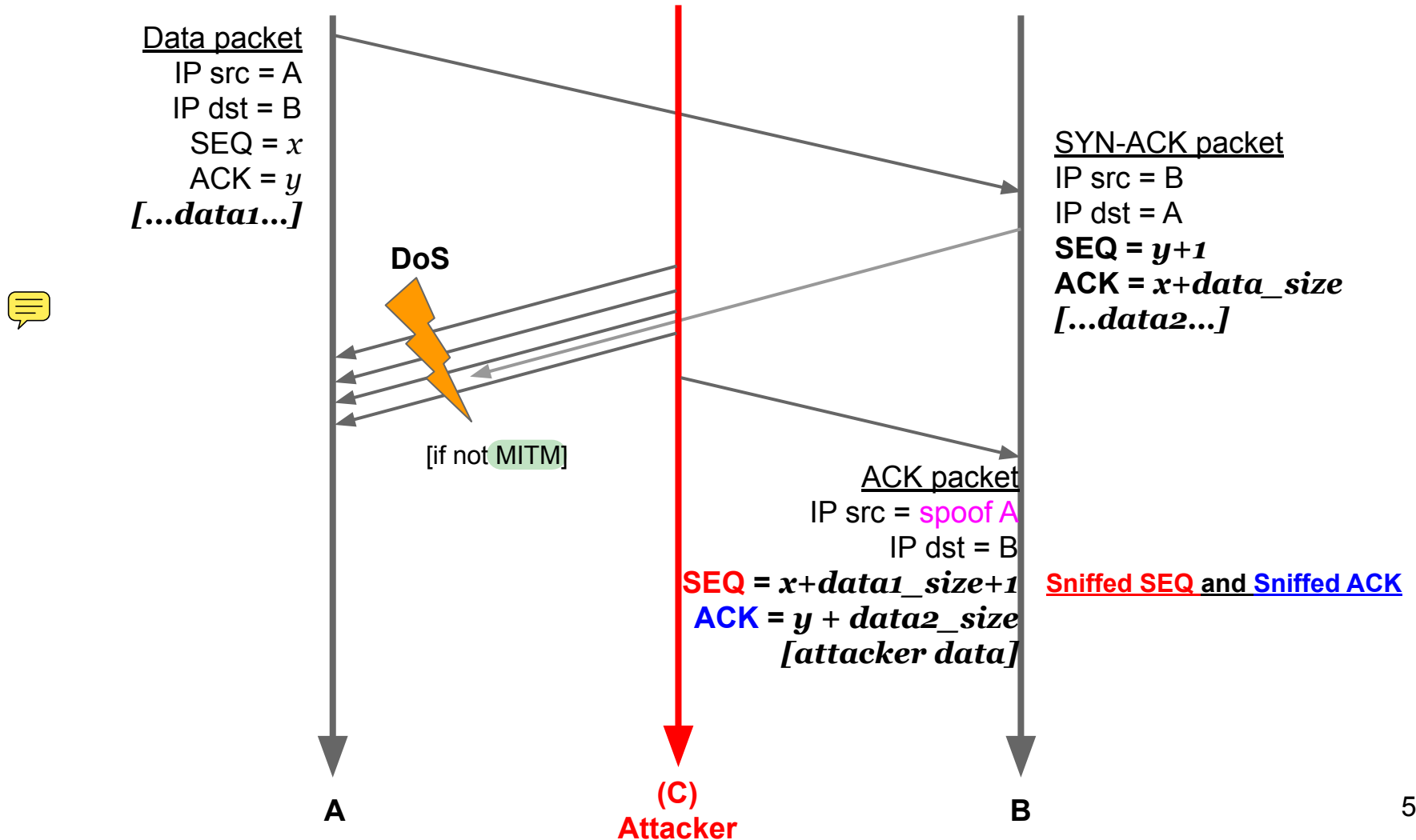
TCP Session Hijacking

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.

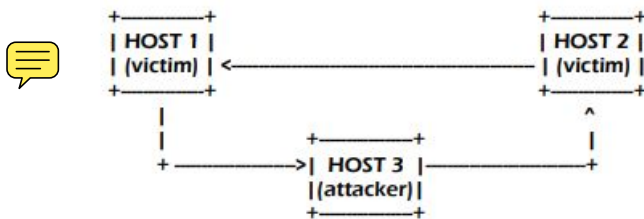
What's a **man in the middle**?

MITM: Man In The Middle

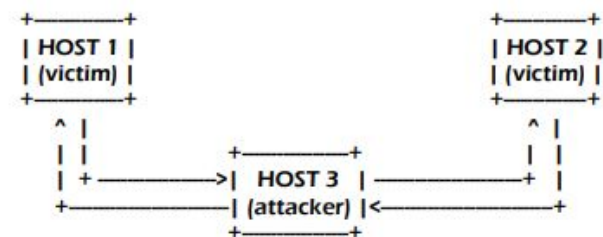
A broad category comprising all the attacks where an attacker can impersonate the server with respect to the client and vice-versa.

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

HALF DUPLEX



FULL DUPLEX



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

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

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

Domain Names System (DNS) (1/2)

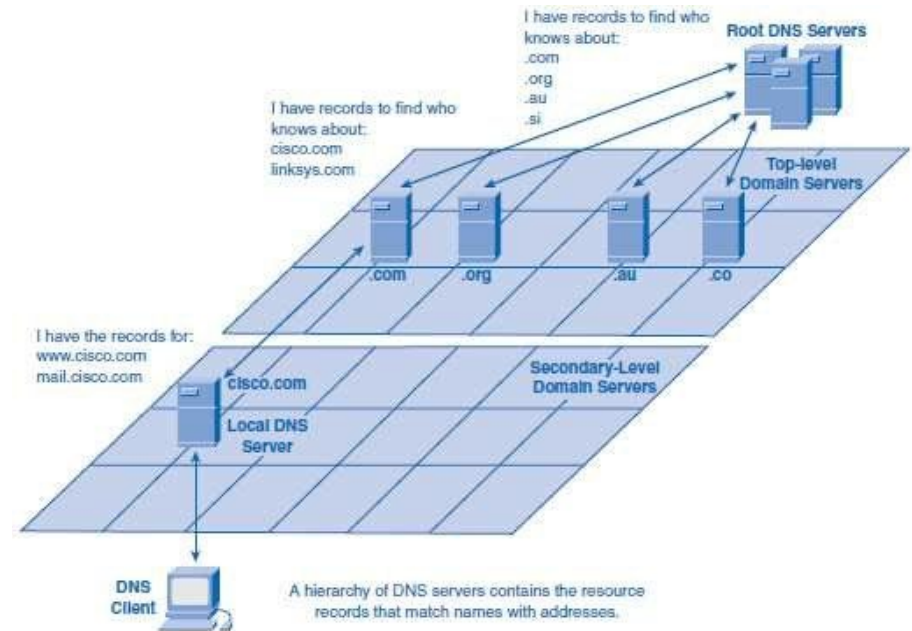
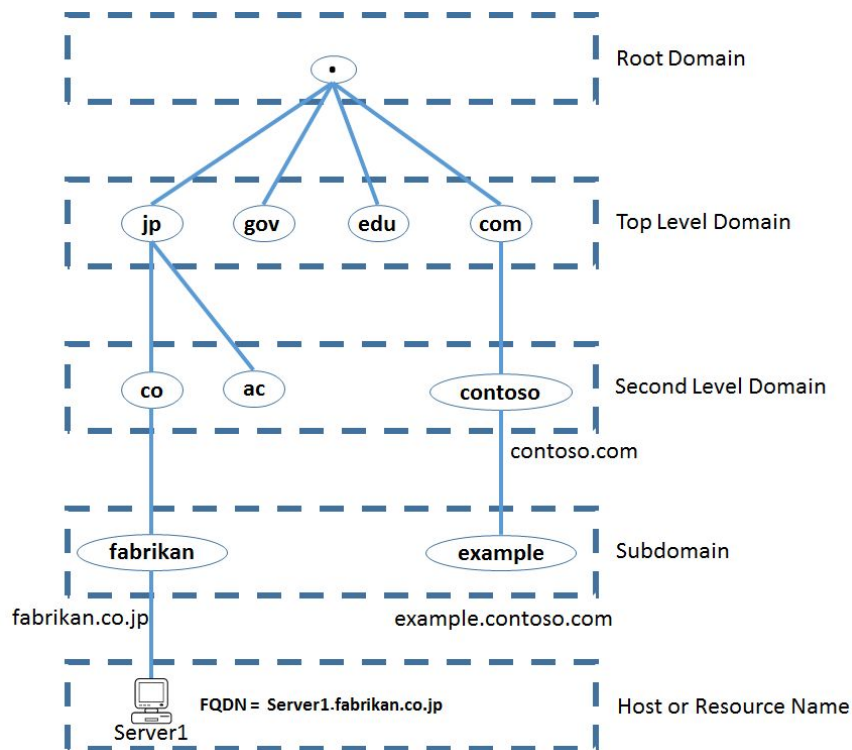
- Maps/Translates “domain names” to numerical IP addresses
 - You can type www.google.com into the browser, and the browser will know to go to 173.194.33.179
- But how might this be done?
 - Some sort of hash (not really practical)
 - A file of all of the mappings (not really practical)

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

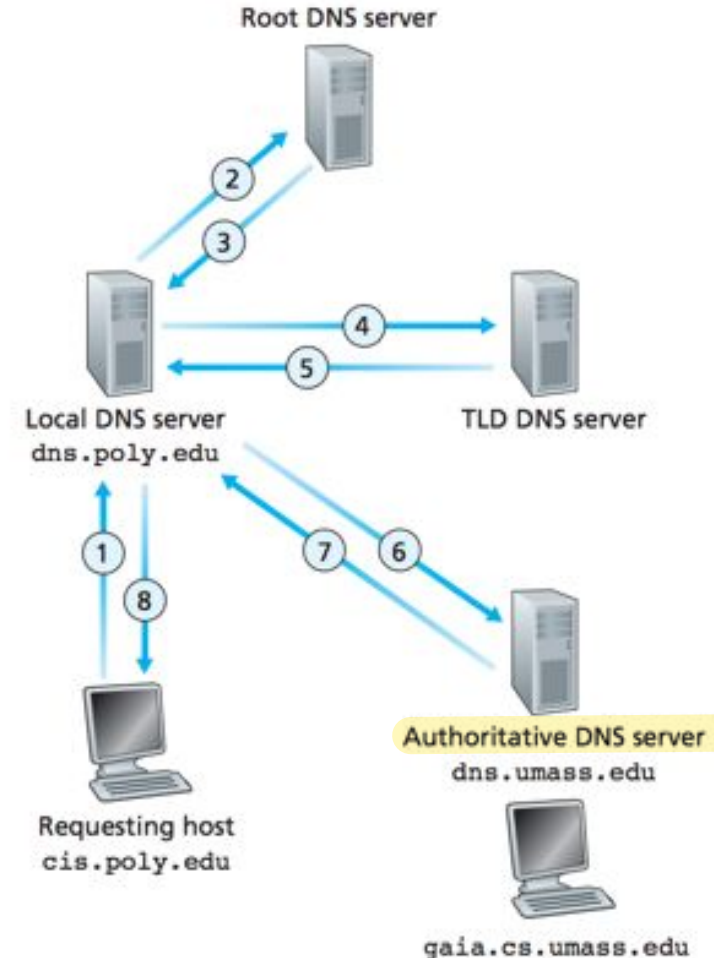
Hierarchical DNS Servers

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



Resolving a Domain Name (1/2)

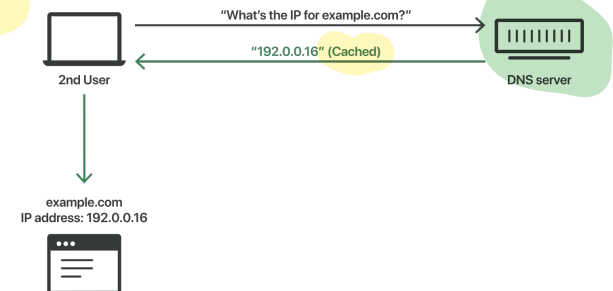
- If I type sports.polimi.com, what happens?
 - Check /etc/hosts
 - Check DNS cache
 - Check local DNS server
 - Go through the hierarchy:
 - Ask . DNS root server
 - Ask .com TLD/SLD (Top/Second Level Domain) server
 - Ask the **Authoritative polimi.com's NS**
 - Send HTTP request to the IP address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



Resolving a Domain Name (2/2)

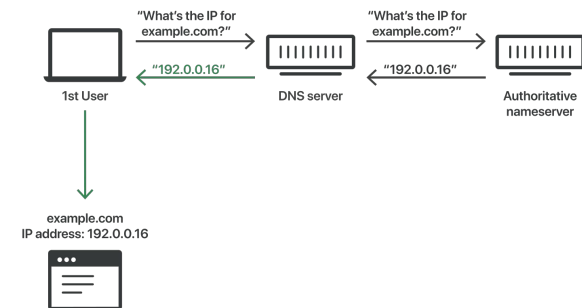
When a **non-authoritative** DNS server receives a request to resolve a domain name:

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



- If **no answer** in cache:

- **Recursive:** resolves the name on behalf of the client.

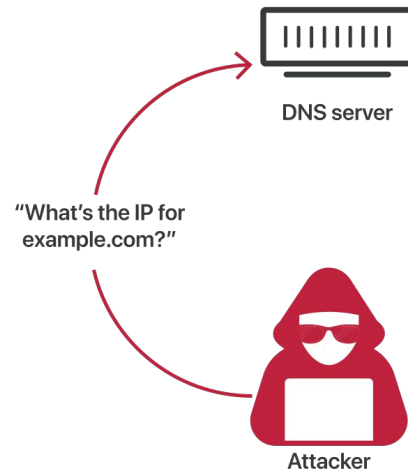


- **Iterative:** gives the authoritative DNS address.

DNS (Cache) Poisoning Attack

Poison the cache of a non authoritative DNS server

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

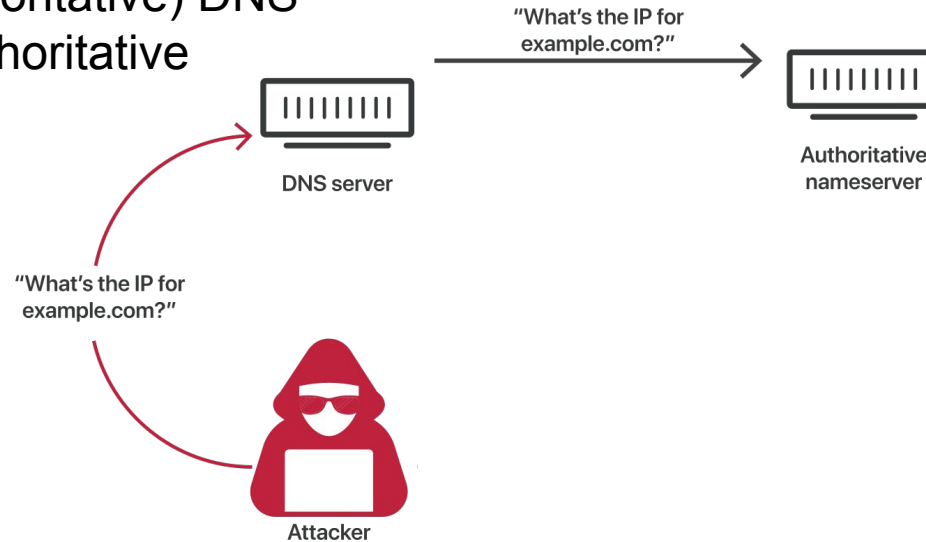


DNS (Cache) Poisoning Attack

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.

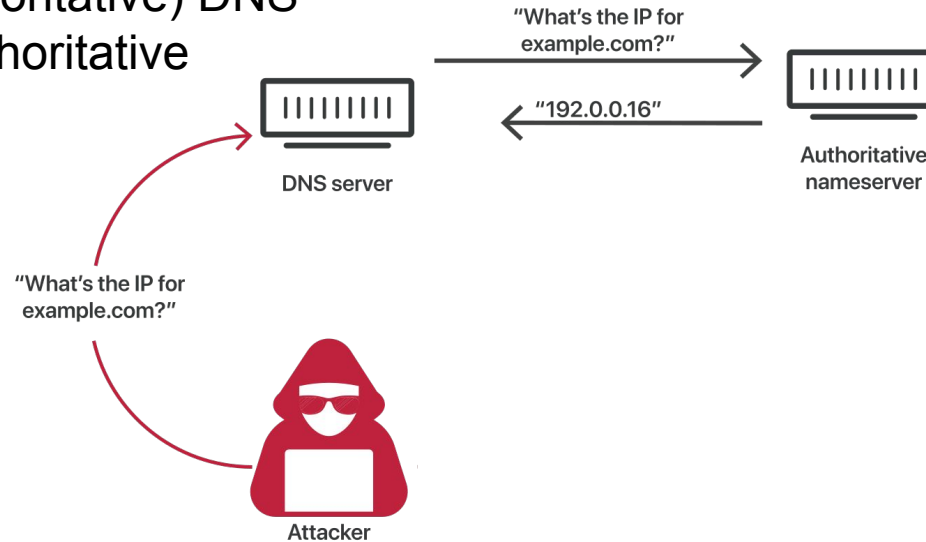


DNS (Cache) Poisoning Attack

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.

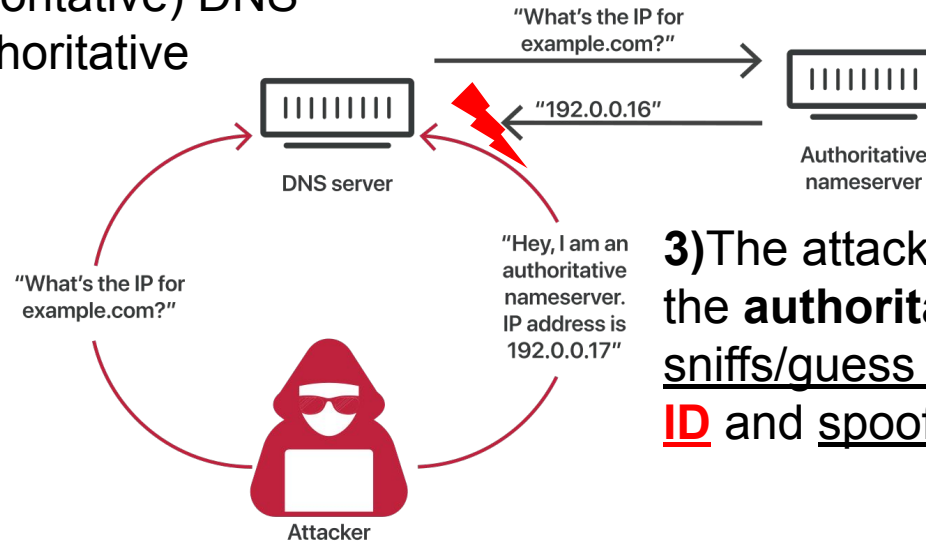


DNS (Cache) Poisoning Attack

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.



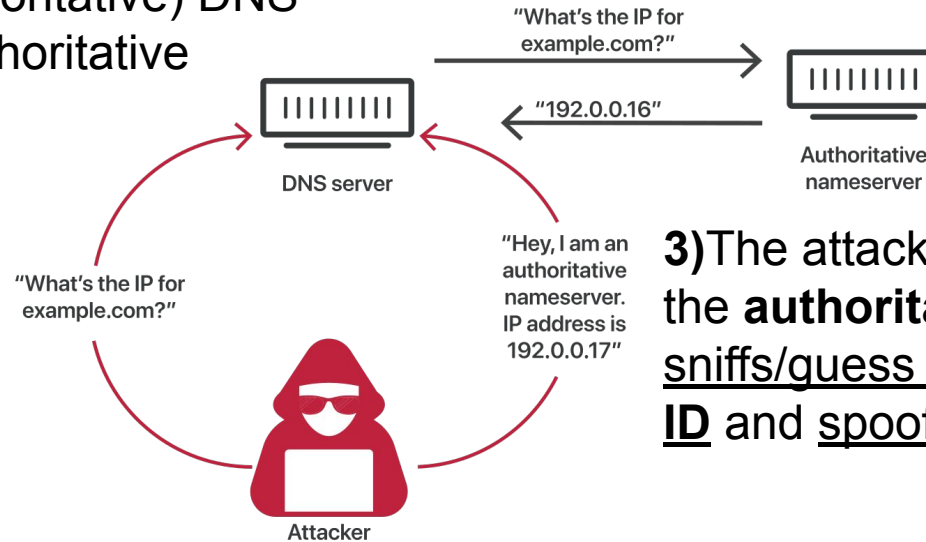
3) The attacker, **impersonating the authoritative DNS server**, sniffs/guesses the **DNS query ID** and spoofs the answer.

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/guesses the DNS query ID and spoofs the answer.

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

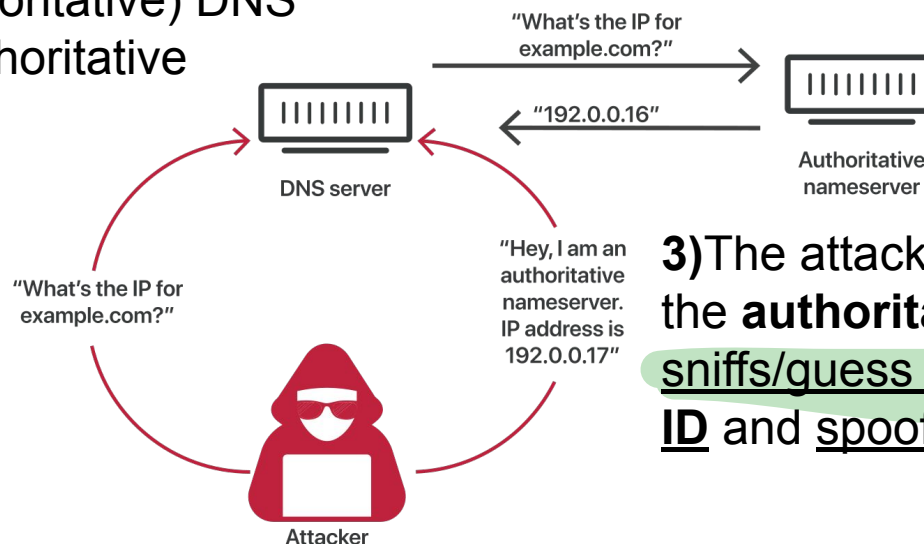


DNS (Cache) Poisoning Attack

Poison the cache of a non authoritative DNS server

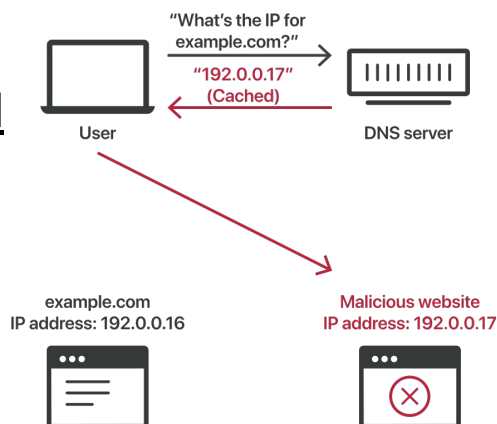
2) The victim (non authoritative) DNS server contacts the authoritative server.

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



3) The attacker, impersonating the authoritative DNS server, sniffs/guesses 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)



Dynamic Host Configuration Protocol (DHCP)

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:

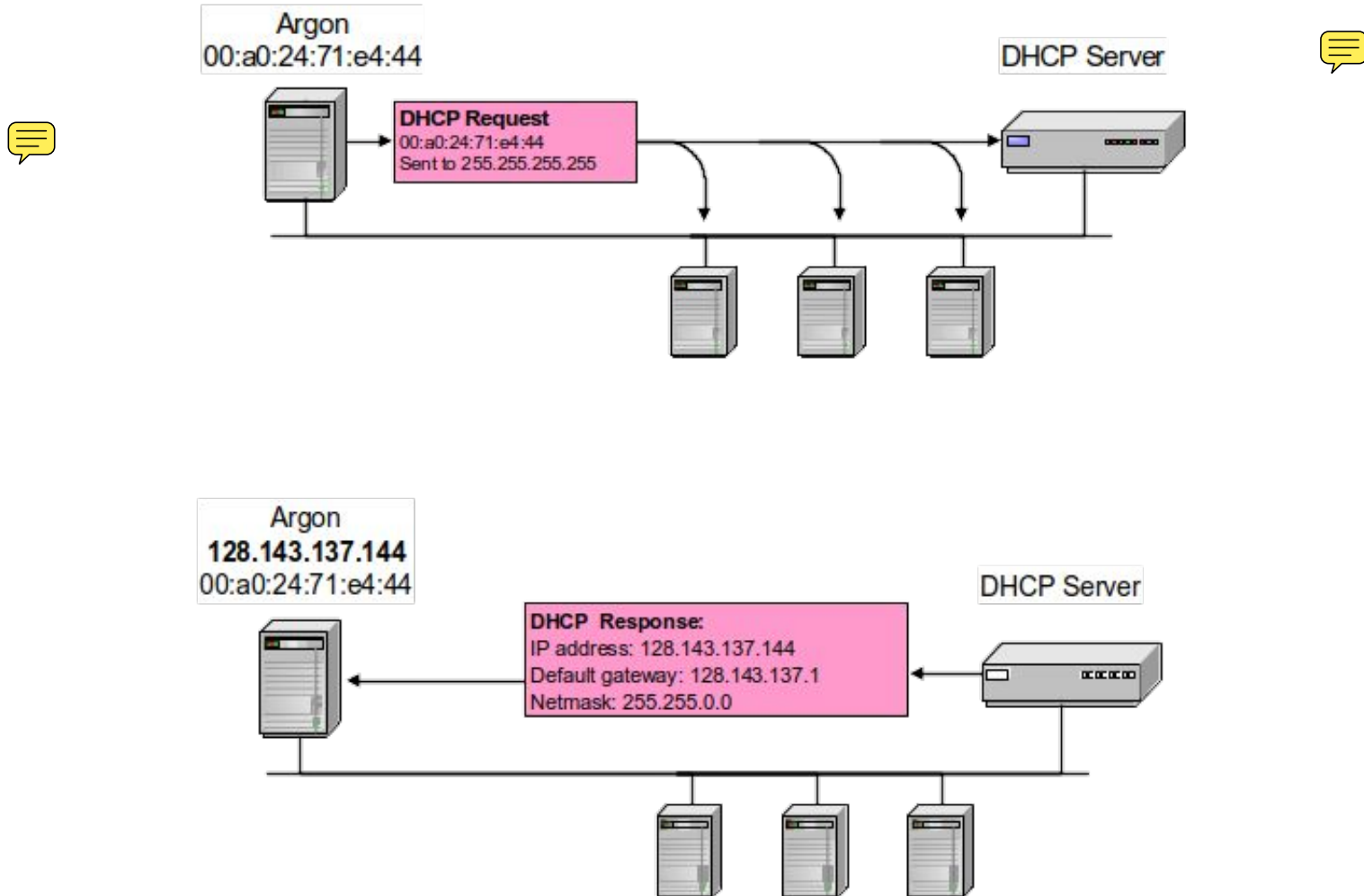
- IP address
- Router
- Subnet Mask
- ...

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.

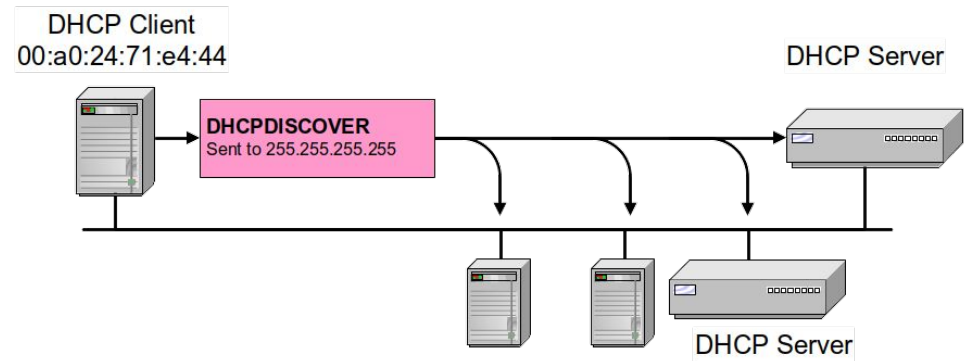


DHCP Interaction (Simplified)

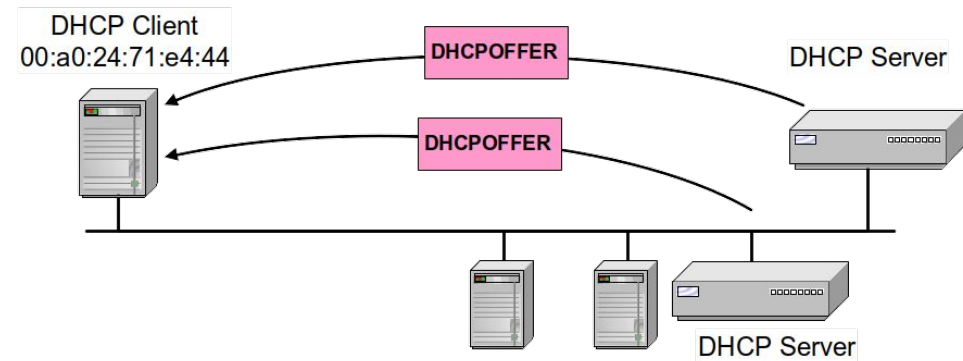


DHCP Operation (1/3)

- DCHP DISCOVER



- DCHP OFFER

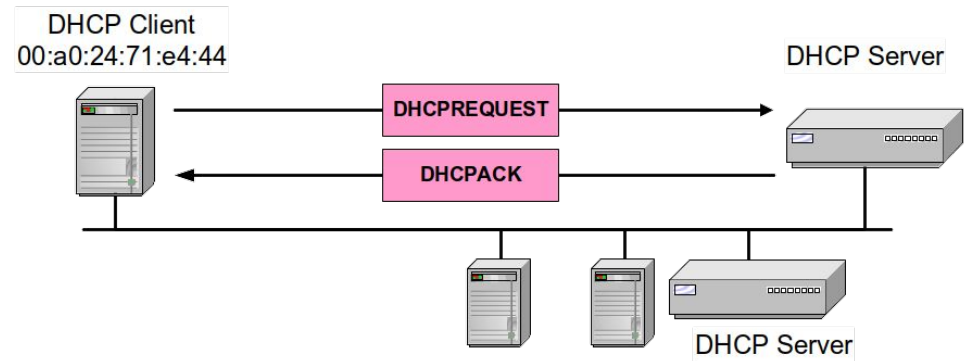


DHCP Operation (2/3)

DCHP DISCOVER

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

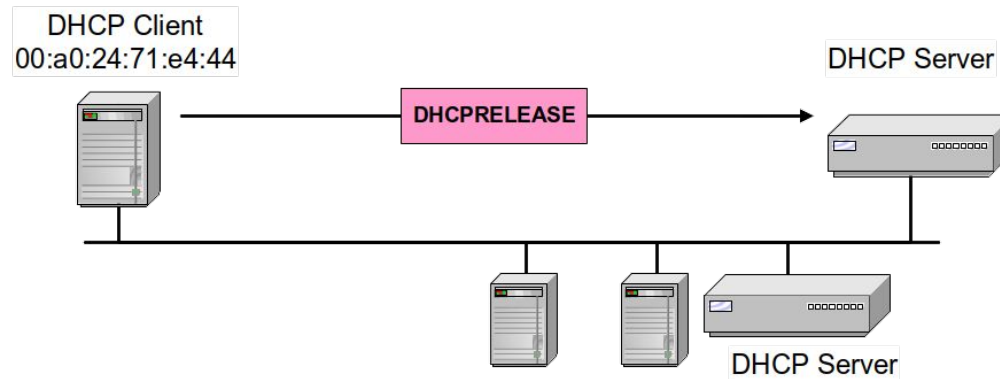
If DHCP server sends **DHCP ACK**, then address is assigned.



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.

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

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

ICMP Echo Request/Reply

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

```
# 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
```

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



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.