Université de Rennes

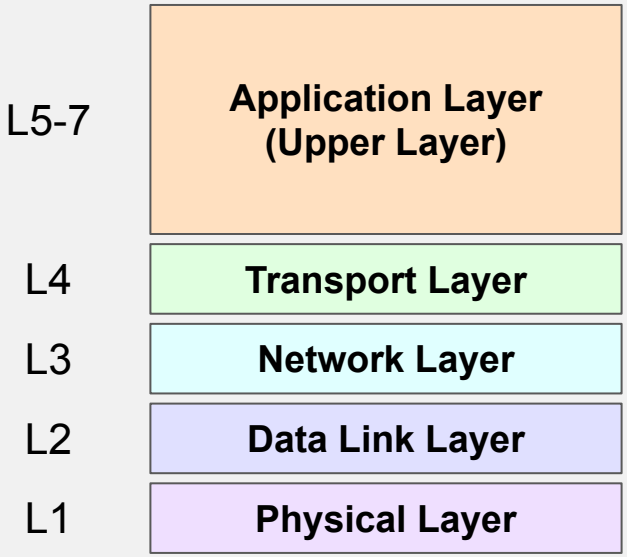
istic Informatique Électronique

Network Security

SYN City: A TCP/UDP Story

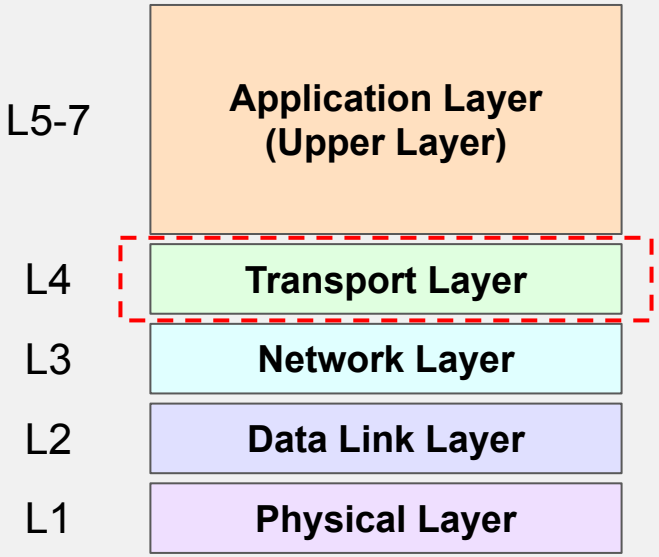
Gwendal Patat
Univ Rennes, CNRS, IRISA
2025/2026

Recall TCP/IP Model



TCP/IP Model

Today's Topic: Transport Layer

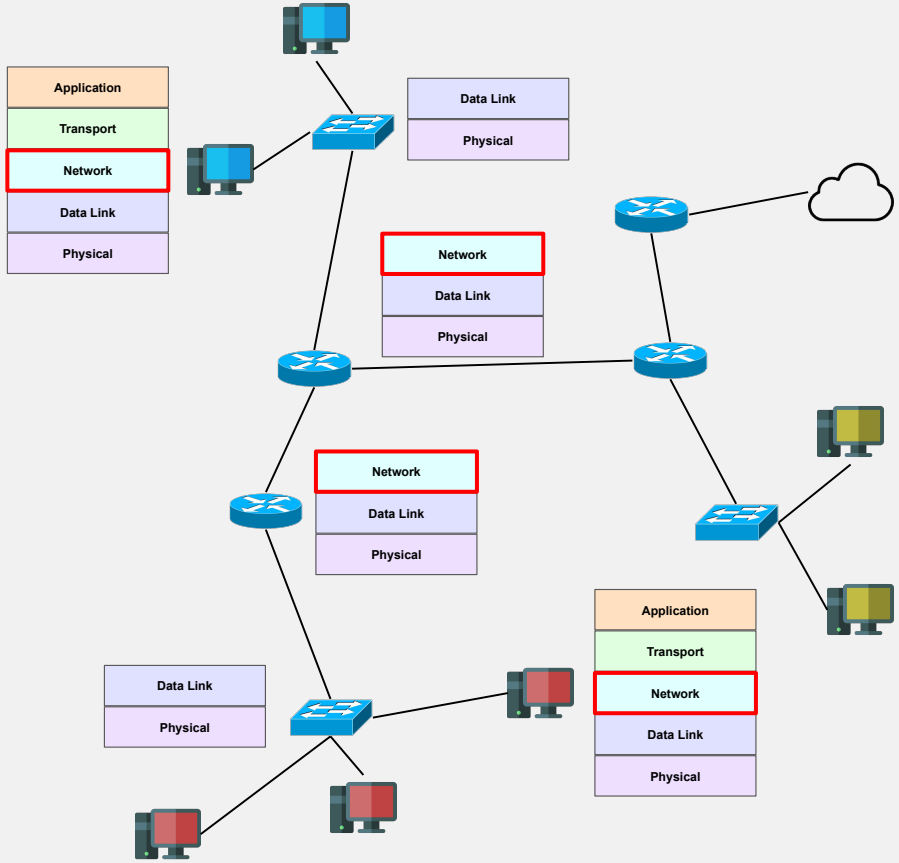


TCP/IP Model

Transport Layer 1/2

From last lecture, we know that the routing from **device to device** is made by the **Network layer**, layer 3.

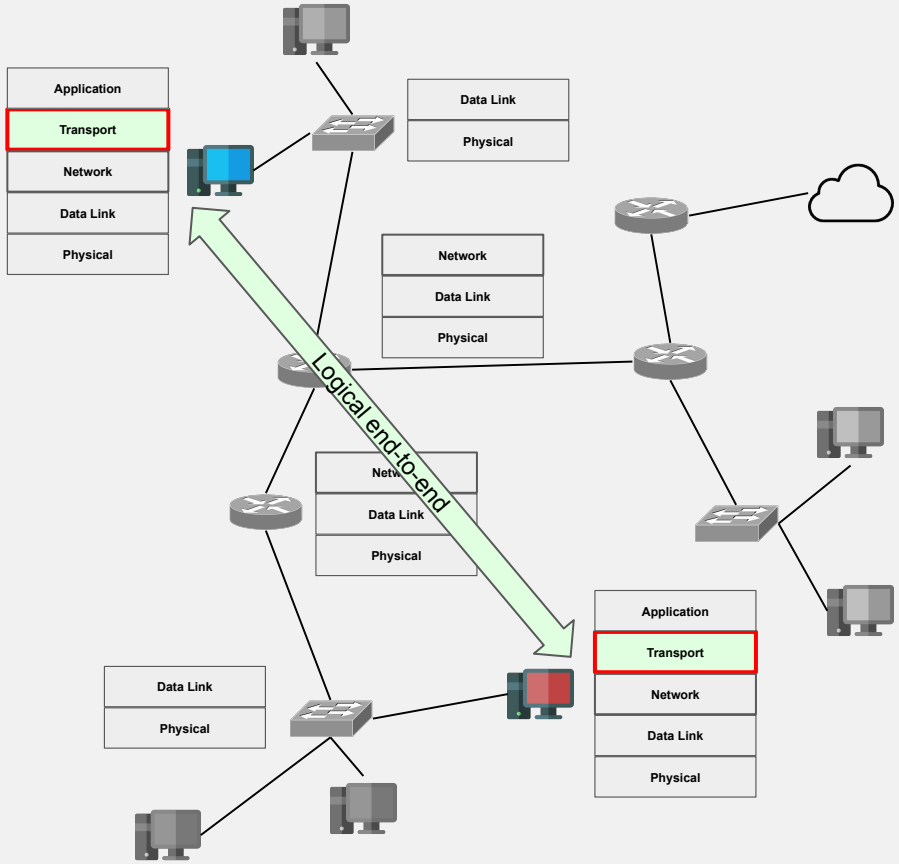
Now what about the **Transport Layer**?



Transport Layer 2/2

Transport Layer:

- Layer 4.
- Provide a logical communication **between processes** on two different hosts.
- Two main protocols:
  - TCP
  - UDP
- Uses ports and sockets.



Transport Layer: Ports & Sockets

- A socket **binds** an application with a port number.
  - It can be seen as a gate to the app.

Port Numbers

- ***"Reserved"* ports**: 0 - 1023
  - For instance: ssh (22), telnet (23), smtp (25), dns (53), http (80), https (443)
- **Registered ports**: 1024 - 49151
  - battle.net (1119), openVPN (1194), Microsoft SQL server (1433), Microsoft Teams (3478–3481)
- **Private ports:** 49152 - 65535
  - Source ports

Note: Nothing is enforced, any application can use the port number they want (except the reserved ones where root rights are needed).

Client-Server Paradigm

- **Server:**
  - Listening to incoming communications.
  - Provide the application layer service to the client.

- **Clients:**
  - Communicate with servers.
  - Do not communicate directly with each other.

- **Examples:** HTTP, IMAP

Processes Communication

Client process: process initiating communication.

Server Process: process waiting for communication.

Process: program running on a host

- Within the same host:
  - Processes communicate using **inter-process communication** (OS defined)
    - e.g., signals, pipes, shared memory, etc…
  - Using messages send to **sockets**.
- With different hosts:
  - Using **sockets**.

Note: For Peer-to-Peer (P2P), both end users have a client process and server process.

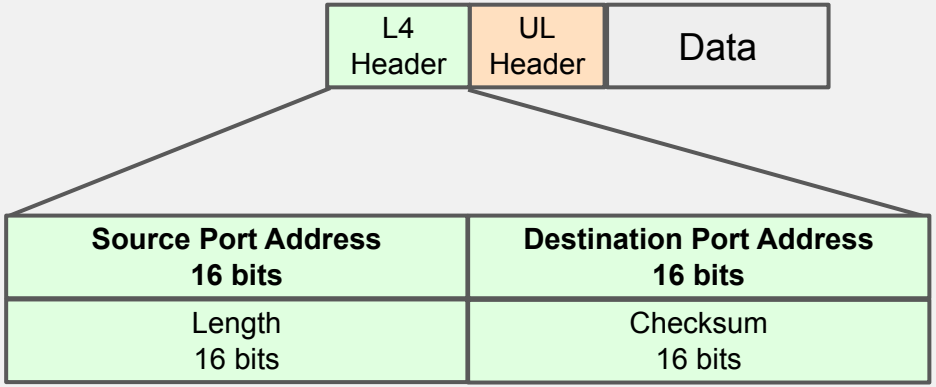
Transport Layer Protocols

	TCP	UDP
Connection	Connection	Connection-less
Reliability	✔	✗
Ordering	✔	✗
Speed	Slower	Faster
Broadcast	✗	✔

UDP
(User Datagram Protocol)

UDP Header

- Application identified through the **destination *port***.
- Responses will be sent to the ***source port***.



Sending and Receiving UDP Packets

- Need an IP address and the destination port
- The server will listen to its open port.

Connectionless Demultiplexing:

Two senders communicating with a server on the same port will fill the same buffer.

- UDP packets with same destination port number will end up in the same socket **without taking care of the source IP.**

Some UDP Applications

- DNS protocol
- Video/Audio Streaming: Skype, Zoom
  - Streaming services uses TCP (non-live performance)
- Real-Time Apps
- VPN (OpenVPN)

UDP Attacks

UDP Flood (Ping-Pong Attack) 1/2

Objective: DoS on two victimes.

How: Launch an ***infinite loop*** of request/reply between two devices by ***spoofing the source IP and port*** of the first message.

- For it to work, we need a process listening and answering regardless of the request on both devices.
  - This was not uncommon in the 90s/early 2000s.

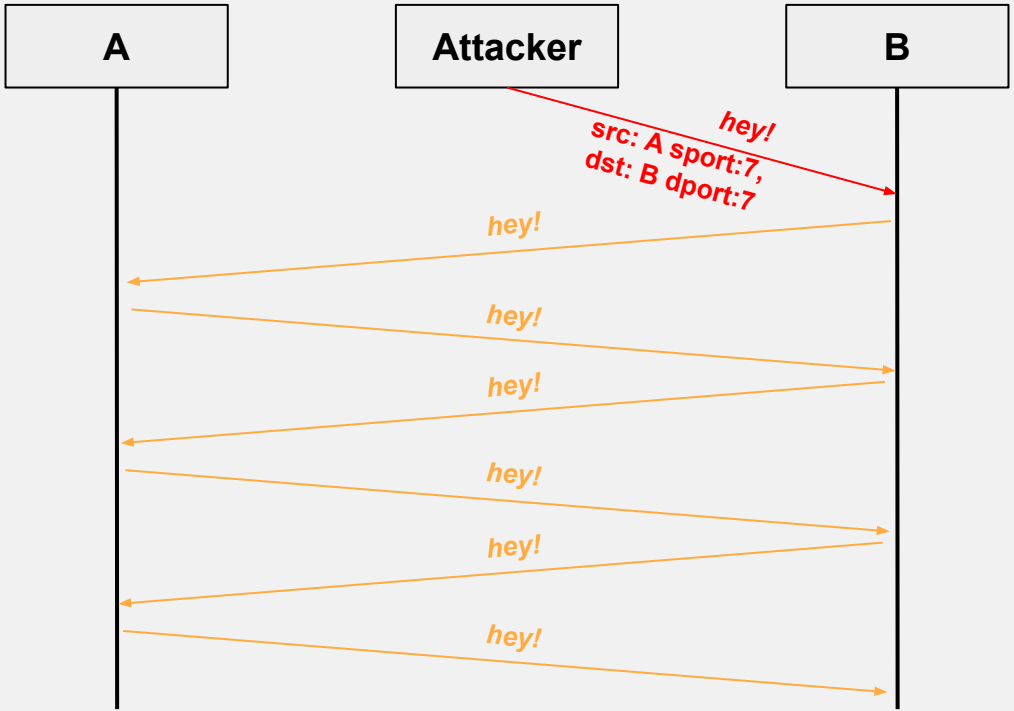
UDP Flood (Ping-Pong Attack) 2/2

Here an Attacker leverages the Echo protocol on port 7.

- When a message is received, it is duplicated and ***send back to the source address***.

By piping both Echo processes from A and B through port 7, the attacker create an infinite loop.

Nowadays, this protocol is disabled by default to avoid such attack.



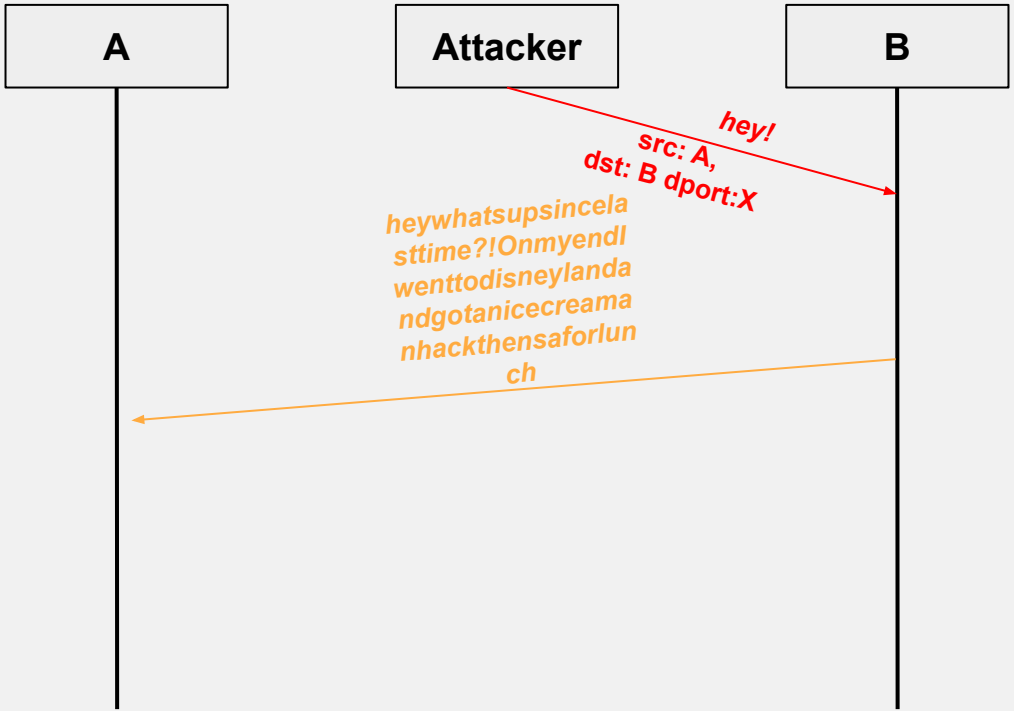
UDP Amplification Attack 1/2

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

Ref: Christian Rossow

UDP Amplification Attack 2/2

Here an Attacker can leverage an open port X with an amplificator process to degrade the bandwidth of the victim A.



UDP Security Consideration

Spoofing UDP: Due to the protocol simplicity, it is really easy to spoof UDP messages.

UDP applications need to take this problem into consideration with additional protection build on top of the UDP protocol (e.g., random number in data, encryption, …)

TCP
(Transmission Control Protocol)

Order & Reliability

One of the main property of TCP: **Maintain order of packets**

- How? By using sequence number and acknowledgement.
- Reliability: Packets are resend if no acknowledgement comes back after a certain time.

Flow & Congestion Control

TCP incorpores a mechanisms to **adjust the interval of packets between two hosts** with different speed.

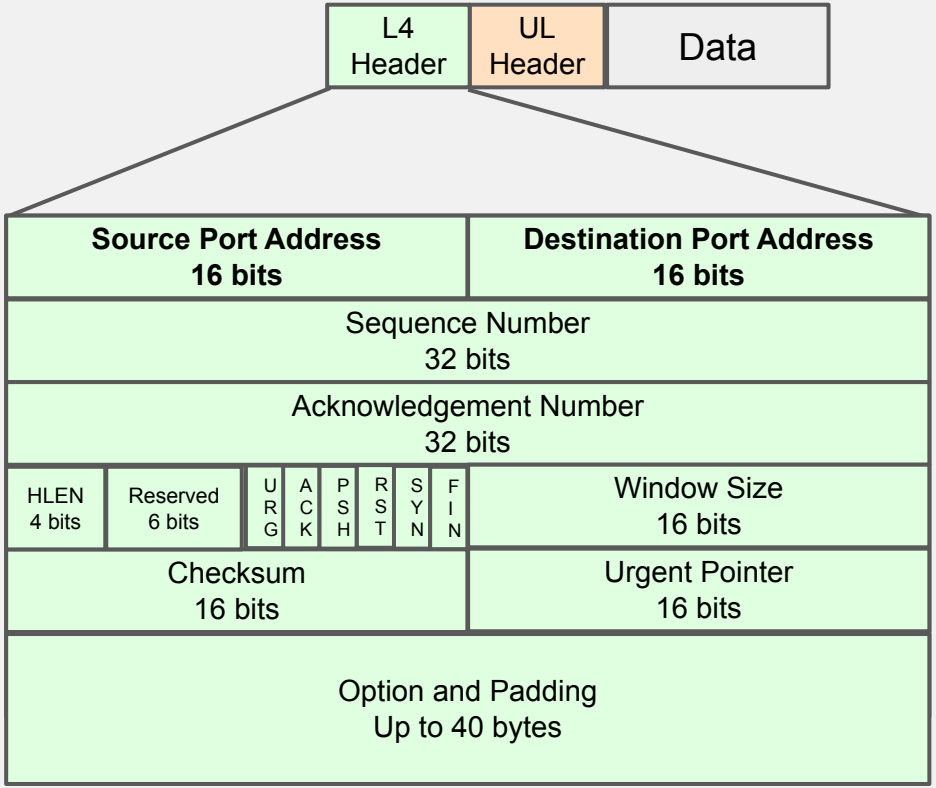
- Using what we call a sliding window.
- Everything in the window can be sent to the destination without waiting for ACK.
- The window slides when the older packet has been acknowledged.

The receiver can ask for the window size to be updated with window advertisement messages.

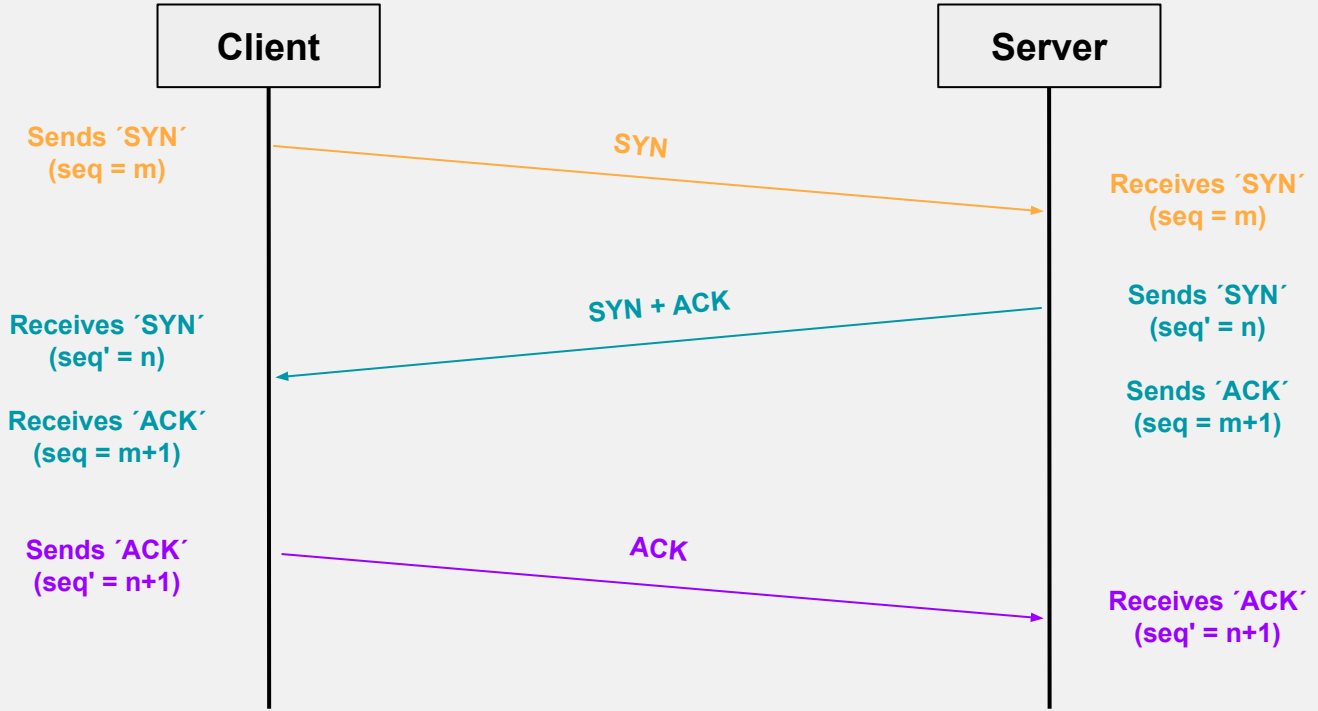
On the sender side, a congestion window can be used when too many packets are lost on the road.

TCP Header

- Communication with IPs, and **source/destination *ports***.
- Reliability via sequence and acknowledgment (ACK) numbers.
- Flow control through the window size.
- Integrity with the checksum.
- Additional functions (SYN, FIN, RST, etc.) that handle connection establishment, termination, or reset.



TCP Connection: 3-Way Handshake



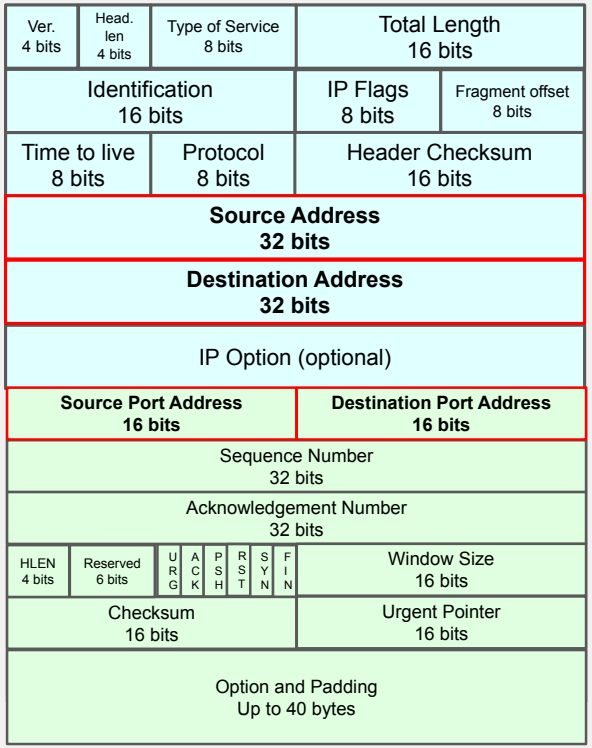
TCP Connection-oriented: 4 tuples

Connections are identified based on a 4 tuple:

- Source port
- Destination port
- Source IP
- Destination IP

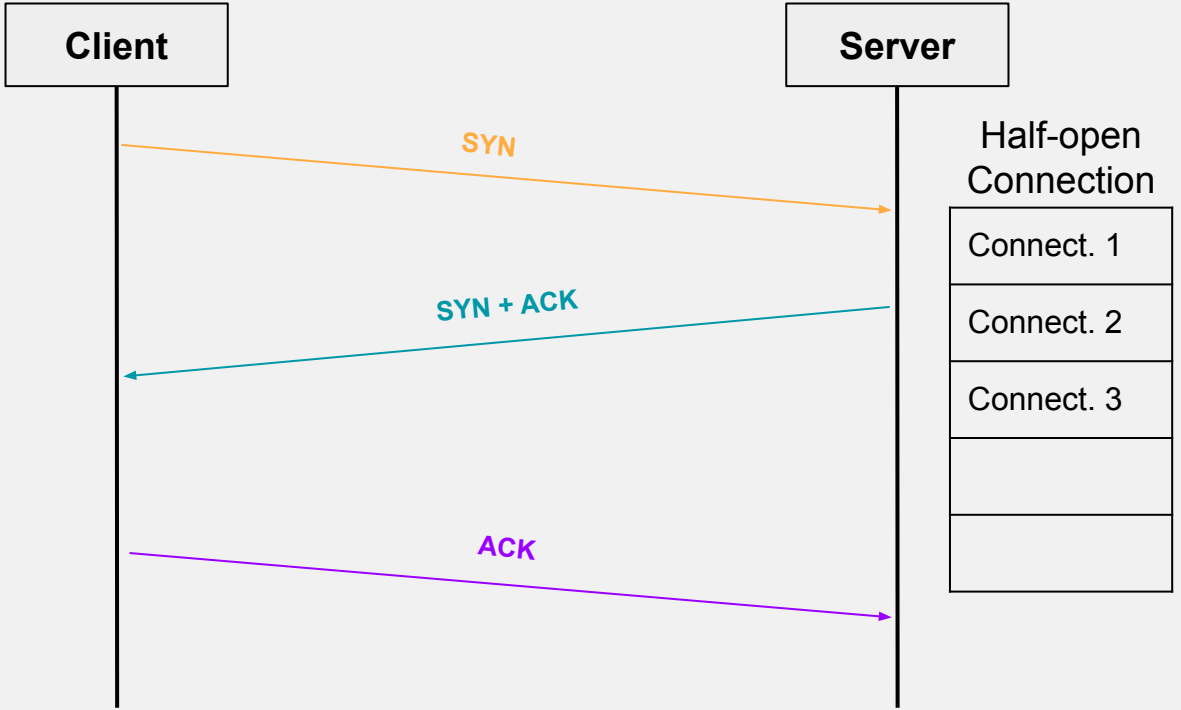
Unlike UDP, a TCP connection is linked to a dedicated socket:

- Two IPs cannot establish multiple connections with the same 4 tuples.

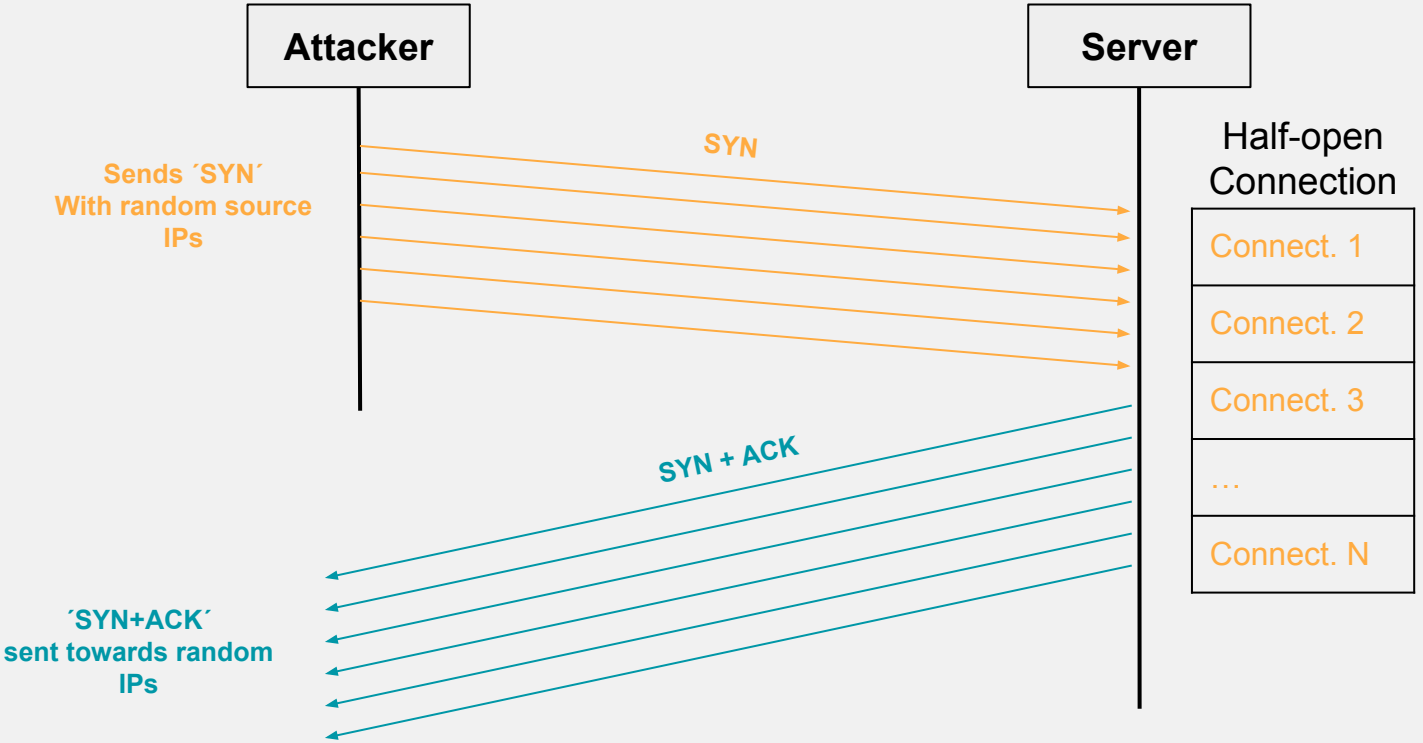


TCP Attacks

SYN Flooding Attack 1/2



SYN Flooding Attack 2/2



SYN Flooding Countermeasure

SYN Cookie:

- Encode information in the sequence number to avoid relying on the half-open connection queue.
- Enabled by default on current OSes.

Firewall:

- Limit the number of SYN packets within a time window.

Queue:

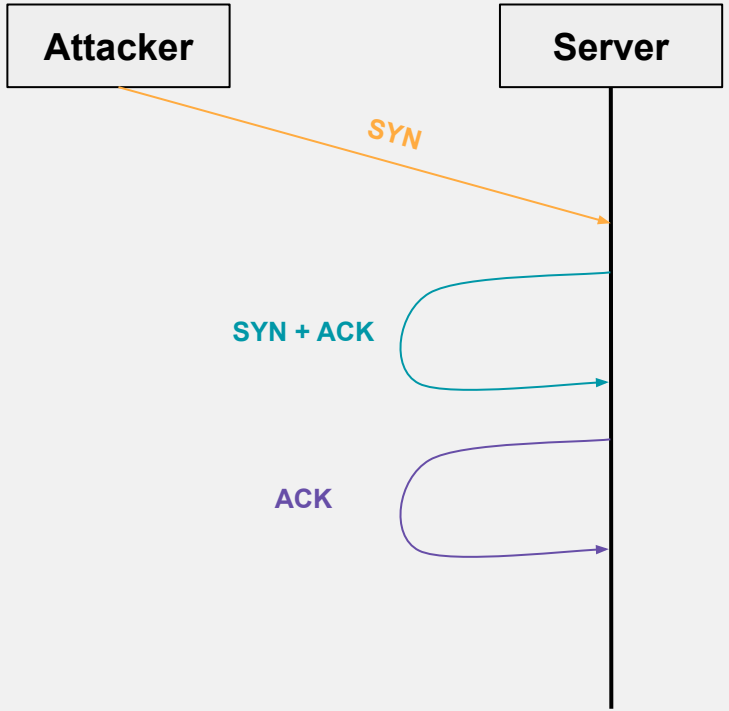
- Random half open connection entry deletion.

LAND Attack

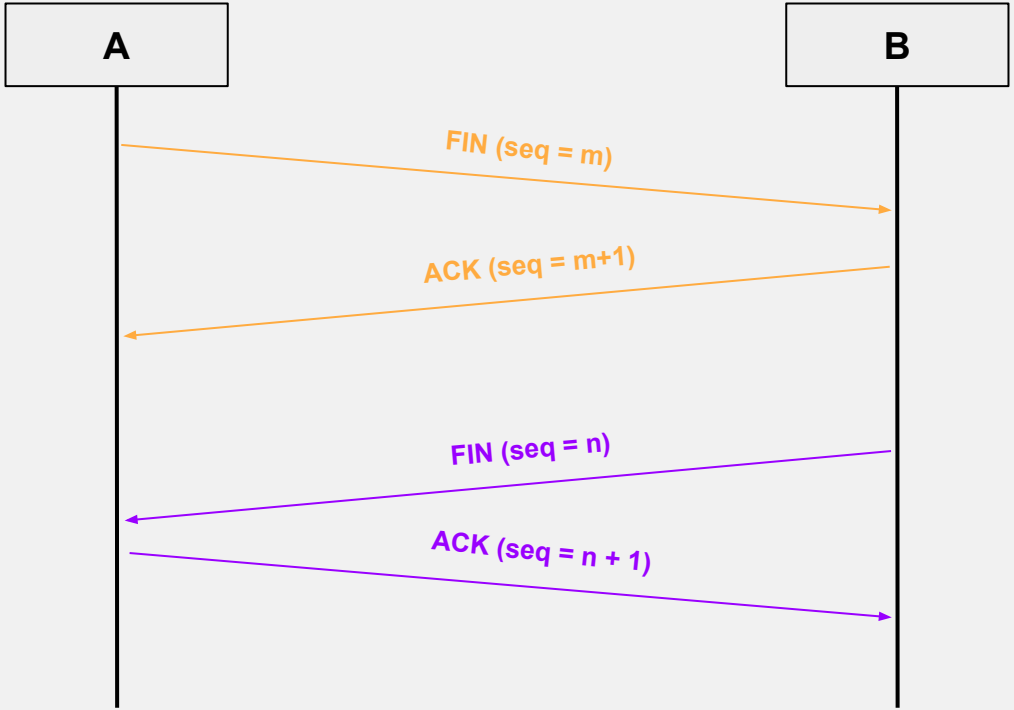
An attacker can forge a SYN packet by spoofing the **IP src and dst** to be both the one of the victim: creating a connection with itself.

Mitigations:

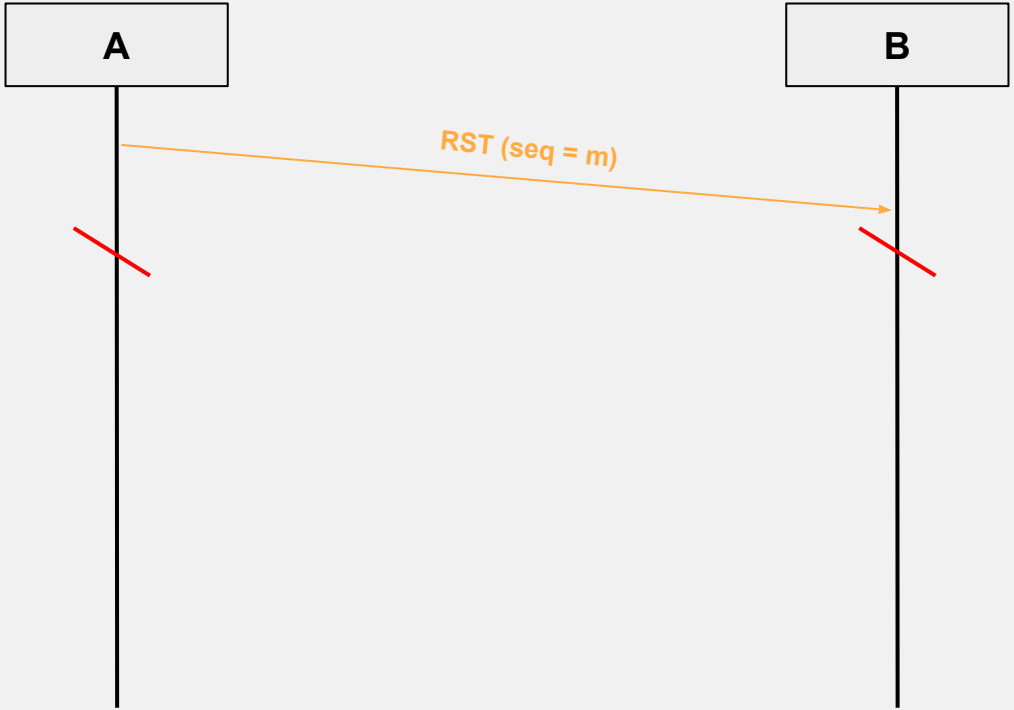
- Packet filtering (Firewall)
- OS mechanisms.



Closing a TCP Connection 1/2



Closing a TCP Connection 2/2



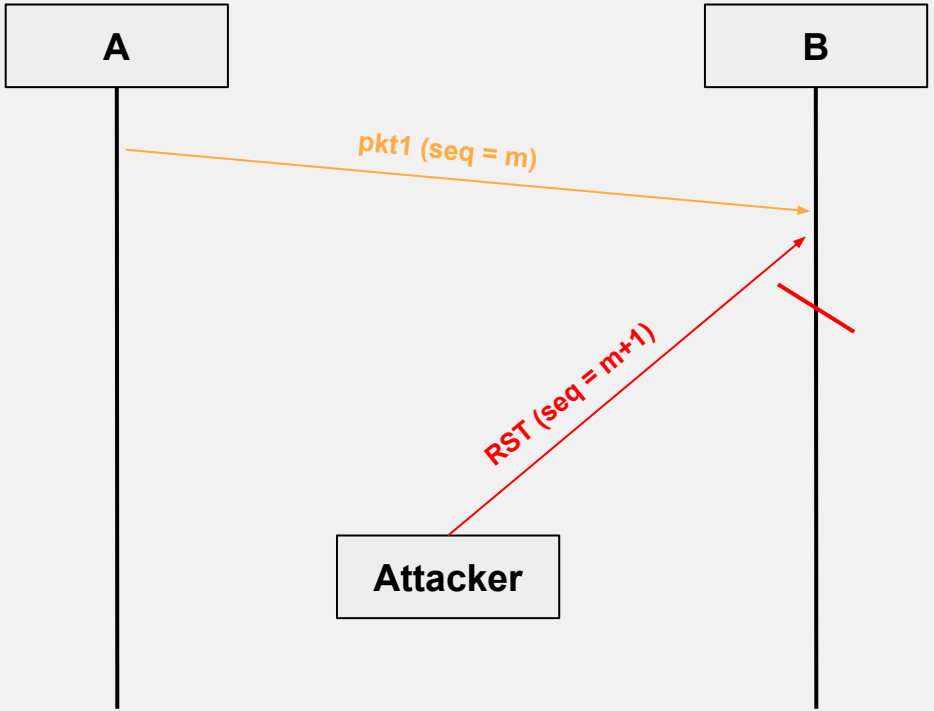
TCP Reset Attack

If A and B are communicating, an attacker can **forge a reset package** by setting the **RST flag** and using the **next sequence number** to close one or both end of the connection.

The attacker needs to create the package using the good values:

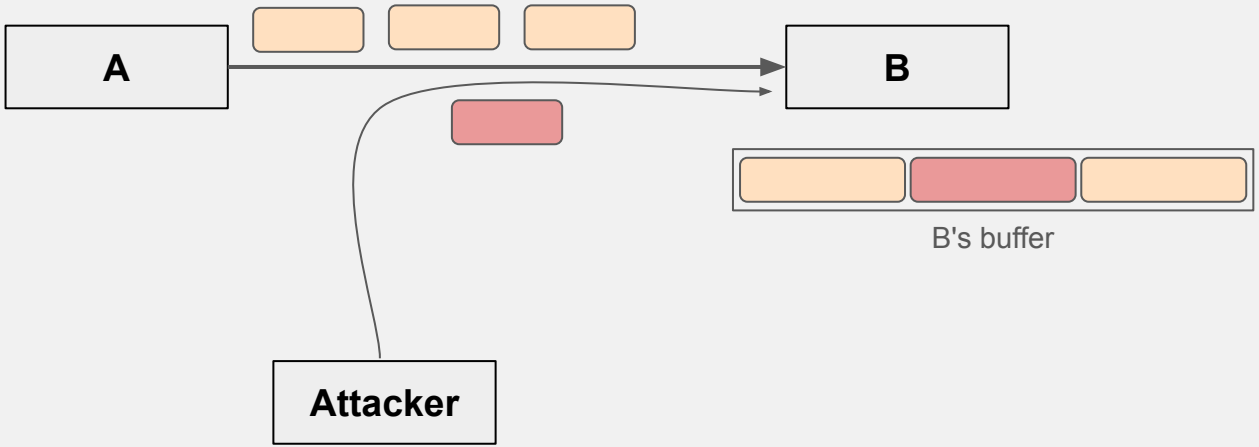
- Source port
- Destination port
- Source IP
- Destination IP
- Seq #

Note: also used for defense.



TCP Session Hijacking 1/2

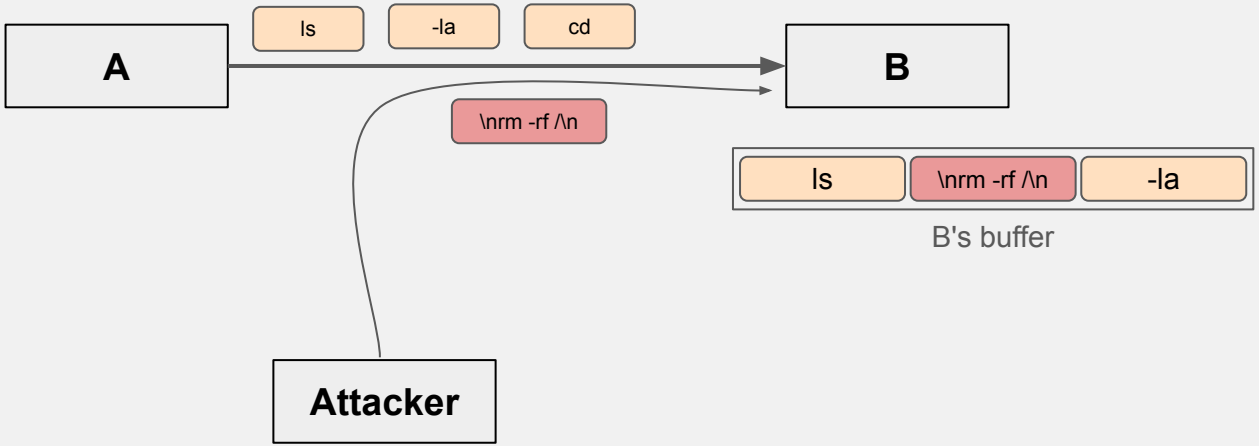
An attacker can forge a TCP packet by **spoofing the 4-tuple and sequence #** to **inject data** in the TCP stream of the victim.



Ver. 4 bits	Head. len 4 bits	Type of Service 8 bits	Total Length 16 bits
Identification 16 bits		IP Flags 8 bits	Fragment offset 8 bits
Time to live 8 bits	Protocol 8 bits	Header Checksum 16 bits	
Source Address 32 bits			
Destination Address 32 bits			
IP Option (optional)			
Source Port Address 16 bits		**Destination Port Address 16 bits**	
Sequence Number 32 bits			
Acknowledgement Number 32 bits			
HLEN 4 bits	Reserved 6 bits	URG ACK PSH RST SYN FIN	Window Size 16 bits
Checksum 16 bits			Urgent Pointer 16 bits
Option and Padding Up to 40 bytes			

TCP Session Hijacking 2/2

Example of impact: This can allow an attacker to inject commands in a telnet session.



Ver. 4 bits	Head. len 4 bits	Type of Service 8 bits	Total Length 16 bits
Identification 16 bits		IP Flags 8 bits	Fragment offset 8 bits
Time to live 8 bits	Protocol 8 bits	Header Checksum 16 bits	
Source Address 32 bits			
Destination Address 32 bits			
IP Option (optional)			
Source Port Address 16 bits		**Destination Port Address 16 bits**	
Sequence Number 32 bits			
Acknowledgement Number 32 bits			
HLEN 4 bits	Reserved 6 bits	URG ACK PSH RST SYN FIN	Window Size 16 bits
Checksum 16 bits			Urgent Pointer 16 bits
Option and Padding Up to 40 bytes			

TCP Reverse Shell 1/2

Specific type of TCP session hijacking.

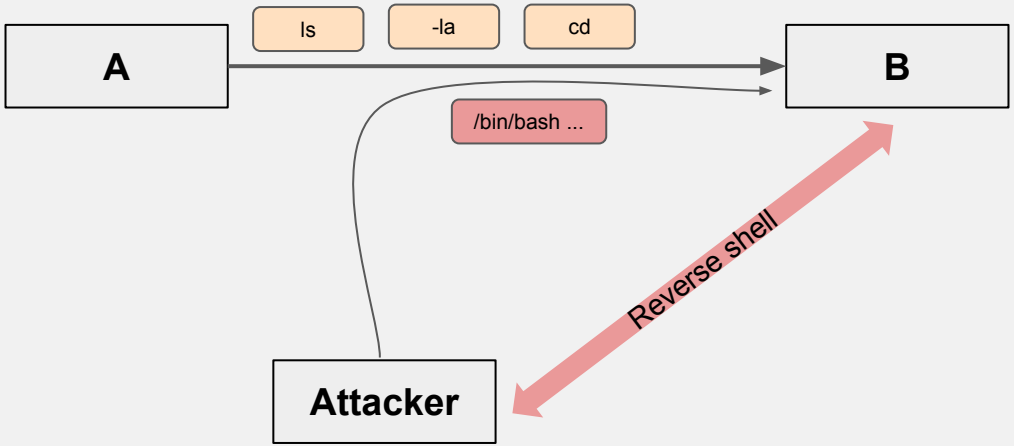
Idea: Run a shell command on the target.

Problem: We can’t just replace our command with a simple */bin/bash* for instance.

- Need to figure out how to redirect IO to a TCP connection.

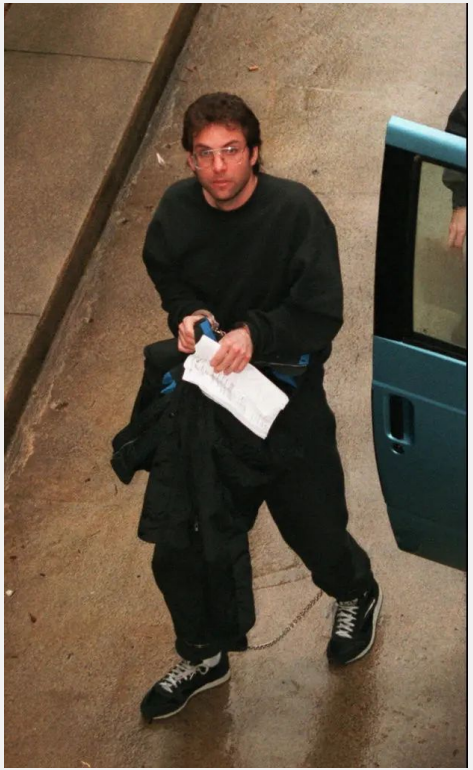
TCP Reverse Shell 2/2

- The payload sent over need to create a shell redirecting IOs to the attacker device, with its IP and a port.



Kevin Mitnick’s Attack

Kevin Mitnick (1963-2023) VS Tsutomu Shimomura (1964-)



WANTED
BY U.S. MARSHALS

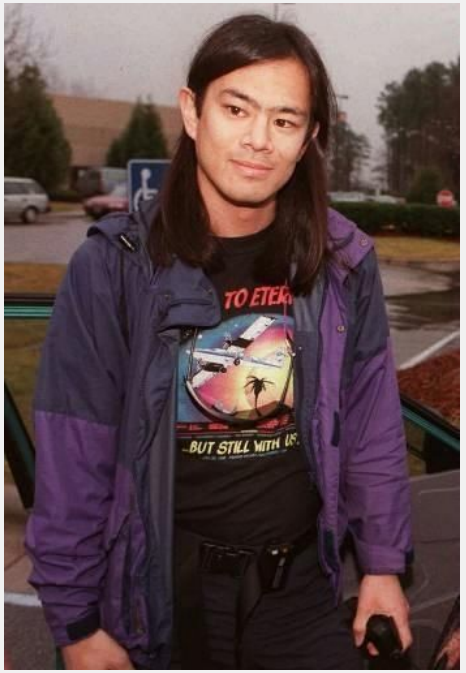
NOTICE TO ARRESTING AGENCY: Before arrest, validate warrant through National Crime Information Center (NCIC).

United States Marshals Service NCIC entry number: (NIC/ W721460021).

NAME: .....MITNICK, KEVIN DAVID

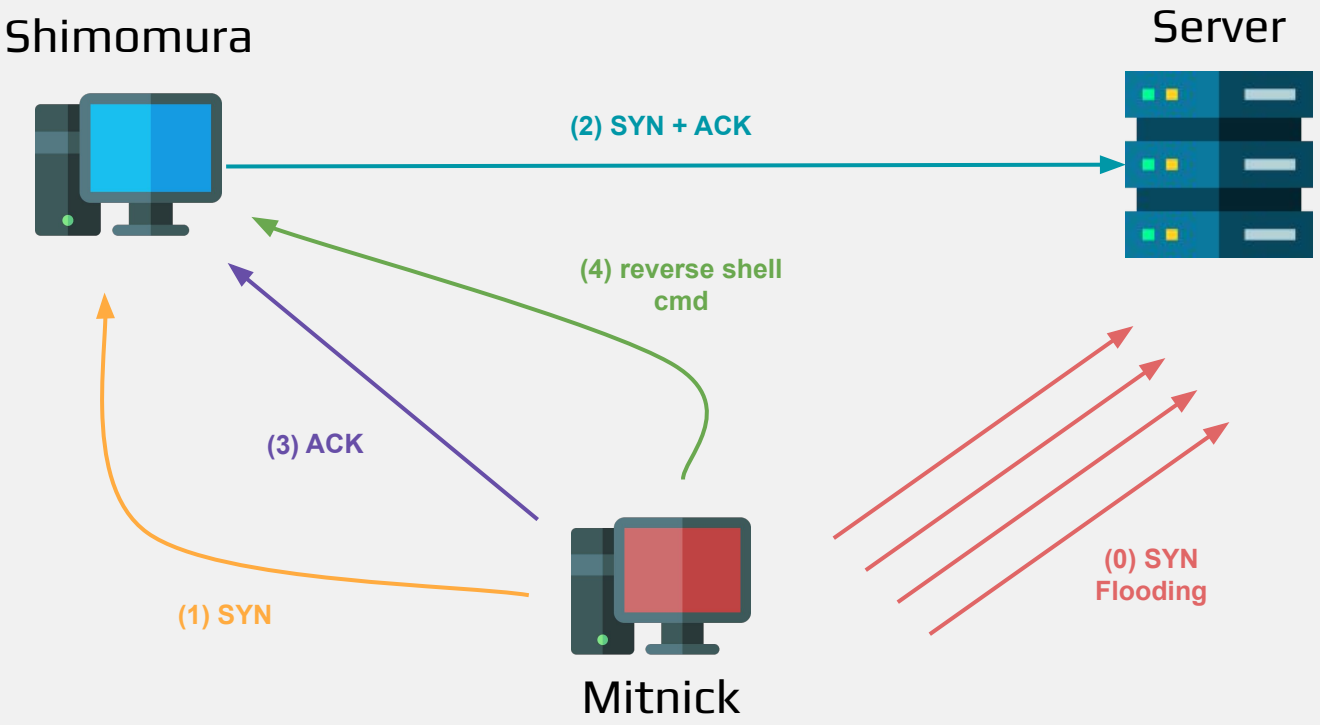
AKS (S): .....MITNIK, KEVIN DAVID
MERRILL, BRIAN ALLEN

DESCRIPTION:
Sex: .....MALE
Race: .....WHITE
Place of Birth: .....VAN NUYS, CALIFORNIA
Date(s) of Birth: 08/06/63; 10/18/70
Height: .....5'11"
Weight: 190
Eyes: .....BLUE
Hair: .....BROWN
Skintone: .....LIGHT
Scars, Marks, Tattoos: .....NONE KNOWN



Mitnick Attack

Mitnick Attack



Countermeasures

Some countermeasures can be applied to avoid TCP attacks, for instance:

- SYN Cookies
- Random sequence number
- Random source port number
- Random half open connection entry deletion
- Encryption (IPSec, or Encrypted data)
- Packet inspection / Firewalls

Bonus: Port Scanning

We have seen that some ports are commonly used for specific services:

- 22 ssh
- 23 telnet
- 80 http
- 443 https

By scanning the open ports on a device, and with specific requests, we can determine the **open services, versions, and even OS** of a target.

Example of tool: nmap

Resources and Acknowledgements

- Computer Networking: A Top-down Approach by James F. Kurose, Keith W. Ross
- Internet Security: A Hands-on Approach, 3rd Edition, Du Wenliang
- External materials from Mathieu Goessens.