Automatic Analysis of Malware Behavior using Machine Learning

Konrad Rieck, Philipp Trinius, Carsten Willems, Thorsten Holz

Peng Su

CISC850
Cyber Analytics
Automatic Analysis of Malware Behavior

• Malware threaten the Internet
• Dynamic VS Static
• binary packers, encryption, or self-modifying code, to obstruct analysis.
• behavior of malicious software during run-time.
Automatic Analysis of Malware Behavior

Malware binaries → Monitoring of behavior → Embedding of behavior → Classification using prototypes → Prototype extraction → Clustering using prototypes → Classified reports

Prototypes of classes
Monitoring of Malware Behavior

- Malware Sandboxes -- CW Sandbox
- Malware Instruction Set

```
<move_file srcfile="c:\foo.exe" dstfile="c:\windows\system32\kernel32.dll" filetype="file" creationdistribution="CREATE_NEW" />
```

(a) CW Sandbox representation of system call

```
move_file | create flags | "exe" | "c:\" | "dll" | "c:\w..." | "foo" | "kernel"
```

(b) MIST representation of system call
Malware Instruction Set

- MIST instruction keep the stable and discriminative patterns such as directory and mutex name at the beginning.
Embedding of Malware Behavior

• Embedding using Instruction Q-grams

• Comparing Embedding reports
Embedding using Instruction Q-grams

\[ \varphi(x) = (\varphi_s(x))_{s \in S} \text{ with } \varphi_s(x) = \begin{cases} 1 & \text{if report } x \text{ contains } q\text{-grams } s, \\ 0 & \text{otherwise.} \end{cases} \]

- For example, if report \( x = '1|A 2|A 1|A 2|A' \), \( A = \{1|A, 2|A\} \), the \( q \) for \( q\)-grams is 2.
Embedding using Instruction Q-grams

- Normalization
- Redundancy of behavior, considered alphabet, length of reports

\[ \hat{\phi}(x) = \frac{\phi(x)}{||\phi(x)||} \]
Comparing Embedding reports

• Euclidean distance

\[ d(x, z) = \| \hat{\phi}(x) - \hat{\phi}(z) \| = \sqrt{\sum_{s \in S} (\hat{\phi}_s(x) - \hat{\phi}_s(z))^2} \]
Clustering and Classification

- Prototypes -> Clustering -> Classification
Prototype Extraction

Algorithm 1 Prototype extraction

1: prototypes ← ∅
2: distance[x] ← ∞ for all x ∈ reports
3: while max(distance) > d_p do
4:   choose z such that distance[z] = max(distance)
5:   for x ∈ reports and x ≠ z do
6:     if distance[x] > ∥ϕ(x) − ϕ(z)∥ then
7:       distance[x] ← ∥ϕ(x) − ϕ(z)∥
8:   add z to prototypes
Clustering using Prototypes

Algorithm 2 Clustering using prototypes

1: for $z, z' \in \text{prototypes}$ do
2:   $\text{distance}[z, z'] \leftarrow ||\hat{\phi}(z) - \hat{\phi}(z')||$
3: while $\min(\text{distance}) < d_c$ do
4:   merge clusters $z, z'$ with minimum $\text{distance}[z, z']$
5:   update $\text{distance}$ using complete linkage
6: for $x \in \text{reports}$ do
7:   $z \leftarrow$ nearest prototype to $x$
8:   assign $x$ to cluster of $z$
9: reject clusters with less than $m$ members
Algorithm 3 Classification using prototypes

1: \textbf{for} \( x \in \text{reports} \) \textbf{do}
2: \hspace{1em} \( z \leftarrow \) nearest prototype to \( x \)
3: \hspace{1em} \textbf{if} \( \|\phi(z) - \phi(x)\| > d_r \) \textbf{then}
4: \hspace{2em} reject \( x \) as unknown class
5: \hspace{1em} \textbf{else}
6: \hspace{2em} assign \( x \) to cluster of \( z \)
**Incremental Analysis**

---

**Algorithm 4 Incremental Analysis**

1. `rejected ← ∅, prototypes ← ∅`
2. **for** `reports ← data source ∪ rejected **do**`
   - classify `reports` to known clusters using `prototypes`
   - extract prototypes from remaining `reports`
   - cluster remaining `reports` using prototypes
   - `prototypes ← prototypes ∪ prototypes of new clusters`
   - `rejected ← rejected reports from clustering`
Experiments & Application

- Evaluation Data
  - Three parameters to decide
- Evaluation of Components
  - How to select the best parameters $d_p$, $d_c$, $d_r$
Evaluation Data

- A reference data set
- Evaluate and calibrate the framework
- An application data set
- See the performance on unknown malwares
## Reference Data Set

<table>
<thead>
<tr>
<th>Malware class</th>
<th>#</th>
<th>Malware class</th>
<th>#</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>a</strong> ADULTBROWSER</td>
<td>262</td>
<td><strong>m</strong> PORNDIALER</td>
<td>98</td>
</tr>
<tr>
<td><strong>b</strong> ALLAPEL*</td>
<td>300</td>
<td><strong>n</strong> RBOT</td>
<td>101</td>
</tr>
<tr>
<td><strong>c</strong> BANCOS</td>
<td>48</td>
<td><strong>o</strong> ROTATOR*</td>
<td>300</td>
</tr>
<tr>
<td><strong>d</strong> CASINO</td>
<td>140</td>
<td><strong>p</strong> SALITY</td>
<td>85</td>
</tr>
<tr>
<td><strong>e</strong> DORFDO</td>
<td>65</td>
<td><strong>q</strong> SPYGAMES</td>
<td>139</td>
</tr>
<tr>
<td><strong>f</strong> EJIK</td>
<td>168</td>
<td><strong>r</strong> SWIZZOR</td>
<td>78</td>
</tr>
<tr>
<td><strong>g</strong> FLYSTUDIO</td>
<td>33</td>
<td><strong>s</strong> VAPSUP</td>
<td>45</td>
</tr>
<tr>
<td><strong>h</strong> LDPINCH</td>
<td>43</td>
<td><strong>t</strong> VIKINGDLL</td>
<td>158</td>
</tr>
<tr>
<td><strong>i</strong> LOOPER</td>
<td>209</td>
<td><strong>u</strong> VIKINGDZ</td>
<td>68</td>
</tr>
<tr>
<td><strong>j</strong> MAGICCASINO</td>
<td>174</td>
<td><strong>v</strong> VIRUT</td>
<td>202</td>
</tr>
<tr>
<td><strong>k</strong> PODNUHA*</td>
<td>300</td>
<td><strong>w</strong> WOIKOINER</td>
<td>50</td>
</tr>
<tr>
<td><strong>l</strong> POSION</td>
<td>26</td>
<td><strong>x</strong> ZHELATIN</td>
<td>41</td>
</tr>
</tbody>
</table>
## Application Data Set

<table>
<thead>
<tr>
<th>Data set description</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Collection period</td>
<td>August 1–7, 2009</td>
</tr>
<tr>
<td>Collection location</td>
<td>Sunbelt Software</td>
</tr>
<tr>
<td>Data set size (kilobytes)</td>
<td>21,808,644</td>
</tr>
<tr>
<td>Number of reports</td>
<td>33,698</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Data set statistics</th>
<th>min.</th>
<th>avg.</th>
<th>max.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reports per day</td>
<td>3,760</td>
<td>4,814</td>
<td>6,746</td>
</tr>
<tr>
<td>Instructions per report</td>
<td>15</td>
<td>11,921</td>
<td>103,039</td>
</tr>
<tr>
<td>Size per report (kilobytes)</td>
<td>1</td>
<td>647</td>
<td>5,783</td>
</tr>
</tbody>
</table>
Evaluation of Components

• Precision and recall

\[ P = \frac{1}{n} \sum_{c \in C} \#_c \quad \text{and} \quad R = \frac{1}{n} \sum_{y \in Y} \#_y \]
Evaluation of Components

• F-measure

\[ F = \frac{2 \cdot P \cdot R}{P + R} \]
Evaluation of Components--$d_p$

(a) MIST level 1

(b) MIST level 2
Evaluation of Components--$d_c$

(a) MIST level 1

(b) MIST level 2
Evaluation of Components--$d_r$
## Comparative Evaluation with State-of-the-Art

<table>
<thead>
<tr>
<th>Clustering methods</th>
<th>$F$-measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Clustering using prototypes with MIST level 1</td>
<td>0.950</td>
</tr>
<tr>
<td>Clustering using prototypes with MIST level 2</td>
<td>0.936</td>
</tr>
<tr>
<td>Clustering using LSH with Anubis features (Bayer et al., 2009a)</td>
<td>0.881</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Classification methods</th>
<th>$F_k$</th>
<th>$F_u$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Classification using prototypes with MIST level 1</td>
<td>0.981</td>
<td>0.997</td>
</tr>
<tr>
<td>Classification using prototypes with MIST level 2</td>
<td>0.972</td>
<td>0.964</td>
</tr>
<tr>
<td>Classification using SVM with XML features (Rieck et al., 2008)</td>
<td>0.807</td>
<td>0.548</td>
</tr>
</tbody>
</table>
An Application Scenario

(a) Comparison with regular clustering

(b) Clustering-classification ratio