Journal of Namibian Studies, 35 (2023): 254-282 ISSN: 2197-5523 (online)

Special Issue on Engineering, Technology and Sciences

Deep Iot: A Deep Learning Model For Anomaly And Botnet Detection In Iot Networks

Mounira Tarhouni[1], Lamaa Sellami[2], Bechir Alaya*[3], and Pascal Lorenz[4]

[1]Higher Institute of Computer Science and Multimedia Gabes, Gabes University, Tunisia.
[2]Numerical Control of Industrial Processes Laboratory (CONPRIS), National School of Engineers of Gabes, Gabes University, Tunisia.
[3]Department of Management Information Systems and Production Management, College of Business and Economics, Qassim University, 6633, Buraidah, 51452, Saudi Arabia.
[4]Université de Haute-Alsace (UHA), MIPS, France.

Abstract

The Internet of Things (IoT) is currently transforming the world by connecting physical objects to the Internet. However, as the number of connected devices and the growth of IoT continue to rise, new network security threats are emerging due to vulnerabilities in these devices. One prevalent threat is the presence of bot malwares, which exploit vulnerable IoT devices to launch cyber attacks. To address these risks, there is a need for novel methods to detect IoT botnet networks. In this study, we propose a network intrusion detection model that utilizes deep learning, specifically an Autoencoder, to identify malicious botnet traffic. Our model takes a one-class classification approach, focusing on modeling the legitimate behavior of devices within the network to detect anomalies without requiring manual labeling. To analyze device behavior, our solution generates network flows from traffic data and selects relevant flow statistics. We evaluated our approach using the IOT-23 dataset, which includes captures of botnets executed on IoT devices as well as legitimate IoT device traffic. The results demonstrate that our detection model achieves a high predictive performance in identifying different types of botnets, with an impressive F1-score of 93%

Keywords: Internet of Things, Botnets, anomaly detection, Autoencoder, IOT-23.

1. Introduction

The Internet has greatly evolved over recent years, allowing an increasing number of objects to interact with each other or with ourselves. Objects have different sizes, capacities, processing and computing power and support different types of applications, thus contributing to the emergence of the Internet of Things. The term "Internet of Things" or IoT first appeared in 1999 in a speech by British engineer Kevin Ashton [1]. This technology mainly consists of connecting a very large number of everyday objects (phone, watch, surveillance camera, etc.) to the Internet network in order to offer services, through the integration of sensors, actuators and communication capabilities, thus linking the physical world to the virtual world [2]. In fact, IoT has introduced technology into daily life through applications in various fields such as health, smart cities, transportation. Due to advanced Internet technology, IoT applications have become a crucial topic. As manufacturers of connected objects accelerate innovation, more and more malicious activities involving such objects, or even cyber-attacks targeting them, are occurring. As these devices are not as secure as other computing devices, but also participate in security-sensitive tasks, they represent a perfect target for attackers. Among several threats, botnets like Mirai are those that can most benefit from IoT security weaknesses. Malicious botnets are compromised device networks called "Bots" that are remotely controlled by a human operator called the "Botmaster" under a common command and control (C&C) infrastructure. They are used to distribute commands to Bots for malicious activities such as distributed denial of service (DDoS) attacks, spamming, and phishing [3].

Many existing security solutions focus narrowly on specific protocols or device characteristics. Additionally, lack of updates from manufacturers leaves devices vulnerable over time. Given these limitations, we propose addressing the important issue of botnet detection in IoT networks [4].

Our work aims to protect IoT networks from botnets without impacting performance or relying on manufacturers. Most solutions are tied to particular protocols or hardware. Instead, we seek to identify botnet activity through analysis of network behavior across devices. By detecting threats at the traffic level, rather than the level of individual protocols or devices, our approach can provide security agnostically. Without timely patches from makers, vulnerabilities persist. We aim to strengthen protection independent of vendor support

through direct examination of inter-device interactions. Rather than confining solutions to constrained parameters, we aim to recognize botnet command and control infrastructure as it functions within the larger network environment. By taking a holistic view and concentrating on dynamic traffic indicators over static device traits, we hope to close gaps left by targeted or outdated solutions. Our overarching goal is fortifying IoT network security and user security against sophisticated threats like botnets through minimally intrusive traffic analysis. This could help counter a significant risk in a protocol- and vendor-agnostic manner.

Given the promising work and challenges related to IoT networks, the main objective of this work is to secure and protect IoT networks from various malicious botnet activities without affecting network performance. Most of the botnet detection mechanisms proposed in the literature for IoT networks typically address only a few botnets and are applicable to only a specific protocol, and thus are based solely on simulated networks. As a result, a botnet network detection system tested on real traffic data from the IoT-23 dataset is needed, which can detect all malicious activities caused by botnets. Specifically, we aim to develop and evaluate a botnet detection system capable of:

- Identifying a wide range of botnet threats using real-world IoT traffic data, rather than limiting to specific botnets or simulated scenarios.
- Operating effectively across different IoT communication protocols, to provide protocol-agnostic security.
- Protecting network performance by detecting botnet command and control traffic patterns without deep packet inspection of individual device communications.

The goal is to provide a more robust and broadly applicable solution for securing IoT networks against real botnet threats represented in the IoT-23 dataset, without compromising usability or network efficiency. Addressing limitations of existing research will help strengthen cybersecurity for this important domain.

This paper is divided into five main sections. Section 2 provides an in-depth examination of the current state of the Internet of Things (IoT), including its definition, architectural framework, application areas, and underlying technologies. This section also evaluates IoT security concerns, exploring vulnerabilities in connected devices and threats posed by

botnets. Section 3 introduces our proposed approach for detecting botnets within IoT networks. This includes an overview of the dataset utilized as well as a detailed breakdown of the preprocessing, feature extraction, and classifier generation steps involved. Section 4 then presents the experimental findings from applying our methodology. Performance results are reported to demonstrate the effectiveness of the botnet detection system. The final section summarizes the key contributions and conclusions of this work. It also identifies potential avenues for future improvement and extensions to further advance this research area

2. Related Work

Recent technological advancements in the fields of electronics, communications, and Internet development, coupled with the increasing proliferation of mobile devices in our daily lives, have given rise to the Internet of Things (IoT). This new era of the Internet is characterized by networks composed of thousands, or even millions, of objects, forming dense and large-scale networks. In this section, we introduce the concept of IoT, define its scope, examine its architecture, applications, and underlying technologies, while also scrutinizing the threats and vulnerabilities that loom over the IoT. We place a particular focus on botnets and cyberattacks targeting these networks, while reviewing the countermeasures proposed in the literature to safeguard IoT networks against a variety of attacks aimed at disrupting their optimal functioning. The term "Internet of Things" [5] refers to an increasingly extensive network of physical objects connected to the Internet, identified and recognized, just like all other conventional devices we use daily, such as computers, tablets, smartphones, etc. Another definition of the Internet of Things, as per [6], is a dynamic global network infrastructure with self-configuring capabilities based on interoperable communication standards and protocols, where physical and virtual "objects" have identities, physical attributes, virtual personalities, and intelligent interfaces, and they are seamlessly integrated into the information network. It represents the world of interconnected "objects," where humans interact with devices, and devices interact with each other (M2M) [7]. The Internet of Things (IoT) represents the new Industrial Revolution, and a critical decision in this realm is the selection of the communication network (or networks) that will facilitate communication among real-world objects, including machines, equipment, and installations. There

are numerous options available, such as machine-to-machine (M2M), Wi-Fi, Sigfox, ZigBee, among others [8]. There are several ways to architect an Internet of Things (IoT) system. One of the most common architectures divides IoT into three or five layers, as described by [9]. Within an Internet of Things (IoT) system, data is gathered through physical IoT devices in the Perception layer, transmitted for processing in the Transport layer, subjected to analysis employing machine learning and artificial intelligence in the Processing layer, supervised and managed via rule sets and service level agreements in the Application layer, and eventually translated into business intelligence to inform decision-making in the Business layer.

2.1 Vulnerabilities in the Internet of Things (IoT)

IoT has a broad range of application domains, and in their article, Gubbi et al. [10] categorized IoT applications into four areas: 1) personal domain, 2) transportation domain, 3) environment, and 4) infrastructure and public services. A schematic representation of these domains is illustrated in Figure 1. Examples of these domains include industry, healthcare, education, and research. However, in the future, the concept of IoT is expected to be pervasive, available anywhere, anytime, and accessible to everyone.

Since most Internet of Things (IoT) devices are designed for simple tasks, they lack strong security measures, making them vulnerable to weak security standards, and malicious actors take advantage of these vulnerabilities to attempt some of the common IoT attacks, as mentioned by [11]. Many mobile applications designed to control and monitor devices over the network do not support modern security standards such as Secure Socket Layer (SSL) for encrypting communications [12]. Additionally, many IoT devices typically do not allow the resetting of default credentials or the installation of software updates, resulting in an inability to address vulnerabilities. For example, in October 2017, a weakness was revealed in the widely used WPA2 encryption protocol in most modern wireless networks [13].

Figure 1: Application Domains of the Internet of Things [10].

Consequently, IoT is becoming increasingly popular as a powerful tool for cybercriminals. There are classifications to list IoT device vulnerabilities [14]. For instance, the Open Web Application Security Project (OWASP) Internet of Things project is designed to help manufacturers, developers, and consumers better understand security issues associated with IoT. OWASP provides a classification that identifies key attack surfaces, including memory, network, web interface, and more. Table 1 lists the most common IoT vulnerabilities identified by OWASP. A security report on IoT devices by HP in July 2014 found an average of 25 vulnerabilities per device. For instance, 80% of devices did not require sufficiently complex and lengthy passwords, 70% did not encrypt local and remote traffic communications, and 60% contained vulnerable user interfaces and/or firmware [15].

Table 1: Common Internet of Things Vulnerabilities [15].

Vulnerability	**Examples**
Insecure web/mobile/cloud interface	- Inability to change default usernames and passwords : Devices or systems using default credentials do not allow users to change them, exposing them to security risks. - Weak passwords: The use of simple, easily guessable, or commonly used passwords makes accounts vulnerable to brute force attacks. - Lack of robust password recovery mechanisms: The absence of secure password recovery processes can allow unauthorized third parties to reset passwords and gain access to accounts.

	- Exposed credentials: User credentials (username, password) can be exposed to third parties, facilitating unauthorized access. - Lack of account locking: Repeated unsuccessful login attempts are not blocked, making accounts vulnerable to brute force attacks. - Susceptibility to cross-site scripting, cross-site request forgery, and/or SQL injection: These security vulnerabilities allow attackers to execute malicious scripts, forge requests, or inject SQL code to compromise the system or data.
Insufficient Authentication/Authorization	- Privilege escalation: refers to the unauthorized elevation of user privileges - Lack of granular access control: means that the system does not have fine-grained control over who can access specific resources or perform particular actions
Insecure network services	- Vulnerability to denial of service attacks - Buffer overflow - Network ports or services unnecessarily exposed to the Internet
Lack of transport encryption/integrity checking	Without transport encryption, data is vulnerable to interception and tampering during transit, potentially exposing sensitive information to unauthorized access or modification.
Privacy issues	- Collection of unnecessary user data - Personal data exposed - Insufficient controls over access to user data - Lack of data retention limits
Insufficient security configuration	- Inability to separate administrators from users - Weak password policies - No security logging - Lack of data encryption options - No user notification of security events
Insecure software/firmware	- Lack of secure update mechanism - Update unencrypted files - Update unverified files before uploading - Insecure update server - Hard-coded credentials
Poor physical security	- Easy to disassemble device - Access to software via USB ports - Removable storage media

Design vulnerabilities, such as weak or non-existent passwords, can lead to authentication or authorization attacks. For instance, in the case of Phillips Hue connected bulbs [16], the authentication password for controlling the lights is limited to only 6 alphanumeric characters, making it vulnerable to attacks. The Mirai botnet [17] is a notable example of attacks that exploit such vulnerabilities, primarily relying on weak or default passwords [18]. Additionally, Fouladi and Ghanoun [19] demonstrated a vulnerability in the implementation of the Z-Wave key exchange protocol. This vulnerability could

potentially allow an attacker to gain complete control over a Z-Wave door lock by only knowing the home ID and node of the target device. These pieces of information can be identified by monitoring Z-Wave network traffic over a short period due to the frequent polling of devices in a Z-Wave network. These examples underscore the importance of addressing design vulnerabilities and implementing robust security measures to protect IoT devices and networks from potential security threats and unauthorized access. Neglecting security measures by IoT device designers/manufacturers can create vulnerabilities for other household equipment. Consequently, the extensive adoption of Universal Plug and Play (UPnP) [20] within IoT devices becomes susceptible to exploitation for launching attacks. UPnP, a protocol used to open ports on network gateways, grants external individuals control over the devices. This feature is exploited by botnets, which leverage Telnet or SSH to connect to and infect IoT devices.

The implementation of traditional computer security mechanisms in IoT devices is challenging due to their limited computational power [18]. Vulnerabilities, such as buffer overflow in the Bluetooth protocol used by popular IoT devices like Arduino Yun, have been exploited by researchers [21]. The low power of IoT devices prevents the application of memory protection techniques like ASLR [18]. Furthermore, a vulnerability in the Bluetooth Low Energy (BLE) protocol has been discovered, compromising the confidentiality of key exchange due to potential brute-force attacks [22].

2.2 Botnets in IoT

Botnets are networks of compromised devices controlled by malicious software, allowing cybercriminals to gain unauthorized access and control over connected devices by employing specialized Trojan horses, resulting in a global network of compromised devices, primarily targeting unsecured IoT devices, with the Mirai botnet being the largest and most notable example [23][24].

The lifecycle of a bot is illustrated in Figure 2, depicting the typical creation of a botnet in five phases [25][24]: initial infection, secondary injection, connection, malicious command and control, and update and maintenance. In the initial infection phase, the attacker scans a target subnet for known vulnerabilities and infects victim machines using various exploitation methods. In the secondary injection phase, infected hosts execute a shell-code script to retrieve the actual bot binary and install it on the target machine, turning it into a "Zombie" that automatically runs the malicious code upon reboot. The connection phase involves establishing a command and control (C&C) channel, connecting the zombie to it, and integrating it into the attacker's botnet army.

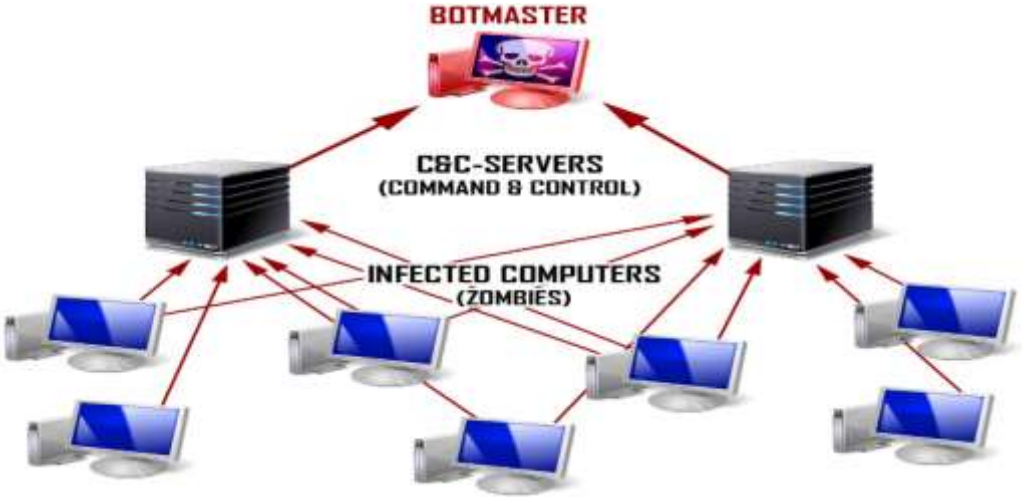


Figure 2. Lifecycle of a bot

The malicious command and control phase is where the botnet's actual commanding and controlling activities take place, with the botmaster using the C&C channel to issue commands to the bot army, which receives and executes them. The maintenance and update phase focuses on keeping the bots active and updated, with botmasters employing dynamic DNS (DDNS) to maintain the botnets' invisibility and portability, allowing frequent server location changes and updates. Various IoT botnets that have undergone testing for botnet detection in IoT networks. These examples shed light on the different types of botnets, their attack strategies, and the specific vulnerabilities they exploit. One such botnet is Mirai, which infects IoT devices and serves a dual purpose of spreading the infection to poorly configured devices and launching DDoS attacks on a targeted server. Mirai establishes communication with a command and control (C&C) server through the Tor network and utilizes a loader to distribute executable files designed for various platforms. Initially, Mirai scans random public IP addresses by targeting TCP ports 23 or 2323. Another botnet called Torii derives its name from its utilization of Tor exit nodes. Torii employs telnet attacks and focuses on extracting sensitive information. It employs a modular architecture that enables it to retrieve and execute additional commands and binaries. Torii communicates with its C&C servers via TCP port 443, utilizing encryption to mimic HTTPS traffic. Okiru, also known as Satori, is a variant of Mirai that exploits a previously unknown vulnerability (CVE-2017-17215) present in Huawei HG532 devices. Okiru injects malicious payloads and inundates its targets with manually crafted UDP or TCP packets. Hajime, on the other hand, is a highly sophisticated and flexible IoT worm that establishes a large peer-to-peer (P2P) botnet. It exploits Mirai's infection methods and employs brute force attacks. Hajime possesses the ability to self-update and swiftly expand its botnet by incorporating additional features. It employs SYN

scanning to discover new victims, targeting TCP ports 23 (Telnet) and 5358 (WSDAPI). Other notable examples include Hide and Seek, which employs complex and decentralized communication techniques while utilizing anti-falsification methods. It spreads like a worm and can perform web exploitation similar to the Reaper botnet. Muhstik leverages web application exploits to compromise IoT devices, using IRC for control and commands. It primarily targets home routers like GPON, DDWRT, Tomato routers, and executes attacks in multiple stages involving payload delivery, downloading an XMRig miner, and incorporating an analysis module to expand the botnet.

Gagfyt targets vulnerable IoT devices such as Huawei and Realtek routers, as well as ASUS devices. It exploits existing vulnerabilities to convert these devices into bots for future DDoS attacks. Gagfyt also reuses certain code modules from Mirai. IRCbot, also known as "Chuck Norris," focuses on vulnerable routers and DSL modems to propagate a worm-like infection. It employs Internet Relay Chat (IRC) for communication with a bot master, utilizing removable drives and instant messaging programs for spreading purposes. Lastly, Hakai, initially based on Qbot, enhances its capabilities by exploiting CVE-2017-17215, which impacts Huawei HG352 routers, Realtek routers, and other vulnerable IoT devices. It incorporates an efficient Telnet scanner to target devices with default or weak passwords.

2.3 Intelligent Intrusion Detection Systems (IDS) in IoT

Research on intelligent Intrusion Detection Systems (IDS) in IoT environments is currently limited compared to traditional networks. The majority of studies focus on simulated networks using the 6LoWPAN protocol, which may hinder their applicability to other IoT protocols.

Botnet detection approaches in IoT environments can be classified into three categories: host-based detection, network-based detection, and hybrid detection. In a host-based detection approach [26], a method employs a One-Class Support Vector Machine (OSVM) model trained with host-based data, such as CPU usage, memory, electrical potential difference, and CPU temperature. Although this approach demonstrates minimal resource consumption, it has only been tested on two types of IoT devices and with a limited set of features.

Another IDS called SVELTE [27] operates on IPv6 networks and utilizes a server module to establish network topology and safeguard sensor systems against internet-based attacks. Meanwhile, the client module provides an IPv6 network topology mapper and detects lost packets. This approach has been evaluated using a software simulator but requires software modification for sensors, which can be challenging for large-scale IoT networks. Nomm and Bahsi [28] propose an

unsupervised learning model for botnet detection. Their model employs one-class classification using methods like Local Outlier Factor (LOF), One-Class SVM, and Isolation Forest (IF) to identify legitimate traffic. However, this method may have a lower detection rate due to the difficulty of generating normal traffic for certain IoT devices. Manzoor and Morgan [29] develop an anomaly-based NIDS that uses a Support Vector Machine (SVM) to predict network traffic as either normal or an attack. Although this approach achieves a precision of 73% when tested against new attacks, it does not address the issue of false alarms. Zeng et al. [30] propose a hybrid detection approach that combines both host-based and network-based detection. They analyze nine features for host analysis and seventeen features for network analysis. This approach demonstrates effectiveness against IRC, P2P, and HTTP botnets. However, the analysis process may be time-consuming due to network violations in P2P bots.

In summary, the development of botnet detection approaches in IoT environments is still ongoing. Network-based approaches appear preferable due to resource limitations and the proliferation of IoT devices. Nevertheless, challenges persist, such as software modification for sensors and the management of false alarms. Further research is necessary to develop efficient and tailored IDS for IoT environments.

3. Proposed Approach

In this section, we introduce our approach to detect botnets in IoT networks while preserving their functionality. We address the need for easy-to-use IoT systems that shield users from the underlying technological complexity and protect against potential threats. Previous discussions highlighted the risks associated with IoT devices connected to the internet and exchanging data, particularly the concern of botnet attacks. We explored existing security solutions but acknowledged their limitations, especially when applied to resource-constrained devices. Our objective is to overcome these challenges and propose an effective botnet detection approach tailored for IoT networks.

3.1 General Description of Proposed Approach

We explore the benefits of traffic analysis in botnet detection and introduce our approach, which incorporates a flow-based traffic analysis model

3.1.1. Traffic Analysis

In traditional botnet detection methods, payload analysis involves inspecting the content of TCP and UDP packets for specific malicious signatures. However, this approach has limitations, including resource-intensive operations, slow

processing speeds, and vulnerability to encryption. To overcome these challenges, we employ traffic analysis as an alternative approach.

Traffic analysis focuses on the overall characteristics of network traffic generated by IoT devices rather than inspecting packet payloads. This approach takes advantage of the notion that legitimate IoT network traffic exhibits a certain level of uniformity and can be characterized by specific attributes. Unlike payload-based methods, traffic analysis is not affected by encryption, and dedicated hardware can efficiently extract relevant traffic information without significantly impacting the IoT network.

Our approach involves examining the traffic flows within a defined time window. By analyzing the patterns and behaviors of these flows, we can identify anomalies associated with botnet activity. We leverage the inherent uniformity present in legitimate IoT network traffic to differentiate between normal and malicious behavior. Specific techniques used in our traffic analysis-based approach may include statistical analysis, machine learning algorithms, or artificial intelligence to identify malicious traffic patterns. By analyzing attributes such as flow duration, packet count, communication patterns, and other relevant features, we can detect and classify botnet activity accurately.

Overall, our traffic analysis-based approach provides a more efficient and effective method for detecting botnets in IoT networks. It overcomes the limitations of payload analysis, such as resource consumption and susceptibility to encryption, by focusing on the characteristics of network traffic. This approach enables us to identify and mitigate botnet threats while preserving the functionality and security of IoT devices.

3.1.2 Overview of the Approach:

Previous research has shown that certain classes of network traffic can be detected by analyzing their traffic patterns. In an IoT network, devices perform specific, repetitive tasks, resulting in a predictable and regular behavior as long as they remain uncompromised. Our approach takes advantage of this by utilizing one-class classifiers to model the normal behavior of IoT devices and detect anomalies, eliminating the need for labeled malicious data for training. To characterize legitimate IoT traffic, we consider static flow-based attributes such as total packet count, flow duration, average and variance of packet sizes, and more. By observing these attributes within a defined time window, we analyze the behavior of a flow. However, selecting an appropriate time window size is crucial. A window that is too small may miss important traffic patterns, while a window that is too large introduces longer detection delays. Our approach consists of two phases: learning and detection. During the learning phase, the detection system is trained using a set of known legitimate data attribute vectors to establish a model

of normal behavior for IoT devices. In the detection phase, the system actively monitors network traffic, generating attribute vectors from active flows and classifying them. If a set of attribute vectors is classified as malicious in real-time data, the corresponding flows are flagged as suspicious.

By leveraging traffic analysis and one-class classification, our approach provides a practical and efficient method for identifying botnet activity in IoT networks. It enables the detection of anomalies without relying on labeled malicious data, and it captures the inherent regularity of legitimate IoT traffic while minimizing false positives.

The proposed methodology, as illustrated in Figure 3, consists of two primary phases:

1- Data Preprocessing Phase: In this phase, we perform four steps. Firstly, we extract the flow features from the raw network capture using Tranalyzer, an open-source flow exporter. This tool calculates statistical flow-based features. The second and third steps involve encoding categorical data and standardizing the features, respectively. Lastly, we select the most representative subset of attributes using an attribute selection method tailored for one-class classification. This helps ensure accurate classification of botnet traffic.

2- Classification Model Construction Phase: The second phase focuses on constructing the classification model based on the selected features. We employ the Autoencoder algorithm, which is a one-class classification algorithm. The Autoencoder learns a compressed representation of normal traffic patterns by training on the chosen attributes. It reconstructs the input data and identifies deviations from normal patterns. Instances that cannot be properly reconstructed are deemed anomalies, potentially indicating botnet activity.

By following this two-phase approach and utilizing the Autoencoder algorithm, our methodology offers an effective solution for detecting botnet traffic in IoT networks. The preprocessing phase ensures the extraction and standardization of relevant features, while the classification phase employs deep learning techniques to identify anomalies associated with botnet activity.

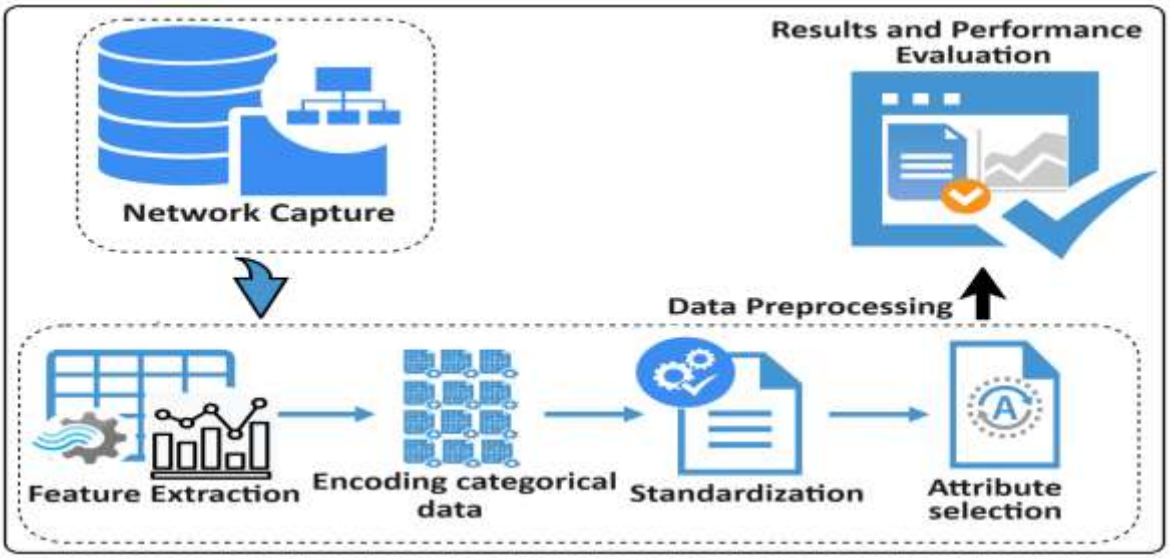


Figure 3. Steps of the proposed approach

3.2 Network capture

The raw network capture data used in our approach is derived from the Aposemat IoT-23 dataset [31]. This dataset aims to provide a comprehensive collection of real and labeled IoT malware infections along with benign IoT traffic. The network traffic was captured at the Stratosphere Laboratory, AIC Group, FEL, CTU University in the Czech Republic.

The IoT-23 dataset consists of twenty-three captures, known as scenarios, representing different types of labeled IoT network traffic. The original PCAP files were processed using the Zeek network traffic analyzer, with a 5-second time window for each capture. The scenarios are categorized into twenty captures of network traffic from IoT devices infected by twelve distinct botnet types, and three captures of legitimate IoT device traffic.

For the malicious scenarios, specific malware was executed on a Raspberry Pi, employing various protocols and performing diverse actions. On the other hand, the network traffic capture for the benign scenarios was obtained from three non-infected, real IoT devices: a Philips HUE smart LED lamp, an Amazon Echo smart home assistant, and a Somfy smart door lock. These devices represent genuine, uninfected hardware, allowing for the capture and analysis of normal network behavior.

Given the extensive size of the IoT-23 dataset, we selected twelve scenarios, one for each botnet type, and the three scenarios representing non-infected real IoT devices. Detailed information about these selected scenarios can be found in Tables 2 and 3.

Table 2: The captures of botnets running on infected Raspberry Pis involve the collection of network traffic, categorized by the number of flows for each specific type, within a defined time interval.

Raspberry PI (Infected)		Malicious traffic type (5 second flow)				
Botnet	Capture Duration (Hrs)	Scan	C&C	DDoS	Attack	e Download
Mirai	24	10114	11816	234	10000	11
Linux Mirai	24	0	6480	55339	0	0
Linux Hajime	24	953	0	0	0	0
Okiru	24	0	15688	0	0	0
Hakai	24	0	82220	0	0	0
Trojan	8	0	3	0	0	3
Torii	24	0	16291	0	0	0
Muhstik	36	147026	4020	0	0	0
Hide and Seek	112	571264	8	0	0	0
Kenjiro	24	0	0	20000	0	0
Gagfyt	24	0	0	20000	0	0
IRCbot	7	248880	0	0	0	0

Our anomaly detection model, which relies on machine learning algorithms, is positioned between the switch connecting the IoT devices and the router (gateway). It is designed to identify five distinct types of flows associated with malicious activities. These flows represent the communication patterns between various servers, Raspberry Pis, and the three IoT devices:

- C&C: This type indicates that the infected device is connected to a command and control server.
- DDoS: It signifies an ongoing distributed denial-of-service (DDoS) attack being executed by the infected device.
- Attack: This label indicates that the infected device is launching a specific type of attack towards another host, such as brute-forcing a Telnet connection or injecting commands into the header of a GET request.
- FileDownload: It suggests that a file is being downloaded onto the infected device. However, we did not utilize this type of traffic for classifier evaluation due to its limited occurrence.
- Scan: This flow indicates that connections are being used to conduct horizontal port scanning, gathering information for potential further attacks.

Table 3: The captured network traffic from the non-infected real IoT devices

Real IoTs	Capture Duration (Hrs)	Traffic (Flow Number)
Amazon Echo	5.4	7682
Philips Hue Bridge	24	78293
Somfy diirlock	1.4	4424

By analyzing the network traffic and classifying it based on these flow types, our model can effectively detect anomalous behavior and identify potential security threats in the IoT network.

Figure 4 illustrates the experimental environment topology used in our approach to simulate the detection of malicious botnet traffic from the IoT-23 dataset. The network configuration includes a botmaster machine and three servers. The botmaster machine serves as the controller, interacting with the command and control (C&C) server to monitor potential new victims and the botnet's current status through communication with the report server.

Additionally, the botmaster can initiate attacks on the target server by issuing commands via the C&C server. The loader server plays a role in connecting to the target device and instructing it to download and execute the appropriate binary version of the botnet upon receiving the infection command from the botmaster. To capture the data flow within the network, the sniffer machine is utilized. Our detection system analyzes this flow and generates alerts if any anomalies are detected, providing valuable information to the IoT user.

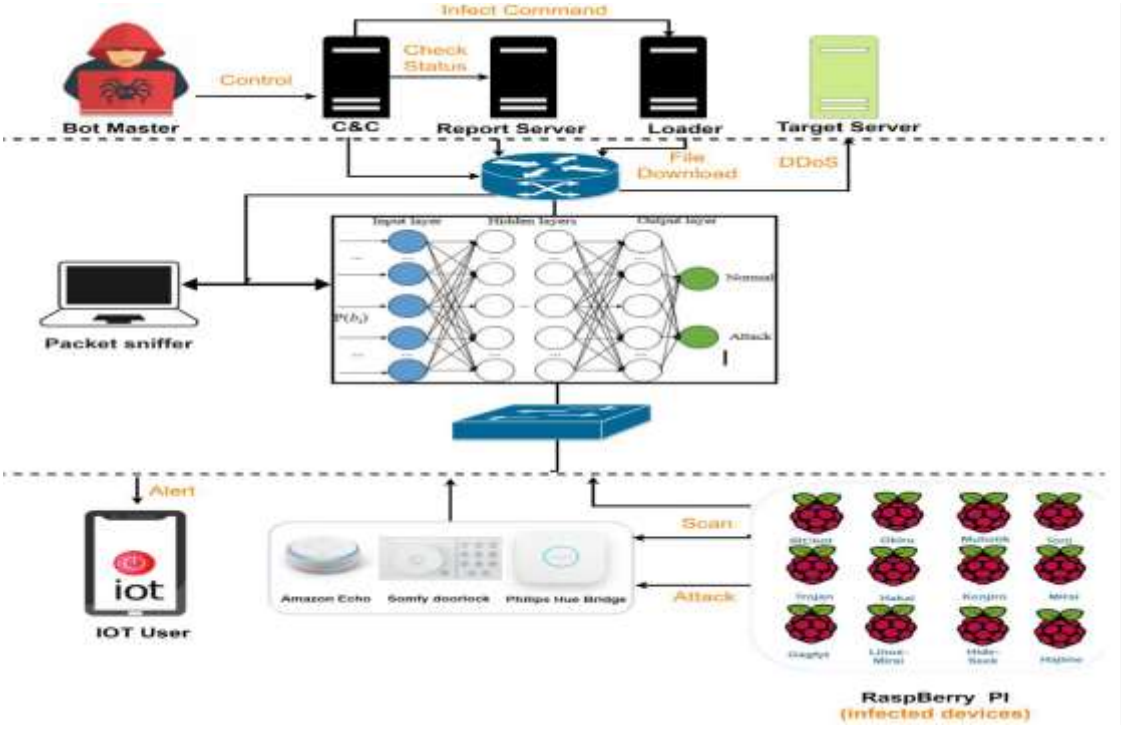


Figure 4. The network topology of the experimental environment

3.3 Data Preprocessing

Data preprocessing is a crucial technique employed to convert raw data into a clean and usable dataset. When data is collected from different sources, it is often in an unstructured format,

unsuitable for analysis purposes. Therefore, preprocessing steps are performed to transform the data into a refined and organized format, ensuring its suitability for subsequent machine learning tasks. In our approach, the data preprocessing phase comprises four key steps. Firstly, we extract relevant features using Tranalyzer. Secondly, we encode categorical data to enable proper representation and analysis. Next, standardization is applied to normalize the data and bring it into a consistent scale. Finally, attribute selection is performed using filters based on attribute importance measures. By executing these preprocessing steps, we ensure that the data is transformed into a clean and manageable dataset, facilitating accurate analysis and effective machine learning outcomes.

3.3.1 Feature Extraction

Feature extraction is a fundamental process in analyzing network traffic, and one common approach is to use flow extractors. In our approach, we employed the Tranalyzer tool for this purpose. Tranalyzer is a lightweight and efficient packet analyzer and flow generator designed for both practitioners and researchers. It extends the capabilities of Cisco NetFlow and assists analysts in handling large volumes of packets. By capturing live IP traces from Ethernet interfaces or pcap files, Tranalyzer enables the exploration of flows and individual packets of interest. It can quickly generate reduced pcap files for in-depth analysis using its text-based packet mode or by loading them into tools like tcpdump or Wireshark. Implemented in C and built on the libpcap library, Tranalyzer offers the flexibility to extract key parameters and statistics from captured IP traces.

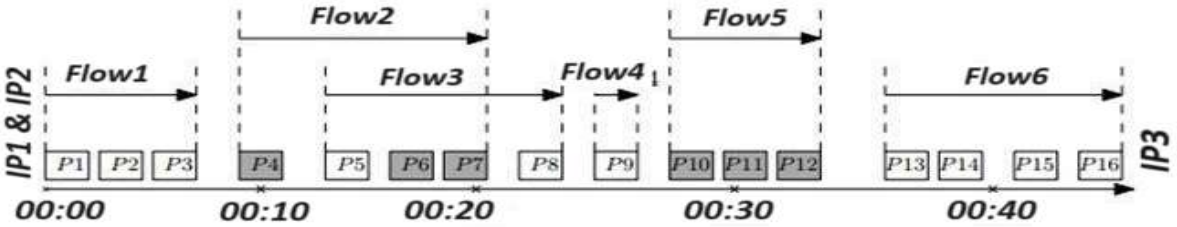


Figure 5. An example of flows exported by Tranalyzer [32].

3.3.2 Encoding Categorical Data

Tranalyzer generates a set of 62 attributes, comprising both numerical and categorical variables. Numerical attributes, such as flow duration, packet count, and Round Trip Time (RTT), have inherent quantitative values that can be directly utilized by machine learning algorithms without any preprocessing.

On the other hand, categorical attributes contain non-quantitative values known as categories. For example, the "tcpAnomaly" attribute denotes anomalies present in a packet and represents a categorical variable with multiple modalities like SYN-FIN flags, which should not coexist within the same

packet. Incorporating categorical variables into the learning process can introduce complexity since most machine learning algorithms operate on numerical input. Therefore, it becomes necessary to find a suitable method for encoding these categorical modalities into numerical representations.

To address this, we employed the "get_dummies" function from the Pandas library in Python for encoding our categorical data. This widely-used function simplifies the transformation of categorical variables into numerical ones. It operates by creating K indicator variables, where K represents the number of modalities. Each indicator variable is a binary vector with a value of 1 for the corresponding modality and 0 for others. By replacing the original categorical variable with these indicator variables, we enable machine learning algorithms to effectively process the data.

However, it's important to note that using the "get_dummies" approach may result in an increased number of variables, especially when dealing with a large number of modalities (e.g., more than 100). This can lead to a larger dataset, occupying more memory and potentially affecting the performance of machine learning algorithms during processing.

3.4 Standardization and Attribute Selection

Standardization is a common requirement for many machine learning estimators because they can perform poorly if individual features do not resemble approximately normally distributed data. In our approach, we utilized the Scikit-Learn library in Python to standardize the data, which involves removing the mean and scaling the variance to a unit value. For instance, many learning algorithms rely on the assumption that all features are centered around zero and have a similar variance. However, if a particular feature has a much larger variance compared to others, it can overshadow the objective function and hinder the estimator's ability to properly learn from other features as intended.

To address this issue, we apply standardization to ensure that features have comparable scales and are centered around zero. By subtracting the mean and dividing by the standard deviation, we transform the features to have a mean of zero and a variance of one. This process allows machine learning algorithms to effectively learn from all features without any single feature dominating the learning process. By utilizing the Scikit-Learn library for data standardization, we guarantee that our features are appropriately scaled, preventing any potential biases or dominance issues that may arise from features with varying scales or variances. In order to perform attribute selection on the dataset, we opted for a filter-based technique implemented in Matlab, as described in [33]. This approach utilizes a set of attribute importance measures that generate various rankings. These rankings are then combined to select the subset of attributes with the highest rankings. Furthermore,

the overall importance of an attribute is enhanced when the rankings produced by each metric are merged. It is important to note that attribute selection through filtering retains a subset of the original features, unlike feature extraction which creates new, potentially less interpretable features. The ranking list for each attribute is obtained by applying metrics specifically designed for single-class variable selection. These metrics assign a specific order to the attributes in the dataset, and subsequently, they are aggregated using a suitable technique.

The following importance measures, detailed in [33], were employed in this work:

1- Spectral Score (SPEC): This measure evaluates the coherence of features with the structure of a graph constructed based on the similarity matrix of labeled and unlabeled data. Features that align well with the graph's structure, indicating consistent and similar positive data, receive higher scores.

2- Information Score (IS): This metric assesses the relevance of each attribute by measuring the decrease in entropy when the attribute is removed from the dataset. If the removal leads to higher similarity among the remaining data, the attribute is considered important.

3- Pearson Correlation (PC): This measure quantifies the degree of association between each attribute and others. It checks for linear dependence among attributes and favors lower values, indicating weaker correlation, to retain uncorrelated attributes.

4- Intra-Class Distance (ICD): This metric quantifies the distance of each attribute's samples from the centroid of their respective class. Attributes that result in smaller distance reductions are considered better as they are closer to the data.

5- Interquartile Range (IQR): This measure takes into account the distribution of feature values across their interquartile range. Attributes with values that tend to be more concentrated are considered more representative of the dataset.

The implementation details for these attribute importance measures can be found in [34]. After obtaining the attribute ranking lists along with their corresponding aggregation values using Matlab, we sorted the attributes in ascending order based on their aggregation values. We selected the top 12 attributes that satisfied two conditions: reaching a minimum number of attributes and either maintaining or improving the performance of the classifier. We also accepted a slight degradation in performance, if necessary. The resulting 12 selected attributes, along with their respective Rank values, are listed below and visualized in Figure 6.

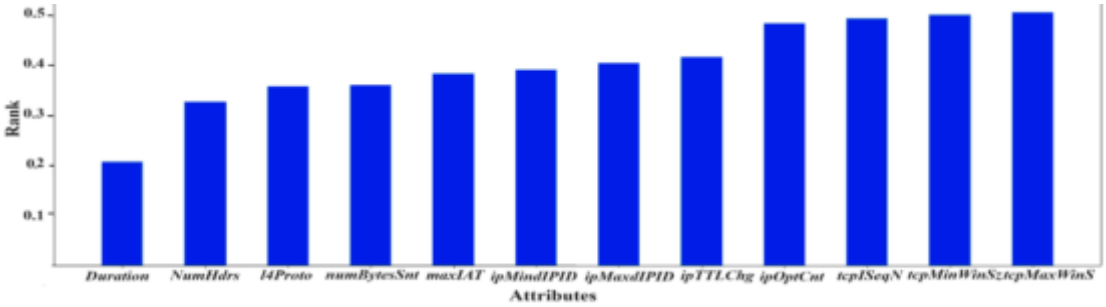


Fig. 6. The 12 attributes selected and sorted according to their Rank value

The correlation matrix, displayed in Figure 7, provides insights into the relationships between attributes. Attributes with high correlation exhibit stronger linear dependence, indicating that they have a similar impact on the dependent variable. However, the 12 selected attributes demonstrate a correlation of less than 50%, indicating a low level of dependence between them. This finding reinforces the reliability of our attribute selection process, as it ensures that the chosen attributes offer distinct and independent information for analysis.

Fig. 7. Correlation Matrix of the 12 Selected Attributes.

4. **Experimental Validation and Performance Analysis**

4.1 Performance Evaluation

In this section, we describe the implementation of our approach for testing and evaluate its performance. We utilized Python 3 for implementing the approach using the free cloud service Google Colab (Colaboratory), which is based on Jupyter Notebook and designed for machine learning research and training. Google Colab provides access to various libraries and services offered by Google. The scripts created in Colab are automatically saved on Google Drive, allowing for easy sharing

of the scripts.

To ensure data persistence and availability, we uploaded our database containing the required CSV files for training and prediction to Google Drive. This way, we could read the data from this location whenever needed, preventing data loss upon disconnection from the service. The single-class classifiers were implemented using the TensorFlow library in Python. For evaluating the performance of our proposed approach, we employed six performance metrics. These metrics are computed based on four measures: true positive (TP), true negative (TN), false positive (FP), and false negative (FN).

The performance metrics used in our evaluation are as follows:

- Accuracy: It measures the ratio of correctly classified abnormal and normal instances to the total number of instances. Accuracy provides an overall indication of how well the model performs, but it does not provide detailed insights into specific applications.

$$\boldsymbol{Accuracy} = \frac{(\boldsymbol{TP} + \boldsymbol{TN})}{(\boldsymbol{TP} + \boldsymbol{FP} + \boldsymbol{FN} + \boldsymbol{TN})}$$

- Precision: It represents the ratio of correctly classified abnormal instances to the total number of instances classified as abnormal. Precision indicates how often the model is correct when it predicts positive data. It should be noted that precision is different from accuracy.

$$\boldsymbol{Precision} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP} + \boldsymbol{FP})}$$

- Recall: Recall measures the ratio of correctly classified abnormal instances to the total number of actual abnormal instances. It signifies the model's ability to identify all relevant instances correctly.

$$\boldsymbol{Recall} = \frac{\boldsymbol{TP}}{(\boldsymbol{TP} + \boldsymbol{FN})}$$

- Specificity: Specificity is the ratio of true negatives to the total number of negatives in the dataset. It measures the model's ability to correctly identify true negatives.

$$\boldsymbol{Specifity} = \frac{\boldsymbol{TN}}{(\boldsymbol{TN} + \boldsymbol{FP})}$$

- F1Score: The F1 score is a measure of the test's precision, considering both precision and recall to calculate the score. It is the harmonic mean of precision and recall, with the best value being 1.

$$F1Score = \frac{2 * (Recall * \text{Precision})}{(Recall + FP\text{Precision})}$$

- Receiver Operating Characteristic (ROC): The ROC curve represents the trade-off between true positive rate and false positive rate. The area under the ROC curve (AUC) is a performance measure indicating the model's ability to distinguish between classes and avoid false classifications

References

By implementing our approach in this manner and evaluating its performance using these metrics, we were able to assess its effectiveness and gain insights into its classification and prediction capabilities.

4.2 Results and Discussion

This section presents the findings and analysis obtained from the implementation of the algorithm described in Chapter 3. The performance of the classification model was evaluated using various metrics such as Accuracy, Precision, Recall, F1 Score, Specificity, and average AUC. The dataset used for training consisted of 70% legitimate traffic, while 30% was used for testing, focusing on binary classification.

Table 4: Optimized Hyperparameters

Hyperparameters	Value
Activation function	(relu, relu, relu, softmax)
Hidden Layers	(Nb attribut, 8, 3, 8, Nb attribut)
Threshold	8
Loss	Mean_squared_error
Optimizer	Adam

Two experiments were conducted to assess the model's performance. In the first experiment, the model was trained using legitimate traffic from real IoT devices and tested with different types of malicious traffic generated by botnets running on Raspberry Pi devices (Scan, C&C, DDoS, Attack). The objective was to evaluate the model's ability to differentiate between legitimate and malicious traffic, as well as its effectiveness in detecting different types of attacks.

The second experiment aimed to evaluate the final model's capability in detecting zero-day attacks in IoT networks with minimal false positives. The normal class included all traffic from real IoT devices, while the abnormal class consisted of simulated malicious traffic from the IOT-23 dataset. The results are presented in tables 5, 6, and 7 and figures 8, 9, and 10. Various techniques were applied to improve the detection performance, including hyperparameter tuning for the AutoEncoder, feature selection, and optimization of runtime. The specific hyperparameters used are detailed in Table 4.

Table 5: Somfy IoT Lock

Malicious traffic type	Recall	Accuracy	F1score	Precision	ROC	Specificity
Scan	97,17%	99,04%	98,83%	97,17%	99,04%	99,80%
DDos	81,12%	86,11%	84,56%	81,12%	86,11%	96,91%
C&C	91,23%	95,11%	92,22%	91,23%	95,11%	94,45%
Attack	97,70%	99,22%	98,57%	97,70%	99,22%	99,92%

Table 5 displays the metrics obtained for the classifier trained on the Somfy dorlock traffic. The F1 Score, representing the balance between precision and recall, was used for result comparison. The classifier successfully distinguished between legitimate and malicious traffic, achieving an F1 Score above 85%.

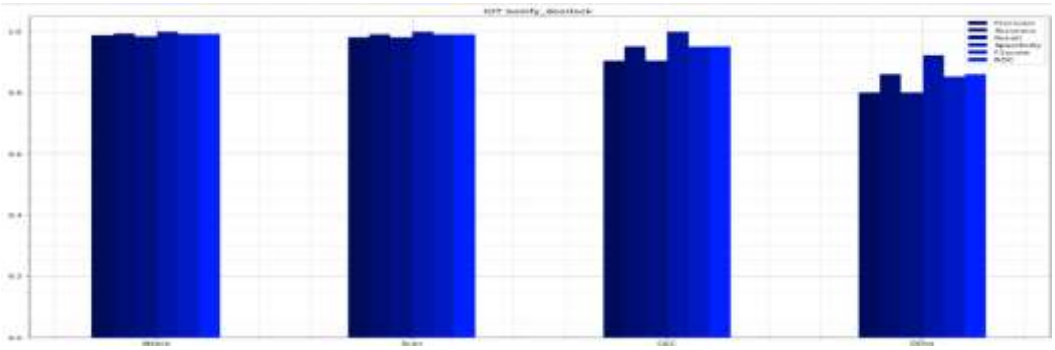


Figure 8. IoT Somfy dorlock traffic

Table 6: IoT Philips Hue Bridge

Malicious traffic type	Recall	Accuracy	F1score	Precision	ROC	Specificity
Scan	83,11%	98,97%	88,11%	83,11%	99,04%	98,83%
DDos	71,98%	74,03%	75,13%	71,98%	86,11%	74,11%
C&C	76,04%	80,95%	79,94%	76,04%	95,11%	81,13%
Attack	98,12%	96,03%	98,15%	98,12%	99,22%	96,17%

The results of the classifier based on the Philips Hue Bridge traffic are shown in Table 6. The performance was slightly lower compared to the first IoT device, indicating some dependency between legitimate and malicious traffic. The DDoS and C&C communications showed F1 scores of 74.48% and 80.16%, respectively.

Similar results were obtained for the classifier trained on Amazon Echo traffic, as presented in Table 7. There was a slight decline in detecting DDoS and C&C traffic, highlighting the challenge of differentiating these types of attacks from legitimate traffic. However, the classifier performed well in detecting Attack and Scan traffic, achieving F1 scores exceeding 97% and 87%, respectively.

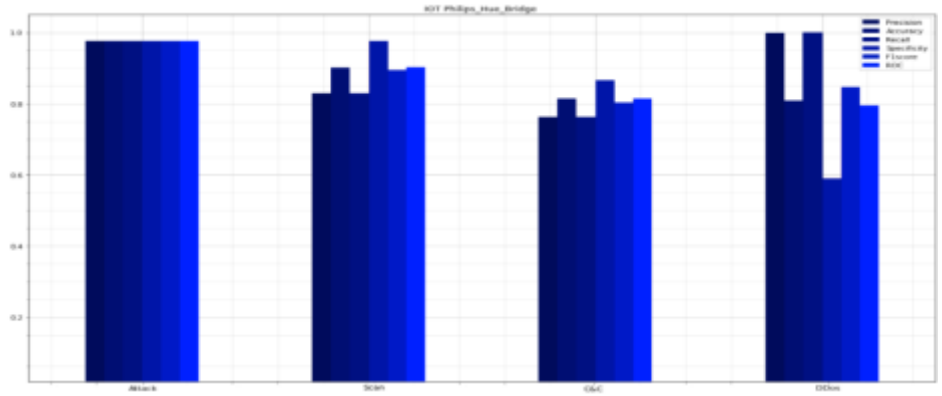


Fig. 9. IoT Philips Hue Bridge traffic

Table 7: Bridge IoT Amazon Echo

Malicious traffic type	Recall	Accuracy	F1score	Precision	ROC	Specificity
Scan	97,89%	86,03%	86,89%	97,89%	86,03%	72,95%
DDos	71,13%	66,25%	88,04%	71,13%	66,25%	72,95%
C&C	57,12%	78,11%	72,31%	57,12%	78,11%	72,95%
Attack	95,39%	98,13%	98,06%	95,39%	98,13%	72,95%

The second experiment's results, shown in Table 8, demonstrated the model's ability to establish a common profile among the legitimate traffic of the three IoT devices, leading to improved precision in detecting anomalies.
Additionally, a feature selection method was applied, resulting in a slight degradation in performance but with a significant improvement in runtime.

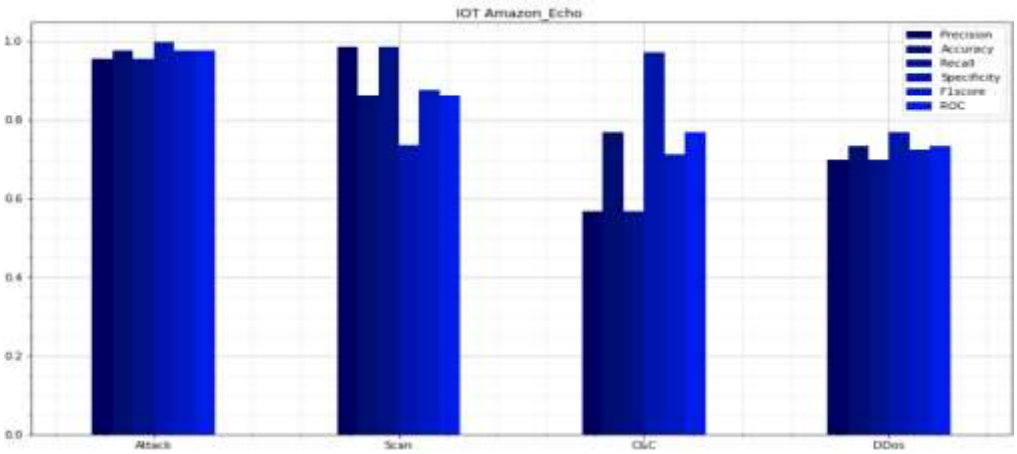
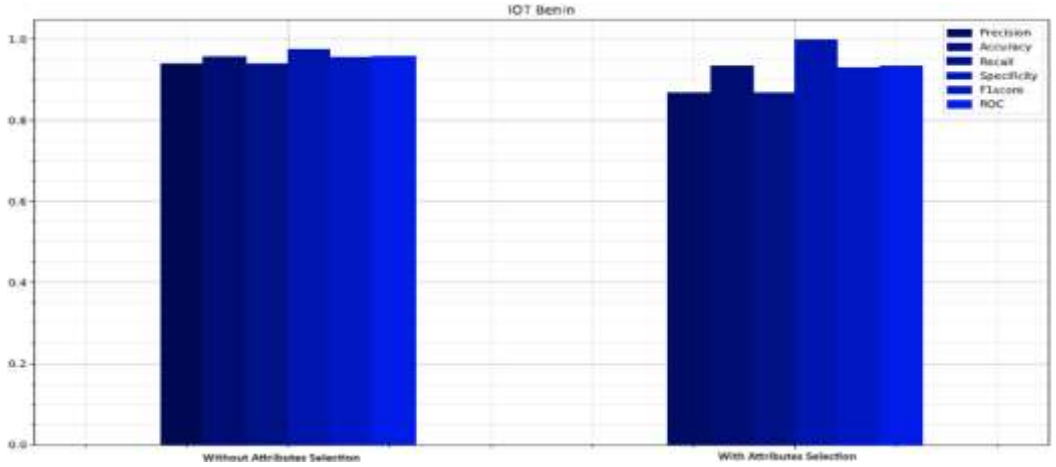


Fig. 10. IoT Amazon Echo traffic

Table 8: Learning IOT traffic with and without selection of attributes

Malicious traffic type	Recall	Accuracy	F1score	Precision	ROC	Specificity
Without Selection	97,44%	95.9%	96	96,89%	96,03 %	94,95%
With Selection	99.2%	92.4%	92.8%	98,83%	93,25%	87%



Comparing our AutoEncoder approach with supervised machine learning algorithms on the IOT-23 dataset, our deep learning-based one-class approach showed promising results. Supervised classifiers often struggle with detecting new attacks due to the need for labeled training data. In contrast, our AutoEncoder approach outperformed the linear SVM classifier used in previous work [35].

Table 9: Comparison table of AutoEncoder and linear SVM results

Classifier	Recall	Accuracy	F1score	Precision
Linear SVM	98%	92%	90%	84%
Our Approach	99%	94%	92%	98%

Table 9 provides a performance comparison between our AutoEncoder approach and linear SVM. In conclusion, this section presents a detailed analysis of the results obtained from our approach and identifies areas for further improvement. The comparison with other approaches demonstrates the effectiveness of our deep learning-based one-class approach in detecting anomalies in IoT networks.

5. Conclusion

This research aimed to develop a botnet detection approach for Internet of Things (IoT) networks without impacting their normal functioning. Botnets are collections of compromised IoT devices that are controlled remotely to launch cyberattacks. The researcher began by studying IoT network vulnerabilities and common attacks to understand the security challenges. Network traffic analysis was identified as a promising method since botnets can be identified by anomalies in device behavior and traffic patterns. Existing literature on botnet detection techniques for IoT was reviewed to inform their approach. Network traffic would be analyzed using Python tools due to their powerful data processing capabilities. An unsupervised deep learning technique called one-class classification was selected to model each device's regular traffic profile without labeled malware samples. It detects abnormal deviations from

normal behavior. An autoencoder-based one-class classifier was implemented and evaluated on the IOT-23 dataset. The experiments demonstrated it can accurately pinpoint malicious traffic with a 93% F1 score. Feature selection was also applied to reduce training/prediction time while maintaining performance.

In summary, the researcher developed and tested a botnet detection solution for IoT networks focused on network traffic analysis and one-class deep learning. It aimed to protect these networks from botnet infections without disrupting normal device operation.

This research successfully achieved its original objective of developing a method to detect botnets targeting IoT networks, thereby enhancing security for connected devices. However, further refining and expanding the work maintains potential for significant progress. One avenue is more rigorously validating the proposed solution across more diverse IoT environments and botnet variants. This could involve generating custom detection models leveraging different device types' legitimate traffic patterns to test generalizability. Evaluating performance when applied to new network conditions would provide invaluable insights. Additionally, the autoencoder performed strongly on scan and attack traffic, indicating promise for identifying compromised devices very early on. Detecting infections during initial targeting and infiltration offers opportunities to prevent many systems from becoming bots. Continued research focusing specifically on detection approaches suited for discriminating abnormal behaviors exhibited uniquely by IoT systems under initial botnet development could offer substantial value. Timely recognition of emerging threats aims to curb the scalability and impacts of these network-abusing programs. While the objectives were met, assessing cross-domain applicability and applying learnings to detecting botnets during their genesis represent especially worthwhile future work avenues. Ongoing experimentation and algorithm refinement applying these concepts maintains potential to significantly enhance capabilities to safeguard connected infrastructures. Overall, refining generalizability testing and expounding detection during early compromise stages represent pathways for impactful research expansions building on foundations established.

References

Luigi Atzori, Antonio Iera, Giacomo Morabito, The Internet of Things: A survey, Computer Networks, Volume 54, Issue 15, 2010, Pages 2787-2805.

Avani Sharma, Emmanuel S. Pilli, Arka P. Mazumdar, Poonam Gera, Towards trustworthy Internet of Things: A survey on Trust Management applications and schemes, Computer Communications, Volume 160, 2020, Pages 475-493.

M. Abu Rajab, J. Zarfoss, F. Monrose, and A. Terzis, “A multifaceted approach to understanding the botnet phenomenon,” in Proceedings of the 6th ACM SIGCOMM on Internet measurement - IMC ’06, (Rio de Janeriro, Brazil), p. 41, ACM Press, 2006.

Thakkar, A., Lohiya, R. A survey on intrusion detection system: feature selection, model, performance measures, application perspective, challenges, and future research directions. Artif Intell Rev 55, 453–563 (2022).

https://doi.org/10.1007/s10462-021-10037-9

Khanna, A., Kaur, S. Internet of Things (IoT), Applications and Challenges: A Comprehensive Review. Wireless Pers Commun 114, 1687–1762 (2020). https://doi.org/10.1007/s11277-020-07446-4

Krishna Kumar, Aman Kumar, Narendra Kumar, Mazin Abed Mohammed, Alaa S. Al-Waisy, Mustafa Musa Jaber, Rachna Shah, Mohammed Nasser Al-Andoli, "Dimensions of Internet of Things: Technological Taxonomy Architecture Applications and Open Challenges—A Systematic Review", Wireless Communications and Mobile Computing, vol. 2022, Article ID 9148373, 23 pages, 2022.

Geng Wu, S. Talwar, K. Johnsson, N. Himayat, and K. D. Johnson, “M2M: From mobile to embedded internet,” IEEE Communications Magazine, vol. 49, pp. 36–43, Apr. 2011.

Salih, Kazhan Othman Mohammed, Tarik A Rashid, Dalibor Radovanovic, and Nebojsa Bacanin. 2022. A comprehensive survey

Latif, Shahid, Maha Driss,Wadii Boulila, Sajjad Shaukat Jamal, Zeba Idrees, Jawad Ahmad, et al. 2021. Deep learning for the industrial internet of things (iiot): a comprehensive survey of techniques, implementation frameworks, potential applications, and future directions. Sensors 21 (22): 7518. on the internet of things with the industrial marketplace. Sensors 22 (3): 730.

Gubbi, J., Buyya, R., Marusic, S., & Palaniswami, M. (2013). Internet of Things (IoT): A vision, architectural elements, and future directions. Future generation computer systems, 29(7), 1645-1660.H. Moore, “Security Flaws in Universal Plug and Play Unplug. Don’t Play,” 2013.

Sadhu, Pintu Kumar, Venkata P Yanambaka, and Ahmed Abdelgawad. 2022. Internet of things: security and solutions survey. Sensors 22 (19): 7433.

Swessi, D., Idoudi, H. A Survey on Internet-of-Things Security: Threats and Emerging Countermeasures. Wireless Pers Commun 124, 1557–1592 (2022). https://doi.org/10.1007/s11277-021-09420-0

M. Vanhoef and F. Piessens, “Key Reinstallation Attacks: Forcing Nonce

Reuse in WPA2," in Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, (Dallas Texas USA), pp. 1313– 1328, ACM, Oct. 2017.

Ma Y, Wu Y, Yu D, Ding L, Chen Y. Vulnerability association evaluation of Internet of thing devices based on attack graph. International Journal of Distributed Sensor Networks. 2022;18(5). doi:10.1177/15501329221097817

Gu, Guofei & Perdisci, Roberto & Zhang, Junjie & Lee, Wenke. (2008). BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. {USENIX} Association, CCS'08. Pp. 139-154.

Patel, Chintan. "Secure Lightweight Authentication for Multi User IoT Environment." arXiv preprint arXiv:2207.10353 (2022).

Syed Muhammad Sajjad, Muhammad Rafiq Mufti, Muhammad Yousaf, Waqar Aslam, Reem Alshahrani, Nadhem Nemri, Humaira Afzal, Muhammad Asghar Khan, Chien-Ming Chen, "Detection and Blockchain-Based Collaborative Mitigation of Internet of Things Botnets", Wireless Communications and Mobile Computing, vol. 2022, Article ID 1194899, 26 pages, 2022. https://doi.org/10.1155/2022/1194899

Khraisat, A., Alazab, A. A critical review of intrusion detection systems in the internet of things: techniques, deployment strategy, validation strategy, attacks, public datasets and challenges. Cybersecur 4, 18 (2021). https://doi.org/10.1186/s42400-021-00077-7

Fouladi, B. and Ghanoun, S., 2013. Security evaluation of the Z-Wave wireless protocol. Black hat USA, 24, pp.1-2.

S. Pastrana, J. Rodriguez-Canseco, and A. Calleja, "ArduWorm: A Functional Malware Targeting Arduino Devices," p. 8.

M. Ryan, "Bluetooth: With Low Energy comes Low Security," 7th USENIX Workshop on Offensive Technologies (WOOT 13) , USENIX Association, p. 7.

E. Bertino and N. Islam, "Botnets and Internet of Things Security," Computer, vol. 50, pp. 76–79, Feb. 2017.

M. Feily, A. Shahrestani, and S. Ramadass, "A Survey of Botnet and Botnet Detection," in 2009 Third International Conference on Emerging Security In-formation, Systems and Technologies, (Athens/Glyfada, Greece), pp. 268–273, IEEE, 2009.

KOLIAS, Constantinos, KAMBOURAKIS, Georgios, STAVROU, Angelos, et al. DDoS in the IoT: Mirai and other botnets. Computer, 2017, vol. 50, no 7, p. 80-84.

Muhammad Hassan Nasir, Junaid Arshad, Muhammad Mubashir Khan, Collaborative device-level botnet detection for internet of things, Computers & Security, Volume 129, 2023.

A. Dunkels, B. Gronvall, and T. Voigt, "Contiki - a lightweight and flexible operating system for tiny networked sensors," in 29th Annual IEEE International Conference on Local Computer

Networks, (Tampa, FL, USA), pp. 455–462, IEEE (Comput. Soc.), 2004.

S. Nomm and H. Bahsi, "Unsupervised Anomaly Based Botnet Detection in IoT Networks," in 2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA), (Orlando, FL), pp. 1048–1053, IEEE, Dec. 2018.

Yuanyuan Zeng, Xin Hu, and K. G. Shin, "Detection of botnets using combined host and network level information," in 2010 IEEE/IFIP International Conference on Dependable Systems Networks (DSN), (Chicago, IL), pp. 291–300, IEEE, June 2010.

M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey," Digital Communications and Networks, vol. 4, pp. 161–175, Aug. 2018. arXiv: 1802.06305.

M. A. Manzoor and Y. Morgan, "Network Intrusion Detection System using Apache Storm," Advances in Science, Technology and Engineering Systems Journal, vol. 2, pp. 812–818, June 2017.

Sebastian Garcia, Agustin Parmisano, & Maria Jose Erquiaga. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic (Version 1.0.0) [Data set].

LuizHNLorena, LuizHNLorena/FilterFeatureOneClass," https://github.com / LuizHNLorena / FilterFeatureOneClass.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, and D. Cournapeau, "Scikit-learn: Machine Learning in Python," MA- CHINE LEARNING IN PYTHON, p.6.

"Tranalyzer documentation." https://tranalyzer.com/documentation.

A.A.; Abu Al-Haija, Q.; Tayeb, A.; Alqahtani, A. An Intrusion Detection and Classification System for IoT Traffic with Improved Data Engineering. Appl. Sci. 2022, 12, 12336.

https://doi.org/10.3390/app122312336