CIS 551 / TCOM 401
Computer and Network Security

Spring 2007
Lecture 11
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 worm and virus propagation & modeling
  – Talk about the impact of firewalls and filters
  – Firewalls, NATs, etc.
Analysis: Random Constant Spread Model

- IP address space = \(2^{32}\)
- \(N\) = size of the total vulnerable population
- \(S(t)\) = susceptible/non-infected hosts at time \(t\)
- \(I(t)\) = infective/infected hosts at time \(t\)
- \(\beta\) = Contact likelihood
- \(s(t) = S(t)/N\) proportion of susceptible population
- \(i(t) = I(t)/N\) proportion of infected population

- Note: \(S(t) + I(t) = N\)
Infection rate over time

- Change in infection rate is expressed as:

\[
\frac{dl}{dt} = l(t) \cdot \beta \cdot s(t)
\]

Rewrite to obtain:

\[
\frac{di}{dt} = \beta \cdot i(t) \cdot (1-i(t))
\]

Integrate to get this closed form:

\[
i(t) = \frac{e^{\beta(t-T)}}{1 + e^{\beta(t-T)}}
\]

\[T = \text{integration constant}\]
Exponential growth, tapers off

- Example curve of $I(t)$ (which is $i(t) \times N$)
- Here, $N = 3.5 \times 10^5$ ($\beta$ affects steepness of slope)
What about the constants?

• N = estimated # of hosts running vulnerable software
  – e.g. Apache or mail servers
  – In 2002 there were roughly 12.6M web servers on the internet
• Reasonable choice for $\beta$ is $r \times N / 2^{32}$
  – Where $r$ = probing rate (per time unit)

• For Code Red I:
  – $\beta$ was empirically measured at about 1.8 hosts/hour.
  – $T$ was empirically measured at about 11.9 (= time at which half the vulnerable hosts were infected)
• Code Red I was programmed to shut itself off at midnight UTC on July 19th
  – But incorrectly set clocks allowed it to live until August
  – Second outbreak had $\beta$ of approximately 0.7 hosts/hour
  – Implies that about 1/2 of the vulnerable hosts had been patched
Predictions vs. Reality

- Port 80 scans due to Code Red I
courtesy Paxson, Staniford, Weaver
What can be done?

- **Reduce the number of infected hosts**
  - **Treatment**, reduce $I(t)$ while $I(t)$ is still small
  - e.g. shut down/repair infected hosts

- **Reduce the contact rate**
  - **Containment**, reduce $\beta$ while $I(t)$ is still small
  - e.g. filter traffic

- **Reduce the number of susceptible hosts**
  - **Prevention**, reduce $S(0)$
  - e.g. use type-safe languages

Reactive

Proactive
Treatment

• Reduce # of infected hosts

• Disinfect infected hosts
  – Detect infection in real-time
  – Develop specialized “vaccine” in real-time
  – Distribute “patch” more quickly than worm can spread
    • Anti-worm? (CRClean written)
    • Bandwidth interference…
Effects of "patching" infected hosts

- Kermack-McKendrick Model
- State transition:
  \[ U(t) = \# \text{ of removed from infectious population} \]
  \[ \gamma = \text{removal rate} \]

\[
\frac{du}{dt} = \gamma \cdot i(t)
\]

\[
\frac{di}{dt} = \beta \cdot i(t) \cdot (1-i(t)) - \frac{du}{dt}
\]
Containment

• Reduce contact rate $\beta$

• **Oblivious defense**
  – Consume limited worm resources
  – Throttle traffic to slow spread
  – Possibly important capability, but worm still spreads…

• **Targeted defense**
  – Detect and block worm
Design Space

• Design Issues for Reactive Defense
  [Moore et al 03]

• Any reactive defense is defined by:
  – *Reaction time* – *how long* to detect, propagate information, and activate response
  – *Containment strategy* – *how* malicious behavior is identified and stopped
  – *Deployment scenario* - *who* participates in the system

• Savage et al. evaluate the requirements for these parameters to build *any* effective system for worm propagation.
Methodology

- Moore et al., "Internet Quarantine:..." paper

- **Simulate spread of worm across Internet topology:**
  - infected hosts *attempt* to spread at a fixed rate (probes/sec)
  - target selection is uniformly random over IPv4 space

- **Simulation of defense:**
  - system detects infection within reaction time
  - subset of network nodes employ a containment strategy

- **Evaluation metric:**
  - % of vulnerable hosts infected in 24 hours
  - 100 runs of each set of parameters (95th percentile taken)
    - Systems must plan for reasonable situations, *not* the average case

- **Source data:**
  - vulnerable hosts: 359,000 IP addresses of CodeRed v2 *victims*
  - Internet topology: AS routing topology derived from RouteViews
Initial Approach: Universal Deployment

• Assume **every host** employs the containment strategy

• Two containment strategies they tested:
  – **Address blacklisting:**
    • block traffic from malicious source IP addresses
    • reaction time is relative to each infected host
  – **Content filtering:**
    • block traffic based on signature of content
    • reaction time is from first infection

• How quickly does each strategy need to react?
• How sensitive is reaction time to worm probe rate?
To contain worms to 10% of vulnerable hosts after 24 hours of spreading at 10 probes/sec (CodeRed):

- Address blacklisting: reaction time must be < 25 minutes.
- Content filtering: reaction time must be < 3 hours
Probe rate vs. Reaction Time

• Reaction times must be fast when probe rates get high:
  – 10 probes/sec: reaction time must be < 3 hours
  – 1000 probes/sec: reaction time must be < 2 minutes
Limited Network Deployment

• Depending on every host to implement containment is not feasible:
  – installation and administration costs
  – system communication overhead

• A more realistic scenario is limited deployment in the network:
  – Customer Network: firewall-like inbound filtering of traffic
  – ISP Network: traffic through border routers of large transit ISPs

• How effective are the deployment scenarios?
• How sensitive is reaction time to worm probe rate under limited network deployment?
Deployment Scenario Effectiveness?

Reaction time = 2 hours

CodeRed-like Worm:

Content filtering firewalls at edge of customer nets.

Content filtering at exchange points in major ISPs.
Above 60 probes/sec, containment to 10% hosts within 24 hours is impossible even with *instantaneous* reaction.
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.
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)  

SYN, SequenceNum = $x$

SYN + ACK, SequenceNum = $y$

ACK, Acknowledgment = $x + 1$

Passive participant (server)
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 * \textit{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.* .