International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

Robust intelligence malware detection using deep learning

P Sudhakar Rao [1],Tadisina Dwarak Reddy[2],Md Lalmahammad[3],Mavarapu Karthik[4], Patha Sneha[5]
[2,3,4,5] UG Scholars, Department of CSE, **AVN Institute of Engineering and Technology,**Hyderabad, Telangana, India.
[1] Assoiate Professor, Department of CSE, **AVN Institute of Engineering and Technology**, Hyderabad, Telangana, India.

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).

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. Specifically, our feature selection method eliminates less instructive OpCodes to mitigate the effects of injecting junk OpCodes. To demonstrate the effectiveness of our proposed approach against code insertion attack, in an iterative manner, a specified proportion (f5%, 10%, 15%, 20%, 25%, 30%g) of all elements in each sample's generated graph were selected randomly and their value incremented by one. For example, in the 4th iteration of the evaluations, 20% of

the indices in each sample's graph were chosen to increment their value by one. In addition, in our evaluations the possibility of a repetitive element selection was included to simulate injecting an OpCode more than once. Incrementing Ei;j in the sample's generated graph is equivalent to injecting OpCodej next to the OpCodei in a sample's instruction sequence to mislead the detection algorithm. Algorithm 2 describes an iteration of junk code insertion during experiments, and this procedure should repeat for each iteration of k-fold validation. To show the robustness of our proposed approach and benchmark it against existing proposals, two congruent algorithms described in Section 1 are applied on our generated dataset using Adaboost as the classification algorithm.

LITERATURE SURVEY

2.1 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.

2.1.1 LIMITATIONS OF EXIXTING SYSTEM

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.

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

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.

2.2 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.

2.2.1 ADVANTAGES OVER EXISTING SYSTEM

- 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

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

in this paper, which is the novelty in this paper.

REQUIREMENT SPECIFICATIONS

SOFTWARE REQUIREMENTS:

A software requirements specification (SRS) is a document that describes what the software will do and how it will be expected to perform. A software requirements specification is the basis for entire project. It lays the framework that every team involved in development will follow. It's used to provide critical information to multiple teams - development, quality assurance, operation and maintenance.This keeps everyone on the same page. Using the SRS helps to ensure requirements are fulfilled. And it can also help you make decisions about your product's lifecycle - for instance, when to retire a feature. Writing an SRS can also minimize overall development time and costs. Embedded development teams especially benefits from using an SRS.

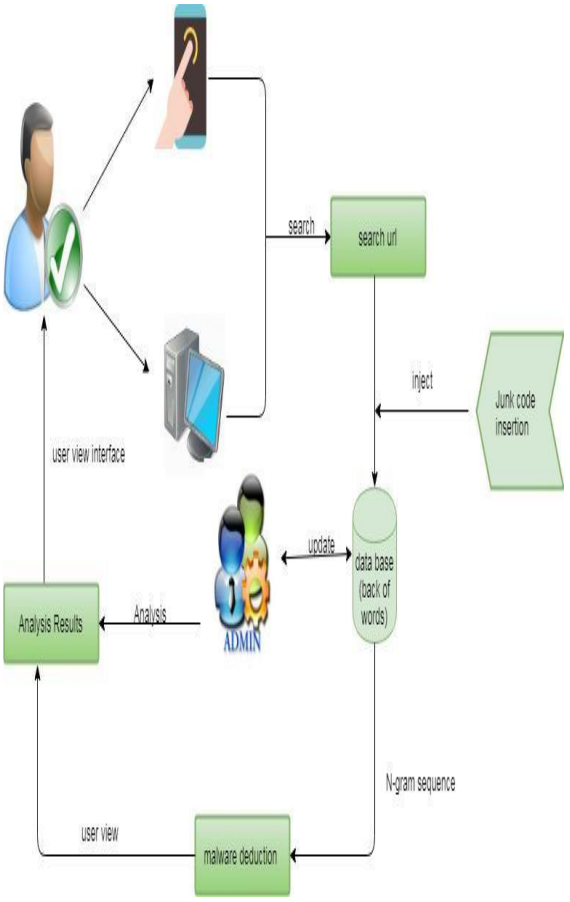


Fig 3.1.1 System Architec

MODULE DESCRIPTION

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.

4.1 USER ACTIVITY:

User handling for some various times of IOT(internet of thinks example for Nest

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

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.

4.2 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).

4.3 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.

Results:

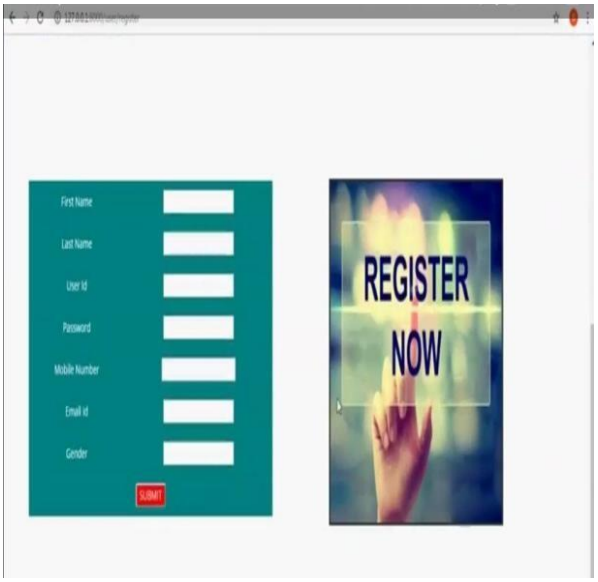
HOME PAGE: This is screen that is opened once the project is run as shown in screen 1. The screen has the title name followed by the tabs which are follows-

Screen 6.3.1 Home page

Registration: A registered user is a user of a website, program, or other system who has previously registered. Registered users normally provide some sort of credentials (such as a username or e-mail address, and a password) to the

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

system in order to prove their identity: this is known as logging in. Systems intended for use by the general public often allow any user to register simply by selecting a register or sign up function and providing these credentials for the first time.



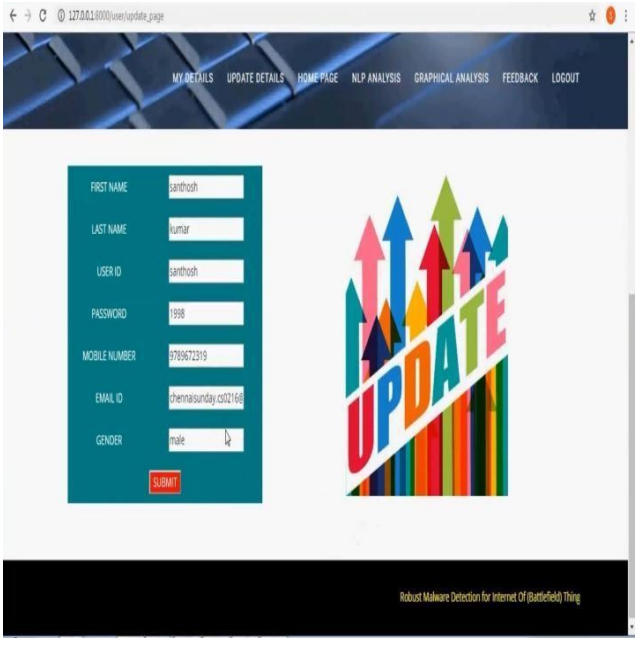
Screen 6.3.2 Registration page

Screen 6.3.3 My Details

Sign in: After registration process user must be sign in with his username and password .A sign in is the period of activity between a user sing in and sign out of a (multi-user) system.

Screen 6.3.4 Sign in

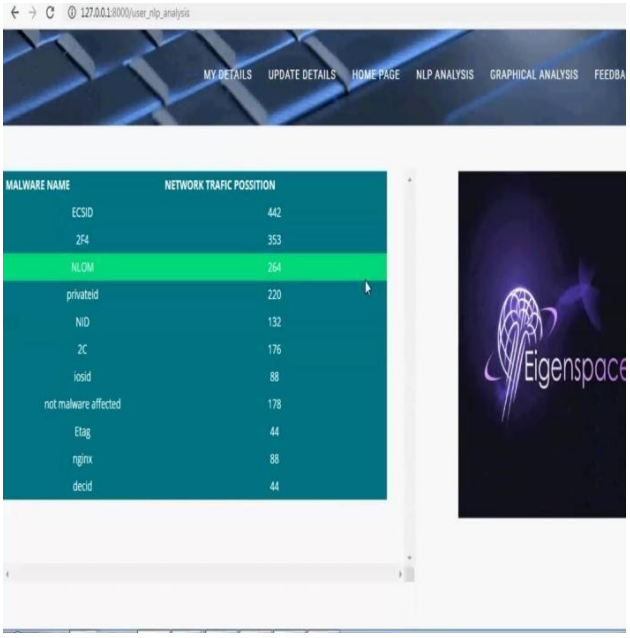
UPDATE DETAILS: In update tab the user can update his details



Screen 6.3.5 Update details

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

NLP ANALYSIS:



Screen 6.3.6 NLP Analysis

GRAPHICAL ANALYSIS:

Screen 6.3.6 Graphical Analysis

ADMIN PAGE:

Admin login:

Screen 6.3.8
Admin login

OPCODE BASED MALWARE:

Screen6.3.9 Opcode sequence analyses

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal
www.ijarst.in
ISSN: 2457-0362

GRAPHICAL ANALYSES:

Result Cases :

		box	
3	Information invalid	If information is not founded	Fail
4	Searching URL	All the category of URL's are listed to search	Pass
5	Verify	If the given URL data is found in database	Pass
6	NLP analysis	Viewing results and graphical representation	Pass
7	Feedback	If the feedback is successfully taken	Pass

International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

		with alphabets and password with only digits	
2	Updating information	Related information must be enter in specified	Pass

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.

FUTURE ENHANCEMENT:

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.

REFERENCES :

- https://ieeexplore.ieee.org/document/8302863/
- https://www.google.com/search?rlz=1C1CHBF_enIN854IN854&sxsrf=ALeKk03MZoDsC7Y3dMmotBRglFMni5-FUw%3A1590205665898&ei=4ZzIXtG4NqOY4-EP4PK3GA&q=robust+malware+detection+for+iot+devices+using+deep+eigenspace+learning+github&oq=robust+malware+detect&gs_lcp
- https://www.researchgate.net/publication/323405239_Robust_Malware_Detection_for_Internet_Of_Battlefield_Things_Devices_Using_Deep_Eigenspace_Learning
- https://github.com/nsslabcuus/Malware
- https://towardsdatascience.com/malware-detection-using-deep-learning-6c95dd235432
- https://www.jetbrains.com/help/pycharm/configuring-project-and-ide-settings.html
- https://sourceforge.net/projects/staruml/