International Conference on Recent Advancement in Science & Technology- 2020
(ICRAST-2020)

Robust Malware Detection for Internet of Thongs (IoT) Devices using Deep Eigenspace Learning

[1]Shipra Singh, [2]Abhishek Saxena
Dept of Computer science
Bansal Institute of Enginnering and Technology, Lucknow, India
bhushipra14@gmail.com, abhisaxena0212@gmail.com

Abstract: Internet of Things (IoT) in military settings generally consists of a diverse range of Internet-connected devices and nodes (e.g. medical devices and wearable combat uniforms). These IoT devices and nodes are a valuable target for cyber criminals, particularly state-sponsored or nation state actors. A common attack vector is the use of malware. In this paper, we present a deep learning based method to detect Internet Of Battlefield Things (IoBT) malware via the device's Operational Code (OpCode) sequence. We transmute OpCodes into a vector space and apply a deep Eigenspace learning approach to classify malicious and benign applications. We also demonstrate the robustness of our proposed approach in malware detection and its sustainability against junk code insertion attacks. Lastly, we make available our malware sample on Github, which hopefully will benefit future research efforts (e.g. to facilitate evaluation of future malware detection approaches).

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

1. Introduction:

Junk code injection attack is a malware anti-forensic technique 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 In our proposed approach, we use an affinity based criteria to mitigate junk OpCode injection anti-forensics technique.

ARL intends to establish a new collaborative venture (the IoBT CRA) that seeks to develop the foundations of IoBT in the context of future Army operations"1. There are underpinning security and privacy concerns in such IoT environment [1]. While IoT and IoBT share many of the underpinning cyber security risks (e.g. malware infection [14]), the sensitive nature of IoBT deployment (e.g. military and warfare) makes IoBT architecture and devices more likely to be targeted by cyber criminals.

In addition, actors who target IoBT devices and infrastructure are more likely to be state-sponsored, better resourced, and professionally trained. Intrusion and malware detection and prevention are two active research areas [11]. However, the resource constrained nature of most IoT and IoBT devices and customized operating systems, existing / conventional intrusion and malware detection and prevention solutions are unlikely to be suited for real-world deployment.

For example, IoT malware may exploit lowlevel vulnerabilities present in compromised IoT devices or vulnerabilities specific to certain IoT devices (e.g., Stuxnet, a malware reportedly designed to target nuclear plants, are likely to be 'harmless' to consumer devices such as Android and iOS devices and personal computers). Thus, it is necessary to answer the need for IoT and IoBT specific malware detection [20].

There has been recent interest in utilizing machine learning and deep learning techniques in malware detection (e.g. distinguishing between malware and benign applications), due to their potential to increase detection accuracy and robustness [15]. Typically, the following criteria are used to evaluate the utility of machine learning and deep learning techniques in malware detection: True Positive (TP): indicates that a malware is correctly identified as a malicious application. True Negative (TN): indicates that a benign is detected as a non-malicious application correctly. False Positive (FP): indicates that a benign is falsely detected as a malicious application. False Negative (FN): indicates that a malware is not detected and labeled as a non-malicious application. Based on the above criteria, the following metrics will then be used to quantify the effectiveness of a given system:

Accuracy is the number of samples that a classifier correctly detects, divided by the number of all malware and goodware Cross-validation is a fundamental technique in machine learning to assess the extent that the findings of an experiment can be generalized into an independent dataset.

While there are many cross validation techniques (e.g., Leave-POut, K-fold and Repeated Random Sub-sampling), when the size of a dataset is limited K-fold validation techniques (e.g. 10-fold) are generally used. K-Fold validation techniques are also commonly used to validate the fitness of a model to a hypothetical validation set in the absence of an independent validation set [2], [6]. Due to the

fast pace of malware development and the significant increase in the number of malware samples, using deep learning techniques for malware detection is gaining prominence.

2. Related Work:

Malware detection methods can be static or dynamic [5]. In dynamic malware detection approaches, the program is executed in a controlled environment (e.g., a virtual machine or a sandbox) to collect its behavioral attributes such as required resources, execution path, and requested privilege, in order to classify a program as malware or benign [6], [7], [8]. Static approaches (e.g., signature-based detection, byte-sequence n-gram analysis, opcode sequence identification and control flow graph traversal) statically inspect a program code to detect suspicious applications. David et al [9] proposed Deepsign to automatically detect malware using a signature generation method. The latter creates a dataset based on behaviour logs of API calls, registry entries, web searches, port accesses, etc, in a sandbox and then converts logs to a binary vector. They used deep belief network for classification and reportedly achieved 98.6% accuracy. In another study, Pascanu et al. [1] proposed a method to model malware execution using natural language modeling. They extracted relevant features using recurrent neural network to predict the next API calls. Then, both logistic regression and multi-layer perceptions were applied as the classification module on next API call prediction and using history of past events as features. It was reported that 98.3% true positive rate and 0.1% false positive rate were achieved. Demme et al. [4] examined the feasibility of building a malware detector in IoT nodes' hardware using performance counters as a learning feature and K-Nearest Neighbor, Decision Tree and Random Forest as classifiers. The reported accuracy rate for different malware family ranges from 25% to 100%. Alam et al. [2] applied Random Forest on a dataset of Internet-connected smartphone devices to recognize malicious codes. They executed APKs in an Android emulator and recorded different features such as memory information, permission and network for classification, and evaluated their approach using different tree sizes. Their findings showed that the optimal classifier contains 40 trees, and 0.0171 of mean square root was achieved. In order to detect crypto-ransomware on Android devices as management nodes of an IoT networks, Azmoodeh et al. [3] recorded the power usage of running processes and identified distinguishable local energy consumption patterns for benign applications and ransomware. They broke down the power usage pattern into sub-samples and classified them, as well as aggregating sub-samples' labels to determine the final label. The proposed approach reportedly achieved 92.75% accuracy. The need to secure IoT backbone against malware attacks motivated Haddad Pajouh et al. [44] to propose a two-layer dimension reduction and two-tier classification module to detect malicious activities. Specifically, the authors used Principle Component Analysis and Linear Discrimination Analysis to reduce the dataset and then used Na¨ıve Bayes and K-Nearest Neighbor to classify samples. They achieved detection and false alarm rates of 84.86% and 4.86%, respectively. While OpCodes are considered an efficient feature for malware detection, there does not appear to have been any attempt to use OpCodes for IoT and IoBT malware detection. In addition, using deep learning for robust malware detection in IoT networks appears to be another understudied topic. Thus, in this paper, we seek to contribute to this gap by exploring the potential of using OpCodes as features for malware detection with deep Eigenspace learning.

3. Methodology:

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++or Java. It provides constructs that enable clear programming on both small and large scales. Python interpreters are available for many operating systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation. Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

Proposed System:

To the best of our knowledge, this is the first OpCodebased deep learning method for IoT and IoBT malware detection. We then demonstrate the robustness of our proposed approach, against existing OpCode based malware detection systems. We also demonstrate the effectiveness of our proposed approach against junk-code insertion attacks. Specifically, our proposed approach employs a class-wise feature selection technique to overrule less important OpCodes in order to resist junk-code insertion attacks. Furthermore, we leverage all elements of Eigenspace to increase detection rate and sustainability. Finally, as a secondary contribution, we share a normalized dataset of IoT malware and benign applications2, which may be used by fellow researchers to evaluate and benchmark future malware detection approaches. On the other hand, since the proposed method belongs to OpCode based detection category, it could be adaptable for non-IoT platforms. IoT and IoBT application are likely to consist of a long sequence

of OpCodes, which are instructions to be performed on device processing unit. In order to disassemble samples, we utilized Objdump (GNU binutils version 2.27.90) as a disassembler to extract the OpCodes. Creating n-gram Op-Code sequence is a common approach to classify malware based on their disassembled codes. The number of rudimentary features for length N is CN, where C is the size of instruction set. It is clear that a significant increase in N will result in feature explosion. In addition, decreasing the size of feature increases robustness and effectiveness of detection because ineffective features will affect performance of the machine learning approach.

Malware Deduction
Users search the any link notably, not all network traffic data generated by malicious apps correspond to malicious traffic. Many malware take the form of repackaged benign apps; thus, malware can also contain the basic functions of a benign app. Subsequently, the network traffic they generate can be characterized by mixed benign and malicious network traffic. We examine the traffic flow header using N-gram method from the natural language processing (NLP).

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.

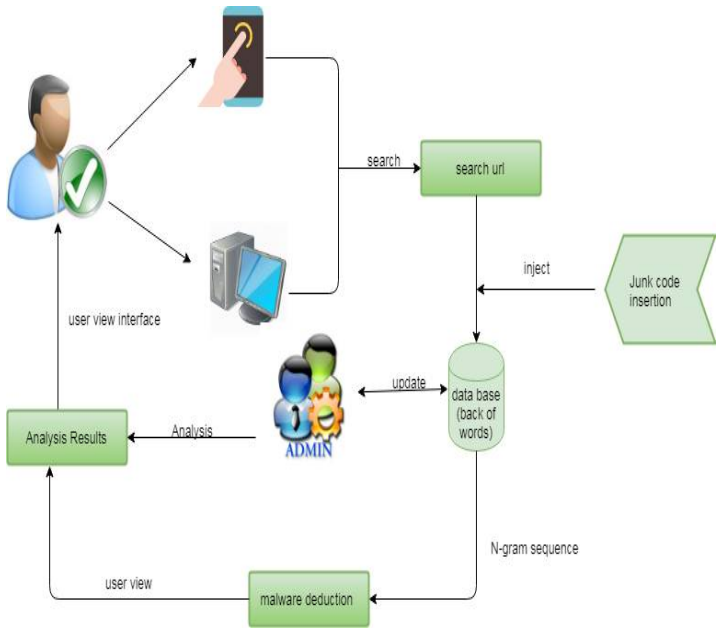


Fig 1: Architecture Diagram.

4. Result and Discussion:

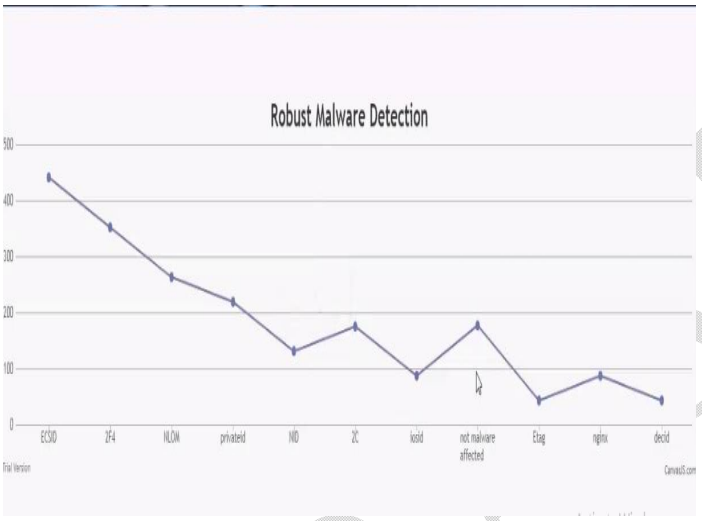
To show the robustness of our proposed approach and benchmark it against existing proposals, two congruent algorithms [1], [2] described in Section 1 are applied on our generated dataset using Adaboost as the classification algorithm. All evaluations were conducted using Python Django Framework a running on a Microsoft Windows 10 Pro personal computer powered by Intel Core i5 2.67GHz and 4GB RAM. A 10-fold cross validation was used in the validating. It is clear that our proposed approach outperforms the proposals of Hashemi et al. and Santos et al.

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 Eigenspace 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.

The approach of Santos et al. is a basic and commonly-known OpCode based malware detection algorithm and the approach of Hashemi et al. is the most similar in terms of using eigenspace as the basis. Accuracy is a general criteria for evaluating performance of an algorithm for both malware and benign class identification. The proposed approach achieves a high accuracy of 99.68%, while the approaches of Hashemi et al. and Santos et al. respectively achieve 98.59% and 95.91% accuracy. Recall or detection rate is an important criteria and the proposed approach achieves 98.37%, in comparison to 81.55% and 77.70% for the other two approaches.

Our proposed approach also outperforms the approaches of Hashemi et al. and Santos et al., in terms of precision rate and F-Measure. Utilizing class-wise feature selection appears to result in beneficial features of minor class to be more effective during classification phase. Also, using Formulation (6) to calculate OpCode's distance leads to the ability to represent more OpCode sequence patterns in the sample's graph. It also appears that employing deep neural networks for classification leads to a better classifier.

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:

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 Eigenspace 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.

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.