IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

Malware Detection and Classification Framework for IOT Devices

Sayali Khirid[1], Sakshi Veer[2], Tanushika Gupta[3], Vishwajeet Waychal[4], Mrs. Asmita R. Kamble[5]
Students, Department of Computer Engineering[1]
Professor, Department of Computer Engineering[2,3,4,5]
Sinhgad Institute of Technology and Science, Pune, Maharashtra, India

Abstract: *Internet of Things (IoT) technology provides the basic infrastructure for a hyper connected society where all things are connected and exchange information through the Internet. IoT technology is fused with 5G and artificial intelligence (AI) technologies for use various fields such as the smart city and smart factory. As the demand for IoT technology increases, security threats against IoT infrastructure, applications, and devices have also increased. A variety of studies have been conducted on the detection of IoT malware to avoid the threats posed by malicious code. While existing models may accurately detect malicious IoT code identified through static analysis, detecting the new and variant IoT malware quickly being generated may become challenging. Due to the complexity of design and implementation in both hardware and software, as well as the lack of security functions and abilities, IoT devices are becoming an attractive target for cyber criminals who take advantage of weak authentication, outdated firmware's , and malwares to compromise IoT devices .This project provides the light on the system named as malware classification and detection of IOT devices, used to detect the cyber-attacks caused by malware on IOT devices by using machine learning techniques. The malware classification and detection system detect and identifies the various types of malwares using static analysis with the help of machine learning algorithm. An easy-to-use user interface for easy uploading of files and checking for virus is designed. Also, acceptance testing is performed on the application to remove vulnerabilities.*

Keywords: Internet of Things, Malware, Malware Classification, Static Analysis.

I. INTRODUCTION

Internet of Things (IoT) occurs to be the largest digital mega-trend that bridges physical and virtual worlds. The increment within the connectivity of humans, objects, machines and the Internet is directing to the emergence of latest business models as well as new interactions between mankind. Due to the complexity of design and implementation in both hardware and software, along with the lack of security functions and capabilities, IoT devices have become an attractive target for cyber criminals who benefit from weak authentication, outdated firmware's, and malwares to compromise IoT devices. There are two fundamental approaches to malware analysis. Static analysis and dynamic analysis. We've used Static analysis; it consists of examining the executable file without viewing the existing instructions. It is used in order to confirm or get an idea of whether the file being inspected is malicious or not. We achieve this by figuring out the functions and libraries that are being called by the executable. A basic static analysis does act as a steppingstone for the rest of the malware analysis and offers a thought about things one should be looking into. There are mainly two types ofsignatures based and behaviour-based detection. We've used signature-based detection, which is the most commonly used detection method. The process uses known patterns to detect malware. These patterns are loaded from an interior database. The process is able to detect malware fast, but it cannot detect new forms of threats if they are not stored within the interior database. Signature-based identification methods have become popular due to their performance and stability in commercial systems until now. Therefore, the best file format is PE Header, it is the commonly used file format because of the wide use of Windows operating system. A PE File is a data framework that contains the data necessary for the Windows OS loader to manage the wrapped executable code. There are no mandatory constraints in many fields of PE files and contain many redundant fields and spaces, creating opportunities for malware propagation and malware attacks. A PE File contains the PE file header, section table and section data. They have many valuable pieces of data for malware analysts, including imports, exports, time-date stamps, subsystems, sections and

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

resources. Hence, we propose an answer to utilize the strengths of static analysis to maximise their chances of detecting malware by using machine learning.

II. Related Work

[1], have stated that, the DAIMD scheme learns IoT malware using the convolution neural network (CNN) model and analyses IoT malware dynamically in nested cloud environment. It was also observed that the DAIMD performs dynamic analysis on IoT malware in a nested cloud environment to extract behaviours related to memory, network, virtual file system, process, and system call. By converting the extracted and analysed behaviour data into images, the behaviour images of IoT malware were classified and trained in the Convolution Neural Network (CNN). DAIMD can minimize the infection damage of IoT devices from malware by visualizing and learning the vast amount of behaviour data generated through dynamic analysis.

[2],[7],[8],[11] have used deep learning methods for classification of malware. [2] they proposed methods to resample the raw bytecodes of the classes.dex files of Android applications as input to deep learning models. They have used two deep learning models, DexCNN and DexCRNN, to train the pre-processed sequences. These models are trained and evaluated in a dataset containing 8000 benign applications and 8000 malicious applications. Experiments show that the proposed methods can achieve 93.4% and 95.8% detection accuracy respectively. [7] have used deep learning-based feature detector and use its results to classify the Android and IoT samples either as Benign or Malware. In order to train Deep Models, they have gathered Android and IoT samples from various sources. Samples are decompiled and opcode are extracted from them. They feed the binary opcodes to the chars2vecmodel for embedding. The embeddings received are used as input to the feature detector. The feature learned by the detector are classified via a fully connected SoftMax network and a LSTM network. Findings reveal that the proposed feature detector achieves significant results with an F1-Score of 98.97% and an accuracy of 98%. [8] have proposed a novel behaviour-based deep learning framework (BDLF) which is built in cloud platform for detecting malware in IoT environment. In the proposed BDLF, they first construct behaviour graphs to provide efficient information of malware behaviours using extracted API calls. They then use a neural network-Stacked Autoencoders (SAEs) for extracting high-level features from behaviour graphs. The layers of SAEs are inserted one after another and the last layer is connected to some added classifiers. The architecture of the SAEs is 6,000-2,000-500. The experiment results demonstrate that the proposed BDLF can learn the semantics of higher-level malicious behaviours from behaviour graphs and further increase the average detection precision by 1.5%. [11] consists of two modules, colour image transformation and DCNN model. End malware image contains patterns in arbitrary form, the use of colour images produces better results with deep learning algorithms. Malware binary file is transformed into colour image. In-depth analysis is done using DCNN model. The performance of model is increased with optimization of convolutional kernel width, the number of hidden units and learning rate. Pooling layers reduce the amount of data by holding meaningful information. After this dense layer converts the 2-D features into 1-D features and further supplies them for classification. Additional, fine tuning number of neurons with activation function and learning error rate in different layers also increase the classification performance. Finally, the classifier identifies the images as malware or benign.

Danish Vasan et al [3], used the unique MTHAEL model using stacked ensemble of heterogeneous feature selection algorithms and state-of-the-art neural networks to learn different levels of semantic features demonstrates enhanced IoT malware detection than existing approaches. MTHAEL is the first of its kind that effectively optimizes recurrent neural network (RNN) and convolutional neural network (CNN) with high classification accuracy and consistently low computational overheads on different IoT architectures. Cross-architecture benchmarking was performed during the training with different architectures such as ARM, Intel80386, MIPS, and MIPS+Intel80386 individually. Two different hardware architectures were employed to analyse the architecture overhead, namely Raspberry Pi 4 (ARM-based architecture) and Core-i5 (Intel-based architecture). The proposed MTHAEL was evaluated comprehensively with a large IoT cross-architecture dataset of 21,137 samples and has achieved 99.98 percent classification accuracy for ARM architecture samples, surpassing prior related works. Overall, MTHAEL has demonstrated practical suitability for cross-architecture IoT malware detection with low computational overheads requiring only 0.32 seconds to detect any IoT malware.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

N. Moses Babu et. al [4], have proposed the system design for detection of malware for multi cloud servers using a intermediate monitoring server, any file transmission to multi clouds should be scanned by intermediate server, providing scanning, detection and removal of malware before transmitting to cloud servers. The Proposed Malware Detection for Multi clouds provides an inter -place routing primarily based on each probabilistic and deterministic forwarding mechanisms, the proposed malware provides tolerant networking, discussing the main necessities and viable solutions, and outlining the open research issues for detection and removal of malware in multi clouds using intermediate server.

Abhijit Yewale et. al [5], have modelled a new method to detect malwares based on the frequency of opcodes in the portable executable file. It was identified that; Opcode frequency can be used to detect the unknown malwares. They found 20 most frequent opcodes can be used as feature vector for machine learning classifier. The dataset for good wares and malwares were containing 20 most frequent Opcode with their frequency. By using their dataset, they have constructed four models which are SVM, RF, BOOST and Decision Tree. Out of four models Random Forest has provided 97% accuracy and zero per cent false positive ratio.

S. Muthurajkumar et. al [6], have put forward the method to detect malware infected files while transmitting the files from server to client and to provide a secure way to transfer files among users The system developed in this project is more efficient than the existing systems as it takes less time since the malwares are blocked in the router itself. Therefore, the client system does not need to carry out any detection and saves time.

A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting was discussed by Hamed Haddad Pajouh et. al. [9] in the year 2018. This paper explores the potential of using Recurrent Neural Network (RNN) deep learning in detecting IoT malware. Specifically, the approach uses RNN to analyse ARM-based IoT applications' execution operation codes. To train the models, we use an IoT application dataset comprising 281 malware and 270 benign wares. Then, they evaluate the trained model using 100 new IoT malware samples (i.e. not previously exposed to the model) with three different Long Short-Term Memory (LSTM) configurations. Findings of validation analysis show that the second configuration with 2-layer neurons has the highest accuracy (98.18%). A comparative summary demonstrates that the LSTM approach delivers the best possible outcome.

Quoc-Dung Ngo et. al. [10] have dealt with IoT malware detection methods which can be divided into two groups: non-graph-based and graph-based methods. The non-graph-based methods can achieve a good result when detecting "simple" and "forthright" malware without customization or obfuscation, but potentially lose accuracy when detecting unseen malware. In opposite, the graph-based methods show advantages when analysing the control flow of IoT malware, thus have the potential to accurately detect unseen or complicated malicious code despite the complexity of these methods. The methods of using non-graph-based features have achieved better results than using a graph-based features method in detecting IoT botnet, both in terms of accuracy and time cost complexity.

Hayate Takase et. al [12] have proposed a malware detection mechanism using the processor information obtained from a CPU. They have implemented a prototype using a virtual machine and evaluated their proposed mechanism. From the evaluation results, it was found that they can detect malware variants including packed malware by training one by one from each malware family.

III. Malware Detection Overview

3.1 Static Analysis

Static malware is malware at rest. Static malware is the process of extracting information from malware while it is not running. In static malware analysis (also called code analysis), the program is analyzed without executing it, and reverse engineering is performed using different tools like debugger, disassemble, decompile, etc. It is the easiest and least risky process. There is no risk of an infection occurring while analysis is taking place.

3.2 Signature Based Detection

It's the most commonly used detection method. The method uses known patterns to detect malware. These patterns are loaded from an internal database. The method is able to detect malware fast, but it cannot detect new kinds of threats if they are not stored in the internal database. This is the reason, why antivirus programs must upgrade their internal databases. Methods also cannot effectively deal with malware obfuscation.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

3.3 PE Format

Portable Executable (PE) is a format defining a form and a section layout of the executable files. PE is a data structure in the binary form that is used by Windows system for exe files, dll libraries and other executable formats. PE keeps the file description information which is used by Windows OS loader and also keeps the program code itself. PE format and blocks are described below:

A. DOS MZ Header

Defined as _IMAGE_DOS_HEADER_ structure. The first 64 bytes of the file is a header guarantee of DOS compatibility, and it is included even in current applications. The first value of the header is named as e_magic_ (so called magic number) and it is used to identify the DOS mode. The value must be always equal to 0x54AD ("MZ" in ASCII). The header also contains the e_lfanew value which is a relative offset to the PE header.

B. DOS Stub

This is a section for the DOS code itself. Nowadays, it is compiled only to show the message: “This program cannot be run in DOS mode.”

C. PE Header

A structure also defined as “IMAGE_NT_HEADERS” containing the signature, “IMAGE_FILE_HEADER” structure and “IMAGE_OPTIONAL_HEADER” structure. It is the main header for the Windows executables and it consists of a variety of fields placed in “signature”, “COFF header” and “PE Optional Header” structures. The description of the fields itself exceeds the scope of this paper but in the research, the “signature” value, “machine” value, “magic” value and “AddressOfEntryPoint” value are the important ones:

PE _signature_ value has the similar meaning as “e_magic” value from the DOS MZ header. The value is equal to 0x50450000 ("PE n0n0" in ASCII) and it is used for the identification of PE header. The machine value holding the type of CPU the code is compiled for (the value is used for compatibility check). “Magic” field is a 2-byte value placed at the beginning of the optional header and representing the architecture type (0x010B for PE32, 0x020B for PE64, 0x0107 ROM). _AddressOfEntryPoint_ is an address of the application entry point (address where the applications code begins).

D. Section Table

Defined in “IMAGE_SECTION_HEADER” structure. Each section has its own section header and these headers are used to describe each of the following sections. The headers contain section name, relative address, section size and other values. The number of sections and their names can be different because the com- piler controls these sections and names.

E. Sections

The sections contain data created by compiler, code itself and metadata corresponding with the code. Sections names are controlled by a compiler, but the most commonly used names are:

- .text = section containing the main executable code.
- rdata = read-only data that is globally accessible within the program.
- .data = global data of the program.
- .idata = stores the information about the import functions, if this section missing, data can be stored in the .rdata. .edata = stores the information about the export functions, if this section missing, data can be stored in .rdata section; .pdata = exception handling for 64-bit architecture.
- .rsrc = stores various resources.
- .reloc = information for relocation of libraries.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

The following picture shows the layout and the structures of PE file format:

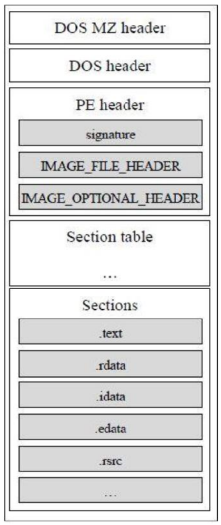

Figure 1: Structure of PE Format

IV. SYSTEM ARCHITECTURE

System architecture is the conceptual model that defines the structure, behaviour and more views of a system. The system architecture of our project is as shown in fig 2,

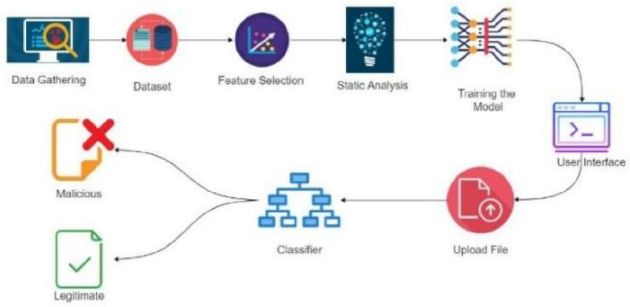

Figure 2: System Architecture

We have a dataset of different (.exe) and (.dll) files with the available information about that file memory Header information. The dataset will first undergo static analysis where features will be extracted. Later feature selection will be performed, and model will be trained using machine learning algorithm having highest accuracy among Random Forest, Ada-boost and Decision trees. Then the user will upload an .exe file using the designed UI and on starting detection the file will be passed through the classifier and classified as malicious or legitimate.

V. IMPLEMENTATION

5.1 Data Acquisition

The dataset is collected from VirusShare.com, which has total 138,047 files out of which 41323 files are legitimate and 96724 are malicious. It contains 57 features. All are (.exe) and (.dll) files with the available information about that file Memory Header information, Size of code, ImageSize, etc. These act as 'features' for our ML Models.

5.2 Feature Selection

The dataset is first loaded, and the feature selection is performed using extra trees classifier for selecting the best features out of total 57 features for accurate classification of malware files. As a result, 14 features were obtained as the important ones and saved into a (.pkl) file.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

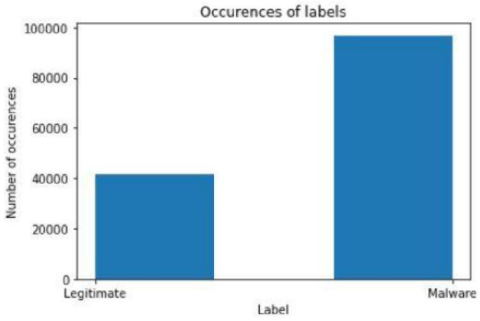

Figure 3: Number of Legitimate and Malware files

5.3 Machine Learning Model

We used three machine learning methods Random Forest, Decision Trees and Adaboost to classify a file as malicious/legitimate. The algorithm which was able to distinguish the malware file with lowest error rate and maximum accuracy was selected and used as a final model. We got the following accuracy from the three methods used:

- Decision Tree : 99.014832 % (Overfitting)
- Random Forest : 99.319075 % (Best)
- AdaBoost : 98.422644 % (Good)

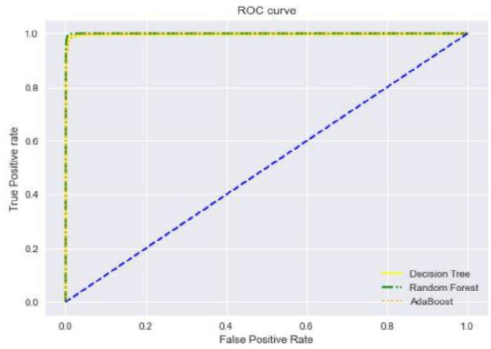

Figure 4: ROC curve for Algorithms.

We found that Random Forest has the best accuracy and used it for classifying an unknown file uploaded through our application. Decision Trees, though they have a better accuracy than Adaboost, are overfitting the train data. Adaboost could be used after Random Forest for malware file classification.

5.4 Static Analysis

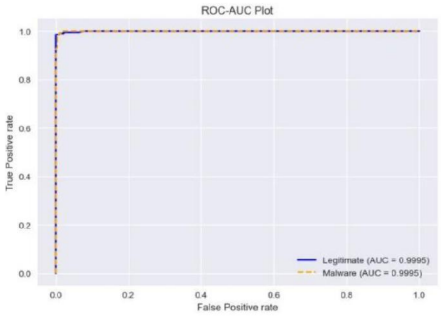

Figure 5: ROC curve for Legitimate and Malicious file.

The dataset was split into training set and testing set out of which 80% was for training and 20% was for testing. The machine learning model was trained using the training dataset and later it was tested using an unknown (.exe) and (.dll)

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

file. When the file is uploaded by the user, the features are extracted from the PE header of the file then the entropy values for each feature are calculated according to the PE header information of the file. Later on, those features are passed to the Random Forest classification model and the file is predicted as legitimate or malicious.

5.5 Malware Report

Malware report is generated by analysing a PE file and find out whether anything suspicious exists. It gives information about file such as its MD5, SHA1, Timestamp. It also checks PEiD Signature, Section, Imports, Exports, Resources and TLS Call-backs Overview done by the file.

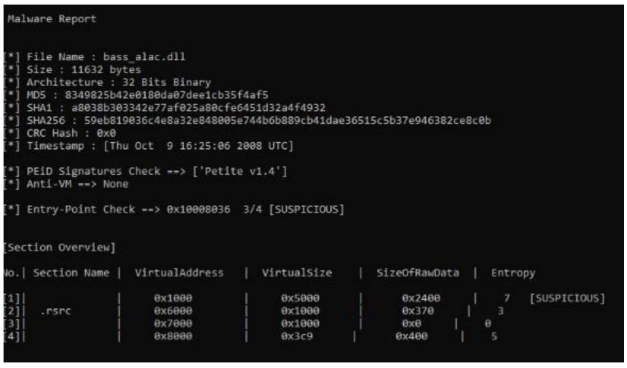

Figure 6: Malware Report

VI. Future Work

In the future, more static characteristics (like control flow graphs) and perform feature selection using more different methods (like chi-square distribution, etc) can be included. Dynamic characteristics can be used along with static characteristics to detect even more complex malware types. A website can be created and hosted on web for real time analysis of files on the cloud.

VII. Conclusion

A Malware Detection tool has been created where the user will upload the (.exe) or (.dll) file and can detect whether the file is malicious or legitimate. For the same we have used extra trees classifier for performing feature selection on the dataset to select the important features needed for classification. Along with it we have used Random Forest algorithm as it has the highest accuracy among the other two algorithms which are Decision Trees and AdaBoost. We got 99.31% accuracy with Random Forest. The true positive rate is 98.81% and true negative rate is 99.53%. We have also made an easy-to-use user interface for easy uploading of files and checking for virus. With this accuracy we were able to successfully detect malicious or legitimate file. With the help of this the user's system will be safe from the viruses spreading across the computer. This tool is fully ready and has been tested for all the corner cases. It is easy to use and very intuitive.

Acknowledgment

Finally, we present our gratitude towards our institute, 'Sinhgad Institute of Technology and Science' for guiding and helping us with every aspect to complete this proposed work. Also, we would like to thank 'Savitribai Phule Pune University' for giving us this opportunity to present our ideas and creativity through this overall work. It wouldn't have been possible without their support.

References

[1]. Jueun Jeon 1 , Jong hyuk park 2 , (member, IEEE), and Young-Sik Jeong , ''Dynamic Analysis for IoT Malware Detection With Convolution Neural Network Model ,'' 2020.

[2]. Zhongru Ren a , Haomin Wu d , Qian Ning c ,Iftikhar Hussain e , Bingcai Chen, "End-to-end malware detection for android IoT devices using deep Learning ," 2019.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 4, May 2022
Impact Factor: 6.252

[3]. Danish Vasan, Mamoun Alazeb, Sitalakshmi Venkatraman, Junaid Akram, Zheng Qin, "MTHAEL: Cross-Architecture IoT Malware Detection Based on Neural Network Advanced Ensemble Learning," 2020.

[4]. N. Moses Babu , Qian Ningc, "Malware Detection for Multi Cloud Servers using Intermediate Monitoring Server," 2019.

[5]. Abhijit Yewale, Maninder Singh, "Malware Detection Based On Opcode Frequency," 2020.

[6]. S. Muthurajkumar, M. Vijayalakshmi, S. Ganapathy, A. Kannan, "Agent Based Intelligent Approach for the Malware Detection for Infected Cloud Data Storage Files," 2019.

[7]. Muhammad Amin, Duri Shehwar, Abrar Ullah, Teresa Guarda, Tamleek Ali Tanveer, "A deep learning system for health care IoT and smartphone malware detection," 2020

[8]. Fei Xiao , Zhaowen Lin ,Yi Sun ,Yan Ma, "Malware Detection Based on Deep Learning of Behavior Graphs," 2019

[9]. Hamed HaddadPajouh , Ali Dehghantanha , Raouf Khayami , Kim-Kwang Raymond Choo, " A deep Recurrent Neural Network based approach for Internet of Things malware threat hunting," 2018.

[10]. Quoc-Dung Ngo, Huy-Trung Nguyen, Van-Hoang Le, Doan-Hieu Nguyen, "A survey of IoT malware and detection methods based on static features," 2019.

[11]. Hamad Naeem, Farhan Ullah, Muhammad Rashid Naeem, Shehzad Khalid, Danish Vasan, Sohail Jabbar, Saqib Saeed, "Malware detection in industrial internet of things based on hybrid image visualization and deep learning model," 2020.

[12]. Hayate Takase, Ryotaro Kobayashi,Masahiko Kato,Ren Ohmura, "A prototype implementation and evaluation of the malware detection mechanism for IoT devices using the processor information," 2019.