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

Scott Coull, Principal Data Scientist

With Christopher Gardner, Reverse Engineer

The Rise of Byte-based Malware Classifiers

- 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


FireEye SECURITY REIMAGINED

Representation Learning for Malware Classification

Jeffrey Johns

Copyright © 2017, FireEye, In

Workshop track - ICLR 2018

DEEP CONVOLUTIONAL MALWARE CLASSIFIERS CAN LEARN FROM RAW EXECUTABLES AND LABELS ONLY

Marek Krčál, Czech Academy of Sciences, krcal@cs.cas.cz
Ondřej Švec, Otakar Jašek, Avast Interns, {ond.svec,jasek.ota}@gmail.com
Martin Bálek, Avast, balek@avast.com

Malware Detection by Eating a Whole EXE

Edward Raff[1,3,4], Jon Barker[2], Jared Sylvester[1,3], Robert Brandon[1,3,4]
Bryan Catanzaro[2], Charles Nicholas[4]
[1]Laboratory for Physical Sciences, [2]NVIDIA, [3]Booz Allen Hamilton, [4]University of Maryland, Baltimore County
{edraff,jared,rbrandon}@lps.umd.edu, {jbarker,bcatanzaro}@nvidia.com, nicholas@umbc.edu

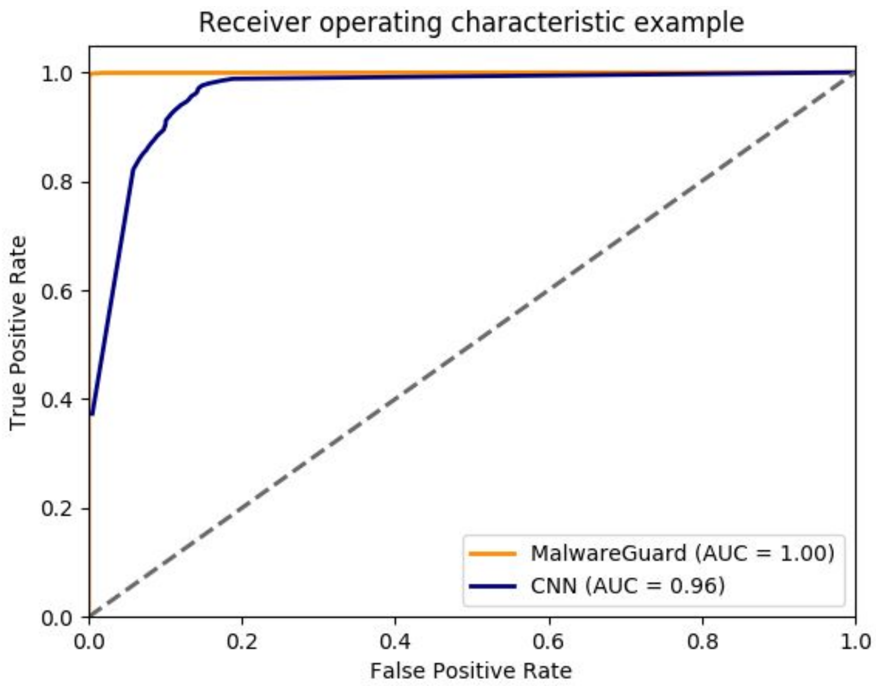
25 Oct 2017

Abstract

In this work we introduce malware detection from raw byte sequences as a fruitful research area to the larger machine learning community. Building a neural network for such a problem presents a number of interesting challenges that have not occurred in tasks such as image processing or NLP. In particular, we note that detection from raw bytes presents a sequence problem with over two million time steps and a problem where

inside a specially instrumented environment, such as a customized Virtual Machine (VM), which introduces high computational requirements. Furthermore, in some cases it is possible for malware to detect when it is being analyzed. When the malware detects an attempt to analyze it, the malware can alter its behavior, allowing it to avoid discovery (Raffetseder, Kruegel, and Kirda 2007; Garfinkel et al. 2007; Carpenter, Liston, and Skoudis 2007). Even when malware

Traditional Classifiers vs. Deep Learning



Results on 16 million PE files from June 1 to August 31, 2018

- Clearly still a large gap between handcrafted features in MalwareGuard and the CNN
- CNN performance is surprising given the level of indirection and variability of syntax/semantics found in Windows PE files
- **How is it doing so well with so little information?**

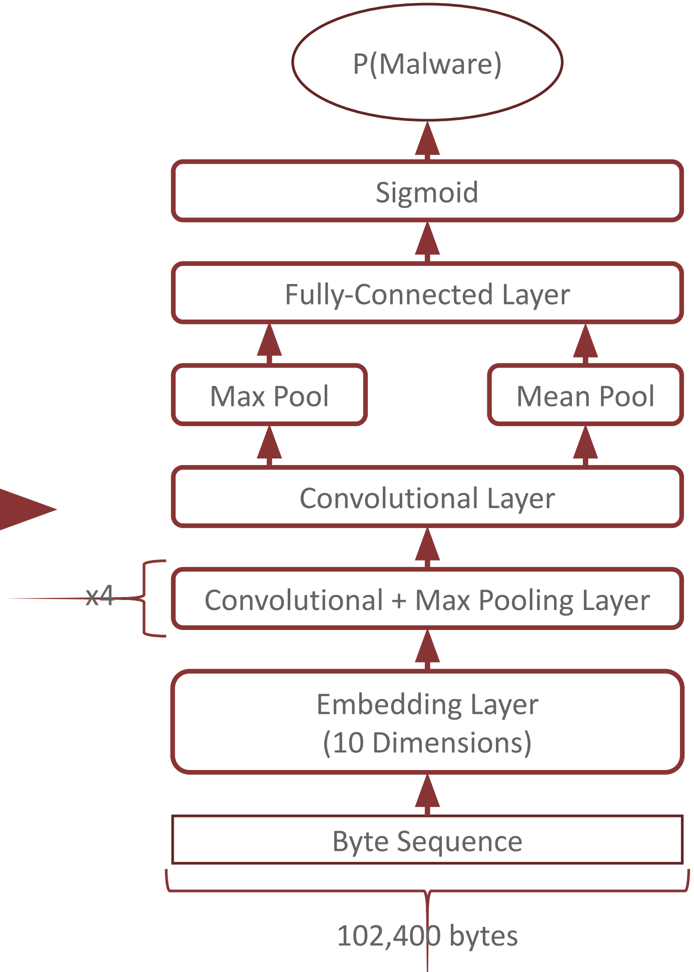
Understanding Byte-based Malware Classifiers

- Predictions from deep learning models are notoriously difficult to interpret even under ideal conditions
- Malware classification on raw binaries makes it even more difficult due to the **semantic gap** between the **byte representation and the disassembled code** that analysts examine
- **Objective: Develop methodologies for understanding what byte-based malware classifiers are learning**

FireEye CNN Malware Classifier

Training

- 15+ million PE files
- 20% Malware, 80% Goodware
- Epochs: 10
- Batch Size: 80



Analysis Overview

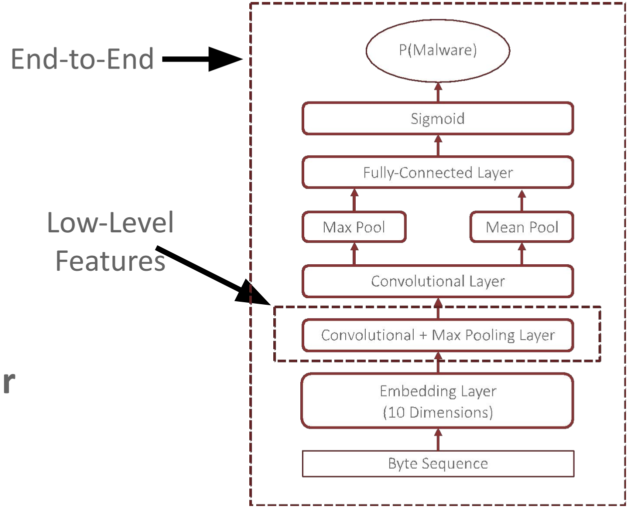
- Broad Trends
  - Gather general information about the locations of interest in goodware and malware
  - Examine locations and strengths using both **low-level feature detectors** and **end-to-end analysis**
- Deep Analysis
  - Dive into specific ransomware samples to provide concrete examples of what features are learned
  - Examine trends in **embedding layer topology**, **byte sequences for frequently-activated filters**, and **contiguous segments that push classification toward malware/goodware labels**
- Deep Learning vs. Reverse Engineer
  - Understand the overlap between **analyst intuition** and **areas of interest identified by the model**

Broad Trends

Analyzing activations across a large dataset

Overview
Broad Trends

- Data
  - 4,000 PE files (50/50 split)
  - Random sample from dataset of 15M binaries

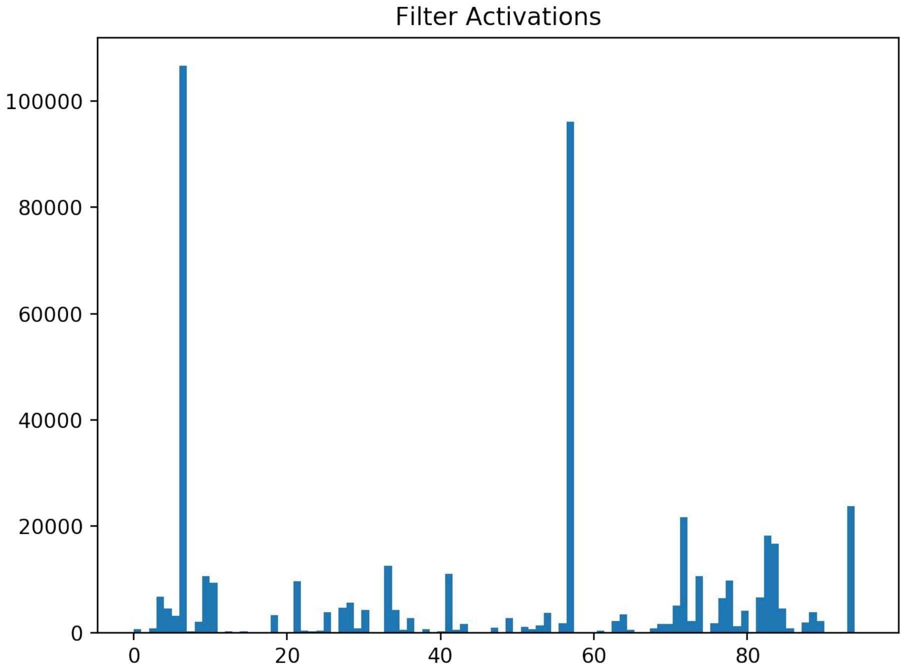


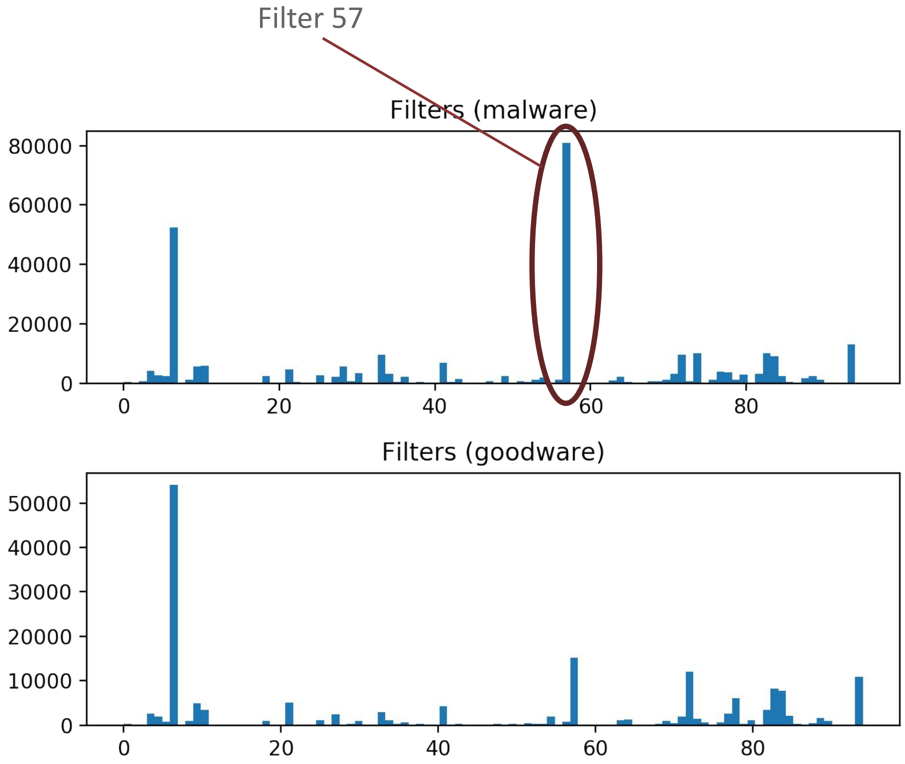
- Analysis Methodology
  - Locations and weight of activations in **first convolutional layer**
  - Comparison using end-to-end analysis with **GradientSHAP**[1]
  - **Differences** between goodware and malware

[1] Lundberg, Scott M., and Su-In Lee. *A unified approach to interpreting model predictions.*

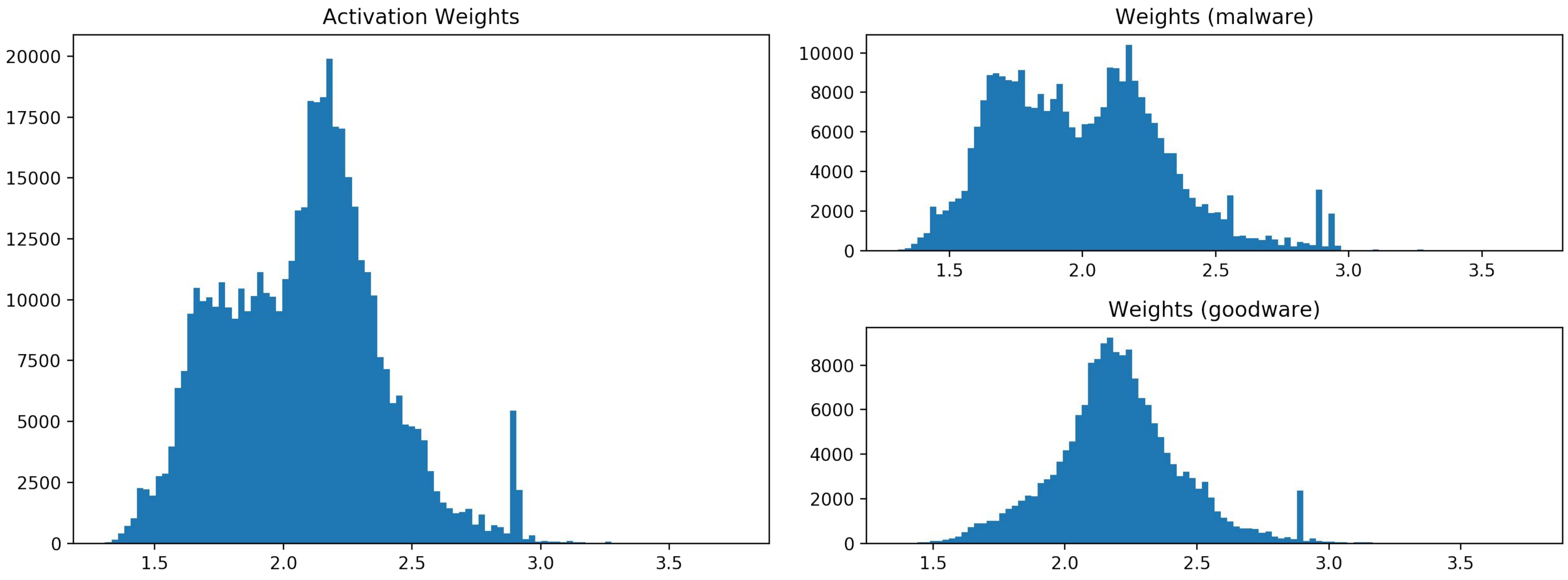
Low-level Feature Detectors

Broad Trends

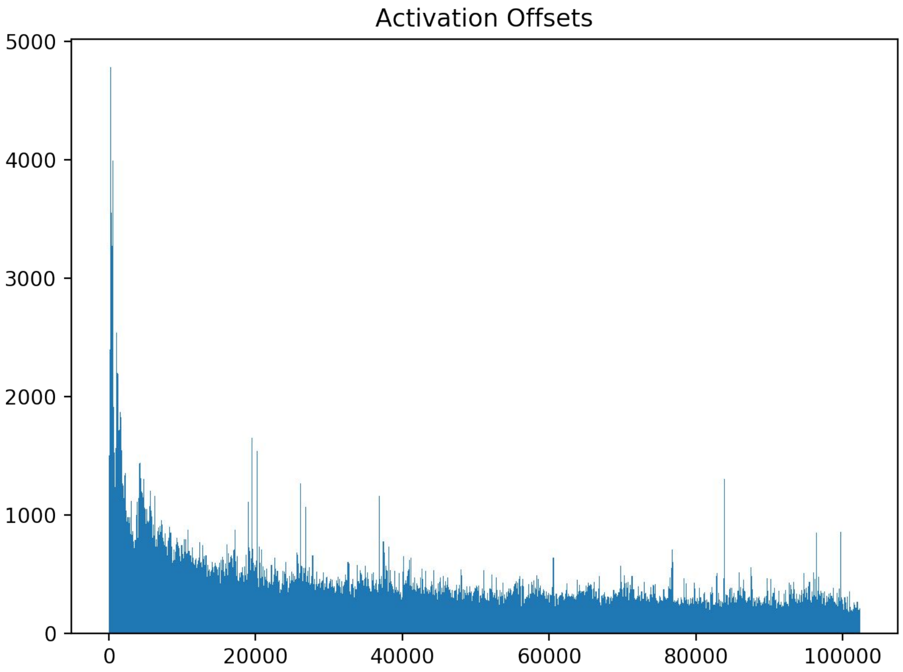


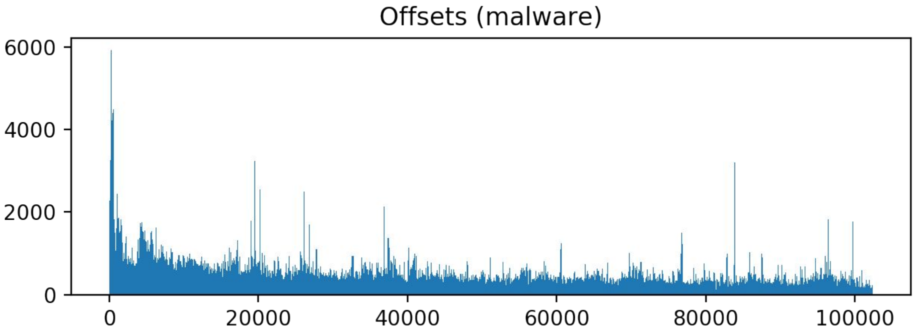


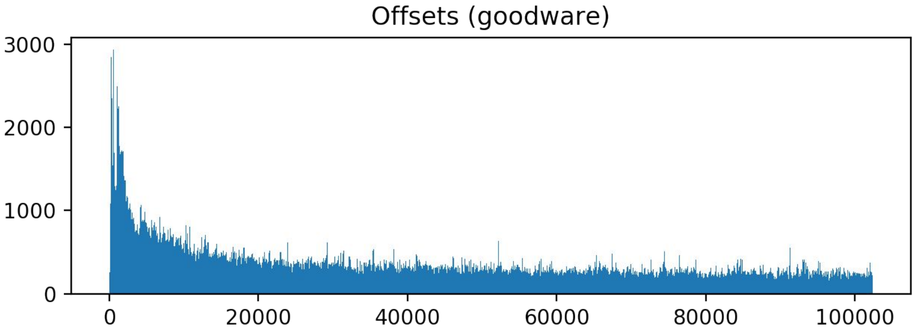
Low-level Feature Detectors
Broad Trends



Low-level Feature Detectors
Broad Trends

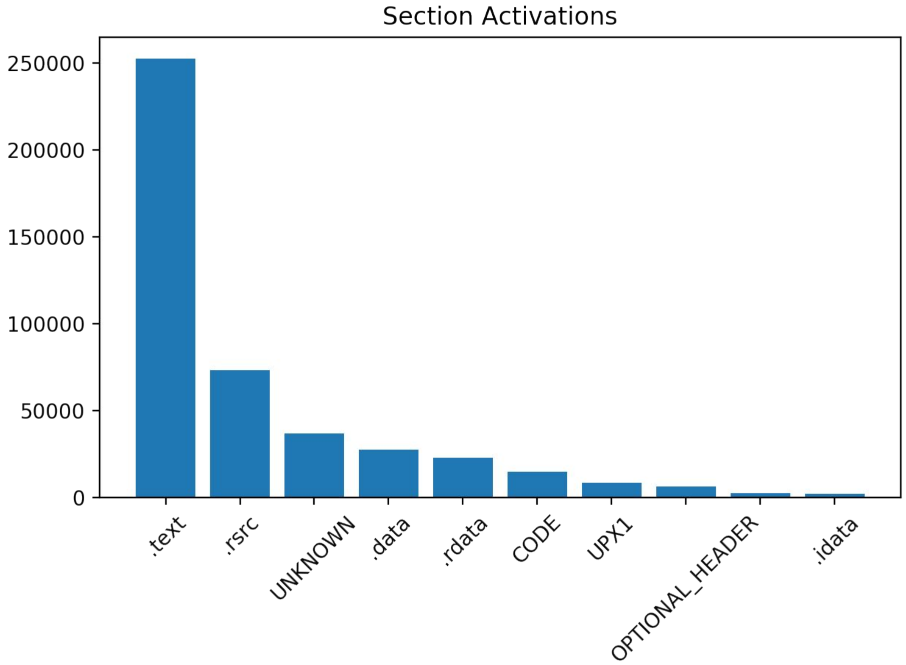


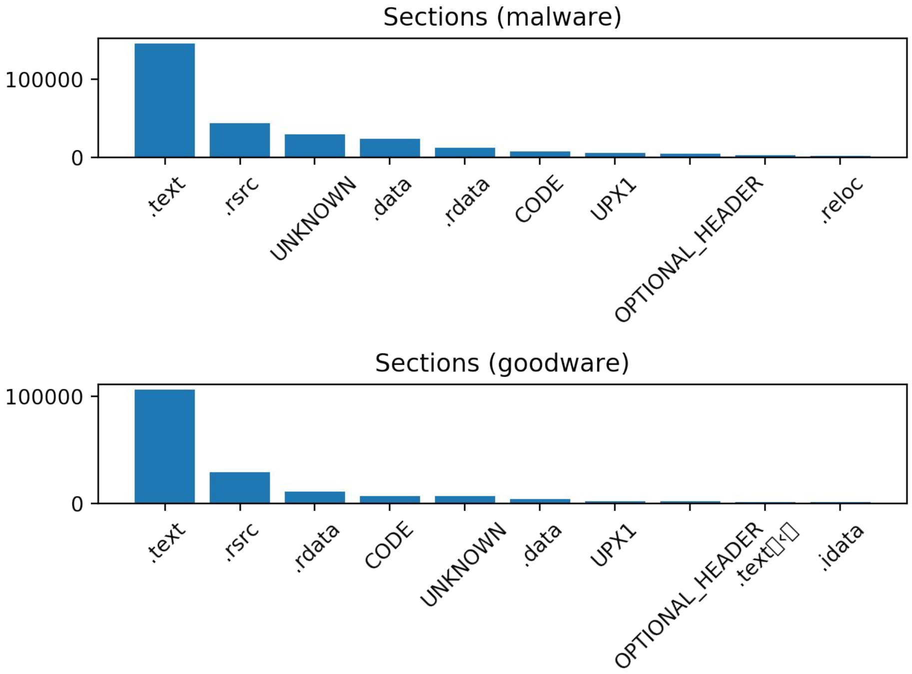




Low-level Feature Detectors

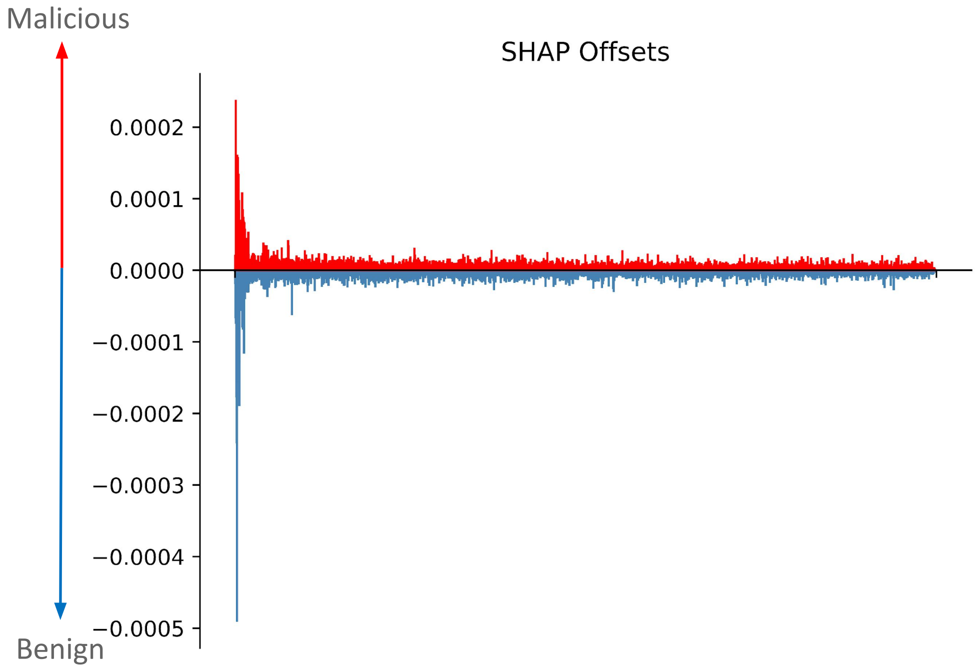
Broad Trends

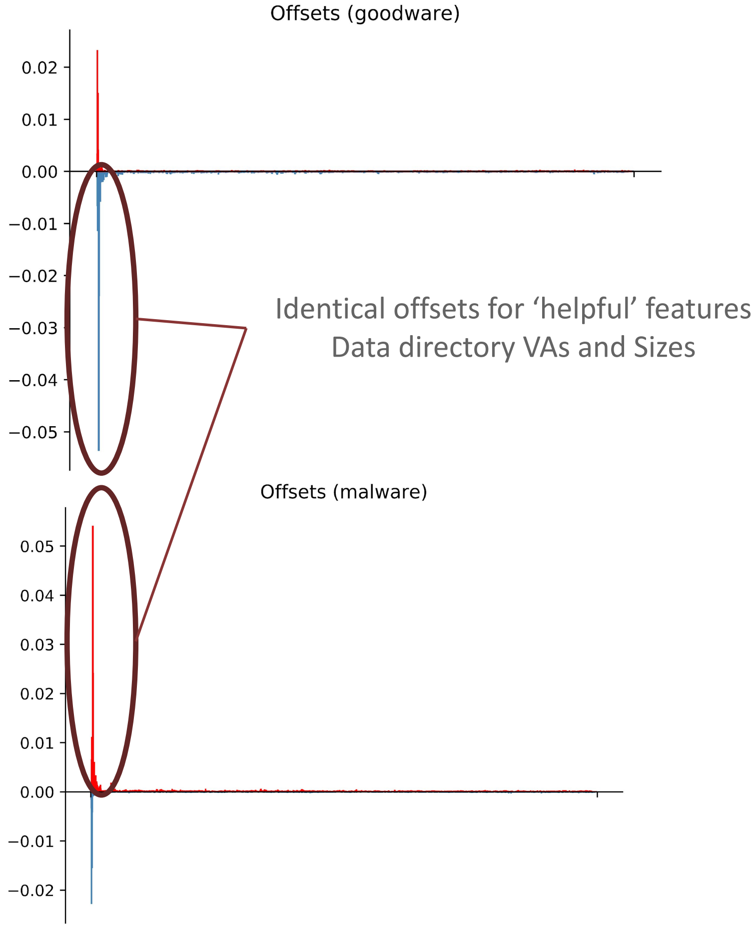




End-to-End Analysis

Broad Trends





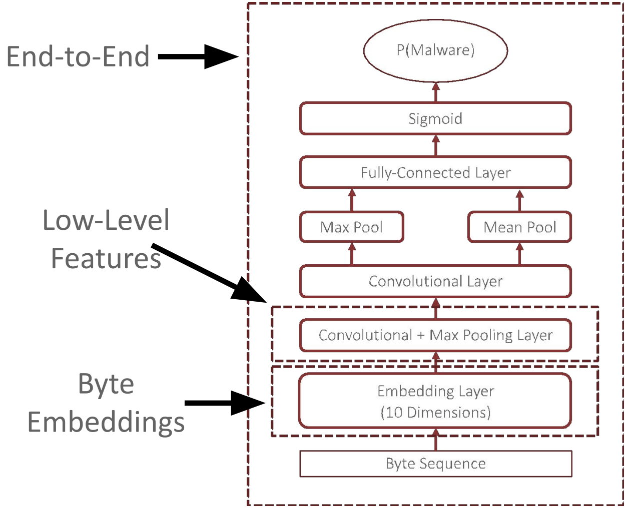
Deep Analysis

Studying interesting features in ransomware

Overview

Deep Analysis

- Data
  - 6 ransomware artifacts (loader, payload, encryptor)
  - NotPetya, WannaCry, BadRabbit

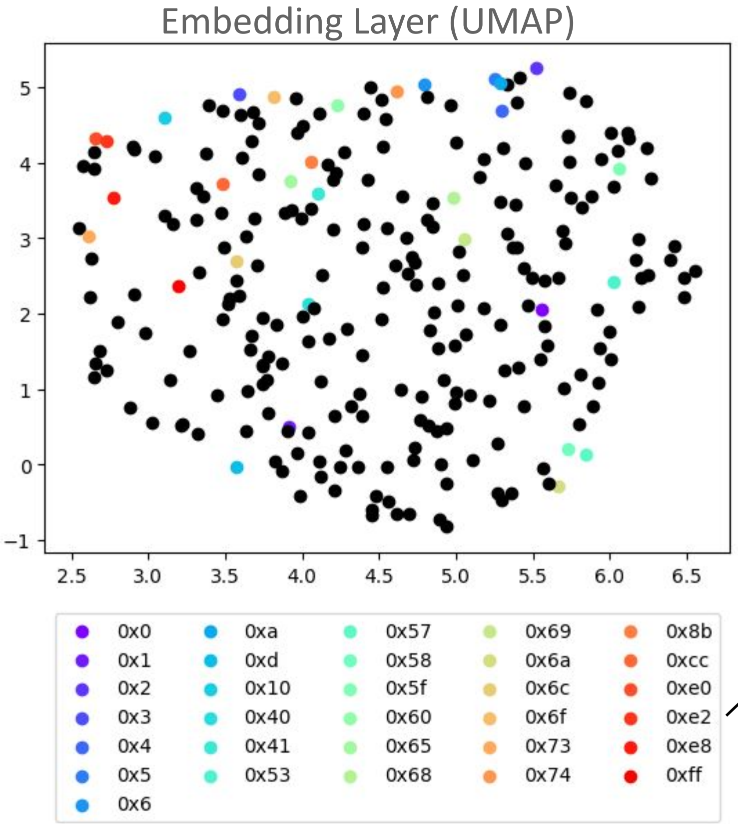


- Methodology
  - **Cluster** and **visualize** of embedding space with HDBSCAN[2] and UMAP[3]
  - Examine **semantics** of recurring activations within first-layer convolutional filters using BinaryNinja disassembly
  - End-to-end analysis of **contribution of specific byte segments** to malware/goodware decision

[2] McInnes, L, Healy, J, *UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction.*
[3] Campello, Ricardo JGB, Davoud Moulavi, and Jörg Sander. *Density-based clustering based on hierarchical density estimates.*

Byte Embedding Outliers

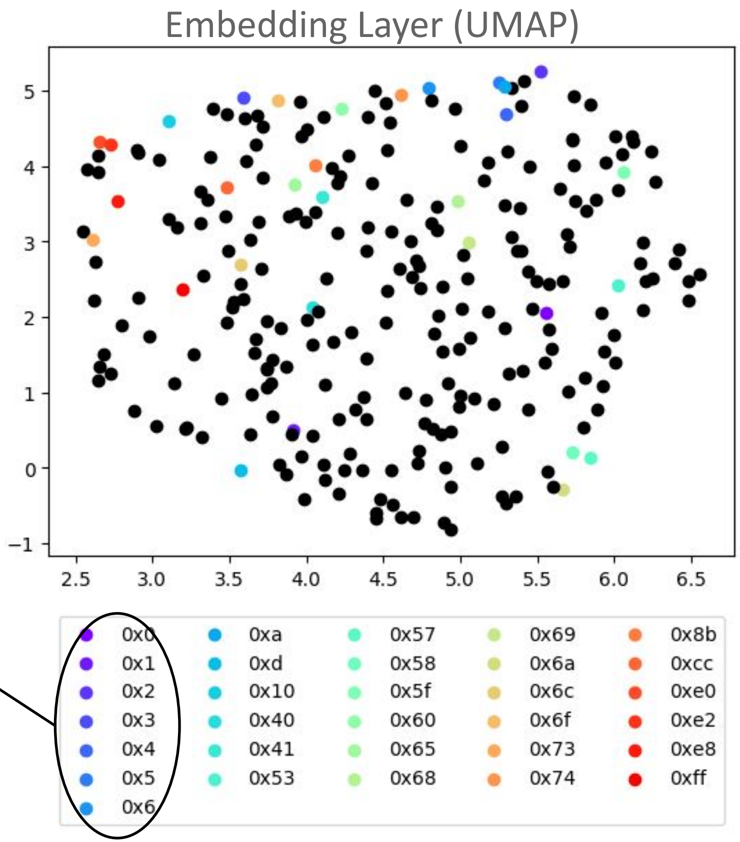
Deep Analysis



Outliers from HDBSCAN indicate bytes the model has learned are unique or "special"

Byte Embedding Outliers
Deep Analysis

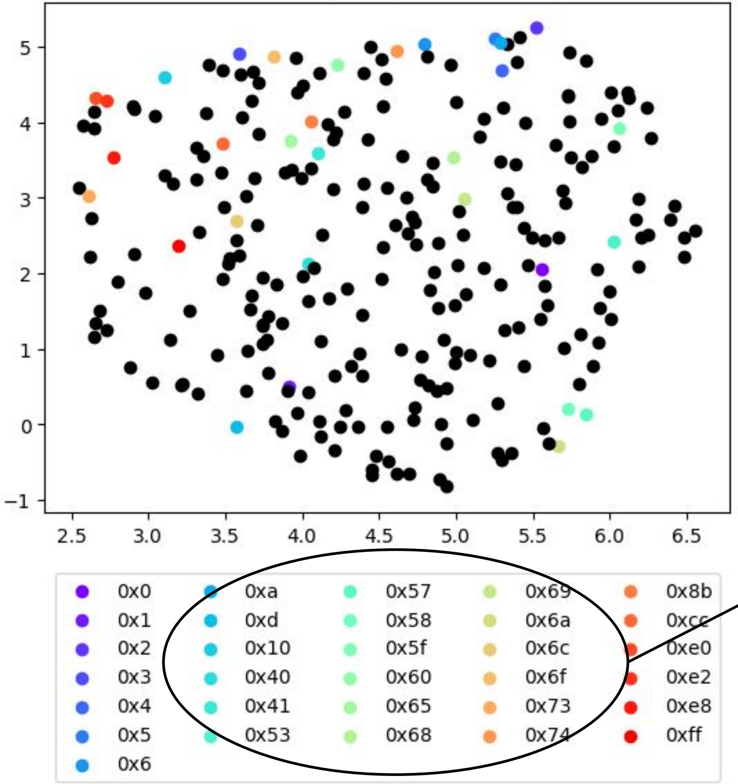
- eax-edx
- nop
- Short Jumps



Byte Embedding Outliers

Deep Analysis

Embedding Layer (UMAP)

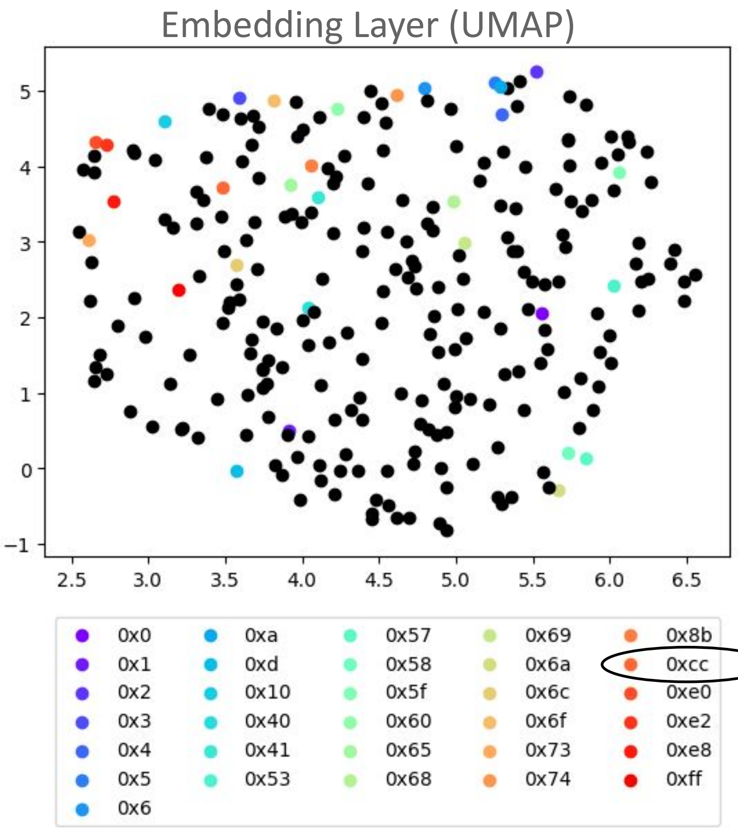


ASCII Characters

- @, \n
- A, W, X
- e, l, j, s, t

Byte Embedding Outliers

Deep Analysis



Anti-Debugging Instruction

Low-level Feature Detectors

Deep Analysis

- Filter 83:
  - Import table activations
  - Lots of process related function
  - Some conditional jumps

```
71768 (0x10012458L):      72 6d 69 6e 61 74 65 50     rminateP
**71776 (0x10012460L):    72 6f 63 65 73 73 00 00     rocess..
**71784 (0x10012468L):    03 02 47 65 74 4c 6f 63     ..GetLoc
71792 (0x10012470L):      61 6c 54 69 6d 65 00 00     alTime..
```

```
71992 (0x10012538L):      74 43 6f 64 65 50 72 6f     tCodePro
**72000 (0x10012540L):    63 65 73 73 00 00 15 02     cess....
**72008 (0x10012548L):    47 65 74 4d 6f 64 75 6c     GetModul
72016 (0x10012550L):      65 48 61 6e 64 6c 65 41     eHandleA
```

```
43024 (0x40a810L):    74 68 72 65 61 64 65 78     threadex
**43032 (0x40a818L):      00 00 c1 02 73 74 72 6e     ....strn
**43040 (0x40a820L):      63 70 79 00 a6 02 72 61     cpy...ra
**43048 (0x40a828L):      6e 64 00 00 a6 00 5f 62     nd...._b
43056 (0x40a830L):    65 67 69 6e 74 68 72 65     eginthre
```

Low-level Feature Detectors
Deep Analysis

- Filter 82:
  - Import table, and some offset tables·
  - File IO functions, memory allocation and time functions

```
50644 (0x1000d1d4L):      int32_t (* KERNEL32!SetFilePointerEx@IAT)() = 0x12110
**50648 (0x1000d1d8L):    int32_t (* KERNEL32!SetEndOfFile@IAT)() = 0x12100
**50652 (0x1000d1dcL):    int32_t (* KERNEL32!GetDriveTypeW@IAT)() = 0x120f0
**50656 (0x1000d1e0L):    int32_t (* KERNEL32!UnmapViewOfFile@IAT)() = 0x120de
**50660 (0x1000d1e4L):    int32_t (* KERNEL32!MapViewOfFile@IAT)() = 0x120ce
50664 (0x1000d1e8L):      int32_t (* KERNEL32!FindFirstFileW@IAT)() = 0x120bc
```

Low-level Feature Detectors

Deep Analysis

- Filter 94 catches parts of EternalRomance exploit code from WannaCry worm

```
10975 (0x100036DFL):      cmp   [ebp+buf], ecx
**10978 (0x100036E2L):    jnz   loc_10003B39
**10984 (0x100036E8L):    push  4
**10986 (0x100036EAL):    xor   eax, eax
**10988 (0x100036ECL):    mov   [edi+1], ax
10992 (0x100036F0L):      pop   eax
```

Low-level Feature Detectors

Deep Analysis

- Remember the strong malware filter?

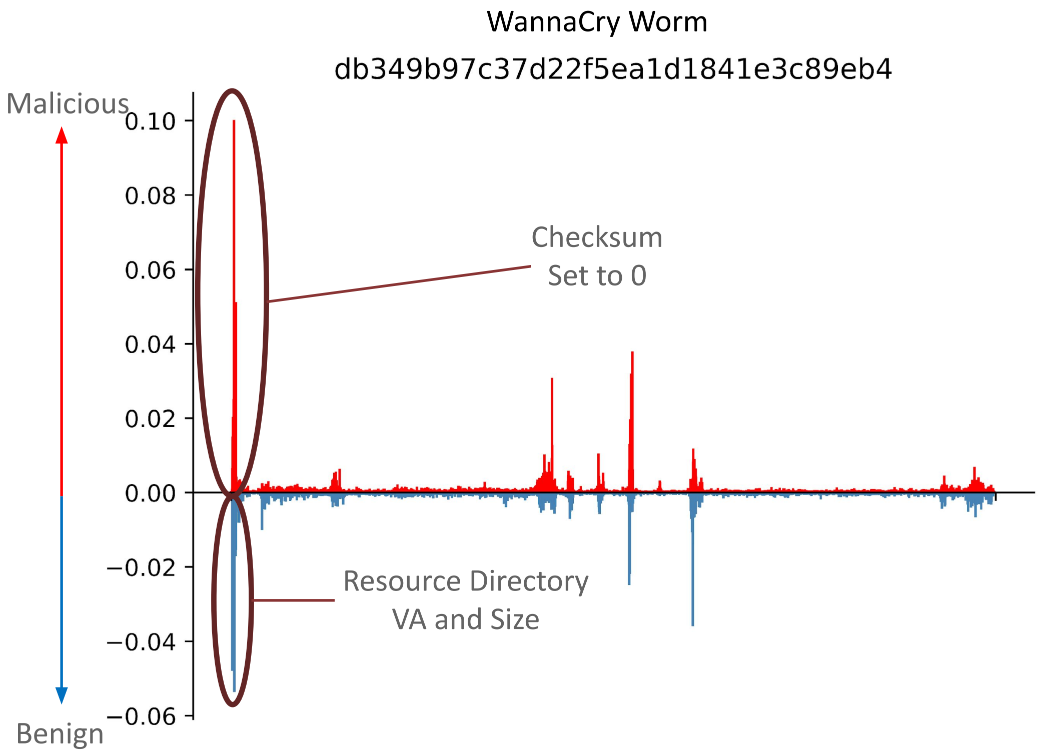
```
32953 (0x4080b9L):    push    edi
**32954 (0x4080baL):          push    0xf003f
**32959 (0x4080bfL): push    0x0
**32961 (0x4080c1L):push    0x0
**32963 (0x4080c3L):call    dword [ADVAPI32!OpenSCManagerA@IAT]
32969 (0x4080c9L):    mov     edi, eax
```

- Filter 57:
  - Push/call sequences
  - Callings functions with (lots of) arguments (like Windows APIs)
  - In-lined memcpy() implementation

```
24652 (0x10006c4cL): push    0x0
**24654 (0x10006c4eL):       call    ebx
**24656 (0x10006c50L):       push    eax
**24657 (0x10006c51L):       call    edi
**24659 (0x10006c53L):
**24659 (0x10006c53L):       push    esi
**24660 (0x10006c54L):       push    0x0
**24662 (0x10006c56L):       call    ebx
**24664 (0x10006c58L):       push    eax
**24665 (0x10006c59L):       call    edi
**24667 (0x10006c5bL):       pop     edi
24668 (0x10006c5cL):         pop     ebx
```

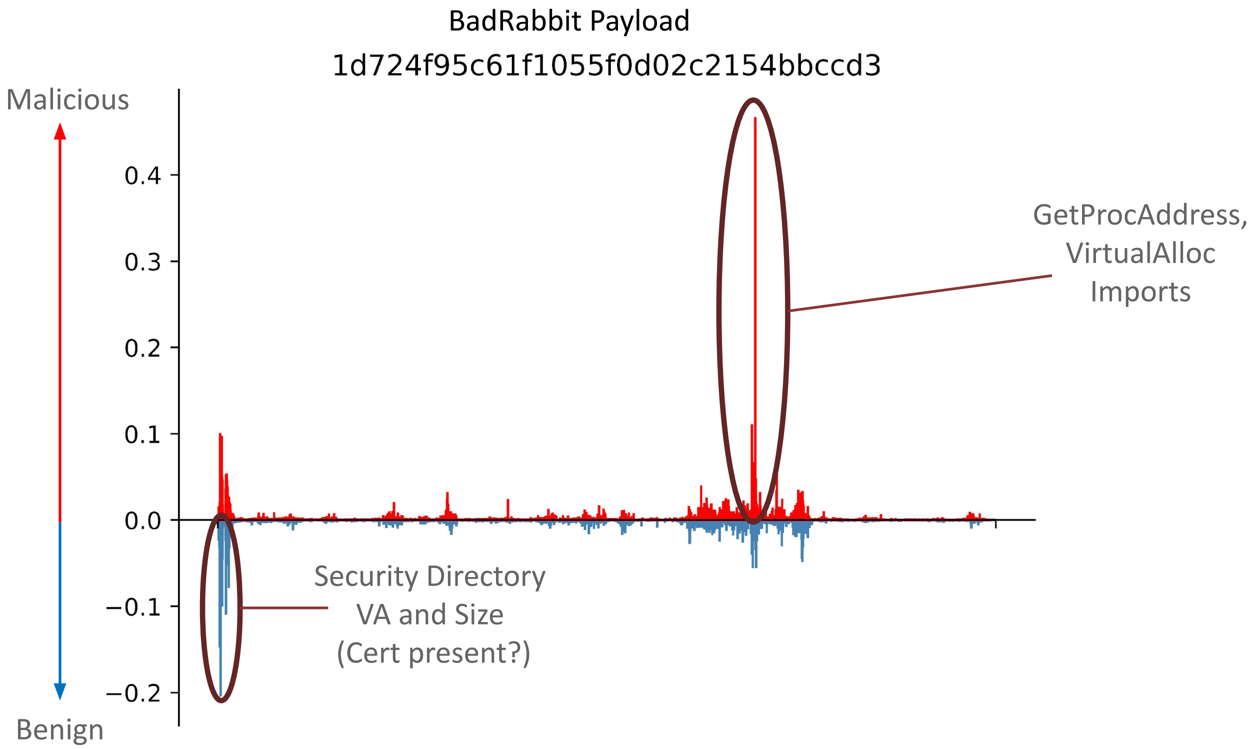
End-to-End Analysis

Deep Analysis



End-to-End Analysis

Deep Analysis



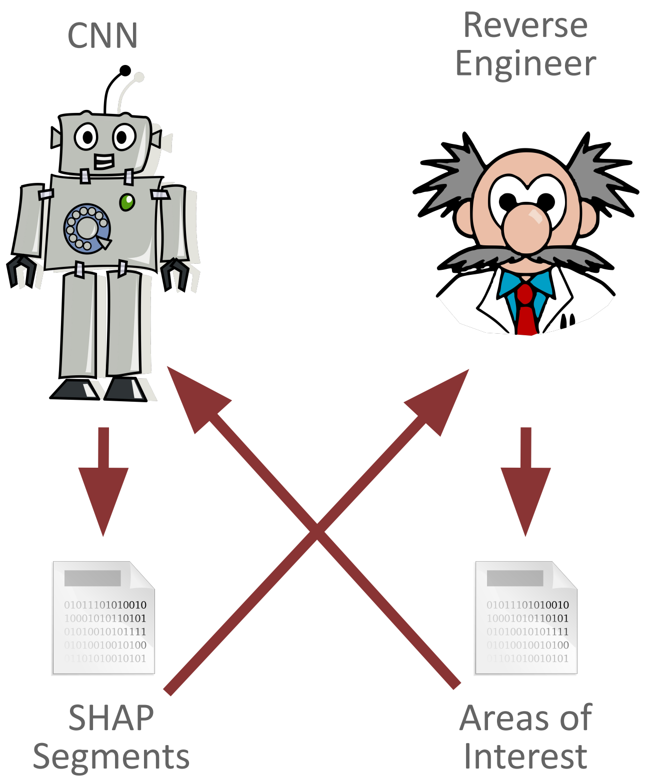
Deep Learning vs. Reverse Engineer

The gap between model and human understanding

Overview

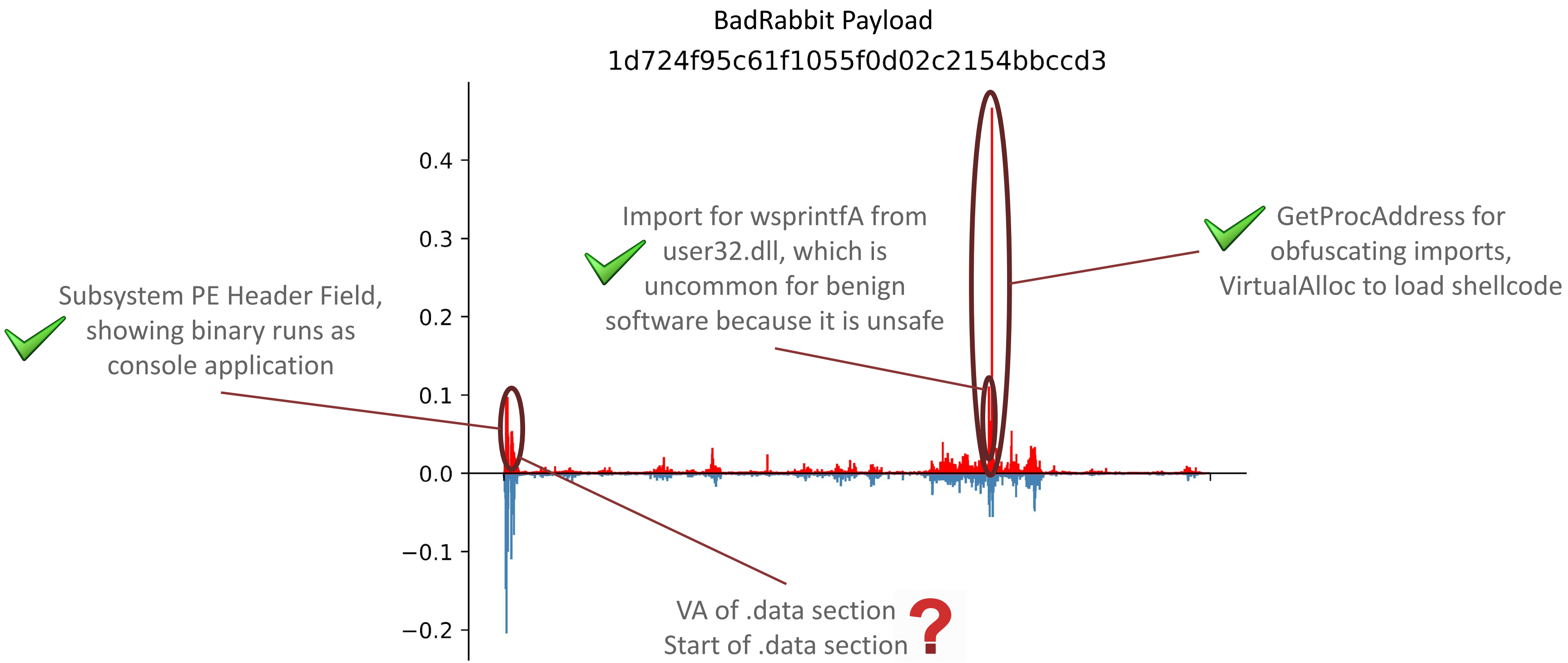
Deep Learning vs. Reverse Engineer

- Data
  - Same 6 ransomware samples from previous analysis
- Methodology
  - RE produces list of file offsets containing interesting indicators of maliciousness, called **areas of interest (AOI)**
  - CNN produces **top-100 convolutional layer activations** and **malicious SHAP segments**
  - Examine overlap between CNN activations/segments and analyst AOIs, as well as analyst feedback on CNN segments



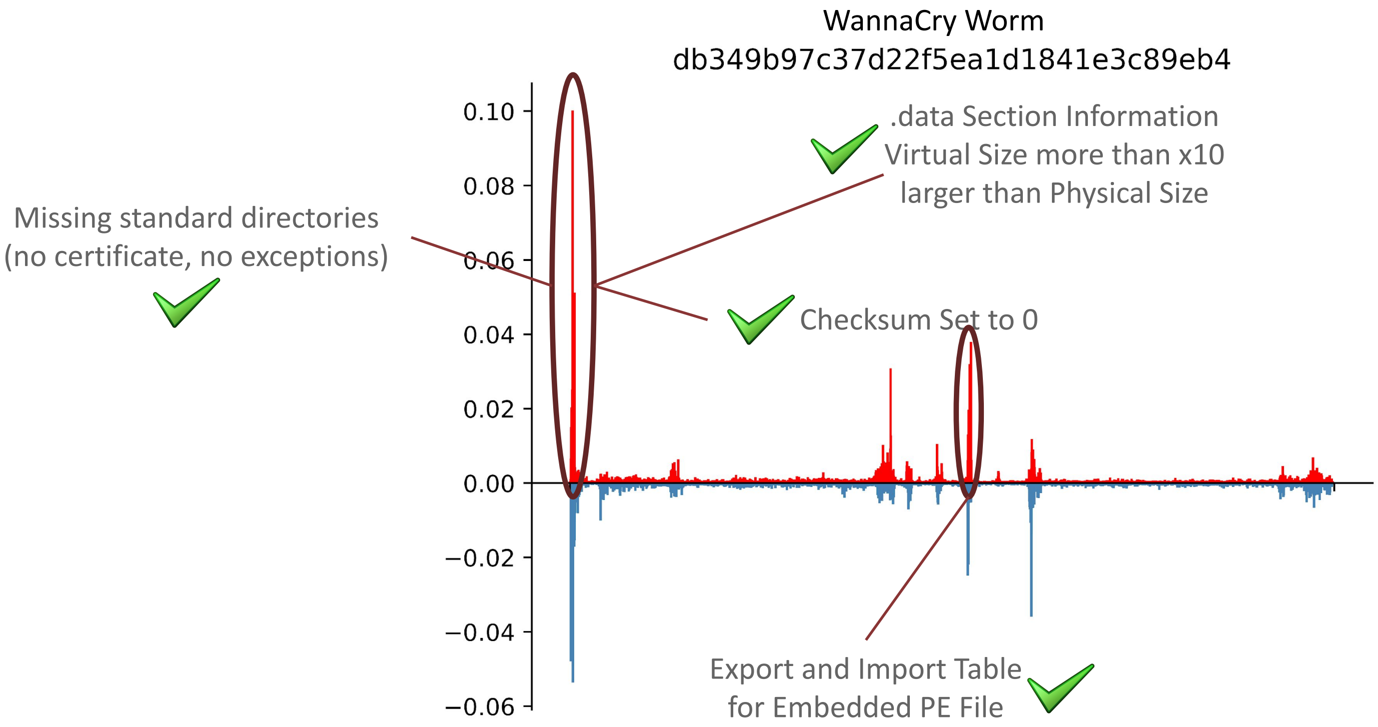
Human Feedback on Model Malicious Segments

Deep Learning vs. Reverse Engineer



Human Feedback on Model Malicious Segments

Deep Learning vs. Reverse Engineer



Model Overlap with Human AOIs

Deep Learning vs. Reverse Engineer

Percentage of Analyst AOIs with Overlap

	BadRabbit Payload	WannaCry Worm	NotPetya Payload	Overall
1st Conv. Layer	23.5%	20.0%	25.0%	**30.0%**
SHAP Segments	11.7%	0.0%	0.0%	**10.8%**

- Creating a scheduled task
- Strange stack strings
- Creating a process as another user

Non-trivial overlap given that we only examined top-100 activations/segments

Summary of Findings
Part 1

- Implicit and explicit import features play a large role in CNN model decisions
  - Artifacts of imports are observed in embedding, convolutional, and end-to-end analysis results
  - These can be easily manipulated and previous work has demonstrated adversarial attacks here

- Interesting code features are observed at lower layers but do not translate to end-to-end importance
  - Exploit code from EternalRomance and push-call sequences learned by convolutional filters
  - Top SHAP segments made up primarily of PE header and import features

Summary of Findings
Part 2

- Many important end-to-end features map closely to common manually-derived features
  - Incorrect checksums and implicit indicators for presence of certificate
- Significant amount of overlap with analyst's areas of interest
  - Convolutional filter activations were much more strongly related to analyst AOIs
- Highly-ranked end-to-end features considered generally useful indicators by analyst
  - Most focused on imports or implicit indicators of non-standard PE file structure

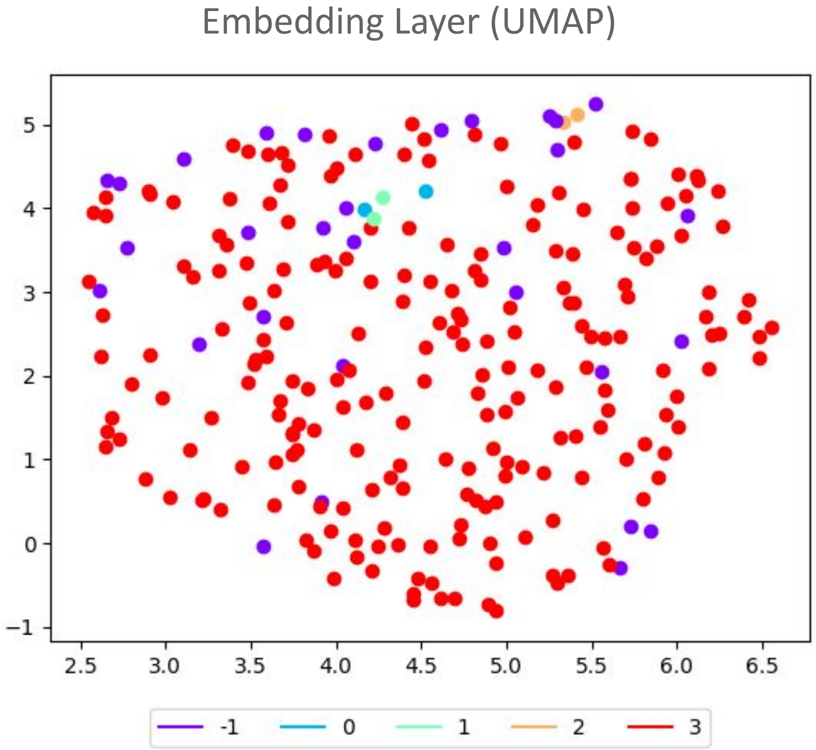
FireEye Data Science is Hiring!

- Come work on interesting data science problems across the entire cyber security spectrum!
  - Threat Intelligence, Email, Endpoint, Network, Cloud, …
- Data scientist positions open at the Senior, Staff, and Principal level
- Software and data engineering positions open at the Staff level

Bonus Material

Byte Embedding Clusters

Deep Analysis



Low-level Feature Detectors

Deep Analysis

- Filter 34:
  - Crypto-related imports
  - Process management
  - ZLIB code snippets

```
72584 (0x10012788L):      61 00 b5 00 43 72 79 70    a...Cryp
**72592 (0x10012790L):    74 44 65 72 69 76 65 4b    tDeriveK
**72600 (0x10012798L):    65 79 00 00 cd 00 43 72    ey....Cr
72608 (0x100127a0L):      79 70 74 53 65 74 4b 65    yptSetKe
```

```
42664 (0x40a6a8L):    72 41 00 00 44 02 53 65    rA..D.Se
**42672 (0x40a6b0L):  74 53 65 72 76 69 63 65    tService
**42680 (0x40a6b8L):  53 74 61 74 75 73 00 00    Status..
**42688 (0x40a6c0L):  34 00 43 68 61 6e 67 65    4.Change
42696 (0x40a6c8L):    53 65 72 76 69 63 65 43    ServiceC
```