IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 30 September 2024, accepted 21 October 2024, date of publication 28 October 2024, date of current version 12 November 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3486734

RESEARCH ARTICLE

CycleGAN-Gradient Penalty for Enhancing Android Adversarial Malware Detection in Gray Box Setting

FABRICE SETEPHIN ATEDJIO[1], JEAN-PIERRE LIENOU[2], FREDERICA F. NELSON[3], SACHIN S. SHETTY[4], (Senior Member, IEEE), AND CHARLES A. KAMHOUA[3], (Senior Member, IEEE)

[1]Department of Mathematics and Computer Science, University of Dschang, Dschang, Cameroon
[2]Department of Computer Engineering of Technology Fotso Victor of Bandjoun, University of Dschang, Dschang, Cameroon
[3]DEVCOM Army Research Laboratory, Adelphi, MD 20783, USA
[4]Department of Computational Modeling and Simulation Engineering, Old Dominion University, Norfolk, VA 23529, USA

Corresponding author: Fabrice Setephin Atedjio (fabriceatedjio@gmail.com)

This work was supported by the Army Research Office under Grant W911NF-21-1-0326.

ABSTRACT Adversarial attacks pose significant threats to Android malware detection by undermining the effectiveness of machine learning-based systems. The rapid increase in Android apps complicates the management of malicious software that can compromise user defense solutions. Many current Android defense techniques rely on deep learning methods. Malicious users exploit GAN-based attacks to achieve adversarial attack transferability and deceive target models by crafting adversarial examples based on known models. We propose a new model based on a Cycle Generative Adversarial Network (CycleGAN) to detect GAN-based attacks. This model incorporates a gradient penalty to enhance the detection rate of the target model. Our investigation focuses on a gray box scenario, where the attacker has partial information about the model. The results show that our model outperforms existing classifiers in detecting adversarial attacks.

INDEX TERMS Adversarial attacks, CycleGAN, gradient penalty, android, deep learning, gray box.

I. INTRODUCTION

Global smartphone shipments increase by 7.8% year over year, reaching 289.4 million units in the first quarter of 2024, according to IDC statistic data [1]. With the vast number of Android users and millions of apps worldwide, malicious actors attempt to compromise these apps to gain access to sensitive information. The benefits of machine learning (ML) models in fields such as image recognition, malware detection, and anomaly detection are significant. However, these models are vulnerable to adversarial examples (AEs) [2], which are crafted inputs designed to mislead the classification model. Robustness of a model against AEs is indispensable and crucial. Numerous studies on Android detection solutions [3], [4], [5] have been published in the literature by proposing classifiers to distinguish between benign and malicious software. However, These classifiers are vulnerable to AEs, which are created by adding specific perturbations to natural inputs. Chen et al. [6] generated AEs by injecting perturbations into the AndroidManifest.xml and the APK's Dex code, effectively concealing well-known Android malware detection systems like Drebin [7]. Adversarial perturbations can enable malware to disguise itself from ML-based detectors. Unfortunately, many Android malware detectors lack adequate defenses against adversarial attacks. Due to the notable distinctions between images and Android apps, defensive techniques designed for image adversarial instances [8] cannot be simply transferred to Android malware detection. Researchers have demonstrated that it is feasible to craft specific examples that can deceive ML models [9]. For instance, Grosse et al. [10] showed how to manipulate Android malware samples to mislead

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

neural network malware detection systems. Their approach modified a method proposed by Papernot et al. [11], utilizing the Jacobian matrix of the neural network to create adversarial samples. It is important to note that the creation of AEs is generally applicable across various ML algorithms due to the inherent properties of these architectures. This vulnerability extends to contemporary deep neural networks as well. The ability of two models, trained on distinct datasets and network topologies, to misclassify the same adversarial example (AE) presents an intriguing scenario in a black box setting. In this context, even with limited knowledge of the victim's model, an adversary can train a source network, create the AE, and transfer it to the victim's network [12]. In the context of malware, an AE attacker can deceive a trained malware detector into failing to identify malicious software by subtly altering certain aspects of the code. Because an AE generated from one ML model can also mislead another model, they can be highly effective. AE transferability refers to the ability of an AE that evades one model to bypass others with different architectures. Even without precise knowledge of the victim model, transferability allows an attacker to create AEs using a substitute or source model and then apply them to the victim model, achieving a high success rate. Many previous studies have focused on how attackers can increase transferability to deceive defense methods. In contrast, this work proposes a model to thwart adversarial attacks generated by generative adversarial networks (GANs). The goal is to enhance defense methods against transferability in a gray box scenario. The contributions are as follows:

- We propose a solution based on CycleGAN to prepare our model to resist adversarially transferable samples from malicious users. Since the attack aims to transfer AEs from a substitute model to the defender or victim model, we aim to train the defender model to recognize these transferable samples, enhancing its robustness against GAN-based attacks in a gray box context.
- We introduce the concept of gradient penalty to CycleGAN to improve model stability. This addition enhances the adaptability of the neural network used for classifying benign and adversarial samples, allowing for better distinction between different classes.
- We generate adversarial samples within the context of a gray box scenario, aligned with the adversarial threat model, and test them on the defender model.

The remainder of the paper is structured as follows: section II presents the related works, then section III describes cycleGAN, section IV showcases our model, section V is dedicated to the results and discussions, and in the end section VI concludes with perspectives on future work.

II. RELATED WORKS

A. ADVERSARIAL EXAMPLE

Many ML algorithms are highly susceptible to malicious attacks. If ML-based malware detection algorithms can be easily defeated by specific adversarial techniques, they are not suitable for real-world applications. Deep learning AEs have garnered significant interest from researchers. Szegedy et al. [8] demonstrated that applying imperceptible perturbations to images can increase classification errors in a trained neural network, preventing accurate image classification. Samples with these perturbations are known as AEs. To generate AEs, Goodfellow et al. [13] proposed a gradient-based approach. When creating AEs, Papernot et al. [11] used the Jacobian matrix to determine which features to modify when creating these examples. Additionally, Grosse et al. [14] recommended a gradient-based technique for generating adversarial Android malware instances. These AEs are designed to deceive malware detection algorithms based on neural networks.

B. GAN FOR ANDROID

MalGAN [15] is an adversarial malware attack technique designed for computers, but it is also applicable to Android due to the significant similarity between PC software and Android applications. The generator produces harmful adversarial malware instances by using noise and malicious samples as input. The basic white box attack is transformed into a black box attack by employing a replacement detector–essentially a model that mimics the target detector and uses its gradient to generate adversarial samples. In addition to MalGAN, various other techniques for generating AEs have emerged in recent years. For instance, Hu et al. [16] proposed sequence generation based on API calls, while MalGAN leverages binary characteristics in software and a Recurrent Neural Network (RNN) model to create adversarial instances. E-MalGAN [17] enhances MalGAN by introducing a new substitute detector that learns to detect AEs within the detection system. Peng et al. [18] developed AEs using a Long Short-Term Memory (LSTM) network generator and a Convolutional Neural Network (CNN) as a substitute model. In another study, Peng et al. [19] used RNNs to generate adversarial API sequences, utilizing the context of API calls and word embeddings. Wang utilized LSGAN-AT [20] to improve the generation of smoother AEs by applying Least Squares (LS) loss to optimize the GAN's network topology. The model p-MalGAN, which incorporates predictive capabilities, was proposed in [21]. This approach also addresses the issue that both the detector and the attacker utilize the same features during the development of these methods. Li et al. [17] proposed an approach using bi-objective GANs to develop an AE attack strategy against Android malware classifiers. Jan et al. [22] utilized GANs to enhance the security of Android malware detectors through features based on intents. Taheri et al. [23] tested four different evasion attack models on Android malware classifiers and developed a countermeasure using GANs, reporting that GAN-based techniques improved evasion detection of Android malware by up to 50%. Additionally, Rafiq et al. [24] introduced DanDroid, a model that categorizes both obfuscated and

unobfuscated malicious and benign Android applications using GANs. They employed the classifier-two sample test (C2ST) to evaluate the accuracy and expected loss of both the discriminator and the generator. In practice, malicious data was input into the GANs to generate artificial data that closely resembled actual malicious programs.

C. TRANSFERABILITY IN ANDROID ADVERSARIAL MALWARE

Wallace et al. [25] employed gradient-based attacks to investigate the transferability of black box machine translation systems in natural language processing applications. They developed an imitation model that mimicked the victim model, allowing them to create AEs that could be transferred to production systems. Afterward, to enhance the transferability of adversarial attacks, the authors in [26] introduced an input transformation-based attack architecture. It is important to note that deep learning-based adversarial attacks have become a significant concern in the field of artificial intelligence, particularly in computer vision applications [26]. The transferability of AEs has also been explored in various computer vision tasks. The authors in [27] proposed AEs to improve cross-task transferability in classification, explicit content detection, and object detection models. Suciu et al. [28] developed a system to assess actual AEs in contexts such as image classification, data breach prediction, Twitter-based exploit prediction, and Android malware detection.They suggested poisoning and evasion attacks against ML systems, demonstrating that a recent evasion attack is less effective in the absence of broad transferability. The authors examined the transferability of these attacks and identified various factors that influence this characteristic. Similarly, Demontis et al. [29] evaluated the transferability of evasion and poisoning techniques across three datasets related to different applications, including face recognition, handwritten digit identification, and Android malware detection. Many state-of-the-art studies focus on enhancing the attacker's capabilities, by demonstrating methods to increase transferability to compromise the victim model. In this paper, our main goal is to examine the transferability problem from a different perspective by proposing a model that strengthens the victim or defense model against GAN-based transferability attacks.

III. CycleGAN

CycleGAN, also known as cycle-consistent generative adversarial network, is a type of usual GAN that is designed to train two generators, G_M and G_B, along with two discriminators, D_M and D_B. It was initially developed for unpaired image-to-image translation [30]. Using a diverse set of samples from both the target and source domains, it employs unsupervised training to build the model. This straightforward approach yields visually striking results across a wide range of applications. CycleGAN learns mappings $G_B : \Sigma_M \rightarrow \Sigma_B$ and $G_M : \Sigma_B \rightarrow \Sigma_M$. The cycle consistency loss for both G_B and G_M ensures the bijectivity of these mapping functions. Importantly, the distinct features of CycleGAN arise from the requirement that both functions must interact with one another. The adversarial loss is combined with the cycle loss to achieve the goal of translating unpaired data effectively.

IV. ADVERSARIAL THREAT MODEL AND DEFENSIVE APPROACH

A. CycleGAN GENERATOR

The CycleGAN generator comprises three key components: the encoder, transformer, and decoder. First, the encoder is a neural network that compresses input data into a lower-dimensional latent space, effectively capturing essential features and information. It extracts relevant characteristics from the input (e.g., features of benign APKs), allowing the model to concentrate on critical aspects necessary for transformation. Second, the transformer modifies the latent representation obtained from the encoder, adjusting features to align with those of the target domain (e.g., characteristics of malware). This component introduces adversarial manipulations, generating a representation that resembles malware while retaining certain attributes of the original benign input. Finally, the decoder reconstructs the transformed latent representation back into the original data format, producing the final output. It translates the modified features into a complete set of APK features that mimic malicious behavior, thereby creating adversarial examples. The discriminator evaluates the output from the decoder, providing feedback to the generator based on whether the output is classified as real or fake.

B. ADVERSARIAL THREAT MODEL

Security threats are classified based on their objectives and capabilities. This section outlines the adversarial threat model, specifically designed to counter malware evasion attacks, which consists of four components: adversarial knowledge, adversarial capabilities, adversarial goals, and attack surface.

1) ADVERSARIAL KNOWLEDGE

Adversarial knowledge refers to the amount of information that an attacker possesses, or is presumed to possess, to conduct attacks against a model. Based on this knowledge, adversarial attacks can be categorized into three types:

- White box attack: In a white box scenario, the attacker has full access to the underlying model. This includes knowledge of the algorithm, training data, hyperparameters, gradient information, and other relevant details. Due to the extensive information available, attacks in a white box context are relatively straightforward. Existing state-of-the-art works in white box environments have demonstrated a significant level of adversarial effectiveness [31].
- Black box attack: In a black box scenario, the attacker only has access to the model's inputs and outputs, with no information about its internal architecture.

Typically, the attacker constructs a surrogate model by inferring the target model's structure from the input-output relationships [32].

- Gray box: This type of attack falls between the black box and white box approaches, where the attacker has some limited knowledge about the model but does not possess complete information.

2) ADVERSARIAL CAPABILITIES

Adversarial capabilities refer to an adversary's capacity and are based on their understanding of the target model or victim model. The training dataset $\mathcal{D}$, the feature representation of a sample $\mathcal{X}$, the model architecture g, and the model parameters θ are the four types of knowledge that an attacker might obtain about the victim model. Depending on how much data the attacker has, a black box attacker does not know anything about the victim model, and a gray box attacker only knows a subset of $\{\mathcal{X}, \mathcal{D}, g, \theta\}$. A white box attacker, on the other hand, is fully aware of all the details of the targeted victim model. In this study, we focus on a gray box attack, where the attacker is aware of a subset of $\mathcal{D}$ and $\mathcal{X}$.

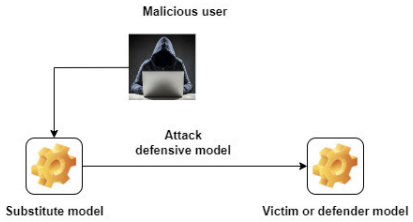

FIGURE 1. Goal of the attacker.

3) ADVERSARIAL GOAL

The primary objective of a malicious user is to deceive the target model into making incorrect classifications, as illustrated in Fig. 1. The attacker utilizes resources such as $\mathcal{X}$ and a subset of D to develop a substitute model against the defense or victim model. Since the attacker constructs this substitute model independently, it functions as a white box model for them. The attacker generates the corresponding AE $\mathcal{X}_{\text{ad}}$ for the substitute model based on a malicious clean sample $\mathcal{X}$; thus, $\mathcal{X}_{\text{ad}}$ is an effective AE on the substitute model. Finally, to assess the effectiveness of $\mathcal{X}_{\text{ad}}$ on the victim model, the attacker applies it. The greater the likelihood that $\mathcal{X}_{\text{ad}}$ succeeds against the victim model, the more transferable the attack becomes.

C. CycleGAN MODEL TO GENERATE ADVERSARIAL SAMPLES

The model presented here includes two generators: generator G_B and generator G_M, and two discriminators: discriminator D_B and the discriminator D_M. Generator G_B is used to create samples for the benign category, while generator G_M generates samples from the malicious category. One of the key advantages of CycleGAN is its ability to be trained on both paired and unpaired samples, allowing the model to leverage patterns within each domain to facilitate transitions that accurately represent each field. The CycleGAN model is employed to map adversarial malware samples to the benign domain and vice versa. This process enables the model to learn the intricate transformations that occur when malware attempts to evade detection. We utilized the Drebin dataset [7], which we divided into two unpaired subsets: malicious and benign. Our architecture consists of two GANs, each with its generator and discriminator. The first GAN generates benign samples from malicious inputs, as illustrated in Fig. 2, while the second GAN creates malicious samples from benign inputs, as shown in Fig. 3. Each GAN also includes a discriminator to evaluate the authenticity of the generated samples. As training progresses, they can produce samples that closely resemble those from the target domains.

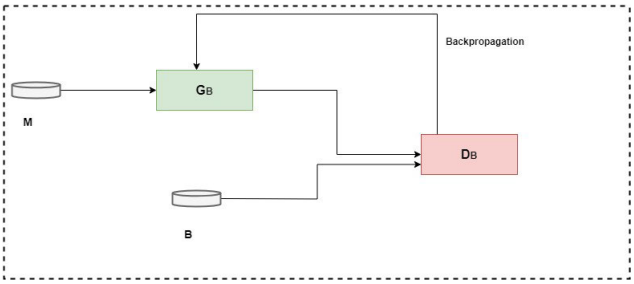

FIGURE 2. Generating fake benign samples.

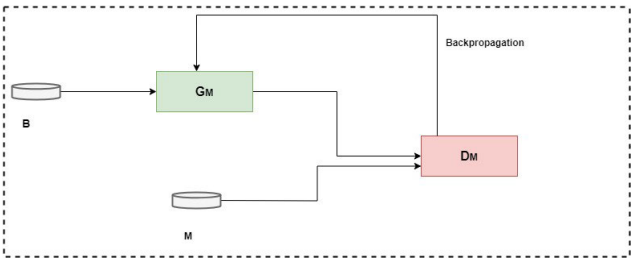

FIGURE 3. Generating fake malicious samples.

1) CYCLE LOSS

Like other GAN models, the discriminator and generator of CycleGANs are trained using the adversarial zero-sum principle. The discriminators focus on identifying generated false data, while the generators work to enhance their ability to deceive them. The generators translate input samples, where G_M receives samples from the set of malicious samples Σ_B and G_B receives samples from the set of benign samples Σ_M. G_M aims to generate adversarial data that closely resembles traffic from Σ_B, and vice versa. Discriminator D_M distinguishes between real malicious samples and adversarial samples generated by G_M, producing a real or fake decision. Similarly, discriminator D_B performs the same function with genuine benign traffic and adversarial traffic from G_B.

Additionally, the generators are designed to produce adversarial data in the target domain while also synthesizing

reconstructed data from the source domain. This is accomplished by inputting the generated samples into the corresponding generator and comparing the results to the original samples. A cycle is defined as the transition of samples across both generators. As illustrated in Fig. 4, each pair of generators is trained to improve the reproduction of the source samples. This process is known as cycle consistency. The cycle-consistency property ensures that our model is better at capturing the underlying structure of the data, making CycleGAN more robust for adversarial defense tasks.

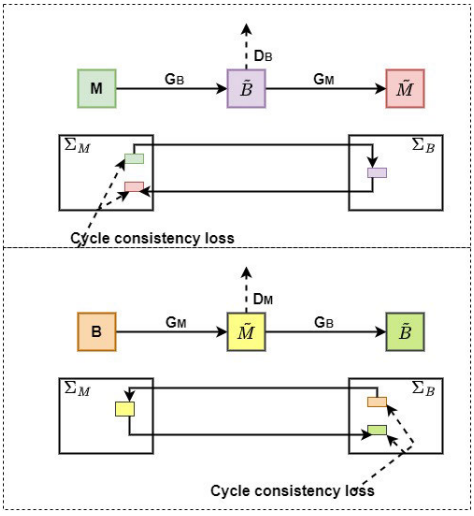

FIGURE 4. Cycle consistency loss [30].

The mathematical formulation of the cycle consistency loss is as follows:

For any malicious sample from Σ_M, the data translation cycle must be able to return MM to its original instance:

$$\forall M \in \Sigma_M, M \to G_B(M) \to G_M(G_B(M)) \approx M \quad (1)$$

For any benign sample from Σ_B, the data translation cycle must return BB to its original instance:

$$\forall B \in \Sigma_B, B \to G_M(B) \to G_B(G_M(B)) \approx B \quad (2)$$

The cycle consistency loss [30] is thus defined as:

$$\mathscr{L}_{cyc}(G_M, G_B) = \mathbb{E}_{M\in\Sigma_M}\left[||G_M(G_B(M)) - M||_1\right] + \mathbb{E}_{B\in\Sigma_B}\left[||G_B(G_M(B)) - B||_1\right] \quad (3)$$

2) IDENTITY LOSS

Identity loss ensures that input data from the target domain produces the same input samples. This is expressed as: $G_M(M) \approx M$ and $G_B(B) \approx B$

The identity loss [30] is defined as:

$$\mathscr{L}_{id}(G_M, G_B) = \mathbb{E}_{M\in\Sigma_M}\left[||G_M(M) - M||_1\right] + \mathbb{E}_{B\in\Sigma_B}\left[||G_B(B) - B||_1\right] \quad (4)$$

where M represents the malicious traffic in the malicious training set Σ_M, and B represents the benign traffic in the benign training set Σ_B. The Fig. 5 illustrates the identity loss.

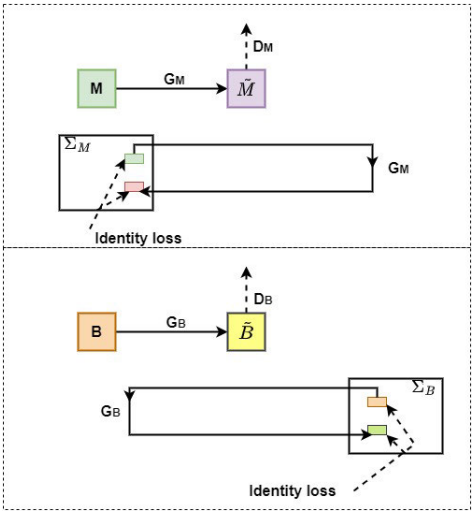

FIGURE 5. Identity loss [30].

3) ADVERSARIAL LOSS

Adversarial losses are closely related to the mapping functions G_M and G_B. For the mapping $G_M : \Sigma_B \to \Sigma_M$ and its discriminator D_M, we have:

$$\mathscr{L}_{adv}(G_M, D_M) = \mathbb{E}_{M\in\Sigma_M}\left[\log(D_M(M))\right] + \mathbb{E}_{B\in\Sigma_B}\left[\log(1 - D_M(G_M(B)))\right] \quad (5)$$

For the mapping $G_B : \Sigma_M \to \Sigma_B$ and its discriminator D_B, we have:

$$\mathscr{L}_{adv}(G_B, D_B) = \mathbb{E}_{M\in\Sigma_M}\left[\log(D_B(B))\right] + \mathbb{E}_{B\in\Sigma_B}\left[\log(1 - D_B(G_B(M)))\right] \quad (6)$$

where M represents the malicious traffic in the malicious training set Σ_M, and B represents the benign traffic in the benign training set Σ_B.

4) GENERATOR LOSS FUNCTION

The generator aims to produce data that closely resembles the original data, while the discriminator attempts to differentiate between original and translated samples. The discriminator seeks to maximize the loss function, whereas the generator aims to minimize it. Therefore, the generator loss function [30] can be expressed as follows:

$$\mathscr{L}(G_B, G_M, D_B, D_M) = \mathscr{L}_{adv}(G_B, D_B) + \mathscr{L}_{adv}(G_M, D_M) + \alpha_1 \cdot \mathscr{L}_{cyc}(G_M, G_B) + \alpha_2 \cdot \mathscr{L}_{id}(G_M, G_B) \quad (7)$$

where α_1 and α_2 are parameters that control the weightings of the cycle consistency loss and the identity loss, respectively.

Algorithm 1 illustrates the process of the CycleGAN architecture for generating systems.

D. DETECTING ADVERSARIAL ANDROID MALWARE BY BUILDING CycleGAN-GRADIENT PENALTY

The model described here is a type of GAN that features two generator architectures and two critic architectures.

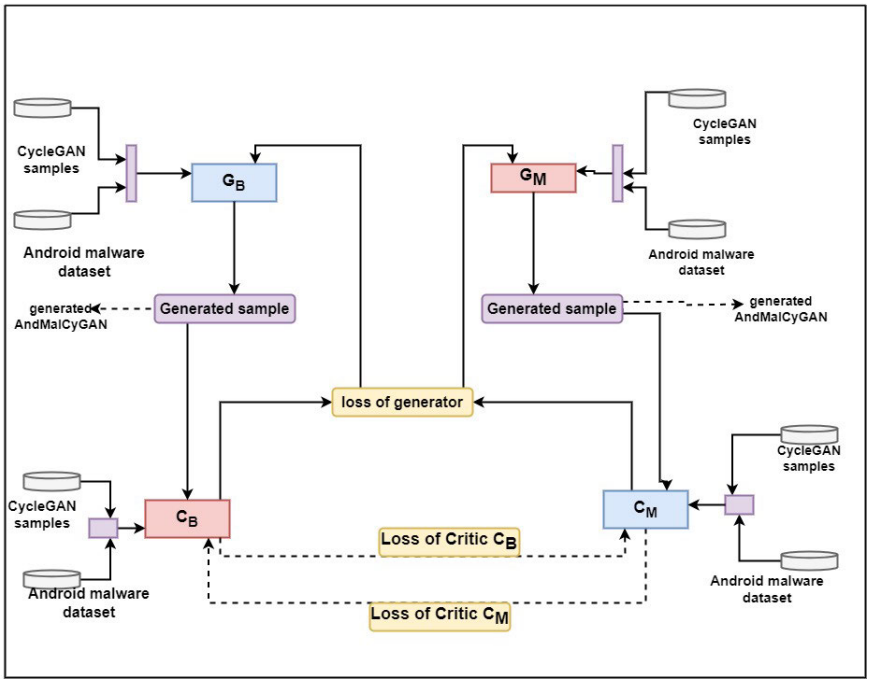

FIGURE 6. Detecting adversarial Android malware by building CycleGAN-gradient penalty.

Algorithm 1 Training of the cycleGAN Architecture Generating System

INPUT:
G_M: malicious samples generator,
G_B: benign samples generator,
Σ_M: malicious domain,
Σ_B: benign domain,
num_epochs
OUTPUT: trained cycleGAN architecture for adversarial samples
for num_epochs **do** $M^{(i)}_{i=1,\ldots,m} \in \Sigma_M$
$B^{(i)}_{i=1,\ldots,m} \in \Sigma_B$
Generate m samples of $G_B(M)$ and $G_M(B)$
Generate m sample of $G_B(G_M)$ and $G_M(G_B)$
Generate m samples of $G_M(M)$ and $G_B(B)$
update the discriminator D_M according to the adversarial loss function using equation (5)
update the discriminator D_B according to the adversarial loss function using equation (6)
update the generator G_M and G_B according to the total cycleGAN loss function using equation (7)

It incorporates a gradient penalty to stabilize the model and enhance performance. As shown in Fig. 6, the traffic generated by the CycleGAN system is included in the training set of the model. The Critic C_B aims to differentiate benign traffic from malicious traffic, while the critic C_M focuses on identifying malicious traffic. Both critics C_M and C_B work to detect adversarial samples. Algorithm IV-D3 details the procedure.

Considering the generator G_B and the critic C_B. The Wasserstein loss (W-loss) approximation of the Earth Mover's [33] distance of the adversarial loss given in Equation 8 is:

$$\mathbb{E}_{B\in\Sigma_B}(C_B(x) + \mathbb{E}_{B\in\Sigma_B}(C(G_B(x)) \tag{8}$$

Here, x is a sample, and C_B is a critic referred to as discriminator D_B. In this context, usually is used the term "critic" instead of "discriminator" because there is no logarithmic function involved. The goal for critic C_B is to maximize the distance between fake benign samples and real benign samples, while the generator G_B aims to minimize it.

The output of discriminator D_B must be a prediction between 0 and 1 for binary cross-entropy (BCE) loss to be effective. Consequently, the neural network of the discriminator for GANs trained with BCE loss typically employs a Sigmoid activation function in the output layer to confine the values between 0 and 1.

However, W-loss does not impose this condition, allowing the discriminator's neural network to include a linear layer at the end, which can output any real value. This output serves as the critic's assessment of how authentic a sample is. Since the output is no longer bounded between 0 and 1, we refer to it as a critic rather than a discriminator. W-loss approximates the Earth Mover's distance in a way that mitigates mode collapse and addresses vanishing gradient problems [33].

1) CycleGAN IN THE ANDROID ADVERSARIAL MALWARE

When implementing CycleGAN for malware detection, our model focuses on feature representations or transformed versions of Android applications rather than traditional images. The input to the model consists of transformed feature sets extracted from Android applications, which the model processes. The output is another structured representation of the application after the model has attempted to translate it, indicating the transformation it would undergo to become either benign or adversarial.

In a gray box context, the CycleGAN model learns to translate an adversarial example (a malicious application with adversarial modifications) back to its benign equivalent, and vice versa. This mechanism ensures that for each adversarial input (the attacked malware sample), a corresponding benign representation is produced as the output. Likewise, the inverse transformation is maintained, allowing for the mapping of benign samples to their potential representations under adversarial conditions.

2) W-LOSS WITH GRADIENT PENALTY

Implementing a gradient penalty serves to regularize the training process of the CycleGAN, ensuring that the generated adversarial samples exhibit smoother and more realistic properties. This enhancement significantly bolsters the defensive model's capacity to identify adversarial attacks effectively. W-loss is a straightforward expression that computes the difference between the expected values of the critic's output for a real example x and its prediction for a fake example $G(z)$:

$$\mathbb{E}(C(x) + \mathbb{E}(C(G(z)) \tag{9}$$

The critic C aims to maximize this expression to effectively distinguish between real and fake examples, while the generator G seeks to minimize it to make the generated example as close to the real example as possible. However, the critic must satisfy a specific requirement for training GANs with W-loss: it must be 1-Lipschitz continuous [34].

To implement the gradient penalty, we add a regularization term to the loss function. When the gradient norm exceeds 1, this regularization term penalizes the critic. The complete expression of the loss function used for training with W-loss and gradient penalty is given by:

$$\begin{aligned}\mathscr{L}(G, C) = \mathbb{E}(C(x)) - \mathbb{E}(C(G(z))) \\ + \beta \cdot \mathbb{E}\left(||\nabla C(\hat{x})||_2 - 1\right)^2\end{aligned} \tag{10}$$

Here, β is a hyperparameter that controls the weighting of the regularization term relative to the primary loss function. Checking the critic's gradient at every point in the feature space is typically impractical. Therefore, during implementation, we interpolate between genuine and fake examples to incorporate the gradient penalty logic:

$\hat{x} = \epsilon x + (1 - \epsilon)G(z)$

where ϵ is a random vector of benign samples.

3) CycleGAN-GRADIENT PENALTY ADVERSARIAL LOSS

From the equation 10 we can derive the following new adversarial loss for CycleGAN-gradient penalty.

For the mapping $G_M : \Sigma_B \rightarrow \Sigma_M$ and its discriminator C_M we have:

$$\begin{aligned}\mathscr{L}'(G_M, C_M) = \mathbb{E}(C_M(x)) - \mathbb{E}(C_M(G_M(z))) \\ + \beta_M \cdot \mathbb{E}\left(||\nabla C_M(\hat{x})||_2 - 1\right)^2\end{aligned} \tag{11}$$

For the mapping $G_B : \Sigma_M \rightarrow \Sigma_B$ and its discriminator D_B we have:

$$\begin{aligned}\mathscr{L}'(G_B, C_B) = \mathbb{E}(C_B(x)) - \mathbb{E}(C_B(G_B(z))) \\ + \beta_B \cdot \mathbb{E}\left(||\nabla C_B(\hat{x})||_2 - 1\right)^2\end{aligned} \tag{12}$$

Here, M represents the malicious traffic in the malicious training set Σ_M, and B represents the benign traffic in benign the training set Σ_B.

The new generator loss is given as follows:

$$\begin{aligned}\mathscr{L}(G_B, G_B, C_M, C_B) = \mathscr{L}'_{adv}(G_B, C_B) + \mathscr{L}'(G_M, C_M) \\ + \alpha_1 \cdot \mathscr{L}_{cyc}(G_M, G_B) \\ + \alpha_2 \cdot \mathscr{L}_{id}(G_M, G_B)\end{aligned} \tag{13}$$

Algorithm 2 Detecting Adversarial Android Malware Algorithm

INPUT: G_M malicious samples generator,
G_B benign samples generator,
C_M: malicious samples critic,
C_B: benign samples critic,
B: benign samples,
M: malicious samples,
adv: adversarial malicious samples generated by cycleGAN,
OUTPUT: trained defense model
Y = combine M and adv samples
for num_epochs **do** generate m samples of $G_M(B)$
generate m samples of $G_B(Y)$
for n_critic_M **do** $\max[\mathbb{E}(C_M(x_Y)) - E(G_M(z_Y)) + \beta_M \mathbb{E}(||\nabla C_M(\hat{x_Y}||_2 - 1)^2$]
for n_critic_B **do**
$\max[\mathbb{E}(C_B(x_B)) - E(G_B(z_B)) + \beta_B \mathbb{E}(||\nabla C_B(\hat{x_B}||_2 - 1)^2]$
update critic C_M according to the equation (11)
update critic C_B according to the equation (12)
update generator G_M and G_B according to the to the new equation (13)

V. EXPERIENCES, RESULTS AND DISCUSSIONS

A. EXPERIENCES AND RESULTS

Our experiments were conducted on an Intel® Xeon® CPU @ 2.20GHz machine with 12GB of RAM, running Ubuntu 22.04.3 LTS. We utilized PyTorch version 2.3.0 and Python 3.10 for programming.

The dataset was split into 70% for training and 30% for testing, with each part further divided into malicious (M) and benign (B) samples. Although CycleGAN was initially

developed for one-to-one mappings (e.g., mapping one image to another), Android adversarial malware attacks involve various transformations, which means multiple adversarial examples could correspond to a single benign app or vice versa. To handle this, we have introduced additional variability during training by using diverse sets of adversarial examples for the same benign app, allowing the CycleGAN to learn a broader range of mappings. This helps the model generalize better and learn to detect different adversarial modifications effectively. Various performance metrics were employed for comparison, including accuracy, precision, F1-score, and the Receiver Operating Characteristic (ROC) curve. The training process is conducted using a supervised learning approach.

TABLE 1. Hyperparameters for attack or substitute model.

Parameter	Value
Epochs/Batch	300/128
Optimizer	Adam
Learning rate	0.001
α_1	3
α_2	0.04

TABLE 2. Hyperparameters for the defender training model.

Parameter	Value
Epochs/Batch	300/128
Optimizer	Adam
Learning rate	0.000154
β_M	18
n_critic_M	13
β_B	10
n_critic_B	7
α_1	1
α_2	0.1

B. RESULTS ON NORMAL SAMPLES

This section presents the performance metrics obtained without any adversarial attacks.

C. RESULTS ON ADVERSARIAL SAMPLES

Here, we present the performance metrics after applying adversarial samples generated from the substitute model.

D. DISCUSSIONS

We first evaluated the model using normal samples without adversarial attacks. Fig. 7 to 11 present the results without adversarial attacks: Fig. 7 shows the accuracy result, Fig. 8 represents the precision metric, Fig. 9 displays the recall, Fig. 10 indicates the F1-score, and Fig. 11 illustrates the ROC curve. Our model achieved outstanding classification results among the five classifiers, with an accuracy of 99.8% accuracy, 99.8% precision, 99.9% recall, 99.9% F1-score, and 99.8% ROC. In the second place, the Random Forest (RF) classifier achieved 99% accuracy, 99% precision, 99.5% recall, 99.2% F1-score, and 99.8% ROC. Following closely

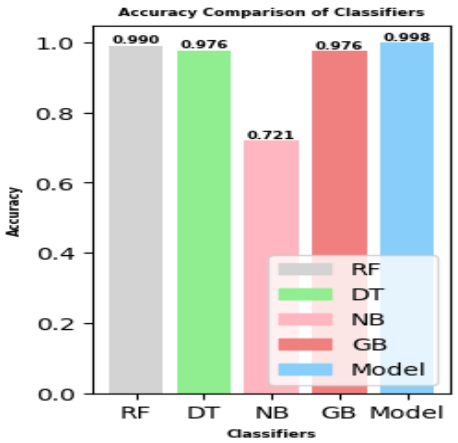

FIGURE 7. Normal samples: accuracy.

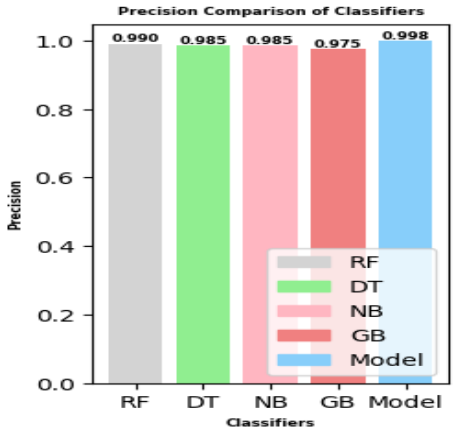

FIGURE 8. Normal classification: precision.

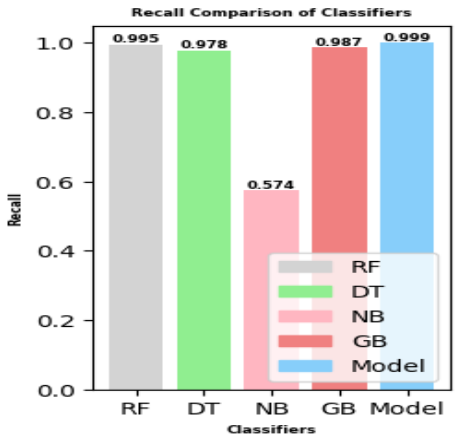

FIGURE 9. Normal classification: recall.

were the Decision Tree (DT) classifier with 97.6% accuracy, 98.5% precision, 97.8% recall, 98.2% F1-score, and 97.5 % ROC.; and the Gradient Boosting (GB) classifier with 97.6% accuracy, 97.5% precision, 98.7 %recall, 98.1% F1-score and

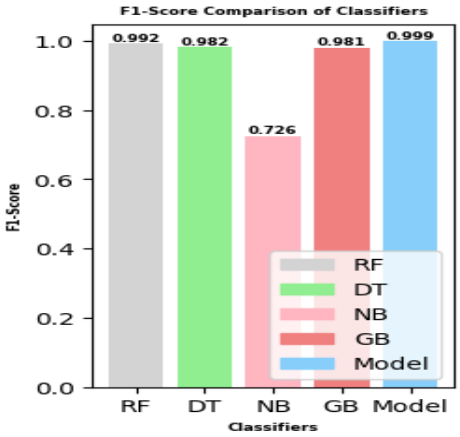

FIGURE 10. Normal classification: F1-score.

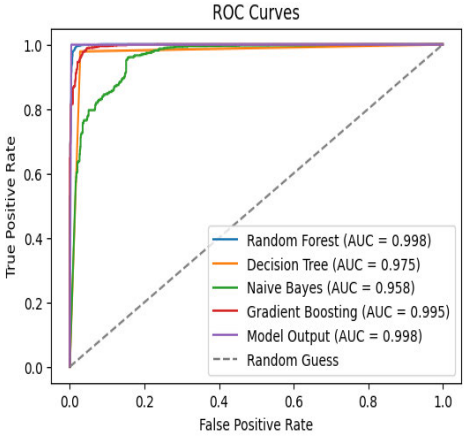

FIGURE 11. Normal classification: ROC curve.

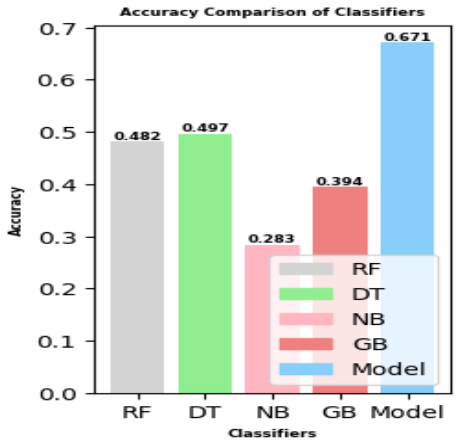

FIGURE 12. Adversarial classification: accuracy.

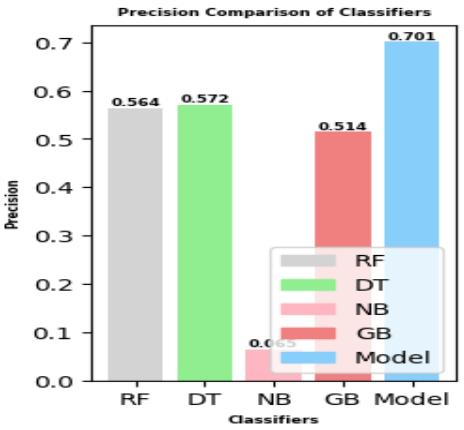

FIGURE 13. Adversarial classification: precision.

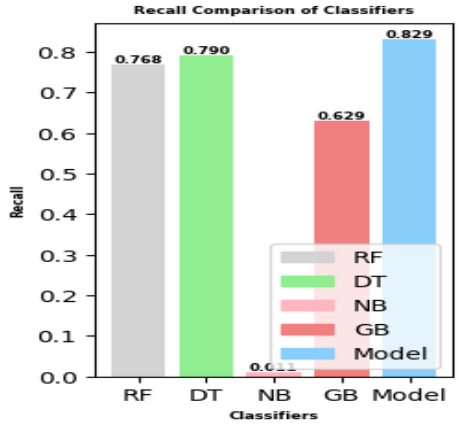

FIGURE 14. Adversarial classification: recall.

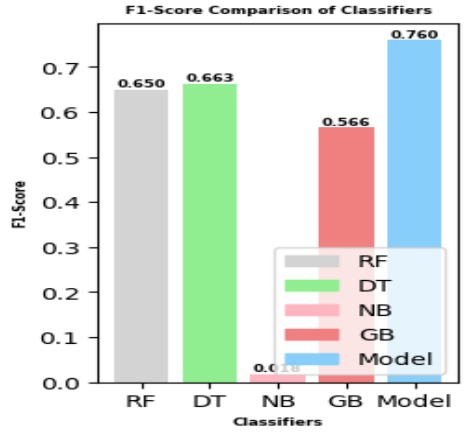

FIGURE 15. Adversarial classification: F1-score.

99.5% ROC. The Naive Bayes (NB) classifier performed the least well, with 72.1%, 98.5% precision, 57.4% recall, 72.6%, and 95.8% ROC.

After generating adversarial samples from the substitute model, we applied them to the defensive model and classifiers to assess new performance metrics. Fig. 12 to 16 show the results of GAN adversarial attacks in the gray box context:

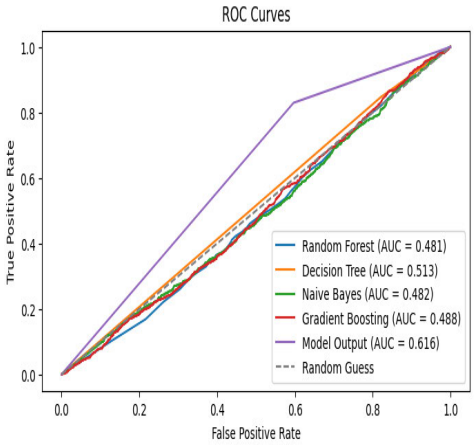

FIGURE 16. Adversarial classification: ROC curve.

Fig. 12 displays the accuracy performance, Fig. 13 presents precision, Fig. 14 points recall, Fig. 15 indicates F1-score, Fig. 16 illustrates the ROC curve.

Despite the attacks, our model remained the best among the classifiers, achieving, 67.1% accuracy, 70.1% precision, 82.9% recall, 76% F1-score, and 61.61% ROC. In contrast, RF's performance decreased significantly, dropping from 99% to 48.2% accuracy, from 99% to 56.4% precision, from 99.5% to 76.8% recall, from 99.2 to 65% F1-score, and from 99.8% to 48.1% ROC. The DT classifier scored 49.7% for accuracy, 57,2% for precision, 79% for recall, 66.3% for F1-score, and 51.31% for ROC. The GB classifier achieved 39.4% of accuracy, 51.4% of precision, 62.9% of recall, 56.6% of F1-score, and 48.8% of ROC. Lastly, NB's performance was significantly diminished, with accuracy at 28.3%, precision at 6.5%, recall at 1.1%, F1-score at 1.8%, and ROC at 48.2%.

Therefore, our model proved to be the most effective in both the presence and absence of GAN-based adversarial attacks.

VI. CONCLUSION AND FUTURE WORKS

The goal of this paper was to propose a solution to the problem of GAN-based adversarial malware. This issue can arise in various contexts, including white box, black box, and gray box scenarios. We focused on a gray box scenario in which an attacker has partial information about the model. The attacker exploits this knowledge to construct a substitute model, generating adversarial samples before applying them to the victim or defender model. Our proposed solution leverages the attacker's existing knowledge to train the model to recognize and prevent these adversarial samples. We built our model based on CycleGAN logic, enabling it to learn how to identify adversarial inputs. Additionally, we modified the CycleGAN to include a gradient penalty, which helps to stabilize the model and increase the detection rate. Future work will address all types of adversarial attacks, particularly in the context of transferability, and will explore adversarial attacks in black box scenarios.

ACKNOWLEDGMENT

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. government. The U.S. government is authorized to reproduce and distribute reprints for government purposes notwithstanding any copyright notation herein.

REFERENCES

[1] (2024). *Worldwide Quarterly Mobile Phone Tracker*. [Online]. Available: https://www.idc.com/getdoc.jsp?containerId=IDC_P8397
[2] A. Rashid and J. Such, "StratDef: Strategic defense against adversarial attacks in ML-based malware detection," *Comput. Secur.*, vol. 134, Nov. 2023, Art. no. 103459.
[3] S. Wang, Q. Yan, Z. Chen, B. Yang, C. Zhao, and M. Conti, "Detecting Android malware leveraging text semantics of network flows," *IEEE Trans. Inf. Forensics Security*, vol. 13, no. 5, pp. 1096–1109, May 2018.
[4] S. Xi, S. Yang, X. Xiao, Y. Yao, Y. Xiong, F. Xu, H. Wang, P. Gao, Z. Liu, F. Xu, and J. Lu, "DeepIntent: Deep icon-behavior learning for detecting intention-behavior discrepancy in mobile apps," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2019, pp. 2421–2436.
[5] H. Li, S. Zhou, W. Yuan, X. Luo, C. Gao, and S. Chen, "Robust Android malware detection against adversarial example attacks," in *Proc. Web Conf.*, Apr. 2021, pp. 3603–3612.
[6] X. Chen, C. Li, D. Wang, S. Wen, J. Zhang, S. Nepal, Y. Xiang, and K. Ren, "Android HIV: A study of repackaging malware for evading machine-learning detection," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 987–1001, 2020.
[7] 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. 23–26.
[8] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," 2013, *arXiv:1312.6199*.
[9] E. Nowroozi, A. Dehghantanha, R. M. Parizi, and K.-K.-R. Choo, "A survey of machine learning techniques in adversarial image forensics," *Comput. Secur.*, vol. 100, Jan. 2021, Art. no. 102092.
[10] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial examples for malware detection," in *Proc. 22nd Eur. Symp. Res. Comput. Secur.*, Oslo, Norway. Cham, Switzerland: Springer, Sep. 2017, pp. 62–79.
[11] 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*, Mar. 2016, pp. 372–387.
[12] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*.
[13] I. J. Goodfellow, J. Shlens, and C. Szegedy, "Explaining and harnessing adversarial examples," 2014, *arXiv:1412.6572*.
[14] K. Grosse, N. Papernot, P. Manoharan, M. Backes, and P. McDaniel, "Adversarial perturbations against deep neural networks for malware classification," 2016, *arXiv:1606.04435*.
[15] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," in *Proc. Int. Conf. Data Mining Big Data*. Springer, 2022, pp. 409–423.
[16] W. Hu and Y. Tan, "Black-box attacks against RNN based malware detection algorithms," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 1–7.
[17] 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.
[18] X. Peng, H. Xian, Q. Lu, and X. Lu, "Generating adversarial malware examples with API semantics-awareness for black-box attacks," in *Proc. 6th Int. Symp.*, Tianjin, China. Cham, Switzerland: Springer, Sep. 2020, pp. 52–61.
[19] X. Peng, H. Xian, Q. Lu, and X. Lu, "Semantics aware adversarial malware examples generation for black-box attacks," *Appl. Soft Comput.*, vol. 109, Sep. 2021, Art. no. 107506.

[20] J. Wang, X. Chang, Y. Wang, R. J. Rodríguez, and J. Zhang, "LSGAN-AT: Enhancing malware detector robustness against adversarial examples," *Cybersecurity*, vol. 4, no. 1, pp. 1–15, Dec. 2021.
[21] Y. Guo and Q. Yan, "Android malware adversarial attacks based on feature importance prediction," *Int. J. Mach. Learn. Cybern.*, vol. 14, no. 6, pp. 2087–2097, Jun. 2023.
[22] S. Jan, T. Ali, A. Alzahrani, and S. Musa, "Deep convolutional generative adversarial networks for intent-based dynamic behavior capture," *Int. J. Eng. Technol.*, vol. 7, pp. 101–103, Jul. 2018.
[23] R. Taheri, R. Javidan, M. Shojafar, P. Vinod, and M. Conti, "Can machine learning model with static features be fooled: An adversarial machine learning approach," *Cluster Comput.*, vol. 23, no. 4, pp. 3233–3253, Dec. 2020.
[24] H. Rafiq, N. Aslam, B. Issac, and R. H. Randhawa, "An investigation on fragility of machine learning classifiers in Android malware detection," in *Proc. IEEE INFOCOM Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2022, pp. 1–6.
[25] E. Wallace, M. Stern, and D. Song, "Imitation attacks and defenses for black-box machine translation systems," 2020, *arXiv:2004.15015*.
[26] 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.
[27] Y. Jia, Y. Lu, S. Velipasalar, Z. Zhong, and T. Wei, "Enhancing cross-task transferability of adversarial examples with dispersion reduction," 2019, *arXiv:1905.03333*.
[28] O. Suciu, R. Marginean, Y. Kaya, H. Daume III, and T. Dumitras, "When does machine learning FAIL? Generalized transferability for evasion and poisoning attacks," in *Proc. 27th USENIX Secur. Symp.*, 2018, pp. 1299–1316.
[29] A. Demontis, M. Melis, M. Pintor, M. Jagielski, B. Biggio, A. Oprea, C. Nita-Rotaru, and F. Roli, "Why do adversarial attacks transfer? Explaining transferability of evasion and poisoning attacks," in *Proc. USENIX Security Symp.*, 2019, pp. 321–338.
[30] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2242–2251.
[31] 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.
[32] K. Aryal, M. Gupta, and M. Abdelsalam, "A survey on adversarial attacks for malware analysis," 2021, *arXiv:2111.08223*.
[33] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 214–223.
[34] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017, pp. 1–11.

FABRICE SETEPHIN ATEDJIO received the bachelor's degree in mathematics and in computer science, in 2017, and the master's degree in computer science, focusing on IoT security. He is currently pursuing the Doctorate degree with the University of Dschang, Cameroon.

He is currently a Researcher in artificial intelligence applied to cybersecurity. In 2021, he was selected to join the "Game Theory and Machine Learning for Cyber Deception, Resilience, and Agility (GMC-DRA)" Project, sponsored by U.S. Army Research Office. He holds professional certifications in cybersecurity and data analysis from Google and a certification in deep learning from DeepMind. In 2022, he was awarded the Best Poster Award at both the International Conference on Machine Learning and the Indabax Cybersecurity Event.

JEAN-PIERRE LIENOU received the Master of Science degree in system engineering from the Kiev Polytechnic Institute (National Technical University of Ukraine) and the Ph.D. degree from the University of Yaound I, Cameroon, in 2011. He is currently an Associate Professor with the Department of Computer Engineering, University of Dschang. Before his academic career, he was a Maintenance Engineer with Labotech Medical, where he was responsible for the upkeep and operation of medical imaging equipment. He joined the University of Dschang, in 2012, he has since established himself as a prominent figure in the field of computer engineering. He is also the Head of the Department of Computer Engineering, College of Technology, University of Bamenda. In addition to his academic responsibilities, he has been an active member of the Engineering Association of the National Group (EANG), since 2018. His research interests include method engineering applied to control systems, multi-agent systems utilized in power electric systems, cyber resilience, and the application of various artificial intelligence techniques in the diagnosis of complex systems. His work in these areas contributes significantly to advancements in both theoretical and practical aspects of engineering and technology.

FREDERICA F. NELSON is currently a Researcher and the Program Lead with U.S. Army Research Laboratory (ARL), Adelphi, MD, USA, where she leads research on machine learning and intrusion detection methods and techniques to promote cyber resilience and foster research on autonomous active cyber defence. She manages and negotiates the Research and Project Agreements for ARL between the network security branch and academia or international organizations. She is the lead for the robust low-level cyber-attack resilience for Military Defense (ROLLCAGE) Program working in collaboration with Army Tank Automotive Research, Development and Engineering Center, Office of Naval Research, and Air Force Research Laboratory to build a cohesive in-vehicular resilient system for defense against sophisticated enemy malware that strives to blend in with normal system activities. She has over 20 years combined experience in Cybersecurity Research, Software Engineering, and Program Management within the DoD and other federal services including the Federal Bureau of Investigation and the Department of Justice. She has expertise in leading projects to success from conception to execution and delivery/transfer. She currently serves as the Chairperson for the International Science Technology (IST-163) Panel—NATO Science & Technology Organization on the topic of deep machine learning for military cyber defence. She is a participant in the Army Education Outreach Program as an Ambassador and a Virtual Judge for the eCybermission Program.

SACHIN S. SHETTY (Senior Member, IEEE) received the Ph.D. degree in modeling and simulation from Old Dominion University, in 2007. He is currently an Executive Director of the Center for Secure and Intelligent Critical System and a Professor with the Department of Electrical and Computer Engineering, Old Dominion University. His research interests include the intersection of computer networking, network security, and machine learning. Within the last 15 years, he completed many large-scale projects with multiple collaborators and institutions and served as the PI/Co-PI on various grants and contracts, funded by various military and federal government departments and private businesses. He has published over 300 research articles in journals and conference proceedings and edited four books. Two research papers were chosen as the top 50 academic papers in blockchain, in 2018. His laboratory conducts cloud and mobile security research and has received over $18 million in funding from the National Science Foundation, Air Office of Scientific Research, Air Force Research Laboratory, Office of Naval Research, Department of Homeland Security, and Boeing. He received the Commonwealth Cyber Initiative Fellow, DHS Scientific Leadership Award, and Fulbright Specialist and was inducted into the Tennessee State University Million-Dollar Club. He was the Winner of the Electric Power Research Institute Cyber Security Challenge Competition, in 2019.

CHARLES A. KAMHOUA (Senior Member, IEEE) received the B.S. degree in electronics from the University of Douala (ENSET), Cameroon, in 1999, and the M.S. degree in telecommunication and networking and the Ph.D. degree in electrical engineering from Florida International University (FIU), in 2008 and 2011, respectively. He is currently a Researcher with the Network Security Branch of U.S. Army Research Laboratory, Adelphi, MD, USA, where he is responsible for conducting and directing basic research in the areas of game theory applied to cyber security. Prior to joining the Army Research Laboratory, he was a Researcher with U.S. Air Force Research Laboratory (AFRL), Rome, New York City, for six years and an educator in different academic institutions for more than ten years. He has held visiting research positions with Oxford and Harvard. He has co-authored more than 100 peer-reviewed journals and conference papers. He has been invited to more than 40 keynote and distinguished speeches and co-organized more than ten conferences and workshops. He has mentored more than 50 young scholars including students, postdoctorals, and AFRL Summer Faculty Fellow. He has been recognized for his scholarship and leadership with numerous prestigious awards, including the 2017 AFRL's Information Directorate Basic Research Award "For outstanding achievements in basic research," the 2017 Fred I. Diamond Award for the best paper published at AFRL's Information Directorate, 40 Air Force Notable Achievement Awards, the 2016 FIU Charles E. Perry Young Alumni Visionary Award, the 2015 Black Engineer of the Year Award (BEYA), the 2015 NSBE Golden Torch Award—Pioneer of the Year, and a selection to the 2015 Heidelberg Laureate Forum, to name a few. He is currently an Advisor of the National Research Council and a member of the FIU Alumni Association and the ACM.