**Barnum**: Detecting Document Malware via Control Flow Anomalies in Hardware Traces

Carter Yagemann, Salmin Sultana, Li Chen, Wenke Lee

September 18, 2019
Document malware is still an open problem.
Fig. 15. An example of obfuscation observed during clustering. The variable `arr` is randomized with key `q` in `cc`. 
Hi Carter, we haven't received your camera-ready of ISC. Any reason for that?
Dream

Signatures

Malware
Reality
Automatically Evading Classifiers
A Case Study on PDF Malware Classifiers

Weilin Xu, Yanjun Qi, and David Evans
University of Virginia
http://www.EvadeML.org
Static Analysis

Dynamic Analysis
APIs Are Complicated

JavaScript® for Acrobat® API Reference

Adobe® Acrobat® SDK
Version 8.1
April 2007

769 Pages
<table>
<thead>
<tr>
<th>Vuln ID</th>
<th>Summary</th>
<th>CVSS Severity</th>
</tr>
</thead>
<tbody>
<tr>
<td>CVE-2019-7089</td>
<td>Adobe Acrobat and Reader versions 2019.010.20069 and earlier, 2019.010.20069 and earlier, 2017.011.30113 and earlier version, and 2015.006.30464 and earlier have a data leakage (sensitive) vulnerability. Successful exploitation could lead to information disclosure.</td>
<td>7.8 HIGH</td>
</tr>
<tr>
<td>CVE-2019-7087</td>
<td>Adobe Acrobat and Reader versions 2019.010.20069 and earlier, 2019.010.20069 and earlier, 2017.011.30113 and earlier version, and 2015.006.30464 and earlier have a type confusion vulnerability. Successful exploitation could lead to arbitrary code execution.</td>
<td>5.0 MEDIUM</td>
</tr>
<tr>
<td>CVE-2019-7086</td>
<td>Adobe Acrobat and Reader versions 2019.010.20069 and earlier, 2019.010.20069 and earlier, 2017.011.30113 and earlier version, and 2015.006.30464 and earlier have a type confusion vulnerability. Successful exploitation could lead to arbitrary code execution.</td>
<td>5.0 MEDIUM</td>
</tr>
<tr>
<td>CVE-2019-7083</td>
<td>Adobe Acrobat and Reader versions 2019.010.20069 and earlier, 2019.010.20069 and earlier, 2017.011.30113 and earlier version, and 2015.006.30464 and earlier have an use after free vulnerability. Successful exploitation could lead to arbitrary code execution.</td>
<td>5.0 MEDIUM</td>
</tr>
<tr>
<td>CVE-2019-7082</td>
<td>Adobe Acrobat and Reader versions 2019.010.20069 and earlier, 2019.010.20069 and earlier, 2017.011.30113 and earlier version, and 2015.006.30464 and earlier have an use after free vulnerability. Successful exploitation could lead to arbitrary code execution.</td>
<td>5.0 MEDIUM</td>
</tr>
</tbody>
</table>

Innumerable and unpredictable.
Different Approach

- APIs
- Functions
- Basic Blocks
Threat Model

- Offline document malware analysis.

- Attack techniques: exploits, API/ABI abuse.
  - Social engineering out-of-scope.

- Fixed duration dynamic analysis.
  - Mimicry? Sleeping malware? Triggering conditions?
Challenges

- ~100M BBs execute per min.
- Most activity is benign, even when attacked.
- Not all malware use exploits.
Design

- QEMU-PT Guest OS
- Target Program
- Input
- BBIDs
- Behavior Modeling
- Prediction
- Confidence
- Anomaly Detection
- wrong predictions + high confidence = anomaly

1. 2. 3.
1) Trace Collection
2) Behavior Modeling

Indirect Control Transfer

Self-Supervised
2) Behavior Model

Sequence Length

Features (32) -> Embedding (32, 128) -> relu -> LSTM (32, 128) -> relu -> Dense (128) -> relu -> Dense (1024) -> softmax

Binning

Labels (1024)

1,350,784 parameters
~5.2 MB in H5 format
3) Anomaly Detection

Behavior Model

BBIDs

- not used
- used in **correct** prediction
- used in **wrong** prediction
Data

- PDF
  - 1,250 benign
  - 1,639 malicious
- DOC
  - 200 benign
  - 379 malicious
Training Metrics

- 7:3, training:testing
- 1.5 days to train (Nvidia GTX 1080ti)
- 3 epochs (many slices per trace, repetitive)
Overhead

• PDF testing data: 3,566 minutes (real time)
• Time to classify: 342 minutes
• Overhead: ~10% (using a GPU)
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>[25]</th>
<th>[47]</th>
<th>[48]</th>
<th>Barnum</th>
</tr>
</thead>
<tbody>
<tr>
<td>TP</td>
<td>91.7%</td>
<td>93.2%</td>
<td>75.9%</td>
<td>97.8%</td>
</tr>
<tr>
<td></td>
<td>(917)</td>
<td>(6,781)</td>
<td>(243)</td>
<td>(1,600)</td>
</tr>
<tr>
<td>FN</td>
<td>8.3%</td>
<td>6.8%</td>
<td>24.1%</td>
<td>2.4%</td>
</tr>
<tr>
<td></td>
<td>(83)</td>
<td>(497)</td>
<td>(77)</td>
<td>(39)</td>
</tr>
<tr>
<td>TN</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>(994)</td>
<td>(7,278)</td>
<td>(1,030)</td>
<td>(375)</td>
</tr>
<tr>
<td>FP</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
<td>0%</td>
</tr>
<tr>
<td></td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
<td>(0)</td>
</tr>
</tbody>
</table>

Accuracy comparable to static analysis. Robustness of dynamic analysis.
Limitations

- **Maintainability:**
  - Patching software invalidates the current model.
  - Transfer learning? Modular models?

- **Not ready for online protection:**
  - Overhead was under 100%, but used a GPU.

- **Mimicry**
Code, Data, Models

https://tinyurl.com/y27clrfl