IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 November 2024, accepted 1 December 2024, date of publication 18 December 2024, date of current version 2 January 2025.

Digital Object Identifier 10.1109/ACCESS.2024.3519524

SURVEY

A Survey on Adversarial Attacks for Malware Analysis

KSHITIZ ARYAL[1], (Graduate Student Member, IEEE), MAANAK GUPTA[1], (Senior Member, IEEE), MAHMOUD ABDELSALAM[2], (Member, IEEE), PRADIP KUNWAR[1], (Graduate Student Member, IEEE), AND BHAVANI THURAISINGHAM[3], (Fellow, IEEE)

[1]Department of Computer Science, Tennessee Technological University, Cookeville, TN 38505, USA
[2]Department of Computer Science, North Carolina A&T State University, Greensboro, NC 27411, USA
[3]Department of Computer Science, The University of Texas at Dallas, Dallas, TX 75080, USA

Corresponding author: Kshitiz Aryal (karyal42@tntech.edu)

This work was supported in part by NSF Grant through Tennessee Tech University under Grant 2416990 and Grant 2230609 and in part by North Carolina A&T State University under Grant 2416992 and Grant 2230610.

ABSTRACT Machine learning-based malware analysis approaches are widely researched and deployed in critical infrastructures for detecting and classifying evasive and growing malware threats. However, minor perturbations or ineffectual byte insertions can easily 'fool' these trained ML classifiers, making them ineffective against these crafted and smart malicious software. This survey aims to provide an encyclopedic overview of adversarial evasion attacks specifically targeting malware detection and classification systems, standing apart from previous surveys by focusing exclusively and comprehensively on this unique application domain. While significant strides have been made in adversarial research in other fields, the specific challenges of adversarial malware remain under-explored due to the intricate nature and constraints of the malware domain. Our survey addresses this gap by analyzing literature on adversarial evasion attacks published between 2013 and 2024, making it one of the first to systematically focus on malware-specific adversarial attacks in a detailed, self-contained manner. The paper will begin by introducing various machine-learning techniques used to generate adversarial malware samples, including the structural nuances of target files, which influence adversarial vulnerabilities. The work presents an in-depth threat model specific to adversarial malware evasion attacks, describing the unique attack surfaces of malware detectors and outlining adversarial goals tailored to the malware domain. We systematically analyze adversarial generation algorithms from broader domains adapted to malware evasion attacks, proposing a taxonomy of adversarial evasion attacks within malware detection based on target domains(Windows, Android and PDF). The survey highlights real-world adversarial evasion attacks on machine learning-based anti-malware engines under each taxonomical heading, demonstrating the evolution and refinement of these attack strategies over time. Our survey outlines current limitations and practical challenges in executing adversarial attacks against malware detectors in real-world environments. We identify open problems and propose future research directions for developing more practical, robust, efficient, and generalized adversarial attacks on ML-based malware classifiers.

INDEX TERMS Adversarial evasion attack, adversary modeling, security for AI, windows PE malware, Android malware, PDF malware.

I. INTRODUCTION

Machine Learning (ML) has revolutionized the modern world due to its ubiquity and generalization power over the humongous volume of data. The transformation of ML approaches from classical algorithms to modern deep learning technologies is providing breakthroughs in state-of-the-art research problems. Further, deep learning (DL) has excelled in areas where traditional ML approaches were infeasible (or unsuccessful) to apply. Needless to

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

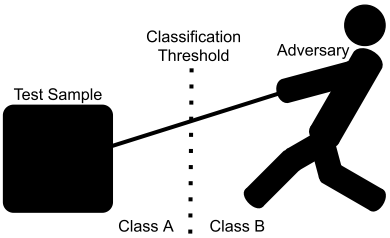

FIGURE 1. Example of adversarial evasion attack that drags test sample from class A to B.

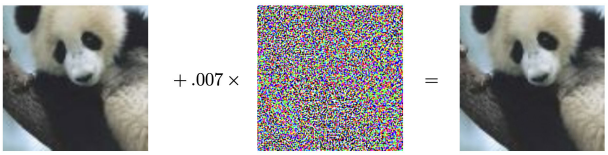

FIGURE 2. An adversarial example against GoogLeNet [20] on ImageNet [21], demonstrated by Goodfellow et al. [22].

say, machine learning is increasingly embedded in our daily life habits; connecting to people on social media, ordering food and groceries from online stores, and listening to music on Spotify are all examples of systems built around recommendation engines powered by deep learning-based models. Machine learning-based solutions not only control our lifestyle, but it has also revolutionized cyber security-critical operations in different domains, including malware analysis [1], [2], [3], [4], [5], spam filtering [6], fraud detection [7], medical analysis [8], access control [9], [10], among others. These solutions exemplify ML's versatility in both enhancing user experience and strengthening cybersecurity across diverse fields.

Malware analysis is one of the most critical fields in which ML is employed significantly and increasingly relied upon. Traditional malware detection methods focus on signature-based approaches, maintaining a database of unique malware identifiers to compare against signatures extracted from suspicious files. However, with security researchers looking for advanced detection techniques addressing sophisticated zero-day and evasive malware, ML-based approaches have become essential in advancing detection capabilities [11]. Most of the modern anti-malware engines, such as Windows Defender, Avast, Deep Instinct D-Client and Cylance Smart Antivirus, are powered by machine learning [12], making them robust against emerging variants and polymorphic malware. Despite the existence of numerous malware detection approaches, including ones that leverage ML, recent ransomware attacks, like the Colonial Pipeline attack where operators had to pay around $5 million for recovering a 5,500-mile long pipeline [13] and the MediaMarkt attack worth an around $50M bitcoin payment [14], highlight the vulnerabilities and limitations of current security approaches and necessitates of more robust, real-time, adaptable and autonomous AI-driven defense mechanisms.

The assumption of similar training and testing settings in machine learning is overly simplified and often does not hold true for real-world use cases where adversaries deceive the ML models into performing wrong predictions (i.e. adversarial attacks). These attacks manifest either as data poisoning—altering the training data—or as test data manipulation through evasion attacks [15]. Data poisoning attacks [16], [17], [18] have been prevalent for some time but are less scrutinized as access to training data by the attackers is considered unlikely. In contrast, Adversarial Evasion(AE) attacks, first introduced by Szegedy et al. [19] against deep learning architectures, are carried out by carefully crafting imperceptible perturbation in test samples, forcing models to misclassify as illustrated in Figure 1. Here, the attacker's effort is to drag a test sample across the ML's decision boundary by adding minimal perturbation to that sample. Given the growing body of research and the increasing security risks, this survey will focus exclusively on adversarial evasion attacks targeting malware detection systems.

Adversarial Evasion (AE) attacks were originally developed for image data, where the primary constraint is to make perturbations imperceptible to the human eye [23]. A widespread example of AE attack in images, shown in Figure 2, is performed by Goodfellow et al. [22] where GoogLeNet [20] trained on ImageNet [21] classifies panda as a gibbon with the addition of minimal perturbations. This threat has moved beyond research settings and has been proven effective in real-world environments. For example, Eykholt et al. successfully performed "sticker attacks" on road signs, causing an image recognition system to misidentify 'STOP' signs as speed limits. Researchers from the Chinese technology company Tencent[1] tricked Tesla's Autopilot in Model S and forced it to switch lanes by adding a few stickers on the road [24]. Such adversarial attacks on real-world applications force us to rethink the increasing reliability of machine learning-based smart technologies.

Generating adversarial examples in the malware domain presents unique challenges compared to tasks in computer vision due to additional constraints. Perturbations in malware files must be crafted to preserve their functionality and executability, which adds complexity to adversarial evasion. Constraints may vary depending on the structure of the malware files. Each file structure contains divisions, bringing their individual significance and resilience to modification. Adversarial evasion attacks on malware are carried out by manipulating or inserting a few ineffectual bytes in the malware executable in a way that does not tamper with its original state but changes the classification decision by the ML model.

For instance, one early demonstrated attack against an anti-malware engine was carried out by Anderson et al. [25] using reinforcement learning. This black-box attack bypassed

[1] https://www.tencent.com/

Random Forest and Gradient-Boosted Decision Trees (GBDT) detectors by modifying a few bytes of Windows PE malware files. Kolosnjaji et al. [26] later employed an evasion attack using a gradient-based approach against a Convolutional Neural Network (CNN) based malware detector. Since then, numerous works have tried to optimize the attacks, discovering better approaches to attack wide domains of malware detectors. Demetrio et al.'s [27] success in crafting adversarial from few header byte modifications and Suciu et al.'s [28] experiment on inserting perturbations in different file locations, further magnified the interest towards improving the standard of attacks. This growing sophistication in adversarial attacks has intensified concerns within the cybersecurity research community, fueling an ongoing battle between adversarial attackers and defenders.

This survey aims to offer researchers a detailed overview of adversarial evasion attacks on malware detectors across platforms such as Windows, Android, PDF, Linux, and hardware-based systems to elucidate existing vulnerabilities and guide future offensive/defensive strategies.

A. MOTIVATION AND CONTRIBUTION

1) PRIOR SURVEYS AND LIMITATIONS

The surveys on adversarial attacks crafted in different domains have been summarized in Table 1. Most surveys on adversarial attacks are focused on computer vision for image mis-classification [32], [33], [34], [50]. Fewer studies have specifically addressed security domains such as malware, intrusion, and network security, which are particularly relevant due to the critical risks they pose. For instance, Barreno et al. [29] worked on one of the first surveys on the security side of machine learning, where different categories of attacks and defences against ML systems are discussed. Gardiner et al. [30] focused on reviewing call and control detection techniques. They identified vulnerabilities and also pointed out limitations of malware detection systems. Duddu et al. [35] highlighted privacy issues within ML systems and introduced a cyber-warfare testbed to evaluate attack and defense strategies. In addition, Martins et al. [45] performed a generalized survey on attacks focusing on cloud security, malware detection and intrusion detection, while Ibitoye et al. [47] surveyed adversarial attacks in the network domain using a risk grid map.

Several recent surveys have attempted to bridge this gap by focusing more closely on adversarial attacks within malware detection, especially on Windows platforms. For instance, Li et al. [48] systematized adversarial malware detection (AMD), addressing both attack and defense perspectives. Ling et al. [49] focused specifically on adversarial attacks against Windows PE malware detectors, discussing the intricacies of PE file structure, specific challenges, and potential attack vectors. From studying all existing surveys, we noticed none of the surveys entirely covered all adversarial evasion attacks carried out against the malware binaries and talked about the practicality of different approaches.

From these surveys, we can draw several conclusions regarding adversarial research trends, particularly in cybersecurity. The surge in interest within adversarial research over the past few years underscores an urgent need to understand and address adversarial threats as our reliance on automated systems grows. However, while adversarial attacks on general domains have been well-surveyed, only a few studies have focused solely on adversarial malware attacks, and even fewer comprehensively cover adversarial evasion attacks within malware detection systems. This leaves a critical gap, as current malware-focused surveys often span multiple domains without a detailed examination of adversarial evasion attacks on malware.

2) OUR CONTRIBUTIONS

This work will contribute to understanding the arms race between attacker and defender by discussing adversarial evasion attacks in different folds of the malware domain. We aim to provide a self-contained survey on adversarial attacks against malware detection techniques. Based on our knowledge, this work is one of the first to focus solely on adversarial attacks on malware detection systems. In this work, our contributions cover the following dimensions:

- Survey provides the threat modelling for adversarial evasion attacks in the malware domain. The threat is modelled in terms of the attack surface of the malware detector, the attacker's knowledge about the malware detector, the attacker's capabilities on malware, and the adversarial goals that are to be achieved through the malware files.
- We systematically analyze different adversarial generation algorithms proposed in different domains, which have been attempted to be used in the malware domain. We then taxonomize adversarial evasion attacks in the malware with respect to various attack domains. As Windows malware is the most abundant and also the most exploited area, we further taxonomize attacks on Windows malware based on the optimization algorithms used. We also discuss attacks in less frequent file structures like Android and PDF.
- We discuss real evasion attacks carried out against anti-malware engines by the researchers under each taxonomical heading. We also cover the strategies researchers used to generate adversarial attacks, showing how the attacks evolved with time.
- We discuss the challenges and limitations of existing adversarial evasion attacks while carrying them out in a real-world environment. We also highlight future research directions for carrying out more practical, robust, efficient, and generalized adversarial attacks on malware classifiers.

B. SURVEY ORGANIZATION

We begin our survey, as discussed in Section I, by introducing the field of adversarial machine learning and the motivation

TABLE 1. Surveys focusing on security of machine learning.

Paper	Year	Application Domain	Taxonomy	Threat Model	Adversarial Example
Barreno et al. [29]	2008	Security	Attack Nature		
Gardiner et al. [30]	2016	Security	Attack Type/Algorithm	√	√
Kumar et al. [31]	2017	General	Attack Type		
Yuan et al. [32]	2017	Image	Algorithm	√	√
Chakraborty et al. [33]	2018	Image/Intrusion	Attack Phase	√	√
Akhtar et al. [34]	2018	Image	Image domains		√
Duddu et al. [35]	2018	Security	Attack Type	√	
Li et al. [36]	2018	General	Algorithm		
Liu et al. [37]	2018	General	Target Phase	√	
Biggio et al. [38]	2018	Image	Attack Type	√	√
Sun et al. [39]	2018	Image	Image Type	√	√
Pitropakis et al. [40]	2019	Image/Intrusion/Spam	Algorithm	√	
Wang et al. [41]	2019	Image	Algorithm	√	√
Qiu et al. [42]	2019	Image	Knowledge	√	√
Xu et al. [43]	2019	Image/Graph/Text	Attack Type	√	
Zhang et al. [44]	2019	Natural Language Processing	Knowledge/Algorithm	√	√
Martins et al. [45]	2019	Intrusion/Malware	Approach		√
Moisejevs [46]	2019	Malware Classification	Attack Phase		√
Ibitoye et al. [47]	2020	Network Security	Approach/Algorithm	√	√
Li et al. [48]	2021	Malware Detection	Attacks/Defenses	√	
Ling et al. [49]	2023	Windows malware	Algorithm		√
Our Work	2024	Malware Analysis	Domain/Algorithm	√	√

***Year**: Published Year, **Application Domain**: Dataset domain for adversarial, **Taxonomy**: Basis of taxonomy, **Threat Model**: Presence of threat modeling, **Adversarial Example**: Discuss actual adversarial attacks crafted in literature*

to study adversarial evasion attacks in the malware analysis domain. Section II models the adversarial threat from different dimensions. Section III discusses various algorithms considered standard techniques for adversarial perturbation generation. Section V taxonomizes existing real adversarial attacks based on the execution domains (Windows, PDF, Android, Hardware, Linux) and algorithms maneuvered to carry out the attack. This section discusses real attacks carried out against malware detection approaches in detail and compares related works. Section VI highlights the challenges of current adversarial generation approaches and sheds light on open research areas and future directions for adversarial generation in malware analysis. Finally, Section VII concludes our survey.

C. LITERATURE SEARCH RESOURCES

We used different digital libraries for computer science scholarly articles to discover the relevant state-of-the-art works and publications in adversarial attacks on malware analysis. Our major sources are IEEE Xplore, ACM Digital Library, DBLP, Google Scholar, Semantic Scholar and arXiv. Among numerous keywords used to fetch the papers from public libraries, ''Adversarial Malware'', ''Adversarial Evasion Attacks'', ''Adversarial Malware Analysis'', and ''Adversarial attacks in malware'' gave us the most relevant papers. After listing all the published works in the adversarial generation between the years 2013 and 2024, we filtered out papers with good impact and relevance and prepared the final list to conduct our detailed survey.

II. ADVERSARIAL THREAT MODEL

Security threats are defined in terms of their goals and capabilities. In this section, we divided the adversarial threat model, tailored to adversarial attacks in malware, into four parts: adversarial knowledge, attack surface, adversarial capabilities, and adversarial goals. This section aims to explain the major components of adversarial attacks to readers.

A. ADVERSARIAL KNOWLEDGE

The adversary's knowledge is the amount of information about a model under attack that the attacker has, or is assumed to have, to carry out adversarial attacks against the model. An adversarial attack can be classified into two groups based on the attacker's knowledge:

- **White box attack:** In a white box approach, an attacker has full knowledge about the underlying target model. Such knowledge might include but is not limited to, the name of the algorithm, training data, tuned hyper-parameter, and gradient information, among others. It is relatively easy to carry out attacks in the white box model due to the large amount of available knowledge. Current state-of-the-art works on white box environments have achieved near-perfect adversarial attacks [28].
- **Black box attack:** In a black box approach, an attacker only has access to the inputs and outputs of the model. No information about the internal structure of the model has been provided. Generally, in a black box attack, a surrogate model is created by guessing the internal

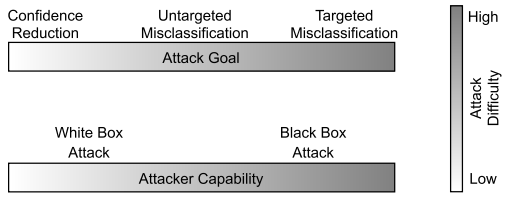

FIGURE 3. An attack difficulty with adversarial knowledge and adversarial goals.

structure of the target model using input and output [32], [51]. In addition, in a gray box attack [52], a type of black box attack, the attacker knows the output performance of the model in the form of accuracy, confusion matrix or some other performance metrics.

In general, it is assumed that black-box adversarial attacks are difficult and inefficient to orchestrate compared to white-box attacks, primarily due to the information available regarding the underlying target model. However, black box attacks reflect more real-world use cases where, in a practical sense, an attacker will not likely have any knowledge of underlying models.

B. ATTACK SURFACE

The attack surface includes different vulnerable points by which an attacker attacks the target model. The flow of data through this machine learning pipeline introduces vulnerabilities in each stage [33]. Attack surfaces comprise all those points in machine learning models (malware detector models in our case) where adversaries can carry out their attacks. Based on different approaches to carrying out attacks, the attack surface is classified into the following broad categories [53]:

- **Poisoning Attack:** This attack is carried out by contaminating training data during the training process of models [54], [55]. Training data is poisoned with faulty data, making machine learning models learn on the wrong dataset.
- **Evasion Attack:** This attack is performed by trying to evade a trained system by adjusting malicious input samples at test time [25], [28]. Evasion attacks do not require any access to training data but require some level of access to the target model.
- **Exploratory Attack:** This attack is carried out against a model with blackbox access [51]. Attackers try to maximize their knowledge without direct access to the underlying algorithm and attempt to reflect similar input data patterns.

C. ADVERSARIAL CAPABILITIES

Adversarial capabilities denote adversaries' abilities and largely depend on their knowledge of the target model. The most straightforward attack approach is the attacker having access to full or partial training data. For adversarial attacks carried out on malware files, adversarial capabilities can be classified into the following categories:

- **Data Injection:** It is the ability of attackers to inject new data. One type of injection can be done on training data before the training process. A different kind of data injection is carried out by inserting a perturbation, which forms a new section or replaces the original section within an existing file. Injected data can corrupt the model or cause the data-injected file to evade detection.
- **Data Modification:** Data modification can also be performed both for training data and testing data. If an attacker has access to training data, data can be modified to cause the model to learn on modified data. An attacker can also modify input data to cause perturbation and lead to evasion.
- **Logic Corruption:** Logic corruption is the most dangerous ability to be possessed by an attacker and also the most improbable. Whenever an attacker has complete access to a model, they can modify the learning parameters and other hyper-parameters related to the model. Logic corruption can go undetected, which makes it hard to design remedies.

D. ADVERSARIAL GOALS

An attacker tries to fool the target model, causing it to produce misclassifications. Details of algorithms used to attack and achieve the adversary's goals successfully are discussed in section III. Typically, the adversarial goals of attackers are categorized as follows:

- **Untargeted Misclassification:** An attacker tries to change the output of the model to a value different from the original prediction. For malware classification, if an ML model predicts a malware file as a family A, the goal is to force the model to misclassify it as a family other than A.
- **Targeted Misclassification:** An attacker tries to change the output of the model to a target value. For example, if an ML model is predicting a malware file as family A, the goal is to force the model to misclassify it as family B.
- **Confidence Reduction:** An attacker's goal is to reduce the confidence of an ML model. Changing the prediction is unnecessary, but a reduction of confidence is enough to meet the goal.

Figure 3 gives an overview of the adversarial attack difficulty with respect to the attacker's knowledge, capabilities and goals. While moving in the direction of increasing attack complexity from confidence reduction to targeted misclassification, attack difficulty also increases for the attacker. However, white-box attacks with higher attacker capability have the least attack difficulty.

III. ADVERSARIAL ALGORITHMS

In this section, we will explore the most distinguished adversarial attack algorithms discovered in different domains and applied to generate adversarial malware samples. Different algorithms are developed in numerous time frames, battling

the trade-off in terms of application domain, performance, computational efficiency and complexity [45]. We will discuss the architecture, implementation and challenges of each algorithm. Most of the attack algorithms are gradient-based approaches where perturbations are obtained by optimizing some distance metrics between original and perturbed samples.

A. LIMITED-MEMORY BROYDEN - FLETCHER - GOLDFARB - SHANNO (L-BFGS)

The Limited-Memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) algorithm is an optimization technique often used in machine learning to minimize functions with many variables, especially when resources for storing large datasets are limited. Szegedy et al. [19] proposed one of the first gradient-based approaches for adversarial example generation in the imaging domain using the box-constrained Limited-Memory Broyden-Fletcher-Goldfarb-Shanno optimization technique. The authors studied counter-intuitive properties of deep neural networks, which allow small perturbations in the images to fool deep learning models for misclassification. Adversarial examples trained for a particular neural network are also able to evade other neural networks trained on completely different hyperparameters. These results are attributed to non-intuitive characteristics and intrinsic unseen spots of deep learning models learned by backpropagation, with structure connected to data distribution in a non-obvious way. Traditionally, for small enough radius $\epsilon>0$ around the given training sample x, $x + r$ satisfying $||r|| < \epsilon$ will be classified correctly by a model with very high probability. However, many underlying kernels are found not holding to this kind of smoothness. The simple optimization procedure is able to find adversarial samples using imperceptibly small perturbations, leading to incorrect classifications by the classifier. While adding noise to an original image is to minimize perturbation r added to the original image under L_2 distance. This groundbreaking use of L-BFGS for adversarial example generation set the stage for an extensive research area focused on probing and mitigating vulnerabilities in deep learning models.

B. FAST GRADIENT SIGN METHOD (FGSM)

The Fast Gradient Sign Method (FGSM) is a foundational gradient-based technique for generating adversarial perturbations, efficiently devised to exploit vulnerabilities in neural networks. Considering the gradient-based optimization technique as a workhorse of modern AI, Goodfellow et al. [22] proposed an efficient approach for generating adversarial perturbation in the image domain. In contrast to earlier works that explained adversarial phenomena as non-linearity and overfitting, the authors argued that the linear nature of neural networks leads to their vulnerability. Linear behaviour in high dimensional space is found sufficient to cause adversarial samples. To define the approach formally, let's consider θ as a parameter of the model, x as input to the model, y as target associated with x and $J(\theta, x, y)$ be the cost function for training neural network. On linearizing the cost function around the current parameter values θ, perturbation can be obtained by

$$\eta = \epsilon sign(\nabla_x J(\theta, x, y)) \tag{1}$$

where the required gradient can be computed using backpropagation. Conversion of features from problem to feature space affects the precision. Commonly, images are represented by 8 bits per pixel and other information below 1/255 of the continuous range is discarded. With limited precision, the classifier may not be able to respond to all perturbations whose size is smaller than the precision of the feature. Classifiers having well-separated decision boundaries are expected to assign the same class for original sample x and perturbed sample x' until $||\eta||_\infty < \epsilon$ where ϵ is small enough to be discarded. FGSM's efficiency and generalizability sparked further research into robust adversarial attack strategies and defense mechanisms for deep learning.

C. ITERATIVE GRADIENT SIGN METHOD (IGSM)

Different from the one-step perturbation approach, where a single large step is in the direction of increasing the loss of the classifier, the Iterative Gradient Sign Method takes iterative small steps while adjusting the direction after each step [23]. The basic iterative method extends the FGSM approach by applying it multiple times with a small step size and clipping the pixel values after each iteration to ensure the perturbation within ϵ neighbourhood of the original image. Formally, each step in the iterative process is represented by:

$$X_{N+1}^{adv} = Clip_{X,\epsilon} X_N^{adv} + \alpha sign(\nabla_X J(X_N^{adv}, y_{true})) \tag{2}$$

where X_{N+1}^{adv} is the perturbed image at N^{th} iteration and $Clip_{X,\epsilon}\{X'\}$ function performs pixel-wise clipping on image X' in order to keep perturbation inside $L_\infty\epsilon$-neighbourhood of source image X. Kurakin et al. [23] extended basic iterative method to iteratively least likely class method to produce adversarial for targeted misclassification. By using this iterative method, finer and controlled perturbations can be added, allowing for successful adversarial examples even with higher values of ϵ while minimizing distortion to the original image.

D. JACOBIAN SALIENCY MAP ATTACK (JSMA)

Most of the adversarial generation techniques are based on observing output variations to generate input perturbations, while Papernot et al. [56] crafted adversarial samples by constructing a mapping of input perturbations with output variations. The approach is based on limiting the l_0-norm of the perturbation, which deals with a minimal number of pixel modifications. The proposed adversarial generation algorithm against feed-forward DNN modifies a small portion of input features by applying heuristic search approaches. Adversarial sample X^* is constructed by adding perturbation δ_X to benign sample X through following

optimization problem:

$$argmin_{\delta_X} ||\delta_X|| \; s.t. \; F(X + \delta_X) = Y^* \tag{3}$$

where $X^* = X + \delta_X$ is the adversarial sample and Y^* is the desired adversarial output. Forward derivative is used to evaluate the changes in output due to corresponding modifications in input, and these changes are presented in matrix form called as Jacobian of the function. This approach, leveraging a forward derivative matrix, provides a precise mechanism for selecting minimal, impactful modifications, allowing adversarial examples to evade detection by conventional defenses.

E. CARLINI & WAGNER ATTACK (C&W)

Carlini & Wagner [57] introduced an advanced adversarial generation technique aimed at defeating defensive distillation, a defense mechanism designed to harden neural networks against adversarial attacks through a single retraining phase [58]. Defensive distillation leverages a softened output layer during training to reduce model sensitivity to perturbations; however, Carlini & Wagner's approach demonstrated the ability to evade such defenses effectively. The proposed approach is able to perform three types of attacks: L_0 attack, L_2 attack and L_∞ attack to evade defensively distilled and undistilled networks. These attacks are based on different distance metrics, which are:

- L_0 distance, measuring the number of pixels modified in an image
- L_2, measuring the standard Euclidean distance between the original sample and the perturbed sample
- L_∞, measuring the highest change among any of the perturbed coordinates

The optimization problem for adversarial generation of input image x is given as:

$$minD(x, x + \delta) \; such \; that \; C(x + \delta) = tx + \delta \in [0, 1]^n \tag{4}$$

where input x is fixed and goal is to reach δ that minimizes $D(x, x + \delta)$. D could be any of distance metric among L_0, L_2 *or* L_∞. This method not only bypasses defensive distillation but also demonstrates effectiveness across both distilled and undistilled networks.

F. DeepFool

Dezfooli et al. [59] introduced DeepFool, an untargeted white-box adversarial generation method that targets misclassification by minimizing the Euclidean distance between the perturbed and original samples. The attack begins by generating a linear decision boundary to separate the given classes and is accompanied by the addition of perturbation perpendicular to the decision boundary that separates classes. The attacker projects the perturbation into a separating line called hyper-plane and tries to push it beyond for misclassification. Decision boundaries are usually non-linear in high dimensional space, so the perturbation is added iteratively by performing multiple attacks till evasion. An attack for such a multiclass finds the closest hyperplane and projects input towards that hyperplane, then proceeds to the other.

G. ZEROTH ORDER OPTIMIZATION (ZOO)

All of the previously discussed adversarial generation algorithms depend on the detector model's gradient, which limits the adversarial attack space within the white-box attack. Chen et al. [60] proposed a black-box adversarial generation approach by estimating the gradients of targeted DNN with only access to the input and output of a target. Zeroth order methods are gradient-free optimization approaches requiring only the Zeroth order oracle for the optimization process. The objective function is analyzed at every two close points $f(x + hv)$ and $f(x - hv)$ with a very small h to estimate a gradient along the direction of vector v. An optimization algorithm like gradient descent follows gradient estimation. While attacking black-box DNN with a large input size, the use of a single minute step of gradient descent can be very inefficient as a large number of gradients needs to be estimated. To address this, ZOO applies a coordinate descent method, optimizing each input coordinate iteratively, thus improving the computational efficiency of the attack without requiring exact gradient calculations.

H. ONE PIXEL ATTACK (OPA)

Another gradient-free adversarial generation approach is proposed by Su et al. [61] by generating one-pixel perturbations based on Differential Evolution (DE). Differential evolution is a population-based optimization algorithm that can find higher quality solutions than gradient-based approaches [62]. Since gradient information is not required for DE, the need for differentiable objective functions is also omitted. One pixel attack perturbs a single pixel using only probability labels. The single-pixel modification allows attackers to hide the adversarial modifications, making them imperceptible. Each image is represented as a vector to carry out the attack, where each scalar element represents one pixel. With f as the target function, $x = (x_1, \ldots, x_n)$ representing n-dimensional inputs, t being the original class, $e(x) = (e_1, \ldots, e_n)$ denoting the perturbation to be added to the input with maximum modification limited to L, the optimized solution is given by Equation 5.

$$\max_{e(x)^*} f_{adv}(x + e(x)) \;\; subject \; to \; ||e(x)||_0 \leq d, \tag{5}$$

where d is a small number. This approach considers determining two values: the dimension to be perturbed and the corresponding magnitude of modification for each dimension. Unlike multi-pixel approaches, OPA uniquely focuses on modifying only one pixel without constraining the intensity of the perturbation, making it a minimalistic yet effective adversarial strategy.

I. UNIVERSAL ADVERSARIAL PERTURBATION (UAP)

The Universal Adversarial Perturbation (UAP) technique, introduced by Moosavi-Dezfooli et al. [63], is a powerful

approach for generating adversarial examples that can mislead a target classifier across a wide range of inputs. Unlike traditional adversarial attacks that are tailored to specific inputs, UAP generates a single perturbation that can be applied universally to different images, effectively compromising the model's robustness. This technique begins by formulating an optimization problem that seeks to find a perturbation δthat minimizes the classification accuracy across a diverse set of images while ensuring that the perturbed images remain visually similar to the original ones. The objective function is expressed as:

$$\min_{\delta} \sum_{x \in \mathcal{S}} \max_{y \in \mathcal{Y}} \mathcal{L}(f(x+\delta), y), \tag{6}$$

where S is a set of input images, Y represents the set of possible labels and L denotes the loss function. The UAP technique effectively captures the inherent vulnerabilities of deep learning models, revealing that even small, universal perturbations can lead to significant misclassification. By utilizing a heuristic approach to iteratively refine the perturbation, the UAP can efficiently generate adversarial samples that maintain their effectiveness across various input instances, making it a significant concern for the security of machine learning systems.

J. AUTOATTACK

AutoAttack, proposed by Croce et al. [64] is a robust framework for evaluating the adversarial robustness of machine learning models through an ensemble of diverse, parameter-free attacks. It aims to provide a reliable assessment of model performance against adversarial examples by leveraging a set of complementary attack methods that can adaptively target vulnerabilities in the model. AutoAttack consists of four different attacks, which are categorized into two groups: a robust white-box attack and two parameter-free attacks. The first one is based on APGD (Adaptive Projected Gradient Descent) method, which effectively explores the decision boundaries of the model by using iterative gradient-based perturbations. The other attacks include FAB (Fast Adaptive Boundary) and Square Attack, which leverage different optimization strategies to generate adversarial examples without requiring precise parameters, ensuring versatility in attacking various models. One of the key features of AutoAttack is its parameter-free nature, which eliminates the need for tuning hyperparameters that often complicate the evaluation process. By incorporating a diverse set of parameter-free attacks, AutoAttack enhances the reliability of robustness assessments.

K. BOUNDARY ATTACK

Boundary Attack, proposed by Brendel et al. [65] is a decision-based adversarial attack method designed to effectively target black-box machine learning models. It operates under the principle of manipulating the input data while staying within the vicinity of the original sample, aiming to achieve a misclassification without the need for gradient information. This technique is particularly notable for its ability to generate adversarial examples in scenarios where the model's internal workings are not accessible. Unlike gradient-based attacks, Boundary Attack leverages the model's output decisions to iteratively refine perturbations. The attack begins with an initial adversarial example, which is chosen based on a strategy that ensures it is misclassified. It then seeks to adjust this example by moving along the decision boundary of the model, gradually refining the perturbation until a suitable adversarial example is found that successfully misclassifies the input. Focusing on decision boundaries and employing a minimal perturbation strategy enhances the feasibility of adversarial attacks in scenarios where traditional gradient-based methods are not applicable.

IV. FILE STRUCTURE

Executable files are structured differently based on the target/host OS. Although detailed discussions on file structure are out-of-scope for a survey, a good understanding of file structure is critical both for shaping the adversarial generation strategy and for successfully generating adversarial examples. Different file sections are classified into two groups: mutable and immutable. Mutable sections can be modified for adversarial generation without altering the functionality of the file, whereas immutable sections either break the file or alter the functionality on modification. This section will provide a brief overview of three kinds of file structures that will be discussed later in the survey.

A. WINDOWS PE FILE STRUCTURE

Windows PE file format is an executable file format based on the Common Object File Format (COFF) specification. The PE file is composed of linear streams of data. The structure of Windows PE file as shown in Figure 4 is derived and confirmed from [66], [67], [68]. The header section consists of the MS-DOS MZ header, the MS-DOS stub program, the PE file signature, the COFF file header and an optional header. File headers are followed by body sections with debug information before closing the file. The MS-DOS header occupies the first 64 bytes of the PE file. This header is required to maintain compatibility with files created on Windows version 3.1 or earlier. The Magic number used in the header determines if the file is of compatible type. MS-DOS runs stub-program after loading the executable and is responsible for giving output messages with errors and warnings.

PE file header is searched by indexing the `e_lfanew` field to get the offset of the file, which is the actual memory-mapped address. This section of the PE file is one of the target areas for modification, and these locations are used as macros to create adversarial examples [27]. The macro returns the offset of the file signature location without any dependency on the executable file type. This injection of adversarial perturbation in this location, being in the header, was found to be highly efficient while increasing the risk

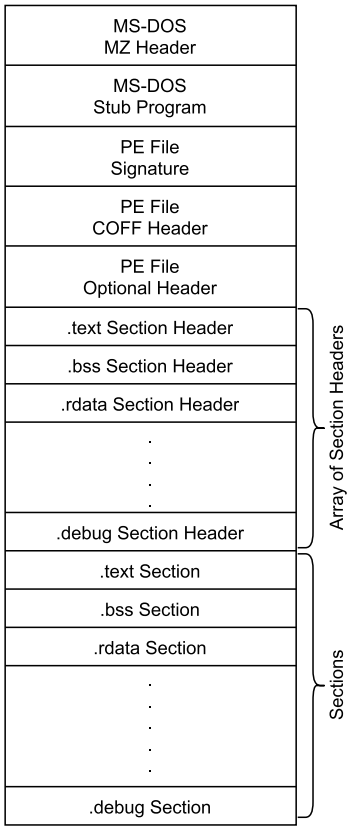

FIGURE 4. A standard structure of windows PE file.

of compromising the file integrity. At offset $0 \times 3c$, a 4-byte signature is placed to identify the file as a PE image. The optional header takes the next 224 bytes. Although it may be absent in a few types of files, it is not an optional segment for PE files. It contains information like initial stack size, program entry point location, preferred base address, operating system version, and section alignment information, among other [67].

Information from both the section header as well as optional header is required to retrieve data directories. The `.text` section contains all the executable code sections along with the entry point. Uninitialized data for the applications is stored in the `.bss` section, which includes all declared static variables, and the `.rdata` section represents all the read-only data like constants, strings, and debug directory information. The `.rsrc` section contains resource information for the module, and export data for an application are present in the `.edata` section. Section data is the major area where perturbation takes place to make a file adversarial. Debug information is in the `.debug` section, but the debug directories reside in the `.rdata` section.

In terms of perturbation injection strategy based on the characteristics of the target location inside the PE file, we can categorize the existing adversarial attack approaches into the following kinds:

- **Append Attack:** This approach is the most common and widely adopted approach for adversarial perturbation injection inside a PE file as it appends the noise at the end of the file without interfering with the execution of a malware file [26], [69].
- **Slack Attack:** This approach looks for existing empty spaces (code caves) inside a PE file and uses those spaces as targets for injecting adversarial perturbations [28]. These approaches are found to be more efficient in creating adversarial samples while being limited in their availability across all malware samples.
- **Header Attack:** This approach uses the unused spaces in the header like the MS-DOS header used for backward compatibility, that does not have relevance to the execution of PE file as a target for adversarial injection [27]. These perturbations are found to be highly effective against end-to-end malware detectors.
- **Code Cave Injection Attack:** This approach is the latest advancement in the field of adversarial malware generation in problem space as it introduces the unused spaces inside the Windows PE file, bringing flexibility to the injection of adversarial perturbation. Yuste et al. [70] injected the perturbation on those locations of raw space that never get mapped to memory, while Aryal et al. [71] used a code loader to restore the malware's original form dynamically.

B. ANDROID FILE STRUCTURE

Android APK file has been recently victimized as a tool for adversarial attacks [72], [73], [74], [75]. APK file is a ZIP file containing different entries. Different sections of APK files are described below:

- **Androidmanifest.xml:** `AndroidManifest.xml` contains information to describe the application. It contains information like the application's package name, components of the application, permissions required and compatibility features [76]. Due to the presence of a large amount of information, `AndroidManifest.xml` is one of the majorly exploited sections in APK files for adversarial attacks.
- **classes.dex:** As Android applications are written in Java, the source code will be with the extension `.java`. These source codes are optimized and packed into this `classes.dex` file.
- **resources.arsc:** This file is an archive of compiled resources. Resources include the design part of apps like layout, strings and images. This file form the optimized package of these resources.
- **res:** Resources of app which is not compiled to store in resources.arsc stays in `res` folder. The XML files present inside this folder are compiled to binary XML to boost performance [77]. Each sub-folder inside `res` stores different types of resources.
- **Meta-INF:** This section is in signed APKs and has all the files in APK with their signatures. Signature verification is done by comparing the signature with the uncompressed file in archive [78].

C. PDF FILE STRUCTURE

Here, we will look into the internal structure of PDF file format. PDF is a portable document with a range of features that are capable of representing documents, including text, images, multimedia, and many others. The basic structure of a PDF file is discussed below:

- **PDF header:** PDF header is the first line of PDF which specifies the version of a PDF file format.
- **PDF Body:** The body of a PDF file consists of objects present in the document. The objects include images, data, fonts, annotations, text streams, etc. Interactive features like animation and graphics can also be embedded in the document. This section provides the possibility of injecting contents and files within it, which makes it the most favourable avenue for adversarial attackers.
- **Cross-reference table:** The cross-reference table stores the links of all the objects or elements in a file. The table helps navigate to other pages and document contents. The cross-reference table automatically gets updated when the PDF file is updated.
- **The Trailer:** The trailer denoted the end of the PDF file and contains a link to the cross-reference table. The last line of the trailer contains the end-of-file marker, %%EOF.

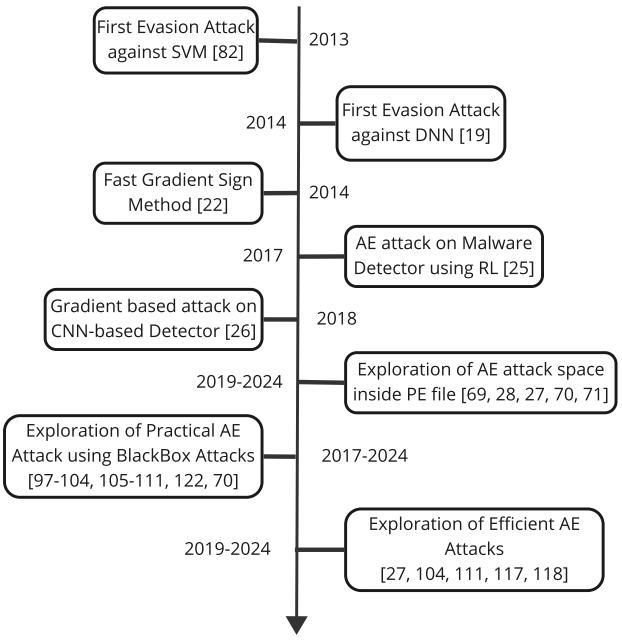

FIGURE 5. Timeline showing research trend for the development of adversarial Evasion(AE) attack.

V. ADVERSARIAL MALWARE EVASION ATTACKS

Adversarial generation methods originating in the image domain did not take long to migrate into the malware field. Among different adversarial threats, evasion attacks have been the most worrisome approach and have already been exploited in various ways. the research trend for the adversarial malware generation specific to evasion attack is shown in a timeline in Figure 5. Initially started by Biggio et al. [79] against SVM and Szegedy et al. [19] against Deep Nets went through different phases to advance in the malware domain. Adversarial malware started with PDF and Windows files due to their abundance and then proliferated into other file formats. Significant work has been done on adversarial generation for Android, PDF, Windows and Linux files. This section deals with adversarial examples generated to evade malware detection systems by making minor perturbations on input malware files.

Though initial research was not concerned about the problem space of adversarial malware, A large volume of research soon started to work on finding suitable locations for adversarial perturbations within a malware file. These subtle modifications to malware files during test time can sneak through the unseen spots of machine learning models without breaking the malware's functionality. As the research progressed, more recent works have focused on creating more stealth and practical attacks while also trying to create more efficient adversarial attacks. The following sections will briefly explain different adversarial generation works by researchers in the malware domain. Adversarial work has been divided based on the attack file format, which includes Windows, Android, PDF, Hardware, and Linux malware files. The following subsection discusses adversarial attacks in Windows files.

A. WINDOWS MALWARE ADVERSARIAL

Microsoft Windows is a dominant PC operating system with more than 70% market share and 1.5 billion users worldwide [80]. Gartner research [81] predicts that 30% of cyberattacks by 2024 will be carried out in the form of adversarial attacks. Abundant availability has placed Windows malware at the core of adversarial threats. Machine learning-based models are data-hungry, so feature engineering is a critical task to feed important features as input. However, the advent of deep neural networks has allowed models to learn features from complex raw data. Deep neural networks have shown impressive performance in malware detection by providing whole binary files as input without any hand-crafted feature engineering effort. We want to mention Raff et al.'s work [82] in this section (referred to as MalConv), which has been an academic standard in the field by making detection considering whole executable. Its architecture combines convolutional activation with global max-pooling before going to fully connected layers, allowing the model to produce its activation regardless of the locations of the detected features. MalConv, as one of the prominent end-to-end static malware detectors, has been considered a baseline for most static adversarial evasion attacks. The subsequent part discusses different adversarial evasion attacks on malware detectors classified based on the optimization algorithm used.

TABLE 2. A gradient based approaches.

Paper/Year	Key Motivation	Approach	Modification	Preserving Functionality	SR
Kolosnjaji et al. 2018 [26]	Adversarial attack on malware detection using raw bytes	• Optimizing byte at a time using gradient descent • Embedded layer to tackle non-differentiable MalConv architecture • Gradient calculation of objective function with respect to embedded representation	Bytes are padded only at the end of the file	• Byte padding only at the end • The Padding byte closest to the embedded byte is chosen	60%
Kreuk et al. 2018 [84]	Gradient based attack with better reconstruction	• Perturbation generation in embedded space • Calculation of weighted distance between generated adversarial embedding from actual embedding • Weighted gradient similar to iterative FGSM	Padding bytes at the end of the file	• Loss function imposing perturbation to embedding matrix row • Payloads inserted into the flagged region	99%
Demetrio et al. 2019 [27]	Explainable technique for efficient adversarial generation	• Feature attribution to determine the most influential feature • Perturbation generation using a gradient of classification function with respect to embedding layer	Changing bytes of file header	• MZ magic number and offset at `0x3C` are not modified	87%
Suciu et al. 2018 [28]	Test existing methods on production-scale datasets and compare different strategies	• Random, gradient-based and fast gradient perturbation • End of file append and slack region insertion • Transferability test across full, EMBER [85] and mini dataset	Padding bytes at the end and in the slack regions	• Slackindexes calculated before adversarial payload insertion • Updates only at the end or in slack regions	27%
Chen et al. 2019 [86]	Enhancing effectiveness of gradient-based approach through benign perturbation initialization	• Saliency vector generation using Grad-CAM • Random benign block initialization and enhanced benign file append, followed by FGSM • Experience-based attack by summarizing the successful trajectories of random benign attacks for black box attack	Bytes are appended only at the end of the file	• No alteration of existing section • Appending only at the end	99%
Jakhotiya et al. 2022 [87]	Attacking state-of-art transformer-based malware detectors	• Train the malware detection system comprised of assembly, static features and neural network modules • Use FGSM to attack the detector	Feature vector	• Modifications in feature space • No considerations for functionality	24%
Aryal et al. 2024 [71]	More flexibility and stealth to adversarial perturbation	• Inject intra-section code cave to make space for adversarial perturbation • Inject code-loader to restore the integrity of malware during execution • Use gradient descent and FGSM to generate the perturbations	Inside the injected code cave	• Any modifications inside code cave don't alter the malware's integrity • Code cave is removed by the code loader during execution	97%
Zhan et al. 2024 [88]	Generate sample agnostic perturbation	• Inject adversarial patch into arbitrary malware samples • Two adversarial patch generation strategies, binary patch and img patch • Use of an evolutionary algorithm to search for universal and robust adversarial perturbations	Add adversarial patches to non-executable regions of the file	• Modification inside non-executable regions does not break the execution	50%

***Key Motivation**: The major motive behind the published work, **Approach**: Key procedures, **Modification**: Changes on file during attack, **Preserving Functionality**: Works towards safeguarding the functionality of a malware, **SR**: Evasion Success Rate*

1) GRADIENT BASED ATTACK

Table 2 presents a comparative study of adversarial attacks using a gradient of cost function against input Windows PE malware. Since Anderson et al. [25] proposed the possibility of manipulating sections of Windows PE malware to form an adversarial sample, various types of research have been conducted to bypass malware detectors. Authors [25] used random actions from action space to modify PE files. To reduce the randomness of payloads, Kolosnjaji et al. [26] proposed appending optimized padding bytes using gradient descent, originally proposed by Biggio et al. [79]. Gradient-based approaches are carried out using either the append or insertion method for perturbation generated using the gradient of cost function as shown in Figure 6. One-hot represented malware vector is combined with gradient-generated perturbation to bypass the malware detector. Representation of the malware can vary based on the approach. Kolosnjaji et al. [26] choose to append bytes only at the end of the file, not to risk altering the functionality of a file. Here, the attacker's goal is to minimize the confidence of the malicious class, limiting the maximum perturbation. Authors achieved an evasion rate up to 60% by only modifying 1% of bytes in the PE file.

Kreuk et al. [69], [83] proposed the enhanced attack method against MalConv [82] using iterative FGSM [22]. The authors focused this approach on enhancing reconstruction by introducing a new surrogate loss function. The representation of binary files as a sequence of bytes is arbitrary, and neural networks are unable to work in this space. Generating adversarial examples deals with adding perturbations to the original sample by increasing or decreasing the gradient. However,

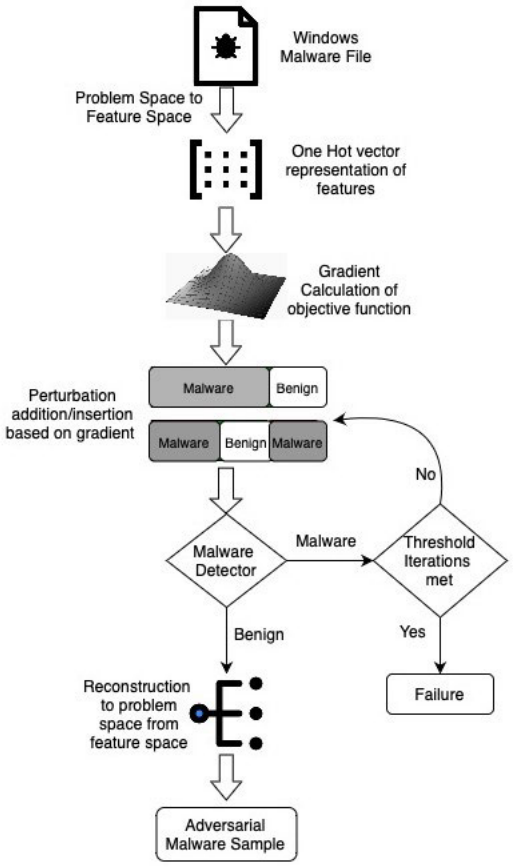

FIGURE 6. Flow diagram for a gradient-based adversarial attack on windows malware.

this process is not that simple, as perturbation in a one-hot vector results in a new vector no longer in one-hot vector space. This approach proceeds by generating perturbations in embedded space. In many cases, the perturbed embedding loses its resemblance to embedding in the lookup table, which contains a mapping between bytes and embeddings. In the absence of resemblance, reconstruction is not possible. Kreuk et al. introduced a new term to the loss function, which causes perturbations to be close to the embedding matrix. To minimise the distance, the introduced term is the weighted distance of generated adversarial embeddings from actual embeddings. The new loss function is:

$$\bar{l}^*(z, y; \theta) = \alpha.\bar{l}(z, y; \theta) + (1 - \alpha)\left[\sum_{i=1}^{L}\sum_{j=1}^{N} d(z_i, M_j)\right] \quad (7)$$

where the first part is the categorical loss called the negative log-likelihood loss, and the second term gives the distance of generated adversarial embedding with the actual embedding in M. The second term is responsible for steering the direction towards reconstructible adversarial embeddings. This approach yielded an evasion rate as high as 99%.

To interpret the blackbox decisions of the malware detection model, Demetrio et al. [27] proposed a technique called integrated gradients initially proposed by Sundararajan et al. [88]. With input model f, a point x and baseline x', the attribution of i_{th} feature is computed as:

$$IG_i(x) = (x_i - x_i') \int_0^1 \frac{\partial f(x' + \alpha(x - x'))}{\partial x_i} d\alpha \quad (8)$$

Equation (8) is the integral of the gradient computed on all points on a line passing through x and x'. Feature attribution determines the most influential feature, leading to meaningful explanations behind classifications of malware binaries. Referencing the research findings, authors can also generate adversarial malware samples by efficiently modifying a few bytes in the file header. This approach is more efficient as it requires a few manipulations to bypass the detector. Authors could evade almost all malware by generating small perturbations on file header sections other than MZ magic number and value at offset $0\times3C$. Perturbation generation using a gradient of classification function with respect to the embedding layer is the same as implemented by Kolosnjaji et al. [26]. Along with success in efficient adversarial attacks from perturbations in file headers, research also introduces new challenges of perturbation being easily detected and patched. This study has directed further research towards hiding modifications from detection. The work claims a high evasion rate of 87%. However, their sample size of just 60 may not demonstrate the true performance on the general dataset.

Suciu et al. [28] trained existing models to study their behaviour on a production-scale dataset to further explore other regions for injecting adversarial perturbations. The author evaluated the effectiveness of adversarial generation strategies at different scales and observed their transferability. Existing adversarial attacks are constrained on appending adversarial noise at the end of a binary file. However, appended bytes are found to be less influential and offset by bytes in the original malware. Inability of byte appending strategies while using size constrained detector like MalConv(Only first 2MB are considered for detection), led authors to use slack attacks. Slack attacks are performed by discovering the region in executable files that are not mapped to memory and will not affect the functionality on modification. Attacking the most influential feature will amplify attack effectiveness, and sufficiently appended bytes can replace legitimate features. Slack attacks yielded an evasion rate as high as 27% in this work. The approach is, however, limited in terms of available slack space as there is no guarantee that the binary files always have enough slack spaces.

All previous works relied on random initial perturbations and were then iteratively updated using a gradient of the model. The role of initializing perturbation in the success rate of adversarial generation can not be disregarded, and Chen et al. [85] proposed the use of saliency vector to select initializing perturbations from benign files. Researchers consider the issue of accuracy and inefficiency in the work of Kreuk et al. [83] and Suciu et al. [28] as a result of random initialization before gradient-driven modification. The benign feature append method was carried out by

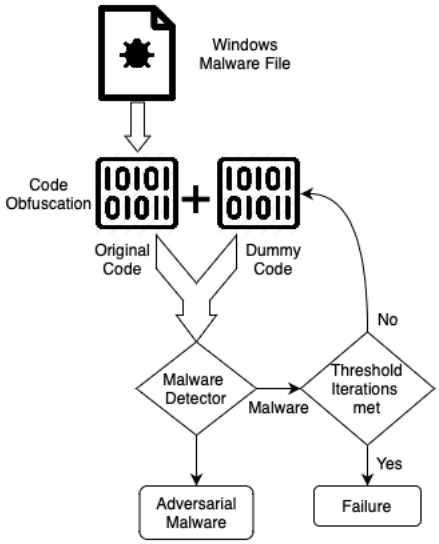

FIGURE 7. An adversarial generation workflow using code obfuscation.

debugging the victim model once to generate saliency vectors. In contrast, continuous debugging of the model is required while incorporating the FGSM algorithm. Avoiding random initialization helps the model obtain backpropagation gradients, and gradient-based algorithms can be implemented more effectively. Benign bytes form saliency vectors and help map between adversarial from continuous space to discrete space, avoiding random perturbations that can not be accurately mapped back to corresponding raw-byte perturbations. This work successfully increased the accuracy of gradient-based adversarial generation techniques up to 99% by replacing random initialization. Jakhotiya et al. [86] tested the gradient-based adversarial attack on a state-of-the-art transformer-based malware detector using FGSM. However, their attack being in feature space is far from addressing the problem space challenge.

Many recent works also use a gradient to optimize the perturbation but without significant changes to the field. Kozak et al. [89] combined multiple adversarial generation approaches, both gradient and non-gradient, to create adversarial malware samples. On the other hand, Yang et al. [90] used multiple strategies for PE malware file modification to preserve its functionality. Li et al. [91] use reverse gradient sign to optimize the perturbations and use the least square method to detect the oscillation in the perturbation optimization while also limiting the injection ratio. To enhance the flexibility and stealth of adversarial perturbation inside Windows PE malware, Aryal et al. [71] proposed an intra-section code cave injection strategy where first the code cave makes spaces within the sections of PE structure. In the next phase, these empty spaces are utilized to inject the adversarial perturbation without breaking the file, as the code loader completely removes the code cave dynamically. Now, the adversarial perturbations are optimized using gradient descent and FGSM. Even though all the discussed approaches used gradient-based optimization, they were different in terms of perturbation location or the motivation behind their research. Their work produced variable success in attacking different regions of the file, taking the evasion rate to as high as 97%.

All existing works using a gradient of the target model produce model-specific perturbations. To make the adversarial generation model agnostic, Zhan et al. [87] proposed a patch injection operation to arbitrary malware samples within non-executable regions of a PE file. The approach initializes the patch and trains it in each malware sample, resulting in an optimized adversarial patch that can be directly applied to any malware binary. Their work produces two different patches: *binary* patch for raw byte sequences and *img* patch for image-based detection. The approach achieved an evasion rate of up to 50%, which, although lower than some targeted methods, is notably high for a general-purpose adversarial patch.

2) CODE OBFUSCATION BASED ATTACK

Code obfuscation changes the pattern of a program without any damage to program logic. Adversarial attacks using obfuscation deal with modifying the code sections without changing the functionality and flow of the program, as shown in Figure 7. Table 3 discusses the code obfuscation attacks done against Windows malware detectors. Park et al. [92] proposed a generative model for generating adversarial through obfuscation in raw binaries. The proposed approach minimally modifies malicious raw binaries using a dynamic programming-based insertion algorithm, obfuscating the `.text` section of a binary in an executable byte sequence. Windows malware binaries are initially converted into grayscale images from byte code for obfuscation. An obfuscation technique called dummy code insertion inserts semantic *nops* (no operation) into a program. At each iteration, the algorithm chooses between inserting a semantic *nop* or not inserting anything based on the distance metric between binary strings. Adding semantic *nops* is easier if the source code is given, but without it, patching techniques are required [95], [96]. The algorithm outputs adversarial malware with the original program's logic after operating in a closed-loop model until the classifier gets fooled. The proposed algorithm is found to be effective against classifiers employing both static and dynamic analysis with features such as API, system calls and n-grams.

Most of the attacks in the adversarial domain are carried out in feature space, and mapping features back to problem space is not always feasible. Song et al. [94] proposed an open-source systematic framework for adversarial malware attacks using code randomization and binary manipulation to evaluate against real-world antivirus systems. The authors collected large categories of features from open-source malware detectors: hash-based signatures, rule-based signatures and data distribution-based features. A generic action set is prepared as micro and macro actions to modify these features. Micro actions are a relative concept, which only changes

TABLE 3. A code obfuscation based approaches.

Paper/Year	Key Motivation	Target Model	Approach	Modification	Preserving Functionality
Park et al. 2019 [93]	Generative model by obfuscation in raw binaries	Inception V3 [94], MalConv [82]	• Dummy code insertion using Adversarial Malware Alignment Obfuscation • Semantic nops insertion to match original malware to standard adversarial • Optimization in closed loop till evasion	Semantic nops insertion	• Modification with executable adversarial • Dummy code insertion in form of semantic nops
Song et al. 2020 [95]	Practical adversarial generation and evaluation against real world anti-virus system	Signature based and machine learning based detectors	• Selection and application of macro actions from action space • Action sequence minimization, traversing through actions and removing unnecessary actions • Entangling macro actions to micro actions to evaluate feature essence	Through sequence of macro and micro action	• Functionality preserving actions • Cuckoo sandbox verification

***Key Motivation**: The major motive behind the published work, **Target Model**: Target defense for adversarial attack, **Approach**: Key procedures to carry out adversarial attack, **Modification**: Changes on file to craft the adversarial perturbation, **Preserving Functionality**: Works towards safeguarding the functionality of a malware*

a subset of actions inside macro-actions. The proposed workflow begins by selecting and applying macro-actions to original samples till the original sample crosses the decision boundary. Those macro-actions with no roles are removed from the action sequence to reach the most efficient evasive form. And finally, to get detailed knowledge about the reason behind evasion, macro actions are broken into micro-action. To provide reasoning for evasion, every actions are entangled into several micro-actions and each macro-actions is replaced with one micro-actions. This process helps in the evaluation of essential feature changes responsible for classification decisions. This research directs future exploration towards the generation of adversarial, which can evade both static as well as dynamic detectors, and also recommends antivirus systems to provide offline dynamic detection.

3) REINFORCEMENT LEARNING BASED ATTACK

To counter the need for a differentiable model for gradient-based approaches, a reinforcement learning agent has been proposed to generate an adversarial sample against static malware detection. Reinforcement learning enables complete blackbox attacks on detector, creating real-world attack scenarios where an attacker is completely unknown about the detector. Table 4 compares all RL approaches on adversarial evasion attacks for Windows malware. Anderson et al. [25] proposed a whitepaper on evading malware detection by modifying Windows PE bytes for the first time. Anderson et al. [97] extended results of work done in [25] to perform generic black box attacks on static PE malware detection without assuming any knowledge of the detector model's structure and features, retrieving only malicious/benign label. Actor-Critic Model with Experience Replay (ACER) is used to learn both policy model π and a Q-function to estimate the state-action value. Countless, infinite features are collapsed into a fixed-size vector using a hashing trick. The obtained feature vector provides a complete view of malware files. Functionality-preserving actions like adding functions to unused import address tables, creating/renaming sections, appending bytes, and manipulating debug info and header checksums are present in action space.

To reduce the instability and increase the convergence speed of Gym-Malware [97], Fang et al. [98] proposed a Deep Q-network to Evade Antimalware engines (DQEAF) framework to evade anti-malware engines. DQEAF can reduce instability caused by higher dimensions, taking binary stream features of only 513 dimensions. It takes only four functionality-preserving actions in its action space to increase convergence and reports a higher evasion rate. Actions proposed for deep Q-network training are appending random bytes, appending a random library with a random function to import address table, appending a randomly named section to section table and removing signature. Rewards are provided based on number of training 'TURN' required to evade malware detection along with discount factor to consider future rewards. DQEAF also uses experience replay, which allows reinforcement learning to remember and reuse experiences from the past. The workflow of adversarial generation begins by reading the original PE malware, followed by modifications using DQEAF, and finally, correcting the virtual address for the sample with integrity insurance using Cuckoo Sandbox. DQEAF was able to alleviate the evasion rate to 70% in the same dataset as used by Gym-Malware.

Chen et al. [99] proposed an approach based on Gym-Malware [25] using Deep Q-Network (DQN) and Advantage Actor Critic (A2C) deep reinforcement algorithm and named the environment as *Gym-malware-mini*. Even though authors claimed to have increased the evasion rate by 18% than that of Gym-Malware, it could be due to data leakage. Gym-malware-mini is trained and tested using the same data as gym-malware. Eleven actions in the action space of Gym-Malware scale to an uncountable number due to randomness in each action. Gym-malware-mini converts those random actions to 10 deterministic actions, making the space for the actions very small. To balance the exploit and exploration, the best actions are chosen using the epsilon-greedy method during the network training. Smaller

TABLE 4. A reinforcement learning adversarial attacks.

Paper/Year	Target	Features	Action Space	Approach	Reward	SR
Anderson et al. 2018 [98]	Gradient Boosted Decision Trees (GBDT)	• 2350-Dimensional feature vector • Hashing trick to collapse into a vector of fixed size	10 stochastic actions for simplicity	• ACER with DQN learns both a policy model and a Q-function • Boltzman exploration and exploitation where mutation are proportional to expected Q-value • Mutations till evasion or 10 rounds	Positive:10, Negative:0	(12-24)%
Fang et al. 2019 [99]	GBDT	• Instability reduction using lower dimensional features • Feature vector of 513D	4 stochastic actions, choosen after assessing malware	• DQN with prioritized version of experience replay • Virtual address correction after modification • Integrity verification using Cuckoo Sandbox	TURN and discount factor based function	75%
Chen et al. 2020 [100]	GBDT	• Features similar to Anderson et al.'s work [98]	10 deterministic actions	• DQN and A2C based approach called as gym-malware-mini • Modifying work of gym-malware	Positive:10, Negative:-1	83%
Fang et al. 2020 [101]	Neural network based DeepDetectNet	• Import function feature, General information feature, Byte entropy features • 2478-D feature vector	200 deterministic actions	• Novel static feature extraction • RLAttackNet using DQN and optimized using double and dueling DQN • Different Q-network for choosing best action and Q-value	r= k*MAXTURN / TURN	19.13%
Quertier et al. 2022 [102]	MalConv and Ember	• Features/Raw bytes extracted as per detector • PE converted to image	16 deterministic actions	• Train DQN and REINFORCE (Monte Carlo Policy Gradient Method) • Compile a vulnerability report for the target model	Detection score dependent function	(30-100)%
Song et al. 2022 [103]	Ember, MalConv and Commercial AV	• Based on the target classifiers	Macro and Micro actions	• Adversarial attacks as a multi-armed bandit problem to balance exploiting and exploring • Limited exploration space by making generation a stateless process • Minimized changes to malware file to assign the reward correctly • Reusing successful payload in modelling	Precise rewards for essential actions	(32-97)%
Rigaki et al.2023 [104]	Ember, Sorel-LGB, Sorel-FFNN, MS-Defender	• Based on target classifier	Adapted from Malware-Gym environment	• Algorithm combining malware evasion and model extraction attacks • Model-based reinforcement learning to adversarially modify Windows PE • Trains surrogate model with the target model to evade	Positive:10, Negative:0	(32-73)%
Zhan et al. 2024 [105]	Ember, FireEye and MalConv	• 2^{20} dimensional feature vector for MalConv and FireEye • 2381 - for EMBER	6 simple operations	• Intrinsic Curiosity Module(ICM) to explore state and action spaces efficiently • Leverage GAN model to generate synthetic contents for actions	Extrinsic rewards for evasion and intrinsic rewards for the novelty of action	(63-85)%

***Target**: Target defense for adversarial attack, **Features**: Properties of features considered for processing, **Action Space**: Nature of actions in action space*
***Approach**: Key procedures, **Reward**: Reward used for learning, **SR**: Success Rate of evasion*

action spaces aid in better learning policy. Gym-malware-mini also uses negative rewards for punishment, which helps to make agents learn faster.

Fang et al. [100] tried to address shortcomings of previous work by proposing their own malware detection and adversarial generation method using DRL. MalConv [82], a standard detector network for Windows PE malware by feeding whole binary bytes, has been exploited by various researchers. Its vulnerability to gradient-based attacks for adversarial motivated authors to build their own malware detection system, DeepDetectNet, with AUC up to 0.989. For feature extraction, DeepDetecNet uses a traditional approach based on feature engineering. Static feature extraction mainly includes three categories: *Import Functions feature, General information feature, and Bytes entropy feature*. Previous success in adversarial generation using reinforcement learning is from `UPX` packing, which is not the actual modifications on PE files. To solve this problem, all random modification operations are expanded to 218 specific operations. The reward is provided in each *turn* based on constants k and *MAXTURN*, which denotes the maximum number of times a file can be modified.

TABLE 5. An adversarial attacks based on GAN.

Paper/Year	Key-Motivation	Target Model	Byte/ Feature	Approach	Feature Count
Hu. et al. 2017 [106]	Need of black-box flexible adversarial attack	ML-based (RF, LR, DT, SVM, MLP, VOTE) detectors	Feature	• Feed Forward Neural Networks are used for both generator and substitute detector • Iterative approach, modifying one feature every iteration	128 APIs
Kawai et al. 2019 [107]	Using single malware for realistic attacks	ML-based (RF, LR, DT, SVM, MLP, VOTE) detectors	Feature	• Deep Convolutional GAN used for Substitutor(S) and Generator(G) • API list from multiple clean ware and single malware	All APIs
Castro et al. 2019 [108]	Automatic byte-level modifications	GBDT Model	Byte Level	• Richer Feature representation • Generates random perturbation sequence with nine different options at each injection	2350 Features
Yuan et al. 2020 [109]	End-to-end blackbox attacks at byte levels	MalConv [82]	Byte Level	• Dynamic thresholding to maintain the effectiveness of payload • Balance in the attention of generator to payloads and adversarial samples are brought using automatic weight tuning	Raw Bytes
Zhu et al. 2022 [110]	Efficient deployment and running time	Random Forest	N-gram	• Extract n-gram features from the bytecode of malware • Train MalGan network using the features	350 features
Zhong et al. 2023 [111]	Attacking the collection of practical AV	VirusTotal	Feature	• MalFox, generation framework based on convolutional generative adversarial networks • Parser to extract features, generator to produce perturbation paths and discriminator to detect malware • Novel frameworks: Obfusmal, Stealmal and Hollowmal	16156 features
Gibert et al. 2023 [112]	Enhanced efficiency, making approach query-free	EMBER, VirusTotal	Feature	• Conditional Wasserstein GAN to generate malware resembling benign samples in feature space • Employed byte histogram, API-based and String-based features • Feature space attack mapped to en-to-end attack	Top K features

***Key Motivation**: The major motive behind the published work, **Target Model**: Target defense for adversarial attack, **Byte/Feature**: Byte or Feature selected to modify, **Approach**: Key procedures, **Feature Count**: Number of features*

Quertier et al. [101] used DQN and REINFORCE algorithms to attack popular malware detection engines, MalConv, EMBER, Grayscale and commercial AVs. On using the same action space as introduced by Anderson et al., [25], they were able to achieve an evasion rate of 67%, 100%, 98% and 30% against Ember, MalConv, GrayScale and commercial AV, respectively, using DQL while reaching even improved performance on the use of REINFORCE. To make adversarial generation more efficient and practical, Song et al. [102] proposed a black box Reinforcement Learning framework, MAB-Malware. The framework takes a problem as a multi-armed bandit to find a balance between exploiting and exploring. They limit the exploration to avoid combinational explosions while minimizing the changes to correctly attribute the rewards. The work by Rigaki et al. [103] focuses on more practical black-box attacks by focusing on model evasion and model extraction of the target. The surrogate model is trained and attacked before attacking the actual target.

All existing reinforcement learning-based approaches rely on evasion rewards for positive feedback, which, in a black-box setting, results in low training efficiency. To enhance the efficiency of adversarial generation using RNN, Zhan et al. [104] introduced the intrinsic curiosity reward into the framework that motivates the agent to explore unknown state spaces. Additionally, the authors also employed a Generative Adversarial Network(GAN) to obtain varying adversarial payloads to replace random or benign payloads.

4) GAN BASED ATTACKS

Most of the existing adversarial generation deals with the use of gradient information and hand-crafted rules. However, obtaining a high true positive rate (TPR) has been challenging due to the constrained representation ability of existing gradient-based models. Generative Adversarial Networks (GAN), originally proposed by Goodfellow et al. [112], have inspired blackbox attacks on malware detectors with very high TPR. GAN uses the discriminative model to distinguish between generated and real samples and a generative model to fool the discriminative model between generated and real samples. Table 5 summarizes adversarial attacks against Windows anti-malware engines.

Hu et al. [105] proposed an adversarial generation technique, MalGAN, which can bypass black-box machine learning models. Binary features obtained by the presence or absence of API are used as input to the model, and the number of input features equals the input dimension. A generator transforms malware to its adversarial version by taking the

probability distribution of adversarial far away from the detector. Concatenating malware feature vectors with noise vectors allows the generator to produce numerous adversarial examples from a single malware feature vector. A substitute detector is used to fit the detector model and provide gradient information to train the generator.

Considering the use of multiple malware to train MalGAN affecting the performance of avoidance, Kawai et al. [106] proposed improved MalGAN with only one malware for training. MalGAN imports malware detectors for training and predicting, which is not convenient for attackers. This improved MalGAN uses Python's sub-process library to import only detection results to MalGAN. The authors also utilized all APIs used for malware to feature quantities instead of the 128 APIs used by the original MalGAN. API lists are extracted by combining multiple cleanware and single malware in order to avoid the malware detection process being driven by the addition of cleanware features to the malware file.

A few assumptions made in designing MalGAN are less realistic and limited in bypassing real malware classifiers. One such assumption is that attackers are assumed to have full access to feature space in the detector model. In addition, API features are considered too extended to represent malware. To overcome these limitations, Castro et al. [107] published a poster using the GAN approach for generating adversarial examples by injecting byte-level perturbations. The proposed model works with real PE files instead of API feature representations. Automatic byte-level real perturbation is combined with feature representation to produce adversarial examples. The use of richer feature representation and the ability to return valid PE binaries allows the system to bypass the GBDT detector and cross-evade different classifiers.

Using API sequences or feature representation demands a lot of manual tasks to get the training data. Current state-of-the-art research is directed towards end-to-end malware detection without any feature engineering effort. Yuan et al. [108] proposed a GAPGAN framework that performs end-to-end black-box attacks against malware detectors using byte-level features. Initial discrete malware binary features are mapped to continuous space before feeding to the generator network of GAPGAN, which generates adversarial perturbations to be appended at the end of original malware binaries. Dynamic thresholding preserves generated subtle perturbations while mapping back to discrete space from continuous space. The balance of the attention of the generator across payloads and adversarial samples is maintained using an automatic weight-tuning strategy. Variable input and output size give great flexibility to the GAPGAN model in contrast to prior research works.

To increase the efficiency of deployment and running time, Zhu et al. [109] introduced the idea of n-gram to expand feature sources from hexadecimal bytecode. The n-gram features obtained from both malicious and benign files are combined to form a 350-dimensional feature vector, trained with MalGAN [105] network. Changing the features to n-gram helps in carrying out attacks more efficiently. Lately, to create adversarial samples against commercial AV, Zhong et al. [110] proposed a generation framework based on convolutional generative adversarial networks. The framework majorly consists of a PE parser to extract features, a generator to produce the perturbation path, a PE Editor to edit, a Detector and a Discriminator to identify malware. Their approach uses their distinct frameworks, Obfusmal, Stealmal and Hollowmal, to attack a VirusTotal as their target detector. Another approach by Gibert et al. [111] attempted to resolve the high query requirement of GAN-based approaches by proposing a conditional Wasserstein GAN. They generate the malware sample resembling to a benign file in feature space and later map it to end-to-end problems. The approach's generalization power is demonstrated on different features: byte distribution, functions, libraries and strings. Devadiga et al. [113] utilized GAN to further enhance the GAN-based approach, fusing opcode and n-gram features with LLM embeddings. Since the approach utilized LLM and GAN as separate entities, exploring how the completely merged approach will generate such attacks is still interesting.

5) RECURRENT NEURAL NETWORK BASED ATTACK

Recent works have focused on the use of Recurrent Neural Networks (RNN) for malware detection and classification [4], [114]. Sequential malware API is used by RNN to predict whether the program is malware or benign. Papernot et al. [115] introduced adversarial sequence for RNN processing sequential data. The authors demonstrated the transferability property of adversarial examples generated from feed-forward neural networks against recurrent neural networks. Table 6 summarizes a comparison among RNN, explainable ML, malware visualization, Generative AI, and Genetic algorithm-based adversarial attacks. Hu et al. [51] proposed an RNN-based adversarial attack for an RNN malware detector. The approximation of the victim RNN model is done by training substitute RNN, and generative RNN outputs sequential adversarial examples. Some irrelevant API sequences are generated and inserted in vulnerabilities in the original sequence. API sequences, represented as a one-hot vector, are the input for the generator network, which generates adversarial API sequences. The generative part of RNN generates small API sequence pieces after each API, which are inserted after the API. A benign sequence and the Gumbel-Softmax [116] output are used to train the substitute network to fit the victim RNN-based detector. The attention mechanism helps by spreading the focus on different parts of the sequence. Using this approach, authors were able to decrease the initial detection rate of around 90% across all malware detectors to around (1)-(3)%, showing almost perfect evasion against all the cases.

6) EXPLAINABLE MACHINE LEARNING BASED ATTACK

One of the biggest challenges of machine learning is the lack of explainability or reasoning behind such intelligent

TABLE 6. RNN, Explainable ML, Visualization, Generative AI, and Genetic algorithm based adversarial.

Paper/Year	Key-Motivation	Target Model	Algorithms Used	Approach	SR
Hu et al. 2017 [51]	Attack against RNN processing sequential data	LSTM and BiLSTM-based detectors	Bidirectional RNN with attention mechanism	• Substitute RNN approximates victim RNN • Generative RNN gives a sequential adversarial example • Irrelevant API sequence generated and inserted in vulnerabilities of the original sequence	97%
Rosenberg et al. 2020 [118]	Use of explainable machine learning for adversarial generation	GBDT Classifer	Integrated Gradient, LRP, DeepLIFT, SHAP	• Unearthing most impactful features using explainability algorithm • Manual selection of easily modifiable features • Feature by feature modification without harming functionality and interdependent features	37%
Aryal et al. 2024 [119]	Using explainability to enhance the efficiency of adversarial perturbation	MalConv	Gradient Descent	• Calculate SHAP values corresponding to different regions of PE malware • Devise adversarial injection strategy based on calculated attribution	58%
Liu et al. 2019 [120]	Adversarial malware against visualization based detection	CNN, SVM and RF based malware detectors	ATMPA framework using GoogLeNet, FGSM and C&W	• Data transformation to convert code segments into grayscale images • Pre-training module to find function of malware detectors • Optimized FGSM and C&W attack is used to generate actual AE	100%
Khormali et al. 2019 [121]	Targetted and Untargetted misclassification on windows and IoT malware dataset	Convolutional Neural Network	FGSM, C&W, DeepFool, MIM and PGD	• Adversarial generation using different algorithms • Conversion of adversarial dimension, same as of original image • Appending pixels at the end or injecting	99%
Benkraouda et al. 2021 [122]	Attack against visualization based detection with ability to evade pre-processing filtering without losing functionality	Convolutional Neural Network	Modified version of CW attack [57], Euclidean distance	• Mask generator to flag the locations for perturbation • Modified version of CW attack to generate optimal perturbation • NOP generator to replace the perturbation from CW attack by semantic NOPs • AE optimizer to choose optimal viable NOPs	98.9%
Hu et al. 2021 [123]	Increase efficiency of attack against black-box detectors	VirusTotal	Fine-tuning GPT2	• Fine-tune the GPT2 language model to generate benign looking byte sequence • Append generated sequence from GPT2 to a malware sample	28%
Yuste et al. 2022 [70]	Create flexible, functionality-preserving attacks using code caves	MalConv and VirusTotal	Genetic algorithm	• Introduction of space in between PE section that won't be loaded to memory • Optimization by exploring search space using genetic algorithm (selection, crossover, and mutation)	81%

***Key-Motivation**: The major motive behind the published work, **Target Model**: Target defense for adversarial attack, **Algorithms Used**: Algorithm used for crafting adversarial example, **Approach**: Key procedures to carry out adversarial attack, **SR**: Evasion Success Rate*

decisions. Recent researchers have been able to bypass malware detectors using the concept of explainable machine learning. The explainability approach involves finding the significance of each feature and then conducting feature-specific modifications based on their importance. Rosenberg et al. [117] proposed an explainable ML approach to generate adversaries against multi-feature type malware classifiers. Adversarial attackers first evaluate the most effective list of features, and the features that are easy to modify are selected. Transferability of explainability allows the proposed attack to achieve a very high impact on the target classifier, even in a black-box attack. This approach assumes that the malware classifier and the substitute model possess similar feature importance, leading to modification in features to impact the target malware classifier. Different explainability algorithms on white-box [88], [123], [124] and black-box [125] are evaluated to make comparisons between substitute model and victim model. The proposed end-to-end PE adversarial attack performs feature modification without harming the malware functionality as well as interdependent features, giving an evasion rate as high as 34% compared to 0.11% when adding random perturbation. Using naive and engineered features of the EMBER dataset, the explainable ML approach successfully bypasses the GBDT classifier. Rosenberg et al.'s work presents explainability as a dual-edged sword that can be used by adversaries to make more explainable models and carry out more robust adversarial attacks.

One of the recent works by Aryal et al. [118] demonstrated the use of explainability as a tool to enhance the efficiency of adversarial attacks. Their approach uses explainability to attribute the different regions of Windows PE malware

based on their contribution towards being detected by malware detectors. The attribution is later used to derive the adversarial injection strategy. They demonstrated an increase of efficiency by more than 200% on using the explainability to assist the attack. The amount of increase and success of the attack approach is dependent on the region of the PE malware file. Their experiments demonstrated an evasion rate as high as 58% on attacking `.text` section, 36% with `.data` section and 49% while targeting `.rdata` section.

7) MALWARE VISUALIZATION BASED ATTACK

Machine learning-based visualization detection has been popular due to its ability to prevent zero-day attacks and make detection without extracting pre-selected features [126], [127]. These approaches convert binary code into image data and visualize the features of the sample, improving the detection speed for malicious programs. Visualization-based techniques are similar to the adversarial generation in the image domain where pixel perturbations are introduced, as shown in Figure 8. Liu et al. [119] introduced an Adversarial Texture Malware Perturbation Attack (ATMPA) against visualization-based malware detection using a rectifier in neural network hidden layers. The framework allows an attacker to probe with the malware image while visualizing and also hiding them from malware detectors. Code segments are converted into grayscale images during the data transformation module. In the adversarial pre-training module, an attacker uses a machine learning approach to train an adversarial example generation model, producing a noise signal δ. For a generation of AEs, optimization algorithms, FGSM and C&W attacks are used. ATMPA method also used L_p-based C&W attack to generate adversarial, including l_0, l_2 and l_∞ attack. Their approach produced a perfect evasion rate of 100% in most of their attacks.

COPYCAT approach proposed by Khormali et al. [120] produced both targeted and untargetted misclassification on Windows and IoT malware datasets. The author used two approaches, AE padding and sample injection, to produce adversarial malware for visualization-based detectors. For padding method, COPYCAT generated adversarial x' using five different attack methods namely: FGSM [22], C&W [57], DeepFool [59], Momentum Iterative Method (MIM) [128] and Projection Gradient Descent (PGD) [129]. The generated adversarial needs to be converted to the same dimensions as that of the original image before appending at the end of the image. The binary samples from the targeted class are injected into an unreachable section of the target sample, producing an evasion rate as high as 99% in almost all attacks.

In order to provide an adversarial attack that can evade visualization-based detection in the presence of preprocessing filtering, Benkraouda et al. [121] proposed a binary rewriting-based attack on malware files. A mask generator creates the space in the instruction boundary to insert the perturbations. Once the perturbation mask is created, the modified version of C&W attack [57] is used to generate an adversarial example in image space. The modified version is in the sense that the perturbation mask is imposed while carrying out an attack to restrict the positions of perturbations. The NOP generator will replace the perturbation introduced by the C&W attack with the corresponding binaries that preserve the malware functionality. Finally, the AE optimizer will use the Euclidean distance metric to choose semantic NOPs that are close to sequences in the allowed perturbation space. Their approach produced a high evasion rate of 98.9% when tested with just 174 malware samples.

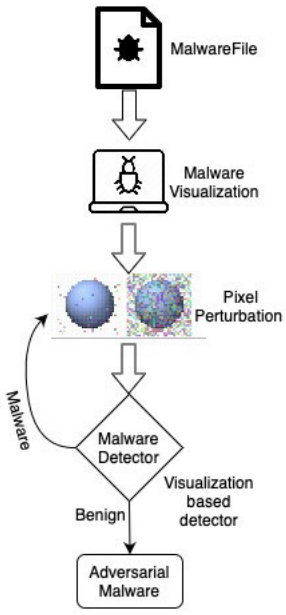

FIGURE 8. An adversarial generation against malware visualization based detection.

8) MISCELLANEOUS

Generative AI has been the biggest buzzword and is being tested across every domain. To make adversarial attacks more efficient, overcoming the query limit imposed in black-box attacks, Hu et al. [122] proposed MalGPT that demonstrates the ability of a Deep Learning-based causal language model to enable a single shot evasion. The approach fine-tunes the GPT-2 model with the benign files and tests it against the VirusTotal to achieve an evasion rate of around 24.51% with just a single shot attack.

In another approach, Yuste et al. [70] tried to bring flexibility while preserving malware's functionality in the adversarial creation process. The author's approach introduces code caves between the PE malware sections in the disk that never get mapped to the memory. In the creation of code cave, authors follow genetic algorithms through selection, crossover and mutation to create an adversarial malware sample. They were able to achieve an evasion rate as high as 99% and significant success against VirusTotal detectors as well.

TABLE 7. An Android adversarial attacks.

Paper/Year	Target/Dataset	Approach	Modification	Limitations
Grosse et al. 2017 [131]	Feed forward neural network based detector / DREBIN	• Binary feature vector extraction using static evaluation • Jacobian matrix of neural network for adversarial generation • Direction for generated perturbation is given by gradient of the given function with respect to the input • Selection of perturbation with maximal positive gradient towards target class	• Feature addition to AndroidManifest.xml • Changing features leading to only one line of code	• Constrained on maximum feature space perturbation • Feature modifications confined inside AndroidManifest.xml
Yang et al. 2017 [132]	KNN, DT, SVM, RF/ DREBIN, VirusShare, Genome	• Malware Recomposition Variation by semantic analysis • Feature mutation analysis and phylogenetic analysis to perform automatic program transplantation • Malware evolution attack focusing on mimicking and automating the evolution of malware • Conufsion attack making features less differentiable	• Resource, temporal, locale and dependency features used • Mutation following feature pattern of existing malware	• Significant alteration of semantic leading to higher failure rate of app
Rosenberg et al. 2018 [73]	RNN variaant and Feed forward neural networks/ VirusTotal	• Mimicry attacks against surrogate model • Surrogate model by querying black-box detectors with synthetic inputs selected by Jacobian based heuristics in prioritizing directions • Closest API call in direction indicated by Jacobian are selected	• No-op attack by adding API call with valid parameters • Functionality verification using sandbox after modification	• Detectable Residual artifacts during app transformation
Liu et al. 2019 [133]	Neural network, logistic regression, DT and RF based detectors/DREBIN	• Random forest to filter most significant features • Disturbance randomly generated and disturbance size calculated using genetic algorithm • Mutation using fitness function till fit and evading individual is produced	• Restricted permission modification on AndroidManifest file • Functionality changing modifications are deemed unfit	• Increased constraint on perturbation • Random perturbation affecting convergence
Shahpasand et al. 2019 [74]	SVM, Neural network, RF and LR / DREBIN	• GAN architecture with threshold on generated distortion • Different loss function to generate benign like adversarial and to produce high mis-classification	• Perturbation addition limited by threshold distortion amount	• Highly unstable learning of GAN architecture
Li et al. 2020 [72]	AdaBoost, CNN, SVM / Tencent Myapp, AndroZoo	• Bi-objective GAN with two discriminator and one generator • One discriminator to distinguish malware and benign sample and another to distinguish original and adversarial sample	• Iterative perturbation addition till evasion • Perturbation evading both malware and adversarial detection	• Very limited feature vectors (Permission, action and API calls) are considered
Pierazzi et al. 2020 [75]	Linear SVM, Sec-SVM/DREBIN	• Formalization of adversarial evasion attacks in the problem feature space including transformations, semantics, robustness and plausibility • Automated software transplantation to extract benign slices • Side effect features to find projections that maps perturbation to feasible problem-space regions • Gradient based strategy based on greedy algorithm to choose perturbation	• Perturbations appended at the end • Restricted addition of permissions • Cyclomatic Complexity to take heuristic approach maintaining existing homogeneity	• Heuristic based approaches are time and resource consuming
Bostani et al. 2024 [134]	DREBIN [135], Sec-SVM [136], MaMaDroid [137] / AndroZoo [138]	• Automated Software Transplantation Technique to prepare action set which includes gadgets extracted from benign Android apps • n-gram-based similarity method to identify benign APKs, closely similar to malware files • Applying extracted gadgets from benign samples into malicious files • Iterative and incremental manipulation	• Random Search(RS) for moving malware sample in problem space applying sequence of transformation in action set • New contents injected inside an IF statement	• In Random Search (RS) algorithm, actions from action space are random • Increase in adversarial size, increasing chances of adversarial detection

***Target/Dataset**: Target defense for adversarial attack/Dataset used, **Approach**: Key procedures to carry out adversarial attack, **Modification**: Changes on file to craft the adversarial perturbation, **Limitations**: Shortcomings of proposed approach*

B. ANDROID MALWARE ADVERSARIAL

Android has over 2.8 billion active users and owns 75% market share in the mobile phone industry [138]. The wide usage of the Android platform has attracted security threats in numerous forms, and adversarial evasion attacks are one of them. Table 7 provides a brief comparison among different adversarial attacks crafted on Android files. Grosse et al. [130], [139] generated adversarial examples for state-of-art Android malware detection trained on DREBIN dataset [134]. Authors migrated the method proposed by Papernot et al. [56] to handle binary features of Android malware while preserving the malicious functionality. Binary features are derived by statically evaluating code based on system calls and usage of specific hardware. Authors adopted

the Jacobian matrix of neural network F for an adversarial generation. To get adversarial, the gradient of the function F with respect to X is calculated to get the direction of perturbation such that the output of classification will change. Perturbation δ with the highest positive gradient in the direction of the target is selected and is kept small enough to prevent negative change due to intermediary alterations of the gradient. Functionality is preserved in this approach by changing features, resulting in the addition of only a single line of code. Research also confines the modifications to manifest features related to `AndroidManifest.xml` file contained within the Android application. With permissions, intents and activities being the most frequently modified features, authors successfully evaded DREBIN classifier [134], preserving the semantics of malware.

To overcome the white box attack issues, Rosenberg et al. [73] implemented the GADGET framework to convert malware binary to an adversarial binary without access to malware source code. The proposed end-to-end black-box method is extended to bypass the multi-feature-based malware classifiers relying on the transferability in RNN variants. For the target RNN detector, a malicious API call sequence is the adversarial example to be generated. Adversaries train a surrogate model with the same decision boundaries as the detector and then execute a white-box attack on the surrogate model. The black-box detector is queried with synthetic input values from chosen Jacobian-based heuristics in the prioritizing directions where model output varies to build the surrogate model. API calls that are nearest to the direction given by Jacobian are inserted to generate the adversarial sequence. The Jacobian matrix of the surrogate model is used for evaluation, and after each iteration, a synthetic example is added to each existing sample. Adversarial generation showed the same success against the substitute and blackbox model with short API sequences, making adversarial generation faster. Framework also uses Cuckoo Sandbox to verify the malicious functionality of generated adversarial malware. GADGET framework wraps malware binary with proxy code and increases the risk even higher, providing malware-as-a-service.

Adversarial attacks on the malware domain have not considered manipulating the feature vector to see the impact of mutation due to the strict functionality-preserving requirements of malware. The Malware Recomposition Variation (MRV) based approach proposed by Yang et al. [131] performed an analysis of malware files semantically and constructed a new malware variant. Mutation strategies synthesized by conducting semantic-feature mutation analysis and phylogenetic analysis are used to perform automatic program transplantation [140]. The proposed framework performs inter-component, inter-app, and inter-method transplantation. A more comprehensive attack is performed on the manifest and the as dex code. Malware evolution attacks aim to imitate and automate malware evolution using phylogenetic evolutionary tree [141].

Several adversarial generation approaches have been conducted with minor changes to existing attacks. Liu et al. [132] proposed a Testing framework for Learning-based Android Malware Detection systems (TLAMD). Framework uses a genetic algorithm to perform black-box attacks against the Android malware detection system. Android files are modified by adding the request permission codes to the AndroidManifest.xml file originally proposed by Grosse et al. [130]. The restriction was imposed on the types and magnitude of permissions that can be added to the AndroidManifest file. A random population is generated giving the characteristics of permission to add and followed by calculating the disturbance size for the sample malware. Using the evaluated perturbation size, adversarial is generated and tested against the detection model. Based on the detection result, either a new disturbance size is calculated using genetic algorithms, or perturbation is successfully added to the Android application. The fitness function searches for optimal solutions to perform a mutation, leading to a new fit individual able to evade detection. A random forest approach filters out insignificant features during feature extraction. Disturbances generated by genetic algorithms can bypass malware detectors trained on neural networks, logistic regression, decision trees and random forest.

Shahpasand et al. [74] implemented GAN to generate adversarial by keeping a threshold on the distortion values of generated samples. The generated optimum perturbation δ is added to existing malware to produce adversarial. Like every other GAN architecture, the generator can learn the distribution of benign samples, generating perturbations that can bypass the learning-based detectors. The discriminator implicitly enhances the perturbation by escalating the loss of the generator while the adversarial samples are identifiable with benign files.

The goal of adversarial generation has been to bypass malware detectors without losing functionality. However, due to the growth in adversarial malware in recent times, defenders are employing firewalls to stop the adversarial samples. Li et al. [72] extended the work of MalGAN [105] to make it robust against a detection system equipped with a firewall. Despite its high evasion rate against malware detectors, MalGAN is found to be less effective against detection systems using firewalls. Bi-objective GAN with two discriminators with different objectives is used. One discriminator helps distinguish between malware and benign, whereas another discriminator helps to find out whether the samples are adversarial or normal. Authors used permissions, actions and application programming interface calls as a feature to generate adversarial.

Pierazzi et al. [75] formalized the adversarial ML evasion attacks in the problem space and proposed a problem space attack on Android malware. This work is focused on attacks modifying the objects in real input space corresponding to

TABLE 8. Adversarial attacks on PDF malware.

Paper/Year	Target	Approach	Modification	Limitations
Maiorca et al. 2013 [144]	PJScan, Malware Slayer and PDFRate	• Reverse mimicry attack by manipulating binary files to make it malicious • Malicious embedded EXE payload insertion • Malicious PDF file insertion inside a benign one • Encapsulating malicious JavaScript code	• Malicious EXE payload as a new version after trailer • Unrestrained embedded PDF structure insertion	• Less control on malicious goal
Biggio et al. 2014 [83], [145]	SVM and neural network based detectors	• Gradient based optimization inspired by Golland's discriminative directions technique • Additional panalizing term to reshape objective function, biasing gradient descent towards region of negative class concentration	• Insertion of objects creating new PDF files	• Feature mapping issues • Non-differential discriminating functions can not be evaded
Srndic et al. 2014 [146]	PDFrate employed on Random Forest	• Taking advantage of discrepancy between operation of PDF reader and PDFrate • Mimicry attack to mimic 30 different benign files • GD-KDE attack to defeat classifier with differentiable decision function	• Insertion of dummy contents, ignored by PDF readers but affect detector • Trailer section moved away from cross reference table for file injection space	• Feature mappings are assumed to be perfect which is unrealistic
Carmony et al. 2016 [147]	PDFrate and PJScan	• Reference JavaScript extractor by directly tapping into a Adobe reader at locations identified by dynamic binary analysis • Parser confusion attack combined with reverse mimicry attack	• Obfuscation based on output of reference extractor	• Useful only for JavaScript based detector • Dependent on versions of Adobe Reader
Xu et al. 2016 [148]	PDFrate and Hidost	• Stochastic manipulations using genetic algorithm to generate population • Iterative population generation till evasion • Successful mutation traces reused for initialization efficiency • Fitness score based on maliciousness detected by oracle	• Inserting new, removing and modifying existing contents • Oracle confirming the maliciousness of file	• Stochastic approaches are resource intensive • No exact way to choose best fitness function

***Target**: Target defense for adversarial attack, **Approach**: Key procedures to carry out adversarial attack, **Modification**: Changes on file to craft the adversarial perturbation, **Limitations**: Shortcomings of proposed approach*

the feature vectors. To overcome the inverse feature-mapping problems from previous research, the author presents the idea of side-effect features. An attack on a feature space is projected towards a feasibility region satisfying the problem space constraints to obtain the side effect features. Though side effect features contribute towards preserving the validity of malware, they alone can positively and negatively influence the classification score. Authors use automated software transplantation [140] to extract byte-codes from benign donor applications to inject into a malicious host, also known as organ harvesting. Prior research relied heavily on adding permissions to the Android Manifest, which is considered dangerous in Android documentation [142]. Authors bind the modifications to inject a single permission into the host app. The gradient-based strategy using the greedy algorithm proposed in this approach overcomes previous limitations of preserving semantics and pre-processing robustness.

To overcome the challenges of limited access to target classifiers while circumventing black-box Android malware detectors, Bostani et al. [133] proposed a novel iterative and incremental manipulation strategy. The attack is carried out in two steps: preparation and manipulation. In the preparation phase, automated software transplantation prepares action sets from Android apps. The n-gram-based similarity method is used to identify benign apps that closely match malware files. Insertion of extracted gadgets of closely matching benign files forces malware samples towards the unseen spots of the classifier. In the manipulation stage, the perturbation on malware samples is applied incrementally, choosing from the collected action set. The search method randomly chooses suitable transformations and applies them to malware samples. This approach shows a high success rate in query-efficient approaches but increases the size of adversarial perturbation, increasing the risk of perturbation being easily detected.

C. PDF MALWARE ADVERSARIAL

Along with widespread applications and adoption, PDF documents have been one of the most exploited avenues for adversarial malware attacks. Initially, JavaScript-based and structural properties detection was prominent in recognising malware in PDF. The freedom to distribute chunks of Javascript code and assemble them at run-time and the high degree of expressiveness in JavaScript led to the failure of Javascript-based detection. Despite significant growth in PDF malware detection from JavaScript using deep learning techniques, the challenges posed by adversarial examples still exist. Early evasion attempts on PDF documents were crafted by Smutz et al. [148] and Šrndić et al. [149] using heuristic approaches. The authors proposed an approach to build more robust PDF malware detection techniques that showcased the adversarial ability to mislead linear classification algorithms successfully.

The flexible logical structure of PDF has allowed us to craft adversarial by carefully analyzing its structure. Maiorca et al. [143] demonstrated an evasion technique called reverse mimicry attack against popular state-of-art malware detectors [148], [150], [151]. Traditionally, malicious PDF files are believed to be structurally different from benign PDF files. Taking advantage of this structural difference, most malware detectors were able to discriminate PDF files with very high accuracy. However, malware files that can imitate the benign file structure or vice-versa can easily fool the detector. Reverse mimicry attacks can make benign files malicious with minimal changes in their structure. Malicious payloads poison the samples, initially classified as benign. Three kinds of malicious payloads introduced to benign files take the sample across the decision boundary of the malware detector. The first one is the EXE payload with malicious embedding, which is introduced using the Social Engineering Toolkit as a new version after its trailer. The new trailer will point to a new object when adding a new root object. In this payload, authors embedded malicious PDF files inside other benign PDF files using the embedded function of PeePDF [152] tool. The embedded PDF file automatically opens without user interaction, allowing malicious PDF to be embedded inside a benign one without any restriction on embedding the file. PDF file injection enabled an attacker to have fine-grained control of structural features in the carrier file. A final kind of payload insertion is carried out by encapsulating a malicious JavaScript code without reference to other objects. Table 8 provides an overview of adversarial attacks carried out on PDF files.

Optimization-based evasion attack against PDF malware detection was introduced by Biggio et al. [79], [144]. The attack was carried out using a gradient-based optimization procedure inspired by Golland's discriminative directions technique [153] to evade linear as well as non-linear classifiers. The proposed work carried out complete knowledge and constrained knowledge attacks on non-linear models like Support Vector Machine(SVM) and neural networks. This approach used a gradient descent procedure with special consideration to avoid getting stuck on local optima. To increase the probability of successful evasion, an attacker needs to reach legitimate attack points and to reach this, the additional penalizer term is introduced using a density estimator. The extra component helps imitate features of known legitimate samples, reshaping the objective function by biasing the gradient descent towards the negative class concentration region.

Srndic et al. [145] further enhanced optimization based attack against deployed system PDFrate [148] using mimicry attack, and Gradient Descent and Kennel Density Estimation (GD-KDE) attack. The attack takes advantage of the discrepancy between the functioning of PDF readers and PDFrate in terms of interpretation of semantic gaps as explained in [154]. The dummy contents to insert should be ignored by PDF readers but affect the feature computation in PDFrate. PDF

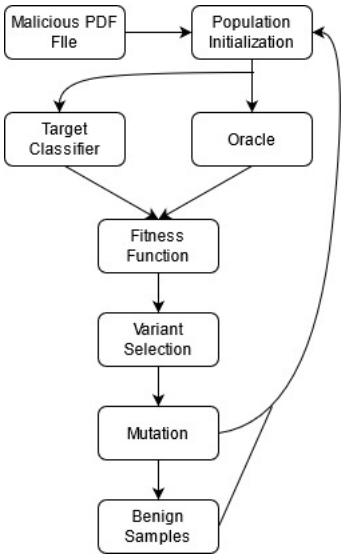

FIGURE 9. A PDF adversarial malware generation based on genetic algorithm [147].

reader looks at the end of the PDF for the cross-reference table and goes to locate the object directly. The trailer section of PDF files was moved arbitrarily far away from the cross-reference table, generating a space for file injection without affecting the functionality of the PDF document.

PDF detection techniques mostly rely on PDF parsers to extract features for classification [150], [155]. These parsers are unable to extract all JavaScript of PDF files. Carmony et al. [146] created a reference JavaScript extractor that measured the difference between the parser and Adobe Reader by tapping Adobe Reader on locations given by binary analysis. Manual analysis refines the few candidate tap points provided by dynamic binary analysis. JavaScript extraction tap points help Adobe Reader extract and execute JavaScript code from PDF documents. The memory accessed by Adobe Readers when reading PDF files using automatically executable JavaScript is analyzed to determine the raw JavaScript extraction tapping points. The proposed PDF parser confusion attack applies obfuscation on malicious PDF samples by analyzing the weaknesses of extractors. Reference extractor enables new obfuscation in comparison to existing extractors, and a combination of these obfuscations was able to bypass all JavaScript extractor-based detectors.

In order to preserve maliciousness, several works take a conservative approach by inserting only new content and refraining from modifying or removing existing content. Xu et al. [147] proposed a black-box generic method to evade the classifier as shown in Figure 9. As in the figure, first, the population is initialized by performing random modifications on malicious files. Then, each member of the population is passed through a target classifier to measure maliciousness and through Oracle to confirm the functionality. Suppose no samples can evade the target classifier with functionality intact. In that case, a subset of the initialized population is chosen for the next generation

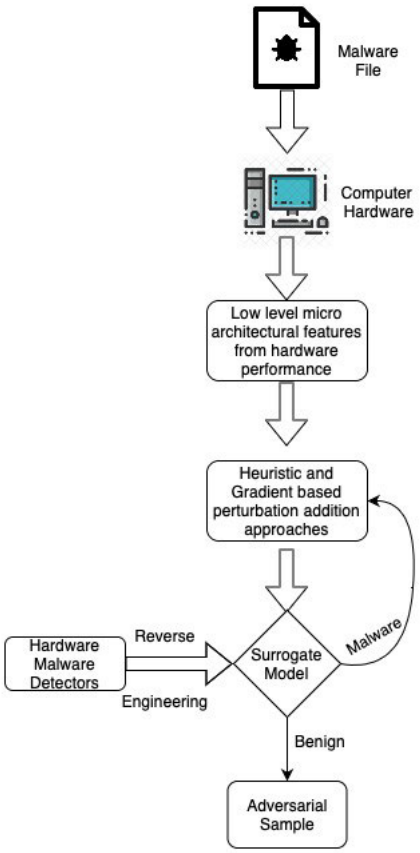

FIGURE 10. An adversarial generation workflow against hardware malware detectors.

based on the fitness score, indicating progress towards the evasive sample. The population generation is repeated, and this process is continued until the evasive sample is found or threshold iterations are met. The author uses genetic programming (GP) to bring off stochastic modifications in an iterative manner till evasion.

D. HARDWARE BASED MALWARE ADVERSARIAL

Hardware malware detectors use low-level information on features from hardware performance monitoring units available in CPUs. Hardware malware detectors are prone to reverse engineering [159], allowing mimicry attack [160] to reverse-engineer the models. Adversarial against such detectors are carried out by generating perturbations in the form of low-level hardware features, following the architecture shown in Figure 10. These adversarial generation approaches differ only in the type of features used in comparison to previous works. Table 9 briefly compares adversarial attacks against hardware malware detectors. Khasawneh et al. [156], [161] demonstrated evasion of Hardware Malware Detectors(HMD) after being reverse-engineered, using low overhead evasion strategies. Data collected by running malware and cleanware programs on a virtual machine operating on Windows 7 are used to train a reverse-engineered model. Data required for training are dynamic traces while executing the program and are collected by using the Pin instrumentation tool [162]. These dynamic traces are profiles of the program's run time behaviour. This dataset is comprised of three types of feature vectors: *Instruction features, Memory address and Architectural events*. Authors [156] constructed a Dynamic Control Flow Graph (DCFG) of the malware to insert instructions into the executing malware dynamically. Injection of instruction features increases the weight of the corresponding feature, while memory feature injection alters the histogram of memory reference frequencies. Khasawneh et al. picked the instructions with negative weights to move the malware away from the decision boundary. A heuristic approach was taken to identify the candidate instructions for insertion. Weighted injection strategy where the probability of selecting particular instruction is proportional to negative weight allowed to bypass HMD with around 10% dynamic overhead.

Dinakarrao et al. [157] also proposed an adversarial attack on low-level micro-architectural events captured through Hardware Performance Counters (HPC). Victim's defense system (HMD) being black-box needs to be reverse engineered to mimic the behaviour. The number of HPC patterns required to bypass HMD is unknown, which leads to the need for an adversarial sample predictor. The HPC patterns perturbing mechanism are implemented using a lower-complexity gradient approach, Fast Gradient Sign Method (FGSM). The adversarial perturbations needed to misclassify the HPC trace are calculated using the cost function of the neural network. With θ being hyperparameters of neural network, x is input HPC trace to the model and y as output, cost function $L(\theta, x, y)$ is defined as:

$$x^{adv} = x + \epsilon \; sign(\Delta_x L(\theta, x, y)) \tag{9}$$

where ϵ is a scaling constant ranging between 0.0 to 1.0 and is used to limit the perturbation to a very small value. The LLC load misses and branch misses are the most significant micro-architectural events of malicious applications [163]. Malicious circuits, or hardware Trojans, can be inserted into circuits producing logically equivalent results. Modifications in the manufacturing stage are more tedious than in the design stage, as few changes in a hardware description language (HDL) are enough to embed hardware Trojans into the circuit. A trigger circuit allows the payload circuit to trigger malicious behaviour, such as information leakage and degrading performance, after satisfying the trigger condition.

Nozawa et al. [158] proposed an architecture to develop the adversarial hardware Trojan using Trojan-net Concealment Degree (TCD) and Modification Evaluating Value (MEV). Feature mapping issues, like in all other adversarial attacks in Windows, Android and PDF, are also prevalent in hardware Trojan. Hardware circuits are represented in a graph structure, and modifications in feature space do not guarantee the transfer back of modification to the graph structure. Two stages in the designing period are known for adversarial attacks. The first is the RTL (Register-Transfer Level) description design step, and the second is after logic

TABLE 9. Summary of hardware based malware adversarial.

Paper/Year	Target Model	Data Collection	Feature Vectors	Approach
Khasawneh et al. 2017 [157]	Logistic Regression and neural network based detectors	Dynamic traces collected using Pin instrumentation tool	• Instructions Feature • Memory address patterns • Architectural Events	• Reverse engineering to create surrogate model of HMD • Dynamically instruction insertion into malware execution through Dynamic Control Flow Graph • Weighted injection strategy with insertion instruction selection proportional to negative weight
Dinakarrao et al. 2019 [158]	Logistic Regression and neural network based detectors	Captured using Hardware Performance Counters (HPC)	• LLC load misses, branch instructions, branch misses and executed instructions	• Reverse engineering of Black-box HMD • HPC patterns perturbation mechanism determined using FGSM • Perturbation calculated using neural network • Adversarial generators running as separate thread to avoid interference with original source code
Nozawa et al. 2021 [159]	Neural network architecture	Structural features analysis	• Gate level netlist	• Hardware circuits represented in graph structure and converted to feature space • During design step or after logic synthesis • Trojan-net concealment degree to prevent from detection

***Target Model**: Defense model under adversarial attack, **Data Collection**: Feature value collection process, **Feature Vectors**: Types of features considered, **Approach**: Process of crafting adversarial*

synthesis. The authors take the assumption of the Trojan detector using neural network architecture and the availability of raw output values from the detector to train the adversarial model.

E. LINUX MALWARE ADVERSARIAL

Distributed edge computing has increased the use of IoT devices. With many devices using Linux systems, robust malware detection is paramount. Both deep learning networks and Control Flow Graph (CFG) based malware detectors in IoT devices are found to be vulnerable against adversarial samples [164]. In off-the-shelf adversarial attacks, authors examined different well-known adversarial algorithms based on feature extraction. Generic adversarial algorithms are successful in adversarial generation with a high evasion rate but are limited in applying practical changes to feature space. In response to these challenges, adversarial based on a control flow graph has been proposed [164]. Programs are structurally analyzed using vertices and edges with the help of CFG. The graph embedding and augmentation (GEA) approach combines the original graph with the target graph, producing misclassification while preserving the original program's functionality. GNU compiler collection command compiles in a way that only functionality related to the original sample is executed. Linux-based malware binaries easily evade IoT malware detection using different graph algorithmic constructs. Our literature search found minimal works carried out as adversarial malware attacks in the Linux domain and also found that Linux and Android file systems are used interchangeably.

VI. CHALLENGES AND FUTURE DIRECTIONS

Following the introduction of evasion attacks against deep learning by Szegedy et al. [19], the research community is concerned about its impact in different domains. To contribute towards the literature, we conducted comprehensive research on various adversarial evasion attacks carried out against the malware detection domain. Although our survey highlights several successful adversarial attacks crafted against anti-malware engines, novel attacks are still evolving. In this section, we will discuss potential research open challenges and future direction as the adversarial approaches in malware analysis domain become more prevalent. Our intention is in no way to overlook or understate the contributions of existing adversarial attack researchers in the malware domain.

A. REALISTIC (PRACTICAL) ATTACKS

Most of the existing adversarial attacks in the survey are carried out using white-box approaches. White-box approach is an unrealistic scenario in itself as it is unlikely that any ML-based anti-malware engine will reveal information such as algorithms used, gradients of the model and hyper-parameters used to fine-tune the model. Getting this information about a target model provides a 'superpower' to attackers as they can camouflage the data in any way they want. Few of the existing black-box attacks also depend on the performance of models provided in numeric form. In addition, most of the works are centred on static malware detection. Modern industrial malware detection engines merge both static and dynamic detection techniques. Further, the attacker rarely gets the privilege to work with data at rest. There have been very few successful attempts to craft adversarial examples against data in motion [165], [166]. The malware domain can have data that is moving at a very high pace and may require performing an attack on data in motion. Adversarial attacks are not always swift enough to work with data moving across network channels. So, more adversarial

attacks are to be experimented with systems deployed with both static and dynamic detection as well as against data at motion.

B. PERTURBATION INSERTION SPACE

Smart perturbation insertion plays a key role in the success of adversarial attacks. Initial adversarial evasion attacks on malware began by placing perturbations at the end of the malware file [83]. Most of the existing attack approaches are concentrated on additive adversarial perturbation. Demetrio et al. [27] later discovered that perturbations embedded at header sections of files resulted in effective adversarial attacks compared to perturbations appended at the end. Suciu et al. [28] further investigated the possibility of inserting perturbations in slack regions of file which are left behind by the compilers. These experiments provide inconclusive information about suitable insertion space for perturbation. Hence, further research is needed to determine optimal locations for perturbation that are more effective as well as undetected.

C. ENHANCING EFFICIENCY

Adversarial efficiency can be defined in terms of different parameters. One of the efficiency criteria is the length of the payload to be generated/injected. The significance of the inserted payload determines the efficiency of perturbation. One way to insert efficient features is to first decipher the importance of each feature in the decision-making of the machine learning model. Despite the gradient helping attackers to generate perturbation in the right direction, efficiency may be limited due to uncountable iterations to reach the adversarial goal. Applying small perturbations iteratively results in high-quality adversarial evasion. However, these approaches will require an immense amount of time, making it impossible for real-time operation. To challenge this limitation approaches like the Fast Gradient Sign Method are proposed, which produce perturbations at a very high pace but are less effective and have a high chance of being detected. Hence, research is needed to ensure that efficiency is looked at both in terms of quantity and quality of noise generated to produce adversarial evasion. In addition, the trade-off between performance and computational complexity should be analysed to evaluate the worth of performing adversarial attacks [167].

D. MAPPING SPACE CHALLENGE

Mapping between problem space and feature space is performed by an embedding layer present in between them. The features in problem space can be of any form, like n-grams, API names or other non-numeric parameters, which can not be directly processed by machine learning models. This causes the problem space vectors to be converted into feature space which are some form of numeric values. The embedding layer, however, is an approximation mapping table between features in problem space and feature space. Hence, there is no exact mapping between problem spaces and feature space, which results in approximate mapping, leading to a slightly altered feature space than the original problem space. After adversarial examples are crafted on malware files, mapping features back to problem space also loses a few crafted perturbations due to a lack of absolute mapping. Therefore, the challenge of defining adversarial space and efficiently searching elements approaching the best replacement has always been there in the adversarial domain.

E. AUTOMATED ATTACKS

All of the discussed adversarial attacks require manual intervention at a few steps of the attack procedure. Human intervention makes the process time-consuming and impractical in many cases. In white-box attacks, the loss function of deep neural networks can be used to determine the most influential features, and the corresponding features can be automatically modified [44]. Current literature relies on human efforts for feature extraction, mapping to adversarial generation and functionality verification. Minimizing human effort while moving towards automated adversarial generation could be an interesting arena to work on in the future [168]. Novel research is needed to fully automate the adversarial attack ecosystem.

F. EXPLAINABLE ADVERSARIAL

Adversarial vulnerabilities have been considered unseen spots of machine learning models but current research work fails to assert concrete reasoning behind these unseen spots. Having no consensus behind such reasoning leaves explaining the existence of an adversarial example in an open research domain. Goodfellow et al. [22] first attributed vulnerability to the linear behaviour of the model in high dimensional space. However, there has been research that contradicts the accountability of adversarial behavior solely to the linearity of the model as highly non-linear models are also evaded successfully [169]. Explaining the adversarial phenomenon both in terms of models' functionality and features' contribution can pave the path for more robust adversarial attacks. Features can be assigned appropriate weights based on their contribution to alleviating the adversarial effect in the model. With the current state of the literature, explainable adversarial is still at an immature stage and requires concrete efforts from the community.

G. TRANSFERABLE ATTACKS

Transferability refers to the generalization property of the attack methods. A machine learning model with transferability property, trained on one particular dataset, can be generalized well for another dataset as well. Transferability is a common property for evasion attacks and is extensively exploited by black-box attacks. Untargeted attacks are found to be more transferable than targeted ones due to their generality [168]. Transferability can also take three different forms such as the same architecture with different data,

different architecture with the same application and different architecture with different data [32]. Although some studies have already been carried out on transferability, there is no universally accepted postulation. The ability to use the same data, model or algorithm to attack all available targets should be one of the goals of future research on adversarial attacks.

H. ATTACKING ADVERSARIAL DEFENSE

The influx of research on the adversarial domain during the last few years demonstrates the extent and importance of work in performing adversarial attacks. The profound activity has not been limited to the attack side only, but considering the threat posed to the entire machine learning family, researchers have been equally active on the defensive side as well. Performing adversarial attacks is turning out to be harder than ever, as many systems are designed robustly with adversarial defence in mind. Defensive approaches like adversarial training [170], defensive distillation [171] are proposed to stop adversarial attacks. Some recent techniques are hiding the gradients of the target model [42], which, if carried out successfully, can completely nullify the threat of gradient-based adversarial attacks. Hence, future adversarial attacks are required not only to bypass machine learning detection but also to overcome adversarial defences.

I. FUNCTIONALITY VERIFICATION

During adversarial evasion attacks, the modifications carried out in a malware file should not alter the functionality of the malware. The contents in the executable file could be very sensitive, and modification of a single byte can completely change the functionality of malware or even break the file. Most of the adversarial attacks have constrained themselves in perturbation type, volume and insertion techniques to preserve the functionality of the executable. Despite such gravity, most adversarial attacks can still not preserve the functionality of modified files. Moreover, limited mechanisms exist to verify the functionality of malware after perturbing the file. One of the available approaches is to run the malware file in an isolated environment like Cuckoo Sandbox [172]. However, running every individual malware in a sandbox is inefficient and unrealistic. Therefore, further research should be directed to develop tools that can automatically and efficiently verify the functionality of malware post perturbations.

J. BENIGN FILES ATTACK

Adversarial attacks are performed in malware files by inserting some non-malicious contents that do not tamper with any functionality other than classification decisions. Modifying malware files slowly has been a mainstream approach for adversarial. However, no limited or no existing research has studied the possibility of inserting malicious content into a benign file. This approach works in a reverse way than the established adversarial approaches. Inserting and hiding malicious payloads at different locations of files without affecting the classification decision is also a future research topic in adversarial and requires attention.

K. TARGETING UNEXPLORED ALGORITHMS

Most of the machine learning algorithms have already been victimized by adversarial attackers, including sophisticated deep neural networks. However, there are some deep neural networks that haven't yet been compromised by adversarial attackers such as Generative Adversarial Networks (GANs), Deep Reinforcement learning (DRL) and Variational Auto-Encoders (VAEs) [44]. These algorithms are in the development stage, which has capped the adversarial attempts against them to date. Differentiable neural computer [173] are only attacked once [174]. These new sets of algorithms are yet to be explored by adversarial attackers.

L. STANDARDIZING TESTBED AND METRICES

Adversarial attacks discussed in the survey are carried out in lab environments, taking numerous assumptions that may be unpragmatic for real-world challenges. Most of the works have assumed unlimited access to machine learning model, favourable datasets and weak classifiers to bolster their results. The current literature lacks standardized datasets and detection mechanisms to measure the exact performance of adversarial attacks. Hence, the attack testbed should be standardized to ensure the assessment uniformity across the research community.

The issue is not limited to the test environment but also evaluation metrics. More often than not, the performance of the attacker is reflected in terms of evasion accuracy inherited from machine learning models. However, accuracy only provides a small fraction of the attacker's performance in the adversarial domain. To provide the overall quality of attacks, metrics such as transferability, universality, and imperceptability need to be studied [167]. The metrics should be descriptive, fair, and complete to evaluate the quality of attacks performed across different environments. Some metrics should measure the degree of functionality preservation while manipulating the files. Metrics can also be designed to determine the sensitivity of file structure, helping attackers determine the level of cautiousness required during modification. These complete and fair metrics will not only help to understand and compare the adversarial quality but also enhance the performance of attacks.

M. ADVERSARIAL DEFENSE

The growth in adversarial attacks and novel approaches will also require developing advanced defense mechanisms. Although our survey is focused on adversarial evasion attacks, we believe it is important to briefly highlight future defence directions to present a comprehensive review paper. Among several defense techniques proposed, defensive distillation [58] and adversarial training [175], [176] are found to be the most effective. Collection of adversarial samples in large amounts to perform adversarial training

is a tedious task as neural networks require a massive volume of adversarial data [177]. An adversarial generation approach was proposed by Goodfellow et al. [112]; however, it is still very far away from being efficient and accurate enough to perform robust adversarial training. In addition, many defensive approaches that have been tested in an image domain [178], [179] are yet to be introduced for defence in malware adversarial domains. Recent research using robust machine learning architectures like Generative Adversarial Networks (GANs) [180] for defending against adversarial attacks requires more exploration to thwart or detect sophisticated evasion attacks. Overall, future research works on adversarial malware should be directed to build more robust, efficient, generalized and reliable defence mechanisms that can protect malware detection models against adversarial attacks.

VII. CONCLUSION

Machine learning and AI solutions are increasingly playing an important role in the cyber security domain. However, these data-driven systems can be easily manipulated, misled and evaded, which can have serious implications. Recent surges and research in adversarial attacks highlight the vulnerability of ML models, making them ineffective against even minor perturbations. In this paper, we provide a comprehensive survey of recent work that focuses on adversarial evasion attacks in the malware analysis domain. We have summarized the state-of-art adversarial attacks carried out against anti-malware engines in different file domains. The survey highlights the limitations of ML architectures against minute perturbations in the form of adversarial attacks. We taxonomize the adversarial evasion world of malware based on the attack domain and the approach taken to realize such attacks. The survey briefly discusses approaches taken by researchers, comparing them with other concomitant works. We conclude the survey by highlighting current challenges, open issues and future research directions in adversarial malware analysis. This work will provide a definitive guide to researchers and the community to understand the current scenarios of adversarial malware evasion attacks, prompting unexplored research territories in this highly dynamic and evolving domain.

REFERENCES

[1] M. Abdelsalam, R. Krishnan, Y. Huang, and R. Sandhu, "Malware detection in cloud infrastructures using convolutional neural networks," in *Proc. IEEE 11th Int. Conf. Cloud Comput. (CLOUD)*, Jul. 2018, pp. 162–169.

[2] A. McDole, M. Gupta, M. Abdelsalam, S. Mittal, and M. Alazab, "Deep learning techniques for behavioral malware analysis in cloud IaaS," in *Malware Analysis Using Artificial Intelligence and Deep Learning*. Cham, Switzerland: Springer, 2021, pp. 269–285.

[3] A. McDole, M. Abdelsalam, M. Gupta, and S. Mittal, "Analyzing CNN based behavioural malware detection techniques on cloud IaaS," in *Proc. Int. Conf. Cloud Comput.* Cham, Switzerland: Springer, Jan. 2020, pp. 64–79.

[4] J. C. Kimmel, A. D. Mcdole, M. Abdelsalam, M. Gupta, and R. Sandhu, "Recurrent neural networks based online behavioural malware detection techniques for cloud infrastructure," *IEEE Access*, vol. 9, pp. 68066–68080, 2021.

[5] J. C. Kimmell, M. Abdelsalam, and M. Gupta, "Analyzing machine learning approaches for online malware detection in cloud," in *Proc. IEEE Int. Conf. Smart Comput. (SMARTCOMP)*, Aug. 2021, pp. 189–196.

[6] T. S. Guzella and W. M. Caminhas, "A review of machine learning approaches to spam filtering," *Expert Syst. Appl.*, vol. 36, no. 7, pp. 10206–10222, Sep. 2009.

[7] J. O. Awoyemi, A. O. Adetunmbi, and S. A. Oluwadare, "Credit card fraud detection using machine learning techniques: A comparative analysis," in *Proc. Int. Conf. Comput. Netw. Informat. (ICCNI)*, Oct. 2017, pp. 1–9.

[8] A. Dhakal, C. McKay, J. J. Tanner, and J. Cheng, "Artificial intelligence in the prediction of protein–ligand interactions: Recent advances and future directions," *Briefings Bioinf.*, vol. 23, no. 1, Jan. 2022, Art. no. bbab476.

[9] M. Gupta, F. Patwa, and R. Sandhu, "POSTER: Access control model for the Hadoop ecosystem," in *Proc. 22nd ACM Symp. Access Control Models Technol.*, Jun. 2017, pp. 125–127.

[10] M. Gupta and R. Sandhu, "Towards activity-centric access control for smart collaborative ecosystems," in *Proc. 26th ACM Symp. Access Control Models Technol.*, Jun. 2021, pp. 155–164.

[11] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," *Inf. Sci.*, vols. 460–461, pp. 83–102, Sep. 2018.

[12] G. Phillips. (Aug. 2018). *These 4 Antivirus Tools Are Using AI to Protect Your System*. [Online]. Available: https://www.makeuseof.com/tag/artificial-intelligence-antivirus-tools/

[13] S. Morrison. (May 2021). *How a Major Oil Pipeline Got Held for Ransom*. [Online]. Available: https://www.vox.com/recode/22428774/ransomeware-pipeline-colonial-darkside-gas-prices

[14] J. Crawley. (2021). *Electronics Retailer MediaMarkt Hit by Ransomware Demand for $50M Bitcoin Payment: Report*. [Online]. Available: https://www.yahoo.com/lifestyle/electronics-retailer-mediamarkt-hit-ransomware-151038565.html

[15] L. Huang, A. D. Joseph, B. Nelson, B. I. P. Rubinstein, and J. D. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, Oct. 2011, pp. 43–58.

[16] A. Shafahi, W. R. Huang, M. Najibi, O. Suciu, C. Studer, T. Dumitras, and T. Goldstein, "Poison frogs! Targeted clean-label poisoning attacks on neural networks," 2018, *arXiv:1804.00792*.

[17] X. Liu, S. Si, X. Zhu, Y. Li, and C.-J. Hsieh, "A unified framework for data poisoning attack to graph-based semi-supervised learning," 2019, *arXiv:1910.14147*.

[18] K. Aryal, M. Gupta, and M. Abdelsalam, "Analysis of label-flip poisoning attack on machine learning based malware detector," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2022, pp. 4236–4245.

[19] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.

[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2015, pp. 1–9.

[21] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2009, pp. 248–255.

[22] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial examples in the physical world," 2016, *arXiv:1607.02533*.

[24] T. Huddleston Jr., (Apr. 2019). *These Chinese Hackers Tricked Tesla's Autopilot Into Suddenly Switching Lanes*. [Online]. Available: https://www.cnbc.com/2019/04/03/chinese-hackers-tricked-teslas-autopilot-into-switching-lanes.html

[25] H. S. Anderson, A. Kharkar, B. Filar, and P. Roth, "Evading machine learning malware detection," in *Proc. Black Hat*, 2017, pp. 1–6.

[26] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial malware binaries: Evading deep learning for malware detection in executables," in *Proc. 26th Eur. Signal Process. Conf. (EUSIPCO)*, Sep. 2018, pp. 533–537.

[27] L. Demetrio, B. Biggio, G. Lagorio, F. Roli, and A. Armando, "Explaining vulnerabilities of deep learning to adversarial malware binaries," 2019, *arXiv:1901.03583*.

[28] O. Suciu, S. E. Coull, and J. Johns, "Exploring adversarial examples in malware detection," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, May 2019, pp. 8–14.
[29] M. Barreno, B. Nelson, A. D. Joseph, and J. D. Tygar, "The security of machine learning," *Mach. Learn.*, vol. 81, no. 2, pp. 121–148, May 2010.
[30] J. Gardiner and S. Nagaraja, "On the security of machine learning in malware C&C detection: A survey," *ACM Comput. Surv.*, vol. 49, no. 3, pp. 1–39, 2016.
[31] A. Kumar and S. Mehta, "A survey on resilient machine learning," 2017, *arXiv:1707.03184*.
[32] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[33] A. Chakraborty, M. Alam, V. Dey, A. Chattopadhyay, and D. Mukhopadhyay, "Adversarial attacks and defences: A survey," 2018, *arXiv:1810.00069*.
[34] N. Akhtar and A. Mian, "Threat of adversarial attacks on deep learning in computer vision: A survey," *IEEE Access*, vol. 6, pp. 14410–14430, 2018.
[35] V. Duddu, "A survey of adversarial machine learning in cyber warfare," *Defence Sci. J.*, vol. 68, no. 4, pp. 356–366, Jun. 2018.
[36] G. Li, P. Zhu, J. Li, Z. Yang, N. Cao, and Z. Chen, "Security matters: A survey on adversarial machine learning," 2018, *arXiv:1810.07339*.
[37] Q. Liu, P. Li, W. Zhao, W. Cai, S. Yu, and V. C. M. Leung, "A survey on security threats and defensive techniques of machine learning: A data driven view," *IEEE Access*, vol. 6, pp. 12103–12117, 2018.
[38] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018, doi: 10.1016/j.patcog.2018.07.023.
[39] L. Sun, M. Tan, and Z. Zhou, "A survey of practical adversarial example attacks," *Cybersecurity*, vol. 1, no. 1, pp. 1–9, Dec. 2018.
[40] N. Pitropakis, E. Panaousis, T. Giannetsos, E. Anastasiadis, and G. Loukas, "A taxonomy and survey of attacks against machine learning," *Comput. Sci. Rev.*, vol. 34, Nov. 2019, Art. no. 100199.
[41] X. Wang, J. Li, X. Kuang, Y.-A. Tan, and J. Li, "The security of machine learning in an adversarial setting: A survey," *J. Parallel Distrib. Comput.*, vol. 130, pp. 12–23, Aug. 2019.
[42] S. Qiu, Q. Liu, S. Zhou, and C. Wu, "Review of artificial intelligence adversarial attack and defense technologies," *Appl. Sci.*, vol. 9, no. 5, p. 909, Mar. 2019.
[43] H. Xu, Y. Ma, H.-C. Liu, D. Deb, H. Liu, J.-L. Tang, and A. K. Jain, "Adversarial attacks and defenses in images, graphs and text: A review," *Int. J. Autom. Comput.*, vol. 17, no. 2, pp. 151–178, Apr. 2020.
[44] W. E. Zhang, Q. Z. Sheng, A. Alhazmi, and C. Li, "Adversarial attacks on deep-learning models in natural language processing: A survey," *ACM Trans. Intell. Syst. Technol.*, vol. 11, no. 3, pp. 1–41, Jun. 2020.
[45] N. Martins, J. M. Cruz, T. Cruz, and P. H. Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35403–35419, 2020.
[46] I. Moisejevs, "Adversarial attacks and defenses in malware classification: A survey," *Int. J. Artif. Intell. Expert Syst.*, vol. 8, 2019.
[47] O. Ibitoye, R. Abou-Khamis, M. E. Shehaby, A. Matrawy, and M. O. Shafiq, "The threat of adversarial attacks on machine learning in network security—A survey," 2019, *arXiv:1911.02621*.
[48] D. Li, Q. Li, Y. Ye, and S. Xu, "Arms race in adversarial malware detection: A survey," *ACM Comput. Surv.*, vol. 55, no. 1, pp. 1–35, Jan. 2023.
[49] X. Ling, L. Wu, J. Zhang, Z. Qu, W. Deng, X. Chen, Y. Qian, C. Wu, S. Ji, T. Luo, J. Wu, and Y. Wu, "Adversarial attacks against windows PE malware detection: A survey of the state-of-the-art," *Comput. Secur.*, vol. 128, May 2023, Art. no. 103134.
[50] K. Ren, T. Zheng, Z. Qin, and X. Liu, "Adversarial attacks and defenses in deep learning," *Engineering*, vol. 6, no. 3, pp. 346–360, Mar. 2020.
[51] W. Hu and Y. Tan, "Black-box attacks against RNN based malware detection algorithms," in *Proc. 32nd AAAI Conf. Artif. Intell. Workshops*, Jun. 2018, pp. 1–7.
[52] B. S. Vivek, K. R. Mopuri, and R. V. Babu, "Gray-box adversarial training," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, Aug. 2018, pp. 203–218.
[53] B. Biggio, G. Fumera, and F. Roli, "Security evaluation of pattern classifiers under attack," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 984–996, Apr. 2014, doi: 10.1109/TKDE.2013.57.
[54] Y. Wang and K. Chaudhuri, "Data poisoning attacks against online learning," 2018, *arXiv:1808.08994*.
[55] X. Chen, C. Liu, B. Li, K. Lu, and D. Song, "Targeted backdoor attacks on deep learning systems using data poisoning," 2017, *arXiv:1712.05526*.
[56] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P)*, Mar. 2016, pp. 372–387.
[57] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2017, pp. 39–57.
[58] N. Papernot, P. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2016, pp. 582–597.
[59] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "DeepFool: A simple and accurate method to fool deep neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2574–2582.
[60] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, Nov. 2017, pp. 15–26.
[61] J. Su, D. V. Vargas, and K. Sakurai, "One pixel attack for fooling deep neural networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 5, pp. 828–841, Oct. 2019.
[62] P. Civicioglu and E. Besdok, "A conceptual comparison of the cuckoo-search, particle swarm optimization, differential evolution and artificial bee colony algorithms," *Artif. Intell. Rev.*, vol. 39, no. 4, pp. 315–346, Apr. 2013.
[63] S.-M. Moosavi-Dezfooli, A. Fawzi, O. Fawzi, and P. Frossard, "Universal adversarial perturbations," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jul. 2017, pp. 1765–1773.
[64] F. Croce and M. Hein, "Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2020, pp. 2206–2216.
[65] W. Brendel, J. Rauber, and M. Bethge, "Decision-based adversarial attacks: Reliable attacks against black-box machine learning models," 2017, *arXiv:1712.04248*.
[66] M. Pietrek. (2019). *Inside Windows: Win32 Portable Executable File Format in Detail*. [Online]. Available: https://docs.microsoft.com/en-us/archive/msdn-magazine/2002/february/inside-windows-win32-portable-executable-file-format-in-detail
[67] J. Plachy. (2018). *Portable Executable File Format*. [Online]. Available: https://blog.kowalczyk.info/articles/pefileformat.html
[68] (2007). *Pe File Structure*. [Online]. Available: https://ivanlef0u.fr/repo/madchat/vxdevl/papers/winsys/pefile/pefile.htm
[69] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Deceiving end-to-end deep learning malware detectors using adversarial examples," 2018, *arXiv:1802.04528*.
[70] J. Yuste, E. G. Pardo, and J. Tapiador, "Optimization of code caves in malware binaries to evade machine learning detectors," *Comput. Secur.*, vol. 116, May 2022, Art. no. 102643.
[71] K. Aryal, M. Gupta, M. Abdelsalam, and M. Saleh, "Intra-section code cave injection for adversarial evasion attacks on windows PE malware file," 2024, *arXiv:2403.06428*.
[72] H. Li, S. Zhou, W. Yuan, J. Li, and H. Leung, "Adversarial-example attacks toward Android malware detection system," *IEEE Syst. J.*, vol. 14, no. 1, pp. 653–656, Mar. 2020.
[73] I. Rosenberg, A. Shabtai, L. Rokach, and Y. Elovici, "Generic black-box end-to-end attack against state of the art API call based malware classifiers," in *Proc. Int. Symp. Res. Attacks, Intrusions, Defenses*. Cham, Switzerland: Springer, Jan. 2018, pp. 490–510.
[74] M. Shahpasand, L. Hamey, D. Vatsalan, and M. Xue, "Adversarial attacks on mobile malware detection," in *Proc. IEEE 1st Int. Workshop Artif. Intell. Mobile (AI4Mobile)*, Feb. 2019, pp. 17–20.
[75] F. Pierazzi, F. Pendlebury, J. Cortellazzi, and L. Cavallaro, "Intriguing properties of adversarial ML attacks in the problem space," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2020, pp. 1332–1349.
[76] Android. (2022). *App Manifest Overview : Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/manifest/manifest-intro
[77] A. Asokan. (Sep. 2016). *APK File Contents—In-Depth Explanation*. [Online]. Available: https://ajinasokan.com/posts/apk-file-postmortem/
[78] W. Kalicinski. (May 2016). *SmallerAPK, Part 1: Anatomy of an APK*. [Online]. Available: https://medium.com/androiddevelopers/smallerapk-part-1-anatomy-of-an-apk-da83c25e7003

[79] B. Biggio, I. Corona, D. Maiorca, B. Nelson, N. Šrndić, P. Laskov, G. Giacinto, and F. Roli, "Evasion attacks against machine learning at test time," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases*. Cham, Switzerland: Springer, Jan. 2013, pp. 387–402.

[80] S. Liu. (Jul. 2021). *Desktop OS Market Share*. [Online]. Available: https://www.statista.com/statistics/218089/global-market-share-of-windows-7/

[81] M. D. Boer. (2019). *AI as a Target and Tool: An Attacker's Perspective on ML*. [Online]. Available: https://www.gartner.com/en/documents/3939991

[82] E. Raff, J. Barker, J. Sylvester, R. Brandon, B. Catanzaro, and C. K. Nicholas, "Malware detection by eating a whole EXE," in *Proc. 32nd AAAI Conf. Artif. Intell. Workshops*, 2018, pp. 1–9.

[83] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Adversarial examples on discrete sequences for beating whole-binary malware detection," 2018, *arXiv:1802.04528*.

[84] H. S. Anderson and P. Roth, "EMBER: An open dataset for training static PE malware machine learning models," 2018, *arXiv:1804.04637*.

[85] B. Chen, Z. Ren, C. Yu, I. Hussain, and J. Liu, "Adversarial examples for CNN-based malware detectors," *IEEE Access*, vol. 7, pp. 54360–54371, 2019.

[86] Y. Jakhotiya, H. Patil, J. Rawlani, and S. Mane, "Adversarial attacks on transformers-based malware detectors," in *Proc. NeurIPS ML Saf. Workshop*, Jan. 2022, pp. 1–12.

[87] D. Zhan, Y. Duan, Y. Hu, W. Li, S. Guo, and Z. Pan, "MalPatch: Evading DNN-based malware detection with adversarial patches," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 1183–1198, 2024.

[88] M. Sundararajan, A. Taly, and Q. Yan, "Axiomatic attribution for deep networks," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2017, pp. 3319–3328.

[89] M. Kozák and M. Jureček, "Combining generators of adversarial malware examples to increase evasion rate," 2023, *arXiv:2304.07360*.

[90] W. Yang and F. Yin, "A multi-strategy adversarial attack method for deep learning based malware detectors," in *Proc. 7th Int. Conf. Cryptography, Secur. Privacy (CSP)*, Apr. 2023, pp. 66–70.

[91] K. Li, F. Zhang, and W. Guo, "FGAM: Fast adversarial malware generation method based on gradient sign," 2023, *arXiv:2305.12770*.

[92] D. Park, H. Khan, and B. Yener, "Generation & evaluation of adversarial examples for malware obfuscation," in *Proc. 18th IEEE Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2019, pp. 1283–1290.

[93] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2016, pp. 2818–2826.

[94] W. Song, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin, "Automatic generation of adversarial examples for interpreting malware classifiers," 2020, *arXiv:2003.03100*.

[95] P. Junod, J. Rinaldini, J. Wehrli, and J. Michielin, "Obfuscator-LLVM—Software protection for the masses," in *Proc. IEEE/ACM 1st Int. Workshop Softw. Protection*, May 2015, pp. 3–9.

[96] X. Meng and B. P. Miller, "Binary code is not easy," in *Proc. 25th Int. Symp. Softw. Test. Anal.*, Jul. 2016, pp. 24–35.

[97] H. S. Anderson, A. Kharkar, B. Filar, D. Evans, and P. Roth, "Learning to evade static PE machine learning malware models via reinforcement learning," 2018, *arXiv:1801.08917*.

[98] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, and H. Huang, "Evading anti-malware engines with deep reinforcement learning," *IEEE Access*, vol. 7, pp. 48867–48879, 2019.

[99] J. Chen, J. Jiang, R. Li, and Y. Dou, "Generating adversarial examples for static PE malware detector based on deep reinforcement learning," *J. Phys., Conf. Ser.*, vol. 1575, no. 1, Jun. 2020, Art. no. 012011.

[100] Y. Fang, Y. Zeng, B. Li, L. Liu, and L. Zhang, "DeepDetectNet vs RLAttackNet: An adversarial method to improve deep learning-based static malware detection model," *PLoS ONE*, vol. 15, no. 4, Apr. 2020, Art. no. e0231626.

[101] T. Quertier, B. Marais, S. Morucci, and B. Fournel, "MERLIN—Malware evasion with reinforcement LearnINg," 2022, *arXiv:2203.12980*.

[102] W. Song, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin, "MAB-Malware: A reinforcement learning framework for blackbox generation of adversarial malware," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, 2022, pp. 990–1003.

[103] M. Rigaki and S. García, "The power of MEME: Adversarial malware creation with model-based reinforcement learning," in *Proc. Eur. Symp. Res. Comput. Secur.*. Cham, Switzerland: Springer, Jan. 2023, pp. 44–64.

[104] D. Zhan, Y. Zhang, L. Zhu, J. Chen, S. Xia, S. Guo, and Z. Pan, "Enhancing reinforcement learning based adversarial malware generation to evade static detection," *Alexandria Eng. J.*, vol. 98, pp. 32–43, Jul. 2024.

[105] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," 2017, *arXiv:1702.05983*.

[106] M. Kawai, K. Ota, and M. Dong, "Improved MalGAN: Avoiding malware detector by leaning cleanware features," in *Proc. Int. Conf. Artif. Intell. Inf. Commun. (ICAIIC)*, Feb. 2019, pp. 40–45.

[107] R. L. Castro, C. Schmitt, and G. D. Rodosek, "Poster: Training GANs to generate adversarial examples against malware classification," in *Proc. IEEE Secur. Privacy*, Jul. 2019, pp. 1–2.

[108] J. Yuan, S. Zhou, L. Lin, F. Wang, and J. Cui, "Black-box adversarial attacks against deep learning based malware binaries detection with GAN," in *Proc. ECAI*. Amsterdam, The Netherlands: IOS Press, Jan. 2020, pp. 2536–2542.

[109] E. Zhu, J. Zhang, J. Yan, K. Chen, and C. Gao, "N-gram MalGAN: Evading machine learning detection via feature N-gram," *Digit. Commun. Netw.*, vol. 8, no. 4, pp. 485–491, Aug. 2022.

[110] F. Zhong, X. Cheng, D. Yu, B. Gong, S. Song, and J. Yu, "MalFox: Camouflaged adversarial malware example generation based on conv-GANs against black-box detectors," *IEEE Trans. Comput.*, vol. 73, no. 4, pp. 980–993, Apr. 2024.

[111] D. Gibert, J. Planes, Q. Le, and G. Zizzo, "A wolf in sheep's clothing: Query-free evasion attacks against machine learning-based malware detectors with generative adversarial networks," in *Proc. IEEE Eur. Symp. Secur. Privacy Workshops (EuroS&PW)*, Jul. 2023, pp. 415–426.

[112] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 27, 2014, pp. 1–11.

[113] D. Devadiga, G. Jin, B. Potdar, H. Koo, A. Han, A. Shringi, A. Singh, K. Chaudhari, and S. Kumar, "GLEAM: GAN and LLM for evasive adversarial malware," in *Proc. 14th Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2023, pp. 53–58.

[114] R. Pascanu, J. W. Stokes, H. Sanossian, M. Marinescu, and A. Thomas, "Malware classification with recurrent networks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Apr. 2015, pp. 1916–1920.

[115] N. Papernot, P. McDaniel, A. Swami, and R. Harang, "Crafting adversarial input sequences for recurrent neural networks," in *Proc. IEEE Mil. Commun. Conf. (MILCOM)*, Nov. 2016, pp. 49–54.

[116] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with Gumbel-Softmax," 2016, *arXiv:1611.01144*.

[117] I. Rosenberg, S. Meir, J. Berrebi, I. Gordon, G. Sicard, and E. O. David, "Generating end-to-end adversarial examples for malware classifiers using explainability," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–10.

[118] K. Aryal, M. Gupta, M. Abdelsalam, and M. Saleh, "Explainability guided adversarial evasion attacks on malware detectors," 2024, *arXiv:2405.01728*.

[119] X. Liu, J. Zhang, Y. Lin, and H. Li, "ATMPA: Attacking machine learning-based malware visualization detection methods via adversarial examples," in *Proc. IEEE/ACM 27th Int. Symp. Quality Service (IWQoS)*, Jun. 2019, pp. 1–10.

[120] A. Khormali, A. Abusnaina, S. Chen, D. Nyang, and A. Mohaisen, "COPYCAT: Practical adversarial attacks on visualization-based malware detection," 2019, *arXiv:1909.09735*.

[121] H. Benkraouda, J. Qian, H. Q. Tran, and B. Kaplan, "Attacks on visualization-based malware detection: Balancing effectiveness and executability," in *Proc. Int. Workshop Deployable Mach. Learn. Secur. Defense*. Cham, Switzerland: Springer, Jan. 2021, pp. 107–131.

[122] J. L. Hu, M. Ebrahimi, and H. Chen, "Single-shot black-box adversarial attacks against malware detectors: A causal language model approach," in *Proc. IEEE Int. Conf. Intell. Secur. Informat. (ISI)*, Nov. 2021, pp. 1–6.

[123] S. Bach, A. Binder, G. Montavon, F. Klauschen, K.-R. Müller, and W. Samek, "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation," *PLoS ONE*, vol. 10, no. 7, Jul. 2015, Art. no. e0130140.

[124] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2017, pp. 3145–3153.

[125] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, Jan. 2017, pp. 4768–4777.
[126] K. S. Han, J. H. Lim, B. Kang, and E. G. Im, "Malware analysis using visualized images and entropy graphs," *Int. J. Inf. Secur.*, vol. 14, no. 1, pp. 1–14, Feb. 2015.
[127] K. Kancherla and S. Mukkamala, "Image visualization based malware detection," in *Proc. IEEE Symp. Comput. Intell. Cyber Secur. (CICS)*, Apr. 2013, pp. 40–44.
[128] Y. Dong, F. Liao, T. Pang, H. Su, J. Zhu, X. Hu, and J. Li, "Boosting adversarial attacks with momentum," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9185–9193.
[129] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.
[130] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. Eur. Symp. Res. Comput. Secur.* Cham, Switzerland: Springer, 2017, pp. 62–79.
[131] W. Yang, D. Kong, T. Xie, and C. A. Gunter, "Malware detection in adversarial settings: Exploiting feature evolutions and confusions in Android apps," in *Proc. 33rd Annu. Comput. Secur. Appl. Conf.*, Dec. 2017, pp. 288–302.
[132] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang, and M. Guizani, "Adversarial samples on Android malware detection systems for IoT systems," *Sensors*, vol. 19, no. 4, p. 974, Feb. 2019.
[133] H. Bostani and V. Moonsamy, "EvadeDroid: A practical evasion attack on machine learning for black-box Android malware detection," *Comput. Secur.*, vol. 139, Apr. 2024, Art. no. 103676.
[134] D. Arp, M. Spreitzenbarth, M. Hübner, H. Gascon, and K. Rieck, "DREBIN: Effective and explainable detection of Android malware in your pocket," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2014, pp. 1–15.
[135] A. Demontis, M. Melis, B. Biggio, D. Maiorca, D. Arp, K. Rieck, I. Corona, G. Giacinto, and F. Roli, "Yes, machine learning can be more secure! A case study on Android malware detection," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 4, pp. 711–724, Jul. 2019.
[136] E. Mariconti, L. Onwuzurike, P. Andriotis, E. De Cristofaro, G. Ross, and G. Stringhini, "MaMaDroid: Detecting Android malware by building Markov chains of behavioral models," 2016, *arXiv:1612.04433*.
[137] K. Allix, T. F. Bissyandé, J. Klein, and Y. L. Traon, "AndroZoo: Collecting millions of Android apps for the research community," in *Proc. IEEE/ACM 13th Work. Conf. Mining Softw. Repositories (MSR)*, May 2016, pp. 468–471.
[138] D. Curry. (Jun. 2022). *Android Statistics (2021)*. [Online]. Available: https://www.businessofapps.com/data/android-statistics/
[139] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, *arXiv:1606.04435*.
[140] E. T. Barr, M. Harman, Y. Jia, A. Marginean, and J. Petke, "Automated software transplantation," in *Proc. Int. Symp. Softw. Test. Anal.*, Jul. 2015, pp. 257–269.
[141] C. J. Bult, "Bioinformatics: A practical guide to the analysis of genes and proteins," *Science*, vol. 282, no. 5389, pp. 635–636, 1998.
[142] A. Developers. (2022). *Permissions on Android : Android Developers*. [Online]. Available: https://developer.android.com/guide/topics/permissions/
[143] D. Maiorca, I. Corona, and G. Giacinto, "Looking at the bag is not enough to find the bomb: An evasion of structural methods for malicious PDF files detection," in *Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur.*, May 2013, pp. 119–130.
[144] B. Biggio, I. Corona, B. Nelson, B. I. Rubinstein, D. Maiorca, G. Fumera, G. Giacinto, and F. Roli, "Security evaluation of support vector machines in adversarial environments," in *Support Vector Machines Applications*. Cham, Switzerland: Springer, 2014, pp. 105–153.
[145] N. Šrndic and P. Laskov, "Practical evasion of a learning-based classifier: A case study," in *Proc. IEEE Symp. Secur. Privacy*, May 2014, pp. 197–211.
[146] C. Carmony, M. Zhang, X. Hu, A. V. Bhaskar, and H. Yin, "Extract me if you can: Abusing PDF parsers in malware detectors," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–16.
[147] W. Xu, Y. Qi, and D. Evans, "Automatically evading classifiers: A case study on PDF malware classifiers," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, 2016, pp. 1–17.
[148] C. Smutz and A. Stavrou, "Malicious PDF detection using metadata and structural features," in *Proc. 28th Annu. Comput. Secur. Appl. Conf.*, Dec. 2012, pp. 239–248.
[149] N. Šrndić and P. Laskov, "Detection of malicious PDF files based on hierarchical document structure," in *Proc. Annu. Netw. Distrib. Syst. Secur. Symp.*, Jan. 2013, pp. 1–16.
[150] P. Laskov and N. Šrndić, "Static detection of malicious JavaScript-bearing PDF documents," in *Proc. 27th Annu. Comput. Secur. Appl. Conf.*, Dec. 2011, pp. 373–382.
[151] D. Maiorca, G. Giacinto, and I. Corona, "A pattern recognition system for malicious PDF files detection," in *Proc. Int. Workshop Mach. Learn. Data Mining Pattern Recognit.* Cham, Switzerland: Springer, Jan. 2012, pp. 510–524.
[152] J. M. Esparza, "PEEPDF–PDF analysis tool," Tech. Rep., 2015.
[153] P. Golland, "Discriminative direction for kernel classifiers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 14, 2001, pp. 1–9.
[154] S. Jana and V. Shmatikov, "Abusing file processing in malware detectors for fun andprofit," in *Proc. IEEE Symp. Secur. Privacy*, May 2012, pp. 80–94.
[155] D. Liu, H. Wang, and A. Stavrou, "Detecting malicious Javascript in PDF through document instrumentation," in *Proc. 44th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw.*, Jun. 2014, pp. 100–111.
[156] K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "RHMD: Evasion-resilient hardware malware detectors," in *Proc. 50th Annu. IEEE/ACM Int. Symp. Microarchitecture (MICRO)*, Oct. 2017, pp. 315–327.
[157] S. M. P. Dinakarrao, S. Amberkar, S. Bhat, A. Dhavlle, H. Sayadi, A. Sasan, H. Homayoun, and S. Rafatirad, "Adversarial attack on microarchitectural events based malware detectors," in *Proc. 56th ACM/IEEE Design Autom. Conf. (DAC)*, Jun. 2019, pp. 1–6.
[158] K. Nozawa, K. Hasegawa, S. Hidano, S. Kiyomoto, K. Hashimoto, and N. Togawa, "Generating adversarial examples for hardware-trojan detection at gate-level netlists," *J. Inf. Process.*, vol. 29, pp. 236–246, Jan. 2021.
[159] Y. Vorobeychik and B. Li, "Optimal randomized classification in adversarial settings," in *Proc. Int. Conf. Auto. Agents Multi-Agent Syst.*, May 2014, pp. 485–492.
[160] K. Wang, J. J. Parekh, and S. J. Stolfo, "Anagram: A content anomaly detector resistant to mimicry attack," in *Proc. Int. Workshop Recent Adv. Intrusion Detection*. Cham, Switzerland: Springer, Jan. 2006, pp. 226–248.
[161] M. S. Islam, K. N. Khasawneh, N. Abu-Ghazaleh, D. Ponomarev, and L. Yu, "Efficient hardware malware detectors that are resilient to adversarial evasion," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2872–2887, Nov. 2022.
[162] C.-K. Luk, R. Cohn, R. Muth, H. Patil, A. Klauser, G. Lowney, S. Wallace, V. J. Reddi, and K. Hazelwood, "Pin: Building customized program analysis tools with dynamic instrumentation," *ACM SIGPLAN Notices*, vol. 40, no. 6, pp. 190–200, Jun. 2005.
[163] N. Patel, A. Sasan, and H. Homayoun, "Analyzing hardware based malware detectors," in *Proc. 54th ACM/EDAC/IEEE Design Autom. Conf. (DAC)*, Jun. 2017, pp. 1–6.
[164] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, and A. Mohaisen, "Adversarial learning attacks on graph-based IoT malware detection systems," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1296–1305.
[165] Y. Gong, B. Li, C. Poellabauer, and Y. Shi, "Real-time adversarial attacks," 2019, *arXiv:1905.13399*.
[166] Y. Xie, C. Shi, Z. Li, J. Liu, Y. Chen, and B. Yuan, "Real-time, universal, and robust adversarial attacks against speaker recognition systems," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2020, pp. 1738–1742.
[167] I. Alsmadi, K. Ahmad, M. Nazzal, F. Alam, A. Al-Fuqaha, A. Khreishah, and A. Algosaibi, "Adversarial attacks and defenses for social network text processing applications: Techniques, challenges and future research directions," 2021, *arXiv:2110.13980*.
[168] B. Liang, H. Li, M. Su, P. Bian, X. Li, and W. Shi, "Deep text classification can be fooled," 2017, *arXiv:1704.08006*.
[169] H. Li, Y. Fan, F. Ganz, A. Yezzi, and P. Barnaghi, "Verifying the causes of adversarial examples," in *Proc. 25th Int. Conf. Pattern Recognit. (ICPR)*, Jan. 2021, pp. 6750–6757.

[170] A. Shafahi, M. Najibi, A. Ghiasi, Z. Xu, J. Dickerson, C. Studer, L. S. Davis, G. Taylor, and T. Goldstein, "Adversarial training for free!" 2019, *arXiv:1904.12843*.
[171] N. Papernot and P. McDaniel, "Extending defensive distillation," 2017, *arXiv:1705.05264*.
[172] S. Jamalpur, Y. S. Navya, P. Raja, G. Tagore, and G. R. K. Rao, "Dynamic malware analysis using cuckoo sandbox," in *Proc. 2nd Int. Conf. Inventive Commun. Comput. Technol. (ICICCT)*, Apr. 2018, pp. 1056–1060.
[173] G. Wayne and A. Graves. (2016). *Differentiable Neural Computers*. [Online]. Available: https://deepmind.com/blog/article/differentiable-neural-computers
[174] A. Chan, L. Ma, F. Juefei-Xu, X. Xie, Y. Liu, and Y. S. Ong, "Metamorphic relation based adversarial attacks on differentiable neural computer," 2018, *arXiv:1809.02444*.
[175] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," *STAT*, vol. 1050, p. 30, May 2017.
[176] A. Shafahi, M. Najibi, Z. Xu, J. W. T. Dickerson, L. S. Davis, and T. Goldstein, "Universal adversarial training," in *Proc. AAAI Conf. Artif. Intell.*, Apr. 2020, vol. 34, no. 4, pp. 5636–5643.
[177] Y. Jang, T. Zhao, S. Hong, and H. Lee, "Adversarial defense via learning to generate diverse attacks," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, Oct. 2019, pp. 2740–2749.
[178] F. Liao, M. Liang, Y. Dong, T. Pang, X. Hu, and J. Zhu, "Defense against adversarial attacks using high-level representation guided denoiser," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 1778–1787.
[179] D. Meng and H. Chen, "MagNet: A two-pronged defense against adversarial examples," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, pp. 1–8.
[180] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," 2018, *arXiv:1805.06605*.

KSHITIZ ARYAL (Graduate Student Member, IEEE) received the B.S. degree in ECE from Tribhuvan University, Nepal, and the M.S. degree in computer science from Tennessee Technological University, Cookeville, TN, USA, where he is currently pursuing the Ph.D. degree with the Department of Computer Science. He is also a Graduate Research Assistant with the Department of Computer Science, Tennessee Technological University. His current research interests include adversarial attacks/defense, malware analysis, AI security, security of AI, explainable AI, and data science.

MAANAK GUPTA (Senior Member, IEEE) received the B.Tech. degree in computer science and engineering from India, the M.S. degree in information systems from Northeastern University, Boston, and the M.S. and Ph.D. degrees in computer science from The University of Texas at San Antonio (UTSA). He was a Postdoctoral Fellow with the Institute for Cyber Security (ICS), UTSA. He is currently an Associate Professor in computer science with Tennessee Technological University, Cookeville, TN, USA. He has worked on developing novel security mechanisms, models, and architectures for next-generation smart cars, intelligent transportation systems, and smart farming. His research has been funded by U.S. National Science Foundation (NSF), NASA, and U.S. Department of Defense (DoD), among others. His research interests include security and privacy in cyberspace, focused on studying foundational aspects of access control, malware analysis, AI and machine learning-assisted cyber security, and their applications in technologies, including cyber-physical systems, cloud computing, the IoT, and big data. He was awarded the 2019 Computer Science Outstanding Doctoral Dissertation Research Award from UT San Antonio.

MAHMOUD ABDELSALAM (Member, IEEE) received the B.Sc. degree from the Arab Academy for Science and Technology and Maritime Transportation (AASTMT), in 2013, and the M.Sc. and Ph.D. degrees from the University of Texas at San Antonio (UTSA), in 2017 and 2018, respectively. He was a Postdoctoral Research Fellow with the Institute for Cyber Security (ICS), UTSA, and an Assistant Professor with the Department of Computer Science, Manhattan College. He is currently an Assistant Professor with the Department of Computer Science, North Carolina A&T State University. His research interests include computer systems security, anomaly and malware detection, cloud computing security and monitoring, cyber-physical systems security, and applied machine learning.

PRADIP KUNWAR (Graduate Student Member, IEEE) received the B.Tech. degree in electronics and communication from NIT Rourkela, India. He is currently pursuing the Ph.D. degree in computer science with Tennessee Technological University, Cookeville, TN, USA. His current research interests include AI-based malware analysis, malware generation, security of LLMs and AI models, and explainable AI. He is interested in researching the underlying vulnerabilities of AI systems and making them more robust against adversarial attacks.

BHAVANI THURAISINGHAM (Fellow, IEEE) was educated from the University of Bristol and the University of Wales, U.K. She is currently a Professor of computer science and the Director of the Cyber Security Research Center, The University of Texas at Dallas (UTD). Prior to joining UTD, she was the Program Director for three years at the National Science Foundation (NSF), Arlington. She has also worked for the Computer Industry, Minneapolis, MN, USA, for more than five years and has served as an Adjunct Professor of computer science and a member of the Graduate Faculty at the University of Minnesota and later taught at Boston University. She has published more than 300 research papers, including more than 90 journal articles, and is the inventor of three patents. She is also the author of nine books on data management, data mining, and data security. Her research interests include information security and data management. She is an elected fellow of three professional organizations, such as IEEE, American Association for the Advancement of Science (AAAS), and the British Computer Society (BCS) for her work in data security. She serves on the editorial board of numerous journals, including *ACM Transactions on Information and Systems Security* and IEEE TRANSACTIONS ON DEPENDABLE AND SECURE COMPUTING.