CS658A: Malware Analysis and Intrusion Detection
Project Report

Team Name: BINARY BEASTS

TUSHAR GAUTAM	20111071 (tushark20@iitk.ac.in)
HARSIKA DIKSHA	20111021 (harsikad20@iitk.ac.in)
SUMESH KUSHWAHA	20111066 (sumeshk20@iitk.ac.in)
SHUBHAM KUMAR	20111063 (shubhamkum20@iitk.ac.in)
ABHISHEK KUMAR SAINI	20111401 (abhik20@iitk.ac.in)
MUSKAN RATHORE	20111036 (muskanr20@iitk.ac.in)

Title : Dynamic Malware Analysis using Machine Learning Algorithms

---

Supervised By: Prof. Sandeep K Shukla

Contents

1 Summary

Malware analysis is a process of analyzing executables and detecting potential malicious software. A new family of malware has emerged with advanced evasion, making them much more challenging to use conventional methods. Malware detection is an essential factor in the computer system's security; however, currently utilized signature-based methods cannot accurately detect zero-day attacks and polymorphic viruses.
Cuckoo sandbox is used for generating analytic reports of portable executable files by executing them in a virtual environment. Classified windows PE files are collected from C3I center IIT, Kanpur which are executed in the sandbox. A parser is built for extracting behavioral features from analytical reports generated by the cuckoo sandbox. Different Machine learning models such as KNN, Decision Tree, multiclass SVM, random forest, and DNN were implemented for the classification of PE files and achieved the highest accuracy with DNN. All related codes are uploaded **here**.

1.1 Problem Statement:

Developing malware detection system for windows to perform dynamic malware analysis using the state-of-the-art machine learning algorithms.

1.2 Goals:

➢ Collection of benign and malicious portable executable files for windows.
➢ Obtaining behavioural data of classified windows PE files using cuckoo sandbox.
➢ Generating raw dataset with significant features from behavioural data.
➢ Performing pre-processing and apply Machine Learning models.
➢ Finally building a malware detection system using ML model with highest accuracy.

1.3 Data Source:

We have collected different sets of Portable Executable (PE) files from the C3i center, IIT Kanpur. It is a collection of labelled benign and malicious PE files for the Windows environment. We have obtained the behavioral data of these raw PE files by executing them in cuckoo sandbox. We created a parser for parsing the analytic reports, that extracts the significant features, and generates a CSV file with these selected features.

Keywords:

Malware Detection, Malware Classification, Dynamic Analysis, Portable Executable, Cuckoo Sandbox, Machine Learning.

2 Introduction

Malicious software, which is most commonly known as **malware**, is any program or file that is intentionally made harmful to a computer, network, or server. Different types of malware include computer viruses, worms, Trojan horses, ransomware, and spyware. These malicious programs steal, encrypt and delete sensitive data, alter or can hijack core computing functions and monitor end users' computer activity. It can infect networks and devices and is designed to harm those devices, networks, and/or their users in some way.

Malware analysis is a process of analyzing executable and detecting potential malicious software. Malware detection is a vital aspect of computer security. There are two different ways of analysis, static and dynamic analysis. Static analysis is performed without running the software, whereas in dynamic analysis software behaviour is captured by executing the software in a safe environment. Various tools are available for malware analysis that can perform static and dynamic analysis. VirusTotal, Strings, IDA Pro, etc are used in static analysis and Cuckoo sandbox, RegShot, Process Hacker, CaptureBAT, etc are used in dynamic analysis. As we are aware that in static analysis important behaviour of malware can be missed, so we choose to perform dynamic malware analysis. In dynamic analysis files are actually executed and behaviour of files are considered which gives more insights on interactions of the software

with the system variables and the files.

We have developed a malware detection system that performs malware analysis using the state of art machine learning algorithms. Although the methods available such as signature-based are capable of detecting malware but struggles in detecting zero-day attacks and polymorphic viruses. We have created our own dataset for the analysis, and Cuckoo sandbox is used for generating analytic reports of portable executable files. Analytics reports are generated in JSON format containing static as well as behavioural information. We have built a parser that extracts the behavioral features from analytical reports. Our parser generates the final dataset which is used for training different Machine learning models such as KNN, Decision Tree, multiclass SVM, random forest, and DNN for the classification of PE files.

3 Cuckoo Sandbox

Detection and classification systems require classified data for classification. Due to the unavailability of raw classified data, we have collected different sets of Portable Executable (PE) files from the C3i center, IIT Kanpur. It is a collection of benign and malicious PE files which is labeled with proper benign and malware class for the Windows environment. We obtained the behavioral data from these raw PE files by executing them using the cuckoo sandbox.

3.1 Overview

Cuckoo Sandbox is an open source automated malware analysis system. One can throw any suspicious file at it and in a matter of minutes Cuckoo will provide a detailed report outlining the behavior of the file when executed inside a realistic but isolated environment. In these evolving times, detecting and removing malware artifacts is not enough: it's vitally important to understand how they operate in order to understand the context, the motivations, and the goals of a breach. By default it is able to:

- ➢ Analyze many different malicious files (executables, office documents, pdf files, emails, etc) as well as malicious websites under Windows, Linux, macOS, and Android virtualized environments.
- ➢ Trace API calls and general behavior of the file and distill this into high level information and signatures comprehensible by anyone.
- ➢ Dump and analyze network traffic, even when encrypted with SSL/TLS. With native network routing support to drop all traffic or route it through InetSIM, a network interface, or a VPN.
- ➢ Perform advanced memory analysis of the infected virtualized system through Volatility as well as on a process memory granularity using YARA.

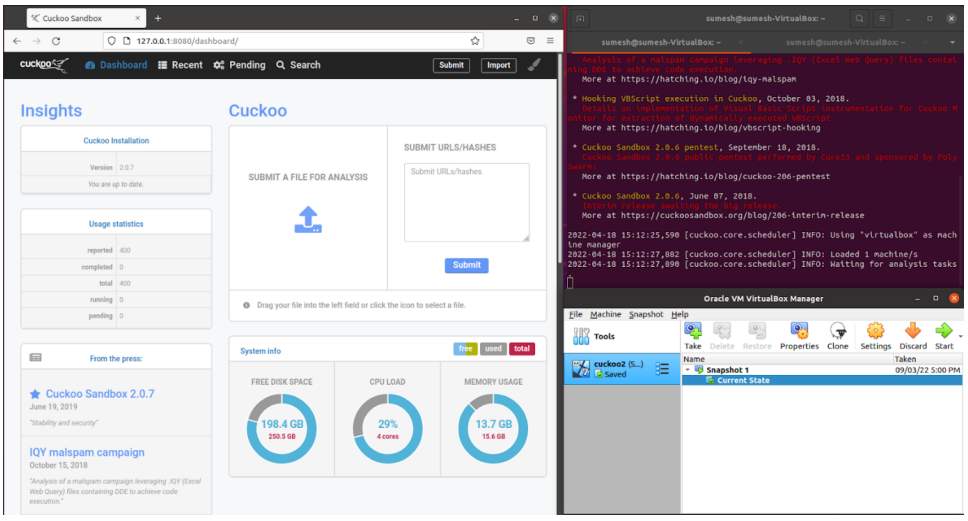

Figure 1: Cuckoo installation

3.2 Setup

- ➢ We had configured Cuckoo Sandbox in a virtual environment.

- ➢ Cuckoo host is in Ubuntu 20.04 Cuckoo guest is in Windows 7 Ultimate.
- ➢ As Cuckoo is still based on Python2.7 which is outdated, we had faced errors several times during the setup.
- ➢ To overcome various errors, we had used various scripts available on the internet but none of them make installation without any errors.
- ➢ So, we have corrected those collection of scripts and used these scripts to install cuckoo on virtual machine and exported that virtual machine in .OVA format.
- ➢ Now anyone can install cuckoo by using corrected script or can use cuckoo without installing it by just importing our virtual machine image in their virtualbox.
- ➢ The Cuckoo we had used for our analysis is of version 2.0.6
- ➢ We had used Cuckoo guest in "Host-only" network environment.
- ➢ For our analysis we configured cuckoo.conf for both static behavioural analysis.
- ➢ We had disabled memory dump option as it was not needed for our analysis and also it was taking very huge memory space.
- ➢ Also, we enabled IP forwarding. So, the internet connection gets routed from host machine to cuckoo guest VM.

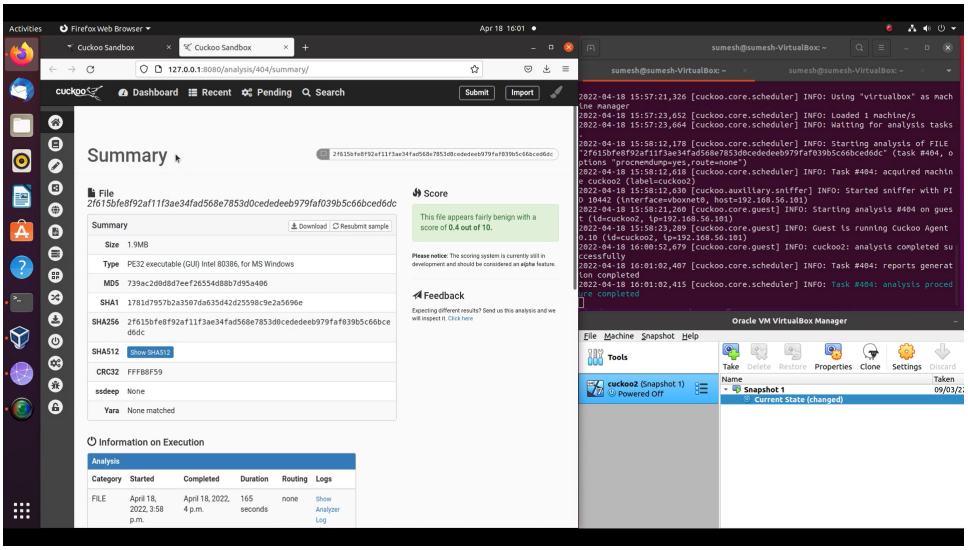

Figure 2: Cuckoo setup

4 Dataset Collection

The Portable Executable (PE) format is a file format for executables, object code, DLLs and others used in 32-bit and 64-bit versions of Windows operating systems. The PE format is a data structure that encapsulates the information necessary for the Windows OS loader to manage the wrapped executable code. This includes dynamic library references for linking, API export and import tables, resource management data and thread-local storage (TLS) data. On NT operating systems, the PE format is used for EXE, DLL, SYS (device driver), MUI and other file types. The Unified Extensible Firmware Interface (UEFI) specification states that PE is the standard executable format in EFI environments.

4.1 Dataset overview

We have collected JSON files for **995 classified PE files** with 399 Benign files, 98 Backdoor files, 100 Trojan-Downloader, 99 Trojan-Dropper, 99 Trojan, 100 Virus and 100 Worms files. By running the Portable Executable (PE) files in cuckoo sandbox environment, we have collected the Analysis Report in JSON format. JSON files consists of lots of information like PE information, signatures, process information, and other corresponding attributes of PE files. We have created a parser for parsing these JSON files, that extracts the significant features, and generates a CSV file with these selected features for further feature selection. We have considered api calls, files information such as file written, file deleted, regkey information like regkey written and deleted, DLL loaded and duration as significant features. We

have not considered cuckoo score as a feature because we wanted to keep our analysis free from cuckoo's influence.

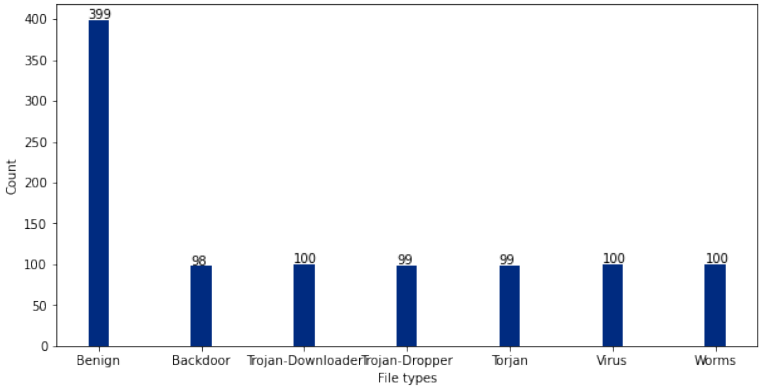

Figure 3: PE files count

4.2 Dataset details

We have created our dataset by taking different combinations of features and tested models on them. We have found an optimal sets of features for our analysis. After feature selection, we are using these features to train our machine learning models and detect malicious files. Brief description of feature set can be found below:

- **API calls:** Number of the times apis have been called by the PE files.
- **DLL laoded:** List of Dynamic libraries loaded by the PE files. We have treated it as binary data, if DLL is loaded then we assigned it as 1 and if not then 0.
- **Files deleted and written:** List of files modified and deleted by PE files. We have considered this feature as binary data, if files are modified or deleted then we assigned it as 1 else 0.
- **Regkey deleted and written:** List of registry keys modified and deleted by PE files. It is treated as binary data, if Regkeys are modified or deleted then we assigned this value as 1 else 0.
- **Duration:** Time duration for which PE is executed in virtual environment.

5 Experiments

We have preprocessed the Data, First, we normalize the data; usually, normalization is done so that all the features are at the same scale. Otherwise, our model will give high importance to higher numerical values. Then we use PCA, which is used to reduce the dimensionality of large datasets. But reducing the number of dimensions means we are also compromising the accuracy. But it is kind of a trade-off with little accuracy for simplicity, also Machine learning algorithms perform faster with less number of dimensions. So we took the top K-features out of 14+ features.

5.1 PCA with Top 100 Features

After seeing the variance plot, we first took only top 100 features Because according to maximum variance formulation definition, we need to maximize the variance and this can be done by maximizing:

$$\sigma^2(\hat{x}) = w_1^T S w_1$$

$$S = \frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})(x_n - \bar{x})^T$$

where S is the co-variance matrix of the observed data in the original high dimensional space.

Results with Top 100 Features:

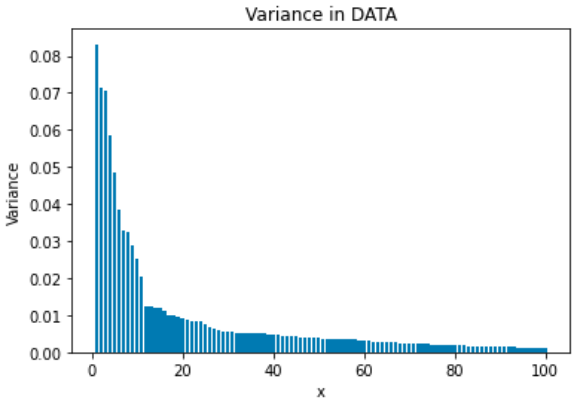

Figure 4: Variance Plot of Top 100 Features

Model(With 100 Features)	**Accuracy**
SVM	72.36
Decision Tree	89.44
Random Forest	**93.46**
K-Nearest Neighbour	89.94
Deep Neural Network	79.30

Table 1: Results for different models with Top 100 Features

5.2 PCA with Top 350 Features

Now we took only top 350 features.

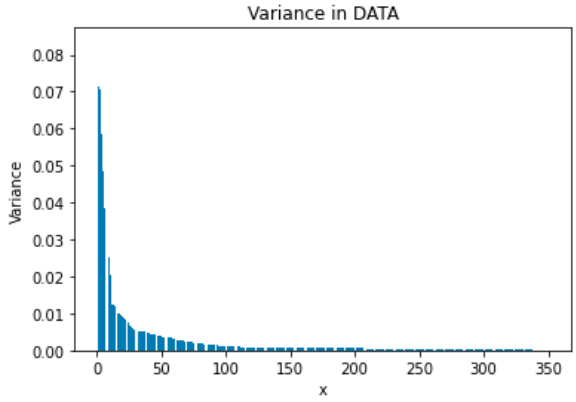

Figure 5: Variance Plot of Top 350 Features

Results with Top 350 Features:

Model(With 350 Features)	**Accuracy**
SVM	80.40
Decision Tree	91.95
Random Forest	**93.46**
K-Nearest Neighbour	80.40
Deep Neural Network	87.84

Table 2: Results for different models with Top 350 Features

We can see as we increase number of features from 100 to 350, most of the models give better accuracy than previous models(with top 100 features).

5.3 PCA with Top 500 Features

Now we took only top 500 features, because we found out that after 500 variance is almost NULL and the best results are found out by trying various Machine Learning Models with various parameters combinations.

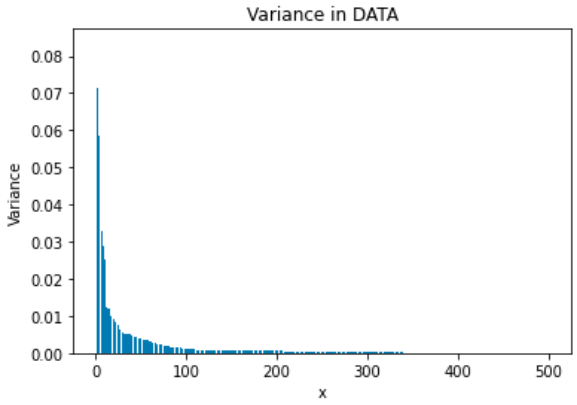

Figure 6: Variance Plot of Top 500 Features

5.4 Machine Learning Models:

➢ **Multiclass SVM:** Basically, SVM doesn't support multiclass SVM. For multiclass SVM same principle is utilized after breaking down multiclass problems into smaller subproblems.

Kernel	Accuracy	F1-score
Poly	41.21	24.05
RBF	67.34	60.77
Linear	**91.46**	**91.18**

Table 3: Accuracy and F1 score for different kernels in SVM

SVM with Linear kernel with default parameters gives 82.42 accuracy and 84.23 F1 score.

➢ **Decision Tree:** The decision Tree used a flow chart like a tree structure to show the prediction that results from a series of feature-based splits. It starts with the root node and ends with a decision made by leaves. Here, Decision tree for our dataset has optimal depth as 8, increasing and decreasing the depth both will decrease accuracy.

Criteria	Accuracy
Gini	88.0
Entropy	**91.45**

Table 4: Accuracy for different Criteria in Decision Tree

➢ **Random Forest:** Random forests are created from subsets of data, and the final output is based on average or majority ranking. It selects observations, builds a decision tree, and takes the average result.

Estimators	Accuracy
1000	**93.46**
500	92.46
100	92.46

Table 5: Accuracy for different estimators in Random Forest

➢ **K-Nearest Neighbour:** It is instance-based learning where we don't learn any weights from training data to predict the output, but it uses the entire training instances to predict the output of unseen data.

N-Neighbours	Accuracy
20	75.37
15	74.87
7	73.86
5	76.88
3	**78.89**

Table 6: Accuracy for different number of neighbours in Random Forest

➢ **Deep Neural Network:** Neural Network made up of layers, our model contains up to 4 layers. Neural network mainly has 6 components:
- ✧ **Weights:** These are theta, which we use in our algorithms.
- ✧ **Layers:** Our Network have up to 4 layers.
- ✧ **Forward Propagation:** Use the Features/Weights to get Z and A.
- ✧ **Back Propagation:** Use the result of forward propagation/weights to get S.
- ✧ **Gradient:** Calculating the cost/gradient of each weight.
- ✧ **Gradient Descent:** Find the best weight/hypothesis.

We experimented our network with different number of layers(optimal answer with 4 layers), different number of epochs and different activation function and To prevent over fitting we use a dropouts between the layers.

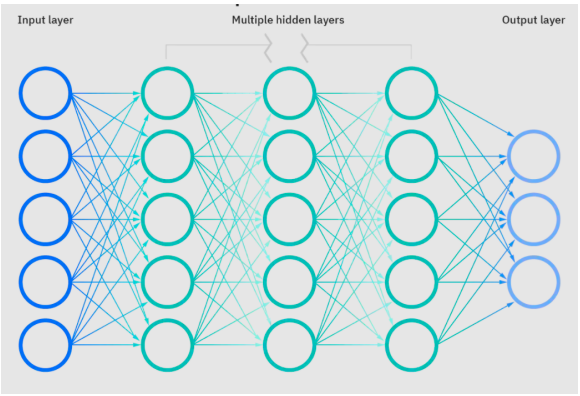

Figure 7: Neural Network
Source: https://www.ibm.com/cloud/learn/neural-networks

Layers	Accuracy
3	94.87
4	**97.69**

Table 7: Accuracy for different number of Layers in Deep Neural Network

6 Results

Models	**Accuracy**
K-Nearest Neighbour	78.89
Decision Tree	91.45
SVM	91,46
Random Forest	93.46
Deep Neural Network(Best)	**97.69**

Table 8: Accuracy for different number of Layers in Deep Neural Network

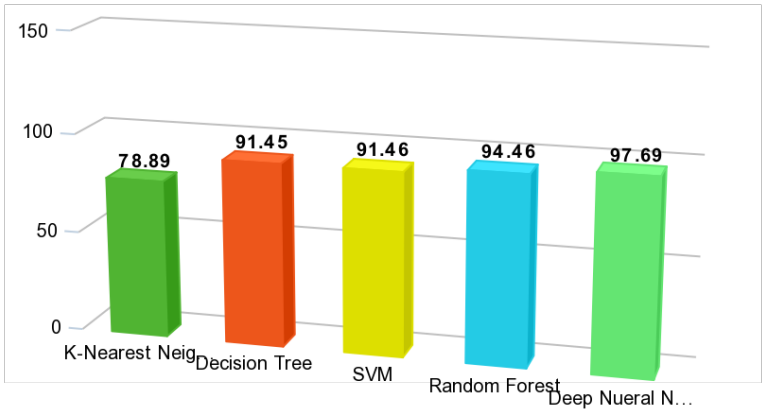

Figure 8: Results

7 Conclusion and Limitations

Signature based methods are very popular in malware detection but could not accurately detect zero-day attacks and polymorphic viruses. It can be overcome by machine learning algorithms by learning the behaviour of malicious softwares. In this project we built a malware detection system using state-of-the-art machine learning algorithms. We have used KNN, multiclass-SVM, random forest, decision tree and DNN for the malware detection. Among all these models we have achieved highest accuracy of **97.69%** using Deep Neural Network.

However our project faced certain limitations which are listed below:

- **Smaller Dataset:** Extraction of Analytics Report from Cuckoo took lots of time, so we have considered only 995 PE files for our analysis.
- **Limited significant features:** Analytic reports contains lots of information about PE files like signatures, cuckoo score, duration, process and memory information, etc. considering all the fields from JSON files will create extremely large feature set.
- **Windows Malware:** Our analysis only covers windows PE files, so our detection system can only detect malicious files for windows environment.

Our analysis could be extended by considering significantly good amount of PE files, resulting in diverse dataset. All the behavioural features of PE files can be extracted for creating larger feature set. By using executable files for different platforms, our detection system can be used as universal malware detection system.

References

[1] D. Gavriluţ, M. Cimpoeşu, D. Anton, and L. Ciortuz, "Malware detection using machine learning," in *2009 International Multiconference on Computer Science and Information Technology*, pp. 735–741, 2009.

[2] L. Liu, B.-s. Wang, B. Yu, and Q.-x. Zhong, "Automatic malware classification and new malware detection using machine learning," *Frontiers of Information Technology & Electronic Engineering*, vol. 18, no. 9, pp. 1336–1347, 2017.

[3] H. Rathore, S. Agarwal, S. K. Sahay, and M. Sewak, "Malware detection using machine learning and deep learning," in *International Conference on Big Data Analytics*, pp. 402–411, Springer, 2018.

[4] I. Firdausi, A. Erwin, A. S. Nugroho, *et al.*, "Analysis of machine learning techniques used in behavior-based malware detection," in *2010 second international conference on advances in computing, control, and telecommunication technologies*, pp. 201–203, IEEE, 2010.

[5] S. I. Bae, G. B. Lee, and E. G. Im, "Ransomware detection using machine learning algorithms," *Concurrency and Computation: Practice and Experience*, vol. 32, no. 18, p. e5422, 2020.

[6] E. J. Alqahtani, R. Zagrouba, and A. Almuhaideb, "A survey on android malware detection techniques using machine learning algorithms.," in *2019 Sixth International Conference on Software Defined Systems (SDS)*, pp. 110–117, IEEE, 2019.

[7] A. Kumar, K. Abhishek, K. Shah, D. Patel, Y. Jain, H. Chheda, and P. Nerurkar, "Malware detection using machine learning," in *Iberoamerican Knowledge Graphs and Semantic Web Conference*, pp. 61–71, Springer, 2020.

[8] F. A. Narudin, A. Feizollah, N. B. Anuar, and A. Gani, "Evaluation of machine learning classifiers for mobile malware detection," *Soft Computing*, vol. 20, no. 1, pp. 343–357, 2016.