Security Part One: Attacks and Countermeasures

15-441

With slides from: Debabrata Dash, Nick Feamster, Vyas Sekar

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

**Security Flaws in IP**

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

![Diagram of IP-level vulnerabilities](image)

**Smurf Attack**

- 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...

**ICMP Attacks**

- Can A claim it is B to the server S?
- ARP Spoofing
- Can C claim it is B to the server S?
- Source Routing

![Diagram of ICMP attacks](image)
Routing attacks

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

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

- BGP vulnerabilities

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

1. SYN (ISN_X)  
   SRC = X
2. SYN (9N, S1)  
   ACK (ISN_X)

Server

Trusted (T)

Malicious (M)
Session Hijack

TCP Layer Attacks

Malicious (M) Using ISN_S1 from earlier connection guess ISN_S2!

1. SYN (ISN_X)
   SRC = T

2. ACK (ISN_S2)
   SRC = T

3. ACK (ISN_X)

Server

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
An Example

- Shimomura (S)
  - Finger @S
  - showmount –e
  - Send 20 SYN packets to S
  - SYN flood T

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

Mitnick
- Send 20 SYN packets to S
- SYN flood T
- Determine ISN behavior
- T won’t respond to packets
- Send SYN to S spoofing as T
- Send ACK to S with a guessed number
- S assumes that it has a session with T

An Example

- Shimomura (S)
- Trusted (T)
- SYN
- SYN|ACK
- ATTACK when no one is around
- What other systems it trusts?

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

• 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

Simple DoS

• Attacker usually spoofs source address to hide origin

• Aside: Backscatter Analysis
  • When attack traffic results in replies from the victim
  • E.g. TCP SYN, ICMP ECHO

Backscatter Analysis

• Attacker sends spoofed TCP SYN packets to www.haplessvictim.com
  ♦ With spoofed addresses chosen at random

• My network sees TCP SYN-ACKs from www.haplessvictim.com at rate R

• What is the rate of the attack?
  ♦ Assuming addresses chosen are uniform
  ♦ \((2^{32}/ \text{Network Address space}) \times R\)
**Reflector Attack**

- **Attacker**
- **Agent**
- **Reflector**
- **Victim**

```
Src = Victim
Destination = Reflector
```

Unsolicited traffic at victim from legitimate hosts

---

**Distributed DoS**

- **Attacker**
- **Handler**
- **Agent**
- **Reflector**
- **Victim**

```
Src = Reflector
Destination = Victim
```

---

**Outline**

- **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

**Random Scanning**

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

- Hits black-holed IP space frequently
  - Only 28.6% of IP space is allocated
  - Detect worms by monitoring unused addresses
    - Honeypots/Honeynet

**Scanning Techniques**

- Random

- Local subnet

- Routing Worm

- Hitlist

- Topological

**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
Routing Worm

- BGP information can tell which IP address blocks are allocated
- This information is publicly available
  - http://www.routeviews.org/
  - http://www.ripe.net/ris/

Hit List

- List of vulnerable hosts sent with payload
  - Determined before worm launch by scanning
- Boosts worm growth in the slow start phase
- Can evade common detection techniques

Topological

- Uses info on the infected host to find the next target
  - Morris Worm used /etc/hosts, .rhosts
  - Email address books
  - P2P software usually store info about peers that each host connects to

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, ...)
    - snprintf(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

• 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 → (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

**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>Dest Addr</th>
<th>Dest Port</th>
<th>Proto</th>
<th>Ack Set?</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>Any</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>Yes</td>
<td>Allow</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

Proxy Firewall

- Data Available
  - Application level information
  - User information

- Advantages?
  - Better policy enforcement
  - Better logging
  - Fail closed

- Disadvantages?
  - Doesn't perform as well
  - One proxy for each application
  - Client modification

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
Classes of IDS

- What type of analysis?
  - Signature-based
  - Anomaly-based

- Where is it operating?
  - Network-based
  - Host-based

Signature-based IDS

- Characteristics
  - Uses known pattern matching to signify attack

- Advantages?
  - Widely available
  - Fairly fast
  - Easy to implement
  - Easy to update

- Disadvantages?
  - Cannot detect attacks for which it has no signature

Anomaly-based IDS

- Characteristics
  - Uses statistical model or machine learning engine to characterize normal usage behaviors
  - Recognizes departures from normal as potential intrusions

- Advantages?
  - Can detect attempts to exploit new and unforeseen vulnerabilities
  - Can recognize authorized usage that falls outside the normal pattern

- Disadvantages?
  - Generally slower, more resource intensive compared to signature-based IDS
  - Greater complexity, difficult to configure
  - Higher percentages of false alerts

Network-based IDS

- Characteristics
  - NIDS examine raw packets in the network passively and triggers alerts

- Advantages?
  - Easy deployment
  - Unobtrusive
  - Difficult to evade if done at low level of network operation

- Disadvantages?
  - Fail Open
  - Different hosts process packets differently
  - NIDS needs to create traffic seen at the end host
  - Need to have the complete network topology and complete host behavior
Host-based IDS

- Characteristics
  - Runs on single host
  - Can analyze audit-trails, logs, integrity of files and directories, etc.

- Advantages
  - More accurate than NIDS
  - Less volume of traffic so less overhead

- Disadvantages
  - Deployment is expensive
  - What happens when host get compromised?

Summary

- 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