Network Vulnerabilities: OSI Layer 4 and above

Segurança Informática em Redes e Sistemas
2024/25

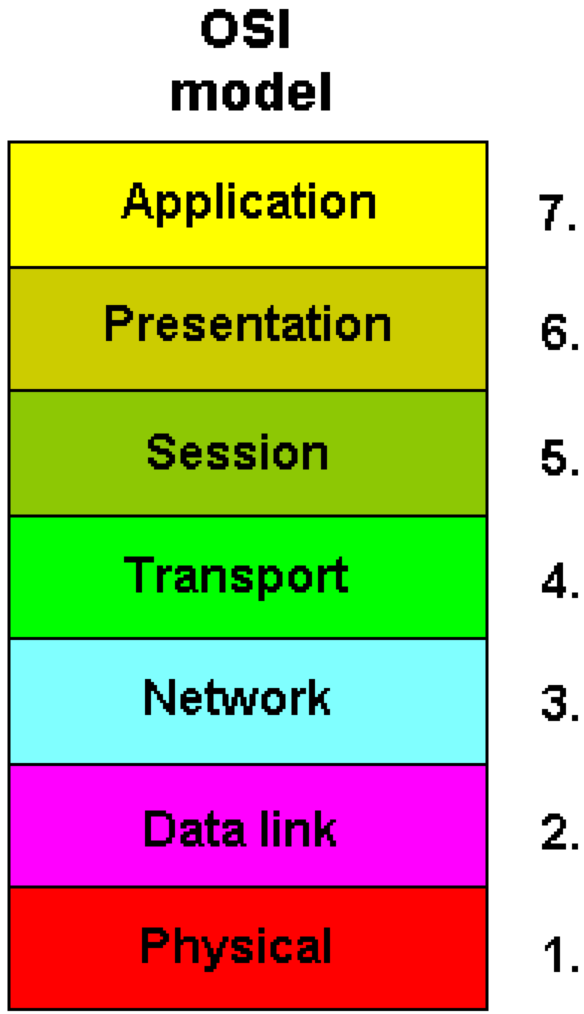
Ricardo Chaves, David R. Matos

Ack: Carlos Ribeiro, André Zúquete,
Miguel P. Correia, Miguel Pardal

Roadmap

- Network vulnerabilities
  - Physical layer
  - Data link layer
  - Network layer
  - Transport layer
  - Application layer
- Network security models

OSI network model



Roadmap

- **Network vulnerabilities**
  - Physical layer
  - Data link layer
  - Network layer
  - **Transport layer**
  - Application layer
- Network security models

(Layer 4) Transport Layer

- Topics:
  - UDP
  - TCP
    - Handshake
    - Hijacking
  - DoS
    - TCP DoS
    - ICMP DoS
  - Solutions

UDP

- User Datagram Protocol
- This protocol can be used to send and receive individual packets, without an established connection
- It is just a thin addition to IP
  - It is vulnerable to the same attacks
  - The attacker can make any change
    - And recalculate the checksum

UDP header format

(RFC 768)

```
 0      7 8     15 16    23 24    31
+--------+--------+--------+--------+
|     Source      |   Destination   |
|      Port       |      Port       |
+--------+--------+--------+--------+
|                 |                 |
|     Length      |    Checksum     |
+--------+--------+--------+--------+
|
|          data octets ...
+---------------- ...
```

TCP

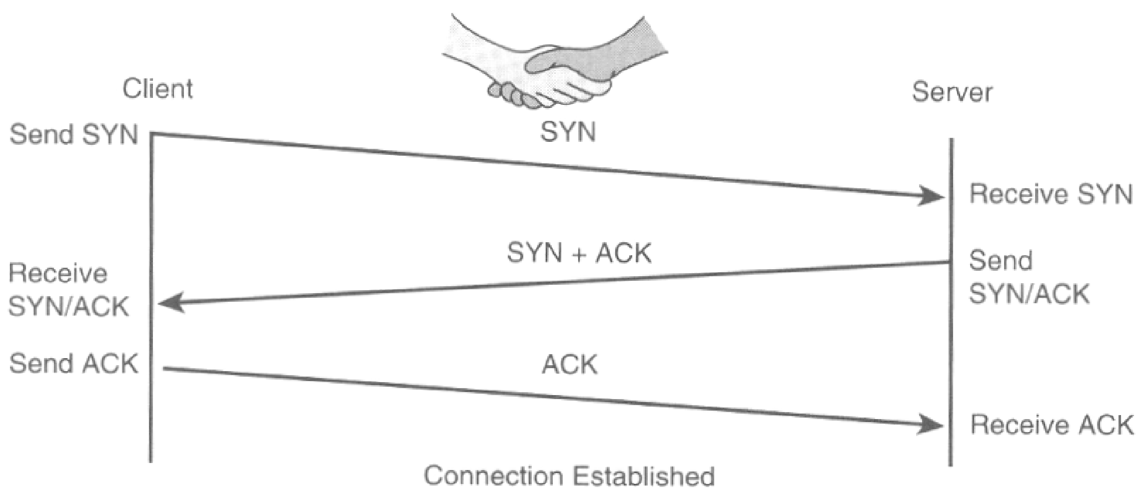
- Transmission Control Protocol
- This protocol can be used establish a connection to send and receive a data stream of bytes
  - Reliable
  - Ordered
  - Error-checked

TCP header format
(RFC 793)

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          Source Port          |       Destination Port        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                        Sequence Number                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Acknowledgment Number                      |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Data |           |U|A|P|R|S|F|                               |
| Offset| Reserved  |R|C|S|S|Y|I|            Window             |
|       |           |G|K|H|T|N|N|                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|           Checksum            |         Urgent Pointer        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                    Options                    |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                             data                              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

TCP/IP 3-way handshake

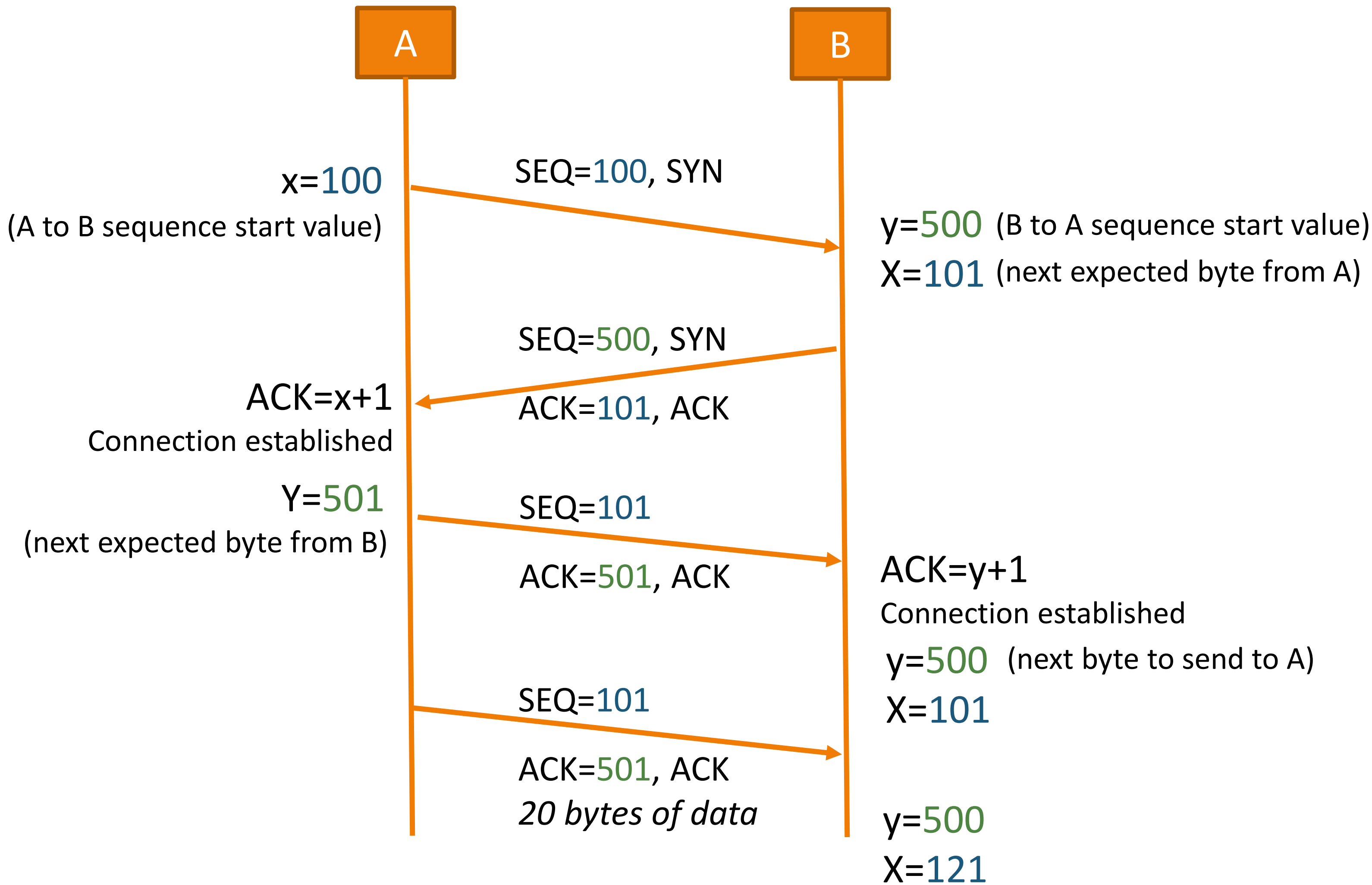
- Process used to make a connection between server and client
- SYN used to initiate and establish a connection
- ACK confirms to the other side that it has received the SYN
  - SYN-ACK is a SYN message from local device and ACK of the earlier packet
- Later, FIN is used for terminating a connection



TCP/IP handshake

- Client sends a SYN request to server with initial sequence number **x**
- Server sends the SYN/ACK packet with its own sequence number SEQ **y** and acknowledgement number ACK **x+1** for client's original SYN packet
  - The ACK indicates the next SEQ number expected from client by the server
- Client acknowledges the receipt of the SYN/ACK packet from server by sending the ACK number **y+1** which will be the next sequence number expected from server
- After the session establishment, packets are sent and received, increasing the sequence and the acknowledgement numbers accordingly

TCP handshake example



TCP connection hijacking

- There are different techniques, depending on the attacker’s capability to intercept communications
  - Full adversary-in-the-middle
  - Weak adversary-in-the-middle
    - De-synchronization
  - No interception
    - Blind

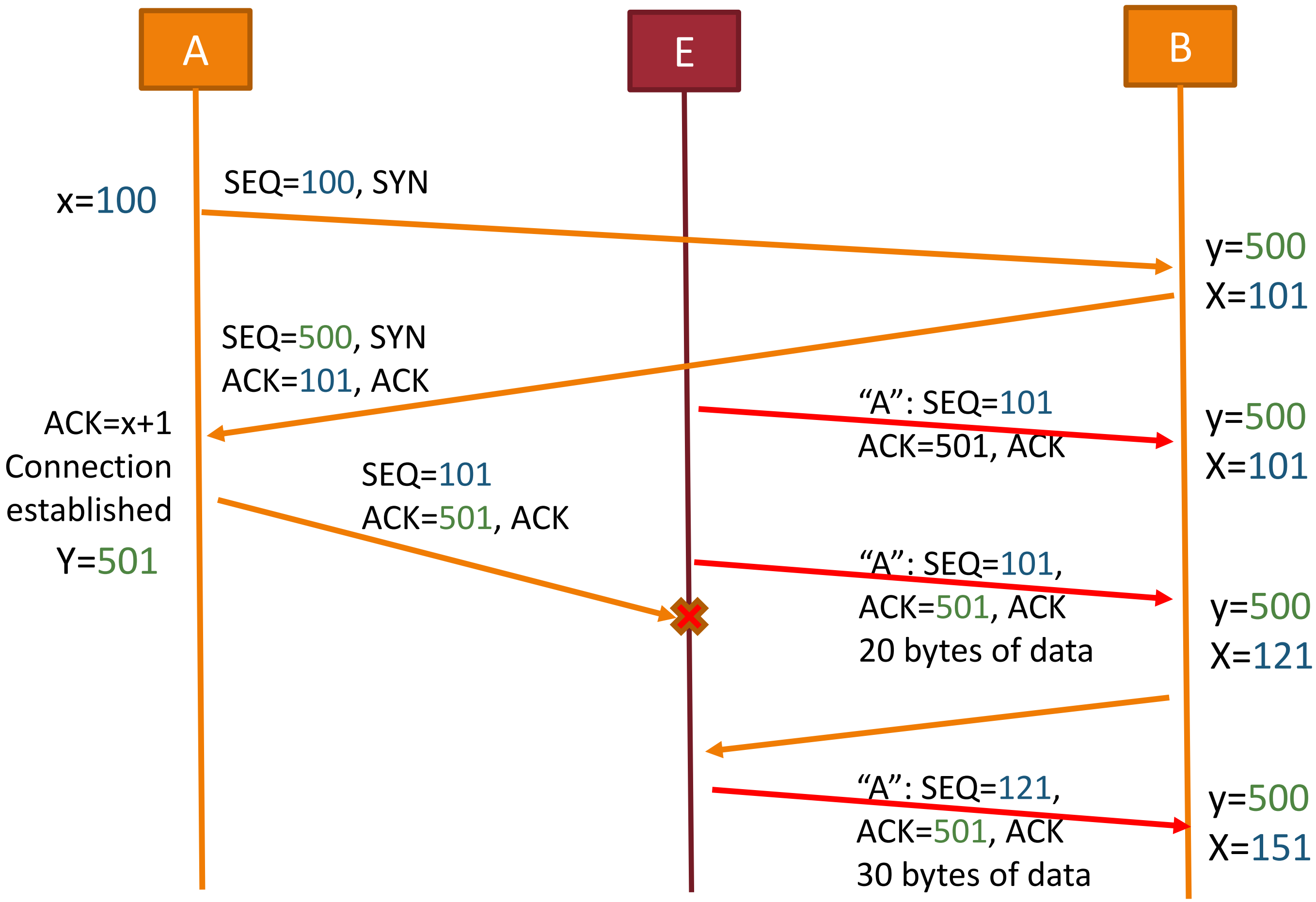
Adversary-in-the-middle TCP hijack

- The attacker is positioned to fully intercept the communication
  - E.g. is at a network gateway
  - E.g. performs ARP poisoning in local network
- The attacker can intercept the sequence numbers and take over the connection
- Example tool:
  - shijack

shijack

- `./shijack eth0 10.0.0.2 53517 10.0.0.1 23`
  - interface you are going to hijack on
  - source IP and port of the connection
  - destination IP and port of the connection
  - [-r] Reset the connection rather than hijacking it
- `Waiting for SEQ/ACK to arrive from the source to the destination`
  - The tool runs and waits for another packet to get a working sequence number
  - As soon as it gets something, it will hijack the connection automatically
- ```
  #Got packet! SEQ = 0xad6e5b8e ACK = 0x5ebaf20d
  #Starting hijack session, Please use ^C to terminate
  #Anything you enter from now on is sent to the hijacked TCP connection
  ```
  - Hijack of telnet session successful! Now we can send everything we want through the session to the server, like shell commands: `mkdir hello`
  ```

TCP adversary-in-the-middle example



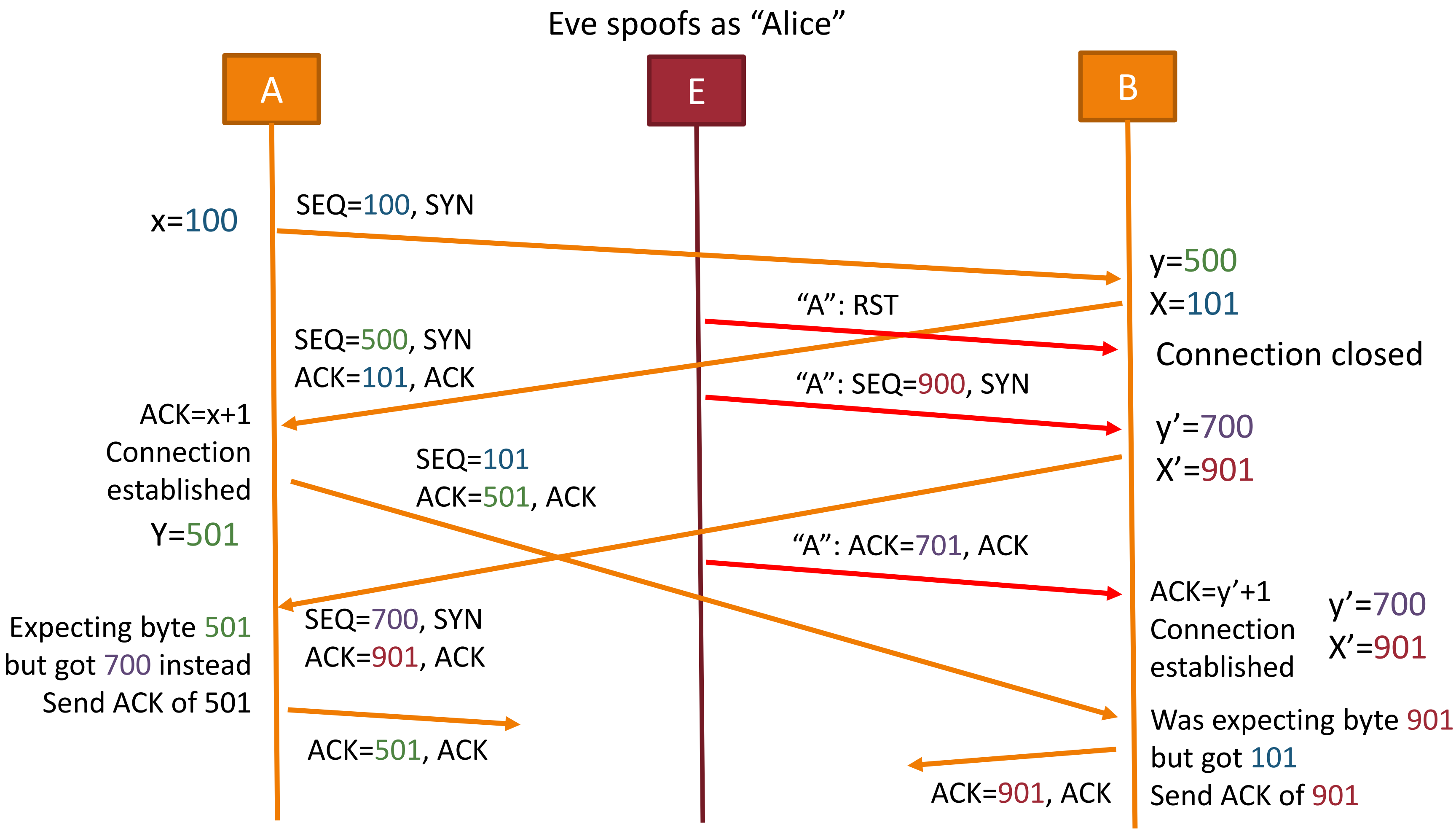
Eve can discard messages (some or all) and
can send some messages as if she were Alice or Bob

Weak adversary-in-the-middle TCP hijack

- Attacker can only eavesdrop and spoof packets
  - The attacker is a man-in-the-middle that CANNOT drop packets
- Attacker must now exploit de-synchronization between hosts
  - Data sent out of the sliding window is discarded by the receiver
- Once the sender and the receiver are desynchronized, only the attacker can create data segments with correct numbers
  - The attacker packets are the ones that are NOT ignored

- How can we forge the de-synchronization?

TCP desynchronization example



Eve knows correct sequence numbers and can send packets that will be accepted

Forging the de-synchronization

- The de-synchronization can be forged during the creation of a TCP/IP connection
  - With a reset and with false acknowledgements
- It can also be done for an already established connection
  - Send blank data to displace sliding windows
    e.g. space chars are usually ignored in a telnet session
- Side-effects: receivers generate many ACK packets trying to acknowledge
  - This "TCP ACK storm" can be used to detect the de-synchronization
  - Meanwhile, the attacker is sending packets that are accepted…

Blind TCP hijack

- The attacker cannot capture return traffic from the host connection
  - The attacker is NOT an adversary-in-the-middle
- The attacker "blindly" sends malicious or manipulated packets
  - Spoofed source IP
  - Guessed sequence number
- The attacker does not receive any confirmation of the desired effect through a packet capture
- For the attack to be successful, the attacker must guess the sequence numbers of the TCP packets
  - Brute force attack on a 32-bit value
  - Unless the initial sequence numbers (ISN) is predictable…
    - *Some older Unix OSes also incremented the ISN with a time dependent algorithm*
    - October 1999 - Microsoft Security Bulletin MS99-046 – Critical
      "Microsoft has released a patch that significantly improves the randomness of the TCP initial sequence numbers (ISNs) generated by the TCP/IP stack in Windows NT 4.0"

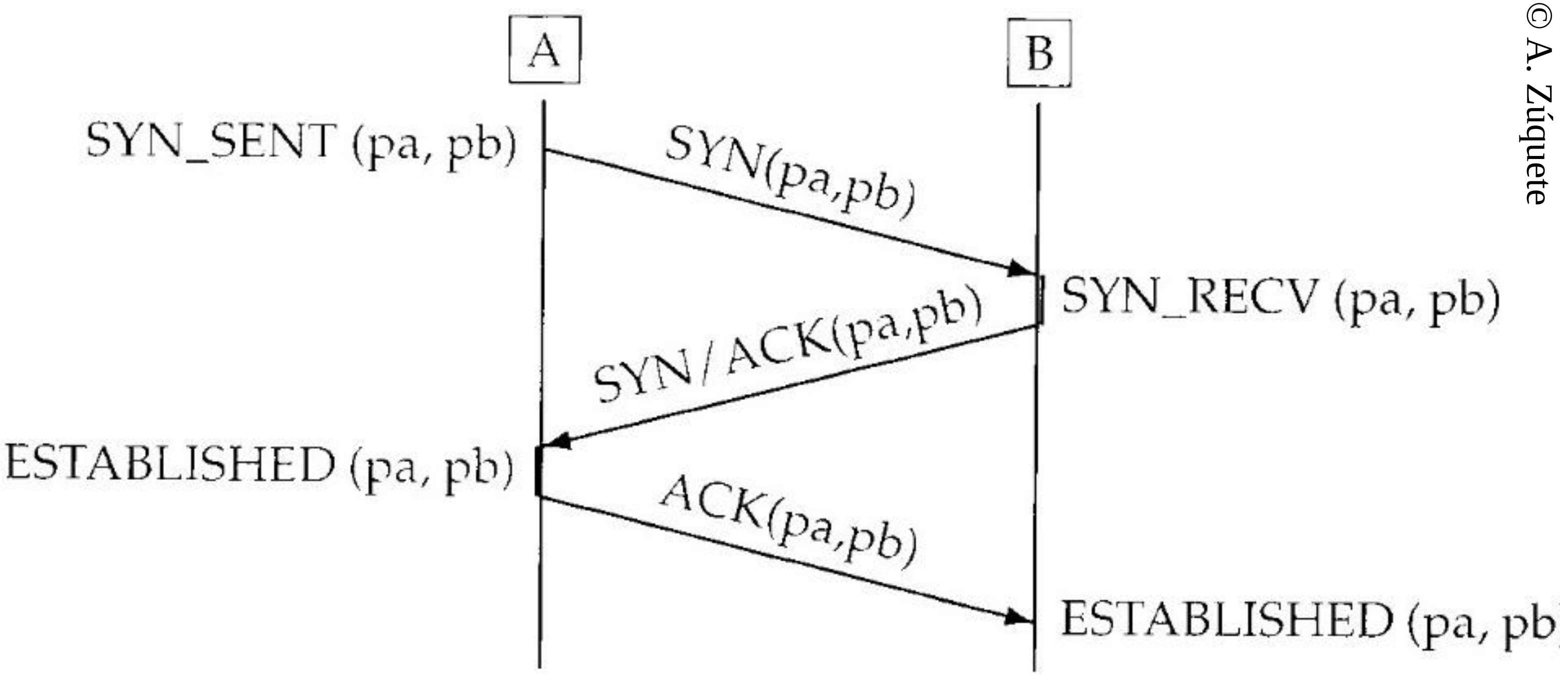
TCP connection hijacking protection

- Random generation of the ISN (initial sequence number)
  - Useful if attacker does not observe the packets
- Avoid any *host-based authentication* based on the IP address
- *Firewalls (we will see more later)*
  - Filter/discard data segments with *source-routing*
  - Use IP masquerading (NAT) for insecure connection nodes
- Protection at the IP level or higher
  - IPsec, TLS, SSH, etc.

TCP DoS attack: SYN flooding (1/2)

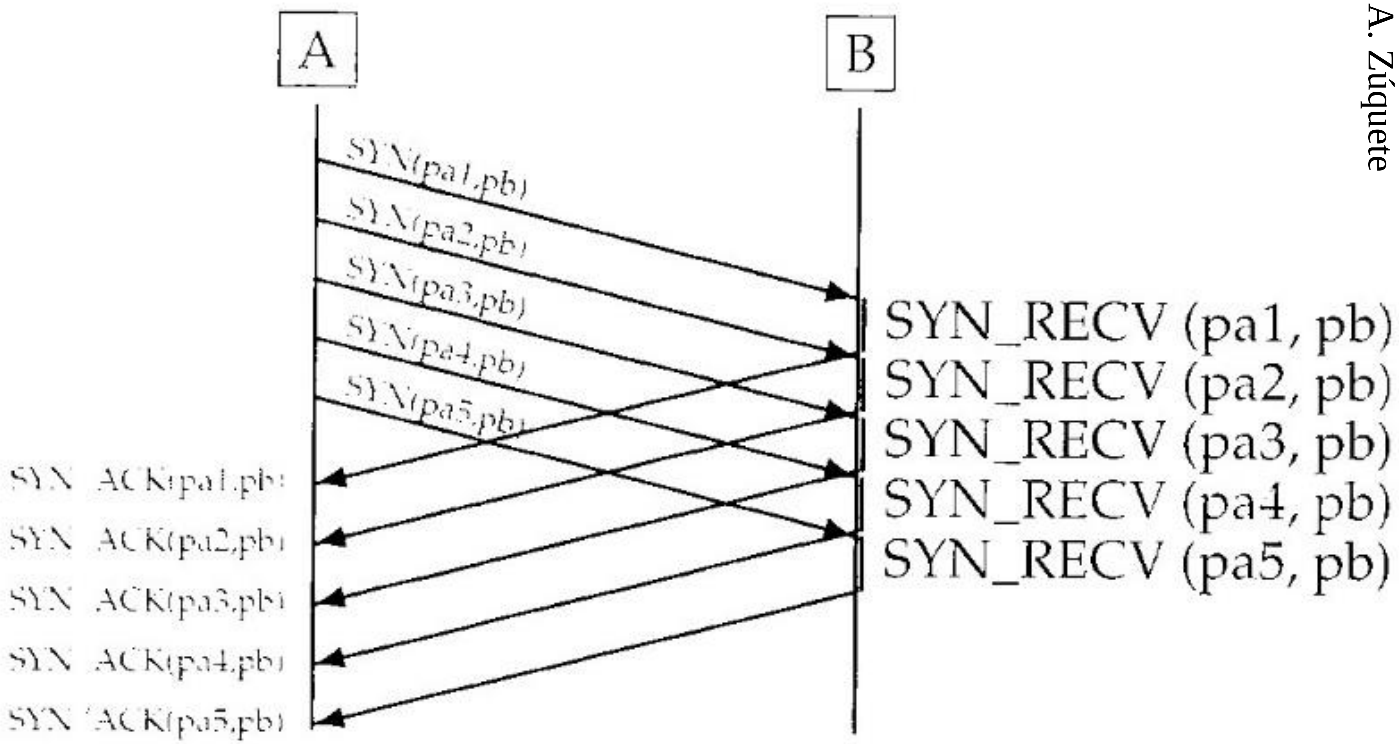
- Consists of overloading a host with incomplete TCP/IP connection requests
  - X → A: SYN
  - A → X: SYN+ACK
  - X → A: ACK ------ missing
- Typically the attacker uses IP *spoofing*
  - Fake the sender IP
  - Often TCP is insensitive (when in the SYN_RECVD state) to ICMP error messages: "host unreachable" or "port unreachable"
  - Forging one or more unused IP addresses
    - Easy to block temporarily
  - Forging random IP addresses
    - Harder to block

TCP connection (again)



pa / pb - ports

SYN flooding attack



pa1..5 and pb are port numbers

TCP DoS attack: SYN flooding (2/2)

- Explored vulnerabilities
  - No authentication in the SYN segments
  - The server needs to reserve more resources that the client/attacker
- Impact on the attacked machine
  - Storage of the connection requests until they are eliminated by timeout
    - TCP connection in the SYN_RECVD state
  - The amount of connection requests per port are limited:
    - The subsequent requests are discarded
    - Correct requests may be discarded due to the existence of false connection requests

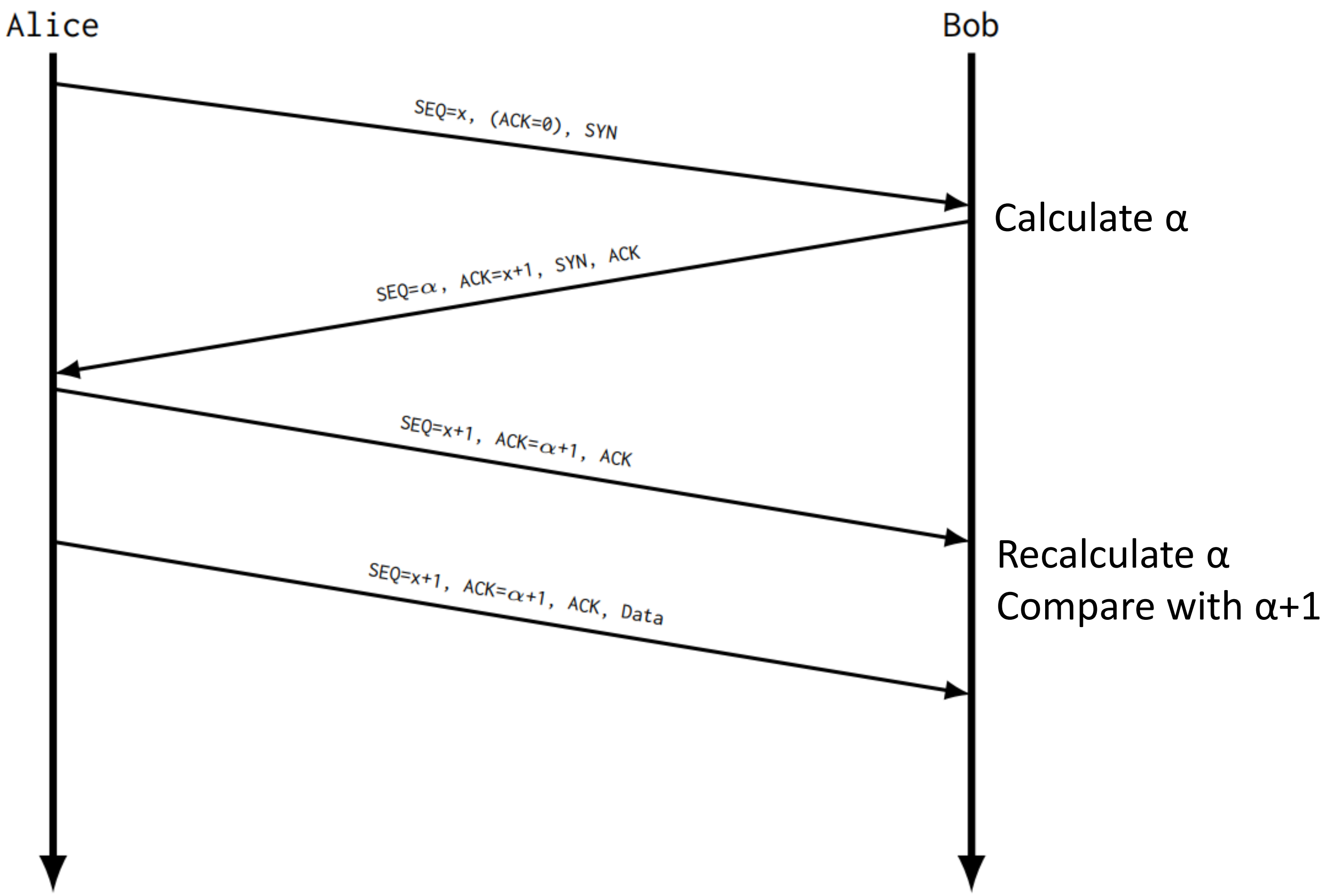
SYN flooding mitigation

- No definite solution for IPv4
- Modifying TCP for the servers
  - Bigger request queues, lower timeouts
  - Random Drop
  - **SYN cookies**
- Cooperation with firewall and attack detector

SYN flooding mitigation with SYN cookies

- SYN cookie: choice of the initial seq number by Bob
  - Bob generates the initial sequence number α such as:
    - α = h(K, SSYN)
    - h is a one-way hash function
    - K: a secret key known only by the server
    - SSYN: source IP address of the SYN packet
  - At arrival of the ACK message, Bob calculates α again
    - If it knows K and received the source IP
    - Then, it verifies if the ACK number is correct
  - If yes, it assumes that the client has sent a SYN message recently and it is considered as normal behavior

Handshake with SYN cookie
(RFC 4987)



SYN cookies tradeoffs

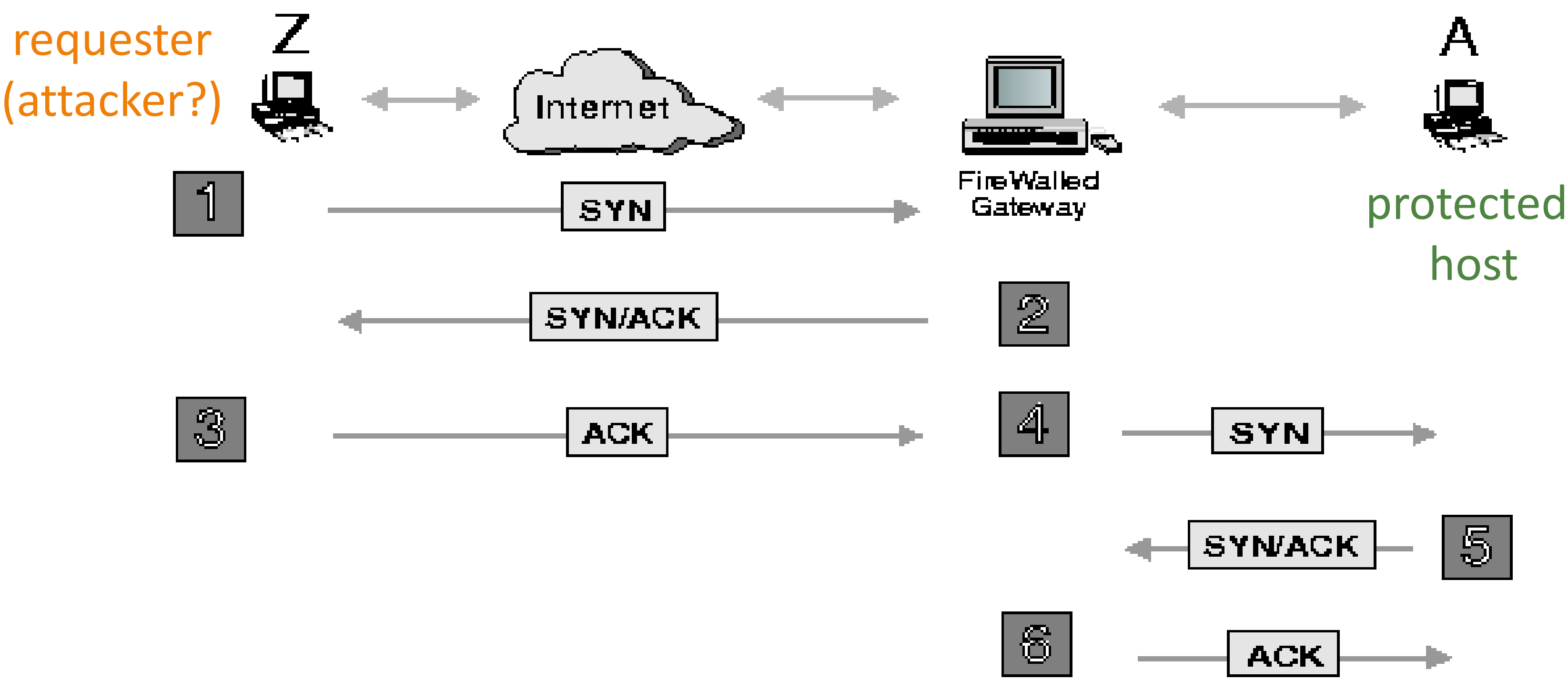
- Advantages:
  - Server does not need to allocate resources after first SYN packet
  - Client does not need to be aware that server is using SYN cookies
  - SYN cookies does not require changes in the specification of the TCP protocol
- Disadvantages:
  - Calculating α may be CPU consuming
    - Moved the vulnerability from memory overload to CPU overload
  - TCP options cannot be negotiated e.g. large window option
    - Use SYN cookies only when an attack is assumed
  - ACK/SEQ number are only 32 bit long
    - May be vulnerable to cryptoanalysis
    - The secret needs to be changed regularly

SYN flooding mitigation with firewall

- Cooperate with firewalls and attack detectors
  - Handshake relay
    - Firewall stands in front of server and protects it until the handshake is complete
  - Gateway
    - Firewall keeps the connection alive on server and terminates it if the client leaves the connection open but without traffic

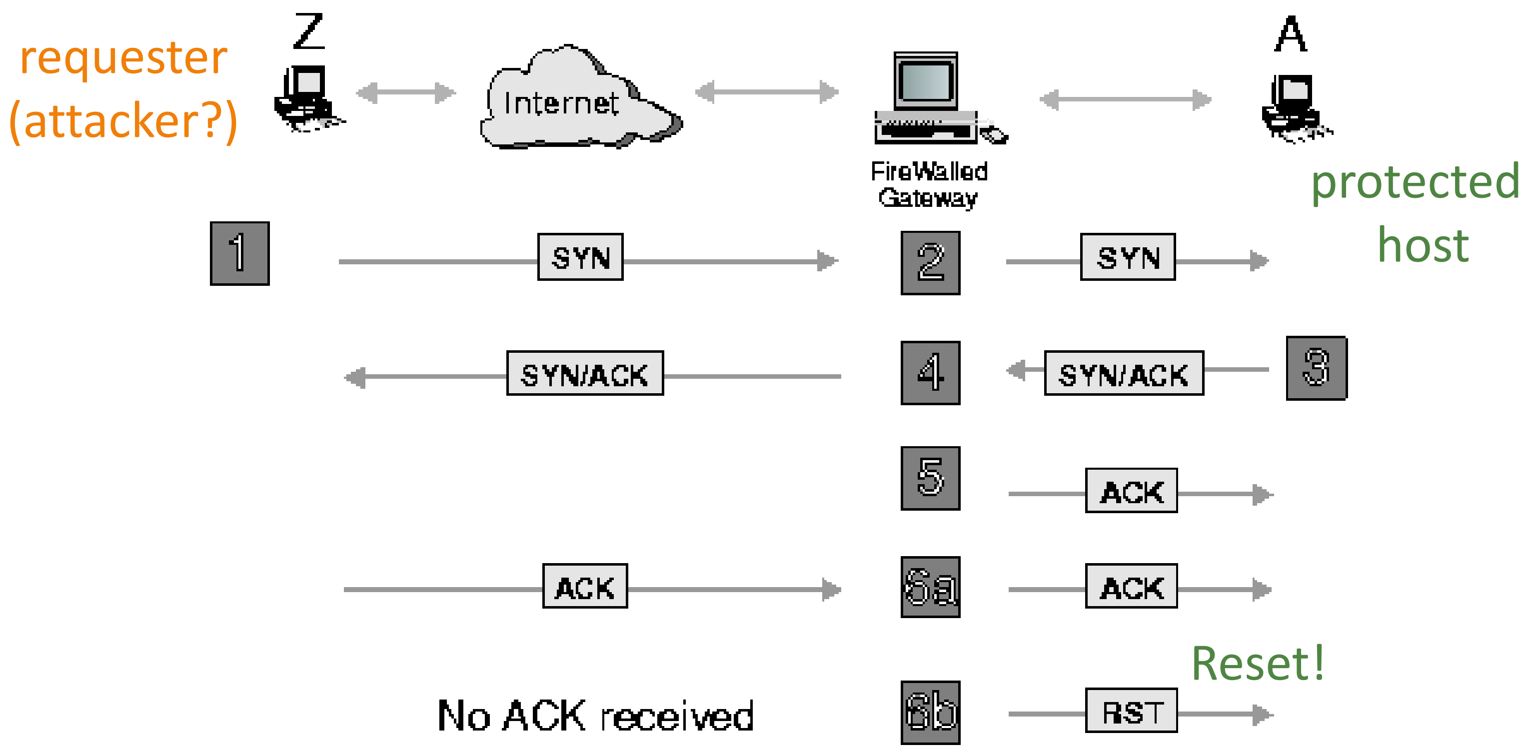
Firewall working as Handshake Relay

- Solution:
  - Firewall does handshake with requester
  - If handshake OK,
    then firewall does it with protected host



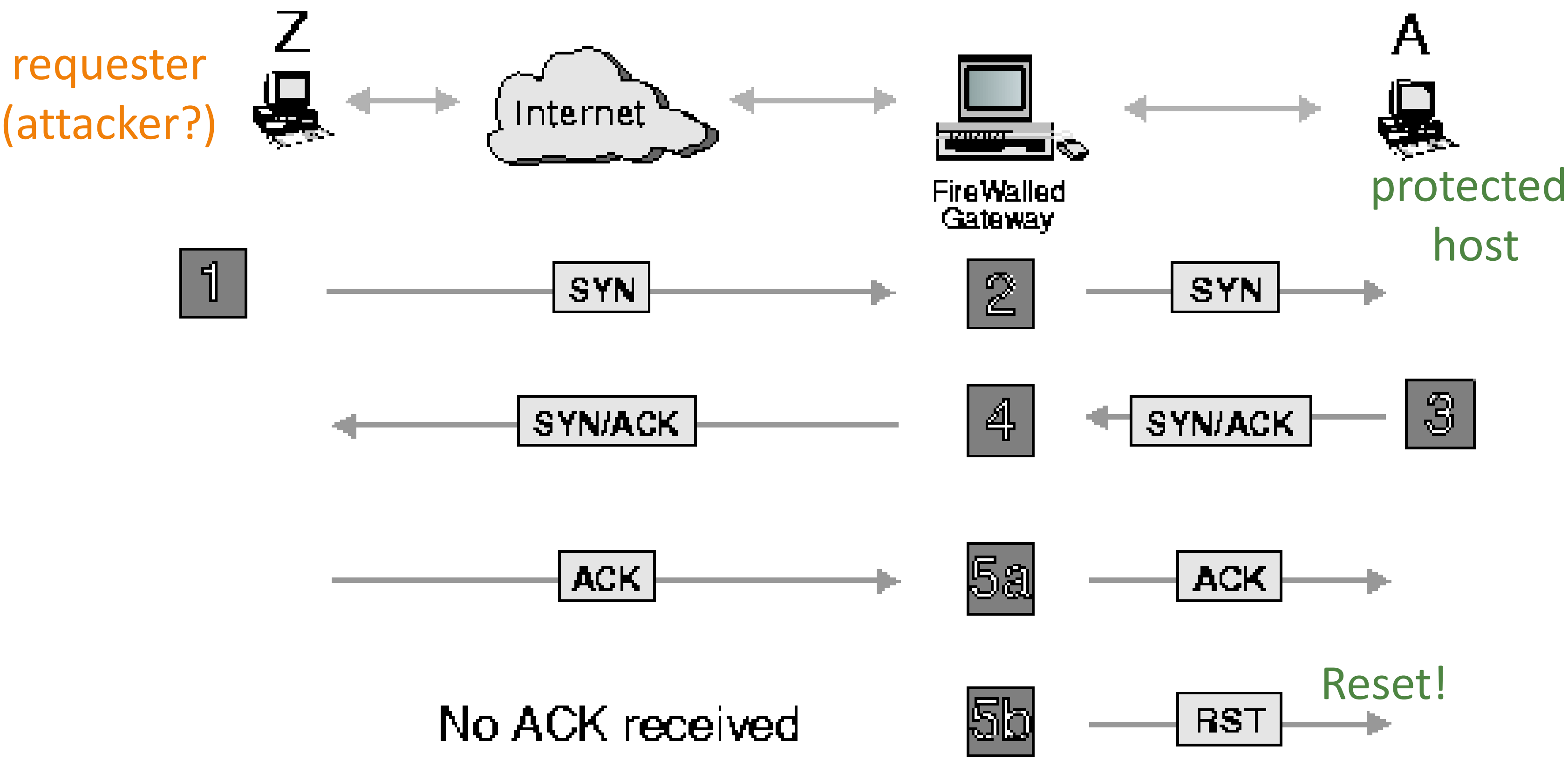
Firewall working as Gateway

- Solution:
  - Firewall does handshake with both and finishes handshake with the protected host
  - If handshake with requester not OK, resets the other



Firewall – Passive gateway

- Solution:
  - Similar but just forwards the requester's packets



Roadmap

- **Network vulnerabilities**
  - Physical layer
  - Data link layer
  - Network layer
  - Transport layer
  - **Application layer**
- Network security models

Application layer

(Layer 7) Application Layer

- Topics:
  - DNS – critical infrastructure service
  - Remote Code Execution
    - Dynamic Code Execution
    - Memory unsafety

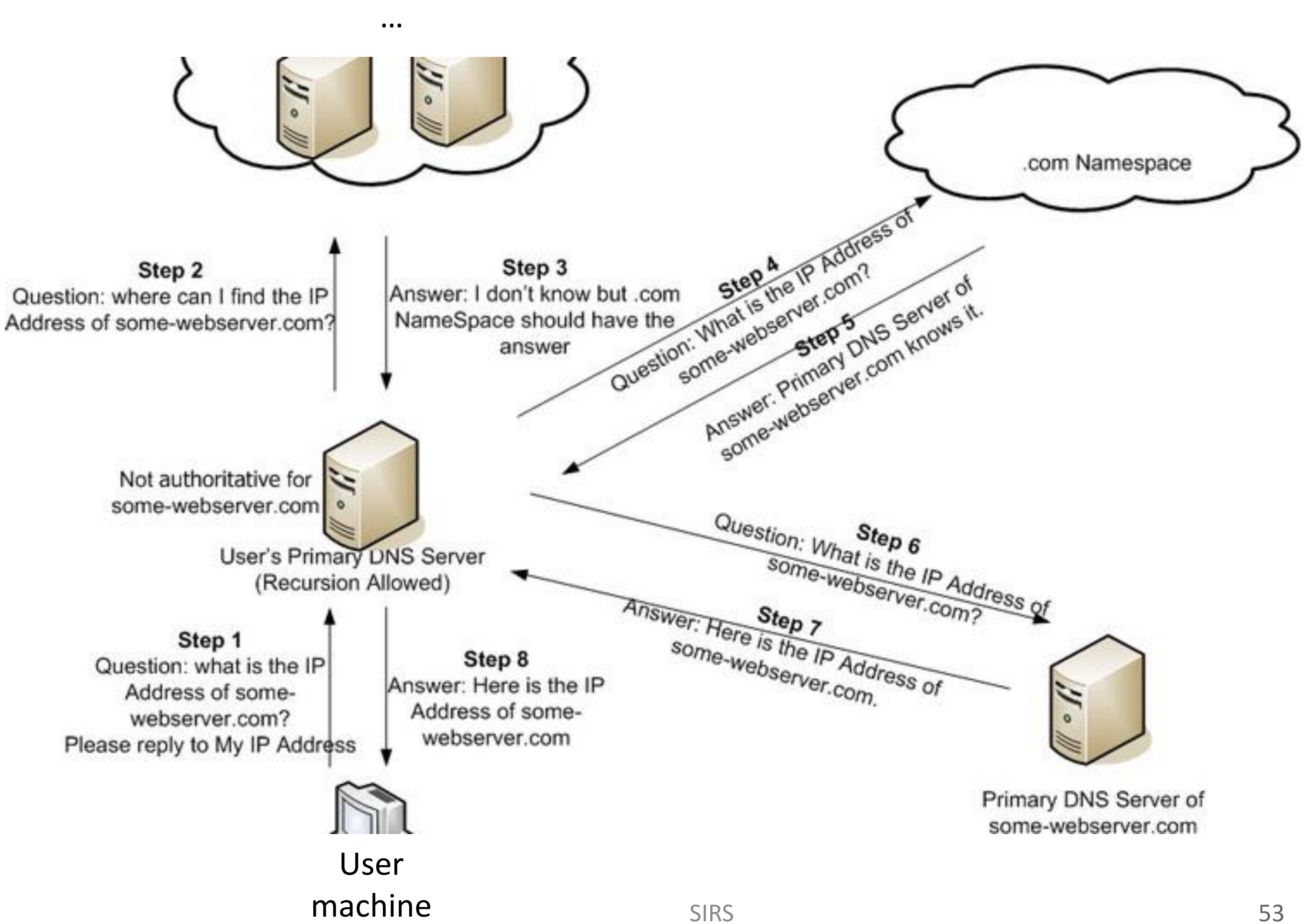
DNS (Domain Name System)

- Entities
- Resource records
- Threats
  - Kaminsky attack
- DNSSEC

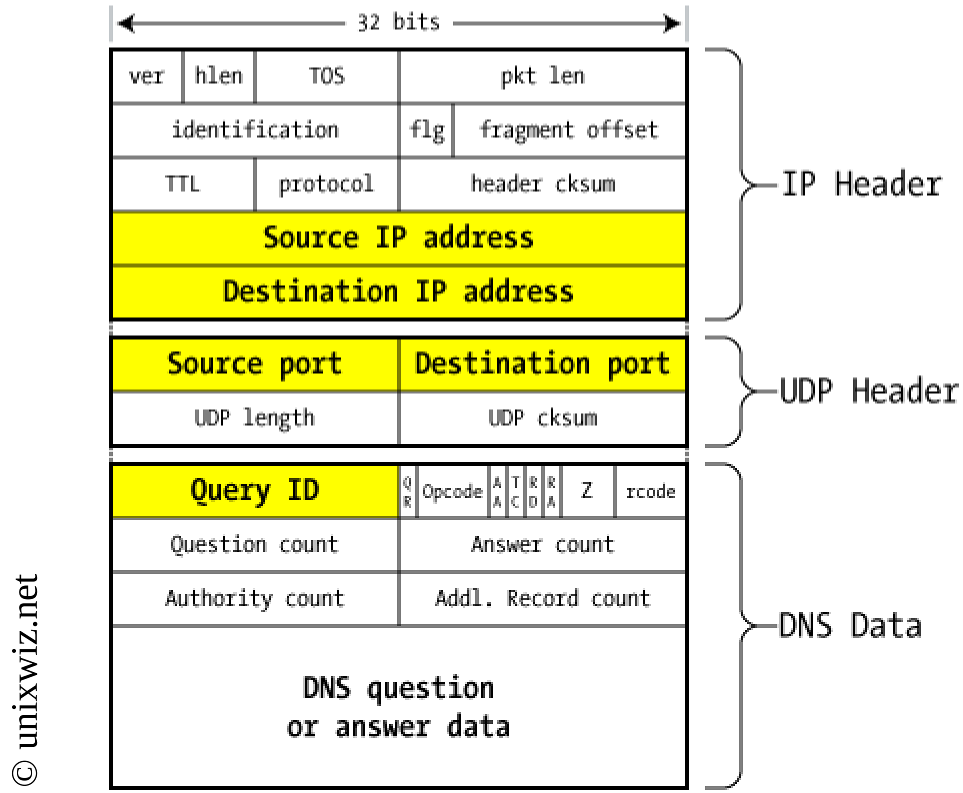
DNS in action

- Translate Domain Names to IP addresses
  - www.tecnico.ulisboa.pt ⇨ 193.136.128.66
- Reverse Translation
  - 66.128.136.193.in-addr.arpa ⇨ www.tecnico.ulisboa.pt
- Mail Server Localization
  - Ricardo.Chaves@tecnico.ulisboa.pt ⇨ smtp.tecnico.ulisboa.pt
- Other name translations

DNS resolving steps

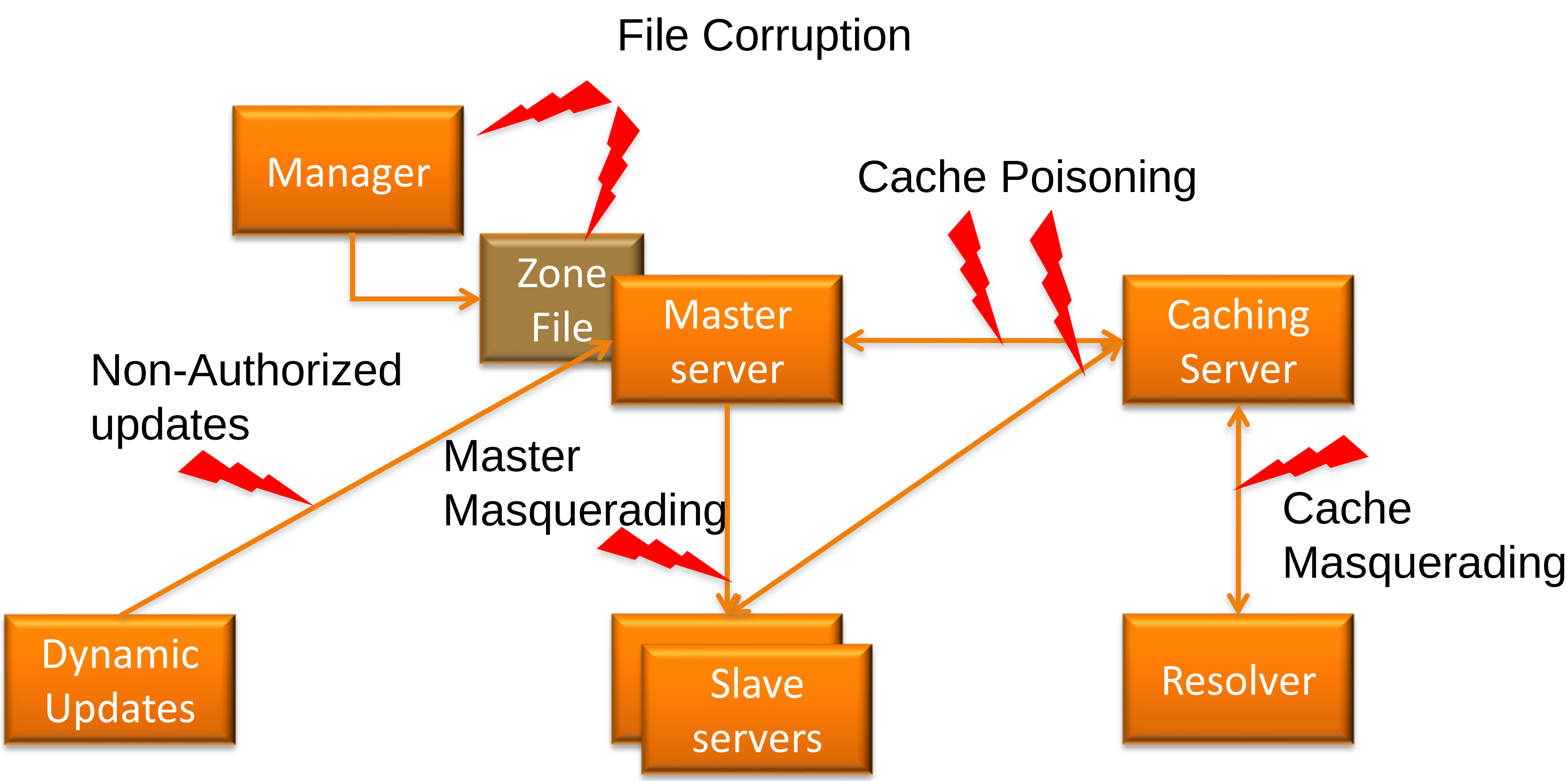


DNS Message



DNS packet on the wire

DNS Architecture Threats



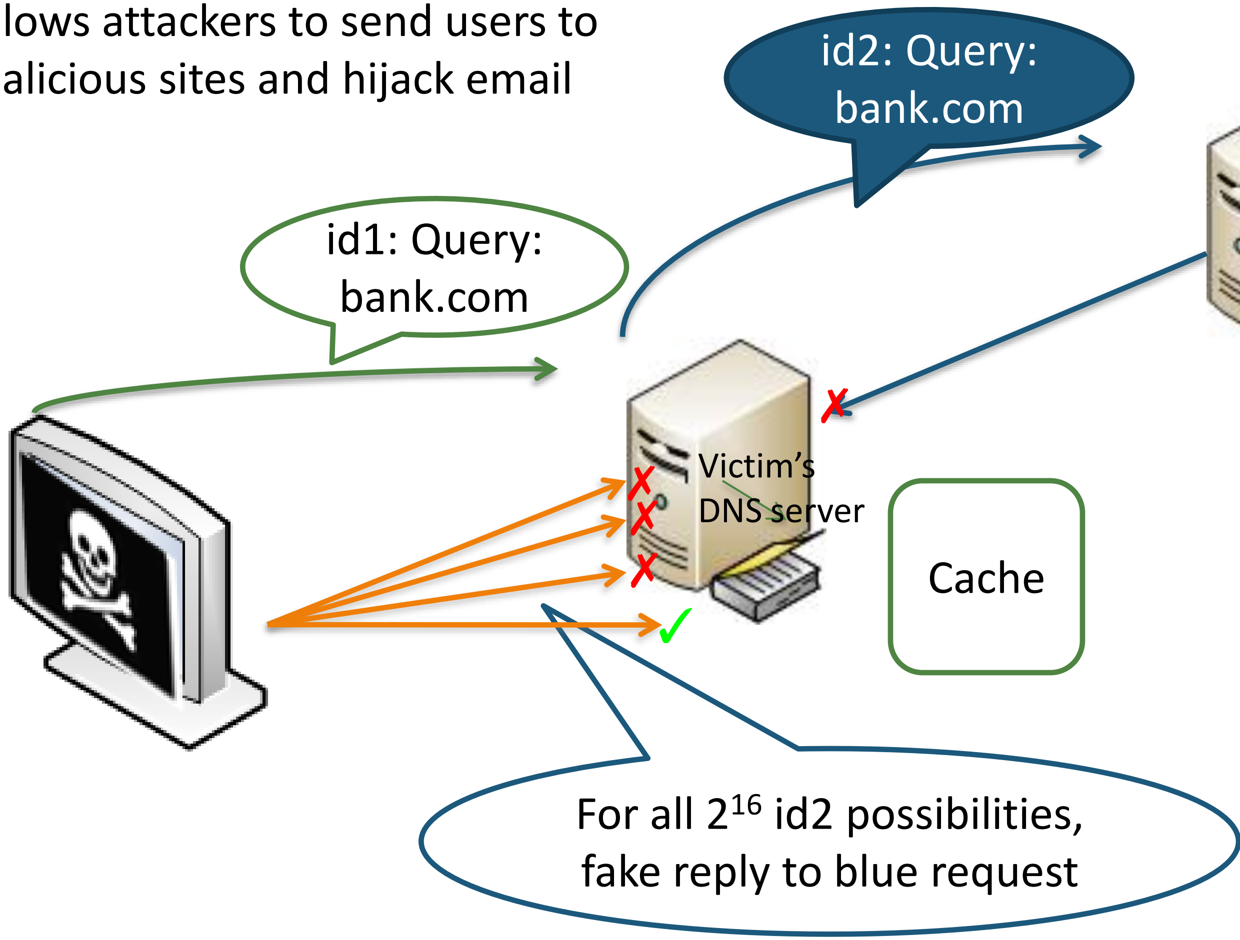
Kaminsky Attack

- Feb/2008 – Dan Kaminsky reports the problem
- 8/Jul/2008 – Patch for several systems
- 21/Jul/2008 – Public knowledge
- 8/Aug/2008 – Details on BlackHat
- 28/Aug/2008 – Memo for adoption of DNSSEC in .gov
- .pt – https://www.dns.pt/pt/seguranca/dnssec/

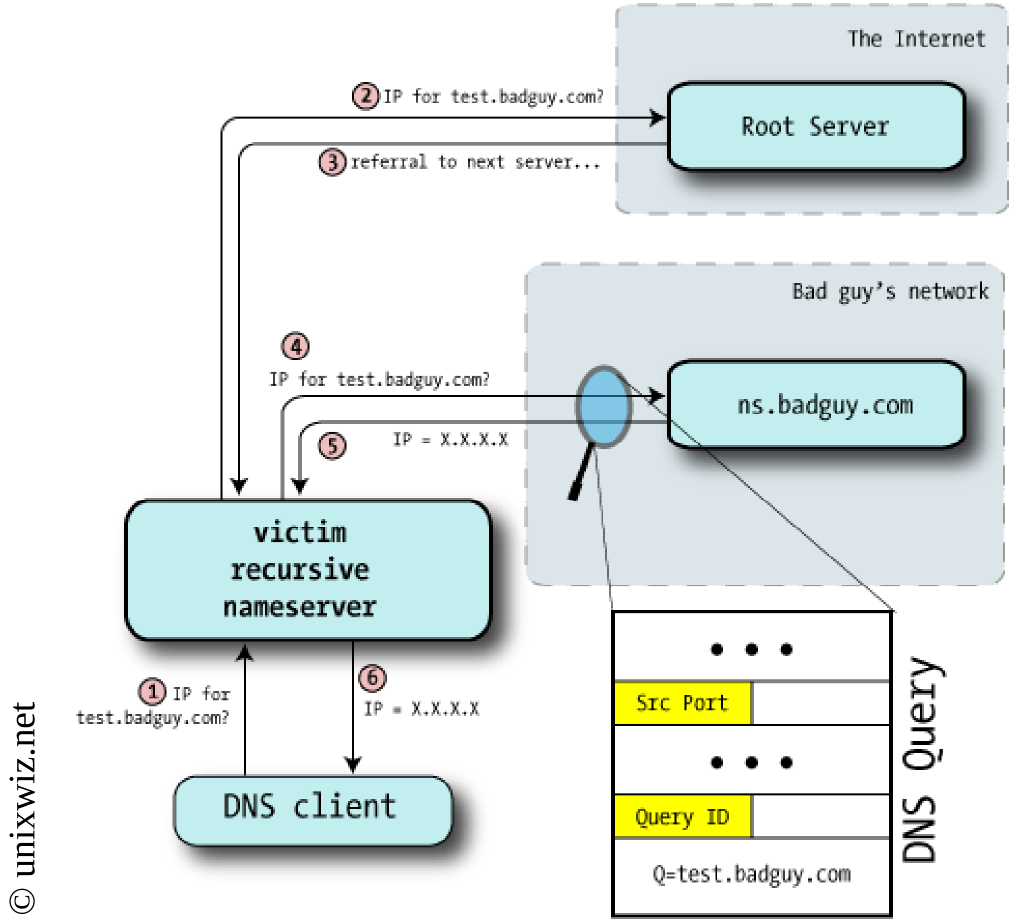
Dan Kaminsky
1979-2021

Kaminsky attack (cache poisoning)

Allows attackers to send users to malicious sites and hijack email



The attack is successful if it can guess the Query ID value

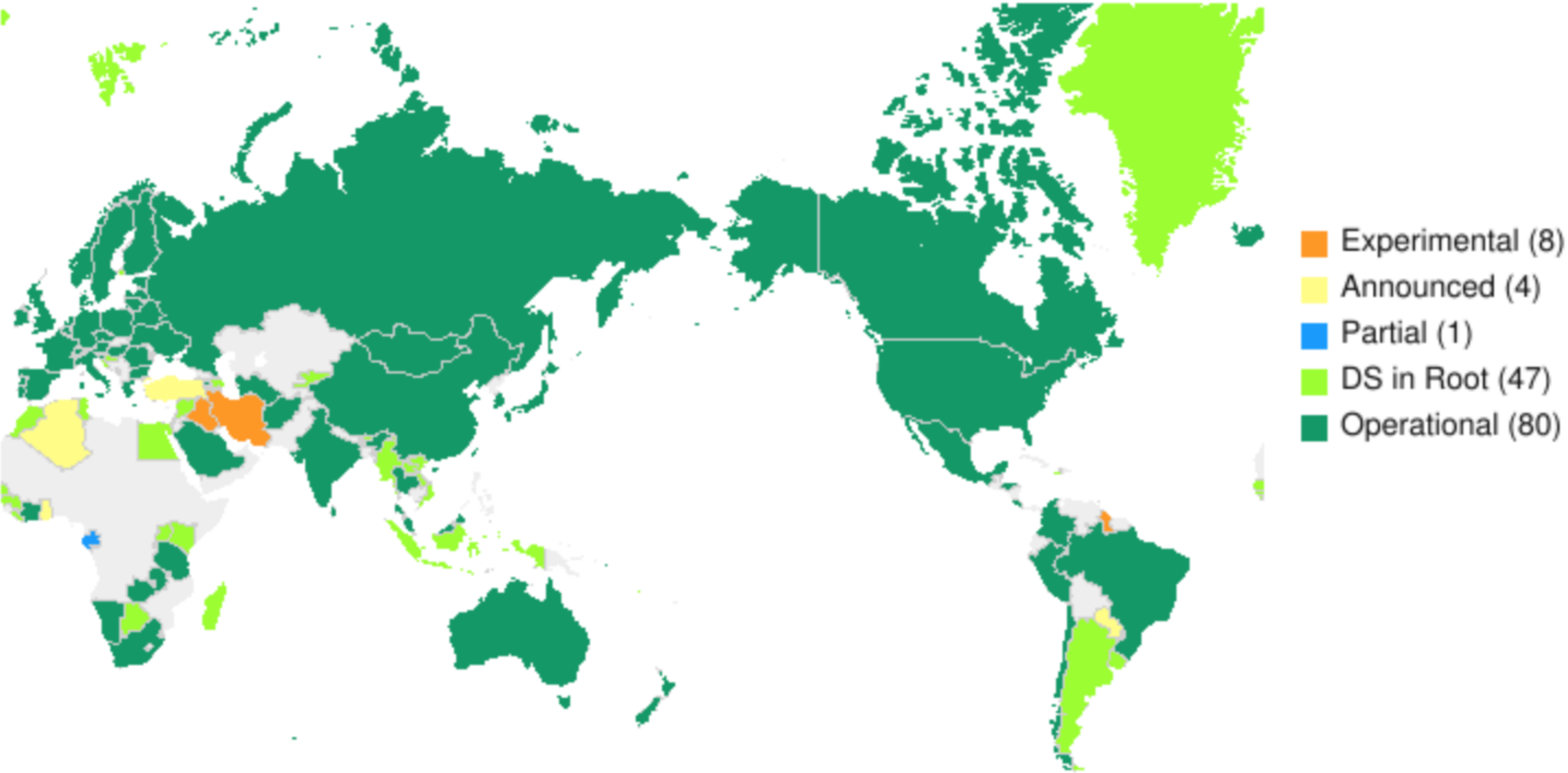


Current solution: request takes
random source port and **random query id**entifier

DNSSEC

- **DNSSEC** – DNS with digitally signed responses
  - Each zone has its own key-pair for signing
    - Responses can be validated using the respective public key
  - Public Keys are published in the DNS itself
    - As a DNSKEY Resource Record
    - One needs to get the public keys from a trusted source
      - Ideally only for the parent zones of the DNS hierarchy
  - DNSSEC provides integrity and authenticity for RRs of the signed zones
    - Does not provide more reliability, confidentiality or protection against DoS

ccTLD DNSSEC status jan. 2019



More about layer 7

- DNS is at the application layer, but it is an infrastructure service
- We also must be concerned with the application code exposed over the network

Remote Code Execution (RCE)

- RCE is a class of software security vulnerabilities
  - Much more about these in SSoft course
- RCE vulnerabilities allow a malicious actor to execute any code of their choice on a remote server machine
  - Arbitrary code execution
  - Over LAN, WAN, or **Internet**
- Exploits:
  - Dynamic code execution
  - Memory unsafety

Dynamic code execution

- Most programming languages have some way to generate code in run-time and execute it
  - E.g., parse a string as code and execute it
  - Powerful programming concept, can be very convenient
- However, a malicious actor can abuse it
  - Often, generated code is based on some user input
- If the user inputs are not vetted, then that code will be executed on the target machine
- Examples:
  - PHP code injection
  - SQL injection

Memory unsafety

- Software may have flaws when managing memory
  - Compiler, interpreter, operating system kernel or libraries
  - Virtual machines too
- Buffer overflow
  - Typically, program accepts input that is bigger than the allocated buffer
  - Memory following the buffer is overwritten
  - Program may “jump” to a different function
- An attacker can carefully craft the requests to a server to cause buffer overflow
  - Modify system memory on the affected machine
  - Cause execution of arbitrary code

Vulnerabilities inside application code

- *Dynamic code execution*:
  - using PHP (Code Injection)
  - using SQL (SQL Injection)
  - using JavaScript (XSS – Cross-site scripting)

- *Memory unsafety*:
  - using C (Overflows)

PHP – Eval Injection

vuln.php

```
<?php
$var = “value”;
$v = $_GET[‘argument'];
eval("\$var = $v;");
?>
```

```
http://victim.com/vuln.php?argument=1;phpinfo()
```

```
eval("value = 1; phpinfo();");
```

Attack effect: run the phpinfo() function

PHP – Local File Inclusion

vuln.php

```
<?php
$page = $_GET[page];
include($page.php);
?>
```

```
http://victim.com/vuln.php?page=../../../../../
../../etc/passwd%00
```

Attack effect: get the content of file /etc/passwd

How to prevent code injection?

- Avoid using data as code as much as possible
- **Sanitize inputs**
  - Remove illegal characters
  - PHP now provides native filters that you can use to sanitize the data
    - Such as e-mail addresses, URLs, IP addresses, etc…

Problem goes beyond PHP

- These attacks are not exclusive to PHP programming
- They can be done whenever inputs are parsed and interpreted as code

SQL Injection

Java code

```
SQLQuery = "SELECT Username FROM Users WHERE Username = '" +
strUsername + "' AND Password = '" + strPassword + "'"
strAuthCheck = getQueryResult(SQLQuery)

if (strAuthCheck.equals("")) bAuthenticated = false; else bAuthenticated = true;
```

Login: admin
Password: ' OR '1' = '1

SELECT Username FROM Users WHERE Username = 'admin' AND Password = '' OR '1' = '1'

Attack effect: login as user admin without knowing the password

Login: ' OR '1'='1' ; DROP TABLE Users --
Password: *does not matter!*

SELECT Username FROM Users WHERE Username = '' OR '1' = '1' ; DROP TABLE Users -- ' AND Password = '???'

Attack effect: delete table Users

How to prevent SQL Injection

- Best solution is to use **prepared statements** with **parameters** that are always properly sanitized and treated as data

```
...
Set cmd = CreateObject("ADOBD.Command")
cmd.Command = "select Username from Users where
Username=? and Password=?"

Set param1 = cmd.CreateParameter(...)
param1.Value = strUsername
cmd.Parameter.Append param1

Set param2 = cmd.CreateParameter(...)
param2.Value = strPassword
cmd.Parameter.Append param2

Set strAuthCheck = cmd.Execute
...
```

Problem goes beyond SQL

- Similar attacks can be made with other database languages, e.g.:
  - MongoDB (NoSQL)
  - Graph query language (Neo4J)
- Input values should be escaped and never used as statements

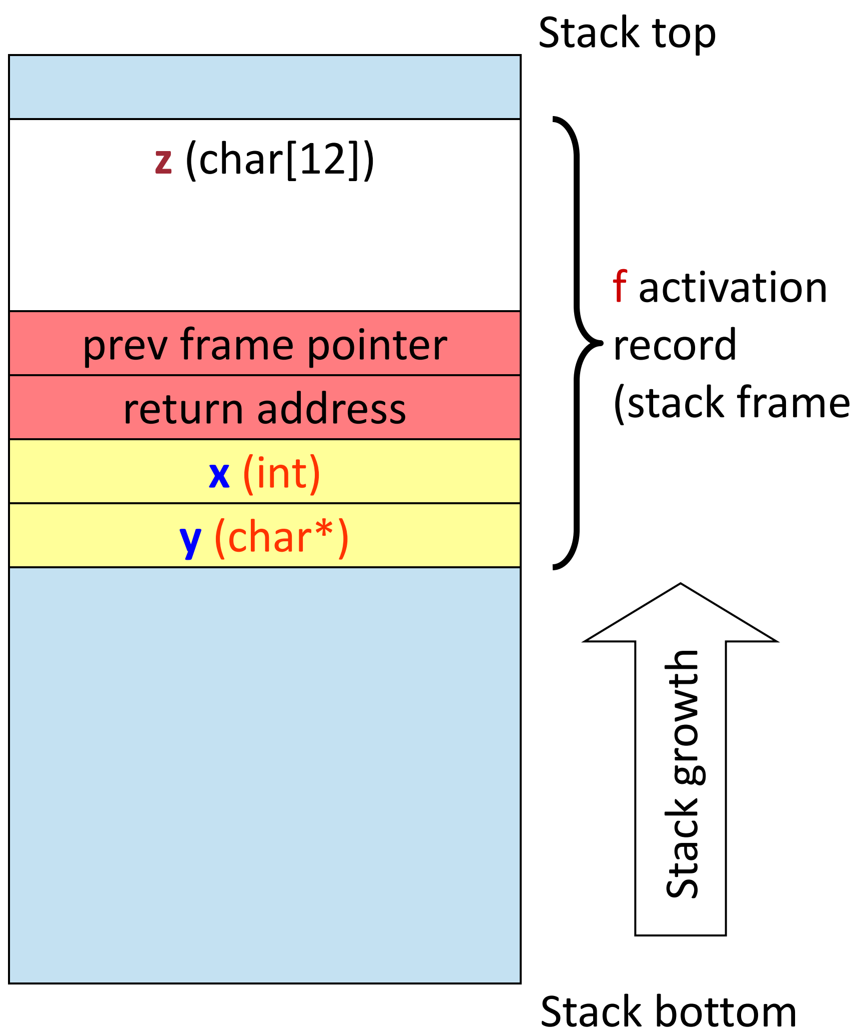
C language and overflows

- **Stack smashing**
- Heap overflow
- BSS overflow
- Print and format overflow

Overflows: *Stack smashing*

Standard usage:

```
void f ( int x, char * y )
{
    char z[12];
    sprintf (z, "%d %s", x, y );
    write ( 2, z, strlen(z) );
}
```



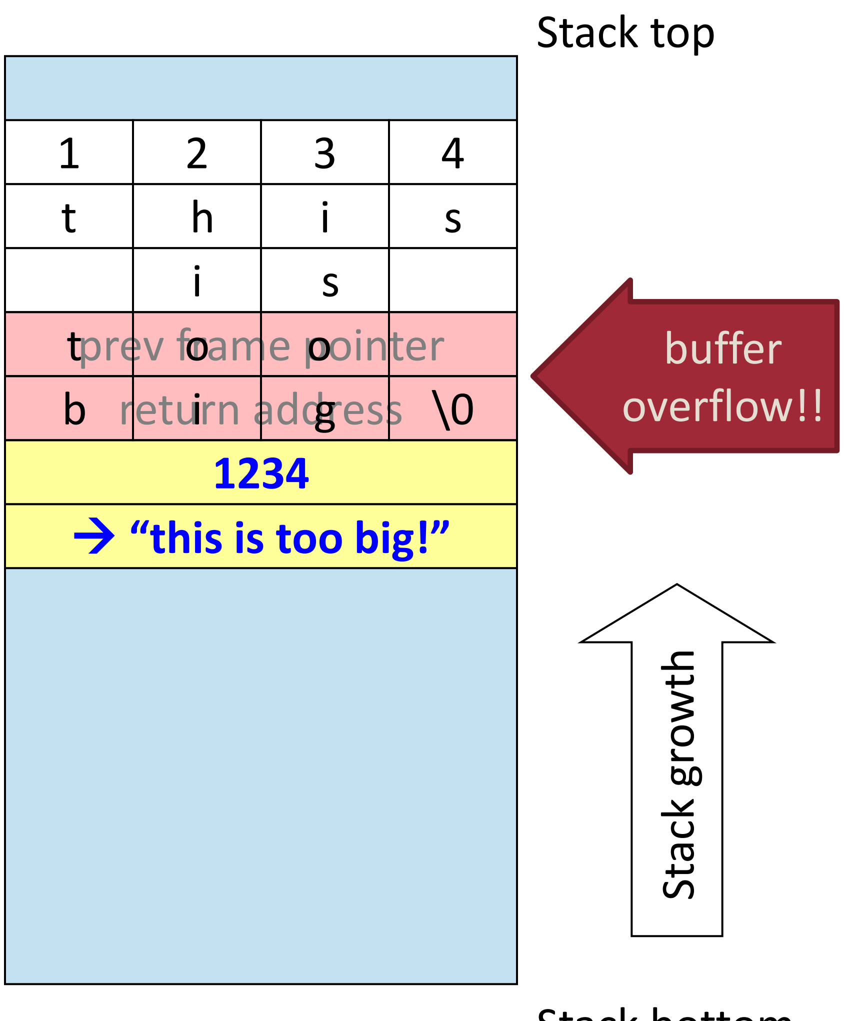
Overflows: *Stack smashing*

```
void f ( int x, char * y )
{
    char z[12];
    sprintf (z, "%d %s", x, y );
    write ( 2, z, strlen(z) );
}

{
    ...

    f (0x1234,"this is too big");

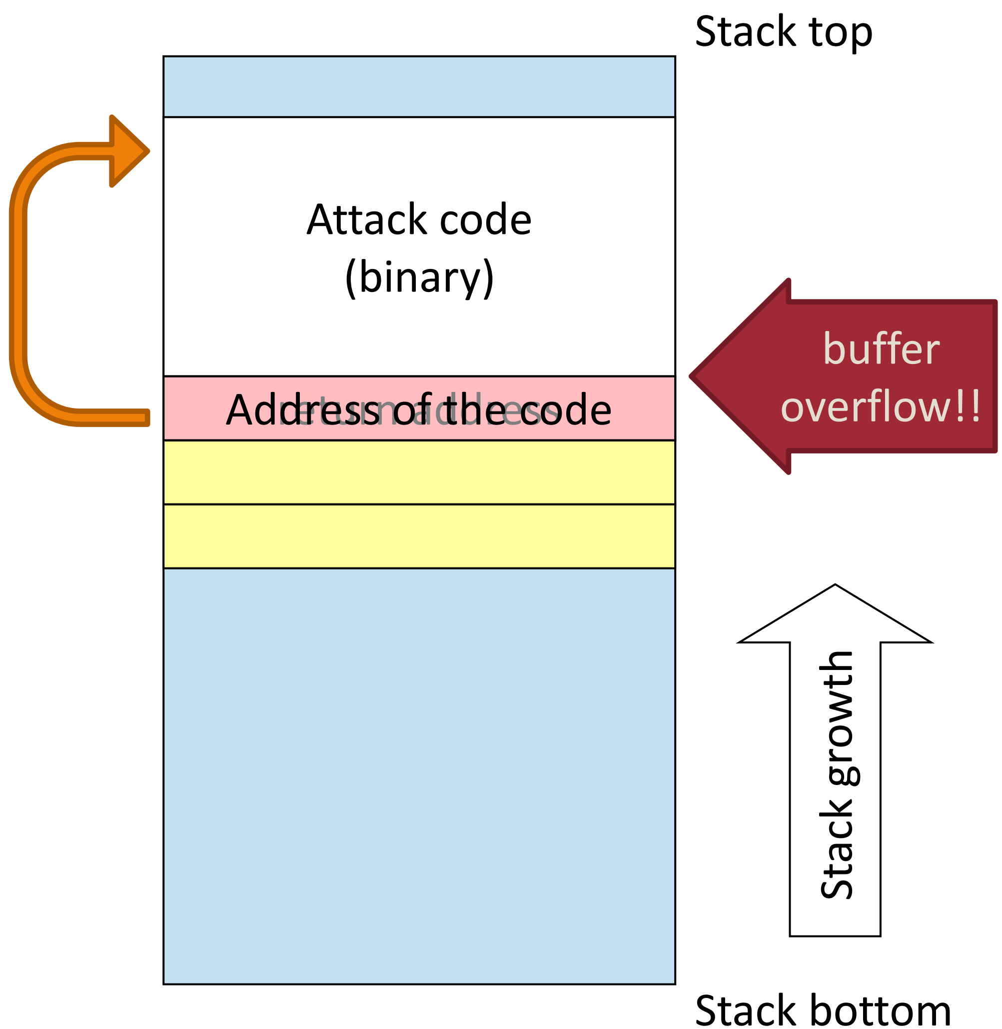
    ...
}
```



Overflows: *Stack smashing*

Code injected by the attacker is executed!

This is worst that can happen…



C/C++ memory unsafety

- Most buffer overflow attacks target C or C++ code since these languages do not have built-in buffer size checks
- So, is this only a concern for C/C++ developers?
  - No, because most other languages end up using C/C++ libraries under the surface
  - Also makes them vulnerable to this kind of attack
- Examples
  - Python calling C libraries
  - Java Native Interface
  - Node.js engine and add-ons

How to prevent stack overflows?

- Non-executable stack
- Randomization of addresses
- Canaries for detecting tampering

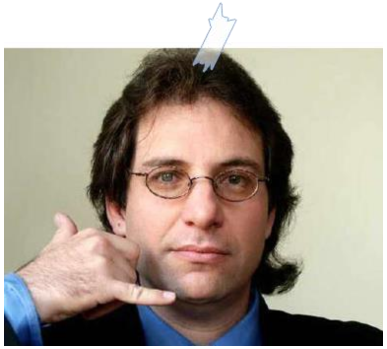
(Detailed in SSof course)

Is there a layer 8 ?

- Layer 8 *informally* refers to the "user"
  - Users are often the **weakest link** in security
- Considering layer 8 explicitly can allow IT administrators to define processes to:
  - Identify users
  - Control Internet activity of users in the network
  - Set user-based policies
  - Generate reports by user
- We can even add more layers:
  - Layer 9: The organization
  - Layer 10: Government or legal compliance

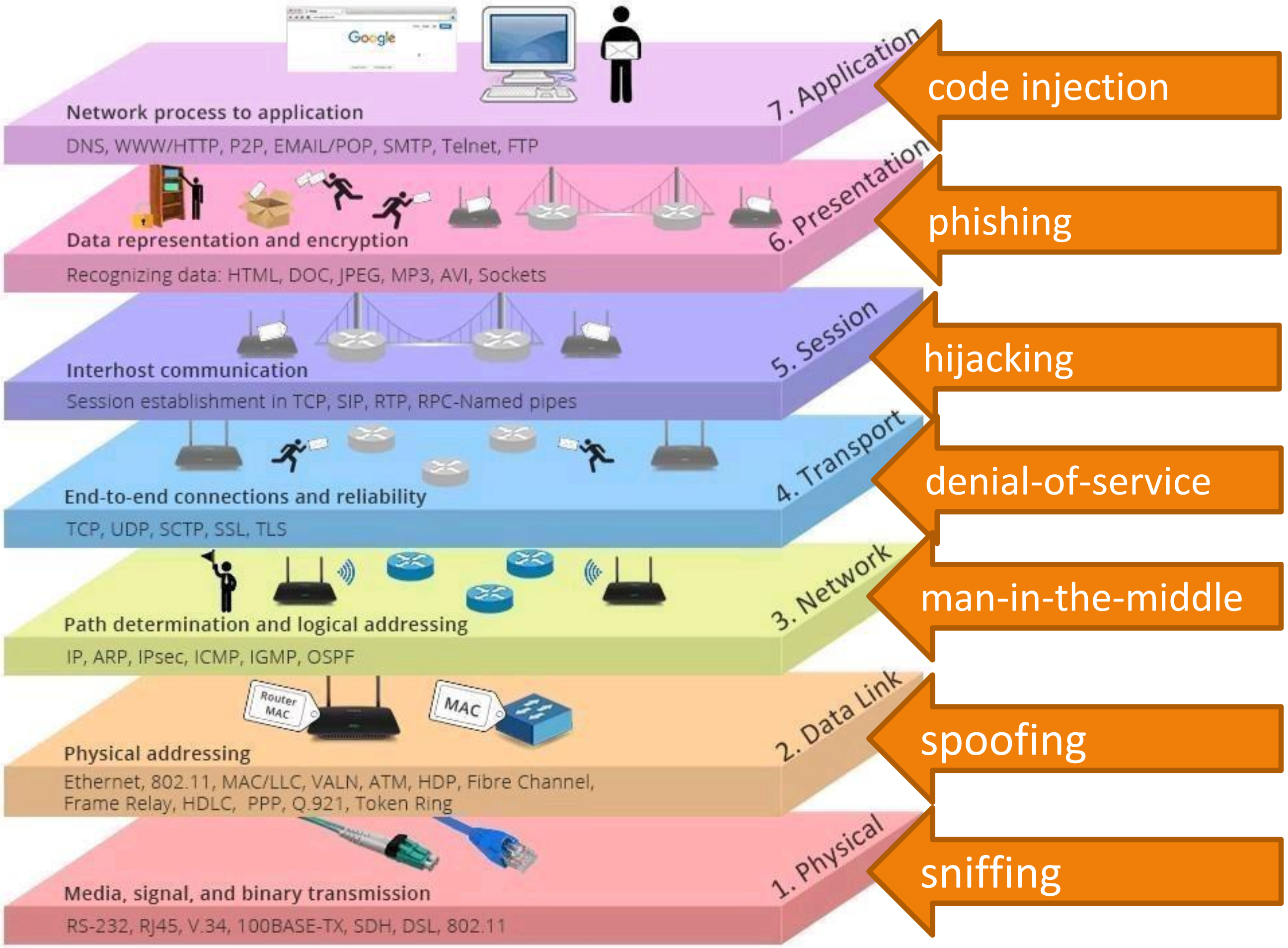
“Social Engineering”

- Psychological **manipulation** of people into performing actions or divulging confidential information
  - Trick a user to grant access to resource or reveal some secret
- Examples of social engineering attacks:
  - Pretexting
    - E.g., attacker claims to be part of the administrative team and asks for password to “repair” the system
  - Baiting
    - E.g., attacker leaves unattended USB drive with malware that the user inserts into the computer
  - Phishing
    - E.g., attacker sends email with malicious attachment or link to be clicked
  - Deep fakes
    - Voice and video



Kevin Mitnick
(1963-2023)

Example attacks on each layer

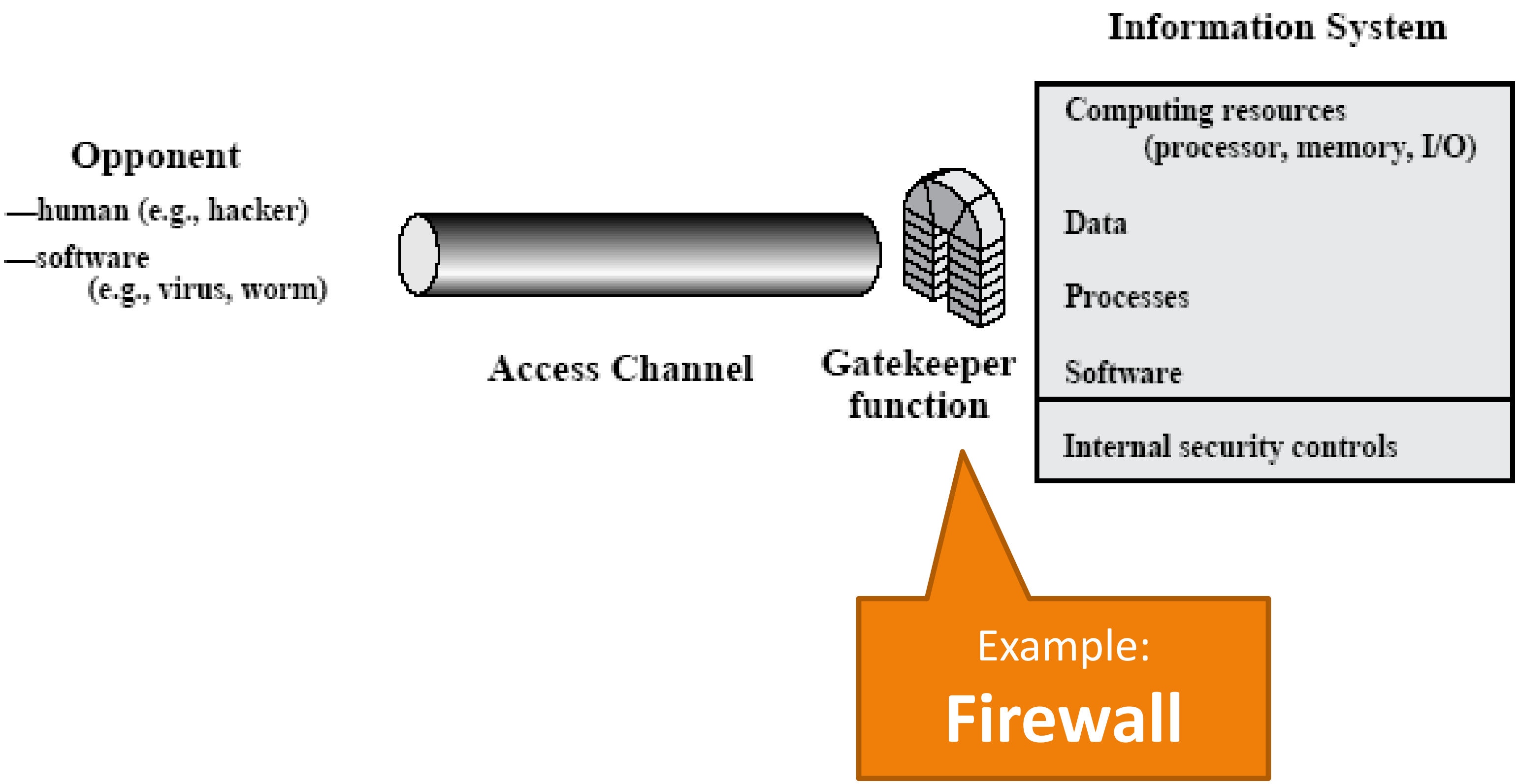


Roadmap

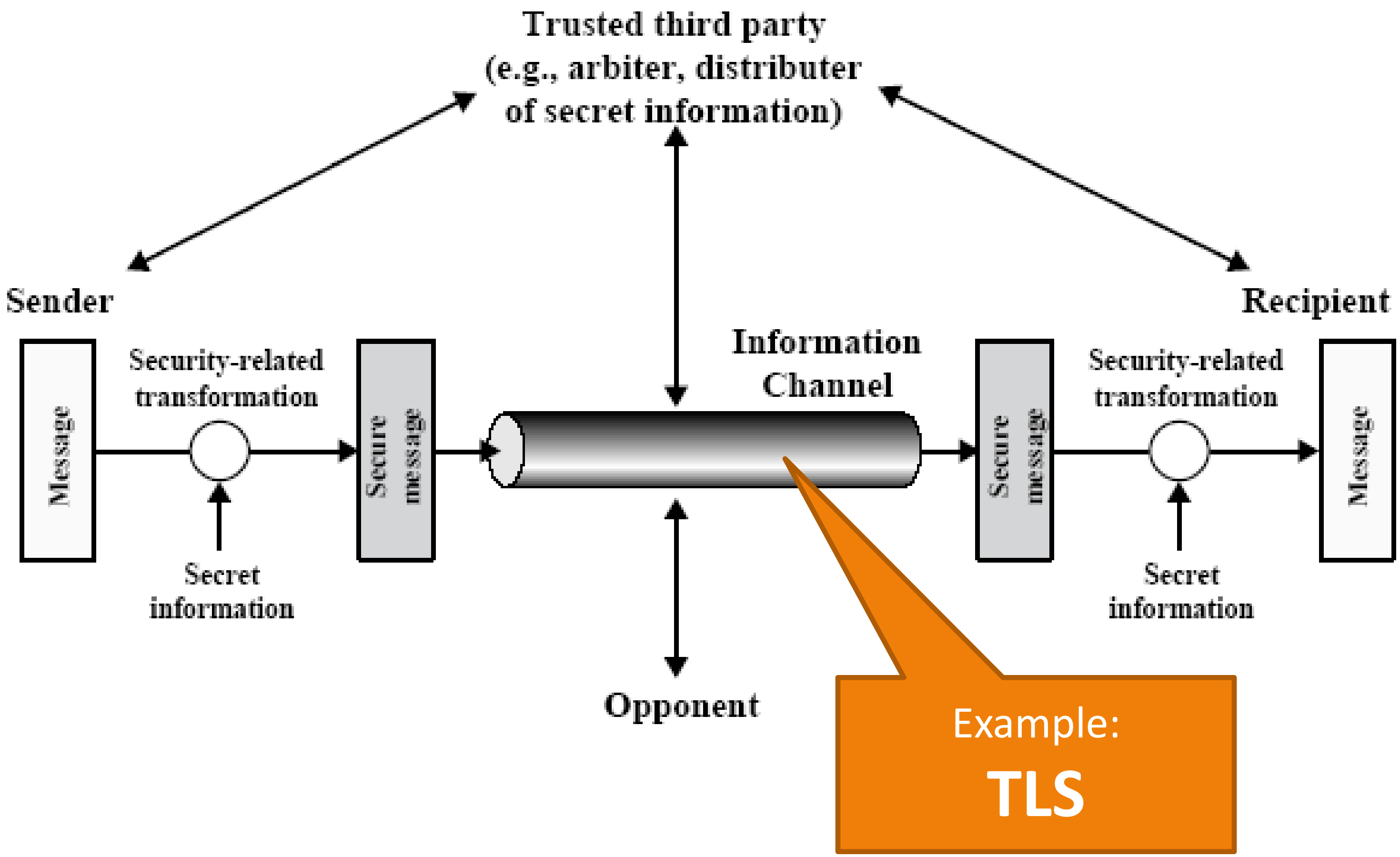
- Network vulnerabilities
  - Physical layer
  - Data link layer
  - Network layer
  - Transport layer
  - Application layer
- **Network security models**

NETWORK SECURITY MODELS

Network Security Model I: Gatekeeper for access control



Network Security Model II: Secure communication channel



Summary

- Network models
  - OSI and Internet
  - Address resolution
- Network attacks
- Network vulnerabilities
  - Physical layer
  - Data link layer
  - Network layer
  - Transport layer
  - Application layer
- Network security models