

9. Network Protocol Attacks

Computer Security Courses @ POLIMI
Prof. Carminati & Prof. Zanero

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

Denial of Service Examples

- 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

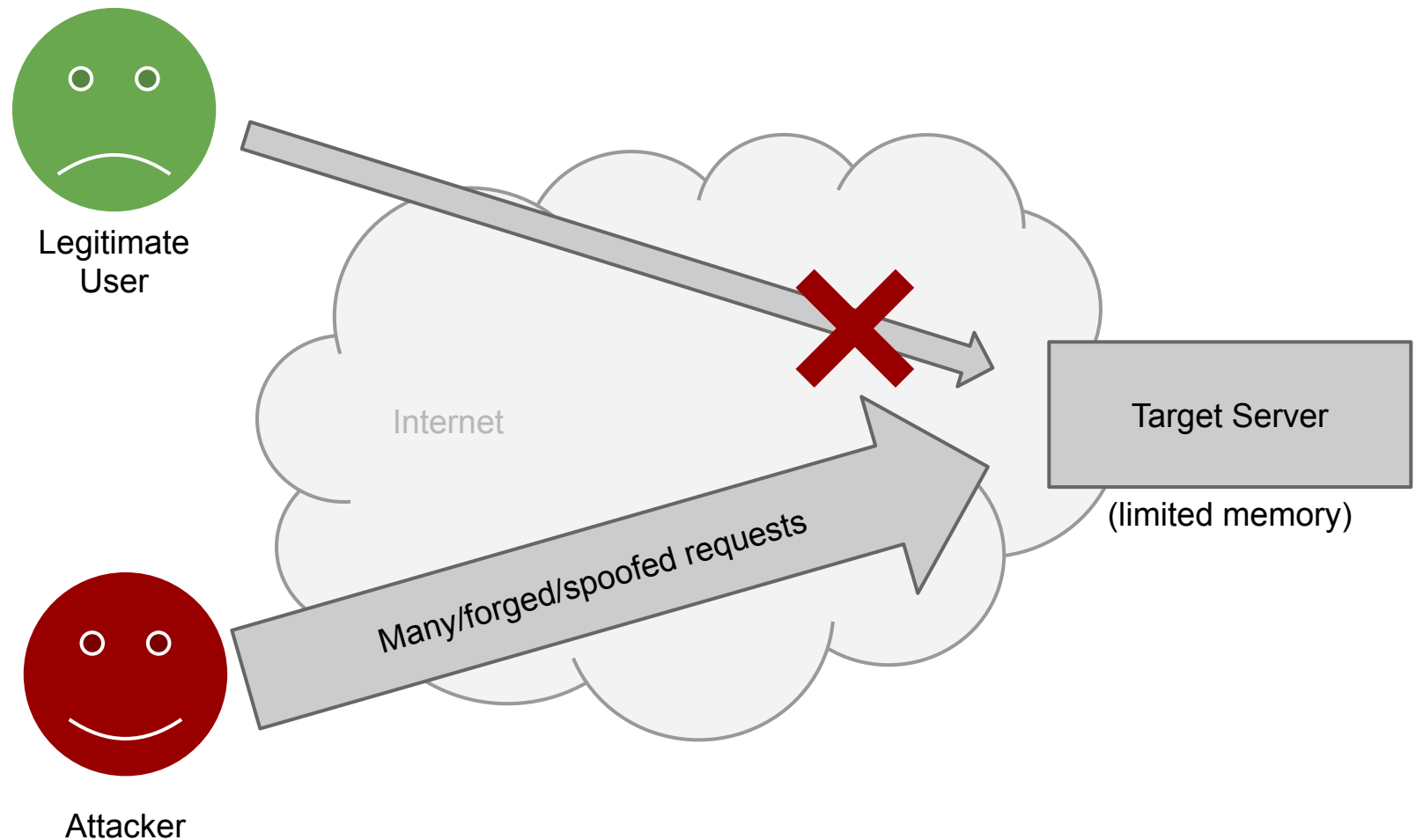
- src IP == dst IP
- SYN flag set

could loop and lock up a TCP/IP stack.

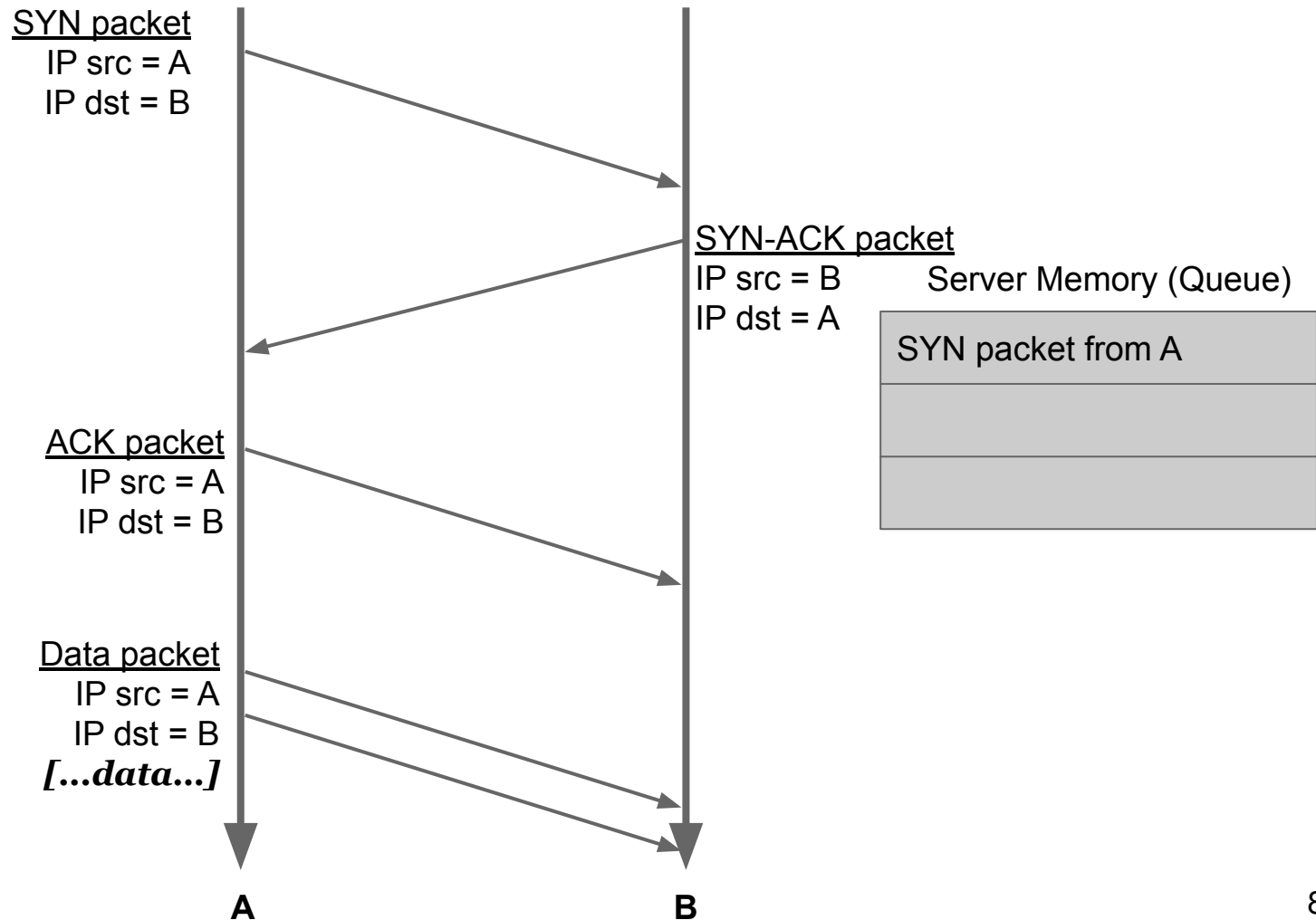
Back to the future, same happened with SP2 in Windows XP: “This thing is like Dracula: it just won't stay dead”

<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 Flood Attack: The TCP/IP Three Way Handshake



Attacker

SYN packet
IP src = A
IP dst = B

~~ACK packet~~
~~IP src = A~~
~~IP dst = B~~

SYN-ACK packet

IP src = B
IP dst = A

Server Memory (Queue)

SYN packet from A

Multiplier Factor

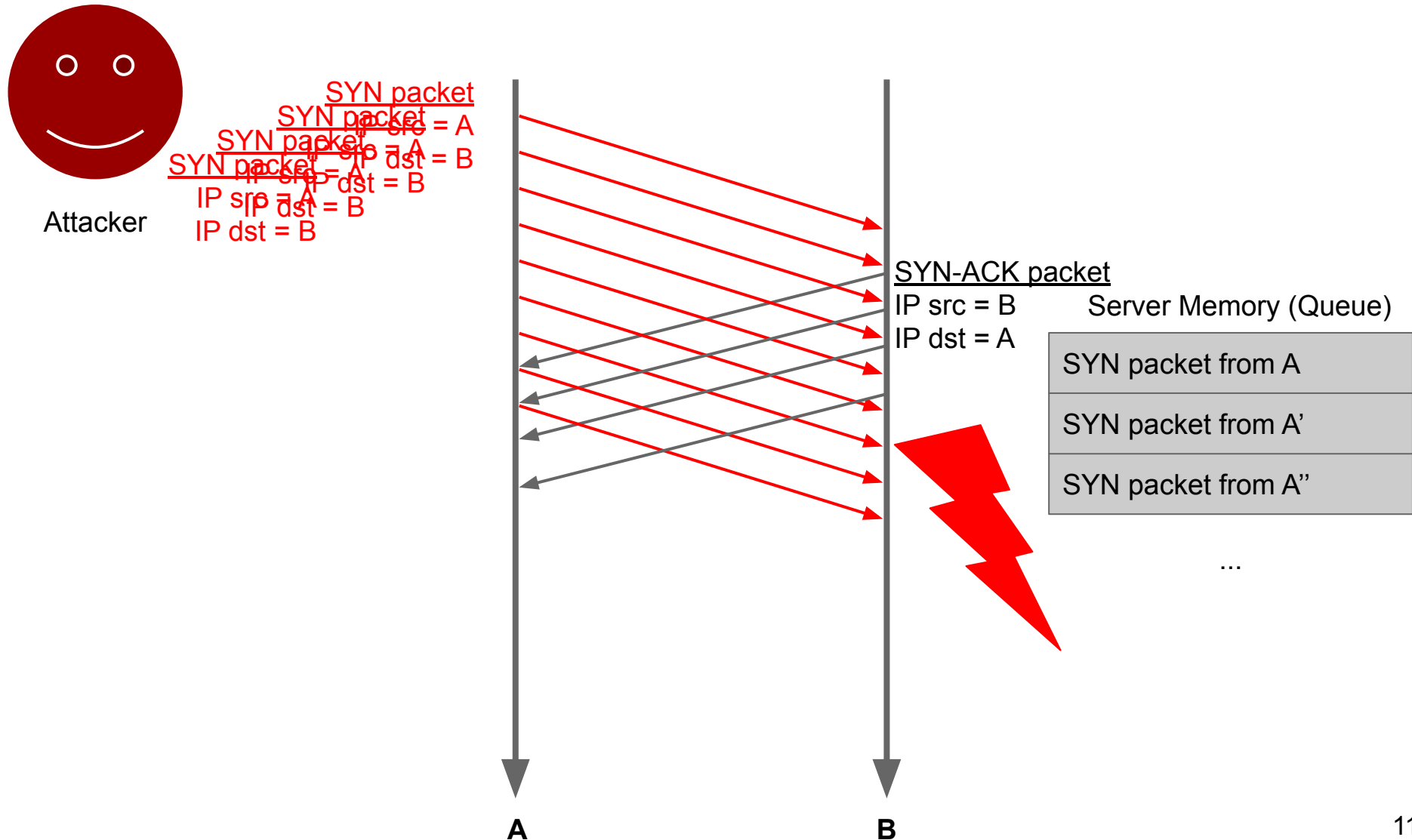
A

B

SYN Flood Attacks

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

SYN Flood Attack: The TCP/IP Three Way Handshake

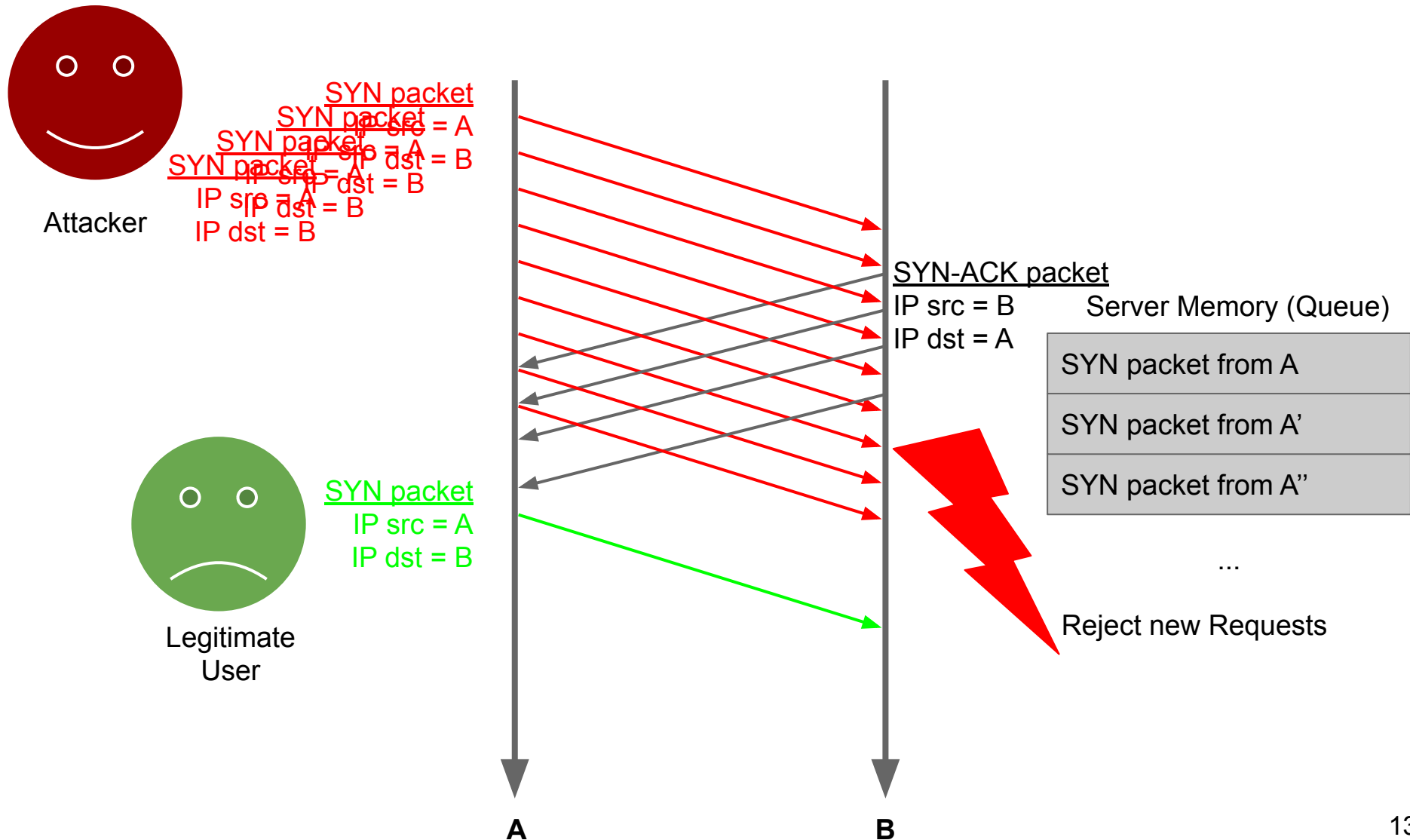


SYN Flood Attacks

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

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

SYN Flood Attack: The TCP/IP 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.

SYN Flood Attacks

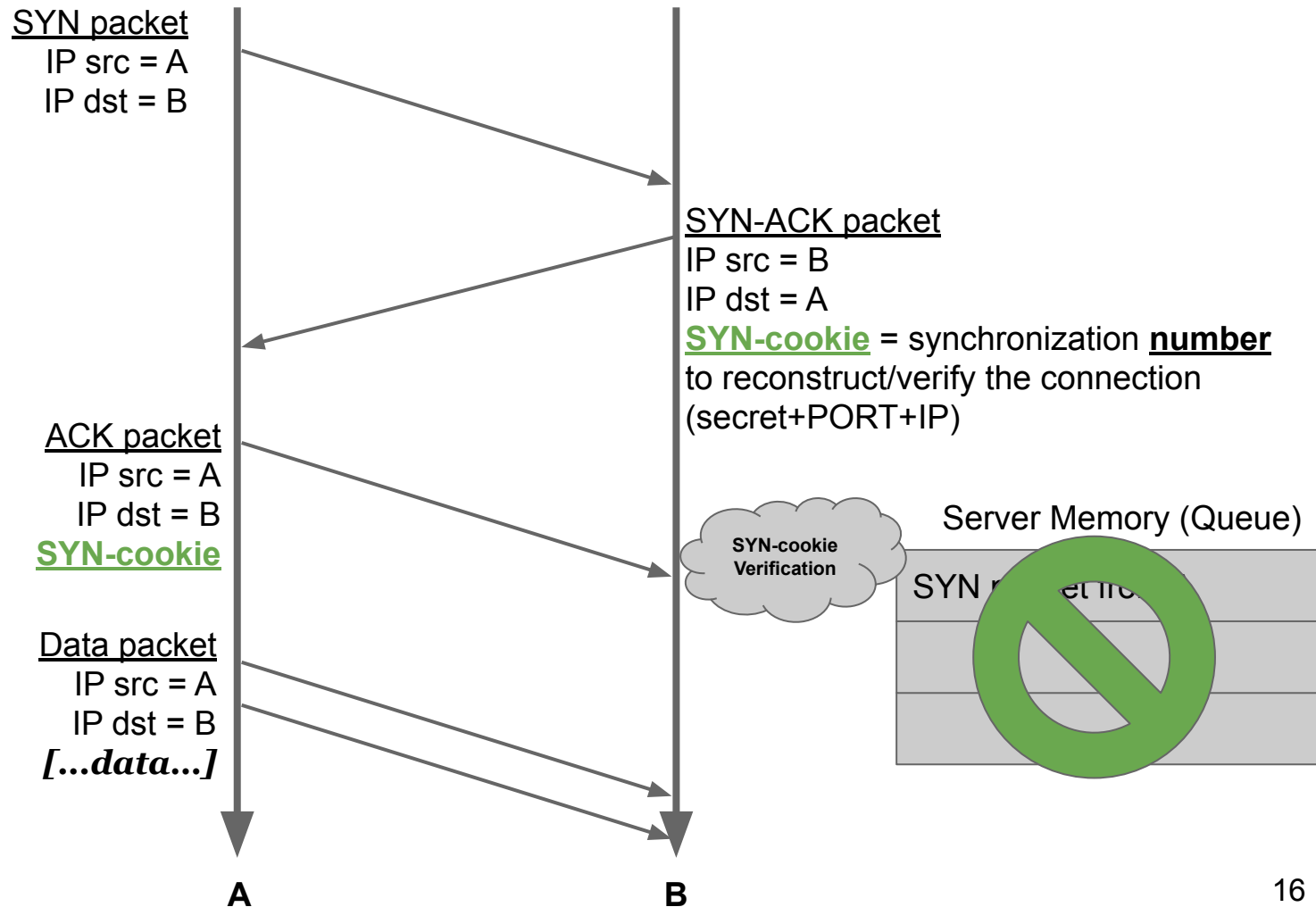
Attacker generates a high volume of SYN requests with **spoofed source address**.

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

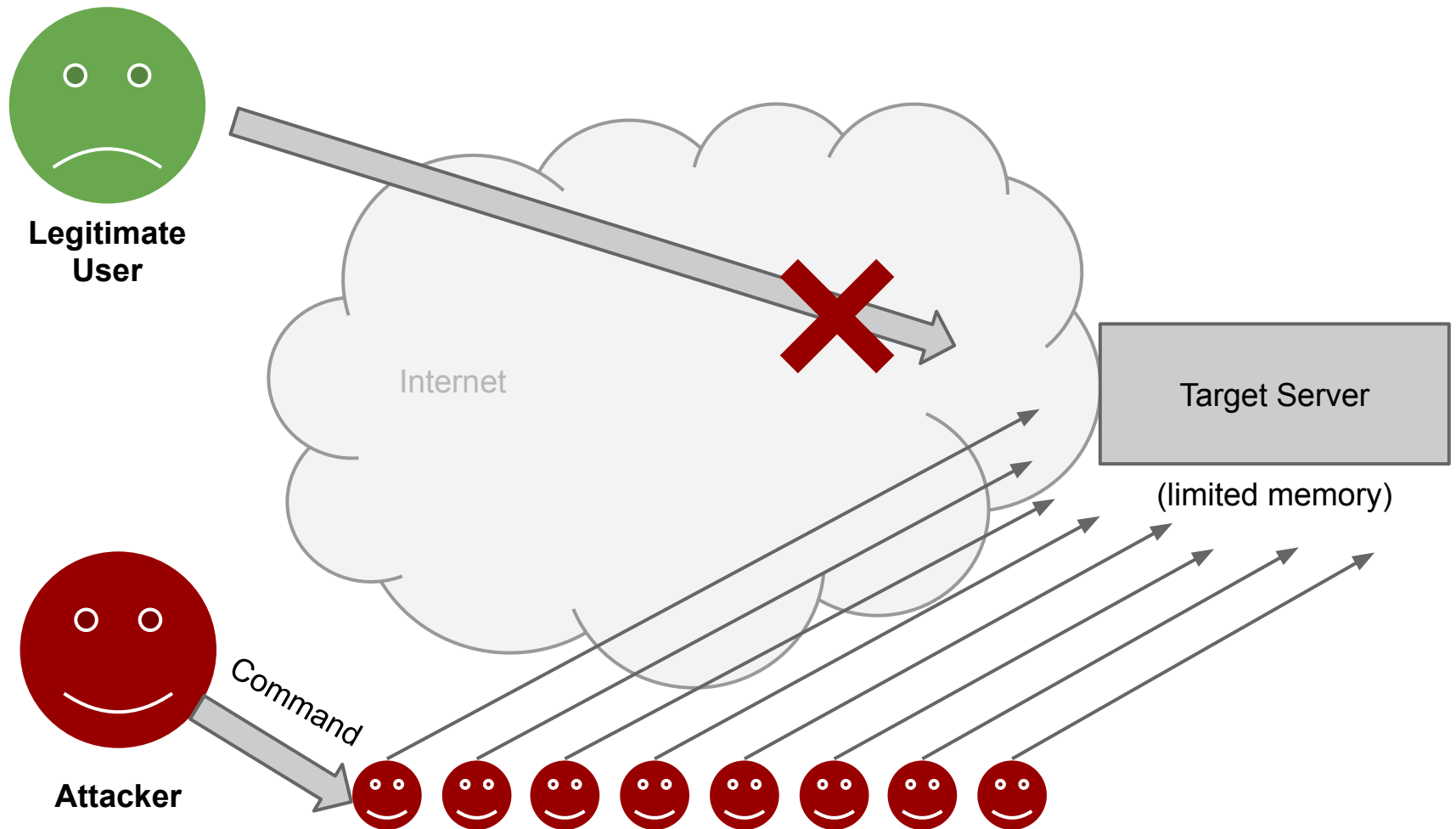
SYN requests from legitimate clients dropped.

SYN-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).

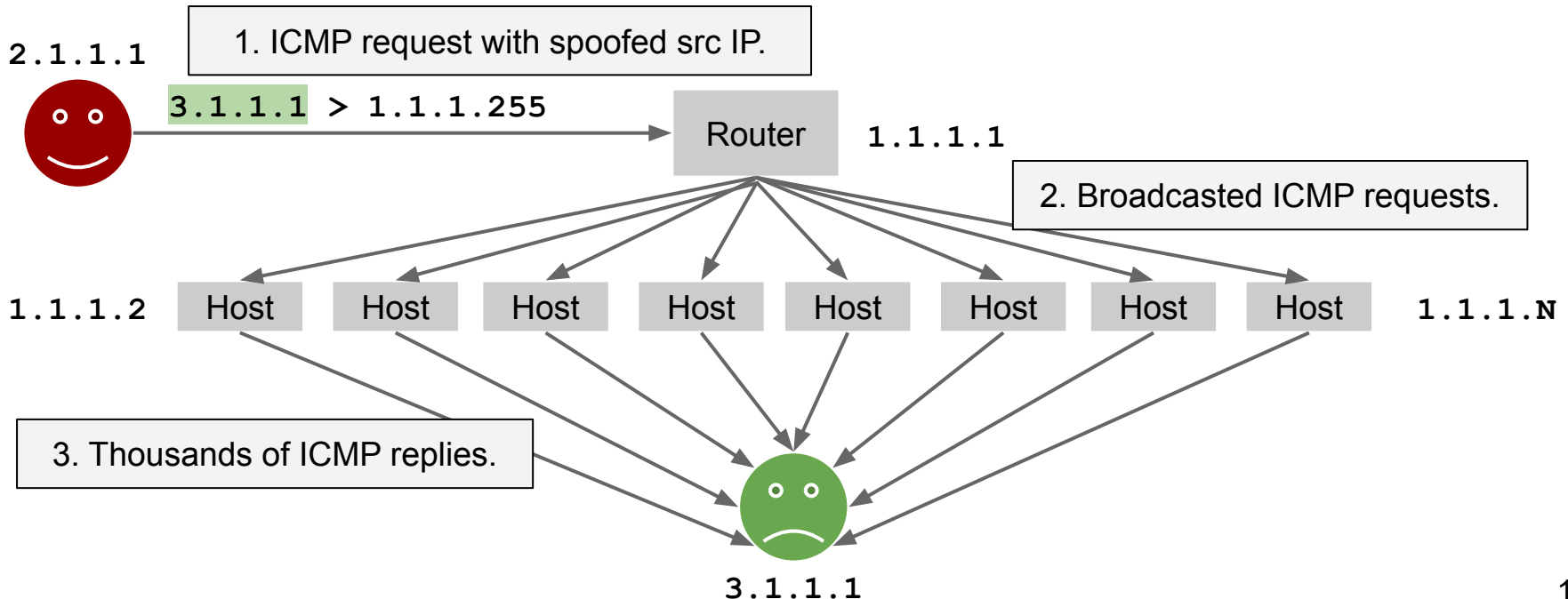
C&C: dedicated command-and-control infrastructure so that the attacker (botmaster) can send commands to the bots.

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 50%	10%	PAF <i>all</i>	Scenario
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 passes to the OS any packet read off of the wire.

DSniff tool www.monkey.org/~dugsong/dsniff

ARP spoofing, MAC flooding, sniffing.

Mitigations Against (basic) sniffing

Use switched networks as opposed to hub-based networks.

Hubs broadcast traffic to every host. NICs can be in promiscuous mode. Broadcast domain.

Switches selectively relay traffic to the wire corresponding to the correct NIC (ARP address based).

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]# ./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
```

...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 conflicts of addresses)
- Add a SEQ/ID number in the request
- ...

Filling up a CAM Table (or FIB)

- 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)

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

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

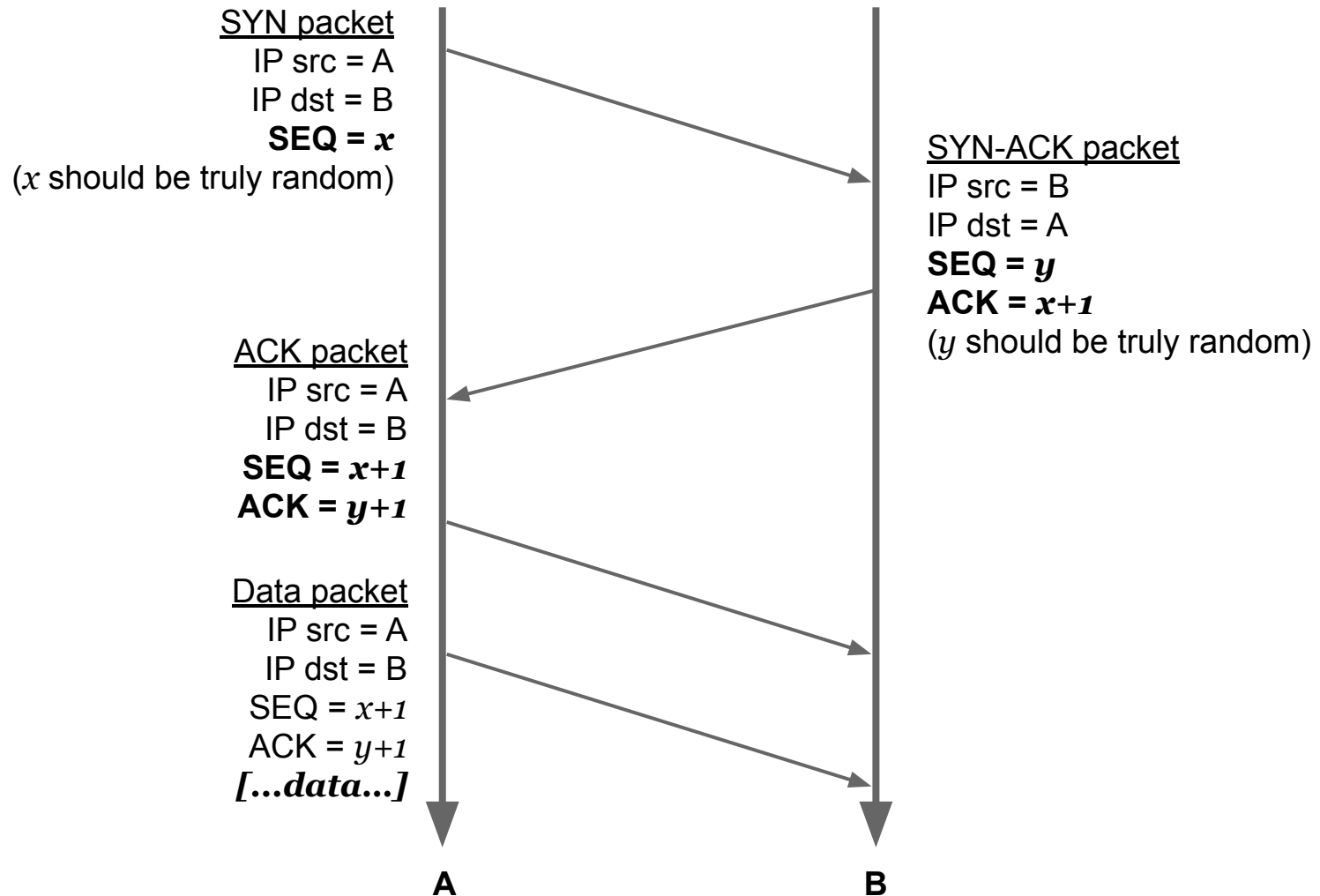
Changing it in **UDP or ICMP** packets is **easy**.

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....

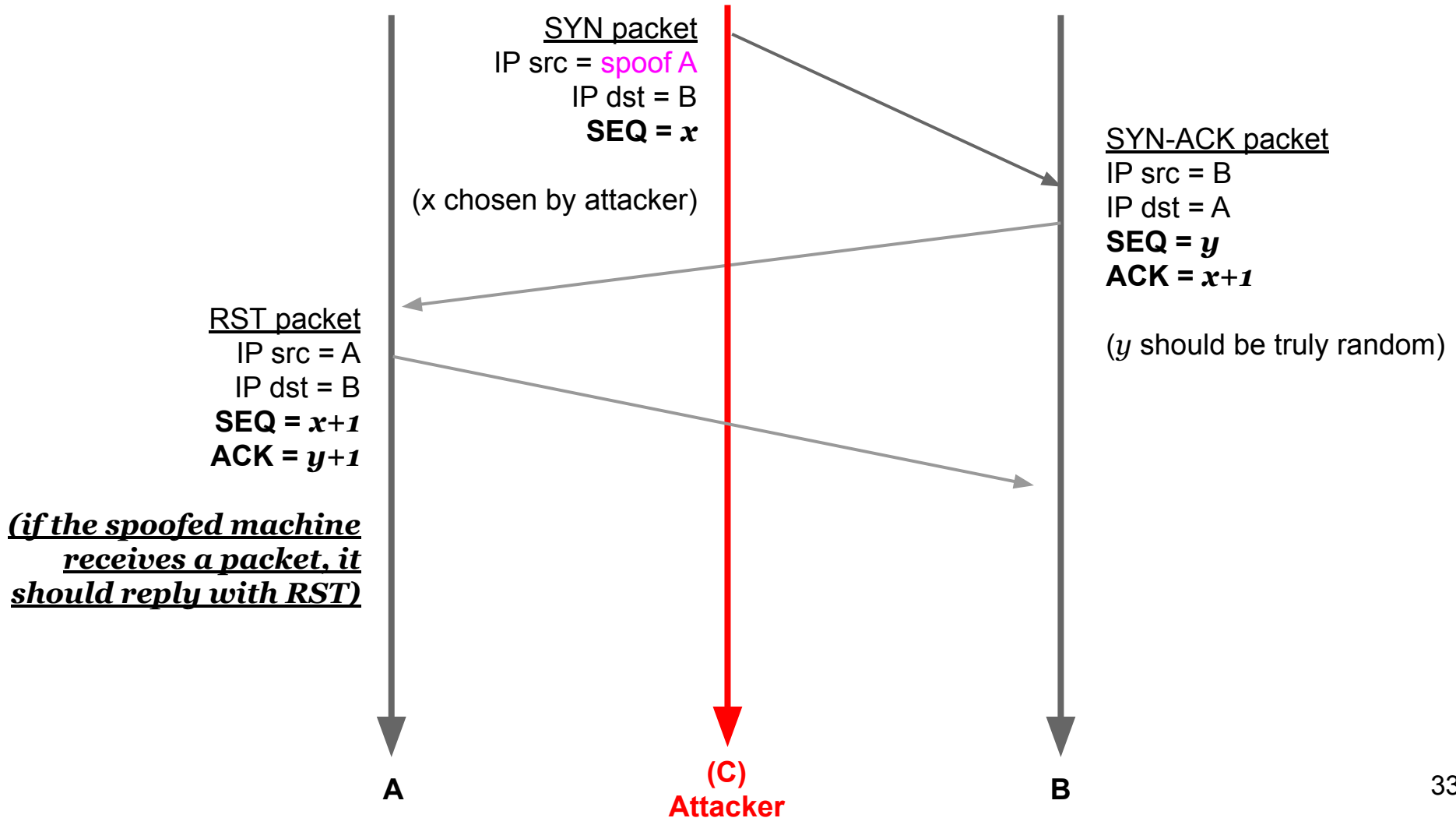
Recall the Three Way Handshake



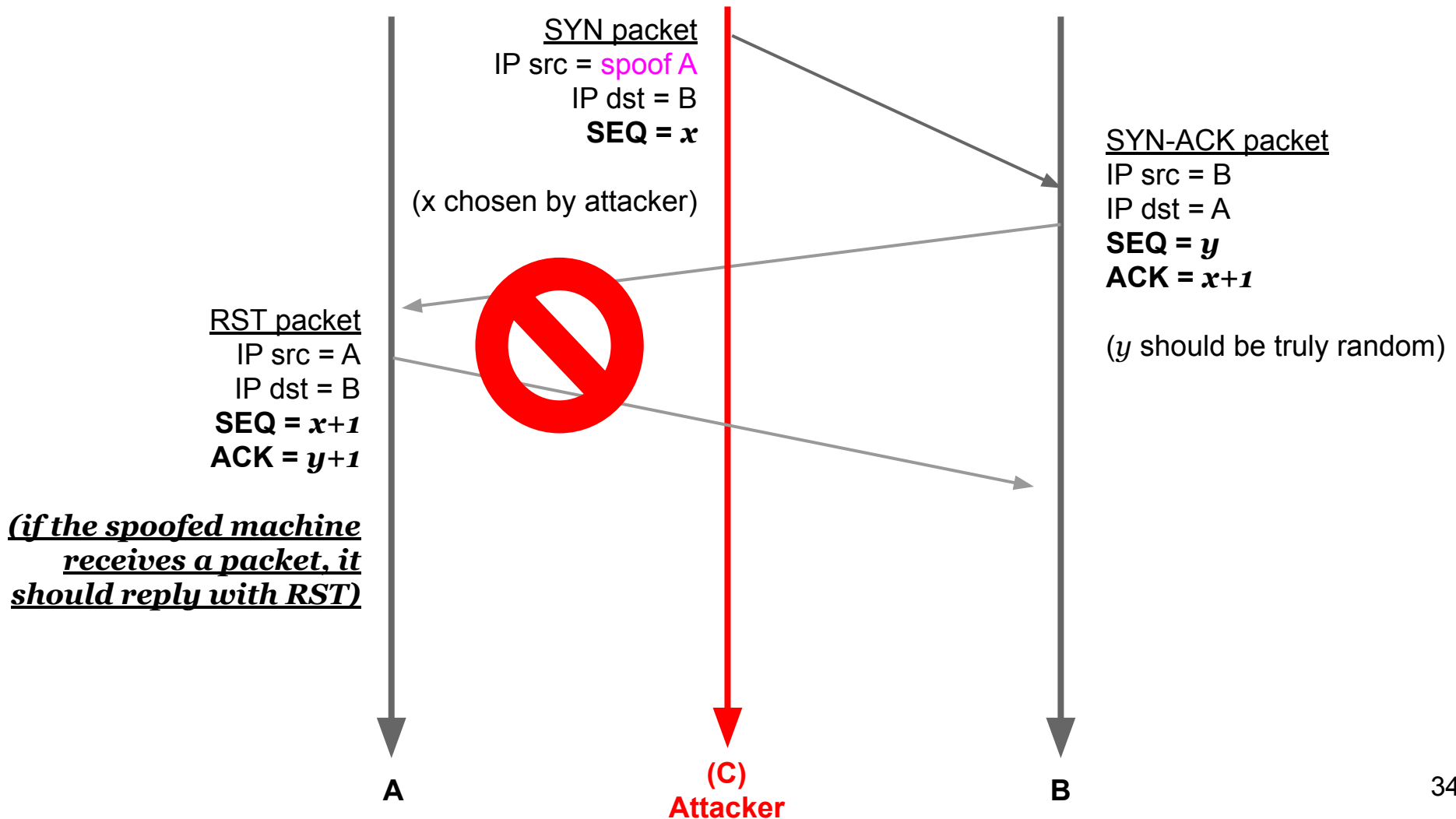
TCP Sequence Number Guessing

- 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 needs not to receive the response packets, otherwise it might answer with a RST.

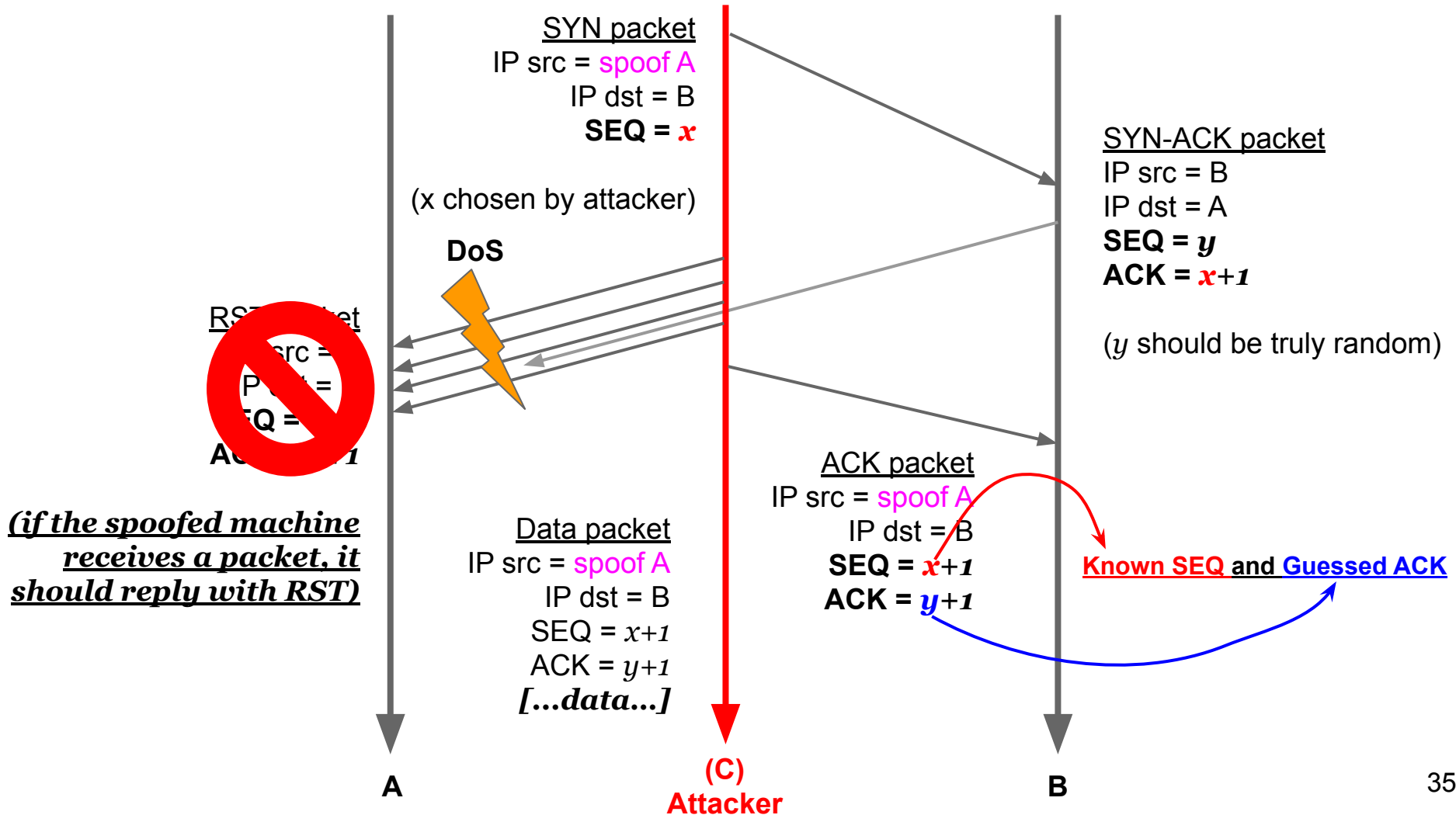
TCP/IP (Blind) Spoofing Attack: Sequence Number Guessing



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



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.

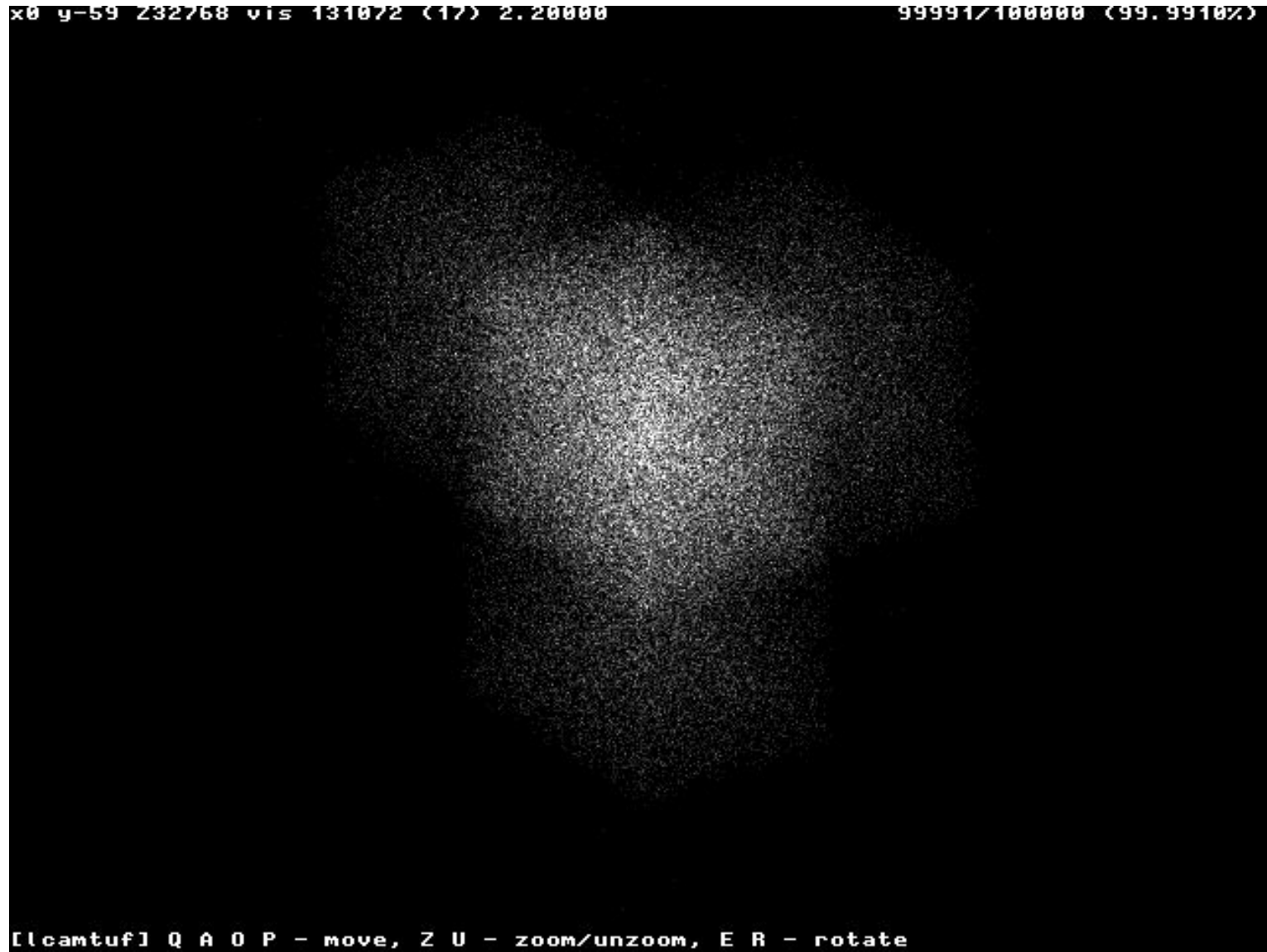
What changed since 1994?

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

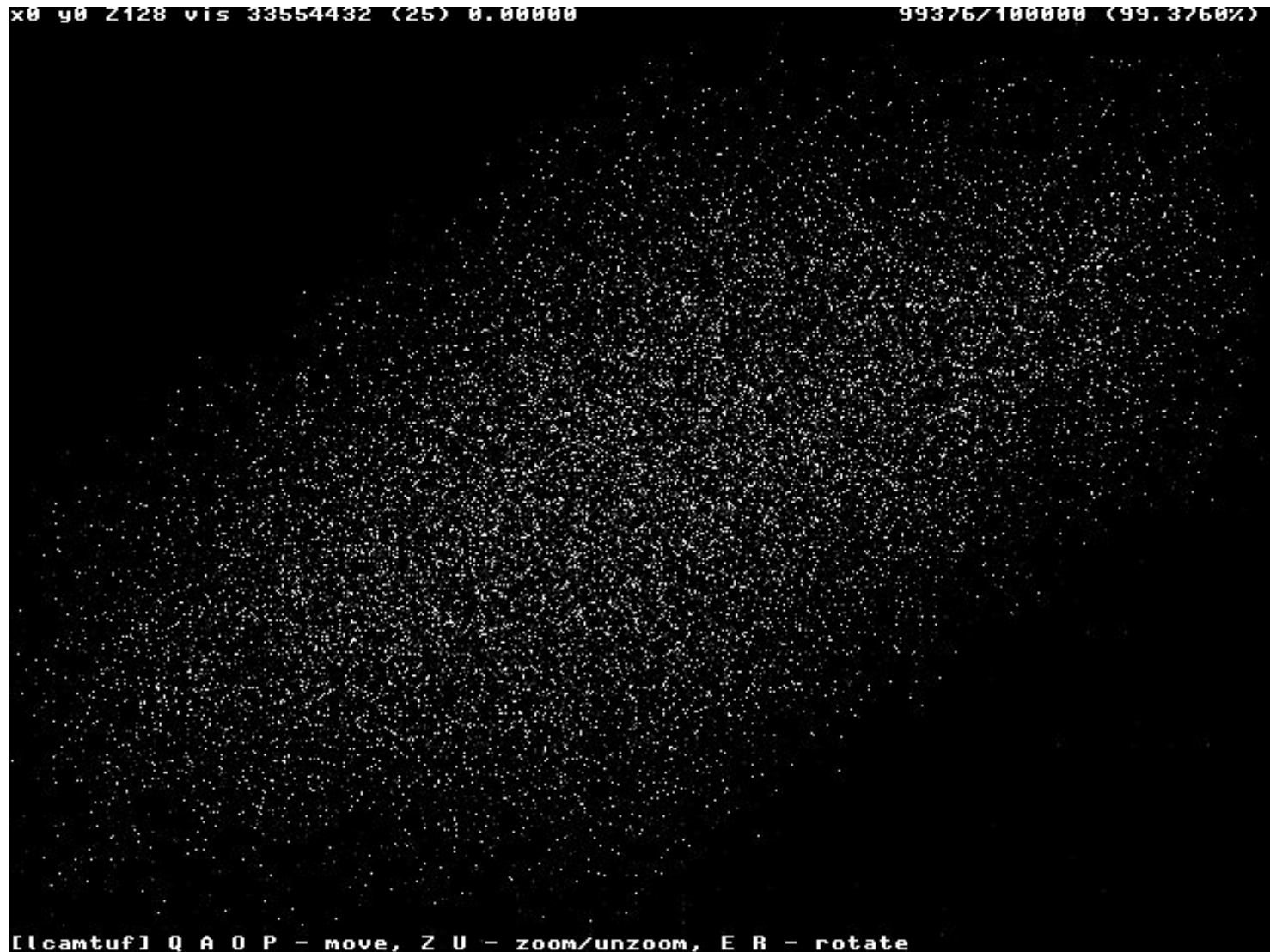
IRIX 6.5.15 ISN Distribution



Windows XP SP2 ISN Distribution



Netware 6 SP2 ISN Distribution



Netware 6 SP3 ISN Distribution



*BSD family ISN Distribution



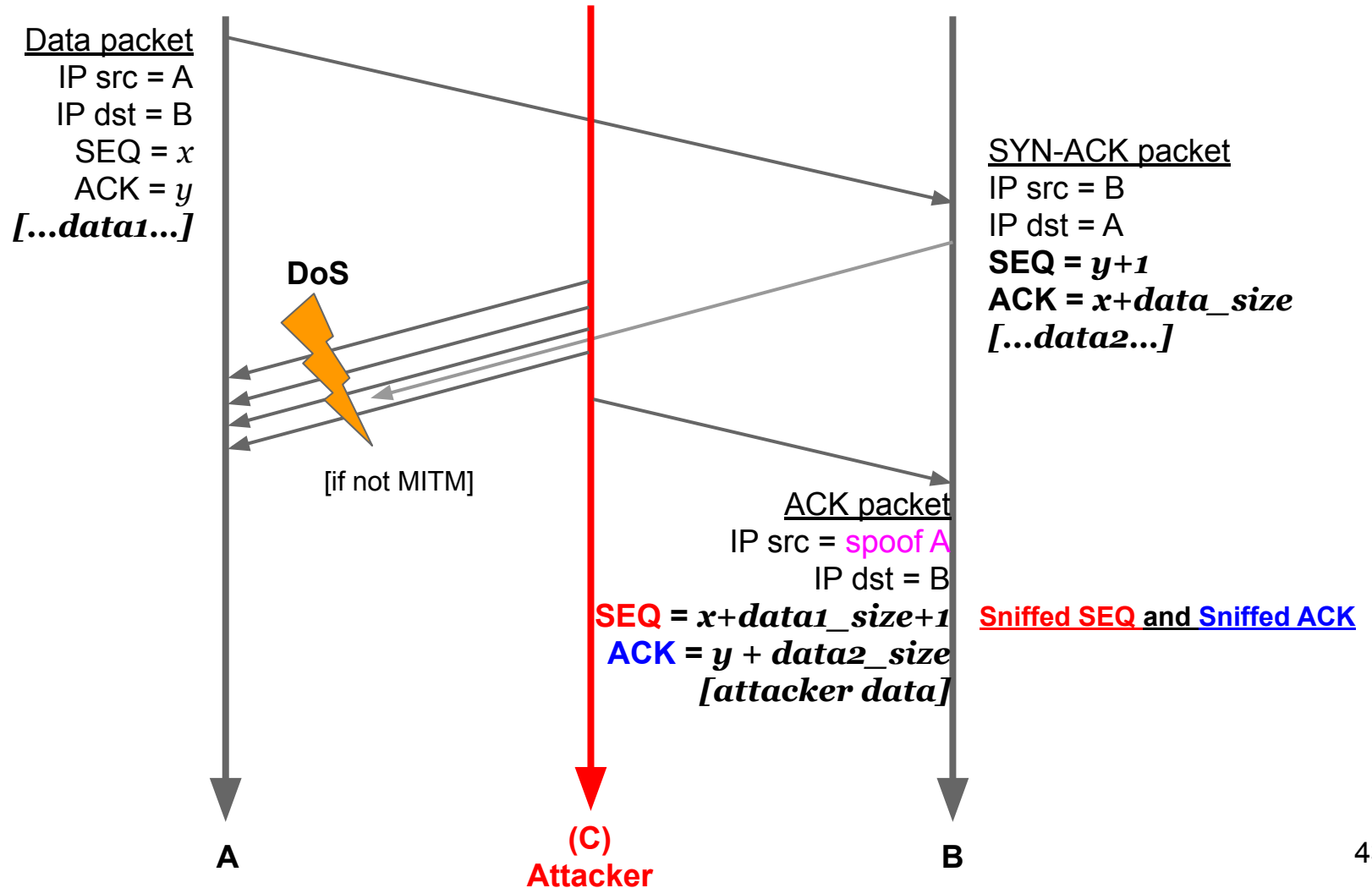
TCP Session Hijacking

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.

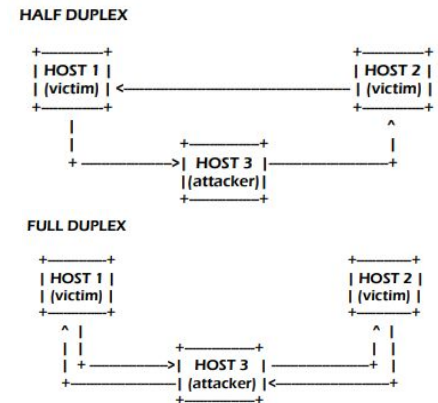
What's a **man in the middle**?

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.

MITM

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



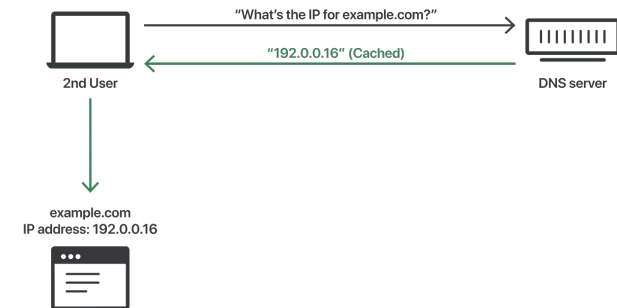
What happens if the attacker is able to ARP-spoof the gateway of a LAN? :-)

DNS: Resolving a Domain Name

DNS translates domain names to the numerical IP addresses.
It is based on UDP and messages are not authenticated.

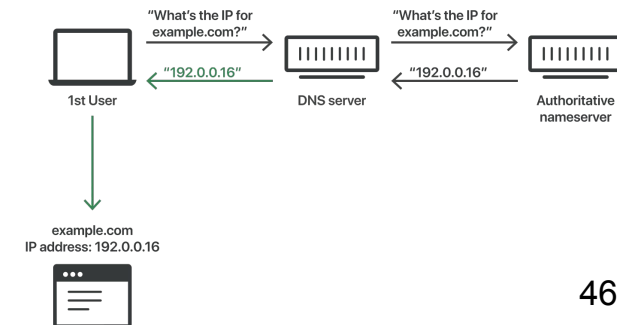
When a **non-authoritative** DNS server receives a request to resolve a domain name:

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



If **no answer** in cache:

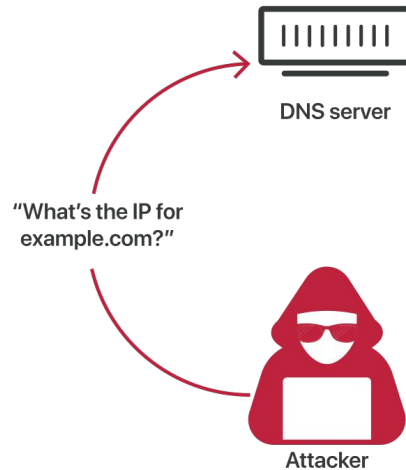
- **Recursive:** resolves the name on behalf of the client.
- **Iterative:** gives the authoritative DNS address.



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.

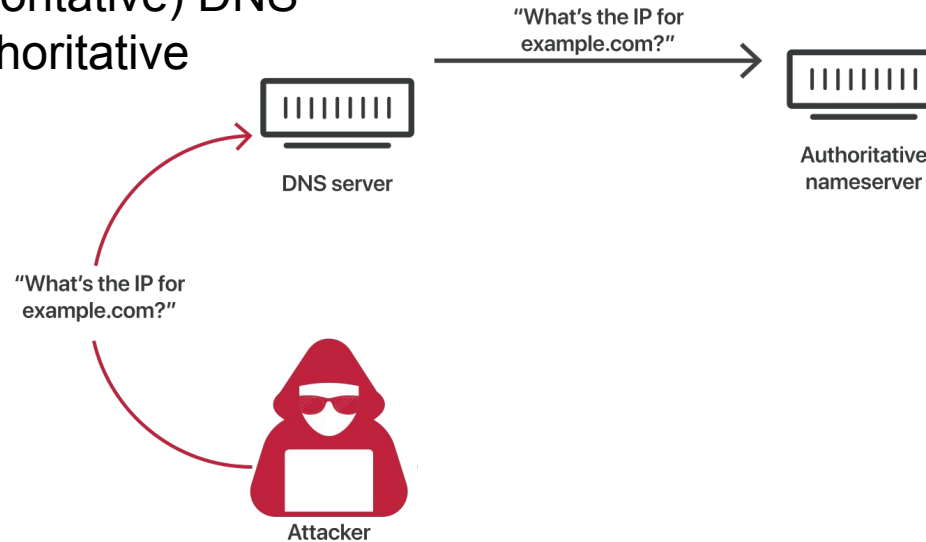


DNS (Cache) Poisoning Attack

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.

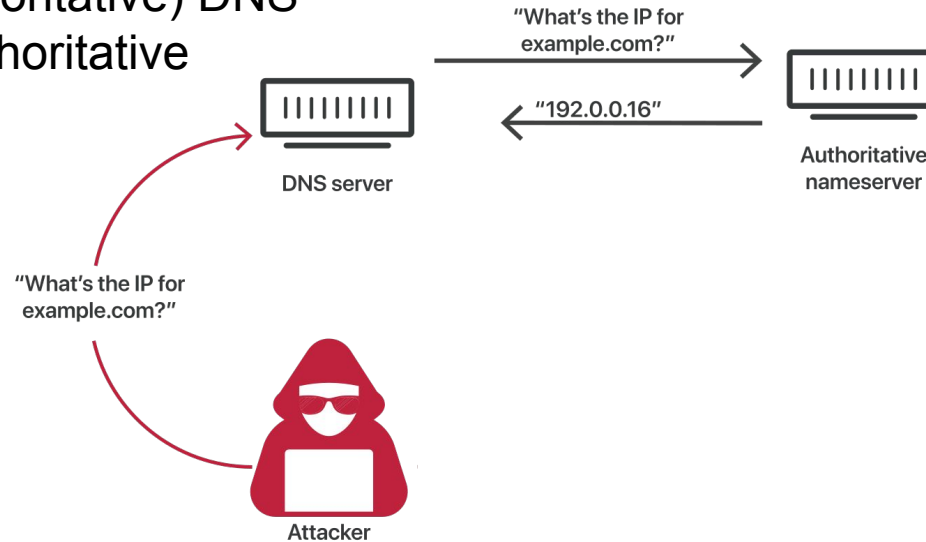


DNS (Cache) Poisoning Attack

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.

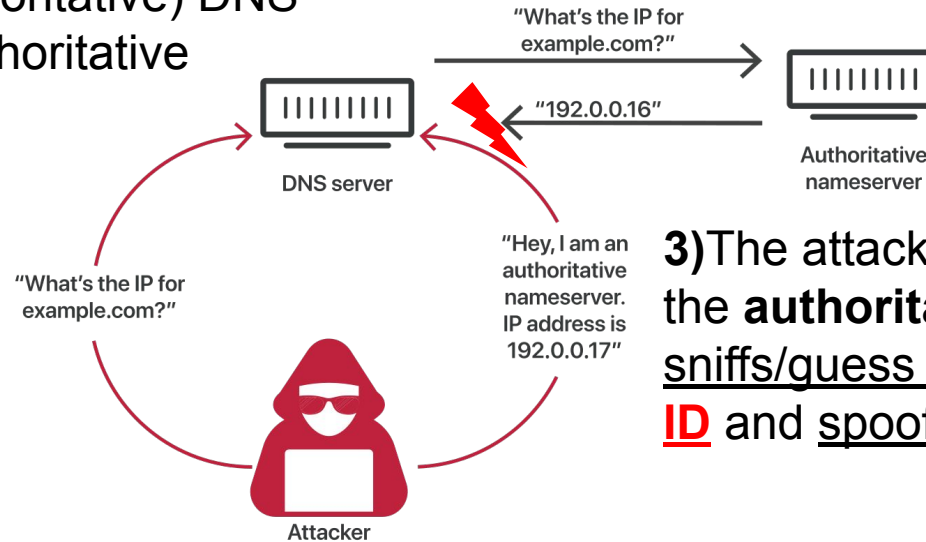


DNS (Cache) Poisoning Attack

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.



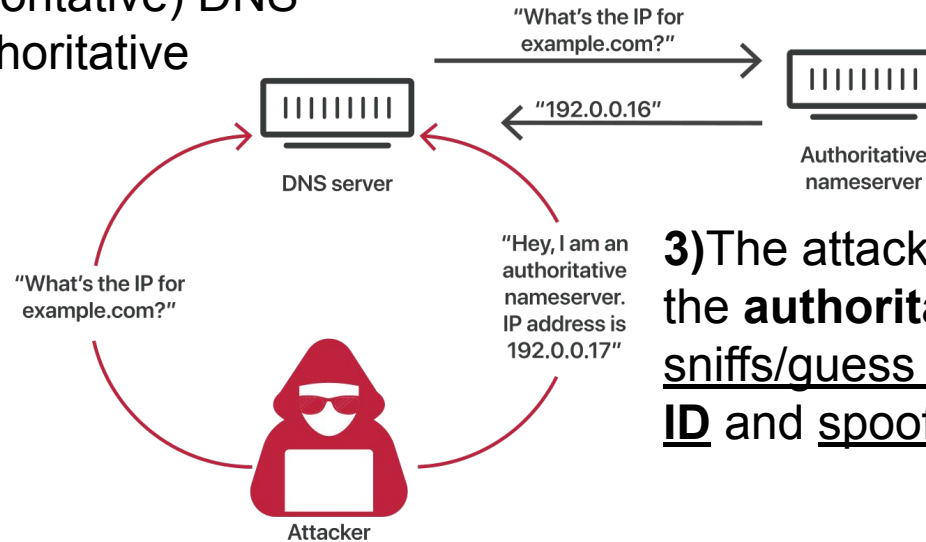
3) The attacker, **impersonating** the **authoritative** DNS server, sniffs/guesses the **DNS query ID** and **spoofs the answer**.

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 the **DNS query ID** and spoofs the answer.

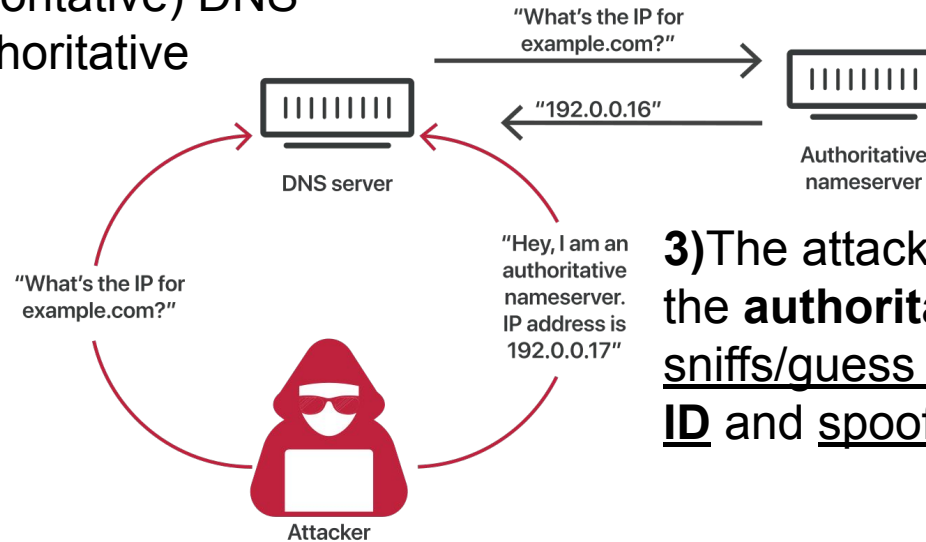
4) The victim DNS server trusts and caches the malicious record **[POISONED]**.

DNS (Cache) Poisoning Attack

Poison the cache of a non authoritative DNS server

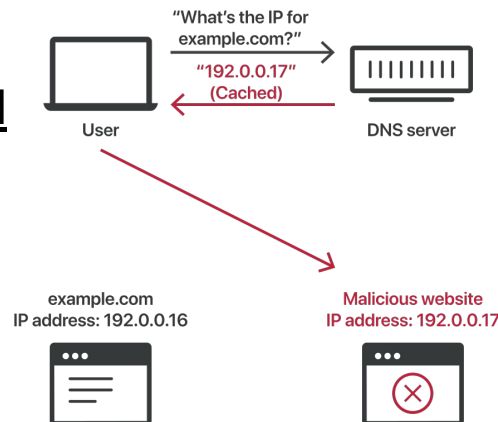
2) The victim (non authoritative) DNS server contacts the authoritative server.

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



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

DHCP Poisoning Attack

The DHCP server dynamically assigns an IP address (and network parameters) to each device on a network.

DHCP is **not authenticated** (not for performance reasons).

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

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

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

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 is not a gateway 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 offending 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.