The network ...
Internet Services

• Internet Protocol (IP)
  • Really refers to a whole collection of protocols making up the vast majority of the Internet

• Routing
  • How these packets move from place to place?

• Network management
  • Administrators have to maintain the services and infrastructure supporting everyone’s daily activities

• Quality of service
  • How do we ensure that we get our fair share of network resources, e.g., bandwidth?
Reality

- Networks are not secure ..
- Never meant to be ....

- Designers of Internet saw security as largely orthogonal to network services ..
Address Resolution Protocol (ARP)

- Protocol used to map IP address onto the physical layer addresses (MAC)
  1) ARP request: who has x.x.x.x?
  2) ARP response: me!
- Policy: last one in wins
- Used to forward packets on the appropriate interfaces by network devices (e.g., bridges)
- Attack: replace good entries with your own
- Leads to
  - Session hijacking
  - Man-in-the-middle attacks
  - Denial of service, etc.

Q: Why would you want to spoof an IP address?
TCP/IP uses a *three-way handshake* to establish a connection

1. C -> S: $Q_C$ where sequence numbers $Q_C$
2. S -> C: $Q_S$, ack($Q_C$) and $Q_S$ are nonces
3. C -> S: ack($Q_S$) … then send data

However assume the bad guy does not hear msg 2, if he can guess $Q_S$, then he can get S to accept whatever data it wants (useful if doing IP authentication, e.g., “rsh”)

---

**Sequence number prediction**

- Client
- Server
- Adversary
Routing Manipulation

• RIP - routing information protocol
  • Distance vector routing protocol
  • Routers exchange reachability and “distance” vectors for all the sub-networks within (a typically small) domain
  • Use vectors to decide which is best, notification of changes is propagated quickly
• So, the big problem is that you receive vast amounts of data that a router uses to form the routing table
  • So, just forge that, and the game is up
  • Manipulate paths, DOS, hijack connections, etc.
• Solutions:
  • Authenticate data, but this is less than obvious how to do this efficiently (a whole lot of people are trying)
**Internet Control Message Protocol (ICMP)**

- ICMP is used as a control plane for IP messages
  - Ping (connectivity probe)
  - Destination Unreachable (error notification)
  - Time-to-live exceeded (error notification)
- These are used for good purposes, and are largely indispensable tools for network management and control
  - Error notification codes can be used to reset connections without any
- Solution: verify/sanity check sources and content
  - ICMP “returned packets”
- Real solution: filter most of ICMP, ignore it
In 1996, someone discovered that many operating systems, routers, etc. could be crash/rebooted by sending a single malformed packet.

It turns out that you can send a IP packet larger than 65,535 ($2^{16}$), it would crash many things.

The real reason lies in the way fragmentation works.

- It allows somebody to send a packet bigger than IP allows
- Which blows up most fixed buffer size implementations
- ... and dumps core, blue screen of death, etc.

Note: this is not really ICMP specific, but easy (try it)

```
% ping -l 65510 your.host.ip.address
```

This was a popular pastime of early hackers.

Solution: patch the implementations.
- Post office protocol - mail retrieval
  - Passwords passed in the clear (duh)
  - Solution: SSL, SSH, Kerberos

- Simple mail transport protocol (SMTP) - email
  - Nothing authenticated: SPAM
  - Nothing hidden: eavesdropping
  - Solution: your guess is as good as mine

- File Transfer protocol - file retrieval
  - Passwords passed in the clear (duh)
  - Solution: SSL, SSH, Kerberos
• DNS maps between IP address (12.1.1.3) and domain and host names (ada.cse.psu.edu)

• How it works: the “root” servers redirect you to the top level domains (TLD) DNS servers, which redirect you to the appropriate sub-domain, and recursively ….

• Note: there are 13 “root” servers that contain the TLDs for .org, .edu, and country specific registries (.fr, .ch)
DNS Vulnerabilities

• Nothing is authenticated, so really the game is over
  • You can not really trust what you hear …
  • But, many applications are doing just that.
  • Spoofing of DNS is really dangerous
• Moreover, DNS is a catalog of resources
  • Zone-transfers allow bulk acquisition of DNS data
  • … and hence provide a map for attacking the network
• Lots of opportunity to abuse the system
  • Relies heavily on caching for efficiency -- cache pollution
  • Once something is wrong, it can remain that way in caches for a long time (e.g., it takes a long time flush)
DNSSEC

- A standard-based (IETF) solution to security in DNS
- Prevents data spoofing and corruption
- Public key based solution to verifying DNS data
- Authenticates
  - Communication between servers
  - DNS data
- Public keys (a bootstrap for PKI?)
• Securing the DNS records
  • Each domain signs their “zone” with a private key
  • Public keys published via DNS
  • *Indirectly* signed by parent zones
  • Ideally, you only need to self-signed root, and follow keys down the hierarchy

---

**DNSSEC Mechanisms**

root \(\rightarrow\) .edu \(\rightarrow\) psu.edu \(\rightarrow\) cse.psu.edu
DNSSEC challenges

• Incremental deployability
  • Everyone has DNS, can’t assume a flag day

• Resource imbalances
  • Some devices can’t afford real authentication

• Cultural
  • Most people don’t have any strong reason to have secure DNS ($$$ not justified in most environments)
  • Lots of transitive trust assumptions (you have no idea how the middlemen do business)

• Take away: DNSsec will be deployed, but it is unclear whether it will be used appropriately/widely
Filtering: Firewalls

• Filtering traffic based on *policy*
  • Policy determines what is acceptable traffic
  • Access control over traffic
  • *Accept* or *deny* *policy*

• May perform other duties
  • Logging (forensics, SLA)
  • Flagging (intrusion detection)
  • QoS (differentiated services)
• **Blacklisting** - specifying specific connectivity that is explicitly disallowed
  - E.g., prevent connections from badguys.com

• **Whitelisting** - specifying specific connectivity that explicitly allowed
  - E.g., allow connections from goodguys.com

• These is useful for IP filtering, SPAM mitigation, …

• Q: What access control policies do these represent?
Stateful/Stateless and Proxy/Transparent

• Single packet contains insufficient data to make access control decision
• State allows historical context consideration
• Firewall collects data over time
  • e.g., TCP packet is part of established session
• Firewalls can affect network traffic
  • Transparent: appear as a single router (network)
  • Proxy: receives, interprets, and reinitiates communication (application)
  • Transparent good for speed (routers), proxies good for complex state (applications)
Example Server Firewall

TCP

UDP

IP

Interface

Sendmail

Apache

named
Example Server Firewall

- **TCP**
  - Ports: 1, 2, 3, 25, 60, 2^16

- **UDP**
  - Ports: 1, 2, 3, 42, 2^16

- **IP**

- **Interface**

- **Sendmail**

- **Apache**

- **named**
Firewall Policy

- Specifies what traffic is (not) allowed
  - Maps attributes to address and ports
  - Example: HTTP should be allowed to any external host, but inbound only to web-server

<table>
<thead>
<tr>
<th>Source Address</th>
<th>Source Port</th>
<th>Destination Address</th>
<th>Destination Port</th>
<th>Protocol</th>
<th>Flags</th>
<th>Actions</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>*</td>
<td>1.1.1.1</td>
<td>80</td>
<td>TCP</td>
<td>SYN</td>
<td>Accept</td>
</tr>
<tr>
<td>1.1.1.*</td>
<td>*</td>
<td>*</td>
<td>80</td>
<td>TCP</td>
<td>SYN</td>
<td>Accept</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>80</td>
<td>TCP</td>
<td></td>
<td>Accept</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>TCP</td>
<td></td>
<td>Deny</td>
</tr>
</tbody>
</table>