Cyber Attack Attribution using Malware Artifacts

University of Houston

Team Members

Michael Tran
mhtran4@uh.edu

Alec Davila
andavil2l@uh.edu

Kevin Chen
ckchen2@uh.edu

Tu Van
tnnguyen66@uh.edu

Attribution

Definition: The action of regarding something as being caused by a person or thing.

Applications:

- Authorship attribution in disputed works
- Plagiarism detection
- Author profiling
- Stylistic inconsistency detection in collaborative work
- Malware attribution

Project Overview

- Attribute malicious PE executables to their respective malware families
- Assumptions:
  - Non-generic malware families are written by a single author or group of authors
  - Infeasible to derive “real” ground truth without claim of ownership
  - Family attribution is best proxy for ground truth
- Commonly cited features of malware for attribution analysis:
  - Static features - PE header information, printable strings, opcodes, etc.
  - Dynamic features - network traffic, file system changes, API calls, etc.
  - Hybrid analysis - static and dynamic combined
- Newer approaches
  - Malware visualization

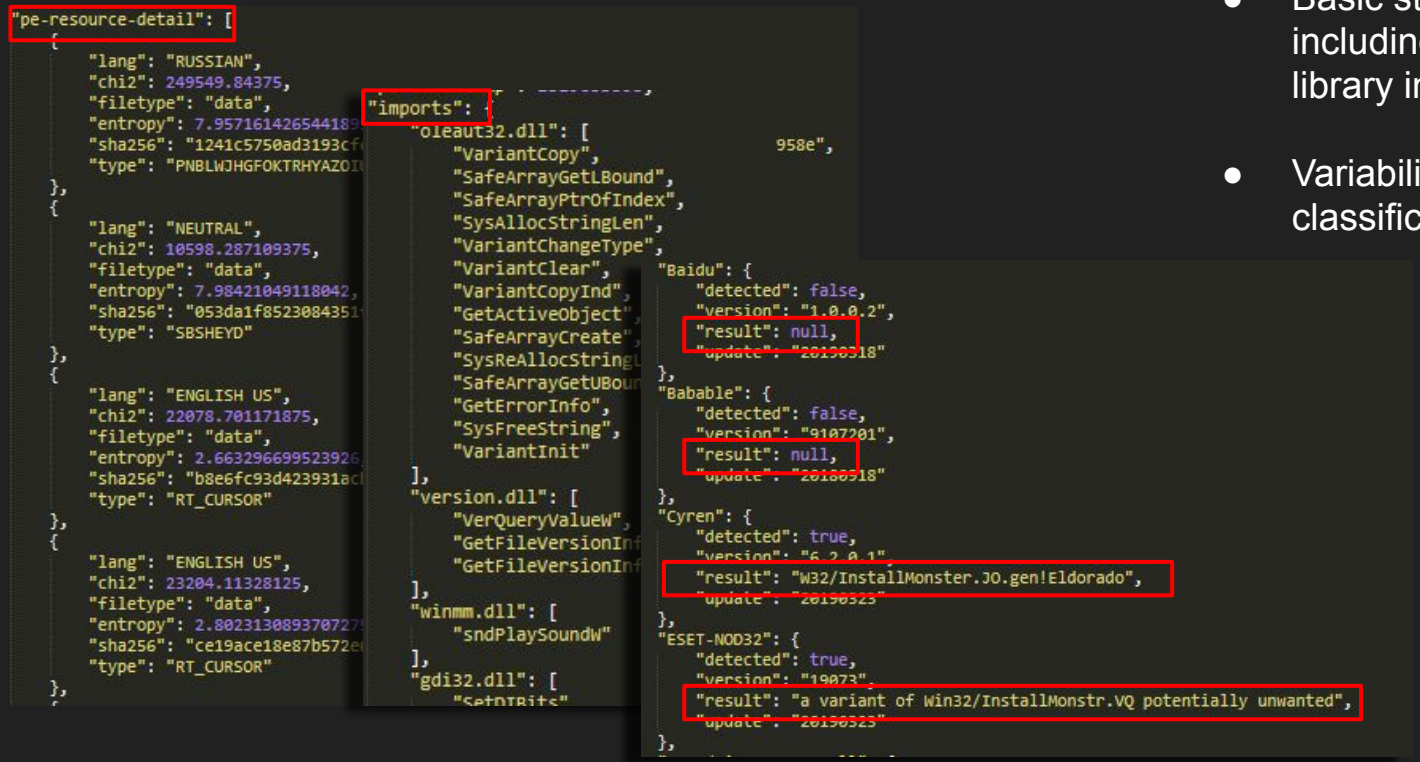
Static Features - VirusTotal

- One year’s worth of samples spanning Feb 2018 - Feb 2019 collected using VirusShare
- After pruning for families with at least 20 samples, we have 95,000 total reports and 248 families
- VirusTotal reports returned on Windows executables uses a combination of different tools such as ExifTool, sigcheck, TrID, etc.

Dynamic Features - Cuckoo

- After pruning for families with at least 50 samples, we further sampled 50 random samples from each of these families
- 20 samples x 124 families = 2,480 samples
- 2 minutes per sample
- Many more features than VirusTotal including API calls, network traffic information, manipulated registries, behavioral descriptions, etc.

Sample VirusTotal Report



- Basic static features including PE resource, library imports, etc.
- Variability in scanner classification

Sample Cuckoo Report


```
{
    "markcount": 4,
    "families": [],
    "description": "Steals private information from local Internet browsers",
    "severity": 2,
    "marks": [
        {
            "category": "file",
            "ioc": "C:\\Users\\sandy\\AppData\\Local\\Google\\Chrome\\User Data\\Default\\Login Data",
            "type": "ioc",
            "description": null
        },
        {
            "category": "file",
            "ioc": "C:\\Users\\sandy\\AppData\\Local\\Chromium\\User Data\\Default\\Login Data",
            "type": "ioc",
            "description": null
        },
```

```
"process_name": "rkverify.exe",
"command_line": "\"C:\\Users\\sandy\\AppData\\Local\\Temp\\is-69KPR.tmp\\rkverify.exe\" ",
"modules": [
    {
        "basename": "rkverify.exe",
        "imgsize": 385024,
        "baseaddr": "0x400000",
        "filepath": "C:\\Users\\sandy\\AppData\\Local\\Temp\\is-69KPR.tmp\\rkverify.exe"
    },
    {
        "basename": "ntdll.dll",
        "imgsize": 1572864,
        "baseaddr": "0x76f80000",
        "filepath": "C:\\Windows\\SysWOW64\\ntdll.dll"
```

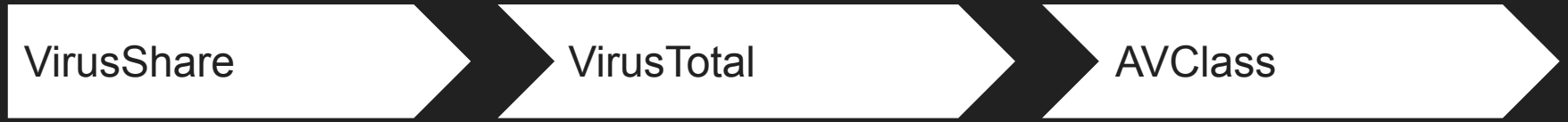

- Example of Cuckoo event descriptions and non-system process information
- JSON formatting convenient for data extraction
- ~ 40MB of data per file with a much larger variation in size across samples

Sample Cuckoo Report


```
{
    "count": 1,
    "body": "",
    "uri": "http://post.securestudies.com/packages/VR/PackageV.exe",
    "user-agent": "InnoTools_Downloader",
    "method": "GET",
    "host": "post.securestudies.com",
    "version": "1.0",
    "path": "/packages/VR/PackageV.exe",
    "data": "GET /packages/VR/PackageV.exe HTTP/1.0\r\nHost: post.securestudies.com\r\nUser-Agent: InnoTools_Downloader\r\n\r\n",
    "port": 80
},
{
    "count": 1,
    "body": "",
    "uri": "http://post.securestudies.com/packages/IR/PackageI2.exe",
    "user-agent": "InnoTools_Downloader",
    "method": "GET",
    "host": "post.securestudies.com",
    "version": "1.0",
    "path": "/packages/IR/PackageI2.exe",
    "data": "GET /packages/IR/PackageI2.exe HTTP/1.0\r\nHost: post.securestudies.com\r\nUser-Agent: InnoTools_Downloader\r\n\r\n",
    "port": 80
},
{
    "count": 1,
    "body": "RK web call",
    "uri": "http://post.securestudies.com/TapAction.aspx?campaign_id=794&tpi=4udiki&action_id=0",
    "user-agent": "InnoTools_Downloader",
    "method": "POST",
    "host": "post.securestudies.com",
    "version": "1.0",
    "path": "/TapAction.aspx?campaign_id=794&tpi=4udiki&action_id=0",
    "data": "POST /TapAction.aspx?campaign_id=794&tpi=4udiki&action_id=0 HTTP/1.0\r\nHost: post.securestudies.com\r\nUser-Agent: Ir
    "port": 80
}
```


- Example of http information
- Some interesting features worth exploring include user-agent, uri, host, path

Gathering and Labeling Samples



Free and **unrestricted** “repository of malware samples to provide security researchers, incident responders, forensic analysts, and the morbidly curious access to samples of live malicious code.”

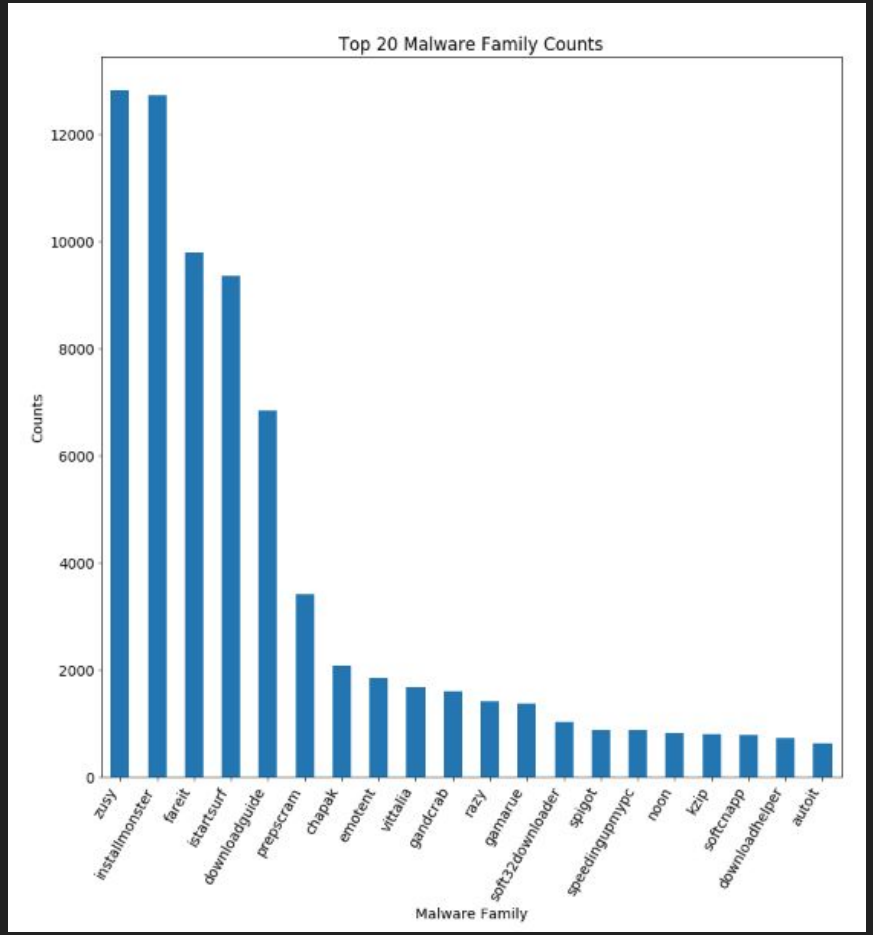
~ 70 AntiVirus engines deployed to classify and label malware based on family

Extracts family classification from each VirusTotal engine and assigns to each sample a malware family based on plurality voting

Top Extracted Classes

- Found dominant classes in our samples:
  - zusy
  - installmonster
  - fareit
  - istartsurf
  - downloadguide

- Many sparse classes necessitates pruning



Datasets Used for Classification

Dataset 1 (VT)

- VirusTotal Reports
- Samples: 95,086
- Classes: 248

Dataset 2 (VT+Cuckoo)

- VirusTotal Reports
- Cuckoo Reports
- Samples: 1,936
- Clases: 101

For all results shown, we use 10-fold cross validation

11 Features Extracted

VirusTotal Features

- TRID
- PE_RESOURCE_LIST
- EMBEDDED_DOMAINS_LIST
- IMPORTS_LIST
- CONTACTED_URLS_LIST

Cuckoo Features

- SIGNATURES
- BEHAVIOR_CALLS
- BEHAVIOR_DLL_LOADED
- NETWORK_HTTP
- NETWORK_HOSTS
- STRINGS

VirusTotal Feature Experiments

- From ~120,000 samples of our dataset, we prune out all samples where there are less than 20 samples per family
- Left with 95,086 Samples
- We do leave-one-out experiments to determine whether a feature is useful or not

SVM using all Features

Accuracy	Precision	Recall	F-score	Time (s)
84.99	83.98	84.99	83.72	3341

SVM Leave-one-out Results

Dropped Feature	Accuracy	Precision	Recall	F-score	Time (s)
TRID	84.81	83.76	84.81	83.48	3028
PE_RESOURCE_LIST	83.77	83.20	83.77	82.45	2117
EMBEDDED_DOMAINS_LIST	84.73	83.71	83.43	83.43	3527
IMPORTS_LIST	80.64	79.65	80.64	78.69	618
CONTACTED_URLS_LIST	84.78	83.81	84.78	83.37	3271

Dataset: 1 (VT)
Samples: 95,086; Classes: 248

Static SVM Final Results

Accuracy	Precision	Recall	F-score	Time (s)
84.99	83.98	84.99	83.72	3337

- None of the 5 features were determined to be non-useful from the leave-one-out experiments and therefore all 5 features were used for the final classifier.

Dataset: 1 (VT)
Samples: 95,086; Classes: 248

Cuckoo Feature Experiments

- Start with set of all malware from families with at least 50 samples = 124 families
- Randomly sample 20 from each family to run through Cuckoo Sandbox = 2,480 samples
- Some provide us with unprocessable JSON files, so we prune out families with less than 18 processable JSON files
- Left with 1,936 Samples
- We do leave-one-out experiments to determine feature usefulness

SVM using Static Features

Accuracy	Precision	Recall	F-score	Time (s)
61.73	64.57	61.73	60.69	41

SVM using Dynamic Features

Accuracy	Precision	Recall	F-score	Time (s)
66.32	70.06	66.32	6534	1187

SVM using all Features (Hybrid Analysis)

Accuracy	Precision	Recall	F-score	Time (s)
67.87	69.72	67.87	66.48	1462

Dataset: 2 (VT+Cuckoo)
Samples: 1,936; Classes: 101

SVM using all Features (Hybrid Analysis)

Accuracy	Precision	Recall	F-score	Time (s)
67.87	69.72	67.87	66.48	1462

SVM Leave-one-out Results (Hybrid Analysis)

Dropped Feature	Accuracy	Precision	Recall	F-score	Time (s)
TRID	67.87	69.92	67.87	66.49	1231
PE_RESOURCE_LIST	67.20	69.27	67.20	66.00	1290
EMBEDDED_DOMAINS_LIST	67.39	69.62	67.36	66.10	1346
IMPORTS_LIST	66.99	69.27	66.99	65.78	1209
CONTACTED_URLS_LIST	67.72	69.40	67.72	66.33	1528

Dataset: 2 (VT+Cuckoo)
Samples: 1,936; Classes: 101

SVM Leave-one-out Results cont. (Hybrid Analysis)

Dropped Feature	Accuracy	Precision	Recall	F-score	Time (s)
SIGNATURES	66.12	68.19	66.12	64.71	1525
BEHAVIOR_CALLS	67.46	69.58	67.46	66.19	1137
BEHAVIOR_DLL_LOADED	67.56	69.38	67.56	66.19	1333
NETWORK_HTTP	67.92	69.60	67.92	66.47	1432
NETWORK_HOSTS	67.92	69.60	67.92	66.47	1217
STRINGS	67.39	69.37	67.36	66.17	112

Baseline Accuracy: 67.87

Dataset: 2 (VT+Cuckoo)
Samples: 1,936; Classes: 101

SVM using all Features (Hybrid Analysis): Initial

Accuracy	Precision	Recall	F-score	Time (s)
67.87	69.72	67.87	66.48	1462

SVM using all Features (Hybrid Analysis): Final

Accuracy	Precision	Recall	F-score	Time (s)
67.98	69.79	67.98	66.66	1946

- NETWORK_HTTP and NETWORK_HOSTS were determined to be non-useful.
- Final accuracy increased by 0.11% by dropping NETWORK_HTTP and NETWORK_HOSTS.

Dataset: 2 (VT+Cuckoo)
Samples: 1,936; Classes: 101

Visualization Experiments

- From 124 families, we sample 50 malicious binaries each
- Left with 6,200 Samples

Grayscale Visualization

- Performed hexdump in unsigned decimal format
- Example:
  Hex: 4D 5A 90 00 03 00
  Dec: 077 090 144 000 003 000
- Each decimal byte converted to grayscale (0 = black, 255 = white)
- Width dependent on file size, height varies

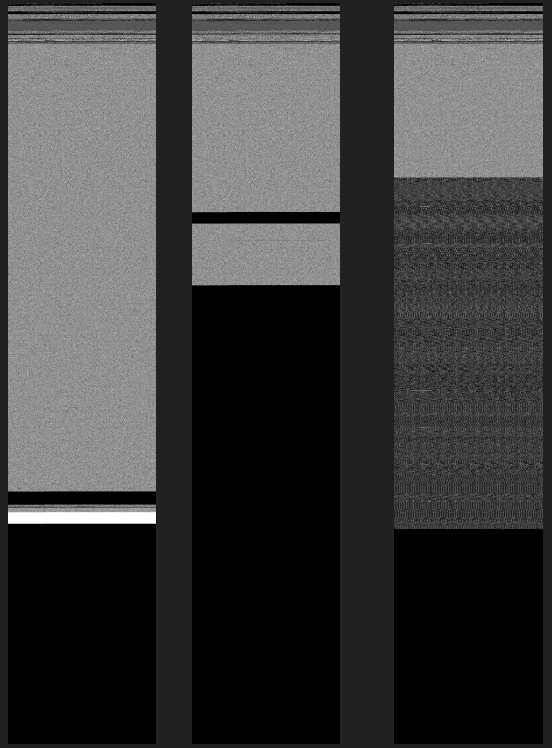
Colored Visualization

- Used trigrams of decimal bytes to map to RGB values using step size 1 and 3
- Example dump:
  Dec: 077 090 144 000 003 000
- Step size 1 (overlap):
  (R: 077, G: 090, B: 144),
  (R: 090, G:144, B: 000),
  (R: 144, G: 000, B: 003), etc...
- Step size 3 (non-overlap):
  (R: 077, G: 090, B: 144)
  (R: 000, G: 003, B: 000).

File Size to Width Reference

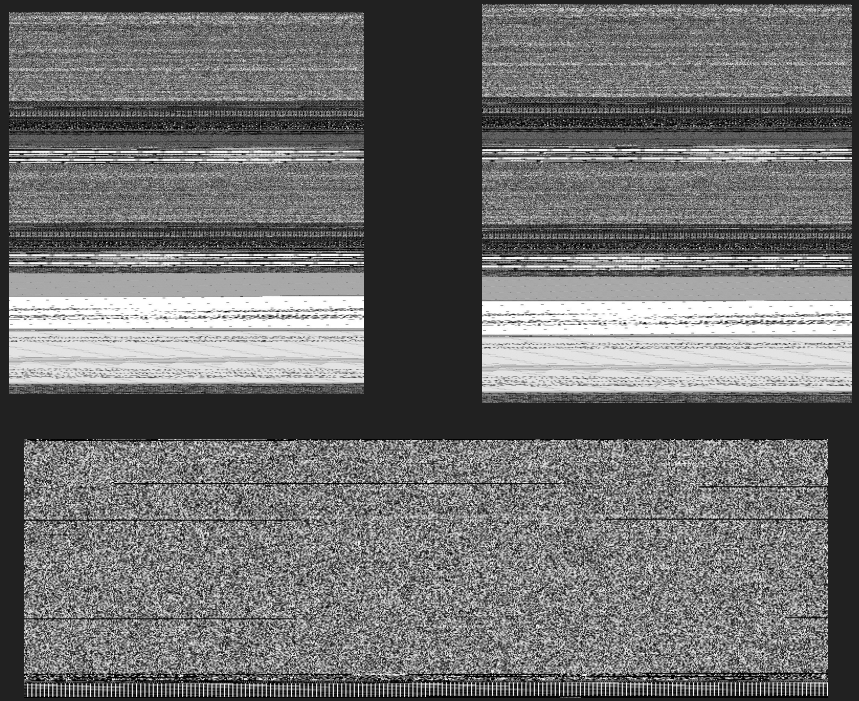
File Size	Image Width (px)
< 10 kB	32
10 kB - 30 kB	64
30 kB - 60 kB	128
60 kB - 100 kB	256
100 kB - 200 kB	384
200 kB - 500 kB	512
500 kB - 1000 kB	768
>1000 kB	1024

wannacry



Left: 0ff8d4056774a72dd5efa1c68bdbf19d
Center: 39a6d9ae74f6d956980db12e44bb98fd
Right: b8e451dd34f6ebfeb1534af098617e31

zusy

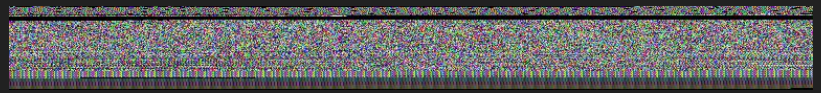


Left: 0ed668e45bc33740eb9bfaf6c32832a1
Right: 6cc7ab5e5cd44345770d3c5d275988ba
Bot: ad70072bd08167bbdcf9e6ed556e83de

emotent

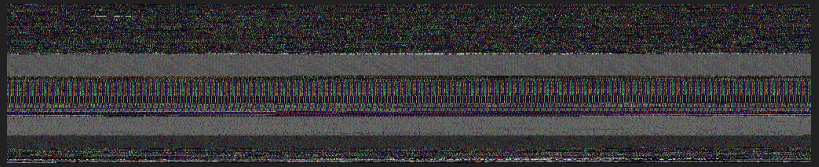
3dd8d36d2cd4c2c401f01f383b6df53a-NONOVERLAP-RGB

11a8df7304254ce2f187ec5caa3b0a01-NONOVERLAP-RGB

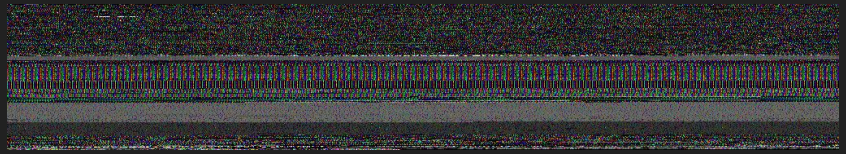


de60df4f984fc69827567e99f647e346-NONOVERLAP-RGB

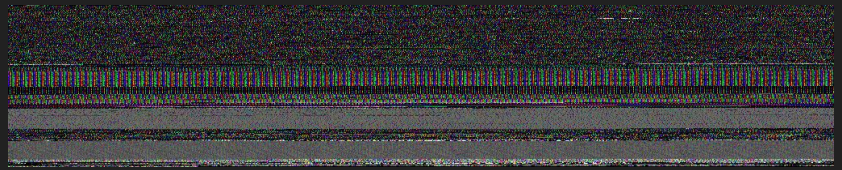
browsefox



3ca88975e9226378da296dfcceb0464c-NONOVERLAP-RGB



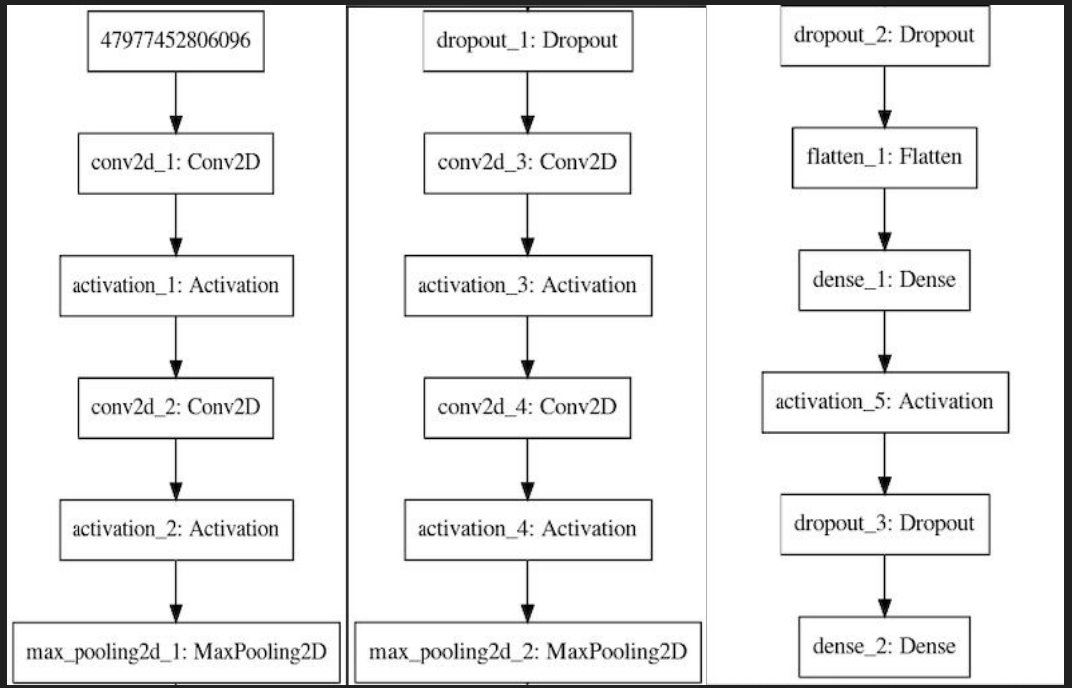
8a17f7ab8eb05238596d1b6b172cb5cc-NONOVERLAP-RGB



cefd40efbeae9d878c6204a57d1c2819-NONOVERLAP-RGB

Convolutional Neural Network Multi-Class Classifier

Simple CNN:



Training Grayscale CNN:

Model was trained in batches of 32, with a callback to stop when validation loss was no longer decreasing over an interval of 100 epochs.

Thanks to University of Houston Core facility for Advanced Computing and Data Science providing allocation on Opuntia.

Image Pre-Processing For Training

Keras Images Preprocessing:

- Images were passed through Keras Image Generator to perform the following in memory:
  - Resize images (performing interpolation)
  - Normalize pixel values between 0 and 1
  - Randomize (with seed)
  - Split Train and Sample
  - Augment Training Data
    - Images were shifted up and down by a factor of .2

Image Augmentation Example:

Source: https://blog.keras.io/building-powerful-image-classification-models-using-very-little-data.html

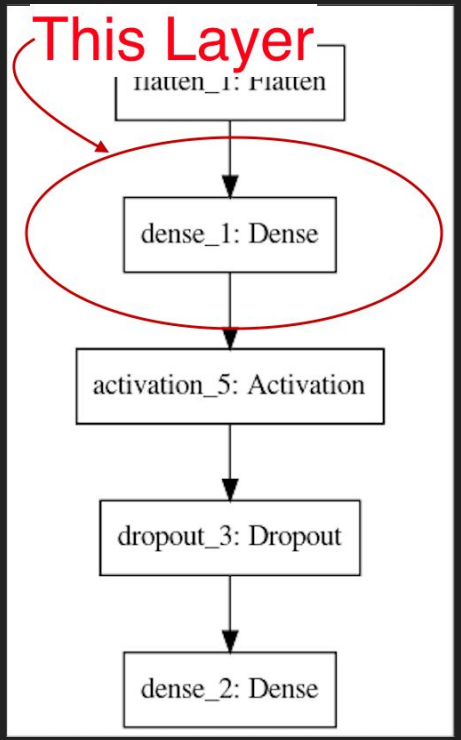
Visualization Moving Forward

- Feature Work
- Future Work

Feature Work With Visualization

Analyzing a new feature set

- Remove Fully Connected layer after flattening takes place
- Use the layer as our feature set
- Apply clustering and classification algorithms using new feature set to determine efficacy for attribution

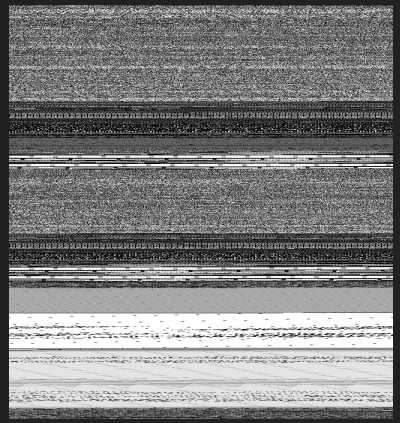


Future Work With Visualization

Preprocessing

- In the paper Malware Images: Visualization and Automatic Classification, Nataraj Et al, the research proposes using Gabor filters on images
- The malware images contain lots of high frequency content
- Apply different low-pass filters in preprocessing step

Original



Magnitude Spectrum

Zusy-0ed668e45bc33740eb9bfaf6c32832a1

Sponsors

Matthew Elder

William La Cholter

Johns Hopkins University
Applied Physics Laboratory

Special Thanks

Dr. Rakesh Verma
Avisha Das

University of Houston
Department of Computer Science

References

[1] Insurehub.org.” [Online]. Available: https://insurehubg.org/
[2] “File statistics during last 7 days.” [Online]. Available: https://www.virustotal.com/en/statistics/
[3] “Desktop operating system market share worldwide.” [Online]. Available: http://gs.statcounter.com/os-market-share/desktop/worldwide
[4] Levinec, “Malware names.” [Online]. Available: https://docs.microsoft.com/en-us/windows/security/threat-protection/intelligence/malware-naming
[5] D. Ucci, L. Aniello, and R. Baldoni, “Survey of machine learning techniques for malware analysis,” Computers & Security , vol. 81, pp. 123–147, 2019. [Online]. Available: https://doi.org/10.1016/j.cose.2018.11.001
[6] Y. Ye, T. Li, D. Adjeroh, and S. S. Iyengar, “A survey on malware detection using data mining techniques,” ACM Comput. Surv., vol. 50, no. 3, pp. 41:1–41:40, Jun. 2017. [Online]. Available:http://doi.acm.org/10.1145/3073559
[7] R. Sihwail, K. Omar, and K. A. Z. Ariffin, “A survey on malware analysis techniques: Static, dynamic, hybrid and memory analysis,” 2018.

References

[8] L. Nataraj, S. Karthikeyan, G. Jacob, and B. S. Manjunath, “Malware images: visualization and automatic classification,” in 2011International Symposium on Visualization for Cyber Security, VizSec ’11, Pittsburgh, PA, USA, July 20, 2011, 2011, p. 4. [Online]. Available: https://doi.org/10.1145/2016904.2016908

[9] E. Gandotra, D. Bansal, and S. Sofat, “Malware analysis and classification: A survey,” Journal of Information Security, vol. 05, no. 02, pp. 56–64, 2014. [Online]. Available: https://doi.org/10.4236/jis.2014.52006

[10] R. M. Verma, M. Kantarcioglu, D. J. Marchette, E. L. Leiss, and T. Solorio, “Security analytics: Essential data analytics knowledge for cybersecurity professionals and students,” IEEE Security & Privacy, vol. 13, no. 6, pp. 60–65, 2015. [Online]. Available: https://doi.org/10.1109/MSP.2015.121

[11] “Virusshare.com.” [Online]. Available: https://virusshare.com/

[12] “Automated malware analysis.” [Online]. Available: https://cuckoosandbox.org/about

References

[14] Z. Cui, F. Xue, X. Cai, Y. Cao, G. Wang, and J. Chen, “Detection of malicious code variants based on deep learning,” IEEE Trans.Industrial Informatics, vol. 14, no. 7, pp. 3187–3196, 2018. [Online]. Available: https://doi.org/10.1109/TII.2018.2822680

[15] A. H. Sung and S. Mukkamala, “Identifying important features for intrusion detection using support vector machines and neural networks,” in 2003 Symposium on Applications and the Internet (SAINT 2003), 27-31 January 2003 - Orlando, FL, USA, Proceedings, 2003, pp. 209–217. [Online]. Available: https://doi.org/10.1109/SAINT.2003.1183050

[16] F. Chollet et al., “Keras,” https://keras.io, 2015.

[17] H. Faridi, S. Srinivasagopalan, and R. Verma, “Performance Evaluation of Features and Clustering Algorithms for Malware,” in 2018 IEEE International Conference on Data Mining Workshops (ICDMW). Singapore, Singapore: IEEE, Nov. 2018, pp. 13–22. [Online]. Available: https://ieeexplore.ieee.org/document/8637452/

[18] ——, “Parameter tuning and confidence limits of malware clustering,” In Proceedings of the Ninth ACM Conference on Data and Application Security and Privacy, CODASPY 2019, Richardson, TX, USA, March 25-27, 2019, 2019, pp. 169–171. [Online]. Available: https://doi.org/10.1145/3292006.3302385