CSE 127 Computer Security
Stefan Savage, Fall 2023, Lecture 13

Network Security I: background & basic attacks
Objectives today

- Understand
  - Architecture of the Internet protocol suite (TCP/IP)
    - CSE123 in 20mins
  - Common weaknesses in basic networking protocols
  - Available mitigations and their limitations
First: packets

- A communications on the Internet is constructed of discrete, self-addressed, chunks of data: packets
  - Apologies in advance, but networking has lots of words for packets when used in different contexts
    - Packet, frame, segment, datagram, cell

- This is different from older circuit switched networks (e.g., the traditional phone system)
  - Which set up circuits between two parties
  - Then send signal (analog) or stream of bits (digital)
Review: Internet Protocol Suite

▪ **Application Layer**
  – Examples: SMTP, FTP, SSH, HTTP, etc.

▪ **Transport Layer**: Port-addressed host-to-host communications (on LAN or WAN).
  – User Datagram Protocol (UDP): single packet transmission with no reliability or ordering mechanisms.

▪ **Internet Layer** (IP): Fragmentation, reassembly, and end-to-end (across network boundaries) routing of data packets.
  – Provides a uniform interface that hides the underlying network topology.

▪ **Link Layer**: Transmission of data frames within a local network (without intervening routers).
  – Example: Ethernet

▪ **Physical Layer**: Transmission of raw bits (rather than logical data packets) over a physical data link connecting network nodes.
  – Example: 1000BASE-T, 5Ghz OFDM WiFi
  – [Technically not part of the Internet Protocol Model, but is still there]
Review: Internet Protocol Suite

Internet protocol suite

Application layer
BGP · DHCP · DNS · FTP · HTTP · IMAP · LDAP · MGCP · MQTT · NNTP · NTP · POP · ONC/RPC · RTP · RTSP · RIP · SIP · SMTP · SNMP · SSH · Telnet · TLS/SSL · XMPP · more...

Transport layer
TCP · UDP · DCCP · SCTP · RSVP · more...

Internet layer
IP (IPv4 · IPv6) · ICMP · ICMPv6 · ECN · IGMP · IPsec · more...

Link layer
ARP · NDP · OSPF · Tunnels (L2TP) · PPP · MAC (Ethernet · DSL · ISDN · FDDI) · more...

https://en.wikipedia.org/wiki/Internet_protocol_suite
Review: Internet Protocol Suite

http://en.wikipedia.org/wiki/Internet_protocol_suite
TCP/IP Protocol Stack by Example

- ROUGHLY, what happens when I click on a URL while UCSD’s network?
Application Layer (HTTP)

- Turn click into HTTP GET request
Application Layer (Name Resolution)

- Where is www.yahoo.com?

What’s the address for www.yahoo.com

Oh, you can find it at 64.58.76.177

Ignore for now:
- How did you know the address of the Local DNS server?
- How did you send a message to it?
- How did the Local DNS server know the answer?
Transport Layer (TCP)

- Break message into pieces (TCP segments)
- Should be delivered reliably & in-order

```
GET http://www.yahoo.com/r/mp HTTP/1.1
Host: www.yahoo.com
Connection: keep-alive
...
```

My computer

```
1 GET http://www.yahoo.com/r/mp HTTP/1.1
2 p://www.
3 yahoo.c
```

“and let me know when they got there”

```
“ACK, I got the first piece”
```

www.yahoo.com
Network layer: IP Addressing

- Address each packet so it can traverse network and arrive at host
- Addresses are generally globally unique
Network Layer: IP Routing

- Each router forwards packet towards destination
Datalink layer (Ethernet)

- Media Access Control (MAC)
  - Can I send now? Can I send now?

- Send individual frames on a link
Physical layer

- 2.4 or 5Ghz Radio x-OFDM Radio (1Mbps – 800Mbps)
- 802.11 Wireless Access Point
- Cat5e Cable (4 wire pairs)
  - 1000BaseT Ethernet
  - 1Gbps
- Ethernet switch/router
- To campus backbone
- 1310nm laser
  - 10GBaseLRM Ethernet
  - 10Gbps
The value of the IP layer

Separate physical networks communicate to form a single logical network (IP addrs *globally unique*)

*except when they aren’t (NAT, anycast)*
### IPv4 Packet Header

<table>
<thead>
<tr>
<th>Field</th>
<th>Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version (ver)</td>
<td>4</td>
</tr>
<tr>
<td>Header Length (HL)</td>
<td>4</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>8</td>
</tr>
<tr>
<td>Total Length (length)</td>
<td>16</td>
</tr>
<tr>
<td>Identification</td>
<td>16</td>
</tr>
<tr>
<td>Flags (F)</td>
<td>3</td>
</tr>
<tr>
<td>Fragment Offset (offset)</td>
<td>13</td>
</tr>
<tr>
<td>Time to Live (TTL)</td>
<td>8</td>
</tr>
<tr>
<td>Protocol</td>
<td>8</td>
</tr>
<tr>
<td>Header Checksum</td>
<td>16</td>
</tr>
<tr>
<td>Source Address</td>
<td>32</td>
</tr>
<tr>
<td>Destination Address</td>
<td>32</td>
</tr>
</tbody>
</table>

Options (if any)

Data (if any)
IP Protocol Functions (Summary)

- **Routing**
  - Your IP host knows location of the **local** router (gateway)
  - IP gateway must know the routes to other networks
    - Packets usually take multiple hops to get to their destination
  - Addresses are globally meaningful
    - 32 bits (IPv4), address separated into network part and host part (127bits w/IPv6)

- **Error reporting**
  - Send Internet Control Message Protocol (ICMP) packet back to source if there was a problem

- **Fragmentation and reassembly**
  - If max-packet-size on next hop link < user-data-size in IP pkt

- **TTL field:** decremented after every hop
  - Packet dropped if TTL=0. Prevents infinite loops.
Fragmentation

- Sender writes **unique value** in *identification* field.
- If router fragments packet, it copies *this id* into each fragment.
- *Offset* field indicates position of fragment in bytes (offset 0 is first).
  - *MoreFragments* flag indicates that this isn’t the last fragment.
  - *DontFragment* flag tells gateway not to fragment.
- All routers must support 576 byte IPv4 packets (MTU)
### IP Fragmentation and Reassembly

<table>
<thead>
<tr>
<th>length</th>
<th>ID</th>
<th>MF</th>
<th>offset</th>
</tr>
</thead>
<tbody>
<tr>
<td>4000</td>
<td>x</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1500</td>
<td>x</td>
<td>1</td>
<td>1480</td>
</tr>
<tr>
<td>1040</td>
<td>x</td>
<td>0</td>
<td>2960</td>
</tr>
</tbody>
</table>

One large datagram becomes several smaller datagrams.
TTL (Time-to-Live)

- How many more routers can this packet pass through?
  - Designed to limit packet from looping forever
- Each router decrements TTL field
- If TTL is 0 then router discards packet
TCP Primer

- TCP provides reliable, ordered delivery of bytes
- Establishes a stateful bi-directional session between two IP:port endpoints
- Each side maintains:
  - Sequence number: sequence base (i.e., \textit{start from here}) + count of bytes sent
  - Acknowledgement number: acknowledgement base + count of bytes received
- Special packet flags
  - SYN: I want to start a connection
  - FIN: I want to shut down a connection
  - RST: We are killing this connection now
**TCP Header Format**

- **Ports** plus IP addresses identify a connection

<table>
<thead>
<tr>
<th>0</th>
<th>4</th>
<th>10</th>
<th>16</th>
<th>31</th>
</tr>
</thead>
<tbody>
<tr>
<td>SrcPort</td>
<td>DstPort</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SequenceNum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HdrLen</td>
<td>0</td>
<td>Flags</td>
<td>AdvertisedWindow</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td>UrgPtr</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options (variable)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Data
Connection Setup: Agree on initial Sequence #’s

- Three-way handshake

<table>
<thead>
<tr>
<th>Active participant (client)</th>
<th>Passive participant (server)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SYN, SequenceNum = x</td>
<td>SYN + ACK, SequenceNum = y,</td>
</tr>
<tr>
<td></td>
<td>Acknowledgment = x + 1</td>
</tr>
<tr>
<td>ACK, Acknowledgment = y + 1</td>
<td></td>
</tr>
</tbody>
</table>

+data
TCP/IP Security (1970’s)

- Original TCP/IP design: Trusted network and hosts
  - Administered by mutually trusted parties

- End-to-end Principle
  - Intelligence is at the edges
  - Network is simple
    - Optimized for speed and simplicity, maintains no state

- Robustness Principle
  - “In general, an implementation must be conservative in its sending behavior, and liberal in its receiving behavior. That is, it must be careful to send well-formed datagrams, but must accept any datagram that it can interpret (e.g., not object to technical errors where the meaning is still clear).”
  - [https://www.ietf.org/rfc/rfc0791.txt](https://www.ietf.org/rfc/rfc0791.txt)
TCP/IP Security

- Built-in trust assumptions:
  - Network protocols used only as intended
    - Correct packet headers
    - Consideration of others’ resources
      - Rate limiting of costly operations
  - Hosts controlled by trusted administrators
    - Random people can’t get onto the network
    - Correct information reported by hosts
    - Protocols implemented correctly
TCP/IP Security (1980’s)

- Wait ... what if we can’t trust everyone?
  - “When describing such attacks, our basic assumption is that the attacker has more or less complete control over some machine connected to the Internet. This may be due to flaws in that machine’s own protection mechanisms, or it may be because that machine is a microcomputer, and inherently unprotected. Indeed, the attacker may even be a rogue system administrator.”

- 1980s threat model
  - Can’t trust the hosts
    - Compromised hosts
    - Untrusted insiders on internal networks
    - Anyone can connect to public Internet
  - But network is still trusted
TCP/IP Security (today)

- Can’t trust the network either
  - Network equipment can be compromised
  - Untrusted network operators
  - Anyone can access the physical channel of wireless networks
**Attacker Models**

- **Person in the middle:** can see, block, and modify traffic
  - E.g., attacker controls wifi access point

- **Passive:** Eavesdrop on traffic
  - E.g., attacker has passive tap or recorded traces

- **Off-path:** attacker can inject traffic into network
  - Anyone with access to network
No Confidentiality

- Who can see the packets you send?
  - Network (routers, switches, access points, etc.)
  - Unprotected WiFi network: everyone within range
    - WPA2 Personal (PSK): everyone on same network
  - Non-switched (i.e. old school) Ethernet: everyone on same network
  - Switched Ethernet: everyone gets their own link, but... sometimes not
No Authentication

- TCP/IP offers no authentication of packets
  - Source address in IP header is set by sender

- Attacker with direct access to network (including PitM) can spoof source address
  - Spoof: forge, set to arbitrary value

- Connectionless protocols (UDP) especially vulnerable

- Some consequences:
  - Can blast packets at a target and, by using spoofed source addresses, make it look like someone else is attacking them
  - Can try to interfere with existing communications between hosts (e.g., by injecting packets purporting to be part of that communication)
  - Can’t count on source address for authentication (but we still do... e.g., UCSD Library)
Link Layer interception

- Physical channel is often shared by multiple hosts on the local network.
  - Examples: open WiFi, non-switched Ethernet

- Link layer controls access to the physical medium.
  - Also known as the Media Access Control (MAC) layer.
  - Apologies: acronym collision

- How to make sure each host only gets frames addressed to it?

- Each host is responsible for picking up frames addressed to it and ignoring the others.
  - Honor system!

- Filtering typically happens on the network card (or equivalent).
  - Only frames addressed to this host are parsed and passed on to the layer above.

- Many support “promiscuous” mode – all frames are picked up.
But you can intercept in other ways too...

- Host configuration
- Address binding (IP to MAC)
- Routing
Network Routing

- Say I want to send packet to 8.8.8.8 ...

- Step 1: Is destination on local network?
  - Check subnet masks of local networks
Status: **Connected**

Ethernet 1 is currently active and has the IP address 132.239.17.19.

Configure IPv4: Manually

**IP Address:** 132.239.17.19

**Subnet Mask:** 255.255.255.0

**Router:** 132.239.17.1

**DNS Server:** 132.239.0.252, 128.54.16.2, 8.8.8.8

**Search Domains:** ucsd.edu

**802.1X:** WPA: UCSD-P...
Network Routing

- Say I want to send packet to 8.8.8.8 ...

- Step 1: Is host on local network?
  - Local?: send directly
  - Not local?: send via *default gateway* (aka the router)
Status: Connected
Ethernet 1 is currently active and has the IP address 132.239.17.19.

Configure IPv4: Manually
IP Address: 132.239.17.19
Subnet Mask: 255.255.255.0
Router: 132.239.17.1
DNS Server: 132.239.0.252, 128.54.16.2, 8.8.8.8
Search Domains: ucsd.edu
802.1X: WPA: UCSD-P...
How do I know all this stuff?
- The address of the router, my own IP address, the netmask, the address of the DNS server, etc

**Dynamic Host Configuration Protocol (DHCP)**
- Automatically configures each new host attached to network
- Basic idea
  - Host broadcasts “DHCP discover” on local network (special broadcast address)
  - DHCP server responds with information for your host (IP addr, gateway, etc)
- What if someone listens for DHCP requests and sends answer?
  - Can tell someone to use the router of their choosing (and DNS server)
  - One partial defense, **dhcp snooping**: network switch configured to block DHCP messages from non-trusted hosts that aren’t known to be DHCP servers
Network Routing

- Say I want to send packet to 8.8.8.8 ...

- Step 1: Is host on local network?
  - Local?: send directly
  - Not local?: send via default gateway

- Step 2: Create IP packet

- Step 3: Create and send link layer (e.g. Ethernet) frame
  - 48 bit source address, 48 bit destination address
Network Routing

- Ethernet frame:
  - Host needs to fill in Ethernet destination address
    - MAC address of host on local network
    - MAC address of gateway for host not on local network
  - How to find Ethernet address from an IP address?
Address Resolution Protocol (ARP)

- Address Resolution Protocol (ARP)
  - used to query hosts on local network to get link-layer address for an IP address

- Protocol sketch
  - Alice (looking for Bob’s IP) broadcasts an ARP request:
    - “What is the MAC address of 10.0.0.3?”
  - Bob sees broadcast (asking about his IP address) and replies:
    - “The MAC address of 10.0.0.3 is 01:02:03:04:05:06.”
  - Alice sends IP packet for 10.0.0.3 in an Ethernet frame to 01:02:03:04:05:06.
Address Resolution Protocol (ARP)

Looking for physical address of a node with IP address 141.23.56.23

The MAC address of destination is broadcast address: 0xFFFFFFFFFFFFFFF

The node physical address is A46EF45983AB

Link

a. ARP request is broadcast

b. ARP reply is unicast

IP = 141.23.56.23
Address Resolution Protocol (ARP)

- ARP messages are link-layer frames (e.g. Ethernet/WiFi)
- ARP requests are broadcast (on the local subnet)
- Anyone can send an ARP reply
ARP Spoofing

- Since
  - ARP requests are broadcast (on the local subnet)
  - Anyone can send an ARP reply

- Attacker on the network can impersonate any other host
  - Who has the MAC address for IP address 8.8.8.8?
  - “Uh... I do... send your traffic to me...”

- Mitigation
  - **Static** ARP tables
    - Impractical for all but small fixed networks
  - “Port binding” on switch
    - Restrict MAC and IP addresses allowed to a single port on a switch at a time; first mover wins
    - Might not work well for WiFi... why not?
  - Depend on higher level host authentication to save you
    - E.g. SSH or TLS
Problems With Addressing

- This problem repeats at every protocol layer:
  - Source needs to send something to destination.
  - How to know which address corresponds to name?
    - Domain name to IP address
    - IP routing
    - IP address to Ethernet address
    - ...
Network Routing

- Say I want to send packet to 8.8.8.8 ...
- Step 1: Is host on local network?
  - Local?: send directly
  - Not local?: send via default gateway
- Step 2: Create IP packet
- Step 3: Create and send link layer (e.g. Ethernet) frame
- Step 4: Gateway picks next router in path and forwards the IP packet
  - Repeat until destination is reached
  - How to know which router to forward to next?
Border Gateway Protocol (BGP)

- Border Gateway Protocol (BGP) is used to manage IP routing information between networks on the Internet.

- Each BGP node maintains connections to a set of trusted neighbors.
  - Connections between neighbors may be (weakly) authenticated.

- Neighbors share routing information.
  - I can reach network X directly, I can reach network Y via ISP Z, I can...

- No authorization.
  - Malicious (or malfunctioning) BGP nodes may provide incorrect routing information that redirects IP traffic.
BGP Hijacking (just two examples)

- 2008 Pakistan tried to block YouTube within the country
  - Pakistan Telecom claimed ownership of YouTube’s IP block via BGP
  - BGP nodes forwarded this routing information
  - YouTube is “sinkholed” globally

- 2018 MyEtherWallet.com compromised, $100,000’s reported stolen
  - Attackers used BGP hijacking to claim ownership of a chunk of Amazon Route 53 (DNS) addresses
  - Used hijacked DNS traffic to direct MyEtherWallet.com-bound traffic to attackers’ servers in Russia
IP Spoofing Attacks

▪ There is no authentication in Link or Internet layers

▪ Even if routing is correct, Eve can still spoof Alice’s IP address
  – Eve can send IP packets claiming to be from Alice
  – Eve may not be able to receive IP packets addressed to Alice

▪ UDP: trivial
  – Stateless protocol, each datagram is independent of others

▪ TCP: more complicated, but still possible
  – Two endpoints maintain a shared state
  – Attacker must be able to guess it
TCP Connection Spoofing

- Eve needs to complete the TCP three-way handshake between “Alice” and Bob
- Eve can’t see traffic between Alice and Bob
  - “TCP off-path attack”
- Eve needs to guess initial sequence number $y$ in order to correctly ACK Bob’s SYN
Recall: Three-Way Handshake
Eve’s Three-Way Handshake problem

- Eve (claiming to be Alice) does not see Bob’s response

TCP Connection Spoofing

- The sequence number field is 32 bits

- Early implementations just incremented a global counter used to initialize sequence numbers (ISN) for TCP connections
  - RFC 793 requires counter incrementing every 4 µs (250 kHz)
  - Early BSD Unix kernels incremented by a large constant every second
  - Attack: Eve could talk to Bob directly get an ISN, then spoof a SYN from Alice and estimate what ISN he was likely to offer her

- Later pseudo-random number generators were used
  - Still global, weaknesses in PRNGs allowed guessing
  - We now use cryptographically strong random number generators for picking initial sequence numbers
Example Denial-of-Service (DoS) vulnerability with TCP spoofing [Watson’04]

- Suppose attacker can guess seq. number for an existing connection:
  - Attacker can send Reset (RST) packet to close connection. Results in DoS.
  - Naively, success prob. is 1/2^{32} per guess (32-bit seq. #'s).
  - But most systems allow for a large window of acceptable seq. #'s corresponding to number of outstanding packets in flight (e.g., if you have 100MB in flight)
    - Achievable success probability.

- Attack is most effective against long lived connections (expensive to set up again; BGP)
Related issue: Blind port scanning

- Context: Attackers would like to know what TCP services are offered on a particular host (so it knows how to attack them)

- Port scan:
  - Send TCP SYN to each port number on the host
  - See if you get a SYN/ACK back (there is a service at that port number) or a RST (there is nothing there)

- Problem
  - You expose your source address when doing this
  - Attackers would like to be able to do it anonymously...
Related issue: Blind port scanning

- **Key trick:** exploiting *IP identification* field in the IP header
  - Recall the IP identification field was used for fragmentation
  - Hosts need to ensure that it is unique across packets have outstanding
  - An super easy way to do this is to use a global counter on the host
    (increment after you send each packet)
    - If host A sends a pkt with id=5, then next pkt will have id=6, followed by id=7, etc
    - So if you *receive* a pkt from host A at time $t_1$ with id =10, and another packet at time $t_2$ with id=12, you can *infer*... that host A sent another packet somewhere between $t_1$ and $t_2$
Blind port scanning

- **1. SYN**
- **2. RST** if id=x
- **3. SYN, srcIP=S, dstPort=X**
- **4. RST** if port closed or **SYN/ACK** if open
- **5. RST** if **SYN/ACK**, else **nothing**
- **6. SYN**
- **7. Response from S.... What is value of id?**
  - x+1 or x+2?
Additional Resources

- **Wireshark**
  - [https://www.wireshark.org/](https://www.wireshark.org/)

- **Attacking Network Protocols**
  - By James Forshaw
  - [https://nostarch.com/networkprotocols](https://nostarch.com/networkprotocols)