Malware Analysis & Classification

Team:

Karol Pierre,Yu Qiu, Cheng Xu, Zichao Yang

Advised By:

Dr. Matthew Elder, JHU Applied Physics Laboratory

William J. La Cholter, JHU Applied Physics Laboratory

Dr. Xiangyang Li, Johns Hopkins University

Background

- Malware (malicious software) are programs intentionally created and designed to cause damage to computers, networks, and more
- Types of malware are distinguished by propagation behavior
  - Virus infects one file (host) and then spreads from file to file
  - Worm is a standalone software and can spread from network to network

Figure 1: Types of Malware
https://enterprise.comodo.com/what-is-malware.php

Why Does This Matter?

- Malware has become so advanced that some variants have the ability to detect if they exist within a virtual environment and terminate self

- Cause greater physical damage:
  - Stuxnet, Loapi

- Ability to accurately detect specific malware signatures or features via machine learning algorithms, will aid in the mitigation and possible development of controls to prevent future strains of malicious software from being able to infect systems

Research Question

Main Focus

- Through the use of dynamic and static analysis of malware samples we will be able to classify these samples into families via the application of Machine Learning Algorithms

Sub-Focus

- Accuracy of Malware Classification: the ability to best determine which malware variant each malicious software belongs to, in order to assist with best methods for malware mitigation
- Dynamic vs. Static Analysis: Comparison of classification results

Literature Review

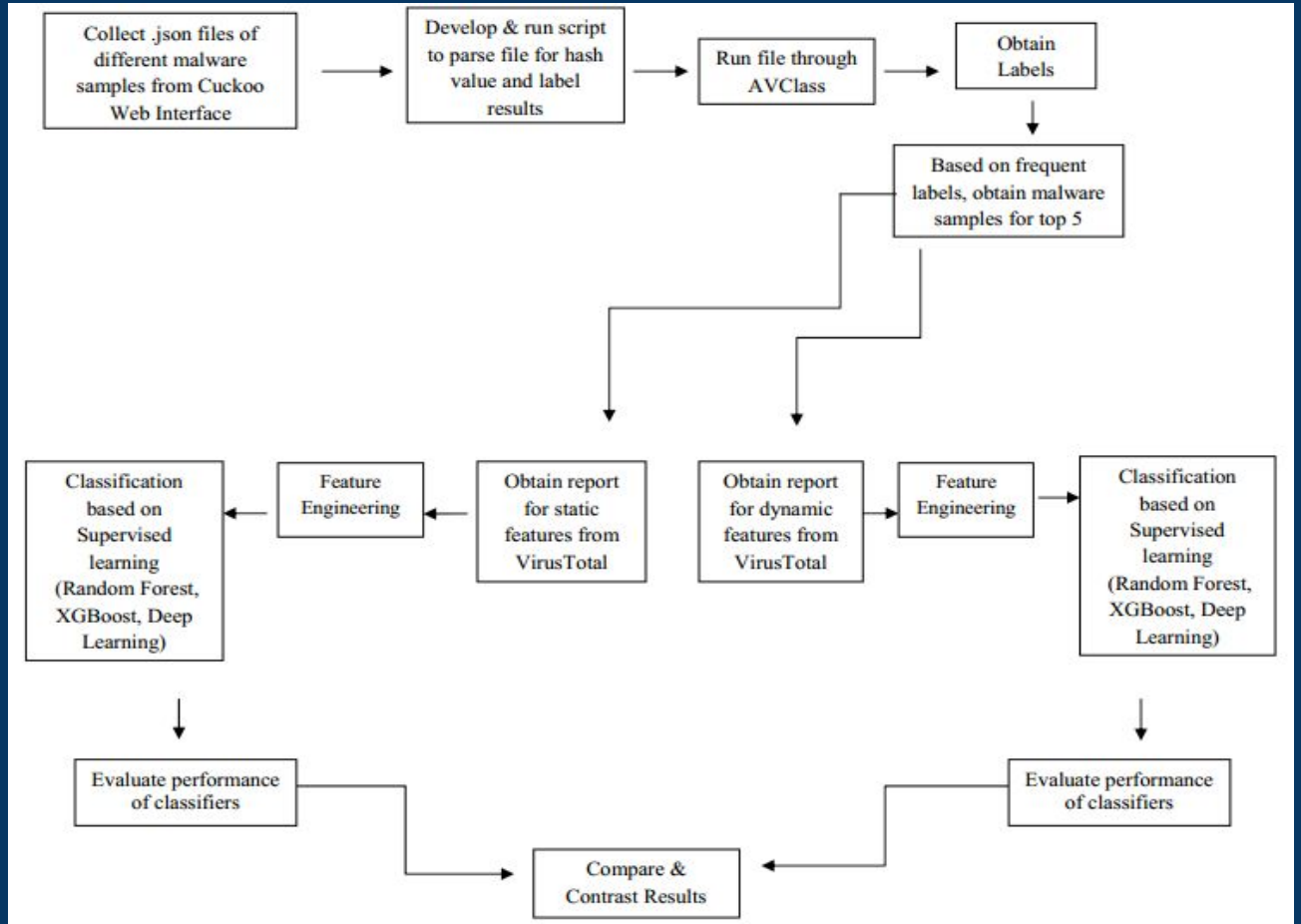
Paper	Classification Problem	Type of Input	Type of Features	Algorithms Used
A machine learning approach for Linux Malware Detection	Malware Categorization	Goodware Binaries, Malware Binaries	System Calls	Naive Bayes Random Forest
A survey on malware detection using data mining techniques	Benign vs. Malware Categorization	Binary Code	Windows API Calls, N-Grams of Program, Opcodes Interpretable Strings	Clustering
An investigation of a deep learning based malware detection system	Benign vs. Malware Categorization	Malware assembled code	Opcodes(frequency), labels(benign or malware), Interpretable strings	Deep neural network
Automatic analysis of malware behavior using machine learning	Behavior	Malware Binaries	Changing registry keys or modifying system files	Scalable clustering
AVclass: A Tool for Massive Malware Labeling	Malware Family Classification	VirusTotal .json reports for malware samples	Labels	N/A

Literature Review

HDM-Analyser: a hybrid analysis approach based on data mining techniques for malware detection	Detection Accuracy	Executed Malware, PE Sections	API Calls	Machine Learning Model best suited for ambiguity points of code and combine both dynamic and static features
Hybrid Analysis and Control of Malware	Identification	Malware binaries Executed Malware	N/A	N/A
Malware Detection using Machine Learning and Deep Learning	Benign vs. Malware Classification Binary Classification	Disassembled benign executables, disassembled malware binaries	Opcode frequencies, API Calls	Random Forest
On the feasibility of Malware Authorship Attribution	Binary Code Authorship	Binary Code	System calls, errors, opcode, compiled and system information	Clustering
When coding style survives compilation: De-anonymizing Programmers from Executable Binaries	Executable Binary Classification	Executable binaries	Features extracted from syntax trees, Word level N-grams	Random Forest

Technical Solution | Design & Analysis

Parallel Approach:
Dynamic & Static Analysis



Technical Solution | Design & Analysis (cont.)

Hybrid Analysis

- Combine features from Static & Dynamic Dataset
- Classification based on Logistic Regression, Random Forest, XGBoost, and Neural Networks
- Compare performance results with Static Performance and Dynamic Performance

Dataset Details

- Labels of Focus:
  - **Emotet:** trojan with worm-like behavior that steals sensitive and private information
  - **Fsysna:** crypto-jacking malware capable of infecting an entire network of systems
  - **Occamy:** executes commands**,** extracts system info, network info and more and sends to remote attacker for analysis
  - **Swrort:** opens backdoor on infected system for remote threat actors to execute more malware
  - **Nanobot:** botnet that can be used for screen captures, crypto-currency mining, webcam session theft, and more

- 520+ total malware samples

Dynamic Features

Behavior:

- Operations:
  - Files Opened
  - Files Deleted
  - Has PCAP (network traffic)
  - Processes Created
  - Processes Injected
  - Files Written
  - etc

Static Features

PE_Sections
- Raw Size
- Virtual Size
- Entropy

- **PE_Imports**
  - Focus on Windows API Dlls
    - GDI32.dll
    - KERNEL32.dll
    - MFC42u.DLL
    - MSVCRT.dll
    - PSAPI.DLL
    - USER32.dll
    - etc

Experimentation

Parallel Approach

1. Collect JSON files of malware samples
2. Process the malware samples and performed triple-check to extract needed data
3. Build static and dynamic datasets
4. Select 4 algorithms to build the classifiers respectively while grid search is performed for parameter optimization

Methods

- 4 Machine Learning Models
  - 3 Traditional: Logistic Regression, Random Forest, XGBoost
  - 1 Modern: Neural Networks
  - Grid Search used for hypertuning parameters

Logistic Regression

- Logistic Regression is a supervised machine learning algorithm which models the probability of a datapoint to a particular class without directly modeling the values of the datapoint. This model uses the logistic function or sigmoid function to predict the binary outcome as either 1 or 0, yes or no, or true or false.

- **Hyperparameters Used?**
  - Regularization
  - Weights of Regularization (Penalty Weight)

Random Forest

- Random forest is an ensemble learning. Like its name implies, it consists of a large number of individual decision trees that operate parallelly. Each individual tree in the random forest spits out a class prediction and the class with the most votes becomes our model's prediction.

- **Hyperparameters Used?**
  - Number of Trees
  - Maximum depth of each tree

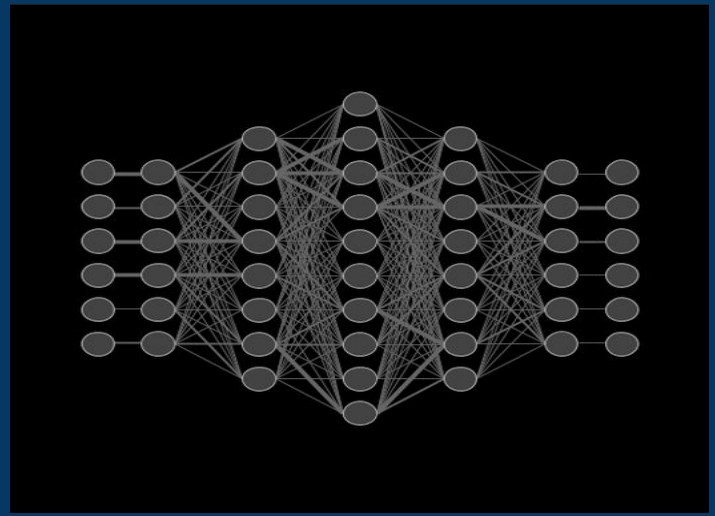
XGBoost

- XGBoost stands for extreme gradient boosting and is also a supervised ensemble learning algorithm. It is an implementation of gradient boosted decision trees designed for speed and performance. Unlike the Random Forest, XGBoost constructs trees sequentially instead of parallelly.

- **HyperParameters Used?**
  - Number of Trees
  - Maximum Depth of each tree
  - Learning Rate

Neural Network

- Neural Networks are a set of algorithms used for machine learning that are modeled from biological neural networks. This process occurs by forward propagating data in the network to obtain the output and then back propagating the gradient of the cost function which will be used to update the weights and bias. In order to prevent overfitting problem, we applied dropout and batch normalization techniques.



- **HyperParameters Used?**
  - Number of Layers
  - Number of Neurons
  - Activation Function
  - Dropout Probability

Evaluation & Result Analysis

Feature Importance Analysis

- Acquired by building random forest tree model of 100 trees (general rule of thumb) and using criterion of entropy(how different)

Static	**Dynamic**	**Hybrid**
.text_virtual_size	dns_lookups	.text_virtual_size
.text_entropy	ip_traffic	.text_raw_size
.text_raw_size	command_executions	.data_raw_size
.rsrc_virtual_size	modules_loaded	.rdata_raw_size
rsrc_entropy	files_copied	.text_entropy

Evaluation & Result Analysis (cont.)

Static Analysis

- Best model: Logistic Regression
- Hyperparameters:
  - L1 Regularization(Lasso Regression)
  - Penalty Weight 4.64
- Performance Value: 82%

Evaluation & Result Analysis (cont.)

Dynamic Analysis

- Best model: Neural Networks with 2 hidden layers
- Hyperparameters:
  - 2 hidden layers
  - 25 neurons/hidden layers
  - Dropout Rate: 20%
  - ReLu activation
- Performance value: 79%

Evaluation & Result Analysis (cont.)

Hybrid Analysis

- Best model: Neural Networks with 2 hidden layers

- Hyperparameters:
  - 2 hidden layers
  - 100 neurons each hidden layer
  - Dropout rate 10%
  - ReLu activation

- Performance value: 87%

Confusion Matrix

Neural Networks		True Fsysna	True Nanobot	True Occamy	True Emotet	True Swrort
1 Layer	Predicted Fsysna	13	1	1	2	0
	Predicted Nanobot	3	8	0	0	0
	Predicted Occamy	1	1	19	1	2
	Predicted Emotet	0	0	0	17	0
	Predicted Swrort	0	0	0	0	10
2 Layers	Predicted Fsysna	13	2	1	1	0
	Predicted Nanobot	2	9	0	0	0
	Predicted Occamy	1	2	20	1	0
	Predicted Emotet	0	0	0	17	0
	Predicted Swrort	0	0	0	0	10
3 Layers	Predicted Fsysna	12	2	1	2	0
	Predicted Nanobot	1	10	0	0	0
	Predicted Occamy	3	2	19	0	2
	Predicted Emotet	0	0	0	17	0
	Predicted Swrort	0	0	0	0	10

Probability of Each Class in Prediction

- SoftMax Function used to obtain probability

	prob_fsysna	prob_nanobot	prob_occamy	prob_emotet	prob_swrort
0	0.000037	2.787066e-08	8.976957e-07	9.999595e-01	2.465697e-06
1	0.000053	6.399413e-06	1.501655e-06	1.288538e-04	9.998098e-01
2	0.000038	9.996002e-01	3.385113e-04	1.876655e-05	4.379531e-06
3	0.985317	4.517088e-04	1.075912e-02	3.310427e-03	1.618540e-04
4	0.000017	5.143971e-07	1.615339e-06	9.999810e-01	1.535007e-08
5	0.000200	9.449723e-08	5.073924e-06	9.997947e-01	4.863345e-08
6	0.721787	3.727260e-04	6.280800e-04	2.771845e-01	2.767466e-05
7	0.878726	1.093795e-02	2.990511e-03	9.713684e-05	1.072488e-01
8	0.006072	8.357639e-01	3.917148e-02	1.176029e-01	1.389510e-03
9	0.000024	1.685386e-04	9.996547e-01	1.526913e-04	2.349616e-07
10	0.879528	1.111073e-02	3.029775e-03	1.088970e-04	1.062222e-01
11	0.317570	1.426599e-02	7.708222e-03	6.555899e-01	4.865931e-03
12	0.000691	4.077718e-03	4.390882e-04	5.315362e-02	9.416389e-01
13	0.000025	5.396026e-05	9.999095e-01	1.118357e-05	7.606883e-08
14	0.462305	4.833749e-01	5.307614e-02	7.785934e-04	4.655649e-04
15	0.646763	3.010790e-01	5.016786e-02	1.434981e-03	5.550865e-04

Conclusion

Best Performer: Neural Network w/ 2 hidden layers

Best Approach: Hybrid Analysis

Future Work

- Expand dataset
  - May help improve performance values

- Stacking some of the different models into one large model

Works Cited

- https://giphy.com/gifs/the-matrix-WoD6JZnwap6s8
- https://enterprise.comodo.com/what-is-malware.php
- https://www.sccpre.cat/maxp/moRTxw/
- Cuckoo Sandbox.https://cuckoosandbox.org/

Questions?