Detecting Stealthy P2P Botnets Using Statistical Traffic Fingerprints

Junjie Zhang¹, Roberto Perdisci², Wenke Lee¹, Unum Sarfraz¹, and Xiapu Luo³

¹Georgia Tech  ²Univ. of Georgia  ³Hong Kong Polytechnic Univ.
Outline

- Introduction
- System Design
- System Evaluation
- Conclusion
Introduction

- Botnet
  - A collection of bots controlled by a botmaster via a *command and control (C&C)* channel
- Botnets serve as the infrastructures for a variety of attacks
  - Spam
  - Identity Theft
  - DDoS
  - …
Introduction

- Botnets Evolve

- C&Cs
  - Centralized -> P2P

- Attacks
  - Noisy -> Stealthy

It is important to detect stealthy P2P botnets

However, existing P2P botnet detection methods may fail!
Introduction

- Existing Method 1
- BotMiner [Gu et al Security 08]
  - Correlates similar activities in communication plane and attack plane

- However, stealthy attacks may not be observable, making the correlation ineffective!
Introduction

- Existing Method 2
- BotGrep [Nagaraja et al Security 10]
  - Uses graph analysis to identify P2P networks
  - Requires global overview of network communications
  - Requires bootstrap information (e.g., honeypots)

- However, it is challenging to get
  - Network global overview
  - A priori detection results
- Resulting in limited usage in practice!
Introduction

- Existing Method 3
- Telling P2P File-Sharing and Bots Apart [Yen et al ICDCS 10]
  - Differentiate P2P bots from P2P file-sharing clients

However, the method may fail if bot-compromised host is running a P2P file-sharing application
Introduction

• We need a new method to detect stealthy P2P botnets!

• Design Targets:

  • Detect P2P bots without observing attack activities

  • Detect P2P bots even if the underlying hosts run legitimate P2P applications
System Design

- Architecture
- Traffic Reduction Using DNS

Phase I: Identify P2P Hosts

- Step 1.1. Traffic Reduction
- Step 1.2. Coarse-Grained Detection of P2P Hosts Using Failed Connections
- Step 1.3. Fine-Grained Detection of P2P Hosts Using Flow Clustering Analysis

Phase II: Identify P2P Bots

- Step 2.1. Coarse-Grained Detection of P2P Bots
- Step 2.2. Fine-Grained Detection of P2P Bots

---

<table>
<thead>
<tr>
<th>Time</th>
<th>SrcIP</th>
<th>DstIP</th>
<th>SrcPort</th>
<th>DstPort</th>
<th>Proto</th>
<th>PktS</th>
<th>PktR</th>
<th>ByteS</th>
<th>ByteR</th>
<th>Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>1305602966</td>
<td>192.168.1.5</td>
<td>74.125.115.103</td>
<td>2000</td>
<td>80</td>
<td>TCP</td>
<td>5</td>
<td>10</td>
<td>30</td>
<td>1000</td>
<td>SAF</td>
</tr>
<tr>
<td>1305602966</td>
<td>192.168.1.5</td>
<td>200.12.15.10</td>
<td>6000</td>
<td>1001</td>
<td>UDP</td>
<td>1</td>
<td>175</td>
<td>75</td>
<td>75</td>
<td>N/A</td>
</tr>
</tbody>
</table>

---
System Design

• Use real-world P2P traces to motivate the system design and parameter selection
  • 5 popular P2P applications
  • 24 hours

<table>
<thead>
<tr>
<th>P2P Applications</th>
<th>Version</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bittorrent</td>
<td>6.4</td>
<td>Bittorrent</td>
</tr>
<tr>
<td>Emule</td>
<td>0.49c</td>
<td>Kademlia</td>
</tr>
<tr>
<td>Limewire</td>
<td>5.4.8</td>
<td>Gnutella&amp;Bittorrent</td>
</tr>
<tr>
<td>Skype</td>
<td>4.2</td>
<td>Skype</td>
</tr>
<tr>
<td>Ares</td>
<td>2.1.5</td>
<td>Gnutella&amp;Bittorrent</td>
</tr>
</tbody>
</table>
System Design – Phase I: Identify P2P Hosts

• Step 1.1 Traffic Reduction
  • Observation: The peer IP addresses of P2P connections are obtained from the overlay routing tables instead of DNS responses.
  • System Design: Filter out the flows, whose destination IP addresses are resolved from DNS queries.

<table>
<thead>
<tr>
<th>Trace</th>
<th>Percentage of flows associated with no-DNS DstIPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-Bittorrent</td>
<td>96.85%</td>
</tr>
<tr>
<td>T-Emule</td>
<td>99.99%</td>
</tr>
<tr>
<td>T-Limewire</td>
<td>99.97%</td>
</tr>
<tr>
<td>T-Skype</td>
<td>99.93%</td>
</tr>
<tr>
<td>T-Ares</td>
<td>99.99%</td>
</tr>
</tbody>
</table>
System Design – Phase I: Identify P2P Hosts

- Step 1.2 Coarse-Grained Detection of P2P Hosts
  - **Observation:** The significant node churn of P2P networks introduces a large number of failed connections.
  - **System Design:** Identify hosts that initiate more than $\Theta_o$ failed connections for each day.

<table>
<thead>
<tr>
<th>Trace</th>
<th># of Failed Conns Per hour</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-Bittorrent</td>
<td>1602</td>
</tr>
<tr>
<td>T-Emule</td>
<td>318</td>
</tr>
<tr>
<td>T-Limewire</td>
<td>1278</td>
</tr>
<tr>
<td>T-Skype</td>
<td>81</td>
</tr>
<tr>
<td>T-Ares</td>
<td>489</td>
</tr>
</tbody>
</table>

Conservatively set $\Theta_o = 10$
System Design – Phase I: Identify P2P Hosts

• Step 1.3 Fine-Grained Detection of P2P Hosts
  • Observation:
    • A P2P application automatically generates a large number of control flows (e.g., ping/pong) to a large number of peers, as long as it is online.
    • The control flows used for the same purpose (e.g., ping/pong or peer discovery) share similar size.
  • System Design:
    • Aggregate flows with similar sizes to clusters respectively.
    • Define a cluster as a fingerprint cluster, if the number of unique BGP prefixes of the destination IP addresses for its flows is greater than $\theta_{BGP}$ ($=50$).
    • Claim a host as P2P host if it has at least one fingerprint cluster.
System Design – Phase I: Identify P2P Hosts

- An example of control flows
System Design – Phase I: Identify P2P Hosts

• Step 1.3. Flow-Clustering analysis for identifying fingerprint
  • Scalable clustering scheme (Birch + Hierarchical Clustering).
  • Refer paper for details.
System Design – Phase II: Identify P2P Bots

• Step 2.1 Coarse-Grained Detection of P2P Bots
  • Motivation:
    • A bot usually is active as long as its underlying system is power on
    • A legitimate P2P application may only be used for a short period of time
  • System Design:
    • Identify P2P clients that are **persistently active** compared to its underlying host
      \[
      \frac{T(P2P)}{T(Host)} > 0.5
      \]
  • Challenge: How can we accurately estimate the active time of a P2P application (i.e., T(P2P))?
  • Solution: Leverage obtained fingerprint clusters!
System Design – Phase II: Identify P2P Bots

- Estimate the active time of a P2P application ($T(P2P)$)
  
  $Dur(P2P) = \text{MAX}(Dur(FC_1), Dur(FC_2), \ldots, Dur(FC_n))$

<table>
<thead>
<tr>
<th>Trace</th>
<th>Actual P2P Active Time ($T(P2P)$)</th>
<th>Estimated P2P Active Time ($Dur(P2P)$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T-Bittorrent</td>
<td>24hr</td>
<td>24hr</td>
</tr>
<tr>
<td>T-Emule</td>
<td>24hr</td>
<td>24hr</td>
</tr>
<tr>
<td>T-Limewire</td>
<td>24hr</td>
<td>24hr</td>
</tr>
<tr>
<td>T-Skype</td>
<td>24hr</td>
<td>24hr</td>
</tr>
<tr>
<td>T-Ares</td>
<td>24hr</td>
<td>24hr</td>
</tr>
</tbody>
</table>

- $T(P2P) \approx Dur(P2P)$.
- $Dur(P2P)$ can accurately approximates $T(P2P)$!
System Design – Phase II: Identify P2P Bots

• Fine-Grained Detection of P2P Bots
  • Motivation
    • Same P2P applications, such as P2P bots of the same botnet, use the same P2P protocol, resulting in similar/same fingerprint clusters

    • P2P bots in the same botnet search for the same commands/contents. Therefore the search paths from bots tend to converge, resulting in a large overlap of peers.

    • Legitimate P2P clients are driven by different users, searching for different contents. Therefore the search paths for legitimate clients tend to diverge, resulting in a small overlap of peers.
Fine-Grained Detection of P2P Bots

System Design

If two persistently active P2P clients share similar fingerprint clusters and a large overlap of destination IP addresses, we take them as bots.

Represent a fingerprint cluster \( FC_i \) using

Define distances between two fingerprint clusters

Define the distance between two hosts

The fingerprint cluster from Emule will not affect the distance of two bot-compromised hosts

The bot-compromised system runs an Emule application
System Design – Phase II: Identify P2P Bots

- Fine-Grained Detection of P2P Bots
  - System Design: If two persistently active P2P clients have small distance, we identify them as bots

```
P2P clients exhibiting persistent activities

Hierarchical Clustering

Cut at \( \theta_{bot} \times \text{height}_{max}, \theta_{bot} = 0.95 \)
```
# System Evaluation

## Experimental Setup

<table>
<thead>
<tr>
<th>Data Sources</th>
<th>Description</th>
<th>P2P Clients/Traces</th>
</tr>
</thead>
<tbody>
<tr>
<td>College networks</td>
<td>Passively collected for 24 hours; Netflows with 200 bytes payload between internal and external networks; DNS responses with IP addresses</td>
<td>3 Bittorrents (ground truth by signatures); 5 Skypes (potential)</td>
</tr>
<tr>
<td>5 P2P applications</td>
<td>Run each application <em>simultaneously</em> on two VMs for 24 (or 5) hours</td>
<td>10 traces totally</td>
</tr>
<tr>
<td>2 P2P botnets</td>
<td>Execute the malware binaries of Storm/Waledac for 24 hours</td>
<td>13 Storm bots; 3 Waledac bots</td>
</tr>
</tbody>
</table>
System Evaluation

- Experimental Design

- Identifying and Profiling P2P applications

- Detecting P2P bots
  - With being mixed with legitimate P2P data

  - Other situations are discussed in the paper including
    - Without being mixed with legitimate P2P data
    - Only two bots from each botnet
    - No bot (clean network)
System Evaluation

- Identifying and Profiling P2P Applications
  - DNS-based filter eliminates 2/3 hosts in the college networks!
  - Effectively identify and profile all the P2P applications!
    - 3 Bittorrent hosts in college networks
    - 5 potential Skypes in college networks
    - 10 P2P application instances we have run
    - 3 Waledac P2P bots
    - 13 Storm bots

Let us have a close look at Fingerprint Clusters!
## System Evaluation

- Identifying and Profiling P2P Applications
  - Fingerprint cluster summaries for Bittorrent clients in college networks
  - Fingerprint cluster summary: average number of pkt_sent/recv, byte_sent/recv, and protocol for one fingerprint cluster

<table>
<thead>
<tr>
<th>Fingerprint</th>
<th>Ground Truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\overline{\text{pkt}_s, \text{pkt}_r, \text{bytes}_s, \text{bytes}_r, \text{proto}}$</td>
<td>$\begin{array}{c}1 1 145 319, \text{UDP} \ 1 1 109 100, \text{UDP} \ 1 1 146 340, \text{UDP} \ 5 3 346 170, \text{TCP} \ 1 1 145 310, \text{UDP} \end{array}$</td>
</tr>
</tbody>
</table>

| Fingerprint identified in the college network | $\begin{array}{c}1 1 109 100, \text{UDP} \\ 1 1 109 91, \text{UDP} \\ 1 1 104 178, \text{UDP} \\ 1 1 319 145, \text{UDP} \\ 1 1 145 319, \text{UDP} \end{array}$ |

<table>
<thead>
<tr>
<th>T-Bittorrent</th>
<th>BT1@C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{array}{c}1 1 145.01 317.66, \text{UDP} \end{array}$</td>
<td>$\begin{array}{c}1 1 145 319, \text{UDP} \end{array}$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>T-Bittorrent-2</th>
<th>BT2@C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\begin{array}{c}1 1 145 319, \text{UDP} \end{array}$</td>
<td>$\begin{array}{c}1 1 145 319, \text{UDP} \end{array}$</td>
</tr>
</tbody>
</table>

| BT3@C | $\begin{array}{c}7 6 1118 1767, \text{TCP} \end{array}$ |
System Evaluation

- Identifying and Profiling P2P Applications
  - Fingerprint cluster summaries for Skype clients in college networks

<table>
<thead>
<tr>
<th>Fingerprint cluster summaries for Skype clients in college networks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Skype</td>
</tr>
<tr>
<td>Skype1@C</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Skype2@C</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Skype3@C</td>
</tr>
<tr>
<td></td>
</tr>
<tr>
<td>Skype4@C</td>
</tr>
<tr>
<td>Skype5@C</td>
</tr>
</tbody>
</table>
System Evaluation

- Identifying and Profiling P2P Applications
- Fingerprint cluster summaries for P2P bots

The fingerprint clusters can effectively identify and profile different P2P applications!
System Evaluation

- Detecting P2P bots
  - Overlaying legitimate P2P data and P2P botnet data to college network data.
  - half of bot-compromised hosts run legitimate P2P applications.

<table>
<thead>
<tr>
<th>Bot</th>
<th>P2P App</th>
<th>Before Overlay (Bot)</th>
<th>After Overlay (Bot+P2PApp)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># of flows</td>
<td># of DstIPs</td>
<td>avg flow size</td>
</tr>
<tr>
<td>Waledac1</td>
<td>Emule1</td>
<td>341784</td>
<td>850</td>
</tr>
<tr>
<td>Waledac2</td>
<td>BT2@C</td>
<td>319119</td>
<td>760</td>
</tr>
<tr>
<td>Storm1</td>
<td>Limewire1</td>
<td>200237</td>
<td>6390</td>
</tr>
<tr>
<td>Storm2</td>
<td>BT3@C</td>
<td>275451</td>
<td>7319</td>
</tr>
<tr>
<td>Storm3</td>
<td>Bittorrent2</td>
<td>133955</td>
<td>5584</td>
</tr>
<tr>
<td>Storm4</td>
<td>Skype4@C</td>
<td>171471</td>
<td>7277</td>
</tr>
<tr>
<td>Storm5</td>
<td>Skype1</td>
<td>164917</td>
<td>6686</td>
</tr>
<tr>
<td>Storm6</td>
<td>Ares1</td>
<td>220459</td>
<td>6618</td>
</tr>
</tbody>
</table>

**TABLE IX: Bot Traces Overlapped with P2P Application Traces**

Bot traffic profiles are significantly skewed by the legitimate P2P data, which could make the existing detection approach such as [Yen et al. ICDCS10] ineffective.
System Evaluation

- Detecting P2P Bots
  - Fingerprint clusters of P2P bots overlapped with legitimate P2P applications

<table>
<thead>
<tr>
<th>Application</th>
<th>Fingerprints</th>
</tr>
</thead>
</table>
| Waledac2+BT2@C | 1 1 145 319, UDP (Bittorrent)  
                                    4 3 224 170, TCP (Waledac)  
                                    3 3 185 162, TCP (Waledac)  
                                    1 1 75 75, UDP (Bittorrent)  
                                    ... |
| Storm4+Skype4@C | 2 2 94 554, UDP (Storm)  
                                    2 2 94 1014, UDP (Storm)  
                                    1 1 73 60, UDP (Skype)  
                                    ... |
System Evaluation

- Detecting P2P Bots

Refer paper for more evaluation of $\Theta_{\text{bot}}$

<table>
<thead>
<tr>
<th>Detection Rate</th>
<th>100%</th>
</tr>
</thead>
<tbody>
<tr>
<td>False Positive Rate</td>
<td>0.2% (2/953)</td>
</tr>
</tbody>
</table>
Conclusion

- A new P2P traffic detection algorithm
  - identify and *profile* P2P applications

- A novel P2P botnet detection system
  - Detect P2P bots without observing malicious activities
  - Detect P2P bots even if legitimate P2P applications are running on the underlying systems
  - High detection rate and low false positive rate
End

- Thanks!
System Evaluation

Detection Rate 100%
False Positive Rate 0.2% (2/953)

Cut at 0.475 = 0.95 * 0.5
System Design – Phase I: Identify P2P Hosts

- Two steps of clustering analysis to generate fingerprint clusters.
System Evaluation

- Experimental Results
- Scalability Regarding Time Consumption

Achieving good detection rates for a large range of $\Theta_{bot}$ (e.g., $[0.7 - 0.95]$).
Achieving good scalability by tuning $\text{Cnt}_{\text{birch}}$. 

A small $\text{Cnt}_{\text{birch}}$ increases system performance, but may decrease accuracy of fingerprint clusters.

<table>
<thead>
<tr>
<th>$\text{Cnt}_{\text{birch}}$</th>
<th>-</th>
<th>0.1</th>
<th>0.3</th>
<th>0.5</th>
<th>$\Theta_{bot}$</th>
<th>0.8</th>
<th>0.9</th>
<th>0.95</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>DR FP</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2/16</td>
<td>3/16</td>
<td>16/16</td>
<td>16/16</td>
</tr>
<tr>
<td>4000</td>
<td>DR FP</td>
<td>2/16</td>
<td>3/16</td>
<td>3/16</td>
<td>16/16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>8000</td>
<td>DR FP</td>
<td>2/16</td>
<td>3/16</td>
<td>3/16</td>
<td>16/16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>10000</td>
<td>DR FP</td>
<td>2/16</td>
<td>3/16</td>
<td>3/16</td>
<td>16/16</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>