IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 23 February 2021; revised 12 March 2022; accepted 22 April 2022.
Date of publication 3 May 2022; date of current version 6 March 2023.

Digital Object Identifier 10.1109/TETC.2022.3170544

PlausMal-GAN: Plausible Malware Training Based on Generative Adversarial Networks for Analogous Zero-Day Malware Detection

DONG-OK WON, (Member, IEEE), YONG-NAM JANG, AND SEONG-WHAN LEE, (Fellow, IEEE)

Dong-Ok Won is with the Department of Artificial Intelligence Convergence, Hallym University, Chuncheon 24252, South Korea
Yong-Nam Jang is with the Department of Brain and Cognitive Engineering, Korea University, Seoul 02841, South Korea
Seong-Whan Lee is with the Department of Artificial Intelligence, Korea University, Seoul 02841, South Korea

CORRESPONDING AUTHOR: SEONG-WHAN LEE (sw.lee@korea.ac.kr)

This work was supported in part by the Institute of Information & communications Technology Planning & Evaluation (IITP) Grant funded by the Korea government (MSIT) under Grant 2019-0-00079 and in part by Artificial Intelligence Graduate School Program (Korea University) under Grant 2021-0-02068, Artificial Intelligence Innovation Hub.

ABSTRACT Zero-day malicious software (malware) refers to a previously unknown or newly discovered software vulnerability. The fundamental objective of this paper is to enhance detection for analogous zero-day malware by efficient learning to plausible generated data. To detect zero-day malware, we proposed a malware training framework based on the generated analogous malware data using generative adversarial networks (PlausMal-GAN). Thus, the PlausMal-GAN can suitably produce analogous zero-day malware images with high quality and high diversity from the existing malware data. The discriminator, as a detector, learns various malware features using both real and generated malware images. In terms of performance, the proposed framework showed higher and more stable performances for the analogous zero-day malware images, which can be assumed to be analogous zero-day malware data. We obtained reliable accuracy performances in the proposed PlausMal-GAN framework with representative GAN models (i.e., deep convolutional GAN, least-squares GAN, Wasserstein GAN with gradient penalty, and evolutionary GAN). These results indicate that the use of the proposed framework is beneficial for the detection and prediction of numerous and analogous zero-day malware data from noted malware when developing and updating malware detection systems.

INDEX TERMS Analogous malware detection, generative adversarial networks, malware augmentation, malware data, zero-day malware

I. INTRODUCTION

Malware can be defined as malicious software that is designed to cause outages, denial of activity, collection of personal data without user consent, unauthorized access to system resources, and similar inappropriate behaviors. With the rapid development of information technology, the exponential increase in malware has become one of the main threats to computer security [1]–[3]. Malicious software detection has become more difficult as the number and variety of applications increase in computer security [4]–[6], with more than 143 thousand new malicious programs targeting mobile devices detected during 2013 [5], and as Kaspersky Lab's research shows that nearly 30% of all computers were threatened at least once during 2018 [7].

Zero-day malware is an unknown or unaddressed software vulnerability that hackers use to do malicious things, such as destroying programs, stealing data, or paralyzing networks [8]. A range of antivirus systems and other strategies are used to help protect against the introduction of malware, which helps in detection if such malware is already present. Antivirus systems typically fail to detect zero-day malware because they rely on signatures to identify malware. Computers are more vulnerable to zero-day malware than to general malware because traditional antivirus systems typically cannot detect zero-day malware. Zero-day malware is an important threat to computer security, and zero-day malware detection is a top priority for malware detection systems.

To detect zero-day malware, we propose a deep learning method of generating arbitrarily modified malware features using the malware's raw code without running it. Malware code based on specific rules and actions generates certain

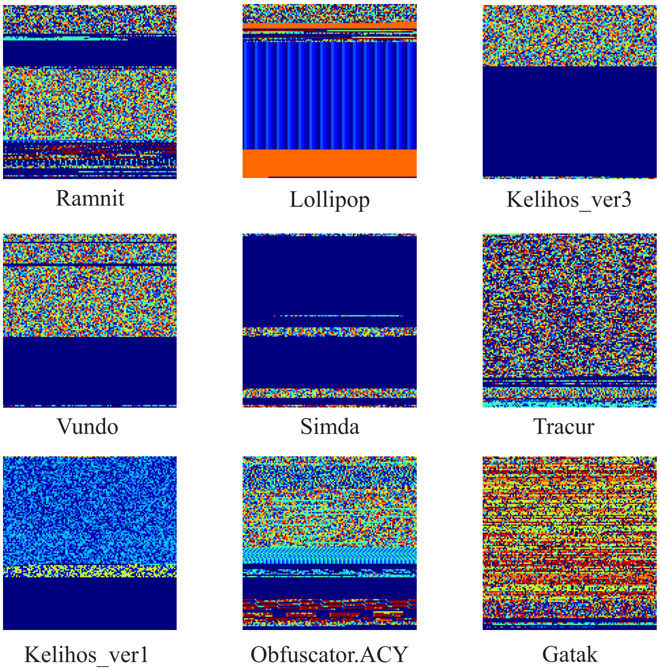

FIGURE 1. **Examples of nine type classes of malware images.**

patterns. Examples of the malware sample used in this study are shown in Figures 1 and 11 [9], [10].

While, when dealing with classification tasks using neural networks, data augmentation techniques have been used to compensate for imbalance or data insufficiency problems. In the malware detection research area, several papers also used simple data augmentation techniques (e.g., sliding window, transformation, etc.) to deal with these issues [11], [12].

In this study, we investigated and focused on the different direction of malware training technique with generating zero-day malware data, not focused imbalance or data insufficiency. We proposed a plausible malware training framework capable of detecting analogous zero-day malware that can handle newly plausible malware (Plausible malware training framework based on generative adversarial networks, PlausMal-GAN). Our main contribution is the proposed malware training framework based on generative adversarial networks (GAN) with generated analogous malware samples. The proposed framework trains a generator and discriminator based on real malware data and the generated malware data in the first phase. In the second phase, the generator is fixed and the discriminator is re-trained based on real malware data and the generated malware data by the fixed generator. Ideally, the proposed framework can apply any kind of GAN model, so we evaluated the performance by applying the latest and repetitive GAN models. Moreover, we obtained stable performance for abundant analogous zero-day malware test data in relatively few training data conditions.

II. BACKGROUND

A. *MALWARE DETECTION*

Owing to the increasing damage caused by malware and zero-day malware, research on malware detection methods have been continuously improving. We discuss two aspects of malware detection: malware detection and zero-day malware detection.

Several reported studies have dealt with malware detection [10], [13]–[17]. Nataraj *et al.* presented a visualization approach that differs from traditional approaches for malware detection [10], where they transformed the malware's binary information into grayscale malware images. Ye *et al.* and Ndibanje *et al.* used Windows Audit Log and API Call for malware detection [18], [19]. Traditional machine learning algorithms such as hidden Markov models, support vector machines (SVMs) and random forests were also used for malware detection [20]–[23]. Singh *et al.* proposed a Big Data analysis framework based on random forests for malware detection [24]. Chen *et al.* attempted to detect malware by analyzing mobile network traffic with machine-learning methods [25]. Recently, there have been many methods to use deep learning and generative adversarial networks (GAN) because the available computing power has increased [11], [12], [26]–[31]. Pascanu *et al.* used recurrent neural networks for time-series information in malware classification [26], [32]. Ye *et al.* presented a heterogeneous deep-learning framework composed of an autoencoder stacked up with a layer of associative memory and multilayer restricted Boltzmann machines [27]. Kabanga *et al.* used data from converted malware images as input to the convolutional neural networks (CNNs) [28]. Yan *et al.* used CNN and long short-term memory networks to learn from grayscale image and opcode sequence, respectively, and takes a stacking ensemble for malware classification [11]. The aforementioned methods have disadvantages that detect only certain variants of malware. The developers of malware use obfuscation techniques, such as null byte injection, code exchange, and subroutine reordering, to create new variants with signatures different from existing malware. However, the aforementioned methods use malware that has been discovered so far. Thus, unlearned malware will not be detected. To detect attacks that bypass deep-learning methods [33], Wang *et al.* proposed a resistant method that is robust to adversarial malware samples by nullifying arbitrary features [33]. However, in this way, malware characteristics are randomly removed, which risks removing not only unnecessary features but also important ones. There are now hybrid methods that combine static and dynamic methods [22], [34]. While these methods can be effective for malware detection, they have the disadvantage of being time-consuming and highly complex.

Recently, there have been some methods developed for zero-day malware detection [13], [14], [35], [36]. Venkatraman and Alazab used a similarity matrix of malware for visualization in order to detect zero-day malware [14]. This method can be used to visually observe that different malware families exhibit significantly different behavior patterns. Gupta and Rani proposed a Big Data framework to address the Big Data problem caused by increase in malware [35]. They also attempted to detect zero-day malware using Big Data analysis techniques and machine-learning algorithms. This method modeled a series of opcodes to detect zero-day malware. Due to the increasing threat of malware in a cyber-physical system, Huda *et al.* proposed a detection method that uses methods like SVM and K-means to detect unknown malware by extracting knowledge and essential structures from already unlabeled, cheap, available data [36]. In the aforementioned zero-day malware detection methods, certain rules are fixed, and zero-day malware that does not follow these rules cannot be detected. Recently, Kim *et al.* has proposed transferred deep-convolutional generative adversarial network (tDCGAN), which generates fake malware and learns to distinguish it from real malware [13]. This method obtained not only enhanced performance in malware detection but also showed possibility in a zero-day attack experiment. Since the method is no consideration of high diversity (e.g., plausible diversity) or quality in generated zero-day malware, nor was it measured numerically (i.e., frÃ©chet inception distance, etc.), it is difficult to assume that focused on zero-day malware detection. While, we implemented analogous zero-day malware classifier with GAN models to create new high-diversity and high-quality malware images for generating plausible malware augmentation. The generated data is used to create a robust detector for zero-day malware detection.

B. DATA AUGMENTATION

Data Augmentation encompasses a suite of techniques that enhance the size and quality of training datasets such that better deep learning models can be built using them [37], [38]. The simple data augmentations based on basic image manipulations are flipping, cropping, rotation, translation, etc [37], [38]. Recently, GAN based approach refers to the practice of creating artificial instances from a dataset such that they retain similar characteristics to the original set [39], [40]. In malware detection, several papers applied data augmentation method to solve imbalance or data insufficiency issues [12], [41].

To our best knowledge, there have been no studies to date which focused on the high diversity and quality of plausible malware in terms of analogous malware augmentation, which is an important factor to be investigated for various transformations or analogous data augmentation using a zero-day malware detection system. In this study, we proposed a plausible malware training framework based on GAN that could consider high diversity in generating analogous zero-day malware data. Moreover, the proposed method showed stable performance even with relatively little training data. We applied different kinds of several recent GAN models (i.e., deep convolutional GAN (DCGAN) [42], least-squares GAN (LSGAN) [43], Wasserstein GAN with gradient penalty (WGAN-GP) [44], evolutionary GAN (E-GAN) [40]) to our design, it could be shown as a potentially reliable adaptation in state-of-the-art GAN models.

C. GENERATIVE ADVERSARIAL NETWORKS

GAN [39] is a deep-learning model that emerged for the purpose of generating data similar to the training data using the given training data. Unlike the original GAN, which uses only one objective function (e.g., minimax), Wang *et al.* proposed E-GAN [40] using several objective functions (i.e., minimax, heuristic, and least-squares). Generators using each objective function are evaluated by a discriminator, and the best-performing generator is chosen to evolve to the next stage. In the process of evolution, the evolved generator is expected to gradually adapt to the discriminator, which means that the evolved generator can provide high-quality, high-diversity samples and learn the real data distribution. The evolutionary process consists of three stages (i.e., variation, evaluation, and selection):

First, the variation stage used the variation operators to produce its offspring $\{G_{\theta_1}, G_{\theta_2}, \ldots\}$, given an individual G_θ in the population. In particular, several copies of each individual or parent were created, each of which was modified by different mutations. Then, each modified copy is regarded as one child. Second, in the evaluation stage, we evaluated the performance or individual quality for each child by a fitness function $\mathcal{F}$ that depends on the current environment (i.e., discriminator D). Third, in the selection stage, we selected all children according to their values and removed the worst ones. The rest remained alive (i.e., free to act as parents) and evolved to the next iteration.

Compared to the generator using multiple objective functions, the discriminator is the same as the objective function of the original GAN. The discriminator D is trained to distinguish between the real data sample $x \sim p_{data}(x)$ and the generated data sample $\hat{x} \sim p_{gen}(\hat{x})$

$$L_D = -\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{\hat{x} \sim p_{\text{gen}}}[\log(1 - D(\hat{x}))]. \tag{1}$$

III. METHODS

In this section, we describe a plausible malware training framework based on generative adversarial networks (GAN) that generates analogous malware with a malware classifier and training discriminator as a malware detector. Figure 2 is an architectures of our proposed framework.

A. PLAUSMAL-GAN FRAMEWORK

To generate analogous malware samples for each kind of malware, the proposed framework trains a generator and discriminator based on GAN with a malware classifier using real malware data and the generated malware data in the first step. The discriminator not only discriminates real or fake, but also learns to classify malware classes. In the second step, the generator is fixed and the discriminator is re-trained based on real malware data and the generated malware data by the fixed generator. Figure 3 shows the overview and process of the proposed framework. The auxiliary classifier GAN (AC-GAN) [45] proposed a structure that produces data that matches class labels as well as data that are close to real data. For malware classifier, the architectures of the proposed framework is following the AC-GAN structures (Figure 2). Our malware generator generates fake malware samples $\hat{x}$ that contain noise sample z by malware class c, and discriminator not only distinguishes between real $x \sim p_{data}(x)$ and fake $\hat{x} \sim p_{gen}(\hat{x})$ but also class c. The difference between our method

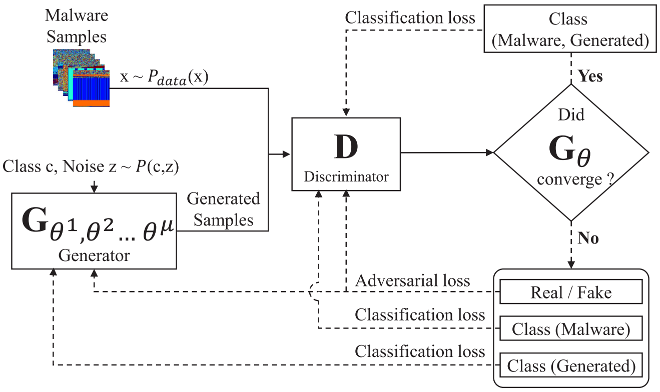

FIGURE 2. **The architectures of the proposed framework for analogous zero-day malware detection.**

and the existing AC-GAN is that the discriminator does not learn the class information of the generated malware sample, only the class information of the real malware sample. Our discrimination training loss is defined as follows:

$$L_D = -\mathbb{E}_{x \sim p_{\text{data}}} [\log D(x) - \log p(c|x)] - \mathbb{E}_{\hat{x} \sim p_{\text{gen}}} [\log (1 - D(\hat{x}))]. \quad (2)$$

And, we considered standard GAN approach (minmax), least-squares approach, heuristic approach, and combining the preceding three-approach for DCGAN, LSGAN, WGAN-GP, and E-GAN model in the proposed framework, respectively. In E-GAN, we considered an evolutionary step consists of three sub-steps: variation, evaluation, and selection. In the variation step, we adopt three objectives that are interpretable and complementary as mutations proposed by Wang *et al.* [40]. As shown in Figure 4, the difference between the three objective functions are minimax mutation, heuristic mutation, and least-squares mutation. In addition, we added a classification loss function to the existing mutation functions, because not only the data is close to real but also data corresponding to the class must be generated. The minimax mutation is similar to the minimax objective function of the original GAN, which aimed to minimize the log probability that the discriminator

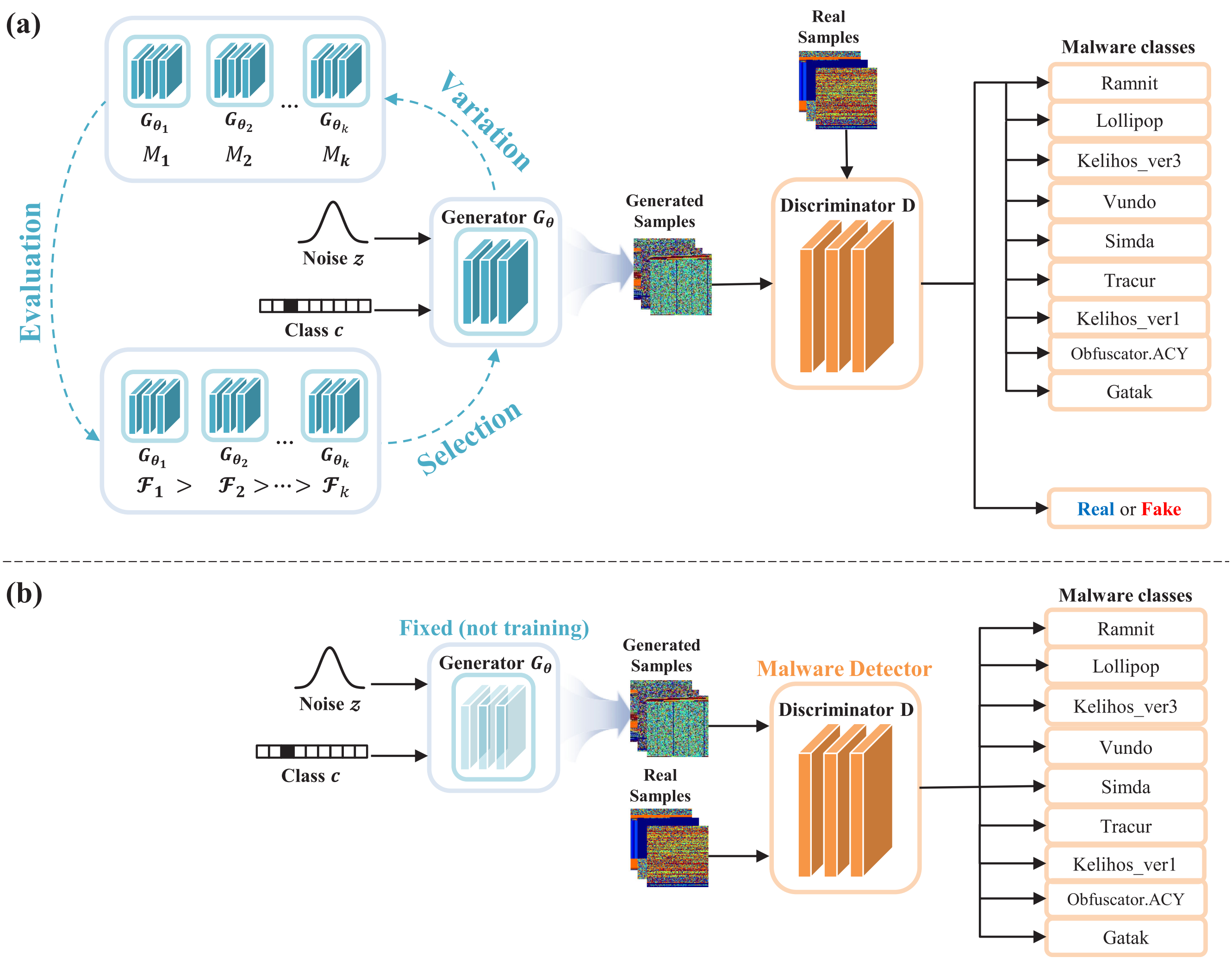

FIGURE 3. **The proposed PlausMal-GAN framework consists of two-phases. (a) The generator and discriminator training based on GAN with malware classifier. (b) Training the discriminator as a zero-day malware detector from plausible malware augmentation. For an intuitive explanation, it is shown using evolutionary GAN, which is one of the representative GANs.**

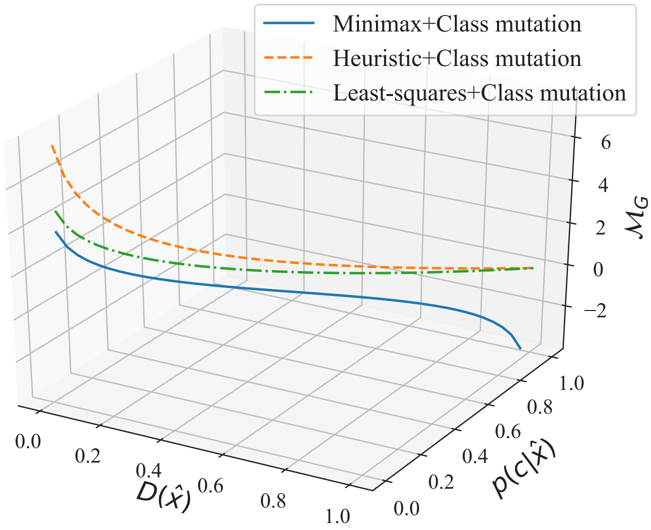

FIGURE 4. Mutation (or objective) functions with classification loss function.

would do well. In the original GAN, gradient vanishing can occur when the discriminator produces a result close to zero (i.e., $D(\hat{x}) \rightarrow 0$). In other words, if the discriminator is confident that the generated malware data is fake malware data, the generator may not train well. However, we have been able to solve this problem to some extent by adding a classification loss. Unlike early gentle gradients, if the generated malware distribution is somewhat similar to the real malware distribution, the minimax mutation provides a steep gradient, which later allows stable learning

$$\mathcal{M}_G^{\text{minimax}} = \mathbb{E}_{\hat{x}\sim p_{\text{gen}}}[\log(1 - D(\hat{x})) - \log p(c|\hat{x})]. \tag{3}$$

The heuristic mutation minimizes the log probability that the discriminator will do well, which maximizes the log probability that the discriminator will go wrong. Using this mutation, the gradient is steep even though the discriminator is convinced that the generated malware data is fake. Thus, the heuristic mutation can avoid a vanishing gradient, unlike the minimax mutation, which suggests the possibility of better learning in the early stages than the minimax mutation

$$\mathcal{M}_G^{\text{heuristic}} = -\mathbb{E}_{\hat{x}\sim p_{\text{gen}}}[\log(D(\hat{x})) + \log p(c|\hat{x})]. \tag{4}$$

Lastly, the least-squares mutation is similar to the least-squares objective function of the LSGAN, which aimed at deceiving the discriminator by penalizing the generator. Using this mutation, we get a gentle slope overall and can avoid a vanishing gradient as in a heuristic mutation. Besides, least-squares mutations, when compared to heuristic mutations, do not assign very high costs to generate fake malware samples but do not assign very low costs to mode dropping, which partially avoids mode collapse [43]

$$\mathcal{M}_G^{\text{least-s.}} = \mathbb{E}_{\hat{x}\sim p_{\text{gen}}}\left[(D(\hat{x}) - 1)^2 - \log p(c|\hat{x})\right]. \tag{5}$$

Algorithm 1. Plausible Malware Training Framework (i.e., With E-GAN Case)

```
Require: batch size $m = 32$. discriminator's updating steps per iteration $n_D = 1$; number of parents $\mu = 1$; number of mutations $n_m = 3$; Adam hyper-parameters $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.99$; the hyper-parameter $\gamma$ of evaluation function.
Require: initial discriminator's parameters $w_0$. initial generator's parameters $\{\theta_0^1, \theta_0^2, \ldots, \theta_0^\mu\}$.
    for number of training iterations do
        for $k$ = 0,..., $n_D$ do
            Sample a batch of $\{x^{(i)}\}_{i=1}^m \sim p_{data}$ (training data), and a batch of $\{(c,z)^{(i)}\}_{i=1}^m \sim p_{c,z}$ (noise sample $z$ by class $c$).
            $g_w \leftarrow \nabla_w \Big[ \frac{1}{m}\sum_{i=1}^m \log D_w(x^{(i)}) + \frac{1}{m}\sum_{j=1}^{\mu}\sum_{i=1}^{m/\mu} \log(1 - D_w(G_{\theta^j}((c,z)^{(i)}))) + \frac{1}{m}\sum_{j=1}^{\mu}\sum_{i=1}^{m/\mu} \log p(c^{(i)}|x^{(i)}) \Big]$
            $w \leftarrow \text{Adam}(g_w, w, \alpha, \beta_1, \beta_2)$
        end for
        for $j$ = 0,..., $\mu$ do
            for $h$ = 0,..., $n_m$ do
                Sample a batch of $\{(c,z)^{(i)}\}_{i=1}^m \sim p_{c,z}$ (noise sample $z$ by class $c$).
                $g_{\theta^{j,h}} \leftarrow \nabla_{\theta^j} \left[ \mathcal{M}_G^h \left( \left\{ (c,z)^{(i)} \right\}_{i=1}^m, \theta^j \right) \right]$
                $\theta_{\text{child}}^{j,h} \leftarrow \text{Adam}(g_{\theta^{j,h}}, \theta^j, \alpha, \beta_1, \beta_2)$
                $\mathcal{F}^{j,h} \leftarrow \mathcal{F}_q^{j,h} + \gamma\mathcal{F}_d^{j,h}$
            end for
        end for
        $\{\mathcal{F}^{j_1,h_1}, \mathcal{F}^{j_2,h_2}, \ldots\} \leftarrow \text{sort}(\{\mathcal{F}^{j,h}\})$
        $\theta^1, \theta^2, \ldots, \theta^\mu \leftarrow \theta_{\text{child}}^{j_1,h_1}, \theta_{\text{child}}^{j_2,h_2}, \ldots, \theta_{\text{child}}^{j_\mu,h_\mu}$
end for
```

In the evaluation step, the 1) malware quality and 2) diversity of the generated malware samples are measured and evaluated. To detect zero-day malware, it was important to generate samples of high-diversity malware with high quality, so we adopted the evaluation step of the E-GAN architecture. First, the quality fitness score was used as a measure of quality. This method puts the generated malware image based on the noise sample by class into discriminator D and uses the output value. We use the output of D multiplied by the probability of that class to measure the image quality score for each class. And, we use the average output value. The closer the value is to 1, the closer to reality the malware data is. In other words, the closer to 1, the higher quality malware data

$$\mathcal{F}_q = \mathbb{E}_{\hat{x}\sim p_{\text{gen}}}[D(\hat{x}) \times \log p(c|\hat{x})]. \tag{6}$$

Second, the diversity fitness score is used as a measure of malware diversity. This method uses the minus log-gradient-norm of the discriminator. When the generator generates data that greatly changes the gradient of the discriminator, the discriminator is likely to determine that the generated malware data is fake. In contrast, when the generator generates data that does not change the discriminator gradient significantly, the generated malware data is not labeled as fake and tends to achieve high diversity

$$\mathcal{F}_d = -\log ||\nabla_D||. \tag{7}$$

Using the two fitness scores mentioned above, the criterion for the E-GAN evaluation is as follows:

$$\mathcal{F} = \mathcal{F}_q + \gamma\mathcal{F}_d \tag{8}$$

where $\gamma > 0$ is the balance between the quality and diversity measurements.

In the selection step, the offspring with the highest fitness score is selected and proceeds to the next variation step. Throughout the evolution process, the generator will gradually generate data for each class as well as generating data similar to real data. We use the converged generator for malware detection in the next step.

3.2 MALWARE DETECTION

For analogous zero-day malware augmentation, the malware generator generates high-quality and high-diversity images. We use the discriminator's classifier as a malware detector. The discriminator has trained anew as a malware detector without adversarial training with the generator. As a malware detector, the discriminator is trained using both generated and real malware images. The objective function of the discriminator is as redefined

$$L_D = -\mathbb{E}_{x\sim p_{\text{data}}}[\log p(c|x)] - \mathbb{E}_{\hat{x}\sim p_{\text{gen}}}[\log p(c|\hat{x})] \tag{9}$$

when training the discriminator, the generator is not trained and only generates malware images. Figure 3 shows training the discriminator with data augmentation as a malware detector.

IV. EXPERIMENTS AND RESULTS

This section describes the experiments and results for evaluating the proposed framework.

A. DATASETS

1) MICROSOFT MALWARE CLASSIFICATION CHALLENGE DATASET

To verify the data generation and detection performance of the proposed framework, we used a malware data from the Microsoft dataset [9]. The malware file was a byte file, and we used binary code written to it. The total number of malware is 10,868, divided into 9,781 training sets and 1,087 test sets (9:1 train-test ratio). Appendix B, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TETC.2022.3170544, shows the malware data types used and the number of malware for each malware type [9].

Algorithm 2. Training Discriminator Based on the Proposed Framework

```
Require: batch size $m = 32$; discriminator's updating steps per iteration $n_D = 1$; Adam hyper-parameters $\alpha = 0.0002, \beta_1 = 0.5, \beta_2 = 0.99$.
Require: initial discriminator's parameters $w_0$; initial generator's parameters $\theta_0$.
  for number of training iterations do
    for k=0,...,$n_D$ do
      Sample a batch of $\{x^{(i)}\}_{i=1}^m \sim p_{data}$ (training data), and a batch of $\{(c,z)^{(i)}\}_{i=1}^m \sim p_{c,z}$ (noise sample $z$ by class $c$).
      $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^m \log p(c^{(i)}|x^{(i)}) + \frac{1}{m}\sum_{i=1}^m \log p(c^{(i)}|G_\theta((c,z)^{(i)}))\right]$
      $w \leftarrow \text{Adam}(g_w, w, \alpha, \beta_1, \beta_2)$
    end for
  end for
```

As Nataraj *et al.* did [10], we convert malware binary code into an image called *malware image*. If k is the length of the binary code, C is the size of the converted column, and R is the size of the converted row, this is how to calculate the size of the converted columns and rows

$$C = 2^{\frac{\log\sqrt{16k}}{\log 2}+1} \tag{10}$$

$$R = \frac{16\ k}{C} \tag{11}$$

The malware images were so large that they were reduced to 128 × 128 using Pillow which python image library. Then we used jet colormaps to represent RGB color images.

2) MALIMG DATASET

In Supplementary Materials Appendix C, available in the online supplemental material, we show the frequency distribution of malware families and their variants in the Malimg dataset [10]. We were able to find malware data from malware class that shared the family name (i.e., Worm: Allaple.A and Allaple.L, PWS: C2Lop.gen!G and C2Lop.P, Trojan: Lolyda.AA1 and Lolyda.AA2, TDownloader: Swizzor.gen!I and Swizzor.gen!E). In Table 1 and Figure 11, eight different malware data have four pairs with two different and similar family names and shared similar properties. For the second zero-day malware experiments, we evaluated malware data with similar

TABLE 1. **Malware Data With Similar Family Names in the Malimg Dataset for the Second Zero-Day Malware Experiment.**

Malware family names	Type	No. of Variants
Allaple.A	Worm	2949
Allaple.L	Worm	1591
C2Lop.gen!G	PWS	200
C2Lop.P	PWS	146
Lolyda.AA1	Trojan	213
Lolyda.AA2	Trojan	184
Swizzor.gen!I	TDownloader	132
Swizzor.gen!E	TDownloader	128

properties family in the Malimg dataset, which consists of 5,543 malware samples from 8 different malware families.

B. EXPERIMENTAL DETAILS

The experiment is divided into two parts: a existing malware classification and a analogous zero-day malware attack experiments. In the existing malware classification experiment, we compared the proposed framework with representative GANs (i.e., DCGAN, LSGAN, WGAN-GP, and E-GAN) and previous methods experimental results [13]. In the proposed framework, we used the same network structure (Supplementary Table S2, available online). In the first analogous zero-day malware attack experiment, we also compared our framework with the four GAN models and previous methods results (i.e., random forest, decision tree, nearest neighbors, Naive Bayes, multi-layer perceptron (MLP) [46], CNN [47], GAN [39], and tDCGAN [13]). In the second zero-day malware experiment, we compared the proposed framework phase 1 and phases 1&2 with the representative four GAN models.

The operating system of the computer used in the experiments was Ubuntu 16.04.2 LTS, and the central processing unit was Intel Xeon Gold 6148. The random-access memory was Samsung DDR4 16 GB × 4, and the graphics processing unit was TITAN XP. When implementing the proposed framework, we used the Pytorch library. The generative and discriminative network architectures used in the generator and discriminator respectively, are shown in Supplementary Table S2, available online.

TABLE 2. **FIDs Between Generated Malware Images and Real Malware Images in the Proposed Framework.**

Model	DCGAN	LSGAN	WGAN-GP	E-GAN ($r = 0.1$, $r = 0.5$)
FID	220.16	190.70	206.23	146.39, **127.96**

C. ANALYSIS OF GENERATED MALWARE DATA

Figure 5 shows examples of the generated malware images using the Microsoft dataset [9]. In qualitative terms, Figure 5 shows the generation of malware images that are similar to the real malware images, which shows that the proposed framework can also generate modified malware or analogous zero-day malware.

We choose the Fréchet inception distance (FID) [48] as a quantitative metric for evaluating generator convergence. The FID uses pre-trained Inception v3 networks to extract features of the generated images and real images. Then model the data distribution for extracted features using a multivariate Gaussian distribution with mean μ and covariance Σ. The FID between the real images x and generated images g is computed as below

$$\begin{aligned}\mathrm{FID}(x,g) = {} & \left\|\mu_x - \mu_g\right\|_2^2 \\ & + \mathrm{Tr}\left(\Sigma_x + \Sigma_g - 2\left(\Sigma_x\Sigma_g\right)^{\frac{1}{2}}\right), \end{aligned} \tag{12}$$

where Tr is the sum of all the diagonal elements.

A lower FID implies that the distribution distance between the real images and generated images is closer. It also means that the generated images have high quality and high diversity. As shown in Table 2, our proposed framework has the lowest FID score. This means that the generator of our proposed framework generated a high-quality and high-diversity malware sample. While low FIDs do not actually produce new malware, it is likely a variant of existing malware. This allows us to expect data augmentation with the generated data.

D. MALWARE CLASSIFICATION

To derive a more accurate estimate of model prediction performance, we used 10-fold cross-validation for all methods and it was used for the existing malware classification experiment

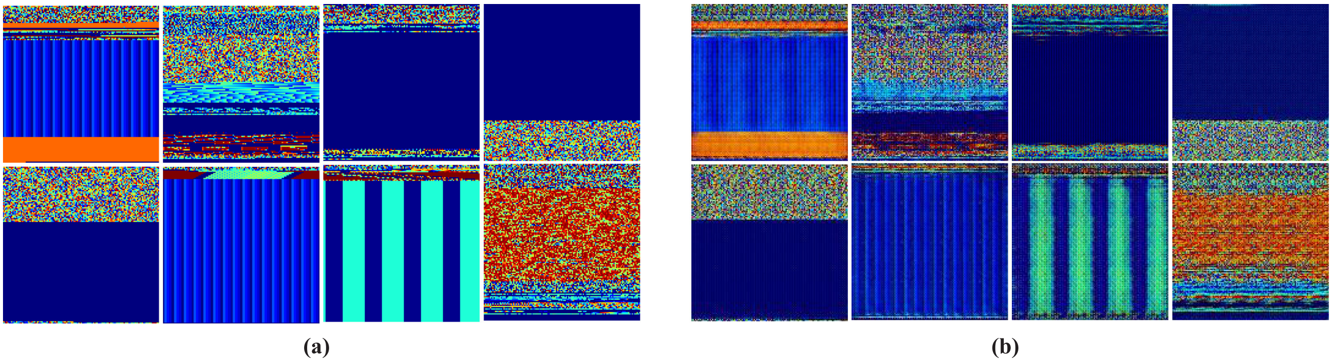

FIGURE 5. **Examples of (a) real malware images and (b) generated malware images in the proposed framework.**

TABLE 3. Comparison of Malware Classification Accuracies in the Proposed Framework With Four Representative GAN Models and Previous Methods.

Model	MLP	CNN	GAN	tGAN	Proposed Framework (PlausMal-GAN)			
					DCGAN	LSGAN	WGAN-GP	E-GAN
Accuracy (%)	83.06	94.63	87.81	88.10	94.99	96.02	94.86	**96.35**
Std. dev.	7.54e-04	2.12e-05	3.44e-05	8.05e-05	0.596	0.351	0.255	0.539

using the Microsoft dataset [9]. The average classification accuracy achieved by the proposed framework was 95.56%, which means that the performance of our proposed framework was much better than the previous methods. Table 3 shows the numerical classification results with four difference models (i.e., DCGAN, LSGAN, WGAN-GP, and E-GAN). Because the performance was the most dominant when using the E-GAN model, only the proposed framework with this model was used for some further analysis (i.e., Table 4, Figures 7 and 9).

To verify the performance of the proposed malware classifier model, we showed a confusion matrix in Figure 7. We calculated the precision, recall, and F1-score for each malware type and summarized them in Table 4. Also, we compared the classification accuracies for the proposed framework with difference four GAN models according to the training iterations in Figure 6. In results, the E-GAN models showed higher classification performance than other Representative models.

E. ZERO-DAY MALWARE

1) ZERO-DAY MALWARE EXPERIMENT I USING GENERATED ANALOGOUS ZERO-DAY MALWARE

We modeled plausible zero-day malware for analogous zero-day malware attack experiments using the Microsoft dataset (Figure 8) [9]. The previous study assumed that the zero-day attacks can be modeled by introducing noise into existing malware data [13]. The noise was generated by the structure similarity (SSIM) method, which uses the structural similarity of images [49]. We likewise used the SSIM method for systematic noise generation. The method of calculating the SSIM values for a pair of images x, y includes calculating μ_x, μ_y as the means for the pixels of the images x, y.

$$\mathrm{SSIM}(x,y) = \frac{\left(2\mu_x\mu_y + c_1\right)\left(2\sigma_{xy} + c_2\right)}{\left(\mu_x^2 + \mu_y^2 + c_1\right)\left(\sigma_x^2 + \sigma_y^2 + c_2\right)} \tag{13}$$

where, $c_1 = (k_1L)^2, c_2 = (k_2L)^2, k_1 = 0.01, k_2 = 0.03, L = 2^{\#\,bits}\ per\,pixel - 1$.

TABLE 4. Results of Precision, Recall, and F1-Score for Each Malware Type in the Proposed Framework.

	R	L	K3	V	S	T	K1	O	G
Precision	0.954	0.971	0.996	0.854	0.666	0.864	1.000	0.982	0.960
Recall	0.961	0.975	0.993	0.979	0.500	0.933	0.975	0.894	0.950
F1-score	0.957	0.973	0.994	0.912	0.571	0.897	0.987	0.936	0.955

We used altered malware images with 0.02 intervals between 0.60 and 0.68 (SSIM value) for analogous zero-day malware evaluation [13], [49]. Then, for more diverse zero-day malware evaluation, we regenerated the transformed malware images with two combined ratios such as 7:3 and 8:2 ratios. We also compared the proposed framework with previous methods results (in 8:2 combined ratio) [13] (Table 5). The plausible zero-day malware modeling with noise is calculated as follows:

$$N_k(x,y) = (1-\xi)x + \xi y, \tag{14}$$

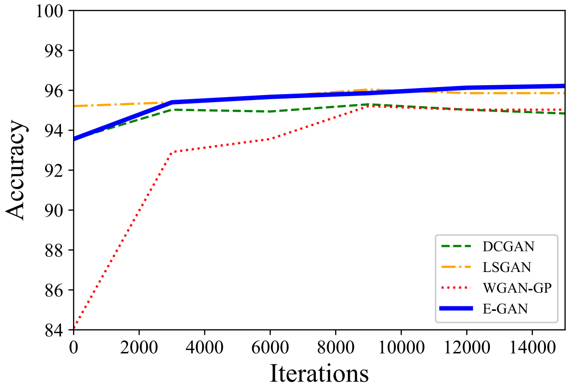

FIGURE 6. Classification accuracy according to the training iterations for the proposed framework with four representative models.

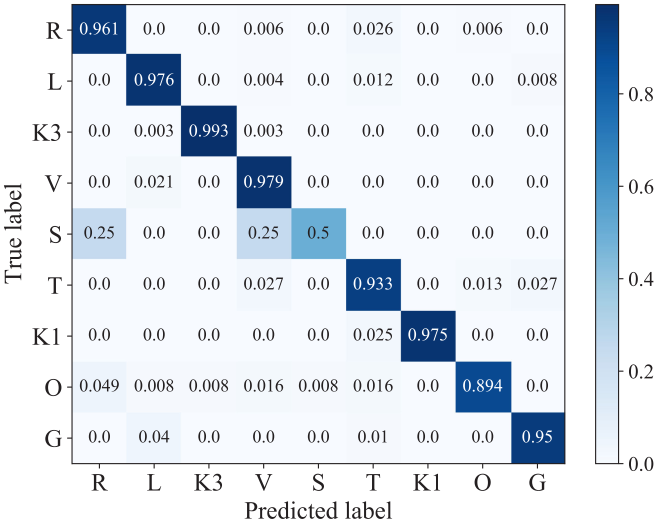

FIGURE 7. Confusion matrix for malware classification results in 9:1 train-test ratio.

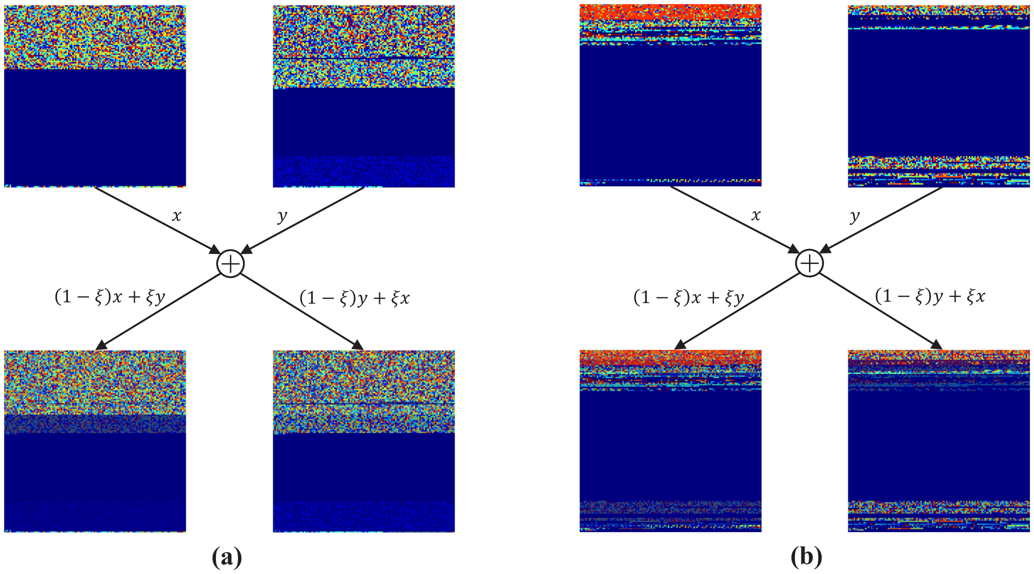

FIGURE 8. **Examples of plausible zero-day malware with SSIM values of (a) 0.60 and (b) 0.68.**

where $\mathrm{SSIM}(x, y) > k$, ξ is 0.3, 0.2 in combined ratio 7:3 and 8:2, respectively. Figure 8 shows examples of deformed plausible zero-day malware.

The results of the analogous zero-day malware attack experiment in Table 5 divided the malware images into an experiment with an 8:2 combined ratio and a 7:3 combined ratio. We used 10-fold cross-validation (i.e., the train-test ratios: 9:1). In 8:2 combined ratio experiments, the proposed frameworks' models were more accurate than other previous recent methods [13], and we obtained stable accuracy performance in our frameworks with tested GAN models in all SSIM conditions. Moreover, in the 7:3 combined ratio experiments, we also obtained reliable high averaged performance 98.62%, 98.37%, 98.51%, and 99.49% for the proposed framework methods with DCGAN, LSGAN, WGAN-GP, and E-GAN model, respectively. In particular, the decreasing SSIM values or combined high noise ratio could be an analogous zero-day attack compared to existing malware, but the proposed framework showed stable performances in any SSIM values or combined ratios. As a result, the proposed framework obtained high and stable performance even the large variations of existing malware (e.g., combined ratio 7:3 or SSIM value 0.6) in a analogous zero-day malware attack. Moreover, we were conducted in few training data condition by the changing train-test ratios experiment (9:1→5:5) for a thorough performance verification

TABLE 5. **Comparison of Analogous Zero-Day Malware Attack Performances in Two Difference Combined Rates (CR) for the Proposed Framework and Previous Methods (%).**

Model \ SSIM	0.60	0.62	0.64	0.66	0.68	CR
Random Forest	91.28	95.19	92.88	95.58	91.40	8:2
Decision Tree	95.64	96.46	96.71	96.41	96.18	
Nearest Neighbors	97.71	97.72	98.36	98.34	98.09	
Naive Bayes	90.60	90.89	91.51	91.16	90.45	
MLP	96.78	96.46	97.26	97.23	96.82	
CNN	98.16	98.23	98.63	98.61	98.41	
GAN	96.32	96.96	96.99	96.95	96.50	
tGAN	97.24	96.96	97.81	97.78	97.45	
tDCGAN	98.39	98.73	98.63	98.61	98.41	
Proposed framework						
(*with DCGAN*)	99.59	99.66	99.60	98.58	99.54	
(*with LSGAN*)	99.43	99.66	99.64	98.41	99.54	
(*with WGAN-GP*)	99.59	99.66	99.60	98.58	98.63	
(*with E-GAN*)	**99.94**	**100.0**	**99.74**	**99.58**	**99.84**	
Proposed framework						7:3
(*with DCGAN*)	99.02	99.25	99.28	96.52	99.05	
(*with LSGAN*)	97.28	99.10	99.42	97.00	99.05	
(*with WGAN-GP*)	99.02	99.70	99.38	96.52	97.94	
(*with E-GAN*)	**99.86**	**100.0**	**99.51**	**98.42**	**99.68**	

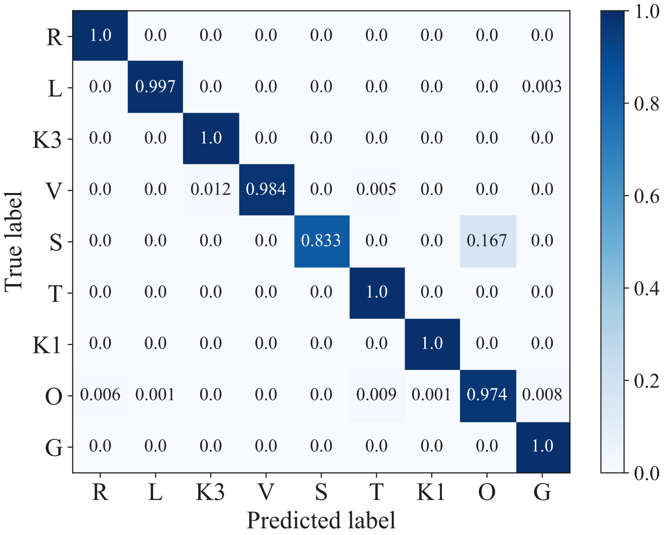

FIGURE 9. **Confusion matrix for zero-day malware classification results in 5:5 train-test ratio with 8:2 combined ratio and 0.64 SSIM.**

TABLE 6. Results of Performances for Zero-Day Malware Attack Experiment in the Few Training Data Conditions (%).

Model \ SSIM	0.60	0.62	0.64	0.66	0.68	CR
Proposed framework						
(*with DCGAN*)	97.58	98.74	98.83	97.76	97.73	8:2
(*with LSGAN*)	98.79	99.17	99.29	98.79	98.74	
(*with WGAN-GP*)	97.92	98.73	98.96	98.43	98.23	
(*with E-GAN*)	**99.11**	**99.42**	**99.51**	**99.06**	**99.25**	
Proposed framework						
(*with DCGAN*)	97.99	98.82	98.96	98.00	98.09	7:3
(*with LSGAN*)	98.98	99.18	99.64	99.06	98.78	
(*with WGAN-GP*)	98.10	98.64	99.04	98.33	98.30	
(*with E-GAN*)	**99.40**	**99.67**	**99.70**	**99.31**	**99.13**	

TABLE 7. Comparison of Zero-Day Malware Classification Accuracies for Second Zero-Day Experiment (Malimg Dataset) in the Proposed Framework With Only Phase 1 and Whole Phases.

Proposed framework	Train session	DCGAN	LSGAN	WGAN-GP	E-GAN
with only	A	87.70	82.67	85.06	**90.53**
phase 1	B	39.15	**39.78**	39.32	**39.78**
with phase	A	**100**	98.19	99.95	99.21
1&2	B	98.42	97.82	96.88	**98.74**

evaluation with 2-fold cross-validation (10-fold → 2-fold cross-validation). This experiment was able to evaluate more various zero-day malware data by increasing the number of existing test data (the average number of zero-day malware data: 506 (122∼1,122) → 14,850 (4,262∼33,710)). As shown in Table 6 and Figure 9, we obtained stable test performance even though not only the relatively few training data (reduced to half) but also increased analogous zero-day malware test data in the proposed framework (> 99%).

2) ZERO-DAY MALWARE EXPERIMENT II USING MALWARE DATA WITH SIMILAR FAMILY NAMES

We conducted a zero-day malware attack experiment II with different class malware data sharing the family name with similar properties from Malimg dataset [10]. We discovered data from the Malimg dataset that are very suitable for use in zero-day malware experiments (Table 1 and Figure 11). We trained and tested four classes using two different family name data with similar properties (Four types (5,543); Worm: Allaple.A (2,949) and Allaple.L (1,591), PWS: C2Lop.gen!G (200) and C2Lop.P (146), Trojan: Lolyda.AA1 (213) and Lolyda.AA2(184), TDownloader: Swizzor.gen!I (132) and Swizzor.gen!E (128)). For richer interpretation and analysis, we designed the zero-day experiment into two sessions and conducted training and testing. For session A, the training dataset (3,494) consists of Allaple.A, C2Lop.gen!G, Lolyda.AA1, Swizzor.gen!I, and the test dataset (2,049) consists of Allaple.L, C2Lop.P, Lolyda.AA2, Swizzor.gen!E. Inversely, session B consists of a training dataset (2,049) and a test dataset (3,494). Session B has a challenging problem of learning with a small amount of training data. This is a big issue not only in the field of machine learning but also in developing malware detection, especially zero-day malware detection technology. Even if it is derived from the same malware family, it is zero-day malware that is not previously learned, and it can cause a big performance degradation problem in the initial period as there is a very limited data to learn. To verify that the proposed framework can handle zero-day malware problems and a few data issues, we designed a second zero-day experiment using a similar malware family from the Malimg dataset. The experiment consists of the training sessions that were not only composed of session A and B, but also we evaluated the proposed framework with only phase 1 and with phases 1&2. The proposed framework deal with analogous new data by composing phase 1 to train the generator and discriminator and phase 2

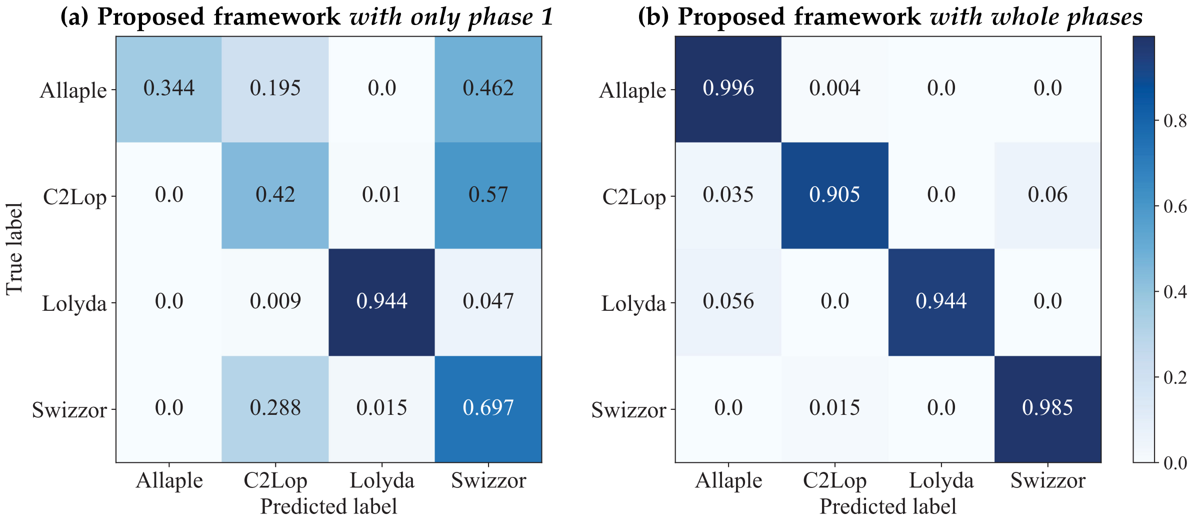

FIGURE 10. Confusion matrix for zero-day malware classification results (session B) in the proposed framework (E-GAN) with (a) only phase 1 and (b) whole phases (1&2) used similar malware family from the Malimg dataset.

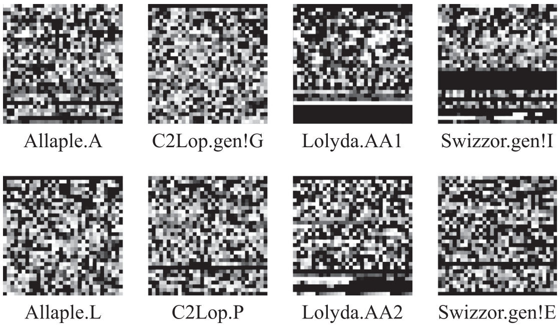

FIGURE 11. Examples of eight malware images from Malimg dataset for second zero-day experiment [10].

to train the discriminator on the analogous zero-day malware data. In Table 7 and Figure 10, we showed that the models trained up to phase 2 performed better than only phase 1 learned in all sessions (A and B). In particular, very interesting results were obtained in session B, where training was performed with a small amount of training data. In session B, the result of learning only phase 1 of the proposed framework was disastrous in all tested GAN models. This experiment demonstrates that existing GAN studies (i.e., phase 1 in the proposed framework) may not respond properly to new data. On the other hand, the final model trained up to phase 2 of the proposed framework showed very stable and high averaged accuracy ($>$ 98.65%) (Table 7 and Figure 10). Consequently, the proposed framework can learn very effectively when there is little data, showing excellent performance in the zero-day malware detection problem.

In practice, it is known that zero-day malware is often derived from variations of existing malware [8], [13]. To explore the limits in the performance of proposed frameworks, we performed on the restricted dataset for evaluation even using two different datasets [9], [10]. The first zero-day experiment designed assumes a plausible zero-day malware attack by transforming existing malware instead of the actual zero-day malware attack data. Additionally, we designed other zero-day experiments using a similar malware family from different malware types. Although we have obtained outstanding results in various zero-day experiments, we might have obtained more meaningful interpretation and discussion if we measured and utilized a richer malware database.

While, the GAN based image-processing approach method has a one-way limitation about malware code to the image in the malware detection field [8], [13], [29]. However, conversion to the malware code is not required to achieve the goals and objectives of this study. In this paper, the proposed framework is to detect a myriad of similar malware that can be made with slight changes. Even if the proposed framework cannot reproduce the malware code, it is a model that can detect and classify the analogous malware with high similarity to the learned sample malware data. In addition, if a new type of zero-day malware that is not used for learning appears, the proposed method also has the advantage of being able to quickly learn about the new type of malware and apply it. Therefore, in terms of practicality and convenience, it is a very helpful framework when developed zero-day malware detection software.

Meanwhile, as it is known from the adversarial attack, the performance of many machine learning based systems is greatly reduced and neutralized by small distortion (e.g., combining noise, etc.) [50], [51]. This is no different in this field, and some hackers will be taking this vulnerability. Therefore, it is necessary to build a robust and stability security system from these easy modifications. The proposed framework is intuitively generating and learning a plausible new malware from existing malware, and it can be a complementary measure to deal with these challenge problems.

V. CONCLUSION

In the present study, the proposed framework based on plausible malware training and augmentation using a generative adversarial network was to solve the problems caused by malware and analogous zero-day malware. In particular, because zero-day malware is often created by the deformation of existing malware, the proposed framework with representative GAN models augmented even for the high-quality and high-diversity evolved malware images. For detection and classification, the discriminator was trained using malware images generated by the generator and robust to zero-day malware. Moreover, the proposed framework achieved high and stable averaged accuracy in the analogous zero-day malware attack experiment. We believe that the proposed framework based plausible zero-day malware detection approach has important advantages for antivirus systems in the computer security because it does not require inefficient malware signatures analysis. In this study, the malware code has been converted to malware images with fixed sizes through crop and pad operations for efficient learning. In fact, the processes could reduce the signatures of malware. In future studies, we will expand the malware types with various malware datasets (including zero-day malware) and solve the problem of various malware lengths. Moreover, further research should be conducted to develop an optimized GAN model performing in our proposed framework for extensive zero-day malware detection. In future studies it will be interesting to use explainable AI techniques (e.g., [52]) to gain a further understanding of zero-day malware features, thus allowing the zero-day malware detection AI and its creators to learn better from their mistakes. Moreover, cases of extreme changes, such as new type of zero-day malware, deserve further investigation to extend the possible application spectrum.

ACKNOWLEDGMENTS

Dong-Ok Won and Yong-Nam Jang authors contributed equally to this work.

REFERENCES

[1] A. Mosenia and N. K. Jha, "A comprehensive study of security of Internet-of-Things," *IEEE Trans. Emerg. Topics Comput.*, vol. 5, no. 4, pp. 586–602, Oct.–Dec. 2016.

[2] A. Li, S. Xue, X.-Y. Li, L. Zhang, and J. Qian, "AppDNA: Profiling app behavior via deep-learning on function call graphs," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 1, pp. 414–427, Jan.–Mar. 2022.

[3] S. Homayoun, A. Dehghantanha, M. Ahmadzadeh, S. Hashemi, and R. Khayami, "Know abnormal, find evil: Frequent pattern mining for ransomware threat hunting and intelligence," *IEEE Trans. Emerg. Topics Comput.*, vol. 8, no. 2, pp. 341–351, Apr.–Jun. 2020.

[4] T. Saha, N. Aaraj, N. Ajjarapu, and N. K. Jha, "Sharks: Smart hacking approaches for risk scanning in Internet-of-Things and cyber-physical systems based on machine learning," *IEEE Trans. Emerg. Topics Comput.*, to be published, doi: 10.1109/TETC.2021.3050733.

[5] W. Zhang, Y. Wen, and X. Zhang, "Towards virus scanning as a service in mobile cloud computing: Energy-efficient dispatching policy under *N*-Version protection," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 1, pp. 122–134, Jan.–Mar. 2018.

[6] S. D. SL and C. Jaidhar, "Windows malware detector using convolutional neural network based on visualization images," *IEEE Trans. Emerg. Topics Comput.*, vol. 9, no. 2, pp. 1057–1069, Apr.–Jun. 2021.

[7] Kaspersky Security Bulletin 2018, *Statistics*, 2018. [Online]. Available: https://securelist.com/kaspersky-security-bulletin-2018-statistics/89145

[8] M. Grace, Y. Zhou, Q. Zhang, S. Zou, and X. Jiang, "Riskranker: Scalable and accurate zero-day android malware detection," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, 2012, pp. 281–294.

[9] R. Ronen, M. Radu, C. Feuerstein, E. Yom-Tov, and M. Ahmadi, "Microsoft malware classification challenge," 2018, *arXiv:1802.10135*.

[10] L. Nataraj, S. Karthikeyan, G. Jacob, and B. Manjunath, "Malware images: Visualization and automatic classification," in *Proc. 8th Int. Symp. Visual. Cyber Secur.*, 2011, Art. no. 4.

[11] J. Yan, Y. Qi, and Q. Rao, "Detecting malware with an ensemble method based on deep neural network," *Security and Communication Networks*, vol. 2018, 2018, Art. no. 7247095.

[12] Z. Cui, F. Xue, X. Cai, Y. Cao, G.-G. Wang, and J. Chen, "Detection of malicious code variants based on deep learning," *IEEE Trans. Ind. Informat.*, vol. 14, no. 7, pp. 3187–3196, Jul. 2018.

[13] 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.*, vol. 460, pp. 83–102, 2018.

[14] S. Venkatraman and M. Alazab, "Use of Data Visualisation for Zero-Day Malware Detection," *Secur. Commun. Netw.*, vol. 2018, 2018, Art. no. 1728303.

[15] F. Xiao, Z. Lin, Y. Sun, and Y. Ma, "Malware detection based on deep learning of behavior graphs," *Math. Problems Eng.*, vol. 2019, 2019, Art. no. 8195395.

[16] J. Zhu, J. Jang-Jaccard, and P. A. Watters, "Multi-loss siamese neural network with batch normalization layer for malware detection," *IEEE Access*, vol. 8, pp. 171542–171550, 2020.

[17] S. Sharmeen, S. Huda, J. Abawajy, and M. M. Hassan, "An adaptive framework against android privilege escalation threats using deep learning and semi-supervised approaches," *Appl. Soft Comput.*, vol. 89, 2020, Art. no. 106089.

[18] K. Berlin, D. Slater, and J. Saxe, "Malicious behavior detection using windows audit logs," in *Proc. 8th ACM Workshop Artif. Intell. Secur.*, 2015, pp. 35–44.

[19] B. Ndibanje, K. H. Kim, Y. J. Kang, H. H. Kim, T. Y. Kim, and H. J. Lee, "Cross-method-Based analysis and classification of malicious behavior by API calls extraction," *Appl. Sci.*, vol. 9, no. 2, 2019, Art. no. 239.

[20] C. Annachhatre, T. H. Austin, and M. Stamp, "Hidden markov models for malware classification," *J. Comput. Virol. Hacking Techn.*, vol. 11, no. 2, pp. 59–73, 2015.

[21] S.-W. Lee and A. Verri, *Proc. Pattern Recognit. Support Vector Mach.: Proc. 1st Int. Workshop*, 2003.

[22] P. Wang and Y.-S. Wang, "Malware behavioural detection and vaccine development by using a support vector model classifier," *J. Comput. Syst. Sci.*, vol. 81, no. 6, pp. 1012–1026, 2015.

[23] F. C. C. Garcia, I. Muga, and P. Felix, "Random forest for malware classification," 2016, *arXiv:1609.07770*.

[24] K. Singh, S. C. Guntuku, A. Thakur, and C. Hota, "Big Data analytics framework for peer-to-peer botnet detection using random forests," *Inf. Sci.*, vol. 278, pp. 488–497, 2014.

[25] Z. Chen *et al.*, "Machine learning based mobile malware detection using highly imbalanced network traffic," *Inf. Sci.*, vol. 433, pp. 346–364, 2018.

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

[27] Y. Ye, L. Chen, S. Hou, W. Hardy, and X. Li, "DeepAM: A heterogeneous deep learning framework for intelligent malware detection," *Knowl. Inf. Syst.*, vol. 54, no. 2, pp. 265–285, 2018.

[28] E. K. Kabanga and C. H. Kim, "Malware images classification using convolutional neural network," *J. Comput. Commun.*, vol. 6, no. 1, 2017, Art. no. 153.

[29] V. S. Bhaskara and D. Bhattacharyya, "Emulating malware authors for proactive protection using GANs over a distributed image visualization of dynamic file behavior," 2018, *arXiv:1807.07525*.

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

[31] M. Kawai, K. Ota, and M. Dong, "Improved MalGAN: Avoiding malware detector by leaning cleanware features," in *Proc. Int. Conf. Artif. Intell. Inf. Commun.*, 2019, pp. 040–045.

[32] S.-W. Lee and H.-H. Song, "A new recurrent neural-network architecture for visual pattern recognition," *IEEE Trans. Neural Netw.*, vol. 8, no. 2, pp. 331–340, Mar. 1997.

[33] Q. Wang *et al.*, "Adversary resistant deep neural networks with an application to malware detection," in *Proc. 23rd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2017, pp. 1145–1153.

[34] Z.-U. Rehman *et al.*, "Machine learning-assisted signature and heuristic-based detection of malwares in android devices," *Comput. Elect. Eng.*, vol. 69, pp. 828–841, 2018.

[35] D. Gupta and R. Rani, "Big Data framework for zero-day malware detection," *Cybern. Syst.*, vol. 49, no. 2, pp. 103–121, 2018.

[36] S. Huda *et al.*, "Defending unknown attacks on cyber-physical systems by semi-supervised approach and available unlabeled data," *Inf. Sci.*, vol. 379, pp. 211–228, 2017.

[37] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *J. Big Data*, vol. 6, no. 1, 2019, Art. no. 60.

[38] H.-G. Jung and S.-W. Lee, "Few-shot learning with geometric constraints," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 11, pp. 4660–4672, Nov. 2020.

[39] I. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[40] C. Wang, C. Xu, X. Yao, and D. Tao, "Evolutionary generative adversarial networks," *IEEE Trans. Evol. Comput.*, vol. 23, no. 6, pp. 921–934, Dec. 2019.

[41] R. Burks, K. A. Islam, Y. Lu, and J. Li, "Data augmentation with generative models for improved malware detection: A comparative study," in *Proc. IEEE 10th Annu. Ubiquitous Comput.*, Electron. Mobile Commun. Conf., 2019, pp. 0660–0665.

[42] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," 2015, *arXiv:1511.06434*.

[43] X. Mao, Q. Li, H. Xie, R. Y. Lau, Z. Wang, and S. Paul Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2017, pp. 2813–2821.

[44] M. Arjovsky, S. Chintala, and L. Bottou, "GAN Wasserstein," 2017, *arXiv:1701.07875*.

[45] A. Odena, C. Olah, and J. Shlens, "Conditional image synthesis with auxiliary classifier GANs," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 2642–2651.

[46] J. L. McClelland et al., "Parallel distributed processing," *Explorations Microstructure Cogn.*, vol. 2, pp. 216–271, 1986.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.

[48] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "GANs trained by a two time-scale update rule converge to a local nash equilibrium," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 6626–6637.

[49] Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli, "Image quality assessment: From error visibility to structural similarity," *IEEE Trans. Image Process.*, vol. 13, no. 4, pp. 600–612, Apr. 2004.

[50] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," 2017, *arXiv:1706.06083*.

[51] N. Carlini and D. Wagner, "Adversarial examples are not easily detected: Bypassing ten detection methods," in *Proc. 10th ACM Workshop Artif. Intell. Secur.*, 2017, pp. 3–14.

[52] S. Lapuschkin, S. Wäldchen, A. Binder, G. Montavon, W. Samek, and K.-R. Müller, "Unmasking clever hans predictors and assessing what machines really learn," *Nature Commun.*, vol. 10, no. 1, 2019, Art. no. 1096.

Dong-Ok Won (Member, IEEE) received the BS degree in computer engineering from the Tech University of Korea, Republic of Korea, in 2012, and the PhD degree from the Department of Brain and Cognitive Engineering, Korea University, Republic of Korea, in 2019. He is currently working as an assistant professor with the Department of Artificial Intelligence, Hallym University, Republic of Korea. His research interests include pattern recognition, machine learning, artificial intelligence, and computer security.

Yong-Nam Jang received MS degree from the Department of Brain and Cognitive Engineering, Korea University, Republic of Korea, in 2020. His research interests include pattern recognition, machine learning, and computer security.

Seong-Whan Lee (Fellow, IEEE) received the BS degree in computer science and statistics from Seoul National University, Seoul, Republic of Korea, in 1984, and the MS and PhD degrees in computer science from the Korea Advanced Institute of Science and Technology, Republic of Korea, in 1986 and 1989, respectively. He is currently the head of the Department of Artificial Intelligence, Korea University, Republic of Korea. His current research interests include artificial intelligence, pattern recognition, and brain engineering. He is a fellow of the International Association of Pattern Recognition (IAPR) and the Korea Academy of Science and Technology.