Our “Narrow” Focus

• Yes:
  - Creating a “secure channel” for communication (Part I)
  - Protecting network resources and limiting connectivity (Part II)
  - “Network Security”

• No:
  - Preventing software vulnerabilities & malware, or “social engineering”.
  - “Software Security”
Flashback .. Internet design goals

1. Interconnection
2. Failure resilience
3. Multiple types of service
4. Variety of networks
5. Management of resources
6. Cost-effective
7. Low entry-cost
8. Accountability for resources

Where is security?
Why did they leave it out?

• Designed for connectivity

• Network designed with implicit trust
  ♦ No "bad" guys

• Can’t security be provided at the edge?
  ♦ Encryption, Authentication etc
  ♦ End-to-end arguments in system design
Security Vulnerabilities

- At every layer in the protocol stack!

- Network-layer attacks
  - IP-level vulnerabilities
  - Routing attacks

- Transport-layer attacks
  - TCP vulnerabilities

- Application-layer attacks
IP-level vulnerabilities

- IP addresses are provided by the source
  - Spoofing attacks

- Using IP address for authentication
  - e.g., login with .rhosts

- Some “features” that have been exploited
  - Fragmentation
  - Broadcast for traffic amplification
Fun with IP Spoofing

- The IP addresses are filled in by the originating host
  - Address spoofing
- Using source address for authentication
  - r-utils (rlogin, rsh, rhosts etc..)

Can A claim it is B to the server S?
- ARP Spoofing
  - Can C claim it is B to the server S?
  - Source Routing
Fun with IP Spoofing (Smurf Attack)

Attacking System

Internet

Broadcast Enabled Network

Victim System

15-411: security
ICMP Attacks

- No authentication
- ICMP redirect message
  - Can cause the host to switch gateways
  - Benefit of doing this?
    - Man in the middle attack, sniffing
- ICMP destination unreachable
  - Can cause the host to drop connection
- ICMP echo request/reply
- Many more…
Routing attacks

• Divert traffic to malicious nodes
  ◦ Black-hole
  ◦ Eavesdropping

• How to implement routing attacks?
  ◦ Distance-Vector:
  ◦ Link-state:

• BGP vulnerabilities
Routing attacks

• Divert traffic to malicious nodes
  ✷ Black-hole
  ✷ Eavesdropping

• How to implement routing attacks?
  ✷ Distance-Vector: Announce low-cost routes
  ✷ Link-state: Dropping links from topology

• BGP vulnerabilities
  ✷ Prefix-hijacking
  ✷ Path alteration
Black-hole Attacks

- All packets to destination network get dropped in network

- Causes:
  - Compromised router drops packets directly
  - Compromised router sends incorrect routing info
  - Maliciously crafted BGP packets
  - Modified BGP packets
  - Dropped BGP packets
TCP-level attacks

- **SYN-Floods**
  - Implementations create state at servers before connection is fully established

- **Session hijack**
  - Pretend to be a trusted host
  - Sequence number guessing

- **Session resets**
  - Close a legitimate connection
Session Hijack

First send a legitimate SYN to server

Malicious (M) → Server
1. SYN (ISN_X)
   SRC = X

Server → Trusted (T)
2. SYN(ISN_S1),
   ACK(ISN_X)

15-411: security
Session Hijack

Trusted (T)  
Malicious (M)

Using ISN_S1 from earlier connection guess ISN_S2!
TCP Layer Attacks

• TCP SYN Flooding
  - Exploit state allocated at server after initial SYN packet
  - Send a SYN and don’t reply with ACK
  - Server will wait for 511 seconds for ACK
  - Finite queue size for incomplete connections (1024)
  - Once the queue is full it doesn’t accept requests
TCP Layer Attacks

- TCP Session Poisoning
  - Send RST packet
    - Will tear down connection
  - Do you have to guess the exact sequence number?
    - Anywhere in window is fine
    - For 64k window it takes 64k packets to reset
    - About 15 seconds for a T1
An Example

- Finger @S
- showmount -e
- Send 20 SYN packets to S

- Attack when no one is around
- What other systems it trusts?
- Determine ISN behavior

Shimomura (S) ➔ Finger SYN ➔ Showmount -e ➔ Trusted (T) ➔ Mitnick
An Example

- Finger @S
- `showmount -e`
- Send 20 SYN packets to S
- SYN flood T

- Attack when no one is around
- What other systems it trusts?
- Determine ISN behavior
- T won’t respond to packets
An Example

- Finger @S
- showmount –e
- Send 20 SYN packets to S
- SYN flood T
- Send SYN to S spoofing as T
- Send ACK to S with a guessed number

- Attack when no one is around
- What other systems it trusts?
- Determine ISN behavior
- T won’t respond to packets
- S assumes that it has a session with T
An Example

Shimomura (S) ->
++ > rhosts

Mitnick

Trusted (T)

• Finger @S
• showmount –e
• Send 20 SYN packets to S
• SYN flood T
• Send SYN to S spoofing as T
• Send ACK to S with a guessed number
• Send “echo + + > ~/.rhosts”

• Attack when no one is around
• What other systems it trusts?
• Determine ISN behavior
• T won’t respond to packets
• S assumes that it has a session with T
• Give permission to anyone from anywhere
Where do the problems come from?

- Protocol-level vulnerabilities
  - Implicit trust assumptions in design

- Implementation vulnerabilities
  - Both on routers and end-hosts

- Incomplete specifications
  - Often left to the imagination of programmers
Outline – Part II

• Security Vulnerabilities

• Denial of Service

• Worms

• Countermeasures: Firewalls/IDS
Denial of Service

• Make a service unusable/unavailable

• Disrupt service by taking down hosts
  ◦ E.g., ping-of-death

• Consume host-level resources
  ◦ E.g., SYN-floods

• Consume network resources
  ◦ E.g., UDP/ICMP floods
Reflector Attack

Unsolicited traffic at victim from legitimate hosts
Distributed DoS

Attacker

Handler

Handler

Agent

Agent

Agent

Agent

Agent

Victim

15-411: security
Distributed DoS

- Handlers are usually high volume servers
  - Easy to hide the attack packets

- Agents are usually home users with DSL/Cable
  - Already infected and the agent installed

- Very difficult to track down the attacker
  - Multiple levels of indirection!

- Aside: How to distinguish DDos from flash crowd?
Outline – Part II

• Security, Vulnerabilities

• Denial of Service

• Worms

• Countermeasures: Firewalls/IDS
Worm Overview

• Self-propagate through network

• Typical Steps in worm propagation
  ✦ Probe host for vulnerable software
  ✦ Exploit the vulnerability (e.g., buffer overflow)
    ▪ Attacker gains privileges of the vulnerable program
  ✦ Launch copy on compromised host

• Spread at exponential rate
  ✦ 10M hosts in < 5 minutes
  ✦ Hard to deal with manual intervention
Scanning Techniques

- Random
- Local subnet
- Routing Worm
- Hitlist
- Topological
Random Scanning

• 32-bit randomly generated IP address
  ◦ E.g., Slammer and Code Red I
  ◦ What about IPv6?

• Hits black-holed IP space occasionally
  ◦ Some percentage of IP space reserved
  ◦ Detect worms by monitoring unused addresses
    ▪ Honeypots/Honeynet
Subnet Scanning

- Generate last 1, 2, or 3 bytes of IP address randomly

- Code Red II and Blaster

- Some scans must be completely random to infect whole internet
Some proposals for countermeasures

- Better software safeguards
  - Static analysis and array bounds checking (lint/e-fence)
  - Safe versions of library calls
    - `gets(buf) → fgets(buf, size, ...)`
    - `sprintf(buf, ...) → snprintf(buf, size, ...)`
- Host-diversity
  - Avoid same exploit on multiple machines
- Network-level: IP address space randomization
- Host-level solutions
  - E.g., Memory randomization, Stack guard
- Rate-limiting: Contain the rate of spread
- Content-based filtering: signatures in packet payloads
Outline – Part II

• Security, Vulnerabilities
• Denial of Service
• Worms
• Countermeasures: Firewalls/IDS
Countermeasure Overview

• High level basic approaches
  ♦ Prevention
  ♦ Detection
  ♦ Resilience

• Requirements
  ♦ Security: soundness / completeness (false positive / negative
  ♦ Overhead
  ♦ Usability
Design questions ..

• Why is it so easy to send unwanted traffic?
  - Worm, DDoS, virus, spam, phishing etc

• Where to place functionality for stopping unwanted traffic?
  - Edge vs. Core
  - Routers vs. Middleboxes

• Redesign Internet architecture to detect and prevent unwanted traffic?
Firewalls

• Block/filter/modify traffic at network-level
  ▶ Limit access to the network
  ▶ Installed at perimeter of the network

• Why network-level?
  ▶ Vulnerabilities on many hosts in network
  ▶ Users don’t keep systems up to date
  ▶ Lots of patches to keep track of
  ▶ Zero-day exploits
Firewalls (contd...)

- Firewall inspects traffic through it
- Allows traffic specified in the policy
- Drops everything else
- Two Types
  - Packet Filters, Proxies
Packet Filters

- Selectively passes packets from one network interface to another

- Usually done within a router between external and internal network

- What/How to filter?
  - Packet Header Fields
    - IP source and destination addresses
    - Application port numbers
    - ICMP message types/ Protocol options etc.
  - Packet contents (payloads)
Packet Filters: Possible Actions

- Allow the packet to go through
- Drop the packet (Notify Sender/Drop Silently)
- Alter the packet (NAT?)
- Log information about the packet
Some examples

• Block all packets from outside except for SMTP servers

• Block all traffic to/from a list of domains

• Ingress filtering
  ▪ Drop pkt from outside with addresses inside the network

• Egress filtering
  ▪ Drop pkt from inside with addresses outside the network
Typical Firewall Configuration

- Internal hosts can access DMZ and Internet
- External hosts can access DMZ only, not Intranet
- DMZ hosts can access Internet only
- Advantages?
  - If a service gets compromised in DMZ it cannot affect internal hosts
Firewall implementation

- Stateless packet filtering firewall
- Rule $\rightarrow$ (Condition, Action)
- Rules are processed in top-down order
  - If a condition satisfied – action is taken
Allow SSH from external hosts to internal hosts

Two rules
   Inbound and outbound

How to know a packet is for SSH?
   Inbound: src-port>1023, dst-port=22
   Outbound: src-port=22, dst-port>1023
   Protocol=TCP

Problems?

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Src Addr</th>
<th>Src Port</th>
<th>Dst Addr</th>
<th>Dst Port</th>
<th>Proto</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSH-1</td>
<td>In</td>
<td>Ext</td>
<td>&gt; 1023</td>
<td>Int</td>
<td>22</td>
<td>TCP</td>
<td>Allow</td>
</tr>
<tr>
<td>SSH-2</td>
<td>Out</td>
<td>Int</td>
<td>22</td>
<td>Ext</td>
<td>&gt; 1023</td>
<td>TCP</td>
<td>Allow</td>
</tr>
</tbody>
</table>
Default Firewall Rules

- **Egress Filtering**
  - Outbound traffic from external address → Drop
  - Benefits?

- **Ingress Filtering**
  - Inbound Traffic from internal address → Drop
  - Benefits?

- **Default Deny**
  - Why?

<table>
<thead>
<tr>
<th>Rule</th>
<th>Dir</th>
<th>Src Addr</th>
<th>Src Port</th>
<th>Dst Addr</th>
<th>Dst Port</th>
<th>Proto</th>
<th>Ack Set?</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>Egress</td>
<td>Out</td>
<td>Ext</td>
<td>Any</td>
<td>Ext</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
<tr>
<td>Ingress</td>
<td>In</td>
<td>Int</td>
<td>Any</td>
<td>Int</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
<tr>
<td>Default</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Any</td>
<td>Deny</td>
</tr>
</tbody>
</table>
Packet Filters

• Advantages
  - Transparent to application/user
  - Simple packet filters can be efficient

• Disadvantages
  - Usually fail open
  - Very hard to configure the rules
  - May only have coarse-grained information?
    ▪ Does port 22 always mean SSH?
    ▪ Who is the user accessing the SSH?
Alternatives

• Stateful packet filters
  • Keep the connection states
  • Easier to specify rules
  • Problems?
    ▪ State explosion
    ▪ State for UDP/ICMP?

• Proxy Firewalls
  • Two connections instead of one
  • Either at transport level
    ▪ SOCKS proxy
  • Or at application level
    ▪ HTTP proxy
Intrusion Detection Systems

• Firewalls allow traffic only to legitimate hosts and services

• Traffic to the legitimate hosts/services can have attacks

• Solution?
  ◦ Intrusion Detection Systems
  ◦ Monitor data and behavior
  ◦ Report when identify attacks
Summary – Part II

- Security vulnerabilities are real!
  - Protocol or implementation or bad specs
  - Poor programming practices
  - At all layers in protocol stack

- DoS/DDoS
  - Resource utilization attacks

- Worm/Malware
  - Exploit vulnerable services
  - Exponential spread

- Countermeasures: Firewall/IDS

15-411: security
Resources

• Textbook: 8.1 – 8.3

• Wikipedia for overview of Symmetric/Asymmetric primitives and Hash functions.

• OpenSSL (www.openssl.org): top-rate open source code for SSL and primitive functions.

• “Handbook of Applied Cryptography” available free online: www.cacr.math.uwaterloo.ca/hac/