EasyChair Preprint

№ 9838

Vigorous Malware Detection in IoT Devices Using Machine Learning

T Renny, Sk Khadar Baba, P Abhilash Reddy and P Deepanth

March 7, 2023

Research Article(Open Access)

VIGOROUS MALWARE DETECTION IN IOT DEVICES USING MACHINE LEARNING

Faculty: Mrs. S. Gayathri Devi Team details: 1. T.Renny, 2. Sk .Khadar Baba,

3.P.Abhilash Reddy, 4. P. Deepanth Reddy

Department of Computer Science and Engineering, J.B. Institute of Engineering and Technology, Hyderabad

*__Corresponding Author:__ Email Id: deepanthreddy222@gmail.com

ABSTRACT:

The use of Internet of Things (IoT) devices has grown significantly due to the expansion of the internet.

However, these devices now contain large amounts of data, making them vulnerable to malware attacks. As a result, detecting malware in IoT devices has become a critical issue. Although many researchers have proposed various methods, accurately identifying advanced malware still poses a challenge.

To tackle this problem, we suggest a deep learning-based ensemble classification method for identifying malware in IoT devices. Our method comprises three steps:

(1) preprocessing the data using scaling, normalization, and de-noising,

(2) selecting features and applying one-hot encoding, and

(3) using an ensemble classifier that combines convolutional neural network (CNN) and long short-term memory (LSTM) outputs for malware detection.

We have evaluated our proposed method using standard datasets and compared it to state-of-the-art techniques. Our approach outperforms existing techniques and achieved an average accuracy of 99.5%.

I. INTRODUCTION

The Junk code injection attack is a type of malware anti-forensic technique that aims to evade OpCode inspection by inserting benign OpCode sequences or instructions like NOP that have no impact on malware activities. This technique obfuscates malicious OpCode sequences and reduces the proportion of malicious OpCodes in the malware.

In our proposed approach, we utilize an affinity-based criteria to counteract the effects of junk OpCode injection anti-forensics technique. Our feature selection method removes less informative OpCodes to mitigate the impact of injecting junk OpCodes.

To demonstrate the effectiveness of our approach against code insertion attacks, we randomly selected a specified proportion of elements (5%, 10%, 15%, 20%, 25%, and 30%) in each sample's generated graph and increment their value by one in an iterative manner. This simulates the injection of an OpCode next to the OpCodei in a sample's instruction sequence to deceive the detection algorithm. Algorithm 2 outlines the iteration of junk code insertion during the experiments, which is repeated for each iteration of k-fold validation.

We evaluated the robustness of our approach and compared it against existing proposals by applying two congruent algorithms described in Section 1 on our generated dataset using Adaboost as the classification algorithm

II.MATERIALS AND METHODS

ALGORITHM:

N-Gram sequence:

The Junk Code Insertion Procedure is an algorithm used in malware detection to simulate the effect of junk code insertion on a trained classifier. The input consists of a trained classifier D, a set of test samples S, and a percentage k representing the amount of junk code to be inserted. The output is the predicted class for the test samples P.

The algorithm works as follows:

1)For each test sample s in S, generate a graph representing the opcode sequence of the sample.

2)Randomly select k% of the graph nodes and insert junk code at each selected node.

3)Reconstruct the modified graph and use it as input to the classifier D to predict the class of the sample.

4)Repeat steps 2 and 3 for each value of k specified in the input.

5)Output the predicted class P for each test sample and the corresponding k value.

The Junk Code Insertion Procedure is used to evaluate the robustness of a classifier against junk code insertion attacks. By inserting junk code at different rates, we can observe how the classifier's performance changes and identify the rate at which the classifier's accuracy starts to degrade. This information can be used to improve the classifier's resistance to junk code insertion attacks.

Support Vector Machine

Support Vector Machine (SVM) is a powerful supervised machine learning algorithm that can be used for solving classification and regression problems. Although it can be used for both types of problems, it is mainly used for classification problems. In this algorithm, we represent each data point as a vector in a highdimensional space, where each feature corresponds to a dimension. The algorithm then tries to find the hyperplane that best separates the two classes by maximizing the margin between the two closest points of different classes. This hyperplane is then used for making predictions on new data points. SVM is a popular algorithm due to its ability to handle complex datasets and its high accuracy in classifying data points

2.1 Detection Techniques

Malware can be detected using either static or dynamic approaches. In dynamic methods, the program is run in a controlled environment, such as a sandbox or virtual machine, to observe its behavior and determine if it is malicious or not. Behavioral attributes such as resource usage, execution path, and privilege requests are collected for classification. Static approaches, on the other hand, involve analyzing the program's code to identify suspicious patterns, such as signature-based detection, byte-sequence ngram analysis, opcode sequence identification, and control flow graph traversal. DeepSign is a signature generation method proposed by David et al that automatically detects malware using deep learning techniques

2.2. Proposed OpCode based Deep Learning Model

It's great to see the details of the proposed OpCode based deep learning model for IoT and IoBT malware detection. It seems that the proposed approach employs a class-wise feature selection technique to overrule less important OpCodes in order to resist junk-code insertion attacks. This is a clever way to address a common issue in malware detection, and it's good to see that the proposed approach leverages all elements of Eigenspace to increase detection rate and sustainability.

The fact that a normalized dataset of IoT malware and benign applications is shared as a secondary contribution is also valuable, as it can be used by fellow researchers to evaluate and benchmark future malware detection approaches. It's worth noting that the proposed method belongs to the OpCode based detection category, but it could be adaptable for non-IoT platforms as well.

The use of Objdump (GNU binutils version 2.27.90) as a disassembler to extract the OpCodes is also interesting, and creating n-gram Op-Code sequences is a common approach to classify malware based on their disassembled codes. It's good to see that the number of features is carefully considered, as a significant increase in N can result in feature explosion. Overall, it's a promising approach to IoT and IoBT malware detection, and I look forward to seeing further developments in this field

III.RESULTS AND DISCUSSION

In this section, we present the general performed methodology for the experiments. Performance on the validation set is presented. Finally, classification performance and screening of the diseases are shown.

3.1. Experimental Framework

1.User Activity:

In the event that any of the mentioned IoT devices are attacked by unauthorized malware software, there is a risk that user's personal data, such as contact information, bank account numbers, and personal documents, may be compromised. To address this issue, it is important for manufacturers of IoT devices to prioritize security and implement robust measures to protect user data. This may include regular software updates, strong encryption protocols, two-factor authentication, and continuous monitoring for unusual activity. Users should also take steps to protect their personal information by using strong passwords, enabling security features, and keeping their devices up-to-date. In the event of a security breach, users should report the incident to the manufacturer and take immediate steps to protect their personal data, such as changing passwords and monitoring bank accounts for suspicious activity

2.Malware Deduction:

It is important to note that malware detection is a complex task and requires a combination of techniques and approaches to effectively identify and remove malware from a system. N-gram analysis is one approach that can be used to analyze network traffic and identify patterns of behavior that may be indicative of malware.

In the context of network traffic analysis, N-gram analysis involves breaking down network traffic into smaller chunks or sequences of data, known as n-grams. These n-grams can then be analyzed to identify patterns of behavior that may be indicative of malware. For example, certain n-gram sequences may be associated with known malware families or types of attacks.

However, it is important to note that N-gram analysis is just one of many techniques that can be used for malware detection. Other techniques include signature-based detection, behavior-based detection, and machine learning-based detection. Additionally, it is important to regularly update antivirus software and

apply security patches to operating systems and applications to prevent vulnerabilities that could be exploited by malware

3.Junk Code Insertion Attacks:

Junk code injection attack is a malware anti-forensictechnique against OpCode inspection. As the name suggests, junk code insertion may include addition of benign OpCode sequences, which do not run in a malware or inclusion of instructions (e.g. NOP) that do not actually make any difference in malware activities.Junk code insertion technique is generally designed to obfuscate malicious OpCode sequences and reduce the 'proportion' of malicious OpCodes in a malware.

3.2. Validation Set Results

. **Unit Testing**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format
- No duplicate entries should be allowed
- All links should take the user to the correct page.

Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

IV. CONCLUSION

IoT, particularly IoBT, will be increasingly important in the foreseeable future. No malware detection solution will be foolproof but we can be certain of the constant race between cyber attackers and cyber defenders. Thus, it is important that we maintain persistent pressure on threat actors. In this paper, we presented an IoT and IoBT malware detection approach based on class-wise selection of Op-Codes sequence as a feature for classification task. A graph of selected features was created for each sample and a deep Eigen space learning approach was used for malware classification. Our evaluations demonstrated the robustness of our approach in malware detection with an accuracy rate of 98.37% and a precision rate of 98.59%, as well as the capability to mitigate junk code insertion attacks.

FUNDING

This research received no external funding.

COMPETING INTERESTS

The authors declare no conflict of interest.

ACKNOWLEDGMENTS

We would like to express our sincere gratitude to our Head of Department, G. Srinivasulu Rao, and our Principal, Dr. P. C. Krishnamacahry, for giving us the opportunity to work on this amazing project about Vigorous Malware Detection in IOT devices using Machine Learning.

Secondly, we would also like to express our gratitude to our guide Mrs. Gayathri Devi S for her assistance in completing this project within the stipulated timeframe.

Lastly, I am thankful to all those who have encouraged me to complete this project before the deadline.

Authors' contributions

We conceived the presented idea also developed the theory and performed the computations. We also verified the analytical methods and provided guidance. All authors discussed the results and contributed to the final manuscript. Gayathri Devi S madam also provided valuable guidance with her expertise.

References:

[1] Sharma, S.; Krishna, C.R.; Sahay, S.K. Detection of advanced malware by machine learning techniques. In Proceedings of the SoCTA 2017, Jhansi, India, 22–24 December

[2]Chandrakala, D.; Sait, A.; Kiruthika, J.; Nivetha, R. Detection and classification of malware. In Proceedings of the 2021 International Conference on Advancements in Electrical, Electronics, Communication, Computing and Automation (ICAECA), Coimbatore, India, 8–9 October 2021; pp. 1–3.

[3] Zhao, K.; Zhang, D.; Su, X.; Li, W. Fest: A feature extraction and selection tool for android malware detection. In Proceedings of the 2015 IEEE Symposium on Computers and Communication (ISCC), Larnaca, Cyprus, 6–9 July 2015; pp. 714–720

[4] Akhtar, M.S.; Feng, T. Detection of sleep paralysis by using IoT based device and its relationship between sleep paralysis and sleep quality. EAI Endorsed Trans. Internet Things 2022, 8, e4.

[5]Gibert, D.; Mateu, C.; Planes, J.; Vicens, R. Using convolutional neural networks for classification of malware represented as images. J. Comput. Virol. Hacking Tech. 2019, 15, 15–28.

[6] Firdaus, A.; Anuar, N.B.; Karim, A.; Faizal, M.; Razak, A. Discovering optimal features using static analysis and a genetic search based method for Android malware detection. Front. Inf. Technol. Electron. Eng. 2018, 19, 712–736.