Not All Malware are Born Equally: An Empirical Analysis of Adversarial Evasion Attacks in Relation to Malware Types and PE Files Structure

Prabhath Mummaneni*, Kshitiz Aryal†, Mahmoud Abdelsalam*, Maanak Gupta†
*Department of Computer Science, North Carolina A&T State University, Greensboro, North Carolina, USA
†Department of Computer Science, Tennessee Tech University, Cookeville, Tennessee, USA
Email: *pmummaneni@aggies.ncat.edu, †karyal42@tntech.edu, *mabdelsalam1@ncat.edu, †mgupta@tntech.edu

2024 IEEE International Conference on Big Data (BigData) | 979-8-3503-6248-0/24/$31.00 ©2024 IEEE | DOI: 10.1109/BigData62323.2024.10826102

***Abstract*—Malware white-box evasion attack is a serious threat to machine learning-based malware classification models, where an attacker carefully inserts perturbations into a malware executable at a test time to evade a target model. Previous research introduced different white-box evasion attacks, namely padding and slack attacks, to craft malware adversarial samples and evaluated them based on the perturbation size and their evasion rate against a target model. However, there is a lack of insights into how the malware file structure and type affect the adversarial malware sample generation and their respective evasion rate. In this work, we provide a comprehensive empirical analysis by factoring in the malware structure and the type. Our analysis quantifies slack space availability in various sections, exploring how the slack space can influence the robustness of detection techniques. We further assess the relationship between malware type and evasion rate to understand how different types of malware respond to evasion attacks. Additionally, we explore the connection between each malware type and the corresponding slack space availability, analyzing how these structural factors influence the evasion rates during adversarial attacks. In our experiments, adversarial malware samples were generated using two different algorithms: gradient descent and iterative gradient sign method. This detailed analysis enhances our understanding of evasion dynamics of adversarial attacks across malware types and different structural characteristics of binary malware files.**

***Index Terms*—Adversarial Malware Analysis; Windows Malware Detection; Append and Slack Attack; Malware PE File Structure**

I. Introduction

Artificial Intelligence (AI) and Deep Learning (DL) have seen significant growth in recent years, becoming a cornerstone in the cybersecurity domain [1], [2]. In the field of malware, deep learning models have emerged as powerful tools for detection and classification. In particular, MalConv [3] is a notable end-to-end CNN model due to its ability to ingest entire executable files and effectively classify them as malicious or benign.

This advancement has also spurred the development of sophisticated adversarial attacks designed to target deep learning models by exploiting their inherent sensitivity to input perturbations. Among these adversarial attacks, optimization algorithms, like Gradient Descent (GRAD), are known to be most effective. In addition, a significant advancement in this field came with the introduction of the Fast Gradient Sign Method (FGSM) by Goodfellow et al. [4]. Building upon FGSM, the Iterative Gradient Sign Method (IGSM) was developed by Madry et al. [5], offering a more refined approach to generating adversarial examples. Goodfellow et al. and Madry et al. demonstrated that using FGSM and IGSM causes DL models to misclassify images by adding carefully crafted minimal adversarial noise. However, a similar straightforward addition of perturbations to malware executables can lead to the breaking of the malware functionality.

To address this issue, research works [6] proposed approaches that limit the injection of perturbation within the executable files to specific areas, keeping their functionality intact while evading the malware detector. For instance, Kruek et al. [7] applied IGSM in the field of adversarial malware analysis and introduced techniques known as the Append and Slack attacks. Append attack, also known as padding attack, pads perturbation at the end of the file, while slack attack adds perturbations in the empty slack spaces inside the PE file sections. The majority of research works have focused on crafting perturbations to enhance the evasion rate (ER) without detailed consideration of the characteristics of the malware being perturbed. We believe factoring in the characteristics of each malware binary can provide a more comprehensive understanding of generating adversarial malware samples, which leads to more successful evasion attacks.

In our work, we have considered three factors throughout the adversarial analysis of binary malware: the amount of perturbations appended at the end of the file, the slack space availability across various sections of the PE file and the malware type. Throughout this paper, we will be referencing malware ”type” or ”family” interchangeably, but both mean the same meaning as classification based on their characterization. Our exploration into these factors raised several pertinent research questions.

RQ1. How does increasing padding size affect the evasion rate of malware? Does the choice of adversarial algorithm influence this behavior?

RQ2. Does the availability of slack space consistently enhance malware detection evasion? How does the choice of adversarial algorithm impact this relationship, and does the location of slack space within the malware affect its evasion effectiveness? Additionally, how does the effectiveness of slack space compare to padding in evading malware detection?

RQ3. Does the type of malware influence its evasion capabilities? How do different malware types respond to various adversarial techniques and algorithms? Additionally, what are the characteristics of each malware family regarding slack space, and how does slack space affect each family's evasion effectiveness?

To answer these, we have conducted a series of experiments comprising three adversarial evasion attacks, namely append, slack, and hybrid attack, which combines the strategies of both append and slack attacks, using both GRAD and IGSM algorithms. All of our attacks target the MalConv [3] model, a well-known end-to-end benchmark model for malware detection in the adversarial domain.

The remainder of this paper is organized as follows. Section II provides the background and related work. Section III outlines the methodology including MalConv (the target model), IGSM and GRAD (adversarial algorithms used), the malware dataset, and the experimental setup. Section IV explains the how different attacks are performed. Section V analyzes the results of the experiments and conducts a discussion pointing towards the answers to the research questions. Finally, Section VI provides potential future direction and concludes the work.

II. Background and Related Work

A. Machine Learning-based Malware Detection

Conventional detection methods such as signature and behavior-based approaches often require "human-in-the-loop" (i.e. analysts), which hinders the malware analysis process significantly. Evidently, with the creation of three to four new malware variants per second [8], such methods struggle to keep up with the diversity and complexity of the malware threat, particularly in the case of zero-day, polymorphic, and metamorphic malware.

The rapid emergence of new malware necessitates a shift towards more advanced and adaptable detection methods. As such, ML-based approaches have been used in the malware domain for static, dynamic and real-time online analysis. In the case of static file detection, ML models are trained on features like hash value, particular string information, opcodes, n-bytes, file system, registry key changes, process operations, network activities and have achieved 99% accuracy using ML classifiers [9], [10], [11], [12], [13], [14].

Beyond traditional machine learning techniques, deep learning models like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) have become pivotal in advancing malware detection. These models are adept at extracting complex patterns from large datasets, crucial for identifying sophisticated, polymorphic, and zero-day malware. With their ability to automatically learn and generalize from raw data, deep learning approaches offer a dynamic solution to the limitations of earlier methods, enhancing the detection accuracy and reducing false positives in the ever-evolving landscape of cyber threats [15], [16], [17], [18].

B. Adversarial Machine Learning

Machine learning (ML) has seen remarkable growth, significantly impacting various sectors. Yet, the effectiveness of ML models solely depends on a critical assumption that train data and test data belong to the same distribution. In real-world situations, we can't guarantee the data comes from a similar distribution, particularly when attackers can intentionally tweak the test or train data to fool the model. These attacks, referred to as adversarial attacks, have become a severe threat in the ML landscape.

The majority of adversarial attacks comprise either data poisoning attacks at training or evasion attacks during model testing. Data poisoning attacks are hard to achieve due to the improbable possibility of attackers' access to training data. On the other hand, evasion attacks can always evade the target model by intentionally tweaking data at the testing/deployment phase. For example, Goodfellow et al. [4] demonstrated how GoogLeNet [19], after being trained on the ImageNet dataset [20], incorrectly identifies a panda as a gibbon when small calculated perturbations are added to the image.

In this paper, we focus on *evasion adversarial attacks*, where the perturbation is carefully crafted and inserted into a malware file, leading to a change of classification from malicious to benign by the target model.

C. Adversarial Malware Analysis

Adversarial attacks have been a rising threat in the domain of malware analysis, especially for Windows malware binaries. Consequentially, more research has been focusing on adversarial malware analysis. Kolosnjaji et al. [21] applied gradient-based methods to manipulate malware executables and create an adversarial malware sample. They appended padding bytes to the malware binary, allowing these files to bypass deep learning-based malware detectors while retaining their malicious functionality. Their approach demonstrated the effectiveness of gradient-informed modifications in evading detection. They randomly sampled just 200 samples and achieved an evasion success of 60%.

Kreuk et al. [7] expanded the realm of malware adversarial attacks by adopting IGSM to the context of malware binaries. They explored two key strategies: appending additional bytes to the end of malware files and refining mid-file perturbation insertion using IGSM. These methods showcased the potential for both appending bytes and intelligently inserting perturbations to increase the likelihood of evading deep learning-based detection systems. Further, Suciu et al. [22] analyzed adversarial examples in malware detection. They examined various attack methods, including appending different types of adversarial noise to the end of malware files and altering the slack regions (i.e. empty spaces within or between the sections of an executable). Their research compared the effectiveness

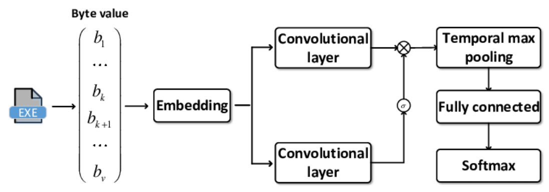

Fig. 1. MalConv Architecture[3]

of these methods, such as using random noise, IGSM modifications, and benign byte appendages. They have randomly sampled 400 Malware samples and performed append attacks, achieving a maximum accuracy of 71% by padding 10000 Bytes. For the slack regions-based attacks, they have achieved a maximum accuracy of 28%. The most recent work in this field is by Kun et al.[23], who developed the FGAM method to optimize the number of alterations in malware and determine the optimal stopping point for introducing perturbations, ensuring the generated adversarial malware remains effective and efficient.

Previous research focused on enhancing the ER by improving the quality of perturbations inserted, using various techniques, such as random noise, benign bytes, and algorithms like GRAD and IGSM. It also focused on where to insert such perturbations, mainly appending them at the end or within the slack spaces.

Unlike previous works, we focus on analyzing the structure and characteristics of the malware executable rather than just the adversarial techniques. As such, we conduct a comprehensive analysis of evasion attacks, focusing on the role of slack space in malware executables. We also include an in-depth exploration of adversarial evasion across various malware types.

III. Methodology

A. MalConv Detector

MalConv is an end-to-end deep learning model specifically designed for malware detection. Proposed by Raff et al. [3], MalConv stands out for its innovative approach to ingesting an entire PE file (and padding/trimming it to 2MB) and processing raw byte sequences. Its architecture is primarily characterized by the presence of an embedding layer, which is crucial for converting raw byte values into a meaningful vector representation. Beyond the embedding layer, MalConv's architecture as shown in Figure 1 includes convolutional layers that are instrumental in extracting local patterns and features from the embedded byte sequences, which are crucial for identifying malicious signatures. The model culminates in a fully connected output layer, which integrates these extracted features to make the final classification decision between benign and malicious files. The effectiveness of MalConv is attributed to its ability to capture intricate patterns within executables, which might be overlooked by conventional malware detection methods that rely on hand-crafted features. This capability makes it a powerful tool for identifying both known and unknown malware variants. As MalConv became a well-known standard as an end-to-end malware detector in academic research, we chose it as a target model for conducting our experiments.

B. Adversarial Algorithms

Gradient Decent (GRAD): Gradient descent is an optimization approach that iteratively adjusts variables to minimize a target function. This quality makes it an invaluable tool in adversarial contexts, particularly for crafting subtle modifications for adversarial malware examples. Unlike brute force methods that make blatant changes, gradient descent fine-tunes the parameters, ensuring efficacy. Its special prowess lies in its ability to identify and exploit the weakest link in a system's defence by making calculated, incremental adjustments. In the adversarial landscape, this translates to a higher likelihood of evading detection, making gradient descent a technique of choice for sophisticated cyber threats.

Iterative Gradient Sign Method (IGSM): IGSM [24] is a technique that builds upon the foundations of the Fast Gradient Sign Method (FGSM). The FGSM introduced a method of generating adversarial examples by applying a single-step perturbation in the direction of the gradient of the loss function. This approach effectively utilized a linear approximation to alter input data, nudging it towards increasing the loss, thereby challenging the model's accuracy. The FGSM's simplicity and efficiency in creating adversarial samples made it a pivotal tool in understanding neural network vulnerabilities. However, the FGSM's straightforward approach also presented limitations, particularly in its ability to generate finely tuned-adversarial examples. This shortcoming stemmed from its one-step nature, which often resulted in either insufficiently subtle or overly aggressive perturbations. To address these drawbacks, the IGSM was introduced. IGSM enhances FGSM's methodology by introducing an iterative process where smaller, controlled perturbations are applied multiple times. This allows for a more gradual and adjustable manipulation of the input data, ensuring that the adversarial examples remain within a defined ϵ-boundary of the original image.

C. Adversarial Techniques

Unlike images, adding perturbations to malware is not straightforward since the malware's functionality has to be maintained. As such, append and slack attacks were introduced in previous works, aiming to create adversarial malware samples without compromising their functionalities. Further, we also analyze the combination of the append and slack attacks, namely a hybrid attack. Figure 2 shows the locations within the PE executable where the perturbations are inserted for each of the three adversarial techniques. Each adversarial technique is discussed as follows.

Append Attack: The append attack, depicted in Figure 2-A, involves appending additional non-malicious bytes to the end of a malware file. This technique doesn't alter the malware's functionality but can significantly impact how detection models based on feature extraction, such as MalConv, perceive the file. Kolosnjaji et al. [21] introduced this approach in the malware binary.

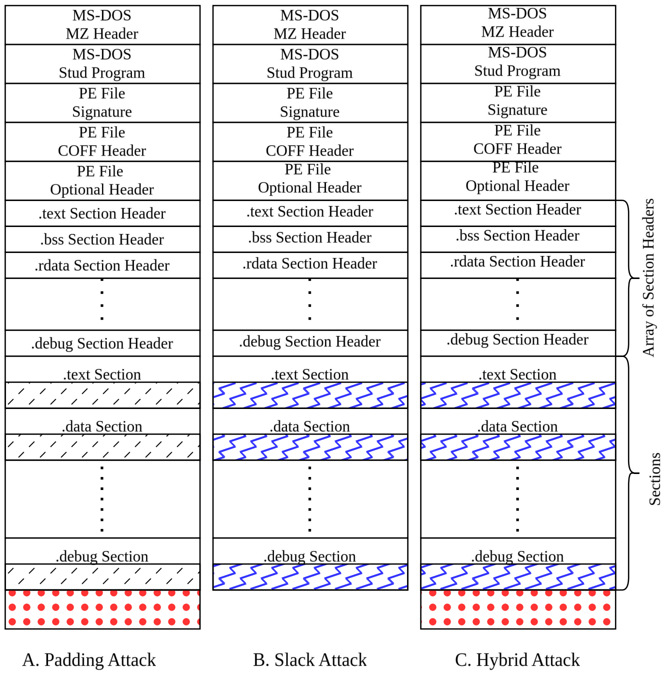

Fig. 2. Adversarial techniques within PE files.

Slack Attack: It is an adversarial evasion attack (shown in Figure 2-B) that exploits the unused or residual space in a file executable, commonly known as 'slack space'. Windows binaries are saved in physical memory in clusters. The last cluster might not be completely occupied, raising the possibility of empty space. These empty spaces are called slack spaces and are identified when the physical size is greater than the virtual size. The slack attack involves embedding adversarial data within this slack space. Using these unutilized areas allows one to insert harmful content into a system without altering the file's apparent size or structure. In our study, we utilized all available slack space to demonstrate the full extent of this attack vector. This attack was first analyzed by Kreuk et al. [7]

Hybrid Attack: In the context of adversarial malware evasion, it is important to note that each technique possesses its own distinct set of limitations. An append attack entails the addition of extraneous data to the end of a file. However, this method may not consistently result in successful evasion, as it lacks a comprehensive integration with the underlying content of the file. It also raises suspicions by the malware detectors, as it is relatively easy to look for out-of-place extra bytes at the end of the file. The utilization of slack space for perturbations, although less conspicuous, is commonly constrained by the quantity of unutilized space available, frequently leading to minimal perturbations that may not adequately deceive advanced detection systems, in our case, MalConv. Therefore, to better analyze the patterns and the behavior of the factors in our consideration, we have combined these two attacks by inserting perturbations both at the end of the file (append) and the mid-file injection (slackspace) to form a hybrid attack as shown in Figure 2-C. The hybrid has all the factors pertaining to both attacks and compliments each other's drawbacks, resulting in better malware evasion. This can be further evidenced in Section IV.

D. Windows PE File Structure & Malware Dataset

PE File Format: Adversarial malware generation requires careful modification of the malware executable. As such, it is imperative to understand the malware file format. The executable file format for the Windows operating system is Windows PE file format. The PE file (shown in Figure 2) is a linear stream of data, starting with the MS-DOS header. MS-DOS stub program is followed by a PE file signature, the COFF file header, and an optional header. This is followed by a sectional header, which contains metadata of sections like physical address, virtual size, virtual address, etc. All executable code and the entry point are present in the `.text` section. The `.bss` section holds uninitialized data for applications, encompassing all statically declared variables. The `.rdata` section is designated for read-only data, such as constants, strings, and debug directory details. The actual data related to the file is stored in the `.data` section. Additionally, the `.rsrc` section is reserved for storing resource information pertinent to the module. In this paper, we only discuss the PE file format since we are focusing on Windows malware.

Malware Dataset: The dataset comprises of $13,971$ latest Windows malware samples collected from VirusTotal[1], a widely recognized and trusted public database for malware analysis. VirusTotal is a remarkable resource known for its extensive and diverse sample collection.

Out of these $13,971$ samples, only $8,107$ of them are classified as malware against the MalConv model with a threshold of 0.5. Since only $8k$ binaries are only detected as malicious by Malconv model, we have performed adversarial activities on these 8k files only. These 8k files are classified using VirusTotal API and are classified into 12 different malware types with 6 types consisting of almost 500 samples.

E. Analysis Setup and Evaluation

In conducting this comprehensive analysis of malware and adversarial techniques, we employed the SECML Malware[2] library implementation of adversarial algorithms (IGSM, Gradient Decent) and utilized SECML's pre-trained model of MalConv as the target. To efficiently process and execute our experiments, we utilized the powerful Lambda-workstation with 3x NVIDIA RTX 4090 24GB GPUs [3].

For the evaluation of our experiments, we focused on a key metric called Evasion Rate (ER). The ER serves as a primary indicator of the effectiveness of our adversarial strategies. It is calculated as the percentage of malware samples that successfully evade detection after being altered by our adversarial techniques, as follows:

$$\text{ER} = \left(\frac{\text{No. of Malware Samples Evaded}}{\text{Total Number of Malware Samples}} \right) \times 100$$

[1]VirusTotal. https://www.virustotal.com/
[2]SECML Malware. https://github.com/pralab/secml_malware
[3]Lambda-Workstation. https://lambdalabs.com/gpu-workstations/vector

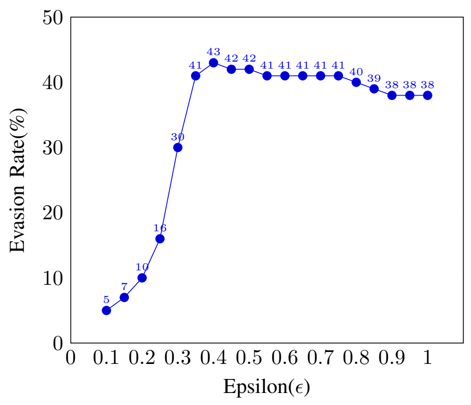

Fig. 3. ER Across Different Epsilon under Append$^{\text{IGSM}}$

Further in the analysis, while talking ER for a selection like ER of malware having 1000-2000 bytes of slackspace or ER for a Worm family, we are considering the total number of malware samples belonging to the selection. In our analysis, we discuss ER with respect to slack spaces and append bytes for various malware families/types under the IGSM and GRAD adversarial algorithms.

IV. Experiments Overview

A. Adversarial Hyperparameter Exploration

Every optimization technique has hyperparameters that need to be tuned to achieve a high success rate for the end goal. This section presents the experiments to optimize hyperparameters and the final optimal values achieved to achieve maximum ER.

1) Epsilon (ϵ): With FGSM being the core of IGSM, Epsilon (ϵ) plays a pivotal role in the Fast Gradient Sign Method (FGSM) by dictating the magnitude of adversarial perturbations. The influence of ϵ on the perturbation is quantified by the formula:

$$perturbation = perturbation + \epsilon \times \text{sign}(\nabla_x J(\theta, x, y))$$

In this formula, $\nabla_x J(\theta, x, y)$ represents the gradient of the loss with respect to the input data, θ denotes the model parameters, x is the input, and y is the target label. The term $\text{sign}(\nabla_x J(\theta, x, y))$ computes the sign of the gradient, indicating the direction in which the input should be modified to maximize the loss. The product $\epsilon \times \text{sign}(\nabla_x J(\theta, x, y))$ represents the actual perturbation applied to the input data. A larger value of ϵ results in a larger step in the direction of the gradient sign.

Our experimental approach to finding the optimal ϵ value is focused on evasion effectiveness. We explored ϵ values from 0.1 to 1.0, in increments of 0.05, to understand how varying levels of perturbation intensity impact ER.

The results, visualized in Figure 3, demonstrated a peak ER at $\epsilon = 0.4$. This suggests an optimal balance at this value, where the perturbations were significant enough to deceive the detection model yet subtle enough to avoid being conspicuous. All subsequent experimental results provided herein are obtained with an epsilon value of 0.4.

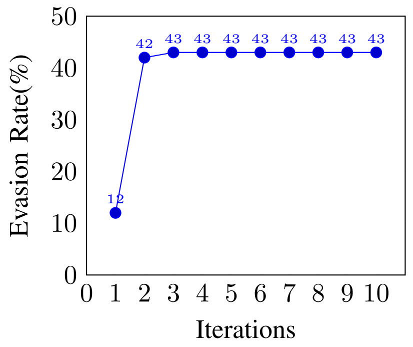

Fig. 4. ER Across Different Iterations under Append$^{\text{IGSM}}$

2) Iterations: The concept of iterations takes on significant importance in adversarial machine learning, particularly in the context of IGSM. In this setting, 'iterations' refer to the number of times an adversarial perturbation is applied and adjusted in an attempt to deceive the target model.

To investigate the influence of iterations on the effectiveness of the IGSM attack, we conducted a detailed experiment. The primary goal was to determine the optimal number of iterations that maximized the ER against MalConv. For consistency and comparability, we employed an append attack strategy with a fixed padding length of 1024 bytes. Our experiment varied the number of iterations from a baseline of 1 iteration, incrementing up to the 10th iteration.

The empirical results, as illustrated in Figure 4, revealed a pattern where the ER stabilized at 43% for iteration counts from 3 to 10. This plateau suggests that increasing the number of iterations beyond a certain point (in this case, 3 iterations) does not significantly enhance the attack's efficacy. This finding provides crucial insights into the diminishing returns of additional iterations in IGSM.

B. Experimental Approach for Append, Slack, and Hybrid Attacks

In this subsection, we discuss various details and parameters that contribute to and shape the experiments conducted using append, slack, and hybrid attacks. All the following experiments are conducted on the 8k files that were detected as malicious by the target MalConv model.

1) Append Attack: To assess how different padding sizes influenced the evasion rate (ER), we conducted experiments by appending varying padding sizes ranging from 512 bytes to 4,096 bytes in increments of 512 bytes while maintaining a consistent experimental setup across all samples for reliable comparison. To understand the impact of the adversarial algorithm, we repeated the same experiments using GRAD and IGSM.

2) Slack Attack: A key factor in our analysis is the availability of slack space in the malware files and its impact on detection evasion. To find the slack space available in the malware files, we have used *pefile* [25] python library to identify the sections of the PE file and used the following equation to calculate slack space in each section of the file.

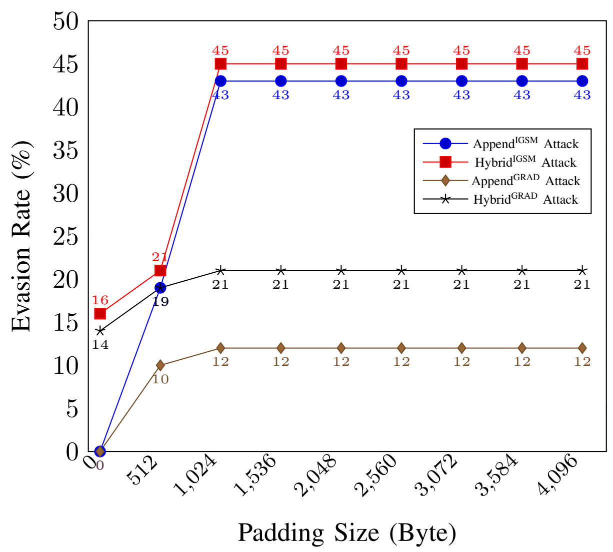

Fig. 5. Evasion rate using different padding lengths for Append^{IGSM}, Hybrid^{IGSM}, Append^{GRAD}, and Hybrid^{GRAD} attacks. Zero padding size represents no perturbation added, hence 0% evasion rate. Note that hybrid attacks utilize the entire slack spaces available within the malware, resulting in evasion success rate even with zero padding.

$$\text{Slackspace} = \text{Size of Raw Data} - \text{Virtual Size}$$

Once the slack spaces were identified, we utilized all the available slack space across all the sections to insert the perturbations. To optimize the perturbations, we have used both IGSM and GRAD adversarial algorithms.

3) Hybrid Attack: As mentioned earlier, the hybrid attack combines the strategies of both append and slack attacks. In this attack, we utilized all the available slackspace and the same padding range, increasing from 512 to 4096 Bytes. Similar to the other two attacks, we carried this attack under IGSM and GRAD.

V. Analysis and Discussion

A. Impact of Padding Length (RQ1)

As shown in Figure 5, when the padding size was increased from 512 to $1,024$ bytes, the ER saw a notable jump from 19% to 43% and 10% to 12% using IGSM and GRAD respectively under the append attack. However, further increases in padding size reveal a critical threshold at 1024 padding bytes, with an evasion rate (ER) of 43% and 12% using IGSM and GRAD, respectively. This contrasts with the earlier work by Suciu et al.[22], who achieved nearly a 70% ER by padding 10,000 bytes. Their higher evasion rate can be attributed to the similarity between the datasets used for training and testing, resulting in stronger gradients to craft adversarial malware samples. Additionally, our analysis considered 8,107 latest malware files, rather than sampling 400 files.

The hybrid attack utilizing both the slack space and appended perturbations started at 16% and 14% using IGSM and GRAD, respectively, with 0 bytes of padding. But with the introduction of 512 bytes padding, the ER increased to 21% using IGSM and 19% using GRAD. A further increase

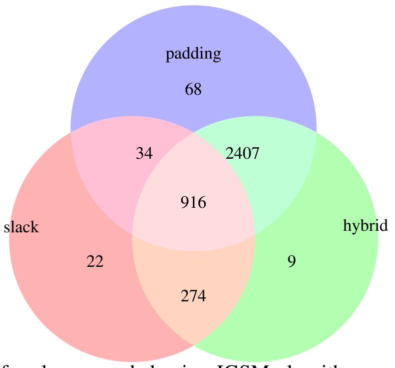

Fig. 6. Count of malware evaded using IGSM algorithm, grouped based on the type of adversarial attack used.

in the padding bytes showed a plateau at max ER of 45% with 1024 Bytes of padding. Similarly, the Gradient decent method achieves an ER of 21%. Hybrid attack has consistently shown a better ER over append attack at all the padding lengths due to the added benefits of slack perturbations. Further, the IGSM approach exhibits a significantly higher evasion success than the Grad across all padding sizes. Notably, both adversarial algorithms (IGSM and GRAD) and both adversarial techniques (Append and Hybrid) reached their respective efficacy thresholds at a padding size of $1,024$ bytes.

Finally, to answer RQ1, the evasion rate initially increases as the padding size is increased. However, it reaches a threshold at a padding length of 1024 bytes. Beyond this point, further increasing the padding size does not enhance the evasion rate. This behavior is consistent across all variants of attack. Thus, the type of adversarial algorithm does not significantly affect the pattern of padding size on the evasion rate. Since we have achieved the maximum evasion rate at 1024 Bytes, we will only reference the padding length of 1024 Bytes in the analysis.

B. Impact of Malware Structure (RQ2)

Out of the $8,107$ samples that were studied, $1,246$ samples using IGSM and 1058 samples using GRAD were successfully evaded under slack attack, giving them an ER of approximately 16% and 14% respectively. The Slack^{IGSM} (16%) has under performed in comparison to Padding$^{IGSM}{}_{1024}$ (43%), but on the contrary Slack^{GRAD} (14%) has a better performance than Padding$^{IGSM}{}_{1024}$ (12%).

On the other hand, the hybrid attack of append and slack yielded ER of 45% and 21% using IGSM and GRAD, respectively. The difference between the evasion rate is only 2% compared to Append$^{IGSM}{}_{1024}$; this points to the very limited advantage of slackspaces when combined with padding bytes. Overall, the hybrid attack with padding of $1,024$ bytes using IGSM has achieved the highest ER in our analysis.

Further, to understand the sole contribution of slack space and padding, we have drawn a Venn diagram that constitutes the count of malware evaded by using IGSM in Fig 6. Overall, from the set of malware evaded, we observed that only 4%(305) samples were evaded specifically due to the contribution of slackspace. On the other hand, 30%(2484) malware samples are evaded due to the contribution of padding. A

Slack Space (Byte)	No. of Malware	No. of Malware Evaded (ER %)			
		Slack^{IGSM}	Slack^{GRAD}	Hybrid^{IGSM}	Hybrid^{GRAD}
0	4,492	0 (0%)	0 (0%)	**1,825 (40%)**	453 (10%)
1-1,000	1138	274 (24%)	265 (23%)	**507 (44%)**	280 (24%)
1,001-2,000	1101	394 (35%)	323 (29%)	**535 (48%)**	380 (34%)
2,001-3,000	359	60 (16%)	52 (14%)	**132 (36%)**	65 (18%)
3,001-4,000	127	56 (44%)	48 (37%)	**81 (63%)**	49 (38%)
4,001-5,000	135	50 (37%)	21 (15%)	**61 (45%)**	23 (17%)
5,001-6,000	112	34 (30%)	29 (25%)	**41 (36%)**	30 (26%)
6,001-7,000	288	87 (30%)	33 (11%)	**117 (40%)**	75 (26%)
7,001-8,000	42	10 (23%)	10 (23%)	**17 (40%)**	11(26%)
8,001-9,000	26	6 (23%)	6 (23%)	**10 (38%)**	7 (26%)
9,001-10,000	12	8 (66%)	8 (66%)	**11 (91%)**	8 (66%)
>10,000	275	267 (97%)	263 (95%)	**269 (97%)**	264 (96%)

TABLE 1
ER USING SLACK^{IGSM}, SLACK^{GRAD}, HYBRID^{IGSM}, AND HYBRID^{GRAD} ATTACKS. MALWARE IS GROUPED BY THE SIZE OF SLACK SPACES AVAILABLE.

Section Name	No. Of malware	No. of Malware Evaded (ER %)			
		Slack^{IGSM}	Slack^{GRAD}	Hybrid^{IGSM}	Hybrid^{GRAD}
.data	1364	622 (46%)	591 (44%)	804 (59%)	660 (49%)
.idata	776	330 (43%)	288 (38%)	458 (60%)	347 (45%)
.rdata	2115	861 (41%)	754 (36%)	1178 (56%)	828 (40%)
.reloc	836	347 (42%)	292 (35%)	576 (69%)	347 (42%)
.rsrc	2477	880 (36%)	725 (30%)	1164 (47%)	793 (33%)
.text	3031	1157 (39%)	960 (32%)	1597 (53%)	1081 (36%)
Total	3615	1246 (35%)	1058 (30%)	1781 (50%)	1192 (33%)

TABLE 2
EVASION RATE USING SLACK^{IGSM}, SLACK^{GRAD}, HYBRID^{IGSM}, AND HYBRID^{GRAD} ATTACKS, MALWARE IS GROUPED BASED ON THE AVAILABILITY OF SLACKSPACE IN THE SECTION.

total of 916 malware samples were evaded by all three types of attacks, indicating that these samples can be successfully evaded regardless of the adversarial attack method employed. Slackspace can be a useful maneuver to evade detection. Still, padding can also evade almost all the malware samples evaded by Slackspace and has a higher success rate in evading other malware samples.

To gain deeper insights into how different sizes of slackspace available impact ER, we grouped the malware samples based on their overall slack space availability in increments of $1,000$ bytes. Table 1 presents the results of this categorization. The ER of Slack^{IGSM} across all the ranges is higher than Slack^{GRAD} attack.

From Table 1, the evasion rate has increased with the increased available slack space until $2,000$ bytes. Thereafter, the evasion rate fluctuates as the slack space increases. The most significant evasion rates are observed in larger slack spaces, peaking at $> 95\%$ for spaces over $10,000$ bytes. This is evident across all four attacks presented in the table, indicating a complex dynamic relation with slack space where the amount of slack space is a crucial determinant of evasion success. Still, the increased available resources don't always guarantee success in evasion.

To understand the granular role of slackspace based on their host PE sections, we have grouped the malware by the Slackspace availability in each PE section. Although there are many sections, we have considered very prominent and well-known ones. From Table 2, the `.text` section showcased the most instances, with slackspaces, followed by `.rsrc` and `.rdata`. Coming to the ER, the `.data` section has showcased the highest ER in three attacks except for the Hybrid^{IGSM} attack, where `.reloc` has the upper hand. Contrary to the high availability of slackspaces in the `.text` section, it has not reciprocated the same upperhand regarding the ER. By this, the section containing the slack space does affect the evasion rate, irrespective of higher availability in a particular section.

In conclusion, to answer RQ2, padding is a significantly more successful strategy in comparison to slack space. The slackspace doesn't constantly enhance the evasion rate. Instead, the evasion rate varies up and down, pointing to a complex relationship between slackspace and the evasion rate. Coming to the impact of the adversarial algorithm in this relationship, as shown in Table 1, both IGSM and GRAD show random fluctuations, pointing that the choice of adversarial algorithm doesn't affect the relationship between slackspace and ER. However, IGSM has a slightly higher ER when compared to the ER of GRAD-based attacks. Finally, the section containing the slackspace also plays a role in affecting the evasion rate. In our analysis, the perturbations in the slack space of `.data` section showcased the highest evasion rate.

C. Impact of Malware Type (RQ3)

To understand the role of malware type, we have grouped the malware by its type/behavior. There are 12 malware families in our dataset (excluding 2 families, which had only 1 sample each). Table 3 has all the ER related to six different approaches (3 attacks $\times$ 2 adversarial algorithms) across all the malware families. To analyze the structure of each malware type, we have added the details of slackspace availability for each section in Table 4. To study each malware type by the amount of slack space, Table 5 presents the details on how each malware type behaves in different slack space ranges. Using these three tables, we discuss the behavior of each type in detail.

Worm: In our dataset, the worm is the most prevalent, represented by $2,753$ samples. The structural characteristics of the Worm, as revealed by Tables 4 and 5A, show that $1,775$ samples are devoid of slack space. Out of the $978(35\%)$ malware samples with slack space, 824 samples showed slack presence in the .text section, and 687 samples showed slack presence in the .rsrc section. A slack attack using IGSM resulted in a 10% ER with 265 samples being evaded, while GRAD achieved 8%. The granular data from Table 5A reveal that ER peaks within the $3,001$ to $4,000$ slack bytes range, highlighting the best spot for evasion attack. Past this point, we observe a decline in evasion success, suggesting a diminishing return on evasion effectiveness as slack space increases. For padding attacks, the worm showed an ER of 37% and 6% for append$_{1024}$ attack using IGSM and gradient techniques, respectively, as presented in Table 3. Although the ER of 37% is significant, it is less than the overall ER of 43% under the append$_{1024}^{IGSM}$. Coming to the hybrid attack, Padding in combination with slack space has no impact on the ER using IGSM and very little impact using GRAD. Although slack spaces are available in 35% of the worm's files, only 10% were successfully evaded under Slack^{IGSM}. Overall, We can observe that an append attack is more effective than a slack attack.

Virus: The structural analysis of the Virus across $2,027$ samples in our dataset, as revealed by Table 4, indicates

Malware Type	No.Of Malware	Slack(%)		Append_{1024}(%)		Hybrid_{1024}(%)	
		IGSM	GRAD	IGSM	GRAD	IGSM	GRAD
Worm	2753	265 (10%)	204 (8%)	995 (37%)	147 (6%)	1004 (37%)	282 (11%)
Virus	2027	458 (23%)	429 (22%)	826 (41%)	237 (12%)	890 (44%)	520 (26%)
Trojan	1002	239 (24%)	143 (15%)	471 (48%)	117 (12%)	505 (51%)	252 (26%)
Dropper	899	58 (7%)	39 (5%)	208 (24%)	25 (3%)	242 (27%)	57 (7%)
Adware	509	38 (8%)	38 (8%)	461 (91%)	195 (39%)	469 (93%)	230 (46%)
Downloader	466	129 (28%)	138 (30%)	254 (55%)	116 (25%)	270 (58%)	169 (37%)
Ransomware	149	22 (15%)	33 (23%)	32 (22%)	0 (0%)	37 (25%)	31 (21%)
Miner	139	5 (4%)	5 (4%)	60 (44%)	3 (3%)	64 (47%)	8 (6%)
Spyware	79	8 (11%)	8 (11%)	71 (90%)	57 (73%)	73 (93%)	65 (83%)
PUA	33	10 (31%)	6 (19%)	20 (61%)	4 (13%)	23 (70%)	10 (31%)
Hacktool	28	5 (18%)	6 (22%)	15 (54%)	7 (25%)	15 (54%)	9 (33%)
Banker	21	9 (43%)	9 (43%)	10 (48%)	4 (20%)	12 (58%)	11 (53%)

TABLE 3
ER(%) BY MALWARE TYPE FOR SLACK, APPEND_{1024}, AND HYBRID_{1024} ATTACKS USING IGSM AND GRADIENT TECHNIQUES.

Malware Type	No.of Malware	Count of Malware with Slack Regions in					
		.data	.idata	.rdata	.reloc	.rsrc	.text
worm	2753	293	228	311	75	687	824
virus	2027	758	367	878	357	612	1042
trojan	1002	137	75	273	99	405	402
dropper	899	57	52	301	148	352	343
adware	509	13	10	68	42	72	73
downloader	466	55	7	168	51	142	180
ransomware	149	17	12	52	32	107	58
miner	139	17	22	15	9	55	55
spyware	79	0	0	7	10	7	10
pua	33	4	2	12	4	12	12
hacktool	28	6	0	14	3	14	15
banker	21	7	1	16	6	12	17

TABLE 4
NO. OF MALWARE WITH SLACK SPACE IN VARIOUS SECTIONS BY TYPE

the presence of slack space in almost all the file sections. The ER for the Virus, as depicted in Table 3, is intriguing to observe that the ER (%) is notably higher, at 23% with IGSM and 22% with GRAD, when compared to others. This anomaly can be attributed to the structural characteristics of the Virus, particularly the high prevalence of substantial slack space within each file section. A more detailed breakdown of these evasion rates based on slack space, as illustrated in Table 5B, reveals that the majority of virus samples exhibited slack space presence in the lower ranges. Notably, the evasion rates are also high within these lower ranges, with 67% of evasion rates under the $\text{Hybrid}_{1024}{}^{\text{IGSM}}$ attack falling within the range of 1000-2000 bytes.

Although the following range of 2000-3000 bytes has around 140 virus samples, only 2 of them are successfully evaded, indicating the short-range threshold at 1000-2000 bytes. In the next ranges, the ER has soared and reached 99% evasion in very high slack space ranges, as shown in 5B. Further analysis revealed that these samples contain slack space ranging from 50,000 to 75,000 bytes The ER for the Virus family under append_{1024} attack is 41% using IGSM and 15% using GRAD, showing their increased susceptibility to these perturbation techniques. This behavior is further emphasized in the hybrid attack scenario, with ER of 44% using IGSM and 26% using GRAD. Overall, the virus family exhibits a high evasion rate against both Append and Slack attacks targeted against MalConv.

Trojan: Despite 55% of the Trojan samples exhibiting slack space, as shown in Table 5C, their ER for slack attacks are moderate - 24% using IGSM and 15% under the Grad attack, as depicted in Table 3. ER of Trojan increases with the amount of available slack space to the range of 5000-6000 Bytes. Remarkably, Trojans with over $10,000$ Bytes of slack space demonstrate a near-perfect ER of 98% in slack attacks, which on further investigation showcased slackspace more than 3 Billion Bytes. On the other hand, the response to padding attacks is more pronounced. With IGSM, the Trojan family shows a significant ER of 48%. This positive response is further amplified in hybrid attacks, where the ER climbs to 51%, as indicated in Table 3. This increase suggests that while slack space provides a foundation for evasion, the addition of padding offers a more substantial boost to the Trojan's ability to evade detection.

Dropper: The structural analysis of the Dropper family, based on Tables 4 and 5D, indicates Dropper malware group is lean in design and conservative in its use of slack space. Table 4 reveals the variation slack space across different sections of the dropper, with the highest counts observed in the .rsrc and .text sections. Table 5D further shows that 507 out of the 899 Dropper samples showcase 0 bytes of slack space, and the droppers with slack space beyond $2,000$ bytes are exceptionally low. Although the smaller size can be an advantage for droppers to evade detection, the limited slack space larger than $2,000$ bytes poses a major challenge from an adversarial perspective as evidenced by the ER from the Table 3. The slack attack yielded a maximum ER of 7% with IGSM on 58 samples, while the GRAD attack yielded ER of 5%. In Table 4, the number of the .data and .idata sections with slack regions are almost equal to the number of files evaded with slack attack. This points us to the possibility of .data and .idata playing a crucial role towards evasion. Further, when droppers are subjected to append_{1024} attack, it yielded an ER of 24% using IGSM and 3% using GRAD. As you can see from Table 3, the dropper family has the second lowest ER; this shows that the Dropper family is more resilient to padding than compared to other. Due to its resiliency to padding and low slack space availability, dropper yielded the second lowest ER under hybrid attack with 27% using IGSM and 7% under gradient attack.

Adware: Padding attacks are highly effective against adware as shown in Table 3, the ER for the append_{1024} attack using IGSM is a remarkable 91%. When examining the Adware family's interaction with slack space (Table 5E), it's evident that only a small fraction, about 70 samples, exhibit slack presence. In line with this observation, the ER for Adware

Slack Space (Byte)	A. Worm					B. Virus					C. Trojan				
	No.Of Malware	Slack(%)		Hybrid$_{1024}$(%)		No.Of Malware	Slack(%)		Hybrid$_{1024}$(%)		No.Of Malware	Slack(%)		Hybrid$_{1024}$(%)	
		IGSM	GRAD	IGSM	GRAD		IGSM	GRAD	IGSM	GRAD		IGSM	GRAD	IGSM	GRAD
0	1775	0 (0%)	0 (0%)	559 (32%)	39 (3%)	870	0 (0%)	0 (0%)	286 (33%)	31 (4%)	451	0 (0%)	0 (0%)	220 (49%)	78 (18%)
1-1,000	249	52 (21%)	59 (24%)	122 (49%)	66 (27%)	284	14 (5%)	17 (6%)	33 (12%)	20 (8%)	162	36 (23%)	29 (18%)	83 (52%)	32 (20%)
1,001-2,000	153	31 (21%)	8 (6%)	40 (27%)	10 (7%)	474	228 (49%)	200 (43%)	316 (67%)	254 (54%)	141	70 (50%)	37 (27%)	76 (54%)	39 (28%)
2,001-3,000	155	26 (17%)	21 (14%)	52 (34%)	31 (20%)	140	2 (2%)	1 (1%)	38 (28%)	4 (3%)	47	25 (54%)	24 (52%)	32 (69%)	24 (52%)
3,001-4,000	81	48 (60%)	41 (51%)	57 (71%)	41 (51%)	13	1 (8%)	1 (8%)	1 (8%)	1 (8%)	7	5 (72%)	4 (58%)	7 (100%)	5 (72%)
4,001-5,000	114	44 (39%)	14 (13%)	53 (47%)	16 (15%)	6	2 (34%)	1 (17%)	3 (50%)	1 (17%)	9	2 (23%)	2 (23%)	2 (23%)	2 (23%)
5,001-6,000	74	28 (38%)	23 (32%)	33 (45%)	23 (32%)	2	0 (0%)	1 (50%)	1 (50%)	1 (50%)	3	2 (67%)	2 (67%)	3 (100%)	2 (67%)
6,001-7,000	120	24 (20%)	25 (21%)	70 (59%)	41 (35%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	136	57 (42%)	3 (3%)	38 (28%)	28 (21%)
7,001-8,000	8	3 (38%)	4 (50%)	5 (63%)	5 (63%)	21	2 (10%)	2 (10%)	2 (10%)	2 (10%)	4	2 (50%)	2 (50%)	3 (75%)	2 (50%)
8,001-9,000	5	0 (0%)	1 (20%)	0 (0%)	1 (20%)	11	4 (37%)	4 (37%)	5 (46%)	4 (37%)	2	1 (50%)	1 (50%)	1 (50%)	1 (50%)
9,001-10,000	4	0 (0%)	0 (0%)	3 (75%)	0 (0%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
> 10,000	15	9 (60%)	8 (54%)	10 (67%)	9 (60%)	206	205 (100%)	202 (99%)	205 (100%)	202 (99%)	40	39 (98%)	39 (98%)	40 (100%)	39 (98%)
	D. Dropper					E. Adware					F. Downloader				
0	507	0 (0%)	0 (0%)	95 (19%)	17 (4%)	423	0 (0%)	0 (0%)	410 (97%)	191 (46%)	258	0 (0%)	0 (0%)	122 (48%)	29 (12%)
1-1,000	218	43 (20%)	23 (11%)	116 (54%)	24 (12%)	28	15 (54%)	15 (54%)	20 (72%)	15 (54%)	134	88 (66%)	95 (71%)	95 (71%)	96 (72%)
1,001-2,000	148	12 (9%)	14 (10%)	26 (18%)	14 (10%)	41	12 (30%)	13 (32%)	27 (66%)	13 (32%)	43	26 (61%)	28 (66%)	30 (70%)	28 (66%)
2,001-3,000	1	1 (100%)	0 (0%)	1 (100%)	0 (0%)	4	1 (25%)	1 (25%)	2 (50%)	1 (25%)	8	3 (38%)	3 (38%)	4 (50%)	3 (38%)
3,001-4,000	1	1 (100%)	1 (100%)	1 (100%)	1 (100%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	2	0 (0%)	0 (0%)	0 (0%)	0 (0%)
4,001-5,000	2	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	4	2 (50%)	4 (100%)	3 (75%)	4 (100%)
5,001-6,000	19	0 (0%)	0 (0%)	0 (0%)	0 (0%)	2	0 (0%)	0 (0%)	0 (0%)	0 (0%)	4	4 (100%)	3 (75%)	4 (100%)	4 (100%)
6,001-7,000	3	1 (34%)	1 (34%)	3 (100%)	1 (34%)	1	1 (100%)	0 (0%)	1 (100%)	1 (100%)	1	0 (0%)	0 (0%)	0 (0%)	0 (0%)
7,001-8,000	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1	0 (0%)	0 (0%)	0 (0%)	0 (0%)	5	1 (20%)	0 (0%)	5 (100%)	0 (0%)
8,001-9,000	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	2	0 (0%)	0 (0%)	2 (100%)	0 (0%)
9,001-10,000	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	8	8 (100%)	8 (100%)	8 (100%)	8 (100%)	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)
> 10,000	0	0 (0%)	0 (0%)	0 (0%)	0 (0%)	1	1 (100%)	1 (100%)	1 (100%)	1 (100%)	5	5 (100%)	5 (100%)	5 (100%)	5 (100%)

TABLE 5
MALWARE FAMILY EVASION RATE BASED ON SLACK SPACE AVAILABLE

under slack attack is 8% for both IGSM and GRAD attacks. Since the majority of Adware samples lack significant slack space, AE attack relies on padding for evasion. This is reinforced in the hybrid attack scenario, which mirrors the results seen in padding-only attacks.

Downloader: Table 3 reveals that the Downloader family responds differently to IGSM and GRAD attacks in a slack attack scenario. With an ER of 28% for IGSM and a slightly higher 30% for GRAD, it suggests that Downloader malware is more sensitive to the Gradient Descent method's one-to-one perturbation optimization approach. The append attack results show the converse result between the two methods, as IGSM outperforms with 55% of ER where GRAD manages only 25%. In the hybrid attack scenario, combining slack with padding, IGSM achieved an ER of 58%, marginally higher than its performance in the append-only attack. However, for GRAD, the ER jumps to 37%, up from 25% in the append-only scenario. This notable increase for GRAD is inline with the higher evasion for slack attack under GRAD than compared to IGSM.

Rest of the Malware types: Ransomware, with 149 samples, shows an ER of 15% using IGSM and 23% with GRAD in slack attacks. Although more than 125 of 149 samples exhibit slackspace, the ER of slack attack is not high. Similarly, the effectiveness of padding is also very low as evidenced by the ER of 22% using IGSM and 0% using GRAD techniques under append attack. Interestingly, ransomware with the GRAD technique yielded a higher ER under slack attack while yielding 0% ER under append attack. This gets more interesting when we observe the hybrid ER reduce to 21% as slack attack alone evaded 23% of the malware. This shows the performance of slack attack is better on using GRAD while padding attack is better for IGSM in ransomware.

Miner malware shows contrasting results as its ER is only 4% in slack attacks, it escalates to 44% in append attacks using IGSM. This low ER under slack attack can be attributed to the lack of slack space in most of the samples as seen in Table 4. Spyware, consisting of 79 samples, demonstrates a notably high ER in append attacks, reaching 90% with IGSM and 73% using GRAD. Despite this, its ER in slack attacks is only 11% (as seen in Table 3), which could be attributed to its limited slack space as depicted in Table 4.

For PUA, Hacktool, and Banker, the ER in slack attacks vary, with PUA at 31% (IGSM) and 19% (GRAD), Hacktool at 18% and 22%, and Banker at 43% for both techniques. While coming to append attack all the three malware types has ER around 50%. This has been further amplified in the hybrid attack with PUA achieving a maximum ER of 70%.

Finally, to answer RQ3, Table 3 clearly shows the varying evasion rates across malware types. By this, we can say that the malware type affects the ER. As discussed, each malware type has a different response to different adversarial techniques. For instance, Adware has only 9% evasion rate under Slack$^{\text{IGSM}}$, whereas it has a 91% and 93% evasion rate under Append$^{\text{IGSM}}$ and Hybrid$^{\text{IGSM}}$ attacks, respectively showing an upper hand for the Append attack. On the contrary, Ransomware has a higher ER under Slack$^{\text{GRAD}}$ compared to Append$^{\text{IGSM}}$.

Each malware type has a different structure to fulfill its functionalities. For instance, the Virus has shown a lot of slack space availability in various sections, thereby increasing its evasion rate (ER) under slack attacks. In contrast, worms, being very concise, have shown a very low evasion rate. The range of slack space in which each family achieves maximum evasion has not been consistent, further indicating the influence of structural characteristics. Thus, the structural characteristics and available slack space within each malware type significantly influence their evasion rates.

VI. CONCLUSION AND FUTURE WORK

This study explores the domain of adversarial malware, using raw executable files as input and employing existing benchmark adversarial sample generation techniques to evade a CNN-based malware classifier, MalConv. We focus our adversarial attack analysis on different malware types and their structure. Our analysis started by analysing the impact of padding on the evasion rate. Initially, the evasion rate increased, but soon, the Malconv showed resistance to padding beyond 1024 bytes. This behaviour has been consistent across both the padding and hybrid attacks. The peak evasion rate achieved by the padding attack alone is 43%. Among two traditional methods, the padding and the slack attack, the padding attack demonstrated a higher evasion success under IGSM. However, the Hybrid attack, which combines padding and slack attacks by complementing each other's drawbacks, consistently outperformed the individual methods. The analysis also noted that IGSM achieved better performance with minimal alterations, as low as three modifications.

Coming to the factors in the scope of the analysis, the relationship between slackspace and the ER is complex. Initially, the ER increased but eventually fluctuated, reaching a peak of 97% at very high slack spaces. Apart from the quantity of slack space, the section containing the slack space is also an important factor. In our analysis, we identified that the `.data` section played an important role towards the ER. Regarding the type of malware, each malware type has showcased various ERs, highlighting the impact of malware type on the ER. Each malware type has a different structure in relation to slackspace, and this structure also plays a key role in evasion in Slackspace-based attacks. One important observation highlighting the impact of malware type is that considering each type of malware with a similar range of slack space, each malware type has shown a different ER.

To mitigate evasion attacks, adversarial training can be employed, which strengthens malware detection models by training them with adversarial examples[26]. Additionally, randomized smoothing, particularly through byte ablation techniques, can enhance robustness by blurring decision boundaries and reducing the model's sensitivity to adversarial perturbations [27]. Finally, input preprocessing, such as marking all slack bytes to 0, can limit the attack surface available for adversarial manipulations, further reducing the effectiveness of such attacks. These combined approaches offer a more resilient defense against malware adversarial attacks.

However, it is important to note that the distribution of malware across types in the dataset was not uniform, and the study was limited to only 10-12 types. Future work could extend this research to explore the behavioral patterns of a broader range of malware types. Additionally, while these results provide valuable insights into the susceptibility of MalConv to adversarial attacks, they are specific to this particular model. Extending this research to a black-box environment would be crucial in real-world scenarios, where the target models are often unknown. Furthermore, the current research focused on adding perturbations only in available slack spaces. Future efforts could explore the potential of exploiting other regions, such as within the sections of PE files, to further enhance the sophistication and effectiveness of adversarial attacks in malware detection.

VII. Acknowledgment

This work is partially supported by NSF grants 2416992, 2230610 at North Carolina A&T State University and 2416990, 2230609 at Tennessee Tech University.

References

[1] H. Sarker *et al.*, "Ai-driven cybersecurity: an overview, security intelligence modeling and research directions," *SN Computer Science*, vol. 2, no. 3, p. 173, 2021.

[2] S. Samtani *et al.*, "Trailblazing the artificial intelligence for cybersecurity discipline: A multi-disciplinary research roadmap," pp. 1–19, 2020.

[3] E. Raff *et al.*, "Malware detection by eating a whole exe," in *Workshops at the thirty-second AAAI conference on artificial intelligence*, 2018.

[4] I. J. Goodfellow *et al.*, "Explaining and harnessing adversarial examples," *arXiv preprint arXiv:1412.6572*, 2014.

[5] A. Madry, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[6] K. Aryal *et al.*, "A survey on adversarial attacks for malware analysis. arxiv 2021," *arXiv preprint arXiv:2111.08223*.

[7] F. Kreuk *et al.*, "Deceiving end-to-end deep learning malware detectors using adversarial examples," *arXiv preprint arXiv:1802.04528*, 2018.

[8] "New malware variants every second," 2023. [Online]. Available: https://www.av-test.org/fileadmin/pdf/reports/AV-TEST_HYAS_Protect_Evaluation_February_2023.pdf

[9] D. Kim *et al.*, "Static detection of malware and benign executable using machine learning algorithm," in *INTERNET 2016: The Eighth International Conference on Evolving Internet*, 2016, pp. 14–19.

[10] Y. Nagano and R. Uda, "Static analysis with paragraph vector for malware detection," in *Proceedings of the 11th International Conference on Ubiquitous Information Management and Communication*, 2017, pp. 1–7.

[11] R. Searles *et al.*, "Parallelization of machine learning applied to call graphs of binaries for malware detection," in *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*. IEEE, 2017, pp. 69–77.

[12] A. N. Jahromi *et al.*, "An improved two-hidden-layer extreme learning machine for malware hunting," *Computers & Security*, vol. 89, p. 101655, 2020.

[13] Ç. Yücel and A. Koltuksuz, "Imaging and evaluating the memory access for malware," *Forensic Science International: Digital Investigation*, vol. 32, p. 200903, 2020.

[14] Q. K. A. Mirza *et al.*, "Cloudintell: An intelligent malware detection system," *Future Generation Computer Systems*, vol. 86, pp. 1042–1053, 2018.

[15] R. Vinayakumar *et al.*, "Robust intelligent malware detection using deep learning," *IEEE access*, vol. 7, pp. 46 717–46 738, 2019.

[16] A. McDole *et al.*, "Deep learning techniques for behavioral malware analysis in cloud iaas," *Malware analysis using artificial intelligence and deep learning*, pp. 269–285, 2021.

[17] McDole *et al.*, "Analyzing cnn based behavioural malware detection techniques on cloud iaas," in *Cloud Computing–CLOUD 2020: 13th International Conference, Held as Part of the Services Conference Federation, SCF 2020, Honolulu, HI, USA, September 18-20, 2020, Proceedings 13*. Springer, 2020, pp. 64–79.

[18] J. C. Kimmel *et al.*, "Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure," *IEEE Access*, vol. 9, pp. 68 066–68 080, 2021.

[19] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[20] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*. Ieee, 2009, pp. 248–255.

[21] B. Kolosnjaji *et al.*, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *2018 26th European signal processing conference (EUSIPCO)*. IEEE, 2018, pp. 533–537.

[22] O. Suciu *et al.*, "Exploring adversarial examples in malware detection," in *2019 IEEE Security and Privacy Workshops (SPW)*. IEEE, 2019, pp. 8–14.

[23] K. Li *et al.*, "Fgam: Fast adversarial malware generation method based on gradient sign," *arXiv preprint arXiv:2305.12770*, 2023.

[24] A. Kurakin *et al.*, "Adversarial examples in the physical world," in *Artificial intelligence safety and security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[25] E. Carrera, "pefile: Python module to read and work with pe (portable executable) files," https://github.com/erocarrera/pefile, 2023.

[26] K. Lucas *et al.*, "Adversarial training for {Raw-Binary} malware classifiers," in *32nd USENIX Security Symposium (USENIX Security 23)*, 2023, pp. 1163–1180.

[27] D. Gibert *et al.*, "A robust defense against adversarial attacks on deep learning-based malware detectors via (de) randomized smoothing," *CoRR*, 2024.