Robust Malware Detection for Internet Of

(Battlefield) Things Devices Using Deep

Eigen space Learning

**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).

**1.6 HARDWARE REQUIREMENTS**

**HARDWARE REQUIREMENTS:**

* **System :** Pentium IV 2.4 GHz.
* **Hard Disk :** 40 GB.
* **Floppy Drive :** 1.44 Mb.
* **Monitor** : 14’ Colour Monitor.
* **Mouse :** Optical Mouse.
* **Ram :** 512 Mb.

**1.7 SOFTWARE REQUIREMENTS**

* **Operating system :** Windows 7 Ultimate.
* **Coding Language :** Python.
* **Front-End :** Python.
* **Designing :** Html,css,javascript.
* **Data Base :** MySQL.

**EXISTING SYSTEM:**

There are underpinning security and privacy concerns in such IoT environment . While IoT and IoBT share many of the underpinning cyber security risks (e.g. malware infection ), 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 area. 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.

DISADVANTAGES:

* Although dynamic analysis surpasses the static analysis in many aspects, dynamic analysis also has some drawbacks. Firstly, dynamic analysis requires too many resources relative to static analysis, which hinders it from being deploying on resource constraint smartphone.
* On contrast to the above mentioned methods, anomaly detection engine in our proposed detection system performs dynamic analysis through Dalvik Hooking based on Xposed Framework. Therefore, our analysis module is difficult to be detected by avoiding repackaging and injecting monitoring code.
* Overall, previous work focuses on detecting malware using machine learning techniques, which are either misuse-based detection or anomaly-based detection. Misuse based detector tries to detect malware based on signatures of known malware

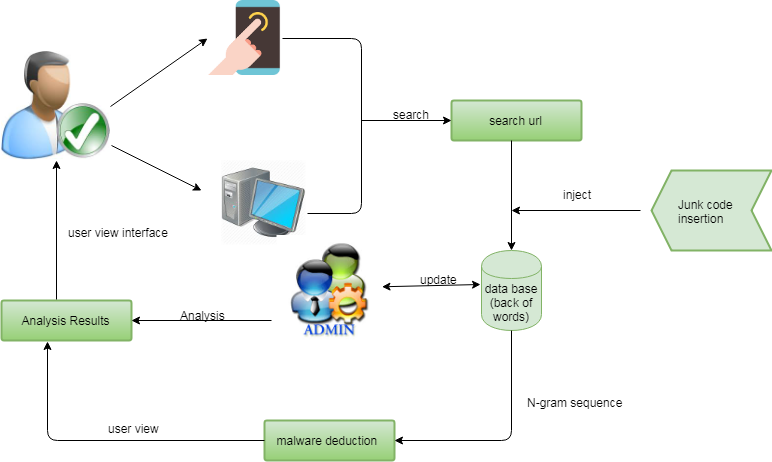
**PROPOSED SYSTEM:**

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

**ADVANTAGE:**

* The choices made in choosing the detection technique can determined the reliability and effectiveness of the Android malware detection system.
* By using this approach the malicious application can be quickly detected and able to prevent the malicious application from being installed in the device.
* Hence, by taking advantages of low false-positive rate of misuse detector and the ability of anomaly detector to detect zero-day malware, a hybrid malware detection method is proposed in this paper, which is the novelty in this paper.

**ARCHITECTURE**

****

**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.

**MODULES:**

There are three modules can be divided here for this project they are listed as below

* User Activity
* Malware Deduction
* Junk Code Insertion Attacks

From the above three modules, project is implemented. Bag of discriminative words are achieved

**MODULES:**

1. **User Activity:**

User handling for some various times of IOT(internet of thinks example for Nest Smart Home, Kisi Smart Lock, Canary Smart Security System, DHL's IoT Tracking and Monitoring System,Cisco's Connected Factory,ProGlove's Smart Glove, Kohler Verdera Smart Mirror.If any kind of devices attacks for some unauthorized malware softwares.In this malware on threats for user personal dates includes for personal contact, bank account numbers and any kind of personal documents are hacking in possible.

1. **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).

3. **Junk Code Insertion Attacks:**

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.

**CONCLUSION:**

Android is a new and fastest growing threat to malware. Currently, many research methods and antivirus scanners are not hazardous to the growing size and diversity of mobile malware. As a solution, we introduce a solution for mobile malware detection using network traffic flows, which assumes that each HTTP flow is a document and analyzes HTTP flow requests using NLP string analysis. The N-Gram line generation, feature selection algorithm, and SVM algorithm are used to create a useful malware detection model. Our evaluation demonstrates the efficiency of this solution, and our trained model greatly improves existing approaches and identifies malicious leaks with some false warnings. The harmful detection rate is 99.15%, but the wrong rate for harmful traffic is 0.45%. Using the newly discovered malware further verifies the performance of the proposed system. When used in real environments, the sample can detect 54.81% of harmful applications, which is better than other popular anti-virus scanners. As a result of the test, we show that malware models can detect our model, which does not prevent detecting other virus scanners. Obtaining basically new malicious models Virus Total detection reports are also possible. Added, Once new tablets are added to training .