Internet Protocol Security
CS155 Computer and Network Security
What is the Internet?

Global network that lets hosts communicate

Internet provides best-effort delivery of **packets** between hosts

**Packet**: a structured sequence of bytes

- **Header**: metadata used by network
- **Payload**: user data to be transported

Packets are forwarded by routers from sender to destination host
Internet Protocol (IP) defines what packets that cross the Internet need to look like to be processed by routers.

Every host is assigned a unique identifier ("IP Address").

Every packet has an IP header that indicates its sender and receiver.

Routers forward packet along to *try* to get it to the destination host.

Rest of the packet should be ignored by the router.
IP Addresses

**IPv4:** 32-bit host addresses
Written as 4 bytes in form A.B.C.D
where A,...,D are 8 bit integers in decimal
(called *dotted quad*) e.g. 192.168.1.1

**IPv6:** 128 bit host addresses
Written as 16 bytes in form AA:BB::XX:YY:ZZ
where AA,...,ZZ are 16 bit integers in hexadecimal
and :: implies zero bytes
e.g. 2620:0:e00:b::53 = 2620:0:e00:b:0:0:0:53
IPv4 Header

Instruct routers and hosts what to do with a packet
All values are filled in by the sending host
Destination Address

Sender sets destination address
Routers try to forward packet to that address
Source Address

Source Address (sender)
Sender fills in. Routers due not verify.
Checksum

16-bit Simple Header checksum (filled in by sender)
IP Security

Client is trusted to embed correct source IP

- Easy to override using lower level network sockets
- **Libnet**: a library for formatting raw packets with arbitrary IP headers

Anyone who owns their machine can send packets with arbitrary source IP

- Denial of Service Attacks
- Anonymous infection (if one packet)
Internet Protocol (IP)

Yes:
Routing. If host knows IP of destination host, route packet to it.
Fragmentation and reassembly: Split data into packets and reassemble
Error Reporting: (maybe, if you’re lucky) tell source it dropped your packet

No:
Protocol Layering

- How does Application structure data?
  - DNS
  - SSH
  - FTP
  - SMTP
  - NNTP
  - HTTP

- How do I get to the right service? How do I have a reliable “stream” of data?
  - UDP
  - TCP

- How do I get to final destination?
  - IP

- How do I get to next hop?
  - Cellular
  - WiFi
  - Ethernet

- How do I have a reliable “stream” of data?
  - Radio
  - Copper
  - Fiber
Protocol Layering

Networks use a stack of layers

Lower layers provide services to layers above
  Don’t care what higher layers do

Higher layers use services of layers below
  Don’t care how lower layers implement services

Layers define abstraction boundaries
  At a given layer, all layers above and below are opaque
Protocol Layering

How does Application structure data?

How do I get to the right service? How do I have a reliable “stream” of data?

How do I get to final destination?

How do I get to next hop?
Protocol N1 can use the services of lower layer protocol N2
A packet P1 of N1 is encapsulated into a packet P2 of N2
The payload of p2 is p1
The control information of p2 is derived from that of p1
Link Layer

Model assumed that hosts can deliver and accept packets from Internet routers

In practice, hosts not connected directly to router

Link layer provides connectivity between hosts and routers
Ethernet

Most common Link Layer Protocol. Let’s you send packets to other local hosts.

At layer 2 (link layer) packets are called *frames*

MAC addresses: 6 bytes, universally unique

EtherType gives layer 3 protocol in payload
0x0800: IPv4
0x0806: ARP
0x86DD: IPv6
Ethernet

Originally broadcast. Every local computer got every packet.
Switched Ethernet

With switched Ethernet, the switch *learns* at which physical port each MAC address lives based on MAC source addresses.

If switch knows MAC address M is at port P, it will only send a packet for M out port P.

If switch does not know which port MAC address M lives at, will broadcast to all ports.
Ethernet

```
zakir@scratch-01:~$ ifconfig
ens160: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
  inet 10.216.2.64  netmask 255.255.192.0  broadcast 10.216.63.255
  inet6 fe80::250:56ff:fe86:b203  prefixlen 64  scopeid 0x20<link>
  ether 00:50:56:86:b2:03  txqueuelen 1000  (Ethernet)
  RX packets 1404151714  bytes 1784388363701 (1.7 TB)
  RX errors 0  dropped 73  overruns 0  frame 0
  TX packets 1155689210  bytes 6010503085464 (6.0 TB)
  TX errors 0  dropped 0  overruns 0  carrier 0  collisions 0
```
Two Problems

Local: How does a host know what MAC address their destination has?

Internet: How does each router know where to send each packet next?
ARP: Address Resolution Protocol

ARP is a Network protocol that lets hosts map IP addresses to MAC addresses.

Host who needs MAC address M corresponding to IP address N broadcasts an ARP packet to LAN asking, “who has IP address N?”

Host that has IP address N will reply, “IP N is at MAC address M.”
ARP Packet

*Note: The length of the address fields is determined by the corresponding address length fields.*
Any host on the LAN can send ARP requests and replies: *any host can claim to be another host on the local network!*  
This is called **ARP spoofing**

This allows any host X to force IP traffic between any two other hosts A and B to flow through X (*MitM!*)

- Claim $N_A$ is at attacker’s MAC address $M_X$
- Claim $N_B$ is at attacker’s MAC address $M_X$
- Re-send traffic addressed to $N_A$ to $M_A$, and vice versa
Routing (BGP)

BGP (Border Gateway Protocol): protocol that allows routers to exchange information about their routing tables.

Each router announces what it can route to all of its neighbors.

Every router maintains a global table of routes.
Pakistan hijacks YouTube

On 24 February 2008, Pakistan Telecom (AS 17557) began advertising a small part of YouTube’s (AS 36561) assigned network

PCCW (3491) did not validate Pakistan Telecom’s (17557) advertisement for 208.65.153.0/24

Youtube offline.
Protocol Layering

How do I get to final destination?
How do I get to next hop?
How do I get to the right service?
How do I have a reliable “stream” of data?
How does Application structure data?
Most Internet applications want a data stream abstraction (not best-effort packets)

Application on host X wants to send a sequence of bytes to application on host Y. Wants reliable, in-order delivery of data

Transmission Control Protocol (TCP) provides a data stream abstraction using a best-effort packet transport (IP)
Transmission Control Protocol

Have: network that will deliver packets
   Packets may be dropped, re-ordered, duplicated

Want:
   Abstraction of a stream of bytes between *applications* on different hosts
   Bytes delivered reliably and in-order
Ports

Each application is identified by a *port number*

TCP connection established between port A on host X to port B on host Y

Ports are 1–65535 (16 bits)

Some destination port numbers used for specific applications by convention
## Ports

<table>
<thead>
<tr>
<th>Port</th>
<th>Application</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>HTTP (Web)</td>
</tr>
<tr>
<td>443</td>
<td>HTTPS (Web)</td>
</tr>
<tr>
<td>25</td>
<td>SMTP (mail)</td>
</tr>
<tr>
<td>67</td>
<td>DHCP (host config)</td>
</tr>
<tr>
<td>22</td>
<td>SSH (secure shell)</td>
</tr>
<tr>
<td>514</td>
<td>RSH (remote shell)</td>
</tr>
<tr>
<td>23</td>
<td>Telnet</td>
</tr>
</tbody>
</table>
Transmission Control Protocol

Bytes in application data stream numbered with a 32-bit sequence number

Data sent in segments: sequences of contiguous bytes sent in a single IP datagram

There are two logical data streams in a TCP session, one in each direction
Transmission Control Protocol

Sequence number in packet header is seq. number of first byte of payload
Acknowledgement number is seq number of next expected byte of stream in opposite direction
Transmission Control Protocol

- Sender sends 3 byte segment
- Sequence number indicates where data belongs in byte sequence (at byte 401)
  - Note: Wireshark shows relative sequence numbers
TCP ACKs

- Receiver acknowledges received data
  - Sets ACK flag in TCP header
  - Sets acknowledgement number to indicate next expected byte in sequence
Transmission Control Protocol

- Sender may send several segments before receiving acknowledgement
Transmission Control Protocol

• Sender may send several segments before receiving acknowledgement
• Receiver always acknowledges with seq. no. of next expected byte
Transmission Control Protocol

• What if the first packet is dropped in network?
• Receiver always acknowledges with seq. no. of next expected byte
Transmission Control Protocol

- *What if the first packet is dropped in network?*
- Receiver always acknowledges with seq. no. of next expected byte
- Sender retransmits lost segment
Transmission Control Protocol

• *What if the first packet is dropped in network?*
• Sender retransmits lost segment
• Receiver always acknowledges with seq. no. of next expected byte
Transmission Control Protocol

- ACKs may be piggybacked on data flowing in opposite direction or sent without data
- All packets after initial connection setup will carry an acknowledgement
- Sequence numbers wrap around: ..., $2^{32}-2$, $2^{32}-1$, 0, 1, 2, ...
Starting a Connection

- **Client**: State changes to **SYN-SENT**
  - SYN-ACK
    - seq: 200
    - ack: 101
  - State changes to **ESTABLISHED**

- **Server**: SYN
  - seq: 100
  - State changes to **SYN-RECEIVED**
  - ACK
    - seq: 101
    - ack: 201
  - State changes to **ESTABLISHED**
Ending a Connection

Sends packet with FIN flag set
Must have ACK flag with valid seqnum

Peer receiving FIN packet acknowledges receipt of FIN packet with ACK
FIN “consumes” one byte of seq. number

Eventually other side sends packet with FIN flag set:
This terminates the TCP session
TCP Connection Reset

TCP designed to handle possibility of spurious TCP packets (e.g. from previous connections)

Packets that are invalid given current state of session generate a reset
  If a connection exists, it is torn down
  Packet with RST flag sent in response

If a host receives a TCP packet with RST flag, it tears down the connection
TCP Connection Spoofing

Can we impersonate another host when *initiating* a connection?

Off-path attacker can send initial SYN to server … … but cannot *complete three-way handshake without seeing the server’s sequence number*

1 in $2^{32}$ chance to guess right if initial sequence number chosen uniformly at random
TCP Reset Attack

Can we reset an *existing* TCP connection?

Need to know port numbers (16 bits)
   Initiator’s port number usually chosen random by OS
   Responder’s port number may be well-known port of service

There is leeway in sequence numbers B will accept
   Must be within window size (32-64K on most modern OSes)

1 in \(2^{16+32}/W\) (where \(W\) is window size) chance to guess right
UDP (User Datagram Protocol)

Sometimes we do only want best-effort delivery

**User Datagram Protocol (UDP)** is a transport layer protocol that is essentially a wrapper around IP

Adds ports to demultiplex traffic by applications
DNS

Application-layer protocols (and people) usually refer to Internet host by host name (e.g., google.com)

DNS is a delegatable, hierarchical name space
A DNS server has a set of records it authoritatively knows about.

$ dig bob.ucsd.edu

;; Got answer:
;; ->>HEADER<<- opcode: QUERY, status: NOERROR, id: 30439
;; flags: qr aa rd ra; QUERY: 1, ANSWER: 1, AUTHORITY: 3, ADDITIONAL: 6

;; QUESTION SECTION:
bob.ucsd.edu. IN A

;; ANSWER SECTION:
bob.ucsd.edu. 3600 IN A 132.239.80.176

;; AUTHORITY SECTION:
ucsd.edu. 3600 IN NS ns0.ucsd.edu.
ucsd.edu. 3600 IN NS ns1.ucsd.edu.
ucsd.edu. 3600 IN NS ns2.ucsd.edu.
DNS Root Name Servers

In total, there are 13 main DNS root servers, each of which is named with the letters 'A' to 'M'.

<table>
<thead>
<tr>
<th>HOSTNAME</th>
<th>IP ADDRESSES</th>
<th>MANAGER</th>
</tr>
</thead>
<tbody>
<tr>
<td>a.root-servers.net</td>
<td>198.41.0.4, 2001:503:ba3e::2:30</td>
<td>VeriSign, Inc.</td>
</tr>
<tr>
<td>b.root-servers.net</td>
<td>199.9.14.201, 2001:500::b</td>
<td>University of Southern California (ISI)</td>
</tr>
<tr>
<td>c.root-servers.net</td>
<td>192.33.4.12, 2001:500:2::c</td>
<td>Cogent Communications</td>
</tr>
<tr>
<td>d.root-servers.net</td>
<td>199.7.91.13, 2001:500:2::d</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>e.root-servers.net</td>
<td>192.203.230.10, 2001:500:a8::e</td>
<td>NASA (Ames Research Center)</td>
</tr>
<tr>
<td>f.root-servers.net</td>
<td>192.5.5.241, 2001:500:2::f</td>
<td>Internet Systems Consortium, Inc.</td>
</tr>
<tr>
<td>g.root-servers.net</td>
<td>192.112.36.4, 2001:500:12::d0d</td>
<td>US Department of Defense (NIC)</td>
</tr>
<tr>
<td>h.root-servers.net</td>
<td>198.97.190.53, 2001:500:1::53</td>
<td>US Army (Research Lab)</td>
</tr>
<tr>
<td>i.root-servers.net</td>
<td>192.36.148.17, 2001:7fe::53</td>
<td>Netnod</td>
</tr>
<tr>
<td>k.root-servers.net</td>
<td>193.0.14.129, 2001:7fd::1</td>
<td>RIPE NCC</td>
</tr>
<tr>
<td>l.root-servers.net</td>
<td>199.7.83.42, 2001:500:9f::42</td>
<td>ICANN</td>
</tr>
<tr>
<td>m.root-servers.net</td>
<td>202.12.27.33, 2001:dc3::35</td>
<td>WIDE Project</td>
</tr>
</tbody>
</table>
Caching

DNS responses are cached
  Quick response for repeated translations
  NS records for domains also cached

DNS negative queries are cached
  Save time for nonexistent sites, e.g. misspelling

Cached data periodically times out
  Lifetime (TTL) of data controlled by owner of data
  TTL passed with every record
DNS Packet

DNS requests sent over UDP

**Four sections:** questions, answers, authority, additional records

**Query ID:**
16 bit random value
Links response to query
# Request

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4</td>
</tr>
<tr>
<td>Header Length</td>
<td>20</td>
</tr>
<tr>
<td>Type of Service (TOS)</td>
<td>0</td>
</tr>
<tr>
<td>Protocol</td>
<td>17</td>
</tr>
<tr>
<td>Source IP</td>
<td>68.94.156.1</td>
</tr>
<tr>
<td>Destination IP</td>
<td>192.26.92.30</td>
</tr>
<tr>
<td>Source Port</td>
<td>5798</td>
</tr>
<tr>
<td>Destination Port</td>
<td>53</td>
</tr>
</tbody>
</table>

**UDP**

- **Length:** 5798
- **Checksum:** 0

**Question ID (QID):** 43561

- **Question Count:** 1
- **Answer Count:** 0
- **Authority Count:** 0
- **Additional Answer Count:** 0

**Query:**

What is A record for www.unixwiz.net?
**Response**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Source IP</strong></td>
<td>192.26.92.30</td>
</tr>
<tr>
<td><strong>Destination IP</strong></td>
<td>68.94.156.1</td>
</tr>
<tr>
<td><strong>Source Port</strong></td>
<td>53</td>
</tr>
<tr>
<td><strong>Destination Port</strong></td>
<td>5798</td>
</tr>
</tbody>
</table>

**UDP**

<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>QID</strong></td>
<td>43561</td>
</tr>
<tr>
<td><strong>Question Count</strong></td>
<td>1</td>
</tr>
<tr>
<td><strong>Answer Count</strong></td>
<td>0</td>
</tr>
<tr>
<td><strong>Authority Count</strong></td>
<td>2</td>
</tr>
<tr>
<td><strong>Additional Record Count</strong></td>
<td>2</td>
</tr>
</tbody>
</table>

**Query:**

What is A record for www.unixwiz.net?

<table>
<thead>
<tr>
<th><strong>A Records</strong></th>
<th><strong>TTL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>linux.unixwiz.net</td>
<td>2 dy</td>
</tr>
<tr>
<td>cs.unixwiz.net</td>
<td>2 dy</td>
</tr>
</tbody>
</table>

**Glue Records**

<table>
<thead>
<tr>
<th><strong>A Records</strong></th>
<th><strong>TTL</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>linux.unixwiz.net</td>
<td>1 hr</td>
</tr>
<tr>
<td>cs.unixwiz.net</td>
<td>1 hr</td>
</tr>
</tbody>
</table>
Authoritative Response

IP

<table>
<thead>
<tr>
<th>src IP = 64.170.162.98</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst IP = 68.94.156.1</td>
</tr>
</tbody>
</table>

UDP

<table>
<thead>
<tr>
<th>src port = 53</th>
</tr>
</thead>
<tbody>
<tr>
<td>dst port = 5798</td>
</tr>
</tbody>
</table>

QID = 43562

Question count = 1
Answer count = 1
Authority count = 2
Addl. Record count = 2

QR=1 - this is a response
AA=1 - Authoritative!
RA=0 - recursion unavailable

What is an A record for www.unixwiz.net?

www.unixwiz.net A = 8.7.25.94 1 hr
unixwiz.net NS = linux.unixwiz.net 2 dy
unixwiz.net NS = cs.unixwiz.net 2 dy

linux.unixwiz.net A = 64.170.162.98 1 hr
cs.unixwiz.net A = 8.7.25.94 1 hr
DNS Security

Users/hosts trust the host-address mapping provided by DNS
Used as basis for many security policies:
   Browser same origin policy, URL address bar
Interception of requests or compromise of DNS servers can result in incorrect or malicious responses
DNSSEC Fixes, but nobody uses. Use TLS!!
DNS Spoofing

**Scenario:** DNS client issues query to server

Attacker would like to inject a fake reply
  Attacker does not see query or real response

How does client authenticate response?
DNS Spoofing

How does client authenticate response?

UDP port numbers must match
   Destination port usually port 53 by convention

16-bit query ID must match
DNS Cache Poisoning

Recursive resolvers cache records to avoid repeating recursive resolution process for each query.

Lifetime of record determined by record TTL. Could also be evicted from cache due to limited memory.

Injecting spoofed records into a resolver’s cache is called *DNS cache poisoning*. No protocol-defined way for to refresh cached record.
**Early Attack Strategy**


2. The user’s computer asks the targeted name server to translate www.BadGuysAreUs.com into an IP address.

3. The targeted name server has not cached the address, so the query is routed through a root name server, a .com name server, and finally the BadGuysAreUs.com name server.

4. The BadGuysAreUs name server responds with an IP address but adds a false IP address for a completely different Web site, www.paypal.com.

5. The targeted name server stores the false IP address for paypal.com.

6. When people using this name server attempt to go to www.paypal.com, they are directed to a Web site that looks like PayPal’s but works only to harvest their user names and passwords.
Kaminsky Attack

- Victim machine visits attacker’s web site, downloads Javascript
- User browser queries `a.bank.com`
- Local DNS resolver queries `.com`
- 256 responses: Random QID `y_1, y_2, ...`
- `NS bank.com=ns.bank.com`
- `A ns.bank.com=attackerIP`
- Attacker wins if `∃ j: x_1 = y_j`
- Response is cached and attacker owns `bank.com`
Try Again!

 victim machine visits attacker's web site, downloads Javascript

user browser → Query: a.bank.com → local DNS resolver

a.bank.com QID=x₁ → .com

response

256 responses: Random QID y₁, y₂, ...

NS bank.com=ns.bank.com
A ns.bank.com=attackerIP

attacker wins if ∃ j: x₁ = yᵢ
response is cached and attacker owns bank.com

attacker
Defenses

Increase QueryID. But how? Don’t want to change packet.

Randomize src port, additional 11 bits

- Now attack takes several hours
Conclusion

The network is out to get you.