Machine Learning Methods for MAC OS malware detection

Students Dolev Abuhazira , Bar Luzon
Advisor dr. Ran Dubin

1 Abstract

The Mac operating system is based on a UNIX-based platform and some consider it a more durable operating system compared to the Windows platform. Mac OS usage grew from a 6.7% market share in 2019 to 7.5% in 2020, while Windows suffered a sharp decline, falling from 85.4% in 2019 to 80.5% in 2020.
As the use of Mac OS operating systems increases, so does the number of different cyber attacks on that operating system. Using machine learning in a supervised learning approach makes it possible to distinguish and learn the differences between malicious files and those that are not.

2 Introduction

The increasing number of cyber attacks on MAC OS platforms highlights the importance of detecting and classifying malicious files. The detection of malware is a challenging problem due to the continuous evolution of malware techniques, which makes it difficult to detect and classify new malware types. Therefore, the need for efficient and accurate classification algorithms is essential for detecting malware in MAC OS platforms. In this project, we present a binary classification algorithm that can differentiate between benign and malicious Mach-O files.

3 Related Work

The Mac OS operating system is ranked as one of the most secure operating systems, but in recent years there has been a large increase in cyber attacks on the system. It is necessary to perform pre-processing to extract the capabilities that best describe the files. There is an approach suggested **[1]** to extract characters and attributes of the files from PE files which is a format for windows executables files.
Following this, another study performed static analysis **[2]** for Mac executables according to the Mach-O format, Which is a native binary executable format in Mac OS and used various machine learning methods to classify the files properly. various researchers proposed methods such as deep learning [**3**] which focuses on learning the characteristics of executable files such as Mach-O By turning them into Bitmap images and identifying patterns that describe malicious actions within the file.

4 Dataset

4.1 Dataset Collection

we collected the data set consisting of 303 Mach-O files. We downloaded 101 examples of normal files from the Apple store, and in addition, we downloaded another 202 malicious files from around the network. We chose to perform the classification on the Mach-O format files which describe these same run time files.

4.2 Mach-O file format

When code is compiled for use on Mac OS, this code is organized using the Mach object (or Mach-O) file format. An executable format determines the order in which code and data in a binary file are read into memory.
The Mach-O file format provides both intermediate (during the build process) and final (after linking the final product) storage of machine code and data. It was designed as a flexible replacement for the BSD a.out format, to be used by the compiler and the static linker and to contain statically linked executable code at run-time. Features for dynamic linking were added as the goals of OS X evolved, resulting in a single file format for both statically linked and dynamically linked code.

Mach-O file consists of three major regions as follows :

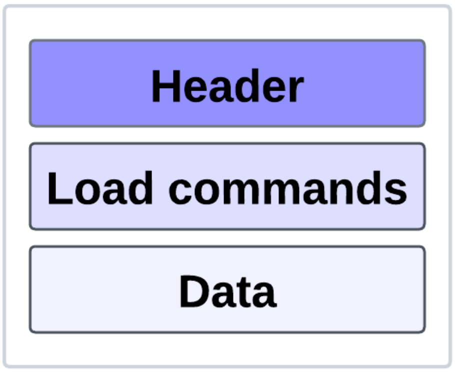

Figure 1: Mach-O structure

Header

contains information about the binary such as byte order (magic number), CPU type, and number of load commands

Load Commands

contains information about logical structure of an executable file and data stored in the virtual memory such as symbol table, dynamic symbol table, etc.

Segments

is the biggest part of each Mach-O file which contains application code and data, Each segment can contain multiple segments as well.

4.3 Data Preprocessing

The raw dataset was processed in several steps. First, the Mach-O files were parsed using the LIEF **[6]** library and converted into JSON files for easy feature extraction. Next, the JSON files were recorded in a DataFrame, with each column representing a feature of the Mach-O file. Since the dataset contained a lot of categorical variables, label encoding was used for most of them. However, for the list of libraries used by the file, the Word2Vec model was used to study their representation in space. The pre-processing step included feature engineering for the following features: libraries names, function start functions, header flags, and uuid.

4.4 Data Modeling

After preprocessing the data, we imputed the missing values by calculating the median of each column in the data set and performed scaling on the same data. The data set was not balanced, with 201 samples of malware and 101 samples of benign files. To balance the data set, we used the SMOTE technique to increase the number of benign files to approximately 402 samples. We then performed feature selection using the RFE algorithm to select the most important features. The RFE algorithm selected 40 features.

5 Results and summary

We trained the XGB model using default hyper-parameters and achieved an accuracy of 98.08% on the test set. In conclusion, our project demonstrated the effectiveness of using machine learning for executable file classification. Our innovation compared to previous works was when we used tools like Word2Vec to represent the names of the libraries in a better way, compared to those who used TF-IDF. Additionally, we chose to compress the entire file first, process it, and choose the appropriate features and not focus on features from the beginning specific as other jobs have done.s

6 References

[1] Data Mining Methods for Detection of New Malicious Executables
[2] Mac Malware Detection via Static File Structure Analysis
[3] Image Based Malware Classification with Multimodal Deep Learning
[4] Mac malware collection - a public collection of malicious code targeting Mac OS
[5] How to Inject Code into Mach-O Apps
[6] LIEF cross-platform library that can parse and modify executable formats