Emulating sequential scanning worms on the DETER testbed

George Kesidis, Lunquan Li, Ihab Hamadeh, Soranun Jiwasurat, Peng Liu
Penn State University

Open-source code downloadable at Http://emist.ist.psu.edu

DETER Community Workshop
Arlington, VA, June 15,16, 2006
1. Experimental methodologies for Worm research

- Mathematical analysis and modeling
- Simulation: By a single CPU or a distributed simulator
- Emulation: 1:1 on a test-bed
- Hybrid: Combination of simulation and emulation
Experiment Overview

- Replay the worm propagation in high fidelity using test bed (DETER)
- Understand the impact of worms on enterprise networks
- Build foundations for worm defense experiments
2. Worm attack traffic modeling: KMSim-II

- KMSim model: a variation of kermack-Mckendrick model

- \( J \) be the number of different groups of peripheral enterprise networks in the Internet,

- \( \sigma_{ji} \) represent the total scanning rate out of the enterprise to the rest of the Internet of a group-\( j \) enterprise network with \( i \) infectives (i.e., infection-level \( i \))

- \( y_{j,i}(t) \) be the number of group-\( j \) enterprises with infection-level \( i \),

- \( C(j) \) be the maximum infection level of group-\( j \) enterprises, and

- the total scan rate to the Internet of the worm be

\[
S(t) \equiv \sum_{j=1}^{J} \sum_{i=1}^{C(j)} \sigma_{ji}y_{j,i}(t).
\]

\[
\begin{align*}
\dot{y}_{j,C(j)}(t) &= \beta_{j,C(j)-1}y_{j,C(j)-1}(t), \quad (1) \\
\dot{y}_{j,i}(t) &= (\beta_{j,i-1}y_{j,i-1}(t) - \beta_{j,i}y_{j,i}(t)) \quad i \in [1,C_j) \quad (2) \\
\dot{y}_{j,0}(t) &= -\beta_{j,0}y_{j,0}(t). \quad (3)
\end{align*}
\]

KMSim-II adds worm death/removal:

\[
\begin{align*}
\frac{dy_{j,i}}{dt} &= -\delta_i y_{j,i} \\
\frac{dy_{j,i-1}}{dt} &= \delta_i y_{j,i}
\end{align*}
\]

where the removal/death rate is \( \delta > 0 \) and \( \delta_i \equiv i\delta \).
2. 1 KMSim-II: Witty and Blaster simulation

Algorithm 1 Removal/Death of the Infected and Susceptible

1: for $j$ do
2:    $newdeath = \delta * ce_j * y_{j,e} * dt$
3:    $death_j + = newdeath$
4:    $y_{j,e} - = newdeath$
5:    for $i = ce_j - 1$ downto 1 do
6:        $y_{j,i} + = newdeath$
7:        $newdeath = \delta * i * (y_{j,i} - newdeath) * dt$
8:    $y_{j,i} - = newdeath$
9:    end for
10: end for

Blaster similarly simulated and used in the following emulation

Parameters: vulnerable population 12,000
j=1, c=4, NE=3,000
$\sigma = [1800, 2400]$
3. Emulation challenge: scaling down

1002 nodes

DETER: 2xx nodes

So virtualization-based abstraction is necessary!

Requirements: (1) scalability
(2) graceful degradation of fidelity
3. 1 Solution: Virtual node approach
## 3.2 Virtualization Comparison

<table>
<thead>
<tr>
<th></th>
<th>VMware</th>
<th>Emulab VM</th>
<th>NS2</th>
<th>Virtual Node</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS Platform</strong></td>
<td>Windows &amp; Linux</td>
<td>FreeBSD</td>
<td>Linux/FreeBSD</td>
<td>Linux/FreeBSD</td>
</tr>
<tr>
<td><strong>Test-bed Support</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes (NSE)</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Scale of Virtualization</strong></td>
<td>4-8</td>
<td>&lt;20</td>
<td>100-1000</td>
<td>20-200</td>
</tr>
<tr>
<td><strong>Supporting unmodified Apps</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Fidelity</strong></td>
<td>High</td>
<td>?</td>
<td>?</td>
<td>?</td>
</tr>
</tbody>
</table>
Step 1. Setup the experiment using the EMIST GUI

Step 2. Setup the DETER environment

Step 3. Run the experiment on DETER

Step 4. Visualize the results using the EMIST GUI

- EMIST topology specification in TCL
  - Virtual sub-network nodes
  - Internet interface
  - Normal & vulnerable nodes
  - Bandwidth, latency, addresses, OS
  - Other auxiliary TCL scripts

- Worm program
- Traffic generator program
- Internet interface program
- Virtual node program
- Normal node program
- Vulnerable node program
- TCPDUMP setup
- EMULAB GUI can be used here

- Worm propagation snapshots
- Worm propagation animation
- Link traffic bar chart (dynamic)
- Worm replay
4. ESVT (Experiment Specification and Visualization Toolkit)

Topology editing

TCL Script Generation

Components Property
- Make changes on: Selected
- Use FreeBSD by default
- Computers/Hosts: 10
- Link Bandwidth: 100000 kbps
- Vulnerability: 50%

Experiment Duration: 60 Seconds
Node Startup Command Directory:

Switch Property
- Index: 15
- Switch Links
- Simulated Link

Example TCL script:
```
# Total Switches: 1, Computer: 50, Susceptible under 50.
eth 0/0/0 [if net=eth0 "filter: ip netmask 0.0.0.0" - 1000 ms]  
# Example network setup script
```
5. Virtual node Design

Physical topology

Programming Implementation
5.1 Address mapping and packet structure

The virtual header is located inside the payload of IP packets

<table>
<thead>
<tr>
<th>IP Header</th>
<th>SourceAddr</th>
<th>SourcePort</th>
<th>DestinationAddr</th>
<th>DestinationPort</th>
<th>Length</th>
<th>Type</th>
<th>Pad</th>
</tr>
</thead>
</table>

Virtual dest addresses are looked up in the mapping table to locate the address and port number of each virtual host, E.G.,

<table>
<thead>
<tr>
<th>Network Address</th>
<th>DETER Node Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.1.3.0/24</td>
<td>10.1.3.2</td>
</tr>
<tr>
<td>10.1.4.0/24</td>
<td>10.1.4.2</td>
</tr>
<tr>
<td>10.1.5.0/24</td>
<td>10.1.5.2</td>
</tr>
<tr>
<td>10.1.6.0/24</td>
<td>10.1.6.2</td>
</tr>
<tr>
<td>default</td>
<td>10.1.8.2</td>
</tr>
</tbody>
</table>
5.2 LAN traffic shaping

- Switched LAN implemented, CSMA/CD LAN and WLAN in the future
- Token-bucket traffic regulator
- Let C be the uplink bandwidth, n be the number of hosts, R be the full transmission speed of any host, the LAN throughput is
  \[ S(n) = \min \{ R \cdot n, C \}, \]
  The maximum sending speed per host is \( s(n)/n \).
5. 3 Blaster Emulation: packet crafting

- Using UDP to simulate TCP worm
- Internet injection scan rate is based on KMSim simulation results
- “Sequential” scanning behavior is reserved in Internet scan injection

```
Algorithm 2 Sequential Scanning Worm Body
repeat
  2: for j = 1 to 20 do
      Calculate Next IP tip_j
      pkt.destinationIP = tip_j
      Send UDP packet
  6: end for
      Sleep 1.8 seconds
  8: for j = 1 to 20 do
      if tip_j is infected then
          pkt.payloadlength = BLASTERSIZE
          pkt.destinationIP = tip_j
          for l = 0 to 8 do
              Send UDP packet {Simulate the actual worm infection}
          end for
          SendingDelay = (wormpacketsize + 28) * 80;
          Delay SendingDelay
      end if
  18: end if
end for
until Program Is Active
```
6. Experiment on DETER

- 1000-plus hosts experiment takes 49 test-bed nodes
- Emulation lasts for 600 seconds
6.1 Blaster Emulation Results

- Emulation: 600 Seconds; 1,400 files and 1GB trace logs
- Infection: 100-400 infected out of 400 plus susceptible nodes
6.1 Blaster Emulation Results

• Characteristics of Blaster: fast local propagation (83% infected between 120sec and 180sec)
• Egress traffic roughly proportional to the number of infected hosts (before saturation)
6.2. Placement of Honeypots

- Node 942 and 955 are two /24 honeypots
- Background traffic generators will skip these addresses
- Not effective for local-scanning worm if no traffic redirection
ESVT Visualization

• Host state change
• Link color change (traffic)
• Link traffic bar chart
• Pause; Rewind; Drag
• Other advanced traffic chart
Future Study

• Topological worms
• Worm’s impact on various services in the enterprise network
• Worm detection, containment, etc.
Questions?

Visit: http://emist.ist.psu.edu