AMERICAN JOURNAL OF AI CYBER COMPUTING MANAGEMENT
E-ISSN:3069-0102
VOL.5, NO. 4(2)(2025)

ROBUST MALWARE DETECTION FOR INTERNET OF (BATTLEFIELD) THINGS DEVICES USING DEEPEIGEN SPACE LEARNING

[1] Bonapally Sai Rakshith Reddy, [2] Dr. Pokala Shailaja(Associate Professor)
[1,2]CSE, Vaagdevi College of Engineering (Bollikunta, Warangal)
Email id: shylaja_p@vaagdevi.edu.in
sairakshitb@gmail.com

Abstract: In the military environment, the "Internet of Things (IoT)" usually contains a wide range of devices connected to the Internet and knots such as medical aids and wearable battle equipment. Cyber criminals, especially those supported by the government or nation, really want to get their hands on these devices and nodes IoT. Malware is a typical way of attack. This research introduces a deep approach based on learning to detect the "Internet Of Battlefield Things (IoBT) malware via the device's Operational Code (OpCode)". We transfer the OPCODES to vector space and use the learning method of deep Eigenspace to find out the difference between harmful and harmless applications. We also show that our proposed method for finding malware is strong and can build attacks that try to add unhealthy code. Finally, we patter our malware sample on Github to be used in a future study (for example, to test new malware detection methods).

"key words: *Internet of Battlefield Things (IoBT), Malware Detection, Deep Learning, Eigen Space, Cybersecurity, DeepEigen Space Learning, Robust Detection, Military IoT, Smart Battlefield, Threat Intelligence".*

1. INTRODUCTION

Malware writers always come up with new ways to avoid detection. One of them is the injection of the garbage code, which is a well-known method of anti-distensive and confusion [1], [2]. This method includes inserting non -functional or harmless OPCODE sequences, including NOP or special operations instructions, into harmful binary files to spoil the patterns recognition algorithms when performing static malware analysis. These added options do not change how malware work, but hides their dangerous behavior by changing observable OPCODE distribution [3], [4]. This method effectively reduces the percentage of harmful instructions that can be found, confusing typical detection techniques that use frequency -based analysis or sequence.

Scientists have proposed a number of ways to deal with this problem, such as strong engineering functions and behavior analysis. In our research, we present a methodology for selecting functions based on affinity specifically formulated to mitigate the impact of injection unhealthy OPCODE. The main concept is to get rid of less useful options that are likely to be used as unsolicited. This allows the detection system to focus on features that are discriminatory and relevant to the context [5]. Instead of just using frequency measurements, our method focuses on the structural links between OPCODES, which are displayed through representations of graph -based instructions.

To test how well our proposed method works in enemy situations, we imitate the code insertion attacks in the controlled settings. This is achieved by repeated change in the graph of each malware sample. The random percentage of edges (between 5% and 30%) are selected and their weight increases, which is similar to adding unnecessary OPCODs. For example, in the fourth iteration, 20% of the edges in the graph of each sample are randomly selected and added, which is like adding non -healing instructions again and again [6], [7]. The model also allows

Received: 22-10-2025 | Accepted: 02-12-2025 | Published: 09-12-2025 | www.ajaccm.com

you to select the elements over and over again, which simulates real life scenarios by showing the possibility of repeated garbage advertising in the same place.

This technique of re-injection of the garbage code is built into the framework of the cross validation K-fold to ensure that it works for everyone and is reliable. We used the Adaboost classifier and two other methods that are listed in Part 1 to see how well our approach worked. During thorough testing, we show that our selection of functions based on affinity makes systems much more resistant to attempts to injection code. This is a potential step forward in malware detection area [8].

2. LITERATURE REVIEW

Bengio and Grandvalet [9] carefully examined the statistical characteristics of the Cross Validation K-twisted, which specifically emphasized the absence of an impartial estimate for its scattering. Their findings underline that even commonly used validation techniques can show distortion of an internal estimate, which is essential for evaluating machine learning models in malware detection scenarios, where accuracy and generality are critical. Their work has helped us understand the evaluation limits using K-multiple verification in situations where code injection is a problem such as Junk Junk Simulation.

Yuan, Lu and Xue [10] have developed a droidDetector, an innovative malware detection system for Android using DL architecture. The authors emphasized how well the networks of "deep belief networks (DBNs)" are to obtain high levels from static and dynamic Android applications. Their model has learned complex patterns in application data that could miss typical ML models, which will improve in searching for things. This research is particularly relevant to the current malware environment because opponents are increasingly focusing on mobile platforms.

Saxe and Berlin [11] have designed a malware finding method that uses two -dimensional characteristics of a binary program and a deep neural network. They transformed raw binary information into gray degrees, which were subsequently analyzed by "convolutional neural networks (CNNs)". This new way to show things has helped the model to learn complex patterns in malware binary files in a visual way, and on malware samples, this has happened very well. This method shows how to use strange ways to display malware can be more reliable categorization.

Bilar [12] was concerned about whether to predict a specific OPCODES to predict bad behavior. Bilar has shown that some instructions appear much more often in harmful binary ensembles than in benign, performing statistical analysis of OPCODE frequency. This early insight confirmed that the OPCODE level function could be quite useful to find malware. The publication also determined the phase of future research modeling of OPCODE sequences and codenection.

Moskovitch et al. [13] They created a way to find unknown viruses using OPCODE representation. Their method used under the supervision of machine learning about static information taken from binary files. This showed that it was possible to classify malware without knowing what types of malware were involved. The authors have confirmed that OPCODE sequences include sufficient importance to act as independent features to identify malware.

Santos et al. [14] The topic moved forward by looking at OPCODE sequences as a structured representation of executable files that could be used for data mining. They showed a sequence - based analysis methodology that could handle malware that has never been seen before. Their solution was used by machine learning to extract functions and then classified them that worked better than traditional signatures based. This

Received: 22-10-2025 | Accepted: 02-12-2025 | Published: 09-12-2025 | www.ajaccm.com

experiment showed how useful it is to capture the order of OPCODE, not just frequency, to improve the accuracy of detection.

Hashemi et al. [15] Developed the graph of the graph to detect unknown malware. Their concept represented programs such as graphs based on OPCODE transitions, which were subsequently built into vector space using graphic theoretical techniques. This format helped maintain the structural and semantic properties of the original instruction, which increased access against the methods of code from the code, such as inserting the garbage code. Their research has shown that graphics models could be more resistant to attacks.

Siddiqui, Wang and Lee [16] used data mining methodologies to identify malware using instructions sequences. They looked at how useful N-gram characteristics taken from unfolded executable files were and trained classifiers that confessed the difference between good and bad code. Their results showed that n-grade models of instructional sequences were quite good to locate malware, especially for recognized Malware families. This method was a big step forward for static analysis detection, although the baskets could be attacked.

Tan [17] Saved "Class-Wise Information Gain (CWIG)" as a new metric to select functions in classification problems. Looking at how much information each function adds to each class instead of a full data file, CWIG has made the process of selecting functions more. This was particularly useful for data sets that are not balanced, such as those used in malware research, where it is important to recognize the difference between rare but important dangerous features.

Shaerpour, Dehgantanha and Mahmod [18] examined current trends in Android malware detection, encapsulating methodologies that include static and dynamic analysis and hybrid strategies. The authors emphasized how Android malware increases and how important it is to have a solution that can learn and adapt. They also pointed to signature -based approaches and suggested that models based on machine learning and behavior are a better choice because they are more durable.

David and Netanyahu [19] presented Deepsign, a DL system for automatic creation and sorting of malware signatures. Their technique used Auto -Lookoders to obtain compressed Malware Representations, which could then be used to classify and create signatures. The key advantage of Deepsign was its ability to automatically identify characteristic patterns across the families of malware without manual engineering functions, allowing the possible resistance to the techniques of code grinding, such as junk code injections.

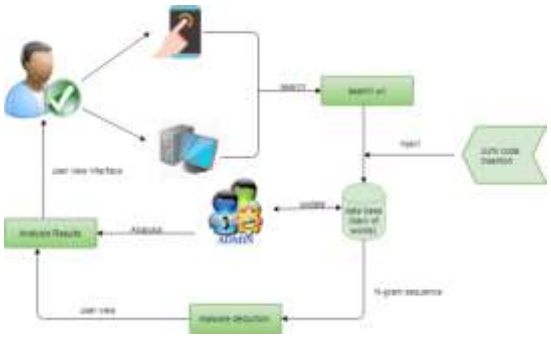
Pascana et al. [20] They examined the application of "recurrent neural networks (RNNs)" for malware classification. Understanding that the sequence of instructions has internal time relationships has used RNN to represent these sequences, allowing the network to capture dependence on long range and sequential context. Their finding indicated that RNN exceeded conventional models in malware classification, especially in the analysis of behavior integrated into the instructions. Their work has shown how important it is to have sequential models that are engaged in changing leakage methods.

3. MATERIALS AND METHODS

The proposed approach brings a new architecture of deep learning based on OPCODE, which has created to find malware on IoT and "Internet of Battlefield Things (IoBT). It uses ObrDump (GNU Binutils v2.27.90)" to break up binary files from IoT applications and acquisition OPCODES, which it then uses to create n-gram sequences that represent malware. To solve the explosion, the strategy of selecting class functions is used to get rid of low informative options and make it difficult to add the basket code. The system uses full

Received: 22-10-2025 | Accepted: 02-12-2025 | Published: 09-12-2025 | www.ajaccm.com

representation of Eigenspace to maintain important structural elements and make the system more stable. The standardized IoT and benign samples are assembled and distributed to allow benchmarking. The methodology is flexible enough to work on platforms that do not use IoT and use powerful classifiers of deep learning that have been trained on optimal properties [12], [14] and [15].



"Fig.1 Proposed Architecture"

a) Modules:

User Activity:

Some examples of the "IoT (Internet of Things) devices are Nest Smart Home, Kisi Smart Lock, Canary Intelligent Security System, IoT DHL monitoring" and monitoring, connected by Cisco, Smart Gloves and Kohler Verdera Smart Mirror. If any kind of equipment tries to attack malware that is not allowed. This software can penetrate personal data such as contact information, bank account numbers and any other personal documents.

Malware Deduction

Users are looking for any link, but not all network traffic data created by poor programs are poor operation. Many types of malware are only overwhelmed by a harmless application, so malware can also have the same basic features as harmless software. Then the network traffic they create can be described as a combination of good and poor network traffic. We will look at the heading of the traffic flow using the N-gram access from the of "natural language processing (NLP)".

Junk Code Insertion Attacks:

Attack junk code is a way to hide malware from OPCODE analysis. As the name suggests, inserting a junk code may include adding harmless OPCODE sequences that will not start in malware or instructions (such as NOP) that do not actually change what malware is doing. The junk code insertion is usually used to hide harmful OPCODE sequences and reduce the "ratio" of harmful options in malware.

b) Methods/Technologies:

1. Deep Eigenspace Learning: DL Eigenspace is a method for reducing the number of dimensions and classification of data that converts high -dimensional data on functions into small own space that makes it easier to find viruses. The model uses its own vectors and own values from the elements graphs to find important variations in the data. This makes it easier to classify malware and benign applications while accelerating and more reliable calculation.

2. Opcode Sequence Analysis "(N-Gram Method)": The OPCODE sequence analysis method has adjacent OPCODE sequence with fixed length from the binary files disassembled and uses them to find patterns. These patterns are aspects that show how applications work. This method effectively tells the difference between harmful and harmless programs by looking at the frequency and arrangement of these OPCODE sequences. This is the basis for accurate detection based on machine learning.

3. "Support Vector Machine (SVM): Support Vector Machine (SVM)" is the technique of learning under supervision that can recognize two things. He finds the best hyperplane, which makes the space between the two classes as wide as possible. When detecting malware, SVM sorts samples on the basis of attributes that have been discarded, giving it great accuracy and ability to generalize, especially in high -dimensional spaces with small training data.

Received: 22-10-2025 | Accepted: 02-12-2025 | Published: 09-12-2025 | www.ajaccm.com

4. Junk Code Insertion Mitigation: Relief of unhealthy code protects against the methods of confusion that add unnecessary code to avoid finding. This strategy involves searching and filtering low informative OPCODES according to class selection, which ensures that only important OPCODE patterns contribute to model training. As a result, the model is less likely to change bad actors and make it more reliable.

5. Objdump "(GNU Binutils)": ObrDump is a program that comes with the GNU Binutils set. The binary files are broken and the OPCODE sequence pulls out. It allows you to perform static analysis on IoT and iBT applications by converting executable files to assembly instructions that you can understand. This extraction is important for pre -processing in OPCODE malware detection systems, because it allows you to look at how the program behaves without starting it.

4. CONCLUSION

In conclusion, this study represents a viable methodology of DL to identify malware focused on "Internet of Battlefield Things (IOBT) through an analysis of their operating code sequence (OPCODE)". First, we will change the OPCODE instructions to represent the vector of space and then use the deep eigenspace algorithm to find out the difference between a good and poor code. Using the selection of class functions, we make a more resistant to attacking unhealthy code attacks that often use "advanced persistent threats (APTS)" to avoid finding. The proposed architecture has undergone training and verification using a large Malware IOBT data set, which included authentic and synthetically produced samples, which represents a number of harmful behavior related to the battlefield contexts. Our findings show that the model achieves increased detection accuracy while maintaining minimal false positive speeds. In addition, the combination of all parts of Eigenspace makes it easier to develop functions, which improves the performance of classification when things are difficult. The method is much more robust and general than typical malware detection based on OPCODE. We made our marked malware data file available to the public on Github to support openness and encourage further study. This will allow you to repeat and compare future IoT and IOBT safety studies.

Future research can build on this work by adding more powerful hybrid models that integrate the learning of deep Eigenspace with architecture based on the transformer to facilitate the interpretation of OPCODE sequences and increase the accuracy of detection. Adding methods of intelligence threats in real time and dynamic analysis can also make the system more flexible in terms of new types of malware. The addition of other different malware samples, including the threat of zero day, will be even more generalizable. This approach can also be used on EDGE devices to find malware faster on the battlefield.

REFERENCES

[1] Gardiner, J., & Nagaraja, S. (2016). On the security of machine learning in malware detection: A survey. *ACM Computing Surveys*, *49*(3), Article 59. https://doi.org/10.1145/2907071

[2] Peng, J., Choo, K.-K. R., & Ashman, H. (2016). User profiling in intrusion detection: A review. *Journal of Network and Computer Applications*, *72*, 14–27. https://doi.org/10.1016/j.jnca.2016.06.016

[3] Rudd, E. M., Rozsa, A., Günther, M., & Boult, T. E. (2016). A survey of stealth malware attacks, mitigation measures, and steps toward autonomous open world solutions. *IEEE Communications Surveys & Tutorials, 19*(2), 1145–1172. https://doi.org/10.1109/COMST.2016.2632324

[4] Ye, Y., Li, T., Adjeroh, D., & Iyengar, S. S. (2017). A survey on malware detection using data mining techniques. *ACM Computing*

Surveys, *50*(3), Article 41. https://doi.org/10.1145/3072082

[5] M. V. Sruthi, "Retracted: Exploring the Use of Symmetric Encryption for Remote User-Authentication in Wireless Networks," 2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON), Bangalore, India, 2023, pp. 1-7, doi: 10.1109/SMARTGENCON60755.2023.10442084.

[6] Paruchuri, Venubabu, Leveraging Generative AI to Streamline Account Approval Processes and Improve the Precision of Risk Assessment in Financial Services (September 30, 2024). Available at SSRN: https://ssrn.com/abstract=5473867 or http://dx.doi.org/10.2139/ssrn.5473867

[6] Faruki, P., Bharmal, A., Laxmi, V., Ganmoor, V., Gaur, M. S., Conti, M., & Rajarajan, M. (2015). Android security: A survey of issues, malware penetration, and defenses. *IEEE Communications Surveys & Tutorials, 17*(2), 998–1022. https://doi.org/10.1109/COMST.2014.2386139

[7] Fadlullah, Z., Tang, F., Mao, B., Kato, N., Akashi, O., Inoue, T., & Mizutani, K. (2017). State-of-the-art deep learning: Evolving machine intelligence toward tomorrow's intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials, 19*(4), 2432–2455. https://doi.org/10.1109/COMST.2017.2707140

[8] GIRISH KOTTE, "Leveraging AI-Driven Sales Intelligence to Revolutionize CRM Forecasting with Predictive Analytics," Journal of Science & Technology, vol. 10, no. 5, pp. 29–37, May 2025, doi: 10.46243/jst.2025.v10.i05.pp29-37.

[8] Sankar Das, S. (2024). Harnessing data lineage: making artificial intelligence smarter using data governance Frameworks. International Journal of Research and Analytical Reviews, 11(1). https://doi.org/10.56975/ijrar.v11i1.322571.

[9] Bengio, Y., & Grandvalet, Y. (2004). No unbiased estimator of the variance of k-fold cross-validation. *Journal of Machine Learning Research, 5*, 1089–1105.

[10] Yuan, Z., Lu, Y., & Xue, Y. (2016). DroidDetector: Android malware characterization and detection using deep learning. *Tsinghua Science and Technology, 21*(1), 114–123. https://doi.org/10.1109/TST.2016.7381449

[11] Saxe, J., & Berlin, K. (2015). Deep neural network-based malware detection using two-dimensional binary program features. In *2015 10th International Conference on Malicious and Unwanted Software (MALWARE)* (pp. 11–20). IEEE. https://doi.org/10.1109/MALWARE.2015.7413680

[12] G. Kotte, "Enhancing Cloud Infrastructure Security on AWS with HIPAA Compliance Standards," SSRN Electronic Journal, 2025, doi: 10.2139/ssrn.5283660.

[12] Bilar, D. (2007). Opcodes as predictor for malware. *International Journal of Electronic Security and Digital Forensics, 1*(2), 156–168. https://doi.org/10.1504/IJESDF.2007.016316

[13] Moskovitch, R., Feher, C., Tzachar, N., Berger, E., Gitelman, M., Dolev, S., & Elovici, Y. (2008). Unknown malcode detection using opcode representation. In *Proceedings of the 1st European Conference on Intelligence and Security Informatics* (pp. 204–215). https://doi.org/10.1007/978-3-540-69125-0_19

[14] Santos, I., Brezo, F., Ugarte-Pedrero, X., & Bringas, P. G. (2013). Opcode sequences as representation of executables for data-mining-based unknown malware detection. *Information Sciences, 231*, 64–82. https://doi.org/10.1016/j.ins.2011.09.028

[15] Bysani Venkata Srinivasulu1* Shaik Ali Moon2 Vijaya Kumar Reddy3 Pamuri Kasukurthy Pavan Kumar4 Pothuganti Sreedhar

Kumar5 “Stress Level Prediction using Pinball Loss Function based Quantile Regression Forest Approach” International Journal of Intelligent Engineering and Systems.-Uploaded, Vol.17, No.6, 2024 DOI: 10.22266/ijies2024.1231.08.

[15] Hashemi, H., Azmoodeh, A., Hamzeh, A., & Hashemi, S. (2017). Graph embedding as a new approach for unknown malware detection. *Journal of Computer Virology and Hacking Techniques*, *13*(3), 153–166. https://doi.org/10.1007/s11416-016-0276-4

[16] Siddiqui, M., Wang, M. C., & Lee, J. (2008). Data mining methods for malware detection using instruction sequences. In *Proceedings of the 26th IASTED International Conference on Artificial Intelligence and Applications* (pp. 358–363).

[17] Tan, Y. (2016). *Class-Wise Information Gain* (pp. 150–172). Hoboken, NJ, USA: John Wiley & Sons.

[18] Shaerpour, K., Dehghantanha, A., & Mahmod, R. (2013). Trends in android malware detection. *Journal of Digital Forensics, Security and Law*, *8*(3), Article 21. https://doi.org/10.15394/jdfsl.2013.1135

[19] David, O. E., & Netanyahu, N. S. (2015). DeepSign: Deep learning for automatic malware signature generation and classification. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8). IEEE. https://doi.org/10.1109/IJCNN.2015.7280422

[20] Pascanu, R., Stokes, J. W., Sanossian, H., Marinescu, M., & Thomas, A. (2015). Malware classification with recurrent networks. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 1916–1920). IEEE. https://doi.org/10.1109/ICASSP.2015.7178302

Received: 22-10-2025 | Accepted: 02-12-2025 | Published: 09-12-2025 | www.ajaccm.com