**Malware Categorization using Machine Learning**

A dissertation submitted in partial fulfilment of the requirements for the degree of Bachelor of Science in Computer Science

In

**The Queen’s University of Belfast**

By

**Thomas Pickup**

# Declaration

Declaration

# Acknowledgements

Acknowledgements……

# Abstract

An important factor in risk assessment is categorisation of malware and its behaviour. It should be noted, a high number of new malware types does not necessarily imply high risk, as malware such as adware does not constitute a high risk. However, a low number of new signature variants does not indicate a low risk, as the new malware signature may relate to a rootkit. Malware programs are often categorised based on Propagation, infection mechanism, Self-Defence (concealment/evasion) or Payload (Criminal Software functionality).

When malware is correctly categorised, it enables an assessment of the risk associated with particular types of malware attacks, thereby enabling Security Operation Centres (SOC) to focus on the highest current threat. Many SOCs have adapted malware categorisation according to type, family and strain is a difficult task and may be impossible to achieve fully. The result is that 66 different AV scanners (VirusTotal) often produce different results, adding to the confusion and impact the ability to assess malware attacks. Therefore this investigates new methods of malware classification that will improve the ability to determine risk assessment of malware. A dynamic runtime dataset (PE file execution) will be mined using unsupervised/clustering algorithms to identify new methods of malware categorisation based on API call structure, which hopefully provides insight to malware risk assessment.

The project will involve:

* Study current publications about dynamic malware analysis techniques
* Establish a run-time environment that can be used to create a programme execution trace dataset (such as cuckoo)
* Write a parser to extract features from the dataset. A literature review is required to determine those features that may yield the best machine learning features.
* Use machine learning clustering algorithms to categorise malware into a cluster that correlates its: risk, family, structure, etc.
* The data mining should be repeated for multiple malware family/categories to determine the optimal category definition.
* Develop and implement an algorithm for measuring agreement/different between existing labels and the new label sets (novel labelling).

# Contents

[Declaration 2](#_Toc5126837)

[Acknowledgements 3](#_Toc5126838)

[Abstract 4](#_Toc5126839)

[Contents 5](#_Toc5126840)

[1.0 Introduction and Problem Area 7](#_Toc5126841)

[1.1 Introduction 7](#_Toc5126842)

[1.2 Malware Types 7](#_Toc5126843)

[1.2.1 Trojan 7](#_Toc5126844)

[1.2.2 Worm 8](#_Toc5126845)

[1.2.3 Ransomware 8](#_Toc5126846)

[1.2.4 Rootkit 8](#_Toc5126847)

[1.2.5 Keylogger 8](#_Toc5126848)

[1.3 Machine Learning 8](#_Toc5126849)

[1.4 Analysis 9](#_Toc5126850)

[1.4.1 Dynamic Analysis 9](#_Toc5126851)

[1.4.2 Static Analysis 9](#_Toc5126852)

[1.4.3 Cuckoo Environment 9](#_Toc5126853)

[2.0 Solution Description and System Requirements 9](#_Toc5126854)

[2.1 Solution Description 9](#_Toc5126855)

[2.2 System Requirements 9](#_Toc5126856)

[3.0 Design 9](#_Toc5126857)

[3.1 Cuckoo Environment 9](#_Toc5126858)

[3.2 Parser 10](#_Toc5126859)

[3.3 Machine Learning 10](#_Toc5126860)

[4.0 Implementation 10](#_Toc5126861)

[4.1 Use of Supporting Tools 11](#_Toc5126862)

[4.1.1 Languages Used 11](#_Toc5126863)

[4.1.2 Development Environment 11](#_Toc5126864)

[4.1.3 Version Control 11](#_Toc5126865)

[4.4 Use of Software Libraries 11](#_Toc5126866)

[4.5 Key Implementation Decisions 11](#_Toc5126867)

[4.6 Important Functions and Algorithms 11](#_Toc5126868)

[4.7 Description of How Each Component Was Implemented 11](#_Toc5126869)

[5.0 Testing 11](#_Toc5126870)

[5.1 Testing Approach 11](#_Toc5126871)

[5.2 Testing Results 11](#_Toc5126872)

[6.0 System Evaluation and Experimental Results 11](#_Toc5126873)

[6.1 System Evaluation 11](#_Toc5126874)

[6.2 Conclusion 11](#_Toc5126875)

[7.0 Appendices 12](#_Toc5126876)

[8.0 Table of Figures 13](#_Toc5126877)

[9.0 References 14](#_Toc5126878)

# 1.0 Introduction and Problem Area

## 1.1 Introduction

Malware: any malicious program or code that is harmful to computers.

With an expected 20 Billion Internet connected devices to come online by 2020 [1], protection against and identification of Malware is becoming more important by the day. The average cost of a malware attack on a company is $2.4million and the cost in time of a malware attack is 50 days [2].

It’s clear that Malware is becoming even more varied, with the NHS Ransomware attack spanning off multiple clones of the WannaCry virus, G Data Software estimates that in 2017 alone there was 7.41 million new malware specimens [3].

The cost and rate at which malware is growing, makes this field one of the most important in the Computer Science industry. Current methods of analysing malware are not proving useful for this ever changing field. Static Analysis is prevalent, but the major downside is that it doesn’t protect against zero day attacks or new strains. Even polymorphic code can fool malware detectors that rely upon Static Analysis.

Machine Learning algorithms using dynamic analysis provide a viable alternative to this limitation, by basing their result on the behaviour of the specimen, the model theoretically can predict not only whether it is benign or malicious but could also be used to classify what family of malware the specimen belongs to.

The goal of this report it to provide an in depth look into how we could use machine learning in the future to classify malware. The report will look into different methods of malware analysis techniques, it will then document the process of setting up a Cuckoo Sandbox environment that will allow us to analyse the behaviour of the specimen. This dataset will be used with a machine learning algorithm to predict what type of malware (or benign) a specimen is. This process will be repeated to determine the optimal category definition.

## 1.2 Malware Types

As this dissertation will focus on identifying malicious files, an enhanced overview of different types of Malware is required, in this section is a descriptive overview of the main types of Malware that are active today. Later on in this dissertation will be an overview of the specific malicious families that will be tested.

### 1.2.1 Trojan

A Trojan Horse, commonly referred to as a Trojan, is a virus that is often disguised as legitimate software. It’s called a Trojan due to the method of attack used by the Greeks in the Trojan war, the Greeks gifted a huge wooden horse that concealed an army. This parallels the attack method used by the Trojan Virus, the payload that would do the damage is hidden in a legitimate program [4].

A cyber criminal would often use social engineering to spread a Trojan Virus. Usually a victim would have to click on a fake link or email that would redirect the user to a webpage often designed in the style of the legitimate counterpart. Once the user downloads and launches the file, the Trojan may execute [5].

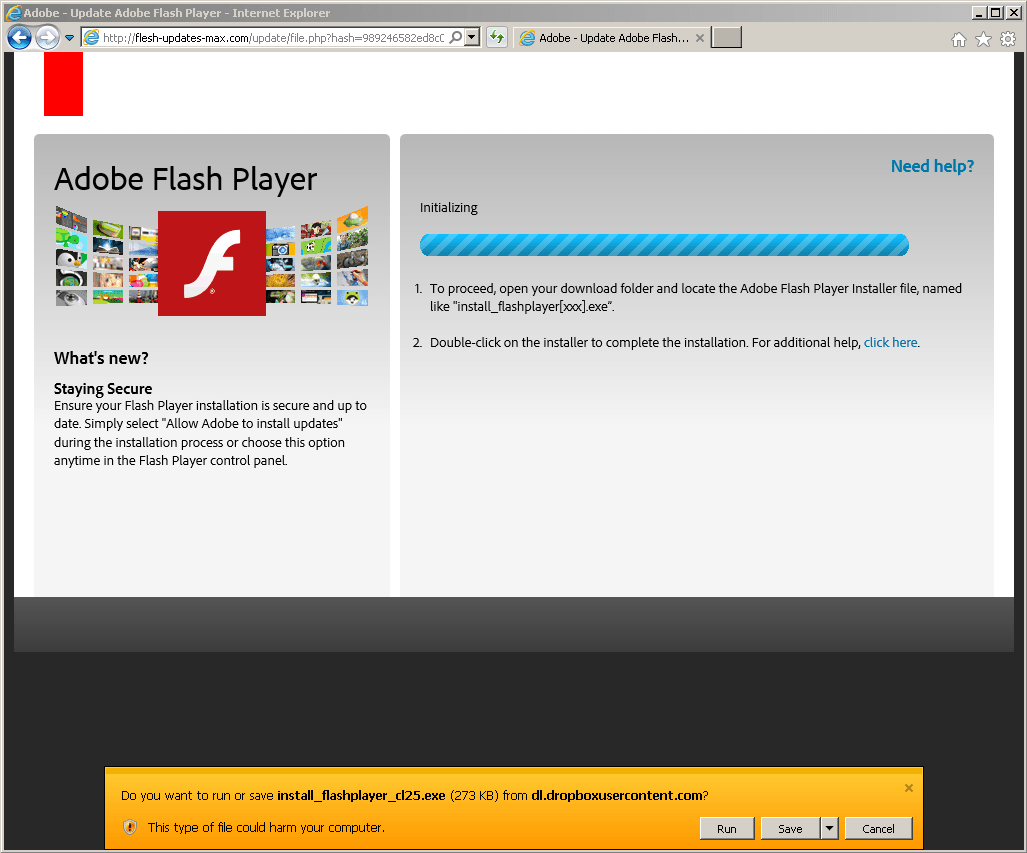


Figure 1. Example of Flash Player Trojan Webpage

In Figure 1above you can see a webpage that looks identical to the Adobe Flashplayer download page, however in this example the webpage is from a website called flesh-updates-max.com, this would be set up by the cyber criminal. This attack method would catch out most unsuspecting users.

The Trojan horse, when on your system can do many things, most would be designed not to alert the user to the Trojan Horses presence. For example Trojans can work as spyware, when they’re working like this, they would capture the users credit card details, passwords and other information that could be valuable to sell on or exploit. Another way a Trojan Horse can cause damage is by using your computer as part of a bot net, this can cause damage to other users, particularly when performing a Distributed Denial of Service attack. The DDoS attack would work with other computers and devices on the bot net to overload a targets network. The target of this attack could be anyone from a single person up to a multinational company, recent examples of this attack are the Boxing Day attacks on online game services such as Xbox Live and PlayStation Network. Being part of a bot net can put the user in great trouble as cyber-attacks are often detected through the originators IP address [5] [6].

### 1.2.2 Worm

### 1.2.3 Ransomware

### 1.2.4 Rootkit

### 1.2.5 Keylogger

## 1.3 Machine Learning

## 1.4 Analysis

### 1.4.1 Dynamic Analysis

### 1.4.2 Static Analysis

### 1.4.3 Cuckoo Environment

# 2.0 Solution Description and System Requirements

## 2.1 Solution Description

## 2.2 System Requirements

# 3.0 Design

There is three main parts to the system that will be developed. The first part is a cuckoo environment, this will take in an executable file and run it, making a report of the API Calls amongst other pieces of information. The next part of the system is a parser, this will take in the JSON file that has been created by the cuckoo environment and will extract features from this creating the training dataset. The final part of the system is the actual Machine Learning mind. It will be trained using the dataset created previously and will be able to take in a set of API Calls and predict the family of malware the executable belongs to.

## 3.1 Cuckoo Environment

The first part is the Cuckoo environment that will perform dynamic analysis on the specimens, this will output a JSON report.

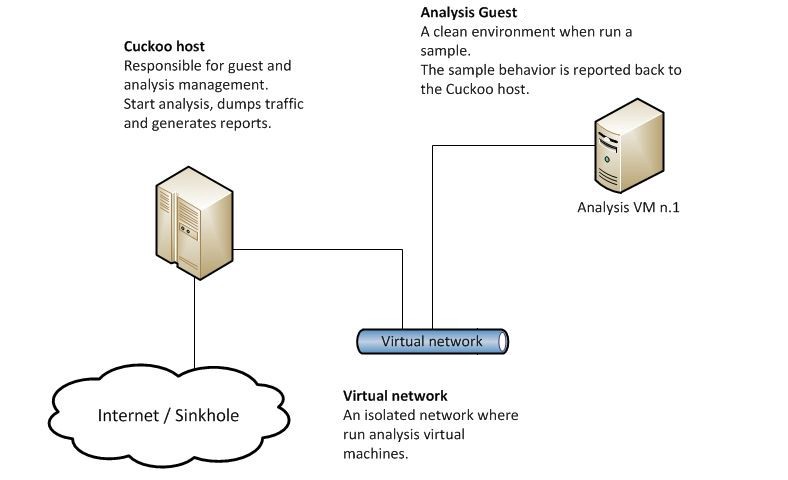


Figure 2. Example of Cuckoo Environment

The environment consists of an Ubuntu machine that has the Cuckoo software installed as well as virtual box by oracle. On this machine there will be a Windows 7 virtual machine.

The Windows 7 virtual machine will be used to run the specimen file. This will allow the Cuckoo Host to perform dynamic analysis, looking at what that file is doing when activated. After it has finished executing, the Cuckoo host will generate the JSON report.



Figure 3. An example of the JSON reports collection.

Out of this report we are interested in the API Calls. API stands for Application Programming Interface, these are functions in dll files that an executable would run to perform various system tasks.

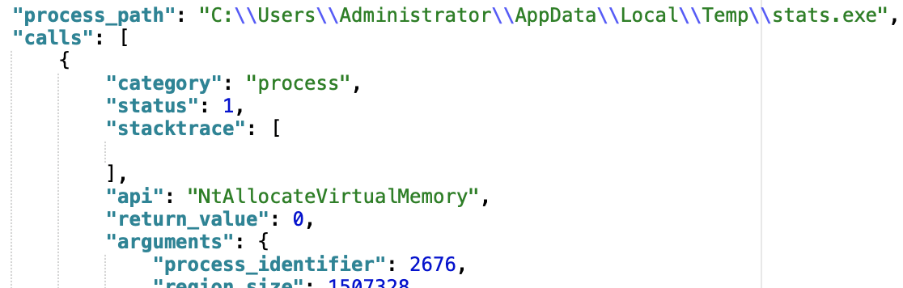


Figure 4. Example of API Call in JSON File

In the JSON report we can see a whole host of details about what the process, in this case a malicious file called “stats.exe” is trying to do on the computer. In this case it is running an API called NTAllocateVirtualMemory.

From there will we will parse the collection of JSON files into one. From there we will be able to apply a machine learning algorithm.

## 3.2 Parser

## 3.3 Machine Learning

# 4.0 Implementation

## 4.1 Use of Supporting Tools

For this project, I needed to use plenty of Supporting Tools and Languages to develop this project. Below is listed the tools and languages that were used in the making of this project as well as the development environment used to create the code.

### 4.1.1 Languages Used

For the development of this project, there was a need for both a scripting language to automate the different subsystems as well as a language aimed towards data science. For this purpose, Python and R were chosen.

I decided on Python as the scripting language because of its ease of use as well as the fact that it was built with scripting in mind. For the Machine Learning portion of the project, I chose R, this is because of its advantages when manipulating large datasets as well as the inbuilt libraries that are provided with R (4.4 Use of Software Libraries).

As a note, Python also contains great support from pre-existing libraries to be used for Machine Learning, however my experience with R in the CSC3060: Artificial Intelligence and Data Analytics module during my final year gave R a slight advantage over Python.

### 4.1.2 Development Environment

The project was developed on a Windows System that leverages Microsoft and Canonical’s recent partnership to run Ubuntu as a subsystem on Windows 10 [7]. This allowed for Cuckoo to be installed as part of this Ubuntu without the need for a separate Virtual Machine to become the Cuckoo Host. As the

For the development of the code, the Atom text editor was used. It is a text editor created by GitHub that has built in support for Python Syntax highlighting. As it was developed by GitHub it also has enhanced support for the Git Version Control System [8]. It was chosen for its simple layout as well as powerful enhancements.

### 4.1.3 Version Control

During the course of the practical part of the dissertation, the Queen’s EEECS Gitlab service was used to host a Git Repository containing the code used for this project. A private Github Repository was also used as a secondary backup location.

The use of Git meant that both myself and the supervisor could access the project’s codebase at any time. By using the commit system to upload incremental changes, it gave an itemised report of the features and changes that had been made, down to the line of code. This meant that in the event of a bug or issue, it was easy to investigate where and when the bug was created. In the event of a catastrophic failure, a full roll back could be carried out by using the Git system.

## 4.4 Use of Software Libraries

### 4.4.1 Python Libraries

#### 4.4.1.1 os

#### 4.4.1.2 datetime

#### 4.4.1.3 json

#### 4.4.1.4 csv

#### 4.4.1.5 sys

#### 4.4.1.6 shutil.copyfile

### 4.4.2 R Libraries

#### 4.4.2.1 caret

#### 4.4.2.2 e1071

#### 4.4.2.3 Boruta

## 4.5 Key Implementation Decisions

## 4.6 Important Functions and Algorithms

## 4.7 Description of How Each Component Was Implemented

# 5.0 Testing

## 5.1 Testing Approach

## 5.2 Testing Results

# 6.0 System Evaluation and Experimental Results

## 6.1 System Evaluation

## 6.2 Conclusion

# 7.0 Appendices

# Table of Figures

[Figure 1. Example of Flash Player Trojan Webpage 7](#_Toc5123681)

[Figure 2. Example of Cuckoo Environment 8](#_Toc5123682)

[Figure 3. An example of the JSON reports collection. 9](#_Toc5123683)

[Figure 4. Example of API Call in JSON File 9](#_Toc5123684)

# References

|  |  |
| --- | --- |
| [1] | Forbes, “Developing the Conected World of 2018 and Beyond,” Forbes, 16 March 2018. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2018/03/16/developing-the-connected-world-of-2018-and-beyond/#156cda811e51. [Accessed 02 April 2019]. |
| [2] | Varonis, “60 Must-Know Cybersecurity Statistics for 2019,” Varonis, 28 March 2019. [Online]. Available: https://www.varonis.com/blog/cybersecurity-statistics/. [Accessed 02 April 2019]. |
| [3] | G-Data Software, “Malware trends 2017,” G-Data Software, 10 April 2017. [Online]. Available: https://www.gdatasoftware.com/blog/2017/04/29666-malware-trends-2017. [Accessed 02 April 2019]. |
| [4] | Britannica, “Trojan Horse,” Britannica, 20 July 1998. [Online]. Available: https://www.britannica.com/topic/Trojan-horse. [Accessed 02 April 2019]. |
| [5] | Comodo Security, “Trojan Horse Definition,” Comodo Security, [Online]. Available: https://enterprise.comodo.com/trojan-horse-definition.php. [Accessed 02 April 2019]. |
| [6] | Comodo Security, “What is a Trojan Horse,” Comodo Security, [Online]. Available: https://enterprise.comodo.com/what-is-a-trojan-horse.php. [Accessed 02 April 2019]. |
| [7] | Kaspersky, “Zeus Virus,” Kaspersky, [Online]. Available: https://usa.kaspersky.com/resource-center/threats/zeus-virus. [Accessed 02 April 2019]. |
| [8] | MalwareBytes Labs, “Zbot with legitimate applications on board,” MalwareBytes, 27 January 2017. [Online]. Available: https://blog.malwarebytes.com/cybercrime/2017/01/zbot-with-legitimate-applications-on-board/. [Accessed 02 April 2019]. |
| [9] | MalwareBytes Labs, “Worm,” MalwareBytes, 09 June 2016. [Online]. Available: https://blog.malwarebytes.com/threats/worm/. [Accessed 02 April 2019]. |