# Dynamics and an efficient malware detection system using op-code sequence graph generation and ml algorithm

*Bharathi* Panduri[1, *], *Madhurika* Vummenthala[2], *Spoorthi* Jonnalagadda[2], *Garwandha* Ashwini[2], *Naruvadi* Nagamani[2] and *Amanagati* Akhila[2]

[1]Asst.Professor, GRIET, Information Technology Department, Hyderabad, India.
[2]UG Student, GRIET, Information Technology Department, Hyderabad, India

**Abstract*:*** IoT(Internet of things), for the most part, comprises of the various scope of Internet-associated gadgets and hubs. In the context of military and defence systems (called as IoBT) these gadgets could be personnel wearable battle outfits, tracking devices, cameras, clinical gadgets etc., The integrity and safety of these devices are critical in mission success and it is of utmost importance to keep them secure. One of the typical ways of the attack on these gadgets is through the use of malware, whose aim could be to compromise the device and or breach the communications. Generally, these IoBT gadgets and hubs are a much more significant target for cyber criminals due to the value they pose, more so than IoT devices. In this paper we attempt at creating a significant learning based procedure to distinguish, classify and tracksuch malware in IoBT(Internet of battlefield things) through operational codes progression. This is achieved by transforming the aforementioned OpCodes into a vector space, upon which a Deep Eigen space learning technique is applied to differentiate between harmful and safe applications. For robust classification, Support vector machine and n gram Sequencing algorithms are proposed in this paper. Moreover, we evaluate the quality of our proposed approach in malware recognition and also its maintainability against garbage code injection assault. These results are presented on a web page which has separate components and levels of accessibility for user and admin credentials. For the purpose of tracking the prevalence of various malwares on the network, counts and against garbage code injection assault. These results are presented on a web page which has separate components and levels of accessibility for user and admin credentials. For the purpose of tracking the prevalence of various malwares on the network, counts and trends of different malicious opcodes are displayed for both user and admin. Thereby our proposed approach will be beneficial for the users, especially for those who want to communicate confidential information within the network. It is also beneficial if a user wants to know whether a message is secure or not. This has also been made malware test accessible, which ideally will profit future research endeavors.

## 1. Introduction

The Internet of Things (IoT) is an add-on of the traditional internet, which allows a large number of smart devices such as home appliances, controllers, network cameras, and sensors to connect and share information. Also, IoT devices are being increasingly deployed in various industries for different purposes. The Internet of Things also has robust military applications in a connected network that increases risk assessment and responsive time. It also genrates a vast amount of data. The Internet of Battlefield Things (IoBT) involves the complete realization of omnipresent sensing, prevalent computing, and practical and remarkable communication, leading to an unparalleled scale of information produced by the sensors and computer units. This increasing presence in a wide range of applications, along with their processing and computing capabilities, making them a valuable attack target, such as malware designed to compromise the security of such devices. Malicious software or malware is the most common type of cyber security threats. Thus, its impact has raised the demand to find a new approach for real-time identification and detection of new malware attacks.
In this document, we explore the potential of using Deep Eigen Space Learning for detecting the IoT and IoBT malware.

## 2. LITERATURE SURVEY

Zhi-Kai Zhang: Presented EDIMA(Early detection of IoT Malware Network activity) using machine learning techniques, a particular arrangement that can IoT towards the discovery of malware in IoT devices during examining stage instead of during an assault. IoT presented a structure to exhibit a potential use of malware dispersion in IoT systems. Andras Rozsa: Proposed a two-dimensional methodology, where a runtime malware identifier (HORM) that utilizes equipment performance counter (HPC) qualities to identify malware. The accuracy obtained is 92.21%. Zubair Md. Fadlullah: Proposed a honeypot based methodology that uses AI procedures for malware identification.The methodology can be taken as a profitable start towards combating zero-day LITERATURE attacks which developed as open test in shielding IoT. Zhenlong Yuan: Declared Naive Bayes far superior to different calculations as of in concern and in the recognition process

## 3. METHODOLOGY

Most of the files like .doc, exe, html etc can be used to setup software. But browsing such files from dubious sources may contain maliciousness, we have obtained a dataset of 1078 benign and 128 malware samples for IoT-based applications. Each of these samples were gathered from a diversity of authentic IOT App stores. Most of the IOBT and IoT systems contain a prolonged series of instructions called Opcodes which should be executed on device central processor. So, as to disassemble this specimen, we make use of Objdump as a disassembler in order to extort these Opcodes. Generating N-gram Opcode series is an easy path to categorize the malware based on their disassembled codes. C^N refers to the primary features for length N, where C is the set of instructions. When N value gets incremented, there is a chance of explosion of features. Consequently, with the decrease of feature size there is a chance to increase effectiveness along with robustness as fruitless attributes will infect performance of machine learning path. For that reason, initially we need to apply feature selection method and select the most excellent features to minimize the feature set to control explosion of features. When working with large amounts of datasets, minimizing number of features plays a vital role as it speeds up the training which ultimately constitutes better classification and gives accurate results. For example, algorithms like Decision Tree, Neural Networks, Information Gain methods tries to select global features based on the amount of data available in classification problem which may lead to the reduction of efficiency of the system and such algorithms requires more computational resources to construct trees. So we proposed N-gram succession technique for developing features. N-gram is a series of n things or items from given samples; the things can be letters, words, syllables based on the application that we use it. We calculated Class wise Informative Gain to identify more useful features, so we have extracted 4513 1-gram and 610109 2-gram definite Opcode sequences. The topmost 82 features were considered which either belong to 1-gram 0r 2-gram and size of the selected feature is set to j=82.

These features were selected based on their CIG values. Such features play a vital role in the progress of our malware discovery. In our proposed approach, we have two phases namely Graph Generation for Opcode sequence and Deep Eigen Space Learning.

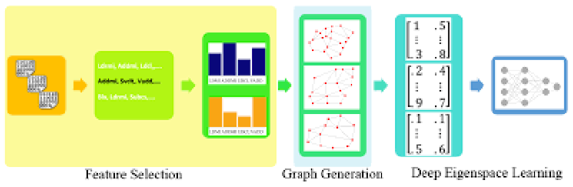


**Fig 1**.Proposed Approach

In the first phase, we generate a graph for the Opcodes. Here graph is a data structure that signifies the arrangement of Opcodes in an executable file. The graph contains edges and vertices where E denotes edges and V denotes vertices. To construct a graph, we need to compute edge values. Using the below-mentioned algorithm, a graph of 82 vertices for each benign and malware specimen is obtained and an adjacency matrix is generated for each specimen within our dataset which is further used for the implementation of Deep Eigen Space Learning phase.

```
Algorithm 1 Graph Generation Algorithm for Each Sample
Input: Sample $P$, Selected Features $F$
Output: Generated Graph for Sample $G$
 1: $k = Number\ Of\ Items\ in\ F$
 2: $G = Zero\ Matrix\ k * k$
 3: for $i = 1$ to $k$ do
 4:     $v_i = F_i$
 5:     for $j = 1$ to $k$ do
 6:         $v_j = F_j$
 7:         $G_{i,j} = E_{v_i,v_j}$ (using Formulation 6)
 8:     end for
 9: end for
10: $Row\_Normalize\ Matrix\ G$
11: return $G$
```

**Fig. 2**: Graph Generation Algorithm for Each Sample

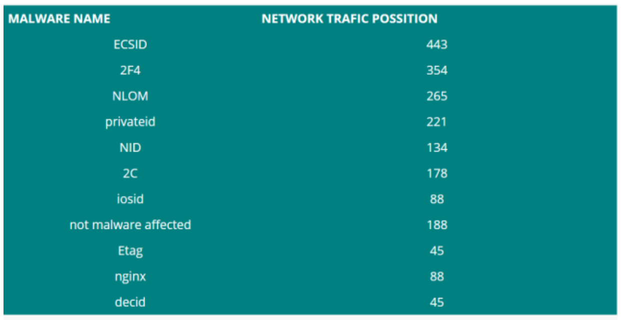
In the Deep Eigen space Learning phase, we convert graph&#39;s adjacency matrix into a vector space. Eigen value and Eigenvectors are two essential components that would linearly convert matrix to vector space. The obtained eigen vectors and values are used as input parameters for classification. In order to show the robustness of our approach in detecting IOT and IOBT malware, we evaluate metrics like accuracy, recall, f-measure, precision. So as to classify and demonstrate the robustness of our model against existing methods we enforced an algorithm named multi-support vector network machine learning. The above mentioned class-wise feature selection constitutes to be more productive during this classification phase. Furthermore, we signify the sustainability of our proposed method against junk code insertion attacks. As the name mentions, insertion of junk code might include adding benign Opcode sequences that do not make any change in malware tasks.

This technique is designed to decrease the proportion of malicious Opcodes in malware.

```
Algorithm 2 Junk Code Insertion Procedure
Input: Trained Classifier $D$, Test Samples $S$, Junk Code
    Percentage $k$
Output: Predicted Class for Test Samples $P$
 1: $P = \{\}$
 2: for each $sample$ in $S$ do
 3:     $W$= Compute the CFG of $sample$ based on Section 4.1
 4:     $R$ = {select $k\%$ of $W$'s index randomly(Allow dupli-
        cate indices)}
 5:     for each $index$ in $R$ do
 6:         $W_{index} = W_{index} + 1$
 7:     end for
 8:     Normalize $W$
 9:     $e_1, e_2$= 1st and 2nd eigenvectors of $W$
10:     $l_1, l_2$= 1st and 2nd eigenvalues of $W$
11:     $P = P \bigcup D(e_1, e_2, l_1, l_2)$
12: end for
13: return $P$
```

**Fig. 3.**Junk Code Insertion Procedure

## 4. Results



| MALWARE NAME | NETWORK TRAFIC POSSITION |
|---|---|
| ECSID | 443 |
| 2F4 | 354 |
| NLOM | 265 |
| privateid | 221 |
| NID | 134 |
| 2C | 178 |
| iosid | 88 |
| not malware affected | 188 |
| Etag | 45 |
| nginx | 88 |
| decid | 45 |

**Fig. 4.** Malware name and its network traffic position

The above figure contains two columns named as malware name and network traffic position. Malware name contains names or lists of the opcodes. Network traffic position contains the number of times that particular malware is affected.

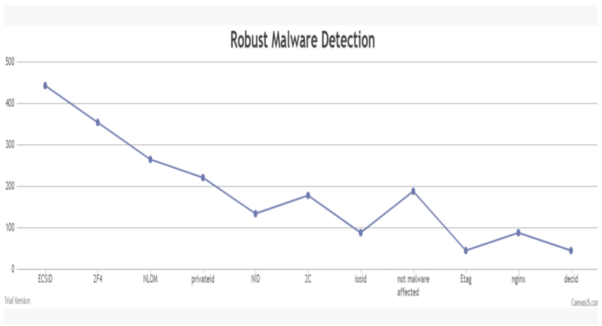


**Fig. 5.** Op-code Graph

The graph gives us information about the number of times the malware got affected by IOBT devices. The graph contains malware names and the number of times it is detected as malware. This graph changes whenever malware gets affected. It provides a clear view of malware that should not be used.

## 5. Conclusion

As we progress with time, IoT and especially IoBT take prominence in our day-to-day lives. Theoretically, we cannot keep them completely secure all the time, but the best we can do is to keep a consistent pressure on agents posing threats. In this paper we present an efficient method for recognizing such threats posed by malwares and characterize them. We used the Support Vector Machine and ngram sequence to make a malware recognition model. After building the model we deploy this on a webpage to track and identify any malware in the network and display this for users.

## References

1. Ekta Gandotra, Divya Bansal and Sanjeev 32(8) 118-123 (2010)
2. J. Zico Kolter and Marcus A. Maloof, vol. 7, (2010)
3. Usukhbayar Baldangombo, Nyamjav Jambaljav and Shi-Jinn Horng, 38(9), (2012)
4. D. Swathigavaishnave and R. Sarala, 22(9), 153-155, May (2012).
5. Ohm Sornil and Chatchai, Liangboonprakong vol. 4, no. 5, July (2013).
6. I. Firdausi, C. Lim and Erwin, pp. 201-203, (2010).
7. Igor Santos, Jaime Devesa, Felix Brezo, Javier Nieves and Pablo G. Bringas, 38, 40-46 (2013).
8. Yogeswara Reddy B, Srinivas Rao J, Suresh Kumar T, Nagarjuna A, Int. J of Inn Tech and Exp Eng., vol. 8, no. 11, (2019)
9. George E. Dahl, Jack W. Stokes, Li Deng and Dong Yu, 1850-1853 (2013).
10. Rafiqul Islam, Ronghua Tian, Lynn M. Battena and Steve Versteeg, CA Labs, vol. 36, pp. 646-656, (2013).
11. Mariano Graziano, Davide Canali, Davide Balzarotti, Leyla Bilge and Andrea Lanzi, Vol 48. pp. 375-382 (2015).
12. Smita Ranveer and Swapnaja Hiray, vol. 120, no. 5, June (2015).
13. Micha Moffie, avid Kaeli and Winnie Cheng pp.883-888 (2010).
14. Chih-Ta Lin, Nai-Jian Wang, Han Xiao and Claudia Eckert Vol. 7, no 8 (2015).
15. Michal Kruczkowoski and Ewa Niewiadomska, pp. 350-353 (2014).