IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 29 October 2023, accepted 12 November 2023, date of publication 20 November 2023,
date of current version 1 December 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3334645

RESEARCH ARTICLE

A Deep Reinforcement Learning Framework to Evade Black-Box Machine Learning Based IoT Malware Detectors Using GAN-Generated Influential Features

RAHAT MAQSOOD ARIF[1], MUHAMMAD ASLAM[1], SHAHA AL-OTAIBI[2], (Member, IEEE), ANA MARIA MARTINEZ-ENRIQUEZ[3], TANZILA SABA[4], (Senior Member, IEEE), SAEED ALI BAHAJ[5], AND AMJAD REHMAN[4], (Senior Member, IEEE)

[1]Department of Computer Science, University of Engineering and Technology, Lahore, Lahore 39161, Pakistan
[2]Department of Information Systems, College of Computer and Information Sciences, Princess Nourah bint Abdulrahman University, Riyadh 11671, Saudi Arabia
[3]Department of CS, CINVESTAV-IPN, 07360 Gustavo A. Madero, Mexico
[4]Artificial Intelligence and Data Analytics (AIDA) Laboratory, CCIS, Prince Sultan University, Riyadh 11586, Saudi Arabia
[5]MIS Department, College of Business Administration, Prince Sattam bin Abdulaziz University, Al-Kharj 11942, Saudi Arabia

Corresponding author: Saeed Ali Bahaj (bahajsaeedali@gmail.com)

This work was supported by Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia, through Princess Nourah bint Abdulrahman University Researchers Supporting Project under Grant PNURSP2023R136.

ABSTRACT In the internet of things (IoT) networks, machine learning (ML) is significantly used for malware and adversary detection. Recently, research has shown that adversarial attacks have put ML-based models at risk. This problem is exacerbated in an IoT environment because of the absence of adequate security measures. Consequently, it is crucial to evaluate the strength of such malware detectors using powerful adversarial samples. The existing adversarial sample generation strategies either rely on high-level image features or an unfiltered feature set, making it challenging to determine which feature modifications are crucial in evading malware detection systems, without compromising the malware functionality. This encourages us to propose an evasion framework named IF-MalEvade, based on Generative Adversarial Network (GAN) and Deep Reinforcement Learning (DRL) that effectively generates fully-working, malware samples with several effective perturbations such as header section manipulation and benign bytes insertion. The DRL framework selects a few suitable action sequences to change malicious samples, thus allowing our malware samples to bypass various black-box ML based malware detectors and the detection search engines of VirusTotal, while maintaining the executability and malicious behavior of the original malware samples. The neural networks of GAN take in the unfiltered feature set of malware dataset and using minimax objective function yields a set of useful features that are subsequently used by the DRL agent to make effective changes. Experimental results illustrated that by utilizing the influential features in sequence of transformations, the adversarial samples generated by our model outperformed the state-of-the-art evasion models with an impressive evasion rate. Additionally, the detection rate of well-known machine learning models was also brought down to up to 97%. Furthermore, when the machine learning models were retrained using adversarial samples, a 35% increase in detection accuracy was observed.

INDEX TERMS Generative adversarial network, portable executable PE malware, adversarial attack, malware evasion, deep reinforcement learning, technological development.

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

I. INTRODUCTION

The Internet of Things (IoT) presents a range of innovative and exciting opportunities, consisting of smart cities, smart

homes, intelligent gadgets, and autonomous transportation, to mention a few. According to one estimate, by the year 2025 there will be around 74.44 billion IoT smart devices running with the use of 5G technology [1]. As the IoT industry, scope, and application areas are expanding, but as they do, so do the security risks. These security threats include malware, spoofing, jamming, etc. which have been explored in the related study [2], [3]. According to a research, McAfee Labs saw an average of 688 malware attacks per minute in the first quarter of 2021, up 40 threats per minute (3%) [4]. Similarly, between January 2021 and April 2022, VirusTotal's database had more than a million signed samples that were regarded as suspicious (more than 15% of anti-viruses identified them as dangerous) [5]. By intruding on IoT devices, malware can steal private information from users to extort or sell privacy data for a huge profit. Due to the exponential rise of new malware, traditional signature-based techniques are unable to keep up, researchers are constantly looking for effective malware detection and classification methods. Commercial antivirus firms found machine learning very effective as the artificial intelligence (AI) gains popularity [6], [7]. Researchers have leveraged the power of ML in detecting IoT malware as well, by employing statistical features of a malware such as opcodes [8], dynamic features like system calls [9] and using image domain [10], [11]. Static features are considered to be a better choice, in comparison because dynamic approaches cost more time and computational resources [12].

While these techniques have shown promising malware detection capabilities, Recent studies have revealed that these learning-based algorithms are susceptible to ''adversarial samples'', which are input changes that have been carefully designed [13]. These samples cause the learning-based malware detection models to misclassify the malware file as a benign file with high confidence. According to Symantec, an internet security company, over fifty percent of the new malware samples created each year are essentially variations of already existing malware [14]. Benefitting from this, the malware developers have been able to successfully change the malware to avoid detection by anti-malware systems.

Numerous research has been published in the literature to test the robustness of machine learning-based systems against adversarial samples. Some researchers took the liberty of using gradient based algorithms [15] and genetic algorithms [16], [17], while tremendous work has been conducted on exploiting reinforcement learning [18], [19], [20], [21] to produce modifications in malware files and help them evade detection methods. Anderson et al. [18] were one of the first to show that reinforcement learning (RL) can be successfully used to generate adversarial examples in the problem space for Windows Portable Executable (PE) files by introducing some semantic preserving actions to modify the malware.

The available literature also implies that generative adversarial networks (GAN) are one of the most sophisticated and modern methods for carrying out these attacks [22], [23], [24], [25], [26], [27], [28], [29]. The concept of GAN is based on minimax two player game in which Generator continuously tries to deceive Discriminator by producing fake inputs, while Discriminator is responsible for determining whether or not given samples are genuine [30]. Hu and Tan [24] employed GAN in the malware domain and produced adversarial modification of feature vectors to target black-box malware detectors. In the computer vision domain, researchers [25], converted malware into binary images and implemented Convolutional Neural Network (CNN) along with GAN to manipulate high level features of malware samples. A most recent evasion framework MalFox [29] focused on retaining the originality of the malware samples by producing carefully crafted perturbation path from CNN based GAN model and attacked online antivirus engine VirusTotal. While these methods are successful in evading machine learning malware detectors, the majority of these techniques either use huge [24], [25] and complex feature action space [29] that may compromise the malware's functionality or do not guarantee if the behavior of the generated variants is exactly the same as the behavior of the original malware samples.

The purpose of this study is to assess the effectiveness of combining GAN with DRL framework to create adversarial samples that evade detection by black-box IoT malware detectors. In addition to that, the research seeks to evaluate the robustness of several machine learning based black-box malware detection algorithms and practical antivirus engines while altering the action space of the DRL environment. We believe that developing a malware mutation system with GAN and Deep Reinforcement Learning effectively improves the ability of mutated malware to avoid detection. The contribution of our research is presented as follows.

- This research presents an efficient and novel malware evasion method that uses GAN-generated list of influential features to produce functionality preserving malware samples in order to assess the strength of black-box machine learning based IoT malware detectors.
- Upon adversarial training, the research shows the improvement in the detection rate of those ML based malware detectors against the malware samples.
- This research suggests the integration of GAN with DRL to create the malware, which helps the training process of the model with less and effective modifications.
- This research also applies the created samples to target the commercial engines of VirusTotal and produces better results than state-of-the-art evasion models.

The rest of research paper is organized as shown below. A thorough literature review of earlier research in this field is provided in Section III. The proposed methodology is presented in Section IV, which also discusses the model's architectural philosophy and how influential features help in the process of creating adversarial malware samples. Section V presents the experiments conducted to evaluate the proposed framework along with the experimental setup. The Results and Discussion sections follows in Sections VI and VII. The paper concludes with Section 8.

II. LITERATURE REVIEW

A. MALWARE DETECTION TECHNIQUES

Malware detection on Windows, Android, and the IoT systems has employed the machine learning techniques. Basically, there are three approaches that can be used to detect malware: static method, dynamic method, and hybrid method. Code inspection is an integral part of static malware detection, which includes examining the header fields [31], the opcode sequences [8], [11], application programming interface (API) calls and requested permissions [32], [33] etc. in a malicious file. Yousaf et al. [31] recently conducted a study on Windows malware detection based on static analysis utilizing PE header fields like import dynamic link libraries (DLLs), API calls and applied several ML algorithms to classify the malware. The dynamic detection approach requires a virtual environment in order to execute a malware [12] and inspect its run-time behavior, similar to the reinforcement learning-based malware detection method proposed by Muhammadkhani and Esmaeilpour [34] using pattern based behavioral dynamic features. The hybrid approach has also been researched as a way to distinguish malware from regular file by integrating both static and dynamic malicious features [35], [36]. However, the researchers have carried out all three approaches over the last decade and found the static method to be more preferrable since the dynamic approach requires more time and computational resources [12]. In addition, analyzing dynamic features requires a suitable virtual environment, which entails a risk. Computer vision domain has also been probed in malware detection [10] where malware binary files are converted to grey scale images and CNN is trained to classify the malware families.

Studies on feature selection have been conducted, using evolutionary algorithms [37] and reinforcement learning [38], to extract a set of reduced significant features that can successfully boost the accuracy of detection models, which in turn improves the performance and resource overhead of malware detection approaches. If incorporated into the creation of a malware detection model, these automatic feature selection algorithms produce good detection results [38].

B. MALWARE EVASION TECHNIQUES

While much attention has been paid to malware detection, researchers have also developed some testing frameworks to verify the accuracy of those detection models. White box attacks have been conducted on the models in the form of gradient based techniques such as the fast gradient sign method (FSGM) [15], [39]. Black box attacks, on the other hand, are given more significance because like a real-world scenario, the attacker doesn't have access to the method's internal parameters or training data and we too aim to target the IoT malware detectors in a black-box setting, requiring only the detector's ability to provide malicious/benign labels. Evolutionary algorithms and graph-based augmentation have been investigated by the authors of [16], [40], and [41] for their potential to generate adversarial cases for IoT systems. Anderson et al. [18] suggested using deep reinforcement learning to produce adversarial examples for PE malware and evade black-box machine learning models. Inspired from Anderson's work, Fang et al. [19] also presented a deep learning model called Deep Q-network anti-malware Engines Attacking Framework (DQEAF) to generate adversarial samples against black-box detectors, which comprises of an AI agent and a target model. First, a short list of actions (file mutations), such as modifying PE headers, inserting overlay bytes, packing, and unpacking is defined. The agent then decides what to do next in accordance with an action policy (i.e. Epsilon Greedy policy) and the current environmental state. OpenAI gym was used as the framework for the environment which allowed four functionality preserving actions. In comparison with white and grey box attacks, this study showed decent results. Song et al. [20] put forth another RL based malware evasion model named MAB-MALWARE using multiple armed bandit problem which basically explored the balancing of exploration and exploitation in the model. Labaca-Castro et al. [21] presented RL based AIMED-RL model which was based on Anderson's feature space and applied Double Deep Q learning agent with penalty to diversify the modification. While these models produced substantial results in evading the ML based malware detection models, the researchers worked with unfiltered import DLLs and sections from the PE header which may include both influential and non-influential features in the training process and makes the training of the RL agent unnecessarily difficult.

Generative adversarial networks (GANs) were first proposed by Goodfellow et al. [30] in 2014 to generate adversarial examples for image files. The generator and discriminator neural networks, collectively make up the GAN model, while simultaneously training using a min-max objective function. The generator aims to produce realistic samples to trick the discriminator while the discriminator attempts to classify both the generated and real data accurately. Due to the overwhelming success of these GAN models, researchers have moved their focus to employing GAN to target the machine learning based malware detection models. MalGAN is a GAN-based technique that was presented by Hu and Tan [24] to create adversarial malicious instances that are able to evade black-box detectors. The MalGAN feeds the GAN network with import feature vector of malicious binaries in order to generate such adversarial vectors which are capable of fooling ML detection models. The model is equipped with a substitute detector and a generator as well as a black-box malware detector. The black-box detector is fitted by the substitute detector, whereas the generator further modifies malware file to reduce its likelihood of detection. However, the issue with GAN is that due to the enormous number of features that are provided to GAN, the learning process is extremely unstable, which results in the creation of nonsensical outputs [22]. In addition to that, MalGAN doesn't produce malware binaries, it just creates adversarial import feature vector representations which is not enough on its own.

To stabilize the training, Kim et al. [22] reduced the number of features by combining autoencoder with GAN. They used CNN-GAN to carry out zero-day malware detection by turning malware into images. Similarly, Bai et al. in [26] proposed an Attack-Inspired GAN (AI-GAN) to create adversarial malware examples conditionally. They incorporated an attacker model into the GAN structure and trained the Discriminator and attacker model simultaneously which made discriminator more robust. The Generator takes malware images as an input to produce a perturbed image. Moti et al. [25] converted malware images as an input to high level features and pass the to the GAN to generate unseen malware files. Although, these image-based GAN models work really work against the target models, they lack specific feature control over the entire process as they produce malware via a random distribution. Moreover, attacks on malware samples by adversaries are distinct from those on images. As long as modifications are bounded by a L_p-norm, adversaries can change the value of any pixel in an image. However, a single byte modification in malware samples might alter the functionality of the original dangerous code or ruin the format of a working PE. Due to this, attackers rarely alter the PE file's raw bytes directly. They instead create a series of actions. Each operation has the potential to change the virus sample without impairing its original functionality [18].

Randhawa et al. [42] utilized the combination of GAN and RL to create an evasion model to attack botnet detectors. The researchers of [27] used dynamic features to produce adversarial attacks against black-box malware classifiers and compared the results of using RL and GAN techniques independently. In the field of industrial IoT, Benaddi et al. [43] focused on anomaly detection in Intrusion Detection Systems (IDS) using Distributional Reinforcement Learning and GAN. Phan et al. [28] proposed an evasion method for black-box malware detectors to evaluate the effectiveness of RL and its combination with GAN, however the focus of their research work is only the comparison between both approaches and did not consider recent state-of-the-art evasion methods in their result analysis. Moreover, they targeted only blackdoor malware samples whereas we conducted our experiments on blackdoor and ransomware malware as ransomware attacks pose a serious threat in IoT devices [44]. Most importantly, there is no evidence that the authors executed the generated malware files in a sandbox and ensured their functionality. Recently, a Conv-GAN based framework MalFox is presented by Zhong et al. [29] which consists of five steps leading to generation of functioning malware variant and targets commercial search engines of 'VirusTotal'. Malware files are parsed to extract system functions and import functions (DLLs). After that, the features vector set gets passed to a complex two CNN networks-based GAN module of the framework which produces a perturbation path for the features to be modified. A PE editor then takes this perturbation path and take functionality preserving actions to modify the malware sample. Table 1 gives the overview of the relevant past approaches in the research subject.

TABLE 1. Comparison with previous studies.

Recent Approaches	RL	GAN	Verify malware functionality	GAN and DRL to produce functional malware samples
MalGAN, 2017 [24]	No	Yes	No	No
DQEAF, 2019 [19]	Yes	No	Yes	No
AIMED-RL, 2021 [21]	Yes	No	Yes	No
AI-GAN, 2021 [26]	No	Yes	No	No
CNN-GAN model, 2021 [25]	No	Yes	No	No
MAB-Malware, 2022 [20]	Yes	No	Yes	No
RL-GAN method, 2023 [28]	Yes	Yes	No	No
MalFox, 2023 [29]	No	Yes	No	No
IF-MalEvade, 2023	Yes	Yes	Yes	Yes

This study focuses on providing an effective evasion model based on DRL framework and GAN together where the GAN model produces an adversarial feature vector based on filtered and influential features and the DRL agent performs a set of few meaningful functionality preserving actions to modify the samples. The proposed model is comprised of a relatively simple feature generation GAN module and employs an influential feature set that improves the training of DRL agent in making header-based mutations. The generated malware variants from the proposed model successfully attacked several black box ML algorithms with high AUC as well as commercial search engines of VirusTotal with impressive results.

III. METHODOLOGY

There has been significant prior work in this field, with researchers modifying the configuration of a malware binary to fool antivirus software with either using the generative adversarial nets or exploiting the deep reinforcement

learning. Our research thoroughly examines the efficacy of integrating a GAN model along with an efficient DRL approach and allowing us to compete with state-of-the-art evasion techniques. While altering the action space of the DRL environment, we seek to escape numerous black-box malware detection methods and antivirus engines of Virus-Total. In contrast to the current MalGan [24] and Deep Reinforcement Learning-based [18], [19], [20] techniques, a malware mutation system built using GAN with DRL greatly improves the malware's ability to avoid detection with less modifications. Figure 1 depicts the proposed evasion framework, which was inspired by the GAN model proposed by Hu and Tan [24] and the Deep RL model proposed by Fang et al. [19].

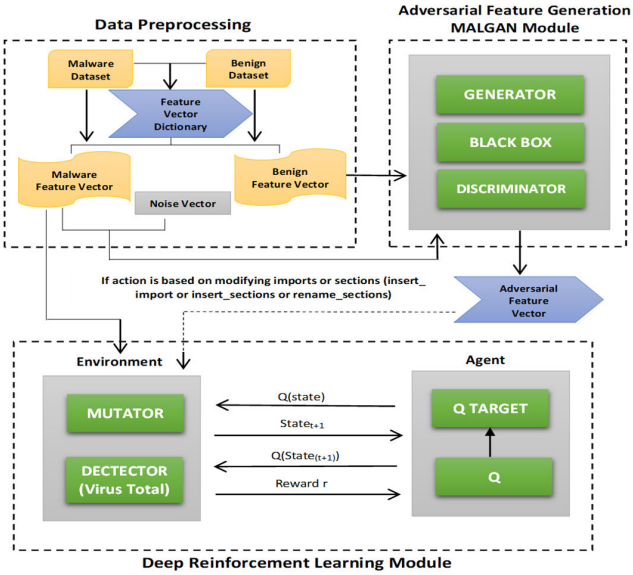

FIGURE 1. Architecture of IF-MalEvade model.

A. WINDOWS IoT

Windows is well known for being the operating of laptop and desktop system that customers and organizations use all around the world. Additionally, a large number of ATMs, point-of-sale terminals, thin clients, industrial automation systems, medical devices, digital signage, kiosks, and other fixed purpose devices run Windows [45]. Windows is the most widely used computer operating system in the world, accounting for 70.39% share of the desktop, tablet, and console OS market in January 2023 [46]. Since Windows systems are employed in IoT devices or are connected to them, PE files are also anticipated to be a potential attack source, considering the multiplatform Windows 11 IoT being in action. In MS Windows operating systems, executables and object code are commonly stored in the Portable Executable (PE) format [47].

B. PE FILE STRUCTURE

The following provides a basic description of the organized structure of PE, as shown in Figure 2 below.

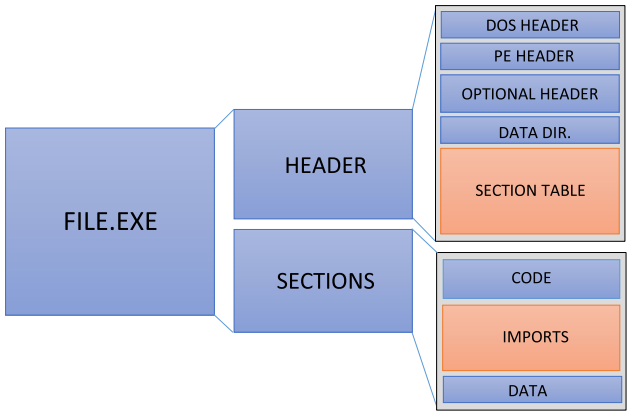

FIGURE 2. PE file structure.

1) HEADER

An executable's essential information, such as the MS-DOS header, the PE signature, the file header, an optional header, and data directories. The file headers are followed immediately by section table. Each section of the file and its specific attributes are included in the Section table.

2) SECTIONS

This part includes relevant real data. The essential ones are.idata (which contains information about imports used in the file),.text (which carries code instructions),.data (which contains initialized global and static variables).

C. MANIPULATING PE FILES

It can be relatively simple to compromise PE files by modifying even one byte, making it generally a non-trivial process. Adding benign import functions, unused sections or even renaming the sections have also proven to be quite effective in rendering the malware file as benign. Injecting bytes into parts of the files that aren't used (such as introducing new sections that the code never gets to) or appending them at the file's end are two conceivable and easy ways to execute modifications, according to Anderson et al. [18]. In addition to that, the executable can also be modified directly, in certain rare instances, without affecting its functioning. Packing is a typical technique in this regard; it involves compressing some part of the executable file and then unpacking it during runtime.

D. IF-MalEvade ARCHITECTURE OVERVIEW

The framework's workflow is divided into two steps: developing an influential feature set and producing adversarial samples based on those features. At first, the adversarial feature set is constructed with the help of MalGan. These newly created features are selected at random from the feature. At first, the adversarial feature set is constructed with the help of MalGan. These newly created features are selected at random from the feature vector library, in accordance with the modifying actions, to create the adversarial

samples. Then, through the process of deep reinforcement learning, adversarial mutated samples are created. Finally, it is determined whether the generated samples still possess the original samples' malicious behavior. We save a generated sample as an adversarial sample if it still exhibits the same malicious behavior like the original sample. The overview of our model's framework is shown in Figure 1.

E. ADVERSARIAL FEATURES GENERATION FROM MAL-GAN

An influential feature set is essential for producing adversarial samples effectively. The Bae et al. proposed MalGan [23] framework served as inspiration for the creation of a feature library that aids in the creation of successful adversarial malware variants. The PE header has been proven to have particularly effective characteristics in the training of machine learning-based malware detectors through the analysis of prior studies and recent survey [48]. Therefore, import functions and section names from the header are retrieved, by parsing malicious and benign samples in order to generate the adversarial feature set. These features are used by MalGan to identify a malware or benign file. During the data preprocessing step, a feature mapping dictionary is built out of all the DLLs and section names that were taken from the datasets of malicious and benign files. This dictionary is utilized to facilitate future operations and feature retrieval in the process. Then, with the help of this dictionary, a binary vector is constructed for each malicious and benign file from the dataset, with '1's representing the presence of that feature and '0's indicating its absence. This binary version of the features is provided to MalGan as input along with a noise vector.

The MalGan architecture consists of a generator followed by a discriminator, and a black-box detector. An AI model called Generator is in charge of creating features that attempt to keep malicious functionality while appearing benign. An AI model called the discriminator is responsible for deciding whether or not the generator has produced malicious output. The generator receives feedback from the discriminator in order to improve itself on the subsequent iterations. To classify the generator's output and aid the discriminator in its decision, the black box detector is trained with machine learning models typically employed in malware detectors. Adversarial feature vectors created through the Generator and feature vectors of benign dataset are labelled during the training of MalGan and then sent into the discriminator learning stage. Discriminator calculates the loss function based on its output and the incoming labels, and uses it to simulate the classification capability of the black box detector. The calculated value from loss function is used by the Generator as input to enhance its output so it substantially bypasses the black-box detector. The MalGan training technique continues until the generated plausible examples are able to fool both the black-box detector and the discriminator.

Once the MalGan generates the adversarial feature vector, import functions and section names are retrieved from adversarial feature vector using the original feature mapping dictionary. These adversarial features (imports and sections) are later on picked out at random by the DRL agent in accordance with the modifying actions in the action space.

F. DEEP REINFORCEMENT LEARNING MODULE

The GAN-based adversarial feature vector is used by the deep reinforcement learning module to build new PE malware samples once the training process of MalGAN is finished. Fang's work on Deep Reinforcement Learning (DQEAF) serves as a foundation for our DRL component where features of malware are used as an input. Following elements are included in the deep reinforcement learning training module:

- The environment known as malware environment which helps the DRL agent in modifying the samples, classifying, and scoring them.
- A neural network model acting as the DRL agent learns the series of mutations that malware samples need to undergo in order to avoid the classifier.
- A classifier is a neural network that analyses a file and classifies it as either a benign file or malware.

The Markov Decision Process (MDP) model is utilized in DQEAF training as follows: State s_t is a vector containing import and section features from malware files. The agent acts with a_t when given the state s_t. Additionally, the agent's 7 actions are represented by a vector in action space including inserting misleading imports (using the adversarial vector from GAN), inserting misleading sections (using the adversarial vector from GAN), adding unused bytes in sections, changing section's name, UPX packing and unpacking, and adding bytes to the end of sections randomly. For each training TURN, the reward r_t is calculated based on the number of actions executed and the label provided by the classifier. Additionally, MAXTURN is defined, which means that if MAXTURN steps of changes have been made and the label is still malicious, the agent shall declare failure. If the classifier detects the file as malware, the agent is given a 0 reward r_t. When the label is benign, reward rt is determined using (1).

$$r_t = 20^{-(\mathrm{TURN}-1)/\mathrm{MAXTURN}} * 100 \tag{1}$$

A discount factor γ is also included which essentially indicates how significant future rewards are to the current state. Discount factor is a number that ranges from 0 to 1. If $= 0$, the agent just considers his first reward. If $= 1$, the agent takes care of all future rewards. To keep it balanced, when it's value is set to 0.9.

The agent optimizes the weights θ during the learning process to reduce the error as indicated by the loss function in (2).

$$l_t(\theta_t) = ((r + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1}; \theta_{t-1})) - Q(s_t, a_t; \theta_t))^2 \tag{2}$$

Essentially, The DRL module takes a malware sample and performs the modifying actions in each turn of the training process to evade antivirus. The adversarial feature vectors produced by GAN contributes in the process when the DRL agent chooses an action that corresponds to manipulating the Imports or Sections features. During the training phase, reward function is determined while the DRL agent and its training environment work together to alter the malware file. Q network turns out to be the most effective policy network that provide an optimal action list for generation of the adversarial malware.

G. CUCKOO SANDBOX

Cuckoo sandbox [49] is a free and open source tool for the automated analysis of malware files that produces a comprehensive report on the actions of any executable. According to Ijaz et al. [50], dynamic malware analysis utilizing the Cuckoo sandbox has an accuracy rate of 94.64%. As a result, the majority of researchers [20], [40], [50] have utilized it to examine the behavior of malware samples when the malicious capability of an adversarial sample is in question. The key benefit is that it allows virtual guest operating systems to run the tool separately from the host system via a virtual switch which practically adds an extra layer of security when carrying out the dynamic malware analysis. Using the virtual switch, which routes API calls from the isolated environment to the host, this tool can monitor and record all network activity and suspicious actions. The sandbox uses signatures to be able to identify malicious actions, like API calls and collected traffic. The signature mainly includes the details of lower-level API requests or traffic generated by a malware sample to higher level information.

Cuckoo Sandbox gives a thorough report of how a suspicious file behaves when it is executed. There are two primary kinds of fields in a Cuckoo sandbox report which are used to analyze the malicious file: static and dynamic. Some examples of static fields are file strings, file hash values, file size etc. The file's actions such as API calls, network characteristics during execution are all examples of dynamic fields. Signatures is one of the fields of dynamic type that contains description of abnormal suspicious activities like if it's creating any additional executable file in the system or creating a service or allocating any memory etc. We validated the behavior of our adversarial malware samples using the signatures field because it's identical for both adversarial and original malware. The flow diagram of generation of an adversarial sample is illustrated in Figure 3 below.

IV. EXPERIMENTATION

This section describes the setup of our experiments which includes the specifications of the datasets, the training parameters of our model, the attacks conducted on ML malware models and the commercial antivirus VirusTotal and the metrics used for evaluation of those attacks.

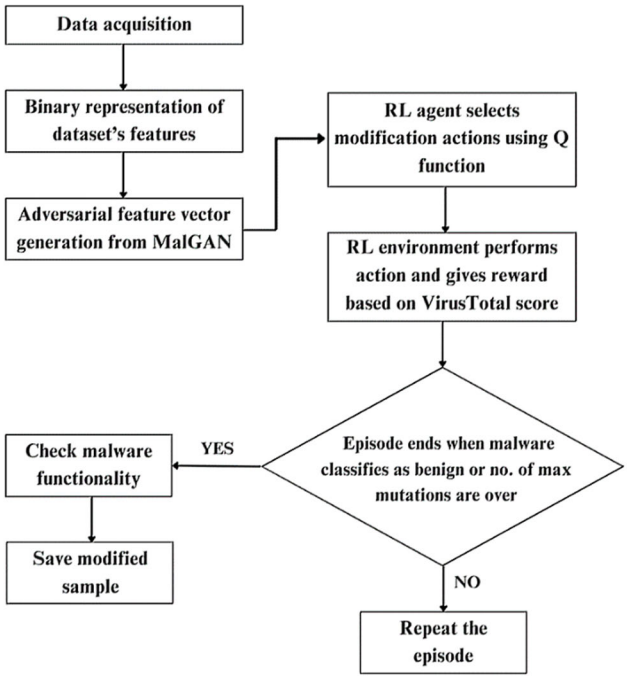

FIGURE 3. Flow diagram of adversarial sample generation.

A. EXPERIMENTAL SETUP

A system with Intel *Core i*7– 1165G7 CPU and Windows 10 was used for the conducted experiments. Since the experiments involved working with malware files, a virtual environment of 200 GB disc space and 16 GB RAM was configured. Tensorflow and sklearn were used to build the target machine learning models. A python library called pelife was installed to parse the PE files and extract features (import functions and section names) from them. OpenAI gym provided the malware environment in which the functionality preserving modifications were performed.

B. DATASET COLLECTION

The whole dataset consisted of 25K malware and 10K benignware windows executable files in total. The malware dataset was collected from the latest online malware repositories of VirusShare [51] and VirusTotal [52] which contained malicious Windows executable files, including backdoor and ransomware. Benign samples were gathered from system files by setting up a virtual environment in a clean Windows 10 and some other known programs were also collected through web crawling. Both malware and benignware needed to be screened to ensure the reliability of our dataset. To check whether or not a file is malicious, we used VirusTotal, which includes 61 different virus detection engines.

C. MALGAN AND DRL TRAINING SPECIFICATIONS

We trained GANs to be able to generate an adversarial feature vector for actions like *add_import* or *add_sections* or *rename_sections* Both the Generator and Discriminator went through training simultaneously in order to improve

the model's effectiveness in producing effective malware samples. Both models were built with identical feed-forward neural networks having 256 nodes in the hidden layer. The generator's input noise vector was scaled to a random distribution of size 10. The size of our MalGAN's feature vector after tracing back all imports and sections were 3523, as opposed to 160 dimensions for the original MalGAN. The 3523-dimensional vector included 3269 imports and 254 sections. Leaky ReLU was chosen for the activation method used by the Generator and Discriminator. The MalGAN was trained using a batch size of 32 and no. of epochs were set to 100. The training and testing portion of dataset used for Deep Reinforcement learning model training is provided in the below Table 2.

TABLE 2. Dataset division for training and testing phases of MalGAN and DRL models.

-	Malicious Dataset	Benign Dataset
Training Dataset	25K	10K
Test Dataset	1000	0

Using OpenGym AI, a DQEAF environment was created in which the agent was trained to perform functionality preserving actions. The malware environment contained the action space listed in the following Table 3. The VirusTotal prediction score was used to determine each action's reward. If the number of antivirus engines that have detected the malware was low, the reward was high and vice versa. Eq (1) is used calculate the reward. Additionally, a simple neural network module was used to train the agent DRL, each having two hidden layers with 256 and 64 nodes. Both layers used ReLU as the activation function for fast computation. The specified number of episodes for the DRL agent training is 500, and the discount factor γ is 0.98. Moreover, by default, each PE file is permitted a limit of 50 mutations. We use the pefile library to interpret and change PE files for the DRL-based mutation phase, and PE Bliss [4] to rebuild modified PE files into new files.

D. ATTACK ON BASELINE ML ALGORITHMS

Firstly, experiments were conducted on the most commonly known machine learning algorithms used by industry Antivirus engines, such as Random Forest, Gradient Boosting, Multi-Layer Perceptron and Decision Tree. The Scikit-learn library's default parameters were used for these algorithms. and trained on the datasets mentioned above. To maintain the consistency of the evaluation method and increase the efficiency of ML models, imports and sections were given as feature inputs in the training phase. While

TABLE 3. Actin space of DRL environment.

List of ACTIONS	Specified by	DESCRIPTION
insert_import	GAN adversarial vector	Add import functions in Import Table
rename_section	GAN adversarial vector	Change the section's name
insert_section	GAN adversarial vector	Add unused (benign) sections
section_append	DRL Action	Append bytes to sections
upx_ pack	DRL Action	Pack using UPX tool
upx_unpack	DRL Action	Unpack using UPX tool
overly_append	DRL Action	Append a random number of bytes

training, the learning rate was set to 0.0003, the epoch number to 150 and the batch size to 50.

To test the robustness of these ML models against our generated samples, 500 original and their respective adversarial malware files were collected from the testing dataset and fed into those models. Malware detection thresholds in the models were established at 0.8 which means a file is considered malicious if its detection score is more than 0.8 to ensure the quality of a generated malware as previously specified by the authors in MAB-Malware's experimental setup [27]. The results of the detection rate with original and adversarial are provided in the Results and Discussion section. Furthermore, to evaluate this experiment's results, two metrics are used. ODR is the black-box detectors' classification rate of original malware from testing dataset while ADR represents the black-box detectors' classification rate of the adversarial samples generated by our proposed method.

An additional experiment was conducted to see if the detection rates of baseline machine learning algorithms were improved by retraining them with the adversarial samples, also known as adversarial training. The algorithms were trained with adversarial samples generated from 50% of the test dataset along with the rest of the malware dataset as it is. The results are shown in the Results and Discussion section.

E. ATTACK ON VIRUSTOTAL AND COMPARISON WITH OTHER EVASION METHODS

To further assess the effectiveness of our model, the adversarial examples generated from two state-of-the-art evasion methods MalFox [31] and MAB-Malware [27] were compared by attacking VirusTotal search engines. The models

were trained with the collected dataset (25K malware and benignware files) and the evasiveness of malware variants generated from this model was checked against the VirusTotal anti-malware solutions. The model parameters of these methods were retained as given in their github code repositories.

The metrics to measure the performance of the proposed method in this experiment are the detection rate and evasive rate of adversarial samples against VirusTotal. Detection rate, as demonstrated in (3), is the percentage of VirusTotal's search engines (n) that identify the original/adversarial malware samples, divided by total number of search engines (N). Evasion rate is calculated using (4), which determines how well the malware is able to evade detection by using the no. of search engines that identify the malware (N_{org}) and the no. of search engines that identify its adversarial variant (N_{adv}). The higher the evasive rate, the higher the likelihood of the adversarial example being detected as benign.

$$\text{detection rate} = \frac{n}{N} \tag{3}$$

$$\text{evasive rate} = \frac{Norg - Nadv}{Norg} \tag{4}$$

V. RESULTS

Figure 4 exhibits the statistical experimental results of most commonly known ML algorithms like Random Forest, Gradient Boosting, Multi-Layer Perceptron and Decision Tree against original and adversarial malware. X-axis shows the ML algorithms while the Y-axis shows the AUC, detection rate of original malware and the detection rate of adversarial malware against each algorithm and detection rate of adversarial malware after retraining.

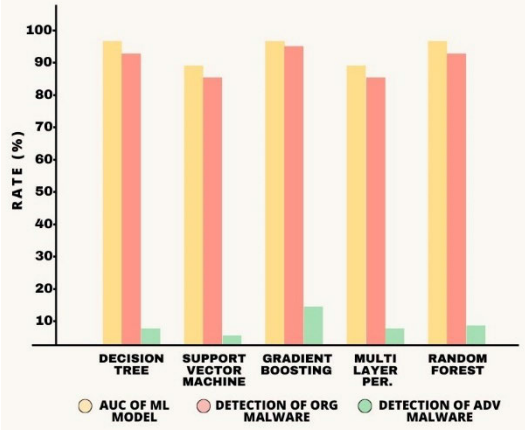

FIGURE 4. Detection rate of different ML algorithms for original malware and proposed adversarial malware samples.

Figure 5 shows the Detection Rate of different ML algorithms before and after adversarial training. X-axis shows the ML algorithms while the Y-axis shows the detection rate of adversarial malware against each algorithm.

Figure 6 illustrates the statistical experimental results of malware generated by MalFox, MAB-Malware and the proposed evasion method against VirusTotal. X-axis shows the minimum, maximum and average values while the Y-axis shows the Detection Rate calculated by formula given in section IV.

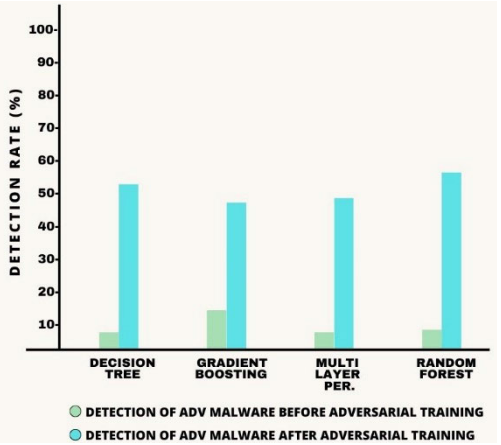

FIGURE 5. Detection Rate of different ML algorithms before and after adversarial training.

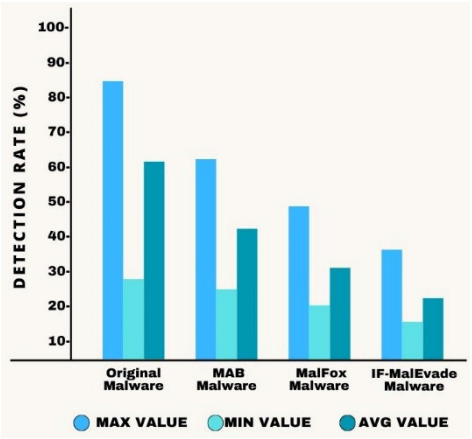

FIGURE 6. Detection rate of different evasion methods against virusTotal search engines.

Figure 7 displays the statistical experimental results of malware generated by MalFox, MAB-Malware and the proposed evasion method against VirusTotal. X-axis shows the minimum, maximum and average values while the Y-axis shows the Evasion Rate calculated by formula given in section IV.

VI. DISCUSSION

In the first experiment, the performance of commonly used ML Detectors against the proposed evasion method was evaluated. Original and adversarial, both samples had been inputted to various ML-based malware detectors models for being classified as malware or benign. The malicious detection threshold value of ML detectors was fixed at 0.8 for the subsequent experiments. Consequently, an input sample was regarded as malware if the detection score is higher than 0.8. Table 4 shows how well ML-based detectors performed prior to and after being targeted by our model's adversarial

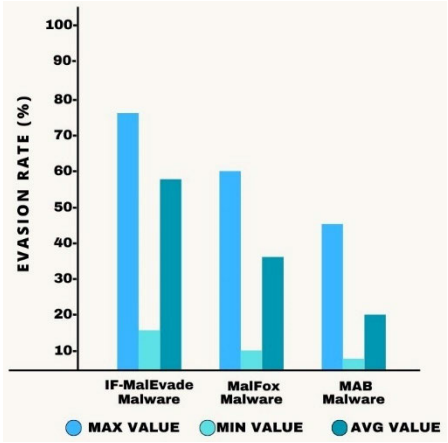

FIGURE 7. Evasion Rate of malware generated by various evasion methods targeting virusTotal.

samples. Measures of original samples' accuracy of such malware detectors, area under the curve (AUC) and original detection rate (ODR), are consistently above 0.8 and 80%, respectively. These ML based detectors, however, incorrectly classify adversarial malicious examples where as the ADR is, in the best case scenario (Gradient Boosting), around 13.56%. It should be noted that ADR is determined by the classification rate of ML based detectors against just the adversarial samples which have been modified from original samples. This finding demonstrates the efficacy of our evasion model in creating convincing adversarial malware vectors to exploit flaws in ML based malware detectors.

TABLE 4. The detection rates of original malware samples and adversarial malware samples against ML based detectors.

ML detectors	AUC score	Original Detection Rate	Adversarial Detection Rate
Decision Tree	96.56	92.98	8.1
Gradient Boosting	95.85	95.24	13.56
Multi-Layer Perceptron	86.50	84.52	7.69
Random Forest	96.12	92.32	9.1

Additionally, after adversarial retraining, it can be seen that the detection rate of those ML detectors was significantly improved, like detection rate of Random Forest algorithm went from 9.1% to 57.6%, as shown in Table 5.

In the second experiment, the performance of proposed method via detection rate and evasive rate is evaluated and compared with state-of-the-art evasions methods: MalFox and MAB-malware. The adversarial malware samples are effective at evading detection by VirusTotal, as seen by

TABLE 5. The detection rate of adversarial malware against ML based detectors with and without adversarial retraining.

ML detectors	Adv. Detection Rate without retraining	Adv. Detection rate with retraining
Decision Tree	8.1	54.6
Gradient Boosting	13.56	48.1
Multi-Layer Perceptron	7.69	49.8
Random Forest	9.1	57.9

their low detection rate and high evasive rate. Our attacks are regarded successful if the adversarial malware example causes a lower detection rate than the original malware, and has a high evasion rate. Due to successful experimental results of these methods, we think it is fair to compare MalFox and MAB-malware with our generated malware samples and judge them based on their ability to detect threats and avoid being detected. The VirusTotal scan engines and anti-virus softwares were created by various organizations and labs, and they each take a slightly different approach to analyzing malware. This means that some entities may pick up on a single piece of malware while others fail to do so. Malware detection rates will never be able to reach perfection. Table 6 shows the average detection rate for malware before and after being processed by our proposed approach. This is a significant decline of roughly 21.8%. As a result, the average percentage of evasion for known malicious cases has increased dramatically (to around 57.9%).

TABLE 6. Comparison results of detection rate between adversarial samples of MalFox/MAB-malware and proposed model against virusTotal.

Evaluation benchmark	Average rate	Min rate	Max rate
Detection rate (Original malware)	61.8	27.2	85.1
Detection rate (MAB-malware)	42.9	25.5	62.2
Detection rate (Malfox malware)	30.2	21.9	49.7
Detection rate (Proposed malware)	21.8	15.4	35.5

Correspondingly, the average evasion rate for adversarial malware examples has been significantly improved (about 57.9%). Table 6 shows that our malware has a detection rate between 61.8% and 85.1%. But their respective adversarial malware instances have a detection rate that is 21.8% to 35.5% lower than that of the original malware. When

TABLE 7. Comparison results of evasion rate between adversarial samples of MalFox/MAB-malware and proposed model against virusTotal.

Evaluation benchmark	Average	Min	Max
Evasion rate (MAB-malware)	19.2	7.8	44.5
Evasion rate (Malfox malware)	38.2	10.2	61.9
Evasion rate (proposed malware)	57.8	15.7	75.5

comparing single malware samples to their corresponding adversarial malware examples, the detection rate drops significantly; in the worst instance, it drops by as much as 58. In addition, as shown in Table 7, the most evasive rate is 75.5%, implying that 75.5 percent of malware detection engines misclassify the respective adversarial samples as being harmless while having labelled the original malware as malware.

VII. CONCLUSION

This research studies the integration of GAN model along with Deep Reinforcement Learning which generates adversarial malicious examples with a significantly higher chance of evading detection against ML based IoT malware detectors. In the proposed framework, DRL agents are trained to produce fully functional adversarial malware samples by feeding them highly influential adversarial features provided by GANs. Malware can improve its ability to evade ML-based detectors with fewer modifications if DRL and GANs are combined. We have analyzed the robustness of various well known ML based malware detectors as well as VirusTotal's detection engines, and compared the results with state-of-the-art evasion models. The experimental results proves that the proposed framework incorporating GAN with DRL outperforms the existing evasion methods and has a promising approach to become a benchmark in testing the robustness of ML-based malware detectors.

As a result, our proposed evasion method successfully exploits machine learning based malware detectors without being aware of their internal implementation specifics, making it a more useful and potent attack framework. Furthermore, the generated adversarial samples also improve the performance of malware detectors in detecting unseen malware variants via the adversarial training.

In the future, we intend to target other file formats for the adversarial malware generation using our framework such as Executable and Linkable format (ELF) used in Linux systems and Android since IoT is a combination of different computer architectures. Defining more concise and effective action space in the DRL environment is also one of our main goals for the future. Moreover, we will also try to explore other influential feature sets like API calls or system calls to create adversarial samples against the IoT malware detectors.

DECLARATION

The authors declare no conflicts of interest.

ACKNOWLEDGMENT

This research is supported by Princess Nourah bint Abdulrahman University Researchers Supporting Project number (PNURSP2023R136), Princess Nourah bint Abdulrahman University, Riyadh, Saudi Arabia. The authors are also thankful to AIDA Lab CCIS Prince Sultan University, Riyadh Saudi Arabia for support.

REFERENCES

[1] Statista. (2016). *Internet of Things (IoT) Connected Devices Installed Base Worldwide From 2015 to 2025*. Accessed: Aug. 5, 2022. [Online]. Available: https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

[2] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Gener. Comput. Syst.*, vol. 82, pp. 395–411, May 2018.

[3] B. D. Deebak, F. H. Memon, S. A. Khowaja, K. Dev, W. Wang, N. M. F. Qureshi, and C. Su, "A lightweight blockchain-based remote mutual authentication for AI-empowered IoT sustainable computing systems," *IEEE Internet Things J.*, vol. 10, no. 8, pp. 6652–6660, Apr. 2023.

[4] R. Samani, "McAfee labs threats report: June 2021," McAfee, Hong Kong, Tech. Rep., Jun. 2021. Accessed: Sep. 22, 2022. [Online]. Available: https://www.mcafee.com/blogs/other-blogs/mcafee-labs/mcafee-labs-report-highlights-ransomware-threats/

[5] VirusTotal, Dublin, Ireland. (2022). *Deception at Scale: How Malware Abuses Trust*. [Online]. Available: https://assets.virustotal.com/reports/2022-deception-at-scale

[6] B. Geluvaraj, P. M. Satwik, and T. A. Kumar, "The future of cybersecurity: Major role of artificial intelligence, machine learning, and deep learning in cyberspace," in *Proc. Int. Conf. Comput. Netw. Commun. Technol.*, 2019, pp. 739–747.

[7] G. McDonald and T. Spangler. (2019). New machine learning model sifts through the good to unearth the bad in evasive malware. Microsoft Defender ATP Research Team. Accessed: Aug. 10, 2022. [Online]. Available: https://www.microsoft.com/en-us/security/blog/2019/07/25/new-machine-learning-model-sifts-through-the-good-to-unearth-the-bad-in-evasive-malware/

[8] H. Darabian, A. Dehghantanha, S. Hashemi, S. Homayoun, and K. R. Choo, "An opcode-based technique for polymorphic Internet of Things malware detection," *Concurrency Comput., Pract. Exp.*, vol. 32, no. 6, Mar. 2020, Art. no. e5173.

[9] M. Shobana and S. Poonkuzhali, "A novel approach to detect IoT malware by system calls using deep learning techniques," in *Proc. Int. Conf. Innov. Trends Inf. Technol. (ICITIIT)*, Feb. 2020, pp. 1–5.

[10] J. Su, D. V. Vasconcellos, S. Prasad, D. Sgandurra, Y. Feng, and K. Sakurai, "Lightweight classification of IoT malware based on image recognition," in *Proc. IEEE 42nd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, vol. 2, Jul. 2018, pp. 664–669.

[11] H. HaddadPajouh, A. Dehghantanha, R. Khayami, and K.-K.-R. Choo, "A deep recurrent neural network based approach for Internet of Things malware threat hunting," *Future Gener. Comput. Syst.*, vol. 85, pp. 88–96, Aug. 2018.

[12] Q.-D. Ngo, H.-T. Nguyen, V.-H. Le, and D.-H. Nguyen, "A survey of IoT malware and detection methods based on static features," *ICT Exp.*, vol. 6, no. 4, pp. 280–286, Dec. 2020.

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

[14] Threat Hunter Team and Symantec. (2022). *The Ransomware Threat Landscape*. Accessed: Oct. 19, 2022. [Online]. Available: https://symantec-enterprise-blogs.security.com/blogs/threat-intelligence/ransomware-threat-landscape-what-expect-2022

[15] A. Al-Dujaili, A. Huang, E. Hemberg, and U.-M. O'Reilly, "Adversarial deep learning for robust detection of binary encoded malware," in *Proc. IEEE Secur. Privacy Workshops (SPW)*, San Francisco, CA, USA, May 2018, pp. 76–82.

[16] X. Liu, X. Du, X. Zhang, Q. Zhu, H. Wang, and M. Guizani, "Adversarial samples on Android malware detection systems for IoT systems," *Sensors*, vol. 19, no. 4, p. 974, Feb. 2019.

[17] F. Wang, Y. Lu, C. Wang, and Q. Li, "Binary black-box adversarial attacks with evolutionary learning against IoT malware detection," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1–9, Aug. 2021.

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

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

[20] W. Song, X. Li, S. Afroz, D. Garg, D. Kuznetsov, and H. Yin, "MAB-malware: A reinforcement learning framework for attacking static malware classifiers," 2020, *arXiv:2003.03100*.

[21] R. Labaca-Castro, S. Franz, and G. D. Rodosek, "AIMED-RL: Exploring adversarial malware examples with reinforcement learning," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discovery Databases (ECML PKDD)*, 2021, pp. 37–52.

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

[23] H. Bae, Y. Lee, Y. Kim, U. Hwang, S. Yoon, and Y. Paek, "Learn2Evade: Learning-based generative model for evading PDF malware classifiers," *IEEE Trans. Artif. Intell.*, vol. 2, no. 4, pp. 299–313, Aug. 2021.

[24] W. Hu and Y. Tan, "Generating adversarial malware examples for black-box attacks based on GAN," in *Proc. Int. Conf. Data Mining Big Data*, 2022, pp. 409–423.

[25] Z. Moti, S. Hashemi, H. Karimipour, A. Dehghantanha, A. N. Jahromi, L. Abdi, and F. Alavi, "Generative adversarial network to detect unseen Internet of Things malware," *Ad Hoc Netw.*, vol. 122, Nov. 2021, Art. no. 102591.

[26] T. Bai, J. Zhao, J. Zhu, S. Han, J. Chen, B. Li, and A. Kot, "Toward efficiently evaluating the robustness of deep neural networks in IoT systems: A GAN-based method," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1875–1884, Feb. 2022.

[27] H. Yadav, A. Handa, N. Kumar, and S. K. Shukla, "Adversaries strike hard: Adversarial attacks against malware classifiers using dynamic api calls as features," in *Proc. 5th Int. Symp. Cyber Secur. Cryptogr. Mach. Learn. (CSCML)*, Be'er Sheva, Israel, Jul. 2021, pp. 20–37.

[28] T. D. Phan, T. D. Luong, N. H. Quoc An, Q. N. Huu, H. K. Nghi, and V.-H. Pham, "Leveraging reinforcement learning and generative adversarial networks to craft mutants of windows malware against black-box malware detectors," in *Proc. 11th Int. Symp. Inf. Commun. Technol.*, Dec. 2022, pp. 31–38.

[29] F. Zhong, X. Cheng, D. Yu, B. Gong, S. Song, and J. Yu, "MalFox: Camouflaged adversarial malware example generation based on conv-GANs against black-box detectors," *IEEE Trans. Comput.*, early access, Jan. 13, 2023, doi: 10.1109/TC.2023.3236901.

[30] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and B. Xu, "Generative adversarial networks," *Commun. ACM*, vol. 63, no. 11, pp. 139–144, 2020.

[31] M. I. Yousuf, I. Anwer, A. Riasat, K. T. Zia, and S. Kim, "Windows malware detection based on static analysis with multiple features," *PeerJ Comput. Sci.*, vol. 9, Apr. 2023, Art. no. e1319.

[32] M. Qiao, A. H. Sung, and Q. Liu, "Merging permission and API features for Android malware detection," in *Proc. 5th IIAI Int. Congr. Adv. Appl. Informat. (IIAI-AAI)*, Jul. 2016, pp. 566–571.

[33] N. McLaughlin, J. M. D. Rincon, B. Kang, S. Yerima, P. Miller, S. Sezer, Y. Safaei, E. Trickel, Z. Zhao, A. Doupe, and G. J. Ahn, "Deep Android malware detection," in *Proc. 7th ACM Conf. Data Appl. Secur. Privacy*, 2017, pp. 301–308.

[34] S. Mohammadkhani and M. Esmaeilpour, "A new method for behavioural-based malware detection using reinforcement learning," *Int. J. Data Mining, Model. Manage.*, vol. 10, no. 4, pp. 314–330, 2018.

[35] J. Zhang, Z. Qin, H. Yin, L. Ou, and K. Zhang, "A feature-hybrid malware variants detection using CNN based opcode embedding and BPNN based API embedding," *Comput. Secur.*, vol. 84, pp. 376–392, Jul. 2019.

[36] W. Ali, "Hybrid intelligent Android malware detection using evolving support vector machine based on genetic algorithm and particle swarm optimization," *Int. J. Comput. Sci. Netw. Secur.*, vol. 19, no. 9, p. 15, 2018.

[37] A. S. Bist, "Genetic and firefly algorithm in instance and feature selection: An approach for malware detection," *Int. J. Digit. Inf. Wireless Commun.*, vol. 8, no. 4, pp. 232–238, 2018.

[38] Z. Fang, J. Wang, J. Geng, and X. Kan, "Feature selection for malware detection based on reinforcement learning," *IEEE Access*, vol. 7, pp. 176177–176187, 2019.

[39] F. Kreuk, A. Barak, S. Aviv-Reuven, M. Baruch, B. Pinkas, and J. Keshet, "Deceiving end-to-end deep learning malware detectors using adversarial examples," 2018, *arXiv:1802.04528*.

[40] B. Jin, J. Choi, J. B. Hong, and H. Kim, "On the effectiveness of perturbations in generating evasive malware variants," *IEEE Access*, vol. 11, pp. 31062–31074, 2023.

[41] A. Abusnaina, A. Khormali, H. Alasmary, J. Park, A. Anwar, and A. Mohaisen, "Adversarial learning attacks on graph-based IoT malware detection systems," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jul. 2019, pp. 1296–1305.

[42] R. H. Randhawa, N. Aslam, M. Alauthman, M. Khalid, and H. Rafiq, "Deep reinforcement learning based evasion generative adversarial network for botnet detection," *Future Gener. Comput. Syst.*, vol. 150, pp. 294–302, Jan. 2024.

[43] H. Benaddi, M. Jouhari, K. Ibrahimi, J. B. Othman, and E. M. Amhoud, "Anomaly detection in industrial IoT using distributional reinforcement learning and generative adversarial networks," *Sensors*, vol. 22, no. 21, p. 8085, Oct. 2022.

[44] M. Humayun, N. Jhanjhi, A. Alsayat, and V. Ponnusamy, "Internet of Things and ransomware: Evolution, mitigation and prevention," *Egyptian Informat. J.*, vol. 22, no. 1, pp. 105–117, Mar. 2021.

[45] Microsoft. (2023). *An Overview of Windows for IoT*. Accessed: Mar. 20, 2023. [Online]. Available: https://learn.microsoft.com/en-us/windows/iot/product-family/windows-iot

[46] Statista. (2023). *Market Share Held by the Leading Computer (Desktop/Tablet/Console) Operating Systems Worldwide From January 2012 to January 2023*. Accessed: Mar. 20, 2023. [Online]. Available: https://www.statista.com/statistics/268237/global-market-share-held-by-operating-systems-since-2009/

[47] Microsoft. (2013). *Microsoft Portable Executable and Common Object File Format Specification*. Accessed: Mar. 20, 2023. [Online]. Available: http://www.microsoft.com/whdc/system/platform/firmware/PECOFF.mspx

[48] M. H. U. Sharif, M. AliMohammed, S. Hassan, and M. H. Sharif, "Comparative study of prognosis of malware with PE headers based machine leaning techniques," in *Proc. Int. Conf. Smart Comput. Appl. (ICSCA)*, Hail, Saudi Arabia, Feb. 2023, pp. 1–6.

[49] C. Guarnieri, A. Tanasi, J. Bremer, and M. Schloesser. (2010). *The Cuckoo Sandbox*. Accessed: Mar. 5, 2023. [Online]. Available: https://www.cuckoosandbox.org

[50] M. Ijaz, M. H. Durad, and M. Ismail, "Static and dynamic malware analysis using machine learning," in *Proc. 16th Int. Bhurban Conf. Appl. Sci. Technol. (IBCAST)*, Islamabad, Pakistan, Jan. 2019, pp. 687–691.

[51] VirusShare. (2022). *Virusshare.com—Because Sharing is Caring*. Accessed: Aug. 5, 2022. [Online]. Available: https://virusshare.com/

[52] *Virustotal*. Accessed: Sep. 25, 2022. [Online]. Available: https://www.virustotal.com/en/

RAHAT MAQSOOD ARIF received the B.S. degree from the Department of Computer Engineering, University of Engineering and Technology (UET), Lahore, Pakistan, in 2018, and the master's degree in computer science from UET, in 2023. Her current research interests include artificial intelligence, machine learning, computer vision, malware analysis and cyber threat intelligence.

MUHAMMAD ASLAM received the Ph.D. degree in computer sciences from CINVESTAV-IPN, Mexico, in 2005, under the Cultural Exchange Scholarships between Pakistan and Mexico.

He has more than 15 years of experience in software architecture design, team leading, team building, and software project, and 14 years of experience in research and development and teaching at postgraduate level (supervising Ph.D. and M.Sc. thesis). He has published his research findings in number of well reputed impact factor journals of Springer, IEE, and Elsevier. He already has supervised six Ph.D. and more than 50 master's theses and final year projects. His research and teaching interests include artificial intelligence, distributed intelligence, knowledge-based systems, expert systems, intelligent agents, human–computer interaction, machine learning, computer-supported cooperative work, cooperative writing and authoring, cooperative learning, and distributed computing.

Dr. Aslam received the Merit Scholarship from the Board of Intermediate and Secondary Education, Sargodha Division, Pakistan, from 1984 to 1986, and the Silver Medal from the Faculty of Agricultural Engineering, University of Agricultural, Faisalabad, Pakistan, from 1987 to 1991. He was awarded the Cultural Exchange Scholarship between Pakistan and Mexico, from 2000 to 2004, for the Ph.D. studies. He received the research grants from HEC and UET.

SHAHA AL-OTAIBI (Member, IEEE) received the M.S. degree in computer science and the Ph.D. degree in artificial intelligence from KSU. She is currently an Associate Professor with the Department of Information Systems, College of Computer and Information Sciences, PNU, Saudi Arabia. Her research interests include data science, artificial intelligence, machine learning, bio-inspired computing, cybersecurity, and information security. She has a SFHEA and the Senior Fellow Recognition from the U.K. Higher Education Academy. She is a reviewer in some journals and an editorial board member in other journals.

ANA MARIA MARTINEZ-ENRIQUEZ received the master's and Ph.D. degrees in computer science from the University of Paris VI, France, in March 1985. Her Ph.D. thesis entitled Automatic Archiving of Knowledge from Textual Documents (in French). She has been a Full Researcher with the Department of CS, CINVESTAV-IPN, Mexico City, since October 1985. Following her Ph.D. research activities, she focused on foundations, methods, and tools for modeling, building and validating applications based on KBS, natural language processing, machine learning, and data mining. Later, her research focuses on DAI, CSCW, deductive and adaptive collaborative interface, and group awareness. Her publications include several journals in JCR, in specialized journal in computer science and international conference.

TANZILA SABA (Senior Member, IEEE) received the Ph.D. degree in document information security and management from the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia, in 2012. She is currently the Associate Chair with the Information Systems Department, College of Computer and Information Sciences, Prince Sultan University, Riyadh, Saudi Arabia, where she is the Artificial Intelligence and Data Analytics Research Laboratory Leader. Her research interests include medical imaging, pattern recognition, data mining, MRI analysis, and soft computing. She is an Active Professional Member of ACM, AIS, and IAENG organizations. She is the PSU Women in Data Science (WiDS) Ambassador at Stanford University and the Global Women Tech Conference. She was a recipient of the Best Student Award from the Faculty of Computing, UTM, in 2012.

SAEED ALI BAHAJ received the Ph.D. degree from Pune University, India, in 2006. He is currently an Associate Professor with the Computer Engineering Department, Hadramout University, Yemen, and the MIS Department, COBA, Prince Sattam bin Abdul-Aziz University, Saudi Arabia. His research interests include information management, forecasting, information engineering, big data analysis, and information security.

AMJAD REHMAN (Senior Member, IEEE) received the Ph.D. degree from the Faculty of Computing, Universiti Teknologi Malaysia (UTM), Malaysia, in 2010, specializing in information security using image processing techniques. He is currently an Associate Professor with CCIS, Prince Sultan University, Riyadh, Saudi Arabia. He is also a PI in several projects and completed projects funded by MoHE Malaysia, Saudi Arabia. His research interests include bioinformatics, the IoT, information security, and pattern recognition. He received the Rector Award for the 2010 Best Student from UTM Malaysia.