Network Intrusion

- **Network Intrusion Detection**
  - Detect when someone breaks into your network
  - Purposes
    - Forensics
    - Learns which end-systems are compromised
  - Examples
    - Snort (Passive)

- **Network Intrusion Prevention**
  - Stop someone from breaking into your network
  - Purpose
    - Security
    - Prevent end-systems from being compromised
  - Example
    - Firewall (Active), Hogwash
Configuration

- **Passive Protection**
  - Router
  - Internet

- **Active Protection**
  - Network Intrusion Detection
  - Network Intrusion Prevention

Network Security Terms

- **Threat**
  - Event or action that can harm a system

- **Attack**
  - Intentional attempt to harm a system

- **Vulnerability**
  - Weakness in a system that can be exploited
    - Software bug (threat caused by code developer)
    - Misconfiguration (threat caused by administrator)
    - Trust abuse (threat caused by system user)

- **Signature**
  - A pattern that identifies an activity
Challenges to Intrusion Detection

- **Alert handling**
  - Hard to deal with thousands of daily alarms

- **False alarms**
  - Most of the alarms can be ignored

- **Evasion**
  - Attackers will try to avoid detection

- **Resource Requirements**
  - Some attacks require a lot of work to detect
    - [Header processing mostly easy]
    - [Payload and Flow processing more difficult]

Header Processing

- **Media header**
  - Ethernet MAC
  - ATM Virtual Path & Circuit Identifiers (VPI/VCI)
    - 12 + 16 bits

- **Internet Header**
  - Source Address
  - Destination Address
  - Source Port
  - Destination Port
  - Protocol

- **Transport Header**
  - Sequence Number
  - Length
  - Checksum
Sample Packet

- **Layer 2 header**
  - ATM Header with VPI + VCI

- **Layer 3 header**
  - IP Header with source and destination addresses

- **Layer 4 header**
  - TCP/IP Header with sequence numbers, packet length, and checksum

- **Application**
  - Email message text

---

**Issues with Header-based Security (1/2)**

- **MAC addresses can be forged**
  - Most Ethernet & wireless adapters allow value of MAC address to modified from software
  - Packet sniffer can find valid MAC address, then allow a host to that address to access the network

- **Source Address can be forged**
  - End-system can can insert bogus address field to packet
  - User can transmit packets using a raw sockets
    - Supported on Windows for all users
    - Supported on UNIX systems for root user
Issues with Header-based Security (2/2)

- Ports do not accurately specify application
  - Many hosts use well-known ports to tunnel traffic for peer-to-peer applications
  - Examples of well-known ports
    - 22 ssh (Secure Shell)
    - 23 Telnet
    - 25 SMTP (send email)
    - 53 DNS (Domain Name Service)
    - 80 HTTP (Web Services)
    - 110 Pop3 (receive email)
    - 443 SSL (Secure Socket Layer)

Application-Level Headers

- Sample Web Request
  
  GET / HTTP/1.1

- Sample HTTP Reply Header
  
  HTTP/1.1 200 OK
  Server: Microsoft-IIS/4.0
  Date: Mon, 22 Sep 2003 03:24:01 GMT
  Content-Type: text/html
  Set-Cookie: USER=293911; path=/
Pattern Matching

- Simple pattern matching
  - Fixed string, regular expression, pattern

- Matching with protocol decode-based
  - Ethernet / ATM
  - Internet Protocol
  - http, mime

- Stateful pattern matching
  - Track more multiple packets in a flow
  - Example: TCP/IP

String Matching Techniques

- Brute force
  - Worst case running time: $O(nm)$
    - where $n$ is string length
    - where $m$ is pattern length

- The Knuth, Morris, and Pratt (KMP) algorithm (1977)
  - Improves running time by reducing character comparisons
  - Worst case running time: $O(n+m)$

- The Boyer and Moore algorithm (1977)
  - Similar to KMP, but searches right-to-left
  - Worst case running time: $O(n+m)$
Regular Expression Matching

- Thompson (1968)
  - Convert regular expression into Nondeterministic Finite Automata (NFA)
  - Worst case running time: $O(nm)$

- Use Deterministic Finite Automata (DFA) instead of NFAs
  - Worst case running time: $O(n)$
  - Theoretically $2^m$ states

Regular Expressions in Hardware

- Haskin and Hollar (1983)
  - Use finite state machines
  - Simulate an NFA by replicating DFAs
    - New DFA starts every time the possible beginning of a term is recognized

- Sidhu and Prasanna (2001)
  - Minimize time/space needed to construct NFAs

- Franklin, Carver, and Hutchings (2002)
  - Presented analysis of the Sidhu and Prasanna approach
FPgrep: Packet Payload Scanner

String Matching for Network Intrusion

- Need the ability to...
  - Scan every character of every packet’s payload
    - To find regular expressions
  - Actively drop packets
    - That match a given expression
  - Generate an alert message
    - To identify which expressions in a given set matched
  - Send an alert message to a log server
    - When a match is detected
  - Easily reconfigure the scanner
    - To search for a new set of expressions
Sample Keywords in packet payloads

- **General SPAM**
  - "(A|a)(M|m)(A|a)(Z|z)(I|i)(N|n)(G|g)"
  - "CALL NOW"
  - "(L|l)imited (T|t)ime (O|o)ffer"

- **Save Money SPAM**
  - "(C|c)onsolidate"

- **Fast Money SPAM**
  - "MAKE MONEY FAST"
  - "(W|w)ork from home"

- **Chains and Forwards**
  - "Read this"
  - "FWD"

- **Jokes**
  - "(J|j)oke"
  - "walks into bar"

- **Work List**
  - "{H|h}omework"
  - "{M|m}achine (P|p)roblem"
  - "{G|g}(S|s)536"
  - "Lockwood"
  - "Washington University"

- **Personal List**
  - "{M|m}om"
  - "{D|d}ad"
  - "{C|c}all (H|h)ome"

- **Urgent**
  - "{U|u}(R|r)(E|e)(N|n)(G|g)(E|e)(N|n)(T|t)"
  - "Emergency"

Regular Expression Matching

- Not necessary to know the beginning of a matching string
  - Prepend a ".*" to the beginning of each string that we are looking for

- Search for ".*ARL" instead of "ARL"
Regular Expression Searching w/Finite Automata

- Searching "A₁R₂A₃R₄L₅" for ".*ARL" succeeds

FPgrep Module
### Combined Header and Payload Processing

- Classify packet based on header –and-
- Classify packet based on content –and-
- Take action on packet

### Using the List Groups with CAMs

- **Drop SPAM, pass others**
  - Example: General SPAM or Chains and Forwards
    - Drop if Match = 1, CAM Value = 1, CAM Mask = 1

- **Keep useful data, drop others**
  - Example Filter: Work or Personal
    - Drop if Match = 0, CAM Value = 0, CAM Mask = 1

- **Ignore the results of the content filters**
  - Urgent or Jokes
    - CAM Mask = 0
### Content Match Vector

<table>
<thead>
<tr>
<th>Urgent</th>
<th>Personal</th>
<th>Work List</th>
<th>Jokes</th>
<th>Chains &amp; forwards</th>
<th>Fast Money SPAM</th>
<th>Save Money Spam</th>
<th>General Spam</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- Bit is set (1) if any phrase in a category is found anywhere in payload
- Bit is clear (0) if none of the phrases in the category appear in the payload

### Sample Content Match Vector

<table>
<thead>
<tr>
<th>Urgent</th>
<th>Personal</th>
<th>Work List</th>
<th>Jokes</th>
<th>Chains &amp; forwards</th>
<th>Fast Money SPAM</th>
<th>Save Money Spam</th>
<th>General Spam</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- “I can’t wait to work on my CS536 assignment!”
Header and Content Fields

- Content: “Consolidate your loans. CALL NOW”
  - Payload Lists = General SPAM (0) & Save Money SPAM (1)
  - Content Vector = “000000011” (binary) = x”03” (hex)

- Source IP Address: 128.252.5.5 (dotted.decimal)
- Destination IP Address: 141.142.2.2 (dotted.decimal)
- Source Port: 4096 (decimal) = 1000 (hex)
- Destination Port: 80 (decimal) = 50 (hex)
- Protocol: TCP (6)

<table>
<thead>
<tr>
<th>Content</th>
<th>Src IP (hex)</th>
<th>Dest IP (hex)</th>
<th>Src Port</th>
<th>Dest Port</th>
<th>Proto</th>
</tr>
</thead>
<tbody>
<tr>
<td>03</td>
<td>80FC0505</td>
<td>8D8E0202</td>
<td>1000</td>
<td>0050</td>
<td>06</td>
</tr>
</tbody>
</table>

All values shown in hex

Multi-field Packet Classifier

- Flow ID
- Resulting Flow Identifier
- 16 bits
- 112 bits
- Bits in IP Header
- CAM Table
- CAM MASK [1]
- CAM VALUE [1]
- CAM MASK [2]
- CAM VALUE [2]
- CAM MASK [3]
- CAM VALUE [3]
- CAM MASK [N]
- CAM VALUE [N]
- Priority Encoder
- Payload Match Bits
- Source Port
- Destination Address
- Source Address
- Protocol
- Mask Matchers
- Value Comparators
- Flow List