Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Deep reinforcement learning based Evasion Generative Adversarial Network for botnet detection[☆]

Rizwan Hamid Randhawa[a,*], Nauman Aslam[a], Mohammad Alauthman[b], Muhammad Khalid[c], Husnain Rafiq[d]

[a] *Department of Computer and Information Sciences, Northumbria University, Newcastle upon Tyne, UK*
[b] *Department of Information Security, University of Petra, Amman 11196, Jordan*
[c] *School of Computer Science, University of Hull, UK*
[d] *Department of Computer Science, Edgehill University, UK*

A R T I C L E I N F O

Article history:
Received 8 January 2023
Received in revised form 29 August 2023
Accepted 3 September 2023
Available online 7 September 2023

Keywords:
Low data regimes
GANs
ACGAN
EVAGAN
Botnet

A B S T R A C T

Botnet detectors based on machine learning are potential targets for adversarial evasion attacks. Several research works employ adversarial training with samples generated from generative adversarial nets (GANs) to make the botnet detectors adept at recognising adversarial evasions. However, the synthetic evasions may not follow the original semantics of the input samples. This paper proposes a novel GAN model leveraged with deep reinforcement learning (DRL) to explore semantic aware samples and simultaneously harden its detection. A DRL agent is used to attack the discriminator of the GAN that acts as a botnet detector. The agent trains the discriminator on the crafted perturbations during the GAN training, which helps the GAN generator converge earlier than the case without DRL. We name this model RELEVAGAN, i.e. ["relieve a GAN" or deep REinforcement Learning-based Evasion Generative Adversarial Network] because, with the help of DRL, it minimises the GAN's job by letting its generator explore the evasion samples within the semantic limits. During the GAN training, the attacks are conducted to adjust the discriminator weights for learning crafted perturbations by the agent. RELEVAGAN does not require adversarial training for the ML classifiers since it can act as an adversarial semantic-aware botnet detection model. The code will be available at https://github.com/rhr407/RELEVAGAN.

1. Introduction

Artificial Intelligence (AI) in cybersecurity has become the new normal. However, AI will take some time to win public trust due to inherent biases and adversarial threats ranging from insiders to intrusion and malware [1,2]. AI-based models can be biased toward the majority class of data on which they are trained due to the imbalance in datasets. Anomaly samples in publicly available datasets are usually scarce compared to the normal data in low-data regimes like cybersecurity. Researchers have used data generation techniques like emulation, synthetic oversampling, and generative models to mitigate data biasing.

Similarly, adversarial learning has been a topic of pivotal interest to research communities for the last decade. Many seminal works have been created to deal with adversarial attacks like poisoning, evasion, and transferability [2–4]. Adversarial defence strategies can be based on preprocessing, adversarial training, architecture, detection, defensive testing, multiclassifiers, and game theory [2].

Adversarial training seems to be a simplistic strategy to provide robustness against adversarial evasion attacks; however, it has some cons. First, increasing the number of samples in the auxiliary data may not have a linear relationship with detection accuracy [4]. Second, the adversarial training cannot guarantee a robust defence as it can be bypassed [5]. Although adopted as an immediate remedy against some adversarial attacks, it cannot be considered an ultimate cure for the grave problem in AI. Third, it consumes additional time for retraining. Several GAN-based research works preserve the functionality of the generated samples by manipulating only non-functional features [6–8]. So, GANs do not play a role in generating a complete feature vector in those works. It is also quite challenging to generate categorical features using a GAN without manual engineering except using a sequence GAN [9]. Researchers have also used deep reinforcement learning (DRL) to generate the functionality preserving

☆ This work was supported under the Research and Development Fund (RDF) of Northumbria University, Newcastle upon Tyne, UK.

∗ Corresponding author.
E-mail addresses: rhr407@gmail.com (R.H. Randhawa), nauman.aslam@northumbria.ac.uk (N. Aslam), mohammad.alauthman@uop.edu.jo (M. Alauthman), m.khalid@hull.ac.uk (M. Khalid), husnain.rafiq@edgehill.ac.uk (H. Rafiq).

https://doi.org/10.1016/j.future.2023.09.011
0167-739X/

adversarial evasion attacks [10–12]. The main goal of employing DRL is to explore functionality preserving adversarial samples since the DRL can guarantee semantic awareness in contrast to GANs; however, these works consider the adversarial training to make the detection models adept at evasion awareness.

RELEVAGAN is an effort toward a unifying model concept that would solve the problem of data imbalance, provide adversarial semantic awareness and save training time. RELEVAGAN is equipped with an integrated DRL agent to achieve the said goals. RELEVAGAN name was chosen for two reasons. First, it is a deep REinforcement Learning-based Evasion Generative Adversarial Network. Second, it relieves the employed GAN model to make its job easier by letting the generator of the GAN explore the semantic aware samples, i.e., within certain boundary conditions. We can call this RELiEVe A GAN as well. Either way, RELEVAGAN proves to be an improved technique compared to the peer models like Auxiliary Classifier GAN (ACGAN) and Evasion Generative Adversarial Network (EVAGAN).

The DRL agent attacks the RELEVAGAN's discriminator, which acts as a botnet detector. The attack generation is based on manipulating the real attack samples to evade the botnet detector. As the RELEVAGAN training proceeds, the agent learns to evade the botnet detector. The discriminator is adversarially trained on the evaded samples from the DRL attacker and synthetic samples from the generator in each training iteration. After a certain number of epochs, the discriminator becomes hardened against the samples from the DRL agent and the generator. The detection estimations for the benign, real, and generated samples and the generator training settle to the desired values in fewer training iterations than the EVAGAN model. The experimental analysis shows the considerable performance of the RELEVAGAN model against EVAGAN in terms of detection estimation and stability of training for three different botnet datasets. We argue that the learning of GAN follows semantic awareness because GAN is also trained on the attacks generated by the DRL model. The following are the main contributions of this paper:

(1) We propose a novel DRL-based GAN model to address the problems of data imbalance, evasion awareness, and functionality preservation in synthetic botnet traffic generation.
(2) We demonstrate by experiments that DRL plays a role in GAN in early convergence of training for detecting synthetic and real attacks.
(3) Because of the integrated self-learning DRL attacker, the proposed model can be envisaged as sustainable against evolving botnet.
(4) We determine that RELEVAGAN outperforms EVAGAN regarding early training convergence.

2. Background

2.1. Data imbalance

The inequitable distribution of botnet datasets makes their ML-based detection models less accurate. To address this issue, data undersampling can be adopted. However, it can result in the loss of diversity and representation of normal traffic [13]. Oversampling can also solve the data imbalance problem to some extent; however, the use of nearest neighbours, and linear interpolation, may not be suitable for the high-dimensional and complex probability distributions [14,15]. Researchers have tested several oversampling techniques in [16] to rank the best performing being SMOTE_IPF, ProWSyn and polynom_fit_SMOTE. However, authors in [4] declared GANs outperforming those three oversamplers in most ML classifiers' adversarial training. Hence, we consider GANs as a suitable candidate for data oversampling compared to other synthetic data generation methods. Table 1 shows the main notations used in this text.

Table 1
Main notations.

Notation	Definition
$\mathcal{G}$	Generator
$\mathcal{D}$	Discriminator
z	Normal distribution from noise space
z	Noise samples
p_{data}	Probability distribution of real samples
p_z	Probability distribution of noise samples
$\mathcal{X}$	Real data distribution
$\mathbb{E}$	Expected value
c_m	Minority class labels
c_M	Majority class labels
y_{x_i}	Actual label of sample x_i in dataset $\mathcal{X}$

2.2. Generative adversarial nets

A GAN combines two neural networks with different structures: a generator and a discriminator. The generator ($\mathcal{G}$) synthesises samples, and the discriminator ($\mathcal{D}$) evaluates those samples. In a classical GAN, $\mathcal{D}$: $\mathcal{X} \rightarrow [0,1]$ acts as a classifier to give us an estimate of probability (between 0 and 1) to mark whether the input data is real or fake. The objective function of the combined model is denoted by Eq. (1).

$$\min_{\mathcal{G}} \max_{\mathcal{D}} V(\mathcal{D},\mathcal{G}) = \mathbb{E}_{x\sim p_{data}(x)}[\log \mathcal{D}(x)] + \mathbb{E}_{z\sim p_z(z)}[\log(1-\mathcal{D}(\mathcal{G}(z)))] \tag{1}$$

In Eq. (1), $\mathbb{E}$ denotes the expected value of the loss, and x and z represent the real and noise data samples consecutively. Similarly, p_{data} and p_z are the real data and noise probability distributions, respectively. The min–max game minimises the $\mathcal{G}$'s loss in generating data similar to the real data. Since $\mathcal{G}$ is not able to control $\mathcal{D}$'s loss on real data, it can maximise the loss of $\mathcal{D}$ on generated data $\mathcal{G}(z)$. The objective function of $\mathcal{G}$ is denoted by Eq. (2).

$$J^{\mathcal{G}}(\mathcal{G}) = \mathbb{E}_{z\sim p_z(z)}[\log(\mathcal{D}(\mathcal{G}(z)))] \tag{2}$$

One complete iteration of the GAN training takes noise as input and the output of the $\mathcal{D}$ as the feedback to update the weights of the $\mathcal{G}$. This training process keeps iterating for a specific number, after which $\mathcal{G}$ and $\mathcal{D}$ do not learn further, resulting in Nash equilibrium.

2.3. Evasion awareness

Adversarial training is widely adopted to proactively make the ML classifiers aware of the evasion samples. However, extending the capability of $\mathcal{D}$ of a GAN from discriminating between real and fake samples to differentiating between normal and anomaly data renders the adversarial training needless because $\mathcal{D}$ can act as an evasion-aware classifier [17,18]. In [19], authors propose EVAGAN that provides such type of $\mathcal{D}$ and compare its performance with the $\mathcal{D}$ of ACGAN and other ML classifiers like xgboost (XGB), naive bayes (NB), decision tree (DT), random forests (RF), k-nearest neighbours (KNN) and logistic regression (LR). EVAGAN's $\mathcal{D}$ outperforms the ML classifiers in black box testing and gives 100% accuracy in normal and evasion sample estimation. However, EVAGAN, like other GAN models, is agnostic of semantics and functionality preservation of malicious synthetic samples. In this paper, we propose a novel type of GAN based on EVAGAN leveraged with DRL to address the problem of functionality preservation.

2.4. ACGAN

The proposed RELEVAGAN is based on EVAGAN, an improved ACGAN version. ACGAN's $\mathcal{D}$, along with differentiating between real or fake samples, also considers the class labels in the training process [20]. That said, $\mathcal{D}$ of ACGAN works as a dual classifier for differentiating between the real/fake data and different classes of the input samples, whether coming from the real source or the $\mathcal{G}$. The objective function of ACGAN can be derived using likelihoods denoted by L_S and L_C for the correct source data and real class labels, respectively. $\mathcal{D}$ trains to maximise $L_C + L_S$ and $\mathcal{G}$ learns to maximise $L_C - L_S$. Hence, the objective of $\mathcal{D}$ is to improve the two likelihoods, while the aim of $\mathcal{G}$ is to assist $\mathcal{D}$ in improving the performance on class labels. $\mathcal{G}$'s other target is to suppress the log-likelihood of $\mathcal{D}$ on fake samples. The $\mathcal{D}$ outputs probability distributions over sources and the class labels $[P(S|\mathcal{X}), P(C|\mathcal{X})] = \mathcal{D}(\mathcal{X})$ where S denotes the sources (real/fake) and C represents the class labels. Eqs. (3) and (4) represent the L_s and L_c respectively.

$$L_S = \mathbb{E}[logP(S = real|\mathcal{X}_{real})] + \mathbb{E}[logP(S = fake|\mathcal{X}_{fake})] \tag{3}$$

$$L_C = \mathbb{E}[logP(C = c|\mathcal{X}_{real})] + \mathbb{E}[logP(C = c|\mathcal{X}_{fake})] \tag{4}$$

2.5. EVAGAN

EVAGAN is a specialised version of ACGAN with binary class consideration for low data regimes. The $\mathcal{G}$ of ACGAN considers generating multiple class samples $\mathcal{X}_{fake} = \mathcal{G}(c, z)$ where c is the class label. Hence the number of the samples can include $C = \{c_1, c_2, c_3, \ldots, c_n\}$, which may not be required in low data regimes as we need to generate only the anomaly samples which are low in number. The generator does not need to improve the classification performance of $\mathcal{D}$ on normal/majority class samples. This considerably reduces the training time of $\mathcal{G}$ as the diversity of the input samples from a single class is less intricate. In this way, we can improve the performance of the $\mathcal{G}$ and harden the $\mathcal{D}$ simultaneously with fewer c_m samples. EVAGAN also hardens itself on the evasion samples generated by its $\mathcal{G}$, which takes noise z and the single class labels c = 1. The minority class labels are embedded into the input layer of the $\mathcal{G}$. Since the $\mathcal{G}$ only needs to generate minority class samples, it should only consider the loss of $\mathcal{D}$ on estimating the minority class and the sources. The goal of $\mathcal{G}$ is to maximise the $\mathcal{D}$'s loss on the fake source and assist in reducing the $\mathcal{D}$'s loss on minority class samples. Let y_{x_i} denotes the actual label of sample x_i in dataset $\mathcal{X}$, $P(S = fake|\mathcal{X}_{m_{fake}})$ represent the prediction of the probability distribution of samples being fake and $P(C = c_m|\mathcal{X}_{m_{fake}})$ is the predicted output for probability distribution from $\mathcal{D}$ for minority class labels c_m, then the loss function of $\mathcal{G}$ for N samples will be given by Eq. (5).

$$G\_Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_{x_i}^{fake}(\log P(S = fake|\mathcal{X}_{m_{fake}})) + y_{x_i}^{c_m}(1 - \log P(C = c_m|\mathcal{X}_{m_{fake}}))] \tag{5}$$

In Eq. (5), $y_{x_i}^{fake}$ and $y_{x_i}^{c_m}$ represent the actual labels for fake and minority classes, respectively. The primary goal of $\mathcal{G}$ is to minimise the *G_Loss*, so it inclines toward reducing the correct estimation of $\mathcal{D}$ on fake samples by suppressing the term $\log P(S = fake|\mathcal{X}_{m_{fake}})$. For the second objective, it will try to increase the value of $\log P(C = c_m|\mathcal{X}_{m_{fake}})$ so that the second term in the equation can also be suppressed in value. The loss function of $\mathcal{D}$ is given by Eq. (6).

$$\begin{aligned} D\_Loss = -\frac{1}{N}\sum_{i=1}^{N}[y_{x_i}^{c_M}(\log P(S = c_M|\mathcal{X}_{M_{real}})) + \\ y_{x_i}^{real}(\log P(S = real|\mathcal{X}_{m_{real}})) + \\ (1 - y_{x_i}^{real})(1 - \log P(S = real|\mathcal{X}_{m_{real}})) + \\ y_{x_i}^{c_{m_{real}}}(\log P(C = c_m|\mathcal{X}_{m_{real}})) + \\ (1 - y_{x_i}^{c_{m_{real}}})(1 - \log P(C = c_m|\mathcal{X}_{m_{real}}))] \end{aligned} \tag{6}$$

In Eq. (6), the loss of $\mathcal{D}$ is calculated per the three different binary cross-entropy losses for majority class, sources, and minority class estimations.

2.6. Semantics/functionality preservation

Semantics or functionality preservation means that the synthetic samples can still execute the original malicious function intended by the attacker. Various researchers have employed GANs to synthesise the network traffic data to address the issue of data imbalance and adversarial evasion attacks [4,21,22]. However, GANs are not good at generating categorical features [9]. Researchers have used one-hot representation [6], or IP2Vec techniques [23,24] to transform IP addresses into integer values to input to GANs for data generation. Since GANs follow the probability distribution of the input data so there is a high chance that the synthetic samples may lie outside the semantic limits. The perturbations could be easily perceived as anomalies by the ML detectors. To preserve the functionality, the perturbation should be small enough not to be perceived by the detectors as malicious and distant enough to be considered normal samples. At the same time, the malicious activity must not be compromised. For this reason, several researchers have proposed functionality preservation by modifying only the non-functionality preserving features using GANs [6,8,9]. However, various research works claim to preserve the malicious functionality of the generated samples using DRL.

2.7. DRL-based evasion generation

A reinforcement learning model comprises an agent and an environment that engage for a defined number of iterations/steps. For a turn t the agent chooses an action $a_t \in \mathcal{A}$ using some policy $\pi(a|s_t)$ and an observable state s_t. The environment returns a reward $r_t \in \mathbb{R}$ against an action and a new state s_{t+1}. The reward r_t and observed state of the environment s_{t+1} are fed back to the agent for defining a new action based on policy $\pi(a|s_{t+1})$. The agent tries to learn through the trade-off between exploration and exploitation. The maximum reward collection is the main goal for exploration, which can be possible if the agent employs a certain policy to boost the expected value given by the Q-value function in Eq. (7).

$$V^{\pi}(s_t) = \mathbb{E}_{a_t}[Q^{\pi}(s_t, a_t)|s_t] \tag{7}$$

Here Q is given by Eq. (8).

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s_{t+1:\infty}, a_{t+1:\infty}}[R_t|s_t, a_t] \tag{8}$$

and R is denoted by Eq. (9)

$$R_t = \Sigma_{i\geq 0}\gamma^i r_{t+i} \tag{9}$$

In Eq. (9), $\gamma \in [0, 1]$ is a discounting factor for rewards from future actions. For catering to the large intractable storage of tabular representation of the Q-function, deep neural networks

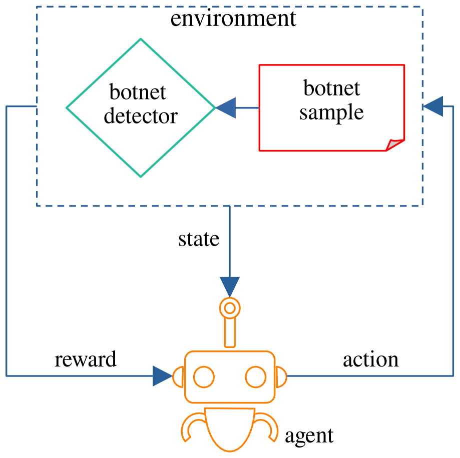


Fig. 1. Markov Decision Process for Botnet evasion generation using Reinforcement Learning.

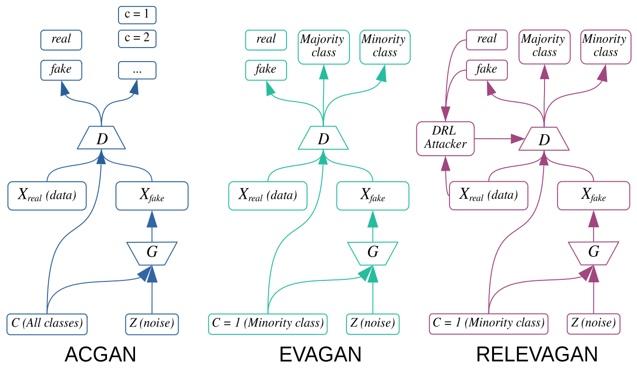


Fig. 2. Comparison of RELEVAGAN model with ACGAN and EVAGAN.

come to the rescue to act as an approximator in the deep Q-Network (DQN) to represent the state–action value function [25]. In RELEVAGAN, we use double DQN as the core technique for DRL-based attack generation.

Fig. 1 shows the Markov Decision Process (MDP) [26] where an agent is interacting with a botnet detector environment by providing a new sample that can act as a potential evasion. The Q-function and the policy help in taking a particular action. The botnet sample is a manipulated feature vector defined within certain boundaries to ensure semantics preservation. The reward is the classification output of the botnet detector. After trying the botnet detector, the reward and the new state are fed back to the agent.

2.8. DRL-based functionality preservation

To preserve the functionality, researchers generate the minority class samples within semantic limits [10–12,27–29] by attacking the trained classifiers using a DRL agent. The evaded samples were collected for adversarial training to make the model adept at adversarial awareness. Similarly, in [27], authors used DRL to attack PDF (Portable Document Format) malware detectors to generate evasive malware samples while preserving the attacks' functionality. The rationale for using the DRL is to restrict it to explore the evasion attacks within a specific range. This paper uses the DRL and the EVAGAN model to generate the evasion attack samples using a DRL agent continuously. The process becomes part of the EVAGAN training. The marriage of GAN and DRL culminates in RELEVAGAN, which uses the power of EVAGAN as a robust evasion-aware detection model and DRL as the semantics check on the samples generated by the $\mathcal{G}$ of EVAGAN. The details of the model will be further discussed in Section 3.

3. RELEVAGAN

In this section, we discuss the motivation behind the design of RELEVAGAN, especially the structural explanation of its DRL attacker. As illustrated by Fig. 2, the DRL attacker has been introduced in RELEVAGAN. The rest of the architecture is similar to EVAGAN. Using the DRL attacker helps improve the EVAGAN performance, the details of which will be discussed in Section 3.

3.1. Motivation

GAN-based samples generation follows the probability distribution of the input data samples as $\mathcal{G}$ trains itself based on the feedback from $\mathcal{D}$. $\mathcal{G}$ tries to explore the new sample spaces that are unknown by $\mathcal{D}$ so that it can fool $\mathcal{D}$; however, this process can lead to the creation of samples that may not follow the real malicious semantics. To address this issue, DRL samples can help $\mathcal{G}$ learn the boundaries of the real samples. For this reason, a DRL agent can be leveraged to explore samples in a defined observation space. $\mathcal{D}$ can be trained on these generated samples, giving the feedback to $\mathcal{G}$ in the GAN training. Eventually, $\mathcal{G}$ can start learning from the updated feedback from $\mathcal{D}$ and generates the samples within a defined range set by the DRL agent. This process can help converge $\mathcal{G}$ training earlier while at the same time achieving high accuracy in a lesser number of epochs. Although semantic awareness comes with additional training costs for the DRL part, motivated by this rationale, the RELEVAGAN is a step toward a more intelligent functionality-preserving GAN design.

3.2. Architecture

Fig. 3 shows the architecture of RELEVAGAN. The structure of RELEVAGAN has mainly two components. One is EVAGAN, summarised in Section 2.5, and the other is a DRL-based model. For more details on EVAGAN, readers are encouraged to refer to the paper [19]. A typical DRL model consisting of an agent and an environment has been coupled with EVAGAN architecture in RELEVAGAN design. Like other DRL-based attackers for evasion generation using a black box attack [10–12,27,28], our proposed DRL agent attacks the $\mathcal{D}$ of EVAGAN, which acts as a black box classifier. $\mathcal{D}$'s output for minority class estimation is the agent's reward for adjusting its weights and generating a new action a_{t+n} based on some policy π. The new action is fed to the environment where the state generator creates a new state taking another seed sample from the real data set. As the result of a single iteration, the new state and the collected reward are fed back to the agent. As a result of the positive reward, the evasion samples are fed to the $\mathcal{D}$ of EVAGAN to train it adversarially. In this way, the $\mathcal{D}$ becomes proactively aware of any possible future evasions and ready to give better feedback to the $\mathcal{G}$ to train to confine its boundaries for evasion generation. The process leads to the early convergence of $\mathcal{G}$ training.

3.3. Environment

The environment in RELEVAGAN consists of mainly two parts:

3.3.1. State generator

The state generator is responsible for three different jobs:

- It takes a botnet seed sample as the current state S_t from the real botnet dataset and transforms it into a feature vector accepted by the $\mathcal{D}$ of EVAGAN based on some action index n coming from the agent.

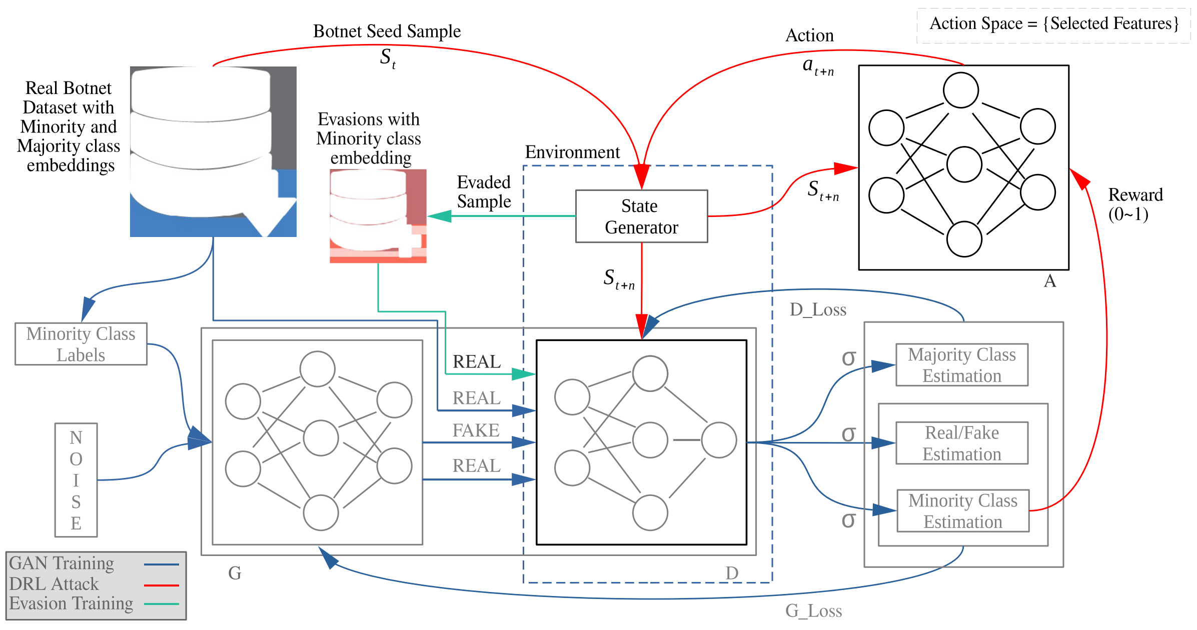


Fig. 3. RELEVAGAN Architecture.

- It feeds back the new state S_{t+n} to the agent.
- If the sample is evaded by the $\mathcal{D}$, the state generator is also responsible for storing and/or feeding it to the $\mathcal{D}$ to train in parallel with EVAGAN training adversarially.

3.3.2. Botnet detector

The target model is the $\mathcal{D}$ of EVAGAN. In Fig. 3, the EVAGAN model has been illustrated in grey-coloured border lines, and the difference has been highlighted in black lines for a better understanding of where the RELEVAGAN is different from EVAGAN.

3.4. Action space

The following feature set gives the action space through which the agent chooses the most appropriate index n to gain the maximum reward by evading the target botnet model. These features have been chosen based on the work in paper [10] for the three datasets used in this work. The details of the datasets are mentioned in Section 4.

- FlowDuration
- FlowBytes/s
- FlowPackets/s
- FwdPackets/s
- BwdPackets/s
- TotalLengthofFwdPacket
- TotalLengthofBwdPacket
- BwdPackets/s
- SubflowFwdBytes
- FwdHeaderLength
- BwdHeaderLength
- Down/UpRatio
- AveragePacketSize

To keep the functionality reservation, we limit the change in the feature value to Δ, which is the minimum value of the particular feature within the data set as given by Eq. (10).

$$\Delta_n = \min_{\forall m \in F_n} X(m) \qquad (10)$$

In Eq. (10), Δ_{F_n} is the minimum value for all the rows m of a particular feature F, and n is the action index coming from the agent as a particular feature number from the action table.

3.5. Agent

The agent is a deep neural network with the size of the observation space as input and the number of actions as the output. The observation space in RELEVAGAN is a complete botnet sample as a feature vector. The agent is responsible for choosing an action index n among the features in the action table as mentioned in Section 3.4. Based on this index, a training step is executed, which feeds back the reward and the new state to the agent.

3.6. Reward

The reward in a typical botnet evasion generation model using a DRL black-box attack is the output of the botnet detector [10, 12], which can be a real number in the range of [0, 1]. In our case, the expected value for the botnet sample is '0'; for a normal traffic sample, the output should ideally be '1'. Hence for a botnet sample to be considered a successful evasion by a DRL attacker, we set the threshold for the reward to be greater than 0.5. In other terms, if the sample generated by the DRL attacker is evaded with more than 50% confidence, the reward will be '1' and '0' otherwise.

3.7. Training

RELEVAGAN training is similar to EVAGAN except for adding a few extra steps for the DRL agent after every training batch. In Algorithm 1, steps 3 and 4 discriminate the training between EVAGAN and RELEVAGAN for a defined number of batches. The sequence of the steps is crucial for understanding the rationale behind RELEVAGAN. Note that we train $\mathcal{G}$ after the evasion training of $\mathcal{D}$. Since $\mathcal{D}$'s weights are adjusted as per the evasions generated by the DRL attacker in Step 4, it will feed the *G_Loss* back to$\mathcal{G}$ more cognitively as compared to the case of EVAGAN training. The DRL attack is executed in every batch of training.

Algorithm 1: RELEVAGAN Training

```
for i= 1, 2, 3, ..., number of batches do
    Step 1: Train $\mathcal{D}$ on real data
    Step 2: Train $\mathcal{D}$ on generated data
    Step 3: Execute DRL $\mathcal{A}$for generating evasion on batch size
    Step 4: Train $\mathcal{D}$ on $\mathcal{A}$ generated evasions
    Step 5: Train $\mathcal{G}$
```

Table 2
Distribution of normal and botnet samples in cybersecurity botnet datasets.

Dataset	Normal	Real_bots	Total samples
ISCX-2014	246929	Virut: 1748	248677
CIC-IDS2017	70374	Ares: 1956	72330
CIC-IDS2018	390961	Ares/Zeus: 2560	393521

4. Implementation details

4.1. Experimental setup

The experiments for the RELEVAGAN were performed on a GPU workstation, AMD Ryzen threadripper 1950x, equipped with a 16-core processor and an 8 GB memory GeForce GTC 1070 Ti graphics card. The OS used was Ubuntu 20.04, running Keras, TensorFlow, Sklearn and Numpy libraries within the Jupyter notebook. The source code of RELEVAGAN is also available on GitHub under MIT license.[1]

4.2. Data preparation

For experimentation, we have used three different botnet datasets, ISCX-2014, CIC-2017 and CIC-2018, from the Canadian Institute of Cybersecurity (CIC) for the quantitative analysis of RELEVAGAN. The choice of the dataset is based on the work done by authors in [4]. An open-source tool, CICFlowMeter-v4 from CIC, was used for feature extraction. The choice of feature set for training and data preprocessing is the same as EVAGAN's. Table 2 shows the distribution of benign and botnet samples in all three datasets. The details of a particular botnet selection are mentioned in EVAGAN paper [19].

4.3. DRL attacker

For the implementation of the DRL attacker, we used the OpenAI Gym and gym-malware tool kits [11,30]. Keras-rl and Keras-rl2 libraries were used for the selection of the DQN agent. Unlike a typical DRL algorithm, we execute a new training session in every batch of the RELEVAGAN training where the weights of the neural network are not reset to ensure that the agent is learning in each batch iteration of RELEVAGAN. The reason for keeping the weights is that we cannot estimate the number of training iterations that would e traverse the whole batch of botnet samples for generating manipulations. Hence, we reinitialise the session after each batch keeping the agent's neural network unchanged. A single training session in each batch iteration lasts until the following two cases appear:

- The evasion is successful.
- The number of tries saturates.

[1] https://www.github.com/rhr407/RELEVAGAN

Table 3
Hyperparameters of DRL Attacker.

Parameter	Value
Agent Type	DQN
Action Space	13
Policy	BoltzmannQPolicy
Double DQN	True
Target Model Update	1e−3
Number of turns	13
Number of rounds	256

Table 4
Hyperparameters of DRL Neural Network.

Parameter	Value
Network Type	FFNN
Number of Layers	4
Activations	ReLU, linear
Neurons in the input layer	size of observation space
Neurons in layer 1	64
Neurons in layer 2	128
Neurons in the output layer	number of actions
Layer Regularisation	*BatchNorm*

In either of the cases mentioned above, a new botnet seed sample is selected until the total number of samples in a batch is traversed. Each training step takes an action index that selects the corresponding feature from the botnet seed sample for manipulation. The modified sample is tried on the trained $\mathcal{D}$ using the Keras *model.predict* function, which gives the estimation of the botnet sample being from a minority class. This output estimation is used to set the agent's reward as mentioned in Section 3.6. As a result of each step, the reward and the new state (alternatively, the manipulated botnet sample) are returned to the agent. The details of the hyperparameters of the DRL attacker part have been mentioned in Table 3 and Table 4, respectively.

5. Results & discussion

The performance analysis of RELEVAGAN is identical to that of EVAGAN for botnet datasets. The metrics used were generated samples validity (GEN_VALIDITY), fake/generated botnet samples evasion (FAKE_BOT_EVA), real normal/majority class estimation (REAL_NORMAL_EST), and real botnet/minority class evasion (REAL_BOT_EVA). Fig. 4 shows the results for the ACGAN, EVAGAN, and RELEVAGAN estimations for comparison. The mathematical expressions for the evaluation metrics have been represented using Eqs. (11)–(14). These estimations were computed using the Keras *model.predict* function. The details of these metrics can be found in EVAGAN paper [19]. Fig. 5 illustrates the losses of $\mathcal{D}$ for real and fake minority classes and majority/normal classes and of $\mathcal{G}$ for ACGAN, EVAGAN, and RELEVAGAN.

$$GEN\_VALIDITY = \frac{\sum[\hat{\mathcal{G}}(z, c_m)[0]]}{N} \tag{11}$$

$$FAKE\_BOT\_EVA = \frac{\sum[\hat{\mathcal{G}}(z, c_m)[1]]}{N} \tag{12}$$

$$REAL\_NORMAL\_EST = \frac{\sum[\hat{\mathcal{X}}_{normal_{test}}[2]]}{N} \tag{13}$$

$$REAL\_BOT\_EVA = \frac{\sum[\hat{\mathcal{X}}_{botnet_{test}}[1]]}{N} \tag{14}$$

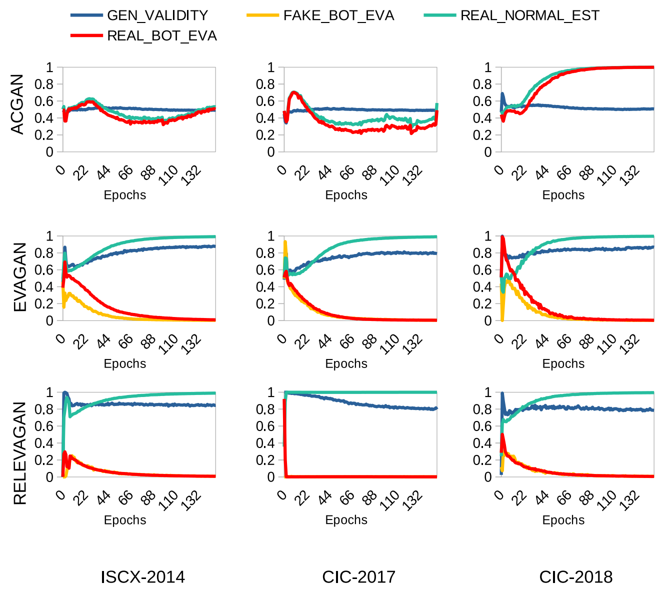


Fig. 4. The estimations on test data and data generated by the relative GANs generated data.

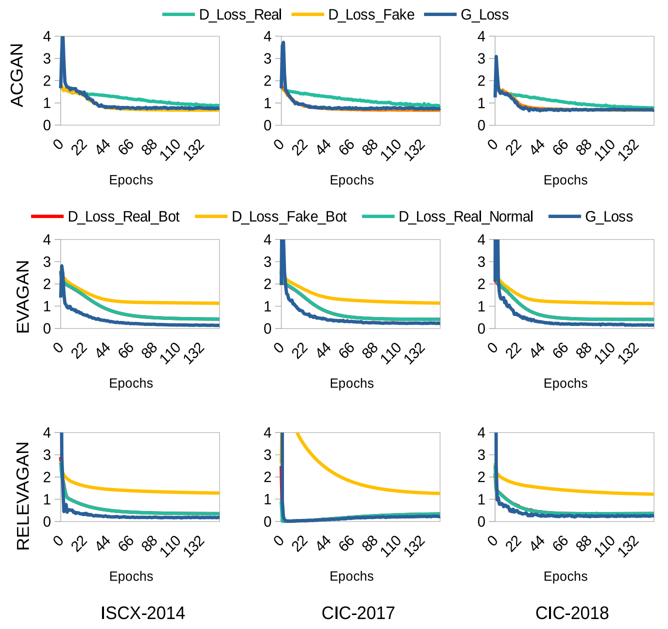


Fig. 5. The training losses for ACGAN, EVAGAN and RELEVAGAN.

5.1. Detection performance

The discussion on the performance comparison of ACGAN with EVAGAN has been mentioned in the EVAGAN paper; however, in Fig. 4, the estimations for ACGAN have also been included for a clear comparison. The performance of ACGAN in discerning the minority class deteriorates in low data regimes, especially in cybersecurity datasets. On the other hand, $\mathcal{D}$ of EVAGAN discerns the difference between the minority of majority classes. However, regarding RELEVAGAN, the estimations tend toward the desired values quicker than EVAGAN after a few initial unstable values. The RELEVAGAN was pre-trained for the values coming from the real and generated sources, especially from epochs 0–44. This is the contribution of the DRL attacker part. The most interesting case occurs for the CIC-2017 dataset, which took only a few epochs to train itself and prepare for the minority and majority classes. This could be due to non-ambiguous discrimination seen by RELEVAGAN between the malicious and normal class samples in CIC-2017 that the other two datasets did not have. A further investigation could be required; however, the role of DRL with EVAGAN demonstrates the value addition, especially for the CIC-2017 dataset. This pattern encourages us to further explore the potential of RELEVAGAN on other datasets, which we leave to future work.

We are using the EVAGAN as the base model, which does not require adversarial training of dedicated ML classifiers because the $\mathcal{D}$ of EVAGAN itself works as an adversarial aware botnet detector. So RELEVAGAN also does not need the adversarial training; however, the DRL attacker's evasions must be fed to the $\mathcal{D}$. This step imitates the back-propagation step in GAN training.

GEN_VALIDITY also illustrates the early convergence of $\mathcal{G}$, which achieves the Nash equilibrium quickly in RELEVAGAN while the $\mathcal{G}$ of EVAGAN is still learning. Since our main objective is to improve the detection performance of our model for botnets in low data regimes, we do not need to let the $\mathcal{G}$ train for a larger number of epochs. For example, if we achieve 100% detection performance, as in the case of the CIC-2017 dataset, we can stop the GAN training after a couple of steps.

5.2. Stability

Fig. 5 demonstrates the $\mathcal{D}$ and $\mathcal{G}$ losses for ACGAN, EVAGAN and RELEVAGAN. It turns out that the losses tend to converge for all the GANs. The values for RELEVAGAN tend to achieve the lowest point sooner than both EVAGAN and ACGAN. The values for D_Loss_Fake for RELEVAGAN are higher than other GANs. This can be because the $\mathcal{D}$ of RELEVAGAN is struggling to discriminate between the evasion generated by the DRL and the $\mathcal{G}$. The evasion samples coming from DRL are labelled as 'REAL', and when similar samples arrive from the $\mathcal{G}$, the wrong estimation increases the loss. However, the overall detection performance improves because the $\mathcal{G}$ now tends to generate the samples within the semantic constraints. There can be a possibility of mode collapse here; however, the detection performance is better than EVAGAN, so we can disregard that factor.

5.3. DRL attacker reward/evasions

Fig. 6 shows the number of evasions the DRL attacker generated or the reward collected during the exploration or training phase of RELEVAGAN of the first ten epochs. No evasions happen after epoch number eight. Note that the highest number of evasions was in the CIC-2017 dataset, which gives the best results for estimations in Fig. 4. A similar pattern is seen in the cases of ISCX-2014 and CIC-2018 datasets, but no evasions for the CIC-2018 dataset were generated after epoch three. If we correlate the number of generated evasions by the DRL attacker with the performance of the $\mathcal{D}$, it turns out that there is an inverse relationship between the number of evasions and convergence of the training of RELEVAGAN. Trying the model on other datasets can give more insights into this relationship.

5.4. Time complexity

Fig. 7 shows ACGAN, EVAGAN and RELEVAGAN training time complexity for the three datasets. RELEVAGAN training time for 150 epochs is always greater than that of EVAGAN because DRL has its own cost. However, we get the benefit of early convergence to achieve the maximum detection performance as manifested in Fig. 4 for the CIC-2017 dataset. Hence we achieve the performance of 100% in the time way less than taken by

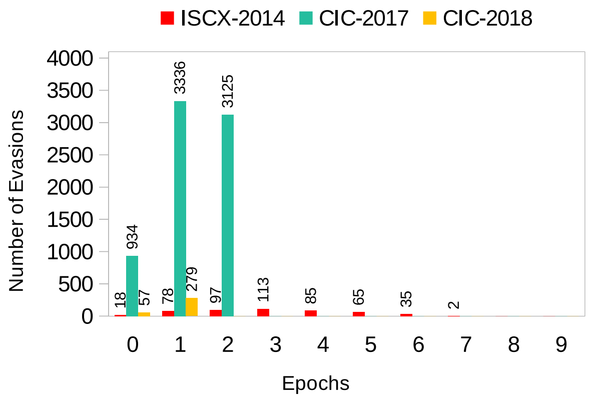


Fig. 6. RELEVAGAN reward/evasions during DRL attack on EVAGAN discriminator for the three datasets.

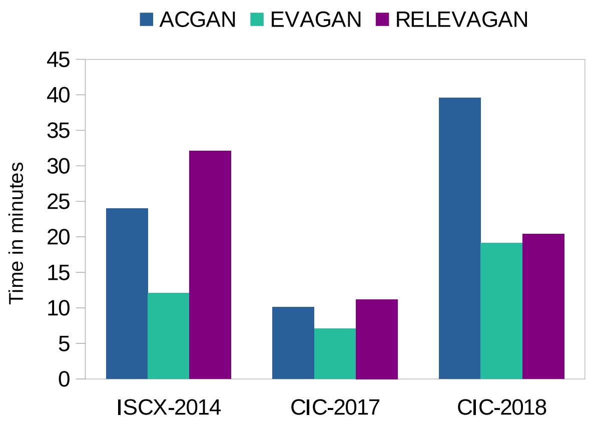


Fig. 7. Time complexity.

EVAGAN, but it turns out to depend on the dataset. As for the case of the ISCX-2014 dataset, the time complexity is even more than ACGAN's training time. Hence, we also consider working toward more datasets for greater insight into the time complexity pattern. Two variables discriminate the CIC-2017 from ISCX-2014 and CIC-2018. One is the botnet type, and the other is the ratio of the botnet samples to the normal traffic samples. It makes the problem non-trivial to be considered as potential future research.

6. Comparison of RELEVAGAN with peer techniques

The RELEVAGAN model is an enhanced version of EVAGAN, dedicated to low data regimes for learning adversarial evasion examples generated during GAN training with the additional support of DRL. So the most suitable existing model for the comparison can be EVAGAN; however, this section mentions peer techniques similar to RELEVAGAN that indirectly address the adversarial evasion problem using DRL while preserving the functionality of the attack operation. Authors in [11,27,28] have provided a DRL-based model to attack a malware detector by making changes based on a DRL agent output as a result of a reward coming out of the classifier. A similar concept has been applied while designing the DRL attacker in RELEVAGAN. A closer work to RELEVAGAN is [12] that uses the botnet datasets to generate evasion samples using DRL; however, RELEVAGAN is different in a way that it automatically makes the target learn the attack samples being generated from the DRL agent so the adjacent EVAGAN model learns simultaneously during training. This synergy makes the generator training faster. The work done by [10] is also significantly relevant to RELEVAGAN; however, they are using adversarial training at the end of the attack generation while RELEVAGAN learns that attacks immediately after its discovery which makes it a proactive approach as compared to the peer work. A tabular comparison of the parent model of the proposed RELEVAGAN with the state-of-the-art GANs has been provided in the EVAGAN paper [19].

7. Conclusion

The semantic-aware adversarial botnet detector is essential for countering modern evasion attacks. In this regard, the detection models must proactively know the possible adversarial perturbations. Researchers employ DRL to generate adversarial evasion examples to preserve the original functionality of the botnet/malware samples. The motivation is to generate samples that could be used later for adversarial training of the ML classifiers. We propose RELEVAGAN, which proves to be an adversarial semantic-aware evasion detection model that does not need exclusive adversarial training. The discriminator of RELEVAGAN is based on EVAGAN, which did not consider the semantic awareness introduced in RELEVAGAN. We have used the three datasets used by the EVAGAN paper for better comparison and coherency. The results demonstrate the supremacy of RELEVAGAN to EVAGAN in early convergence to the detection model training. Although at the cost of more training time, the RELEVAGAN model converges at least 20 iterations before the EVAGAN does in all the datasets. In CIC-2017, the convergence is acquired in a couple of iterations. The training time is somehow greater than EVAGAN for all the datasets; however, the cost is worth it as it provides robustness against DRL-generated evasion attacks.

We highly recommend testing RELEVAGAN's performance against other cybersecurity datasets for future research. Few-shot learning could also be tried on the datasets used in this paper to compare performance.

CRediT authorship contribution statement

Rizwan Hamid Randhawa: Conceived and designed the analysis, Collected the data, Contributed data or analysis tools, Performed the analysis, Writing – original draft. **Nauman Aslam:** Conceived and designed the analysis, Support and guidance. **Mohammad Alauthman:** Collected the data, Contributed data or analysis tools. **Muhammad Khalid:** Conceived and designed the analysis. **Husnain Rafiq:** Conceived and designed the analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] M. Gupta, C.M. Parra, D. Dennehy, Questioning racial and gender bias in AI-based recommendations: Do espoused national cultural values matter? Inf. Syst. Front. 24 (5) (2022) 1465–1481.

[2] A. McCarthy, E. Ghadafi, P. Andriotis, P. Legg, Functionality-preserving adversarial machine learning for robust classification in cybersecurity and intrusion detection domains: A survey, J. Cybersecur. Priv. 2 (1) (2022) 154–190.

[3] N. Papernot, P. McDaniel, X. Wu, S. Jha, A. Swami, Distillation as a defense to adversarial perturbations against deep neural networks, in: 2016 IEEE Symposium on Security and Privacy, SP, IEEE, 2016, pp. 582–597.

[4] R.H. Randhawa, N. Aslam, M. Alauthman, H. Rafiq, F. Comeau, Security hardening of botnet detectors using generative adversarial networks, IEEE Access 9 (2021) 78276–78292.

[5] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, P. McDaniel, Ensemble adversarial training: Attacks and defenses, 2017, arXiv preprint arXiv:1705.07204.

[6] M. Usama, M. Asim, S. Latif, J. Qadir, et al., Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems, in: 2019 15th International Wireless Communications & Mobile Computing Conference, IWCMC, IEEE, 2019, pp. 78–83.

[7] Z. Lin, Y. Shi, Z. Xue, IDSGAN: Generative adversarial networks for attack generation against intrusion detection, in: Pacific-Asia Conference on Knowledge Discovery and Data Mining, Springer, 2022, pp. 79–91.

[8] P.T. Duy, N.H. Khoa, A.G.-T. Nguyen, V.-H. Pham, et al., DIGFuPAS: Deceive IDS with GAN and function-preserving on adversarial samples in SDN-enabled networks, Comput. Secur. 109 (2021) 102367.

[9] Q. Cheng, S. Zhou, Y. Shen, D. Kong, C. Wu, Packet-level adversarial network traffic crafting using sequence generative adversarial networks, 2021, arXiv preprint arXiv:2103.04794.

[10] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, M. Colajanni, Deep reinforcement adversarial learning against botnet evasion attacks, IEEE Trans. Netw. Serv. Manag. 17 (4) (2020) 1975–1987.

[11] H.S. Anderson, A. Kharkar, B. Filar, D. Evans, P. Roth, Learning to evade static PE machine learning malware models via reinforcement learning, 2018, arXiv preprint arXiv:1801.08917.

[12] D. Wu, B. Fang, J. Wang, Q. Liu, X. Cui, Evading machine learning botnet detection models via deep reinforcement learning, in: ICC 2019-2019 IEEE International Conference on Communications, ICC, IEEE, 2019, pp. 1–6.

[13] N.S. Alfaiz, S.M. Fati, Enhanced credit card fraud detection model using machine learning, Electronics 11 (4) (2022) 662.

[14] N.V. Chawla, K.W. Bowyer, L.O. Hall, W.P. Kegelmeyer, SMOTE: synthetic minority over-sampling technique, J. Artif. Intell. Res. 16 (2002) 321–357.

[15] J. Engelmann, S. Lessmann, Conditional wasserstein GAN-based oversampling of tabular data for imbalanced learning, 2020, arXiv preprint arXiv:2008.09202.

[16] G. Kovács, An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets, Appl. Soft Comput. 83 (2019) 105662.

[17] Y. Yin, An enhancing framework for botnet detection using generative adversarial networks, in: 2018 International Conference on Artificial Intelligence and Big Data, ICAIBD, IEEE, 2018.

[18] C. Yin, Y. Zhu, S. Liu, J. Fei, H. Zhang, Enhancing network intrusion detection classifiers using supervised adversarial training, J. Supercomput. (2019) 1–30.

[19] R.H. Randhawa, N. Aslam, M. Alauthman, H. Rafiq, EVAGAN: Evasion generative adversarial network for low data regimes, 2021, arXiv preprint arXiv:2109.08026.

[20] A. Odena, C. Olah, J. Shlens, Conditional image synthesis with auxiliary classifier GANs, in: International Conference on Machine Learning, PMLR, 2017, pp. 2642–2651.

[21] S. Huang, K. Lei, IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks, Ad Hoc Netw. 105 (2020) 102177.

[22] M.H. Shahriar, N.I. Haque, M.A. Rahman, M. Alonso, G-IDS: Generative adversarial networks assisted intrusion detection system, in: 2020 IEEE 44th Annual Computers, Software, and Applications Conference, COMPSAC, IEEE, 2020, pp. 376–385.

[23] M. Ring, A. Dallmann, D. Landes, A. Hotho, IP2vec: Learning similarities between IP addresses, in: 2017 IEEE International Conference on Data Mining Workshops, ICDMW, 2017, pp. 657–666.

[24] M. Ring, D. Schlör, D. Landes, A. Hotho, Flow-based network traffic generation using generative adversarial networks, Comput. Secur. (2018).

[25] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, et al., Human-level control through deep reinforcement learning, Nature 518 (7540) (2015) 529–533.

[26] M.M. Alauthman, An Efficient Approach To Online Bot Detection Based on a Reinforcement Learning Technique, University of Northumbria at Newcastle (United Kingdom), 2016.

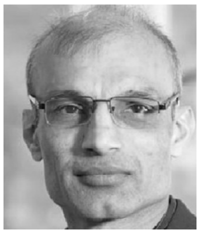
[27] Z. Mao, Z. Fang, M. Li, Y. Fan, EvadeRL: Evading PDF malware classifiers with deep reinforcement learning, Secur. Commun. Netw. 2022 (2022).

[28] Z. Fang, J. Wang, B. Li, S. Wu, Y. Zhou, H. Huang, Evading anti-malware engines with deep reinforcement learning, IEEE Access 7 (2019) 48867–48879.

[29] P.C. Nguyen, N.N. Vlassis, B. Bahmani, W. Sun, H. Udaykumar, S.S. Baek, Synthesizing controlled microstructures of porous media using generative adversarial networks and reinforcement learning, Sci. Rep. 12 (1) (2022) 9034.

[30] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, W. Zaremba, Openai gym, 2016, arXiv preprint arXiv:1606.01540.

Rizwan Hamid Randhawa received a BS degree in Electronic Engineering from International Islamic University Islamabad, Pakistan and Master in Computer Science from Information Technology University, Lahore, Pakistan. He has vast experience with embedded systems in Pakistan's private and public sector organisations. He is pursuing his Ph.D. in Computer Science from Northumbria University, Newcastle upon Tyne, UK. His research interests include AI-based botnet detection, IoT Security and Embedded Systems Design & Development for IoT platforms.



Nauman Aslam is a Professor in the Department of Computer and Information Science, Northumbria University, UK. Before joining Northumbria University as a Senior Lecturer in 2011, he worked as an Assistant Professor at Dalhousie University, Canada. He received his Ph.D. in Engineering Mathematics from Dalhousie University, Canada, in 2008. Dr Nauman leads the Network Systems and Security research group at Northumbria University. His research interests cover diverse but interconnected areas related to communication networks. His current research focuses on addressing wireless body area networks and IoT, network security, QoS-aware communication in industrial wireless sensor networks, and Artificial Intelligence (AI) application in communication networks. He has published over 100 papers in peer-reviewed journals and conferences. Dr Nauman is a member of IEEE.

Mohammed Alauthman received his Ph.D. degree from Northumbria University Newcastle, the UK, in 2016. He received a B.Sc. degree in Computer Science from Hashemite University, Jordan, in 2002 and an M.Sc. in Computer Science from Amman Arab University, Jordan, in 2004. He is Assistant Professor at the Information Security Department at Petra University, Jordan. His research interests include Cyber-Security, Cyber Forensics, Advanced Machine Learning and Data Science applications.

Muhammad Khalid received the Ph.D. degree in computer science from Northumbria University, Newcastle Upon Tyne, U.K. He lectures at the School of Computer Science, University of Hull, UK. Earlier, he worked as a research fellow at the University of Lincoln, UK. His research interests include reinforcement learning, autonomous systems, safety in robotics, the Internet of Things, and autonomous valet parking.

Husnain Rafiq received the B.S. and M.S. degrees in Computer Science from the Capital University of Science and Technology, Islamabad, Pakistan in 2015 and 2017, respectively. He did a Ph.D. from Northumbria University Newcastle upon Tyne, UK, in 2022. He is currently working as a lecturer in Cybersecurity at the Department of Computer Science, Edgehill University, UK. His area of research includes Information Security and Forensics, Machine learning and Malware Analysis.