NETWORK SECURITY:

ARP, TCP/IP, UDP

ADDRESS RESOLUTION PROTOCOL (ARP)

→ connects the network layer to the data link layer

Devices on LAN have → IP addresses (network layer)
→ MAC addresses (data link layer)

→ maps IP addresses to MAC addresses

→ is based on broadcast messages + local caching

→ No security built into it; anybody can see what IP address belong to which MAC address and anyone can change this info.

↳ No confidentiality / integrity / authentication

ARP MESSAGES

→ ARP broadcasts requests of type: who has <IP address C>
tell <IP address A>

→ Machine w/ <IP Address C> responds: <IP address C> is at <MAC address>

→ The requesting machine will cache the response

ARP CACHE

→ The command arp -a displays the ARP table

even if it does NOT send any ARP request!

→ The ARP table is updated whenever an ARP response is received

Machines trust e/o

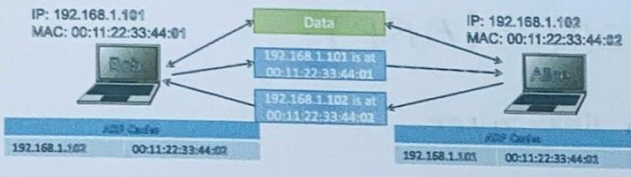
→ Requests are not tracked & ARP announcements are not authenticated

↳ can lead to ARP Cache Poisoning / ARP Spoofing

where an attacker sends spoofed ARP messages onto a LAN

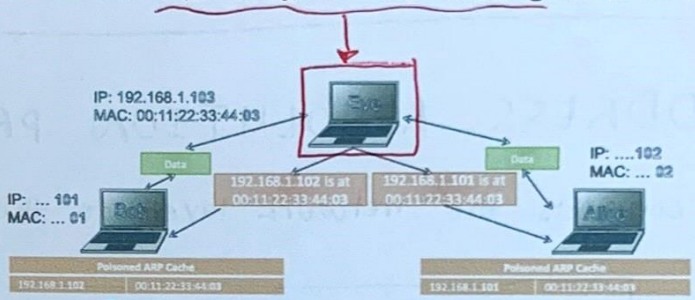
→ Using static entries solves the problem but it is impossible to manage!

ARP Normal Operation



ARP Spoofing

→ There's a Mal actor-in-the-middle (MITM)



↖ Incorrect info. in their caches ↗

IP VULNERABILITIES

→ All the transmissions are unencrypted

→ No source authentication

- sender can spoof source address, making it difficult to trace packet back to attacker

→ No integrity checking

- Entire packet can be modified, enabling content forgeries, MITM attacks, redirections

→ No bandwith constraints

- Large no. of packets can be injected into network to launch a DOS attack

---

USER DATAGRAM PROTOCOL (UDP)

↗ It is at the transport layer

→ is a stateless, unreliable datagram protocol built on top of IP

→ can distinguish data for multiple concurrent applications on a single host (which IP cannot)

→ Applications using UDP must accept a fair amt. of corrupted & lost data

e.g. VoIP, streaming video/audio all use UDP [Time-sensitive applications]

→ While UDP features a 16-bit checksum to verify integrity of each indiv. packet, there is no sequence no. scheme so transmissions can arrive out of order or not arrive at all

TRANSMISSION CONTROL PROTOCOL (TCP)

→ is a stateful (↳ keeps track of connection state in memory) transport layer protocol for reliable data trf. and in-order delivery of messages

e.g. HTTP and SSH are built on top of TCP

→ has ability to distinguish multiple applications on same host

→ packages a data stream into segments for transport by IP

↳ Order of segments are maintained by marking each packet w/ a sequence number

↳ Every time TCP receives a packet, it sends out an acknowledgement (ACK) to indicate successful receipt of the packet ↳ How you get reliability!

→ checks data transmitted by comparing a checksum of the data w/ a checksum encoded in the packet → Ensures integrity of data!

PORTS

→ allow TCP and UDP to support concurrent app. on the same server

→ are 16 bit numbers identifying which app. the data is meant for

→ The header (of TCP/UDP packet) includes both a source & dest. port

→

Ports	0 – 1023	1024 – 49151

1024 – 49151 → User ports

0 – 1023 ↓ reserved for use by known protocols

TCP DATA TRF.

→ is done through some initialization using the 3-way handshake, which allows initial sequence numbers to be exchanged next pg.

→ TCP header includes 16-bit checksum of the data

→ ACKs (or lack thereof) and window size are used by TCP to keep track of:

* [• Packet loss
  • Network congestion
  • Flow control]

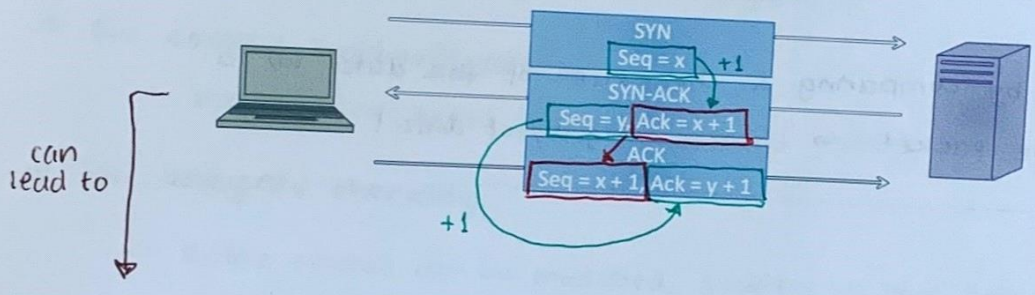
ESTABLISHING TCP CONNECTIONS

→ done through a **three-way handshake**:

1) Client requests a connection by sending out a SYN packet (contains seq. num.)

2) Server responds by sending a SYN/ACK packet, acknowledging the connect
(contains new seq. num. that server has picked and the ack. num. is the seq. num. (in the SYN packet) + 1)

3) Client responds by sending an ACK to the server, thus establishing connection



SYN FLOODING

→ where thousands of SYN requests are sent to the victim

e.g. Alice sends many SYN packets w/o acknowledging any replies
Bob accumulates more SYN packets that he can handle → runs out of space in state table. → It will stop taking connections from potentially legitimate users wanting to connect to it.

→ **PROBLEMS:**

1) Attribution – attacker uses their own IP address which could be traced
2) Bandwith – attacker uses their own bandwith which is likely smaller than the server's
↓
effective against a small target

SPOOFING

→ same as SYN flooding, but forge the source IP address of the TCP packet

→ **ADVANTAGES:** — Harder to trace
— ACKs are sent to a second comp., less attacker bandwith used

→ **PROBLEM:** — Ingress filtering is commonly used to drop packets w/ source addresses outside their origin network fragment

SMURFING (Directed broadcast)

→ This attack exploits ICMP ping requests whereby remote hosts respond to echo packets to say they are online

→ IDEA – Ping a LAN on a broadcast address, then all hosts on the LAN will reply to the sender of the ping.

↓

ATTACK – 1) Make a forged packet w/ victim's IP address as the source

2) Send it to a Smurf amplifier, which causes a huge no. of replies to the victim

↓ networks that respond to pings

→ LANs that allow Smurf attacks are badly configured & one approach is to blacklist these LANs

DISTRIBUTED DENIAL OF SERVICE (DDoS)

→ A large no. of machines work tgther. to perform an attack that prevents valid users from accessing a service