Activation Analysis of a Byte-based Deep Neural Network for Malware Classification

Scott Coull
Sr. Manager, Data Science

Christopher Gardner
Reverse Engineer
Feature engineering for malware classification tasks is hard. Can deep learning do it for us?

Convolutional neural networks (CNNs) automatically and efficiently learn feature representations directly from data.

Recent work has shown promising results competitive with (though not better than) traditional machine learning:

- Accuracy: 90-96%, AUC: 0.96-0.98
CNN Models

- **Baseline**
  - 15.6M Windows PEs (80% goodware)
  - July 2015 to July 2017
  - Stratified sampling

- **Small**
  - 7.3M Windows PEs (50% goodware)
  - July 2016 to November 2016
  - No sampling

- **Baseline+Dropout**
  - Same data as Baseline
  - Dropout layers before convolutional layers
# Model Evaluation

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Data</th>
<th>Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Size</td>
<td>Mal:Good</td>
</tr>
<tr>
<td>Small</td>
<td>7.27M</td>
<td>50:50</td>
</tr>
<tr>
<td>Baseline</td>
<td>15.62M</td>
<td>20:80</td>
</tr>
<tr>
<td>Baseline+Dropout</td>
<td>15.62M</td>
<td>20:80</td>
</tr>
</tbody>
</table>

16.55M binaries (50:50) from June 1, 2018 to August 31, 2018

Model trained on *small dataset performs noticeably better* despite older data and fewer samples
What are byte-based malware classifiers learning?

What is the impact of dataset volume and regularization on learned features?
Analysis Overview

- **Cluster and visualize** embedding layer with HDBSCAN\(^1\) and MDS\(^2\)
- **Disassembly of byte sequences** with large activations in first convolutional layer
- **End-to-end analysis** of byte segments with GradientSHAP\(^3\)

---

The analysis overview includes:

- **Raw Bytes:** 102,400 bytes
- **Embedding Layer:** (10 Dimensions)
- **Fully-Connected Layer**
- **Sigmoid**
- **Convolutional + Max Pooling Layers** x5
- **Embedding Layer** (10 Dimensions)
- **Raw Bytes**
- **P(Malware)**

---


Byte Embeddings

ASCII Characters
@, \n, A,..., j, s, t

eax-edx
padding
short jumps
Byte Embeddings

Increase in number of outliers with more data/regularization
Learned features appear to be less flexible
Low-Level Feature Detectors

Distribution of Top-100 Activations Across First-Level Filters

- Some unused filters
- Same filters, bias toward malware
Low-Level Feature Detectors

Distribution of Top-100 Activations Across First-Level Filters

More data and regularization appears to lead to more features that are equally applicable across the two classes.

Supports earlier observation about feature specificity.
## Low-Level Feature Detectors

<table>
<thead>
<tr>
<th>Model</th>
<th>Strings</th>
<th>Features</th>
<th>Instructions</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Small</strong></td>
<td><strong>Filter 71: ‘C’, ‘r’, ‘@’</strong></td>
<td><strong>Filter 16: Push sequences</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0x40f0c8L): tGenKey.</td>
<td></td>
<td>(0x10007edbL): je, 0x10007ff1</td>
</tr>
<tr>
<td></td>
<td>(0x40f0d0L): CryptDec</td>
<td></td>
<td>(0x10007ee1L): push, 0xff</td>
</tr>
<tr>
<td></td>
<td>(0x40f0d8L): rype...</td>
<td></td>
<td>(0x10007ee6L): push, edi</td>
</tr>
<tr>
<td></td>
<td>(0x40f0e0L): CryptEnc</td>
<td></td>
<td>(0x10007ee7L): push, 0x10007ca5</td>
</tr>
<tr>
<td></td>
<td>(0x40f0e8L): rype...</td>
<td></td>
<td>(0x10007eecL): push, 0x4</td>
</tr>
<tr>
<td><strong>Baseline</strong></td>
<td><strong>Filter 83: ‘r’, ‘s’</strong></td>
<td><strong>Filter 57: Function calls</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0x40d850L): ....GetP</td>
<td></td>
<td>(0x4046b4L): push, 0x0</td>
</tr>
<tr>
<td></td>
<td>(0x40d858L): rocAddre</td>
<td></td>
<td>(0x4046b6L): push, 0x0</td>
</tr>
<tr>
<td></td>
<td>(0x40d860L): ss...R.Lo</td>
<td></td>
<td>(0x4046b8L): push, 0x1</td>
</tr>
<tr>
<td></td>
<td>(0x40d868L): adLibrar</td>
<td></td>
<td>(0x4046baL): push, 0x0</td>
</tr>
<tr>
<td></td>
<td>(0x40d870L): yA...Gl</td>
<td></td>
<td>(0x4046bcL): call,dword, 15042</td>
</tr>
<tr>
<td><strong>Baseline+Dropout</strong></td>
<td><strong>Filter 11: ‘Directory’</strong></td>
<td><strong>Filter 61: mov sequences</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>(0x40d9e0L): ctoryW..</td>
<td></td>
<td>(0x408d5L): je, 0x408d6a</td>
</tr>
<tr>
<td></td>
<td>(0x40d9e8L): N.Create</td>
<td></td>
<td>(0x408d67L): mov, dword, edx</td>
</tr>
<tr>
<td></td>
<td>(0x40d9f0L): .Director</td>
<td></td>
<td>(0x408d6aL): mov, esi, dword</td>
</tr>
<tr>
<td></td>
<td>(0x40d9f8L): yW...Ge</td>
<td></td>
<td>(0x408d6dL): mov, dword, esi</td>
</tr>
<tr>
<td></td>
<td>(0x40da00L): tTempPat</td>
<td></td>
<td>(0x408d70L): mov, ecx, dword</td>
</tr>
</tbody>
</table>
End-to-End Features

SHAP Values for WannaCry Worm

- Baseline
- Missing standard directories (no certificate, no exceptions)
- Checksum Set to 0
- Contains Resource Directory
- Export and Import Table for Embedded PE File

End-to-end features map closely to manual feature engineering
End-to-End Features

SHAP Values for WannaCry Worm

Data and regularization result in more focused areas of interest
Model appears to learn presence/absence of structural features
The Case of the Rich Header

- Rich header is added by Microsoft’s linker and contains metadata about the binary
- Should be effectively ‘random’ due to XOR encryption using key derived from checksum
- **Hypothesis:** Hierarchical pooling can detect presence of fixed bytes around header (e.g., ‘Rich’)
- Proxy for whether **non-Microsoft compiler** was used, which is **common in malware**
Summary

- CNN architectures can learn **meaningful features**
  - Imports, presence of Rich header, incorrect checksums, etc.
  - Many features mimic manually-derived features from traditional ML models
  - Partly contradicts findings by Demetrio et al. on MalConv

- **Model depth**, **dataset**, and **hierarchical pooling** appear to be key

- Malware classification performance relies on **detecting malware indicators**
  - Increased data and regularization lead to more specific features that were equally applicable across the two classes but worse detection performance

FireEye Data Science is Hiring!

- Data scientist positions open at the Senior, Staff, and Principal level
- Perform cutting-edge ML research and apply it to cybersecurity problems
- Work on problems from across the entire cybersecurity spectrum!
  - Threat Intelligence, Email, Network, Endpoint …

Thank you!

- scott.coull@fireeye.com
- @DrScottCoull
- https://scottcoull.com