IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 16, 2021, accepted May 18, 2021, date of publication May 24, 2021, date of current version June 3, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3083421

Security Hardening of Botnet Detectors Using Generative Adversarial Networks

RIZWAN HAMID RANDHAWA[1], NAUMAN ASLAM[1], (Member, IEEE), MOHAMMAD ALAUTHMAN[2], HUSNAIN RAFIQ[1], AND FRANK COMEAU[3], (Member, IEEE)

[1]Department of Computer and Information Systems, Northumbria University, Newcastle upon Tyne NE1 8ST, U.K.
[2]Department of Information Security, University of Petra, Amman 11196, Jordan
[3]Department of Engineering, St. Francis Xavier University, Antigonish, NS B2G 0B7, Canada

Corresponding author: Rizwan Hamid Randhawa (rizwan.randhawa@northumbria.ac.uk)

This work was supported by Northumbria University Research and Development Fund (RDF).

ABSTRACT Machine learning (ML) based botnet detectors are no exception to traditional ML models when it comes to adversarial evasion attacks. The datasets used to train these models have also scarcity and imbalance issues. We propose a new technique named *Botshot*, based on generative adversarial networks (GANs) for addressing these issues and proactively making botnet detectors aware of adversarial evasions. *Botshot* is cost-effective as compared to the network emulation for botnet traffic data generation rendering the dedicated hardware resources unnecessary. First, we use the extended set of network flow and time-based features for three publicly available botnet datasets. Second, we utilize two GANs (vanilla, conditional) for generating realistic botnet traffic. We evaluate the generator performance using classifier two-sample test (C2ST) with 10-fold 70-30 train-test split and propose the use of 'recall' in contrast to 'accuracy' for proactively learning adversarial evasions. We then augment the train set with the generated data and test using the unchanged test set. Last, we compare our results with benchmark oversampling methods with augmentation of additional botnet traffic data in terms of average accuracy, precision, recall and F1 score over six different ML classifiers. The empirical results demonstrate the effectiveness of the GAN-based oversampling for learning in advance the adversarial evasion attacks on botnet detectors.

INDEX TERMS Botnet detection, GANs, adversarial evasion attacks, unbalanced datasets.

I. INTRODUCTION

The ever-increasing sophistication in the design of botnets is alarming for digital economies. With the increase in the number of online transactions, mobile payments and cryptocurrencies, botnets could be in greater demand by cybercriminals in the dark web. Soon botnets will be used to mine Bitcoins on a gigantic scale and rallied to perform more seriously detrimental DDoS attacks than ever [1] on traditional as well as Internet of Things [2]. These attacks exploit the inherent weaknesses in existing intrusion detection systems (IDSs) and analysis frameworks.

Recently, machine learning (ML) has emerged as a powerful tool to design sophisticated IDSs for botnet detection. With the modern enhancements in hardware capabilities of computer systems, deep learning (DL) has become more prominent among the other ML techniques. Deep Neural Networks have been successfully used for solving classification problems with high accuracies. To facilitate online learning, researchers use deep reinforcement learning (DRL) which does not require classifiers to be trained exclusively offline for new updates [3]. However, due to vulnerabilities to adversarial attacks, the training model may learn malicious samples as benign and like a snowball effect train itself on false negatives. This malicious effect is called a poisoning attack which can be caused by an adversarial evasion attack. The adversarial evasion attack can be defined as crafting the input sample to evade the machine learning-based detectors as a result of probing an IDS [4]. In this way, the behaviour of the decision boundary of an ML model can be learned and replicated to a local machine. As a result, example inputs are generated that evade the copied learning model. These evading examples are fed to the target system that most likely is unable to classify those as malign hence poisoning the online learning. This crucial concern about the security of the ML-based systems has also gained significant attention in botnet detection [5]. The state-of-the-art IDS for botnet detection would ideally learn beforehand such crafting

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Hui Yeh.

by an attacker to minimize the effect of an evasion attack. The proposed work in this paper is an attempt in this direction.

To this end, we propose a technique to generate realistic botnet data using Generative Adversarial Networks (GANs) to improve the classifiers' decision making to detect potential evasion samples. GANs have proved to be highly effective in some recent research works [6]–[10]. A GAN is a combination of two different AI models competitively learning to generate realistic samples. The use of GANs in a particular way makes our research novel as compared to some previous works [6], [11]. In our proposed methodology, we use a classifier two-sample test (C2ST) for assessing the quality of generated traffic data. We propose the use of 'recall' as the evaluation metric of the generator performance in C2ST instead of 'accuracy' for the purpose of learning evasions. Since the false negatives are possible evasions, our objective function should minimize the value of 'recall' during GAN training for the exploration of generator weights that generate samples with maximum similarity with the real samples, The detail of this proposition will be discussed in SectionIV.

The other major aspect of this research is addressing the botnet dataset scarcity and imbalance issues in low data regimes without incurring the additional cost of generating actual attacks using multiple computers. In such scenarios, the machine learning classifiers over-fit the majority class and fall short in generalizing the test set [12]. The motivation for using GANs for data oversampling is their effectiveness in mimicking complex probably distributions [10]. To address the low-data regime problem, synthetic oversampling techniques like SMOTE [13] are employed, but these techniques depend on algorithms like nearest neighbours and linear interpolation which make them unsuitable for high-dimensional and complex probability distributions of data [10]. Although the classifiers' performance could be improved to a certain extent by using synthetic data oversampling, GANs can be more effective [14].

To maintain the coherency of the experiments, we have utilized an extended flow and time-based feature set of network traffic [5] for three botnet datasets, i.e. ISCX-2014, CIC-IDS2017 and CIC-IDS2018, from Canadian Institute of Cybersecurity. Traffic was generated using two different GANs. Six different classifiers (extreme gradient boosting or xgboost (XGB), random forests (RF), decision trees (DT), linear regression (LR), k-nearest neighbours (KNN) and naive bayes (NB) with default parameters) were used for generator evaluation and post augmentation detection. The main purpose of using two different GANs was to compare their performance overall three botnet datasets using the six classifiers and to choose the best pair (GAN & classifier) to be used in continued research work. The choice of six different classifiers is based on a related work that used all of these classifiers for comparative analysis for black-box testing [11]. In order to save time, we did not include support vector machine (SVM) due to its expensive training time. Instead of gradient boosting (GB), we used XGB which has lesser timer complexity [15]. The neural network (NN) was not used because the discriminator in GAN is also an NN, we wanted to test the performance of GAN only on non-neural network based ML-classifiers.

The botnet traffic data generated after the GANs' training was used to augment the original train set to improve the performance of the classifiers. The tests were performed using 10-fold, 70-30 train-test split for the above-mentioned datasets for both generator performance in C2ST and post augmentation testing. The performance of the botnet detectors after augmentation of the generated data was evaluated based on accuracy, recall, precision and F1 scores for all six classifiers. We demonstrate by experiments that using our proposed approach, we could not only better evaluate the GAN generated data but also improve the botnet traffic detection.

The contributions of this work can be summarised as follows:

1) We propose Botshot as a technique to proactively learn adversarial evasions in terms of recall score for quantitative evaluation of GANs.
2) To the best of our knowledge, this is the first work to use an extended set of features [16] for botnet data generation using GANs.
3) To the best of our knowledge no previous work has provided the comparison of GAN based oversampling with state-of-the-art synthetic oversamplers for improvement in botnet detection.
4) We address the data imbalance issue in three botnet datasets to improve the model's detection performance.

The rest of this paper is organized as follows. Section II covers the related work, section III throws some light on background concepts, Section IV illustrates the proposed methodology, Section V explains the implementation details. Section VI shows results, Section VII gives the reasoning for the inferences from the experimental outcomes and Section VIII concludes this paper.

II. RELATED WORK

Currently, adversarial attacks on AI-based systems are of considerable attention in botnet detection as well [5]. AI is an open-source tool that can be compromised by cybercriminals for the incarnation of more intelligent evasions. AI-based IDSs need continuous online learning to keep them up to date regarding zero-day attacks. To address the problem of online learning, researchers used DRL [17], [18]. In this way, it is not necessary to train the system exclusively offline for new updates. The training model gets mature with time. However, due to the vulnerabilities of deep learning models to adversarial examples, the use of DRL may be disastrous because the system may learn a malicious input as a benign sample and like a snowball effect train its model on false negatives [19]. This type of evasion of a malign sample can be caused by adversarial attacks such as model extraction or black-box attacks [4]. In this type of attack, the behaviour of the decision boundary of an ML model can be learned and replicated to a local machine. After this step, the example

inputs are generated that evade the copied learning model. These evading examples are input to the target system that most likely is unable to classify those as malign. In [11], the authors have proposed an attack and a countermeasure to demonstrate that AI-based IDSs are prone to adversarial attacks. Another hot issue currently being faced by the research community is the property of transferability of adversarial examples [20]. It means that if an adversarial attack is valid for one type of ML model, then it is highly likely that it will be effective against another model without added effort by the adversary [21].

The adversarial ML deals with ruggedizing the existing models by learning the adversarial examples [22]. This term was coined for computer vision problems but is also valid in network and computer security, since both these fields are highly dependent on ML models [23]–[25]. To address the transferability of adversarial examples, researchers use generative adversarial networks (GANs) to train the existing deep learning model, with synthetic adversarial examples. In this way, an IDS may become aware of a zero-day attack and avoid it. A GAN is a combination of a generative and a discriminative model [26]. The concept of GANs is related to game theory, where a generative neural network creates adversarial examples and a detector neural network tries to classify the input samples as generated or real examples. Based on a mini-max game the system tends to damp down to a Nash equilibrium. Our work is also based on using GAN generated samples to address the adversarial evasion attacks on botnet detectors.

In [27], the authors proposed a DRL-based model to evade the botnet classifiers based on the output of the machine learning classifiers under black-box attack. According to the authors, this work was the first on evasion attacks using a DRL agent. Following similar research, the authors in [28] proposed a technique to generate evasion attacks on the botnet detector as a black-box. In a recent paper, the authors present a new dataset generated using DRL evasion attacks on botnet detectors [29]. These adversarial sample generation using DRL can be used in conjunction with GANs to oversample the adversarial examples which we leave to future work.

Adversarial examples as a result of black-box attacks can be detrimental for an IDS because malign inputs can become part of benign network applications. The research community has highlighted the gravity of this issue [5], [30], [31]. In [32], the authors have proposed ensemble techniques to address the problem of transferability for adversarial attacks in computer vision but the efficient and reliable techniques in IDS design for the majority of botnet behaviours are missing. For example, in [33] authors deal only with the domain generation aspect of the botnet evasion using GANs; however, not all modern botnets use DNS-based communication because these techniques are more prone to detection than peer-to-peer botnets [34]. In [6], the authors propose another GAN based technique for botnet detection named Bot-GAN. The authors have extended the discriminator output from two to three traffic types, i.e. normal, fake botnet and real botnet. By using GANs, they have achieved better accuracy and lower false positives than previous studies. However, the feature set selection is limited in their work. In addition, the authors are not balancing their data set with increasing generated examples as the accuracy drops beyond a certain value which is one of the goals of our research work. In another work, authors have proposed the avoidance of model theft or extraction [35]. The same concept can be applied in conjunction with GAN implementation for botnet detection.

In network-based anomaly detection for botnets, mainly two schemes are employed: deep packet-based and network flow-based inspection. Since botnets have evolved with time, most now use packet encryption. Another issue with deep packet inspection (DPI) is the serious concern about user privacy [16]. As a solution to these problems, most of the research work is now performed using network flow-based features to detect anomalies. For botnet detection, researchers have extensively studied different flow-based features based on their effectiveness for the desired outcome [36]–[38]. The researchers in [39] have devised new time and flow-based traffic features for detecting Tor traffic. This extended set of features can improve the detection of the network anomaly in case of botnet detection as well because most botnet traffic types have underlying similarities in spacio-temporal communication patterns [27].

The importance of ML in IDS, especially botnet detection, has gained significant attention in the last decade [40]–[45]. The major concern raised by the authors in [40] is that most of the research works had addressed only 25% of the modern attacks because they were based on outdated datasets. With the continuous evolution in botnet evasion techniques, it is necessary to update benchmark data sets [46]. To fulfill this objective, many researchers have proposed the generation of new data sets [47]–[49]. However, these proposed methods of generating data sets have two drawbacks:

- Data set generation can be costly due to the involvement of multiple machines
- The generated traffic may not depict the real scenario of attacks [16]

The above-mentioned problems can be addressed in two ways. One is to merge all the benchmark data sets publicly available into a new data set. The authors of [16] used this approach to generate a new data set, taking into consideration the common updated attacks with eleven different evaluation characteristics of IDS data sets. But the problem of obsolescence [40] remains. We need to continuously update the data sets to maintain deterrence against evolving botnets [16]. Hence, for the generation of botnet datasets, we need unseen attacks data in large amount to address the issues of dataset imbalance and obsolescence. The data generation can be opted using synthetic oversampling methods that can solve both the aforementioned problems.

In the light of the above discussion, we consider GANs as a suitable choice to generate unexplored botnet data to

address the dataset imbalance and scarcity issues along with mitigating the effects of adversarial evasion attacks.

III. BACKGROUND

This section gives some background on the common types of adversarial attacks, the datasets with the feature set and a brief explanation of the functionality of GANs used in this research.

A. ADVERSARIAL EVASION ATTACKS

The adversarial evasion can be defined as a perturbed version of an input sample x as x* such that the $x^* = x + \eta$, where η is a carefully crafted perturbation. When making an adversarial attack, η could be sought and selected so that the classifier is unable to discriminate the x* from x. The adversary can make two types of adversarial attacks in general, respectively white-box and black-box attacks [50]. In white-box attacks, the attacker knows not only the gradient information for the loss function of the model but also source of the training data, hyper-parameters, model architectures, numbers of layers, activation functions and model weights. In other words, the attacker has the full knowledge of the model and even the direction of the gradient. The attacker can create a perturbation that most likely could increase the loss value. In black-box attacks, the adversary has no inside information except the output on a particular input. This is also called probing. The attacker has no knowledge of the model and can only know the response of the system. The white-box attacks can be transferable so that they can be used to attack a black-box service due to this property of transferability. Hence most of the attacks generated are white-box attack due to the ease of generation, but the transferability property can be effectively exploited for black-box attacks [21], [50]. The purpose of this research work is to make the classifiers adept at proactively knowing the adversarial examples so that the effect of black-box attacks can be mitigated, especially on ML-based botnet detectors.

B. GENERATIVE ADVERSARIAL NETWORKS (GANs)

The GAN is a combination of two neural networks among which the one that generates samples is called generator ($\mathcal{G}$) and the other that evaluates the generated samples is called discriminator ($\mathcal{D}$). The loss of $\mathcal{D}$ over generated data is fed back to $\mathcal{G}$ while $\mathcal{D}$'s weights are not updated so that $\mathcal{G}$ can try to mimic the real data probability density function more efficiently and fool the $\mathcal{D}$. In this work, two different GAN architectures have been used to generate botnet traffic. We chose these two primitive versions vanilla and conditional GAN, to keep the experiments simple for estimating their potential for botnet data generation. Exploration of a suitable GAN for generating botnet data is left as a future work. The following is a brief detail of each GAN architecture, used in this research, in terms of its loss function.

1) VANILLA GAN (GAN)

The generator model $\mathcal{G}$ in original/vanilla GAN can be represented as $\mathcal{G}$:z$\rightarrow$ $\mathcal{X}$ where z is the normal distribution from noise space and X is the real data distribution. The discriminator $\mathcal{D}$:$\mathcal{X}$ $\rightarrow$ [0,1] model is a classifier that outputs an estimate of probability whether the data coming from $\mathcal{G}$ is real or fake. The loss function of the combined model can be represented by Equation 1.

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

Here, $\mathbb{E}$ stands for the estimation of probability and x and z are the real and noise samples respectively, while p_{data} and p_z represent the probability distributions of real and noise data respectively. The goal in the mini-max game is to minimize the generator loss in creating data similar to the real data since the generator can not control the loss of $\mathcal{D}$ on real data but it can maximize the loss of $\mathcal{D}$ on generated data $\mathcal{G}(z)$. The loss function of $\mathcal{G}$ is given by Equation 2.

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

Figure 1 shows the block diagram of a classical/vanilla GAN with real and generated botnet traffic samples. Note that we refer to real botnet traffic samples as real_bots and GAN generated samples as GAN_bots in the remainder of the text. The noise samples labelled in three different colours depict the random space of all the possible samples including very close and partially close to the real_bots and even samples with very different distributions. The red-coloured samples are real_bots from X distribution and blue coloured samples are GAN_bots from $G(z)$ distribution. Figure 1 illustrates that $\mathcal{G}$ tries to generate a botnet example from noise z (Gaussian distribution) and gives it to $\mathcal{D}$ to get it evaluated. First, $\mathcal{D}$ is trained on samples from real botnet traffic data distribution X before getting input samples from $\mathcal{G}$ in Q space. The losses of $\mathcal{D}$ on real $L_{D(X)}$ and generated data ($L_{D(G(z))}$) are fed to $\mathcal{D}$ using backpropagation. In the next step, for training $\mathcal{G}$ in the forward propagation, the evaluation is done by $\mathcal{D}$ and the loss L_G is fed back to $\mathcal{G}$ to update its weights. This process keeps iterating until we reach a certain number of epochs. When the generator and discriminator achieve Nash equilibrium they do not learn further.

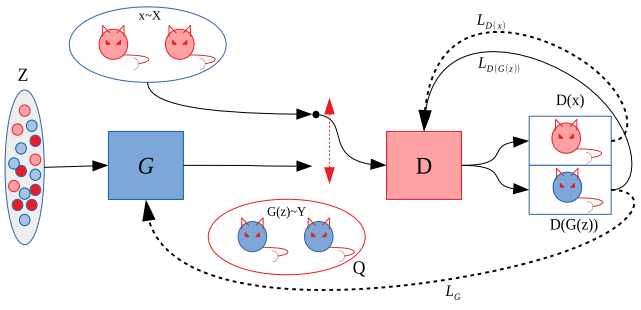


FIGURE 1. Block diagram of a classical GAN with real and GAN_bots.

TABLE 1. Existing vs Extended Feature Set: The features can be categorized in five different groups. Features in bold text were used in the current work for and the rest were dropped in preprocessing using algorithm 3.

Group	**Features [6]** (used in Existing Work)	**Extended Features [16]** (used in current work)
Categorical	Protocol	-
Number of packets	PX, NNP, PSP, IOPR, Reconnect	**fin_cnt**, **syn_cnt**, **rst_cnt**, **psh_cnt**, **ack_cnt**, urg_cnt, **cwe_cnt**, **ece_cnt**, **down_up_ratio**, fw_byt_blk_avg, fw_pkt_blk_avg, fw_blk_rate_avg, **bw_byt_blk_avg**, **bw_pkt_blk_avg**, **bw_blk_rate_avg**, **subfl_fw_pk**, **subfl_fw_byt**, subfl_bw_pkt, **subfl_bw_byt**, **fw_win_byt**, **bw_win_byt**, **fw_act_pkt**, **fw_seg_min**
Time	Duration	**fl_dur**, **fw_iat_tot**, **fw_iat_avg**, **fw_iat_std**, **fw_iat_max**, **fw_iat_min**, **bw_iat_tot**, **bw_iat_avg**, **bw_iat_std**, **bw_iat_max**, **bw_iat_min**, atv_avg, atv_std, atv_max, atv_min, , **idl_avg**, **idl_std**, **idl_max**, **idl_min**, **idl_min**
Size	FPS, TBT, APL, DPL, PV	**tot_fw_pk**, **tot_bw_pk**, **tot_l_fw_pkt**, **tot_l_bw_pkt**, **fw_pkt_l_max**, **fw_pkt_l_min**, **fw_pkt_l_avg**, **fw_pkt_l_std**, **Bw_pkt_l_max**, **Bw_pkt_l_min**, **Bw_pkt_l_avg**, **Bw_pkt_l_std**, **fw_hdr_len**, **bw_hdr_len**, **pkt_size_avg**, **fw_seg_avg**, **bw_seg_avg**
Flow	BS, PS, AIT, PPS	**fl_byt_s**, **fl_pkt_s**, **fl_iat_avg**, **fl_iat_std**, **fl_iat_max**, **fl_iat_min**, **fw_pkt_s**, **bw_pkt_s**, **pkt_len_min**, **pkt_len_max**, **pkt_len_avg**, **pkt_len_std**, **pkt_len_va**,
Flags	-	**fw_psh_flag**, bw_psh_flag, fw_urg_flag, bw_urg_flag

2) CONDITIONAL CLASSICAL GAN (CGAN)

The conditional version of vanilla GAN adds another feature as a class label to help GAN learn the probability distribution quicker as compared to the vanilla GAN. The loss functions for $\mathcal{D}$ and G, in this case, can be represented in Equations 3 and 4.

$$J_{\mathcal{D}} = -\frac{1}{2m}\sum_{i=1}^{m}\log \mathcal{D}(x_i, y_i) + \sum_{i=1}^{m}\log(1-\mathcal{D}(\mathcal{G}(z_i, y_i), y_i))] \tag{3}$$

$$J_{\mathcal{G}} = -\frac{1}{m}\sum_{i=1}^{m}\log \mathcal{D}(\mathcal{G}(z_i, y_i), y_i) \tag{4}$$

Here, m is the total number of examples in one batch selected for training the model, x is real data and y are the additional labels as a condition due to which this GAN is called conditional GAN. The batch size is the same for noise and real data inputs to $\mathcal{G}$ and $\mathcal{D}$, respectively.

IV. PROPOSED METHODOLOGY

We propose a GAN based methodology that could mimic and generate the botnet traffic and then augment the original dataset with the generated traffic to improve the detection performance of the botnet classifiers. The purpose of training the botnet classifiers on GAN generated data is to proactively learn the unseen samples with similar but slightly different probability distributions. Figure 2 shows the proposed scheme for generating realistic botnet traffic using a GAN and then augmenting the generated data to the original train set to improve detection. The following sections throw some light on the details of the methodology. The classifier two-sample test (C2ST) is a quantitative metric to evaluate whether two different samples of data have been taken from the same distribution. In other words, if we have samples real_bots ($\mathcal{X}_b$) and GAN_bots ($G(z)$) then we can assess if both samples have similar or the same probability distributions. The more the distributions overlap, the more is the chance that GAN_bots are realistic. The C2ST method has been shown in Algorithm 1. The GAN evaluation used in Botshot is different than C2ST in a single parameter. The intuition is that the metric in C2ST (i.e. accuracy) should be replaced with recall if we want to reduce the false negatives in the classifier performance in post augmentation testing. The false negatives are the possible evasions that are already present in the test set which the classifiers are not trained on. The main purpose of the Botshot is to suppress the false negatives i.e. improve the recall score. Hence, the C2ST has been tweaked so that the objective function becomes as given in the Equation 5

$$\hat{r}_{argmin} = \frac{1}{n_{test}}\sum_{z_i, l_i \in D_{test}} \mathbb{I}[\mathbb{I}(f(z_i) > \frac{1}{2}) = l_i] \tag{5}$$

Algorithm 1 C2ST Algorithm

Input: $\mathcal{X}_b$ (real_bots), $G(z)$ (GAN_bots), botnet_classifier
Output: $\mathcal{A}$(accuracy)

begin

/* Evaluation of Generated Data ($G(z)$) */

$t_r \leftarrow \mathcal{X}_b[0 : m(7/10)] \cup G(z)[0 : m(7/10)]$;
// Create a train set from 70% of examples in the unified set where m = total number of examples in $\mathcal{X}_b$ and $G(z)$

$t_s \leftarrow \mathcal{X}_b[m(7/10) : m] \cup G(z)[m(7/10) : m]$;
// Create a test set from 30% of examples in $\mathcal{X}_b$ and $G(z)$

train botnet_classifier on t_r

test botnet_classifier on t_s

compute $\mathcal{A} = (\text{TP} + \text{TN})/(\text{TP} + \text{TN} + \text{FP} + \text{FN})$;
// Compute accuracy using confusion matrix

In the above Equation, $\hat{r}$ is the recall score on D_{test} which is test set, n_{test} is the total number of samples in test set, z_i are the samples in test set, l_i are the labels, $f(z_i)$ is the conditional probability distribution $p(l_i = 1|z_i)$. The intuition is that if a GAN-bot is very close in probability distribution with a real bot, then the recall in Equation 5 should remain close

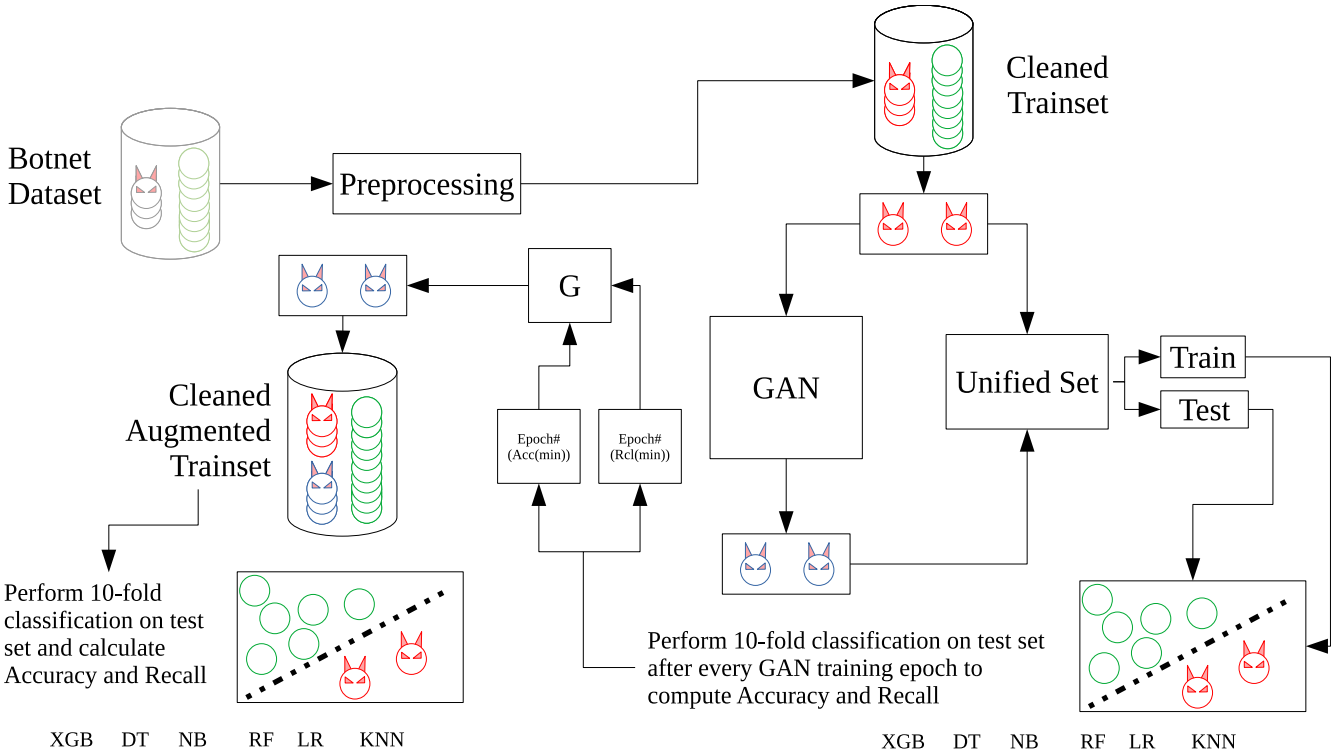


FIGURE 2. Functional diagram of Botshot.

to 50% which is also called a chance level. This means that the classifier was totally evaded or the sample was misclassified as a real bot. So if we use recall as the metric instead of accuracy, we can better minimize the false negatives because accuracy includes the value for false positives (FP) and true negatives (TN) given by Equation 6.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

However, in the recall, we only have true positives (TP) and false negatives (FN) as given by Equation 7.

$$Recall = \frac{TP}{TP + FN} \tag{7}$$

Since our objective function is to minimize false negatives in generator evaluation, we must choose the epochs in which the recall was the lowest instead of the epochs where accuracy was the lowest.

The complete functional block diagram of Botshot is illustrated in Figure 2. First, we preprocess the dataset as mentioned in Section V-F. From the cleaned train set, we extract the real bot samples of the set size for each dataset as mentioned in Table 2 and train our selected GAN. The GAN is trained for a certain number of epochs as given in Table 3. Once an epoch is completed, the trained $\mathcal{G}$ is used to generate data of size equal to the total number of real_bots given as an input to the GAN (Algorithm 3 line 16). The generated bot samples also called GAN_bots along with real_bots are combined into a unified set (Algorithm 3 line 17). This unified set is reshuffled and used to perform 10-fold train-test splitting using the selected classifier (Algorithm 3 lines 18-24).

TABLE 2. Distribution of normal and bot samples in three engineered 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

The results are compiled based on accuracy for C2ST and recall for Botshot (Algorithm 3 line 25-26). The epoch numbers for the minimum values of recall and accuracy for each classifier are saved. After generating the GAN-bots (of size = normal samples - real_bot samples) using the epoch number with minimum accuracy value for C2ST and minimum recall value for Botshot, we augment these into two sets $\mathcal{Y}_{C2ST}$ and $\mathcal{Y}_{Botshot}$ respectively (Algorithm 3 lines 27-32). Finally, we compute average accuracy, recall, precision and F1 scores after 10-fold train-test splitting of the two augmented datasets. The values have been recorded in Table 4 for the case of no augmentation, interpolations based synthetic oversamplers, GAN and CGAN based C2ST and Botshot evaluation for oversampling the botnet data.

V. IMPLEMENTATION

A. DATASETS

We have used three different datasets from the same source i.e. Canadian Institute of Cybersecurity (CIC) to keep the

TABLE 3. GAN Models.

Parameter	GAN	CGAN
Network Type	Densely Connected Feed Forward	
Number of Layers	$\mathcal{G}$: 6, $\mathcal{D}/\mathcal{C}$: 5	
Activations	$\mathcal{G}$: relu (All Layers), $\mathcal{D}$: relu (All Layers except output)	
Batch Size (b)	256	
Multiplier (n)	64	
Neurons in input layer	64	65
Neurons in layer 1	$\mathcal{G}: n \times 1 = 64$, $\mathcal{D}: n \times 4 = 256$	
Neurons in layer 2	$\mathcal{G}: n \times 2 = 128$, $\mathcal{D}: n \times 2 = 128$	
Neurons in layer 3	$\mathcal{G}: n \times 4 = 256$, $\mathcal{D}: n \times 1 = 64$	
Neurons in layer 4	$\mathcal{G}: n \times 8 = 512$	
Neurons in output layer	$\mathcal{G}: 64$, $\mathcal{D}: 1$	$\mathcal{G}: 65$, $\mathcal{D}: 1$
Layer Regularization	$\mathcal{G}, \mathcal{D}$: $BatchNorm$	
Optimizer	Adam (beta_1=0.5, beta_2=0.9)	
Loss Function	binary cross entropy	
Learning Rate	5e-1	

coherency of the experiments. The features were extracted using a utility called CICFlowmeter-v4 provided by CIC as well. Table 1 shows the features used in this work (in bold text). The features can be categorized into different groups based on the number of packets, time, size, flow and flags. It can be observed that we have not used any categorical features like IP addresses or port numbers while a previous work [6] did use protocol. Although previously used for Tor traffic characterization [39] and data set generation for CIC-IDS2017 and CIC-IDS2018 [16], these extended sets of features can improve the detection of the network anomaly in case of botnet detection as well [27]. To the best of our knowledge, this is the first work to use these extended features for botnet datasets for data generation using GANs. The number of samples of benign vs botnet is mentioned in Table 2. The following is the detail of each dataset and the botnet samples that were used in this work.

B. ISCX-2014 DATASET

The ISCX-2014 data set [51] is an amalgamation of three publicly available datasets ISOT [36], ISCX 2012 IDS [37] and CTU-13 [52]. The composition of dataset as explained on the ISCX website, seems promising in terms of generality, realism and representativeness. The generality can be defined as the richness of diversity of botnet behaviour. The realism is the closeness with the actual traffic captured and representativeness determines the ability to reflect the real environment, a detector would face in deployment. We have only used VIRUT botnet for this work among the 7 different botnets. The VIRUT was chosen because it had very few samples as compared to other botnets and not insufficiently scarce as Zeus bot. We could use SMTP or NSIS but their labels were not available on the website.[1] As a result we used a subset including all the normal traffic flows with VIRUT samples. In this way, we could use this dataset as a good example of an unbalanced set.

C. CIC-IDS2017 DATASET

The CIC-IDS2017 is a relatively recent dataset by CIC with the traffic of ARES botnet. The traffic was collected on Friday, July 7, 2017, during the day from 10:02 AM – 11:02 AM in the CIC facility. The dataset can be found on the CIC website.[2] We made a subset of this dataset using all the normal flows with botnet. The ratio of the number of samples has been mentioned in Table 2. The subset as the previously mentioned dataset is also a good kind of an unbalanced dataset for the particular research problem of this work.

D. CIC-IDS2018 DATASET

To create another subset of unbalanced dataset, we used CIC-IDS2018. This dataset included samples for ARES and ZEUS botnets. We created a subset of all the normal samples with just 2560 samples of botnet flows to mimic another unbalanced dataset.

E. FEATURE SELECTION

The performance of the botnet detectors can be highly dependent on the quality of dataset in general and the number of distinct features in particular. Using a limited feature set may not necessarily give a stronger classification decision as compared to an enhanced set of non-redundant features. Previously, [51] summarised the most important features from network flows that could be helpful in botnet detection. Almost all these features except a few are included in [5] that we have used in our work. We used the utility used by the authors, available on github named as CICFlowMeter-v4 to extract more than 80 flow & time-based features from the three different datasets'.pcap files. This utility can be used to extract the said features for any input.pcap files. Table 1 shows the feature set. The details of the features can be found on the source website.[3]

F. PREPROCESSING

For preprocessing of ISCX-2014 dataset, one may need to label it as well. For this purpose, we used the information provided on the CIC website for IPs associated with the particular botnets. After labelling, we performed preprocessing as mentioned in Algorithm 2. We removed all the high and low skewed values to suppress outliers, cleaned the data of the NaN and InF values, removed columns with zero standard deviation and scaled the features in [0,1] range. This is necessary to apply GANs with rectified linear unit (relu) activation function in its layers because the range of the output with relu function lies within this range. The CIC-IDS2017 and CIC-IDS2018 were already labelled. So we only did

[1] https://www.unb.ca/cic/datasets/botnet.html

[2] https://www.unb.ca/cic/datasets/ids-2017.html

[3] https://www.unb.ca/cic/datasets/ids-2018.html

TABLE 4. Results of data augmentation for different oversampling techniques: The original train set was augmented with botnet traffic samples generated from each of the different synthetic oversampling techniques including GANs, and the testing was done on the original test set.

Results on ISCX-2014 Dataset																								
	Accuracy						Recall						Precision						F1					
	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN
No. Aug	99.930	99.907	41.231	**99.895**	**99.443**	99.788	92.751	92.920	**96.222**	86.634	22.500	78.288	97.239	93.935	1.144	97.995	**94.676**	90.456	94.939	93.418	2.261	91.960	36.341	83.926
SMOTE_IPF	98.128	97.920	50.940	98.305	85.285	97.490	**97.620**	**97.102**	92.902	**97.200**	87.648	**97.271**	96.319	96.147	37.763	97.262	71.057	94.676	96.966	96.621	53.699	97.232	**78.487**	95.958
ProWSyn	98.240	97.978	49.711	98.319	84.683	97.552	96.586	96.825	94.066	96.636	**90.106**	96.662	97.636	96.578	37.277	97.849	69.187	95.402	97.107	**96.701**	53.393	97.242	78.271	96.028
polynom_fit_SMOTE	98.272	97.959	52.708	98.316	85.110	97.723	96.280	96.542	91.351	96.229	79.837	95.927	**98.046**	**96.787**	**38.522**	98.237	73.715	**96.612**	**97.156**	96.662	**54.191**	**97.223**	76.653	**96.268**
GAN(C2ST)	99.930	99.904	99.132	99.887	99.409	99.791	92.589	92.783	0.000	85.470	23.751	78.785	97.510	93.737	0.000	98.232	75.859	90.345	94.976	93.288	6.078	91.625	39.796	84.096
GAN(Botshot)	99.932	**99.908**	99.132	99.887	99.409	**99.791**	92.589	92.783	0.000	85.470	23.751	78.785	97.559	93.737	0.000	98.232	75.859	90.345	95.004	93.272	0.000	91.398	36.168	84.163
CGAN(C2ST)	99.932	99.904	99.292	99.887	99.405	99.790	92.896	92.783	0.000	85.506	23.139	78.780	97.441	93.737	0.000	98.422	76.282	90.341	95.068	93.425	0.000	91.540	4.387	83.827
CGAN(Botshot)	**99.932**	99.904	**99.292**	99.887	99.405	99.790	92.896	92.783	0.000	85.506	23.139	78.780	97.441	93.737	0.000	**98.422**	76.282	90.341	95.109	93.253	0.000	91.505	35.500	84.154
Results on CIC-2017 Dataset																								
	Accuracy						Recall						Precision						F1					
	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN
No. Aug	99.94	99.855	70.853	99.799	**98.037**	99.558	98.54	97.492	99.793	92.816	33.89	93.984	99.168	97.104	8.407	99.609	**82.598**	89.989	98.851	97.294	15.509	96.089	48.036	91.936
SMOTE_IPF	99.922	99.862	70.894	98.709	87.734	99.331	99.159	98.698	99.793	**99.073**	98.81	98.243	97.889	96.306	8.421	67.861	17.795	80.906	98.52	97.487	15.529	80.5	30.157	88.73
ProWSyn	99.875	99.767	72.214	98.316	87.329	98.923	**99.314**	**99.175**	99.793	98.782	**99.07**	**98.421**	96.132	92.684	8.785	61.634	17.352	71.825	97.692	95.816	16.148	75.887	29.53	83.04
polynom_fit_SMOTE	99.942	**99.865**	75.396	99.787	89.327	99.547	98.714	97.681	99.52	92.377	94.719	94.052	99.135	**97.326**	**9.782**	99.665	19.417	89.588	**98.922**	**97.501**	**17.813**	95.874	32.228	91.759
GAN(C2ST)	99.942	99.856	73.184	99.79	98.035	99.555	98.629	97.489	**99.898**	92.475	34.275	93.931	**99.186**	97.19	9.075	99.685	81.764	89.954	98.905	97.338	16.643	95.937	**48.285**	91.894
GAN(Botshot)	99.942	99.853	94.511	**99.808**	98.026	99.558	98.645	97.473	0	93.228	34.241	93.984	99.134	97.103	0	99.614	81.305	89.989	98.886	97.287	0	**96.307**	48.171	91.936
CGAN(C2ST)	99.942	99.854	94.547	99.802	97.666	99.558	98.647	97.473	0	92.968	20.238	93.984	99.169	97.137	0	99.68	64.088	89.989	98.853	97.311	0.149	96.009	47.774	91.921
CGAN(Botshot)	**99.942**	99.854	**94.547**	99.802	97.666	**99.558**	98.612	97.421	0	92.968	20.238	93.984	99.185	97.104	0	**99.687**	64.088	**89.989**	98.896	97.262	0	96.196	29.881	**91.936**
Results on CIC-2018 Dataset																								
	Accuracy						Recall						Precision						F1					
	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN	XGB	DT	NB	RF	LR	KNN
No. Aug	100	**100**	99.99	100	**99.955**	100	99.961	99.885	99	**100**	100	**100**	99.959	99.866	**100**	100	**93.536**	99.921	99.961	99.876	99.5	**100**	**96.658**	**99.96**
SMOTE_IPF	100	100	99.99	100	99.842	100	**99.987**	**99.935**	98.94	99.973	100	100	99.959	**99.974**	100	100	80.64	99.805	99 .973	**99.955**	99.47	99.987	89.28	99.902
ProWSyn	100	100	**100**	100	99.839	100	99.987	99.92	**100**	99.933	100	100	**100**	99.961	99.24	100	80.381	99.678	**99.993**	99.941	**99.62**	99.967	89.121	99.937
polynom_fit_SMOTE	100	100	99.99	100	99.842	100	99.933	99.923	98.61	99.987	100	100	99.973	99.855	100	100	80.597	99.729	99.953	99.89	99.3	99.994	89.252	99.864
GAN(C2ST)	100	99.999	99.34	100	99.944	100	99.884	99.769	0	99.974	100	99.974	99.961	99.856	0	100	92.172	99.922	99.921	99.811	0	99.986	89.926	99.947
GAN(Botshot)	100	99.99	99.34	100	99.944	100	99.884	99.769	0	99.974	100	99.974	99.987	99.856	0	100	92.172	99.922	99.935	99.811	0	99.987	95.926	99.947
CGAN(C2ST)	100	99.998	99.35	100	99.944	100	99.884	99.742	0	99.987	100	99.974	99.935	99.83	0	100	92.172	99.922	99.908	99.785	0	99.993	95.926	99.947
CGAN(Botshot)	**100**	99.998	99.35	**100**	99.944	**100**	99.884	99.742	0	99.987	**100**	99.974	99.935	99.83	0	**100**	92.172	**99.922**	99.908	99.785	0	99.993	95.926	99.947

preprocessing as mentioned earlier for these two data sets after extracting the unbalanced subsets.

Algorithm 2 Preprocessing

Input: $\mathcal{X}$ (original train set in csv format)
Output: $\mathcal{T}$ (preprocessed train set in csv format)

begin

$\mathcal{X}_n \leftarrow \mathcal{X}[Label = 0]$, $\mathcal{X}_b \leftarrow \mathcal{X}[Label = 1]$;
`// label flows based on malicious IPs`

Preprocess Data ; `// remove: outliers, rows with NaN and InF values, columns with std = 0 and scale the features in range(0,1)`

G. GAN HYPERPARAMETERS TUNING

The GAN tuning can be challenging especially working on tabular data because unlike images we have to assess the GAN performance based on some quantitative parameters (one of which is C2ST). However, if we tune a classical GAN carefully [53], we can make it work to generate realistic samples as we did in our particular case. After a random search of batch size and the multiple for the number of neurons in hidden layers of $\mathcal{G}$ and $\mathcal{D}$, we could explore the optimum values as highlighted in Figure 3. The choice was made based on the stability of losses of $\mathcal{D}$ and $\mathcal{G}$.

The rest of the hyperparameter details can be found in Table 3. We have used densely connected feed-forward neural network as the architecture in both $\mathcal{G}$ and $\mathcal{D}$. The activation functions are relu in all the hidden layers and sigmoid in the output layer of $\mathcal{D}$. The batch normalization was used in all the hidden layers. This technique regularizes the output of each hidden layer and stabilizes the GAN loss. The optimizer type used was Adam with binary cross-entropy (BCE) as the loss function. The learning rate was set as 5e-1 becasue that classifiers tend to get fooled more at this learning rate as compared to the lower values. So it can also be considered an exploratory value in our case.

Figure 3 shows that the loss values for the three losses are stable in case of a batch size of 256 and multiplier (n) equal to 64 for both GAN and CGAN. In all the other cases, the values are overshoot or not converging. These results were taken in ISCX-2014 dataset and were considered for the other two datasets as well. However, the individual loss values for the three datasets are manifested for each GAN in Figures 4, 5, 6, 7, 8 and 9. The GAN losses for each of the GAN are different for every data set used. Red coloured peaks are shown where $\mathcal{G}$ has a high loss L[G(z)]; then damps down in next few epoch and remains close to zero. This result means that $\mathcal{G}$ has achieved Nash equilibrium but it can be observed that more peaks arrive in which $\mathcal{D}$'s loss L[D(G(z))] also is increased along with L[G(z)] and sometimes even higher. This result happened when $\mathcal{G}$ has generated some sample which fooled the $\mathcal{D}$ pretending to be real but it was fake. It can be observed that most of the time during the training of the desired epochs, the GAN losses remain stable. The result could be very different if we do not use batch normalization which regularizes values in each hidden layer.

H. DATA AUGMENTATION

We can generate an unlimited amount of data from GAN's generator virtually. However, we generated the GAN-bot samples equal to the normal flow samples. The generated samples were augmented into the 70% train set only while 30% of the test set remains unchanged for each k-fold split where k = 10 in our case.

I. EXPERIMENTAL SETUP

The experiments were performed on a GPU workstation AMD Ryzen threadripper 1950x 16-core processor with three units of GeForce GTC 1070 Ti, running ubuntu 20.04.

Algorithm 3 Botshot Algorithm

Input: $\mathcal{T}$ (preprocessed train set in csv format), GAN_{type}, batch_size, botnet_classifier
Output: $\mathcal{A}$ (average accuracy), $\mathcal{R}$ (average recall), $\mathcal{P}$ (average precision), $\mathcal{F}$ (average F1)

begin
  $X_b \sim \mathcal{T}$; // Extract real_bots from the data set $\mathcal{T}$
  Create $\mathcal{D}$, $\mathcal{G}$ and $\mathcal{C}$ models; // Define and compile discriminator, generator and combined models for the GAN
  if $GAN_{type} == CGAN$ **then**
    $\mathcal{X}_b[Label] \leftarrow$ K-Means labels ; // Replace the Label column with k-means class labels
  for $i = 1, 2, 3, \ldots,$ *epochs* **do**
    /* Adversarial Training GAN */
    for $j = 1, 2, 3, \ldots,$ *batches* **do**
      $x_i \sim \mathcal{X}_b$; // Extract botnet data of batch_size
      $z_j \leftarrow \mathcal{N}_{\{mean=0,std=1,size=batch\_size\}}$; // Extract noise from normal distribution of batch_size
      $g_{z_j} \leftarrow \mathbb{E}_{z_i \sim p(z_j)}$; // Probability Estimation from $\mathcal{G}$ using z_j
      $\theta_{\mathcal{D}j} \leftarrow \theta_{\mathcal{D}j} - \eta \nabla \theta_{\mathcal{D}j} \mathcal{L}(x_j)$; // Train $\mathcal{D}$ on x_j with Adam optimizer
      $\theta_{\mathcal{D}j} \leftarrow \theta_{\mathcal{D}j} - \eta \nabla \theta_{\mathcal{D}j} \mathcal{L}(g_{z_j})$; // Train $\mathcal{D}$ on g_{z_j} with Adam optimizer
      $\theta_{\mathcal{G}j} \leftarrow \theta_{\mathcal{G}j} - \eta \nabla \theta_{\mathcal{G}j} \mathcal{L}(z_j)$; // Train $\mathcal{C}$ (combined model) on z_j with Adam optimizer
    $\theta_{\mathcal{G}i} \leftarrow \theta_{\mathcal{G}j}$; // Save the weights of $\mathcal{C}$ after every epoch
    $z_i \leftarrow \mathcal{N}_{\{mean=0,std=1,size=sizeof(\mathcal{X}_b)\}}$; // Extract noise from normal distribution with botnet data size
    $G_{z_i} \leftarrow \mathbb{E}_{z_i \sim p(z)}$; // Probability Estimation from $\mathcal{G}$ using z_i
    /* Evaluation of Generated Data (g_z) */
    U $= \mathcal{X}_b \cup G_{z_i}$
    for $k = 1, 2, 3, \ldots, 10$ **do**
      split_pointer = k
      $t_r \leftarrow$ 70% of U ; // Create a train set from 70% of examples in unified set
      $t_s \leftarrow$ 30% of U ; // Create a test set from 30% of examples in unified set
      train botnet_classifier on t_r
      test botnet_classifier on t_s
      compute $\mathcal{A}_i, \mathcal{R}_i$; // Compute accuracy and recall using confusion matrix
    $L_{\mathcal{A}} \leftarrow$compute average $\mathcal{A}$; // Compute average accuracy and save into a list
    $L_{\mathcal{R}} \leftarrow$compute average $\mathcal{R}$; // Compute average recall and save into a list
  $z \leftarrow \mathcal{N}_{\{mean=0,std=1,size=Normal-real\_bots\}}$; // Extract noise from normal distribution of size = (Normal - real_bots)
  $Gz_{\mathcal{I}_{argmin}(\mathcal{A})} \leftarrow \mathbb{E}_{z \sim p(z)}$; // Probability Estimation from $\mathcal{G}$ using z_j
  $\mathcal{Y}_{C2ST} \leftarrow \mathcal{T} \cup Gz_{\mathcal{I}_{argmin}(\mathcal{A})}$; // Augment the original dataset $\mathcal{T}$ with GAN_bots generated based on the epoch of GAN training with minimum value of accuracy
  $z \leftarrow \mathcal{N}_{\{mean=0,std=1,size=Normal-real\_bots\}}$; // Extract noise from normal distribution of size = (Normal - real_bots)
  $Gz_{\mathcal{I}_{argmin}(\mathcal{R})} \leftarrow \mathbb{E}_{z \sim p(z)}$; // Probability Estimation from $\mathcal{G}$ using z_j
  $\mathcal{Y}_{Botshot} \leftarrow \mathcal{T} \cup Gz_{\mathcal{I}_{argmin}(\mathcal{R})}$; // Augment the original dataset $\mathcal{T}$ with GAN_bots generated based on the epoch of GAN training with minimum value of recall
  for $K = 1, 2, 3, \ldots, 10$ **do**
    split_pointer = K
    $T_r \leftarrow$ 70% of $\mathcal{Y}_{Botshot}$; // Create a train set from 70% of examples in unified set
    $T_s \leftarrow$ 30% of $\mathcal{Y}_{Botshot}$; // Create a test set from 30% of examples in unified set
    train botnet_classifier on T_r test botnet_classifier on T_s compute $\mathcal{A}_K, \mathcal{R}_K, \mathcal{P}_K$ and $\mathcal{F}_K$; // Compute accuracy, recall, precision and F1 score using confusion matrix
  compute average $\mathcal{A}, \mathcal{R}, \mathcal{P}$ and $\mathcal{F}$; // Compute average accuracy, recall, precision and F1 score

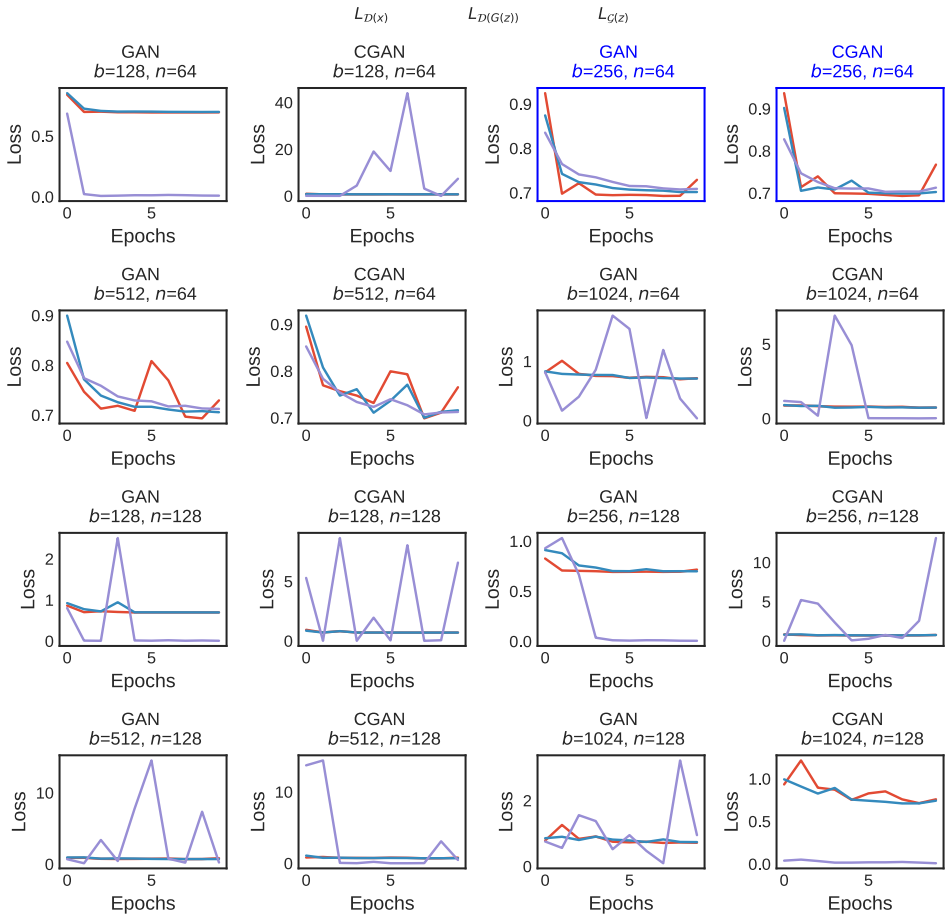


FIGURE 3. Hyperparameters exploration for batch size (*b*) and hidden layer neurons count multiple (*n*).

The platform used was jupyter notebook with libraries mainly keras, tensorflow, sklearn and numpy.

VI. EXPERIMENTAL RESULTS

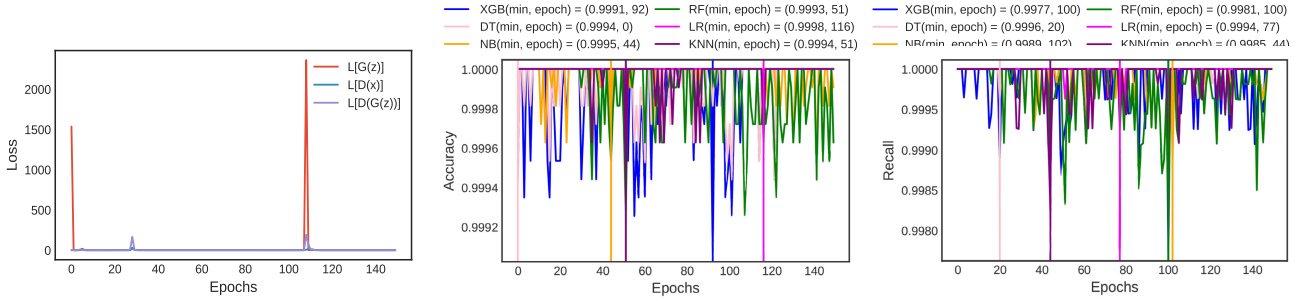
The results are demonstrated in Table 4 for the performance evaluation of six classifiers after the augmentation of data generated from different data generation techniques. The case of no augmentation was used as the benchmark to compare with the oversampling based techniques. For making a performance comparison, we employed three variants of the best performing synthetic minority oversampling techniques (SMOTEs) among 85 as suggested by the author in [54]. The results of these oversamplers have been summarized in Table 4 along with the GAN and CGAN based C2ST and Botshot oversampling. The post augmentation results for the performance evaluation metrics used were accuracy, recall, precision and F1 score. Figures 10, 11 and 12 show the average performance of all the classifiers after augmenting the train set with GAN-bots in 10-fold train-test split for the ISCX-2014, CIC-IDS2017 and CIC-IDS2018 datasets and testing on the fixed 30% test set. The following sections will discuss the results in detail in terms of the performance metrics of the classifiers used:

A. ACCURACY

The accuracy values for both GAN and CGAN based oversampling are almost always greater than or equal to the no-augmentation values and in some cases more than SMOTEs. Specifically, the value of accuracy in case of ISCX-2014 for GAN based Botshot is higher than GAN based C2ST but equal in case of CIC-IDS2017 and CIC-IDS2018. As per equation 6, the accuracy is the ratio of the sum of TP and TN to the total number of samples (TP + TN + FP + FN) in the confusion matrix, so due to the large number of TN (benign samples), the improvement in post augmentation results may not be significantly more accurate. So the decisive metrics could be recall and precision.

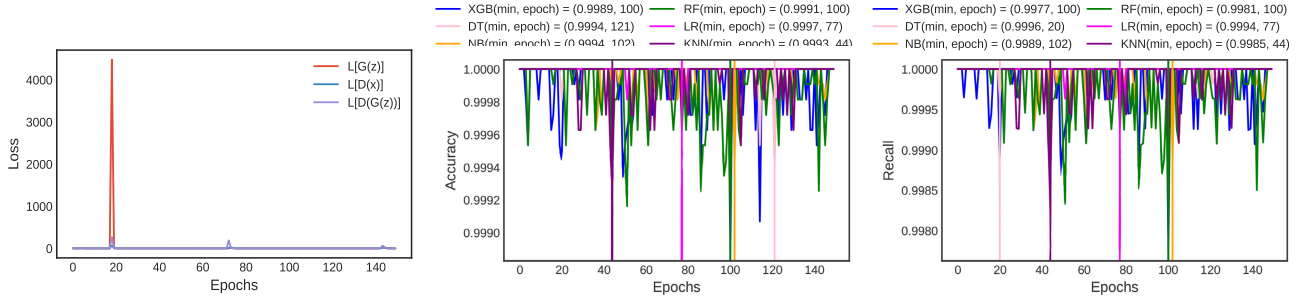
B. RECALL

The recall improvement is the highest value proposition of this research work because we need to mitigate the effects of adversarial evasions. However, the recall could not be improved significantly for all classifiers for the three datasets. In fact, for NB and KNN the recall deteriorated adversely especially in case of the ISCX-2014 dataset. This behaviour of the classifiers is specific to their inherent



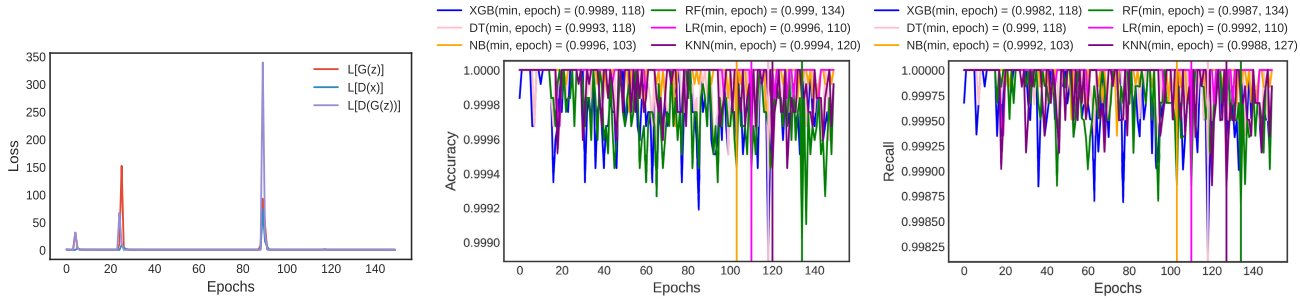
(a) Losses (b) C2ST (c) Botshot

FIGURE 4. GAN training & evaluation on ISCX-2104 dataset.



(a) Losses (b) C2ST (c) Botshot

FIGURE 5. CGAN training & evaluation on ISCX-2104 dataset.



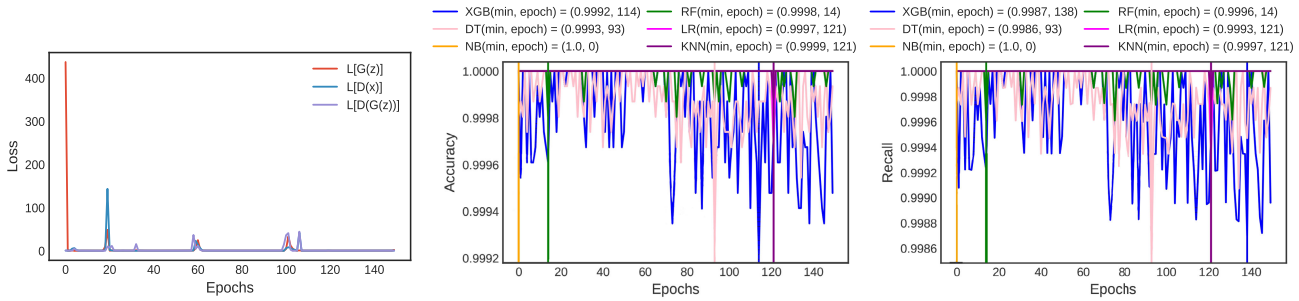
(a) Losses (b) C2ST (c) Botshot

FIGURE 6. GAN training & evaluation on CIC-IDS2017 dataset.

properties, the explanation of which is beyond the scope of this research. In the case of CIC-IDS2017, the recall was improved as compared to the benchmark for all the GAN and CGAN based oversampling methods. This proves the effectiveness of the Botshot as compared to C2ST as well. The values for recall in almost all cases for Botshot based GAN/CGAN oversampling is greater than or equal to C2ST based GAN/CGAN oversampling except in the case of CGAN(Botshot) with recall equal to 98.65% as compared to 98.61% of CGAN(C2ST) for CIC-IDS2017 dataset. It can be confidently inferred that Botshot is competitive not only with the SMOTE based oversamplers in case of CIC-IDS2017 and CIC-IDS2018 datasets but in certain cases better for particular classifiers like XGB which almost always shows improvement in recall for almost all the oversampling techniques as compared to the benchmark. This information can greatly help us choose the right classifier to be used in Botshot in future works.

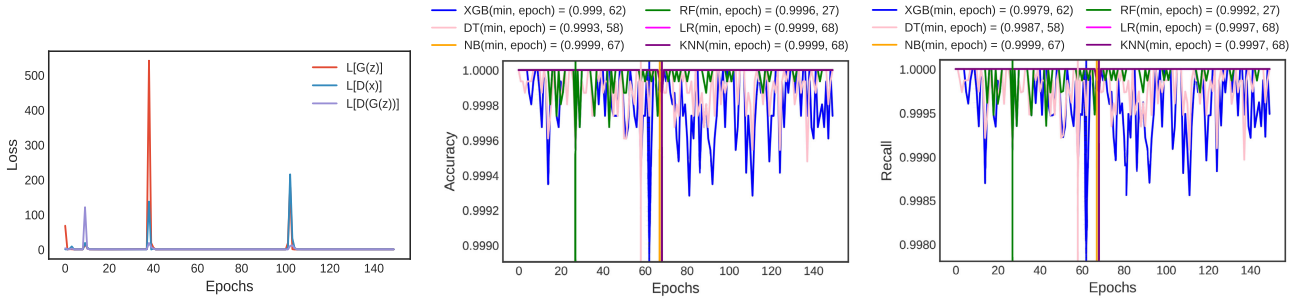
C. PRECISION

The minimization of false positive is specified as the property of a classifier being precise in its classification decisions. The precision values for almost all the classifiers have significantly higher values as compared to the SMOTE based



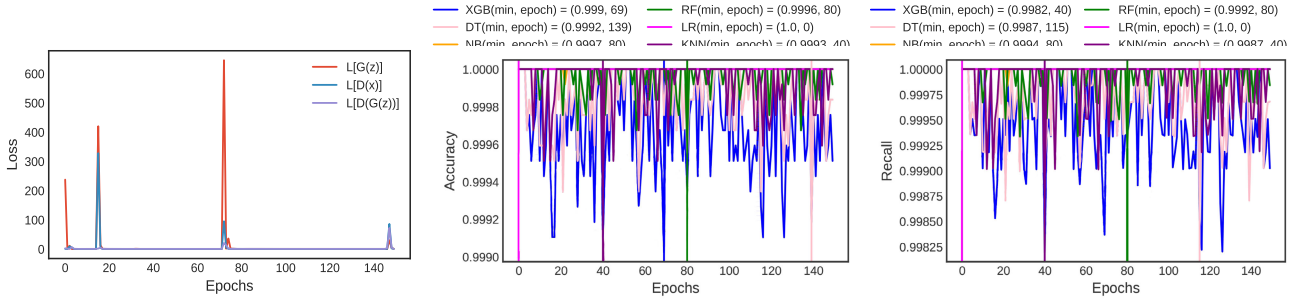
(a) Losses (b) C2ST (c) Botshot

FIGURE 7. CGAN training & evaluation on CIC-IDS2017 dataset.



(a) Losses (b) C2ST (c) Botshot

FIGURE 8. GAN training & evaluation on CIC-IDS2018 dataset.



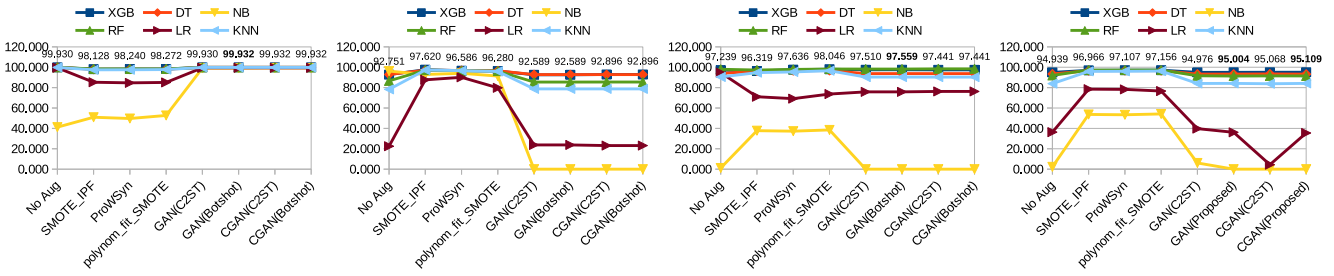
(a) Losses (b) C2ST (c) Botshot

FIGURE 9. CGAN training & evaluation on CIC-IDS2018 dataset.

techniques, which gives us strong confidence in estimating Botshot for reducing the FP except for NB which has the worst performance in case of all the three datasets. However, the precision value for GAN(Botshot) is higher than GAN(C2ST) and equal in case of CGAN for ISCX-2014 dataset. Similarly, the value for CGAN(Botshot) is higher than CGAN(C2ST) but lower than GAN(C2ST) for GAN(Botshot) for CIC-IDS2017 dataset. There is also one higher precision vote that goes to GAN(Botshot) as compared to GAN(C2ST) while the values remain the same for CGAN. The improvement in precision was unexpected since our main objective was to reduce FN and not the FP. GAN based oversampling have helped the classifiers make better decisions about the normal traffic samples, which is an added benefit of the proposed technique.

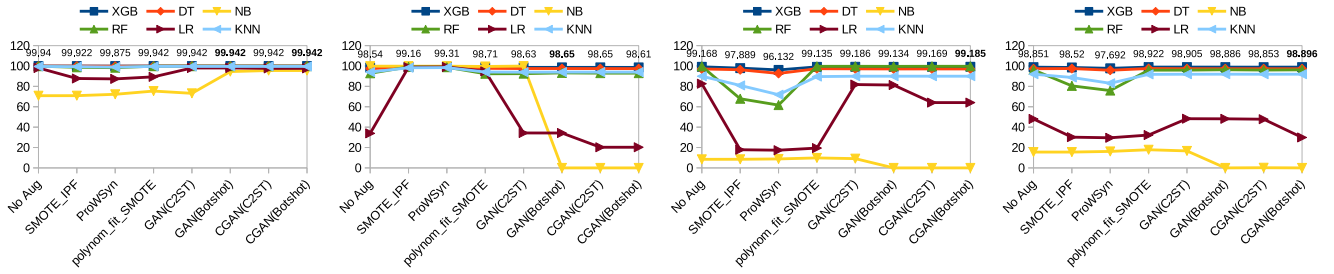
D. F1

The F1 score is the harmonic mean of the precision and recall so it reflects the combined effect of both these parameters. It can be observed from the results that F1 in case of CIC-IDS2017 and CIC-IDS2018 is better for Botshot in the majority of cases as compared to the C2ST based GAN/CGAN oversampling and even better than the benchmark in certain cases. This result also validates the effectiveness of the proposed technique.



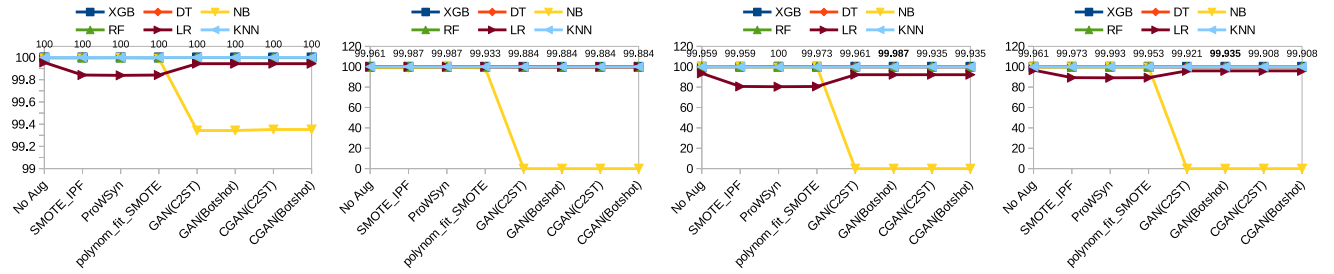
(a) Accuracy (b) Recall (c) Precision (d) F1

FIGURE 10. Results on ISCX-2014 dataset.



(a) Accuracy (b) Recall (c) Precision (d) F1

FIGURE 11. Results on CIC-IDS2017 dataset.



(a) Accuracy (b) Recall (c) Precision (d) F1

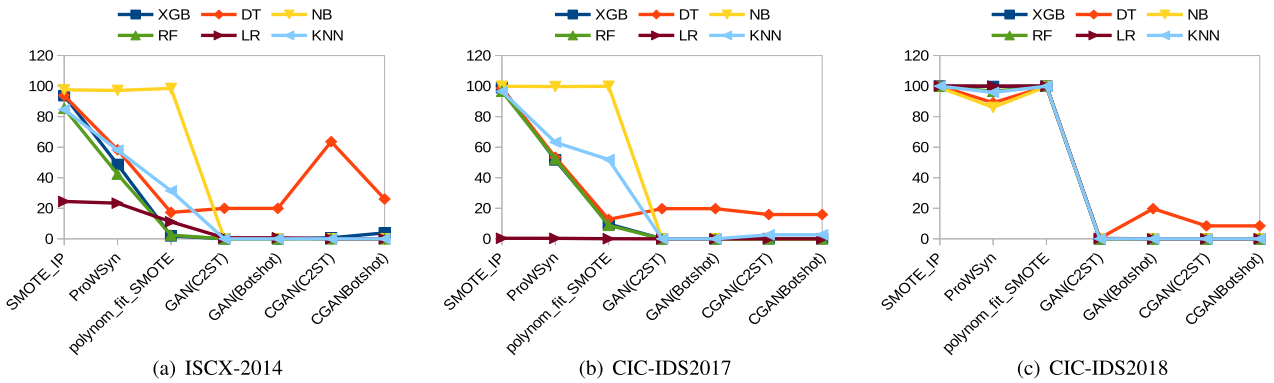
FIGURE 12. Results on CIC-IDS2018 dataset.

VII. DISCUSSION

With the help of the extensive and interesting results, we can empirically estimate the effectiveness of the proposed technique. Although the existing works show the effectiveness of GANs, we have proposed one step further by showing that GANs can be used effectively for security hardening of botnet detectors. In the light of the results, the discussion can be summarised in subsections as follows:

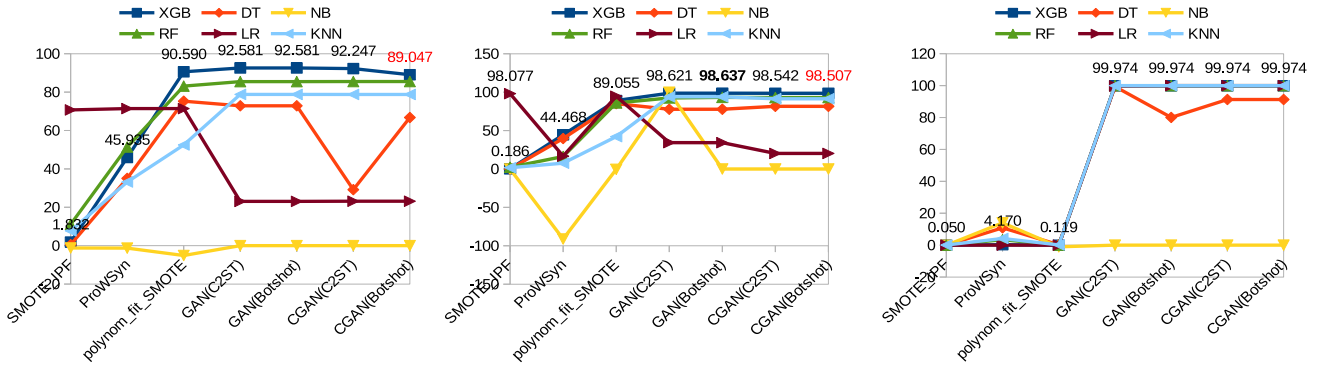
A. RELATIONSHIP OF GAN LOSS WITH C2ST AND BOTSHOT

The minimum values of accuracy and recall for each classifier during the GAN training are depicted in Figures 4, 5, 6, 7, 8 and 9 for all the three datasets. The lines that touch the base (x-axis) represent the minimum values of the accuracy or recall for the particular classifier on a specific epoch. There are two points worth noticing in these diagrams. First, it is not necessary that with each training epoch, the accuracy or the recall decreases gradually. In fact, the values are stochastic and unpredictable in each epoch for every dataset for both GANs. This result shows that the GAN loss only is not a strong metric for evaluating the $\mathcal{G}$ performance. Although, as mentioned in Subsection V-G, the losses converge and remain close to zero; the adopted evaluation, using C2ST and Botshot, behaves differently. Second, it is not necessary that there will always be an epoch that will have both accuracy and recall minimum at the same time. In fact in some cases the epochs for both



(a) ISCX-2014 (b) CIC-IDS2017 (c) CIC-IDS2018

FIGURE 13. Recall score (y-axis) after blackbox test for each oversampling technique for all the three datasets.



(a) ISCX-2014 (b) CIC-IDS2017 (c) CIC-IDS2018

FIGURE 14. Improvement in recall score (y-axis) after adversarial training for all the three datasets.

these metrics are different as can be seen in case of XGB, DT and RF in Figure 4, DT in Figure 5, KNN in Figure 6, XGB and DT in Figure 7 and XGB in Figure 9. Moreover, in only one case were accuracy and recall both minimum in the same epoch for all the classifiers in Figure 8. This reinforces our notion that recall could be a more effective metric for assessing the similarity between the generated and real samples.

B. COMPARISON OF BOTSHOT WITH PEER TECHNIQUES

1) OVERSAMPLING

While comparing GAN based oversampling with SMOTEs, a question may arise whether it is better than its interpolation-based counterpart. The answer can be a definite 'yes' if we can explore a GAN model with appropriate hyperparameters that can generate more realistic samples as compared to what has been proposed in the current work. The accuracy and recall both have values during GAN evaluation no less than 99% (ideal case is 50%), which means that our GANs are generating few realistic samples of botnet traffic. However, a significant improvement in the final results can be observed in Table 4. Since the performance of the SMOTE oversamplers is better in many cases, we are not in a position to definitely say 'yes' to the question. For that reason, we leave the research of exploring a suitable GAN for this purpose to future work. For example, the ISCX-2014 dataset could not give us good results in recall for the Botshot so we can start with a set of an appropriate classifier, a GAN and this dataset to improve the detection performance by making necessary exploration of a suitable GAN model.

2) BLACKBOX TEST AND IMPROVEMENTS IN RECALL

The blackbox test could be used to evaluate the aptness of the generated botnet data to be fooled as benign by the classifiers. The trained classifiers were tested with generated samples added in test set in each 70-30 split. The results for the three datasets for recall are illustrated in Figure 13. If we compare the GAN based oversampling with the other synthetic techniques for blackbox testing, the results are significantly remarkable. The recall score drops excessively for GAN based techniques inferring that the GAN generated samples can evade the classifiers more than peer oversampling techniques do. Figure 14 shows the improvement in the value of recall after adversarial training of generated data for each oversampling technique. It can be observed that in case of the best performing classifier (XGB), the improvement is 98.58%, 98.64% and 99.97% for Botshot being the highest among the peer techniques. The improvement in recall as compared to the C2ST was lower for ISCX-2014 and CIC-IDS2017 (values marked in red in Figure 14)

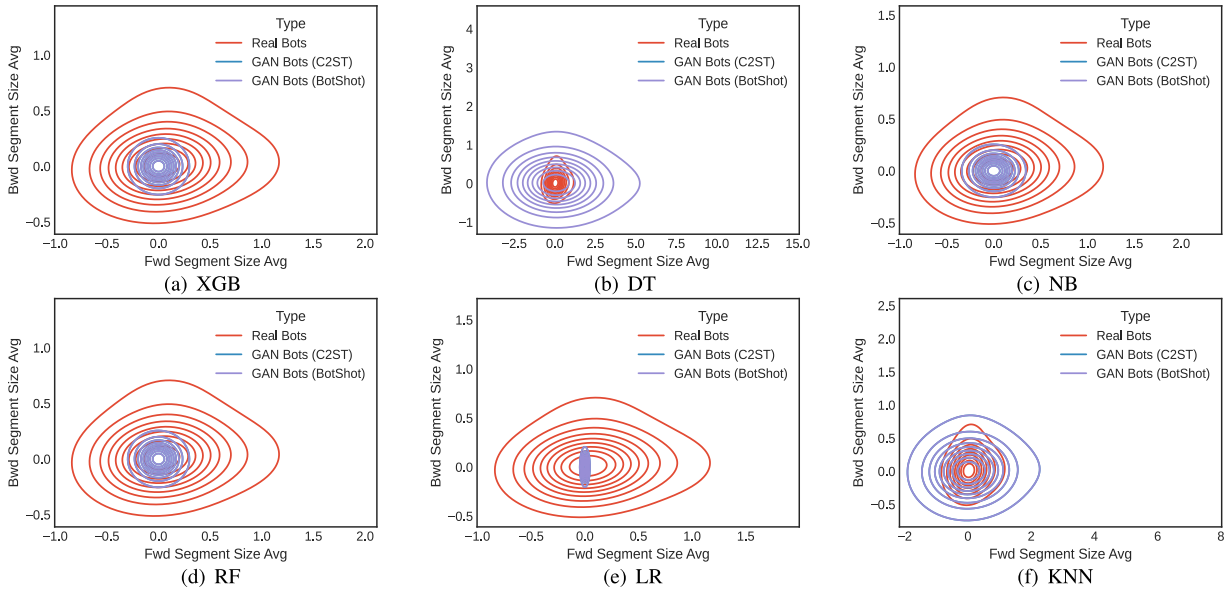


FIGURE 15. Contour plot of two random features from the ISCX-2014 dataset: Both GAN Bots (C2ST) and GAN Bots (Botshot) try to follow the Real Bots distribution.

for CGAN(Botshot). This opens further research horizons towards exploring more GANs to validate Botshot which is a further extension of this work that we leave as a future work.

C. LIMITATION OF BOTSHOT

In Figure 15, the GAN mimicry of the two distinct features from the ISCX-2014 dataset for all the six classifiers has been demonstrated in the form of contour plots. The graphs show that the GAN Bots generated by both C2ST and Botshot are trying to follow the distribution of the Real Bots for the two features 'Fwd Segment Size Avg' and 'Bwd Segment Size Avg'. We chose these two features based on the fact that in most of the training iterations, their distribution is unchanged. This implies that generator is good at generating these two features after a certain number of initial training iterations. The same principal is valid for the CIC-2017 and CIC-2018 datasets so we did not include their results. However, it is imperative to know if the generated samples are as valid as real-life traffic samples since the GANs may not generate all the valid network traffic samples. We do not perform encoding/decoding of generated attack samples to relay on the internet to validate the semantics. Since, the purpose of this work is to devise a defence strategy against adversarial attacks, improving the detection performance with the addition of data generated by GAN in ML domain (i.e. training the detectors in post-processing of network traffic data) is adequate. Hence we are generating realistic samples to improve detection rather than generating real traffic samples to fail botnet detectors. We leave this as future work to filter valid samples out of the generated data from GANs.

VIII. CONCLUSION AND FUTURE DIRECTIONS

GANs have proved to be very effective in computer vision-based applications. However, we can infer from this work that GANs are suitable candidates to address multiple challenges within the cybersecurity domain as well. Especially, the effects of adversarial evasion attacks can be mitigated proactively using GANs' generated traffic augmentation to the original train sets. The general method to utilize GANs to generate the quality samples is based on the loss functions of generator and discriminator networks. However, the quality of generated data could be further evaluated using the classifiers under test. We have proposed a technique to generate improved quality samples to fulfil this objective in case of botnet detection. The results show that GANs can provide a better alternative to the traditional traffic generation methods for all the classifiers used. They can also be used to balance the datasets and further enhance the security hardening of the botnet detectors, especially against adversarial evasion attacks as well as decreasing the false positives.

The behaviours and landscapes of modern botnets need to be explored further. This means that new traffic features must be introduced to differentiate botnets from normal traffic. Modern GANs could be harnessed to further enhance better quality adversarial examples. The validity of GANs' generated traffic in terms of semantics could be another extension of this research work. A further envisaged research direction could be making the IDS autonomous to be proactively trained against novel evasion samples using deep reinforcement learning.

REFERENCES

[1] N. Goodman, "A survey of advances in botnet technologies," 2017, *arXiv:1702.01132*. [Online]. Available: http://arxiv.org/abs/1702.01132
[2] R. H. Randhawa, A. Hameed, and A. N. Mian, "Energy efficient cross-layer approach for object security of CoAP for IoT devices," *Ad Hoc Netw.*, vol. 92, Sep. 2019, Art. no. 101761.
[3] M. Alauthman, N. Aslam, M. Al-Kasassbeh, S. Khan, A. Al-Qerem, and K.-K. R. Choo, "An efficient reinforcement learning-based botnet detection approach," *J. Netw. Comput. Appl.*, vol. 150, Jan. 2020, Art. no. 102479.

[4] N. Papernot, P. McDaniel, I. Goodfellow, S. Jha, Z. B. Celik, and A. Swami, "Practical black-box attacks against machine learning," in *Proc. ACM Asia Conf. Comput. Commun. Secur.*, Apr. 2017, pp. 506–519.

[5] R. K. Sharma, H. K. Kalita, and B. Issac, "Are machine learning based intrusion detection system always secure? An insight into tampered learning," *J. Intell. Fuzzy Syst.*, vol. 35, no. 3, pp. 3635–3651, 2018.

[6] C. Yin, Y. Zhu, S. Liu, J. Fei, and H. Zhang, "An enhancing framework for botnet detection using generative adversarial networks," in *Proc. Int. Conf. Artif. Intell. Big Data (ICAIBD)*, May 2018, pp. 228–234.

[7] P. Samangouei, M. Kabkab, and R. Chellappa, "Defense-GAN: Protecting classifiers against adversarial attacks using generative models," 2018, *arXiv:1805.06605*. [Online]. Available: http://arxiv.org/abs/1805.06605

[8] M. Ring, D. Schlör, D. Landes, and A. Hotho, "Flow-based network traffic generation using generative adversarial networks," *Comput. Secur.*, vol. 82, pp. 156–172, May 2019.

[9] K. R. Mopuri, U. Ojha, U. Garg, and R. V. Babu, "NAG: Network for adversary generation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 742–751.

[10] J. Engelmann and S. Lessmann, "Conditional Wasserstein GAN-based oversampling of tabular data for imbalanced learning," 2020, *arXiv:2008.09202*. [Online]. Available: http://arxiv.org/abs/2008.09202

[11] M. Usama, M. Asim, S. Latif, J. Qadir, and Ala-Al-Fuqaha, "Generative adversarial networks for launching and thwarting adversarial attacks on network intrusion detection systems," in *Proc. 15th Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Jun. 2019, pp. 78–83.

[12] A. Antoniou, A. Storkey, and H. Edwards, "Data augmentation generative adversarial networks," 2017, *arXiv:1711.04340*. [Online]. Available: http://arxiv.org/abs/1711.04340

[13] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *J. Artif. Intell. Res.*, vol. 16, pp. 321–357, Jun. 2002.

[14] H. Ba, "Improving detection of credit card fraudulent transactions using generative adversarial networks," 2019, *arXiv:1907.03355*. [Online]. Available: http://arxiv.org/abs/1907.03355

[15] C. Bentéjac, A. Csörgő, and G. Martínez-Muñoz, "A comparative analysis of gradient boosting algorithms," *Artif. Intell. Rev.*, vol. 54, no. 3, pp. 1937–1967, 2020.

[16] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. ICISSP*, 2018, pp. 108–116.

[17] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "A brief survey of deep reinforcement learning," 2017, *arXiv:1708.05866*. [Online]. Available: http://arxiv.org/abs/1708.05866

[18] Y. Li, "Deep reinforcement learning: An overview," 2017, *arXiv:1701.07274*. [Online]. Available: http://arxiv.org/abs/1701.07274

[19] A. Kurakin *et al.*, "Adversarial attacks and defences competition," 2018, *arXiv:1804.00097*. [Online]. Available: http://arxiv.org/abs/1804.00097

[20] Y. Liu, X. Chen, C. Liu, and D. Song, "Delving into transferable adversarial examples and black-box attacks," 2016, *arXiv:1611.02770*. [Online]. Available: http://arxiv.org/abs/1611.02770

[21] N. Papernot, P. McDaniel, and I. Goodfellow, "Transferability in machine learning: From phenomena to black-box attacks using adversarial samples," 2016, *arXiv:1605.07277*. [Online]. Available: http://arxiv.org/abs/1605.07277

[22] L. Huang, A. D. Joseph, B. Nelson, B. I. Rubinstein, and J. Tygar, "Adversarial machine learning," in *Proc. 4th ACM Workshop Secur. Artif. Intell.*, 2011, pp. 43–58.

[23] A. Kurakin, I. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," 2016, *arXiv:1611.01236*. [Online]. Available: http://arxiv.org/abs/1611.01236

[24] S. Zhang, X. Xie, and Y. Xu, "A brute-force black-box method to attack machine learning-based systems in cybersecurity," *IEEE Access*, vol. 8, pp. 128250–128263, 2020.

[25] B. Biggio and F. Roli, "Wild patterns: Ten years after the rise of adversarial machine learning," *Pattern Recognit.*, vol. 84, pp. 317–331, Dec. 2018.

[26] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[27] D. Wu, B. Fang, J. Wang, Q. Liu, and X. Cui, "Evading machine learning botnet detection models via deep reinforcement learning," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019, pp. 1–6.

[28] G. Apruzzese, M. Andreolini, M. Marchetti, A. Venturi, and M. Colajanni, "Deep reinforcement adversarial learning against botnet evasion attacks," *IEEE Trans. Netw. Service Manage.*, vol. 17, no. 4, pp. 1975–1987, Dec. 2020.

[29] A. Venturi, G. Apruzzese, M. Andreolini, M. Colajanni, and M. Marchetti, "DReLAB–deep REinforcement learning adversarial botnet: A benchmark dataset for adversarial attacks against botnet intrusion detection systems," *Data Brief*, vol. 34, Feb. 2021, Art. no. 106631.

[30] G. Apruzzese and M. Colajanni, "Evading botnet detectors based on flows and random forest with adversarial samples," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–8.

[31] N. Martins, J. M. Cruz, T. Cruz, and P. Henriques Abreu, "Adversarial machine learning applied to intrusion and malware scenarios: A systematic review," *IEEE Access*, vol. 8, pp. 35403–35419, 2020.

[32] F. Tramèr, A. Kurakin, N. Papernot, I. Goodfellow, D. Boneh, and P. McDaniel, "Ensemble adversarial training: Attacks and defenses," 2017, *arXiv:1705.07204*. [Online]. Available: http://arxiv.org/abs/1705.07204

[33] H. S. Anderson, J. Woodbridge, and B. Filar, "DeepDGA: Adversarially-tuned domain generation and detection," in *Proc. ACM Workshop Artif. Intell. Secur.*, Oct. 2016, pp. 13–21.

[34] G. Vormayr, T. Zseby, and J. Fabini, "Botnet communication patterns," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2768–2796, Sep. 2017.

[35] Y. Kilcher and T. Hofmann, "The best defense is a good offense: Countering black box attacks by predicting slightly wrong labels," 2017, *arXiv:1711.05475*. [Online]. Available: http://arxiv.org/abs/1711.05475

[36] D. Zhao, I. Traore, B. Sayed, W. Lu, S. Saad, A. Ghorbani, and D. Garant, "Botnet detection based on traffic behavior analysis and flow intervals," *Comput. Secur.*, vol. 39, pp. 2–16, Nov. 2013.

[37] A. Shiravi, H. Shiravi, M. Tavallaee, and A. A. Ghorbani, "Toward developing a systematic approach to generate benchmark datasets for intrusion detection," *Comput. Secur.*, vol. 31, no. 3, pp. 357–374, May 2012.

[38] S. Saad, I. Traore, A. Ghorbani, B. Sayed, D. Zhao, W. Lu, J. Felix, and P. Hakimian, "Detecting P2P botnets through network behavior analysis and machine learning," in *Proc. 9th Annu. Int. Conf. Privacy, Secur. Trust*, Jul. 2011, pp. 174–180.

[39] A. H. Lashkari, G. D. Gil, M. S. I. Mamun, and A. A. Ghorbani, "Characterization of tor traffic using time based features," in *Proc. 3rd Int. Conf. Inf. Syst. Secur. Privacy*, 2017, pp. 253–262.

[40] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy of network threats and the effect of current datasets on intrusion detection systems," 2018, *arXiv:1806.03517*. [Online]. Available: http://arxiv.org/abs/1806.03517

[41] P. A. A. Resende and A. C. Drummond, "A survey of random forest based methods for intrusion detection systems," *ACM Comput. Surv.*, vol. 51, no. 3, p. 48, 2018.

[42] A. Nisioti, A. Mylonas, P. D. Yoo, and V. Katos, "From intrusion detection to attacker attribution: A comprehensive survey of unsupervised methods," *IEEE Commun. Surveys Tuts.*, vol. 20, no. 4, pp. 3369–3388, Jul. 2018.

[43] D. Santana, S. Suthaharan, and S. Mohanty, "What we learn from learning–understanding capabilities and limitations of machine learning in botnet attacks," 2018, *arXiv:1805.01333*. [Online]. Available: http://arxiv.org/abs/1805.01333

[44] A. Azab, M. Alazab, and M. Aiash, "Machine learning based botnet identification traffic," in *Proc. IEEE Trustcom/BigDataSE/ISPA*, Aug. 2016, pp. 1788–1794.

[45] A. Bansal and S. Mahapatra, "A comparative analysis of machine learning techniques for botnet detection," in *Proc. 10th Int. Conf. Secur. Inf. Netw. (SIN)*, New York, NY, USA, 2017, pp. 91–98. [Online]. Available: http://doi.acm.org/10.1145/3136825.3136874

[46] J. O. Nehinbe, "A critical evaluation of datasets for investigating IDSs and IPSs researches," in *Proc. IEEE 10th Int. Conf. Cybern. Intell. Syst. (CIS)*, Sep. 2011, pp. 92–97.

[47] N. Moustafa, J. Hu, and J. Slay, "A holistic review of network anomaly detection systems: A comprehensive survey," *J. Netw. Comput. Appl.*, vol. 128, pp. 33–55, Feb. 2019.

[48] G. Creech and J. Hu, "Generation of a new IDS test dataset: Time to retire the KDD collection," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2013, pp. 4487–4492.

[49] J. Uramová, P. Scgeč, M. Moravčík, J. Papán, M. Kontšek, and J. Hrabovskỳ, "Infrastructure for generating new IDS dataset," in *Proc. 16th Int. Conf. Emerg. eLearn. Technol. Appl. (ICETA)*, Nov. 2018, pp. 603–610.

[50] X. Yuan, P. He, Q. Zhu, and X. Li, "Adversarial examples: Attacks and defenses for deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 30, no. 9, pp. 2805–2824, Sep. 2019.
[51] E. B. Beigi, H. H. Jazi, N. Stakhanova, and A. A. Ghorbani, "Towards effective feature selection in machine learning-based botnet detection approaches," in *Proc. IEEE Conf. Commun. Netw. Secur.*, Oct. 2014, pp. 247–255.
[52] S. García. *Malware Capture Facility Project*. Accessed: Apr. 10, 2021. [Online]. Available: https://mcfp.weebly.com/
[53] M. Lucic, K. Kurach, M. Michalski, S. Gelly, and O. Bousquet, "Are GANs created equal? A large-scale study," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 700–709.
[54] G. Kovács, "An empirical comparison and evaluation of minority oversampling techniques on a large number of imbalanced datasets," *Appl. Soft Comput.*, vol. 83, Oct. 2019, Art. no. 105662.

RIZWAN HAMID RANDHAWA received the B.S. degree in electronic engineering from International Islamic University Islamabad, Pakistan, and the master's degree in computer science from Information Technology University, Lahore, Pakistan. He is currently pursuing the Ph.D. degree in computer science from Northumbria University, Newcastle upon Tyne, U.K. He has vast experience with embedded systems in different private and public sector organizations of Pakistan. His research interests include AI-based botnet detection, the IoT security, and embedded systems design and development for the IoT platforms.

NAUMAN ASLAM (Member, IEEE) received the Ph.D. degree in engineering mathematics from Dalhousie University, Canada, in 2008. He is currently a Professor with the Department of Computer and Information Science, Northumbria University, U.K. Before joining Northumbria University, as a Senior Lecturer, in 2011, he worked as an Assistant Professor with Dalhousie University. He is also leading the Network Systems and Security Research Group, Northumbria University. He has published over 100 articles in peer-reviewed journals and conferences. His research interest includes communication networks. His current research interests include addressing problems related to wireless body area networks and the IoT, network security, QoS-aware communication in industrial wireless sensor networks, and application of artificial intelligence (AI) in communication networks.

MOHAMMAD ALAUTHMAN received the B.Sc. degree in computer science from Hashemite University, Jordan, in 2002, the M.Sc. degree in computer science from Amman Arab University, Jordan, in 2004, and the Ph.D. degree from Northumbria University Newcastle, U.K., in 2016. He is currently an Assistant Professor with the Information Security Department, Petra University, Jordan. His research interests include cyber-security, cyber forensics, advanced machine learning, and data science applications.

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 is currently pursuing the Ph.D. degree from Northumbria University, Newcastle upon Tyne, U.K. From 2015 to 2018, he was a Junior Lecturer with the Capital University of Science and Technology. His research interests include information security and forensics, machine learning, and malware analysis.

FRANK COMEAU (Member, IEEE) received the Ph.D. degree in engineering mathematics from Dalhousie University, Canada, in 2008. He also holds the position of an Adjunct Professor with Dalhousie University. He is currently an Associate Professor and the Chair of the Department of Engineering, St. Francis Xavier University, Canada. His research interests include wireless sensor and *ad hoc* networks, communication protocols, and underwater sensor networks. He has published 11 articles in peer-reviewed journals and conferences. He is a member of Engineers Nova Scotia (Canada).