Contemporary Mathematics
http://ojs.wiserpub.com/index.php/CM/

Research Article

Black-Box Adversarial Attacks Against SQL Injection Detection Model

Maha Alqhtani*, Daniyal Alghazzawi, Suaad Alarifi

Information Systems Department, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 80200, Saudi Arabia
E-mail: mmohammadalqhtani@stu.kau.edu.sa

Received: 12 July 2024; **Revised:** 27 August 2024; **Accepted:** 4 September 2024

Abstract: Structured Query Language (SQL) injection attacks represent a substantial threat to the security of web applications, making the development of effective detection techniques crucial. These techniques have evolved from traditional signature-based techniques to more advanced techniques based on machine learning models. Machine learning detection models are often vulnerable to adversarial examples. Adversarial examples are deliberately crafted inputs designed to deceive models into making incorrect predictions by subtly altering the original dataset in ways that are typically imperceptible to humans. To train and test these machine learning models, datasets comprising both malicious and normal data are indispensable. However, the lack of sufficient and balanced datasets presents a significant challenge, particularly for models intended to detect SQL injection attacks. Most network traffic datasets exhibit a substantial class imbalance, with a disproportionate amount of normal traffic compared to malicious traffic, making it difficult to train effective and reliable detection models. This study addressed the shortcomings of current SQL injection detection techniques and proposed a conditional tabular generative adversarial network adversarial attack method. We evaluated the effectiveness of the generated adversarial SQL injection examples using qualitative and quantitative methods, measuring their ability to evade the detection model. A conventional neural network algorithm detection model was built and tested, and the generated adversarial examples successfully bypassed the detection model at a rate of up to 6%. The evidence demonstrated that the conditional tabular generative adversarial network successfully captures the statistical properties of real data and generates synthetic data that accurately represents the real data. This method is also expected to address the problem of insufficient and imbalanced SQL injection datasets, which could aid in training various machine learning models beyond the one used in our study.

***Keywords*:** SQL injection attacks, generative adversarial network, conditional tabular generative adversarial network, adversarial attacks, conventional neural network

MSC: 68T07, 68M25

Abbreviation

CTGAN	Conditional Tabular Generative Adversarial Network
CNN	Conventional Neural Network
ML	Machine Learning
DL	Deep Learning

DOI: https://doi.org/10.37256/cm.5420245292

IDS	Intrusion Detection System
WAFs	Web Application Firewalls
SQLIAs	SQL injection Attacks
KL	Kullback-Leibler

1. Introduction

As web applications are used more and more for diverse tasks, the security of these applications has become a paramount concern. Among the numerous threats that web applications face, Structured Query Language (SQL) injection attacks remain a significant challenge [1]. SQL injection attacks (SQLIAs) involve exploiting vulnerabilities in the application's input validation mechanisms to inject malicious SQL code into the underlying database query. Successful SQL injection attacks can result in unauthorized access and data manipulation and can potentially lead to severe security breaches. Traditional approaches to detecting SQL injection attacks often rely on signature-based methods or rule-based heuristics; however, attackers are increasingly employing advanced tools advanced tools for automated injection attacks, posing a challenge for traditional methods to effectively mitigate such threats. As SQL injection attacks continue to evolve, there is a need for more advanced and adaptive detection techniques. Machine learning (ML) models can be an effective solution for detecting SQL injection attacks, potentially providing better results than traditional methods. By training models on labeled datasets containing both normal and malicious SQL queries, ML algorithms can learn patterns and features indicative of SQL injection attacks. These models can then be used to classify and identify potential attacks in real time [2]. ML models are vulnerable to adversarial examples, which are carefully crafted inputs designed to mislead the models into making inaccurate predictions [3]. The emergence of adversarial examples presents an increasing difficulty, and ML models' protection and robustness against such attacks need to be improved.

Another challenge in the field of detecting SQL injection attacks is the lack of robust datasets specifically designed for testing and validating detection techniques. The concerns with the available datasets is compounded by several factors. Firstly, many datasets containing relevant traffic patterns cannot be openly shared due to privacy considerations. These datasets often include internal data used for specific research objectives or anomalous data traffic recorded by security companies. Furthermore, some publicly available datasets are outdated and inadequate for modern detection methods [4]. In addition, there is a significant class imbalance between benign and malicious traffic in most network traffic datasets. These datasets typically contain more examples of normal benign traffic compared to instances of malicious traffic. This skewed distribution, with the majority class of benign traffic, presents a difficulty for intrusion detection systems in accurately identifying and categorizing the rare malicious in accurately identifying and categorizing the rare malicious samples, leading to an overall low detection rate for those minority attack classes. Recently, numerous studies have aimed to tackle the challenges of imbalanced or missing data by employing methods such as the Synthetic Minority Over-sampling Technique (SMOTE) [5, 6] and generative adversarial networks (GAN) [7–13]. These techniques are used to expand network traffic samples and balance datasets, thereby improving the recognition rate of ML algorithms.

To study the potential influence of adversarial examples on the performance of the SQL injection detection models and address the challenges mentioned above, in this study we propose a novel approach for generating adversarial SQL injection examples based on the Conditional Tabular Generative Adversarial Networks (CTGAN) framework. CTGAN is a machine learning model used for generating synthetic tabular data that mimics the statistical properties of a given real (i.e. original) dataset [6]. We test the effectiveness of the generated adversarial SQL injections using qualitative and quantitative methods by measuring their ability to evade detection. The CTGAN model, a variant of GANs, has been successfully employed in generating synthetic data that closely resemble the original data distribution. The primary objective of this study is to develop a method that generates SQL injection examples capable of evading existing defense mechanisms. By leveraging the power of CTGAN, we aim to generate realistic attacks that mimic the characteristics of real-world SQL injection attempts. These adversarial examples can aid in the evaluation and improvement of defense mechanisms, facilitating the development of more robust and resilient web applications. The main contributions that distinguish our research from previous research work are the following:

We designed a CTGAN architecture that can effectively generate synthetic SQL injection examples.

We assessed the effectiveness of the generated adversarial SQL injection examples.

We trained and tested the performance of a CNN model using a dataset containing both real and synthetic data to examine if the adversarial SQL injection examples can successfully bypass the detection algorithm.

We conduct a comparison between the proposed approach and existing adversarial methods in the literature, examining how CTGAN performs in generating structured tabular data, particularly for creating synthetic SQL injection attacks.

The paper is structured as follows: We present a review of the related research in this area in Section 2. The methodology is presented in Section 3. Experimental results and discussion are reported in Section 4. In the last section, we provide the conclusion and discuss future work.

2. Related work

The application of ML techniques in detecting SQL injection attacks has garnered significant attention in recent research. Various studies have explored the efficacy of different ML and DL models in mitigating the risks posed by SQL injection attacks. For instance, ML models such as naive bayes classifier [14, 15], decision trees [16], random forest [17, 18], adaBoost classifier [19, 20], and support vector machines (SVM) [21, 22] have been employed to classify SQL queries as either benign or malicious based on extracted features. These models have demonstrated promising results in detecting SQL injection attacks by leveraging pattern recognition and anomaly detection methodologies. Moreover, the advent of deep learning (DL) architectures, such as convolutional neural networks (CNNs) [23–27], artificial neural networks (ANNs) [28, 29] and recurrent neural networks (RNNs) [30–32] , has shown promise in detecting SQL injection attacks due to their ability to learn complex patterns and relationships within data. Furthermore, recent advancements in adversarial machine learning have led to the development of innovative approaches for detecting SQL injection attacks. Adversarial training techniques such as generative adversarial networks (GANs) have been utilized to support ML and DL frameworks against adversarial attacks that seek to evade SQL injection detection systems. Various studies have highlighted the effectiveness of using GANs and their derivatives to produce adversarial examples that are able to deceive sophisticated ML detection models [33, 34]. Other studies have proposed GAN-based approaches to produce adversarial malicious traffic samples of cross-site scripting (XSS) [35], denial-of-service (DoS) [36, 37], and other types of IDS attacks such as SSH [7] that aim to attack IDSs by evading their detection systems. Study in [38] proposed a new approach to synthesizing traffic data while maintaining statistical similarity to the original data. The proposed approach, called a conditional tabular traffic generative adversarial network (CTTGAN), utilize a CTGAN algorithm. The authors demonstrate that CTTGAN outperforms other data synthesizing methods in terms of preserving individual privacy while maintaining statistical similarity to the real data. Alqarni et al. [39] proposed an IDS approach using a CTGAN model to generate more data of the minority class during training to make the NSL-KDD dataset balanced. To evaluate the efficiency of CTGAN to overcome the imbalance problem, three ML algorithms include DT, SVM and KNN were used to train and evaluate the dataset before and after applying CTGAN. The findings indicated that CTGAN demonstrated greater capability in capturing the distribution of discrete features. Kim et al. [6] proposed an training method to mitigate the generalization error problem of the ML model that detects IoT DDoS attacks, even in situations where 5G traffic is unbalanced. The SMOTE and GAN-based CTGAN was applied to address the issue of insufficient data. Experimental results showed that the ML model trained with CTGAN to augment attack data for training data in the 5G environment minimizes the generalization error problem. Asimopoulos et al. [40] proposed an AI-powered IDS against cyberattacks that targets the IEC 60870-5-104 protocol utilizing four ML methods: XGBoost, decision tree , random forest , and multi-layer perceptron. The researcher utilized the fast gradient sign method (FGSM) to assess the robustness of the previous methods against a common adversarial attack. Additionally, they employed FGSM-generated adversarial datasets to develop a CTGAN adversarial attack generator. The experimental results showed that compared to the CTGAN datasets, the tested models perform better on the FGSM adversarial datasets. Alabsi et al. [41] propose an intrusion detection system based on CTGAN for detecting DDoS and DoS attacks on IoT networks. The data produced by CTGAN is employed to train multiple ML classifiers. The experimental results showed that the generated dataset significantly enhanced the

detection model performance of these classifiers. Alabdulwahab et al. [42] proposed an ML pipeline using a CTGAN model to produce a synthetic dataset for the IoT environment. The generated synthetic dataset was assessed using several types of statistical metrics and ML classifiers that include decision tree, random forest, naïve bayes, multi-layer perceptron, XGBoost, gradient boost, and lightGBM. The experimental results indicated that it was possible to replace the generated dataset with a real dataset. Research on applying GANs in the domain of SQLIAs is still in the early stages.

Research on adversarial attacks against detection models of SQL injection attacks is relatively nascent, with only a limited number of studies conducted in this area [43, 44]. Liu et al. [45] proposed a tool named as DeepSQLi to produce examples for SQL injection attacks using a sequence-of-words prediction and deep learning model. This tool uses a neural language model that can be trained to capture the semantic features of SQL injection attacks that have not been captured by patterns in the training datasets. Appelt et al. [46] proposed automated method named 4SQLi for producing test inputs that could evade security filters, resulting in executable SQL injection queries. This method was based on a set of several mutation operators that modified inputs to produce new test inputs to trigger SQLi attacks, making it possible to create inputs that contained new patterns of the attack, thus increasing the possibility of producing successful SQL injection attacks. Demetrio et al. [17] proposed a tool named WAF-A-MoLE for producing adversarial SQL injection examples against web application firewalls (WAFs). This tool use a set of syntactical mutations that modify payload syntax while preserving the semantics of the original payload. Comprehensive evaluation of the tool demonstrates that it outperforms the existing machine learning-based WAF systems. These studies, however, only assess the effectiveness of their frameworks on a limited number of WAFs, and testing on a broader range of WAFs would strengthen the results. Guan et al. [47] proposed a novel adversarial SQL injection example generator based on reinforcement learning. The main drawbacks of this study is that the training time is relatively long. Table 1 summarizes the most relevant adversarial SQLIA methods in the literature.

Table 1. Adversarial SQL injection attacks proposed in the literature

Ref.	Methodology	Dataset	Detection models	Synthetic data evaluation metrics
Appelt et al. [46]	Java tool, called xavier	Not mentioned	GreenSQL (open-source database firewall)	Comparing with Std, which consists of a set of known SQL injection attack patterns
Demetrio et al. [17]	Unguided mutational fuzzer	OWASP ZAP and sqlmap	Naive bayes classifier, random forest, Gaussian SVM (G-SVM), and linear SVM (L-SVM)	Bypass the WAF
Liu et al. [45]	Deep natural language processing (NLP) and neural language models	GitHub	Neural language model transformer	Not mentioned
Lu et al. [4]	Deep convolutional GAN and genetic algorithms	CNVD, CVE and exploit-db	WAF	Phpstudy2018, safeDog, and sqli-lab range
Guan et al. [47]	Reinforcement learning	GitHub	ModSecurity, SafeDog, CNN, LSTM, MLP, Random forest, SVM, XGBoost, AdaBoost, GradientBoost	libinjection library, MySQL executions

3. Methodology

The general architecture for the proposed approach represented in Figure 1. It consists of three primary phases: (1) data preprocessing, (2) adversarial SQL injection examples generation using CTGAN, and (3) CNN-based SQL injection detection model training and evaluation.

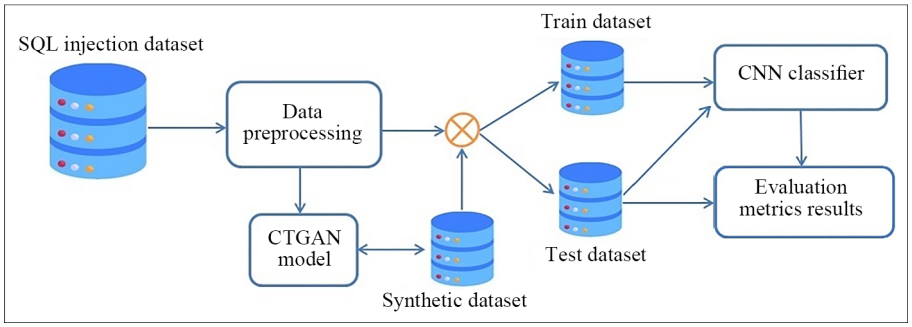


Figure 1. Architecture of the proposed approach

3.1 *Dataset*

In the field of SQL injection detection, many available datasets cover a wide range of intrusion detection traffic but with limited availability of SQL injection instances. In this research, we require a publicly available dataset focused on SQLIAs. After examining our options, we selected a publicly available dataset from [48], known for its emphasis on SQL injection. This dataset was previously employed by researchers in [2, 20, 26, 48–51] to detect SQLIAs using ML models. It consist of a total of 30,919 records, with 19,537 normal queries and 11,382 malicious queries. Each record contains two main features: "Query", which represents the SQL queries, and "Label", which indicates whether the query is normal (0) or malicious (1). The used dataset statistics are shown in Figure 2.

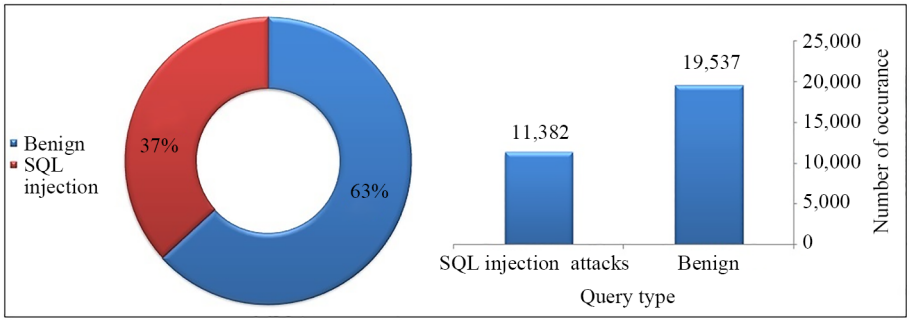


Figure 2. Distribution of normal and SQL injection attacks in the dataset

3.2 *Data pre-processing*

Data preprocessing is a crucial part of data mining where raw data is cleaned and transformed to enhance dataset quality. It encompasses various processes, such as cleaning and preparing the data in a format suitable for training the detection models, particularly for text-based tasks [10]. Data cleaning is essential, as it stops the model from learning wrong relationships or making predictions based on incomplete or duplicate data. To prepare the dataset for the training phase, we perform the following:

Removing missing values and duplicated records. We obtained a total of 30,907 rows, consisting of 19,529 normal queries and the remaining rows classified as SQL injection attacks (SQLIAs). Identifying and correcting errors in the dataset by manually correcting the queries that have syntax errors.

Performing feature extraction, which is the process of transforming raw data, such as text or images, into a numerical representation that can be used as input for ML algorithms.

Balancing the dataset. Figure 2, indicates that the dataset was imbalanced because there were significantly fewer instances of SQLIAs than normal queries, which can affect the detection model performance. To address this problem,

we manually balance the dataset using undersampling by reducing the amount of the class that has more samples to matches the amount of samples in the minority class to achieve a balanced distribution. The resulting dataset comprised 20,000 rows, with 50% representing normal statements and the remaining 50% classified as SQL injection attacks (SQLIAs).

3.3 *CTGAN model training*

CTGAN was preferred to traditional GAN models for two reasons. The first is its ability to overcome the challenges faced by GANs in effectively capturing complex dependencies and distributions in discrete, structured data [52]. Second, SQL injections have complex structured patterns, such as keywords, punctuation, operators, and strict syntactic rules that must be adhered to, and the CTGAN is better equipped to learn these patterns. The CTGAN architecture is specifically designed for producing synthetic tabular data. It addresses the challenge of producing realistic synthetic data that preserves the statistical properties and dependencies present in the real dataset. The structure of the proposed CTGAN model is shown in Figure 3.

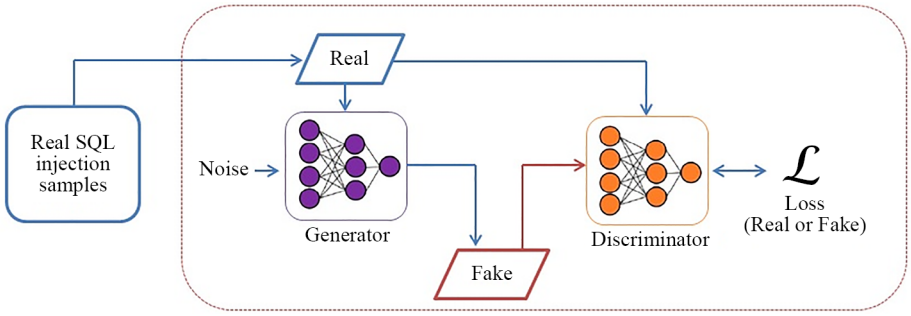


Figure 3. Proposed CTGAN model

The CTGAN architecture consists of two main components: a generator (G) and a discriminator (D). Both components are trained iteratively, as the generator learns to produce synthetic samples that resemble the real dataset and try to fool the discriminator. Discriminator learns to differentiate between real and synthetic data [47]. The loss function of the generator is defined in equation (1).

$$\mathscr{L}_{\text{gen}} = -\mathbb{E}_{z\sim p(z)}[\log(D(G(z)))] \tag{1}$$

where:

z is a random noise sampled from a prior distribution $p(z)$.

$G(z)$ represents the produced samples obtained by adding the noise (z) to the generator.

D is the discriminator, which outputs a probability score indicating the likelihood that a given sample is real.

The discriminator loss is defined as:

$$\mathscr{L}_{\text{dis}} = -\mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(D(x))] - \mathbb{E}_{z\sim p(z)}[\log(1-D(G(z)))] + \lambda \cdot \mathscr{L}_{\text{gp}} \tag{2}$$

where:

x represents real data taken from the original dataset with distribution $p_{\text{data}}(x)$.

$\mathscr{L}_{\text{gp}}$ represents the gradient penalty term, which encourages smoothness of the discriminator's decision boundary.

λ is a hyperparameter that controls the influence of the gradient penalty term.

The generator network's goal is to produce synthetic data samples that closely match the distribution of the real data and deceive the discriminator. It achieves this by minimizing the distance between the distribution of the synthetic data and that of the real data. Simultaneously, by maximizing the discriminator loss, D learns to improve its ability to classify real data as close to 1 and fake data as close to 0. We present an overview of the implementation of the CTGAN model in Algorithm 1.

Algorithm 1 Training of the CTGAN

Input:

x: Real data samples (from training data distribution $p_{\text{data}}(x)$)

z: Random noise vectors sampled from noise distribution $p_z(z)$

c: Conditional variables (labels/categories)

Initialize the generator and the discriminator

number of training iterations

for Generator-step do

Generator generates the adversarial SQL injection examples based on x

Update the parameters of generator according to equation (1).

for Discriminator-steps do

Classify the training set (x, $x_{\text{gen}} = G(z,\ c)$), getting predicted labels

Update the parameters of discriminator according to equation (2).

Output: $x_{\text{gen}} = G(z,\ c)$: New synthetic data generated by the model.

3.4 *CNN model training*

In the literature, many researchers have highlighted the use of several ML and DL algorithms to detect SQLIAs. In this section, we trained and compared the performance of several classifiers including RNN autoencoder, CNN, ANN, Naïve Bayes, Decision Tree, SVM, Random Forest, and Logistic Regression. The model that achieves the highest accuracy will be selected to perform our experiment. The grid search technique was used to optimize the hyperparameters of these models to achieve the best performance. The results are shown in Figure 4.

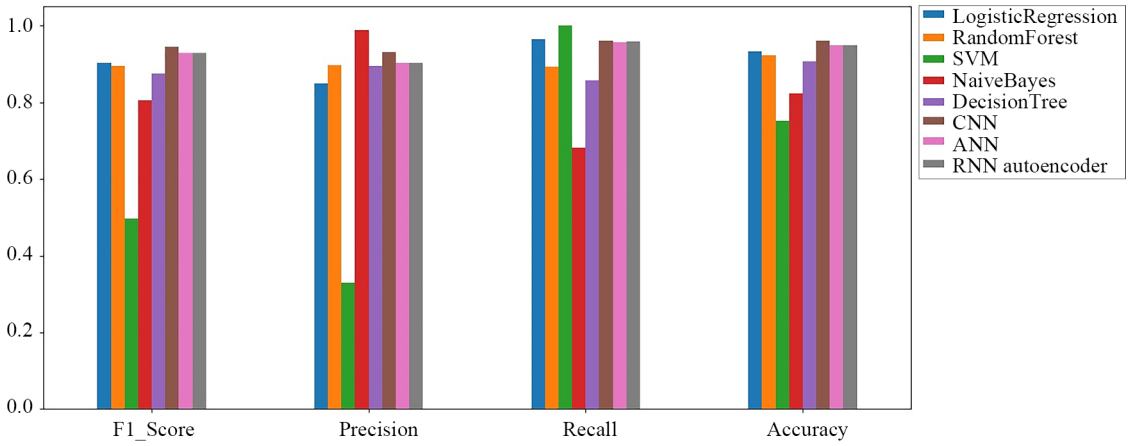


Figure 4. The evaluation metrics comparison for different detection algorithms

The findings in Figure 4. show that the ANN and RNN autoencoder are effective in detecting SQL injection attacks, achieving a high accuracy of 94%. The random forest, decision tree and logistic regression models also performed well, achieving accuracy scores of 92%, 90%, and 93%, respectively. The CNN model achieves the highest accuracy of 95%. However, the SVM and Naive Bayes models had the lowest accuracy scores of 75% and 82%, respectively. According

to the this findings, the CNN model had the highest accuracy, indicating its potential for detecting SQL injection attacks. For that reason, we developed a defense model based on the CNN model. The proposed CNN architecture consists of a Conv1D layer with ReLU activation function. This layer processes input data shaped as (X.shape [1], 1). Then a maxPooling1D layer is added to reducing spatial dimensions. The subsequent flatten layer reshapes input data into a one-dimensional array. Two dense layers are then added, the first with ReLU activation, and the second dense layer with a sigmoid activation function for binary classification. By training the model on labeled data, the CNN learns to classify input queries as either normal or SQLIAs. Table 2 presents the optimized parameters for the CNN model.

Table 2. Proposed CNN classifier structure

Layer Type	Input	Parameters	Activation Function
Conv1D	(X.shape [1], 1)	Filters: 32, Kernel Size: 3	ReLU
MaxPooling1D	None	Pool Size:2	None
Flatten	(X.shape [1], 1)	None	None
Dense	128	None	ReLU
Dense	1	None	Sigmoid

Table 2 shows main information, such as kernel size, number of filters, and activation function for each layer.

3.5 *Results and discussion*

This section discuss the experimental results. The experiment was conducted in a Python environment on a Windows 10 Pro with 4 GB of RAM and a clock speed of 2.00 GHz. In Table 3, we provide an overview of the hyperparameters used in the training of the CTGAN model and CNN classifier, such as loss function, optimizer, batch size , epoch, learning rate.

Table 3. Hyperparameters used in CTGAN and CNN classifier training

Hyperparameter	CTGAN	CNN
Loss Function	Binary Cross-Entropy	Binary Cross-Entropy
optimizer	Adam	Adam
Learning Rate	2e-4	0.001
Epochs	500	10
Batch Size	Default	32

First, the real SQL injection queries was pre-processed and used to train the CTGAN to produce adversarial SQL injection examples. To preserve the syntax and structure of real SQL injection queries during the training process, carefully designed noise (z) was introduced. This included adding comments and altering the case of keywords (e.g., “select” replaced with “sELEcT”). This noise was incorporated into the generator (G) during the training of the CTGAN. An example of the added noise is shown in Figure 5.

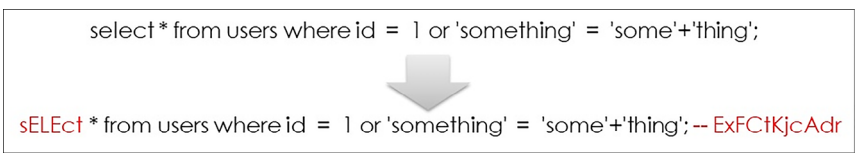
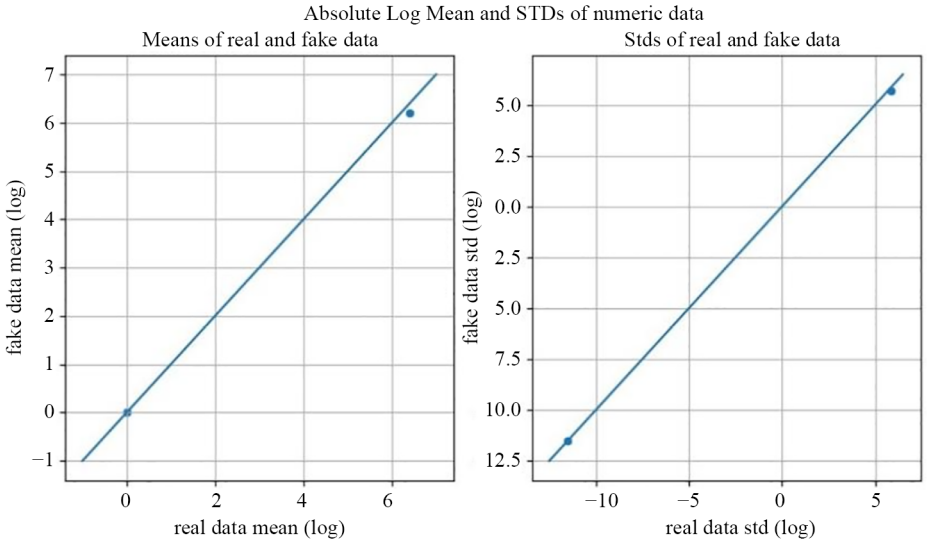


Figure 5. Example of the noise added to the generator during the training of the CTGAN

We validated the syntax of the synthetic queries using MySQL system. Figure 6 shows an example of the queries validation process.

Figure 6. Example of synthetic queries syntax validation

Furthermore, to assess the similarity and quality of both the real and synthetic data, visual and statistical methods were used. In visual evaluation, we use feature-by-feature comparison between the real data and the synthetic data. Also, the Absolute Log Mean and standard deviations (STDs), cumulative sum and density of data points in both datasets are visually represented.



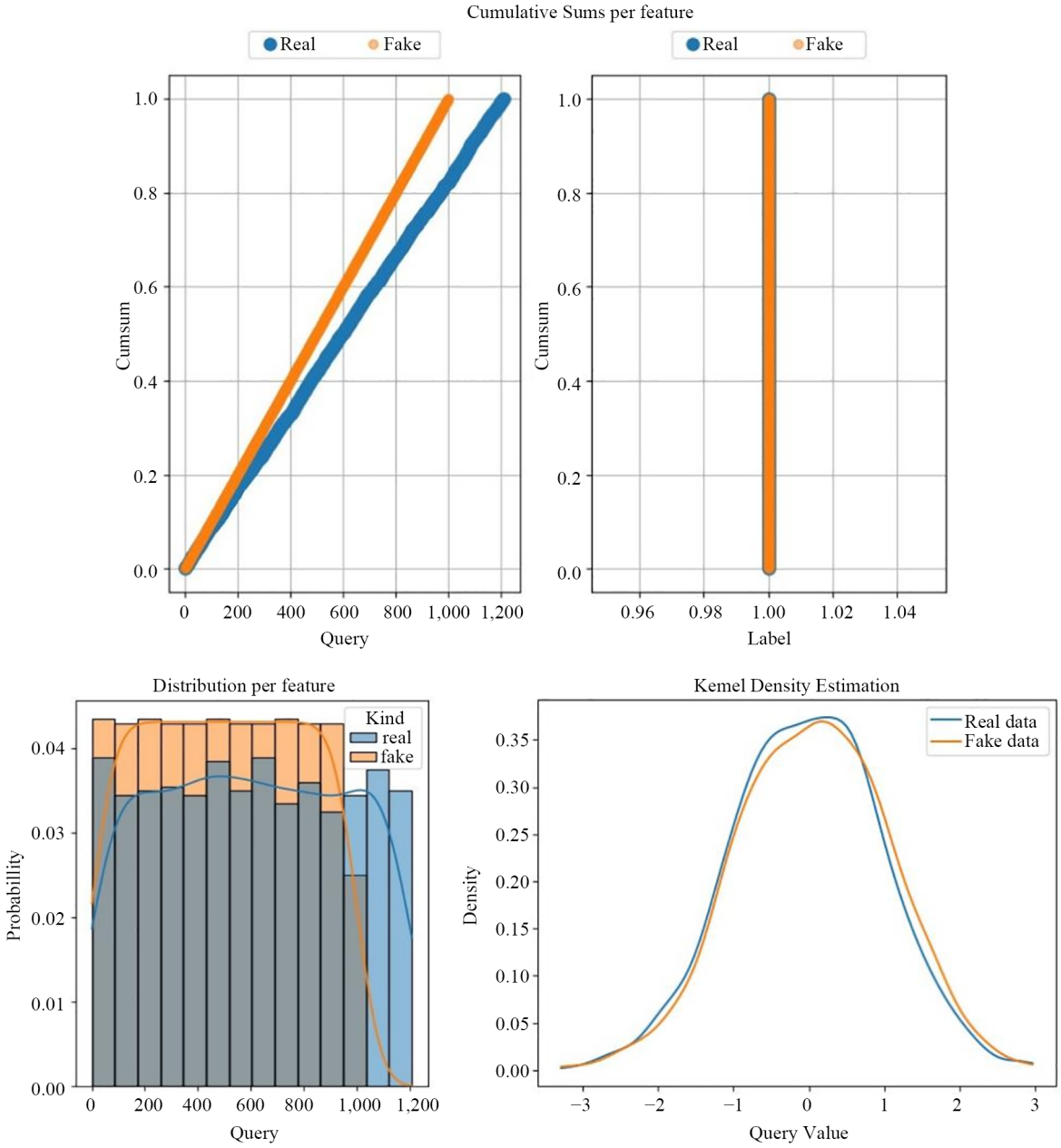


Figure 7. The plots show the absolute log mean, standard deviation diagram, cumulative sums, kernel density estimation, and distribution per feature for both the real and synthetic dataset

Figure 7 presents a comparison of the Absolute Log Mean and standard deviations (STDs) of the numeric data plots for both real and synthetic data, plotted on a log scale. The figure demonstrates a strong correlation between the means and STDs of the real and synthetic datasets. This indicates that the CTGAN has effectively captured the mean values of the numeric features and preserved the spread of the synthetic data features. The cumulative sum plots illustrate the cumulative distribution of the real and synthetic data for the 'Query' and 'Label' features. This allows for a visual comparison of the distribution similarities between the synthetic columns (orange line) and their real counterparts (blue line). The synthetic data closely follows the real data distribution for the 'Query' and 'Label' features. The chart displays the data distribution of the 'Query' feature generated by the CTGAN alongside the distribution of the 'Query' feature in the real data. The blue histogram represents the distribution of real dataset, the orange histogram represents the distribution of the synthetic

dataset, and the gray columns represents the overlap between the distributions of both datasets. Kernel density estimation in the same chart also shows a close overlap between the synthetic and real dataset distributions.

Overall, the visual assessments indicate that the CTGAN model has performed reasonably well in capturing the statistical characteristics and distributions of the real data for SQLIAs. These visual assessments provide insights into the synthetic data's quality, but may not capture all aspects of the data. To address this, additional evaluation metrics, such as quantitative statistical tests (e.g., KL divergence test, Wasserstein distance), were used to capture more comprehensive aspects of the data. Kullback-Leibler (KL) divergence and Wasserstein distance were used to validate the similarity between the synthetic SQL injection dataset distribution and that of the real SQL injection dataset. KL divergence and Wasserstein distance show a lower value, which indicates that the two distributions are relatively identical, as shown in Table 4.

Table 4. Statistical evaluation scores of the synthetic SQL injection data

Metric	Value
Discrete KL divergence	0.0
Wasserstein distance	0.04

The KL divergence value of 0.0 suggests that the discrete features (i.e., SQL injection queries) of the synthetic data closely resemble those of the real data, indicating a high level of similarity. Additionally, the Wasserstein distance value of 0.04 signifies a relatively small difference between the distributions of both datasets. These results suggest that CTGAN, along with the added crafted noise, effectively preserves the syntax of the queries and generates synthetic SQL injection data that closely mimics the real data.

The CNN model was developed and trained. The process of determining the best hyperparameters, such as batch size and number of epochs is one of the most important factors affecting the CNN model performance. Grid search techniques were used to find CNN hyper-parameters best values. After the optimization of the CNN hyperparameters, the model was trained in two stages. Initially, the training was conducted using the real SQL injection dataset. Subsequently, the model was further trained with a hybrid dataset that combined both real and synthetic SQL injection data. The performance of the CNN model was tested using various metrics, including accuracy, precision, recall, and F1-score.

Table 5. Performance metrics of the CNN model using real and hybrid datasets

Evaluation metrics	Real dataset	Hybrid dataset
Accuracy	95%	89%
Precision	92%	82%
Recall	95%	96%
F1-score	94%	88%

According to the findings in Table 5 the CNN model achieves higher performance metrics with the real dataset compared to the hybrid dataset. Specifically, the model achieves an accuracy of 95% with the real dataset, which decreases to 89% with the hybrid dataset. Similarly, precision decreases from 92% to 82%, while recall increases slightly from 95% to 96% when using the hybrid dataset. The F1-score also shows a decrease from 94% to 88% with the hybrid dataset. While the high accuracy of 95% with the real dataset indicates that the chosen CNN architecture effectively captured relevant features and accurately distinguished between SQL injection queries and normal queries. The addition of synthetic SQL injection data introduces a level of complexity, as the model needs to differentiate between real and synthetic data, resulting in misclassifications and reducing its overall accuracy. These differences provide strong evidence that the CTGAN is capable of accurately capturing the statistical characteristics of the real data and is able to generate synthetic data that are

representative of the real data. After evaluating the accuracy of the CNN, we demonstrated that CTGAN had a significant impact on detection model performance. We compared the proposed CTGAN model with other adversarial attack methods proposed in the literature.

Compared to other adversarial attack methods proposed in [4, 14–17], which employed various techniques such as Java tools, mutational fuzzers, deep natural language processing, DCGAN, and reinforcement learning, our study utilized a more advanced approach based on CTGAN for generating synthetic SQL injection queries. For instance, compared to the framework proposed in [16], the CTGAN had a more flexible strategy for attack generation. With a sophisticated generator and discriminator, CTGAN can adapt and learn from the target dataset, capturing intricate characteristics and correlations. The discriminator's feedback during training ensures continuous improvement and alignment with the target distribution, enabling CTGAN to generate realistic and diverse adversarial examples that possess the same statistical properties as real data. Additionally, CTGAN's learning time depends on the dataset complexity and required training iterations, allowing for rapid generation of synthetic data samples once trained. In contrast, the reinforcement learning model proposed in [17] has a relatively longer training time.

While previous studies proposed various evaluation metrics, such as comparing with known attack patterns and bypassing syntactic analysis performed by WAFs, our evaluation methodology went beyond traditional approaches by utilizing visual assessment and quantitative metrics. We employed visual assessment techniques such as cumulative sum and distribution per feature analysis. These techniques provided insights into the characteristics of the queries, enhancing the evaluation process. Furthermore, we introduced statistical metrics such as KL divergence and Wasserstein distance. The lower scores obtained through these metrics demonstrated the effectiveness of our approach in generating realistic queries. Moreover, our study incorporated a hybrid training approach. We initially trained a CNN model using a real dataset, achieving a high accuracy of 95%. Subsequently, we combined the real dataset with the synthetic data produced by CTGAN and trained the CNN model again. This highlighted the influence of the synthetic data on the model's performance and its ability to bypass the model, as most previous studies focused on bypassing WAFs. In summary, our research surpasses prior studies in methodology used, evaluation approach, and effectiveness of synthetic data produced by CTGAN. CTGAN's ability to produce structured tabular data makes it suitable for generating synthetic SQLIAs. However, it may be sensitive to the quantity and quality of the original SQL injection dataset used for training. The main limitation of our study is the small dataset size. Overall, this research demonstrates the potential of using CTGAN-generated adversarial data to enhance SQLIA detection and emphasizes the need for additional research in this domain.

4. Conclusion

In this research, we proposed a framework that leverages the conditional tabular generative adversarial network (CTGAN) to produce adversarial SQL injection examples. The primary objective was to assess the effectiveness of these synthetic examples in evaluating the robustness of SQL injection detection models. Additionally, we aimed to solve the issue of limited or insufficient training data by augmenting the real dataset with the generated adversarial SQL injection data. CTGAN model captures statistical properties of real data , as confirmed by visual assessment and quantitative metrics. To assess the effectiveness of the produced adversarial examples, we developed a detection model based on CNN algorithm and trained it on the original and a hybrid dataset consisting of both real and synthetic examples. The CNN model was tested using various metrics, including accuracy, precision, recall, and F1-score. The CNN model showed better performance when trained on the real dataset rather than the hybrid dataset, as indicated by the results. This underscores the capability of CTGAN-generated adversarial data to evade detection and the need for further research in this area. It is important to highlight that the study was conducted on a relatively small dataset, which may have influenced the results. Future research could investigate the utilization of a larger and more varied datasets to further assess the effectiveness of CTGAN in generating adversarial SQL injection examples and its impact on detection model performance.

Overall, this study significantly contributes to the cybersecurity domain by demonstrating the potential of generative adversarial networks, in creating synthetic data that captures the characteristics of adversarial SQL injection examples. It effectively tackles the problem of limited data, particularly when the real dataset lacks a sufficient number of instances

representing SQL injection attacks. The synthetic data produced by CTGAN serves as a valuable augmentation, providing additional samples that capture the intricate patterns and characteristics of these attacks. By incorporating this augmented dataset, the model can generalize better and its overall detection performance can be improved. Furthermore, this study offers an efficient alternative to the traditional approach of collecting real instances for training detection models. Instead of investing significant time and effort in finding and collecting real instances, CTGAN enables the rapid generation of synthetic data. This time-efficient approach proves particularly beneficial in scenarios where there is an urgent need to train detection models or when real instances are scarce or challenging to obtain.

Acknowledgement

I extend my heartfelt appreciation to King Abdulaziz University for their support and funding, which were crucial in facilitating this research. Thank you for enabling my academic pursuits.

Conflict of interest

The authors declare no competing financial interest.

References

[1] Alnabulsi H, Islam MR, Mamun Q. Detecting SQL injection attacks using SNORT IDS. In: *Asia-Pacific World Congress on Computer Science and Engineering*. Nadi, Fiji: IEEE; 2014. p.1-7.

[2] Zhang W, Li Y, Li X, Shao M, Mi Y, Zhang H, et al. Deep neural network-based SQL injection detection method. *Security and Communication Networks*. 2022; 2022(12): 1-9.

[3] Ibitoye O, Abou-Khamis R, Matrawy A, Shafiq MO. The threat of adversarial attacks on machine learning in network security-a survey. *arXiv:191102621*. 2019.

[4] Lu D, Fei J, Liu L, Li Z. A GAN-based method for generating SQL injection attack samples. In: *2022 IEEE 10th Joint International Information Technology and Artificial Intelligence Conference*. Chongqing, China: IEEE; 2022. p.1827-1833.

[5] Alshamy R, Ghurab M, Othman S, Alshami F. Intrusion detection model for imbalanced dataset using smote and random forest algorithm. In: *Advances in Cyber Security*. Heidelberg: Springer; 2021. p.361-378.

[6] Kim YS, Kim YE, Kim H. A model training method for ddos detection using CTGAN under 5GC traffic. *Computer Systems Science and Engineering*. 2023; 47(1): 1125-1147.

[7] Vu L, Bui CT, Nguyen QU. A deep learning based method for handling imbalanced problem in network traffic classification. In: *SoICT 2017: The Eighth International Symposium on Information and Communication Technology*. New York: Association for Computing Machinery; 2017. p.333-339.

[8] Yilmaz I, Masum R. Expansion of cyber attack data from unbalanced datasets using generative techniques. *arXiv:191204549*. 2019.

[9] Chen H, Jiang L. Efficient GAN-based method for cyber-intrusion detection. *arXiv:190402426*. 2019.

[10] Shahriar MH, Haque NI, Rahman MA, Alonso M. G-ids: Generative adversarial networks assisted intrusion detection system. *arXiv:200600676*. 2020.

[11] de Araujo-Filho PF, Kaddoum G, Campelo DR, Santos AG, Macêdo D, Zanchettin C. Intrusion detection for cyber-physical systems using generative adversarial networks in fog environment. *IEEE Internet of Things Journal*. 2020; 8(8): 6247-6256.

[12] Huang S, Lei K. IGAN-IDS: An imbalanced generative adversarial network towards intrusion detection system in ad-hoc networks. *Ad Hoc Networks*. 2020; 105: 102177. Available from: http://doi.acm.org/10.1016/j.adhoc.2020.102177.

[13] Ullah I, Mahmoud QH. A framework for anomaly detection in iot networks using conditional generative adversarial networks. *IEEE Access*. 2021; 9: 165907-165931. Available from: http://doi.acm.org/10.1109/ACCESS.2021.3132127.

[14] Pham TS, Nguyen QU, Nguyen XH. Generating artificial attack data for intrusion detection using machine learning. In: *SoICT '14: Fifth symposium on Information and Communication Technology*. New York: Association for Computing Machinery; 2014. p.286-291.

[15] Anamika J, Geetha V. SQL Injection detection using machine learning. In: *2014 international conference on control, instrumentation, communication and computational technologies*. Kanyakumari, India: IEEE; 2014. p.1111-1115.

[16] Mejia-Cabrera HI, Paico-Chileno D, Valdera-Contreras JH, Tuesta-Monteza VA, Forero MG. Automatic detection of injection attacks by machine learning in NoSQL databases. In: *Mexican Conference on Pattern Recognition*. Heidelberg: Springer; 2021. p.23-32.

[17] Demetrio L, Valenza A, Costa G, Lagorio G. Waf-a-mole: Evading web application firewalls through adversarial machine learning. In: *SAC'20: The 35th ACM/SIGAPP Symposium on Applied Computing*. New York: Association for Computing Machinery; 2020. p.1745-1752.

[18] Chindove H, Brown D. Adaptive machine learning based network intrusion detection. In: *icARTi '21: International Conference on Artificial Intelligence and its Applications*. New York: Association for Computing Machinery; 2021. p.1-6.

[19] Tripathy D, Gohil R, Halabi T. Detecting SQL injection attacks in cloud SaaS using machine learning. In: *2020 IEEE 6th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing,(HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)*. Baltimore, MD, USA: IEEE; 2020. p.145-150.

[20] Roy P, Kumar R, Rani P. SQL injection attack detection by machine learning classifier. In: *2022 International Conference on Applied Artificial Intelligence and Computing*. Salem, India: IEEE; 2022. p.394-400.

[21] Priyaa B, Devi MI. Hybrid SQL injection detection system. In: *2016 3rd International Conference on Advanced Computing and Communication Systems*. Coimbatore, India: IEEE; 2016. p.1-5.

[22] Kar D, Panigrahi S, Sundararajan S. SQLiGoT: Detecting SQL injection attacks using graph of tokens and SVM. *Computers and Security*. 2016; 60: 206-225. Available from: https://doi.org/10.1016/j.cose.2016.04.005.

[23] Wu Y, Wei D, Feng J. Network attacks detection methods based on deep learning techniques: A survey. *Security and Communication Networks*. New York: John Wiley and Sons; 2020. p.1-17.

[24] Krishnan SA, Sabu AN, Sajan PP, Sreedeep A. SQL injection detection using machine learning. *Geintec Magazine-Innovation and Technology Management*. 2021; 11(3): 300-310.

[25] Chen D, Yan Q, Wu C, Zhao J. Sql injection attack detection and prevention techniques using deep learning. *Journal of Physics Conference*. 2021; 1757(1): 012055.

[26] Ketema A. *Developing Sql Injection Prevention Model Using Deep Learning Technique*. London: St. Mary's University; 2022.

[27] Sun H, Du Y, Li Q. Deep learning-based detection technology for SQL injection research and implementation. *Applied Sciences*. 2023; 13(16): 9466.

[28] Kamtuo K, Soomlek C. Machine Learning for SQL injection prevention on server-side scripting. In: *2016 International Computer Science and Engineering Conference*. Chiang Mai, Thailand: IEEE; 2016. p.1-6.

[29] Ross K, Moh M, Moh TS, Yao J. Multi-source data analysis and evaluation of machine learning techniques for SQL injection detection. In: *ACMSE'18: Proceedings of the 2018 ACM Southeast Conference*. New York: Association for Computing Machinery; 2018. p.1-8.

[30] Yin C, Zhu Y, Fei J, He X. A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*. 2017; 5: 21954-21961. Available from: https://doi.org/10.1109/ACCESS.2017.2762418.

[31] Park SH, Park HJ, Choi YJ. RNN-based prediction for network intrusion detection. In: *2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. Fukuoka, Japan: IEEE; 2020. p.572-574.

[32] Alghawazi M, Alghazzawi D, Alarifi S. Deep learning architecture for detecting SQL injection attacks based on RNN autoencoder model. *Mathematics*. 2023; 11(15): 3286.

[33] Hu W, Tan Y. Generating adversarial malware examples for black-box attacks based on GAN. In: *International Conference on Data Mining and Big Data*. Heidelberg: Springer; 2022. p.409-423.

[34] AlEroud A, Karabatis G. Bypassing detection of URL-based phishing attacks using generative adversarial deep neural networks. In: *IWSPA'20: Proceedings of the Sixth International Workshop on Security and Privacy Analytics*. New York: Association for Computing Machinery; 2020. p.53-60.

[35] Zhang X, Zhou Y, Pei S, Zhuge J, Chen J. Adversarial examples detection for XSS attacks based on generative adversarial networks. *IEEE Access*. 2020; 8: 10989-10996. Available from: https://doi.org/10.1109/ACCESS.2020.2965184.

[36] Abdelaty M, Scott-Hayward S, Doriguzzi-Corin R, Siracusa D. Gadot: Gan-based adversarial training for robust ddos attack detection. In: *2021 IEEE Conference on Communications and Network Security*. Tempe, AZ, USA: IEEE; 2021. p.119-127.

[37] Novaes MP, Carvalho LF, Lloret J, Proença Jr ML. Adversarial deep learning approach detection and defense against DDoS attacks in SDN environments. *Future Generation Computer Systems*. 2021; 125(1): 156-167.

[38] Wang J, Yan X, Liu L, Li L, Yu Y. CTTGAN: Traffic data synthesizing scheme based on conditional GAN. *Sensors*. 2022; 22(14): 5243.

[39] Alqarni AA, El-Alfy ESM. Improving intrusion detection for imbalanced network traffic using generative deep learning. *International Journal of Advanced Computer Science and Applications*. 2022; 13(4): 959-967.

[40] Asimopoulos DC, Radoglou-Grammatikis P, Makris I, Mladenov V, Psannis KE, Goudos S, et al. Breaching the defense: Investigating FGSM and CTGAN adversarial attacks on IEC 60870-5-104 AI-enabled intrusion detection systems. In: *ARES'23: Proceedings of the 18th International Conference on Availability, Reliability and Security*. New York: Association for Computing Machinery; 2023. p.1-8.

[41] Alabsi BA, Anbar M, Rihan SDA. Conditional tabular generative adversarial based intrusion detection system for detecting ddos and dos attacks on the internet of things networks. *Sensors*. 2023; 23(12): 5644.

[42] Alabdulwahab S, Kim YT, Seo A, Son Y. Generating synthetic dataset for ml-based ids using ctgan and feature selection to protect smart iot environments. *Applied Sciences*. 2023; 13(19): 10951.

[43] Liu Z, Yin X. Lstm-cgan: Towards generating low-rate ddos adversarial samples for blockchain-based wireless network detection models. *IEEE Access*. 2021; 9: 22616-22625. Available from: https://doi.org/10.1109/ACCESS.2021.3056482.

[44] Ahsan R, Shi W, Ma X, Lee Croft W. A comparative analysis of CGAN-based oversampling for anomaly detection. *IET Cyber-Physical Systems: Theory and Applications*. 2022; 7(6): 40-50.

[45] Liu M, Li K, Chen T. DeepSQLi: Deep semantic learning for testing SQL injection. *arXiv:200511728*. 2005.

[46] Appelt D, Nguyen CD, Briand LC, Alshahwan N. Automated testing for SQL injection vulnerabilities: An input mutation approach. In: *ISSTA 2014: Proceedings of the 2014 International Symposium on Software Testing and Analysis*. New York: Association for Computing Machinery; 2014. p.259-269.

[47] Guan Y, He J, Li T, Zhao H, Ma B. SSQLi: A black-box adversarial attack method for SQL injection based on reinforcement learning. *Future Internet*. 2023; 15(4): 133.

[48] Sajid576. SQL Injection Dataset. 2021. Available from: https://www.kaggle.com/datasets/sajid576/sql-injection-dataset [Accessed Accessed 6th July 2021].

[49] Alkhathami JM, Alzahrani SM. Detection of SQL injection attacks using machine learning in cloud computing platform. *Journal of Theoretical and Applied Information Technology*. 2022; 100(15): 5446-5459.

[50] Deriba F, Salau AO, Mohammed SH, Kassa TM, Demilie WB. Development of a Compressive Framework Using Machine Learning Approaches for SQL Injection Attacks. *Electrical Engineering Review*. 2022; 1(7): 183-189.

[51] ALAzzawi A. SQL injection detection using RNN deep learning model. *Journal of Applied Engineering and Technological Science*. 2023; 5(1): 531-541.

[52] Watanuki S, Nomura Y, Kiyota Y, Kubo M, Fujimoto K, Okada J, et al. Applying a method for augmenting data mixed from two different sources using deep generative neural networks to management science. *Applied Sciences*. 2023; 14(1): 378.