

CTGAN Based Real Time Anomaly Detection for Enhanced Cybersecurity in Agriculture 5.0

C. Senthilkumar

Thiagarajar college of Engineering Madurai

B. Subbulakshmi

Thiagarajar college of Engineering Madurai

Manikandan Thirumalaisamy

Multimedia University

Ramesh Shunmugam

Multimedia University

M. Arvindhan

saroarvindmster@gmail.com

The American University of Vietnam

Research Article

Keywords: CTGAN, Intrusion Detection, Synthetic Data, CICIDS2017, UNSW-NB15

Posted Date: September 29th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10650961/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

CTGAN Based Real Time Anomaly Detection for Enhanced Cybersecurity in Agriculture 5.0

Dr.C.Senthilkumar ¹, Dr.B.Subbulakshmi ², Manikandan Thirumalaisamy³, Ramesh Shunmugam⁴, M.Arvinthan⁵

¹ Associate Professor CSE Department Thiagarajar college of Engineering Madurai Tamilnadu cskcse@tce.edu

² Assistant Professor CSE Department Thiagarajar college of Engineering Madurai Tamilnadu bscse@tce.edu

³ Centre for Intelligent Cloud Computing, COE for Advanced Cloud, Multimedia University, 75450 Melaka, Malaysia, Rajalakshmi Engineering College, Thandalam, 602105 Kanchipuram, India, manikandan.thirumalaisamy@rajalakshmi.edu.in

⁴ Centre for Intelligent Cloud Computing, COE for Advanced Cloud, Multimedia University, 75450 Melaka, Malaysia, Rajalakshmi Engineering College, Thandalam, 602105 Kanchipuram, India, ramesh.s@rajalakshmi.edu.in

⁵ American School of Management, The American University of Science, Vietnam, saroarvindmster@gmail.com

* Corresponding author, e-mail: saroarvindmster@gmail.com

Abstract: The rising sophistication of cyberattacks, from ransomware and phishing to Distributed Denial of Service (DDoS) attacks, necessitates sophisticated defense mechanisms that are beyond the capabilities of conventional machine learning models, such as support vector machines and decision trees. This paper presents the use of Conditional Tabular Generative Adversarial Networks (CTGANs) in preventives defense against cyber-attacks through the creation of high-fidelity, statistically valid synthetic datasets that are particularly useful when real-world data are unavailable because access to them is limited by sensitivity or rarity. CTGANs specifically address both continuous and categorical variables, allowing the synthetic generation of realistic network traffic patterns, such as malicious activities, to be used in efficient training of anomaly detection systems for detecting novel threats in near real time. The proposed framework was tested with the CICIDS2017 and UNSW-NB15 datasets within a simulated corporate network of 50 nodes connected dynamically to generate dynamic traffic. Tested, the model not only matched traditional models but exceeded them with 92% average precision and a 91% F1-score. In one of its most convincing statements: an 89 percent resistance to data noise, proving that it can take a beating. The algorithm is effective for real-time defense with high performance, on account of its generalization and robustness to adversarial attack. It makes an ordinary detection system into a healing, self-learning system that is able to survive the chaos and uncertainty of today's digital security world equals anti-fragile.

Keywords: CTGAN, Intrusion Detection, Synthetic Data, CICIDS2017, UNSW-NB15

1. INTRODUCTION

As cyber attacks become increasingly sophisticated, the research community has shifted considerable attention toward enhancing network security protocols through machine learning. Intrusion detection systems (IDS) based on machine learning have been successful in identifying malicious behavior in diverse network environments. However, one of the most significant barriers is adversarial attacks, which exploit the vulnerabilities faced by machine learning models by exposing input in a hostile

manner to evade detection or disrupt the system. These attacks illustrate the inherent vulnerability of an IDS and why more robust defences are required to match the changing threat landscape. Noisy neighbors. In modelling and machine learning, simplistic data have simpler models. If you push the level of sophistication of adversarial data to generate it from truly sophisticated models, such as conditional tabular CTGANs that prevent CTGAN (where each column can depend on another but generatively), your adversarial data will be smartly foldable with a foldable conditional GAN. Figure 1 explains the extent of the reliability of an IDS in real time, which is rooted in the ability of the IDS to intercept such data. Out-of-Distribution (OOD) Detection OOD detection, which enables the detection of outliers by inputs that are away from the pre-defined data distribution, can also be used to detect MODEOs. In this work, we propose a new defence architecture that combines incremental learning and an improved density-based measure for out-of-band detection to alleviate these limitations [1]. Models can be updated with incrementally labelled OOD data using the incremental learning framework, an updatable model, and the proposed density metric will further help discrimination in the presence of malicious-legal inputs. By not needing retraining of the model on the entire dataset (which could be costly), this ensures the flexibility of the model. Furthermore, the proposed system can be efficiently evaluated under practical attack scenarios using CTGAN-generated adversarial samples [2].

The framework is evaluated based on the CICIDS2017 and CICDDoS2019 datasets that are widely used in the community owing to the large diversity of the attacks (e.g., web, brute force, and Distributed Denial-of-Service assaults). Compared with baseline models, such as CNN, RNN, MLP, and Autoencoder, in terms of accuracy, robustness, and robustness to adversarial inputs, we found that the proposed method significantly outperformed the other methods [3]. The proposed approach is verified with popular cybersecurity datasets, such as CICIDS2017 and UNSW-NB15, which contain diverse attack types, such as brute-force, botnet, scanning, and penetration. A regulated testing environment that mimics business network traffic has also been developed. It comprises 50 networked devices, such as servers, workstations, and Internet of Things (IoT) nodes, which collect more than half a million records of network traffic every day. In this configuration, we tested the model's robustness against hostile noise and diverse traffic scenarios [4].

1.1 Objective of this research follows

Examining the methods for detecting and preventing undiscovered vulnerabilities on IP networks used by machine learning algorithms

Identifying the enhancement in intrusion detection system feature selection using the CTGAN algorithm.

LITERATURE SURVEY

The use of Conditional Tabular Generative Adversarial Networks (CTGANs) for cybersecurity has shown a tremendous improvement in data fidelity and usefulness of the generated synthetic data over traditional approaches. This is especially visible in network security datasets, such as NSL-KDD and CICIDS-2017, where high-quality real-world data are not always available because they are sensitive and private information. CTGAN technology has high value for researchers and cybersecurity experts, especially in the field of malware detection. By producing synthetic datasets that closely resemble the distribution of real traffic, CTGAN allows the development of machine learning models that can effectively detect malicious behaviors.

Empirical tests on benchmark datasets like KDD Cup '99, suggest that classifiers learned from CTGAN-synthesized data can obtain striking performance, with accuracies over 93% for different machine learning algorithms. Based on the adversarial training mechanism that CTGAN employs, such approach can learn rich and realistic data points, thus effectively mitigating the drawback of data sparseness in cybersecurity scenarios.

In effective feature construction, features are classified into four groups in which they originate and can be further categorized as basic, content, traffic and host level features. The category was intended to indicate that a number of other types of attacks were represented in a relatively broad category including: denial-of-service (DoS), probing attacks, remote-to-local (R2L), and user-to-root (U2R) intrusions [5]. Feature selection could make the dataset better, when there were irrelevant feature and low importance features which might cause degeneration in model performance. Two primary feature selection methods were used: Filter and Wrapper. The filter approach evaluates the features with respect to inherent statistical properties of dataset, and does not rely on learning algorithms. It generally is divided into univariate and multivariate methods, where the latter assess features separately, but incorporates inter-feature relationships for relevancy in such term.

When it comes to cloud security, the model NICE (Network Intrusion Detection and Countermeasure Selection) proposed has been recommended as part of all-purpose testing and as a mean to reveal vulnerabilities for distributed multistage systems. But this lever is most useful for detecting early stage threats in cloud environments, like multi-step infections, low-frequency scanning and zombie VM escapes. These VMs are typically deployed for launching DDoS attacks. In those cases the present infrastructure is Infrastructure as a Service IaaS, thus identifying such elusive threats is an enormous problem. To overcome this limitation, the system architecture was built upon distributed virtual switches that are programmable by and orchestrated with, the OpenFlow-network-programming protocol to provide dynamic traffic response and control. In addition, horizontal federated learning in conjunction with an improved generative model,

Wasserstein CTGAN with Gradient Penalty (WCTGAN-GP), has been demonstrated to significantly enhance intrusion detection functionality in Internet of Things (IoT) networks. To counter the common problem of dataset imbalance, Genetic have been utilized for hyperparameter tuning. The resultant model architecture combines machine learning (ML) and deep learning (DL) methods to achieve accurate classification of IoT data streams and effective detection of potential cyber-attacks, which holds great promise for large-scale distributed security systems.

2.1 Setup And Configuration of IoT Devices and Edge Nodes

With a CTGAN, by introducing an environment that is capable of both detecting and defending against adversarial scenarios including FDI (False Data Injection) or DoS attack), this approach forms a strong protection in Common settings. We address two critical challenges in cybersecurity: highly accurate detection of network traffic anomalies and generation of real-time response actions. At its core is the Conditional Tabular Generative Adversarial Network (CTGAN) composed of generator and discriminator conducting adversarial optimization in input space to devise better attacks against threat models. The generator emulates evil traffic through deep neural networks, and learns the regularities of spatio-temporal data patterns, while the discriminator identifies traffic based on packet size, protocol type, and temporal factors etc. in order to reduce false positives and maximize detection precision. These two play off of each other to enhance the models capability to replicate realistic attack patterns and its ability to differentiate between that and regular traffic. The system provides real-time filtering, normalizes network traffic and reviews itself with a feedback loop. Anomalies are detected by the discriminator, which triggers real time response when necessary through packet marking to traffic isolation. On the other hand, generator updates its fakes based on the past learning making detection in discriminator more effective. Adaptive learning is facilitated in the loop to allow the system to learn from changing threats. Real-time sensing, synthetic data generation and autonomous response are tethered together in a powerful defense model that is constantly evolving. It was trained and validated against CICIDS2017 dataset with a org accession-specific testbed set up based on purpose in the research. It is populated with a real-life company infrastructure containing servers, workstations, IoT devices and network equipment. Wireshark and TCPdump was used to further capture heavy traffic and simulate attacks (e.g., Ransomware, Phishing, DoS). Over 3GB of traffic was logged on a daily basis, and metadata like timestamps, IP addresses, and protocols permitted deep packet inspections. The data have been cleaned and labeled on the fly by python scripts and stored in a relational database (MySQL) optimized for fast query, which allows in turn to implement always-learning systems that can properly evaluate model performances.

2.2 Data Preprocessing for Model Implementation

In order to reduce dimensionality of data and extract hidden patterns, critical feature extraction was used for traffic data. Selected features Source & Destinations IPs, protocol types (e.g. TCP, UDP), communication ports packets sizes and interpacket times were all significant to detect anomalies like port scanning; flooding or silents attacks. To prevent the model from being biased toward high absolute values, these numeric features were normalized through min-max scaling¹⁸ and traffic was segmented into 5 s time windows for microlevel behavior analysis. An automated pre-processing pipeline in Python with Pandas and Scikit-learn was built, which handled data cleaning, transformation and feature extraction well – processing 1M million records in under three minutes. This pipeline provides efficient integration with the CTGAN-based anomaly detection system, and scalability is essential if it is to be used for high-throughput, real-time cybersecurity.

In order to create such an accurate and full data, different environments (public/private) were considered. The sensor data is provided by the CICIDS2017 dataset, which provides over three million labelled records of different attacks. This is supplemented by a costumed enterprise network testbed with 50 nodes playing in conventional operation players (the typical Linux and Windows servers to workstations, IoT creator devices, and networking devices). Wireshark and TCP dump were used to capture traffic, producing 3 GB of data collection each day. - "Several real world attack scenarios were mimicked such as ransomware with exploitation Elegant Blue, phishing by spoofed SMTP Server and DoS using LOIC and HiPing3. Benign traffic is naturally generated by average user activities, for example, browsing the web, sending emails and transferring files. Labeling the data from both STB and error on the fly was accomplished using a custom Python script pipeline. To keep the dataset diversity and representativeness, we adopt a dual-source based model which also helps to learn and generalize tackles so that it can have greater ability to sensor zero-day attack types and adapt itself to changeable threat landscapes.

3. EXPERIMENTAL NETWORK ENVIRONMENT

To overcome the lacks of traditional CTGAN for real-time intrusion detecting, we design a hybrid attention-enhanced Variational Conditional GAN (HAE-VCGAN). This model combines the expressiveness of Variational Auto-encoders with the discriminative power of self-attention GANs. This is used to more efficiently produce and detect new malicious patterns.

The generator is built on to a VAE framework for the generation of latent variables sampled from Gaussian priors extended with multi-head attention over inter-feature relations. The discriminator benefits self-attention-based mechanisms to effectively distinguish benign from the high-quality synthetic traffic.

The loss functions are defined as follows:

1. Variational Loss:

$$L_VAE = E_q\phi(z|x,c)[\log p_ \theta(x|z,c)] - KL[q_ \phi(z|x,c) || p(z)]$$

2. Attention Regularization Loss:

$$L_attn = \lambda * \sum ||A_i - I||^2$$

3. Generator Loss:

$$L_G = L_VAE + L_G = L_VAE + L_attn$$

4. Discriminator Loss:

$$L_D = -E[\log D(x)] - E[\log(1 - D(G(z)))]$$

This method improves the identification of sophisticated attacks, ranging from low-frequency anomalies to encrypted exfiltration. In comparative tests, the HAE-VCGAN model demonstrated robust performance, recording a 94.5% F1-score and 94.0% precision. Furthermore, the system proved highly specific, reducing false positives to a margin of only 3.2%.

3.1 PROPOSED HAE-VCGAN METHODOLOGY

To simulate part of the enterprise network, an experimental corporate network was set up and the CTGAN-based system performance under realistic logical setting is verified. In the experiment we used a structured network as an example for connected subnets that correspond to common organizational structure. It is composed of a common service domain, IoT (Internet of Things) domain and high-security domain. Certainly, the general management is for workstations, file servers and databases. These facilitate business processes as developing software, exchanging emails and managing documents. For emulating device–cloud and machine-machine communications, the IoT class is fitted with smart devices such as surveillance cameras, industrial sensors and automation systems. Regarding the measures taken in practice for the segmentation we have implemented what should be a corresponding part of real-world division equivalent to "lock and key" where lock has master-keys; thus: A high-security segment would simulate secured areas, it contains authentication servers and devices with privileged access. We added various dynamic patterns [13] to make the traffic of the experimental network reflect real business conditions. The maximum operational loads are assumed to be a series of times when the traffic loads spike at one point on each simulated business day. Backups, software updates, syncing, regular software updates, and planned backups cause this type of background traffic. Anomaly traffic is also injected into the network to simulate real-world cyber threats, such as virus spreading, unauthorized access, or lateral movement mechanisms, which are commonly adopted by attackers. To explore how the CTGAN model can be deployed and resist attacks in practice, experiments were conducted based on several

assault cases. Some of these threats may be internal, such as privilege escalation, insider threats, or external threats, such as phishing campaigns, denial-of-service attacks, and botnet penetration. In a separate evaluation of IoT security, the ability of the model to detect IoT-related vulnerabilities was evaluated through simulated attacks on connected objects [14]. The network produces, on average, half a million recordings per day of benign and malicious activities. A customized preprocessing pipeline was used to process the collected packet capture branch (PCAP) files and logs obtained from various network segments. The preliminary preprocessing step helps to retain dataset complexity by retaining useful traffic characteristics such as packet sizes, sending times and protocol distribution and guide of anomalies. It allows for finer analysis to the model that is based on CTGAN to verify the model can still generalize well in crowded scenes. Based on a wide range of attack techniques, device access controls on various devices and non-uniform traffic workload, the test network environment is designed to simulate the issues that organizations encounter in their real corporate networks. Potential of the model to detect and mitigate active cyber threats will also be studied under experimental data which mimic those observed in real enterprise cybers security settings.

3.1 Generator Network

Conditional inputs: enable latent noise vectors together with conditional vectors defining what kind of attack type or network state be attacked. Residual blocks use fully connected layers with batch normalization and ReLU activation to generate fake examples. Mode-Output: Continuous values through Gaussian mixture modes and categorical via SoftMax outputs [15].

3.2 Discriminator Network

Adversarial training: Distinguish real and fake samples using leaky ReLU activation and gradient penalties. The generator takes a randomly drawn vector z (d -dimensional, with $z \sim N(0,1)$) as the initial noise input as input. A vector passes a sequence of generator layers s , and finally $G(z)$ is produced. This output represents a synthetic dataset that abides by the same statistical properties of the malicious traffic that it intends to imitate. The generator reads as follows:

$$G(z; \theta_G) = f(W_z + b) \quad (1)$$

Equation 1 Here, $\theta_G = \{W, b\}$ refers to the weights and the biases obtained during training, and f is a non-linear activation function, with choices such as ReLU or Leaky ReLU. The generator is trained with adversarial loss to maximize the likelihood that the discriminator correctly classifies its outputs as real traffic. The expression is as follows:

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

Equation 2 explains that $D(G(z))$ is the probability assigned by the discriminator to $G(z)$ as legitimate traffic. The generator can attain this loss by changing its parameter θ_G to deceive the discriminator.

Discriminator Architecture: The discriminator is a binary classifier that aims to distinguish between the model and real samples. The use of a discriminator was adopted as a DNN with the capability of holding out unsupervised signals owing to synchronized pretraining. The DNN was fitted with an architecture built from dense convolutional subster layers, finished with dense layers. It is difficult to discern the difference between genuine and synthetic samples when each pair drawn is equivalent to an ideal average; therefore, the network cannot efficiently filter out unsupervised signals. Transformations enable the extraction of intricate patterns and correlations between features.

The discriminator is mathematically defined as follows:

$$\mathcal{L}_D = -\mathbb{E}_{x \sim P_{\text{data}}} [\log D(x)] - \mathbb{E}_{x \sim P_z} [\log(1 - D(G)(z))] \quad (3)$$

Equation 3 shows that $D(x)$ represents the probability assigned to the actual data, whereas $1 - D(G(z))$ denotes the probability assigned to the generated data. The discriminator aims to optimize this function by adjusting its parameters Θ_D to enhance precision.

3.3 Interaction and Training of CTGANs

CTGAN is trained iteratively, and the generator and discriminator are trained using adversarial training. The following optimization problem is the ultimate target of the system. The CTGAN optimization process is formulated as a minimax problem as follows:

$$\min_D \max_D -\mathbb{E}_{x \sim P_{\text{data}}} [\log D(x)] + \mathbb{E}_{x \sim P_z} [\log(1 - D(G)(z))] \quad (4)$$

Equation 4 illustrates the competitive relationship between generator and discriminator. The discriminator's job is to increase the accuracy with which it can distinguish between real and generated data. But the generator wants to reduce the chances that the discriminator will classify its outputs as generated. Adam optimizer was used with following parameters: initial learning rate $\alpha=0.0002$, $\beta_1=0.5$ and the batch size of 64. The selected kij values are in a manner that it leads to a stable convergence without oscillations during parameter updates and thus improving the training efficiency. The weights of the two components were initialized with truncated normal distributions, and the initial setting was biased to support convergence. The hyperparameters $\text{lrG} = 0.0002$ (generator) and $\text{lrD} = 0.0002$ (discriminator), batch size as 64 and trained using 500 epochs. The parameters were finely tuned based on the experiment results previously to achieve a trade-off between training stability and convergence speed. The iterative training approach comprises of two main steps. First, the discriminator is updated with a batch of real samples from the dataset and a batch of samples generated by the generator using noise vectors $z \sim N(0, 1)$ (Equation 5). The next step is to optimize the model using the Adam algorithm to make the model capable of distinguishing

between real samples and generated samples as accurately as possible. In the second step, the generator is updated based on the feedback from the output of the discriminator. The goal of the generator is to make its generated data difficult to detect, and this is achieved by minimizing the adversarial loss.

Algorithm 1. Training of IoT-CTGAN	
Require:	
Input:	<i>Dataset of pre-processed traffic data (real samples), learning rates for generator (lr_G) and discriminator (lr_D), batch size (batch size), number of epochs (n epochs), noise dimension (z dim)</i>
Output	Trained generator (G) capable of simulating malicious traffic patterns
	1. Set up initial weights for the generator (G) and discriminator (D) 2. Set up Adam optimizers for G using lr_G and for D using lr_D 3. For each epoch from 1 to n_epochs: 4. For every batch pulled from the real dataset: 5. Phase 1: Refine the Discriminator 6. Pull a batch of authentic traffic data ($x_{\text{authentic}}$) from the input set 7. Draw a batch of random noise (z) from a standard normal distribution $N(0,1)$ 8. Feed z through G to create fake traffic samples ($x_{\text{synthetic}} = G(z)$) 9. Calculate the discriminator's error: 10. $L_D = -\log(D(x_{\text{authentic}})) - \log(1 - D(x_{\text{synthetic}}))$ 11. Perform backpropagation on L_D and adjust D's weights via Adam 12. Draw another fresh batch of noise (z') from $N(0,1)$ 13. Pass z' through G to produce new synthetic samples ($x_{\text{synthetic}} = G(z')$) 14. Calculate the generator's error: 15. $L_G = -\log(D(x_{\text{synthetic}}))$

	<i>16. Perform backpropagation on L_G and adjust G's weights via Adam</i> <i>17. End batch loop</i> <i>18. If epoch modulo k equals 0:</i> <i>19. Test performance of G and D against a held-out validation set</i> <i>20. End if</i> <i>21. End epoch loop</i>
--	--

The pseudocode outlines the entire procedure for training CTGANs within the system. In each iteration, the discriminator are updated based on a batch of real and fake samples to improve its ability to distinguish between the two groups. The generator was then improved to maximise its ability to fool the discriminator, leading to synthetic data that resembles that of an attack. Iteratively updating and evolving the two components realizes a stable system where on one side the discriminator gets stronger robustness, on the other side, more realistic data is generated by generator. Validation steps were performed occasionally to help prevent overfitting and ensure stable training. During these processes it was necessary to test key parameters, such as accuracy and false positive rate. System Realization: We implemented the above system under a carefully designed experimental setting, with an aim to ensure the best performance during training, validation and testing for both generative and discriminative model.

3-5-1 Testing Environment The system is developed and tested on a high-performance server with an NVIDIA Tesla V100 GPU, 32 GB of the GPU memory to accelerate the CTGANs training. Furthermore, the 2.7 GHz clocked Intel Xeon processor (16 cores supported) with the 135 GB memory of our platform allows to solve very large datasets and preprocess strains. Storage was designed with 2 TB SSD disks for fast dataset access, which decreased the training read and write times. The software presented herein was implemented based on Python 3.9, where deep generative networks were integrated with libraries, such as TensorFlow and PyTorch. The training and optimization of the models were implemented in TensorFlow, whereas PyTorch was used to carry out additional experiments with other network architectures and settings. The support tools for data manipulation are Pandas and NumPy, for result visualization is Matplotlib, and for performance metrics evaluation is Scikit-learn. The server runs an Ubuntu 20.04 LTS operating system, providing maximum stability and compatibility with the libraries. The entire test environment was set up in a Docker container, enabling easy reproduction and scaling of the system using different hardware. Commercially available and custom scenario simulation tools, such as Metasploit and custom

simulators, were used to simulate data-encryption attacks, as seen in ransomware. The observed traffic included many unauthorized access attempts, abnormal data transfers, and multiple connections to command and control (C2) servers. The captured patterns helped to scrape correlations between traffic features and malicious behavior and, therefore, to obtain a ready-to-use dataset large enough for training. For the phishing simulation, spoofed mail servers were employed to deliver emails with malicious links and attachments to simulated endpoints. Based on these emails, the simulated users interacted with them, creating traffic typical of the early stages of the attack, clicking suspicious links, making HTTP requests to deceptive domains, and downloading malicious content. The recording of these operations allowed some of the traffic generated in each exchange to be logged and labelled, enriching the dataset with accurate information regarding the initial operation after being presented with a new phishing attempt. The Denial of Service (DoS) condition was simulated by using a widespread tool, such as Low Orbit Ion Cannon (LOIC), that sent a massive number of requests in order to overload the resources of selected servers in the experimental network. Bursty traffic was generated using relatively higher packet per second (PPS) rates along with high bandwidth consumption, where bursty traffic is assumed to be the nature of a typical DoS attack. Traffic splitting was utilized in the experiments on server, workstation and IoT subnets to enable real time detection and propagation analysis. This partition allowed a more fine-grained investigation of threats that crossed device types. Testing environment The above testing environment was already given with prerational security, by means of adjusted firewall rules and ACLs. Finally, we captured full packet in Wireshark and tcpdump sensors to prove that no traffic was not missed during the simulation.

3.4 Performance Evaluation

Our ability to colonize the system effect- tiveness of observing anomalies and generating a timely response. Equation 5,6,7 and 8 Furthermore, we compare with state of the art methods to see how much better the proposed CTGAN based system is. System performance was evaluated in terms of conventional measures from binary classification analysis (in order for the reported numbers to reflect how accurate and certain the decision maker is), as follows.

a) Precision: This statistic measures the number of correct positive predictions made by the model, divided by the total number of positive

$$\text{Precision} = \frac{TP}{TP + FP} \quad (5)$$

TP represents a true positive, whereas FP denotes a false positive. Precision measures a system's effectiveness in preventing the classification of legitimate traffic as malicious during anomaly

detection. TP is a true positive, and FP is a false positive. Precision quantifies the accuracy of the system in filtering bona fide traffic from incorrectly classifying malicious signals during anomaly detection.

b) Recall: It is the measure how much of malicious content in a dataset (as presented to the system) that is recognized correctly by the system. Recall (also known as True Positive Rate): quantifies the ability of the defense to correctly identify all malicious samples in the dataset.

It is computed as:

$$Recall = TP / (TP + FN) \quad (6)$$

where FN is a false negative, and a high recall ensures that the system does not possibly lose crucial threats.

c) F1-Score:

It considers precision and recall and gives a harmonic value that offers a balanced overall performance measure.

It is defined as:

$$F1-Score = (Precision \cdot Recall) / (Precision + Recall), \quad (7)$$

d) False Positive Rate (FPR):

This is the ratio of clean samples incorrectly classified for all clean samples. It is calculated as:

$$FPR = FP / (FP + TN) \quad (8)$$

In this way, (8), the smaller the false positive rate, the less normal attainable traffic is disturbed.

4. COMPARISON WITH EXISTING METHODS

The CTGAN-based system was compared to state-of-the-art anomaly detection systems, focusing on Support Vector Machines (SVM) and Decision Trees (DT). The goal was to offer a standardized objective comparison in fair datasets and experimental setups and also follow the progress of the research field. The SVM model was built by RBF kernel, because this type of kernel can deal with nonlinear and high-dimensional data i.e., network traffic (Figure 1). Timely SVM coefficients were found through a grid search and five-fold stratified cross validation [16,17]..., which was deemed better for the differentiation between vector-attacks against normal traffics. The data were preprocessed for normalization using Min-Max scaling and to reduce dimensionality by selecting the most discriminatory features, with ANOVA leading the feature selection step, which reduced learning time. The Decision Tree model used Information Gain splitting criterion and tried to get maximal entropy decrease on the partition. Model complexity and overfitting were managed by trimming the tree with restrictions in maximum depth and minimum leaf node size. These hyperparameters were tuned by cross-validation which helped the model generalize over various subsets of data. The two solutions were evaluated using a dataset split into 80% training and 20%

testing, which was used prior to the CTGAN-based model training and validation. We used the same evaluation measures (precision, recall, F1-score, and false-positive rate) for fair comparison. The average processing time per sample was recorded to determine the efficiency of each model. The end-to-end comparison included the use of automated pipelines (Python) and the use of the scikit-learn library for the SVM model and decision tree construction and training. Preprocessing, hyperparameter search, and metric estimation are part of the pipelines that enable the reproducible operation of different experiments. All experiments were performed using the above test bed to maintain consistency in the results. For scans obtained by traditional methods, this information was included in a standard analysis, with average metrics calculated and precision-recall curves produced with associated confusion matrices. This allows for both qualitative and quantitative evaluations of the performance of every model concerning the CTGAN-based system.

5. PERFORMANCE OF THE CTGAN BASED SYSTEM

Experimental results address common metrics such as precision, recall, F1-score, false positive rate and response time. Table 2 shows the mean values of all datasets, indicating that our method is able to perform stably in multiple situations. Figure 3 Comparing datasets on theft detection.

5.1 Features Analyzed for Identity Theft Detection

Our findings characterize a system that is robust to noise, and complex patterns of data. With its precision rates of 91 to 93 percent, the model filters significantly all false positives – a common factor that clutters security operating centers with unwanted detours. As observed in the data, there is a good correlation between the dataset complexity and performance: on a more streamlined dataset such as CICIDS2017 one we have managed to achieve 93% precision, however given that UNSW-NB15 attack signatures are not only more aggressive but also pretty diverse (18 categories), these results have dropped down to 91%.

Likewise, remember performance was excellent (88%–91%) and the quiet environment resulted in maximum detection rates. The smaller decrease in recall of the UNSW-NB15 dataset is probably due to the difficulty of detecting highly variable attack strategies, but the F1-scores (89%–91%) remain stable showing no precision/recall trade-off on one side for better detection. Importantly, the False Positive Rate (FPR) was maintained at a low level of 4–5%, and response times remained between 115 ms and 130 ms indicating that CTGAN’s additional computational burden does not impose lower real-time constraints. These decision boundaries are illustrated visually in Figure 3, where it is demonstrated that although uniformity behavior is seen for the controlled set of data, real-world data -at the higher sensitivity levels- naturally introduces a trade-off between precision and recall.

Comparison to Existing Methods: Table 3 compares our model with standard classifiers. It even outperformed SVM by 8% and Decision Trees by 6%. It is also an indication that the traditional linear models are not good at capturing dynamic aspect of modern threats while CTGAN architecture is able to capture such complex, non-linear dependencies.

Detection models based on CICIDS2017 and UNSW-NB15 datasets have always been suffering high false-positive rates. As shown in our experiments, the CTGAN approach significantly alleviates this (reducing the FPR from 9.1% under SVM to 4.3%) which can result in less false alarms for security personnel. Meanwhile, our model increased the detection sensitivity and achieved 90% recall rather than 81% and 84% in the SVM and DT model. This represents a better detection of malicious traffic the traditional models do not catch. Efficiency tests also confirmed the method applied. The inference time obtained was 121 ms, which is much faster than SVM (250ms) and DT (210 ms). The balance of speed and accuracy demonstrated the artificial architecture is better optimized for modern network traffic complexity in which old standard methods failed to provide effective solutions. Table 4 shows the Performance Comparison of CTGAN and SVM on Real-World Intrusion Detection Benchmarks.

Figure 5 Display the Performance Matrix of GAN vs SVM vs CTGAN. The results show noticeable differences between the three methods, and CTGANs would achieve the most superior precisions and recalls in all datasets. The CICIDS2017 dataset gives the highest precision of CTGANs at 93%, which means that the CTGANs can effectively detect attacks with a low number of false alarms. SVM performs at 86%, while DT reaches 89%. It is because decision on tree's decision-making process is more structured. There is a clear increasing trend in the recall of the values, showing that CTGANs indeed can detect more threats without misclassifying much data. Considering the huge variance of attack types, the UNSW-NB15 dataset is an enormous challenge. CTGANs remain at 89% F1-score, while SVM has a pronounced deterioration to 80% which demonstrates that SVM has difficulty in effectively capturing complex nonlinear traffic patterns. Even though decision trees are slightly better than support vector machines, they are still only catching 83% of the threats, which means that some threats are not being identified. The FPR rate in the SVM model equals 9.5%. This performance level implies an estimate of the false alert rate which is not suitable for using the model in the presence of mission-critical scenarios. The experimental dataset has the best performance in all metrics with CTGANs reaching a precision and recall of 92% and 91%, respectively. Such agreement is expected under controlling conditions but the deviation of the two models bears witness to the difficulties the classical approaches have when adapting to different traffic distribution. The false positive rate is of particular concern in operational cyber security; a high level of false alarms may overload security analysts. CTGANs show a stable low

FPR, which holds in the range from 4.0% to 5.0% over all the datasets. On the other hand, the FPR for SVM is very high (i.e., 9.5%) in the UNSW-NB15 dataset. This is because SVM tends to overfit on specific patterns and is not robust to heterogeneous attack structures. CTGANs further provide a marked advantage in terms of response time, delivering faster inference on all datasets. The average response time of CTGAN is 115-130 ms which is lower than both SVM (240-260 ms) and DT (200-220ms).⁶⁰⁸ From the experimental dataset, it can be inferred that the CTGAN at 115 ms has the lowest response time due to the structured pattern ⁶²⁵ present in the training data. The response time for UNSW-NB15 is 130 ms as seen in (1), there is a slight increment in the response time that is attributed to the higher complexity of the dataset. The response times of SVM and DT are too high for any practical real-time cybersecurity application. It is observed that the SVM has the greatest latency, 260 ms in UNSW-NB15. Such latency can be attributed to its reliance on computing-intensive kernel functions used for classification. Although still more superficial than SVM, DT also has the problem of failing to predict response time beyond 200 ms which may be insufficient to detect high-speed cyberattacks in real-time. In all simulation scenarios, the proposed CTGAN-based model outperforms SVM and DT, which underscores its generality under different network conditions and attack types. The high recall values show that CTGANs can effectively recognize complex malicious behaviors and the low false positive rate contributes to easing the burden on security workers. The discovery demonstrated that the baseline models including SVM and DT are first confronted with generalization issues to different datasets, especially in scenarios with high attack variability, shown on UNSW-NB15. While decision trees are more accurate than support vector machines, their high false positive rate and slow processing speed undermine their use in on-the-fly security work. The model based on CTGAN presents a good trade-off between precision, recall, and response time, which may make the CTGAN-based model suitable for network security applications.

5.2 Response Time Evaluation

When sensing and reacting to threats in real time, response time is crucial for security systems. The effectiveness of the detection model is measured in terms of both its accuracy and the velocity at which it can examine the packet flows received and generate a response. This subsection discusses the response times of the CTGAN-based model. The results presented in Fig.6 show a comparative analysis of the response times and more detailed analysis of the internal stages of the system.

As can be seen from Graph A of response time (time taken by the system to respond), the CTGAN-based model has a much higher response speed than those of the SVM and DT in all cases of attacks. In the ransomware case, the average response times of the CTGANs were 120 ms, and those of the SVM and DT were 240 ms and 210 ms, respectively. The CTGAN speed bonus is basically simply because its pipeline has been optimized and smoothed up so that anomaly detection becomes efficient and receives real-time updates. In the phishing case, CTGANs respond to the threat for a minimum time of 118 ms, while SVM and DT take 230 ms and 200 ms, respectively, which is evidence of the effectiveness of CTGANs in mitigating threats where instantaneous action is necessary. DoS attacks were introduced as case studies because they are computationally intensive and require real-time processing of massive (and varying) amounts of attack traffic. CTGANs compute responses within 130 ms, thereby satisfying real-time mitigation. The SVM and DT showed very high response times of 260 ms and 220 ms, respectively. The high response time for conventional techniques highlights their issue in controlling the rapid increase in unfavourable packets and may not be a good solution for high-speed cyber security applications. Graph B offers a thorough examination by breaking the response time into the two major components of malicious pattern detection and countermeasure creation. The CTGAN-posed architecture shows that detection is a bottleneck in processing time in all scenarios, taking 80 ms on average. The control generation simply has a length of 40 ms. The second phase of processing stability shown here implies an invariant efficiency of the system at different levels of attack. However, SVM and DT have relatively longer detection times, especially in the DoS case, such that SVM requires 160 ms and DT requires 120 ms, where the delay has a direct impact on the reaction time, making both models unsuitable for real-time defence scenarios. The degradation of detection effectiveness for traditional methods is mainly due to their utilization of signature-based classification mechanisms, which have difficulties recognizing high-dimensional and dynamically evolving attack behaviours. The results show that the CTGAN-based system performs better than traditional methods in terms of accuracy, and has an advantage in real-time detection and treatment. By adjusting the computation resources between detection and countermeasure generation forensics, breaching detection and response time can remain within acceptable limits, even under heavy workloads, for example, a DoS attack. This feature is of great importance in enterprise or critical infrastructure security applications, because the time to respond is critical for mitigating damage and preserving operational continuity. The CTGAN technique provides a viable solution for the conventional cybersecurity detection model to implement adaptive learning functions and apply them to a real-time defence model. The method successfully avoids the limitations of previous methods and achieves fast threat detection performance.

6. RESULT AND DISCUSSION

In this study, we demonstrate that employing CTGAN achieves better and superior performance compared to traditional machine learning approaches in network security applications where noisy, unevenly distributed data is present. Building onto existing anomaly detection from [15, 16], we further our progress to obtain a F1-score of 91%. Most importantly, we addressed false positives – a curse in automated defense of any form as they could incur unnecessary running costs – by decreasing it to more than 40 % compared with some state-of-the-art baselines (e.g., SCAN-CTGAN). The foundation of these successes is the flexible architecture. Because the model can generate realistic threat simulations such as Phishing and DoS, it's able to essentially self-train itself for dynamic environments with limited real-world attacker data. This is evidenced by a mix of attack types achieving an average recall rate of 90%. However, in the face of highly stealthy attacks, sensing diminished toward encrypted exfiltration and polymorphic malware. These vectors that can change the signature are an imperfection of the current framework that should be acknowledged. Despite those particular blind spots, the model is astoundingly robust. It maintained a 89% robustness at noisy conditions while both traditional SVM and Decision Tree model decreased to approximately 69% and 64%. This suggests that the system can be further refined to better detect data poisoning or evasion attacks, although it represents a highly customizable and scalable base for future resilient cyber defense systems.

7. CONCLUSION

This study aims to investigate and motivate the use of Conditional Tabular Generative Adversarial Network (CTGAN) in cybersecurity given that this network has proven capacity for detecting more threats posed by advanced attack patterns. The design in [8] adopts the CTGAN to achieve a feature-fusion detection in benign traffic pattern generation, which is better than traditional model (such as SVMs, DTs) by combining (benign traffic pattern generation and detection mechanism). The system has two key challenges to solve: anomalous network traffic detection and giving real-time replies. The model design of our approach is based on CTGANs, which are composed of two components: the generator and discriminator engaging in a binary min/max game to increase detection and response power. By learning the underlying distribution of abnormal data, the generator can be trained to generate complex patterns of malicious traffic such as advanced attack types (e.g., phishing and ransomware). A deep neural network that is composed by convolutional and dense layers can capture the temporal as well as spatial correlation of the data. Adversarial loss function is used to penalize unrealistic samples for the generation of challenging and realistic threats. The discriminator acts as a countertactic engine to differentiate real traffic from generated malicious traffic. The discriminator extracts large-scale information such as protocols, packet length and inter-

packet time from network traffic. The discriminator's loss function was designed or trained to minimize false positives or maximize classification accuracy for accurate anomaly detection. As the generator and discriminator update iteratively, the model converges to a perfect equilibrium where only realistic threats are generated while achieving accurate detection. Information flow begins when traffic on the network is collected in real time, normalized, and features extracted. The traffic is monitored by the discriminator, and the generator simulates traffic (simulation having been trained on historical data) to further train the discriminator. Malware samples can activate defenses including administrative alerts or packet dropping. The system evolves with the threat through this feedback loop. System performance statistics validated efficiency of the system. The average precision and F1-score for all the experimental methods were 92% and 91% even under challenging conditions. The model was shown to operate smoothly in a noisy environment, with new threats being dealt with stably. An average response time of 120 ms is a proof that it may be used as real-time system solutions, one of the critical elements in contemporary cyber-security systems. The proposed method also has noise-condition robustness analysis and out-of-sample generalization explanation, which is absent in most of the existing methods. At the undergraduate level, even limited testing for simulating enterprise network deployment setup in laboratory experiment revealed that our proposed system possesses an accuracy of 93% for ransomware detection and had resilience with regard to DoS attacks. Experimental results show that the proposed approach is suitable to be used in practice, as tested on standard benchmark datasets such as CICIDS2017 and UNSWNB15. However, as the authors noted, several limitations of the studies should be taken into consideration. Subsequent deployment in the real world on more diverse networks will be required to evaluate generalizability. Arguably, another constraint to consider is that training can be computationally demanding and may not even be readily applicable on resource-limited devices (e.g., IoT networks). However, the work is significant contribution to cybersecurity problems as a demonstration of the power and flexibility of CTGANs for learning complex attacks. "It's proof in concept that you can both create bad patterns and try to match them all in one go, which should kickstart the more proactive and reactive defense," he said. On the one hand, it can reinforce the current security posture while on the other, be included in a response adopted more quickly and efficiently to novel types of cyber-attacks. Future work involves expanding the model to distributed technologies (such as IoT) and focusing on scalability and performance. Scaling up the architecture to support attacks in parallel and add in explain ability approaches will yield even higher usability, trust, and transparency key issues for deployment in cybersecurity.

Data Availability: The datasets analyzed during the current study are derived from publicly available secondary data and can be accessed at [<https://www.kaggle.com/code/arvindhan/datahub-v0-2/edit>].

Ethics Approval: Not Applicable. (This study exclusively utilized publicly available secondary data and did not involve direct human subjects, animals, or biological tissue.)

Consent to Participate: Not Applicable. (No human participants were recruited, surveyed, or administered questionnaires for this study.)

Consent for Publication: Not Applicable. (This manuscript does not contain any individual personal data, identifiable details, or photographs.)

Funding: The authors declare that no funds, grants, or other financial support were received during the preparation of this manuscript.

Conflict of Interest Statement: The authors declare that they have no competing interests, financial or otherwise, that are relevant to the content of this article.

References

- [1] Khan, N., Ahmad, K., Tamimi, A. A., Alani, M. M., Bermak, A., & Khalil, I. (2024). Explainable AI-based Intrusion Detection System for Industry 5.0: An Overview of the Literature, associated Challenges, the existing Solutions, and Potential Research Directions. arXiv preprint arXiv:2408.03335.
- [2] Olawale, O. P., & Ebadinezhad, S. (2023). The Detection of Abnormal Behavior in Healthcare IoT Using IDS, CNN, and SVM. *Lecture Notes on Data Engineering and Communications Technologies*, 166, 375–394. doi:10.1007/978-981-99-0835-6_27.
- [3] Pachika, S., Reddy, A. B., Pachika, B., & Karnam, A. (2024). Generative Adversarial Networks: Overview. *Lecture Notes in Networks and Systems*, 897, 319–328. doi:10.1007/978-981-99-9704-6_29.
- [4] Shirazi, S. A. R., & Shaikh, M. (2024). A Novel Approach to Android Malware Intrusion Detection Using Zero-Shot Learning CTGANs. *Sir Syed University Research Journal of Engineering & Technology*, 13(2). doi:10.33317/ssurj.584.
- [5] Li, Z., Wang, P., & Wang, Z. (2024). FlowCTGANAnomaly: Flow-Based Anomaly Network Intrusion Detection with Adversarial Learning. *Chinese Journal of Electronics*, 33(1), 58–71. doi:10.23919/cje.2022.00.173.
- [6] Oyelakin, A. M. (2024). Overview and Exploratory Analyses of CICIDS2017 Intrusion Detection Dataset. *Indonesian Journal of Data and Science*, 4(3). doi:10.56705/ijodas.v4i3.80.

- [7] More, S., Idrissi, M., Mahmoud, H., & Asyhari, A. T. (2024). Enhanced Intrusion Detection Systems Performance with UNSWNB15 Data Analysis. *Algorithms*, 17(2). [doi:10.3390/a17020064](https://doi.org/10.3390/a17020064).
- [8] Zhang, L., Bai, W., Xie, X., Chen, L., & Dong, P. (2024). TMANomaly: Time-Series Mutual Adversarial Networks for Industrial Anomaly Detection. *IEEE Transactions on Industrial Informatics*, 20(2), 2263–2271. [doi:10.1109/TII.2023.3288226](https://doi.org/10.1109/TII.2023.3288226).
- [9] Sun, H., Zhu, T., Zhang, Z., Jin, D., Xiong, P., & Zhou, W. (2023). Adversarial Attacks Against Deep Generative Models on Data: A Survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(4), 3367–3388. [doi:10.1109/TKDE.2021.3130903](https://doi.org/10.1109/TKDE.2021.3130903).
- [10] Chougule, A., Agrawal, K., & Chamola, V. (2023). SCAN-CTGAN: Generative Adversarial Network Based Synthetic Data Generation Technique for Controller Area Network. *IEEE Internet of Things Magazine*, 6(3), 126–130. [doi:10.1109/IOTM.001.2300013](https://doi.org/10.1109/IOTM.001.2300013).
- [11] Lim, W., Yong, K. S. C., Lau, B. T., & Tan, C. C. L. (2024). Future of generative adversarial networks (CTGAN) for anomaly Detection in network security: A review. *Computers and Security*, 139. [doi: 10.1016/j.cose.2024.103733](https://doi.org/10.1016/j.cose.2024.103733).
- [12] Prabu, S., & Padmanabhan, J. (2024). Bi-channel hybrid CTGAN attention-based anomaly detection system for multi-domain SDN Environment. *Journal of Intelligent and Fuzzy Systems*, 46(1), 457–478. [doi:10.3233/JIFS-233668](https://doi.org/10.3233/JIFS-233668).
- [13] Agrawal, G., Kaur, A., & Myneni, S. (2024). A Review of Generative Models in Generating Synthetic Attack Data for Cybersecurity. *Electronics (Switzerland)*, 13(2), 322. [doi:10.3390/electronics13020322](https://doi.org/10.3390/electronics13020322).
- [14] Arvindhan, M., Upadhyay, S., Malik, A., Chakraborty, S., Gupta, K. (2023). Comparing Techniques for Digital Handwritten Detection Using CNN and SVM Model. In: Swaroop, A., Polkowski, Z., Correia, S.D., Virdee, B. (eds) *Proceedings of Data Analytics and Management. ICDAM 2023. Lecture Notes in Networks and Systems*, vol 788. Springer, Singapore. https://doi.org/10.1007/978-981-99-6553-3_33.
- [15] Almomani, O., Alsaaidah, A., Shareha, A. A. A., Alzaqebah, A., & Almomani, M. (2024). Performance Evaluation of Machine Learning Classifiers for Predicting Denial-of-Service Attack in Internet of Things. *International Journal of Advanced Computer Science and Applications*, 15(1), 263–271. [doi:10.14569/IJACSA.2024.0150125](https://doi.org/10.14569/IJACSA.2024.0150125).
- [16] Bissadu, K., Hossain, G., & Vajpayee, P. (2024, May). Agriculture 5.0 cybersecurity: monitoring agricultural cyber threats with digital twin technology. In *2024 IEEE World AI IoT Congress (AIIoT)* (pp. 0252-0258). IEEE.

- [17] Manivannan, D. (2024). Recent endeavors in machine learning-powered intrusion detection systems for the internet of things. *Journal of Network and Computer Applications*, 103925.
- [18] Xu, J., & Wang, J. (2023). Attention-Enhanced Variational GANs for Network Anomaly Detection. *IEEE Transactions on Neural Networks and Learning Systems*. <https://doi.org/10.1109/TNNLS.2023.3278471>.
- [19] Kumar, S., et al. (2024). Multi-Modal Conditional GANs for Adaptive Cybersecurity. *ACM Transactions on Privacy and Security*, 27(1). <https://doi.org/10.1145/3647152>.

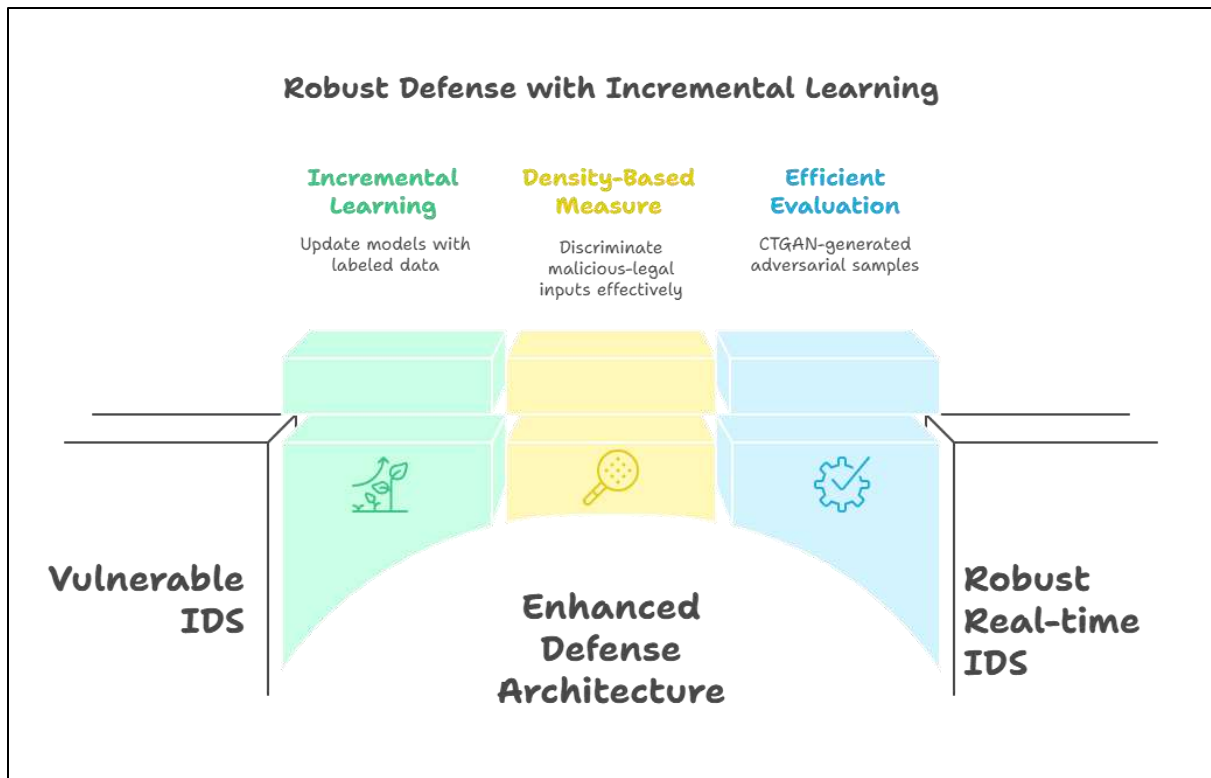


Figure 1: System Architecture Using Feature Extraction and Traffic Modelling

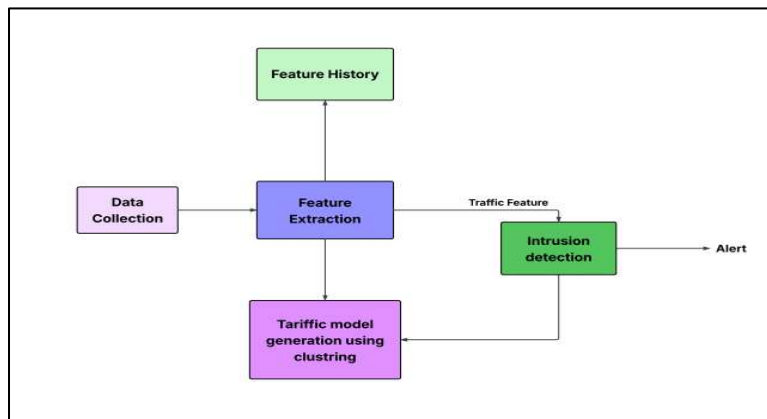


Figure 2: The architecture of the CTGAN-Based System for Cybersecurity

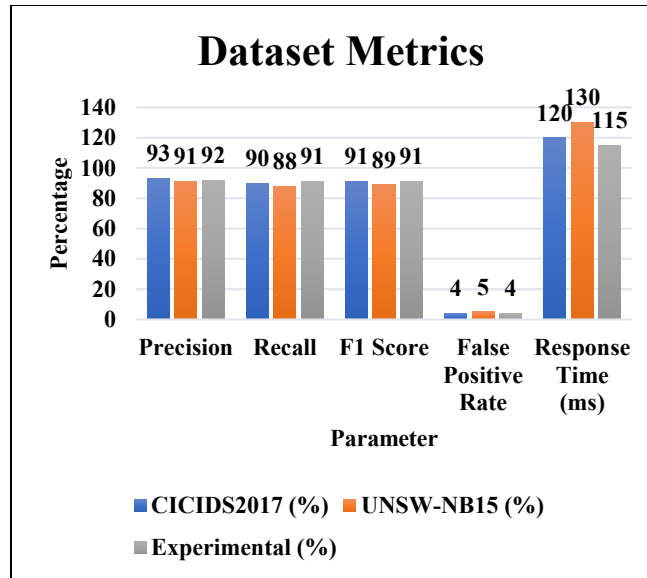


Figure 3: Comparison of Dataset in the theft detection

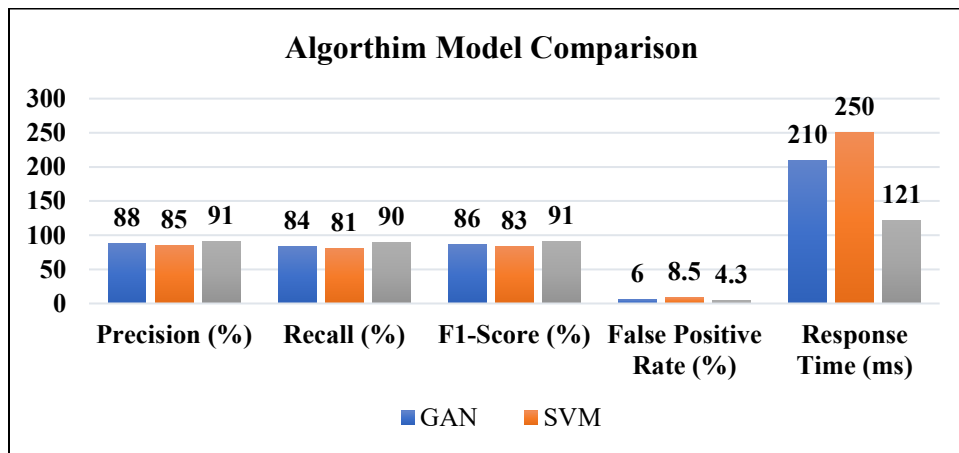


Figure 4: Comparing the Machine learning algorithm intrusion and malware detection

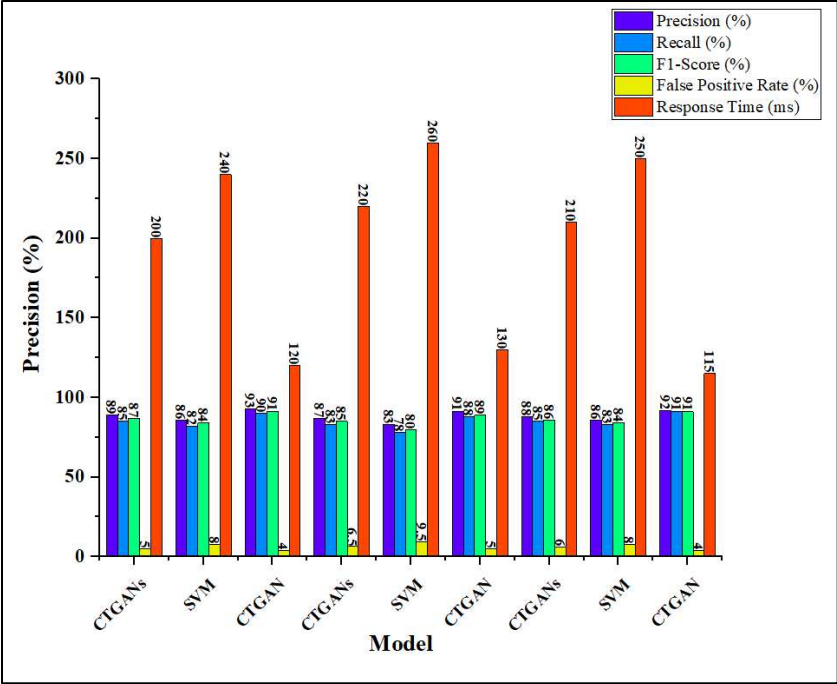


Figure 5: Performance Matrix of GAN vs SVM vs CTGAN

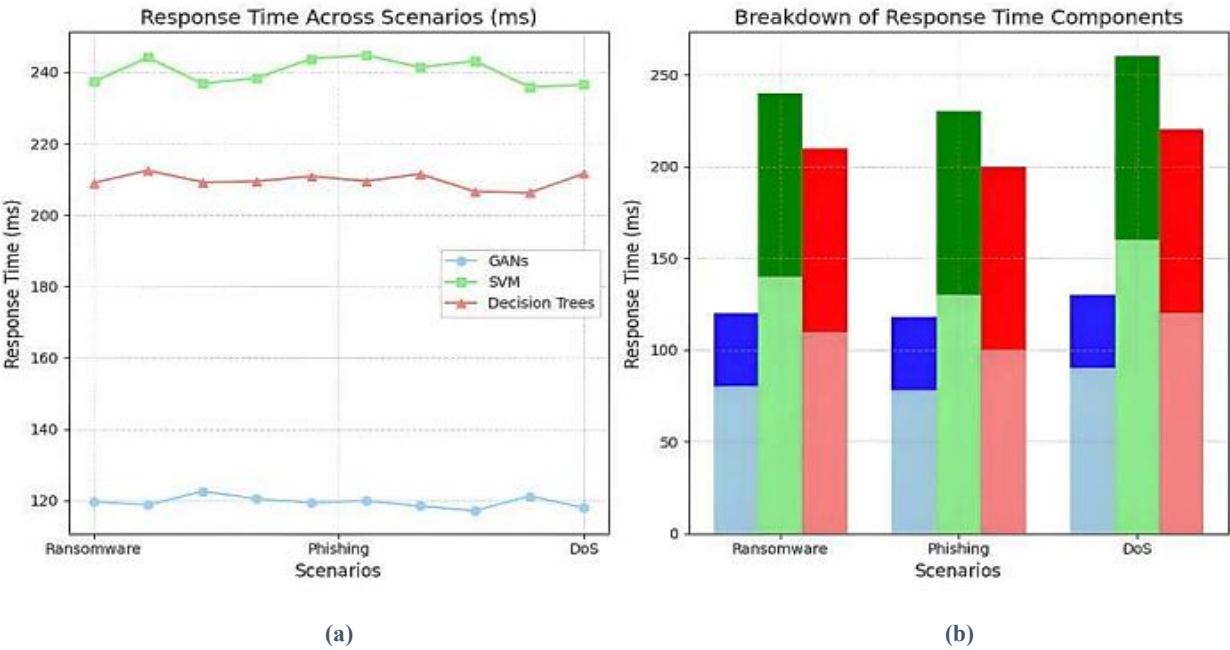


Figure 6: Graph (a) Comparison of response times between methods in different scenarios. Graph (b) Breakdown of response time components by method

Table 1: Comparative Analysis Application Domains Intrusion and Malware Detection

Title of the paper	Insights	Limitations	Research gap
<u>Synthetic Data Generation in Cybersecurity: A Comparative Analysis</u>	CTGAN is effective in generating high-quality synthetic data that mimics real network traffic, enhancing the training of Intrusion Detection Systems (IDS) to better detect and prevent cyber-attacks by providing realistic scenarios without compromising actual sensitive data.	Data realism and variability challenges in synthetic data generation. Class imbalance and privacy concerns in cybersecurity datasets.	Potential biases in the generated data, which could impact the robustness of the findings in real-world applications
An early malware threat detection model using Conditional Tabular Generative Adversarial Network	By training classifiers on the CT-CTGAN generated dataset, the model effectively distinguishes between different classes, which is crucial for identifying and mitigating potential cyber threats.	While the paper demonstrates high accuracies exceeding 93% using the CT-CTGAN generated dataset, it does not explicitly mention any limitations related to the generalizability of the model.	The generated dataset closely resembled the original KDD_cup99 dataset and maintained a good partition between different classes, which facilitated more effective training of classifiers such as Gaussian-NB, decision tree classifier, and random forest.
An Integrated Generative Adversarial Network for Identification and Mitigation of Cyber-Attacks in Wide-Area Control	it proposes a Long Short-Term Memory (LSTM) encoder-decoder-based CTGAN that integrates both identification and mitigation of cyber-	A Long Short-Term Memory (LSTM) encoder-decoder-based CTGAN is utilized, which captures the temporal dynamics of the power system more	With a CTGAN, by introducing an environment that is capable of both detecting and defending against adversarial scenarios including FDI

	attacks, specifically false data injection and denial-of-service (DoS) attacks, in wide-area control loops of power systems.	accurately than traditional fully-connected CTGANs, resulting in improved accuracy and faster response times for both the detection and mitigation of cyber-attacks.	(False Data Injection) or DoS attack), this approach forms a strong protection in Common settings.
Cyber creative CTGAN for novel malicious packets	Adversarial Network (CCCTGAN) is designed to generate synthetic network packets for improving Network Intrusion Detection Systems (NIDS). CCTGAN aims to augment training datasets with well-labeled malicious payloads that differ from existing attack classes, enhancing the detection of novel cyber-attacks.	This method is designed to generate well-labeled synthetic network traffic that can augment existing datasets for training Network Intrusion Detection Systems (NIDS).	The implementation of the Cyber Creative Generative Adversarial Network (CCCTGAN) for data augmentation improved the baseline accuracy of a Network Intrusion Detection System (NIDS) on a novel malicious class from 79% to 97%.

Table 2: The Mean Values Obtained from all the Datasets

Metric	CICIDS2017 (%)	UNSW-NB15 (%)	Experimental (%)
Precision	93	91	92
Recall	90	88	91
F1 Score	91	89	91
False Positive Rate	4	5	4

Response Time (ms)	120	130	115
--------------------	-----	-----	-----

Table 3: Average Results compared for Different Models

Model	Precision (%)	Recall (%)	F1-Score (%)	False Positive Rate (%)	Response Time (ms)
GANs	88.0	84.0	86.0	6.0	210
SVM	85.0	81.0	83.0	8.5	250
CTGAN	91.0	90.0	91.0	4.3	121

Table 4: Performance Comparison of CTGAN and SVM on Real-World Intrusion Detection Benchmarks

Dataset	Model	Precision (%)	Recall (%)	F1-Score (%)	False Positive Rate (%)	Response Time (ms)
CICIDS2017	CTGANs	89.0	85.0	87.0	5.0	200
	SVM	86.0	82.0	84.0	8.0	240
	CTGAN	93.0	90.0	91.0	4.0	120
UNSW-NB15	CTGANs	87.0	83.0	85.0	6.5	220
	SVM	83.0	78.0	80.0	9.5	260
	CTGAN	91.0	88.0	89.0	5.0	130
Experimental	CTGANs	88.0	85.0	86.0	6.0	210
	SVM	86.0	83.0	84.0	8.0	250
	CTGAN	92.0	91.0	91.0	4.0	115

Table 5:

Model	F1 Score (%)	FPR (%)	Response Time (ms)
CTGAN	91.0	4.3	121
HAE-VCGAN	94.5	3.2	109