Topic 7.4 Firewalls

Outline

• What are firewalls?
• Types
  – Filtering
    • Packet filtering
    • Session filtering
  – Proxy
    • Circuit Level
    • Application Level
• Brief introduction to Linux firewall
What is a firewall?

- Device that provides secure connectivity between networks (internal/external; varying levels of trust)
- Used to implement and enforce a security policy for communication between networks

Firewalls

- From Webster’s Dictionary: *a wall constructed to prevent the spread of fire*
- Internet firewalls are more the moat around a castle than a building firewall
- Controlled access point
Firewalls can:

- Restrict incoming and outgoing traffic by IP address, ports, or users
- Block invalid packets

Convenient

- Give insight into traffic mix via logging
- Network Address Translation
- Encryption
Firewalls Cannot Protect…

- traffic that does not cross it
  - routing around
  - Internal traffic
- when misconfigured

Access Control

Security Requirement
- Control access to network information and resources
- Protect the network from attacks
Filtering

- Typically route packets
- Packets checked then passed
- Inbound & outbound affect when policy is checked
- Client ↔ Server

Filtering

- Packet filtering
  - Access Control Lists
- Session filtering
  - Dynamic Packet Filtering
  - Stateful Inspection
  - Smart packet filtering
  - Context Based Access Control
Filtering

- Fragmentation/reassembly
- Sequence number checking
- ICMP

Packet Filtering

- Decisions made on a per-packet basis
- No state information saved
Typical Configuration

- Ports > 1024 left open
- If dynamic protocols are in use, *entire ranges of ports must be allowed* for the protocol to work.

Packet Filter
Session Filtering

- Packet decision made in the context of a connection
- If packet is a new connection, check against security policy
- If packet is part of an existing connection, match it up in the state table & update table

Typical Configuration

- All denied unless specifically allowed
- Dynamic protocols (FTP, H323, RealAudio, etc.) allowed only if supported
Session Filtering

- **Screens** ALL attempts, **Protects** All applications
- **Extracts & maintains** ‘state’ information
- **Makes** an intelligent **security / traffic** decision

Proxy Firewalls

- Relay for connections
- **Client ↔ Proxy ↔ Server**
- Two flavors
  - Application level
  - Circuit level
Application Gateways

- Understands specific applications
  - Limited proxies available
  - Proxy ‘impersonates’ both sides of connection
- Resource intensive
  - process per connection
- HTTP proxies may cache web pages

Application Gateways

- More appropriate to TCP
- ICMP difficult
- *Block all unless specifically allowed*
- Must write a new proxy application to support new protocols
  - Not trivial!
Application Gateways

- Clients configured for proxy communication
- Transparent Proxies

Application Layer GW/proxy
Circuit-Level Gateways

- Support more services than Application-level Gateway
  - less control over data
- Hard to handle protocols like FTP
- Clients must be aware they are using a circuit-level proxy
- Protect against fragmentation problem

SOCKS

- Circuit level Gateway
- Support TCP
- SOCKS v5 supports UDP, earlier versions did not
- See http://www.socks.nec.com
### Comparison

<table>
<thead>
<tr>
<th></th>
<th>Security</th>
<th>Performance</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Filter</td>
<td>3</td>
<td>1</td>
<td>No dynamic w/o holes</td>
</tr>
<tr>
<td>Session Filter</td>
<td>2</td>
<td>2</td>
<td>Dependent on vendor for dynamic support</td>
</tr>
<tr>
<td>Circuit GW</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>App. GW</td>
<td>1</td>
<td>4</td>
<td>Typically &lt; 20</td>
</tr>
</tbody>
</table>

*Lower is better for security & performance*

### Comparison (Cont’d)

<table>
<thead>
<tr>
<th></th>
<th>Modify Client Applications?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Filter</td>
<td>No</td>
</tr>
<tr>
<td>Session Filter</td>
<td>No</td>
</tr>
<tr>
<td>Circuit GW</td>
<td>Typical, SOCKS-ify client applications</td>
</tr>
<tr>
<td>App. GW</td>
<td>Unless transparent, client application must be proxy-aware &amp; configured</td>
</tr>
</tbody>
</table>
Comparison (Cont’d)

<table>
<thead>
<tr>
<th></th>
<th>ICMP</th>
<th>Fragmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet Filter</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Session Filter</td>
<td>Yes</td>
<td>Maybe</td>
</tr>
<tr>
<td>Circuit GW</td>
<td>(SOCKS v5)</td>
<td>Yes</td>
</tr>
<tr>
<td>App. GW</td>
<td>No</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Linux Firewall: iptables

- History
  - ipfw
  - ipfwadm
  - ipchains
  - iptables
    - Based on the netfilter framework
The Netfilter Framework

• A framework for packet mangling

Kernel
Protocol stack
hooks
netfilter
Kernel modules

User space

The Netfilter Framework (Cont’d)

• Current protocols
  – IPv4, IPv6, and DECnet.
• Five hooks for IPv4
  – [1]: Pre-routing hook; [2]: Local-in hook;
  – [3]: Forward hook; [4]: Local-out hook;
  – [5]: Post-routing hook

A packet traversing the netfilter system:
Packet Filtering

A packet traversing the netfilter system:

[1] → [ROUTE] → [3] → [4] → [ROUTE] → [5]

Packet filtering only uses these three hooks

IP Tables

- A packet selection system
  - Direct descendent of ipchains
- Used for
  - Packet filtering
  - Network Address Translation (NAT)
    - Masquerading, port forwarding, transparent proxying
  - Packet mangling
    - Actual changing of packet information
User Space Tool: iptables

- **iptables**
  - Command to configure and communicate with the kernel modules

- **iptables for packet filtering**
  - Three chains
    - INPUT
    - OUTPUT
    - FORWARD

Iptables for Packet Filtering

- You need three things to configure a firewall rule
  - Which chain?
  - What packet pattern?
  - What action to apply?

- Example
  - Drop all packets from 200.200.200.1
    - `iptables -A INPUT -s 200.200.200.1 -j DROP`
  - Use “man iptables” on Linux to get more information.