IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2024.0429000

A defensive strategy against Android Adversarial Malware attacks

FABRICE SETEPHIN ATEDJIO[1], JEAN PIERRE LIENOU[2], FREDERICA F. NELSON[3], SACHIN S. SHETTY[4], CHARLES A. KAMHOUA [5]
[1]Department of Mathematics and Computer Science, University of Dschang, PO Box 67 Cameroon(e-mail: fabriceatedjio@gmail.com)
[2] Dept. of Comp Engineering of Techn. F. Victor of Bandjoun, University of Dschang, PO Box 39 Cameroon, CO 80523 USA (e-mail:lienou@gmail.com)
[3]DEVCOM Army Research Laboratory, Adelphi, MD 20783 USA (e-mail:frederica.f.nelson.civ@army.mil)
[4]Dept. of Comput. Model. and Simul. Eng., Old Dominion University, Virginia, USA (e-mail:sshetty@odu.edu)
[5]DEVCOM Army Research Laboratory, Adelphi, MD 20783 USA (e-mail:charles.a.kamhoua.civ@army.mil)

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

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

ABSTRACT Due to the popularity of Android mobile devices over the past ten years, malicious Android applications have significantly increased. Systems utilizing machine learning techniques have been successfully applied for Android malware detection to counter the constantly changing Android malware threats. However, attackers have developed new strategies to circumvent these systems by using adversarial attacks. An attacker can carefully craft a malicious sample to deceive a classifier. Among the evasion attacks, there is the more potent one, which is based on solid optimization constraints: the Carlini-Wagner attack. Carlini-Wagner is an attack that uses margin loss, which is more efficient than cross-entropy loss. We propose a model based on the Wasserstein Generative Adversarial Network to prevent adversarial attacks in an Android field in a white box scenario. Experimental results show that our method can effectively prevent this type of attack.

INDEX TERMS Adversarial Attack, Carlini-Wagner attack, Generative Adversarial Network, Android Adversarial Malware.

I. INTRODUCTION

ANDROID is the open-source and most widely used operating system in the world. Many attackers create various malicious applications to gain unauthorized access to any personal user device. However, the defenders on their side try to overcome the obstacle by developing solutions to counter the attackers. Machine learning (ML) algorithms have been created under the assumption that training and test data follow the same underlying probability distribution, which renders them susceptible to well-crafted attacks that violate this assumption, as first noted by Barreno et al. [1]. This implies that the weakest link in the security chain may be ML itself [2]. Despite enormous community efforts at defensive measures, malware continues to present an important threat to cyber security. ML is used to automate the identification of malware in the wild to deal with the increasingly serious scenario [3]. However, adversarial evasion attacks, in which an adaptive attacker perturbs or alters malware instances into adversarial examples that would be considered benign rather than harmful, might compromise ML-based solutions [4]. Many evasion attacks have been proposed to deceive ML algorithms, as presented in [5, 6, 7]. These evasion attacks have been proposed first for computer vision and later adapted for the cyber security domain. However, both fields are not the same. Computer vision is a continuous environment, whereas cyber security is a discrete domain where it is more difficult to manage adversarial attacks. Some authors have started to have more interest in Android adversarial malware. Malware detection has evolved into an ML application as a result of the rise in the number of released programs and applications. The presented features, however, have a significant impact on the detection's quality. Static and dynamic features are two categories that are typically distinguished in the literature. Static features can be easily gathered and included from the source code of the application. The more common type today is dynamic features, which sample features from an application while it is in use and observe usage patterns for access and communication. The last approach combines static and dynamic analysis [8]. While some authors have proposed solutions such as distillation defense [9] and adversarial training [6], [10], [11], which do not work with some attacks

Distribution Statement A: Approved for public release. Distribution is unlimited.

such as the Carlini-Wagner attack, others have focused on the Generative Adversarial Network (GAN) in the context of misleading classifier [12] and others for preventing attacks [13]. We aim to utilize Wasserstein GAN, a variation of the standard Generative Adversarial Network, which enhances stability and training by employing a distinct loss function based on the Wasserstein distance. This approach measures the divergence between the real data distribution and the generated data distribution, providing a solution to mitigate the Carlini-Wagner attack. Our contributions to this paper are as follows:

- Proposing model to thwart adversarial evasion attack. The model leverages the underlying logic of Wasserstein GAN to build a model robust against those adversarial attacks;
- Focusing our protection by countering the Carlini-Wagner attack. This will make the model robust against this attack;
- Evaluating the dataset's vulnerability to the Carlini-Wagner attack, as well as the effectiveness of our method across five different classification techniques.

The rest of this paper is organized as follows: in section II, we present a background on an explanation of Android, GAN, and machine learning classifiers. Section III describes the related works on Android malware detection and MalGAN. Section IV presents the attack model and dataset. Section V explains the proposed approach. Section VI, gives evaluation results and performance. Finally, section VII concludes this paper and presents future works.

II. BACKGROUND

A. PRELIMINARIES ON ANDROID

The Android app is compiled and packed in a single archive .apk file that contains the Androidmanifest.xml file, Dalvid executable (class.dex) file, assets, and resources as presented in Fig.1.

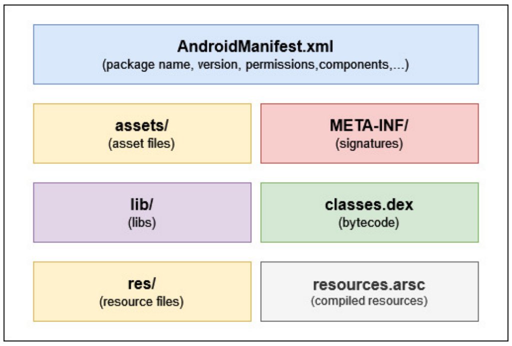

FIGURE 1. Android Architecture.

The AndroidManifest file represents a framework based on components for constructing mobile apps, which keeps information about its structure [14]: Services, Activities, Broadcast Receivers, and Content Providers. The components are first configured using a set of application attributes to specify default values for relevant elements. The activities of each component are further described through filtered intentions, which declare the types of intentions it can reply to. The AndroidManifest file also includes a list of the permissions that the app requests to carry out tasks. Since permissions, filtered intents, and application attributes can show how an app interacts with other apps and the operating system, we extract them as features from the Android manifest file. Dalvik executable (dex) files can be created from Android apps and executed on the Dalvik Virtual Machine (DalvikVM) [15]. The Dalvik VM uses API calls to access operating system resources and functionality. It can interpret the dex file, which contains compiled code created for Android, as user-implemented methods and classes. Exceptions are another way the dex file highlights conditions that an application might want to avoid. Hence, the dex file may be used to represent the actions of an Android app through new instances, API calls, and exceptions. We use Virtual Machine Santoku, which contains all the tools we need for extracting features such as Androguard, APKtool, Baksmali, and JD-Gui.

B. GENERATIVE ADVERSARIAL NETWORK

A GAN is a generative model that was initially described by Ian Goodfellow et al. [16] and is based on deep learning (DL). Its objective is to create adversarial samples from input datasets that are strikingly comparable to the original data. As in a two-player game, a GAN is implemented by two neural networks that challenge each other. It enables a model to gain additional knowledge from the available data and strives to replicate a data distribution. The generator receives a random number (random noise) as input, generates samples that are comparable to those in the dataset, and then sends those samples to the discriminator, which analyzes them and determines whether they are samples of genuine data or generated data [17]. The discriminator gains knowledge of original data properties and, using this understanding decides how to interpret the data that the generator passes to it. The generator constantly enhances its adversarial samples to make it harder for the discriminator to recognize them. The discriminator strives to gain more knowledge and accurately recognize the adversarial samples that the generator introduces. The capabilities of GAN are subject to several restrictions. One of them is the vanishing gradient problem when the generator reaches saturation and can no longer generate new samples to deceive the discriminator. In other words, the discriminator correctly identifies samples produced by the generator with high confidence values, leaving the generator with no gradient [18]. By carefully balancing the training between the generator and discriminator networks and ensuring that the discriminator is not overtrained, this problem can be avoided. To prevent such problems, it is essential to make sure that the discriminator is trained to the best possible level for each iteration of the generator training. In [19] , some strategies for stabilizing GAN training are addressed and examined.

C. CLASSIFICATION ALGORITHMS

Classification algorithms are essential to the effectiveness of ML-based models. The algorithms in this work were therefore chosen based on the level of expertise they represented to the scientific community, particularly in handling datasets with various classes:

a) Decision Trees: The core capability of the Decision Tree (DT) method is the efficient classification of data into logical trees composed of leaf nodes, root nodes, and branchlike structures with nodes [20].

b) Random Forest: The Random Forest (RF) classifier is a well-known and effective ensemble learning technique that consists of several DTs that have been trained on various dataset subsets. In addition, different data patterns affect the features that each tree takes into account while making decisions. To make the final judgment on an input, all individual tree's predictions are taken into account and averaged [21]

c) Support Vector Machines: Support Vector Machines (SVMs) are a traditional technique that successfully manages huge data applications. Even though this technique requires sophisticated and time-consuming computations, we chose it because of how well it handles non-linear classification issues [22] .

d) Extreme Gradient Boosting (XGBoost): The XGBoost technique provides a way to create K Classifications and Regression Trees as well as Gradient Boosting Machines. The algorithm is built on the concept of "boosting," which combines all of the predictions made by a group of "weak" learners to create a "strong" learner through additive training techniques. In addition to avoiding over-fitting, XGBoost optimizes processing resources. This is achieved by condensing the objective functions that permit the combination of regularization and prediction terms while still maintaining a perfect computation speed [23].

D. LEARNING-BASED CLASSIFIER FOR ANDROID MALWARE DETECTION

The issue with learning-based classifiers for Android malware detection can be expressed in the form $f : X \longrightarrow Y$, which uses the function f to assign the label $y \in Y$ to an input app $x \in X$. Thus, the following may be written as a general linear classification model for Android malware detection:

$$g = sign(f(X)) = sign(X^T w + b) \tag{1}$$

Each column of matrix X represents the feature vector of an app, g is a vector whose elements are each the harmful or benign label of an app to be predicted, w is the weight vector, and b is the bias. A learning-based classifier can be described as an optimization problem in more detail:

$$argmin_{g,w,b} L(y, g) + \beta||w|| + \gamma||b|| \tag{2}$$

with $R_f = \beta||w|| + \gamma||b||$.

Equation (2) is subject to Equation (1), where R_f is a regularization term to prevent overfitting, β and γ are the regularization parameters, and y is the labeled information vector, L(y,f), a loss function. The classifier in Equation 2 is an example of a learning-based classifier. Depending on the choice of the loss function and regularization variables, the equation can be translated without losing generality into several learning models[15].

III. RELATED WORK

A. ADVERSARIAL EXAMPLE

Many ML algorithms are quite susceptible to malicious attacks. If ML-based malware detection algorithms may be easily defeated by specific adversarial approaches, they cannot be applied in real-world applications. Deep learning adversarial examples have attracted the interest of many academics. Adversarial examples can be referred to as examples that have perturbations added to them. To create adversarial examples, Szegedy et al. [5] applied imperceptible perturbations to the images to increase a trained neural network's classification mistakes and prevent the network from accurately classifying the images. Goodfellow et al. [6] suggested a gradient-based approach. Papernot et al. [24] employed the Jacobian matrix to decide which features to change when generating adversarial examples. Grosse et al. [25] recommended using a gradient-based technique to craft adversarial Android malware instances.

B. GAN FOR ANDROID

MalGAN [12] is an adversarial malware attack technique for computers. MalGAN is appropriate for Android because of the significant degree of similarity between PC software and Android applications. To produce harmful adversarial malware instances, the generator uses noise and malicious samples as input. In addition to MalGAN, a variety of other adversarial example-generating techniques have emerged in recent years. To attack the model with adversarial defensive capabilities, E-MalGAN [26] modified the structure of MalGAN by introducing a new substitute detector that learns the adversarial examples detector in the detection system. In a series of API calls, Peng et al. [27] create adversarial examples using a Long Short-Term Memory networks generator and a Convolutional Neural Network as a stand-in model. By applying Least Square (LS) loss to improve border examples and generate smoother adversarial examples, Wang et al. [28] proposed LSGAN-AT to optimize the network topology of the GAN. Many of these techniques for creating malware adversarial examples involve replacement models to convert black-box detectors into white boxes to generate gradients. A significant ML idea that makes use of the transferable qualities of ML is substitute models. A model called p-MalGAN, a MalGAN with predictive capability, was proposed in [13]. They also take into account the issue that throughout the development of these methods, both the detector and the attacker employ the same features. Li et al. [26] suggested an approach based on bi-objective GANs to produce an adversarial example attack strategy against Android malware classifiers. Through the usage of features based on intents, Salman et al. [29] leveraged GANs to bolster the security of Android malware detectors. On Android malware classifiers, Taheri et al. [30] tested four different evasion attack models and developed a countermeasure using GANs. According to the authors of [30], GAN-based techniques increase the evasion detection of Android malware by up to 50%. By utilizing GANs, Millar et al.[31] introduced DanDroid, a model to categorize malicious and benign Android applications that are both obfuscated and unobfuscated. They used the classifier-two sample test (C2ST) to assess the accuracy and expectancy loss of the discriminator as well as the generator of the GAN. In reality, malicious data was fed to GANs to produce artificial data that resembled the actual malicious programs. Many of these works investigated how to deceive malware classifiers by using GAN. In our work, we use a specific generative adversarial network, Wasserstein GAN, to prevent a particular evasion attack, the Carlini-Wagner attack.

IV. ATTACK MODEL AND DATASET

A. ATTACK MODEL

The integrity of a classifier is intended to be compromised by evasion attacks that are either targeted (misclassifying a specific group of input) or untargeted (misclassifying any sample) [32]. The understanding of an adversary can be partial or full on the training data, feature set, learning algorithm, parameters, and hyperparameters for the target classifier. The capabilities of an adversary determine how they can take advantage of the classifier during training or testing [32] or what obstacles they can surmount during sample perturbation [33]. Attack utilizing feature selection: Grosse et al. [10] produce adversarial samples of Android malware that target a feed-forward neural network classifier using a feature selection method. Consider an Android application that has M dimensions and is represented by the binary vector x $\in \{0, 1\}^M$. They suggest only adding features to the AndroidManifest.xml file and limiting the total amount of features introduced to address the limits for malware disturbance. The approach for creating adversarial samples based on forward derivatives is suggested by [24]. They first calculate the gradient, also known as the forward derivative, of the classifier F with respect to the input X. They next choose a perturbation that maximizes the gradient with respect to X. For the altered sample to behave like the original sample, the perturbation should ideally be modest. This limitation is simpler to examine in an image. Malware samples, on the other hand, are represented by discrete and binary features; a feature is either present or not. Therefore, to apply this constraint, they modify one X feature from 0 to 1 based on the forward derivative information, and they repeat the procedure until there has been a maximum number of feature changes or the classifier has produced a misclassification.

The Carlini-Wagner Attack: A targeted evasion attack, sometimes known as a CW attack, was proposed by Carlini and Wagner [8] as a means of overcoming the common defense technique known as defensive distillation. The majority of defensive strategies were rendered useless by this white-box attack, which became more effective than many other white-box attack techniques in the research community. The fundamental element of an adversarial attack algorithm is the formulation of an optimization problem that results in misclassification. CW attack [7] is presented in the following ways:

$$min||\delta||_2^2 + c.f(X + \delta) \tag{3}$$

where $f(X^{'}) = max[max_{i \neq t} Z(X^{'})_t - Z(X^{'})_i, -k]$. f is the best objective function. where c > 0 is constant. Z is the set of input objects (apps), and z $\in$ Z is a sample. δ is a perturbation vector. The parameter k encourages the solver to find an adversarial instance $X^{'}$ that will be classified as class t with high confidence. The value of a feature must fall within the range of 0 and 1 according to the first constraint, X + δ $\in$ $[0; 1]^n$. The norm of the features in each family or package of the API call is restricted by the second constraint, which is presented as follows:

$||X_g + \delta_g||_1 = 1, g \in 1...k$, additionally, it should be 1. The optimization variable is changed from δ to w since it is being done to add more API calls, and w is defined as follows:

$\delta_i^g = \frac{a_{i_g} + w_i^g}{a^g + w^g} - \frac{a_{i_g}}{a^g}$ (4)

where $w^g = \sum_i w_{i^g}$ and a_{i_g} represents all of the API calls made by the i^{th} feature in the g^{th} group. They limit themselves to adding API calls.

B. DEFENSE METHODOLOGY

C. DATASET OVERVIEW

To represent each gathered Android app, we initially extract the features and transform them into a vector space, which can then be input to the classifier for either training or testing. Permissions (S1), filtered intents (S2), and application attributes (S3) from manifest files, API calls (S4), new-instances (S5), and exceptions (S6) from dex files are among these six sets of features. We define our dataset K as having the form K = $\{x_i, y_i\}_{i=1}^n$ of n apps, where x_i is the characteristics extracted from app i, and y_i is the class label of app i $y_i \in \{0, 1\}$, 1:malicious, 0: benign. Let n_f be the total number of features in dataset K's S_{1-6}. Consequently, each program can be represented by a few binary feature vectors. If a feature is associated with app j, then $x_{ij} = 1$ or $x_{ij} = 0$, and $x_i \in \{0, 1\}^{n_f}$.

We use Drebin [34] as a benchmark dataset in this investigation. The dataset contains 9476 benign applications and 5560 malicious ones. In addition, we extract the Java source code from the Android application packages (APKs) in the dataset

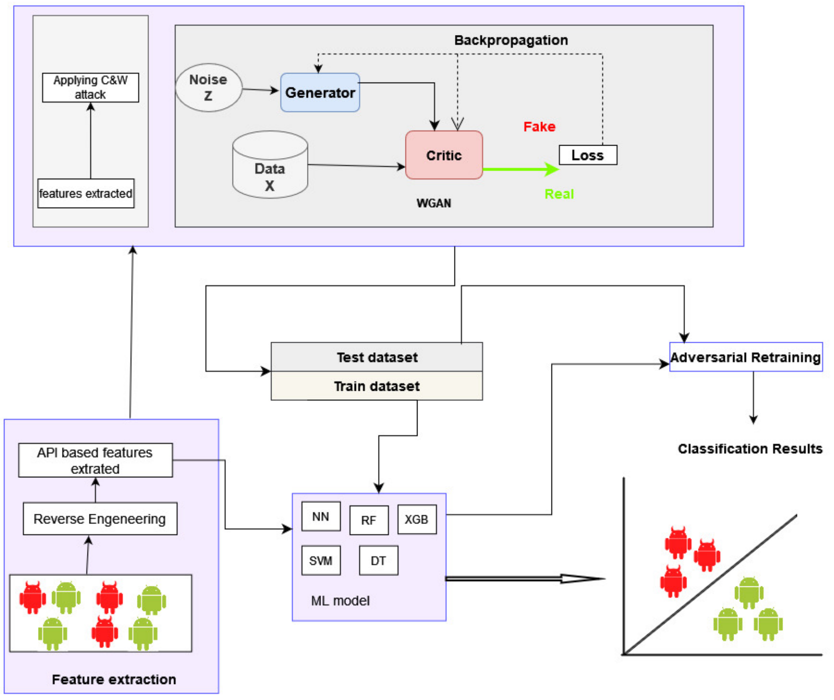

FIGURE 2. This figure presents a general description of the model. The part on feature extraction shows how we obtain features from Android apps. Thereafter, the Wasserstein Generative Adversarial Network is used on the training dataset to teach the Critic model how to detect adversarial samples. Then we apply the Carlini-Wagner attack to obtain adversarially infected samples. Finally, the classifiers DT, RF, SVM, Neural Network, and XGBoost are used to show the classification results.

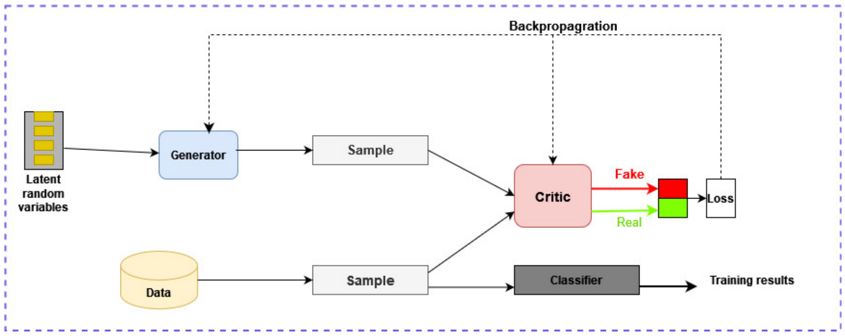

FIGURE 3. Training Phase of the model. Here critic model receives a generated sample from the generator and a training sample; it learns the difference between a real and fake sample. By doing backpropagation, it permits the generator to generate a realistic sample. In the end, the generator and the critic reach the equilibrium point where they cannot increase their ability to generate and differentiate between real and fake anymore. The normal training sample is classified by the classifier to see the classification results without adversarial attacks.

by reverse engineering them. APKs are first decompiled into .dex files, which are subsequently converted into .jar files. To extract features, the .jar files are then disassembled into Java source code.

V. PROPOSED APPROACH

A. PROBLEM DEFINITION

Let the feature space be $X \in R^{n_f}$ and the number of features is n_f. Assume that $(x_i; y_i)$ is the i^{th} instance in the training set, which consists of true class labels $y_i \in Y$ and feature vectors $x_i \in X$ produced by an unidentified distribution

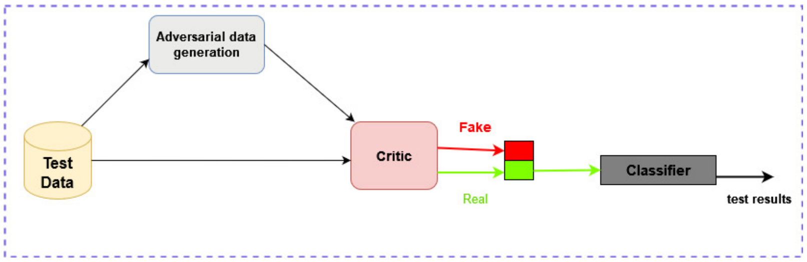

FIGURE 4. Test phase of the model. We apply the Carlini-Wagner attack on the testing samples to obtain the adversarial sample, and we pass them through our critic model for removing the adversarial samples, which is already trained to do that. Thereafter, we pass them through classifiers to see the classification results.

$x_i \sim P_{data}$. the goal of the learning system is to learn a classifier $f : X \rightarrow Y$ from the domain X to the set of classification outputs Y, where $|y|$ represents the variety of classification outputs that may be possible. The objective of an adversary is to provide an adversarial example x_A, which can be either $f(x_A) = t$ (targeted attack), where t is the target class, or $f(x_A) \neq y$ (untargeted attack), where y is the true label.

1) General representation

Fig. 2 highlights the major components of the system and illustrates the proposed approach to Carlini-Wagner attacks. In the feature extraction module, we reverse-engineer the Android applications to extract API-based features. The collected features are further used to train several ML classifier models. The extracted features are then used for training numerous ML classifier models. To circumvent the trained classifiers, we generate adversarial data based on Wasserstein GAN in the adversarial samples generation module. The adversarial samples are subsequently tested on the existing pre-trained classifiers. Finally, we perform adversarial training to improve the security of Android malware classifiers against the adversarial CW attack. First, employing each of the selected classification methods on the original dataset in non-adversarial situations, the model performance baseline is assessed. After that, the CW attack is used to evaluate the model. The phases of training and testing are shown in Fig.3 and Fig.4, respectively.

2) Generator model

The generator captures the data distribution and generates an adversarial example. It uses malicious samples and noise as input for generating adversarial examples of malware, and it aims to have the detector misclassify adversarial instances as much as possible.

$$\mathfrak{L}_{gen}(w) = min_{\theta} - \mathbb{E}_{z \sim Z}[f_w(g_{\theta}(z))] \quad (4)$$

3) Critic model

The Critic model predicts whether the input sample is from the training data or the generator, and it differentiates whether the sample is generated by the generator as accurately as feasible. The objective function of the critic is as follows:

$$\mathfrak{L}_{critic}(w) = max_{w \in \mathbf{W}} \mathbb{E}_{x \sim P_r}[f_w(x)] - \mathbb{E}_{z \sim P_z}[f_w(g_{\theta}(z))] \quad (5)$$

Here, $\mathbb{E}$ stands for the probability estimation; z and x are the noise and real samples, respectively, while P_r and P_z represent the probability distributions of real and noise data.

4) Algorithm Wasserstein GAN adversarial example (WGAE)

Algorithm 1 WGAE algorithm

```
Require: $\Sigma$, classifier[1,2,3,4,5], epochs, batches
Ensure: Result
 1: $\Sigma_{train} \leftarrow 70\%$
 2: $\Sigma_{test} \leftarrow 30\%$
 3: build generator G and critic model C
 4: for k in epoch do
 5:     for q in batches do
 6:         $x_k \leftarrow$ element of$(\Sigma_{train})$
 7:         $z_q \leftarrow$ generate_noise_sample$(0, 1, batch\_size)$
 8:         $W_{C_q} \leftarrow W_{C_q} - \alpha \nabla W_{C_{qL(X_k)}}$
 9:         $W_{C_q} \leftarrow W_{C_q} - \alpha \nabla W_{C_{qL(G_{z_q})}}$
10:         $W_{G_q} \leftarrow W_{G_q} - \alpha \nabla W_{G_{qL(Z_q)}}$
11:     end for
12: end for
13: $\Sigma_{adver} \leftarrow$ CW_attack$(\Sigma_{test})$
14: $\Sigma_T \leftarrow \Sigma_{adver} \cup \Sigma_{test}$
15: for j in [1,2,3,4,5] do
16:     Result_normal[j] $\leftarrow$ train_classifier[j]$(\Sigma_{train})$
17:     Result_adver[j] $\leftarrow$ [j]$(\Sigma_T)$
18:     Result_final[j] $\leftarrow$ [j]$(C(\Sigma_T))$
19: end for
20: Result$[1, 2, 3] \leftarrow$ Result_normal, Result_adver, Result_final
21: return Result
```

The WGAE algorithm shows the pseudocode of the model. It takes as input, Σ set of samples, the five classifiers, epochs, and batches. As output, it returns Result aims to output all the results, like accuracy, Recall, etc. At lines 1 and 2, we split the set of samples into Σ_{train} and Σ_{test}. Line 3 aims to construct the generator and the critic models, as explained in Sections V-A3 and V-A2. From line 4 to 11, we have described the process of generating fake samples that mimic real samples by teaching the critic model how to detect fake samples. At line 13, we generate adversarial samples Σ_{adver} by running the CW attack. Line 14 shows the combination of adversarial and test samples. We get the results of normal classification without adversarial samples of the five classifiers at line 16. Line 17 gives us the results with adversarial samples. In line 18, we get the result with our model on infected samples. In line 20, we get the results from classifiers after applying WGAN to remove adversarial examples. In the end, we return the results.

VI. RESULTS AND PERFORMANCE EVALUATION

A. GENERATOR CONFIGURATION

The generator is made up of a five-layer NN with a random noise input layer of size 216 and three inner layers, and uses the ReLU activation function for learning the real data distribution. It has an output layer that produces a data sample of size 216, which is similar to the original data. A generator loss function is calculated based on the prediction outcomes produced by the discriminator and fed back to the generator for it to elevate in a way to minimize the loss value.

B. CRITIC OR DISCRIMINATOR CONFIGURATION

The critic network contains an input layer that accepts inputs from the generator, the training dataset during the learning phase, and four hidden layers with the ReLU activation function. The output result shows how confident the critic is about the sample being real or fake.

C. EXPERIMENTAL RESULTS AND ANALYSIS

In this section, we evaluate the performance of different ML classifiers for normal classification, against the CW attack and performance after applying Wasserstein GAN. ' The PC that was used for the experiment has the following specifications: Ubuntu 22.04 64-bit, Intel Xeon E5-2686 v4 (Broadwell), and 61 GB RAM. The software specifications are Python 3.6.5, Tensorflow 1.13.2, Scikit-learn 0.24.2, and Keras 2.1.5. The CW attack was implemented using the Cleverhans 3.0.1 library [23]

All the results are shown in the Table 1.

In case of no adversarial attacks, we train the LSVM, NN, RF, XGB, and DT classifiers on default hyper-parameters settings with a distribution of 70% train and 30% test set. Fig. 5 presents the normal classification without adversarial attack: DT has 93.88% of classification, RF has 96.45%, LSVM achieves 96.47%, XGB obtains 96.91%, and NN achieves 97.55%.
Next, we performed the Carlini-Wagner attack and got the results shown in Fig.6: DT becomes 92.46%, RF becomes 96.01%, XGB gives now 96.11% and NN gives 77.55%. When we apply our model we get the results shown in Fig. 7: DT gives 93.07%, LSVM becomes: 96.47% RF gives 96.07%, XGB becomes 96.91 and NN gives 97.21%.

Table 2 presents a comparison of our results on neural networks (NN) with the work of Taheri et al. [30]. The first column provides the basis for comparison, the second column displays the results without adversarial attacks, the third column shows the outcomes under the Carlini-Wagner (CW) attack, and the fourth column presents the results after applying the adversarial detection model. Our findings indicate that our model outperforms the better.

Fig. 8 presents classifier accuracy. The first four bands show the baseline of classification without adversarial attack: RF is 93.79%, LSVM gives 96.89%, RF gives 96.67%, XGB gives 97.23%, and NN gives 97.55%. Thereafter, with adversarial attack DT becomes 92.46%, LSVM becomes 75.86%, RF becomes 88.15%, XGB becomes 96.23% and NN becomes 73.53%. When we apply our method RF gives now 93.45%, LSVM gives 96.89, RF gives 96.45% XGB gives 97.23% and NN gives 97.67%.

Fig. 9 presents classifier Recall. For classification without adversarial attack: DT gives 94.29%, LSVM gives 96.89%, RF gives 96.89%, XGB gives 97.23 and NN gives 97.67%. With adversarial attack DT becomes 71.46% LSVM becomes 56.93%, RF becomes 88.43% XGB becomes 96.23% and NN becomes 73.53%. After applying Our model we obtain 93.78% for DT, 96.86% for LSVM, 96.78% for RF, 97.23% for XGB and 97.67% for NN.

Figure 10 illustrates the neural network (NN) confusion matrix results for normal classification, yielding 99% true negatives (TN), 96% true positives (TP), 4% false negatives (FN), and 1% false positives (FP).

Figure 11 presents the NN confusion matrix results under the Carlini-Wagner (CW) attack, indicating 85% TN, 74% TP, 15% FN, and 26% FP.

Figure 12 displays the NN confusion matrix results after applying our detection model, showing 98% TN, 96% TP, 4% FN, and 2% FP.

TABLE 1. The different classification results

	Normal classification					Classification with CW attack					Result with our method				
	DT	**LSVM**	**RF**	**XGB**	**NN**	**DT**	**LSVM**	**RF**	**XGB**	**NN**	**DT**	**LSVM**	**RF**	**XGB**	**NN**
F1-score (%)	**94.29**	**96.89**	**96.89**	**97.20**	**97.60**	**71.40**	**56.93**	**88.43**	**96.23**	**73.53**	**94.18**	**96.89**	**96.45**	**97.23**	**97.67**
Recall (%)	**93.35**	**96.89**	**96.89**	**97.20**	**97.60**	**71.40**	**57.47**	**88.43**	**96.23**	**73.53**	**93.78**	**96.86**	**96.78**	**97.23**	**97.67**
AUC(%)	**93.88**	**96.47**	**96.47**	**96.90**	**97.50**	**92.40**	**75.86**	**96.50**	**96.11**	**77.55**	**93.07**	**96.47**	**96.47**	**96.91**	**97.12**
Accuracy (%)	**93.79**	**96.89**	**96.86**	**97.20**	**97.60**	**75.40**	**58.86**	**88.15**	**96.23**	**73.53**	**93.45**	**96.89**	**96.89**	**97.23**	**97.67**

TABLE 2. Our result compare to Taheri et al. [30] work

	Normal classification		Classification with CW attack		**Taheri et al.[30]**	Result with our method
	Taheri et al.[30]	**NN**	**Taheri et al.[30]**	**NN**		**NN**
F1-score (%)	**97.45**	**97.60**	**72.85**	**73.53**	**72.85**	**97.67**
Recall (%)	**97.45**	**97.60**	**72.85**	**73.53**	**72.85**	**97.67**
AUC(%)	**97.15**	**97.50**	**66.61**	**77.53**	**66.61**	**97.12**
Accuracy (%)	**97.45**	**97.60**	**72.85**	**73.53**	**73.53**	**97.67**

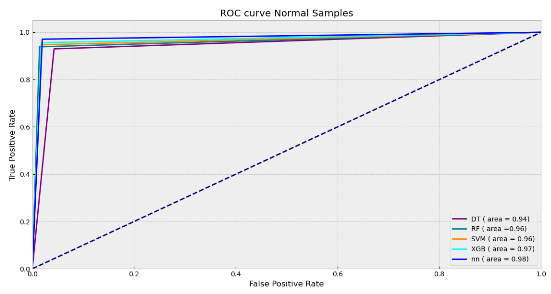

FIGURE 5. ROC curve classification without C&W attack.

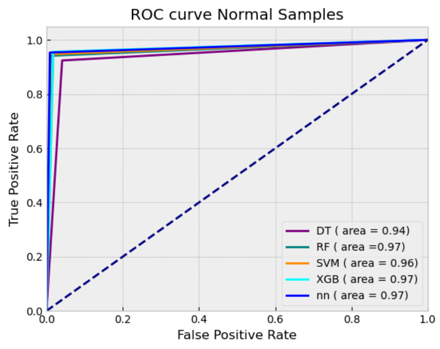

FIGURE 7. ROC curve with with our model.

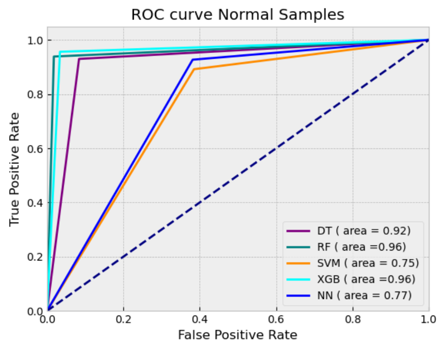

FIGURE 6. Accuracy of classification.

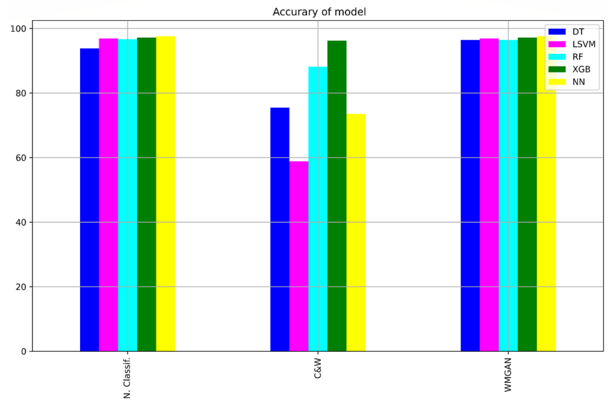

FIGURE 8. Accuracy of classification.

VII. CONCLUSION

In this paper, we presented a method to detect white-box adversarial attacks. We principally focus on the Calini-Wagner attack. Our proposed model leveraged the logic of Wasserstein GAN to build a robust model against this attack. We

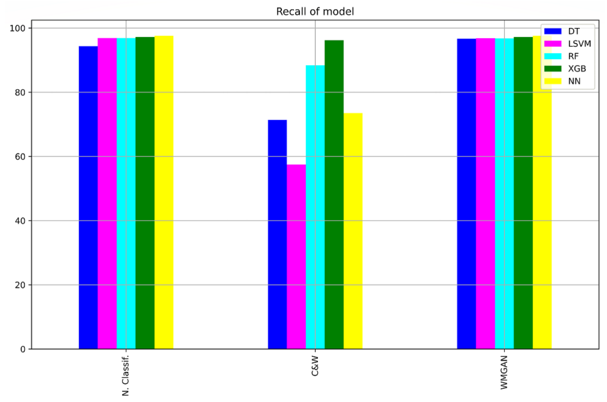

FIGURE 9. Recall of classification.

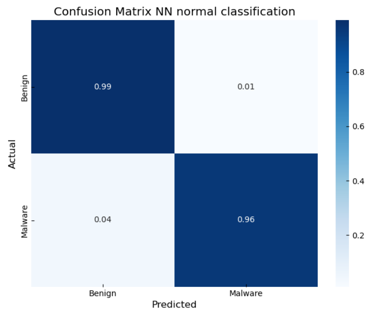

FIGURE 10. Confusion matrix NN normal classification.

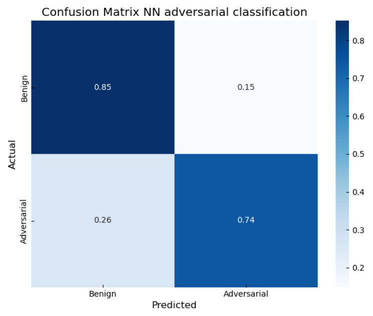

FIGURE 11. Confusion matrix NN under adversarial samples.

generated adversarial by using CW attacks, and we then used them to train our model. The experience is done by extracting features from malware and benign apps from the Drebin dataset. The results showed that our model can detect this adversarial attack with good performance. In future work we are planning to look into another scenario of adversarial attacks: first, The adversarial attack can be crafted on the attacker model with partial knowledge of the target model and transferred to the defender model. Second, an attack can also be built in a black box scenario where an attacker does not have knowledge about the target model.

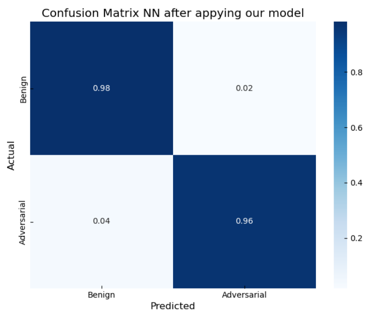

FIGURE 12. Confusion matrix NN with adversarial detection model.

ACKNOWLEDGMENT

Research was sponsored by the Army Research Office and was accomplished under Grant Number W911NF-21-1-0326. 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] Marco Barreno, Blaine Nelson, Anthony D Joseph, and J Doug Tygar. The security of machine learning. *Machine Learning*, 81:121–148, 2010.

[2] Ambra Demontis, Marco Melis, Battista Biggio, Davide Maiorca, Daniel Arp, Konrad Rieck, Igino Corona, Giorgio Giacinto, and Fabio Roli. Yes, machine learning can be more secure! a case study on android malware detection. *IEEE transactions on dependable and secure computing*, 16(4):711–724, 2017.

[3] Yanfang Ye, Tao Li, Donald Adjeroh, and S Sitharama Iyengar. A survey on malware detection using data mining techniques. *ACM Computing Surveys (CSUR)*, 50(3):1–40, 2017.

[4] Deqiang Li, Qianmu Li, Yanfang Ye, and Shouhuai Xu. A framework for enhancing deep neural networks against adversarial malware. *IEEE Transactions on Network Science and Engineering*, 8(1):736–750, 2021.

[5] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013.

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014.

[7] Nicholas Carlini and David Wagner. Towards evaluating the robustness of neural networks. In *2017 ieee sympo-*

sium on security and privacy (sp), pages 39–57. Ieee, 2017.

[8] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. On the (statistical) detection of adversarial examples. *arXiv preprint arXiv:1702.06280*, 2017.

[9] Jack W Stokes, De Wang, Mady Marinescu, Marc Marino, and Brian Bussone. Attack and defense of dynamic analysis-based, adversarial neural malware detection models. In *MILCOM 2018-2018 IEEE Military Communications Conference (MILCOM)*, pages 1–8. IEEE, 2018.

[10] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial examples for malware detection. In *Computer Security–ESORICS 2017: 22nd European Symposium on Research in Computer Security, Oslo, Norway, September 11-15, 2017, Proceedings, Part II 22*, pages 62–79. Springer, 2017.

[11] Wei Yang, Deguang Kong, Tao Xie, and Carl A Gunter. Malware detection in adversarial settings: Exploiting feature evolutions and confusions in android apps. In *Proceedings of the 33rd Annual Computer Security Applications Conference*, pages 288–302, 2017.

[12] Weiwei Hu and Ying Tan. Generating adversarial malware examples for black-box attacks based on gan. In *International Conference on Data Mining and Big Data*, pages 409–423. Springer, 2022.

[13] Yanping Guo and Qiao Yan. Android malware adversarial attacks based on feature importance prediction. *International Journal of Machine Learning and Cybernetics*, 14(6):2087–2097, 2023.

[14] Shifu Hou, Aaron Saas, Lifei Chen, and Yanfang Ye. Deep4maldroid: A deep learning framework for android malware detection based on linux kernel system call graphs. In *2016 IEEE/WIC/ACM International Conference on Web Intelligence Workshops (WIW)*, pages 104–111. IEEE, 2016.

[15] Lingwei Chen, Shifu Hou, Yanfang Ye, and Shouhuai Xu. Droideye: Fortifying security of learning-based classifier against adversarial android malware attacks. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 782–789. IEEE, 2018.

[16] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

[17] Simon Msika, Alejandro Quintero, and Foutse Khomh. Sigma: Strengthening ids with gan and metaheuristics attacks. *arXiv preprint arXiv:1912.09303*, 2019.

[18] Samuel A Barnett. Convergence problems with generative adversarial networks (gans). *arXiv preprint arXiv:1806.11382*, 2018.

[19] Maciej Wiatrak, Stefano V Albrecht, and Andrew Nystrom. Stabilizing generative adversarial networks: A survey. *arXiv preprint arXiv:1910.00927*, 2019.

[20] Yan-Yan Song and LU Ying. Decision tree methods: applications for classification and prediction. *Shanghai archives of psychiatry*, 27(2):130, 2015.

[21] Mariana Belgiu and Lucian Drăguţ. Random forest in remote sensing: A review of applications and future directions. *ISPRS journal of photogrammetry and remote sensing*, 114:24–31, 2016.

[22] Shan Suthaharan. Machine learning models and algorithms for big data classification. *Integr. Ser. Inf. Syst*, 36:1–12, 2016.

[23] Odey Alshboul, Ali Shehadeh, Ghassan Almasabha, and Ali Saeed Almuflih. Extreme gradient boosting-based machine learning approach for green building cost prediction. *Sustainability*, 14(11):6651, 2022.

[24] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. The limitations of deep learning in adversarial settings. In *2016 IEEE European symposium on security and privacy (EuroS&P)*, pages 372–387. IEEE, 2016.

[25] Kathrin Grosse, Nicolas Papernot, Praveen Manoharan, Michael Backes, and Patrick McDaniel. Adversarial perturbations against deep neural networks for malware classification. *arXiv preprint arXiv:1606.04435*, 2016.

[26] Heng Li, ShiYao Zhou, Wei Yuan, Jiahuan Li, and Henry Leung. Adversarial-example attacks toward android malware detection system. *IEEE Systems Journal*, 14(1):653–656, 2019.

[27] Xiaowei Peng, Hequn Xian, Qian Lu, and Xiuqing Lu. Generating adversarial malware examples with api semantics-awareness for black-box attacks. In *Security and Privacy in Social Networks and Big Data: 6th International Symposium, SocialSec 2020, Tianjin, China, September 26–27, 2020, Proceedings 6*, pages 52–61. Springer, 2020.

[28] Jianhua Wang, Xiaolin Chang, Yixiang Wang, Ricardo J Rodríguez, and Jianan Zhang. Lsgan-at: enhancing malware detector robustness against adversarial examples. *Cybersecurity*, 4:1–15, 2021.

[29] Salman Jan, T Ali, A Alzahrani, and S Musa. Deep convolutional generative adversarial networks for intent-

based dynamic behavior capture. *International Journal of Engineering & Technology*, 7(4.29):101–103, 2018.

[30] Rahim Taheri, Reza Javidan, Mohammad Shojafar, P Vinod, and Mauro Conti. Can machine learning model with static features be fooled: an adversarial machine learning approach. *Cluster computing*, 23:3233–3253, 2020.

[31] Husnain Rafiq, Nauman Aslam, Biju Issac, and Rizwan Hamid Randhawa. An investigation on fragility of machine learning classifiers in android malware detection. In *IEEE INFOCOM 2022-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 1–6. IEEE, 2022.

[32] Octavian Suciu, Radu Marginean, Yigitcan Kaya, Hal Daume III, and Tudor Dumitras. When does machine learning {FAIL}? generalized transferability for evasion and poisoning attacks. In *27th USENIX Security Symposium (USENIX Security 18)*, pages 1299–1316, 2018.

[33] Fabio Pierazzi, Feargus Pendlebury, Jacopo Cortellazzi, and Lorenzo Cavallaro. Intriguing properties of adversarial ml attacks in the problem space. In *2020 IEEE symposium on security and privacy (SP)*, pages 1332–1349. IEEE, 2020.

[34] Daniel Arp, Michael Spreitzenbarth, Malte Hubner, Hugo Gascon, Konrad Rieck, and CERT Siemens. Drebin: Effective and explainable detection of android malware in your pocket. In *Ndss*, volume 14, pages 23–26, 2014.

FABRICE SETEPHIN ATEDJIO is a researcher in artificial intelligence applied to cybersecurity. He is currently a doctoral researcher at the University of Dschang, Cameroon. He received his bachelor's degree in mathematics and computer science in 2017. Two years later, he obtained his master's degree in computer science where he worked in IoT security. He was selected to join the "Game Theory and Machine Learning for Cyber Deception, Resilience, and Agility (GMC-DRA)" project in 2021, sponsored by the U.S. Army Research Office. He is certified in professional cybersecurity and data analysis by Google. He is also certified in Deep learning by DeepMind. He obtained the best poster Award at the International Conference on Machine Learning, and Indabax Cybersecurity in 2022.

JEAN-PIERRE LIENOU is an Associate Professor in the Department of Computer Engineering at the University of Dschang. He earned his Master of Science in System Engineering from the Kiev Polytechnic Institute (National Technical University of Ukraine) and his PhD from the University of Yaoundé I in Cameroon in 2011. Before his academic career, he worked as a Maintenance Engineer at 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. Currently, he serves as the Head of the Department of Computer Engineering at the College of Technology, University of Bamenda. In addition to his academic responsibilities, he has been an active member of the EANG (Engineering Association of the National Group) since 2018. His research interests are diverse and 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 NELSON is a Researcher and the Program Lead with the 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 to 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 SHETTY is an Executive Director of Center for Secure and Intelligent Critical System and Professor with the Department of Electrical and Computer Engineering at Old Dominion University. He obtained his PhD in Modeling and Simulation at the Old Dominion University in 2007. His research interests lie at the intersection of computer networking, network security and machine learning. Within the last 15 years, Sachin 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 4 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 Lab, Office of Naval Research, Department of Homeland Security, and Boeing. He has 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. He is a Senior Member of the IEEE.

CHARLES KAMHOUA received the BS degree in electronics from the University of Douala (ENSET), Cameroon, in 1999, the MS degree in telecommunication and networking and the PhD degree in electrical engineering from the Florida International University (FIU), in 2008 and 2011, respectively. He is a researcher with the Network Security Branch of the U.S. Army Research Laboratory, Adelphi, Maryland 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 the U.S. Air Force Research Laboratory (AFRL), Rome, New York for six years and an educator in different academic institutions for more than 10 years. He has held visiting research positions with the Oxford and Harvard. He has co-authored more than 100 peer-reviewed journal and conference papers. He has been invited to more than 40 keynote and distinguished speeches and co-organized more than 10 conferences and workshops. He has mentored more than 50 young scholars including students, postdocs, 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 for the National Research Council, a member of the FIU alumni association, the ACM, and a senior member of the IEEE.

• • •