Journal of
Sensor and Actuator Networks

MDPI

Article

Create a Realistic IoT Dataset Using Conditional Generative Adversarial Network

Miada Almasre * **and Alanoud Subahi**

Department of Information Technology, Faculty of Computing and Information Technology, King Abdulaziz University, Jeddah 21589, Saudi Arabia; asubahi@kau.edu.sa

* Correspondence: malmasre@kau.edu.sa

Abstract: The increased use of Internet of Things (IoT) devices has led to greater threats to privacy and security. This has created a need for more effective cybersecurity applications. However, the effectiveness of these systems is often limited by the lack of comprehensive and balanced datasets. This research contributes to IoT security by tackling the challenges in dataset generation and providing a valuable resource for IoT security research. Our method involves creating a testbed, building the 'Joint Dataset', and developing an innovative tool. The tool consists of two modules: an Exploratory Data Analysis (EDA) module, and a Generator module. The Generator module uses a Conditional Generative Adversarial Network (CGAN) to address data imbalance and generate high-quality synthetic data that accurately represent real-world network traffic. To showcase the effectiveness of the tool, the proportion of imbalance reduction in the generated dataset was computed and benchmarked to the BOT-IOT dataset. The results demonstrated the robustness of synthetic data generation in creating balanced datasets.

Keywords: Conditional Generative Adversarial Network (CGAN); Internet of Things (IoT); machine learning; Traffic Analyzer Tool; synthesize datasets; network attack scenarios

check for updates

Citation: Almasre, M.; Subahi, A. Create a Realistic IoT Dataset Using Conditional Generative Adversarial Network. *J. Sens. Actuator Netw.* **2024**, *13*, 62. https://doi.org/10.3390/jsan13050062

Academic Editors: Donald Elmazi and Elis Kulla

Received: 30 August 2024
Revised: 27 September 2024
Accepted: 1 October 2024
Published: 3 October 2024

1. Introduction

In recent years, technologies such as the Internet of Things (IoT), Cloud Computing, and Artificial Intelligence (AI) have become increasingly popular. The rapid proliferation of IoT devices, particularly in the context of smart homes, has led to a significant increase in the number of devices connected to the Internet, resulting in a substantial rise in the volume of data generated by these devices [1]. Smart homes utilize a variety of technologies and standards, which can often be incompatible. However, this diversity presents new security vulnerabilities, leaving them vulnerable and making IoT environments attractive targets for cyberattacks.

IoT applications often rely on lightweight communication protocols and devices with limited computing and storage capabilities. This makes it challenging to implement security measures that require high computational power. Due to their unique characteristics, traditional security tools such as firewalls and encryption may not perfectly fit IoT systems. For instance, it is difficult to detect attacks like DoS, MITM, and password cracking using traditional programming methods. Security is a critical component in safeguarding IoT environments by identifying malicious activities and preventing potential breaches. Therefore, researchers have improved security mechanisms by employing advanced detection techniques, such as machine learning based on Intrusion Detection System (IDS) ML-based IDS methods, which utilize various algorithms to distinguish between normal and malicious behaviors by classifying the data based on selected features [2–4]. However, previous studies often overlook the challenge of data imbalance, which tends to bias classification models towards dominant classes with more attack samples. In smart home environments, for example, this significantly impacts the performance of the often used security applications and IDS. Amongst these are firewalls and Security Information and Event Management (SIEM) systems, which might fail to accurately detect and correlate sporadic attack types, resulting in false detection instances or alarms. In a similar manner, Endpoint Detection and Response (EDR) applications and anomaly detection systems, which recognize instances of anomalies in the context of normal behavior, may disregard less frequent but nonetheless dangerous anomalies if such cases are underrepresented in the data. Implemented smart home behavioral analytics solutions, usually used to collect user behavior data, might also be unsuccessful in detecting advanced insider attacks or vulnerable devices because of the lack of various attack scenarios in the trained dataset.

To address this issue, more recent approaches emphasize generating balanced datasets, ensuring that classification models perform accurately across all classes, leading to more reliable detection of both common and rare attack types [1].

In recent years, Generative Adversarial Networks (GANs) have emerged as a powerful tool for generating synthetic data samples for minor classes, particularly in addressing class imbalance issues in datasets. In some studies, these generative models are used as classifiers by training the discriminator differently.

This research builds on previous studies highlighting the importance of synthetic data generation in cybersecurity. For instance, Kumar and Sinha [5] utilized a Wasserstein CGAN model to generate synthetic IDS datasets, demonstrating significant improvements in detection performance. Similarly, Liu et al [6] introduced GAN-FS, a GAN-based oversampling technique combined with feature selection, which effectively addressed class imbalance and high dimensionality in IDS datasets. These studies underscore the potential of GANs in enhancing IDS by providing high-quality synthetic datasets. The proposed CGAN-based tool called "Synthetic Data Generator Tool" is designed to generate realistic and balanced synthetic datasets for IDSs in IoT environments. The objectives of the proposed Synthetic Data Generator Tool are not only to address the class imbalance issue but also to offer a comprehensive statistical dashboard for data analysis. This feature is particularly beneficial for researchers who want to understand the underlying patterns and distributions within their datasets. Furthermore, the tool incorporates advanced feature engineering techniques to rank the importance of features, thereby aiding in the development of more robust IDS models.

The objectives and contributions of this paper can be summarized as follows:

1. Development of a novel Synthetic Data Generator Tool capable of generating realistic and balanced synthetic data which can be used in cybersecurity applications to detect network attack types.
2. Building a multi-class IoT network attack type dataset that includes six attack types and a normal class.
3. Development of a user-controlled Synthetic Data Generator Tool capable of generating realistic and balanced datasets which can be used in cybersecurity applications to detect network attack types.
4. Constructing a comprehensive pre-processing pipeline that addresses challenges in synthetic data generation using CGANs.
5. Evaluating the generated datasets' balance.

The subsequent sections of this paper discuss the literature review in Section2, whereas Section 3 details the methodology used to collect and analyze IoT network traffic data and the design and implementation of the Synthetic Data Generator Tool. Section 4 discusses the results of using the Synthetic Data Generator Tool and evaluating the synthetic datasets generated by the tool. Finally, Sections 5 and 6 conclude the research and present limitations and future work.

2. Literature Review

This section provides a comprehensive understanding of current advancements in datasets in IoT environments; we divided the research into two subsections. Section 2.1 provides an in-depth analysis of the methodologies used to gather datasets for both benign and malicious activities in IoT environments. Section 2.2 examines the latest research focused on generating synthetic datasets using GANs, detailing how these techniques improve dataset quality and performance by simulating diverse and representative attack scenarios.

2.1. Comprehensive Review of Public IoT Datasets

In this subsection, a comparative review of various publicly available IoT datasets is conducted, evaluating them from multiple perspectives, such as data composition, feature types, and applicability. The discussion aims to highlight the strengths and weaknesses of each dataset, providing insights into their suitability for different IDS applications. Table 1 summarizes the key characteristics of these datasets.

A recently published dataset by Tomás Sureda Riera et al. [7] focuses on the classification of web attacks using CAPEC (Common Attack Pattern Enumeration and Classification) with machine learning techniques. This multi-label dataset contains data specific to web attack types such as SQL Injection (SQLI), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF), which are critical for evaluating IDSs in environments deploying web servers and applications. Given the increasing frequency of web-based attacks in IoT devices that often deploy web servers, this dataset provides valuable insights into identifying and mitigating web attacks using advanced classification techniques. By incorporating this dataset, researchers can improve IDS models to handle both traditional network attacks and complex web-based threats, which are underrepresented in many existing IoT-focused datasets.

The 5G-NIDD dataset [5] is created to assess the performance of network IDSs in 5G networks. It contains network traffic data from a fully operational 5G test network, capturing both normal activities and nine types of cyberattacks. The dataset consists of 112 features derived from network traffic, such as flow-based features, packet details, and statistical attributes. However, using the dataset requires significant storage and computational resources. Additionally, the high dimensionality of features can complicate analysis and model training.

The TON_IoT dataset [4] was designed to evaluate IoT IDSs and includes network traffic data, operating system logs, and telemetry data from IoT devices. It captures normal activities and various types of attacks in a realistic IoT environment. However, its large size (∼80 GB) and the variety of features from multiple sources complicate analysis and model training.

The IoT-23 dataset [8] consists of network traffic from 23 different IoT devices, containing both benign and malicious scenarios. The dataset, which is approximately 50GB in size, offers diversity in device types and attack coverage, such as Mirai and Gafgyt attacks. However, due to its volume and labeling complexity, analysis may require significant computational resources, similar to other large datasets.

The MQTT-IoT dataset [9] captures network traffic from IoT environments using the MQTT protocol, reflecting real-world communication scenarios. While the dataset is smaller (∼5 GB) than others, it still requires considerable resources for processing. It focuses on attacks relevant to MQTT communications, which may limit its generalizability to other IoT environments.

The Bot-IoT dataset [10] focuses on IoT botnet activities and contains data from various botnets such as Mirai and Bashlite. This dataset, though comprehensive in its attack coverage, includes synthetic scenarios that may not fully replicate real-world conditions. Its large size (∼69 GB) and high feature count require significant computational resources.

The UNSW-IoT dataset [11], which also has a size of approximately 50GB, was created to evaluate IDS in IoT environments. Like IoT-23, this dataset includes a wide range of attacks and feature sets. Both datasets, due to their large volumes and complex feature sets, present similar challenges for analysis and resource demands.

The UNSW-IoT Trace dataset [12] is a subset of the UNSW-IoT dataset that focuses on detailed packet-level information. It provides packet-level features such as timestamps, IP addresses, port numbers, and protocol attributes, offering insights into IoT traffic patterns and behaviors under both normal and attack conditions. However, being a subset of the larger UNSW-IoT dataset, it has a smaller scope (∼10 GB) but still requires significant computational resources for detailed analysis.

The CICIDS2017 dataset [13] is intended for use in network intrusion detection and prevention systems. It contains various types of attacks, including DDoS, brute force, and botnet, as well as normal traffic, to replicate real-world scenarios. The dataset comprises 80 network traffic features, but its large size (∼70 GB) and high dimensionality make feature selection and model training more complex. Additionally, class imbalance issues can affect model performance.

The MedBIoT IoT dataset [14] contains network traffic data from IoT devices that have been targeted by various types of cyberattacks, including those initiated by the MedBIoT, to evaluate real-world scenarios from IoT environments, providing detailed packet-level and flow-level features for thorough analysis. However, its significant data volume (∼30 GB) can pose challenges in processing and analysis.

The IoTID20 dataset [15] is focused on IoT device identification and anomaly detection. It contains network traffic data from various IoT devices, offering a comprehensive dataset for device identification and IDS testing. The dataset includes over 80 features related to network traffic, such as flow statistics, packet sizes, and protocol-specific attributes. However, its large dataset size (∼100 GB) and high dimensionality complicate analysis and model training, particularly for diverse device types.

In addition to the datasets discussed, the Web Attack Dataset of Sureda Riera et al. [16] presents a valuable resource for evaluating the detection of web-based attacks. This dataset uses machine learning techniques for the classification of web attacks based on the CAPEC (Common Attack Pattern Enumeration and Classification) taxonomy, which is particularly useful for assessing vulnerabilities in web applications. While IoT devices often deploy web servers and applications, web attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) are increasingly common. The inclusion of such datasets would complement the current focus on Distributed Denial of Service (DDoS) attacks and other network-based threats covered in this work. Additionally, OWASP's catalog of web application vulnerabilities highlights the growing need for securing IoT web applications. This reinforces the relevance of integrating web-based attack detection into IoT environments, as web attacks have become one of the most frequent and dangerous threats to IoT ecosystems.

Table 1. A comprehensive comparison of major IoT device datasets from different perspectives.

Dataset	Year	Description	Size	Features	Labels	Strengths	Limitations
CAPEC Web Attacks	2022	Multi-label dataset classifying web-based attacks (SQLI, XSS, CSRF) using CAPEC classification; focuses on web server vulnerabilities and web attacks prevalent in IoT environments	∼5 GB	CAPEC attack patterns, including web-based attacks (SQLI, XSS, CSRF)	Normal traffic and web attack labels	Valuable for IDS development against web threats in IoT, underrepresented in IoT datasets	Limited to web-based attacks
5G-NIDD	2022	Data from a functional 5G test network, capturing normal and malicious activities	∼5 GB	112 features, including flow-based, packet, and statistical attributes	Normal traffic and nine attack types	Realistic 5G traffic, detailed feature set; generated dataset for testing missing attacks	Large size, requires significant resources

Table 1. *Cont.*

Dataset	Year	Description	Size	Features	Labels	Strengths	Limitations
TON_IoT	2020	IoT network traffic, OS logs, and telemetry data	∼80 GB	Multiple features across network traffic, log data, and telemetry data	Normal and malicious activities	Realistic environment, diverse data sources	Large size, complexity of analysis
IoT-23	2020	Network traffic from 23 IoT devices with both benign and malicious scenarios	∼50 GB	Network traffic features including packet details, flow statistics, protocol-specific attributes	Normal traffic and different types of attacks	Device diversity, comprehensive attack coverage	Large size, labeling complexity
MQTT-IoT IDS	2020	Network traffic data from MQTT environments	∼5 GB	Packet details, flow characteristics, MQTT-specific attributes	Normal traffic and various attack types (DoS, scan, brute force)	Realistic traffic, MQTT focus	Smaller dataset, protocol specificity
Bot-IoT	2019	IoT network traffic dataset capturing botnet activities	∼69 GB	Over 50 features	Different attack types and normal traffic	Comprehensive attack coverage, generated dataset used for modeling certain botnet behaviors	Synthetic scenarios, large size
UNSW-IoT	2019	IoT network traffic capturing normal and attack scenarios	∼50 GB	Flow-based, statistical, and protocol-specific features	Normal traffic and various attack types (DDoS, reconnaissance, etc.)	Comprehensive feature set, realistic traffic	Large size, complexity of analysis
UNSW-IoT Trace	2018	Subset of UNSW-IoT dataset with packet-level details	∼10 GB	Packet-level features including timestamps, IP addresses, and port numbers	Normal traffic and attack types	Detailed packet-level data	Limited scope
CICIDS2017	2018	Network traffic dataset for intrusion detection systems	∼70 GB	80 network traffic features	Normal traffic and various attack types	Realistic traffic, comprehensive data	High dimensionality, class imbalance
IoT Dataset by MedBIoT	2018	Network traffic from IoT devices under attack	∼30 GB	Packet-level details, flow features, and statistical measures	Normal traffic and different attack types	Real-world scenarios, detailed traffic analysis	Large size, diverse attack types
IoTID20	2017	Network traffic for IoT device identification and anomaly detection	∼100 GB	Over 80 features	Different device types and anomaly types	Covers a wide range of IoT devices, labeled anomalies	Large size, high dimensionality

2.2. Recent Advances in Synthetic Dataset Generation for IoT Cybersecurity Applications

In recent years, Generative Adversarial Networks (GANs) have gained traction as a solution for addressing the limitations of traditional (IDS) datasets, particularly in terms of class imbalance and data scarcity. Several GAN-based approaches have been proposed to generate synthetic data for training IDS models, with varying levels of success and limitations. This section provides an overview of these approaches and highlights their shortcomings.

2.2.1. GAN-Based Approaches for IDS Dataset Generation

Wu et al. [17] proposed a GAN-based method for generating adversarial examples to enhance the detection of jamming attacks in wireless networks. Their approach leverages GANs to simulate adversarial scenarios, enabling intrusion detection systems to become more robust to hard-to-detect network-based attacks. This study represents a step forward in applying GANs to generate realistic network traffic for testing IDS robustness, making it highly relevant for IoT cybersecurity applications.

Kumar and Sinha [1] introduced a Wasserstein Conditional Generative Adversarial Network (WCGAN) combined with an XGBoost classifier, achieving high F1 scores on multiple datasets. However, their approach utilized a combination of real and synthetic data for training rather than relying solely on generated samples, potentially affecting the generalization capabilities of the generated datasets. Moreover, their feature reduction technique, while effective in certain datasets like NSL-KDD, struggled with maintaining high performance in others, such as UNSW-NB15, due to a lack of feature standardization across datasets.

Ranka et al. [18] introduced a GAN-enhanced approach for malware detection, focusing on generating synthetic data for underrepresented attack types. Their work demonstrates the capability of GANs to augment cybersecurity datasets for multi-class classification, improving detection rates for attacks that are traditionally hard to detect due to their scarcity in training datasets. This method complements existing approaches by addressing the challenge of data imbalance and showcasing GAN's versatility in cybersecurity applications.

Strickland et al. [19] introduced an approach using Conditional GAN (CTGAN) and CopulaGAN for synthetic dataset generation in combination with deep reinforcement learning (DRL). The study achieved a relatively high accuracy of 0.85 with CTGAN. However, their methodology lacked feature selection, which limited the ability to optimize model performance. Moreover, they did not reduce training and testing times, leading to potential scalability issues in large-scale environments. One of the key limitations of this work is that the generated synthetic data were used solely for training, while the real data were used for testing, leaving unanswered questions about the effectiveness of the generated data in real-world applications. Without evaluating the generated dataset independently, it is unclear if it could fully replace real data for model training and testing.

Alabdulwahab et al. [3] employed CTGAN to generate synthetic IoT network traffic based on pre-existing datasets such as TON-IoT, MQTT-IoT-IDS2020, and BoT-IoT. While their model effectively simulated real IoT traffic and attack scenarios, it relied on augmenting these pre-existing datasets rather than generating new, distinct traffic patterns. Additionally, while they employed feature selection techniques to enhance the model's efficiency, their approach did not account for real-time applicability, a crucial factor in IoT environments that require swift detection and mitigation of attacks.

Liu et al. [6] introduced GAN-FS, a novel oversampling technique combining GAN with feature selection to address the issues of class imbalance and high dimensionality in IDS datasets. They tested GAN-FS on the NSL-KDD, UNSW-NB15, and CICIDS-2017 datasets, achieving significant improvements in accuracy, recall, and F-measure compared to other methods. However, their approach also exhibited certain limitations. While feature selection improved detection performance, it did not always yield better results for high-performing models, sometimes disrupting optimal feature combinations. Furthermore, the method's effectiveness varied across different classifiers, with tree-based models and neural networks benefiting more than others. GAN-FS performed well in balancing the datasets but may not generalize across all classifier types, limiting its application in diverse environments.

Dina et al. [20] explored the use of WCGAN combined with an XGBoost classifier to address the problem of data imbalance in IDS datasets. Their experiments on the NSL-KDD, UNSW-NB15, and BoT-IoT datasets demonstrated significant improvements in precision, recall, and F1 score when using a mixed dataset (generated and real data combined). However, the False Alarm Rate (FAR) improvement was modest, particularly for the NSL-KDD dataset. While their approach was effective in enhancing detection capabilities for minority attack classes, the model's generalizability to other types of network traffic remains uncertain. Additionally, the computational cost of training GAN models, particularly WCGAN, poses a significant challenge, especially in real-time IDS deployment.

2.2.2. Non-GAN-Based Toolkit for Dataset Generation

In contrast to GAN-based methods, Vasilomanolakis et al. [21] introduced the ID2T toolkit, which creates labeled datasets for evaluating IDS models by addressing the limitations of outdated or incomplete datasets. The ID2T toolkit allows users to generate datasets with user-defined attacks, enhancing the flexibility and relevance of the synthetic data. The toolkit architecture consists of four core modules: statistics, packet splitter, attack controller, and merger. It utilizes two main techniques for attack generation: script-based generation for attacks with similar parameters and pcap modification for more complex attacks.

The ID2T tool processes large network packet capture files (pcap) to generate realistic background traffic and inject various network attacks, including resource exhaustion attacks (e.g., TCP-SYN DoS, DDoS) and specific malware (e.g., Angler malware, Aurora exploit). While the toolkit offers considerable flexibility for generating realistic network scenarios, several limitations persist. The tool requires manual intervention for certain functionalities, and the authors noted the need for new metrics to comprehensively measure the quality of the synthetic datasets. Additionally, the toolkit requires further enhancements to automate attack generation processes and support IPv6 environments.

While the above GAN-based methods have advanced the field of IDS dataset generation, they exhibit several limitations:

- Dependence on Real Data: Most studies, including Kumar and Sinha [1] and Dina et al. [20], depend on a combination of real and synthetic data for training. This reliance can introduce bias and limit the generalizability of IDS models when exposed to new or unseen attack patterns.
- Limited Dataset Representation: Studies such as Alabdulwahab et al. [3] and Strickland et al. [19] focus on generating synthetic data based on pre-existing datasets. This approach can restrict the variety of attack scenarios simulated, as the generated data may only replicate patterns already present in the original datasets. Consequently, these methods may fail to capture novel or evolving IoT threats.
- Lack of Focus on Real-Time Applicability: Many existing studies, such as those by Strickland et al. [19] and Alabdulwahab et al. [4], do not account for the computational demands of real-time data generation and analysis. IoT environments require immediate response times, and the lack of optimization for speed and efficiency in data generation limits the practical use of these methods in real-world applications.
- Feature Selection and Optimization: GAN-based methods like GAN-FS [6] integrate feature selection but sometimes disrupt optimal feature combinations. While feature selection can improve performance in certain contexts, it does not always yield better results, particularly for high-performing models. Additionally, methods without feature selection (e.g., Strickland et al. [19]) risk inefficiencies, leading to longer processing times.

Unlike previous studies, our research focuses on addressing the above limitations by:

- Generating Fully Synthetic Datasets: Our approach eliminates the reliance on real data by generating entirely synthetic datasets. This ensures that the model is trained on balanced, unbiased data, improving its ability to generalize to new and previously unseen IoT traffic patterns.
- Attack Scenario Generation: We go beyond augmenting pre-existing datasets by generating new IoT attack scenarios that are underrepresented in current datasets.

This allows our model to train on a more diverse set of threats, which enhances its applicability in real-world IoT environments.

- Real-Time Applicability: Our method is designed with real-time IoT environments in mind, optimizing both the generation and processing of synthetic data. This ensures that the model can respond to network threats swiftly, making it suitable for real-time IDS deployment.
- Integrated Feature Selection: We incorporate advanced feature selection techniques that streamline the model's performance, reducing the computational overhead while maintaining or improving detection accuracy. This makes our approach more efficient and scalable, particularly for large IoT networks.

3. Methodology

In this section, the researchers explain the methodology used to collect, analyze, and extract the features for creating the dataset used for this research. We divide it into five Subsections: Section 3.1 describes in detail the network configuration testbed of the smart home. In Section 3.2 we explain the attack scenarios implemented against the IoT devices to collect and analyze the pcapng traffic. In Section 3.3 we discuss the creation process of our own dataset named "Joint Dataset". In addition, in Section 3.4 we describe the BoT-IoT dataset that is used in our experiment as a benchmark for synthetic data generation evaluation. Finally, in Section 3.5, we describe the Synthetic Data Generator Tool, as well as the architecture of the CGAN model used for synthetic data generation.

3.1. Network Configuration

To collect realistic data for network attack types in a smart home context that uses IoT devices, we configured a testbed. Figure 1 illustrates the configuration of the smart home testbed used for gathering and generating genuine network traffic. The testbed comprises four IoT devices, two smart cameras, one smart plug, and one smart lamp, strategically positioned across five crucial areas. To ensure internet connectivity for the IoT devices, we have set up a Kali Linux laptop as a gateway and connected the devices to the internet via IoT apps using the Kali hotspot. This setup allows us to conduct tests on potential attacks from both inside and outside the smart home network while also closely monitoring and analyzing network traffic between the IoT devices and the IoT cloud.

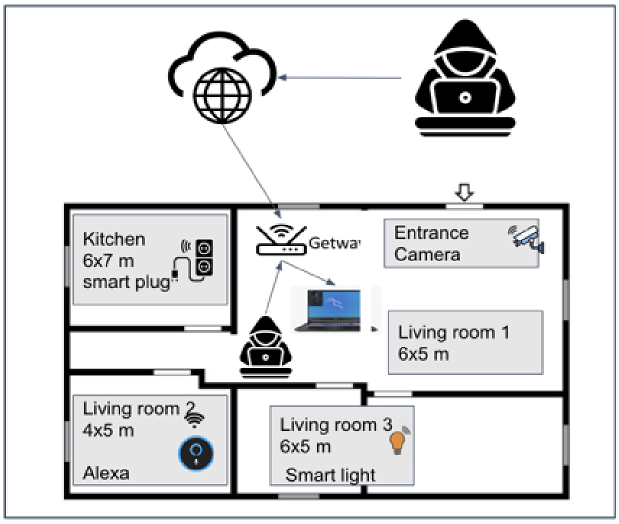


Figure 1. The IoT testbed for collecting attack traffic.

3.2. Attack Scenarios

The attack scenarios implemented in this study were designed to simulate some of the most prevalent cyberattacks targeted at IoT devices. Each attack type was selected based on its relevance and impact on IoT devices in smart home environments, as well as its frequency in real-world attack patterns.

In these scenarios, the attacks are mainly either coming from the inside or outside. In an outside attack simulation, the attacker is not a part of the smart home network, while in an inside attack, the attack can be initiated by compromised or malicious nodes that are part of the network. The IoT device and attacker are connected wirelessly to the access point, providing IPs in the range 192.168.2.2/254.

The traffic generated between the IoT device, attacker, internet, and all network traffic during these attacks was collected using Wireshark installed on Kali Linux as mentioned in the network configuration, then saved in a pcapng format for later analysis.

The selected attack types for our experiment can be defined as follows:

1. Reconnaissance Attack: Reconnaissance is the first step of any cyberattack. By gathering information about available services and open ports, attackers can identify vulnerabilities. In our experiment, we used Nmap to perform scanning and information gathering, simulating how an attacker could probe for weaknesses in IoT devices. To better understand potential reconnaissance attacks on these devices, we recorded Nmap traffic, as seen in Figure 2.
2. Man-in-the-Middle (MITM) Attack: MITM attacks are highly effective in IoT environments where communication between devices and their controlling apps is frequent and often unencrypted. Attackers can intercept or manipulate this communication without detection. This attack was prioritized because many IoT devices lack robust encryption, making them susceptible to this type of intrusion. In our testbed, data were intercepted between IoT devices and the cloud, allowing us to analyze the extent to which sensitive information could be accessed, as shown in Figure 3.
3. Deauthentication Attack: Deauthentication attacks were selected due to their ability to disrupt IoT devices by disconnecting them from their wireless networks. Since most IoT devices rely on Wi-Fi, such an attack causes significant service disruption; see Figure 4. This causes the user to lose access to the internet and all network services until they reconnect. As previously mentioned, the IoT device is wirelessly connected to the access point, providing IPs within the range of 192.168.2.2/254, and the attacker (for this attack) is not connected to the access point. The attacker achieves this by sending a fake request to the access point, causing the user's device to disconnect.
4. UDP flood Attack: This type of denial-of-service attack targets IoT devices by overwhelming them with UDP packets. The attack was simulated on our smart lamp, which communicates via UDP, to test how it handles excessive network traffic. In the following illustration, Figure 2, we can see the attacker sends a large number of User Datagram Protocol (UDP) packets to a specific server or device. It is worth noting that we found the Xiaomi smart lamp to be controlled via UDP. To test this, we sent random UDP messages with a falsified source address that matched the control server.
5. SYN Flood Attack: The SYN Flood attack is another type of denial-of-service attack that disrupts the three-way handshake process in TCP communication. This attack was particularly effective against the web server running on the smart plug, rendering it unresponsive; see Figure 2. During a previous Nmap scan, we found that the smart camera was running API services and the Shelly smart plug had a web server, both of which are vulnerable to a SYN Flood attack.
6. Password Cracking Attack: Password cracking attacks were significant in our experiment because many IoT devices in smart home networks use weak or default passwords. Our smart plug and camera both required password authentication, and we were able to use brute force techniques to successfully gain unauthorized access. Given that many IoT devices have weak password mechanisms, these attacks remain a serious threat. We discovered that our smart plug has a security feature that prompts web users to enter a password once activated. Additionally, the smart camera has an RTSP stream feature that requires a username and password. As a result, we utilized brute force and dictionary attacks to guess the passwords of IoT devices.

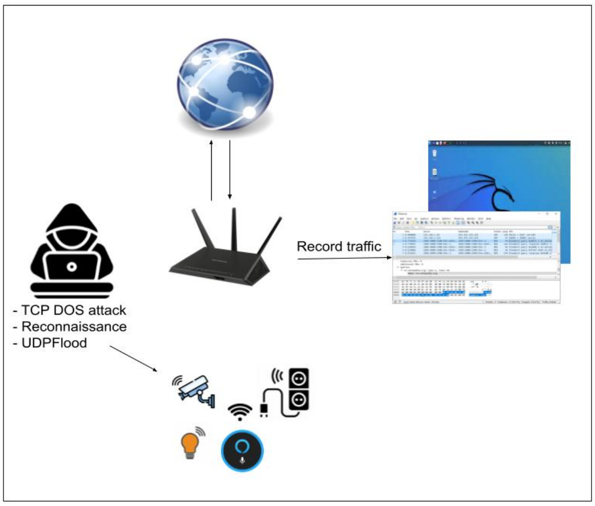


Figure 2. Reconnaissance, TCP-DOS, UDP-DOS attack.

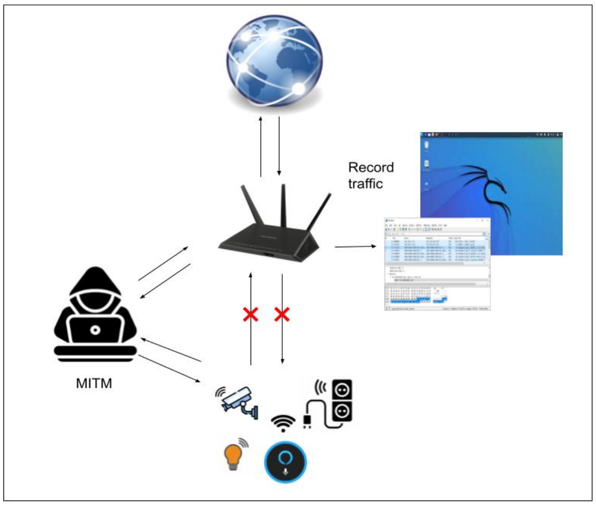


Figure 3. Man-in-the-Middle attack.

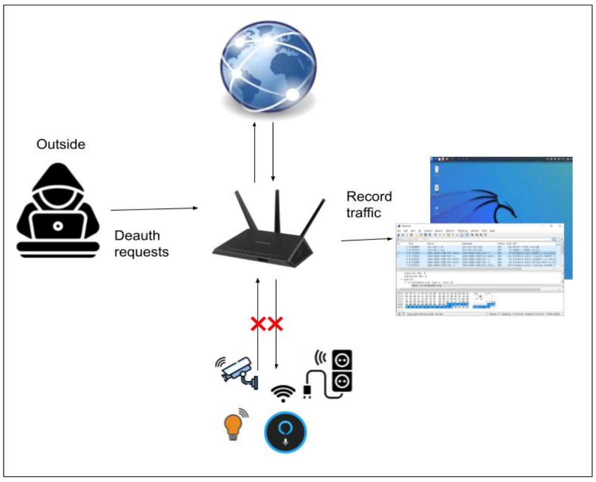


Figure 4. Deauthentication attack.

The selected, previously defined attack types, were executed in the testbed and data were collected in pcapng. Table 2 summarizes the attack results on the smart plug and states whether the attack was successful from outside or inside the network. The rest of the attack results on each IoT device are separately presented in Appendix A.

Table 2. Attacks on smart plug (Shelly) device.

No.	Attack Type	Used Library	Attack Successful	Results	Inside/Outside Attack
1	Nmap	Nmap	yes	80/tcp open HTTP	Inside
2	MITM	Ettercap	yes	Data were intercepted. However, some of the data were readable as some devices' data were encrypted.	Inside
3	Deauth	Aireplay-ng	yes	The device is no longer authenticated to the network	Outside
4	DOS (tcp)	slowhttptest, slowloris, slowite	yes	Device became unresponsive and offline and required reboot.	Inside
5	DOS (UDP Flood)	hping3	no	Device does not use udp or have udp ports open	Inside
6	Password crack	Brute force	yes	The device has a login feature that can be activated in the browser, which prompts the user to enter a username and a password	Inside

3.3. Building the Dataset

As mentioned in the previous section, we collected 24 pcapng files using Wireshark on our Kali Linux, which was set up as a hotspot. We joined these files so the name 'Joint Dataset' is used to emphasize that the dataset combines traffic files from both normal and attack scenarios across multiple IoT devices. Our "Joint Dataset" was then developed by combining selected features from multiple IoT devices.

Compared to existing real-world datasets, such as IoT-23 and Bot-IoT, which provide valuable but generalized network traffic data, the "Joint Dataset" offers several advantages in terms of representativeness. While publicly available datasets typically focus on large-scale, diverse environments, they often lack detailed coverage of certain attack types or device-specific behaviors. The "Joint Dataset" focuses on critical, targeted IoT attacks that are frequently encountered in smart home environments, such as man-in-the-middle attacks, password cracking, and deauthentication attacks, providing more granular insights into IoT vulnerabilities.

To create our "Joint Dataset" IoT dataset, we utilized the IoT TAHFE tool (Traffic Analyzer Tool with Automated and Holistic Feature Extraction Capability) developed by Subahi and Almasre [22]. This tool automatically extracts network features from a pcap file, generating three CSV files: in-depth server/IoT per-packet analysis.csv, general IoT traffic analysis within the network.csv, and full communication pattern mapping.csv. Such a tool helps IoT researchers by speeding up their research efforts through automatically extracting all network features, including network packets and flow, from a pcap file and profiling the normal behavior of an IoT device.

We only used the "in-depth analysis" CSV file, which describes the network traffic between the IoT device and a remote IP, to build our dataset. We combined the selected features from all IoT devices, including benign and attack traffic, resulting in 12,779 samples.

Our dataset consisted of a total of 12,779 instances, with 148 normal traffic examples and five categories of attack: MITM (Man-in-the-Middle), UDP-DOS (User Datagram Protocol Denial of Service), browser password cracking, deauthentication attacks, DOS-SYN (Denial of Service–SYN Flood), and reconnaissance. It is a multi-class dataset that, unlike traditional IoT intrusion detection datasets, does not only consider binary outputs (normal/attack), which often limit the feasibility of using of such datasets in real-world scenarios. Moreover, a multi-class dataset is essential for our research focus, i.e. developing a robust synthetic data generation model that is capable of realistically creating attack type instances that can be used in ML or IDS applications

In the process of creating our dataset, we focused on identifying important features to label the attacks. After analyzing each attack mechanism and network traffic behavior during attacks, we concluded the following:

1. The Protocol feature can define a deauth attack due to its fixed protocol usage.
2. The combination of "Protocol" and "Flow volume" features can identify deauth attacks and MITM attacks, as MITM attacks consist of four ICMP packets.
3. The "IoT_Respond_401" feature identifies false web and RTSP login credentials to identify password-cracking attacks.
4. The "IsServer" feature helps identify a trusted destination.
5. The combination of "Protocol", "No_of_received_packets_per_minutes", and "No_of_sent_packets_per_minutes" helps identify DOS-SYN attacks.

It is important to note that our dataset was imbalanced, with the majority classes being DOS and reconnaissance. This is typical as DOS attacks, for example, are launched from thousands of destinations simultaneously, while other attacks are from a single IP. Additionally, reconnaissance attacks can be launched against thousands of ports simultaneously. We acknowledge that developing ML and DL models to predict attack types would be challenging, especially ones intended to be used in IDS systems. There is a great need for our dataset to be balanced so that it can be used effectively in other predictive systems. Therefore, training a CGAN to accurately generate synthetic data while being conditioned by the types of network attacks, particularly minority ones, such as Man in the Middle (MitM) and denial of service (DoS), is highly in demand in the case of our multi-class Joint Dataset.

3.4. Benchmark Dataset

In ore der to evaluate our "Joint Dataset" created for this research, we select the BoT-IoT sub-dataset [23,24] as a benchmark datset. The BoT-IoT sub-dataset consists of 733,705 records, which were originally extracted from a 7-million-record dataset that was created in the Cyber Range Lab of UNSW Canberra Cyber. This "Joint Dataset" was developed to support research in network security, with a special focus on attacks in IoT environments. Datapoints corresponding to four attack types and a normal state were extracted and compiled into a benchmark dataset for our experiment. Tables 3 and 4 describe the categorical and numerical values of the BoT-IoT sub-dataset. The classes being represented are:

1. DDoS (Distributed Denial of Service): This class simulates attacks where multiple systems overwhelm a target or its surrounding infrastructure with a flood of Internet traffic.
2. DoS (Denial of Service): Similar to DDoS but typically involves a single attacking system, focusing on overwhelming or crashing the target by flooding it with excessive requests.
3. Reconnaissance: These activities are exploratory in nature, aiming to gather information about a network to identify potential vulnerabilities for future attacks.
4. Theft: This class includes scenarios involving unauthorized access and extraction of sensitive data, representing data breaches.
5. Normal: Represents normal network activities to provide a baseline for detecting anomalous behavior and differentiating between benign and malicious traffic.

3.5. Designing the Synthetic Data Generator Tool

The tool is based on a CGAN and is used to create balanced datasets from multi-class imbalanced network attack type datasets. It consists of two main modules: the Exploratory Data Analysis (EDA) module and an IoT Network Traffic Synthetic Data Generator module.

In Figure 5, you can see that the first module allows users to upload their multi-class network attack type data, conduct Exploratory Data Analysis (EDA), and perform initial feature engineering, whereas the second module consists of a CGAN model pipeline that includes steps for extensive pre-processing, feature selection, and generating synthetic data.

Table 3. Statistical summary of the categorical features in Joint Dataset.

Feature	Count	Unique Values	Top Value	Top Value Count
pkSeqID	733,705	733,705	2	1
proto	733,705	5	udp	399,618
saddr	733,705	16	192.168.100.147	189,606
sport	733,705	65,538	0x0303	1794
daddr	733,705	45	192.168.100.3	475,171
dport	733,705	4111	80	714781
seq	733,705	249,513	380	12
N_IN_Conn_P_SrcIP	733,705	100	100	369,260
state_number	733,705	11	4	399,567
N_IN_Conn_P_DstIP	733,705	100	100	573,744
category	733,705	5	DDoS	385,309

Table 4. Statistical summary of the numerical features in Joint Dataset.

Feature	Count	Mean	Std	Min	25%	50%	75%	Max	Number of Samples
stddev	733,705	0.887894	0.804013	0	0.030132	0.795481	1.745595	2.496758	733,705
min	733,705	1.018868	1.484235	0	0	0	2.163444	4.98047	733,705
mean	733,705	2.233429	1.517572	0	0.182193	2.691715	3.566569	4.981785	733,705
drate	733,705	0.506298	74.33018	0	0	0	0	58823.53	733,705
srate	733,705	2.262398	403.4081	0	0.156231	0.283784	0.488849	333333.3	733,705
max	733,705	3.023	1.860725	0	0.281688	4.011386	4.296505	4.999999	733,705

3.5.1. Synthetic Data Generator Tool Algorithm

The tool was developed using Python and utilized libraries such as TensorFlow to construct the CGAN model. Streamlit was also used to create the interactive user interface. The CGAN model's algorithm can be represented as follows:

Input:Dataset D with categorical C and numerical columns N, class column class, latent dimension latent_dim.
Output: Synthetic dataset D

1. Initialization:
    - Construct label encoders for C and a standard scaler for N.
    - Define generator G and discriminator D architectures.
2. Prepossessing:
    - Impute and encode C; impute and scale N using median and mode strategies.
3. Model Construction:
    - Generator G: Combine noise vector z and embedded categorical labels to generate synthetic N.
    - Discriminator D: Classify combined real or synthetic N and embedded labels.
4. Training:
    - Alternate training of D and G using batches of real and generated data.
5. Synthesis:

- Generate and decode synthetic samples for each class, ensuring feature fidelity and balance.

6. Output: Return D; matching the distribution and characteristics of D.

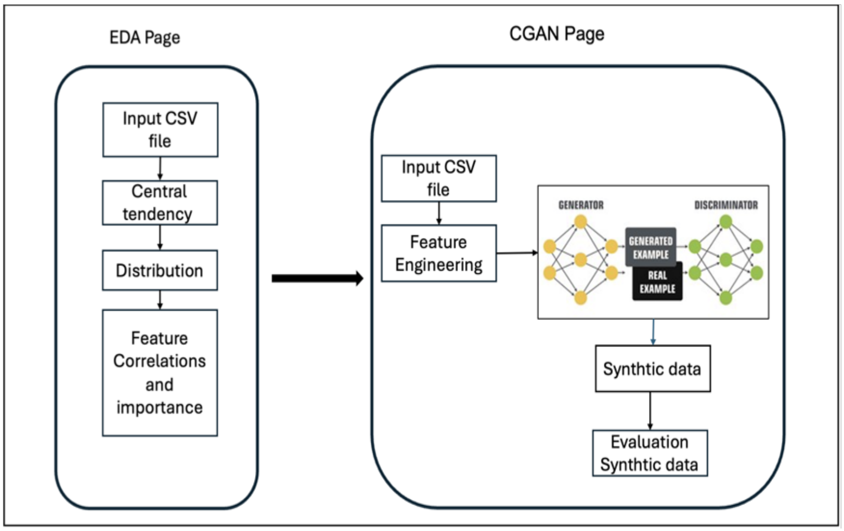
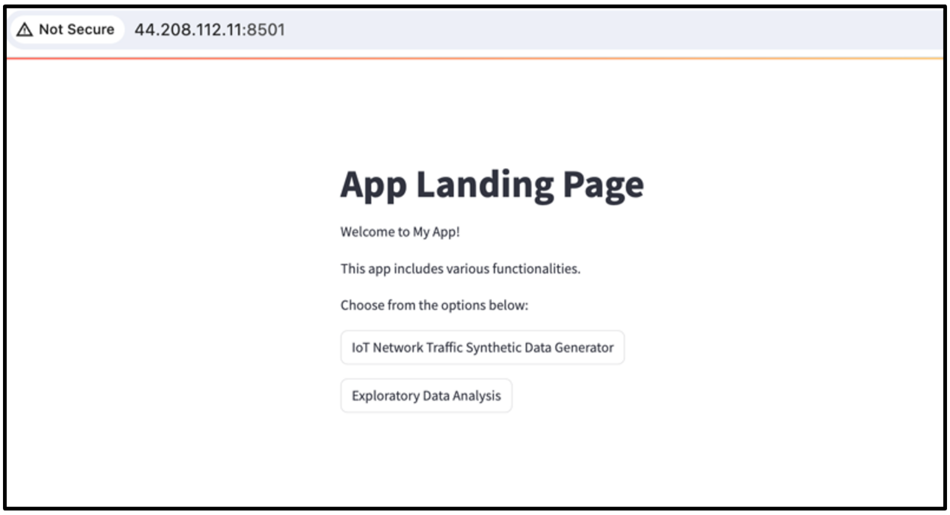


Figure 5. The overview of the tool prototype.

3.5.2. GUI of the Proposed Synthetic Data Generator Tool:

As depicted in Figure 6, the landing page of our system offers two primary modules, allowing the IoT researcher to select either Exploratory Data Analysis (EDA) or IoT Network Traffic Synthetic Data Generator.


⚠ Not Secure 44.208.112.11:8501

App Landing Page

Welcome to My App!

This app includes various functionalities.

Choose from the options below:

IoT Network Traffic Synthetic Data Generator
Exploratory Data Analysis

Figure 6. The main page of the CGAN-based tool.

Users can download and access the tool through this link https://github.com/Alanoud-Subahi/CGAN-based-Tool (accessed on 20 July 2024).

A- Exploratory Data Analysis (EDA) Module: When the IoT researcher selects the second option, the EDA module provides the statistics module, which parses the data, either the original datasets or the generated ones, outputs information with respect to the input, and computes various statistics.

This module has an interactive interface for data visualization and analysis. It allows users to visualize the distribution of numerical features using histograms and assess the balance of categorical features through frequency plots. Basic statistical descriptions of the dataset are also displayed to provide immediate insight into its composition. This module aims to assist researchers in understanding underlying patterns and distributions that may influence the performance of machine learning models (Figure 7).

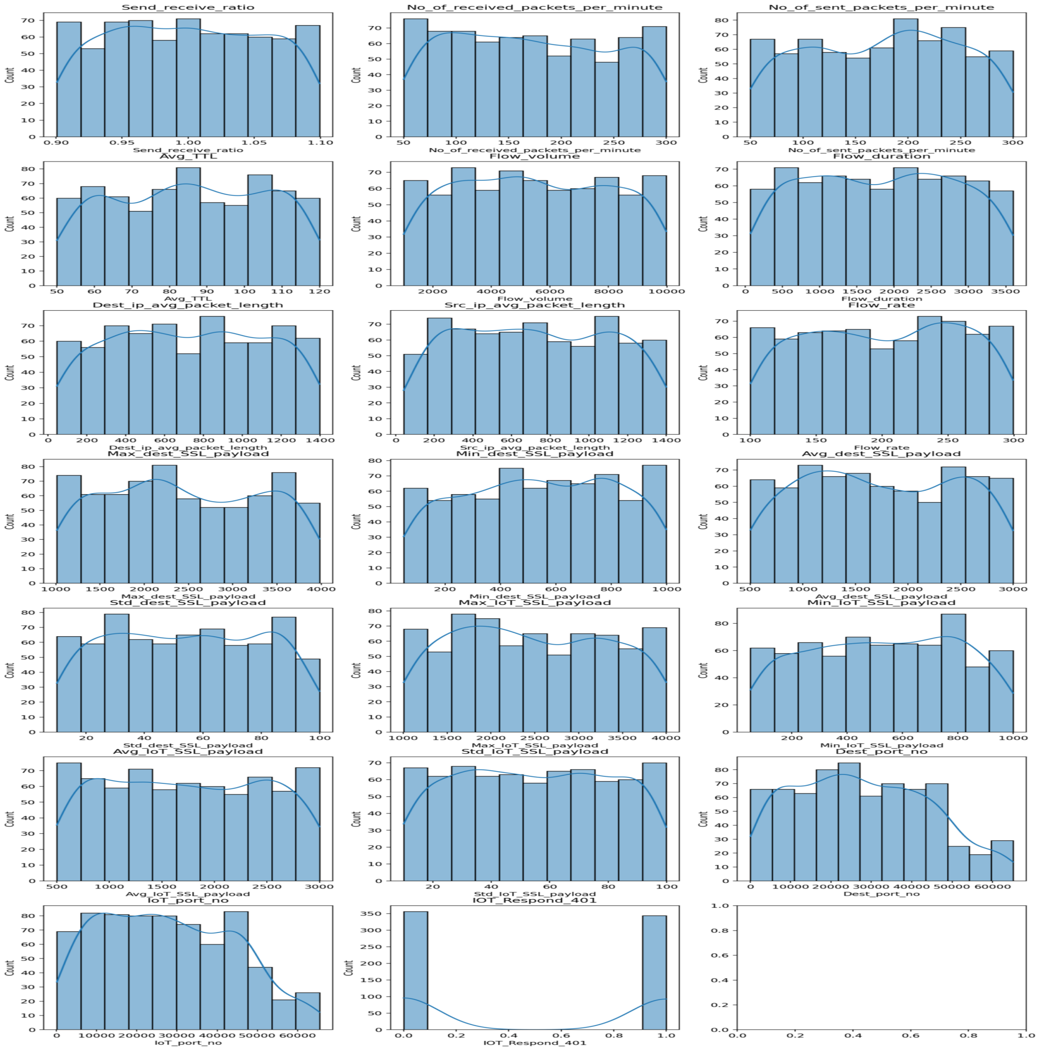


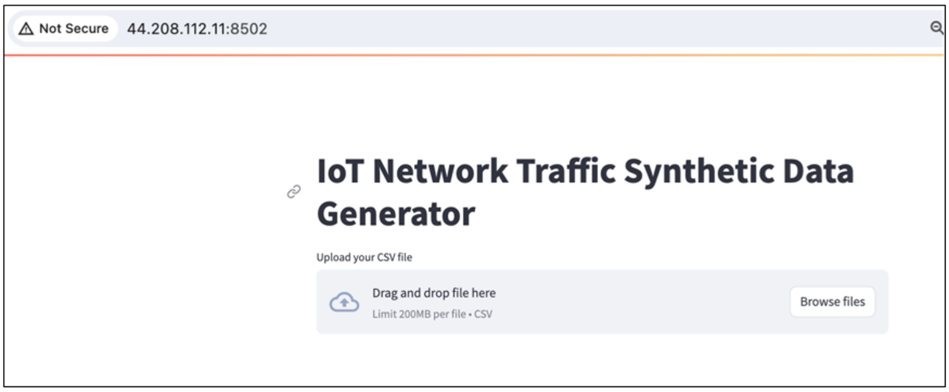
Figure 7. EDA of the dataset analysis.

B- IoT Network Traffic Synthetic Data Generator: The way the CGAN model is designed, in our research, observes some of the most challenging aspects of synthetic data generation when using GANs, as well as when dealing with tabular datasets that include both categorical and numerical features. Research has documented several challenges when using CGANs for generating synthetic tabular data that include both numerical and categorical features, including data pre-processing of mixed features, handling of categorial features, maintaining model stability (mode collapse issues) through robust generators and discriminator structures, conditional generation based on dataset classes, and observing data ranges for numerical data generation [25,26]. To address these issues in the CGAN model proposed for this task, the researchers ensured the following steps are performed:

1. Pre-processing: the user uploads his or her CSV dataset using the GUI (Figure 8). Proper data pre-processing with a special focus on mixed data types is ensured, as it is considered one of the main challenges in using CGANs for tabular data generation. This is important to ensure these diverse data types are represented accurately in the generated data. Our tool, which implements the CGAN model, allows the user to initially select the class column, the categorical and numerical features, as well as the numerical features that should be generated as non-decimal values (Figure 9). These user-selected features are then handled using techniques such as label encoding for

categorical values and scaling for numerical ones to help overcome issues related to non-Gaussian distributions, making it suitable for the CGAN pipeline. Without these pre-processing steps, the model might be unsuccessful in capturing the relationships between the data features, leading to poor-quality synthetic data generation.

This preprocessing pipeline considers the impact of missing values on the quality of generated data, so imputation is used to handle them before feeding the data into a CGAN using simple imputation strategies to ensure the data remain representative and unbiased [27]. At this preprocessing stage as well, the minimum and maximum of each numerical value are stored for further processing. To capture numerical data distribution and reproduce it in the CGAN-generated synthetic data, Kernel Density Estimation (KDE) is implemented to fit each numerical feature.



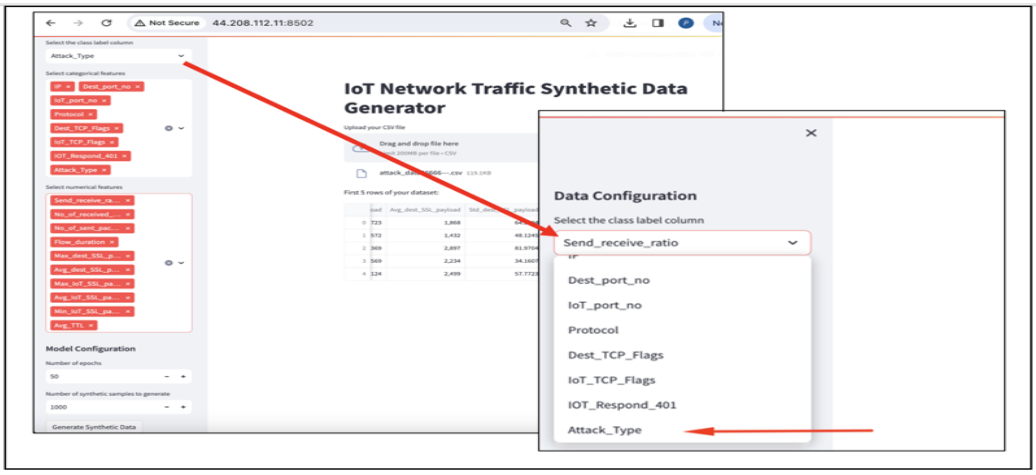
⚠ Not Secure 44.208.112.11:8502

IoT Network Traffic Synthetic Data Generator

Upload your CSV file

Drag and drop file here
Limit 200MB per file • CSV

Figure 8. Select the dataset as CSV file.



⚠ Not Secure 44.208.112.11:8502

Select the class label column: Attack_Type

Select categorical features: IP, Dest_port_no, IoT_port_no, Protocol, Dest_TCP_Flags, IoT_TCP_Flags, IOT_Respond_401, Attack_Type

Select numerical features: Send_receive_ra..., No_of_received_..., No_of_sent_pac..., Flow_duration, Max_dest_SSL_p..., Avg_dest_SSL_p..., Max_IoT_SSL_pa..., Avg_IoT_SSL_pa..., Min_IoT_SSL_pa..., Avg_TTL

Model Configuration

Number of epochs: 50

Number of synthetic samples to generate: 1000

IoT Network Traffic Synthetic Data Generator

Upload your CSV file

Drag and drop file here
Limit 200MB per file • CSV

attack_da...6666---.csv 119.1KB

First 5 rows of your dataset:

	oad	Avg_dest_SSL_payload	Std_dest..._payload
0	723	1,868	64...
1	572	1,432	48.1245
2	369	2,897	81.9704
3	569	2,234	34.1607
4	124	2,499	57.7723

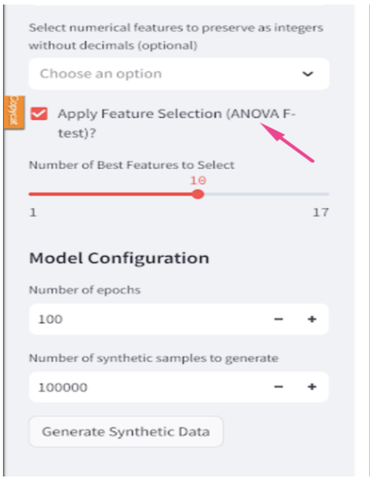
Data Configuration

Select the class label column: Send_receive_ratio

Figure 9. Example of user selection.

2. Feature Selection: To enhance the model's performance, a feature selection step is incorporated to identify the most important numerical features for synthetic data generation. This step is also controlled by the user from the tool's interface. The objective is to reduce the dimensionality of the data and overcome issues related to computational efficiency.

The ANOVA F-test (f_classif) is used for feature selection, where it identifies the most important features that significantly contribute to the variability in the dataset in relation to the target variable Figure 10.



Select numerical features to preserve as integers without decimals (optional)

Choose an option

☑ Apply Feature Selection (ANOVA F-test)?

Number of Best Features to Select

10

1 17

Model Configuration

Number of epochs

100 − +

Number of synthetic samples to generate

100000 − +

Figure 10. ANOVA F-test with the threshold to select the best number of features.

Once the data are pre-processed and feature selection is enabled, the "SelectKBest" function from "scikit-learn" is used in conjunction with the (f_classif) score function to perform the ANOVA F-test. As a first step, the F-test calculates the F-score for all features to quantify the linear dependency connecting every feature to the target variable. Then, each feature is ranked based on the computed F-scores. Features with higher scores possess stronger relationships with the target variable. When selecting this test to be implemented, the user can set the "Number of Best Features to Select" which is the top 'k'. Features with the highest F-scores are then included as input to the model.

For this feature selection, user-controlled step, the researchers tested mutual information as well because of its reported ability to capture non-linear relationships between dataset features and a target variable, ultimately offering a more detailed understanding of feature importance. Nonetheless, after testing it, the researchers opted out of implementing it as it demands careful tuning of binning strategies, which led to difficulties in implementation. This is especially important as the tool is intended to offer users a consistent and interpretable solution that will not be majorly affected by varying feature distributions in the dataset. In addition, mutual information's estimates can be unreliable specifically in small datasets, as it does not offer a easy to understand ranking of features, thus complicating the selection process. These implementation challenges led the researchers to use the ANOVA F-test due to its simplicity, computational efficiency, and clear feature ranking process.

3. Model Architecture: Primarily, the model consists of a generator and discriminator networks, which are trained alternately to ensure that both models work competitively, mitigating issues related to mode collapse. The generator has three dense layers with 128 and 256 neurons, implementing ReLU activations. This structure provides sufficient capacity for the generator to capture the complexity and high-dimensionality of relationships between noise, categorical, and numerical features. Layers for batch normalization and dropout layers are added as well to ensure training stability and prevent overfitting. The generator works by combining two input layers: one for random noise and the other for categorical labels. The categorical labels are embedded into a dense vector using an embedding layer. This layer maps each category to a high-dimensional space. Both the noise vector and the previously created embeddings are then concatenated to create a combined input, which is fed to several dense layers with ReLU activation. As seen in Figure 11, the architecture facilitates the generation of realistic and balanced synthetic data that imitate the statistical distribution of the original multi-class dataset.

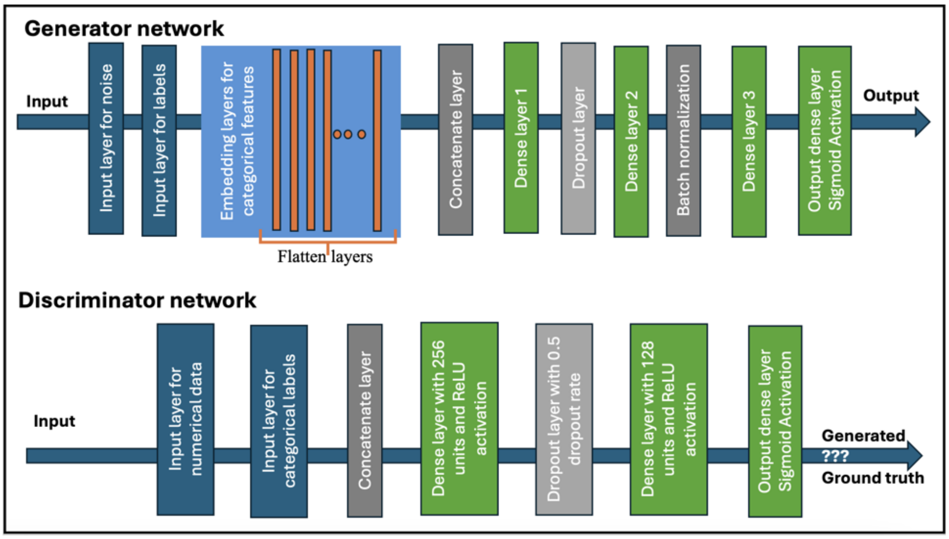

Generator network
Input
Input layer for noise
Input layer for labels
Embedding layers for categorical features
Flatten layers
Concatenate layer
Dense layer 1
Dropout layer
Dense layer 2
Batch normalization
Dense layer 3
Output dense layer Sigmoid Activation
Output
Discriminator network
Input
Input layer for numerical data
Input layer for categorical labels
Concatenate layer
Dense layer with 256 units and ReLU activation
Dropout layer with 0.5 dropout rate
Dense layer with 128 units and ReLU activation
Output dense layer Sigmoid Activation
Generated
???
Ground truth

Figure 11. The generator and discriminator networks.

The actual structure of the CGAN model also considers a number of recurrent challenges in handling tabular data with mixed types, like in the case of the datasets reflecting IoT network attack types. The generator is structured in such a way as to handle mixed data types by incorporating embedding layers that represent encoded categorical variables. The objective is to transform high-cardinality categorical features into dense vectors to reduce dimensionality and capture the relationships between them, enhancing the generator's ability to produce realistic synthetic datasets. These embeddings are also combined with normalized noise vectors, ensuring the accurate representation of numerical data. Ultimately, this method enables the generator to learn the relations between the noise and the labels, facilitating the generation of data specific to the given class labels. Moreover, the generator contains dense layers with dropout and batch normalization to prevent overfitting and contribute to the stabilization of training, thus attempting to overcome the challenge of non-Gaussian and multimodal distributions in numerical data, especially since the users of our tool might use datasets that inherently have these issues. An inverse transform step at the end ensures that the numerical values are within their original data scale.

The discriminator network architecture consists of dense layers with 256 and 128 neurons, which slightly imitates that of the generator in terms of complexity but with attention to reducing it to avoid overfitting. The objective is to maintain a balance in which the discriminator's complexity helps it distinguish real from synthetic data without overpowering the generator. To ensure that the discriminator does not become too confident in the classification of synthetic data as fake, dropout is used. In this way, the discriminator ensures that it evaluates the real and synthetic data and differentiates them. It consists of dense and dropout layers for regularization. The dense layers with ReLU activation learn the distribution of both data types, numerical and categorial. The inclusion of a dropout layer with a 0.5 dropout rate helps prevent overfitting, a common issue in GANs, by randomly omitting some units during training, which forces the network to generalize better. The final dropout layer with a sigmoid activation function probabilistically discriminates real from synthetic input, allowing for better convergence during training. Basically, the discriminator is conditioned on the categorical labels so that the network can capture the nuances of different classes. This helps overcome the mode collapse issue in which the generator might generate a limited varieties of outputs.

Through using the CGAN model, the researchers were basically utilizing its conditional structure to address the complex nature of data generation from a multi-class attack-type dataset. The class labels are, for example, embedded into the generator and discriminator networks so that the model output values are not only realistic but also representative of specifically labeled attack types. The generator takes these class-conditioned values to learn the distinctive statistical features of each class, while the discriminator assesses whether the generated values coincide with the actual distribution of these classes.

4. Training: The training process targets the enhancement of the generator and discriminator models' performance through adopting an adversarial training framework.

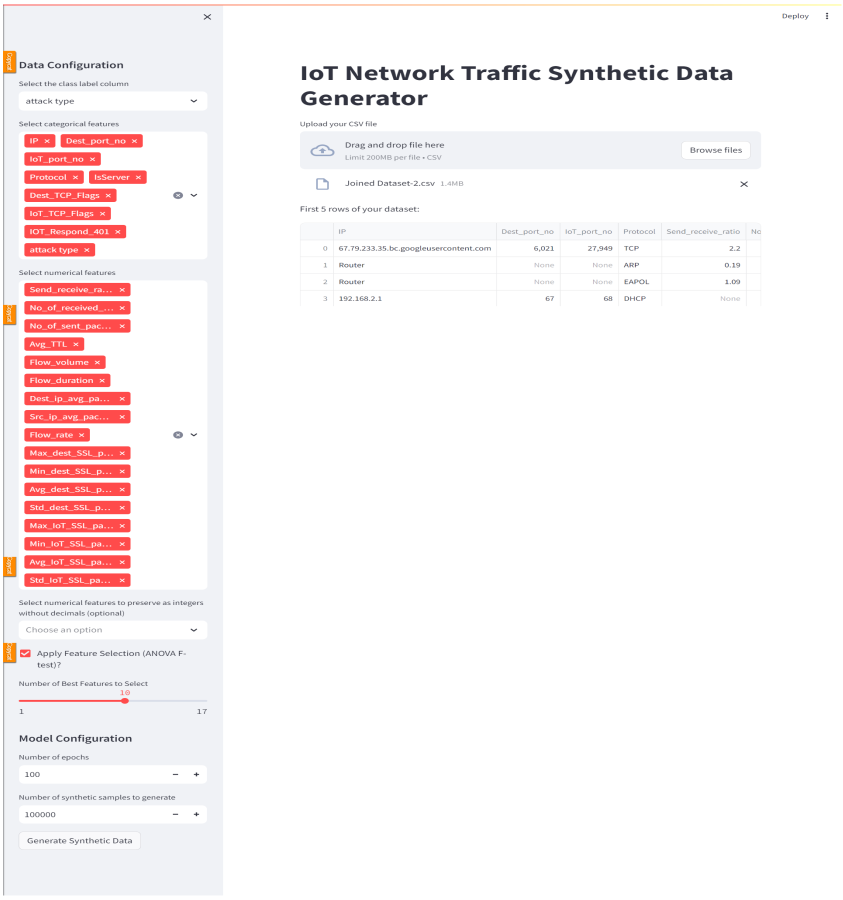
The learning rate, set to 0.0002 for the Adam optimizer, is applied to both the generator and discriminator with the purpose of achieving stable convergence. The assumption, as research demonstrates, is that a lower learning rate assists in the gradual weight updates, hence overcoming issues related to overshooting minima during optimization. Similarly, this learning rate helps in reducing the expected mode collapse, instances where the generator outputs a limited range of values. For the combined model, the learning rate is set to half of the one used for the discriminator to ensure that the generator learns at a slower pace, gradually fooling the discriminator.

The learning rate of the discriminator is set slightly higher than that of the generator. This facilitates the discriminator providing useful gradients that can be used in the generator's training. This ensures the training balance so that neither network surpasses the other. A dual-phase training is performed in mini-batches of the real dataset to optimize computational efficiency with the objective of classifying these samples as real (with outputs close to 1). Then, it should classify the generated synthetic data samples as fake (with outputs close to 0).

The training process is designed to consider solutions when the research reported the challenges faced when implementing GANs. Generally, the instability of the training process, which ultimately results in non-convergence, is one issue. Utilizing the Adam optimizer with meticulously selected learning rates for both the generator and discriminator helps maintain stable updates and does not allow for oscillations [28,29].

5. Data Generation: The actual process of data generation entails ensuring that the generated data adhere to the distributions and characteristics of the original multi-class dataset, reproducing values for both numerical and categorical features (Figure 12). This function is performed post-training, utilizing the generator's learned parameters to generate new data points. The user of the tool selects the number of samples to be generated, then downloads the output as CSV files for both datasets (with feature selection and without) (Figure 13).

The function is designed to ensure that the selected number of samples are evenly distributed across the distinct categories present in the class column (in this case, the attack type). The adequate representation of each class is thus guaranteed in the synthetic dataset. This is intended to address the imbalance issues which might exist in the original data. We designed the model in such a way that for each class, random noise vectors are to be sampled from a standard normal distribution of the dataset, ensuring the realization of the stochastic input required to generate diverse data points.



Data Configuration

Select the class label column

attack type

Select categorical features

IP ×, Dest_port_no ×, IoT_port_no ×, Protocol ×, IsServer ×, Dest_TCP_Flags ×, IoT_TCP_Flags ×, IOT_Respond_401 ×, attack type ×

Select numerical features

Send_receive_ra… ×, No_of_received_… ×, No_of_sent_pac… ×, Avg_TTL ×, Flow_volume ×, Flow_duration ×, Dest_ip_avg_pa… ×, Src_ip_avg_pac… ×, Flow_rate ×, Max_dest_SSL_p… ×, Min_dest_SSL_p… ×, Avg_dest_SSL_p… ×, Std_dest_SSL_p… ×, Max_IoT_SSL_pa… ×, Min_IoT_SSL_pa… ×, Avg_IoT_SSL_pa… ×, Std_IoT_SSL_pa… ×

Select numerical features to preserve as integers without decimals (optional)

Choose an option

☑ Apply Feature Selection (ANOVA F-test)?

Number of Best Features to Select: 10 (1–17)

Model Configuration

Number of epochs: 100

Number of synthetic samples to generate: 100000

Generate Synthetic Data

IoT Network Traffic Synthetic Data Generator

Upload your CSV file

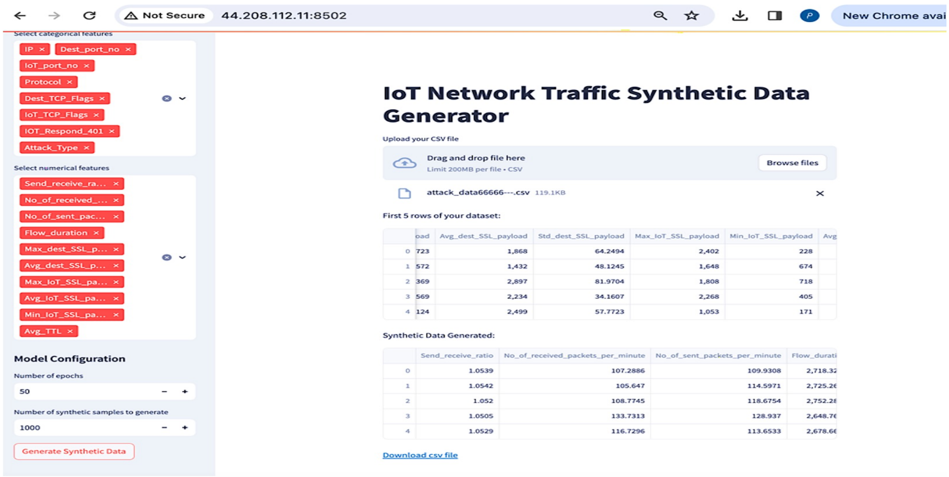
Drag and drop file here
Limit 200MB per file • CSV
Browse files

Joined Dataset-2.csv 1.4MB

First 5 rows of your dataset:

	IP	Dest_port_no	IoT_port_no	Protocol	Send_receive_ratio	Nc
0	67.79.233.35.bc.googleusercontent.com	6,021	27,949	TCP	2.2	
1	Router	None	None	ARP	0.19	
2	Router	None	None	EAPOL	1.09	
3	192.168.2.1	67	68	DHCP	None	

Figure 12. Full page view of CGAN processing the user dataset.



Not Secure 44.208.112.11:8502

Select categorical features

IP ×, Dest_port_no ×, IoT_port_no ×, Protocol ×, Dest_TCP_Flags ×, IoT_TCP_Flags ×, IOT_Respond_401 ×, Attack_Type ×

Select numerical features

Send_receive_ra… ×, No_of_received_… ×, No_of_sent_pac… ×, Flow_duration ×, Max_dest_SSL_p… ×, Avg_dest_SSL_p… ×, Max_IoT_SSL_pa… ×, Avg_IoT_SSL_pa… ×, Min_IoT_SSL_pa… ×, Avg_TTL ×

Model Configuration

Number of epochs: 50

Number of synthetic samples to generate: 1000

Generate Synthetic Data

IoT Network Traffic Synthetic Data Generator

Upload your CSV file

Drag and drop file here
Limit 200MB per file • CSV
Browse files

attack_data66666---.csv 119.1KB

First 5 rows of your dataset:

	oad	Avg_dest_SSL_payload	Std_dest_SSL_payload	Max_IoT_SSL_payload	Min_IoT_SSL_payload	Avg
0	723	1,868	64.2494	2,402	228	
1	572	1,432	48.1245	1,648	674	
2	369	2,897	81.9704	1,808	718	
3	569	2,234	34.1607	2,268	405	
4	124	2,499	57.7723	1,053	171	

Synthetic Data Generated:

	Send_receive_ratio	No_of_received_packets_per_minute	No_of_sent_packets_per_minute	Flow_durati
0	1.0539	107.2886	109.9308	2,718.32
1	1.0542	105.647	114.5971	2,725.26
2	1.052	108.7745	118.6754	2,752.28
3	1.0505	133.7313	128.937	2,648.76
4	1.0529	116.7296	113.6533	2,678.66

Download csv file

Figure 13. Synthetic and balanced IoT dataset ready to downloaded on the user device using CGAN.

4. Results and Discussion

In this section, we present the results of applying the Synthetic Data Generator Tool to our "Joint Dataset" to create a balanced dataset. Furthermore, we evaluate how effectively the synthetic dataset mitigates class imbalance.

4.1. Implementation

In this experiment, we used the Synthetic Data Generator Tool to generate a multi-class attack type balanced dataset from both our previously created dataset "Joint Dataset" and the BOT-IoT Dataset, with and without feature selection. We set the model configuration to 100 epochs of training, and requested the generation of 100,000 samples. We investigated the computation usage and cost for the two scenarios of feature selection and without feature selection. When feature selection was implemented, the total training time for our Joint Dataset was lower at 180.04 s in comparison to 195.17 s without feature selection, which indicated that feature selection resulted in reduced training time due to faster convergence. The peak memory usage continued to be similar in both scenarios, with 32.25 MB for the feature selection scenario and 32.51 MB without it. This demonstrates that while feature selection can assist in reducing the training time by simplifying the input space, it rarely impacts memory usage because the model architecture and batch sizes are the same.

Due to the slight differences in computational cost, we focused on the generated datasets without feature selection to test the generated data adherence to the balance criteria we set, considering the presence of all features being previously collected or benchmarked.

4.2. Class Imbalance Evaluation

To assess the effectiveness of the synthetic dataset in addressing the class imbalance present in the "Joint Dataset" dataset, we employed a method that involved calculating the deviation of each class's proportion from an ideal, perfectly balanced distribution. This method allowed us to quantitatively measure how far each class in both the original and synthetic datasets deviated from what would be expected if the classes were equally represented. By comparing these deviations, we could then determine the extent to which the synthetic dataset corrected the imbalances found in the "Joint Dataset". We benchmarked the results of synthetic data generation for both the Joint and BOT-IoT datasets.

Joint Dataset: In our "Joint Dataset", a number of classes (attack types) demonstrated significantly indicative deviations from the ideal value, representing a very clear imbalance. For instance, the 'reconnaissance' class with the highest deviation of 0.3047 and a proportion value of 30.47% demonstrates an imbalanced dataset. In a similar manner, the 'SYN Flood' class showed a deviation of 0.2719, and the 'MITM' and 'deauth' classes both had deviations above 0.1422, which indicates that these classes are a minority in the dataset. Table 5 compares the "Joint Dataset" to its synthetic version

Table 5. Comparison of original and synthetic datasets across various attack types.

Joint Dataset	**Attack Type**						
	Normal	**MITM**	**Deauth**	**Password-Cracking**	**Reconn-Aissance**	**SYN Flood**	**UDP-Flood**
Original Number of Samples	148	4	8	8	5718	5299	1590
Original Deviation	0.131272	0.142544	0.142231	0.142231	0.304736	0.271937	0.018395
Synthetic Number of Samples	1428	1428	1428	1428	1428	1428	1428
Synthetic Deviation	0	0	0	0	0	0	0
Imbalance Reduction Proportion	0.131272	0.142544	0.142231	0.142231	0.304736	0.271937	0.018395

In contrast, the synthetic dataset achieved a perfect balance across all classes, as indicated by the fact that the Synthetic Deviation for every class was reduced to 0. This outcome shows that the synthetic data effectively adjusted the proportions of each class to match the ideal distribution, correcting any imbalance (Figure 14). Consequently, the Imbalance Reduction was equal to the Original Deviation for each class. For example, the 'reconnaissance' class, which initially had a 30.47% deviation, was perfectly balanced in the synthetic dataset, resulting in a full reduction of the imbalance. Similarly, the classes 'SYN Flood', 'MITM', 'deauth', and 'password-cracking' all saw their deviations entirely corrected, reflecting the synthetic dataset's success in achieving a uniformly distributed representation.

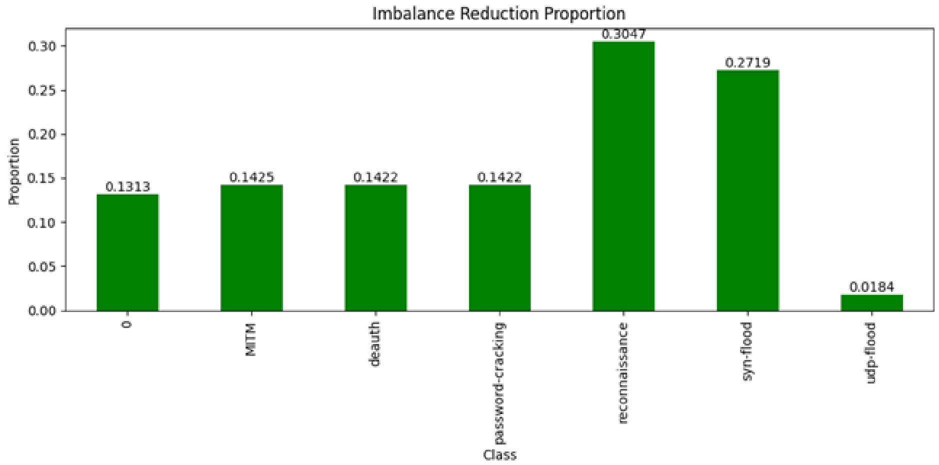


Figure 14. The imbalance reduction proportion of Joint Dataset.

BOT-IoT Dataset: In the BOT-IoT Dataset, class imbalances were also significant, with the 'DDoS' attack type showing the highest deviation at 0.3252, suggesting a considerable overrepresentation compared to an ideal balanced distribution. Other classes like 'DoS' and 'Theft' also showed substantial deviations, with the 'Theft' class being the most underrepresented, having a deviation close to 0.2000, see Table 6.

Table 6. Comparison of BOT-IOT original and synthetic datasets across various attack types.

Bot Dataset	Attack Type				
	DDoS	**DoS**	**Normal**	**Reconnaissance**	**Theft**
Number of Samples	385309	330112	18163	107	14
Original Deviation	0.325155	0.249925	0.199854	0.175245	0.199980
Synthetic Number of Samples	20,000	20,000	20,000	20,000	20,000
Synthetic Deviation	0	0	0	0	0
Imbalance Reduction Proportion	0.325155	0.249925	0.199854	0.175245	0.199981

The synthetic version of the BOT-IoT dataset effectively neutralized these imbalances, achieving a synthetic deviation of 0 for all classes, reflecting an ideal balanced state. This correction allowed for an exact match to the ideal distribution, with the imbalance reduction proportion mirroring the original deviation for each class (Figure 15). The 'DDoS' class, for example, saw a complete elimination of its initial deviation, indicating that the synthetic dataset provides a well-balanced platform for model training and further analysis.

Overall, the results demonstrate that the synthetic datasets not only mitigated the class imbalances but completely resolved them, leading to datasets where all classes are equally represented (Figures 16 and 17). This balanced representation is crucial for downstream tasks, such as machine learning, where class imbalance can lead to biased models and inaccurate predictions. By ensuring that each class is equally represented, the synthetic datasets enhance the overall quality and reliability of the data, making them superior alternatives to the original, imbalanced datasets.

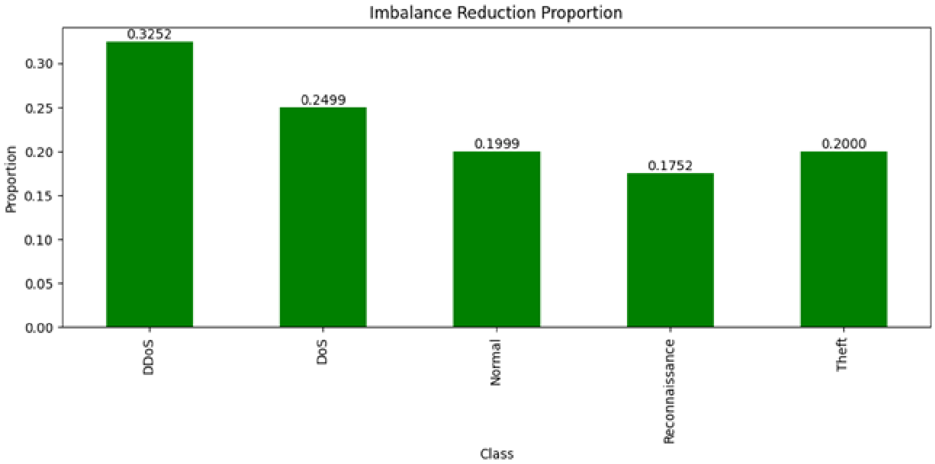


Figure 15. The imbalance reduction proportion of BoT-IoT.

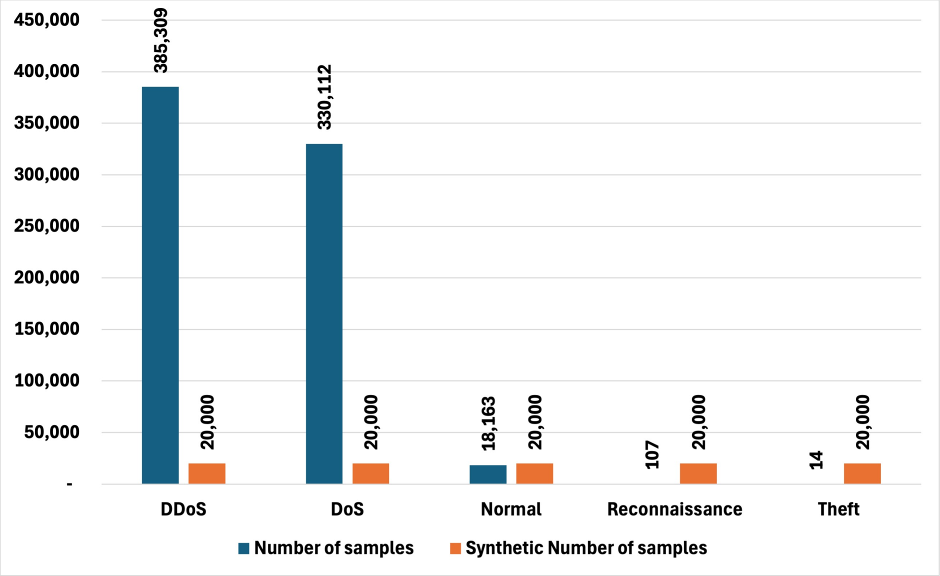


Figure 16. The comparison of synthetic and original in BOT dataset.

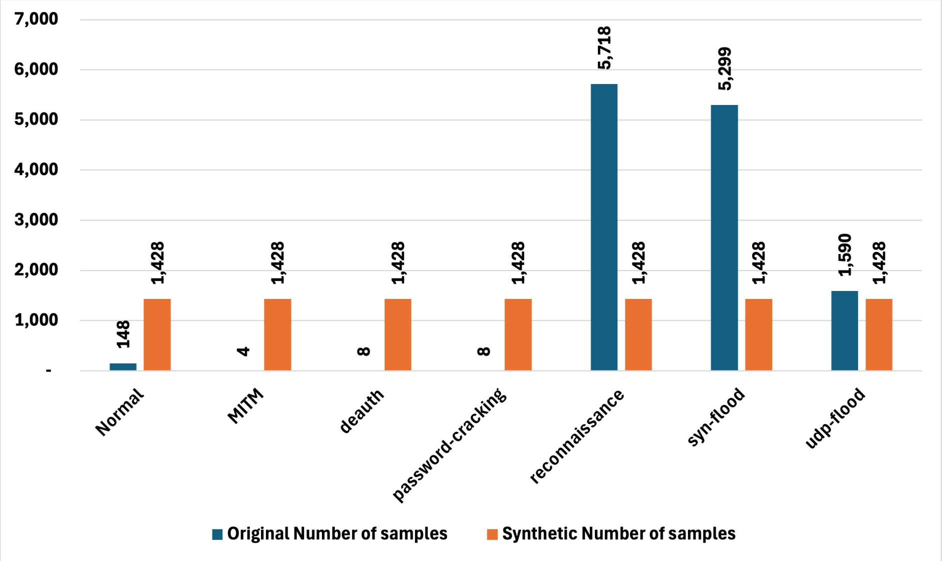


Figure 17. The comparison of synthetic and original in Joint Dataset.

4.3. Evaluation of the Ranges of Numerical Features in Both Datasets

One of the main aspects of the synthetic dataset that the researchers wanted to investigate qualitatively is its features' replication of the ranges of the real dataset, especially when considering the numerical values. Therefore, to evaluate that the generated numerical values are within the ranges of the original data, we compared the minimum-maximum ranges of each feature. Table 7 presents the comparison:

Table 7. Summary statistics for selected features.

Feature	Count	Mean	Std	Min	25%	50%	75%	Max	Number of Samples
Dest_port_no	9996	41,659.04	12,909.2	53	36,065.5	43,553	51,254	60,988	9996
IoT_port_no	9996	10,112.15	15,022.46	1	1152	4001	9101	65389	9996
Dest_TCP_Flags	9996	1161.495	2844.339	0	2	11	18	8180	9996
IoT_TCP_Flags	9996	22.59104	32.59825	0	2	12	18	100	9996
IOT_Respond_401	9996	0.493597	0.499984	0	0	0	1	1	9996

A straightforward comparison of the real and synthetic datasets value ranges for the numerical features reveals that almost all features, except for "drate" largely replicate the ranges. However, it is demonstrated as well that while the synthetic dataset closely reproduces the variability observed in the original data, there are slight differences at the lower end of the ranges, which we believe is normal and acceptable within the context of synthetic data generation as such data initiates the often-observable incongruities in real datasets.

4.4. Evaluation of Categorical Features

To evaluate the representation of categorical features between the original Joint Dataset and synthetic one, we employed two primary methods: descriptive statistics and cumulative difference analysis. The descriptive statistics provided a detailed summary of each categorical feature's distribution in both datasets, including measures such as mean, standard deviation, and range. This allowed us to assess how closely the synthetic data replicated the original dataset's distribution for each feature. Additionally, we conducted a cumulative difference analysis to quantify the overall deviation between the two datasets. By calculating the absolute differences in category proportions and summing these across all categories within each feature, we derived a cumulative difference value. This value serves as a metric to evaluate how well the synthetic data align with the original, particularly in terms of capturing the variability and distribution of categorical features.

The evaluation of Joint Dataset categorical features using these methods revealed varying degrees of alignment between the original and synthetic datasets, reflecting both the effectiveness and limitations of the synthetic generation process. The original dataset, derived from a real-world scenario, was characterized by missing values and class imbalance, which likely influenced the representation of categorical features. These imperfections in the original data posed a significant challenge for the synthetic generation process (Table Summary Statistics of categorical features). The cumulative difference was as well computed (Figure 18).

Features like 'Dest_port_no', 'IsServer', and 'IOT_Respond_401' exhibited lower cumulative differences, indicating that the synthetic dataset closely mirrors the original distributions of these categories. For example, the Dest_port_no feature showed a mean of 41,659.04 in the synthetic data compared to 42,422.96 in the original dataset, with a consistent range across both datasets. The relatively low cumulative difference for these features suggests that the synthetic data effectively captured the original distributions, even in the presence of missing values and class imbalance.

In contrast, features such as ‘IP’, ‘Dest_TCP_Flags’, and ’Protocol’ demonstrated higher cumulative differences, signaling significant deviations between the original and synthetic datasets. For instance, the ‘Dest_TCP_Flags’ feature in the synthetic dataset had a mean of 1161.50 and a standard deviation of 2844.34, compared to the original dataset’s mean of 3.49 and standard deviation of 109.75. Similarly, the ‘IP’ feature showed the highest cumulative difference, indicating that the synthetic dataset struggled to replicate the original data’s distribution accurately. These differences suggest that the synthetic dataset may have introduced more variability, possibly as a corrective measure for the underrepresentation and imbalance present in the original data, particularly in complex features or those heavily affected by missing values.

Overall, while the synthetic dataset successfully addresses many of the original dataset’s deficiencies, such as class imbalance and missing data, it also introduces new variations that were not present in the original data. This analysis underscores the complexity of generating synthetic data that not only replicates but also potentially improves upon the original data’s representation, especially when the original data are plagued by real-world imperfections. The results highlight both the strengths and limitations of the synthetic data, demonstrating its effectiveness in certain areas while also revealing where further refinement may be necessary to achieve a more accurate and nuanced representation.

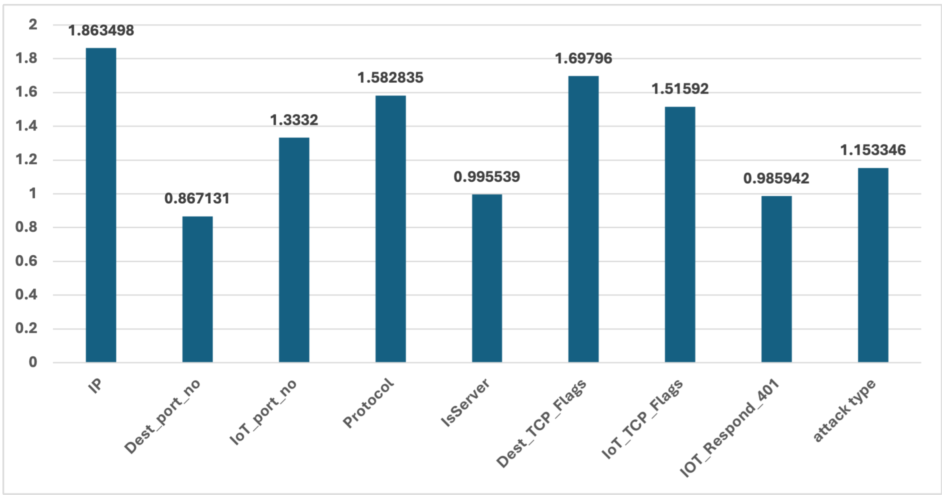


Figure 18. The Joint Dataset categorical features cumulative difference.

4.5. Evaluation of Numerical Features

To assess the alignment between the Joint Dataset and synthetic dataset in terms of the numerical features’ representation, we applied the cumulative difference measure, as well as the mean absolute error (MAE), root mean squared error (RMSE), and the correlation coefficient. as for the cumulative difference, it is expected to reflect the absolute difference in mean values when comparing the Joint and synthetic datasets per feature. The MAE and RMSE measures are used to compute the average and squared deviations, respectively, allowing us access to insights about how individual data values differ between the datasets. in addition, the correlation coefficient is used to assess the strength and direction of the linear relationship between the Joint and synthetic datasets, with values close to 1 or -1, indicating strong positive or negative correlations, respectively; see Table 8.

Some synthetic data features showed a relatively good replication of the Joint Dataset, as indicated by lower cumulative differences and acceptable correlation coefficients. For instance, the ’Send_receive_ratio’ feature has a cumulative difference of 3.08, scoring an MAE of 3.14 and RMSE of 3.18. In spite of the fact that the correlation coefficient value is low (0.01), the small difference in mean values suggests that the synthetic data capture the overall scale of this feature reasonably well, even if the linear relationship is weak. In a similar way, the feature ’Flow_duration’ has a low cumulative difference of 2.58, and its MAE and RMSE values are also low (2.59 and 2.60, respectively). Nonetheless, this feature has a negative correlation (−0.031), indicating that while the synthetic dataset captures the mean value closely, the direction of the relationship between the Joint and synthetic datasets may differ.

Features like 'Avg_dest_SSL_payload' and 'Min_IoT_SSL_payload' also exhibit moderate cumulative differences (234.16 and 122.74, respectively), with scored MAE and RMSE values that are relatively low. These metrics suggest that the synthetic data capture the general distribution of these features reasonably well, though the correlation coefficients remain low or slightly negative, which demonstrates that the linear relationship is not strongly preserved.

Table 8. Evaluation of numerical features.

Feature	Cumulative Difference	MAE	RMSE	Correlation
Send_receive_ratio	3.082799	3.137454	3.183326	0.010839
No_of_received_packets_per_minutes	672.1486	672.1486	674.19	0.016176
No_of_sent_packets_per_minutes	1809.712	1810.018	1815.296	−0.00774
Avg_TTL	48.10356	48.38865	49.17743	0.00193
Flow_volume	879907.2	879982.6	882675.5	0.013071
Flow_duration	2.580025	2.589503	2.603747	−0.03076
Dest_ip_avg_packet_length	214.148	214.301	216.9572	−0.00157
Src_ip_avg_packet_length	716.8348	716.9825	718.8302	0.000316
Flow_rate	1746863	1746863	1751104	−0.01433
Max_dest_SSL_payload	728.1321	728.3052	733.4881	0.012872
Min_dest_SSL_payload	155.2861	155.441	156.3907	−0.00866
Avg_dest_SSL_payload	234.1595	234.2863	236.5934	−0.00195
Std_dest_SSL_payload	129.1881	131.8779	133.9625	−0.00594
Max_IoT_SSL_payload	746.4051	747.3188	749.6432	−0.00565
Min_IoT_SSL_payload	122.7403	122.7745	123.4037	−0.00796
Avg_IoT_SSL_payload	660.2485	660.3813	662.3838	0.008442
Std_IoT_SSL_payload	313.2357	313.48	314.5114	−0.00133

However, there are features which exhibited discrepancies between the Joint and synthetic datasets, as indicated by higher cumulative differences, MAE, RMSE, and low or negative correlation coefficients. Features, including 'No_of_received_packets_per_minutes', 'Flow_rate', Max_dest_SSL_payload', 'Std_dest_SSL_payload', and 'Max_IoT_SSL_payload' generally do not align with the distribution of the Joint Dataset as observed by the moderate to high cumulative differences, high MAE and RMSE values, and low correlation coefficient.

5. Conclusions

This research presented in this paper introduces a novel Synthetic Data Generator Tool designed to generate realistic and balanced synthetic datasets for IDSs in IoT environments. The primary contribution of the tool lies in its ability to address the persistent challenge of data imbalance. By leveraging the unique capabilities of CGANs, the tool generates high-quality synthetic data that closely mirror real-world IoT network traffic, thereby enhancing the training of IDS models. Through comprehensive data preprocessing, feature selection, and model training, this tool ensures the generation of high-quality synthetic data that closely mimic the characteristics of real-world IoT network traffic.

The tool's feature engineering capabilities and statistical dashboard provide valuable insights for researchers, aiding in analyzing and interpreting network traffic data. The successful implementation and validation of the tool mark a significant advancement in the field of IoT security, offering a powerful resource for researchers and practitioners. The evaluation results demonstrate the tool's efficacy in creating balanced datasets that can be used to enhance the performance of ML, DL, and IDS models' performance.

In practical terms, the tool is particularly valuable for industry professionals, especially in sectors like smart homes, where IoT integration is prevalent. Its advanced ability to detect anomalies and adapt to different usage patterns ensures robust security against potential threats, such as unauthorized access or abnormal device behavior. The tool's compatibility with common IDS platforms like Snort, Suricata, or Zeek allows for easy integration without major infrastructure changes. This makes it a key asset for professionals seeking intelligent and secure solutions in the growing IoT and smart home industry.

Overall, this study contributes significantly to research done in IoT environments, by providing access to a tool that can generate realistic datasets replicating network attacks thus addressing key challenges in dataset generation.

6. Limitations and Future Work

Future work will expand the tool's capabilities to support a wider range of IoT protocols and environments, ensuring its applicability across diverse network scenarios.

Future work will expand the tool's capabilities to support a wider range of IoT protocols and environments, ensuring its applicability across diverse network scenarios. Specifically, we plan to enhance the tool to simulate a broader array of cyberattacks, which are becoming increasingly relevant as IoT ecosystems grow more complex.

In this study, we focused on selected network attack types that almost always feature as minority classes in IoT datasets, including reconnaissance, deauthentication, and MITM attacks. While these attacks are representative of common threats in IoT environments, we acknowledge the importance of expanding our research to encompass a broader array of cyberattacks, such as ransomware, insider threats, and advanced persistent threats (APTs). These attacks are emerging as significant threats in IoT contexts, particularly in smart home environments and industrial IoT deployments. We plan to expand the testbed by incorporating specific malware simulations and internal compromise scenarios that reflect these evolving threats, improving the tool's utility in real-world cybersecurity applications, including IDS training.

In addition to network-based attacks, future work will extend the tool to simulate web-based attacks, which are increasingly prevalent for IoT devices that integrate web interfaces or host web servers. Attacks such as SQL Injection (SQLi), Cross-Site Scripting (XSS), and Cross-Site Request Forgery (CSRF) represent significant vulnerabilities, particularly in smart devices that interact with external web applications.

To achieve this, we will design a dedicated testbed for simulating web-based attacks. This testbed will include IoT devices with integrated web servers or web-based control panels, which will be subjected to various attack scenarios. For example, SQL injection attacks will be simulated by targeting vulnerable input fields within the web interfaces of the IoT devices, while XSS and CSRF attacks will be launched to manipulate user sessions and hijack control over the devices.

Furthermore, we will expand the dataset by generating web-based traffic that accurately reflects real-world IoT web communication patterns. The tool will be adapted to handle these new attack types by extending the feature set to include web-related attributes such as HTTP requests, URL parameters, and session tokens. This will allow the tool to generate realistic synthetic datasets that can train machine learning models to detect web-based attacks in IoT environments.

Author Contributions: Conceptualization, A.S.; methodology, A.S.; software, A.S. and M.A.; validation, A.S. and M.A.; formal analysis, A.S. and M.A.; investigation, A.S. and M.A.; resources, A.S. and M.A.; data curation, A.S.; writing—original draft preparation, A.S.; writing—review and editing, A.S. and M.A.; visualization, A.S. and M.A.; project administration, A.S.; funding acquisition, A.S. and M.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research work was funded by the Institutional Fund Project under grant no. (IFPIP: 1478-865-1443). The authors gratefully acknowledge the technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia, Jeddah.

Data Availability Statement: Not applicable

Acknowledgments: Both authors are working at and sponsored by King Abdulaziz University in Saudi Arabia. This research work was funded by the Institutional Fund Project under grant no. (IFPIP: 1478-865-1443). The authors gratefully acknowledge the technical and financial support provided by the Ministry of Education and King Abdulaziz University, DSR, Jeddah, Saudi Arabia. https://dsr.kau.edu.sa/Default-305-ar (accessed 6 August 2024).

Conflicts of Interest: The authors declare no conflicts of interest.

Appendix A. Results of Applying the Attacks on the IoT Devices

Table A1. Attacks on smart lamp (Xaiomi) device.

No.	Attack Type	Used Library	Attack Successful	Results	Inside/Outside Attack
1	Nmap	Nmap	yes	No open ports found	Inside
2	MITM	Ettercap	yes	Data were intercepted. However, some of the data were readable as some devices' data were encrypted.	Inside
3	Deauth	Aireplay-ng	yes	The device is no longer authenticated to the network.	Outside
4	DOS (tcp)	slowhttptest, slowloris, slowite	no	Device does not have TCP servers running.	Inside
5	DOS (UDP Flood)	hping3	yes	Device was halted sometimes and its response was delayed. The device changed status randomly, occasionally when the UDP had a meaning by coincidence.	Inside
6	Password crack	Brute force	no	The device has no login/password prompts on any services.	Inside

Table A2. Attacks on smart camera (tapo tc70) device.

No.	Attack Type	Used Library	Attack Successful	Results	Inside/Outside Attack
1	Nmap	Nmap	yes	443/tcp open https —554/tcp open rtsp —2020/tcp open xinupageserver —8800/tcp open sunwebadmin —20,002/tcp open commtact-http	Inside
2	MITM	Ettercap	yes	Data were intercepted. However, some of the data were readable as some devices' data were encrypted.	Inside
3	Deauth	Aireplay-ng	yes	The device is no longer authenticated to the network	Outside
4	DOS (tcp)	slowhttptest, slowloris, slowite	yes	Service on 443 went down. However, the device did not go offline, nor was functionality affected	Inside

Table A2. *Cont.*

No.	Attack Type	Used Library	Attack Successful	Results	Inside/Outside Attack
5	DOS (UDP Flood)	hping3	no	Device does not use UDP or have UDP ports open	Inside
6	Password crack	Brut force	yes	The device has a user-password prompt on rtsp streaming which yields 401 when wrong credentials are provided.	Inside

Table A3. Attacks on smart camera (EZVIS) device.

No.	Attack Type	Used Library	Attack Successful	Results	Inside/Outside Attack
1	Nmap	Nmap	yes	−80/tcp open https −554/tcp open rtsp	Inside
2	MITM	Ettercap	yes	Data were intercepted. However, some of the data were readable as some devices' data were encrypted.	Inside
3	Deauth	Aireplay-ng	yes	The device is no longer authenticated to the network	Outside
4	DOS (tcp)	slowhttptest, slowloris, slowite	yes	Service on 443 went down. However, the device did not go offline, nor was functionality affected	Inside
5	DOS (UDP Flood)	hping3	no	Device does not use UDP or have UDP ports open.	Inside
6	Password crack	Brut force	yes	The device has a user-password prompt on rtsp streaming, which yields 401 when wrong credentials are provided.	Inside

References

1. Kumar, V.; Sinha, D. Synthetic attack data generation model applying generative adversarial network for intrusion detection. *Comput. Secur.* **2023**, *125*, 103054. [CrossRef]
2. Jeong, J.; Lim, J.Y.; Son, Y. A data type inference method based on long short-term memory by improved feature for weakness analysis in binary code. *Future Gener. Comput. Syst.* **2019**, *100*, 1044–1052. [CrossRef]
3. Alabdulwahab, S.; Kim, Y.T.; Seo, A.; Son, Y. Generating Synthetic Dataset for ML-Based IDS Using CTGAN and Feature Selection to Protect Smart IoT Environments. *Appl. Sci.* **2023**, *13*, 10951. [CrossRef]
4. Alsaedi, A.; Moustafa, N.; Tari, Z.; Mahmood, A.; Anwar, A. TON_IoT telemetry dataset: A new generation dataset of IoT and IIoT for data-driven intrusion detection systems. *IEEE Access* **2020**, *8*, 165130–165150. [CrossRef]
5. Samarakoon, S.; Siriwardhana, Y.; Porambage, P.; Liyanage, M.; Chang, S.Y.; Kim, J.; Kim, J.; Ylianttila, M. 5g-nidd: A comprehensive network intrusion detection dataset generated over 5g wireless network. *arXiv* **2022**, arXiv:2212.01298.
6. Liu, X.; Li, T.; Zhang, R.; Wu, D.; Liu, Y.; Yang, Z. A GAN and feature selection-based oversampling technique for intrusion detection. *Secur. Commun. Netw.* **2021**, *2021*, 9947059. [CrossRef]
7. Riera, T.S.; Higuera, J.R.B.; Higuera, J.B.; Herraiz, J.J.M.; Montalvo, J.A.S. A new multi-label dataset for Web attacks CAPEC classification using machine learning techniques. *Comput. Secur.* **2022**, *120*, 102788. [CrossRef]
8. Parmisano, A.; Garcia, S.; Erquiaga, M.J. *A Labeled Dataset with Malicious and Benign Iot Network Traffic*; Stratosphere Laboratory: Praha, Czech Republic, 2020.
9. Hindy, H.; Bayne, E.; Bures, M.; Atkinson, R.; Tachtatzis, C.; Bellekens, X. Machine learning based IoT intrusion detection system: An MQTT case study (MQTT-IoT-IDS2020 dataset). In *Proceedings of the International Networking Conference;* Springer: Cham, Switzerland, 2020; pp. 73–84.
10. Koroniotis, N.; Moustafa, N.; Sitnikova, E.; Turnbull, B. Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset. *Future Gener. Comput. Syst.* **2019**, *100*, 779–796. [CrossRef]
11. Hamza, A.; Gharakheili, H.H.; Benson, T.A.; Sivaraman, V. Detecting volumetric attacks on lot devices via sdn-based monitoring of mud activity. In Proceedings of the 2019 ACM Symposium on SDN Research, San Jose, CA, USA, 3–4 April 2019; pp. 36–48.
12. Sivanathan, A.; Gharakheili, H.H.; Loi, F.; Radford, A.; Wijenayake, C.; Vishwanath, A.; Sivaraman, V. Classifying IoT devices in smart environments using network traffic characteristics. *IEEE Trans. Mob. Comput.* **2018**, *18*, 1745–1759. [CrossRef]

13. Sharafaldin, I.; Lashkari, A.H.; Ghorbani, A.A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp* **2018**, *1*, 108–116.
14. Meidan, Y.; Bohadana, M.; Mathov, Y.; Mirsky, Y.; Shabtai, A.; Breitenbacher, D.; Elovici, Y. N-baiot—Network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Comput.* **2018**, *17*, 12–22. [CrossRef]
15. Sivanathan, A.; Sherratt, D.; Gharakheili, H.H.; Radford, A.; Wijenayake, C.; Vishwanath, A.; Sivaraman, V. Characterizing and classifying IoT traffic in smart cities and campuses. In Proceedings of the 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), Atlanta, GA, USA, 1–4 May 2017; pp. 559–564.
16. Sureda Riera, T.; Bermejo Higuera, J.R.; Bermejo Higuera, J.; Sicilia Montalvo, J.A.; Martínez Herráiz, J.J. SR-BH 2020 Multi-Label Dataset 2022. Available online: https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/OGOIXX (accessed on 15 September 2024).
17. Mashrur Arifin, M.; Shoaib Ahmed, M.; Ghosh, T.K.; Zhuang, J.; Yeh, J.h. A Survey on the Application of Generative Adversarial Networks in Cybersecurity: Prospective, Direction and Open Research Scopes. *arXiv* **2024**, arXiv:2407.08839.
18. Ranka, P.; Shah, A.; Vora, N.; Kulkarni, A.; Patil, N. Computer Vision-Based Cybersecurity Threat Detection System with GAN-Enhanced Data Augmentation. In *International Conference on Soft Computing and Its Engineering Applications*; Springer: Cham, Switzerland, 2023; pp. 54–67.
19. Strickland, C.; Zakar, M.; Saha, C.; Soltani Nejad, S.; Tasnim, N.; Lizotte, D.J.; Haque, A. Drl-gan: A hybrid approach for binary and multiclass network intrusion detection. *Sensors* **2024**, *24*, 2746. [CrossRef] [PubMed]
20. Dina, A.S.; Siddique, A.; Manivannan, D. Effect of balancing data using synthetic data on the performance of machine learning classifiers for intrusion detection in computer networks. *IEEE Access* **2022**, *10*, 96731–96747. [CrossRef]
21. Vasilomanolakis, E.; Cordero, C.G.; Milanov, N.; Mühlhäuser, M. Towards the creation of synthetic, yet realistic, intrusion detection datasets. In Proceedings of the NOMS 2016—2016 IEEE/IFIP Network Operations and Management Symposium, Istanbul, Turkey, 25–29 April 2016; pp. 1209–1214.
22. Subahi, A.; Almasre, M. IoT Traffic Analyzer Tool with Automated and Holistic Feature Extraction Capability. *Sensors* **2023**, *23*, 5011. [CrossRef] [PubMed]
23. Ashraf, J.; Keshk, M.; Moustafa, N.; Abdel-Basset, M.; Khurshid, H.; Bakhshi, A.D.; Mostafa, R.R. IoTBoT-IDS: A novel statistical learning-enabled botnet detection framework for protecting networks of smart cities. *Sustain. Cities Soc.* **2021**, *72*, 103041. [CrossRef]
24. UNSW, S. The Bot-IoT Dataset. 2021. Available online: https://research.unsw.edu.au/projects/bot-iot-dataset (accessed on 27 August 2024).
25. Figueira, A.; Vaz, B. Survey on synthetic data generation, evaluation methods and GANs. *Mathematics* **2022**, *10*, 2733. [CrossRef]
26. Saxena, D.; Cao, J. Generative adversarial networks (GANs) challenges, solutions, and future directions. *ACM Comput. Surv. (CSUR)* **2021**, *54*, 1–42. [CrossRef]
27. Couplet, E.; Lee, J.A.; Verleysen, M. Tabular Data Synthesis Using Generative Adversarial Networks: An Application to Table Augmentation. Master's Thesis, UCLouvain, Ottignies-Louvain-la-Neuve, Belgium, 2021.
28. Nayak, A.A.; Venugopala, P.; Ashwini, B. A Systematic Review on Generative Adversarial Network (GAN): Challenges and Future Directions. *Arch. Comput. Methods Eng.* **2024**, 1–34. [CrossRef]
29. Ahmad, Z.; Chen, M.; Bao, S. Understanding GANs: Fundamentals, variants, training challenges, applications, and open problems. *Multimed. Tools Appl.* **2024**, 1–77. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.