International Conference on Intelligent Technologies & Science - 2022
(ICITS-2022)

Deep Eigenspace Learning for Robust Malware Detection in Internet of Things (IoT) Devices

Anshika Singh[1], Deepti Ranjan Tiwari[2]
Computer Science and Engineering,
Lucknow Institute of Technology & Management, Lucknow, India
Anshika.anshi2593@gmail.com

Abstract: The Internet of Things (IoT) typically consists of a wide variety of Internet-connected devices and nodes in military settings, such as wearable combat uniforms and medical devices. Cybercriminals, particularly state-sponsored or nation-state actors, use these IoT devices and nodes as a valuable target. Malware is a common method of attack. Using the device's Operational Code (OpCode) sequence, we present a deep learning-based strategy for detecting Internet of Battlefield Things (IoBT) malware in this paper. To distinguish between malicious and benign applications, we use a deep Eigenspace learning technique to transform OpCodes into a vector space. In addition, we demonstrate the viability of our suggested method against junk code insertion attacks and malware detection. Last but not least, we make our malware sample accessible on Github, with the hope that it will aid future research efforts (for example, by making it easier to evaluate new methods for detecting malware).

Keywords: IOT, OpCode, Eigenspace learning, Malware detection.

1. Introduction:

An OpCode inspection-abusing malware technique is the junk code injection attack. Junk code insertion, as the name suggests, may include the inclusion of benign OpCode sequences that do not execute malware or instructions (like NOP) that have no effect on malware activities. An affinity-based criterion is used to mitigate the junk OpCode injection anti-forensics technique, which typically aims to obscure malicious OpCode sequences and decrease the malware's "proportion" of malicious OpCodes.

The IoBT CRA, a new collaborative venture that aims to develop the foundations of IoBT in the context of future Army operations, will be established by ARL"1. In such an IoT environment, there are fundamental concerns regarding privacy and security [1]. Cybercriminals are more likely to target IoBT architecture and devices due to the sensitive nature of IoBT deployment (e.g., military and warfare), despite the fact that IoT and IoBT share many of the fundamental cyber security risks (e.g., malware infection [14]).

Additionally, actors who target IoBT infrastructure and devices are more likely to be backed by the state, have better resources, and have received professional training. Two areas of active research are malware prevention and intrusion detection [11]. However, existing or conventional intrusion and malware detection and prevention solutions are unlikely to be suitable for real-world deployment due to the resource constrained nature of the majority of IoT and IoBT devices and customized operating systems.

IoT malware, such as Stuxnet, which is said to have been designed to target nuclear plants, is likely to be "harmless" to consumer devices like Android and iOS devices and personal computers because it exploits lowlevel vulnerabilities in compromised IoT devices. As a result, IoT and IoBT-specific malware detection must be addressed [20].

Due to their potential to improve detection accuracy and robustness, machine learning and deep learning techniques have recently sparked interest in malware detection (such as distinguishing between malicious and benign applications) [15]. In most cases, the following criteria are used to determine whether or not machine learning and deep learning methods are useful for detecting malware: Positive 100% (TP): indicates that a malware application has been correctly identified. Authentic Negative (TN): indicates that a benign application is correctly identified as non-malicious. Positive False (FP): indicates that a benign application is mistakenly identified as a malicious one. Negative False (FN): indicates that an application that is not malicious is not identified as malware. The following metrics will then be used to measure a system's effectiveness based on the aforementioned criteria:

Cross-validation is a fundamental method in machine learning that is used to evaluate the extent to which the results of an experiment can be generalized into an independent dataset. Accuracy is calculated by dividing the number of samples that a classifier correctly detects by the total number of malware and goodware.

Even though there are a lot of cross validation methods, such as Leave-POut, K-fold, and Repeated Random Sub-sampling, K-fold validation methods, like 10-fold, are usually used when a dataset is small. In the absence of an independent validation set, the fitness of a model to a hypothetical validation set is frequently validated using K-Fold validation methods [2, 6]. Utilizing deep learning techniques for malware detection is becoming increasingly popular as a result of the rapid pace of malware development and the significant increase in malware samples.

Organized by: International Journal of Research and Development in Applied Science and Engineering, India

2. Related Work:

Methods for detecting malware can be either static or dynamic [5]. In dynamic malware detection methods, the program is run in a controlled environment (such as a virtual machine or sandbox) to collect its behavioral attributes, such as the required resources, execution path, and requested privilege, so that it can be classified as malware or benign [6, 7], 8]. In order to identify potentially malicious software, static methods such as signature-based detection, byte-sequence n-gram analysis, opcode sequence identification, and control flow graph traversal statically inspect a program's code. Deepsign, which uses signature generation to automatically detect malware, was proposed by David et al. [9]. In a sandbox, the latter generates a dataset from behavior logs of API calls, registry entries, web searches, port accesses, and other activities before converting the logs into a binary vector. For classification, they reportedly used a deep belief network and achieved 98.6% accuracy. Pascanu et al. in another study [1] proposed a natural language modeling-based method for modeling malware execution. In order to anticipate the subsequent API calls, they used a recurrent neural network to extract relevant features. The next API call prediction classification module used a combination of logistic regression and multi-layer perceptions, with the past events' history as a feature. It was stated that a true positive rate of 98.3 percent and a false positive rate of 0.1 percent were achieved. Demme and others 4] looked into whether performance counters could be used as a learning feature and K-Nearest Neighbor, Decision Tree, and Random Forest could be used as classifiers to build a malware detector into the hardware of IoT nodes. The various malware families' reported accuracy rates range from 25% to 100%. Alam and co. 2] identified malicious code by employing Random Forest on a dataset of smartphone devices connected to the Internet. They used various tree sizes to evaluate their approach and ran APKs in an Android emulator, recorded various features for classification, such as memory information, permission, and network. According to their findings, the best classifier has 40 trees and a mean square root of 0.0171. Azmoodeh et al. [] developed a method for detecting crypto-ransomware on Android devices used as management nodes in IoT networks. 3] logged the amount of power used by running processes and discovered distinct local patterns of energy consumption for legitimate applications and ransomware. They classified each sub-sample of the power usage pattern and gathered the labels of those sub-samples to create the final label. According to reports, the proposed method was accurate by 92.75 percent. Haddad Pajouh et al. were motivated by the need to protect the IoT backbone from malware attacks. [44] to suggest a module for malicious activity detection that uses two-tier classification and two-layer dimension reduction. In particular, the dataset was reduced by means of Principle Component Analysis and Linear Discrimination Analysis, and samples were classified by means of Naive Bayes and K-Nearest Neighbor. Their false alarm and detection rates were 84.86% and 4.86%, respectively. Although OpCodes are regarded as an effective malware detection feature, it does not appear that OpCodes have been utilized for IoT and IoBT malware detection. Another unexplored area is the use of deep learning to effectively detect malware in IoT networks. By investigating the potential of OpCodes as features for malware detection with deep Eigenspace learning, our goal in this paper is to fill this void.

3. Methodology:

Python is a high-level, interactive, object-oriented, general-purpose interpreted programming language. Python is an interpreted language with a design philosophy that places an emphasis on code readability (for example, using whitespace indentation rather than curly brackets or keywords to delimit code blocks) and a syntax that lets programmers express concepts in fewer lines of code than they might in languages like C++ or Java. It offers structures that make it possible to program clearly at both small and large scales. There are Python interpreters for a lot of different operating systems. Like nearly all of its variant implementations, the reference implementation of Python, CPython, is open source software with a community-based development model. The non-profit Python Software Foundation oversees CPython. Python has an automatic memory management system and a dynamic type system. It has a large and extensive standard library and supports a variety of programming paradigms, including object-oriented, imperative, functional, and procedural.

Proposed Method:

This is, to the best of our knowledge, the first OpCode-based deep learning approach for the detection of IoT and IoBT malware. The viability of our suggested strategy against existing OpCode-based malware detection systems is then demonstrated. In addition, we demonstrate how well our suggested strategy defends against junk-code insertion attacks. To be more specific, in order to ward off junk-code insertion attacks, the strategy we propose makes use of a class-wise feature selection method to overrule OpCodes that are not as important. In addition, we make use of every Eigenspace feature to improve sustainability and detection rates. Last but not least, as a secondary contribution, we offer a normalized dataset of IoT malware and benign applications2, which other researchers can use to compare and contrast possible future methods of malware detection. On the other hand, the proposed method falls under the OpCode-based detection category, so it may be applicable to platforms other than IoT. The IoT and IoBT applications will probably be made up of a long series of OpCodes, which are instructions for the device processing unit to carry out. We extracted the OpCodes using the disassembler Objdump (GNU binutils version 2.27.90) for disassembling the samples. A common method for classifying malware

based on its disassembled codes is to create an n-gram Op-Code sequence. For a length of N, the number of basic features is CN, where C is the size of the instruction set. Clearly, feature explosion will occur with a significant increase in N. Because ineffective features will affect the machine learning approach's performance, reducing the size of the feature also improves detection robustness and efficiency.

Malware Deduction Users look for any link, but not all network traffic data produced by malicious apps matches malicious traffic. Many forms of malware are repackaged versions of safe apps; Consequently, malware may also contain a benign app's fundamental features. As a consequence of this, the kind of network traffic that they produce can be described as a mix of benign and malicious traffic. The natural language processing (NLP) N-gram method is used to examine the traffic flow header..

Algorithm

N-Gram sequence:

In the fields of computational linguistics and probability, an n-gram is a contiguous sequence of n items from a given sample of text or speech. The items can be phonemes, syllables, letters, words or base pairs according to the application. The n-grams typically are collected from a text or speech corpus.

Algorithm : Junk Code Insertion Procedure
Input: Trained Classifier D, Test Samples S, Junk Code Percentage k
Output: Predicted Class for Test Samples P
1: P = fg
2: for each sample in S do
3: W= Compute the CFG of sample based on Section 4.1
4: R = fselect k% of W's index randomly(Allow duplicate indices)g
5: for each index in R do
6: Windex = Windex + 1
7: end for
8: Normalize W
9: e1; e2= 1st and 2nd eigenvectors of W
10: l1; l2= 1st and 2nd eigenvalues of W
11: P = P
S
D(e1; e2; l1; l2)
12: end for
13: return P

Support Vector Machine

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot). The SVM algorithm is implemented in practice using a kernel. The learning of the hyper plane in linear SVM is done by transforming the problem using some linear algebra, which is out of the scope of this introduction to SVM. A powerful insight is that the linear SVM can be rephrased using the inner product of any two given observations, rather than the observations themselves. The inner product between two vectors is the sum of the multiplication of each pair of input values. For example, the inner product of the vectors [2, 3] and [5, 6] is 2*5 + 3*6 or 28. The equation for making a prediction for a new input using the dot product between the input (x) and each support vector (xi) is calculated as follows:

$$f(x) = B0 + sum(ai * (x,xi))$$

This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B0 and ai (for each input) must be estimated from the training data by the learning algorithm.

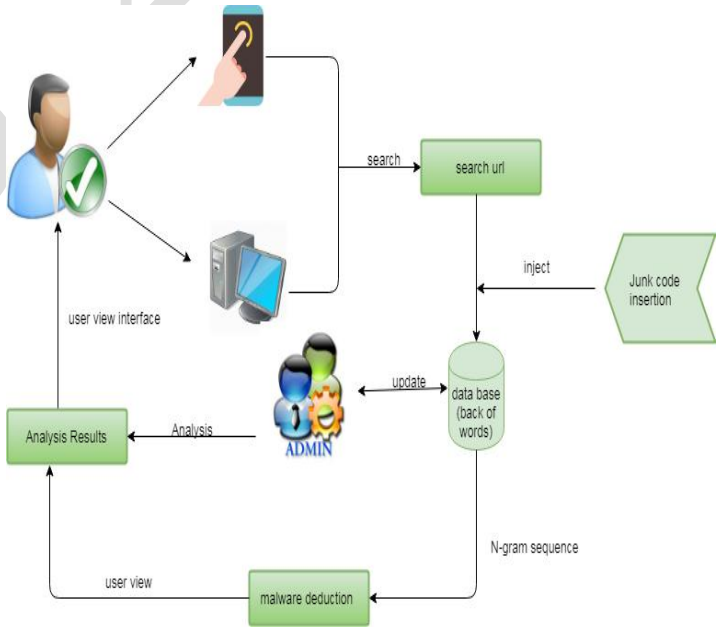


Fig 1: Architecture Diagram.

4. Result and Discussion:

Two congruent algorithms—1 and 2—described in Section 1 are applied to our generated dataset using Adaboost as the classification algorithm to demonstrate the robustness of our proposed strategy and compare it to other proposals. The Python Django Framework was used in all tests on a Microsoft Windows 10 Pro computer with an Intel Core i5 2.67GHz and 4GB of RAM. During the validation, a 10-fold

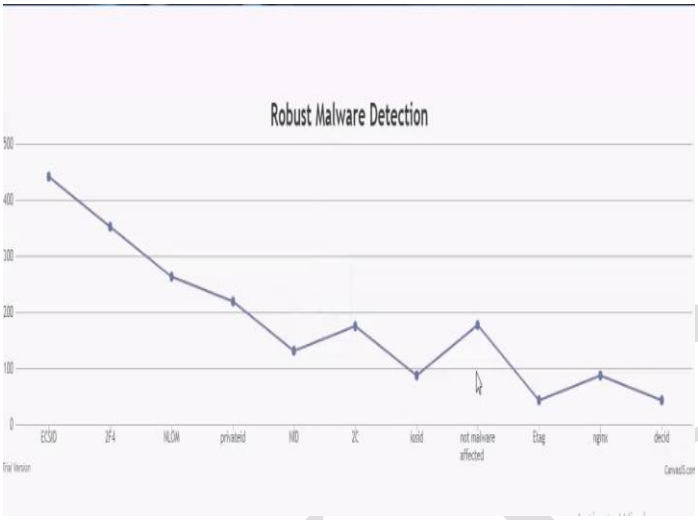
cross validation was used. It is abundantly clear that our proposed method performs better than those of Hashemi et al. and Santos and others

In the not-too-distant future, IoT, particularly IoBT, will become increasingly significant. There will never be a malware detection solution that works perfectly, but we can be sure that cyber attackers and defenders are always in a race. As a result, it is critical that we continue to exert pressure on threat actors. An IoT and IoBT malware detection strategy based on class-wise selection of Op-Code sequence as a feature for the classification task was presented in this paper. For each sample, a graph of selected features was created, and a deep Eigenspace learning method was used to classify malware. With an accuracy rate of 98.37 percent and a precision rate of 98.59 percent, our evaluations demonstrated the robustness of our malware detection strategy, as well as its ability to prevent junk code insertion attacks.

The method used by Santos et al. is a fundamental OpCode-based malware detection algorithm and Hashemi et al.'s strategy. is the closest in that it uses eigenspace as its foundation. Precision is an overall measures for assessing execution of a calculation for both malware and harmless class recognizable proof. The proposed method has a high accuracy of 99.68 percent, whereas Hashemi et al.'s and Santos and others respectively achieve an accuracy of 98.59% and 95.91%. A crucial criterion is the recall or detection rate, and the proposed method achieves 98.37 percent, compared to 81.55% and 77.70% for the other two methods.

Additionally, our proposed strategy performs better than those of Hashemi et al. and Santos and others, in terms of F-Measure and precision rate. During the classification phase, it appears that beneficial features of minor classes are more effective when class-wise feature selection is used. Additionally, when OpCode's distance is calculated using Formulation (6), more OpCode sequence patterns can be represented in the sample's graph. Additionally, it appears that a superior classifier is produced by utilizing deep neural networks for classification.

MALWARE NAME	NETWORK TRAFIC POSSITION
ECSID	442
2F4	353
NLOM	264
privateid	220
NID	132
2C	176
iosid	88
not malware affected	178
Etag	44
nginx	88
decid	44



5. Conclusion:

In the not-too-distant future, IoT, particularly IoBT, will become increasingly significant. There will never be a malware detection solution that works perfectly, but we can be sure that cyber attackers and defenders are always in a race. As a result, it is critical that we continue to exert pressure on threat actors. An IoT and IoBT malware detection strategy based on class-wise selection of Op-Code sequence as a feature for the classification task was presented in this paper. For each sample, a graph of selected features was created, and a deep Eigenspace learning method was used to classify malware. With an accuracy rate of 98.37 percent and a precision rate of 98.59 percent, our evaluations demonstrated the robustness of our malware detection strategy, as well as its ability to mitigate junk code insertion attacks.

References:

[1] E. Bertino, K.-K. R. Choo, D. Georgakopolous, and S. Nepal, "Internet of things (iot): Smart and secure service delivery," ACM Transactions on Internet Technology, vol. 16, no. 4, p. Article No. 22, 2016.
[2] X. Li, J. Niu, S. Kumari, F. Wu, A. K. Sangaiah, and K.-K. R. Choo, "A three-factor anonymous authentication scheme for wireless sensor networks in internet of things environments," Journal of Network and Computer Applications, 2017.
[3] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," Future generation computer systems, vol. 29, no. 7, pp. 1645– 1660, 2013.
[4] F. Leu, C. Ko, I. You, K.-K. R. Choo, and C.-L. Ho, "A smartphonebased wearable sensors for monitoring real-time physiological data," Computers & Electrical Engineering, 2017.
[5] M. Roopaei, P. Rad, and K.-K. R. Choo, "Cloud of things in smart agriculture: Intelligent irrigation monitoring by thermal imaging," IEEE Cloud Computing, vol. 4, no. 1, pp. 10–15, 2017.
[6] X. Li, J. Niu, S. Kumari, F. Wu, and K.-K. R. Choo, "A robust biometrics based three-factor authentication scheme for global mobility networks in smart city," Future Generation Computer Systems, 2017.
[7] L. Atzori, A. Iera, and G. Morabito, "The internet of things: A survey," Computer networks, vol. 54, no. 15, pp. 2787–2805, 2010. [8] D. Miorandi, S. Sicari, F. De Pellegrini, and I. Chlamtac, "Internet of things: Vision, applications and research challenges," Ad Hoc Networks, vol. 10, no. 7, pp. 1497–1516, 2012.
[9] A. Kott, A. Swami, and B. J. West, "The internet of battle things," Computer, vol. 49, no. 12, pp. 70–75, 2016.
[10] C. Tankard, "The security issues of the internet of things," Computer Fraud & Security, vol. 2015, no. 9, pp. 11 – 14, 2015.
[11] C. J. DOrazio, K. K. R. Choo, and L. T. Yang, "Data exfiltration from internet of things devices: ios devices as case studies," IEEE Internet of Things Journal, vol. 4, no. 2, pp. 524–535, April 2017.
[12] S. Watson and A. Dehghantanha, "Digital forensics: the missing piece of the internet of things promise," Computer Fraud & Security, vol. 2016, no. 6, pp. 5–8, 2016.
[13] M. Conti, A. Dehghantanha, K. Franke, and S. Watson, "Internet of things security and forensics: Challenges and opportunities," Future Generation Computer Systems, vol. 78, no. Part 2, pp. 544 – 546, 2018.
[14] E. Bertino and N. Islam, "Botnets and internet of things security," Computer, vol. 50, no. 2, pp. 76–79, Feb 2017.
[15] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware c&c detection: A survey," ACM Computing Surveys, vol. 49, no. 3, p. Article No. 59, 2016.