I Trust My Zombies: A Trust-Enabled Botnet

Emmanouil Vasilomanolakis, Jan Helge Wolf, Leon Böck, Shankar Karuppayah, Max Mühlhäuser
Introduction #1

• Botnet monitoring is turning into a cat and mouse game...
• What if we start thinking like the bad guys?
Introduction #2

• Think as the attacker

• Envision the botnets of the future
  - Exploit the limitations of defenders
  - Mechanism for detecting the presence of sophisticated defenders

• Research Goal:
  - Botnet in which monitoring is difficult/infeasible
Terminology #1

P2P Botnet:

a number of bots that communicate in a P2P fashion
and in which a botmaster can issue commands
Terminology #2

**Membership Maintenance (MM) mechanism**

- Ensures overlay remains connected
- Periodically maintains a **Neighborlist (NL)**
  - Probes responsiveness frequently
  - Update/Replace entries as needed
  - Request additional neighbors

*The size of an NL ranges between 50-1000 entries*
How can P2P botnets be taken down?

- Reverse engineering
- Analysis
- Preliminaries

- Crawlers
- Sensors
- Monitoring

- Partition
- Sinkhole/Disarm
- Disrupt
Background: crawlers & sensors

Crawler
• Aggressively crawls the botnet
• Attempts to create a holistic image of the botnet asap
• Can be **easily** detected and contended

Sensor
• Acts like a normal bot and builds up its knowledge (slowly)
• Harder to create a holistic view of the botnet
• Very passive compared to crawlers
• **Cannot** be easily detected and contended
Background: Computational Trust

Two classes of evidence:
- interactions
- recommendations
- stereotypes
- credentials
- trustee analysis

Trust Computation

Subjective utility based decision

sound models (math, ML)
Background: the Sality P2P Botnet

• Early versions: 2003-2004!
• Very sophisticated all-around malware
• **P2P** since 2008
• Extremely resilient

• Communication protocol
  - Membership maintenance
  - Command dissemination
• Basic trust management
  - `goodCount`

Worldwide, 1st half 2016 (Source: Statista)
Background: Sality “Hello” messages

<table>
<thead>
<tr>
<th>Id</th>
<th>IP</th>
<th>lastOnline</th>
<th>goodCount</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1.2.3.4</td>
<td>2017-10-05 13:38:03</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>2.3.4.5</td>
<td>0000-00-00 00:00:00</td>
<td>5</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

cmdSeq (command sequence):
- an identifier of the latest command
  (simplified: the (internal) malware version)
Meet our Botnet

• Cautious: careful to whom you talk to
• Smart: learn from your past experiences

Core idea

• Defenders are bound to legal and ethical limitations
  □ They should not forward commands
  □ Exploitation via sending special messages (to neighbors)
• Utilization of computational trust
  □ Calculation and modeling of local knowledge
Meet our Botnet: Bogus Command Sequence (BCS) Messages

• Extend basic botnet protocol
• Introduce a novel type of message
  ❑ Based on the ethical/legal limitations of sensors/crawlers
  ❑ BCS message: indistinguishable from common hello messages
  ❑ Forces zombies to reveal their true self
Meet our Botnet:
BCS Messages #1

I Trust My Zombies | Blackhat Europe 2017

cmdSeq: 620

Bot A

(ridiculously low cmdSeq)

Hello (cmdSeq=310)

cmdSeq: 618

Bot B

HelloReply (cmdSeq=618, cmdSet=<...>)
Meet our Botnet:

BCS Messages #2

Bot A

Hello (cmdSeq=310)

HelloReply (cmdSeq=310, cmdSet=NULL)

Sensor
Meet our Botnet:
Trust Threshold and Blacklisting

• Autonomous trust score calculation
• Trust score check after every new experience
• Trust score below pre-defined trust threshold:
  - Remove peer from neighborlist
  - Add to blacklist
    - Prevent re-adding to neighborlist
    - Drop all incoming messages
• Irreversible decision
Meet our Botnet:
Utilized Botnet Trust Models

- Four trust models
  - EbayUserRating
  - BetaDistribution
  - SubjectiveLogic
  - CertainTrust
Experiments: objectives of monitoring

- Enumeration of the botnet
  - Sensor popularity (indegree)

- Decrease sensor popularity
- Blacklisting precision
  - \( p = \frac{TP}{TP+FP} \)
Experiments: setup

• Simulation environment
  • Botnet Simulation Framework (BSF) based on OMNeT++

• 5500 benign nodes
  • Churn

• 1/10/50 sensors
  • Permanently online
  • Cooperation among sensors

• Simulation time: 7 days

• 16 simulations per experiment
Experiments: Results – single-sensor

<table>
<thead>
<tr>
<th>Sensor</th>
<th>Popularity</th>
<th>TP</th>
<th>FP</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{eBay}(15)$</td>
<td>131</td>
<td>1959</td>
<td>24</td>
<td>0.98786</td>
</tr>
<tr>
<td>$T_{Beta}(3,3)$</td>
<td>393</td>
<td>2519</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_{CT}(0.5,5,0.95)$</td>
<td>717</td>
<td>2771</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>$T_{SL}(1)$</td>
<td>1120</td>
<td>2855</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Maximum Popularity <97%
Experiments: Results – multi-sensor (10)
Experiments: Results – colluding sensors

Comparison by number of sensors (BetaDistribution(3,3))
Conclusion

• The *cat and mouse* game will always benefit the mouse
  - Infinite ways to improve botnets
  - Cannot predict them all

• Monitoring P2P Botnets might become infeasible (soon)
  - We have shown how to decrease sensor effectiveness
    by *up to 97%*

• The war is still not lost: *collaboration* might be the key
  - Colluding sensors can provide an answer
Thank you!

Questions?