AEDGAN: A Semi-Supervised Deep Learning Model for Zero-Day Malware Detection

Abdullah Marish Ali[1], Fuad A. Ghaleb[2]*, Faisal Saeed[3]
Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah, Saudi Arabia[1]
College of Computing, Birmingham City University, Birmingham B4 7XG, UK[2, 3]

***Abstract*—Malware presents an increasing threat to cyberspace, drawing significant attention from researchers and industry professionals. Many solutions have been proposed for malware detection; however, zero-day malware detection remains challenging due to the evasive techniques used by malware authors and the limitations of existing solutions. Traditional supervised learning methods assume a fixed relationship between malware and their class labels over time, but this assumption does not hold in the ever-changing landscape of evasive malware and its variants. That is malware developers intentionally design malicious software to share features with benign programs, making zero-day malware. This study introduces the AEDGAN model, a zero-day malware detection framework based on a semi-supervised learning approach. The model leverages a generative adversarial network (GAN), an autoencoder, and a convolutional neural network (CNN) classifier to build an anomaly-based detection system. The GAN is used to learn representations of benign applications, while the auto-encoder extracts latent features that effectively characterize benign samples. The CNN classifier is trained on an integrated feature vector that combines the latent features from the autoencoder with hidden features extracted by the GAN's discriminator. Extensive experiments were conducted to evaluate the model's effectiveness. Results from two benchmark datasets show that the AEDGAN model outperforms existing solutions, achieving a 5% improvement in overall accuracy and an 11% reduction in false alarms compared to the best-performing related model.**

Keywords*—Malware detection; zero-day; anomaly detection; generative adversarial network; autoencoder; convolutional neural network

I. Introduction

This Malware, or malicious software, refers to any program specifically designed to damage, disrupt, or exploit digital systems. Common types include viruses, worms, Trojan horses, ransomware, spyware, rootkits, and bots. Over the past decade, malware threats have continuously evolved, posing a persistent and growing challenge [1]. Cybercriminals employ advanced techniques to disguise and distribute malicious code, often using obfuscation and evasion tactics to bypass security defenses, making detection and analysis increasingly difficult. Attacks targeting critical infrastructure, including power plants, financial institutions, and mobile networks, can have severe and widespread consequences. A notable example is the 2021 ransomware attack on a major U.S. pipeline, which led to a complete operational shutdown and substantial financial losses [2]. As Internet of Things (IoT) technologies continue to proliferate within critical infrastructure, it becomes increasingly likely that malware attacks will exploit heightened vulnerabilities. This susceptibility arises from the complexity of modern attacks and the digital environment, rather than a simple lack of security measures or computational resources [3].

Many detection approaches were proposed and can be categorized into signature, anomaly-based approaches [3-8]. Several previous malware detection systems have relied on the signature-based approach [4, 6, 9], which effectively identifies malicious patterns extracted from static or dynamic malware analysis. This method has proven particularly successful when combined with supervised machine learning (ML) techniques, enhancing its ability to detect known threats based on predefined signatures. These techniques learn to distinguish between benign and malicious samples, leading to significant improvements in detection accuracy [6, 10-14]. However, the signature-based approach has a significant disadvantage in that it only concentrates on well-known malware patterns, which severely restricts its use. In addition, supervised based solutions assume such patterns are static to both malware and benign software, limiting detection to known malware manifestations. As a result, it is inefficient in identifying zero-day vulnerabilities, which are previously unknown or extremely complex threats that deviate from existing signatures. Automated malware development toolkits provide techniques like packing, obfuscation, and polymorphism to conceal malicious code and mimic normal patterns, making it relatively easy to create new malware variants or modified versions that can evade machine learning-based detection. This underscores the importance of identifying previously unseen malware instances to effectively combat emerging novel threats.

Anomaly detection is a powerful method for identifying abnormal patterns or behaviors that deviate from expected norms. Many malware detection solutions have leveraged one-class classification, which focuses on modeling benign data to capture its essential characteristics. This approach enables the system to effectively distinguish malicious instances by detecting deviations from the learned benign profile [15-17]. Consequently, any sample not aligning with the acquired model representation is classified as a potential malware occurrence. However, this approach has two major drawbacks. First, it tends to generate a high false alarm rate because benign samples are often significantly outnumbered by malware samples during training. This imbalance creates biased learning, leading to an increased likelihood of misclassifications. Second, due to the use of evasive and obfuscation techniques by malware developers, the learned representation of benign samples often overlaps with malicious features. This overlap makes it

challenging to distinguish between benign and malicious instances, resulting in a high degree of uncertainty in classification decisions.

To address these challenges, this study aims to design and develop a zero-day malware detection model using a semi-supervised learning approach for anomaly detection. Semi-supervised learning effectively utilizes a limited amount of labeled data combined with a larger set of unlabeled malware samples. The benign samples were used to train an anomaly detection model. Anomaly detection works by identifying abnormal patterns that deviate from expected norms. In the context of malware detection, many traditional systems use a one-class classification approach, which models benign data to capture its essential characteristics and detects deviations from this benign profile as potential malware. This method enables the model to capture diverse and evolving malware patterns, enhancing its ability to generalize to unseen threats. Learning deviations from known behaviors improves the detection of novel attacks, enabling the proposed approach to identify threats that traditional methods may fail to recognize. However, this approach has two significant drawbacks. First, there is often a data imbalance between benign and malware samples, leading to a high false alarm rate due to biased learning. Second, malware frequently uses evasive techniques, causing overlap between benign and malicious features, making it difficult to distinguish between them and leading to misclassifications.

To overcome these limitations, the proposed model, named AEDGAN, combines generative adversarial network (GAN), autoencoder (AE), and convolutional neural network (CNN) architectures. To mitigate the data imbalance caused by the limited availability of benign samples, the GAN model generates more accurate representations of benign applications, thereby enhancing the ability to distinguish them from evolving malware. Meanwhile, the autoencoder is customized to extract latent features that best characterize benign samples. Finally, the CNN model is trained on a consolidated feature vector derived from both the latent attributes obtained from the autoencoder and the hidden features extracted by the discriminator within the GAN model. Extensive experiments were conducted to assess and validate the proposed model's performance. These experiments employed two datasets, encompassing evasive and novel malware attacks, for validation. The model's efficacy was evaluated by benchmarking it against state-of-the-art solutions. This study presents the following contributions:

1) Develop AEDGAN, an advanced architecture integrating Generative Adversarial Networks (GAN), deep autoencoding, and Convolutional Neural Networks (CNN) to create a zero-day malware detection model based on semi-supervised learning and anomaly detection.

2) Design and implement a GAN architecture, trained exclusively on benign instances, to generate realistic representations of normal samples. This approach is grounded in the hypothesis that benign software exhibits lower dynamism compared to malware, making it well-suited for GAN-based generation to enhance the modeling of normal behavior.

3) Construct an anomaly-based detection model, utilizing deep autoencoding to improve feature representation and detection accuracy. The auto-encoder leverages benign samples generated by the GAN to refine the distinction between normal and malicious behavior.

4) Develop a CNN model to reduce false positives produced by the autoencoder, specifically addressing the challenge of feature overlap between benign and malicious instances. The CNN is trained on a combined feature set, integrating latent features from the autoencoder and outputs from the GAN discriminator, to strengthen its ability to differentiate be-tween benign and malware samples.

The remainder of this paper is structured as follows: Section II reviews related work, while Section III elaborates on the proposed model. Section IV provides comprehensive details on the experimental design, encompassing dataset selection, performance metrics, validation, and evaluation procedures. Section V presents the results and Section VI includes the discussions of the results, as well as the limitations of the proposed solution, while Section VII offers concluding remarks.

II. Related Works

Please Zero-day malware refers to previously unknown or newly discovered malware that exploits vulnerabilities for which no patch or signature exists [18]. Detecting zero-day malware is a significant challenge for existing solutions [19]. Traditional signature-based detection methods rely on known patterns or signatures of malware, making them ineffective against zero-day malware [3, 18]. However, there are several approaches and techniques that have been proposed to address this issue.

One approach is the use of behavioural analysis, which focuses on the actions and behaviour of software to identify malicious activities [9, 11, 13, 20]. This technique can detect zero-day malware by analysing the behaviour of an application during its execution. By monitoring system calls and analysing their patterns, it is possible to identify suspicious or malicious behaviour [13]. However, this approach has limitations, as some malware can evade detection by modifying their behaviour or using obfuscation techniques [19]. Authors in study [4] proposed a CNN architecture for zero-day malware detection based on static analysis. CNN is employed to extract a small binary fragment from the text section of the Portable Executable (PE) malware file. However, the limitation of this model is that these small fragments may not be available due to the use of the obfuscation and evasion techniques by malware authors. Authors in study [5] presents the Cyber Resilience Recovery Model (CRRM), an epidemiological model designed to combat zero-day outbreaks in closed networks. Authors in study [7] pro-posed a zero-day malware detection model based on multiple views learning with convolutional method. Three sources of information were integrated to increase the chance of recognition of the malicious patterns with the hope of detecting zero-day malware. The main drawback of such an approach is the reliance of static analysis where it is complex to extract representative patterns due to the use of obfuscations and evasive techniques. Authors in study [8] proposes a novel method, the transferred deep-convolutional generative adversarial network (tDCGAN), to robustly detect malware,

including zero-day attacks, by generating fake malware and using deep autoencoders for feature extraction, achieving 95.74% average classification accuracy and demonstrating superior stability and resilience against zero-day attacks compared to other models. However, the reliance on generating fake malware data to train the model will not resolve the inherent issue of overlapping malware features with benign features, which arises from unrepresentative benign samples and the obfuscation and evasive techniques used by malware authors, potentially leading to lower generalization capabilities for unseen or novel attacks.

Many researchers used machine learning techniques, such as supervised machine learning and random forest algorithms [6, 10, 12, 14, 21-24]. These techniques can learn from existing information and detect new malware apps, including zero-day malware [25]. Machine learning models can be trained on known malware samples and then used to classify unknown samples based on their features. This approach has shown promising results in detecting zero-day malware that cannot be detected by conventional methods. Authors in study [6] proposes Malware-SMELL, a zero-shot learning method for classifying malware using visual representation and a new S-Space representation, achieving 80% recall and outperforming other methods by 9.58% in classifying malware with a model trained solely on goodware code. Authors in study [26] argued that the use of sandboxing techniques can help detect zero-day malware. Sandboxing involves running an application in a controlled environment to observe its behaviour and identify any malicious activities. According to authors in study [26] analysing the interactions between the application and the sandbox make it possible to detect zero-day malware based on its behaviour. However, detecting zero-day malware remains a challenge [19]. Zero-day malware often employs obfuscation techniques to evade detection, making it difficult for existing solutions to identify them. Furthermore, some techniques may have limitations in terms of accuracy or the ability to detect complex malware [27].

Anomaly detection approach also have been utilized for detecting zero-day mal-ware by characterizing typical patterns and identifying malicious actions based on their deviation from normal patterns [28]. These techniques aim to identify anomalies or deviations from expected behavior, which can indicate the presence of zero-day attacks or malware. By comparing the behavior of an application or system to a base-line or normal profile, any deviations or anomalies can be flagged as potentially malicious. Hybrid methods that combine both anomaly detection and anomaly identification techniques have been proposed for detecting zero-day attacks. These methods leverage the strengths of both approaches to improve the accuracy and effectiveness of detection. Anomaly detection techniques can identify deviations from normal behavior, while anomaly identification techniques can classify these deviations as malicious or benign.

Unsupervised anomaly detection algorithms have also shown potential in detecting zero-day attacks [29]. These algorithms do not require labeled training data and can automatically learn patterns and identify anomalies in data. By analyzing the be-haviour of applications or systems, unsupervised algorithms can detect deviations from normal behavior and flag them as potential zero-day attacks. However, it is important to note that the performance of unsupervised algorithms for zero-day detection can be influenced by the availability of quantitative analyses and me-ta-learning techniques. Authors in study [3] proposed autoencoder architecture based on neural network for anomaly detection. The model was trained based on the benign instances. The aim is to create a model with no idea of high to reconstruct the malware instances as the model originally trained based on benign instances. Although autoencoder method is promising for binary classification, selecting proper threshold is challenging.

Generative adversarial networks (GANs) have been widely used for anomaly detection in various domains, including time series data, image processing, and network analysis [30-32]. In the context of anomaly detection, GANs have shown promise in capturing the normal patterns of data and identifying deviations from these patterns as anomalies. GAN was used in two approaches: unsupervised and semi-supervised anomaly detection. In the unsupervised anomaly detection GAN is trained solely on normal data without any labeled anomalies while in semi-supervised the GAN is trained on both normal and anomalous where a small portion of anomalous labels are minority class. Kolosnjaji et al. [33] leveraged data extracted from malware samples, including header fields, instruction sequences, and raw bytes, to train models that discriminate between benign and malicious software. By using GANs, they aimed to enhance the detection of adversarial malware binaries that can evade traditional deep learning-based detection methods. Although, GANs offer a promising avenue for anomaly detection by capturing the underlying patterns and distributions of data, the effectiveness of GANs for anomaly detection can be influenced by factors such as the quality and representativeness of the training data, the architecture and hyperparameters of the GAN, and the choice of anomaly scoring or thresholding methods. In mal-ware detection domain, GAN has not been investigated much in the literature for detecting malware threats. Some works focused on generating adversarial mal-ware samples [34]. Accordingly, a model is trained to classify the benign samples including the synthesis generated benign samples from the malware samples. Authors in study [8] proposed a zero-day malware detection model by training a generative adversarial network with deep autoencoder (DAE) using transfer learning.

In conclusion, existing zero-day malware detection solutions employ various approaches including signature and anomaly analysis. Various techniques are used in the machine learning and sandboxing analysis. These approaches aim to identify malicious patterns either based on static features or based on behavior that can indicate the presence of novel malware pattern. The signature based static features were the most employed form of zero-day detection in malware domain. While these techniques have shown promise in detecting zero-day malware, this approach assume that the zero-day malware is a malware variant that have known characteristics with the previous one. Such assumption is not accurate because zero-day malware may show different treats and might not follow any known patters due to the use of obfuscation techniques by malware authors. Few researchers employ the concept of

anomaly detection to device zero-day malware detection model by identifying deviations from normal patterns or behavior. Unsupervised and semi-supervised learning was utilized to train the anomaly detection models. Research using autoencoders [3, 8], GAN [8, 18], and CCN [4, 24, 35] architectures showed promise in detecting zero-day attacks. However, the selection of proper threshold that can discriminate the malware from benign is challenging task due to the overlapping features between the benign and malware instances caused using the obfuscation and evasion techniques. Though further research is needed to enhance their performance through quantitative analyses and meta-learning techniques.

To this end, this study devised a zero-day malware detection (Fig. 1) model through de-signing an architecture that incorporate GAN, deep autoencoding, and CCN to improve the detection rate while reduce the false alarm rate. The GAN architecture was trained on normal instances, to generate realistic benign samples. Our hypothesis is that benign samples exhibit less dynamism compared to malware samples, making them suitable for GAN-based generation to represent normal instances effectively. The deep autoencoding is trained to model benign distribution for anomaly detection leveraging the benign samples generated by the GAN networks to enhance representation and improve detection performance. To reduce the false alarm rate resulted from con-figuration of the anomaly detection threshold. A CNN architecture aimed at mitigating false positives generated by the autoencoder, particularly addressing the issue of feature overlap between benign and malware representations was designed and developed. The latent features extracted by the autoencoders were fused with the GAN discriminator's output to train the CNN model for robust differentiation between benign and malware instances. The detailed description of the proposed model is presented in the following section.

III. The Proposed Model

The proposed model has been constructed through five main phases features ex-traction and pre-processing, data representation, GAN model, the autoencoder model, and the CNN classifier. In the first phase, the malware features are extracted and pre-processed for the training. In the second phase, the GAN model is constructed using semi-supervised approach. The GAN model consists of two adversarial modules, a generator and a discriminator. The Generator and Discriminator always competes against each other. The generator tries to generate a fake sample look like benign software while discriminator try to recognize real sample as real and generated sample as fake. Autoencoders consist of two integral components: the encoder and the decoder. The encoder is responsible for transforming input data, which can encompass various types such as images or text, into a condensed representation known as a bottleneck or latent code, characterized by lower dimensions. Subsequently, the decoder's role is to utilize this latent code to perform an optimal reconstruction of the initial input data. The fundamental goal of an autoencoder is to minimize the reconstruction error, quantified as the disparity between the input data and the reconstructed output.

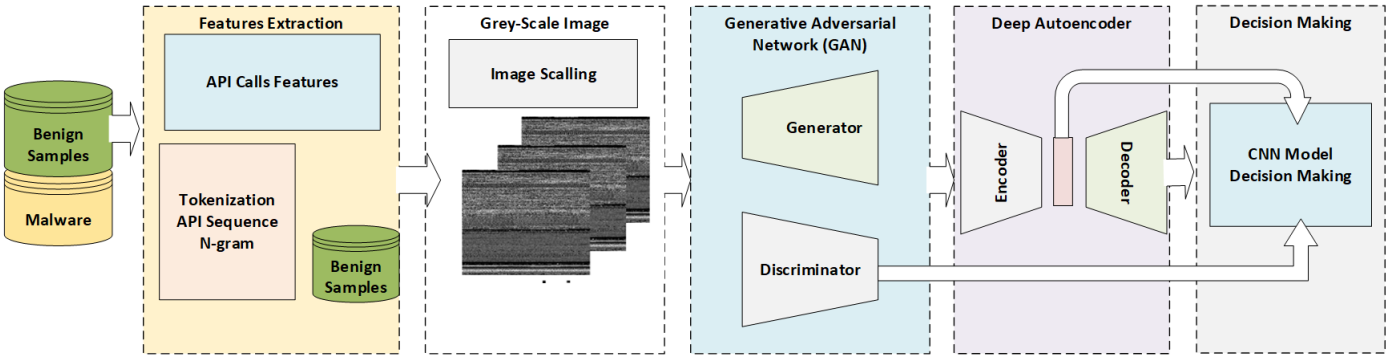

Fig. 1. The proposed zero-day malware detection model.

A. Features Extraction Phase

In this phase, malware features are extracted by monitoring and analyzing the interactions between an application (whether malicious or benign) and the operating system during runtime, specifically when API calls are made. Dynamic analysis is performed using the Cuckoo sandbox, which employs a technique called hooking to intercept and track API calls. Hooking works by injecting code into an application's execution flow, allowing the system to capture function calls to APIs. These intercepted calls are then logged into files, with each log file containing the recorded API calls of a specific application. For each application, the API calls are extracted from the log file and arranged sequentially based on their occurrence during execution. Each API function call is then treated as a distinct feature.

To enhance feature representation and capture behavioral patterns, n-gram analysis is applied to extract meaningful API call sequences. N-grams help identify important patterns in API sequences, making them a valuable technique for feature extraction. Numerous studies have validated the effectiveness of n-grams across various domains, including malware detection, where they improve classification accuracy by providing richer contextual information [36].

B. Data Representation Phase

In this study, each API calls and API sequence extracted using n-gram is used as a feature (term). Then the TF-IDF which is a well-established technique for feature ex-traction from text data, was used for representing the APIs features. TF-IDF considers both the frequency of terms (API calls in this case) within a sequence and their importance across multiple sequences [37]. It assigns higher weights to terms that are

frequent within a sequence but relatively rare across all sequences. This is useful in identifying unique or significant API call patterns associated with specific malware samples. TF-IDF helps in reducing the dimensionality of the feature space by focusing on the most relevant terms (API calls). This can make subsequent analysis and machine learning tasks more computationally efficient and interpretable. Rare or unique API calls that are common in malware but rare in legitimate applications can be weighted more heavily [10].

The TF/IDF vectors then are converted to image format. The process typically begins by reshaping the TF-IDF feature matrix into a grid-like structure, where each cell represents the TF-IDF score of a specific term (word) in a document. This grid, often referred to as a term-document matrix, forms the basis of the 2D image representation. To generate the image, TF-IDF scores are mapped to pixel intensities, converting the continuous values into values. Once the TF-IDF values are transformed into pixel values, the image is ready for the classification. According to study [10], the Inverse Document Frequency (IDF), which measures the global importance of an API across the entire corpus, can be calculated as follows:

$$idf_i = log\left(\frac{number\ of\ Applications}{(number\ of\ Applications\ call\ the\ API\ i+1)}\right) \quad (1)$$

where the idf_i is the inverse document frequency. TF-IDF is calculated by multiplying the TF (term frequency) and IDF (inverse document frequency) values for each term in each document. This results in a TF-IDF score for each API or API sequence in each document. It quantifies how unique or common a term is in the corpus. Next, for each feature in the corpus, the term frequency-inverse term frequency (tf_idf) is calculated as follows.

$$t\_idf_i = tf_i * idf_i \quad (2)$$

The t_idf_i score for a term in a document is higher if the term appears frequently in that document but is relatively rare across the entire corpus. The t_idf_i features are scaled using min-max normalization as follows.

$$scalled_{tf_{idf_{features}}} = \frac{tf\_idf\_features - min(tf\_idf\_features)}{max(tf\_idf\_features) - min(tf\_idf\_features} \quad (3)$$

Finally, the features vector is created from the unique terms of the corpus. The maximum length of the feature vector is n features. These features vector was converted to $w \times h$ image size as follows.

$$image\_width\ w = floor(\sqrt{n})\ ) \quad (4)$$

$$image\_hight\ h = floor\left(\left(\frac{(n-1)}{w}\right) + 1\right) \quad (5)$$

Where w and h are the width and height of the represented images and n is the max length of the features vector.

C. GAN Model Construction Phase

In this phase, the Generative Adversarial Network (GAN) model is constructed. GANs are a type of deep learning model that consists of a generator and a discriminator. The generator aims to generate synthetic data that resembles the real data, while the discriminator tries to distinguish between real and synthetic data. When the discriminator is no longer able to distinguish between real data and synthetic data, then the model is converging and can be used in the production. In this study the GAN is trained on the benign data samples. By training the GAN on a dataset of normal data (benign samples), it learns to capture the underlying distribution of the normal data [38]. GANs have emerged as a promising approach in the anomaly detection [38, 39]. The aim is to measure the anomaly score of given samples based on its deviation from the learned distribution of normal samples. This is done by comparing the reconstruction error of a given sample with the reconstruction error of the benign samples. This approach is promising and have been widely adopted by many researchers in the anomaly detection field [38, 39, 40].

The Generator was trained based on the benign samples as represented by images in the previous phase. The generator network learns to generate synthetic images that resemble the benign images, while the discriminator network learns to distinguish between real and synthetic images. Once the GAN is trained, the constructed GAMN uses an iterative process to find the latent vector in the generator network that best re-constructs a given test image. This is done by optimizing the latent vector to minimize the difference between the reconstructed image and the original test image. The anomaly score is then calculated based on the reconstruction loss and the loss between the intermediate discriminator feature of the test image and the reconstructed image. The generator is trained to reconstruct the samples represented by 1D vector extracted randomly from latent space and map them to 2D images in the image space created from the applications samples. The generator network is architected using stack of convolutional decoder equivalent to a convolutional decoder. The Discriminator D is constructed using standard CNN layers that maps 2D images to a single scalar represent the anomaly score of the sample. Fig. 2 shows the architecture of the proposed GAN network.

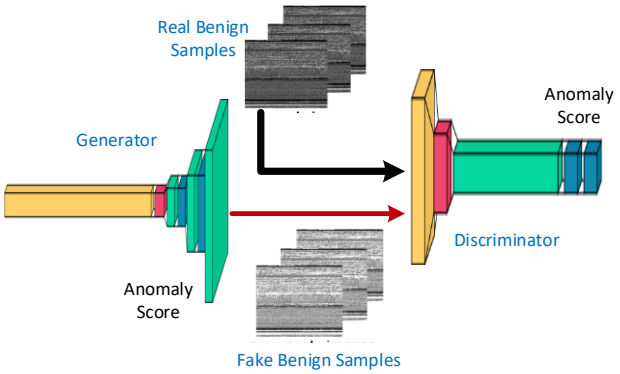

Fig. 2. The proposed semi-supervised GAN network.

The generator network of the GAN is trained to produce synthetic samples that look similar to the fraud samples, while the discriminator network is trained to distinguish between the original and the synthetic samples. For the learning, let G and D denote the generator and the discriminator, respectively, and let $Z = \{z1, z2, .. zn\}$ and $X = \{x1, x2, \dots, xn\}$ denote the distribution of latent and problem space, respectively. G and D

G(z) are the output of the generator (the fake sample) and $D(G(z))$ is the output of the discriminator, which is the probability of getting $G(z)$ belonging to real data. The error $e = log(1 - D(G(z))$ should be minimized to generate a fake sample that is drawn from the distribution of the real data. The error e is also used to penalize the generator G and thus to minimize $\log(D(x))$. Thus, based on [38], the following min-max game must be played by G and D to minimize the generator error and maximize the divergence.

$$\underbrace{min}_{G}\ \underbrace{max}_{D} V(G,D) = E_x\big(log\big(D(x)\big)\big) + E_z(log(1 - D(G(z)))) \quad (6)$$

The training of the GAN model continues until the generator can fool the discriminator into believing that the generated samples are real, namely when adversarial loss converges, indicating that the generator is producing realistic fraudulent samples.

D. Deep Autoencoder Construction Phase

It is widely believed by researchers that the performance of the anomaly detection using one class learning fall behind the supervised learning approach. This is because the classification approach does not relay much on selecting the classification thresholds as the model learn automatically the best discrimination threshold [3, 8]. The ability of neural network in performing abstractions is attractive. Considering this, it is reasonable to assume that autoencoders, a type of neural network specializing in encoding input data, would yield a latent representation that faithfully represents the specific attributes of input data samples. As a result, our strategy in this work relies on autoencoding to gain the benefits of strong abstraction and one class model to make judgments automatically and without the need for thresholds. In this study the auto-encoder based model was trained based on the benign samples. As shown in Fig. 3, the data with latent distribution was used to construct one class model for anomaly detection. The autoencoder learns how to minimize the reconstruction errors.

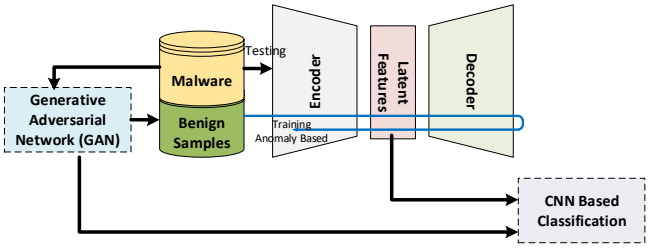

Fig. 3. The training and testing bath of the Autoencoder Anomaly based model.

E. CNN Classification Phase

In this stage, the latent features extracted from the autoencoder was used to develop a CNN classifier that can effectively distinguished between benign and malware samples. Fig. 4 shows the proposed CNN model for Decision Making about the anomaly status of the sample normal for benign samples and anomaly for malware samples.

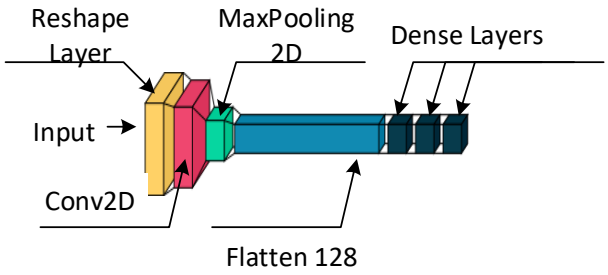

Fig. 4. The proposed CNN model for decision making.

The CNN Model consists of eight layers. The CNN model in this stage is designed for binary classification, where the sigmoid activation function is suitable for producing binary output probabilities (0 or 1) zero for normal and one for anomalies. The first layer defines the input shape, indicating that the model expects input data with a dimension of 256. The input layers are taken form the last hidden layer (the flatten layer) of the discriminator and concatenated with the latent layer form the discriminator to form the 256 input dimension. The second layer is used to transform the input data into a 16x16 grid with a single channel (grayscale image). The third layer is a 2D convolutional layer with 32 filters and a 3x3 kernel size. It uses the ReLU (Rectified Linear Unit) activation function, which introduces non-linearity into the model. The fourth layer performs max-pooling with a 2x2 pool size. Max-pooling reduces the spatial dimensions of the feature maps, helping to capture essential information while reducing computational complexity. The fifth layer is the flatten layer which reshapes the output from the previous layer into a one-dimensional vector. This prepares the data for fully connected layers. The sixth layer is a fully connected dense layer with 128 units and ReLU activation. The seventh layer is fully connected dense layer with 64 units and ReLU activation. The output layer with a single neuron and sigmoid activation.

IV. Experimental Design and Performance Evaluation

The dataset, the experimental procedures, and the performance evaluation are described in the following sub-sections.

A. Datasets

In this study, two datasets were used to validate and evaluate the proposed model. The first dataset, which is referred to Dataset I, is the API call sequences have been extracted form dynamic analysis environment. The malware samples were originally collected by [41, 42]. The extracted API call sequence represents behaviours of 7208 evasive malware sample. The benign samples, namely 3,848 benign, were collected from a newly installed copy of Windows 7 and from [43]. Fig. 5(a) illustrates the distribution of samples in Datasets I. The dataset was split into two parts 70% for training and 30% for testing. The 30% of the real benign samples represents the unseen benign samples while the whole malware samples were hidden during the training of the anomaly-based models in this study. As shown in Table I the model is trained based on the real and synthesized benign samples. For CNN model 70% of the malware samples were used in the training and 30% for the testing.

The second dataset referred as Dataset II which is publicly available online and can be downloaded from IEEEDataPort Web portal [44]. The dataset contains 10,654 samples 3,097 are benign samples while 7557 are malware samples. The malware samples distributed as follows, 451 ransomware, 1,051miner, 797 DDoS Trojan, 89 worm, 3353 infective virus, 454 backdoor, and 1362 trojan (see Fig. 5(b)). Table I presents the distribution of samples in Datasets I and II for training and testing, including both real and generated benign and malware samples. To enrich the datasets, the GAN model was used to generate diverse sets of benign samples, enhancing the training process and improving model performance. Accordingly, 2469 benign samples were used for the training of the GAN network and 14814 benign samples used for the training of the deep autoencoding model.

B. Performance Measures

To evaluate the detection performance of the proposed model, we utilized five key performance metrics, namely overall accuracy, detection rate (recall), precision, F1 score, false-positive rate (FPR), and false-negative rate (FNR). These metrics are widely acknowledged and commonly employed in the assessment of malware detection solutions within the existing body of literature. The performance metrics utilized in this study were computed using the following formulas.

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \quad (7)$$

$$FPR = \frac{FP}{TP+FN} \quad (8)$$

$$FNR = \frac{FN}{TN+FP} \quad (9)$$

$$DR\ (Recall) = \frac{TP}{TP+FN} \quad (10)$$

$$Precision = \frac{TP}{TP+FP} \quad (11)$$

$$\text{F1 Score} = \frac{2\times Precision \times Recall}{Precision+Recall} \quad (12)$$

The F1 score measures the balance between accuracy and recall to assess the model's overall performance. True positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) are all equally considered by MCC. As a result, it provides additional information about the model's performance. Table 1 lists the samples used for Fig. 5.

TABLE I. DATASETS I AND II SAMPLES DISTRIBUTION

	Dataset I			Dataset II		
	Training		Testing	Training		Testing
	Real	Generated		Real	Generated	
Benign	4694	9388	1154	2469	12345	1503
Malware	5045 (for CNN only)	-	2162	6054 (for CNN only)	-	628
Total	19127		3316	20868		2131

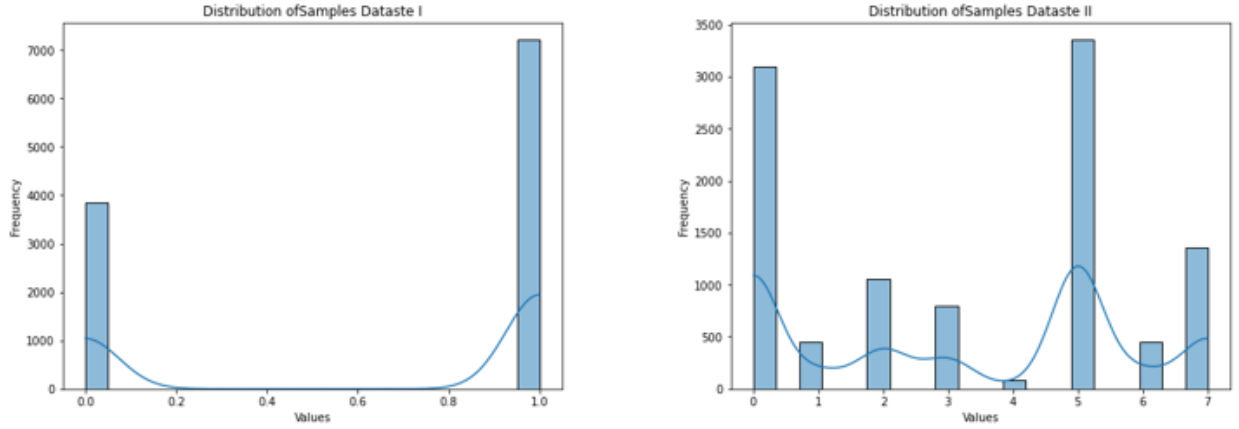

Fig. 5. (a) Dataset I samples distribution (b) Dataset II samples distribution.

C. Evaluation Procedure

In this study, extensive experiments were conducted to evaluate the proposed model. Because CNN was reported by many researchers to have promising classification performance, two different anomaly models were trained for the comparison. The benign samples were used to train the CNN model. The input is the features vector represented as image based on n-gram and TF/IDF features extraction and presentation schemes. The output of these models are the anomaly scores of the samples. The autoencoder model which can be considered a type of semi supervised learning to con-struct anomaly detection model was implemented in this study for the evaluation. The autoencoder model is trained based on the benign samples. The aim is to minimize the reconstruction error of the benign samples. However, in case of the malware which is considered zero-day attack for the anomaly-based model the

construction error likely to be greater than the errors generated by reconstructing the benign samples because the model has not been learnt to represent the malware instances [3]. Although autoencoder is promising for the anomaly detection, selecting proper threshold is challenging task. Therefore, in this study, the autoencoder model was implemented according to the model presented in study [3]. The autoencoder was also cascaded with the CNN model for the comparison. Both one- and two-dimensions image representation were used in the experiments. The autoencoder model firstly trained using the normal samples and then the latent space features were used to train the CNN classifier. Generative Adversarial Network (GAN) based model with autoencoder were also implemented for the comparison. GAN models were widely used for anomaly detection in literature due to their ability of generating samples similar the minority class instances and their ability to model high dimensional data distribution [32]. The GAN model is trained to regenerate the normal samples and the autoencoder was trained based on the data generated by the GAN model. In doing so, a variety of noise that resample the normal data is included in the representation.

V. RESULTS

Table II and Fig. 6 (a)-(f) present a comparison of the performance of the pro-posed model with other models using dataset I. The proposed AEDGAN outperforms all other models, achieving a remarkable 95% accuracy and precision, a 93% detection rate (recall), and an impressive 94% overall accuracy. The false positive rate is only 5%, with a corresponding 5% reduction in the false negative rate. Notably, the CNN models with 2D representation exhibit superior performance compared to the other models studied. It is worth noting that the CNN model without the autoencoder out-performs the CNN model with autoencoder, primarily because the CNN model's supervised learning approach enables effective discrimination between benign and malware samples.

Table III and Fig. 7 (a)-(f) present the classification performance of the proposed model compared to the other models using Dataset II. The proposed model AEDGAN achieved the highest performance, attaining an 88% overall accuracy in terms of F1 Score, while all the other models scored lower than 84% overall performance. Notably, the proposed AEDGAN significantly reduces the false positive rate to 10%, compared to 26%, 21%, 23%, and 35% for AEGAN, AECNN(2D), AECNN(1D), and AE models, respectively.

TABLE II. PERFORMANCE COMPARISON BASED ON DATASET I

	Accuracy	Precision	Recall	F1 Score	FNR	FPR
CNN(1D)	0.90	0.94	0.82	0.88	0.12	0.06
CNN(2D)	0.92	0.95	0.86	0.90	0.09	0.05
AE	0.84	0.79	0.83	0.81	0.13	0.21
AECNN(1D)	0.90	0.87	0.91	0.89	0.07	0.13
AECNN(2D)	0.91	0.88	0.92	0.90	0.06	0.12
AEGAN	0.87	0.85	0.84	0.85	0.11	0.15
AEDGAN	0.95	0.95	0.93	0.94	0.05	0.05

TABLE III. PERFORMANCE COMPARISON BASED ON DATASET II

	Accuracy	Precision	Recall	F1 Score	FNR	FPR
CNN(1D)	0.89	0.89	0.70	0.78	0.12	0.11
CNN(2D)	0.90	0.90	0.76	0.82	0.09	0.10
AE	0.80	0.65	0.71	0.68	0.13	0.35
AECNN(1D)	0.88	0.77	0.83	0.80	0.07	0.23
AECNN(2D)	0.89	0.79	0.86	0.83	0.06	0.21
AEGAN	0.84	0.74	0.72	0.73	0.11	0.26
AEDGAN	0.93	0.90	0.87	0.88	0.05	0.10

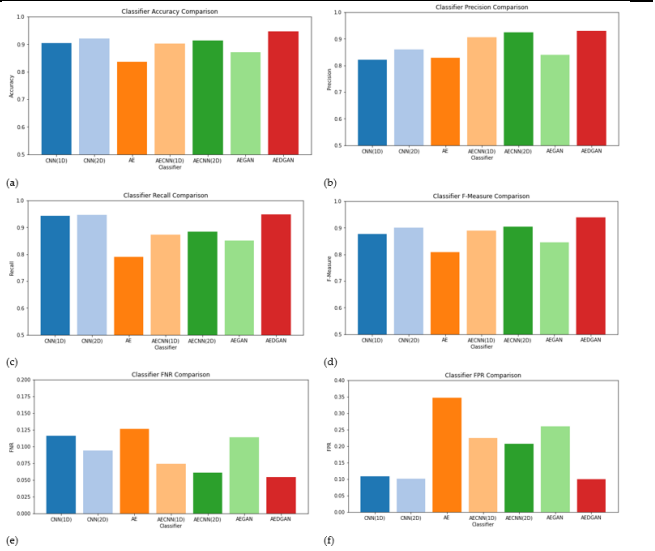

Fig. 6. Comparison of the detection performance (Dataset I) in terms of (a) Accuracy, (b) Precision, (c) Recall, (d) F-measure, (e) False negative rate, and (f) False positive rate.

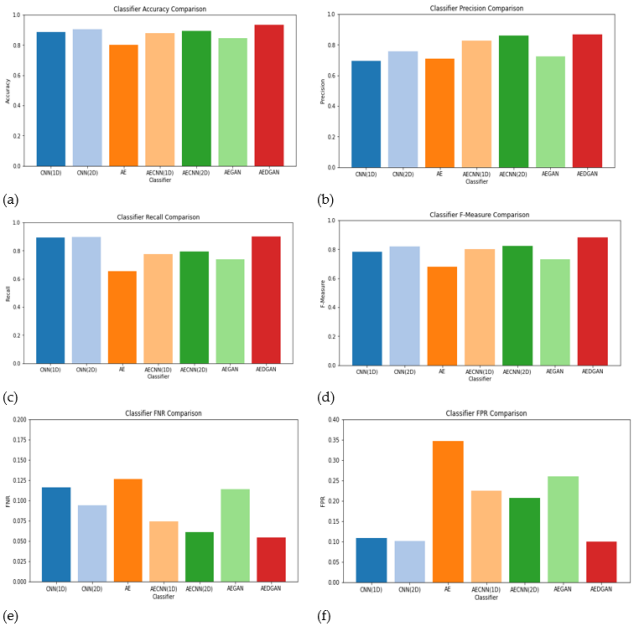

Fig. 7. Comparison of the detection performance (Dataset II) in terms of (a) Accuracy, (b) Precision, (c) Recall, (d) F-measure, (e) False negative rate, and (f) False positive rate.

VI. DISCUSSION

The results indicate that the autoencoder exhibited the poorest performance compared to the other models under study. This can be attributed to the possibility that the features learned by the autoencoder may not adequately represent benign samples, resulting in low detection accuracy, as indicated by the precision score for the AE model in Table 2. Furthermore, the output of the autoencoder requires additional analysis, and the detection relies on identifying an appropriate threshold for the constructed error. In contrast, the CNN models outperform the autoencoder model due to their capability to learn high-level features that effectively discriminate benign samples from malware samples.

In terms of the false alarm rate, the proposed model outperforms the others, achieving a low 5% rate for both false positives and false negatives. In contrast, the AE and AECNN models fail to strike a balance between false positives and false negatives, with high false positive rates due to the challenge of determining an appropriate threshold for distinguishing between benign and malware instances. The overlapping features of malware and benign samples, caused by the obfuscation nature of malware, hinder effective discrimination. Even with adversarial networks enhancing the representation of benign instances, the AEGAN still exhibits high false positives. Notably, the proposed AEDGAN substantially reduces the false positive rate to 5%, compared to 15%, 12%, 13%, and 21% for AEGAN, AECNN (2D), AECNN(1D), and AE models, respectively.

It can be observed from both Table II and Table III that the proposed model outperforms all other models studied for both datasets. However, the model's performance with Dataset II is inferior to its performance with Dataset I. The reason behind this discrepancy is that in Dataset I, the benign samples were extracted from the Windows 7 operating system, which exhibited distinguishable traits compared to the malware samples. On the other hand, the benign samples in Dataset II were derived from applications developed by a more diverse range of developers. Applications developed by Microsoft or integrated into the Windows OS by Microsoft may possess distinct API call sequences, especially in areas such as authentication and error handling, when compared to those developed by other software development firms.

Despite its promise, the proposed model has several limitations. One major challenge is the higher false positive rate (10%, as shown in Table II). This is because the second dataset contains a diverse set of malware samples from different families, leading to greater feature variability and overlap between benign and malicious applications. Such diversity makes it more difficult for the model to accurately distinguish between benign and malware instances, increasing the likelihood of false positives. Additionally, the model exhibits a lower detection rate for certain types of malwares, particularly those that employ obfuscation or evasive techniques, resulting in significant feature overlap with benign applications. This overlap makes it difficult for the model to reliably distinguish between malicious and non-malicious behavior. Moreover, while GAN-based data augmentation enhances generalization, the generated synthetic data may not fully capture the complexity of real-world benign applications, potentially introducing biases. The computational complexity of training GANs, autoencoders, and CNNs together also poses a challenge, making real-time malware detection in resource-constrained environments difficult. Furthermore, the model's effectiveness in practical, real-world scenarios remains uncertain, as it has not been extensively tested against evolving malware threats outside controlled environments. To address these issues, incorporating ensemble methods, leveraging diverse feature sets, and conducting real-world evaluations could further enhance the model's accuracy and robustness. Another key limitation is the dataset itself. The datasets used in this study may be quite obsolete (Windows 7), and based on our best knowledge, there is a lack of newly available datasets for malware detection. This limitation may affect the generalizability of our findings to more recent threats. In the future, we plan to collect datasets from newer versions of Windows to enhance the relevance and effectiveness of our detection methods.

VII. CONCLUSION

In this study, an anomaly-based zero-day anomaly-based malware detection model utilizing semi-supervised deep learning has been designed and developed. The model's development comprises three main phases: In the initial phase, we trained a Generative Adversarial Network (GAN) to acquire representations of benign applications, enabling the detection of malware and malicious applications. Given the relative stability of benign application behavior compared to malicious behavior, GAN-based data augmentation contributes to the generality and stability of the detection model. Furthermore, GAN is leveraged to generate a diverse set of synthetic data closely resembling real-world benign samples, thereby enhancing the model's capability to distinguish malware instances in subsequent learning stages. The second phase involved the development of an autoencoder, aimed at learning latent representations of benign samples and capturing essential features that characterize benign applications. In the third and final phase, we concatenated the latent representation with the last hidden layer of the GAN discriminator, representing them as an image. Subsequently, a Convolutional Neural Network (CNN) classifier was constructed to classify samples as either benign or malicious. This CNN model obviates the need for threshold selection to identify anomalous instances. The results indicate that the proposed model holds promise for detecting zero-day malware. In the worst-case scenario, it achieved an overall performance of 88% accuracy with a 10% false positive rate, surpassing the best existing solution by 5% in overall performance and reducing the false positive rate by 11%.

Despite its promise, the proposed model exhibits a lower detection rate and a higher false positive rate. The primary challenge lies in the inherent overlap between benign and malware features. The obfuscation and evasive characteristics of malware often lead to feature overlap between these classes. To address this challenge, we advocate for the use of a diverse set of features in the representation. Furthermore, we propose an ensemble approach involving anomaly detection models trained on diverse feature sets, incorporating both GAN and Autoencoder models, to enhance detection accuracy and mitigate false alarms.

ACKNOWLEDGMENT

This Project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (GPIP: 1826-611-2024). The authors, therefore, acknowledge with thanks DSR for technical and financial support.

FUNDING

This Project was funded by the Deanship of Scientific Research (DSR) at King Abdulaziz University, Jeddah, under grant no. (GPIP: 1826-611-2024). The authors, therefore, acknowledge with thanks DSR for technical and financial support.

REFERENCES

[1] AVTEST. "Malware Statistics and Trends Report," 5/10/2023, 2024; https://www.av-test.org/en/statistics/malware/.

[2] NPR. "What we know about the ransomware attack on a Critical u.s. pipeline," 5/10/2023, 2023; https://www.npr.org/2021/05/10/995405459/what-we-know-about-the-ransomware-attack-on-a-critical-u-s-pipeline.

[3] C. Kim, S. Y. Chang, J. Kim, D. Lee, and J. Kim, "Automated, Reliable Zero-day Malware Detection based on Autoencoding Architecture," IEEE Transactions on Network and Service Management, pp. 1-1, 2023.

[4] Q. Wen, and K. P. Chow, "CNN based zero-day malware detection using small binary segments," Forensic Science International: Digital Investigation, vol. 38, pp. 301128, 2021/10/01/, 2021.

[5] H. Tran, E. Campos-Nanez, P. Fomin, and J. Wasek, "Cyber resilience recovery model to combat zero-day malware attacks," Computers & Security, vol. 61, pp. 19-31, 2016/08/01/, 2016.

[6] P. H. Barros, E. T. C. Chagas, L. B. Oliveira, F. Queiroz, and H. S. Ramos, "Malware-SMELL: A zero-shot learning strategy for detecting zero-day vulnerabilities," Computers & Security, vol. 120, pp. 102785, 2022/09/01/, 2022.

[7] S. Millar, N. McLaughlin, J. Martinez del Rincon, and P. Miller, "Multi-view deep learning for zero-day Android malware detection," Journal of Information Security and Applications, vol. 58, pp. 102718, 2021/05/01/, 2021.

[8] J.-Y. Kim, S.-J. Bu, and S.-B. Cho, "Zero-day malware detection using transferred generative adversarial networks based on deep autoencoders," Information Sciences, vol. 460-461, pp. 83-102, 2018/09/01/, 2018.

[9] N. Kumar, S. Mukhopadhyay, M. Gupta, A. Handa, and S. K. Shukla, "Malware Classification using Early Stage Behavioural Analysis." pp. 16-23.

[10] F. A. Aboaoja, A. Zainal, F. A. Ghaleb, N. S. Alghamdi, F. Saeed, and H. Alhuwayji, "A Kullback-Liebler di-vergence-based representation algorithm for malware detection," PeerJ Computer Science, vol. 9, pp. e1492, 2023.

[11] A. A. Al-Hashmi, F. A. Ghaleb, A. Al-Marghilani, A. E. Yahya, S. A. Ebad, M. Saqib, and A. A. Darem, "Deep-Ensemble and Multifaceted Behavioural Malware Variant Detection Model," IEEE Access, vol. 10, pp. 42762-42777, 2022.

[12] J. Palša, N. Ádám, J. Hurtuk, E. Chovancová, B. Madoš, M. Chovanec, and S. Kocan, "MLMD—A Mal-ware-Detecting Antivirus Tool Based on the XGBoost Machine Learning Algorithm," Applied Sciences, vol. 12, no. 13, pp. 6672, 2022.

[13] A. A. Darem, F. A. Ghaleb, A. A. Al-Hashmi, J. H. Abawajy, S. M. Alanazi, and A. Y. Al-Rezami, "An Adaptive Behavioural-Based Incremental Batch Learning Malware Variants Detection Model Using Concept Drift Detec-tion and Sequential Deep Learning," IEEE Access, vol. 9, pp. 97180-97196, 2021.

[14] S. Baek, J. Jeon, B. Jeong, and Y.-S. Jeong, "Two-stage hybrid malware detection using deep learning," Hu-man-centric Computing and Information Sciences, vol. 11, no. 27, pp. 10.22967, 2021.

[15] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth, and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery." pp. 146-157.

[16] A. Abusitta, G. H. de Carvalho, O. A. Wahab, T. Halabi, B. C. Fung, and S. Al Mamoori, "Deep learning-enabled anomaly detection for IoT systems," Internet of Things, vol. 21, pp. 100656, 2023.

[17] N.-A. Stoian, "Machine learning for anomaly detection in iot networks: Malware analysis on the iot-23 data set," University of Twente, 2020.

[18] D. O. Won, Y. N. Jang, and S. W. Lee, "PlausMal-GAN: Plausible Malware Training Based on Generative Ad-versarial Networks for Analogous Zero-Day Malware Detection," IEEE Transactions on Emerging Topics in Computing, vol. 11, no. 1, pp. 82-94, 2023.

[19] M. A. Ashawa, and S. Morris, "Analysis of android malware detection techniques: a systematic review," 2019.

[20] E. Amer, I. Zelinka, and S. El-Sappagh, "A Multi-Perspective malware detection approach through behavioural fusion of API call sequence," Computers & Security, vol. 110, pp. 102449, 2021/11/01/, 2021.

[21] J.-Y. Kim, and S.-B. Cho, "Obfuscated Malware Detection Using Deep Generative Model based on Global/Local Features," Computers & Security, vol. 112, pp. 102501, 2022/01/01/, 2022.

[22] S. Srinivasan, R. Vinayakumar, A. Arunachalam, M. Alazab, and K. Soman, "DURLD: Malicious URL Detection Using Deep Learning-Based Character Level Representations," Malware Analysis Using Artificial Intelligence and Deep Learning, pp. 535-554: Springer, 2021.

[23] R. Elnaggar, L. Servadei, S. Mathur, R. Wille, W. Ecker, and K. Chakrabarty, "Accurate and Robust Malware Detection: Running XGBoost on Runtime Data From Performance Counters," IEEE Transactions on Comput-er-Aided Design of Integrated Circuits and Systems, vol. 41, no. 7, pp. 2066-2079, 2021.

[24] S. Saadat, and V. Joseph Raymond, "Malware classification using cnn-xgboost model," Artificial Intelligence Techniques for Advanced Computing Applications, pp. 191-202: Springer, 2021.

[25] T. A. A. Abdullah, W. Ali, and R. Abdulghafor, "Empirical Study on Intelligent Android Malware Detection Based on Supervised Machine Learning," International Journal of Advanced Computer Science and Applica-tions, 2020.

[26] F. Alhaidari, and A. Rahman, "ZeVigilante: Detecting Zero-Day Malware Using Machine Learning and Sand-boxing Analysis Techniques," Computational Intelligence and Neuroscience, 2022.

[27] A. Arfeen, Z. H. Khan, R. Uddin, and U. Ahsan, "Toward Accurate and Intelligent Detection of Malware," Concurrency and Computation Practice and Experience, 2021.

[28] S. Manimurugan, S. Almutairi, M. Aborokbah, N. Chilamkurti, S. Ganesan, and R. Patan, "Effective Attack Detection in Internet of Medical Things Smart Environment Using a Deep Belief Neural Network," Ieee Access, 2020.

[29] T. Zoppi, A. Ceccarelli, and A. Bondavalli, "Unsupervised Algorithms to Detect Zero-Day Attacks: Strategy and Application," Ieee Access, 2021.

[30] M. Dietrichstein, D. Major, M. Wimmer, D. Lenis, P. Winter, A. Berg, T. Neubauer, and K. Bühler, "Anomaly Detection Using Generative Models and Sum-Product Networks in Mammography Scans," 2022.

[31] X. Gong, X. Wang, and N. Li, "Research on DUAL-ADGAN Model for Anomaly Detection Method in Time-Series Data," Computational Intelligence and Neuroscience, 2022.

[32] X. Xia, X. Pan, N. Li, X. He, L. Ma, X. Zhang, and N. Ding, "GAN-based anomaly detection: A review," Neu-rocomputing, vol. 493, pp. 497-535, 2022/07/07/, 2022.

[33] B. Kolosnjaji, A. Demontis, B. Biggio, D. Maiorca, G. Giacinto, C. Eckert, and F. Roli, "Adversarial Malware Binaries: Evading Deep Learning for Malware Detection in Executables," 2018.

[34] D. Li, and Q. Li, "Adversarial deep ensemble: Evasion attacks and defenses for malware detection," IEEE Transactions on Information Forensics and Security, vol. 15, pp. 3886-3900, 2020.

[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," Computers & Security, vol. 84, pp. 376-392, 2019/07/01/, 2019.

[36] B. M. Khammas, A. Monemi, I. Ismail, S. M. Nor, and M. Marsono, "Metamorphic malware detection based on support vector machine classification of malware sub-signatures," TELKOMNIKA (Telecommunication Com-puting Electronics and Control), vol. 14, no. 3, pp. 1157-1165, 2016.

[37] B. A. S. Al-Rimy, M. A. Maarof, M. Alazab, F. Alsolami, S. Z. M. Shaid, F. A. Ghaleb, T. Al-Hadhrami, and A. M. Ali, "A Pseudo Feedback-Based Annotated TF-IDF Technique for Dynamic Crypto-Ransomware Pre-Encryption Boundary Delineation and Features Extraction," IEEE Access, vol. 8, pp. 140586-140598, 2020.

[38] F. Di Mattia, P. Galeone, M. De Simoni, and E. Ghelfi, "A Survey on GANs for Anomaly Detection," 2019.

[39] H. Zenati, M. Romain, C. S. Foo, B. Lecouat, and V. Chandrasekhar, "Adversarially Learned Anomaly Detec-tion," 2018.

[40] C. P. Ngo, A. A. Winarto, C. K. K. Li, S. J. Park, F. Akram, and H. K. Lee, "Fence GAN: Towards Better Anomaly Detection," 2019.

[41] N. Galloro, M. Polino, M. Carminati, A. Continella, and S. Zanero, "A Systematical and longitudinal study of evasive behaviours in windows malware," Computers & Security, vol. 113, pp. 102550, 2022/02/01/, 2022.

[42] D. Kirat, and G. Vigna, "MalGene: Automatic Extraction of Malware Analysis Evasion Signature," in Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security, Denver, Colorado, USA, 2015, pp. 769–780.

[43] C. Wei, Q. Li, D. Guo, and X. Meng, "Toward Identifying APT Malware through API System Calls," Security and Communication Networks, vol. 2021, pp. 8077220, 2021/12/09, 2021.

[44] Z. Zhang. "MALWARE_API_CLASSIFICATION," 12/06/2023, 2023; https://ieee-dataport.org/documents/malwareapiclassification