



GIDS: Anomaly Detection Using Generative Adversarial Networks

Rinoy Macwan¹, Sankha Das², and Manik Lal Das¹(✉)

¹ DA-IICT, Gandhinagar, India

rinoymacwan@gmail.com, maniklal@gmail.com

² BITS Pilani, Rajasthan, India

umasankhadas@gmail.com

Abstract. Cyber security in modern digital age has become a major challenge before an individual/organization to protect assets from malicious entities. Machine learning technique has been used in advanced intrusion detection system (IDS) which detects new attacks by analyzing existing attacks' metrics with the help of rich collection of datasets. Generative Adversarial Networks (GAN) has found lots of attentions in recent times, in particular, for forgery detection in image data. GAN also shows its potential that can be used in text-based traffic inspection to check whether the traffic contains any suspicious strings. In this paper, we present an intrusion detection system using GAN, termed as GIDS, that detects anomalies in input strings with a reasonable accuracy. GIDS minimizes the mapping error without using an external encoder. The analysis and experimental results show that GIDS detects anomalies with an accuracy of 83.66%, while keeping false positive rate low.

Keywords: Network security · Intrusion detection · Generative adversarial network · Anomaly detection

1 Introduction

In modern digital age, Information and Communication Technology enables household objects to reach out to powerful cloud server and vice-versa. With this Internet of Things scenario, security of data as well as communicating entities is of prime concern. At the same time, the intellectual game between application developer and attacker has also been evolved in multi-fold defense-attack layers. Intrusion Detection Systems (IDS) [1] have been widely deployed by organizations to protect their valuable assets as well as perimeter from the defense-attack game.

An intrusion is defined as an activity that poses a threat to system and network resources. Intrusion detection is the process of monitoring and identifying an attempt that may bypass or break systems' defenses. Based on the nature of monitoring, IDS is classified into two types - Host Intrusion Detection System (HIDS) and Network Intrusion Detection System (NIDS). HIDS analyzes an

individual system by inspecting networks and logging events. Typically, HIDS monitors the events on a system and checks the systems' logs and/or the events' logs for suspicious/malicious activity. NIDS analyzes network traffic that consists of network tap or port mirroring. A network tap is a device which captures the network traffic passing through it. Port Mirroring, on the other hand, makes a copy of the network traffic and sends it to another port for analysis. Generally, NIDS are placed at strategic points in the network to analyze network traffic where any plausible attack may happen in near term.

Broadly, an IDS works by the principle of signature/pattern-based behaviour detection or by anomaly-based detection. Signature-based IDS monitors the network traffic and match the packet characteristics against a database of known attack signatures. The drawback of using this approach is that if there is large list of signatures in the database, the matching time would be impractical for stopping an attack. Another drawback of signature-based IDS is the inability to detect new attacks, as the signature/pattern for such upcoming attack does not exist in the database. To overcome these limitations on signature-based IDS, anomaly-based IDS is commonly preferred, which can observe any abnormal behavior that deviates from the normal behavior. Anomaly-based IDS uses the network traffic to determine normal or benign traffic and then compares the incoming traffic with the baseline observation. If any anomaly or deviation from the normal behaviour exceeds a predefined threshold, the IDS raises an alarm, which not only stops the known attacks but also detects any new attacks. However, one main drawback of anomaly-based IDS is that the system hits a higher number of False Positives (FP).

In recent times, Generative Adversarial Networks [2] has shown significant potentials for discriminating anomalies from normal behaviour. Although GAN is mostly employed in image-based characteristics [3, 13], the model has shown potential in other input samples. In [3], an encoder-based adversarial training is proposed, which samples data into Gaussian distribution space and then a discriminator checks whether the input comes from normal latent space or it is an anomaly.

In this paper, we present a GAN-based intrusion detection system, termed as GIDS, which trains textual data on the normal training samples and identifies anomalies with a reasonable accuracy. The proposed GIDS minimizes the mapping error without using an external encoder. In our analysis and experiment, GIDS is trained with NSL-KDD dataset [8] to prevent the system from any bias and the results show that GIDS detects anomalies with an accuracy of 83.66%, while keeping false positive rate substantially low.

The paper is organized as follows: Sect. 2 discusses the related work. Section 3 presents the proposed intrusion detection system using GAN. Section 4 provides the analysis and experimental results of the proposed GIDS. We conclude the paper with Sect. 5.

2 Related Work

Network security has been studied extensively by numerous researchers in past decades. We put our focus primarily on machine learning based IDS. Sangkatsanee et al. [4] proposed a real-time IDS using decision trees, where several existing machine learning algorithms have been studied using decision tree, ripper rule and neural networks. The study in [4] showed that IDS powered decision tree outperforms in terms of detection accuracy. Principle component analysis is quite successful for dimensionality reduction, where the training phase reduces time because of decrease in the number of features. Heba et al. [5] used Support Vector Machine to create an IDS using principle component analysis. Garcia-Teodoro et al. [6] and the work in [14] have discussed multiple approaches useful for anomaly detection in networks. Schlegl et al. [3] proposed a system using GAN to detect disease markers using anomaly detection. In their work [3], a deep convolutional GAN is used to train the system on benign images.

In [11], the authors proposed anomaly detection for medical images using GAN. Ganomaly was proposed in [12], which uses the original idea of AnoGAN [3] by adding an additional encoder.

3 Background and the Proposed System

3.1 Generative Adversarial Networks

Generative Adversarial Networks (GAN) [2] is generative in nature as opposed to discriminative algorithm. Discriminative algorithms classify the input data, that is, the features of the input data instance, into a category of output to which the data may belong. For example, a discriminative algorithm could predict whether a packet is an attack from the details it can gather from the packet. A discriminative algorithm gives a probability of the data instance belonging to an output class. Therefore, discriminative algorithm maps features to labels. Generative models on the other hand perform the opposite of classification. Generative algorithm gives the probability of the features given that the data instance belongs to a specific class. Using the network analogy, it can be noted that the generative algorithm tries to answer the question that how likely are these features given that a string/packet is an attack, and provides a probability for the same. In general, discriminative model learns the boundary between output classes and generative model learns the data distribution of the classes. A generative adversarial network consists of two neural networks: a generator and a discriminator. The generator generates new data instances, whereas, the discriminator tries to tell apart the real data from the generated data. One can think of the generator as a counterfeiter and the discriminator as the cop, where the counterfeiter tries to produce real samples and fool the cop and the cop's job is to identify which samples are from real data and which are the counterfeits.

3.2 Anomaly Score

Once the model using GAN has learned the real data distribution, samples are generated which can fool the discriminator. After the adversarial training is done, the model can map noise to a realistic data instance. The anomaly score of a query x tells how likely it is to be an anomaly, and based on a threshold one can classify whether it as an anomaly. To find an anomaly score for data x , one has to find the point z in the latent space that corresponds to $G(z)$, such that x and $G(z)$ are similar. The extent of similarity depends on how close x is to the data that was used to train the network. Schlegl et al. [3] proposed AnoGAN, which uses a GAN to detect anomalies on image samples. The summary of the AnoGAN is captured below, as the posed GIDS goes by the similar principles. In AnoGAN, two losses - residual and discriminator - play important roles. The residual loss measures the dissimilarity between the query data and the generated data $G(x)$, defined as:

$$L_R(z_y) = \Sigma|x - G(z_y)| \quad (1)$$

Discriminator loss is the difference in the discriminators output when the query data and the generated data $G(z)$ are passed through the discriminator. However, instead of using the scalar output of the discriminator, a new loss function was proposed which uses an intermediate feature of the discriminator.

$$L_D(z_y) = \Sigma|f(x) - f(G(z_y))| \quad (2)$$

Here, $f(\cdot)$ is the output of an intermediate layer of the discriminator. Instead of using only the decision of the discriminator, whether or not the generated image fits the learned distribution, this new loss takes the information of the feature representation which is learned by the discriminator during the adversarial training. The overall loss as weighted sum of both components is defined as:

$$L(z_y) = (1 - \lambda).L_R(z_y) + \lambda.L_D(z_y) \quad (3)$$

Using the above equations the corresponding latent variable z is obtained by minimizing $L(z)$. The minimization is done by iterative methods. After a fixed number of iterations the anomaly score of the query data is computed.

In our proposed GIDS, we have used the similar principles of AnoGAN with the objective of anomalies detection on textual data. To detect anomalies, KDD CUP 99 dataset [7] is used, which consists of around 48,98,431 data instances each of which is a single connection [9]. In the proposed GIDS, NSL KDD dataset [8] is used. KDD'99 dataset contains 34 continuous features and 7 categorical features for a total of 41 features. Before using the 7 categorical features, they have to be one-hot encoded. What one hot encoding does is that it creates columns for all possible values for a feature and assigns '1' to the column that the particular instance belongs to and '0' to all other columns. Using this one-hot encoding technique, GIDS ends up with 126 features. The data is then normalized to

between -1 and 1 , so that they are at a comparable range. GIDS uses the following architectures for the generator and the discriminator for experimentation. The generator contains 4 dense layers. A dense layer is a fully connected neural network layer. The first, second, third and the fourth dense layers contain 256, 512, 1024 and 126 neurons, respectively. The relatively higher number of neurons in the generator help the generator learn more features about the dataset. Batch normalization is also used. Similar to the normalization done before providing input to the networks, the batch normalization normalizes the data, and is usually placed before the activation layers. Batch normalization reduces the amount by which the hidden unit values shift. Multiple dropout layers were also used, where the dropout method prevents over fitting. Dropout method drops out some random neurons while training which helps prevent over fitting. A dropout rate is a value between 0 and 1, which decides what percentage of neurons have to be dropped out on random.

An activation function is used in GIDS that introduces non-linearity to the network. For example, ReLu (Rectified Linear Unit) [10] is one of the most widely used activation functions in neural networks. In such model, the discriminator contains 3 dense layers. The first, second and third layer contain 126, 512 and 126 neurons, respectively. In addition, there is one more dense layer which is the output of the discriminator that contains just one neuron, which uses the sigmoid activation function and essentially outputs the probability of the data originating from the real dataset. One important aspect of training a GAN is the stopping point. There is no fixed formula to calculate it; however, a good point to stop GAN training is when both the generator and the discriminator have gotten extremely good at their tasks, especially the generator. When a generator creates data which is similar to the training data and when the discriminator has a hard time telling apart real and generated data, the GAN is said to be trained. However, it is difficult to tell, if a generated sample resembles a somewhat similar sample from the real dataset. To mitigate this, one can use the outputs of the discriminator. When the losses of both the generator and the discriminator have converged sufficiently, a few samples are generated from the generator and pass them through the discriminator. The discriminator outputs how confident the discriminator is in labelling the data real or generated. If we get a value of the range 0.4 to 0.6, we can assume that the discriminator is having a hard time placing a binary label on the data sample, this it, is a good indication that the generated sample has a close resemblance to the training data. Following parameters are considered in the training and testing phases:

```

batch size = 128
epochs = 7500
dim z = 100
lr = 0.0002
slope of LeakyRelu = 0.2
dropout rate for G = 0.5
dropout rate for D = 0.3

```

The batch size of training was set to 128 for 7500 epochs. The input noise dimension dim_z was set to 100. The learning rate of the GIDS was set to 0.0002. The dropout rate was set to 0.5 and 0.3 for the generator and the discriminator, respectively.

4 Analysis and Experimental Results

Figure 1 depicts various losses of the GIDS during the training phase, where ‘d_loss_real’ is the loss of the discriminator while being trained on real data samples from the dataset, ‘d_loss_fake’ is the loss of the discriminator while being trained on the generated data samples created by the generator, and ‘g_loss’ is the loss of the proposed GIDS. The generated samples are passed on as being real data and by using the output of the discriminator the generator updates its weights accordingly. As there is no specific stopping criteria for GAN, a good point to end training is when the network is in equilibrium, that is, when the discriminator is not able to distinguish efficiently between real data and generated data. We have experimented the proposed GIDS on an Intel i7 CPU with 6 cores and 12 threads with 16 GB of RAM. The training of the GIDS took 8 min. Anomaly score was calculated for all the entries. Anomaly score for benign data was calculated with a mean of 10.36 and anomaly score for attack data was calculated with mean of 23.18. Attack data generates a higher anomaly score than benign data.

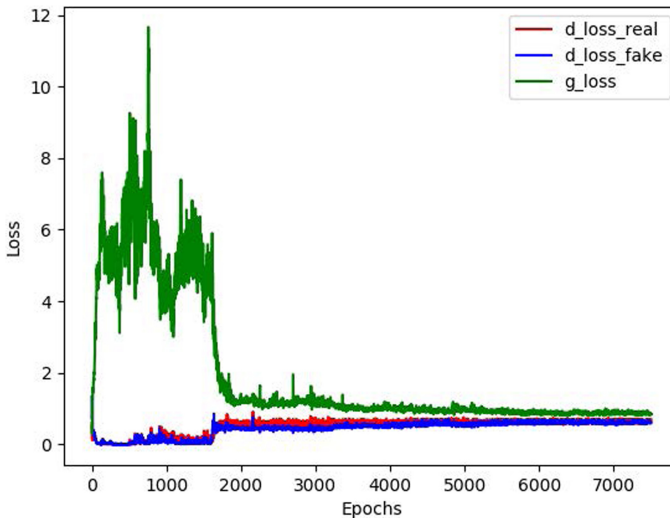


Fig. 1. Loss graph

Table 1. Confusion matrix

	Actual attack	Actual benign
Predicted attack	83.66	16.34
Predicted benign	7.67	92.33

Table 2. Comparisons of the models

	Normal	DoS attack	Probe
GAN based model	92.33	85.64	79.56
Decision Tree based model	99.43	99.17	98.73
Neural Network based model	98.60	96.37	95.72

A confusion matrix is a common way to represent the results of a classification model. There are two possible classes: attack and benign. Actual class of the data, that is, whether it actually is an attack or not. It is noted from the confusion matrix Table 1 that the GIDS model is able to detect 83.66% of all attacks. GIDS is able to classify 92.33% of the benign data correctly. As GIDS is anomaly-based, false positives observe 16.34% and false negatives stood at 7.67%. The following confusion matrix is obtained:

Setting a threshold to detect anomalies depends on the level of defenses one wants to getting away. In our experiment, a threshold value 18 is used. Table 2 provides the performance of the proposed GIDS. The table shows the classification rates of different attack types like DoS and Probe using different methods. Sangkatsanee et al. [4] shows an accuracy of 99.6%; however, they used only 10% of the KDD dataset. The proposed GIDS provides an accuracy of 96.7%, which uses the whole KDD dataset, which is more practical with reasonable accuracy than having tested the model with 10% data in the case of [4].

5 Conclusion

We discussed GAN-based intrusion detection for anomalies in perimeter defense. We proposed an adversarial generative training architecture, termed as GIDS, for a network intrusion detection system that minimizes the mapping error without the use of an external encoder. We have shown with experiments that GIDS successfully detects network intrusions for anomalies with an accuracy of 83.66%. Although GAN is commonly used for image dataset for identifying the correctness of images, we show that GAN can also be used with textual data for identifying anomalies with a reasonable accuracy.

References

1. Crothers, T.: Implementing intrusion detection systems: a hands-on guide for securing the network. Wiley (2020)
2. Goodfellow, I.J., et al.: Generative adversarial networks. *Commun. ACM* **63**(11), 139–144 (2014)
3. Schlegl, T., Seeböck, P., Waldstein, S.M., Schmidt-Erfurth, U., Langs, G.: Unsupervised anomaly detection with generative adversarial networks to guide marker discovery. In: Niethammer, M., et al. (eds.) *IPMI 2017*. LNCS, vol. 10265, pp. 146–157. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-59050-9_12
4. Sangkatsanee, P., Wattanapongsakorn, N., Charnsripinyo, C.: Practical real-time intrusion detection using machine learning approaches. *Comput. Commun.* **34**(18), 2227–2235 (2011)
5. Heba, F.E., Darwish, A., Hassaniien, A.E., Abraham, A.: Principle components analysis and support vector machine based intrusion detection system. In: *Proceedings of International Conference on Intelligent Systems Design and Applications*, pp. 363–367 (2010)
6. Garcia-Teodoroa, P., Diaz-Verdejoa, J., Macia-Fernandez, G., Vazquez, E.: Anomaly-based network intrusion detection: techniques, systems and challenges. *Comput. Secur.* **28**(1–2), 18–28 (2009)
7. KDD cup 1999 data. <http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html>
8. NSL KDD dataset. <https://www.unb.ca/cic/datasets/nsl.html>
9. Tavallae, M., Bagheri, E., Lu, W., Ghorbani, A.A.: A detailed analysis of the KDD cup 99 data set. In: *Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6 (2009)
10. Chintala, S.: How to train a GAN? tips and tricks to make GANs work (2016). <https://github.com/soumith/ganhacks>
11. Schlegl, T., Seebock, P., Waldstein, S.M., Langs, G., Erfurth, U.S.: f-AnoGAN: fast unsupervised anomaly detection with generative adversarial networks. *Med. Image Anal.* **54**, 30–44 (2019)
12. Akcay, S., Atapour-Abarghouei, A., Breckon, T.P.: GANomaly: semi-supervised anomaly detection via adversarial training. In: Jawahar, C.V., Li, H., Mori, G., Schindler, K. (eds.) *ACCV 2018*. LNCS, vol. 11363, pp. 622–637. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-20893-6_39
13. Yi, X., Walia, E., Babyn, P.: Generative adversarial network in medical imaging: a review. *Med. Image Anal.* **58**, 101552 (2019)
14. Radford, A., Metz, L., Chintala, S.: Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint [arXiv:1511.06434](https://arxiv.org/abs/1511.06434) (2015)