Insecure network services

- **NFS (port 2049)**
  - Read/write entire FS as any non-root user given a dir. handle
  - Many OSes make handles easy to guess

- **Portmap (port 111)**
  - Relays RPC requests, making them seem to come from localhost
  - E.g., old versions would relay NFS mount requests

- **FTP (port 21) – server connects back to client**
  - Client can specify third machine for “bounce attack”

- **YP/NIS – serves password file, other info**

- **A host of services have histories of vulnerabilities**
  - DNS (53), rlogin (513), rsh (514), NTP (123), lpd (515), …
  - Many on by default—compromised before OS fully installed

---

Firewalls

- **Separate local area net from Internet**
  - Prevent bad guys from interacting w. insecure services
  - Perimeter-based security

---

Two separable topics

- **Arrangement of firewall and routers**
  - Separate internal LAN from external Internet
  - Wall off subnetwork within an organization
  - Intermediate zone between firewall and rest of network (called demilitarized zone or “DMZ”)
  - Personal firewall on end-user machine

- **How the firewall processes data**
  - Packet filtering router
  - Application-level gateway
    Proxy for protocols such as ftp, smtp, http, etc.
  - Personal firewall
    E.g., disallow telnet connection from email client

---

Packet filtering

- **Filter packets using transport layer information**
  - Examine IP, and ICMP/TCP/UDP header of each packet
  - IP Source, Destination address
  - Protocol
  - TCP/UDP source & destination ports
  - TCP flags
  - ICMP message type

- **Example: coping with vulnerability in lpd**
  - Block any TCP packets with destination port 515
  - Outsiders shouldn’t be printing from outside net anyway

---

Example: blocking forgeries

- **Should block incoming packets “from” your net**
- **Egress filtering: block forged outgoing packets**

---

Example: blocking outgoing mail

- **At Stanford, all mail goes out through main servers**
  - Result of Sircam worm
    …infected & mailed users’ files around as attachments
  - Could have disclosed sensitive information
  - Mail servers now scan attachments for worms
  - Also reduces threat of Stanford being used to spam

- **How to enforce?**
- **Block outgoing TCP packets**
  - If destination port is 25 (SMTP – mail protocol)
  - And if source IP address is not a Stanford mail server
Blocking by default

- Often don’t know what people run on their machines
- In many environments better to be safe:
  - Block all incoming TCP connections
  - Explicitly allow incoming connections to particular hosts
    E.g., port 80 on web server, port 25 on mail server, …
  - But still must allow outgoing TCP connections
    (users will revolt if they can’t surf the web)
- How to enforce?
  - Recall all but first packet in TCP flow has ACK flag set
  - Block incoming TCP packets w. SYN flag but not ACK flag

Abnormal fragmentation

- Recall IP fragmentation—Why might this complicate firewalls?

Fragmentation

- Recall IP fragmentation—Why might this complicate firewalls?

Abnormal fragmentation

- Low offset allows second packet to overwrite TCP header at receiving host

Blocking UDP traffic

- Some sites block most UDP traffic
  - UDP sometimes viewed as “more dangerous”
  - Easier to spoof source address
  - Used by insecure LAN protocols such as NFS
- Often more convenient to block only incoming UDP
  - E.g., allow internal machines to query external NTP servers
  - Don’t let external actors to exploit bugs in local NTP software
    (unless client specifically contacts bad/spoofed server)
- Must keep state in firewall – like a NAT
  - Remember ⟨local IP, local port, remote IP, remote port⟩ for each outgoing UDP packet
  - Allow incoming packets that match saved flow
  - Time out flows that have not been recently used

Network intrusion detection

- Many holes exploited over the network
  - Buffer overruns in servers
  - Servers with bad implementations
    ("login -f root", telnet w. LD_LIBRARY_PATH)
- Want to detect people exploiting such bugs
  - Want to detect activities performed by people who’ve penetrated server
    - Setting up IRC bot
    - Running particular commands, etc.
- Do so with network-based intrusion-detection system (IDS)
Detect in network monitor

- Attach IDS machine to DMZ
- Sniff all packets in and out of the network
- Process packets to identify possible intruders
  - Secret, per-network rules identify possible attacks
  - Is it a good idea to keep rules secret?
- React to any threats
  - Alert administrators of problems in real time
  - Switch on logging to enable later analysis of potential attack
  - Take action against attackers – E.g., filter all packets from host that seems to be attacking

Deep packet inspection

- May want to block attacks as they are happening
  - E.g., Stanford can detect your broken software, but can’t force you to patch it
  - But if your PC joins a botnet, it’s Stanford’s problem
  - Best to block attacks as they happen
- Many attacks require particular fingerprints
  - E.g., attack packet may include copy of a worm
- Can amass database of “bad” fingerprints to block
  - Manually or semi-manually widely done, but slow to adapt to new attacks
  - Heuristics can catch attacks as they happen...
- But if such countermeasures were uniformly and widely deployed, attackers would defeat them

Virtual Private Networks (VPNs)

- What if firewall must protect more than one office
- Extend perimeter w. Virtual Private Networks (VPNs)
- Two popular VPN protocols:
  - IPsec encrypts at IP layer (bad for NATs)
  - OpenVPN tunnels IP inside SSL (inside TCP)

ESP high-level view

- Encapsulates one IP packet inside another
- Each endpoint has Security Association DB (SAD)
  - Is a table of Security Associations (SAs)
  - Each SA has 32-bit Security Parameters Index (SPI)
  - Also, source/destination IP addresses, crypto algorithm, keys
- Packets processed based on SPI, src/dest IP address
  - Usually have one SA for each direction betw. two points
- SAD managed “semi-manually”
  - Manually set key
  - Or negotiate it using IKE protocol

ESP details

- Must avoid replays
  - Keep counter for 64-bit sequence number
  - Receiver must accept some packets out of order (e.g., up to 32)
  - Only low 32 bits of sequence number in actual packet (would be bad if you lost 4 billion packets)
- Support for traffic flow confidentiality (TFC)
  - Can pad packets to fixed length
  - Can send dummy packets
- Support for encryption without MAC… Bummer!
  - Rationale: App might be SSL, which has MAC-only mode
  - But then attacker can mess with destination address!

IPsec ESP protocol

- MACed data
- Encrypted data
- Clear text IP packet
- Payload
- Encrypted data
- Integrity tag
SSL/TLS [RFC 5246] Overview

- SSL offers security for HTTP protocol
  - That’s what the padlock means in your web browser
- Authentication of server to client
- Optional authentication of client to server
  - Incompatibly implemented in different browsers
  - CA infrastructure not in widespread use
- Confidentiality of communications
- Integrity protection of communications

Ciphersuites: Negotiating ciphers

- Server authentication algorithm (RSA, DSS)
- Key exchange algorithm (RSA, DHE)
- Symmetric cipher for confidentiality (RC4, DES, AES)
- MAC (HMAC-MD5, HMAC-SHA)

Overview of SSL Handshake

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>Supported ciphers, client random</td>
<td>Chosen cipher, server random, certificate</td>
</tr>
<tr>
<td>Compute keys</td>
<td>Compute keys</td>
</tr>
<tr>
<td>Encrypted pre-master secret</td>
<td>MAC of handshake messages</td>
</tr>
<tr>
<td>MAC of handshake messages</td>
<td>MAC of handshake messages</td>
</tr>
</tbody>
</table>

SSL Handshake

- Client and server negotiate on cipher selection
- Cooperatively establish session keys
- Use session keys for secure communication
- Details
  - Multiple messages per stage
  - Get an idea of protocol in action:
    openssl s_client -connect www.paypal.com:443

Establishing a Session Key

- Server and client both contribute randomness.
- Client sends server a “pre-master secret” encrypted with server’s public key
- Use randomness and pre-master secret to create session keys:
  - Client MAC
  - Server MAC
  - Client Write
  - Server Write
  - Client IV
  - Server IV

Establishing a Session Key

From “SSL and TLS” by Eric Rescorla
Session Resumption

- Problem: Public key crypto expensive
- New TCP connection, reuse master secret.
  - Avoids unnecessary public key cryptography.
- Combines cached master secret with new randomness to generate new session keys.
- Works even when the client IP changes (servers cache on session ID, clients cache on server hostname).

What does CA mean by certificate?

- That a public key belongs to someone authorized to represent a hostname?
- That a public key belongs to someone who is associated in some way with a hostname?
- That a public key belongs to someone who has lots of paper trails associated to a company related to a hostname?
- That the CA has no liability, or $100,000, or $250,000?
- >100-page Certification Practice Statement (CPS)

So many CAs...

CA Convenience vs. Security

- How convenient is a Verisign certificate?
  - Need fee + cooperation from Stanford IT to get one here
  - Good for credit cards, but shuts out many other people
- How trustworthy is a Verisign certificate?
  - In mid-March 2001, VeriSign, Inc., advised Microsoft that on January 29 and 30, 2001, it issued two… [fraudulent] certificates. The common name assigned to both certificates is “Microsoft Corporation.”
  - Microsoft Security Bulletin MS01-017

2-minute stretch