GAN-AIIPot: GAN-based Cyber Deception for Probing Attacks on IoT Devices

Volviane Saphir Mfogo, Alain Zemkoho, Laurent Njilla, Marcellin Nkenlifack, Charles Kamhoua

***Abstract*—The Internet of Things (IoT) is an emerging technology that has transformed the global network by interconnecting internet-enabled devices, people, intelligent things, and valuable data, leading to significant advancements in various domains. As IoT devices become more interwoven into our daily lives, the security of these devices is a huge concern. Many IoT devices are connected to the internet, making them open to security threats. Researchers have been exploring new methods for detecting and mitigating cyberattacks on IoT devices. One promising approach is the use of Generative Adversarial Networks (GANs) for cyber deception. Cyber deception is a cybersecurity technique used to mislead attackers and hackers from their intended targets. GANs have shown promise in the field of cybersecurity for creating realistic synthetic data to test the security of systems. In the case of probing attacks on IoT devices, GAN-based cyber deception can be used to create fake devices/information that can mimic real IoT devices and deceive attackers into thinking that they have successfully compromised a target. This paper proposes a novel GAN-based cyber deception technique called GAN-AIIPot, which is designed for probe attacks on IoT devices. GAN-AIIPot is an extended version of AIIPot that adds a GAN model on top of the Bidirectional Encoder Representations from the Transformers (BERT) model used in AIIPot. We evaluate our approach using a publicly available IoT dataset and show that GAN-AIIPot captures more sophisticated attacks and improves session length with attackers, showing the effectiveness of the deception technique compared to the existing honeypots. We believe that such a solution can enhance the security of IoT devices and protect them from malicious actors.**

***Index Terms*—Generative Adversarial Networks (GAN), Internet of Things (IoT) Devices, Honeypot, Machine Learning, Probe Attacks.**

I. Introduction

The Internet of Things (IoT) attracts a lot of investment in various domains and applications like healthcare, military, transportation, public services, and electronics [1]. Concerns have been expressed about the security of network-connected devices as IoT has grown in popularity. Unfortunately, IoT devices can have security weaknesses just like other traditional computers. Such weaknesses include insecure open ports and the usage of passwords that are either weak or hard-coded. Additionally, there are instances where passwords are easily predictable, publicly accessible, or unmodifiable. Attackers can exploit open ports and weak passwords to gain unauthorised access to devices, while software vulnerabilities, particularly those found in the device's operating system, enable malicious activities to be carried out. IoT vulnerabilities, on the other hand, are typically too reliant on the type of device, firmware version, or even the vendor. As a result, after scanning the network, attackers notice network vulnerabilities and tend to execute many inquiries on the remote target device to gain additional information about the particular device before launching the attack code (exploit code). This process is known as the pre-check attack step or probe attack. Probing is a type of attack employed by attackers to assess the vulnerabilities of a network. Attackers conduct various scans on a system to identify potential weaknesses. Quick scan, instant scan, and mscan are examples of probing techniques. The life cycle of an IoT attack is depicted in Figure 1 by the three boxes. Cyber deception is a technique that can help improve the security posture of IoT devices. Adversarial learning can be used to enhance the effectiveness of cyber deception by creating sophisticated decoys or honeypots that appear to be legitimate IoT devices but are actually designed to lure attackers away from the real devices. A honeypot is a deception strategy used by security practitioners to uncover vulnerabilities. Generally, a honeypot simulates real-world contact and encourages unauthorised users (attackers) to launch attacks. Generative Adversarial Networks (GANs) are a type of adversarial learning approach used in machine learning (ML). GANs involve the simultaneous training of a generator and discriminator, enabling the generation of realistic synthetic data that can effectively mimic real data. In the context of IoT security, GANs can generate realistic synthetic data that mimics real IoT device responses, making it difficult for attackers to distinguish between real and fake devices. This dynamic adaptability is a key motivation for using GANs in our proposed GAN-AIIPot system.

In the context of IoT, where billions of devices are connected and communicating with each other over the internet, the need for robust security measures is essential. IoT vulnerabilities depend on device type; a honeypot with a low amount of engagement will fail during the probing step and fail to capture the real attack code. There are numerous commercial honeypot products on the market, as well as many honeypot projects on GitHub. We discovered that some IoT honeypots are based on a machine-learning model. One example is AIIPot [3] for Adaptive Intelligent-Interaction honeyPot

Volviane Saphir Mfogo and Marcellin Nkenlifack are with the Department of Mathematics & Computer Science, University of Dschang, BP 96 Dschang, CAMEROON (e-mail: smfogo, marcellin.nkenlifack@aimsammi.org, gmail.com).

Alain Zemkoho is with the School of Mathematical Sciences, University of Southampton, SO17 1BJ Southampton, UK (e-mail: a.b.zemkoho@soton.ac.uk).

Laurent Njilla is with the Information Assurance Branch, Air Force Research Laboratory, Rome, NY, USA (e-mail: laurent.njilla@us.af.mil).

Charles Kamhoua is with the Network Security Branch, DEVCOM Army Research Laboratory, Adelphi, MD, USA (e-mail: charles.a.kamhoua.civ@army.mil).

for IoT devices. AIIPot is a honeypot that is based on the Bidirectional Encoder Representations from the Transformers (BERT) model to learn and automatically reply to attackers. In this context, the BERT model is trained on a real IoT dataset where existing responses are those collected from real IoT devices. This is a critical problem because an attacker who interacts with this system can learn from those responses to make more sophisticated attacks later. So AIIPot [3] can contribute to making the IoT system more vulnerable.

Our goal in this paper is to create a honeypot that can engage with attackers during the checking phase in order to successfully analyse attacks against IoT devices. This is done to gain access to the attacker's exploit code. This honeypot must be able to provide appropriate responses to the attacker's requests. However, those responses should not help the attacker to learn about IoT devices since the attacker may be using a machine learning model to perform the attack. Also, this honeypot must be able to pass the attacker's pre-check stage in order to capture the attack code. Because of the vast number, diversity, and heterogeneity of IoT devices, manually creating low and high-interaction honeypots is not feasible [4]. Some device tests, on the other hand, are straightforward, such as confirming that the response from the target device is correct. As a result, a honeypot that responds correctly to the received request may avoid these tests. Others may be more complicated and necessitate more steps before the exploit code is executed.

In this paper, we propose a novel GAN-based cyber deception technique called *GAN-AIIPot*. GAN-AIIPot is an extended version of the AIIpot honeypot, which is designed for IoT devices. GAN-AIIPot uses a transformer-based architecture model (BERT) and a generator from GAN to construct a response that is able to fool an attacker who used a machine learning model to probe an IoT network. GAN-AIIPot keeps the adaptability property as new requests that arrive at the system are broadcast to the IoT device network, which gives a highly expected response to the attacker. All the new requests are evaluated and broadcast to all the IoT devices of the network in order to gather more accurate responses and collect datasets. With these methods, our honeypots return expected responses to the attacker even if the request is not represented on the initial database, therefore increasing the likelihood that the attacker will launch the attack on the honeypot, thinking that it is a real IoT device. This allows us to extend the session length and the time of deception and captures the exploit code. GAN-AIIPot is a deception technique that has several advantages over traditional honeypots, which are typically used to detect and lure attackers. Traditional honeypots are often static and easy to detect, and attackers may be wary of them. GAN-AIIPot, on the other hand, can create dynamic honeypots and is difficult for attackers to distinguish from real devices. Additionally, GAN-AIIPot can be more cost-effective than deploying physical honeypots, as it can be implemented entirely in software.

The main contributions of this paper are summarised as follows:

1) A honeypot based on a combination of BERT and GAN models is proposed to capture 0-day vulnerabilities on IoT during a one-shot and multiple-step probe process.
2) We successfully used GAN to create a response that significantly increased the deception time.
3) We proposed a dynamic and cost-effective honeypot for IoT devices.

Although GAN-AIIPot extends AIIPot [3], its contributions are fundamentally distinct and go beyond incremental improvements. Specifically: (i) GAN-AIIPot introduces adversarial training to model attacker–defender interactions, enabling resilience against ML-driven attackers; (ii) A novel fusion strategy integrates BERT embeddings with GAN outputs, accelerating convergence and improving response realism; (iii) Unlike AIIPot, which risks attackers learning from authentic responses, GAN-AIIPot generates dynamic, synthetic, and diverse responses that mitigate information leakage; (iv) The system explicitly addresses adversarial robustness, an aspect absent in AIIPot and existing ML-based IoT honeypots.

The rest of this paper is organised as follows: Section II shows the background and related works. Section III describes the system model and problem formulation. Section IV elaborates on the main design of our approach. The experimental result is provided in Section V. Finally, we conclude in Section VII.

II. Background and Related Work

A. Probe attack on IoT

Attackers dedicate a significant portion of their time to probing networks, as this reconnaissance phase is crucial for identifying vulnerabilities and potential entry points for exploitation [2]. Within the context of probing attacks, attackers conduct network scans on devices to identify potential weaknesses in their network topology or port configurations. These vulnerabilities can be later exploited to gain unauthorised access to sensitive information. Probing attacks encompass various types, such as IPsweep, Nmap, and Portsweep. It is important to note that the vulnerabilities of IoT devices can be influenced by factors such as the device type, firmware version, and even the vendor, thereby emphasising the significance of these variables in IoT security. An attacker generally follows the steps shown in Figure 1 while launching an attack on an IoT device.

During the initial pre-attack phase known as reconnaissance, the attacker systematically engages in activities to discover, gather, identify, and document information pertaining to the targeted device. The objective is to acquire as much knowledge as possible about the victim. This initial step primarily involves passive information gathering, wherein the attacker undertakes actions such as collecting information, delineating the network scope, identifying active devices, detecting open ports and access points, performing OS fingerprinting, service fingerprinting, and mapping the network. On IoT devices, some attackers need only a one-shot step probe while in others, attackers need a multiple-step probe before launching the attack. So there are two types of probes that attackers can use to gather information on IoT devices: one-shot probes and multiple-step probes [5]. In the one-shot probe situation, only one interaction (request/response) is enough for the attacker

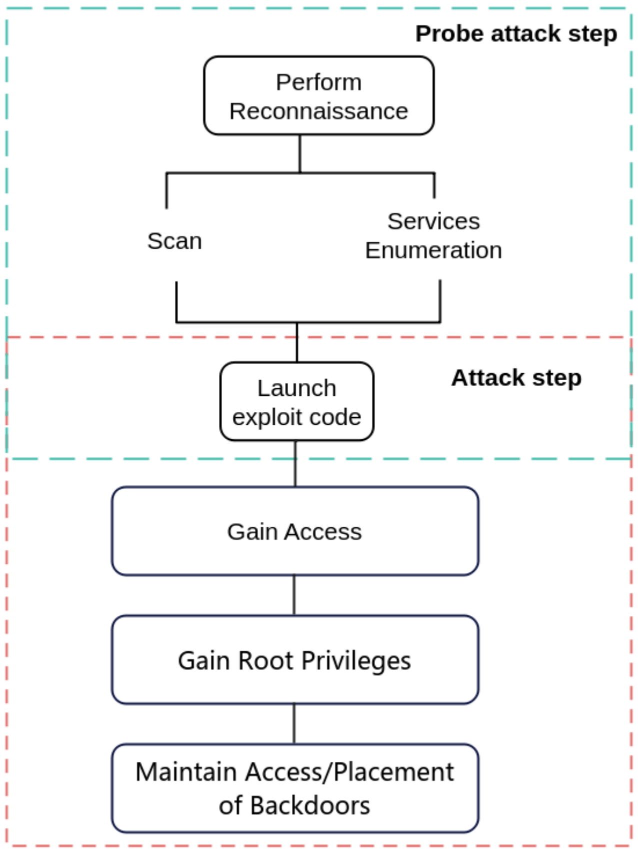


Fig. 1: Steps in performing an attack on IoT device.

to have sufficient information to launch the attack (exploit code). For instance, some device tests are straightforward, such as confirming that the target device's response is correct. In a multiple-step probe, the attacker needs to interact with the target device for a while. In this case, the information provided by the target device should be consistent, otherwise, the attacker will not launch the attack. Existing network reconnaissance software tools, such as Nmap, are focused on devices with IP addresses. However, several popular IoT protocols, such as BLE, Zigbee, Z-Wave, and LoRa, do not support IP addressing. IoT-Scan [6] is a universal IoT network reconnaissance tool.

B. Cyber deception for IoT devices

Deception has been used for defensive purposes since the dawn of civilisation and has been implemented for both offensive and defensive purposes [7]. Deception has also been used in computer networking, most notably in the form of honeypots, honeynets, and the use of canary files. As IoT deployments accelerate, the replication of decoys for security purposes is a crucial basis for defending the IoT ecosystem and the users linked with accompanying services. By incorporating cyber deception techniques into IoT device security strategies, it is possible to create a more robust defence against cyber attacks. However, cyber deception should be used as part of a comprehensive security strategy that includes other measures such as encryption, secure communication protocols, and regular software updates [8].

C. Generative Adversarial Networks (GANs)

Generative Adversarial Networks (GANs) are a class of ML models that consist of two neural networks: a generator and a discriminator. The generator creates synthetic data, while the discriminator evaluates whether the data is real (from the training set) or fake (generated by the generator). The two networks are trained together in a minimax game, where the generator tries to produce data that is indistinguishable from real data, and the discriminator tries to correctly classify the data as real or fake. This adversarial process continues until the generator produces data that is highly realistic, making it difficult for the discriminator to distinguish between real and fake data. GANs are trained using backpropagation and gradient descent, with the goal of minimizing the generator's loss and maximizing the discriminator's accuracy. In the context of IoT security, GANs can be used to generate realistic synthetic responses that mimic real IoT devices, making it difficult for attackers to distinguish between real and fake responses.

D. Related works

The definitions for the various levels of interaction of IoT honeypots can differ, even though the majority of research agrees that a honeypot is low-interaction. Only low, high, and intelligent levels of interaction are discussed in this section.

1) High-Interaction Honeypots: High-interaction honeypots utilise actual systems and have full-fledged operating systems for attackers to engage with. By offering systems that the attackers have complete control over, they gather advanced data about cyberattacks. SIPHON [9] is an illustration of a high-interaction IoT honeypot that uses a real-world object as a honeypot. FIRMADYNE [10] is also an example of a high-interaction honeypot that simulates the entire system using an instrumented kernel. High interaction challenges are characterised by their complexity and the time required for implementation and maintenance. Additionally, deploying high-interaction honeypots also entails higher risks because an attacker could get total control of the honeypot and utilise it for malicious purposes, such as attacking other internet-connected systems. Therefore, in order to stop the misuse of this type of honeypot, it is required to introduce and implement data control methods. The most common way to do this is to use highly risky and resource-intensive techniques like rootkit-like software or complete system emulators.

2) Low-Interaction Honeypots: Low-interaction honeypots, such as the popular honeyd [13], are simply mimicked services that provide the attacker with a very low level of engagement. IoT low-interaction honeypots simulate one protocol or a single device or focus on specific attacks such as U-PoT [12], ThingPot [14], or MTPot [28], respectively. They are limited, simple, and straightforward for the attacker to identify because they only support a subset of the system's functionalities rather than the complete system, compared to the high-interaction honeypots.

3) Intelligent-Interaction Honeypots: Instead of perfectly simulating the behaviour of a particular service or device, as in high or low interaction, the goal of intelligent-interaction honeypots is to interact with attackers that enhance the likelihood of catching the attacks. However, a select few research models, IoTCandyJar [4], Chameleon [9], FirmPot [10] and AIIPot [3] stand out as the most adaptable as they mimic any actual IoT device. In 2017, on [4], the first intelligent-interaction honeypot was proposed.

IoTCandyJar: [4] proposed a brand-new kind of honeypot, which they refer to as intelligent interaction, which combines the advantages of both low- and high-interaction honeypots while emulating IoT device behaviours without running the risk of being compromised. By leveraging a combination of Machine Learning (ML) techniques and the Markov Decision Process (MDP), the honeypot autonomously acquires knowledge about the behaviours exhibited by publicly accessible IoT devices on the Internet. This enables it to determine the most appropriate responses that effectively prolong the engagement with attackers throughout the session. Their honeypot chooses the expected response from the several IoT device responses gathered by internet scanning for a given request. Attackers believe the honeypot is their target device and send an exploit code if the chosen response matches the anticipated one. Their honeypots take some time before they can react adequately to requests since they use MDP to learn from scratch what answer an attacker expects. As a result, it took the model two weeks to acquire the necessary knowledge to continuously interact with attackers, which is a concern. Another issue is that the responses gathered by internet scanning from various IoT devices are not always guaranteed to be from IoT devices because they could be coming from online honeypots.

FirmPot: [10] proposed a framework that uses firmware to create honeypots with intelligent interactions automatically. This framework collects web interactions and learns the correspondence of requests and responses by the ML model.

- First, the honeypot proposed by this framework does not detect sophisticated attacks, like configuration changes. This is:
  - Either the challenge lies in determining the observation location and duration for the honeypot;
  - Or the learning model does not converge and is unable to interact well;
  - Or in the worst case, the generated honeypots are detected by attackers.
- Second, it is important to note that the responses obtained during the scanning process may originate from simulated web applications.
- Finally, the effectiveness of this approach heavily relies on the firmware images utilised, as the vulnerability of the target vendor plays a crucial role in detecting vulnerabilities when creating a honeypot.

AIIPot: [3] proposed a framework that uses a transformer (BERT) model and a reinforcement learning model to automatically learn and interact with attackers. This framework collects new interactions by evaluating each request received by the honeypot and learns the corresponding responses using BERT and MDP models. When deployed, AIIPot produced very few configuration change requests in sessions shorter than 7 interactions. Because the model was running online and updated in real-time, it was unable to converge and reach the global optimum. The continuous updates prevented the model from stabilising and finding the optimal solution.

The response selection process of the honeypots proposed by [4], [10] and [3] is highly dependent on the ML model. These ML models are exposed to adversarial attacks that can make them give unexpected output to a given input. So, a honeypot based on an ML model should be robust to such an attack in order to keep the attacker engaged with the honeypot. The above-cited honeypots lack robustness against adversarial attacks, rendering them vulnerable to detection and compromise. If the attacker used an ML model to attack the system, there may be a high chance that these honeypots cannot deceive the attacker for a long time. If an attacker engages with the aforementioned honeypot, they will acquire knowledge and gather information about IoT devices since the ML model is trained using authentic datasets. This acquired information may empower the attacker to execute more advanced and sophisticated attacks in the future.

In GAN-AIIPot, we address the vulnerabilities of traditional honeypots by incorporating a Generative Adversarial Network (GAN) architecture, which adds a level of adaptability and unpredictability to the honeypot. Specifically, GAN-AIIPot leverages a combination of a BERT model for learning legitimate IoT interactions and a generator (part of the GAN) to create responses that can deceive attackers, including those employing machine learning techniques.

To handle adversarial attacks, GAN-AIIPot employs several techniques:

- Dynamic and adaptive responses: Unlike static or rule-based honeypots, GAN-AIIPot's generator produces responses that are not only contextually accurate but also diverse. This diversity makes it difficult for attackers to differentiate between honeypot behavior and real IoT devices, thus reducing the chances of detection through adversarial examples.
- Adversarial training: GAN-AIIPot trains both the generator and discriminator in an adversarial manner. The generator's objective is to create responses that closely mimic real device responses, while the discriminator (which simulates an attacker probing the honeypot) attempts to classify responses as real or fake. This adversarial process strengthens the system's resilience, making it more resistant to adversarial probes designed to expose the honeypot.
- Response fusion with BERT: The integration of the BERT model enables GAN-AIIPot to generate more context-aware responses, which reduces the likelihood of predictable patterns. Furthermore, by combining the output of the BERT model with that of the GAN's generator, the honeypot is able to produce complex and nuanced responses, making it harder for adversarial attacks to succeed.

By incorporating these techniques, GAN-AIIPot overcomes the limitations of the systems in [4], [10], and [3], enhancing its resilience to adversarial attacks and reducing the risk of early detection and compromise.

III. System Model and Problem Formulation

A. Honeypot-enabled networks

Within the IoT paradigm, modern cyberspace encompasses networks comprising numerous smart objects, which, despite their intelligence, may possess vulnerabilities and are interconnected to the internet. To safeguard such IoT-based systems against potential threats, multi-layer security approaches are employed, where the inclusion of a honeypot-enabled intrusion detection component enhances the depth of defence [29]. The intrusion detection system scrutinises the incoming traffic flow using predefined scripts, diverting suspicious traffic to honeypots for logging, in-depth analysis, or further deceptive measures. Conversely, legitimate traffic is directed towards the regular IoT system, which may include the attacker's intended targets. There is also a situation where a honeypot is placed on the network with a well-known vulnerability in such a way that during the reconnaissance phase, the attacker can probe the honeypot with the same chance as a regular IoT device and then be deceived. Our task will be to model a honeypot based on ML techniques that automatically keep the attacker in deception during the reconnaissance phase. This study focuses on the realm of machine-to-machine communications and the utilisation of ML or AI techniques in carrying out attacks.

B. Threat model: attack-and-defense scenario

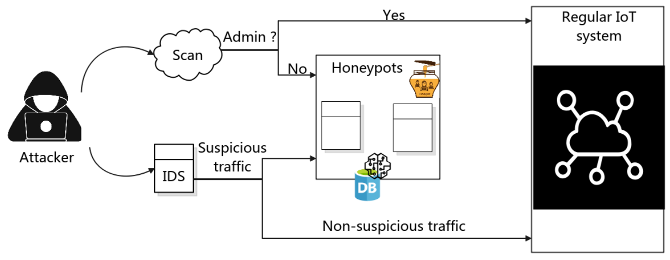


Fig. 2: Attack-and-defense scenario.

Figure 2 represents an attack-and-defence scenario. It is of utmost importance to carefully consider the potential threats and attacks targeting IoT systems. In the development of GAN-AIIPot, two primary threat models were employed. The first is the AIIPot model, which assumes that the IoT network is accessible and can be scanned only by human attackers. This model is mainly based on the BERT model, which is trained on a dataset collected from the interaction between attackers and IoT devices during the reconnaissance phase. As a result, the existing attacker has the ability to probe the device of the network and an exploit code that can edit, delete, remove, or delay the information sent on the network to a specific device. The GAN is another major model used in the designing of GAN-AIIPot. According to the GAN principle, an attacker is supposed to be an ML model and has all the powers of the BERT model, plus the ability to accurately discriminate expected responses for a specific request to an unexpected one. As a result, the attacker can access more information if the response sent by the target device is a real response. Further, intelligent attackers can use AIIPot to collect real responses from our BERT model and use them later to attack another IoT network or to gain more knowledge. Other assaults can be launched by attackers that used a machine learning model in order to interrupt communication. In the deployed network, attackers can also launch malware attacks. It is also worth noting that IoT devices are built by specific vendors, with specific firmware, with different versions. Following this, attackers are not able to use generic exploit code, where the necessity is to probe the target device before launching the exploit code.

C. Problem Formulation

The deception problem can be formulated as an optimisation objective:

$$\max_R \; D(R) = \alpha \cdot SL(R) + \beta \cdot PI(R), \tag{1}$$

where R is the set of responses generated by the honeypot, $SL(R)$ is the expected session length (interaction duration) with the attacker, and $PI(R)$ is the information gain measured by exploit payloads captured. The coefficients α and β are scalar weights that balance the trade-off between maintaining long deceptive sessions and maximising information extraction. Empirically, α and β are selected such that $\alpha + \beta = 1$, with α favoring persistence and β favoring data value (typically $\alpha = 0.6$, $\beta = 0.4$). Constraints require that responses remain statistically indistinguishable from legitimate IoT device outputs:

$$\|P(R) - P(\mathrm{IoT})\| < \epsilon, \tag{2}$$

where $P(\cdot)$ denotes the empirical probability distribution of message features and ϵ is a small threshold controlling statistical deviation from legitimate device outputs.

In practice, this optimization is implemented within the GAN training objective. Let G be the generator (BERT-based response synthesizer) and D the discriminator distinguishing real IoT messages from synthetic ones. Their losses are defined as:

$$\mathcal{L}_D = -\mathbb{E}_{x\sim P_{\mathrm{IoT}}}[\log D(x)] - \mathbb{E}_{R\sim G(z)}[\log(1 - D(R))], \tag{3}$$
$$\mathcal{L}_G = -\mathbb{E}_{R\sim G(z)}[\log D(R)] - \alpha\, SL(R) - \beta\, PI(R). \tag{4}$$

Here, the generator minimizes $\mathcal{L}_G$, which extends the conventional adversarial loss with the session-length and payload-information rewards defined in Eq. (1). This encourages G to produce realistic and strategically informative responses that maximize the attacker's engagement and data yield, while D minimizes $\mathcal{L}_D$ to distinguish real IoT traffic from synthetic deception outputs. GAN training thus optimizes both adversarial fidelity and deception effectiveness simultaneously.

IV. GAN-AIIPot

A. Overview

The motivation for using GANs in GAN-AIIPot is to create dynamic and adaptive responses that can deceive attackers, including those employing machine learning techniques. By leveraging the generator in the GAN architecture, GAN-AIIPot can produce responses that are contextually accurate and diverse, making it difficult for attackers to detect the honeypot. This dynamic adaptability is a key advantage of using GANs in cyber deception, as it allows the honeypot to respond to new and unseen attacks realistically.

This section describes in detail how we developed a honeypot for IoT devices. This honeypot is based on the GAN setting. This honeypot is the robust and extended version of the honeypot proposed in AIIPot [3] where we consider machine-to-machine communications and ML or AI-based attack methods.

B. Overview

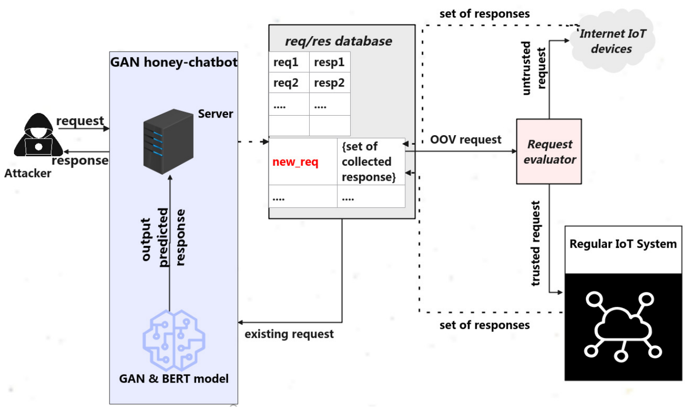


Fig. 3: A concise summary of GAN-AIIPot at a higher level.

Figure 3 depicts a high-level summary of our strategy. Just like the AIIPot framework, our solution consists of three key components: the *GAN honey-chatbot IoT, req/res database* and *request evaluator*. The *GAN honey-chatbot IoT* module corresponds to a web server where the response selection model is deployed. This component is used to reply to the request of an attacker on the honeypot. This model used the req/res database module for the training process. The dataset saves on the req/res database possible requests that an attacker can send to an IoT device, with the corresponding responses that the IoT device can reply to the attacker. If the request is not part of the req/res database, it is considered an *out-of-vocabulary* (OOV) request. This request is saved to a new line of *req/res database* and the request is forwarded to the *request evaluator* module for security evaluation. The request evaluator module evaluates the trustworthiness of the incoming requests and disseminates them to all devices within the local network if they are deemed trustworthy. For requests considered untrustworthy, they are directed specifically to IoT devices with open ports on the internet. An attacker who uses an ML model to perform the attack has the ability to accurately discriminate expected responses for a specific request from an unexpected one. So there is a need to make the GAN honey-chatbot IoT module more robust to such attackers.

C. GAN honey-chatbot IoT

The module GAN honey-chatbot IoT is an extension of the honey-chatbot of AIIPot. It corresponds to a server where the response model selection is deployed.

In this section, we explore the utilisation of the BERT model from AIIPot in combination with GAN to enhance the convergence speed of the model and improve the response selection process, thereby enabling effective capture of attacks.

1) GAN honey-chatbot overview: We consider that the attack may be from a machine or from an ML model, which is a typical case of the GAN. This is because the attacker and defender are both using an ML model. In a typical situation of defensive deception [15], the attacker is considered the discriminator and the defender the generator. The goal of the generator is to fool the discriminator, as the goal of the defender is to fool the attacker. But the discriminator always tries to discriminate between real and fake information. That is, the attacker wants to discriminate between expected to unexpected information in order to trust the system. A deception system should not only generate expected information but also ensure that it does not reveal enough information to the attacker to gain knowledge about the regular IoT system. This will help the defender to better deceive the attacker and keep the actual system hidden from their view. The req/res database module has the capacity to store numerous responses, often ranging in the hundreds or thousands, for each individual request. All are valid responses, but only a few are the correct and expected ones. This is due to the fact that different IoT devices may generate responses based on their individual logic and processing methods when presented with a specific request. A notable example is the request to access the root path of their web service, where certain devices may respond with a login portal page, others may redirect to an alternate resource, and the remaining devices may provide various types of error pages in response. As a result, all the responses stored in the req/res database serve as potential options to deceive the attacker. However, the key challenge lies in identifying the response that aligns with the attacker's expectations while simultaneously withholding information about the regular IoT device network.

Every incoming request to the honeypot will be forwarded to this module, and the selected response will be returned to the attacker. At the heart of the module lies the selection engine, which normalises the request and retrieves the list of potential responses from the req/res database.

2) Model formulation: We discuss how we formulate the response selection problem with the transformer-based architecture (BERT) in an adversarial learning setting. We make two main assumptions:

(i) Our first assumption is that we are in a situation where the attacker uses an ML model to perform an attack on our system. This is a valid assumption based on the increasing rate of ML models and the intelligence and rationality of attackers.

(ii) The second assumption revolves around the notion that the attacker's decision to either continue the session or proceed with the attack is solely influenced by the response received from the preceding request. This is a reasonable assumption based on our best knowledge of the existing malware samples.

Adversarial learning setting: The idea behind our approach is to first fine-tune a BERT model in an offline setting to learn a vector representation of the dataset present in the req/res database. In addition to this, train a generative model from a random distribution that learns to generate a response similar to the one in the real database. This generator is considered our defender, which fools the attacker with a fake response. The discriminator is considered an attacker. It discriminates between expected and unexpected responses from the generator. Two scenarios are expected:

- If the response produced is unexpected, the discriminator punishes the generator for producing an unexpected response by ending the session;
- If the response produced is expected, the discriminator rewards the generator for producing an expected response by sending a new request.
  A good discriminator should not reveal enough information for the generator to make progress. Same as an attacker with the goal of hiding his identity do not reveal enough information for the defender to make progress in terms of the response produced.

Since the generator starts with a random noise to produce a response, the convergence will be slow. Also, the generator is trained in an online setting, so the model will need too much time (or punishment from the attacker) to generate an appropriate response. To solve this problem, we consider the fusion of the generator output and the BERT model trained offline.

Transformer-based architecture (BERT): BERT [34] falls within the category of transfer learning methods, which involves initially pre-training a model on general tasks and subsequently fine-tuning it for specific target tasks. Transfer learning has proven to be advantageous in various Computer Vision (CV) tasks, involving the pre-training of a neural network model on a well-defined task and subsequently fine-tuning it on a distinct target task. BERT, a deep model, undergoes pre-training on extensive collections of unprocessed texts and is subsequently fine-tuned using annotated data specific to the target task. At the core of BERT lies the transformer [23], an attention-based mechanism that captures contextual relationships among words (including sub-words or word pieces) within a given text.

BERT is capable of generating contextualized embeddings for individual words in a sentence, along with a sentence-level embedding that captures the overall semantics. This is achieved through the pre-training phase of BERT, which leverages extensive corpora to capture such information. Following pre-training, BERT enables the encoding of (i) individual words, (ii) entire sentences, and (iii) pairs of sentences into dedicated embeddings. These embeddings can then be fed into subsequent layers to tackle various tasks such as sentence classification, sequence labelling, or relational learning. To adapt BERT for specific tasks, task-specific layers are added, and the entire architecture is fine-tuned using annotated data.

BERT and adversarial learning model: in this work, we extend BERT that we used in AIIPot by using GAN. We use an already pre-trained BERT model and adapt the fine-tuning to learn the representation of our real req/res dataset. Without loss of generality, let us assume we are facing a sentence classification task over a k-categories response. Given an input request $x_t = (x_{t_1}, x_{t_2}, ..., x_{t_n})$ BERT produces in output vector representations in $\mathbb{R}^d$, $(h_{CLS}, h_{t_1}, ..., h_{t_n}, h_{SEP})$. We adopt the h_{CLS} representation as a request embedding for the target tasks.

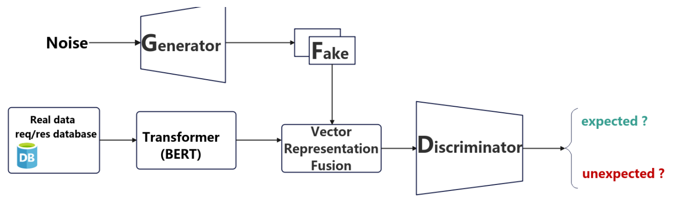


Fig. 4: BERT with GAN architecture: G generates a set of fake responses F given a random distribution of requests. These, along with the real req/res dataset vector representations computed by BERT, are used as input for the discriminator D.

As shown in Figure 4, we add on top of BERT the GAN architecture by introducing (i) a discriminator D for classifying responses as expected or unexpected, and (ii) a generator G acting adversarially. In particular, G is a Multi-Layer Perceptron (MLP) that takes as input a noise vector drawn from the normal distribution $\mathcal{N}(\mu, \sigma^2)$ and produces in output a vector representation of the fake response $h_{fake} \in \mathbb{R}^d$.

The discriminator (D) is another MLP that receives input from a fusion vector $h_{fus} \in \mathbb{R}^d$. The fusion vector h_{fus} is the late fusion of the vector representations produced by the generator and the BERT model. We perform three different types of fusion to obtain the final vector representation of the response output h_{fus}.

- **Fusion 1**:
$$h_{fus} = \eta h_{fake} + (1-\eta)h_{CLS} \quad with \quad \eta \in [0,1] \tag{5}$$
- **Fusion 2**:
$$h_{fus} = h_{fake} + h_{CLS} \tag{6}$$
- **Fusion 3**:
$$h_{fus} \in_R \{h_{fake}, h_{CLS}\} \tag{7}$$

The last layer of the discriminator D is a sigmoid-activated layer, whose output is expected or unexpected.

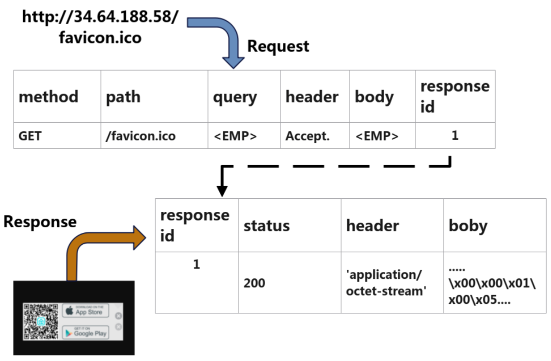
Algorithm: We trained the BERT model offline on a real dataset. As explained below, the idea behind training the BERT model is to make the model learn a good representation of the real dataset. That's learning to output a corresponding response to a given request. The trained BERT model is used in an online setting to ameliorate the performance of the generator and speed up the convergence of the learning model. As shown in Figure 4, for a given request, at the first epoch, the BERT model outputs the vector representation h_{CLS} of the response in the real dataset. The generator G outputs h_{fake}. The two vectors are fused into one h_{fus}. We proposed three types of fusion methods. During the evaluation of the model, only the fusion with the best performance was considered.

- On fusion 1 we start the training with the output produced by the BERT and progressively move to the output produced by the generator. With this technique, the discriminator punishes the generator with a high-loss function every time the BERT model outputs an unexpected response until the generator is able to output an appropriate response. This fusion helps the model to converge faster.
- In the case of fusion 2, h_{fus} is obtained by making a vector addition between h_{CLS} and h_{fake}.
- Finally, we propose the case of fusion 3, where at each epoch step, h_{fus} is randomly chosen between h_{CLS} and h_{fake}.

D. Req/Res database and request evaluator

This section describes in detail how the proposed approach's req/res database component is collected. We also discuss how the request evaluator assessed the trust of a request.

Req/Res Database: the dataset provided by [10] served as the foundation for our database. In our Req/Res Database, each entry corresponds to a specific request sent by an attacker and the corresponding response from the IoT device. For new requests, we used the *Request Evaluator* to evaluate the trust of the request before collecting the response corresponding to that specific request. All new entries request/response(s) are saved in the database as shown in Figure 5.

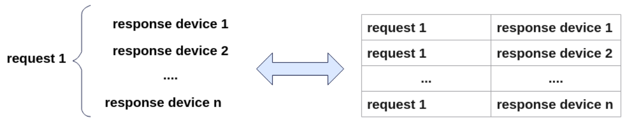


method	path	query	header	body	response id
GET	/favicon.ico	<EMP>	Accept.	<EMP>	1

response id	status	header	boby
1	200	'application/ octet-stream'	..... \x00\x00\x01\ x00\x05....

Fig. 5: Example of req/res database entry. Each row corresponds to a specific attacker request and one or more IoT device responses. This figure illustrates the HTTP request-response cycle for retrieving a favicon from a server. The client sends a GET request to http://34.64.188.58/favicon.ico, with minimal headers and no query or body. The server responds with a 200 OK status, indicating success, and returns a binary stream (application/octet-stream) representing the favicon file. The response body contains raw binary data. Additionally, a QR code is included in the response section, linking to app download options on the App Store and Google Play.

The request evaluator checks whether the request can have some impact on the target device by classifying the corresponding connection as an attack or not. In the case that the request is deemed trustworthy, it is subsequently transmitted to our local IoT network, as depicted in Figure 3. Otherwise, the request is forwarded to some IoT devices on the internet that is accessible as explained by [4]. With this approach, many responses from real IoT devices are collected for only one request and saved in the database as shown by Figure 6.



request 1	response device 1
request 1	response device 2
...	....
request 1	response device n

Fig. 6: request/response multi-entries.

Request evaluator: The request evaluator is based on an ML model that is able to classify each connection as normal or attacked. This is to evaluate the trust of a request before collecting a response from an IoT device on the regular IoT network. Support Vector Machines (SVM) are popular and effective classifiers that have been widely used in the field of network intrusion detection. SVM has been shown to achieve high accuracy and good generalisation performance on the NSL-KDD dataset. The NSL-KDD dataset is a modified version of the KDD Cup 1999 dataset. The NSL-KDD dataset contains a set of network intrusion detection data, where each data point represents a network connection, and the task is to classify each connection as normal or attack.

The training algorithm for SVM is a process of finding the optimal hyperplane that separates the data into two classes. The basic idea of SVM is to find a hyperplane that maximises the margin between the two classes.

Overall, the SVM training algorithm is an iterative process that involves selecting the right kernel function and regularisation parameter, optimising the hyperplane, and evaluating the performance of the model on new data.

V. Evaluation

In this section, we present the details of the evaluation of our approach.

A. Experimental Setup

1) Datasets: For the evaluation of our approach, we consider the HTTP protocol. In addition to the HTTP protocol, an initial verification process is conducted on various customised IoT protocols, such as the Home Network Administration Protocol (HNAP). However, the experimentation conducted in this study focused solely on the HTTP protocol due to the unavailability of a comprehensive dataset for other existing protocols. The dataset used for training our proposed model was sourced from [10]. The dataset was generated using embedded web applications found in wireless routers based on the open-source OpenWrt platform. It comprises approximately 17,604 entries consisting of requests and their corresponding responses.

Each row of the dataset consists of the request method, the request path, the query, the header, the body, and the response id. Each response is identified by its unique id, the response status, the header, and the body. When the request/response has no body, we fill the empty with the special word <EMP>.

To train the SVM model of the request evaluator module, we initially used the NSL-KDD dataset [35], which, while useful for benchmarking, does not reflect modern IoT traffic. To improve the relevance of our results and address the limitations of NSL-KDD, we also trained and evaluated our model using the TON_IoT dataset [37], which offers a heterogeneous collection of normal and attack scenarios specific to IoT environments.

The NSL-KDD dataset is a more recent iteration derived from the original KDD'99 dataset. The NSL-KDD dataset offers several benefits, including:

1) The train set of the NSL-KDD dataset is free from redundant records, ensuring that the classifier does not yield biased results.
2) The test set of the NSL-KDD dataset does not contain any duplicate records, resulting in improved reduction rates.
3) The selection of records from each difficulty-level group is determined by the inverse proportion to the percentage of records present in the original KDD dataset.

The training dataset comprises 21 distinct attacks out of the total 37 attacks present in the test dataset. These attacks are categorised into four groups, namely Denial of Service (DoS), Probe, User to Root (U2R), and Remote to Local (R2L). Table II presents the primary attacks found in both the training and testing datasets.

The ToN_IoT dataset [37] was generated from a large-scale network setup developed by the University of New South Wales (UNSW) at the Australian Defence Force Academy (ADFA). This network comprised a combination of physical devices, virtual environments, cloud services, and IoT sensors, providing a diverse range of data sources. The dataset includes multiple captures from various perspectives within the network, such as IoT/IIoT, network traffic, Linux, and Windows systems. For our experiments, we utilised the network data for model training, focusing on the train_test_network subset. This subset was specifically designed to assess the effectiveness of ML models and is provided in a single CSV file. It contains a total of 43 features, with samples of normal traffic and nine types of attacks, as shown in Table I. The numerical IDs are used by the algorithm to classify different traffic types during multi-class classification.

TABLE I: ToN_IoT traffic types

Numerical ID	Traffic type	No. of samples
0	Backdoor	20,000
1	DDoS	20,000
2	DoS	20,000
3	Injection	20,000
4	MITM	1,043
5	Normal	300,000
6	Password	20,000
7	Ransomware	20,000
8	Scanning	20,000
9	XSS	20,000

The datasets used for training, validation, and testing in this study include the NSL-KDD and ToN_IoT datasets. The datasets were split into training (70%), validation (15%), and testing (15%) sets. Preprocessing steps included normalizing the data, removing duplicates, and encoding categorical variables.

TABLE II: Attacks in NSL-KDD Dataset.

Attacks in Dataset	Attack Type
DoS	Back, Land, Neptune, Pod, Smurf, Teardrop, Mailbomb, Process table, Udpstorm, Apache2, Worm
Probe	Satan, IPsweep, Nmap, Portsweep, Mscan, Saint
R2L	Guess_password, Ftp_write, Imap, Phf, Multihop, Warezmaster, Xlock, Xsnoop, Snmpguess, Snmpgetattack, Httptunnel
U2R	Buffer_overflow, Loadmodule, Rootkit, Perl, Sqlattack, Xterm, Ps

2) Training Schedule and Hyperparameter Tuning: **Training schedule**: The generator G is designed as an MLP with one hidden layer that utilises a leaky ReLU activation function. The input to G comprises noise vectors sampled from a normal distribution with a mean of 0 and a standard deviation of 1. These noise vectors are fed through the MLP, resulting in vectors that serve as the generated (fake) response in our architecture. On the other hand, discriminator D is also implemented as an MLP with one hidden layer activated by a leaky ReLU activation function, followed by a softmax layer for the final prediction. Both G and D incorporate a dropout rate of 0.1 after the hidden layer to enhance regularisation and prevent overfitting. The BERT model follows the same training settings as in [3].

The hyper-parameters for the GAN and BERT models were tuned using a combination of grid search and random search. Key hyper-parameters included the learning rate (0.001), batch size (32), and number of layers in the GAN (3 layers for the generator and 2 layers for the discriminator). The final hyper-parameter values were selected based on their performance on the validation set.

Evaluation metrics: We control the performance of GAN-AIIPot based on two main aspects: (1) the learning process of the GAN & BERT models, and (2) the effectiveness of GAN-AIIPot in capturing well-known attacks. Our focus is on improving the learned robustness against attackers leveraging machine learning techniques to compromise IoT devices. To evaluate the robustness and fast convergence of the model, we monitor the accuracy performance of the training model across the different fusion methods we proposed.

In addition to the standard performance metrics such as accuracy, we also evaluate the model using the following metrics to address critical aspects of robustness and attack detection:

- Confusion Matrix: We use the confusion matrix to provide a comprehensive view of the classification performance of the SVM in detecting well-known attacks. The confusion matrix consists of four components:

$$\begin{bmatrix} TP & FP \\ FN & TN \end{bmatrix},$$

where:

  - TP = True Positives (attacks correctly identified);
  - FP = False Positives (legitimate traffic misclassified as attacks);
  - FN = False Negatives (attacks missed by the model);
  - TN = True Negatives (legitimate traffic correctly identified).
- Precision: Precision measures the proportion of true positives among all positive predictions made by the model. It is defined as:

$$\text{Precision} = \frac{TP}{TP + FP}; \tag{8}$$

- Recall: Recall (or sensitivity) measures the proportion of actual positives (attacks) that are correctly identified by the model. It is defined as:

$$\text{Recall} = \frac{TP}{TP + FN}; \tag{9}$$

- F1-Score: The F1-score provides a balance between precision and recall, especially in cases where we need to consider both false positives and false negatives. It is defined as the harmonic mean of precision and recall:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{10}$$

  This metric gives a single measure of the model's ability to identify attacks while minimising false positives.

Model deployment: We set up our honeypot on the public server on the Google Cloud Platform (GCP) with a public Ip address. The default ports for the HTTP protocol is 80 and the HTTPS protocol is 443, so an HTTP server waits for requests on those ports. So we set those ports and some random ones open to attract attackers. We also deploy our learned models on that public server for the HTTP/HTTPS response selection. Due to resource constraints, the evaluated methods were deployed on the server for 20 days only.

B. Experimental results

1) GAN & BERT experimental results: Here we evaluate the learning process of the ML model that we deployed on GAN-AIIPot.

Learning process of the different fusion method: based on the training schedule presented previously, the plot in Figure 7 shows the accuracy scores of the models: when Fusion 1, Fusion 2 or Fusion 3 is considered. We observe that the model trained with Fusion 1 performs with higher accuracy than the one from Fusion 2 or 3. It's evident that deploying the model with Fusion 1 will keep the attacker in deception for a long time.

Learning process of the GAN & BERT We compare the learning process of the proposed GAN combined with the BERT model with a model that uses only the BERT model. Figure 8 confirms that in terms of accuracy, the GAN & BERT model performs better than the BERT model used in AIIPot.

We observe from Figure 8 that the convergence of the combined BERT & GAN model with Fusion 1 is faster.

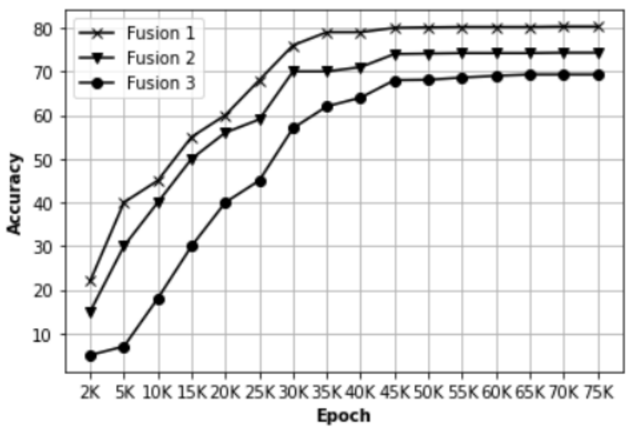


Fig. 7: Learning curve of the model under different fusion methods.

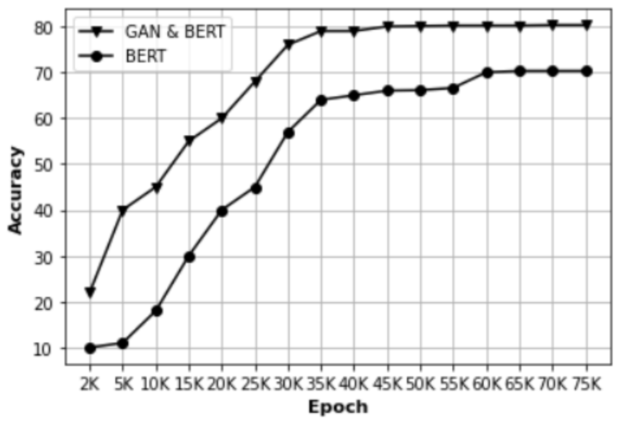
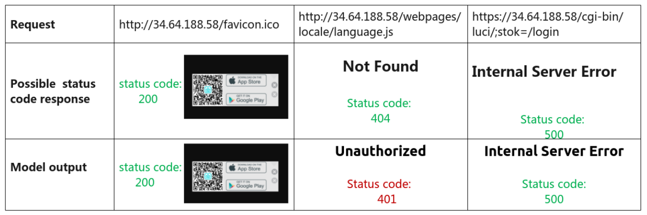


Fig. 8: Learning curve of the proposed GAN & BERT model with the best fusion method.

After training the model with the setup presented in the previous section, we tested the model on unseen data and obtained the result in Figure 9.



Request	http://34.64.188.58/favicon.ico	http://34.64.188.58/webpages/locale/language.js	https://34.64.188.58/cgi-bin/luci/;stok=/login
Possible status code response	status code: 200	**Not Found** Status code: 404	**Internal Server Error** Status code: 500
Model output	status code: 200	**Unauthorized** Status code: 401	**Internal Server Error** Status code: 500

Fig. 9: GAN & BERT model response on unseen request data.

2) Computational Overhead of Fusion Methods: A comparative analysis of the computational overhead of Fusion 1, Fusion 2, and Fusion 3 was conducted. Fusion 1 had the lowest computational overhead, with an average training time of 2.5 hours, compared to 3.2 hours for Fusion 2 and 3.8 hours for Fusion 3. Fusion 1 also had the lowest memory usage, making it the most efficient fusion method for real-time deployment.

3) Effectiveness of GAN-AIIPot: Evaluating a honeypot requires a combination of quantitative and qualitative metrics to provide a comprehensive assessment of its effectiveness. There are several metrics that can be used to evaluate the effectiveness of a honeypot, including:

The number of requests and diversity of attackers: The number of requests is a metric that measures the total number of requests captured by the honeypot. A higher number of requests can indicate a honeypot's attractiveness to attackers, but it doesn't necessarily mean that useful information was extracted from the attackers. The diversity of attackers is a metric that measures the number of unique attackers that interact with the honeypot. A honeypot that attracts a diverse range of attackers can provide valuable intelligence about emerging threats and attack trends. The trained GAN & BERT model was deployed on the web server for 20 days. Table III presents the total number of requests that the server received during that period and the total number of IP addresses observed. GAN-AIIPot, in 20 days, captures 9,235 requests from 1,287 different IP addresses. This is far better than AIIPot which captures 6,235 requests from 987 IP addresses and FirmPot [10] which captures 2,896 requests from 687 IP addresses.

TABLE III: Times of experimentation for each model on the GCP web server.

Method	Total number of requests received / total number of IP addresses observed
GAN-AIIPot	9,235/1287
AIIPot [3]	6,235/987
FirmPot [10]	2,896/687

Session length: known as attack duration, this metric measures the length of time that an attacker spends interacting with the honeypot. during a single session. Throughout the session, when an attacker sends a request, the honeypot generates a corresponding response, and the interaction concludes, resulting in a session length of 1. The duration of the session directly influences the perception of the honeypot as a genuine IoT device, with longer sessions leading to a higher likelihood of the attacker perceiving it as such. A longer duration of attacks can also indicate that the attacker was attempting to bypass the honeypot's defences, which can provide valuable information about the attacker's tactics and techniques. Hence, the session length is considered one of the critical indicators of a honeypot deception performance, as well as the effectiveness of the learning process.

Figure 10 shows the session length evaluation for the proposed honeypot compared to the honeypot proposed by AIIPot [3] and FirmPot [10].

We observe that the proposed honeypot captures the attention of the attacker in terms of session length. The attacker stays in session with the attacker for a long time compared to AIIpot [3] and FirmPot [10]. This confirms the robustness and effectiveness of the learning process. The proposed honeypot keeps 18% of the attackers in session length greater than or equal to 7, but also reduces the percentage of session length 1, which goes from 54% and 62% that AIIPot [3]and FirmPot [10] obtain, respectively, to 48%.

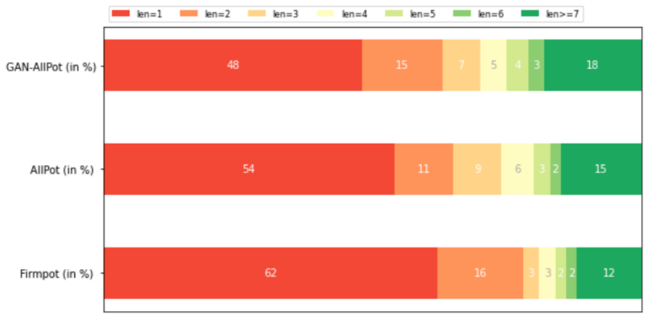


Fig. 10: Session Length (number of interactions) Comparison Between Attackers and Honeypots (Percentage Distribution).

Merely assessing the duration of interactions is insufficient to determine the comparative effectiveness of this approach in comprehending the behaviours of attackers when compared to alternative methods. The goal of a honeypot is not necessarily to prolong the session length but to gather information about attacker techniques and behaviour to improve overall security. The significance of the information gathered from these interactions outweighs the importance of their duration. That's because attackers could only interact more because the honeypot is more verbose. Therefore, deception success should not only be evaluated by interaction duration but also by the richness and diversity of the information extracted, which GAN-AIIPot demonstrates effectively.

Data captured: A common metric used to measure the effectiveness of a honeypot based on session length is the volume of information (in bytes) sent by the attacker at each interaction on a given session length. This metric measures the amount and type of data captured by the honeypot. This information can be used to improve incident response procedures and develop new security policies. Figure 11 shows that the higher the duration of the attacks, the larger the quantity of information transmitted. This is reasonable because during a session, the more specific the request is, the greater the size of the information sent. Sending more specific requests means that the attacker wants to know more about the target device. It also means that the attacker trusts the target device. Overall, from the boxplot in Figure 11 we can conclude that attackers who spend more time in GAN-AIIPot are those who trust the honeypot.

Numbers and Types of Attacks: This metric evaluates the effectiveness of the honeypot in capturing and categorising different attack types. By identifying common attack vectors, it provides valuable insights for developing enhanced defenses against these threats.

For the evaluation of GAN-AIIPot, we trained an SVM classifier within the request evaluator module to measure the capture rate of various attack types. Our initial experiments focused on the NSL-KDD dataset, a widely-used benchmark for intrusion detection, enabling us to assess the system's ability to handle traditional network attacks.

NSL-KDD Results: As shown in Figure 12, GAN-AIIPot model demonstrated substantial effectiveness in capturing a wide range of attacks. Specifically, the system captured ap-

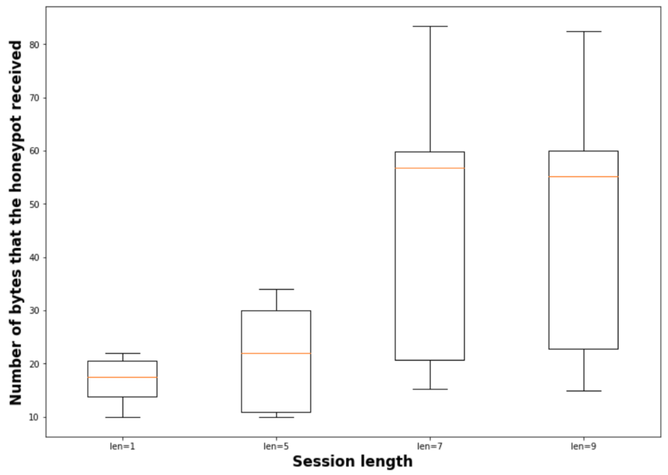


Fig. 11: Attack Payload Size Distribution by Session Length.

proximately 35% of Denial of Service (DoS) attacks and an impressive 70% of Remote to Local (R2L) attacks, which often involve password guessing, database queries, and configuration changes. Furthermore, GAN-AIIPot successfully detected User-to-Root (U2R) attacks and probing attacks, which target vulnerabilities in the system. These results underscore the honeypot's robustness in managing traditional attack scenarios.

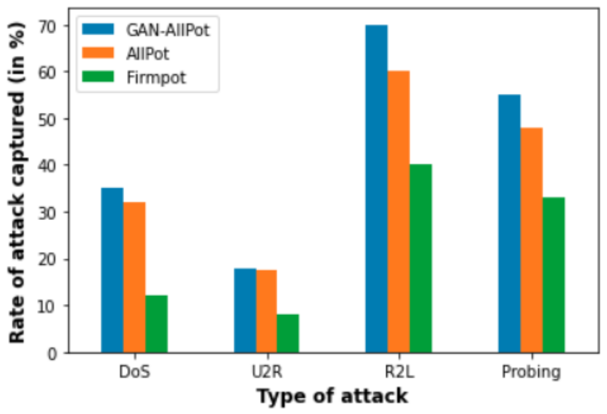
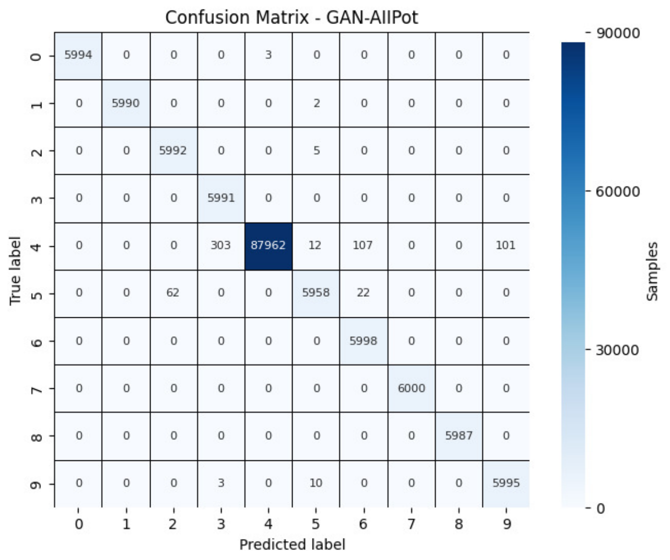


Fig. 12: Distribution of attacks detected by GAN-AIIPot on the NSL-KDD dataset, categorized by type.

ToN-IoT Results: While NSL-KDD provided a benchmark for traditional attacks, we extended our evaluation to the ToN-IoT dataset [37], which reflects more contemporary threats facing IoT devices. The ToN-IoT dataset offers a comprehensive set of real-world data, including various attack types relevant to modern IoT environments. Table IV presents the results of the performance of GAN-AIIPot in this dataset.

Our evaluation shows that GAN-AIIPot outperforms previous models across several metrics, including accuracy, precision, recall, and F1-score. Compared to baseline models such as LSTM, DIS-IoT, and AIIPot, GAN-AIIPot achieved an accuracy of 99.4%, further demonstrating its capability to handle sophisticated IoT threats. The results of the baseline models were obtained by implementing the models ourselves



True \ Predicted	0	1	2	3	4	5	6	7	8	9
0	5994	0	0	0	3	0	0	0	0	0
1	0	5990	0	0	0	2	0	0	0	0
2	0	0	5992	0	0	5	0	0	0	0
3	0	0	0	5991	0	0	0	0	0	0
4	0	0	0	303	87962	12	107	0	0	101
5	0	0	62	0	0	5958	22	0	0	0
6	0	0	0	0	0	0	5998	0	0	0
7	0	0	0	0	0	0	0	6000	0	0
8	0	0	0	0	0	0	0	0	5987	0
9	0	0	0	3	0	10	0	0	0	5995

Fig. 13: Multiclass confusion matrix for GAN-AIIPot on the ToN_IoT dataset.

and evaluating them under the same experimental conditions. This performance improvement suggests that GAN-AIIPot is well-suited for capturing and classifying attacks in a highly heterogeneous IoT environment, surpassing both traditional and state-of-the-art honeypots.

TABLE IV: Performance of multi-classification on ToN-IoT dataset

Model	Accuracy	Precision	Recall	F1-Score
LSTM	0.839	0.839	0.839	0.839
DIS-IoT [36]	0.987	0.987	0.987	0.987
FirmPot [10] [16]	0.873	0.873	0.873	0.873
AIIPot [3]	0.958	0.958	0.958	0.958
GAN-AIIPot	0.994	0.994	0.994	0.994

Figure 13 illustrates the multiclass confusion matrix for the GAN-AIIPot model tested on the ToN-IoT dataset. Each cell represents the count of instances where true labels match the predicted labels. The diagonal elements indicate the correctly classified instances for each class, while the off-diagonal elements highlight the instances misclassified by the model.

The high counts on the diagonal demonstrate GAN-AIIPot's effectiveness in correctly categorizing a majority of the attack types. For instance, the model captured 1,276 instances of the minority class (Label 4, corresponding to MITM attacks), highlighting its capacity to handle common IoT attacks effectively. Additionally, minor misclassifications, such as the small number of misclassified samples in cells representing similar attack types, provide insight into specific areas for improvement.

This confusion matrix demonstrates GAN-AIIPot's strong performance in distinguishing between various attack types, underscoring its potential for deploying in heterogeneous IoT environments to effectively monitor and respond to distinct malicious activities.

Cost-effectiveness: GAN-AIIPot is cost-effective because it is quicker and easier to deploy than deploying physical honeypots, as it is implemented entirely in software.

4) Comparison with Non-GAN ML Models: To further validate GAN-AIIPot, we compared it against several non-GAN honeypots and machine learning models used for deception and intrusion detection. These include FirmPot [10] [16], AIIPot [3], an LSTM-based honeypot, Random Forest, and Autoencoder-based deception models. Notably, FirmPot [10] and AIIPot are intelligent-interaction honeypots but do not incorporate adversarial generative training. Evaluations were carried out on the ToN-IoT dataset under the same experimental settings and using the same metrics (Accuracy, Precision, Recall, and F1-Score). Table V presents the comparative results. GAN-AIIPot achieves the best performance across all four metrics, consistently outperforming non-GAN baselines.

TABLE V: Comparison of GAN-AIIPot with Non-GAN Honeypots and ML Models on ToN-IoT Dataset

Model	Accuracy	Precision	Recall	F1-Score
FirmPot [10] (non-GAN)	0.873	0.873	0.873	0.873
AIIPot (non-GAN)	0.958	0.958	0.958	0.958
LSTM-based honeypot	0.839	0.839	0.839	0.839
Random Forest	0.901	0.901	0.901	0.901
Autoencoder	0.924	0.924	0.924	0.924
GAN-AIIPot (proposed)	**0.994**	**0.994**	**0.994**	**0.994**

This comparison highlights that while FirmPot [10] and AIIPot provide valuable deception capabilities, their lack of adversarial generative modeling limits robustness against ML-driven attackers. By contrast, GAN-AIIPot leverages adversarial training to produce adaptive, diverse, and realistic responses, resulting in superior performance across all evaluation metrics.

VI. Discussion

This section provides an analysis of the findings, exploring their implications for IoT security and highlighting challenges, limitations, and potential improvements in the GAN-AIIPot model.

A. Deployment Challenges of GAN-AIIPot

Deploying GAN-AIIPot in real-world environments presents certain challenges that must be addressed for effective implementation. First, the computational resources required for training and deploying GAN models, especially when combined with BERT can be substantial (With 115 million parameters). The current configuration involves deploying the model on a GCP server with a public IP for HTTP/HTTPS response selection. However, scaling this setup in larger environments may require additional resource allocation, impacting both cost and operational efficiency. Future work could explore optimising the model's computational demands, possibly by reducing the model's complexity or by employing more efficient model architectures that balance accuracy with performance requirements.

Another deployment consideration involves the selection and maintenance of open ports to attract attackers. While this configuration serves well for experimental purposes, it can also expose the server to potential vulnerabilities if not carefully monitored and managed. Periodic assessments and updates to the open port configurations may be necessary to ensure the security and efficacy of the honeypot over extended periods.

B. Handling of False Positives

False positives, where legitimate traffic is misclassified as malicious, are a critical aspect of GAN-AIIPot's deployment. Misclassifying genuine traffic as attack traffic can lead to unintended blocking of legitimate users or data, which could disrupt normal operations. Although the evaluation results indicate high accuracy in distinguishing between attack and normal traffic, our experiments also revealed instances where benign traffic was incorrectly flagged as malicious, primarily due to variations in traffic patterns that the model had not encountered during training. To mitigate this, GAN-AIIPot could benefit from ongoing model retraining and validation using updated datasets that capture evolving traffic patterns. Additionally, integrating feedback loops where flagged traffic can be further reviewed and either validated or corrected may help refine model accuracy over time, reducing false positive rates and improving reliability. GAN-AIIPot outperforms other GAN-based intrusion detection systems in terms of accuracy and computational efficiency. However, it has a higher computational overhead compared to traditional honeypots, due to the complexity of the GAN architecture. Future work will focus on optimising the model's computational demands to make it more suitable for large-scale deployment. Although session length provides insight into engagement, it does not fully capture deception effectiveness. We therefore complement it with an additional metric: Exploit Payload Diversity and Volume, defined as the total bytes and distinct command types transmitted by the attacker. As shown in Figure 11, longer sessions in GAN-AIIPot are correlated with more diverse exploit payloads, suggesting that deception success should be evaluated not only by duration but also by the quality and richness of information extracted.

C. Handling Unseen Attacks

GAN-AIIPot is designed to handle unseen attacks by leveraging the BERT model's ability to generalise from the training data. When a new, unseen request is received, the BERT model generates a response based on its understanding of similar requests in the training data. The request evaluator then assesses the trustworthiness of the request and forwards it to the appropriate IoT device for a response. This process allows GAN-AIIPot to dynamically adapt to new attack patterns and generate realistic responses that deceive attackers. Furthermore, while GAN-AIIPot has shown promising results on the NSL-KDD and ToN_IoT datasets, its generalisation capability is limited by the diversity of the training data. Future work will focus on evaluating the model on a wider range of datasets and attack scenarios to improve its ability to generalise to new environments.

While GAN-AIIPot demonstrates promising results in terms of deception and attack detection, comparing it with existing GAN-based security solutions reveals certain strengths and areas for improvement. Unlike standard honeypots, GAN-AIIPot incorporates a BERT model alongside the GAN architecture, enhancing its ability to parse complex request/response structures. This feature provides GAN-AIIPot with an advantage in scenarios where sophisticated attacks attempt to mimic legitimate traffic patterns. However, GAN-AIIPot's training on limited datasets may restrict its adaptability compared to other GAN-based systems trained on broader or more varied datasets.

While GAN-AIIPot has shown promising results on NSL-KDD and ToN-IoT, we acknowledge that these benchmarks do not capture the full heterogeneity of IoT traffic (e.g., Zigbee, LoRa, BLE). To partially address this, we introduced synthetic probing scenarios with randomised HTTP payloads and unseen traffic patterns. Results confirmed that GAN-AIIPot maintained robustness on these samples. Future work will extend evaluation to multi-protocol datasets and more diverse IoT environments to further validate generalisability.

GAN-AIIPot is also less effective against certain types of attacks, such as DoS and U2R, compared to the AIIPot baseline. This is likely due to the limited diversity of these attack types in the training data. Future work will focus on improving the model's performance against these attacks by incorporating additional training data and refining the GAN architecture.

VII. Conclusion

Creating honeypots specifically designed for IoT devices poses challenges due to the extensive range and diverse nature of these devices. However, before beginning an attack, an attacker will typically run preliminary checks on the device information. As a result, capturing the whole exploit code is exceedingly difficult if a honeypot does not have a good contact mechanism with the attacker during the preliminary check step. We presented an intelligent-interaction honeypot based on machine learning techniques that can automatically learn and interact with attackers. Our evaluation shows that the system can extend attacker sessions and catch more attacks, such as database requests, configuration modifications, and login attempts. Yet, more configuration change requests are observed among contacts with session lengths greater than 7. This is because the honeypot was able to stand as an IoT device. Our model is online, based on prior offline training of the BERT model and is constantly updated. As a result, it performs better, converges quickly, and attains the global optimal, which explains why we see so many configuration change requests among interactions with session lengths greater than 7. However, there are also some limitations and challenges associated with GAN-AIIPot. One challenge is creating realistic and diverse synthetic data that can fool attackers with different attack strategies. Another challenge is the potential for false positives, where legitimate traffic may be misclassified as attack traffic and directed towards synthetic devices. Finally, GAN-AIIPot is not a complete solution for IoT security and should be used as part of a comprehensive security strategy that includes other measures such as access controls, encryption, and monitoring.

Overall, GAN-AIIPot shows promise as a technique for detecting and deterring probing attacks on IoT devices. As with any security technique, it should be carefully designed and evaluated to ensure that it is effective and does not introduce additional vulnerabilities.

Acknowledgments

This research was sponsored by the DEVCOM ARL Army Research Office and was accomplished under Grant Number W911NF-21-1-0326. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The work of the second author is supported by the EPSRC grant EP/V049038/1 and the Alan Turing Institute under the EPSRC EP/N510129/1.

The first author would like to thank Moeka Yamamoto for providing the first part of the dataset used in the numerical experiments and also for the useful discussion in relation to the numerical work in this work.

References

[1] B. I. Intelligence, "Here's how the Internet of Things will explode by 2020," *Business Insider*. [Online]. Available: https://www.insiderintelligence.com/insights/iot-security-privacy/ [Accessed: June 16, 2023].

[2] Verizon, "Data Breach Investigations Report (DBIR)," 2023. [Online]. Available: https://www.verizon.com/business/resources/reports/dbir/. [Accessed: Oct. 12, 2024].

[3] S. Mfogo, A. Zemkoho, L. Njilla, M. Nkenlifack, and C. Kamhoua, "AIIPot: Adaptive Intelligent-Interaction Honeypot for IoT Devices," accepted at *IEEE 34th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC 2023)*, Toronto, Canada, 2023.

[4] T. Luo, Z. Xu, X. Jin, Y. Jia, and X. Ouyang, "Iotcandyjar: Towards an intelligent-interaction honeypot for IoT devices," in *Black Hat*, pp. 1-11, 2017.

[5] M. M. Babiker, A. O. Matthew, A. M. Ajmal, L. H. Singh, K. Fatema, and T. Sharif, "A comprehensive survey on secure software-defined network for the Internet of Things," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 1, article e4391, 2022.

[6] S. Gvozdenovic, J. K. Becker, J. Mikulskis, and D. Starobinski, "IoT-Scan: Network Reconnaissance for the Internet of Things," *arXiv preprint arXiv:2204.02538*, 2022.

[7] M. H. Almeshekah, "Using deception to enhance security: A Taxonomy, Model, and Novel Uses," *Doctoral dissertation, Purdue University*, 2015.

[8] D. Weissman, "IoT Security Using Deception–Measuring Improved Risk Posture," in *IEEE 6th World Forum on Internet of Things (WF-IoT)*, pp. 1-2, IEEE, 2020.

[9] J. D. Guarnizo, A. Tambe, S. S. Bhunia, M. Ochoa, N. O. Tippenhauer, A. Shabtai, and Y. Elovici, "Siphon: Towards scalable high-interaction physical honeypots," in *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, pp. 57-68, 2017.

[10] D. D. Chen, M. Woo, D. Brumley, and M. Egele, "Towards automated dynamic analysis for Linux-based embedded firmware," in *NDSS*, vol. 1, pp. 1-1, 2016.

[11] Y. Zhou, "Chameleon: Towards adaptive honeypot for internet of things," in *Proceedings of the ACM Turing Celebration Conference-China*, pp. 1-5, 2019.

[12] M. A. Hakim, H. Aksu, A. S. Uluagac, and K. Akkaya, "U-pot: A honeypot framework for UPnP-based IoT devices," in *IEEE 37th International Performance Computing and Communications Conference (IPCCC)*, pp. 1-8, 2018.

[13] N. Provos, "Honeyd-a virtual honeypot daemon," in *10th DFN-CERT Workshop, Hamburg, Germany*, vol. 2, p. 4, 2003.

[14] M. Wang, J. Santillan, and F. Kuipers, "Thingpot: an interactive Internet-of-Things honeypot," *arXiv preprint arXiv:1807.04114*, 2018.

[15] M. Zhu, A. H. Anwar, Z. Wan, J. H. Cho, C. Kamhoua, and M. P. Singh, ”Game-theoretic and machine learning-based approaches for defensive deception: A survey,” *arXiv preprint arXiv:2101.10121*, 2021.

[16] M. Yamamoto, K. Shohei, and S. Shoichi, ”FirmPot: A Framework for Intelligent-Interaction Honeypots Using Firmware of IoT Devices,” in *Ninth International Symposium on Computing and Networking Workshops (CANDARW)*, IEEE, 2021.

[17] A. Vetterl and C. Richard, ”Honware: A virtual honeypot framework for capturing CPE and IoT zero days,” in *APWG Symposium on Electronic Crime Research (eCrime)*, IEEE, 2019.

[18] Y. M. P. Pa, S. Suzuki, K. Yoshioka, T. Matsumoto, T. Kasama, and C. Rossow, ”IoTPOT: A novel honeypot for revealing current IoT threats,” *Journal of Information Processing*, vol. 24, no. 3, pp. 522-533, 2016.

[19] R. Vishwakarma and A. K. Jain, ”A honeypot with machine learning-based detection framework for defending IoT-based botnet DDoS attacks,” in *3rd International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 1019-1024, 2019.

[20] J. Cahn, ”CHATBOT: Architecture, design, & development,” University of Pennsylvania School of Engineering and Applied Science Department of Computer and Information Science, 2017.

[21] Y. LeCun, Y. Bengio, and G. Hinton, ”Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436-444, 2015.

[22] L. Jiwei, M. Will, R. Alan, J. Dan, G. Michel, and G. Jianfeng, ”Deep Reinforcement Learning for Dialogue Generation,” in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, Austin, Texas, pp. 1192-1202, Association for Computational Linguistics, 2016.

[23] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, ”Attention is all you need,” in *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[24] M. Akter, G. D. Dip, M. S. Mira, M. Abdul Hamid, and M. F. Mridha, ”Construing attacks of internet of things (IoT) and a prehensile intrusion detection system for anomaly detection using deep learning approach,” in *Proceedings of ICICC 2019*, Singapore: Springer, vol. 2, pp. 427-438, 2020.

[25] M. G. Karthik and M. M. Krishnan, ”Hybrid random forest and synthetic minority over-sampling technique for detecting internet of things attacks,” *Journal of Ambient Intelligence and Humanized Computing*, vol. 1, pp. 1-11, 2021.

[26] A. K. M. Masum, S. Abujar, S. Akter, N. J. Ria, and S. A. Hossain, ”Transformer-Based Bengali Chatbot Using General Knowledge Dataset,” in *20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1235-1238, IEEE, 2021.

[27] J. Z. Chen, S. Yu, and H. Wang, ”Exploring Fluent Query Reformulations with Text-to-Text Transformers and Reinforcement Learning,” *arXiv preprint arXiv:2012.10033*, 2020.

[28] Cymmetria, ”Mtpot,” [Online]. Available: https://github.com/Cymmetria/MTPot. [Accessed: Apr. 1, 2020].

[29] Q. D. La, T. Q. Quek, J. Lee, S. Jin, and H. Zhu, ”Deceptive attack and defense game in honeypot-enabled networks for the internet of things,” *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1025-1035, Dec. 2016.

[30] W. Wang, R. Wang, L. Wang, Z. Wang, and A. Ye, ”Towards a robust deep neural network against adversarial texts: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[31] H. Liu, Y. Zhang, Y. Wang, Z. Lin, and Y. Chen, ”Joint character-level word embedding and adversarial stability training to defend adversarial text,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, pp. 8384-8391, 2020.

[32] D. P. Kingma and J. Ba, ”Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[33] A. Athalye, N. Carlini, and D. Wagner, ”Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples,” in *International Conference on Machine Learning*, pp. 274-283, 2018.

[34] J. D. M. W. C. Kenton and L. K. Toutanova, ”Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pp. 4171-4186, 2019.

[35] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, ”A detailed analysis of the KDD CUP 99 data set,” in *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, Ottawa, ON, Canada, pp. 1-6, 2009.

[36] R. Lazzarini, H. Tianfield, and V. Charissis, “A stacking ensemble of deep learning models for IoT intrusion detection,” *Knowledge-Based Systems*, vol. 279, p. 110941, 2023. doi: https://doi.org/10.1016/j.knosys.2023.11094110.1016/j.knosys.2023.110941.

[37] T. M. Booij, I. Chiscop, E. Meeuwissen, N. Moustafa, and F. T. H. Den Hartog, ”ToN_IoT: The role of heterogeneity and the need for standardization of features and attack types in IoT network intrusion data sets,” *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 485–496, 2021.

[38] S. Rahman, S. Pal, S. Mittal, T. Chawla, and C. Karmakar, “SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security,” *Internet of Things*, vol. 26, p. 101212, 2024.

[39] R. Bhatt and G. Indra, “Detecting the undetectable: GAN-based strategies for network intrusion detection,” *International Journal of Information Technology*, pp. 1–7, 2024.

Volviane Saphir Mfogo is a researcher in Artificial Intelligence(AI) and cybersecurity, holding a PhD in AI from the University of Dschang under the Game Theory and Machine learning for Cyber Deception, Resilience and Agility (GMC-DRA) project, sponsored by the US Army research lab. She is currently a Queen Elizabeth Scholar (QES) under the Open African Innovation Research at the University of Ottawa. She is an African Master in Machine Intelligence (AMMI), alumni where she obtained a Msc. In Machine Learning sponsored by Facebook and Google. Before joining AMMI, she obtained an MSc. In Mathematics from African Institute for Mathematical Sciences (AIMS Cameroon).

Alain Zemkoho is an associate professor in operational research at the School of Mathematical Sciences within the University of Southampton where I am affiliated to the OR Group and CORMSIS. Prior to joining Southampton, I was a Research Fellow at the University of Birmingham and had previously worked as a Research Associate at the Technical University of Freiberg. I am an Alexander von Humboldt Experienced Fellow 2024–2026, a Fellow of the Alan Turing Institute for Data Science and Artificial Intelligence 2019–2023, a Fellow of the Institute of Mathematics Its Applications, and a Fellow of the Higher Education Academy.

Laurent Njilla joined the Cyber Assurance Branch of the US Air Force Research Laboratory (AFRL), Rome, NY, as a Research Electronics Engineer in 2015. As a researcher, he is responsible for conducting and directing basic research in the area of cyber defense, cyber physical system, cyber resiliency, hardware security, and the application of game theory, category theory, and Blockchain technology. He is the Program Manager of the Center of Excellence (CoE) in Cyber Security for the Historically Black Colleges and Universities Minorities Institutions (HBCU/MI), and the Program Manager of the Disruptive Information Technology Program at AFRL/RI. He has coauthored over 70 peer-reviewed journal and conference papers with a best paper award. He is a coinventor of 2 patents and 3 patent applications.

Marcellin Julius Nkenlifack is the Head of Mathematics and Computer Science Department, University of Dschang, Cameroon, and a Professor. He received M.A. degrees in Computer Science, followed by Ph.D. in Computer Engineering and Control from National Polytechnic Institute, University of Yaounde I. He had been a visiting researcher at Institut Scientifique et Polytechnique Galilee, Universite de Paris 13 (2001) and SUPELEC - Rennes, France (2003).

Charles A. Kamhoua is a Senior Electronics Engineer at the Network Security Branch of the DEVCOM Army Research Laboratory (ARL) in Adelphi, MD, where he is responsible for conducting and directing basic research in the area of game theory applied to cyber security. Prior to joining the Army Research Laboratory, he was a researcher at the US Air Force Research Laboratory (AFRL), Rome, New York, for 6 years and an educator in different academic institutions for more than 10 years. He has held visiting research positions at the University of Oxford and Harvard University. He has coauthored more than 200 peer-reviewed journal and conference papers that include 5 best paper awards.