International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 07 Issue: 05 | May 2020 www.irjet.net p-ISSN: 2395-0072

Detecting Malware using Deep Learning Model

Shalini Mahalunge[1], Iftesam Shaikh[2], Vaishakhi Kabir[3], Diksha Tiwary[4], Prashant Lokhande[5]

[1,2,3,4]Student, Dept. .of Computer Engineering, Pillai College of Engineering, Maharashtra, India
[5]Professor, Dept. of Computer Engineering, Pillai College of Engineering, Maharashtra, India

---

Abstract - *This project demonstrates an approach to detect malware targeted at Microsoft windows. When we download any software or file from unknown links it may inject malware to the system. These malwares that originates from these sites may travel around the internet and land on an innocent users computer redirecting their browsing experience to these sites. So to prevent systems from such threats we are implementing a project which will detect the files which has malware. With the help of this project user can get prior knowledge of malware file and he will not execute it. This project demonstrates a novel approach to detecting ransomware targeted at Microsoft Windows, combining 2 deep learning neural network classifiers to create an ensemble, taking files as input in Microsoft's standard PE file format, such as those with a '.exe' file extension, and returning a prediction of the file belonging to 1 of 3 classes: benign, generic malware, or ransomware. The model's ability to distinguish between ransomware and other forms of malware allows it to be applied as an extension to an existing malware detection system such as anti-virus software.*

Key Words: **Ransomware, ensemble, benign, Malware.**

1. INTRODUCTION

This project aimed to detect ransom ware using an ensemble of ML (Machine Learning) classifiers, classifying Windows executable files as belonging to 1 of 3 possible classes: benign, malware, or ransom ware. The model can be used in practice to identify ransom ware on a user's machine, or can be integrated into an existing AV (Anti-Virus) system to facilitate or improve its ability to distinguish between ransom ware and other types of malware.

2. LITERATURE REVIEW

A. Application of ML for Detecting Malware

In this paper ANN (Artificial Neural Network)to detect viral infections of the boot sector of HDDs (Hard Disk Drives), reporting an 82% detection rate on test samples unseen in the model's training, though only training on a total of 195 samples, which may not be a large enough sample to accurately generalise these results to the full in- the-wild population of boot sector infections In 2001, Schultz et al[1]. compared various ML methods against each other and against a signature based technique in the accuracy of their binary classification of Windows executables as benign or malware.[1]

B. Malware Detector Using Naive Bayes

Their most successful method was an ensemble of Naive Bayes classifiers trained on the raw bytes of the executable files, which achieved a 97.76% malware detection rate, nearly 3 times more accurate than the signature method which was evaluated at just 33.75% Again the number of sample files used, only 3622, was likely not enough to accurately generalize these results to the full population of in-the-wild malware, estimated to be around 50,000 executables at the time (Microsoft, 2012, 26).[2]

C. Deep Android Malware Detection and Classification

Vinayakumar et al.[3] had similar success in using the frequency of API calls to train ML models, but found that a type of DLNN (Deep Learning Neural Network) called an MLP (Multi-Layer Perceptron) achieved greater accuracy than an SVM in both multi-class (98%) and binary (100%) classification of executable files as either benign or belonging to 1 of 7 different families of ransomware. This study however, like many recent investigations into ML for ransomware detection, used a small sample, just 974 executable files.[3]

D. Deep Android Malware Detection

Due to security concerns and time requirements this project takes a static approach to malware analysis, as opposed to a dynamic approach. Static malware analysis uses features from within the contents of executable files, whereas dynamic analysis relies on recording the behaviours of software during its

execution, typically within a contained environment such as a sandbox or virtual machine. There are 4 features that can be seen frequently being extracted from Windows executables within research into automating static analysis of malware using ML: bytes, opcodes, strings, and metadata encoded in PE files.[4]

3. PROPOSED WORK

This project aimed to detect ransom ware using an ensemble of ML (Machine Learning) classifiers, classifying Windows executable files as belonging to 1 of 3 possible classes: benign, malware, or ransom ware. The hope is that by releasing source code to the public, the model can be used in practice to identify ransom ware on a user's machine, or can be integrated into an existing AV (Anti-Virus) system to facilitate or improve its ability to distinguish between ransom ware and other types of malware.

3.1 System Architecture

The system architecture is given in Figure 1. Each block is described in this Section.

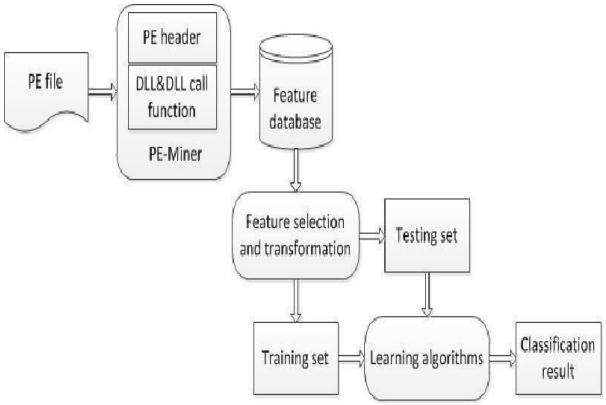

Fig. 1 Proposed system architecture

A. PE File: The **Portable Executable (PE)** format may be a file format for executable, code, DLLs, FON Font files, et al. utilized in 32-bit and 64-bit versions of Windows operating systems. The PE format may be an arrangement that encapsulates the knowledge 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 employed for EXE, DLL, SYS (device driver), and other file types. The Extensible Firmware Interface (EFI) specification states that PE is that the standard executable format in EFI environment

B. PE header: The following list describes the Microsoft PE executable format, with the base of the image header at the top. The section from the MS-DOS 2.0 Compatible EXE Header through to the unused section just before the PE header is that the MS-DOS 2.0 Section, and is employed for MS-DOS compatibility only.

- MS-DOS 2.0 Compatible EXE Header
- unused
- OEM Identifier
- OEM Information
- Offset to PE Header
- PE Header (aligned on 8-byte boundary)
- Section Headers

C. Learning Algorithms: Deep learning uses layers of neural- network algorithms to decipher higher-level data at alternative layers supported raw input file. Neural networks will facilitate cluster points at intervals an oversized sample of knowledge supported the similarities of its options, classify knowledge supported labels from previous knowledge, and extract distinct options from knowledge. The numerical patterns these networks acknowledge are kept in vectors that depict real-world inputs. Deep neural networks may be thought of as elements of larger machine- learning applications involving algorithms for reinforcement learning, classification, and regression. Deep learning uses self-taught learning and algorithmic program constructs with several hidden layers, big data, and powerful machine resources. The recursive framework is named the neural network, whereas the hidden layers within the network provides it the appellation of deep learning. The Google Brain Team project and deep learning computer code like TensorFlow have

given more traction to the event of deep learning techniques. Such techniques are supported mathematical functions and parameters for achieving the specified output.

D. Classification results: The deciliter ensemble model developed during this project automates static analysis and classification of Windows feasible files as either benign, malware, or ransom ware, with a high degree of accuracy. The model is incontestable to accurately classify 3000 files unseen in its coaching, suggesting generalizability to sleuthing malware and ransom ware within the

wild. For end-users of an easy terminal program developed for ransom ware detection as a part of this project, it's vital to the safety of their or their organizations devices that the excellence between benign and malicious files is correct, and then this was tested to be ninety eight correct on the set of 3000 unseen samples. Jewish calendar month developers and security researchers will use, and re-use the ensemble model in their own systems, and retrain the models with a bigger dataset and newer samples to take care of its ability to classify in-the-wild executable.

4. REQUIREMENT ANALYSIS

The implementation detail is given in this section.

4.1 Software

The minimal software requirement is given in Table

1. Table 1 minimal Software requirement

Operating System	Linux/Windows
Programming Language	Python

4.2 Hardware

The minimal Hardware requirement is given in Table

2. Table 2 minimal Hardware requirement

Processor	Pentium IV & above
HDD	250 GB
RAM	1 GB

5. IMPLEMENTATION

Obtain samples of ransom ware, other malware, and benign Windows Portable Executable (PE) files. Labeled by multiple antiviruses. Gathering at least 10,000 samples of each class, though more will be required if this volume proves too small to train an accurate machine learning (ML) model. Extract UTF-
8 strings encoded in the binary contents of the sample files and build a numerical model of the strings extracted from all training samples. The same process will be applied to the opposes of assembly code disassembled from the binary contents of the executable files, as well as imported library names and exported function / library names extracted from the import and export sections of the PE files. Train a variety of machine learning classification models on a subset of the sample data and compare their accuracy in labeling another subset of the sample data that the models are not shown in training, finding the most accurate models for classifying the unseen sample data. Implement the machine learning classification model in a user- friendly, GUI featured, compiled executable program that allows users to select and analyses multiple files. This program will report back the model's estimated probabilities for the files' being ransom ware; other forms of malware, or benign software, and will allow the user to delete these files and report them to a popular public malware repository. Evaluate the accuracy, usability, and source disk encryption.applicability of the solution to real-world problems, and compare it to similar existing solutions.

5.1 Reflective Analysis:

Discovery of model architectures and data representations which would lead to high validation accuracy was far more time consuming and challenging than anticipated, with many approaches not even achieving training accuracy better than the ~33% baseline achieved by random guessing. It challenged me to learn and practice many new skills in data mining, pre-processing, and development and testing of DLNNs. Once the architectures had

been designed and implemented that achieved high validation accuracy, the results of their performance in classifying unseen test data were surprisingly impressive, at 96% test accuracy for the ensemble model, considering their ability to compete with contemporary research in the field which is often performed by teams of academics. And the test accuracy was even more impressive when the ensemble model was adapted to perform binary classification of executable as either benign or malicious, calculated at 98%, which is the most critical distinction for end-users of the model who are using it to decide whether or not a Windows program is safe to execute.

6. CONCLUSION

So we are implementing the deep learning model which will detect if there is malware in selected file.

For that we have to download an .exe file or we can try with existing file also by uploading it.

We have used a deep learning model to verify among the .exe files.

Convolution neural network is one of the deep learning model we have used.

7. REFERENCES

[1] Kephart et al (1995)"Application of ML for detecting malware".

[2] Schultz et al (2001, 10)"Malware Detector using Naive Bayes".

[3] Vinayakumar et al, K.P. Soman, Prabaharan Poornachandran(2017)"Deep android malware detection and classification".

[4] Shabtai et al,Uri Kanonov, Yuval Elovici (2009)"Deep android malware detection and classification".

[5] N Scaife, H Carter, P Traynor,2016, Cryptolock(and drop it):stopping ransomware attacks on user data(2016).

[6] MQ Li, B Fung, P Charland, SHH Ding, 2019, "A Novel Interpretable Malware Detector Using Hirarchical Transformer".

[7] S Mohurle, M Patil, 2017, A brief study of wannacry threat:Ransomware attack (2017).

[8] SM Reasor, AJ Newman, RA Franczyk, J Garms, 2010 driven Malware Detector(2010).

[9] Shabtai et al, Uri Kapoor, Yuval Elovici, 2009, A Behavioral Malware Detection Framework for Android Devices(2009).

[10] Nwokedi Idika, Aditya Mathur 2007, Malware Detection Technique(2007).