Attack On Sequential Embedding-Driven Attentive Malware Classification

Abhignya Bhat
Tandon School of Engineering,
New York University
MetroTech Center, Brooklyn, NY, 11201
ayb5037@nyu.edu

Avani Vaishnav
Tandon School of Engineering,
New York University
MetroTech Center, Brooklyn, NY, 11201
av3141@nyu.edu

Sharad Tembhurne
Tandon School of Engineering,
New York University
MetroTech Center, Brooklyn, NY, 11201
st4870@nyu.edu

Abstract

The escalating threat of malware poses a significant risk to the integrity of digital systems. This project addresses the crucial requirement for early detection and mitigation of malware to prevent potential catastrophic effects on devices and networks. Employing advanced natural language processing (NLP) techniques, our project presents lightweight yet highly effective classifier adaptable to a diverse range of devices. The proposed model undergoes rigorous testing on a benchmark dataset, demonstrating an impressive accuracy of 0.9908 and a log loss score of 0.07.Moreover, recognizing the necessity for robust security measures, our project includes a comprehensive evaluation of the classifier's resilience against adversarial attacks. GitHub: https://github.com/Abhignya-Bhat/ML_CyberSEC_Project

1 Introduction

Malware poses a significant threat to smart devices, ranking among the top 5 according to the Open Web Application Security Project (OWASP). Smart devices store diverse information, including sensitive data like health records, making their protection paramount. Limited resources in smart devices, such as memory and processing power, make them susceptible to attacks, especially since traditional antivirus programs are often impractical. There is a pressing need for a lightweight malware detection technique that can integrate seamlessly into embedded devices and provide swift detection to prevent exploitation.

Three primary techniques are commonly employed for malware detection. They are:

1. **Static Analysis (Signature-Based):** This technique compares the structure of a program with known malware signatures without considering its execution behavior. However, it falls short when malware exhibits polymorphic and metamorphic characteristics.
2. **Dynamic Analysis (Behavioral-Based):** Based on the program's execution behavior in a sandbox environment, dynamic analysis collects metrics like system calls and file access information. While more effective, it has limitations, particularly in identifying zero-day malware or new malware families lacking reference metrics.
3. **Heuristic-Based:** Heuristic methods are employed for detecting unknown malware families, leveraging advances in Machine Learning. This approach uses a pipeline involving

Preprint. Under review.

preprocessing techniques and machine learning algorithms to classify incoming sequences, enabling the detection of new malware and prevention of zero-day attacks.

Building on the advantages of heuristic-based techniques, our project adopts a similar approach. The project utilizes the "Microsoft BIG 15" benchmark dataset, addressing highly imbalanced classes for classification. The primary objectives of the project include developing an efficient classifier, comparing results with existing methods on the same dataset, and reporting performance after adversarial attacks .

2 Literature Review

2.1 Malware Classification

Over the recent years, various techniques rooted in machine learning, deep learning, and natural language processing have been applied for preprocessing and detecting malware in networked devices. The process involves a comprehensive analysis of files transmitted to devices to distinguish between malware and benign content.

In recent studies, sequential pattern mining technology was utilized to identify the most frequent opcode sequence patterns, aiding in malware identification. Another similar research [10] focused on constructing behavior sequence chains for certain malware families, calculating the similarity between these chains and the sequence of the target process.

Traditional signature-based methods, employed historically for malware detection in devices like mobile phones [11] primarily involve matching the structure of new files with known malware signatures. However, this approach is limited in detecting previously unidentified malware. It involves extracting features from known malware families, creating a signature database. When classifying a new program as malware or benign, its features are extracted and cross-checked against the existing signature database [12].

Antivirus vendors have relied on signature-based detection for its speed and effectiveness in identifying known threats, especially those from the same family. Nevertheless, this method has shortcomings in detecting newer malware that utilizes obfuscation and polymorphic techniques [13].

Malware binaries, typically denoted by file extensions such as ".exe" or ".bin," are malicious programs capable of harming computer operating systems. These binaries often exhibit variations with shared fundamental patterns, implying they can be categorized into multiple families. Effectively detecting malware binaries and recognizing their variations is crucial. But simply, recognizing malware binaries is not enough, the classification software should be robust enough to defend against adversarial attacks which aim to help malware evade mitigation.

2.2 Adversarial Attack on Malware classifiers

Adversarial attacks pose a significant challenge to the effectiveness of malware classifiers, which are instrumental in identifying and mitigating malicious software threats. These attacks involve the manipulation of input data by attackers to deceive the classifier and evade detection. Various types of adversarial attacks exploit vulnerabilities in machine learning models, necessitating the development of robust classifiers capable of withstanding such manipulations. In this section sections, we delve into basic information on different types of attacks and present strategies aimed at mitigating these evolving threats.

Types of Adversarial Attacks:

1. **Evasion Attacks:** In evasion attacks, adversaries craft malicious samples intentionally designed to be misclassified by the classifier. This is often achieved by making subtle changes to the input features, such as modifying specific bytes or introducing noise, while maintaining the overall functionality of the malware.
2. **Poisoning Attacks:** Poisoning attacks involve injecting malicious samples into the training dataset used to train the classifier. By influencing the training process, attackers can manipulate the model to become more vulnerable, potentially allowing the introduction of backdoors or biases that can be exploited later.

3. **Model Inversion Attacks:** Model inversion attacks aim to reverse-engineer the underlying decision boundaries of a classifier. Attackers leverage queries to the classifier to infer sensitive information about the training data, enabling them to identify potential weaknesses and devise effective evasion strategies.
4. **Transfer Attacks:** In transfer attacks, adversaries exploit the transferability of adversarial examples across different models. By crafting attacks on a surrogate model and applying them to the target model, attackers can exploit shared vulnerabilities, circumventing the need to have access to the target model during the attack crafting phase.

Defense Strategies:

1. **Adversarial Training:** One effective strategy is to augment the training data with adversarial examples, forcing the model to learn robust features and improving its ability to withstand adversarial manipulations. This approach helps the model generalize better to various attack scenarios.
2. **Input Preprocessing:** Applying input preprocessing techniques, such as input normalization and noise filtering, can make the classifier more resilient to adversarial perturbations. These techniques aim to reduce the impact of subtle changes in input features that adversaries might exploit.
3. **Ensemble Learning:** Utilizing ensemble models that combine predictions from multiple classifiers can enhance the robustness of the system. Adversarial attacks often struggle to simultaneously compromise multiple models, making ensemble methods effective in mitigating the impact of individual vulnerabilities.
4. **Continuous Monitoring and Updating:** Regularly updating and retraining the classifier with new data helps it adapt to evolving threats. Continuous monitoring of model performance and the deployment of timely updates are crucial for maintaining the effectiveness of malware classifiers in the face of dynamic adversarial tactics.

Developing robust malware classifiers is an ongoing challenge that requires a combination of advanced machine learning techniques, thoughtful design considerations, and a proactive approach to adapting to emerging threats. For our specific project focus, we prioritize addressing Evasion Attacks. In response to this threat, we employed a dual strategy combining Adversarial Training and Ensemble Learning. Adversarial Training boosts classifier robustness with adversarial examples in training, while Ensemble Learning strengthens defense by combining multiple models' strengths.

3 Data Exploration

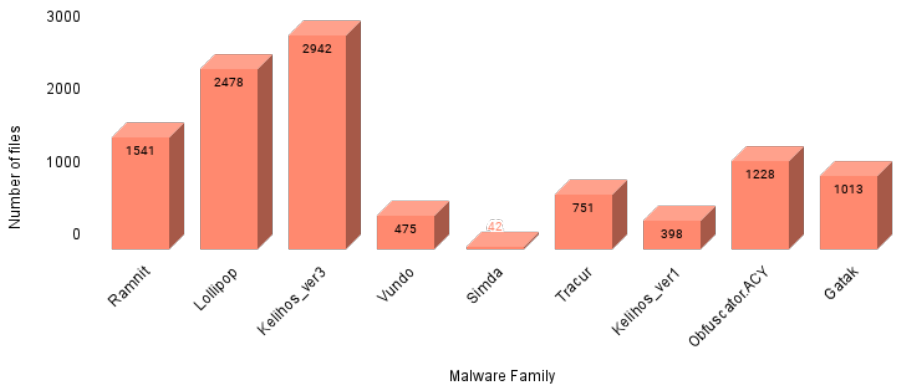
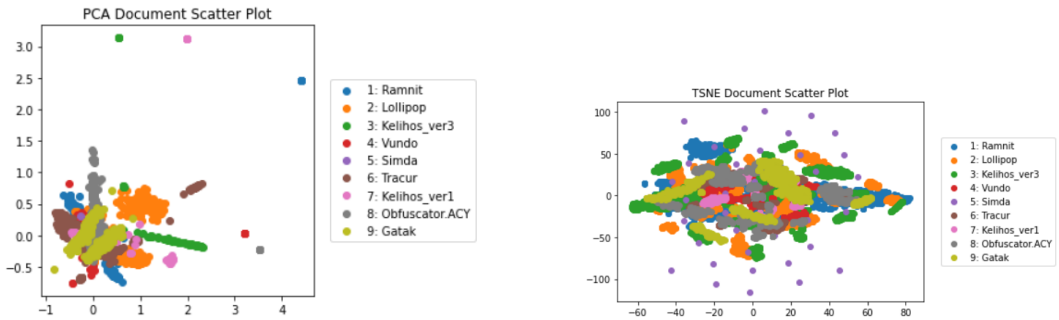


Figure 1: Distribution of files across different families.

In our project, we make use of the benchmark dataset called "Microsoft BIG 15," which Microsoft released on Kaggle [9]. This dataset consists of nine distinct malware families, presenting a significant imbalance in class distribution for the classification task. In this section, we describe the patterns

and behaviors demonstrated by different malware families by employing exploratory data analysis techniques.

1. **Sample Distribution by Class:** Within this segment, we conduct an examination of the entire dataset. Each malware family is characterized by an opcode sequence, and every malware file is assigned a unique identifier in the form of a 20-character hash value (Id). Figure 1 illustrates the distribution of files across different families.
2. **Embedding Clusters:**Delving into the realm of embedding clusters, we delve into the resemblances and distinctions among clusters originating from diverse families, shaped by the acquired embeddings of malware opcodes. The training of our model involves the utilization of the opcode sequences as tokens, treating each opcode as a distinct word. Subsequently, document embedding vectors are constructed by computing the average of tokens within each document. Figures 3 (a) and (b) visually present the Principal Component Analysis (PCA)components and t-SNE components, respectively, of these 100-dimensional document embeddings.



(a) Principal Component Analysis components of document embedding.

(b) t-SNE components of document embedding.

Figure 2: Embedding Clusters

4 Proposed Methodology

4.1 Model Architecture

The diagram in Figure 8 outlines our proposed methodology, aimed at creating an efficient yet effective malware classifier in terms of time and computing resources. Balancing performance metrics and real-time efficiency often involves trade-offs. Our approach[1] involves two key principles: 1) Select appropriate methods at each stage, and 2) Use efficient alternatives to reduce time consumption. The goal is to develop a classifier that can identify hidden patterns without bias towards the majority class.

1. **Learning Opcodes Embedding:** The focus is on learning both context and semantics of opcodes. Context refers to understanding how an opcode is used in a specific context, while semantics involves learning a special representation vector for each opcode. We learn the embeddings by sequentially passing the opcodes through a hidden layer, and then averaging it for the entire sequence
2. **Sequential Block:** The sequential block processes the embeddings using LSTM cells to capture long-term dependencies in extensive opcode sequences. The input gate filters out irrelevant information by assigning importance scores through a sigmoid activation. These scores are multiplied with the previous cell output to retain crucial details. The LSTM processes information through a tanh activation, combining it with preserved data from the forget gate. The output gate produces the final learned vector, passing it to the next cell after a tanh activation.To address the vast data and enhance diversity, a parallel architecture incorporates a GRU alongside LSTM. The GRU, a time-efficient alternative, complements LSTM for comprehensive sequence understanding.
3. **Averaging Mean Values:** Two feature vectors are obtained from the LSTM and GRU, representing opcode information, long-term relations, and hidden contextual patterns. These

vectors are combined by cascading their outputs to the same dimensions. A simple mean is taken to center their means, eliminating extreme values.

4. **Attention Block:** The Attention Block is a pivotal component of the model architecture, emphasizing the prioritization of important information while filtering out less critical details. . This mechanism, lightweight yet effective, ensures that the model doesn't overlook the minority malicious parts while overly focusing on the majority bengin parts of the opcode embedding. By preparing an attention vector and taking the dot product with the embeddings, the model accurately assigns attention to the critical malicious part, contributing to an accurate and efficient classification process. While this ends the Sequential Embeddings based Attentive Classifier, for our project we combine it with Gradient Boosted Classifiers.
5. **Gradient Boosted Classifiers:** Two gradient boosted classifiers are employed to classify the malware based on opcode frequency. This is in line with traditional methods which focus on frequency and signature based analysis. The output from the attention block and these classifiers are averaged to give a final prediction

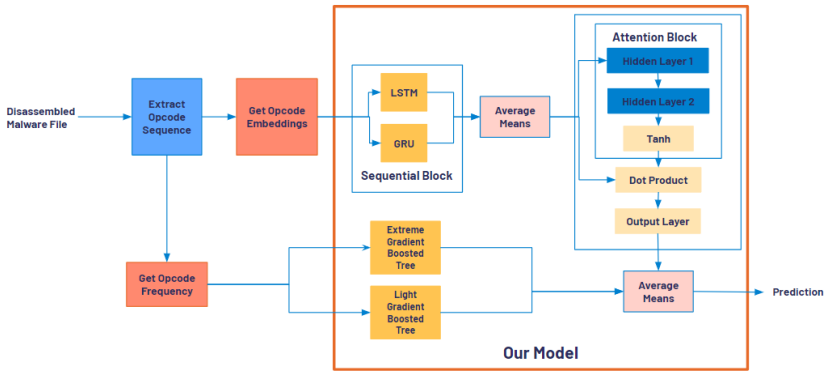


Figure 3: Model Architecture.

4.2 Attack Strategies

While having effective classifiers is crucial, it is equally important that they exhibit resilience against attacks. Therefore, this project seeks to examine the vulnerabilities of models against the following four attack strategies, as outlined in this section.

1. **Append Attack:** Append-based strategies tackle the semantic integrity constraints of PE files by adding adversarial noise to the original file's end. As this noise is appended at the conclusion, it remains beyond the PE file boundary, ensuring it does not impact the file's execution. In this project, we applied this approach by adding 500 push commands to the end of each opcode sequence as an adversarial attack.
2. **FGSM Append Attack:** The attack strategy is designed to underscore the vulnerabilities of the model, particularly as adversarial leverage increases. Drawing inspiration from the append-based approach, the FGSM Append Attack involves augmenting extracted opcode sequences with push commands. Subsequently, these sequences are updated using a policy guided by the Fast Gradient Sign Method (FGSM). In this method, the attack leverages the classification loss (l) of the model's output concerning the target label. FGSM operates by adjusting each embedding value by a user-specified amount (epsilon) in a direction that minimizes the classification loss on the input. This adjustment is determined by the sign of the gradient, aligning with the overall objective of enhancing the model's susceptibility to adversarial inputs.
3. **Rule Based Attack:** The Rule-Based Attack method operates by adhering to a predefined set of rules that guide the introduction of perturbations after each opcode in the sequence. Following these rules, the generated embeddings undergo adjustments by a small value epsilon, aligning with the direction that minimizes the loss—a methodology reminiscent of

the Fast Gradient Sign Method (FGSM) attack. In the context of our project, this strategy entailed injecting additional push commands for each existing push command in the original file. Additionally, extra XOR operations were introduced for every XOR command in the sequence. Similar to the Append Attack, these injections are made only by setting the rules such that the execution of the PE file is undisturbed. Subsequently, an FGSM attack was applied to the adjusted embeddings, further exploiting the model's vulnerability to adversarial inputs through these rule-based modifications.

4. **Feature-Importance Based Attack:** In this approach, push commands are appended to the end of the opcode sequences, resembling the Append Attack. However, the Feature-Importance Based Attack introduces complexity by assigning priority to values within the embeddings based on their feature importance. Typically utilized for removing redundant features and improving model training efficiency, the feature selection algorithm takes on a unique role in this attack, contributing to the construction of adversarial samples. After employing various feature selection algorithms, the attack ranks the features. Perturbations are then applied to a small fraction of these prioritized values, representing the most crucial features. In our project, feature importance was assessed by averaging scores from ANOVA F-value, Information Gain, and the FDE feature selection algorithm. Following this, 20 out of the total 100 values in the embedding space were updated, showcasing a focused and well-informed adversarial strategy.

It's important to highlight that the rationale behind implementing each of the attacks alongside the append attack was to validate the efficacy of the model's sequential and self-attentive characteristics. The primary objective of all attacks, excluding the append attack, is not only to target the frequency analysis aspect but also to disrupt the embedding space. This approach enables aims an exploration of the model's resilience and effectiveness against a broader spectrum of adversarial challenges.

4.3 Adversarial Retraining and Ensemble Learning

Additionally, we performed model retraining on both original and adversarial datasets, assessing their performance against both normal and perturbed samples. Our analysis extended to various other models, and we integrated them with the proposed SEA classifier. This comprehensive approach was that we could us to identify the most effective combinations among the models and the SEA classifier.

5 Results

This section presents outcomes of our proposed approach. Additionally, it includes a comparative analysis with state-of-the-art methods and a comprehensive evaluation of model performance in conjunction with other models against diverse attack strategies.

1. **Comparative results with state of the art classifiers:** From Table 1 its clear our model showcases competitive performance, boasting a commendable accuracy of 99.08% and a log loss of 0.0795, with a moderate number of trainable parameters at 811,530.It's noteworthy that our model compares favorably with existing models. Each model excels in specific aspects, but our approach strikes a balance, offering efficiency with fewer parameters while maintaining high accuracy.

Table 1: Model Comparison

Model	Accuracy	Logloss	Trainable Parameters
Our model	0.9908	0.0795	811,530
Original Model [1]	0.9912	0.0431	812,030
Hierarchical CNN [2]	0.9913	0.0419	20,863,302
CNN with Structural Entropy [3]	0.9828	0.0750	900,148
HYDRA [4]	0.9975	-	2,666,617
Orthus [5]	0.9924	-	973,989

2. **Comparative results of the Attack and Retraining:** In our project, we conducted experiments using the Structural Embedding Adversarial (SEA) technique alongside different

classifiers such as the original Gradient Boosted Classifier, Logistic Regression, Support Vector Machine (SVM), and Random Forest classifiers. Specifically, all models underwent evaluation against the append attack, while only those integrated with SEA were subjected to testing against FGSM Append Attack, Rule-Based Attack, and Feature Importance Attack. This choice was influenced by the conventional use of machine learning models without embedding space, focusing solely on opcode frequency analysis.

Tables 2 and 3 consistently highlight the superior performance of our model across different scenarios when compared to other models. It is evident that basic machine learning pipelines do not match up to the baseline performance of our proposed model, even when facing a simple append attack. Notably, the possibility of using a Random Forest classifier in conjunction with is observed as an alternative to the initially recommended Gradient Boosted Classifier. However, it is crucial to acknowledge that the Random Forest classifier shows a specific vulnerability to gradient-based attacks, as elaborated below.

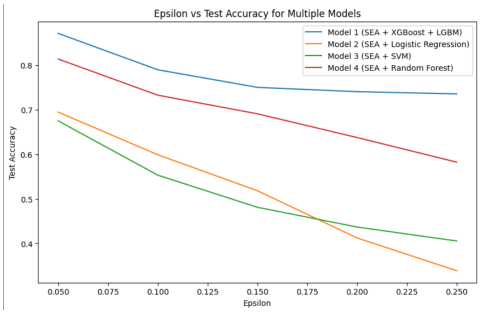
Table 2: Comparative Analysis of Append Attack and FGSM Append Attack

Model	Test Accuracy	Append Attack		FGSM Append Attack($\epsilon = 0.15$)	
		Before Retraining	After Retraining	Before Retraining	After Retraining
Our model: SEA + XGB + LGBM	0.9908	0.8991	0.9978	0.8719	0.9971
SEA + Logistic Regression	0.9328	0.8952	0.9377	0.695	0.9622
SEA + SVM	0.9425	0.9055	0.953	0.695	0.9775
SEA + Random Forest	0.95814	0.9315	0.9646	0.8143	0.9877
XGB + LGBM	0.9885	0.7478	0.99788	-	-
Logistic Regression	0.9144	0.8253	0.9146	-	-
SVM	0.965	0.6403	0.9644	-	-
Random Forest	0.9885	0.89915	0.99788	-	-

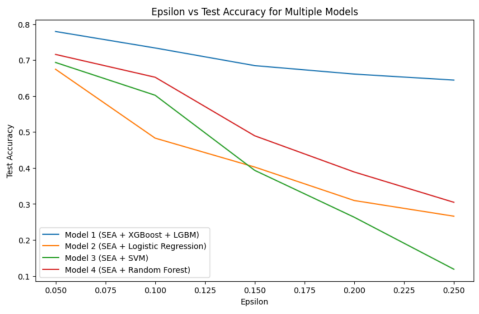
Table 3: Comparative Analysis of Rule-Based Attack and Feature Importance Based Attack

Model	Test Accuracy	Rule-Based Attack ($\epsilon = 0.15$)		Feature Importance Based Attack features=20 update=0.25	
		Before Retraining	After Retraining	Before Retraining	After Retraining
Our model: SEA + XGB + LGBM	0.9908	0.7794	0.9978	0.8958	0.9978
SEA + Logistic Regression	0.9328	0.6744	0.9567	0.8740	0.9367
SEA + SVM	0.9425	0.6744	0.9773	0.8740	0.9521
SEA + Random Forest	0.95814	0.7156	0.9858	0.9135	0.9631
XGB + LGBM	0.9885	-	-	-	-
Logistic Regression	0.9144	-	-	-	-
SVM	0.965	-	-	-	-
Random Forest	0.9885	-	-	-	-

Upon scrutinizing the attacks, it becomes evident that gradient-based attacks result in more significant performance drops when compared to append and Feature Importance attacks. Furthermore, Rule-Based attacks consistently lead to the most substantial degradation in performance across all models. Figure 4 (a) and (b) visually illustrates that as epsilon increases, performance continually deteriorates. Notably, our proposed model shows a plateau in its degradation, while other combinations degrade to the point of almost complete misclassification. Additionally, the figures indicate that the SEA and Random Forest Combination struggle against rule-based attacks, suggesting a potential overfitting of the Random Forest Classifier to the dataset due to its small size.



(a) Epsilon vs Test Accuracy for FGSM Append Attack



(b) Epsilon vs Test Accuracy for Rule-Based Attack

Figure 4: Performance against gradient based attacks

Finally, it is noteworthy that the models demonstrated unexpected robustness against Feature Importance Based attacks, even when these attacks targeted both the frequency and embedding domains.

6 Conclusion

We implemented a malware detection model that goes beyond traditional frequency analysis by focusing on embeddings alongside frequency analysis. This lightweight model utilizes sequential opcode-based embeddings and an attentive mechanism for classification. Our model demonstrates resilience against Append attacks, with only a minimal decrease in Test Accuracy from 0.9908 to 0.8991. This highlights the effectiveness of our proposed attention mechanism in filtering out appended adversarial noise. However, the model is susceptible to gradient-based attacks, such as the FGSM attack, resulting in a significant Test Accuracy drop from 0.9908 to 0.7359. Interestingly, our model's inherent defense partially succeeds, maintaining Test Accuracy at around 0.74 even with increasing epsilon values. Additionally, a rule-based attack leads to a substantial Test Accuracy drop from 0.9908 to 0.6443. Notably, no other combination of SEA and machine learning models, aside from our originally proposed model, proves resilient enough. To address this vulnerability, we implemented a successful adversarial retraining defense, resulting in a remarkable improvement in Test Accuracy to nearly 0.9971 after retraining for both gradient-based attacks. This underscores the importance of efficient and robust malware classifiers, like the one we've developed, in addressing the evolving landscape of cybersecurity threats.

References

[1] M. Ahmed, A. Qureshi, J. Ahmed Shamsi, and M. Marvi, "Sequential Embedding-based Attentive (SEA) classifier for malware classification," 2022 International Conference on Cyber Warfare and Security (ICCWS), Islamabad, Pakistan, 2022

[2] D. Gibert, C. Mateu and J. Planes, "A Hierarchical Convolutional Neural Network for Malware Classification," 2019 International Joint Conference on Neural Networks (IJCNN), 2019

[3] Gibert, D., Mateu, C., Planes, J. and Vicens, R., 2018, April. Classification of malware by using structural entropy on convolutional neural networks. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 32, No. 1).

[4] Daniel Gibert, Carles Mateu, Jordi Planes, HYDRA: A multimodal deep learning framework for malware classification, Computers & Security, Volume 95, 2020

[5]Daniel Gibert, Carles Mateu, Jordi Planes,"Orthus : A Bimodal Learning Architecture for Malware Classification," 2020 International Joint Conference on Neural Networks (IJCNN), 2020,

[6] Suciu, Octavian, et al. "Exploring adversarial examples in malware detection." 2019 IEEE Security and Privacy Workshops (SPW), 2019, https://doi.org/10.1109/spw.2019.00015.

[7] Kivlichan, David. Rule-Based Adversarial Learning: Disguising and Recognizing Maliciousness in Opcodes. Diss. University of Toronto (Canada), 2021.

[8]Li, X., et al.: Feature selection-based android malware adversarial sample generation and detection method. IET Inf. Secur. 15(6), 401–416 (2021). https://doi.org/10.1049/ise2.12030

[9]Ronen, R., Radu, M., Feuerstein, C., Yom-Tov, E. and Ahmadi, M., 2018. Microsoft malware classification challenge. arXiv preprint arXiv:1802.10135

[10] Zhang, Y., Huang, Q., Ma, X., Yang, Z., and Jiang, J., 2016, August. Using multi-features and ensemble learning method for imbalanced malware classification. In 2016 IEEE Trustcom/BigDataSE/ISPA (pp. 965-973). IEEE.

[11] Venugopal, D. and Hu, G., 2008. Efficient signature based malware detection on mobile devices. Mobile Information Systems, 4(1), pp.33- 49.

[12] K. Alzarooni ., 2012, "Malware variant detection," Ph.D. dissertation, Dept.Comput. Sci., Univ. College London, London, U.K

[13] Saxe, J. and Berlin, K., 2015, October. 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