Network Forensics and Next Generation Internet Attacks

Moderated by: Moheeb Rajab

Background singers: Jay and Fabian
Agenda

- Questions and Critique of Timezones paper
  - Extensions

- Network Monitoring (recap)

- Post-Mortem Analysis
  - Background and Realms
  - Problem of Identifying Patient zero
  - Detecting Initial hit-list

- Next Generation attacks (Omitted from slides)
  - Implications and Challenges?
Botnets or Worms？!

“The authors don’t provide evidence that botnets propagate in the same way like regular worms”

Opening Sentence:
Student questions
Data Collection

- “The original data collection method itself is worth mentioning as a strength of this paper”

- “Can’t someone who sees all the traffic intended for a C&C server do more than simply gather SYN statistics”

- “It is not clear to me how do they know that they captured the propagation phase in their tests”
Measuring Botnet Size
SYN Counting

- Only looking at the Transport Layer
  - Do we even know what this traffic is?

- DHCP’d hosts
  - DHCP will cause SYN(s) coming from different addresses.

- How does the Tarpit help?

- Totally unrelated traffic
  - Scans, exploit attempts, etc.
Estimating botnet size

- How do we quantify these effects and relate them back to the claimed 350 K size?
  - Are we counting wrong? If we assume DHCP lease of Δ hours, how do these projections change?

- Studied 50 botnets but we have 3 data points.

- Fitting the model to the collected data
  - What parameters did they use?
Evidence from “Da-list”

<table>
<thead>
<tr>
<th>Date and Time</th>
<th>DNS</th>
<th>Non-DNS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feb, 1st 4:00 AM EST</td>
<td>49</td>
<td>4</td>
</tr>
<tr>
<td>Feb 1st 11:00 AM EST</td>
<td>23 (&gt; 4 public IRCds)</td>
<td>4</td>
</tr>
</tbody>
</table>
General consensus

- Contrary to authors the attackers could use the timezones effect to their benefit
  - How?

- This is old-school, right?:
  - Botnet Herders can hide behind VoIP. InfoWeek, 2/27/06
    - Okay, this is getting ridiculous

- Cherry-picking: some weird indications ...
Extensions

- Can we use this idea for containment?
  - Query to know if someone is infected
  - How to preserve privacy and anonymity?
    - See *Privacy-Preserving Data Mining*. R. Agrawal and R. Srikant. *Proceedings of SIGMOD, 2000*

- Patching rates?
  - More grounded parameters might really affect model
  - How might we get this?

- Lifetime?
Student Extensions

- Is there better ways to track botnets other than poisoning DNS?
  - Crazy idea #1: Anti-worm

- Crazy idea #2: Statistical responders

- What would you have liked to see with this data?
Using telescopes for network forensics
Forensic (Post-mortem) analysis

- Infer characteristics of the attack
  - Population size, demographics, distribution
  - Infection rate, scanning behavior .. etc

- Trace the attack back to its origin(s)
  - Identifying patient zero
  - Identifying the hit-list (if any)
  - Reconstructing the infection tree
Worm Evolution Tracking Realms

- Graph Reconstruction
- Reverse Engineering
- Timing Analysis
Infection Graph Reconstruction


- Proposed a random walk algorithm on the hosts contact graph
  - Provides who infected whom tree
  - Identifies the worm entry point(s) to a local network or administrative domain.
Random Moonwalks

- A random moonwalk on the host contact graph:
  - Start with an arbitrarily chosen flow
  - Pick a next step flow randomly to walk **backward in time**
- Observation: epidemic attacks have a **tree** structure
  - Initial causal flows emerge as high frequency flows

Slide by: Ed Knightly
Random Moonwalk (Limitations)

- Host Contact graph is known.
  - requires extensive logging of host contacts throughout the network

- Only able to reconstruct infection history on a local scale

- *Careful* selection of parameters to guarantee the convergence of the algorithms
  - How to address this is left as open problem
Outwitting the Witty

*Kumar et al.*, “Exploiting Underlying Structure for Detailed Reconstruction of an Internet-scale Event”, IMC 2005

- Exploits the structure of the random number generator used by the worm
  - Careful analysis of the worm payload allows us to reconstruct the infection series
Witty Code!

\[
\text{\texttt{\textit{srand}}(seed) \ { \ X \leftarrow seed \ }} \ \\
\text{\texttt{\textit{rand}}()} \ { \ X \leftarrow X*214013 + 2531011; \ \text{return} \ X } \ \\
\]

\textbf{main()}
1. \texttt{\textit{srand}}(get\_tick\_count());
2. \texttt{for(i=0; i<20,000; i++)}
3. \hfill \text{\texttt{dest\_ip} \leftarrow \texttt{\textit{rand}()}[0..15] || \texttt{\textit{rand}()}[0..15]}
4. \hfill \text{\texttt{dest\_port} \leftarrow \texttt{\textit{rand}()}[0..15]}
5. \hfill \text{\texttt{packetsize} \leftarrow 768 + \texttt{\textit{rand}()}[0..8]}
6. \hfill \text{\texttt{packetcontents} \leftarrow \textit{top-of-stack}}
7. \hfill \texttt{sendto()}
8. \hfill \text{\texttt{if(open\_physical\_disk(\texttt{\textit{rand}()}[13..15])})}
9. \hfill \text{\texttt{write(\texttt{\textit{rand}()}[0..14] || 0x4e20)}}
10. \hfill \texttt{goto 1}
11. \texttt{else goto 2}
Witty Code!

- Each Witty packet makes 4 calls to `rand()`

- If first call to `rand()` returns $X_i$:

  3. $\text{dest\_ip} \leftarrow (X_i)[0..15] \| (X_{i+1})[0..15]$
  4. $\text{dest\_port} \leftarrow (X_{i+2})[0..15]$

Given top 16 bits of $X_i$, now brute force all possible lower 16 bits to find which yield consistent top 16 bits for $X_{i+1} \& X_{i+2}$

⇒ **Single** Witty packet suffices to extract infectee’s *complete* PRNG state!
Interesting Observations

- Reveals interesting facts about 700 infected hosts:
  - Uptime of infected machines
  - Number of available disks
  - Bandwidth Connectivity
  - Who-infected whom
  - Existence of hit-list
  - Patient zero (?)
Reverse Engineering (Limitations)

- Not easily generalizable
  - Needs to be done on a case by case basis

- Can be tedious (go back to the paper to see).

- There must be an easier way, right?
Timing Analysis


- Uses blind analysis of inter-arrival times at a network telescope to infer the worm evolution.
Problem Statement and Goals

Consider a uniform scanning worm with scanning rate $s$ and vulnerable population size $V$ and a monitor with effective size $M$.

- To what extent can a network monitor trace the infection sequence back to patient zero by observing the order of unique source contacts?

- For worms that start with a hitlist, can we use network monitors to detect the existence of the hitlist and determine its size?
Evolution Sequence and “Patient Zero”

- We distinguish between two processes:
  
  - **Time to Infect** $T_{in}$
    - Time elapsed before the worm infects an additional host

  - **Time to Detect** $T_{d}$
    - The time interval within which a monitor can reliably detect at least one scan from a *single* newly infected host
Time to Infect \textit{and} Time to Detect
Time to Infect and Time to Detect

- Time to infect a new host \( T_{in} \)

\[
T_{in} = \frac{\log\left(1 - \frac{1}{V - n_i}\right)}{\log\left(1 - \frac{1}{2^{32}}\right)}
\]
Monitor Accuracy

- Monitor Detection time, $T_d$

- Probability of error

$$P_e = 1 - \prod_{i=1}^{n} \left( 1 - \frac{M}{2^{32}} \right)$$
$T_{in}$ and $T_d$

Uniform scanning worm:
$s = 350$ scans/sec,
$V = 12,000$
Monitor size = /8

Probability of Error
Infection Sequence Similarity

Sequence Similarity

Actual (A)  

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |   | m-1 | m |

Monitor (B)  

| 1 | 2 | 3 | 4 | 9 | 6 | 7 | 8 | 5 |   | m-1 | m |

\[
Y_{B \rightarrow A} = \sum_{i=0}^{m} \frac{(m - r_{(e_i,A)})}{1 + \left| r_{(e_i,B)} - r_{(e_i,A)} \right|}
\]
Is this any good?

- Two (interesting) cases:
  - Varying monitor sizes
  - Non-homogeneous scanning rates
Bigger is Better

Larger telescopes provide a highly similar view to the actual worm evolution

/16 view is completely useless!
Effect of non-homogeneous scanning

Scanning rate distribution derived from CAIDA’s dataset
So, of what good is this?

Who cares what happens after the first 200 infections :-(
Problem Statement and Goals

Consider a uniform scanning worm with scanning rate $s$ and vulnerable population size $V$ and a monitor with effective size $M$.

- To what extent can a network monitor trace the infection sequence back to patient zero by observing the order of unique source contacts?

- For worms that start with a hitlist, can we use network monitors to detect the existence of the hitlist and determine its size?
What if the worm starts with a hit-list?

- Hit-lists are used to
  - Boost initial momentum of the worm
  - (Possibly) hide the identity of patient zero

**Trick:** Exploit the pattern of inter-arrival times of unique sources contacts at the monitor to infer the existence and the size of the hitlist
Hit-list detection and size estimation

Simulation ( $H = 100$ )

Estimated hit-list
$H$ approx. 80
80% in the same /16
88% belong to the same institution

Witty Worm (CAIDA)
Will we always see this pattern?

- Same pattern was noticed also when varying population size and with non-homogeneous scanning rates.
Why is that?

- With a hit-list of size $h_0$ the average worm infection time $T_{in}$ should be less than $T_d / h_0$

$$\log \left(1 - \frac{1}{(V - h_0)}\right) \leq \frac{\log(1 - \alpha) \log \left(1 - \frac{1}{2^{32}}\right)}{\log \left(1 - \frac{M}{2^{32}}\right)}$$

- With a /8 monitor there is no $h_0$ that can satisfy this inequality
  - Of course, for uniform scanning worms