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”
Outline – Part II

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

Security Vulnerabilities

• Exist 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
  - Should be rare today

- Some “features” that have been exploited
  - Fragmentation: resources on routers
  - Broadcast for traffic amplification: bandwidth, endpoints

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
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 does not accept requests
- The solution is to use SYN cookies
  - The server keeps no state after the SYN
  - Instead, it embeds all the necessary state in the packet as carefully crafted initial sequence number

Where do the problems come from?

- Protocol-level vulnerabilities
  - Implicit trust assumptions in design
  - Many protocols were designed at a time that security was not a concern

- 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
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: generate random addresses
• Local subnet: generate last 1, 2, or 3 bytes of IP address randomly
• Routing Worm: uses information about allocated addresses from BGP
• Hitlist: provide list of vulnerable hosts
• Topological: exploit information on the infected hosts
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

Some proposals for countermeasures

- Better software safeguards
  - Static analysis and array bounds checking (lint/e-fence)
  - Safe versions of library calls
    - `gets(buf)` $\rightarrow$ `fgets(buf, size, ...)`
    - `sprintf(buf, ...)` $\rightarrow$ `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
    • Manage false positive / negative tradeoff
  • Overhead
  • Usability

• Where to place functionality: edge vs core
Firewall Motivation

- Block/filter/modify traffic at the perimeter of the network
  - Limit access to the network and all hosts/devices

- Why network-level?
  - Vulnerabilities on many hosts in network
  - Hosts/devices are very heterogeneous
  - Users do not keep systems up to date
  - Lots of patches to keep track of
  - Zero-day exploits

Firewalls Design

- Firewall inspects traffic that flows through it
- Allows traffic specified in the policy
- Drops everything else ("default off")
- Two Types
  - Packet Filters, Proxies
Packet Filters

- Selectively passes packets from one network interface to another
  - Options: forward, drop, or forward + log
- 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)

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 “Demilitarized Zone” (DMZ) and Internet
- External hosts can access DMZ only, not Intranet
- DMZ hosts can access Internet only
- Advantages?
  - A compromised service in DMZ it cannot affect internal hosts

Packet Filter Implementation

- Stateless packet filtering firewall
- Rule $\rightarrow$ (Condition, Action)
- Rules are processed in top-down order
  - If a condition satisfied – action is taken
Sample Firewall Rule

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

- Default rules are placed at end of the list – after “Allow” 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
  - Very hard to configure the rules – order matters, history
  - 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

• Traffic to the legitimate hosts/services can have attacks

• Intrusion detection system monitors traffic and traffic patterns
  • Looks for unusual behavior
  • Compares with known attacks, e.g., list of signatures based on attacks observed elsewhere
  • Block or report 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