DETECTING MALICIOUS CLIENTS IN ISP NETWORKS USING HTTP CONNECTIVITY GRAPH AND FLOW INFORMATION

Lei Liu¹ Sabyasachi (Saby) Saha² Ruben Torres² Jianpeng Xu¹ Pang-Ning Tan¹ Antonio Nucci² Marco Mellia³

¹Michigan State University, Michigan, USA
²Narus, Inc., Sunnyvale, California, USA
³Politecnico di Torino, Italy
Introduction

• Malware is …
  – Malicious software
  – Virus, Phishing, Spam, …

• Increasing threats
  – 4500 new Web attacks launched per day (Symantec Security Report)
  – Continuous and increased attacks on infrastructure
  – Threats to business, national security
    • Huge financial stake (Conficker: 10 million machines, loss $9.1 Billion)
    • Zeus: 3.6 million machines [HTML Injection]
    • Koobface: 2.9 million machines [Social Networking Sites]
    • TidServ: 1.5 million machines [Email spam attachment]

• Attacks are becoming more advanced and sophisticated!
Introduction

• Limitation of existing techniques
  – Signature-based approach
    • Fails to detect zero-day attacks.
    • Fails to detect threats with evolving capabilities such as metamorphic and polymorphic malwares.
  – Anomaly-based approach
    • Producing high false alarm rate.
  – Supervised Learning based approach
    • Poor performance on novel malware

There is no Silver Bullet
Introduction

• However, the malware cannot hide the communication
  – We know who talks to whom (Connectivity Graph)
  – We can extract some information about what has been communicated

• **Goal**: Augment current security solutions using connectivity graph and flow information to find hidden malicious nodes
HTTP Connectivity

- We focus on HTTP Traffic
  - Most of the malwares are in HTTP
  - Difficult to block
One Example

- CL1|cantst0pme11124never2287.net|/run/file.php
- CL1|google.com|
- CL1|autoupdates2012.in|/cb/file.php
- CL2|cantst0pme11124never2287.net|/run/file.php
- CL2|google.com|
- CL2|autoupdates2012.in|/cb/file.php
- CL3|cantst0pme11124never2287.net|/run/file.php
- CL3|google.com|
- CL3|autoupdates2012.in|/cb/file.php
- CL4|google.com|
- CL4|autoupdates2012.in|/cb/file.php
- CL5|google.com|
- CLX|google.com|
- CLN|google.com|
Our Approach

• We propose a two step malicious score propagation approach to identify other malicious nodes in the network
  – Initialize with the malicious nodes having a non-zero score and others having a zero score
VirusTotal is a free service that analyzes suspicious files and URLs and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware.
Example Contd.

- CL1|cantst0pme11124never2287.net|/run/file.php
- CL1|google.com|
- CL1|autoupdates2012.in|/cb/file.php
- CL2|cantst0pme11124never2287.net|/run/file.php
- CL2|google.com|
- CL2|autoupdates2012.in|/cb/file.php
- CL3|cantst0pme11124never2287.net|/run/file.php
- CL3|google.com|
- CL3|autoupdates2012.in|/cb/file.php
- CL4|google.com|
- CL4|autoupdates2012.in|/cb/file.php
- CL5|google.com|
- CLX|google.com|
- CLX|NOTSOCOMMON.com|index.html
- CLN|google.com|
- CLN|NOTSOCOMMON.com|index.html
- CLY|NOTSOCOMMON.com|index.html
Our Approach

• We weigh the edges of the graph using malicious flow similarity
  – If the flow through the edge has any similarity with the malicious flows in the data
  – Used a SVM based classifier with 270 flow-based features

• Then use the same two step malicious score propagation approach to identify other malicious nodes in the network
System Architecture
HTTP Graph Construction

- HTTP bipartite graph
  - $G = ((C,S),E)$ denote a directed graph constructed from the HTTP connections
  - $C$, set of client IP addresses
  - $S$, set of server IP addresses
  - $E$, set of directed links
Two-phase Alternating Score propagation

- **Objective Function**

\[
Q(y) = \frac{1}{2} \sum_{i,j} W_{ij} \left[ \frac{y_i}{\sqrt{D_{ii}}} - \frac{y_j}{\sqrt{D_{jj}}} \right]^2 + \frac{\mu}{2} \sum_i (y_i - y_i^{(0)})^2
\]

Where \( W \) is a adjacency matrix, and \( D \) is a diagonal matrix whose diagonal elements are given by \( D_{ii} = \sum_j W_{ij} \)

- Our objective can be reduced to:

\[
x_i = (1 - \beta_s)x_i^0 + \beta_s \sum_{k \in C} w_{ki}^{cs} y_k \quad (1)
\]

\[
y_k = (1 - \beta_c)y_k^0 + \beta_c \sum_{j \in S} w_{jk}^{sc} x_j \quad (2)
\]

- Iteratively updating the malicious score based on its initial value and weighted average of scores for its neighbors until convergence
Two-phase Alternating Score propagation

- **Link-only**: the weight of a link depends on the existence of a flow between the node pair

\[
\begin{align*}
\hat{w}_{ij}^{(l)} &= \begin{cases} 
1, & \text{if } (v_i, v_j) \in \mathcal{E} \text{ (or } \pi_{ij} \neq \emptyset) \\
0, & \text{otherwise.}
\end{cases}
\end{align*}
\]

- Normalizing the link weight by its out-degree:

\[
\hat{W}_{ij} = \frac{w_{ij}}{\sum_j w_{ij}}
\]
Two-phase Alternating Score propagation

• Example

Transition matrix

\[ W^{cs} = \begin{bmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 1 & 0 & 0 \\ 0 & \frac{1}{2} & \frac{1}{2} \end{bmatrix}, \quad W^{sc} = \begin{bmatrix} \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & 1 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{bmatrix} \]

Score propagation

<table>
<thead>
<tr>
<th>node</th>
<th>initial score</th>
<th>iteration 1</th>
<th>iteration 2</th>
<th>iteration 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>( c_1 )</td>
<td>1</td>
<td>0.7556</td>
<td>0.7230</td>
<td>0.7220</td>
</tr>
<tr>
<td>( c_2 )</td>
<td>0</td>
<td>0.2444</td>
<td>0.2981</td>
<td>0.3222</td>
</tr>
<tr>
<td>( c_3 )</td>
<td>1</td>
<td>0.8725</td>
<td>0.8485</td>
<td>0.8253</td>
</tr>
<tr>
<td>( s_1 )</td>
<td>1</td>
<td>0.575</td>
<td>0.6165</td>
<td>0.6529</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>0</td>
<td>0.425</td>
<td>0.4346</td>
<td>0.4258</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>0</td>
<td>0.85</td>
<td>0.8195</td>
<td>0.7908</td>
</tr>
</tbody>
</table>
HTTP Flow Features

| SessionID | SourceIP | DestIP | 1314281856 | example.com | mozilla/2.0 | 0 | 1 | 0 | 63973 | 80 | 477 | 1628 | 189 | 1460 | 7 | 4 | 1 | / | blog/images/3521.jpg | GET | tq=RA1DQxZBDVlUFQN0AQUDAh | 57 | 200 OK | image/jpeg | 190984 |

- **Hostname**
  - Length of the second domain
  - Randomness of the second domain
  - Is it a IP address?
  - Reliability score from .com, .info, etc.
    - Etc.

- **User agent**

- **URI**
  - Keywords separated by delimiters
  - Length of the URI
  - # of fields

- **Referrer**

- **Method (GET, POST)**

- **Additional parameters**
  - Length of the first key, value (categorical)
  - (For each key-value pair) characters, numbers or mix

- **Server status**
- **Content type**
- **Error on request**
- **Number of requests**
- **Number of URLs requested in a session**
- **Number of pages requested in a session**
- **Bytes Sent**
- **Bytes Received**
- **Data sent, Data received, Pkts sent, Pkt rcvd**
- **Response time**
- **Bytes transferred**
- **Content length**
Flow Classification

- Classifying from malicious and unknown flows

\[
\min_{\omega, b} \quad \frac{1}{2} \omega^T \omega + C^l \sum_{i=1}^{k-1} \xi_i + C^u \sum_{j=k}^{m} \xi_j \\
\text{subject to} \quad y_{ii} (\omega^T \phi(x_{ii}) + b) \geq 1 - \xi_i \\
y_{ju} (\omega^T \phi(x_{ju}) + b) \geq 1 - \xi_j \\
\xi_i \geq 0, \quad i = 1, 2, \ldots, k - 1 \\
\xi_j \geq 0, \quad j = k, k + 1, \ldots, m
\]

Cost for misclassifying malicious flows

Cost for misclassifying unknown flows

in which, assigning \( C^l > C^u \) could guide the classifier towards classifying more accurately flows that belong to malicious class.
Flow-based Graph Construction

• Combining Flow information and Link structure:
  – The weight of a link is determined from its malicious score:

\[ w_{ij}(f) = \begin{cases} 1, & \text{if } \exists f_{ijk} \in \pi_{ij} : \mathcal{I}(f_{ijk}) = 1; \\ \max_{\sigma_k \in \Sigma_{ij}} \{\sigma_k\}, & \text{otherwise.} \end{cases} \]

• Link is associated with set of flows \[ \pi_{ij} = \{f_{ij1}, f_{ij2}, \cdots, f_{ij|\pi_{ij}|}\} \]

• If the malicious score for the link from IDS is 1, we set the weight of the edge as 1
• Otherwise, we set it to the maximum value of the flow classification
Experimental Settings

• Data Sets
  – Four 1-hour data sets, which we name D1~D4, more detailed information in following table.

<table>
<thead>
<tr>
<th>Data set</th>
<th>Time of day</th>
<th>Number of HTTP flows</th>
<th>Number of labeled flows</th>
<th>Number of malicious clients/ Total number of clients</th>
<th>Number of malicious servers/ Total number of servers</th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>4 pm</td>
<td>1926620</td>
<td>1088</td>
<td>26/5856</td>
<td>25/25627</td>
</tr>
<tr>
<td>D2</td>
<td>6 pm</td>
<td>973271</td>
<td>1649</td>
<td>30/6533</td>
<td>25/26346</td>
</tr>
<tr>
<td>D3</td>
<td>7 pm</td>
<td>1033292</td>
<td>1736</td>
<td>26/7360</td>
<td>22/26216</td>
</tr>
<tr>
<td>D4</td>
<td>9 pm</td>
<td>1078765</td>
<td>65</td>
<td>21/7936</td>
<td>23/27518</td>
</tr>
</tbody>
</table>

– Used a commercial IDS to identify flag malicious flows
Experimental Evaluation

• Validation
  – To validate a predicted client, which receives a high propagation score, we check if this client connects to any malicious web server
  – To validate the web server
    • Google SafeBrowsing, Malware Blacklists
    • WOT (Web of Trust) score
Experimental Evaluation

• **Metric**
  
  – The precision of top n ranking clients is used to indicate the final performance.

\[
P@n = \frac{num(TM)}{n}
\]

• *Num(TM)* is the number of true malicious clients show up in top n clients. We report the precision from p@1 to p@1000.
Results

• Results Comparison for All Clients

Precision at top n clients on D1 ~ D4 with threshold Num = 1,5,10
Results

- Results of Clients that are indirectly Connect to Malicious nodes

Precision at top n highest ranked clients that are indirectly connected to malicious hosts
Conclusion

• Proposed a method that combine the links and flow-level information in the HTTP communication graph for malicious clients detection

• Proposed an efficient two-phase score propagation algorithm to identify malicious clients

• Experimental results on large ISP data verified that our proposed method could detect clients that infected with known/new malwares
Future Work

• Extent the framework beyond HTTP traffic
  – Flow features will change
• To be added
Thank You
Future Work

• Validate with diverse data sets
• Propagating malicious score for server by treating each URL as a node instead of server IP.
• Consider the hidden connections in URL-URL connectivity graph.