Announcements

• Project 2 is on the web.
  – Due: March 15th
  – Send groups to Jeff Vaughan (vaughan2@seas) by Thurs. Feb. 22nd.

• Plan for today:
  – Talk about the impact of firewalls and filters
  – Firewalls, NATs, etc.
Summary: Reactive Defense

- Reaction time:
  - required reaction times are a couple minutes or less
    (far less for bandwidth-limited scanners)

- Containment strategy:
  - content filtering is more effective than address blacklisting

- Deployment scenarios:
  - need nearly all customer networks to provide containment
  - need at least top 40 ISPs provide containment
Kinds of Firewalls

• Personal firewalls
  – Run at the end hosts
  – e.g. Norton, Windows, etc.
  – Benefit: has more application/user specific information

• Network Address Translators
  – Rewrites packet address information

• Filter Based
  – Operates by filtering based on packet headers

• Proxy based
  – Operates at the level of the application
  – e.g. HTTP web proxy
Network Address Translation

- Idea: Break the invariant that IP addresses are globally unique
NAT Behavior

• NAT maintains a table of the form:
  <client IP> <client port> <NAT ID>

• Outgoing packets (on non-NAT port):
  – Look for client IP address, client port in the mapping table
  – If found, replace client port with previously allocated NAT ID (same size as PORT #)
  – If not found, allocate a new unique NAT ID and replace source port with NAT ID
  – Replace source address with NAT address
NAT Behavior

• Incoming Packets (on NAT port)
  – Look up destination port number as NAT ID in port mapping table
  – If found, replace destination address and port with client entries from the mapping table
  – If not found, the packet is not for us and should be rejected

• Table entries expire after 2-3 minutes to allow them to be garbage collected
Benefits of NAT

• Only allows connections to the outside that are established from *inside*.
  – Hosts from outside can only contact internal hosts that appear in the mapping table, and they’re only added when they establish the connection
  – Some NATs support firewall-like configurability

• Can simplify network administration
  – Divide network into smaller chunks
  – Consolidate configuration data

• Traffic logging
Drawbacks of NAT

- Rewriting IP addresses isn’t so easy:
  - Must also look for IP addresses in other locations and rewrite them (may have to be protocol-aware)
  - Potentially changes sequence number information
  - Must validate/recalculate checksums

- Hinder throughput

- May not work with all protocols
  - Clients may have to be aware that NAT translation is going on

- Slow the adoption of IPv6?

- Limited filtering of packets / change packet semantics
  - For example, NATs may not work well with encryption schemes that include IP address information
Firewalls

• Filters protect against “bad” packets.
• Protect services offered internally from outside access.
• Provide outside services to hosts located inside.

Diagram:

Filter → Gateway → Filter

Inside — Gateway — Outside
Filtering Firewalls

• Filtering can take advantage of the following information from network and transport layer headers:
  – Source
  – Destination
  – Source Port
  – Destination Port
  – Flags (e.g. ACK)

• Some firewalls keep state about open TCP connections
  – Allows conditional filtering rules of the form “if internal machine has established the TCP connection, permit inbound reply packets”
Three-Way Handshake

Active participant (client)  Passive participant (server)

SYN, SequenceNum = x

SYN + ACK, SequenceNum = y, Acknowledgment = x + 1

ACK, Acknowledgment = y + 1
Ports

- Ports are used to distinguish applications and services on a machine.
- Low numbered ports are often reserved for server listening.
- High numbered ports are often assigned for client requests.

- Port 7 (UDP, TCP): echo server
- Port 13 (UDP, TCP): daytime
- Port 20 (TCP): FTP data
- Port 21 (TCP): FTP control
- Port 23 (TCP): telnet
- Port 25 (TCP): SMTP
- Port 79 (TCP): finger
- Port 80 (TCP): HTTP
- Port 123 (UDP): NTP
- Port 2049 (UDP): NFS
- Ports 6000 to 6xxx (TCP): X11
Filter Example

<table>
<thead>
<tr>
<th>Action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td>*</td>
<td>*</td>
<td>BAD</td>
<td>*</td>
<td>untrusted host</td>
</tr>
<tr>
<td>allow</td>
<td>GW</td>
<td>25</td>
<td>*</td>
<td>*</td>
<td>allow our SMTP port</td>
</tr>
</tbody>
</table>

Apply rules from top to bottom with assumed *default* entry:

<table>
<thead>
<tr>
<th>Action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>block</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>default</td>
</tr>
</tbody>
</table>

Bad entry intended to allow connections to SMTP from inside:

<table>
<thead>
<tr>
<th>Action</th>
<th>ourhost</th>
<th>port</th>
<th>theirhost</th>
<th>port</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>25</td>
<td>connect to their SMTP</td>
</tr>
</tbody>
</table>

This allows all connections from port 25, but an outside machine can run *anything* on its port 25!
Filter Example Continued

Permit *outgoing* calls to port 25.

<table>
<thead>
<tr>
<th>Action</th>
<th>src</th>
<th>port</th>
<th>dest</th>
<th>port</th>
<th>flags</th>
<th>comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>allow</td>
<td>123.45.6.*</td>
<td>*</td>
<td>*</td>
<td>25</td>
<td>*</td>
<td>their SMTP</td>
</tr>
<tr>
<td>allow</td>
<td>*</td>
<td>25</td>
<td>*</td>
<td>*</td>
<td>ACK</td>
<td>their replies</td>
</tr>
</tbody>
</table>

This filter doesn’t protect against IP address spoofing. The bad hosts can “pretend” to be one of the hosts with addresses 123.45.6.*.
Snort

- Snort is a lightweight intrusion detection system:
  - Real-time traffic analysis
  - Packet logging (of IP networks)
- Rules based logging to perform content pattern matching to detect a variety of attacks and probes:
  - such as buffer overflows, stealth port scans, CGI attacks, SMB probes, etc.
- Example Rule:
  ```
  alert tcp any any -> 192.168.1.0/24 143 (content:"|E8C0 FFFF FF|/bin/sh"; msg:"New IMAP Buffer Overflow detected!");
  ```
  - Generates an alert on all inbound traffic for port 143 with contents containing the specified attack signature.
- The Snort web site:
  - http://www.snort.org/docs/
- Question: How do you come up with the filter rules?
Internet Telescopes

• Can be used to detect large-scale, wide-spread attacks on the internet.
Internet Telescopes

- Can be used to detect large-scale, wide-spread attacks on the internet.

UCSD monitors 17M+ addresses

Worm sends randomly

Telescope monitors packets for large range of unused addresses
Internet Telescopes

- Can be used to detect large-scale, wide-spread attacks on the internet.

Worm sends randomly

UCSD monitors 17M+ addresses

Telescope monitors packets for large range of unused addresses
Automated Worm Fingerprinting

• Paper by Singh, Estan, Varghese, and Savage

• Assumptions:
  – All worms have invariant content
  – Invariant packets will appear frequently on the network
    • Worms are trying to propagate, after all
  – Packet sources and destinations will show high variability
    • Sources: over time number of distinct infected hosts will grow
    • Destinations: worms scan randomly
    • Distribution will be roughly uniform (unlike regular traffic that tends to be clustered)
High-prevalence strings are rare
Naïve Content Sifting

- ProcessTraffic(packet, srcIP, dstIP) {
  count[packet]++;
  Insert(srcIP, dispersion[packet].sources);
  Insert(dstIP, dispersion[packet].dests);
  if (count[packet] > countThresh
      && size(dispersion[packet].sources) > srcThresh
      && size(dispersion[packet].dests) > dstThresh) {
    Alarm(packet)
  }
}

- Tables count and dispersion are indexed by entire packet content.
Problems with Naïve approach

- Frequency count is inaccurate:
  - Misses common substrings
  - Misses shifted content
  - Ideally, would index count and dispersion by all substrings of packet content (of some length)

- Counting every source and destination is expensive.

- Too much data to process every packet.
  - Most packets are going to be uninteresting.
  - Tables count and dispersion will be huge!
Engineering Challenges

• To support 1Gbps line rate have 12us to process each packet.

• Naïve implementation can easily use 100MB/sec for tables.

• Don't want to just do naïve sampling
  – E.g. don't want to just look at 1/N of the packets because detecting the worm will take N times as long
Practical Content Sifting

• Reduce size of count table by:
  – Hashing the packet content to a fixed size (*not* cryptographic hashes)
  – Hash collisions may lead to false positives
  – So, do multiple different hashes (say 3) -- worm content is flagged only if counts along all hashes exceed a threshold

• Include the destination port in the hash of the packet content
  – Current worms target specific vulnerabilities, so they usually aim for a particular port.

• To check for substring matches they propose to use a Rabin fingerprint
  – Probabilistic, incrementally computable hash of substrings of a fixed length.
Multistage Filters, Pictorially
Tracking Address Dispersion

• In this case, we care about the number of distinct source (or destination) addresses in packets that contain suspected worm data.

• Could easily keep an exact count by using a hash table, but that becomes too time and memory intensive.
  – In the limit, need one bit per address to mark whether it has been seen or not.

• Instead: Keep an *approximate* count
• Scalable bitmap counters
  – Reduce memory requirements by 5x
Scalable Bitmap Counters

• Suppose there are 64 possible addresses and you want to use only 32 bits to keep track of them.

• High-level idea:
  – Hash the address into a value between 0 and 63
  – Use only the lower 5 bits (yielding 32)
  – To estimate actual number of addresses, multiply the number of bits set in the bitmap by 2.
Multiple Bitmaps, Pictorially

- Recycle bitmaps after they fill up
- Adjust the scale factors on the counts accordingly
Results

- Earlybird successfully detects and extracts virus signatures from every known recent worm (CodeRed, MyDoom, Sasser, Kibvu.B, ...)
- Tool generates content filter rules suitable for use with Snort

![Packet Header and Payload Example](image-url)
Analysis

• False Positives:
  – SPAM
  – BitTorrent
  – Common protocol headers
    • HTTP and SMTP
    • Some P2P system headers
  – Solution: whitelist by hand

• False Negatives:
  – Hard (impossible?) to prove absence of worms
  – Over 8 months Earlybird detected all worm outbreaks reported on security mailing lists