

Anomaly Detection in IoT Devices using LogBERT

Eniela Vela

A Thesis

in

The Department

of

Computer Science and Software Engineering

Presented in Partial Fulfillment of the Requirements

for the Degree of

Master of Computer Science (Computer Science) at

Concordia University

Montréal, Québec, Canada

November 2022

© Eniela Vela, 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: **Eniela Vela**

Entitled: **Anomaly Detection in IoT Devices using LogBERT**

and submitted in partial fulfillment of the requirements for the degree of

Master of Computer Science (Computer Science)

complies with the regulations of this University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

Dr. Yann-Gael Gueheneuc Chair

Dr. Yann-Gael Gueheneuc Examiner

Dr. Olga Ormandjieva Examiner

Dr. Brigitte Jaumard Supervisor

Dr. Fehmi Jaafar Co-supervisor

Approved by

Dr. Lata Narayanan, Chair
Department of Computer Science and Software Engineering

2022

Dr. Mourad Debbabi, Dean
Faculty of Engineering and Computer Science

Abstract

Anomaly Detection in IoT Devices using LogBERT

Eniela Vela

A rapid increase in the use of the Internet of Things (IoT) devices by corporates and consumers are changing the topology of the Internet. The design of IoT devices has often been centered on completing functional requirements (↔ observing, remote controlling), but has sporadically ignored security requirements.

Recently, due to the increase in cyber-attacks on IoT devices, researchers have focused on finding alternative machine-learning solutions. LogBERT is a new deep-learning approach based on the BERT algorithm, which has shown promising results in identifying anomalies in computer logs. LogBERT incorporates two self-learning tasks, Masked Log Key Prediction (MLKP) and Volume of Hypersphere Minimization (VHM). MLKP predicts random log keys and learns contextual information about log sequences while VHM maps the data in a hypersphere where the normal data are concentrated around the center and the abnormal data far from the center. After training LogBERT on normal data, the LogBERT algorithm can identify the abnormal data which deviate from the normal learned path.

In light of the positive results shown in computer logs, we propose to extend the usage of the LogBERT algorithm in IoT data. Our experiment is based on using real-life data which we generated by exploiting the six most common IoT devices; an indoor camera, outdoor camera, DVR, two different home routers, and a smart light bulb, using three different malware that has different architectures but similar exploitation techniques; Mirai botnet, RouterSploit, and UFONet. Our experiment showed that LogBERT can be used for anomaly detection in IoT devices and that it achieves better results than some existing machine or deep learning approaches.

Acknowledgments

After two years of researching for my graduate program at Concordia University, I have to say, "Life can teach you more than you can imagine. I still can remember my 20 years old ambitious self when I was first introduced to the CyberSecurity class at the University of Maine during my exchange semester and my confused self right before my bachelor thesis defence at the American University of Bulgaria. I was torn between two choices: iOS development or CyberSecurity. Fast-forward 5 years later, I am happy to have made my 20 years old self proud by pursuing a Research-Based Master's of Computer Science with a focus on CyberSecurity. However, I would not have been able to achieve this dream without the amazing support of a few people who never gave up on me.

I am lucky enough to have found a soulmate like Arnaud Y. Th. Gerard has been with me through every tear, obstacle, and little victory of this thesis. I would like to officially thank him because he has indirectly been part of this thesis and has brought me back on my feet even when I lost myself.

I want to thank my parents, Pellumb and Teuta Vela, for all of their sacrifices so I could get a chance to study abroad. Without their help, I would not be able to conquer all the problems along the way.

I want to thank my sister, Telma Vela, who brightened my day throughout my research and took off some of my panics when the research was not going how I expected.

I want to express my deep gratitude to my research supervisor, Dr. Brigitte Jaumard, for her constant support and invaluable guidance throughout this research. She has taught me how to clarify my ideas and prepared me for high academic writing. Besides the academic advice, she

has taught me some most valuable life lessons I will carry with me for the rest of my journey.

I am incredibly grateful I had the opportunity to have Dr. Leila Kosseim, the Graduate Director of the Computer Science department at Concordia University, as part of the second half of this research journey.

Lastly, I thank myself for not breaking down in the most difficult times and keeping strong despite losing faith. I owe this to my younger self, and I am glad I passed through this path. This thesis is a reminder that I am stronger than I think; all context considered.

∇VIIΔVIII

Contents

Contents	vi
List of Figures	ix
List of Figures	ix
List of Tables	xi
List of Tables	xi
1 Introduction	1
1.1 Motivation	1
1.2 Background	2
1.3 Research Questions	3
1.4 Challenges	5
1.5 Contributions	5
1.6 Outline of the Thesis	7
2 Literature Review	8
2.1 Traditional Approaches	8
2.2 Machine Learning Approaches	11
2.2.1 Supervised Machine Learning-Based Approaches	11
2.2.2 Unsupervised Machine Learning-Based Approaches	14
2.3 Blockchain Based Approaches	18

2.4	Concluding Remarks	18
3	Research Methodology	20
3.1	IoT Device Selection	20
3.2	Data Generation	21
3.2.1	Normal Data Generation	23
3.2.2	Abnormal Data Generation	24
3.3	Analysis of LogBERT	30
3.3.1	LogParser	30
3.3.2	BERT	31
3.3.3	LogBERT	34
3.4	Energy Consumption	37
4	Experimental Results	39
4.1	Data Generation Results	40
4.2	LogBERT Results in IoT Data	43
4.2.1	Data Analysis	43
4.2.2	Data Preprocessing	44
4.2.3	Data Process	45
4.2.4	LogBERT	50
4.2.5	Comparison of LogBERT with PCA, SVM, IsolationForest and LogClustering	56
4.2.6	Comparison of LogBERT Performance on Computer Logs and IoT Packets	57
4.3	Energy Consumption Preliminary Work Results	59
5	Conclusion and Future Work	61
5.1	Conclusion	61
5.2	Limitation	63
5.3	Future Work	63
	Appendix A	65
	Appendix B	66

List of Figures

Figure 3.1	IoT Architecture Environment	23
Figure 3.2	Mirai Botnet Architecture	25
Figure 3.3	RouterSploit Cred Model Options for DVR Device [36]	27
Figure 3.4	P2P Botnet Architecture of UFONet [36]	29
Figure 3.5	Structure of Parse Tree in Drain (depth = 3) [55]	32
Figure 3.6	BERT's Transformer Encoder [29]	33
Figure 3.7	Next Sentence Prediction Chart [29]	34
Figure 3.8	Overview of the LogBERT Architecture [47]	35
Figure 4.1	Normal Behaviour Data Wyze Cam WYZEC device (Indoor Smart Home Cam- era)	44
Figure 4.2	Google IoT Log Data taken from Google Cloud https://cloud.google.com/community/tutorials/cloud-iot-rtdp	44
Figure 4.3	Normal Behaviour Data Wyze Cam WYZEC device	45
Figure 4.4	Wyze Cam WYZEC Device Packet Block	45
Figure 4.5	The Logic and the Connection between the Data Preprocessing and Data Process	46
Figure 4.6	Event Template File Screenshot Taken from DVR Camera Data	48
Figure 4.7	Vocabulary Generated for DVR Camera Data	48
Figure 4.8	Structured Data File Screenshot Taken from DVR Camera Data	49
Figure 4.9	Event Sequence File Screenshot Taken From DVR Camera Data	49
Figure 4.10	DVR minibatch train validation loss graph. (Normal 50:50 Abnormal)	52

Figure 4.11 DVR Confusion Matrix.	52
Figure 4.12 DVR Minibatch Train Validation Loss Graph. (Normal 70:30 Abnormal) . . .	53
Figure 4.13 DVR Confusion Matrix	53
Figure 4.14 DVR Minibatch Train Validation Loss Graph.(Normal 90:10 Abnormal) . . .	54
Figure 4.15 DVR Confusion Matrix	55
Figure 4.16 LogBERT Results on Computer Logs Presented in "LogBERT: Log Anomaly Detection via BERT" [47]	58
Figure 4.17 ANNKE DVR 8CH Energy Consumption Graph	60
Figure B.1 Wyze Cam WYZEC Energy Consumption Graphs	66
Figure B.2 HOSAFE Outdoor Wifi Camera Energy Consumption Graphs	67
Figure B.3 TP-Link ACS1750 Energy Consumption Graphs	68
Figure B.4 D-Link AC1200 Energy Consumption Graphs	69
Figure B.5 Phillips Hue Smart Hub Energy Consumption Graphs	70

List of Tables

Table 3.1	IoT Device Information	21
Table 3.2	RouterSploit Models	26
Table 3.3	IoT Devices Internet Protocol	26
Table 3.4	Terminology Used Across Energy Consumption Sections	37
Table 3.5	IoT Devices Voltage Information	38
Table 4.1	Network Traffic and Electric Current Dataset Features	40
Table 4.2	Number of Data Packets Generated in Network Traffic Legend : First Set Normal Data - ND-1 , Second Set Normal Data - ND-2 , Mirai Data - MD , RouterSploit Data - RD , UFONet Data- UD ,	41
Table 4.3	F1 Score with the First Set of Normal Data Legend : M - First Set of Normal Data + Mirai Data , R - First Set of Normal Data + RouterSploit Data, U - First Set of Normal Data + UFONet Data, M-AVG - Macro Average of F1 score,	51
Table 4.4	Number of Abnormal Mini Batch Size Legend : Second set of Normal Data - ND-2 , 10% Mirai Data - mini-MD , 10% RouterSploit Data - mini-RD , 10% UFONet Data- mini-UD	55
Table 4.5	F1 Score with Second Set of Normal Data Legend : M - Second Set of Normal Data + Mirai data , R - Second Set of Normal Data + RouterSploit data, U - Second Set of Normal Data + UFONet,	56
Table 4.6	Indoor and Outdoor F1, Recall and Precision	57
Table 4.7	Home Router (1) and (2) Smart Camera F1, Recall and Precision	57
Table 4.8	Smart Hub and DVR Camera F1, Recall and Precision	57

Chapter 1

Introduction

1.1 Motivation

The term Internet of Things, or IoT for short, was introduced by Kevin Ashton in 1999 [13]. It refers to all intelligent devices which are instantly connected to the Internet, communicating, gathering, and sharing information between each other and other information systems. In recent years, IoT has been gaining attention on both individual and professional levels. At an individual level, IoT enhances the quality of living in the form of smart personal assistants, smart homes, and electronic health gadgets. At a professional level, IoT finds use in transportation, health care, and remote quality control. IoT usage is no longer limited to the information technology field. Its usage has spread into areas such as science, business, and medicine [101]. The expansion of IoT beyond the IT field has increased IoT device production to accommodate the growing demand. According to Statista [8], the current number of IoT devices all over the world stands at ten billion, and this number is projected to double within the next decade.

As IoT devices have gained popularity, the threat of cyberattacks has also increased. According to Gartner, [110], 20% of the organizations which have implemented IoT devices have reported an IoT-based cyber attack. In addition to the general security challenges faced by the Internet, cellular networks, and Wireless Sensor Networks (WSNs), the Internet of Things also poses many security challenges, including privacy issues, authentication issues, data management issues, and storage of information [52]. What in the first place can be seen as a minor

functional disruption of an IoT device can potentially lead to the abruptness of the other components within the network, differently known as Distributed Denial of Service attack. One of the main problems with the DDoS attack is that it generates invalid requests and occupies a large number of network resources. Therefore, real customers cannot use the service. Considering the severity of the attack and the poor security of some of the IT devices, further research is required in this area.

Several studies have addressed security issues in IoT devices. Various approaches have been suggested for detecting different types of attacks in an IoT device, ranging from simple firewalls to anti-viruses using advanced machine learning techniques. On the one hand, we have IoT devices that are increasing in number along with cyberattacks, and on the other hand, we have powerful tools like Machine Learning and Deep Learning that have been facilitating impressive results in the early detection of these IoT attacks. Needless to say, the implementation of such powerful ML and DL algorithms requires a large data collection to have high-accuracy results and to take advantage of such solutions. The increase of cyberattacks in IoT devices and the implementation of machine learning techniques to identify such attacks, motivated this thesis.

1.2 Background

The term cyberattack is not a new concept in the technology world. A cyber attack involves the use of a variety of techniques and tools to harm a system or disrupt normal operations [53]. The motive of the attacker who launches the attack ranges from revenge to personal satisfaction. There are various attacks, but the most common cyber attacks are: Physical attacks, Denial-of-service (DoS)/ Distributed-Denial-of-service (DDoS), Access attacks, Attacks on privacy, and Reconnaissance attacks [2]. The most popular attack in recent years has been Denial-of-service (DoS)/ Distributed-Denial-of-service (DDoS) with 9.7 million DDoS attacks launched in 2021 [111]. The first documented DDoS attack occurred on 22 July 1999 during the Internet of Computers era, when the Internet was intended for computer usage only [51]. Ever since then, DoS and DDoS attacks have increased proportionally with the number of devices connected to the Internet over the years.

As computers increased the layers of protection, cyber attackers turned their attention toward IoT devices which appear to be more vulnerable. There is a huge gap between the complexity and advanced nature of cyberattacks and the simple operating systems of IoT devices. For instance, CCTV cameras, which were once used to prevent crime, are today the most vulnerable IoT devices whose authentication factor is the easiest one to guess [92]. Although, CCTV is not the only IoT device which are facing major security issues. In addition, some IoT devices cannot update their software to fix critical bugs [89]. As a result, the security risk is not only mitigated for that specific IoT device but also for other devices connected to the same network. The attacker can scale the attack by passing through the most vulnerable device, leading to a higher-dimension attack.

The increasing number of attacks on IoT devices calls for researchers to focus on the detection of attacks on those devices. While preventing all attacks will not be an absolute solution, minimizing risks will be a major step forward in this area. To do that, we need to examine how the network traffic data change under attack. There are times when threats and attacks become a chain reaction, where a simple brute force algorithm [85] which is based on try/error logic, can lead to much more damage, including data leaks and physical threats. Rather than improving a method, try/error logic involves trying every possible solution.

1.3 Research Questions

Recent years have shown that the value of real-life data has increased so much that it has surpassed the value of oil [17]. In such a case, getting a large amount of real-life data of IoT device attacks can be challenging and not easily accessible. To avert this problem, some researchers, in the absence of real data from IoT attacks, use synthetic data generation. As the data is synthetically generated, the solutions proposed by those researchers do not necessarily solve real-life problems. The lack of real-life data leads us to ask **RQ 1**: how can we generate real-life IoT datasets to get as close as possible to real-life solutions?

To perform our study and to address the first research questions, we needed several datasets of normal and abnormal IoT network traffic (See Section 3.2). We generated real-life data by

attacking six IoT devices with three different malware. By generating real-life data, we are contributing to the research community, and we are basing this thesis on real-life IoT attack events. This data is used later on in LogBERT algorithm [47] in Chapter 3.

Researchers have developed and improved machine learning algorithms that mimic our brain's neuron processing and how human brains process different languages throughout the years. Bidirectional Encoder Representations from Transformers (BERT) [29] is an open-source Natural Language Processing (NLP) algorithm developed by researchers at Google AI Language. Since it is open-source, BERT inspired many other algorithms such as Google's TransformerXL [25], RoBERTa [80], LogBERT [47] etc. In this thesis we will investigate whether it is possible to use LogBERT for anomaly detection in IoT devices using real-life data.

In this thesis, we picked the LogBERT algorithm as it is a BERT-based algorithm that has presented promising results in computer log messages [47]. To the best of our knowledge, there has not been any research testing the performance of LogBERT in IoT network traffic data packets. This leads us to the second and third research question. **RQ 2:** Considering that LogBERT is initially used in computer log data, can LogBERT [47] extend its usage in IoT network traffic data? **RQ 3:** What is the performance of LogBERT algorithm with real life data in IoT devices in comparison to other machine learning algorithms?

In Section 3.3, we describe how we plug the real-life data that we generated in the LogBERT algorithm to detect the IoT attacks based on data packet exchanges in network traffic data. We explain the similarities between the network traffic data and a log. We determine what balance of the dataset leads to better performance. We compare the performance of the LogBERT in IoT device network traffic data packets with the results of the computer log messages presented in the LogBERT paper [47]. Lastly, we compare the performance of the LogBERT on IoT datasets with some of the machine learning approaches mentioned in Chapter 2. In Chapter 4 we present that the LogBERT algorithm can have promising results in real log IoT data.

Finally, we investigate with **RQ 4:** whether energy measures can give an indication of anomaly detection? We present some preliminary work on energy consumption based on the data we collected in Chapter 4. We investigate the change in energy consumption between normal and abnormal states.

1.4 Challenges

We faced a few challenges while going through the three parts of this research; **1)** the real-life data generation, **2)** the data analysis using the LogBERT algorithm, and **3)** data interpretation based on the energy consumption of the IoT devices.

Challenges while generating the real-life data

- It is difficult to find malicious code that can successfully run on IoT devices. There is usually a significant amount of logic missing from the code, or the target is not part of the IoT spectrum.
- Some attacks camouflage the data packet's transportation, which makes it hard to distinguish them from the normal data packet exchange.

Challenges while analyzing the data using LogBERT algorithm

- IoT network traffic data are unstructured alphanumeric data [119], which makes it difficult for machine learning algorithms to interpret. Various IoT devices use different formats and syntax for exchanging messages, making it difficult to adopt a single technique to parse network traffic data and create structured data.
- IoT network traffic data has a high number of anomalies when under attack, which makes the dataset unbalanced, so it is difficult for the algorithm to predict normal behaviour patterns. As a result, training a binary classifier is sub-optimal for distinguishing network traffic data anomalies from network traffic data normalizes [19].

Challenges while interpreting the energy consumption data of the IoT devices

- IoT devices have low energy consumption, which cannot be detected through commercial energy monitors.

We explain how we overcame these challenges in Chapter 3.

1.5 Contributions

Our most significant contributions are listed below:

- We collect the energy consumption and network traffic data in the abnormal state under Mirai malware, RouterSploit malware, and UFONet disruptive cryptography toolkit for six different IoT devices.
- We present and analyze results found by the usage of LogBERT on IoT devices. We determine the best balance of the dataset that leads to better performance. The balanced dataset is composed of the same number of input samples for each output class (or target class). We compare the performance of LogBERT on the computer logs presented in the LogBERT paper [47] and on the IoT network traffic packets, to investigate if the LogBERT algorithm is applicable also in IoT devices. Lastly, we compare the performance of the LogBERT algorithm with different algorithms; PCA, LogCluster, One SVM, and Isolation Forest. Overall, LogBERT performs better than the other algorithms.
- We present the results of our data interpretation on energy consumption in the normal and abnormal states for the six IoT devices. IoT devices registered an increase in energy consumption when under attack.

The data generation for both energy consumption and network traffic under Mirai attacks was done by another student, Khaireddine Mejri, under the supervision of Dr. Fehmi Jaafar in his thesis on "Identification of Infected IoT Devices Using Network Traffic and Power Consumption Analyses" [84]. I took care of the implementation of the RouterSploit and UFONet attacks for the IoT devices to generate real-life data under attack, collected the network traffic and energy consumption data generated during the attacks, investigated LogBERT for IoT network traffic data, and did preliminary work on data interpretation for energy consumption of the IoT devices in normal and abnormal behaviour. Due to the COVID-19 restrictions, the lab in CRIM (Computer Research Institute of Montreal) was closed. As a result, the IoT devices were connected through a secured network with the help of Louis-David Perron. I performed the attacks using a VPN and virtual box for security reasons.

1.6 Outline of the Thesis

The thesis is organized as follows. In Chapter 2, we discuss various related work that focuses on the security of IoT devices and different solutions proposed for securing IoT devices.

In Chapter 3, we explain the procedure of generating real-life data in normal and abnormal states. The normal state is considered the moment that the IoT devices are plugged in and connected to the Internet and the abnormal state is when the IoT devices are exploited with Mirai Botnet, RouterSploit, and UFONet. We describe the LogBERT algorithm and explain the relation between LogParser and BERT algorithms with the LogBERT one.

In Chapter 4, we present the results of our experiments and explain how we overcame the challenges mentioned in Section 1.4. From our findings, the LogBERT algorithm can be used in IoT network traffic data packets and has a better performance than the other algorithms; Principal Component Analysis (PCA), Isolation Forest, One Class Support Vector Machine (OCSVM), and LogClustering. We finish our thesis with concluding remarks and future work in Chapter 5.

Chapter 2

Literature Review

Numerous studies have focused on security issues in IoT devices, proposing different approaches to tackle those problems. Note that there are dedicated studies on security issues, although it is related to anomaly detection. However, not all anomaly detection techniques can be reused for building secure software. We will not thoroughly review anomaly detections as it is a whole field in itself, even if related. We have focused on the highly cited publications in cybersecurity tools in IoT over the years. In the other survey, Noor *et al.* [50] give an overview of the IoT security research trends between 2016 and 2019. They present Software-Defined Networking technology as the newest trend of 2019 in IoT security. However, as the prevalence and sophistication of the IoT have increased, researchers have shifted their focus to more advanced methods like machine learning, and deep learning [5].

This Chapter is organized around three sections: the first reviews traditional approaches to IoT attack detection, the second section discusses machine learning approaches to IoT attack detection, and the third section presents a last set of approaches, i.e., blockchain techniques. In the last section, we draw some conclusions.

2.1 Traditional Approaches

In this section, we consider traditional approaches and each of the proposed solutions for IoT attack detection that does not implement machine learning (ML) techniques. Over the

years, many tools that do not implement ML techniques have been proposed. The key ones are Software-Defined Networking (SDN) approaches [102], trust frameworks [86], and IoT-secured middleware [43]. Many additional tools have been proposed, such as Cyber-Physical System (CPS) [121], [20], cross-layer IoT architecture [122], compressed internet protocol (IP) security for IoT [96], and Intrusion Detection Systems (IDS) in IoT [87],[18]. Unable to pick each solution proposed over the years due to the volume of the paper produced, we are going to select the key traditional approaches among the most cited ones.

The first reference refers to Software-Defined Networking (SDN). SDN uses software-based controllers or application programming interfaces (APIs) to communicate with underlying hardware infrastructures and direct traffic on a network. Shirali-Shahreza *et al.* [102] propose a Software-Defined Networking (SDN) based firewall to detect and block suspicious port scans in IoT devices. The proposed firewall runs as a cloud-based service which eliminates the need for a dedicated firewall in the client network. According to the authors, the SDN-based firewall achieved a 99% accuracy in identifying suspicious logs from the history logs. Although, the two limitations the paper suggest are; 1) the log history was synthetically generated using NetFlow-like logs, and 2) the detection of attack logs from the history logs interferes with timing as detection is not done in real-time but after the logs are saved. Other references on SDN solutions in IoT can be found in [63], [69].

The second reference refers to trust frameworks that achieve a secure management system overall for IoT devices. Ning *et al.* [86] proposed an IoT system architecture that offers a broad array of solutions in system security, network security, and application security. The authors call the proposed system architecture Unit and Ubiquitous IoT (U2IoT), where a unit represents a single IoT application, and a ubiquitous stands for the local, national, and industrial IoT. The proposed U2IoT has three layers; the perception layer, which senses physical objects and converts them into cyber entities; the network layer, which includes all the communication channels such as interfaces, routers, and gateways; and the application layer, which connects unit IoTs in a geographical region. According to the authors, U2IoT enhances cybersecurity requirements with respect to the Confidentiality, Integrity, and Availability (CIA) triad, authority, nonrepudiation, and privacy preservation. The proposed architecture does not take into consideration specific

trust issues such as trust relationship and decision (TRD) which assists IoT devices in making a wise decision to communicate and collaborate with each other [120]. Other frameworks, such as one for preventing sleep deprivation attacks, were described in [66], and another for securing the initialization of IoT was proposed by Pecorella *et al.* [93].

The last key reference is the middleware. Middleware has been defined as computer software that has an intermediary function between the various applications of a computer and its operating system [49]. Middleware has also found its usage in the mitigation or prevention of various attacks in IoT devices [109]. FIWARE [43] is a middleware designed for IoT devices to be the basis of a Future Internet. FIWARE is among the few systems that claim to use PBD as a design basis [41]. The proposed middleware is based on plugins known as Generic Enablers (GE). As part of the security model, GEs such as Identity Management (IDM), the Authorization Policy Decision Point (PDP) GE, and the Policy Enforcement Point (PEP) Proxy, are implemented. FIWARE utilizes OAuth2 and XACML is the standard authentication method. FIWARE also supports cross-system interoperability for exchanging identities. As a security design, FIWARE fits modern authentication and authorization models. FIWARE is equipped with a gateway model to support IoT devices. The connection between IoT devices and the gateway, which is part of IoT Edge, is done using IoT-specific protocols. The FIWARE protocol is used to communicate with an IoT Backend that allows Device Management and Discovery capabilities. Other middlewares for IoT security have been proposed in [11], [35] and [24].

So far, the traditional approaches to IoT attack detection, such as SDN, trust framework, and IoT-secured middleware, have been used to protect IoT devices from cyber attacks. However, with the evolution of IoT devices and the advancement of IoT attacks, such technologies [102], [86], [16], [43] will not be able to detect advanced attacks in real-life as they are built based on predefined static rules, and conditions [60]. Additionally, the papers on the proposed technologies mentioned above were published more than five years ago. Hence, the relevance of such solutions could have been understandable at that time, but they do not resonate with today's security problems of IoT devices. As IoT attacks have become more sophisticated, the need for more advanced technology implementations has increased. In recent years, the usage of machine learning and deep learning models has grown. Nevertheless, the implementation of such

technologies, ML and DL, requires numerous data to train and test the models.

2.2 Machine Learning Approaches

In recent years, applications using machine learning and deep learning technologies have successfully detected IoT attacks. We refer the reader to [6] where several classical machine learning methods, such as decision trees, support vector machine, and linear regression, and a few advanced deep learning methods for handling big data were studied in the context of IoT. Therein, we can see that most machine learnings have been used in one way or the other for detecting IoT attacks. We, therefore, present in the sequel the main classical supervised and unsupervised machine learning models and a few advanced deep learning models and some of their applications for IoT security, and limit ourselves to those published in the last five years.

2.2.1 Supervised Machine Learning-Based Approaches

The goal of supervised learning is to model the distribution of classes based on the characteristics of predictors. The first step in supervised machine learning is to deal with the dataset. In order to perform well in training the model, a machine learner or researcher should select specific features in a dataset. Ultimately, in most cases, this step is followed by data preprocessing [71], where the dataset is cleaned from noises and missing feature values. Once the dataset is clean and ready to be used, the next step is to pick the correct algorithm for the dataset. Over the years, researchers have proposed several supervised machine-learning algorithms to detect IoT attacks. Below, we select the most recent studies on classical algorithms applied to IoT attacks, such as decision trees [106], k-nearest neighbour [91], and support vector machines [48], [62].

The decision tree (DT) method classifies samples by sorting them according to their features. The nodes (vertices) in such a tree represent features, and the edges (branches) identify the values that each vertex can represent in a sample. Starting at the origin vertex, each sample is classified according to its feature values. An origin vertex is a feature that optimally divides the training samples [71]. The information gain index [95] and Gini index [32] are used to determine the best feature to use when splitting training samples. In their paper, Taghavinejad *et al.*

[106] use the Classification And Regression Trees (CART) method, which is a DT-based method. Since CART is a DT-based algorithm, it can handle both classification and regression tasks. The experiment was done using the NSL-KDD dataset [107] which includes seven different types of attacks: brute-force, denial of service (DoS), and distributed denial of service (DDoS), web attacks, botnet, heartbleed, infiltration of the network from inside. The authors conclude that CART achieves an accuracy of 83.14% compared to DT 80.9%.

The support vector machine (SVM) is used for classification by creating a splitting hyperplane between the data attributes, such that the distance between the hyperplane and each class's most adjacent points will be maximized [113]. SVMs have been widely used in various security applications, including intrusion detection systems [58], [100]. One study [48] developed an Android malware detection system for securing IoT systems and applied a linear SVM to their system. Researchers compared the detection performance of SVM with that of other ML algorithms, such as NB and DT. According to the comparison results, SVM outperformed the other ML algorithms with an f-score of 95.4% and a precision of 95.7%. These results confirmed the robust application of SVM for IoT malware detection. However, the results proposed in this study [48] are a bit too promising; consider the fact that in another study [47] authors use One-Class SVM (OCSVM) in a Thunderbird dataset [88] and the SVM based algorithm performed the worse with only 25.48% f-score. Yet the performance of the algorithms relies heavily on the features and the size of the dataset.

In another research paper, Ioannou and Vassiliou [62] used a Radial Basis Function (RBF) with a c-optimizer in a Support Vector Machine (c-SVM) to detect a large-scale attack by analyzing the network behaviour. RBF was used as a better classification linear function compared to polynomial and sigmoid kernel. The c-optimizer was used to increase the correct classification of data in the training phase. C-SVM was trained and tested on live network data of IoT systems collected via a Resource Monitoring Tool (RMT). The proposed model had 100% accuracy with Blackhole and Sinkhole attacks and 81% on different network typologies. However, more research must investigate the performance of SVMs for different attacks and environments.

K-nearest neighbour (KNN) is another classic supervised machine learning technique based on the Euclidean distance [104]. Unlike DT and SVM, KNN is a nonparametric method. The KNN

classifies unknown samples by majority vote among its nearest neighbours. KNN has wide usage in network anomaly detection [74], [90]. However, its usage is also found in IoT device attack detections. A study [91] proposed a KNN-based model to detect User to Root (U2R) and Remote Local (R2L) attacks in IoT devices. The authors implemented two layers of feature reduction to enhance efficiency, followed by a two-tier classification algorithm that used both NB and KNN classifiers. The dataset used for this experiment was the NSL-KDD [107]. The proposed model reduced the dimensionality of the features and showed good results in detecting both attacks. This model detection rate was 84.82%, to be compared with DT, which achieved 81.05%.

Supervised Deep Learning

Other researchers have taken a step further and have implemented deep learning models to solve security problems in IoT devices. Deep learning is based on multi-layer neural network algorithms which imitate the human brain neurons. Deep Learning approaches have gained popularity in much literature for IoT attack detection. In this section, we mention some of the Deep Learning approaches based on the supervised learning technique.

Goyal *et al.* [44] use Artificial Neural Network (ANN) with a single hidden layer and Sigmoid function to detect the abnormal patterns, which use HyperText Transfert Protocol (HTTP) [39] based on Command and Control (C&C) Servers in the IoT environment. The authors analyze the gap duration on GET/POST requests. GET and POST requests are two different types of HTTP requests. The difference between GET and POST is that GET is used for viewing something without changing it, while POST is used for changing something. The algorithm was trained and tested on real-life data, which were collected through Wireshark [42]. The ANN model performs better than the other compared to machine learning techniques SVM and Logistic Regression. As compared to other machine learning algorithms with higher false-negative rates (SVN has a false-negative of 0.21%, Logistic Regression has a false-negative of 0.14%), ANN has the best accuracy with a false-negative of 0.07%. However, limitations include the authors' subjective filtering of the traffic and the small number of features considered.

Another approach to detecting DDoS attacks in IoT devices was suggested by Dosh *et al.*

[30]. They tackled the DDoS prevention problem using Machine learning where they tested different algorithms such as k Nearest Neighbour (kNN), support vector machine (SVM), and deep neural network (CNN) in 5 different IoT devices; Raspberry Pi, WeMo Smart Switch, A Blood Pressure Monitor and Android devices. The most promising result of 99% was achieved during the deep neural network performance. The authors claim that a deep neural network will be the easiest and fastest approach to detecting and preventing DDoS attacks in the early stages.

A deep neural network was used by Kim *et al.* [67] for evaluating ever-evolving network attack strategies using the KDD 1999 [108] data set. The proposed intrusion detection model uses two parameters: four hidden layers and 100 hidden units. Deep neural networks are trained by using the Rectified Linear Unit (ReLU) function [4] as the activation function and stochastic optimization method for deep neural network training. The proposed model has an accuracy of approximately 99%.

Studies regarding supervised machine learning techniques have provided promising results in the IoT security field. However, some of these results can be questionable when it comes to an f-score of 95.7% [48] or an accuracy of 100% [62] using SVM. These results can be true depending on the dataset and the features the authors have considered for their models. For instance, [67], [91], [32] use the same real-life public dataset (KDD) [107] which was generated in 1999 and achieve an accuracy of 99% with DNN, 84.82% with KNN and 83.14% with DT respectively. The results are reasonable considering the performance of each supervised machine learning algorithm, notwithstanding the relevance of the dataset as it is an old one generated in 1999. All three studies performed on real-life data [67], [91], [32], make you question the results of [30] where a 99% accuracy is achieved on an experiment where the authors collected IoT real-life data. Nonetheless, the authors [30] does not make their dataset public, so we cannot compare the results to other algorithms.

2.2.2 Unsupervised Machine Learning-Based Approaches

Unsupervised machine learning is a sub-field of machine learning in which almost all work is based on the data that is used to learn a probabilistic model. One of the advantages of using unsupervised machine learning is that even when the machine is given no supervision,

it may make sense for the machine to estimate a model that represents the probability distribution for a new input based on the previously given inputs. Additionally, unsupervised ML applications are growing in popularity as supervised ML applications are becoming restricted in their application. Some of the most famous unsupervised learning algorithms are K-means Clustering [76], Principal Component Analysis (PCA) [1], Isolation Forest [79]. Below we are mentioning some of the most cited unsupervised learning algorithms used in IoT security; hybrid approach, Machine Learning Intrusion Detection Systems (ML-IDS) [103] which is based on K-means, and a PCA approach [123].

K-mean clustering [76] is an unsupervised learning technique that creates small groups in order to categorize the given data samples as a cluster. The main objective of the method is to determine the k-centroid for each cluster. Among the simple rules for implementing this method are i) first, differentiate the data set into various clusters; each cluster has a centroid; ii) after selecting each cluster, relate it to the nearest centroid and repeat this process until all nodes have been contacted. iii) The method then recalculates based on the average value of each node from every cluster, and iv) Finally, it repeats its previous steps until it coincides with the K-mean value. In his paper, Shukla [103] proposed a hybrid Machine Learning Intrusion Detection System (ML-IDS) based on K-means and Decision Trees. The proposed hybrid approach attains a 71-71% detection rate and significantly reduces the number of false positives. The author compared the results with unsupervised K-means clustering and supervised decision tree algorithms which attains a much higher percentage rate, 70-93%, and 71-80% respectably, but the number of false positives is much higher compared to the hybrid proposed approach. Therefore the hybrid ML-IDS is more accurate than the simple implementation of K-means and decision trees.

Another usage of K-means clustering is introduced in the healthcare IoT (H-IoT) security study [83]. In the healthcare IoT (H-IoT), MacDermott *et al.* [83] proposed a multi-agent-based approach for threat detection. It employs machine learning models for predictive analysis to spot vulnerabilities, make predictions, and find outliers. To communicate with one another, agents were positioned at various strata. Because of this, computational redundancy was reduced. - predictive analysis was used to identify vulnerabilities, make predictions, and detect outliers

by employing machine learning techniques. The dataset is generic LAN data rather than IoT-specific data. A single K-means clustering classifier was used in the study. The author conducted research with WEKA [116] and a single K-means clustering algorithm. The results were presented to achieve an accuracy of 80%. Although, the findings were not compared or optimized against other machine learning classes. IoT systems have limitations on resources; implementing a multi-agent solution may be difficult, especially if it is implemented on IoT devices.

Primary Component Analysis (PCA) [1] is an unsupervised ML also known as a feature reduction technique because it reduces a large set of data into smaller ones while maintaining the same amount of information as the large set. Thus, PCA reduces the complexity of a system. This method can be used to detect real-time intrusion attacks in an IoT system. In this paper, [123], the authors present a new intrusion detection model that utilizes a dimension reduction algorithm and a classifier to perform online machine learning. To reduce the number of features in the dataset from many to a small number, the proposed model applies Principal Component Analysis (PCA). The softmax regression and k-nearest neighbour algorithms are applied and compared in order to develop a classifier. Results from experiments on KDD Cup 99 Data Set [108] show that our proposed model performs optimally in labelling benign behaviours and identifying malicious behaviours. PCA outperforms KNN and reaches 92.68% accuracy on 3-dimension. KNN is left with 85%. In terms of computing complexity and time performance, the model is suitable for detecting intrusions in the IoT. However, if the features of the dataset are increased, the PCA will not perform highly.

Unsupervised Deep Learning

When large datasets are involved, Unsupervised DL algorithms perform better than ML algorithms. IoT environments are characterized by producing enormous amounts of different kinds of data, so DL is especially relevant in IoT applications [73]. Furthermore, Unsupervised DL can automatically model complex features from sample data without any supervision [73].

One of the most famous unsupervised deep learning algorithms is Long Short-Term Memory (LSTM) [45]. LSTM is a recurrent neural network that can learn sequence prediction problems based on order dependence. One of the benefits of LSTM is that they do not suffer from

the optimization hurdles that affect simple recurrent networks [56]. In their paper, Hwang *et al.* [61] propose an online deep learning LSTM-based approach to identify hostile traffic in the Internet of Things network. They focused their work on obtaining semantic information from the packet header. The aim of the study is to discover large-scale assaults in IoT networks by examining packet header details and not the entire traffic flow and reducing the time it takes to find an attack. However, no comparison with other deep learning or machine learning algorithms was presented. The dataset took over 17 hours to preprocess and train the model, according to the study. One potential solution is to use cloud computing to offload the overhead and acquire more GPUs, allowing for faster model time. Using the same data for training, validation, and testing results with 100 percent accuracy. The final accuracy is 97.22% when using a distinct dataset for training and validation. Other studies that use LSTM for security purposes are [75], and [65].

Roopak *et al.* [97] used four different deep learning algorithms to evaluate the public dataset CICIDS2017 [98]; Multi-layer Perceptron (MLP), one-dimensional Convolutional Neural Network (CNN), Long Short-Term Memory (LSTM), and CNN+LSTM. For the combination of CNN and LSTM, the authors used a sigmoid activation function. The results showed that CNN+LSTM performed better than the rest, with an accuracy of 97.16%. Although, the public dataset is unbalanced. The solution proposed by the authors was by duplicating the dataset rather than solving this issue statically. The duplicating of the data can also skew the accuracy in the end. However, there is no proof of the data over-fitting documented in the paper.

In another paper, Guo *et al.* [47] proposed a deep learning technique based on the Bidirectional Encoder Representations from Transformers (BERT) algorithm, named LogBERT. The authors detect anomalous logs from computer systems in order to protect those systems from malicious attacks or malfunctions. The learning algorithm of LogBERT is based on self-supervised data analysis. It learns the typical log sequences' patterns through two novel self-supervised training tasks, and it may spot aberrations in the underlying patterns when they deviate from normal log sequences. The authors performed the experiment on three different datasets Hadoop Distributed File System (HDFS) [118], BlueGene/L Supercomputer System (BGL) [88], and Thunderbird [88], and the results outperformed the state of the art of the paper by an F1 score of 82.32%, 90.83% and 96.64% for each dataset respectably.

Unsupervised machine learning algorithms can show promising results due to their nature of self-learning through different patterns of data. [103], [45] and [75] have based their research on synthetic data. Hence, the results mentioned in those papers can be taken with reserve when it comes to real-life attacks. On the other hand, [47], on which we have based this thesis, provides encouraging results on the computer systems dataset. This is why we would like to take [47] a step further and test it in a real-life IoT dataset.

2.3 Blockchain Based Approaches

There is one more set of approaches that are not contained in those reviewed in Sections 2.1 and 2.2, i.e., blockchain-based techniques. However, we do not review these techniques extensively but mention the next reference [16].

Research on IoT security and blockchain is limited, with the majority of research focused on how blockchain technology can be used to benefit IoT. In essence, a blockchain (blockchain ledger) is an immutable, decentralized, and distributed database ledger that stores asset records and transactions across a peer-to-peer (P2P) network. The data is chained with time stamps and has been verified by miners. In their paper, Bagha *et al.* [16] proposed a blockchain-based framework for industrial IoT devices. The framework enables IoT devices to communicate with the cloud and the blockchain network. The IoT devices are equipped with a single-board computer (SBC) designed to send data to the cloud for storage and analysis and send/receive transactions to other devices on the blockchain network. As a proof of concept, the authors implement a simple platform using Arduino Uno [12] board in order to track the timestamps and Ethereum [117] smart contracts, which are functions that control the data in the blockchain. The authors [16] describe briefly how their framework can be used for machine maintenance and smart diagnostics. Other applications of blockchain in the IoT were reviewed by Christidis *et al.* [23].

2.4 Concluding Remarks

Traditional approaches such as Software-Defined Networking (SDN) [102], trust frameworks [86], and IoT-secured middleware [43] have been a success in IoT security during the time

of their publication. Although with time passing, technology evolving and attacks advancing, machine learning approaches have been a better fit for IoT security. In the studies we reviewed above, we consisted that models of supervised learning [30], [48], [62], [67] tend to be more accurate than unsupervised learning models [83], [47], but the supervised learning models need upfront human intervention to label the data appropriately. However, the accuracy of the machine learning models depends strongly on the dataset, which in the papers we reviewed was either synthetically generated or produced more than twenty years ago. Considering the importance of a relevant and up-to-date dataset and the divergent behaviour of IoT attacks, we found it interesting to generate a real-life IoT dataset through three different malware and apply an unsupervised deep learning approach like LogBERT [47] to detect those IoT attacks.

Chapter 3

Research Methodology

This chapter is structured based on the research questions regarding the thesis we presented in Section 1.3. Following Fenton *et al.* [38], we will follow a typical empirical process in which we will use a quantitative research approach to collect and analyse IoT dataset (**RQ 1** and **RQ 2** in Section 1.3), and investigate whether we can use LogBERT on IoT dataset to identify abnormal behaviour (**RQ 3** and **RQ 4**).

To generate and collect the IoT dataset for our thesis and address the **RQ 1**, we carried out a sequential process to ensure the efficiency of the experiments. The first process is the selection of six IoT devices (see Section 3.1). It is then followed by the data generation process in Section 3.2 and an explanation of attacks performed on each IoT device: Mirai Botnet, RouterSploit and UFONet. The third process consists of an explanation of the LogBERT algorithm [47] in Section 3.3, which is the primarily unsupervised deep learning algorithm we have selected for attack detection in the IoT dataset. The third process is linked with **RQ 2** and **RQ 3**. Lastly, we investigate energy consumption in Section 3.4 to address the **RQ 4**. Results are presented in 4.

3.1 IoT Device Selection

Selecting the right IoT devices for our experiments is one of the first steps of this research. In this section, we will describe the process we went through to select the six devices used in our study. We looked at various factors, including features and compatibility. We believe that our

selection process will be helpful for anyone who wants to embark on their IoT experimentation.

Device	Model
1. Indoor Smart Home Camera	Wyze Cam WYZEC
2. Outdoor Camera	HOSAFE Outdoor Wifi Camera
3. Home Router (1)	TP-Link ACS1750
4. Home Router (2)	D-Link AC1200
5. Smart Home Hub	Phillips Hue Smart Hub
6. DVR Digital Video Recorder	ANNKE DVR 8CH

Table 3.1: IoT Device Information

A 2016 report by ForeScout Technologies [27] presents the top IoT devices that are most vulnerable, highlighting the ease of compromise of security for some cameras, routers, and light bulbs. Based on this report [27] and the extensive usage of cameras, routers, and light bulbs in smart homes and smart cities [59], we decided to research the most commonly purchased devices in the three main categories: cameras, home routers, and smart light bulbs. We referred to BestBuy Canada [105] for the most used devices and their brands. However, the most used are not necessarily the most vulnerable, so we filtered them out based on customer reviews and security descriptions. The result of this study is presented in Table 3.1.

3.2 Data Generation

As the first contribution to this thesis, we produced data for all six IoT devices in network behaviour and energy current (measure unit: ampere) under normal and abnormal states. The normal state is considered when the IoT device is plugged into an outlet and connected to the Internet. The abnormal state is achieved by exploiting the IoT devices through different malware. As part of our experiments, we selected three malware types that share the same typology but have different architectures (refer to Section 3.2.2). Nevertheless, there are many malware in the world, and some of the malware published on GitHub or exchanged in hackers' forums are not necessarily working on IoT devices or they have broken code parts.

For this research, we first tested the following malware: Hybrid Bonensi V 0.15 [21], LOIC [22], ZMAP [34], MANA V 4.1 [94], Wiseg3ck0-AIM-DDOS [14], DDOSimm [7], SAKURA [46],

Paraxinal [15] - which were not successful on IoT devices. Upon further researches and code implementations, we selected the following three malware tools: Mirai botnet [99], RouterSploit[112] and UFONet [36]. We are describing each malware tools in Section 3.2.2.

As mentioned in the paragraph above, there are many different malware in the cyber world. For our experiment, we focused on three malware that have different architectures but similar exploiting techniques. Mirai Botnet, RouterSploit exploitation framework, and UFONet disruptive toolkit can be used to participate in launching a DoS/DDoS attack. In each of the three malware, different communication protocols like TCP, UDP, and HTTP are used to launch a series of data packets to exhaust normal communications among the IoT devices. However, Mirai Botnet uses command and control server, RouterSploit uses five different modules within its framework and UFONet uses third party websites, to launch a series of data packets in the IoT devices.

For each device, we collected the network behavior and electric current during the attacks. The network behavior activity for all IoT devices was recorded using the Wireshark software [42]. Wireshark is a network protocol analyzer which captures data packets exchanged between different devices within the same network connection. We connected the IoT devices with a computer using the two home routers represented in Figure 3.1 in order for the Wireshark to detect and record all the network activity of the IoT devices. A detailed table of the measured data for the network behaviour can be found in Table 4.1.

As per energy consumption, data activity was challenging since we could not find any commercial energy monitor that could record, collect, and accurately measure low voltage IoT devices. IoT devices are known for low voltage operation [57]. Hence, we had to create an Arduino Uno [12] based solution. Arduino Uno is a microcontroller board that uses the Allegro CT Sensor ACS712 for measuring low voltage. The device is programmed in C++ to read and register the electric current (in ampere) for the IoT devices every ten milliseconds.

All the attacks and data collection were performed in a virtual box environment for security issues. The data that was being gathered by the devices were used to generate graphs and statistics that can be used for further analysis. The experiment setup is represented in Figure 3.1.

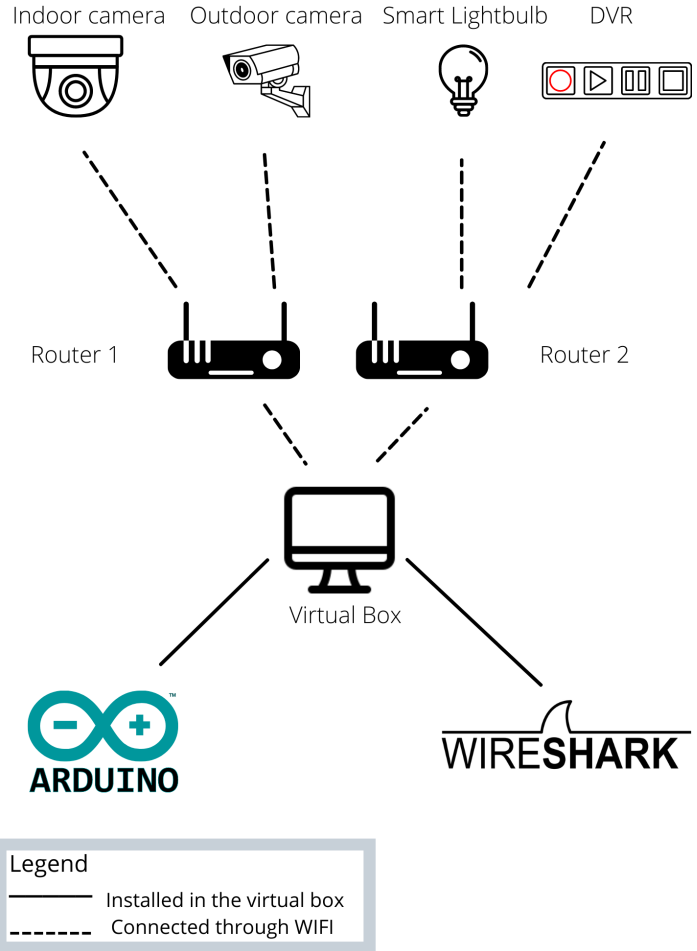


Figure 3.1: IoT Architecture Environment

3.2.1 Normal Data Generation

The first step in our experiment phase was to generate real data on the normal state of the network behavior in all six IoT devices. In the normal state, we consider the data gathered during a 24h period that the device is plugged in and connected to the Internet. There is no further simulation or imposed attack during this period. Real-time data was generated and saved for each IoT device in its normal state.

3.2.2 Abnormal Data Generation

We then proceeded to perform three different malware on each of the devices. The malware tools were chosen based on typology of the malware; we were looking for different malware within the same typology performed on IoT devices, as explained in the previous section (Section 3.2). In this study we used three different malware; Mirai Botnet [99], RouterSploit [112] and UFONet [36]. Initially, we picked Mirai Botnet as it was the first powerful malware to launch DDoS through remotely controlled bots or botnets [10]. RouterSploit and UFONet were selected after extensive experiments to infect the six IoT devices with different malware.

Mirai Botnet

This first malware, Mirai Botnet is a worm-like malware that generates a single type of threat using two servers; control and command server (C&C) and scanner and loader server, to launch an enormous number of data packets in order to exhaust the IoT devices. In addition to the two servers, Mirai malware uses bots or botnets which are controlled by a botmaster, to launch a brute-force attack to exploit the credentials of weakly configured IoT devices. For security reasons, we used a virtual machine as a botmaster to launch and stop the attack.

According to Fairy *et al.* [37], botnets are the most serious threats among other forms of malware like phishing, fraud, and DDoS. Unlike other types of malware where hackers need to attack a specific person or industry, botnet looks for devices with vulnerable endpoints. The goal of the botnet is to infect multiple devices on a large scale and convert them into zombies. It will be easier for the hacker to deploy millions of payloads and exhaust the network.

Botmaster is like a 'lighthouse' that controls the overall operation of the attack, managing and multiplying bots. The control and command server, or C&C server, deals mainly with the bots' delivery. It is used as a link between the botmaster and bots. If an attack is launched by the botmaster, C&C delivers this information to bots.

The command server (C&C) helps with the storing of the bots information which are later on used in attacks towards the IoT device. To set up this server, we created a MySQL [33] database. We set up the MySQL relational database using the open source database management system

MySQL, and the MySQL Client was used to access the database remotely.

The scanner and loader server is usually used to scan through different Internet Protocol (IP) and check for vulnerabilities. In this case, we already knew the IP of our devices (refer to Table 3.3 so in order to elude the possibility of spreading the attack beyond the desired scope, we provided the IP to the scanner and loader server manually.

Both the control and command server (C&C) and the scanner and loader server, were created following the steps from the Mirai Botnet documentation in Github [99]. Meanwhile, we recorded both the IoT devices' network traffic data and electric current using Wireshark and Arduino Uno, respectively (See Table 4.1). The attack was performed using one device at a time rather than attacking all the IoT devices simultaneously.

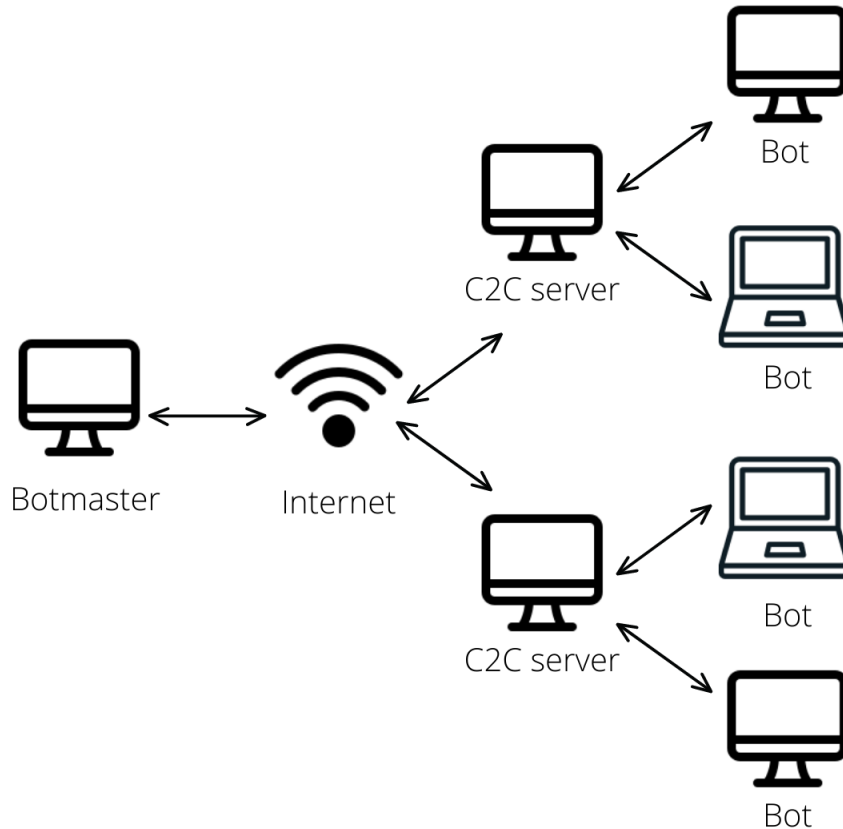


Figure 3.2: Mirai Botnet Architecture

RouterSploit

This second malware framework can generate different types of threats, which are summarized in Table 3.2, each of them targeting some particular vulnerability. RouterSploit is an open-source framework that allows an attacker to scan through the network and exploit a vulnerable device [112]. Model is introduced in the RouterSploit documentation [112] as the different functionalities which they have the ability to check whether the remote target is vulnerable before sending off an exploit. The framework consists of five different modules that aid penetration testing operations; exploits, creds, scanners, payloads, and generic (Table 3.2).

Table 3.2: RouterSploit Models

Model	Explanation
Exploits	identify the vulnerabilities
Creds	test different credentials
Scanners	check for vulnerable devices
Payloads	generate payloads to different architectures
Generics	perform generic attacks

Before getting started with RouterSploit [112] to exploit the IoT devices, we used Network Mapper (NMAP) [82] to identify the IP and port number of the devices which are used for this experiment (See Table 3.3). NMAP scanning was performed as a precautionary measure to prevent accidental exploits on other IoT devices in the lab. RouterSploit automatically assigns the port number 80 because it is the default port for Linksys routers [112]. The most frequently used IoT device ports are 22, 80, 88, and 8080 [40].

IoT Device	IP
Indoor Smart Home Camera	172.31.1.1
Outdoor Camera	172.31.1.21
Home Router (1)	172.31.1.18
Home Router (2)	172.31.1.15
Smart Home Hub	172.31.1.12
DVR Digital Video Recorder	172.31.1.16

Table 3.3: IoT Devices Internet Protocol

In this experiment we performed the RouterSploit attack separately for each IoT device, which made it easier to identify the port number through NMAP. After the port number and

the IP of IoT device was identified, we moved to the second step of finding the vulnerabilities of the IoT device under investigation. From the Scanner option in the RouterSploit menu, we used Autopwn to check the IoT device for vulnerabilities. The algorithm goes through all the vulnerabilities list applicable to that particular device and flags out the ones to which the device is most vulnerable to. On the other hand, the exploitation module can provide the username and the password for the IoT device in any form; hashed, plain, or authenticated without a password.

DVR Digital Video Recorder: The best example from our IoT device tested is the DVR Digital Video Recorder which, after running a brute force, its username and password were easily guessed admin, admin. Brute force is a hacking method which uses trial and error to guess the username and password of a device. Figure 3.2.2 displays all the brute forces suggested from the Creds Model for the DVR Digital Video Recorder. Through the exploitation module, we managed to enter the Command Prompt (CMD) of the CCTV camera and perform a reverse Transmission Control Protocol (TCP) from the RouterSploit.

```
creds/generic/snmp_bruteforce
creds/generic/telnet_default
creds/generic/ssh_default
creds/generic/ftp_bruteforce
creds/generic/http_basic_digest_bruteforce
creds/generic/ftp_default
creds/generic/http_basic_digest_default
creds/generic/ssh_bruteforce
creds/generic/telnet_bruteforce
creds/routers/ipfire/ssh_default_creds
creds/routers/ipfire/telnet_default_creds
creds/routers/ipfire/ftp_default_creds
creds/routers/bhu/ssh_default_creds
creds/routers/bhu/telnet_default_creds
creds/routers/bhu/ftp_default_creds
creds/routers/linksys/ssh_default_creds
creds/routers/linksys/telnet_default_creds
creds/routers/linksys/ftp_default_creds
creds/routers/technicolor/ssh_default_creds
creds/routers/technicolor/telnet_default_creds
creds/routers/technicolor/ftp_default_creds
creds/routers/asus/ssh_default_creds
creds/routers/asus/telnet_default_creds
creds/routers/asus/ftp_default_creds
creds/routers/billion/ssh_default_creds
creds/routers/billion/telnet_default_creds
creds/routers/billion/ftp_default_creds
creds/routers/zte/ssh_default_creds
creds/routers/zte/telnet_default_creds
creds/routers/zte/ftp_default_creds
```

Figure 3.3: RouterSploit Cred Model Options for DVR Device [36]

The internet's Transmission Control Protocol (TCP) is how one IoT talks to another computer or other IoT via the internet by compressing data packets and sending them to the correct

destination. The transport layer, more commonly known as TCP, is in charge of packaging massive data into network packets and sending them to a lower-layer component for processing. The transport layer task is to take the extensive data and arrange it into network packets before sending them to a lower level for analysis. Reverse TCPs are when the attacker forces the host to connect to the attacker's shell. That is the fundamental concept of a reverse TCP. When the host starts a connection, we call it a forward connection. However, when the server establishes a link with the host, we refer to it as a reverse connection (extremely uncommon). There are two different types of shells used mainly in Reverse TCP.

Bind Shell: This is a shell in which the target machine waits for an incoming connection by opening a communication port on the victim machine. As soon as the attacker connects to the listener on the victim machine, commands are broadcast.

Reverse Shell: It is a form of shell in which the target computer initiates the connection with the attacking machine. The listener port on the attacking machine is used to receive the connection, allowing for code or command execution depending on how it is used.

Similar to the DVR approach was taken for the rest of the IoT devices. We used the reverse shell to load the payloads on all six devices after being exploited through RouterSploit, and we monitored the energy consumption and network behavior during the malware performance.

UFONet

UFONet is a peer-to-peer (P2P) [78] and disruptive cryptography toolkit which allows the performance of DoS and DDoS attacks through exploitation using third party websites to act as a botnet and abusing protocols [36]. The architecture of P2P partitions tasks between equally privileged peers without a need of a central coordination server. For this experiment, the open ports and the IP for all the IoT devices were done in the previous experiment of RouterSploit using NMAP. In UFONet, we used Slowloris, SPRAY, SMURF, UFOUDP, and UFOACK as traffic generating tools. The UFONet traffic generating tools target the UDP, TCP, ICMP, ACK, and HTTP protocols that the six IoT devices used for data packet exchange and communication.

The List of UFONet Traffic Generating Tools is show below:

- **Slowloris:** is a HTTP get based attack which can send rapid amounts of HTTP requests into the device.
- **SMURF:** attack targets specifically the ICMP protocol, which sends faked echoed packets to the victim's IP.
- **SPRAY:** is a TCP-SYN Flood type of attack which targets specifically the TCP protocol. The victim IoT devices are bombarded with fabricated SYN requests which contain fake IP addresses.
- **UFOUDP:** aims for UDP protocols to overwhelm them with fake data packets.
- **UFOACK:** is another attack which focuses on TCP-ACK protocols by overloading both of the protocols with fake data packets.

UFONet offers a combination of all attacks by generating the fake datapackets from third party malicious websites. The UFONet toolkit is able to exploit vulnerabilities in protocols used by the IoT devices and generate traffic to these devices. The experiment was performed on all six IoT devices, where the DVR was successfully hacked.

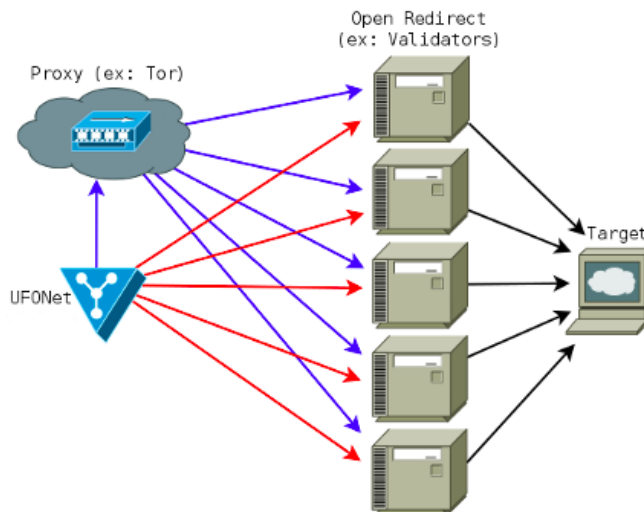


Figure 3.4: P2P Botnet Architecture of UFONet [36]

3.3 Analysis of LogBERT

The LogBERT [47] algorithm is a combination of LogParser [54] and the famous Google's algorithm BERT [29]. In order to explain the LogBERT algorithm in details, we start off with LogParser which includes two algorithms of Spell [31] and Drain [55] which are used in the data preprocessing in LogBERT algorithm. Secondly, we explain the logic behind the BERT algorithm which employs two training techniques Masked Language Modeling and Next Sentence Prediction. Both techniques are used in LogBERT training method to mask and interpret the logs.

3.3.1 LogParser

Log parsing [54] is a process of extracting information from log files and converting it into a format that is more suitable for analysis. The parsed data can then be used to generate reports or to visualize trends and patterns. There are many different ways to parse log files, depending on the structure of the logs and the desired output. Some common methods include regular expression matching, text processing commands, and dedicated log parser software programs. In most cases, a combination of these methods is used in order to achieve the best results.

The first step in any log parsing process is to identify the structure of the log file. This can be done by looking at an example of the file or by reading the documentation that accompanied the creation of the file. Once the structure is understood, it is then possible to identify the specific information that needs to be extracted. This process can be repeated for each log file that needs to be parsed. In some cases, it may be necessary to write custom scripts or programs in order to automate the parsing of large numbers of files. However, there are many off-the-shelf software solutions that can parse log files with little or no configuration required.

Once the data has been parsed, it can then be analyzed in order to extract valuable insights. This step usually involves generating reports or visualizations that make it easy to identify trends and patterns. Depending on the nature of the data, different methods of analysis may need to be used to maximize its effectiveness. Log parsing can be a complex and time-consuming task, but it is often essential in order to make sense of large volumes of log data. By understanding the structure of log files and using the right tools, it is possible to extract valuable information that

can help improve the performance of systems and identify potential problems. To understand the structure of the log, two different parser are used; Streaming Structures Log Parser and Drain.

Streaming Structured Log Parser

Spell, a streaming structured log parser [31] for system event logs, uses the longest common subsequence (LCS) algorithm to find matches across different log formats with high accuracy. The LCS problem is the problem of finding the longest sub-sequence common to all sequences in a set of sequences (often just two). It differs from problems like edit distance and longest increasing sub-sequence by being concerned with a static rather than dynamic solution. For example, given the strings "ABCDGH" and "AEDFHR," the longest common sub-sequence is "ADH" and has a length of three. There may be more than one LCS of a given input set; for example, in English texts, one frequently finds instances of the phrases "a chip off the old block" and "chipping away at." The SPELL algorithm is used in LogBERT to go through the logs and find a 'meaning' or a common path in the normal behavior data.

Drain

Drain [55] analyses the raw log messages. It can extract log templates from raw log messages and automatically split them into separate log groups. It uses a parse tree with a fixed depth to guide the log group search, effectively avoiding building a deep, unbalanced tree. In addition, specially designed parsing rules are compactly encoded in the nodes of the parse tree. Drain is used in LogBERT to through the raw data and automatically split the log groups (See Figure 3.5).

3.3.2 BERT

Bidirectional Encoder Representations from Transformers (BERT) is the second part of the LogBERT. As mentioned in the previous section, LogBERT uses two of the most powerful training methods of BERT, Masked Language Modeling (MLM) and Next Sentence Prediction (NSP) to mask and interpret the logs .

Bidirectional Encoder Representations from Transformers BERT's [29] major technical innovation is applying Transformer [3], a well-known attention model, to language modeling. In

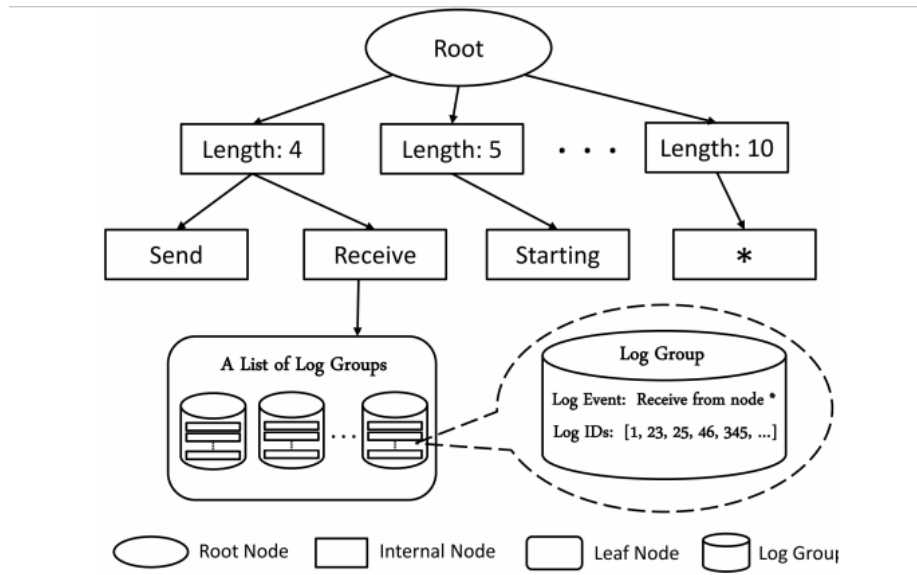


Figure 3.5: Structure of Parse Tree in Drain (depth = 3) [55]

contrast to earlier studies that examined a text sequence from left to right or combined left-to-right and right-to-left training, the new paper investigates bidirectionally trained language models. The findings of the study demonstrate that bidirectionally trained language models have a more profound sense of language context and flow than single-direction language models.

A transformer, an attention mechanism that learns contextual connections between words (or sub-words) in a text, is used by BERT. Two separate components make up the Transformer in its standard form: a text encoder reads input from a text file, and a text decoder predicts output based on what the text says. Only the encoder mechanism is required since BERT's objective is to create a language model.

The Transformer encoder, as opposed to directional models that read text input sequentially (left-to-right or right-to-left), reads the entire sequence of words in one go. It is thus considered bidirectional, albeit it might be more accurate to call it non-directional. This feature allows the model to identify the context of a term based on all of its surroundings (both left and right). The Transformer encoder is shown in the figure below. The input is a sequence of tokens that are first encapsulated and then analyzed in the neural network. The output is a sequence of vectors with H elements, each vector representing an input symbol with the same index.

There is a problem with identifying a prediction objective while training language models.

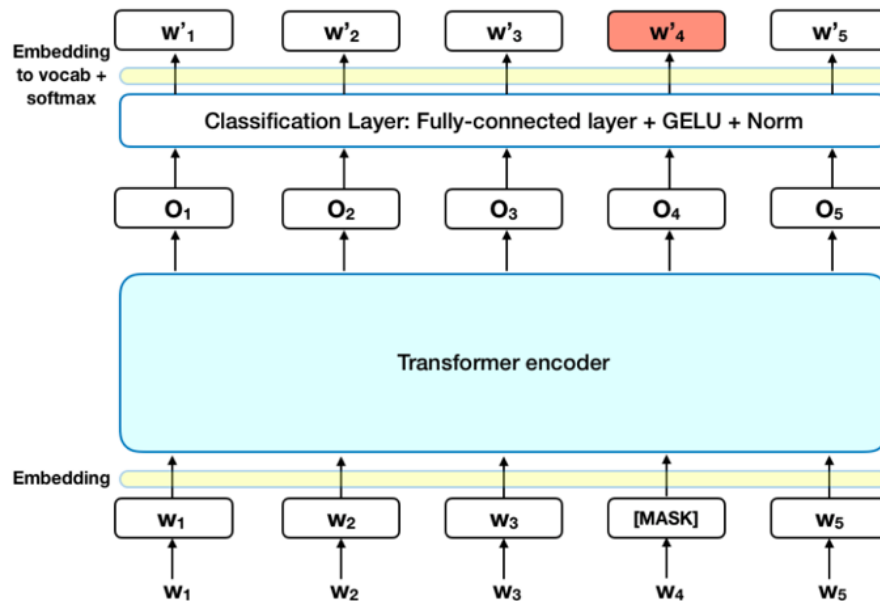


Figure 3.6: BERT's Transformer Encoder [29]

Many models forecast the next word in a sequence, which is an inherently restricting directional approach. BERT employs two training techniques to overcome this issue:

Masked Language Modeling (MLM)

Masked Language Modeling is used to change 15% of the words in each sequence with a [MASK] token before being fed into BERT. The model next attempts to predict the original value of the masked words based on the context supplied by the other non-masked words in the sequence. The BERT loss function only considers the masked values' prediction and disregards the prediction of non-masked words. As a result, directional models converge more slowly than directional ones, which is offset by their enhanced context-awareness.

Next Sentence Prediction (NSP)

In order to train the model to predict whether the second sentence in a pair of sentences is the following statement in the original document, pairs of statements are received as input,

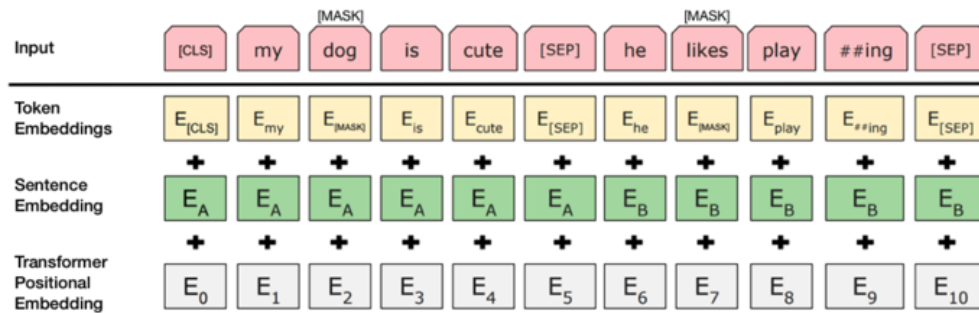


Figure 3.7: Next Sentence Prediction Chart [29]

and the second sentence is learned. In training, half of the inputs are pairs in which the subsequent sentence is the second sentence from the source document, with the other half choosing a random phrase from the corpus as the second statement. The conclusion is that the random phrase will be unlinked from the preceding statement. Masked LM and Next Sentence Prediction are trained together in this case, with the aim of reducing the overall loss function of the two algorithms.

3.3.3 LogBERT

This section covers the fundamentals of LogBERT, the LogBERT association with LogParser and BERT algorithms, as well as its incorporation into the IoT dataset. BERT-based LogBERT is a deep learning neural network for log sequence anomaly detection. LogBERT's objective is to learn a log representation that can be used for anomaly detection. Before moving to the training of the LogBERT model, the data is firstly processed using the two algorithms from LogParser, SPELL and DRAIN. DRAIN analyses the raw log sequences and extracts them to a template by marking certain letters with a symbol ('*'). SPELL on the other hand looks to find the longest sub-sequence with the same pattern and 'find' a meaning or a common path behaviour between normal behaviour data. Later, LogBERT trains two self-supervised tasks, namely Masked Log Key Prediction (MLKP) and Volume of Hypersphere Minimization to analyze log sequenced bidirectionally. The transformer encoder is designed to capture long-term dependencies in sequences by attending to all query and key vectors in the sequence. The self-attention mechanism allows

LogBERT to automatically learn which log keys are important for predictions and which can be safely ignored. The MLKP and VHM tasks are used to train LogBERT such that it can learn from both normal and abnormal log sequences (refer to Figure 3.8). This makes LogBERT robust to concept drift and able to detect novel anomalies. LogBERT is also able to generalize well to new data sets because it is not tuned to any specific anomaly type. The architecture of LogBERT algorithm is presented in Figure 3.8.

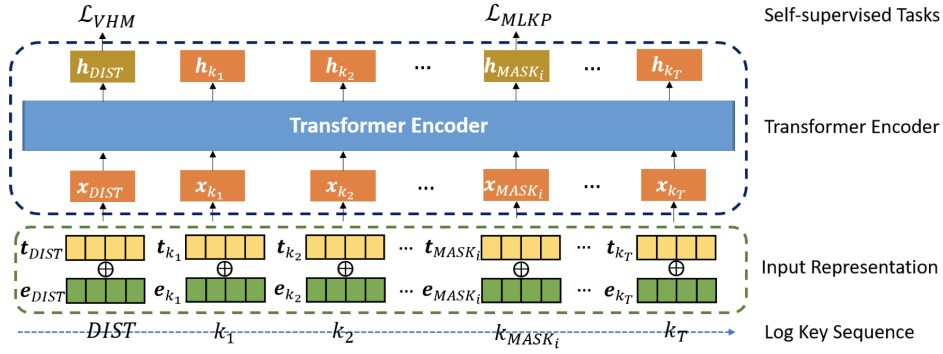


Figure 3.8: Overview of the LogBERT Architecture [47]

The objective of the training is to accurately predict the randomly masked log keys. As a result, LogBERT is encoding the prior knowledge of normal log sequences. Specifically, we feed the contextualized embedding vector of the i th [MASK] token into a softmax function, which outputs the following probability distribution over the entire set of log keys:

$$\hat{y}_{[MASK_i]}^j = \text{Softmax}(\mathbf{W}_C \mathbf{h}_{[MASK_i]}^j + \mathbf{b}_C) \quad (1)$$

Both \mathbf{W} and \mathbf{b} are trainable parameters [47]. Masked log key prediction is then performed using cross entropy loss as the objective function:

$$\lambda_{MLKP} = -1/N \sum_{j=1}^N \sum_{i=1}^N y_{[MASK_i]}^j \log \hat{y}_{[MASK_i]}^j \quad (2)$$

The function of LogBERT training is the summation of the balanced hyper-parameter of the two training tasks.

$$\lambda = \lambda MLKP + \alpha \lambda VHM \quad (3)$$

Rather than individual log sequences, each log sequence consists of a sequence of log keys [k1, k2, k3, ...]. A special token k[dist] is also added at the beginning of each log sequence, which is used as a Distance token. Using this token, we are able to calculate the distance between this log sequence and the center (the center is computed by taking all log sequences in the input and multiplying them together).

The prediction of the data presented through testing is normal or abnormal. Using the Softmax probability distribution as a predictor of the MASK tokens, we mask the tokens randomly in the sequence as with training. Upon reaching the MASK token predictions, we take top- g predictions for each MASK token prediction. The top- g predictions are expected logs if the patterns are normal. If the actual log from the sequence is present in the top- g candidates, then the key is considered normal. If the log is not present in the prediction list of that MASK token, then the log is considered anomalous. For each given sequence, if there is more than r anomalous log we consider the whole sequence as anomalous. Both g and r hyperparameters values are tuned in based on the dataset.

3.4 Energy Consumption

Energy comes in many different forms, e.g. chemical energy [77], thermal energy [9], and motion energy [28]. We use the term energy to refer to the movement of electrically charged particles, which produce electrical energy. In this work, we are interested in identifying energy consumption and investigating the changes in energy consumption when IoT devices are in a normal and abnormal state. This is not, however, the primary aim of our dissertation. The energy consumption results are included in Section 4.3 to inspire future research projects.

As mentioned in the Section 3.2, we generated network behaviour and electric current data for all six IoT devices under normal and abnormal states. In this section we are going to explain the technology used to record the electric current data and how the power data was generated. The terminology explained in Table 3.4 is based on Ohm's Law [68] and is used across Section 3.4 and Section 4.3.

Term	Uni	Explanation
Voltage	Volt	the force that drives electrons through electrical component.
Current	Ampere	the flow rate of electrons through said electrical components.
Power	Watt	describes electrical energy rate transferred in an electric circuit.
Energy	Kilowatt hour or watt second	is the product of power in watts or kilowatts multiplied by running time in hours or seconds.

Table 3.4: Terminology Used Across Energy Consumption Sections

Each of the six IoT devices operates under different voltages, which are presented in Table 3.4. As mentioned in Section 3.2 we were able to register the electric current during normal and abnormal states using an Arduino Uno microcontroller with the Allegro CT Sensor ACS712. Given the voltage (volt) and current (ampere) of the IoT devices, we can calculate the power of each IoT device based on Ohm's law [68] (see Equation 4).

$$P = I * E \tag{4}$$

where: **P** represents Power (Watts), **I** represents Current (ampere), and **E** represents the Energy (watts seconds).

In physics, power is defined as the rate at what work is done or energy is transformed from one form to another. For example, power can be the rate at which electrical energy is converted into heat or light. It can also be the rate at what mechanical work is done (for example, when a machine is used to lift a weight). Power is typically measured in watts. The higher the wattage, the higher is the power the device or process requires. This is important to consider when evaluating the energy consumption of different devices.

Energy consumption of an IoT device can be calculated based on the power consumption of the device and the time for which it is used. The power consumption of an IoT device can be measured in watts (W) or kilowatts (kW). The time for which an IoT device is used can be measured in seconds (s), hours (h), days (d), weeks (w), months (m), or years (y). The formula of energy is presented in Equation 5.

$$E = P \times t, \tag{5}$$

where: **E** represents Energy (Watts second), **P** represents Power (Watts), and **t** represents the time (Seconds).

Based on Ohm's physics law [68] from the information provided, the voltage of the device and the electric current we have registered in our experiment, we can also measure different properties like and power (measured in watts) and energy (measured in watt second) (see Equation 4). The results of this work are presented in Section 4.3.

Devices	Voltage
1. Wyze Cam WYZEC Indoor Camera	5V
2. HOSAFE Outdoor Wifi Camera	12V
3. TP-Link ACS1750	12V
4. D-Link AC1200	9V
5. Phillips Hue Smart Hub	5V
6. ANNKE DVR 8CH	12V

Table 3.5: IoT Devices Voltage Information

Chapter 4

Experimental Results

In this chapter, we answer our research questions by presenting the results and the performance of LogBERT, a deep-learning neural network for log sequence anomaly detection based on Bidirectional Encoder Representations from Transformers (BERT), on IoT devices.

In Section 4.1, we answer **RQ 1**: how can we generate real-life IoT datasets to get as close as possible to real-life solutions? We present the data generated during normal and abnormal states, whereas for the abnormal states, we use Mirai Botnet, RouterSploit, and UFONet, see Section 3.2.2 for their descriptions. Then, we describe the experimental setup in Section 4.2.1, 4.2.2 and 4.2.3, and explore what balance dataset (see Section 1.5) leads to a better performance of LogBERT in Section 4.2.4. All these processes ensure consistency along the dataset for better analysis and performance.

To address our two research questions regarding the performance of LogBERT: **RQ 2**: can LogBERT [47] extend its usage in IoT network traffic data, and **RQ 3**: what is the performance of LogBERT algorithm with real-life data in IoT devices in comparison to other machine learning algorithms; we compare the performance of the LogBERT algorithm in IoT devices, with Principal component analysis (PCA), One class support vector machines (OCSVM), Isolation Forest (IF), and LogClustering in Section 4.2.5. We further compare the performance of the LogBERT algorithm in IoT devices with the performance of the LogBERT in computer logs to showcase that LogBERT usage can be extended to IoT devices as well.

Finally, we address our last research question **RQ 4**: whether energy measures can give

an indication of anomaly detection, we present the results of our preliminary work on energy consumption in Section 4.3.

4.1 Data Generation Results

RQ 1. How can we generate real life IoT dataset in order to get as close as possible to real-life solutions?

Data Sets	Feature	Description
Network Traffic	No.	Packet id
	Time	Time of the packet capture in milliseconds
	Source	IP/MAC source from where the packet is launched
	Destination	IP/MAC destination of where the packet is going
	Protocol	Protocol type
	Length	Size of the packet in bytes
	Source Port	Source port of the packet
	Destination Port	Destination port of the packet
	Info	Data packet's content
Electric Current	Timestamp	Time registration.
	Current	Current of the device in Ampere

Table 4.1: Network Traffic and Electric Current Dataset Features

In this section, we present a total number of 9,187,987 data packets generated in normal and abnormal states. We describe the data features for network traffic which are collected using Wireshark [42] and electric current data collected via Arduino Uno [12] (See Section 3.2. Along this section, we provide We further explain the differences between the first set of normal behaviour data and the second one.

In Table 4.1, we describe the features of the network traffic and electric current respectably. The network traffic dataset consists of multiple rows of packets. It is important to highlight the Info feature which consists of packet content on which the LogBERT algorithm is trained. Source, Destination, and Protocol are features used in data preprocessing in Section 4.2.2 and data process in Section 4.2.3.

The number of data packets generated is presented in Table 4.2. We present two sets of data packets; the first and second normal datasets.

The experiment for generating normal behavior data was redone due to the data imbalance problem we faced while analyzing the results using LogBERT. The Outdoor Camera and Home Router (1) had a collection of 70 and 162 data packets which makes it insufficient for LogBERT to train and find a pattern in a normal dataset. The normal network traffic data generation (second set of normal data) was redone to a 24h time frame, which resulted in an increase of LogBERT performance (refer to Section 4.2.4)

IoT	ND-1	ND-2	MD	RD	UD
Indoor Smart Camera	2,203	-	3,514,402	9,522	733
Outdoor Camera	70	136,008	1,059,031	5,421	-
Home Router (1)	162	89,076	1,059,031	3,241	-
Home Router (2)	2,317	88,863	1,057,975	1,232	-
Smart Home Hub	43,181	101,261	789,717	55,641	-
DVR Digital Video Recorder	591	89,711	1,069,315	8,712	571

Table 4.2: Number of Data Packets Generated in Network Traffic **Legend** : First Set Normal Data - **ND-1**, Second Set Normal Data - **ND-2**, Mirai Data - **MD**, RouterSploit Data - **RD**, UFONet Data- **UD**,

Table 4.2 shows the number of packets generated in first experiment (normal state) (**ND-1**), followed by the second set in normal state (**ND-2**) and the three abnormal states; (**MD**) - Mirai Data, (**RD**) - RouterSploit Data, (**UD**) - UFONet Data.

As we see in Table 4.2, there are a few missing data; indoor smart camera ND-2, outdoor camera UD, home router 1 and 2 UD, and smart home hub UD. The ND-2 of the Indoor Smart Camera is missing because by the time we had to redo the experiment, the Indoor Smart Camera was no longer available as its firmware got upgraded, and we could not record the new data. The firmware provides instructions that aid hardware in starting up, communicating with other devices, and performing basic input/output tasks. By upgrading the firmware, the instructions for communications and data packet exchanges have been changed and do not resemble the initial set of instructions. As of the attacks, the Mirai botnet and RouterSploit were successfully exploited on all six IoT devices. In contrast, we were able to successfully attack the ANNKE DVR 8CH IoT and Wyze Cam WYZEC devices using the UFONet. However, UFONet was unable to perform successfully on other IoT devices for two reasons; first, most data packet content in UFONet occurs over HTTP, which some devices do not use, and second, websites from which

UFONet launches the bots are frequently shut down due to the location of the servers and the time of the malware's execution. UFONet has successfully worked on two devices because the database of this UFONet toolkit includes a known vulnerability in those devices.

4.2 LogBERT Results in IoT Data

The process we followed in analyzing the network traffic data with the LogBERT algorithm is divided into five steps:

- Step 1.** Data analysis in Section 4.2.1 where we explain the similarities between the network traffic data and IoT log data,
- Step 2.** Data preprocessing in Section 4.2.2 where we describe the process of removing noisy data and grouping the data packets with the same IP exchanges,
- Step 3.** Data processing in Section 4.2.3 where the preprocessed data get processed for the LogBERT algorithm,
- Step 4.** LogBERT performance in Section 3.3
- Step 5.** Comparison 4.2.5 of LogBERT results with other algorithms PCA, SVM, IsolationForest, and LogClustering. We picked these four algorithms as they represent one of each category we have mentioned in our literature review in Chapter 2.

4.2.1 Data Analysis

In the LogBERT paper [47], the authors perform the experiment on computer logs. However in this thesis we are using IoT data packets to train and test the LogBERT algorithm.

The network traffic data gathered with the help of Wireshark software [42] consists of data packet exchanges of the IoT devices, see Figure 4.1.

In computer networks [72], a data packet is a small segment of a larger message. On the other hand, log files contain information about usage patterns, activities, and operations within IoT devices. Considering the similarities between network traffic data and log data in IoT devices (refer to Figure 4.1 and Figure 4.2) we are going to apply the LogBERT algorithm on IoT data packets, while it is traditionally applied on computer logs.

No.	Time	Source	Destination	Protocol	Length	Info
0	1	0.000000	52.33.129.122	172.31.1.14	TCP	66 6601 > 58300 [ACK] Seq=1 Ack=1 Win=7 Len=0 T...
1	2	5.071380	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
2	3	6.095379	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
3	4	7.119381	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
4	5	40.471413	::	ff02::1:ff2f:6208	ICMPv6	78 Neighbor Solicitation for fe80::212:41ff:fe2f:...

Figure 4.1: Normal Behaviour Data Wyze Cam WYZEC device (Indoor Smart Home Camera)

Filter by label or text search

Cloud Function, iot, us-central1 All logs Any log level Last hour Jump to now

2017-10-06 JST

↓ ...Loading...

▶	λ	14:25:09.294	iot	157833377882970	Function execution took 7 ms, finished with status: 'ok'
▶	i	14:25:09.292	iot	157833377882970	device29, 1507267509221, 20, 35.6603984, 139.7281902, 20
▶	λ	14:25:09.288	iot	157833377882970	Function execution started
▶	λ	14:25:09.049	iot	157833994309797	Function execution took 102 ms, finished with status: 'ok'
▶	i	14:25:09.045	iot	157833994309797	device9, 1507267508875, 30, 35.6603124, 139.7285123, 20

Figure 4.2: Google IoT Log Data taken from Google Cloud <https://cloud.google.com/community/tutorials/cloud-iot-rtdp>

4.2.2 Data Preprocessing

Data preprocessing consists of two main steps: data cleaning and then cleaned data grouping.

Step 1. Data Cleaning

Action 1.1 Identify all the missing information in the dataset (refer to Figure 4.3).

Action 1.2 Remove all the packets from the dataset which have missing information. In the entire dataset, there are 0.1% of missing values. Cleaning the dataset from the missing values packets leads to an increase in productivity and a higher quality of decision-making in machine learning algorithms.

Step 2. Data Grouping

Action 2.1 Group data packets together based on their communication exchange between Source

No.	Time	Source	Destination	Protocol	Length	Info
0	1	0.000000	52.33.129.122	172.31.1.14	TCP	66 6601 > 58300 [ACK] Seq=1 Ack=1 Win=7 Len=0 T...
1	2	5.071380	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
2	3	6.095379	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
3	4	7.119381	PcsCompu_73:1d:1d	a2imarke_2f:62:08	ARP	42 Who has 172.31.1.14? Tell 172.31.1.1
4	5	40.471413		← fe80::212:41ff:fe2f:...	ICMPv6	78 Neighbor Solicitation for fe80::212:41ff:fe2f:...

Figure 4.3: Normal Behaviour Data Wyze Cam WYZEC device

IP, Destination IP and Protocol which made the communication between two IP-s possible. We achieve that by looping throughout the data and joining the same the Source IP and Destination IP in one string. We map the joined string with the same Protocol and generate the block grouping (blk_n.o) (See Figure 4.4). The reason for grouping the packets together, is to have the full message exchange between two IP-s and to better train the LogBERT algorithm [47].

Action 2.2 Create a separate file that labels the packets into **0** for packets exchanged under normal state and **1** for packets exchanged under abnormal state. This is done to check the performance of LogBERT after training in the later steps.

```

3 43 12.48949278 47.52.128.161 172.31.1.11 TCP 66 blk_13: 19888 > 49158 [ACK] Seq=1 Ack=113 Win=29056 Len=0 TSval=2494743257 TSecr=429
4 44 12.48954133 47.52.128.161 172.31.1.11 TCP 159 blk_13: 19888 > 49158 [PSH, ACK] Seq=1 Ack=113 Win=29056 Len=93 TSval=2494743257 TSe
5 45 12.49039138 172.31.1.11 47.52.128.161 TCP 66 blk_13: 49158 > 19888 [ACK] Seq=113 Ack=94 Win=29200 Len=0 TSval=4294939460 TSecr=24
6 46 28.51171715 47.52.128.161 172.31.1.11 TCP 66 blk_13: 19888 > 49158 [FIN, ACK] Seq=94 Ack=113 Win=29056 Len=0 TSval=2494747263 TSe
7 47 28.55558104 172.31.1.11 47.52.128.161 TCP 66 blk_13: 49158 > 19888 [ACK] Seq=113 Ack=95 Win=29200 Len=0 TSval=4294941067 TSecr=24
8 48 34.33587059 172.31.1.11 39.96.168.250 UDP 60 blk_14: 55351 > 32100 Len=4
9 49 34.33594405 172.31.1.11 47.52.5.251 UDP 60 blk_14: 55351 > 32100 Len=4
0 50 34.33597255 172.31.1.11 47.254.22.224 UDP 60 blk_14: 55351 > 32100 Len=4

```

Figure 4.4: Wyze Cam WYZEC Device Packet Block

4.2.3 Data Process

The data process is the key element of the LogBERT algorithm. It consist of

Step 1. Train and Test Data Preparation

Action 1.1 Parse the grouped data packets using SPELL [31] and DRAIN[55] algorithms. Parsing, in this context, means extracting the packet keys from the packet information. SPELL (see Section 3.3.1) uses the longest common subsequence (LCS) algorithm to find matches across different packet formats with high accuracy and DRAIN (see Section 3.3.1) parses

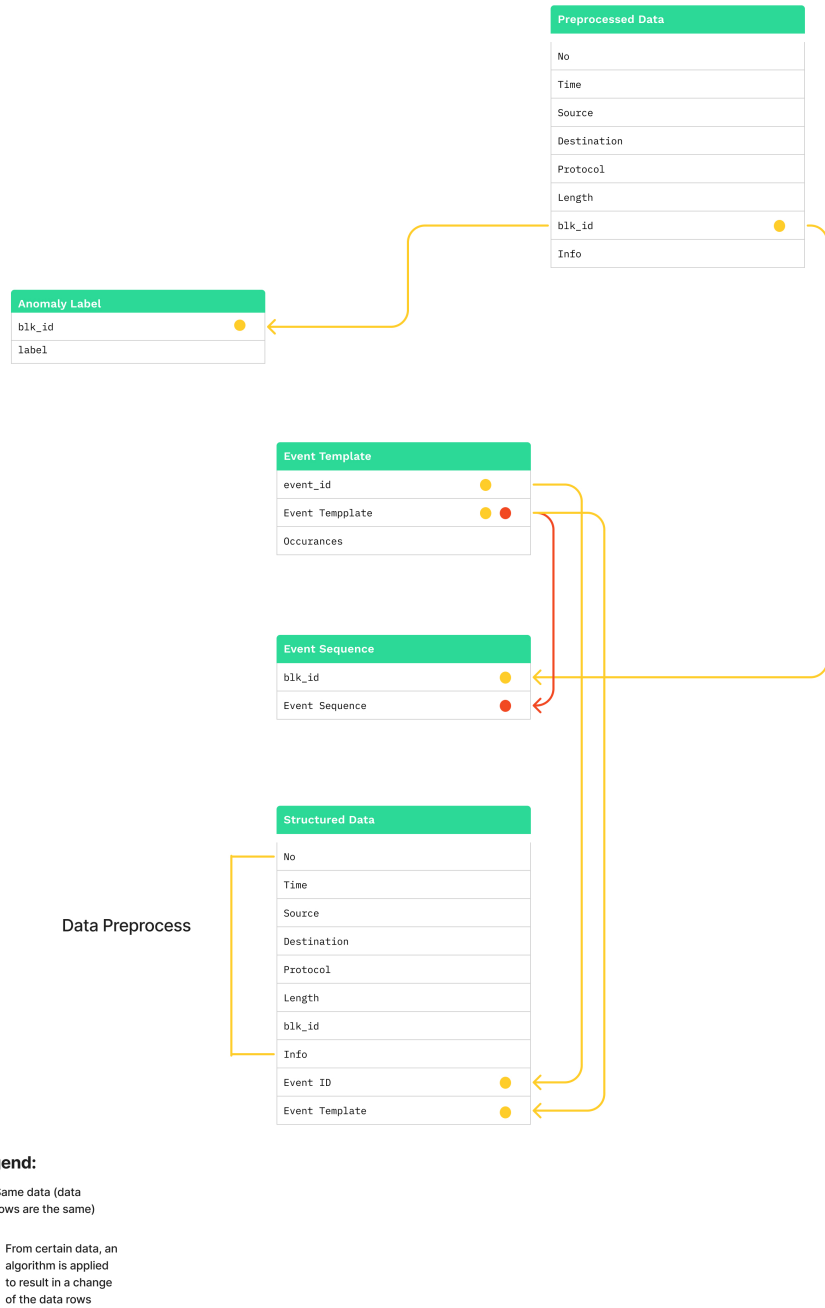


Figure 4.5: The Logic and the Connection between the Data Preprocessing and Data Process

the packet message into packet key. The hyperparameters of SPELL and DRAIN are set based on [31], [55].

Action 1.2 Mask 15% of the numbers in the info column of the preprocessed data with the ('*') symbol using the MLKP algorithm (refer to Section 3.3.3). The percentage of masked numbers is based on the BERT algorithm [29]. The researchers in BERT paper [29] found that increasing the percentage of masking more than 15% reduces the overall performance of BERT. Since LogBERT is a BERT based algorithm, we kept the same parameter for the masked numbers. Masking the part of the information message helps LogBERT in better training.

Action 1.3 Each parsed packet key (in Action 1.2) is assigned an event id, which is used in the vocabulary. The event id maps the packet keys with the packet information in the vocabulary. The vocabulary helps in unmasking the numbers in the packet information and checks the LogBERT prediction(refer to Figure 4.7).

Action 1.4 Using the mapping function attached in A we count the number of occurrences of the same masked packet.

Action 1.5 Create **event template file** (Figure 4.6) which shows a template of the packets after masked with ('*') along with the time of the occurrence of the same template across the log file, and **structured data** (Figure 4.8). The event template file helps LogBERT find a pattern between normal packets.

Action 1.6 Create **event sequence** by converting the packet lines. Figure 4.9 shows the event sequences generated in the DVR camera device. There are several approaches like the packet attribute-based approach [114]and fixed time window technique [64] to achieve the event sequence but, for our data, we used the sliding window technique [70]. Event sequence helps in determining the distance between the normal packets and the abnormal ones using the VHM algorithm (refer to Section 3.3.3).

We followed the same training and testing ratio (70:30) as in the LogBERT HDFS code provided in [47] as the HDFS dataset resembles the most with our IoT dataset.

EventId	EventTemplate	Occurrences
06ea27f7	"blk_<*>: Echo (ping) request <*> <*> ttl=64 <*> <*> <*>	4
cbe37eb2	blk_<*>: DHCP <*> - Transaction ID <*>	22
c9205c65	blk_<*>: Who has <*>? Tell <*>	120
2c0b4a44	blk_<*>: <*> is at <*>	109
3ccc71ef	blk_<*>: Standard query <*> A <*>	5
7.53E+10	blk_<*>: Standard query response <*> A <*> A <*>	4
6f4dc54d	blk_<*>: <*> > <*> [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 <*> TSecr=0 WS=8	9
7639289f	"blk_<*>: <*> > <*> [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1452 SACK_PERM=1 <*> <*>	2
adb6f25a	blk_<*>: 54940 > 30019 [ACK] <*> <*> Win=29200 Len=0 <*> <*>	3
1005173	"blk_<*>: 54940 > 30019 <*> ACK] <*> <*> Win=29200 <*> <*> <*>	2
4f82b6b3	blk_<*>: <*> > <*> [ACK] Seq=1 <*> Win=29056 Len=0 <*> <*>	2
7265de05	"blk_<*>: 30019 > 54940 <*> ACK] <*> <*> Win=29056 <*> <*> <*>	2
4130bbeb	blk_<*>: <*> > 32100 <*>	285
90762bc9	blk_<*>: 32100 > <*> <*>	266
1b6019c0	blk_<*>: 49158 > 19888 [ACK] <*> <*> Win=29200 Len=0 <*> <*>	3

Figure 4.6: Event Template File Screenshot Taken from DVR Camera Data

```

Counter({'1': 645,
        '10': 24,
        '11': 21,
        '12': 15,
        '13': 21,
        '14': 21,
        '15': 12,
        '16': 3,
        '17': 16,
        '18': 12,
        '19': 9,
        '2': 599,
        '20': 6,
        '21': 6,
        '23': 3,
        '24': 9,
        '25': 9,
        '26': 6,
        '27': 6,
        '28': 3,
        '29': 3,
        '3': 264,
        '30': 6,
        '31': 9,
        '32': 3,
        '33': 3,
        '34': 9,
        '35': 9,
        '36': 6,
        '37': 6,
        '38': 3,
        '39': 6,
        '4': 255,
        '40': 6,
        '41': 6,
        '44': 6,
        '45': 3,
        '46': 3,
        ...

```

Figure 4.7: Vocabulary Generated for DVR Camera Data

LineId	Time	IP	Source	Protocol	Content	EventId	EventTemplate
1	0	172.31.1.1	172.31.1.11	ICMP	"blk_1: Echo 06ea27f7"	blk_1: Echo (ping) request <*> ttl=64 <*> <*>	
2	1.00033312	172.31.1.1	172.31.1.11	DHCP	blk_1: DHCP cbe37eb2	blk_1: DHCP - Transaction ID <*>	
3	1.00225619	172.31.1.1	172.31.1.11	DHCP	blk_1: DHCP cbe37eb2	blk_1: DHCP - Transaction ID <*>	
4	4.1991462	00:82:19:30: Broadcast		ARP	blk_4: Who is at c9205c65	blk_4: Who has <*>? Tell <*>	
5	4.19917582	PcsCompu_7	00:82:19:30: Broadcast	ARP	blk_5: 172.31:2c0b4a44	blk_5: <*> is at <*>	
6	4.199928	172.31.1.11	8.8.8.8	DNS	blk_6: Stand: 3ccc71ef	blk_6: Standard query <*> A <*>	
7	4.21942638	8.8.8.8	172.31.1.11	DNS	blk_7: Stand: 753E+10	blk_7: Standard query response <*> A <*> A <*>	
8	4.22102505	172.31.1.11	47.90.2.43	TCP	blk_8: 54940 6f4dc54d	blk_8: <*> <*> [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 <*> TSecr=0 WS=8	
9	4.46370175	47.90.2.43	172.31.1.11	TCP	"blk_9: 3001 7639289f"	"blk_9: <*> <*> [SYN, ACK] Seq=0 Ack=1 Win=28960 Len=0 MSS=1452 SACK_PERM=1 <*> <*> WS=12"	
10	4.46434761	172.31.1.11	47.90.2.43	TCP	blk_10: 5494 adb6f25a	blk_10: 54940 > 30019 [ACK] <*> <*> Win=29200 Len=0 <*> <*>	
11	4.46440701	172.31.1.11	47.90.2.43	TCP	"blk_11: 549 1005173"	"blk_11: 54940 > 30019 [ACK] <*> <*> Win=29200 <*> <*>	
12	4.7070554	47.90.2.43	172.31.1.11	TCP	blk_12: 3001 4f82b6b3	blk_12: <*> <*> [ACK] Seq=1 <*> Win=29056 Len=0 <*> <*>	
13	4.70710253	47.90.2.43	172.31.1.11	TCP	"blk_13: 300 7265de05"	"blk_13: 30019 > 54940 [ACK] <*> <*> Win=29056 <*> <*>	
14	4.70792045	172.31.1.11	47.90.2.43	TCP	blk_14: 5494 adb6f25a	blk_14: 54940 > 30019 [ACK] <*> <*> Win=29200 Len=0 <*> <*>	
15	4.70798884	172.31.1.11	47.90.2.43	TCP	"blk_15: 549 1005173"	"blk_15: 54940 > 30019 [ACK] <*> <*> Win=29200 <*> <*>	
16	4.95120421	47.90.2.43	172.31.1.11	TCP	blk_16: 300 7265de05	blk_16: 30019 > 54940 [ACK] <*> <*> Win=29056 <*> <*>	
17	4.95197261	172.31.1.11	47.90.2.43	TCP	blk_17: 5494 adb6f25a	blk_17: 54940 > 30019 [ACK] <*> <*> Win=29200 Len=0 <*> <*>	
18	5.04015763	PcsCompu_7	00:82:19:30: Broadcast	ARP	blk_18: Who is at c9205c65	blk_18: Who has <*>? Tell <*>	
19	5.04072605	00:82:19:30: Broadcast		ARP	blk_19: 172.31:2c0b4a44	blk_19: <*> is at <*>	
20	11.0498772	172.31.1.11	39.96.168.25	UDP	blk_20: 4271 4130bbeb	blk_20: <*> <*> 32100 <*>	
21	11.0499538	172.31.1.11	47.52.5.251	UDP	blk_21: 4271 4130bbeb	blk_21: <*> <*> 32100 <*>	
22	11.0503582	172.31.1.11	47.254.33.23	UDP	blk_22: 4271 4130bbeb	blk_22: <*> <*> 32100 <*>	
23	11.1249671	47.254.33.23	172.31.1.11	UDP	blk_23: 3210 90762bc9	blk_23: 32100 > <*>	
24	11.2897936	39.96.168.25	172.31.1.11	UDP	blk_24: 3210 90762bc9	blk_24: 32100 > <*>	
25	11.3015559	47.52.5.251	172.31.1.11	UDP	blk_25: 3210 90762bc9	blk_25: 32100 > <*>	
26	12.0452449	172.31.1.11	47.52.128.16	TCP	blk_26: 4915 6f4dc54d	blk_26: <*> <*> [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 <*> TSecr=0 WS=8	
27	12.0473581	172.31.1.11	8.8.8.8	TCP	blk_27: 3917 6f4dc54d	blk_27: <*> <*> [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 <*> TSecr=0 WS=8	

Figure 4.8: Structured Data File Screenshot Taken from DVR Camera Data

BlockId	EventSequence
blk_7	[2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1, 2, 2, 2, 1, 1, 1]
blk_8	[13, 13, 13, 13, 13, 13]
blk_9	[1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2, 1, 1, 2, 2, 2, 2]
blk_10	[1, 1, 45, 2, 2, 1, 1, 45, 2, 2, 1, 1, 45, 2, 2]
blk_11	[1, 1, 1, 1, 1, 1]
blk_12	[32, 24, 32, 24, 32, 24]
blk_13	[47, 53, 32, 53, 32, 47, 53, 32, 53, 32, 47, 53, 32, 53, 32]
blk_14	[1, 1, 1, 2, 2, 1, 1, 1, 2, 2, 1, 1, 1, 2, 2]
blk_15	[3, 4, 3, 4, 3, 4]
blk_16	[1, 1, 2, 1, 1, 2, 1, 1, 2]
blk_17	[1, 2, 1, 2, 1, 2]
blk_18	[3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4]
blk_19	[1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 1, 1]
blk_20	[1, 2, 2, 2, 1, 2, 2, 2, 1, 2, 2, 2]
blk_21	[3, 4, 3, 4, 3, 4]
blk_22	[1, 1, 1, 2, 2, 2, 3, 4, 3, 4, 1, 1, 1, 1, 2, 2, 2, 3, 4, 3, 4, 1, 1, 1, 2, 2, 2, 3, 4, 3, 4]
blk_23	[1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 2]
blk_26	[1, 2, 2, 1, 2, 2, 1, 2, 2]
blk_24	[1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 2, 1, 2, 1, 1, 1, 2, 2]
blk_27	[3, 4, 3, 4, 3, 4]
blk_28	[1, 2, 1, 2, 1, 2]
blk_29	[1, 2, 1, 2, 1, 2]
blk_30	[1, 2, 1, 2, 1, 2]
blk_31	[3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4]
blk_32	[1, 25, 1, 25, 1, 25]
blk_33	[1, 2, 1, 2, 1, 2]
blk_34	[1, 2, 1, 2, 1, 2]
blk_35	[3, 4, 3, 4, 3, 4]

Figure 4.9: Event Sequence File Screenshot Taken From DVR Camera Data

4.2.4 LogBERT

RQ 2. Can LogBERT [47] extend its usage in IoT network traffic data?

As mentioned in Section 3.3.3, the training of the model is done only in normal data in order to find a pattern in normal datasets. Since the LogBERT model is trained on normal logs, it understands the normal log patterns and can accurately predict the masked tokens in the log sequence once the training is complete.

The rest of the Section is divided in three cases. The first case consists of the initial normal data packets in which we have a significant data imbalance between normal and abnormal behaviour. The results of the first case led to re-collect the data packets in the normal state (refer to Table 4.2). The second case consists of creating mini-batches with different weights to determine the best balance by resampling with different ratios of the dataset and determine which ratio that leads to a better LogBERT performance (refer to Section 1.5). The third and last case, we calculate the average F1 for each minibatch of each IoT device. Refer to Table 4.2 for the number of the packets for each IoT device dataset.

$$F1 \text{ score} = TP / (TP + 1/2 \times (FP + FN)), \quad (6)$$

where

- **TP:** True Positive is when the model correctly predicts the positive results.
- **TN:** True Negative is when the model correctly predicts the negative results.
- **FP:** False Positive is when the model predicts incorrectly the positive results.
- **FN:** False Negative is when the mode predicts incorrectly the negative results.

Case I - Predicting LogBERT with Initial Normal Behaviour Data

In the first case, we take each old normal behavior (Refer to Table 4.2), and we randomly mix it with Mirai data, RouterSploit data, and UFONet data separately. The results of each IoT device are presented in Table 4.3. We calculate the macro average, which treats all the classes

and the data sets as equally important. Considering that we have a significant dataset imbalance and very low normal data behavior, the LogBERT algorithm cannot train successfully. In the case of the outdoor camera and home router (1), we had only 70 and 162 normal data behaviour out of 1,514,532 and 1,062,434 total data which was impossible for the algorithm to learn and find a pattern in normal behaviour data. After the results of the first case, we redid the recording of the normal behaviour data for each IoT device (Refer to Table 4.2 NN data).

F1 Score				
IoT	M	R	U	M-AVG
Indoor Smart Camera	18.43	32.54	48.62	33.19
Outdoor Camera	1.24	5.67	-	3.45
Home Router (1)	1.76	10.38	-	6.07
Home Router (2)	23.12	36.72	-	29.92
Smart Home Hub	29.17	30.15	-	29.66
DVR Digital Video Recorder	2.30	14.21	43.45	19.98

Table 4.3: F1 Score with the First Set of Normal Data **Legend** : **M** - First Set of Normal Data + Mirai Data , **R** - First Set of Normal Data + RouterSploit Data, **U** - First Set of Normal Data + UFONet Data, **M-AVG** - Macro Average of F1 score,

Case II - Predicting LogBERT in Mini Batch New Normal Behaviour Data

Initially, we divided the data 50% normal and 50% abnormal. However, the graph in Figure 4.10 shows an unrepresentative dataset. An unrepresentative dataset does not provide enough information to evaluate the model's performance In this case we end up training the model also with abnormal data (the splitting ratio of train and test dataset in LogBERT is 0.3) which makes it harder for LogBERT to train. The F1 score for this case is 26.28 where (Refer to confusion matrix in Figure 4.11).

We redistributed the data in the mini-size batch into 700 normal data and 300 abnormal ones. So the model is completely trained in normal data and it is tested and validated in abnormal data. We saw a significant improvement in the F1-score which showed 55.56% (Refer to confusion matrix Figure 4.13). Also, the graph in Figure 4.12 indicates that the model is capable of further learning and improvement. The loss is lower than the one in Figure 4.10 but it remains at a high rate for a model with a higher f1 score.

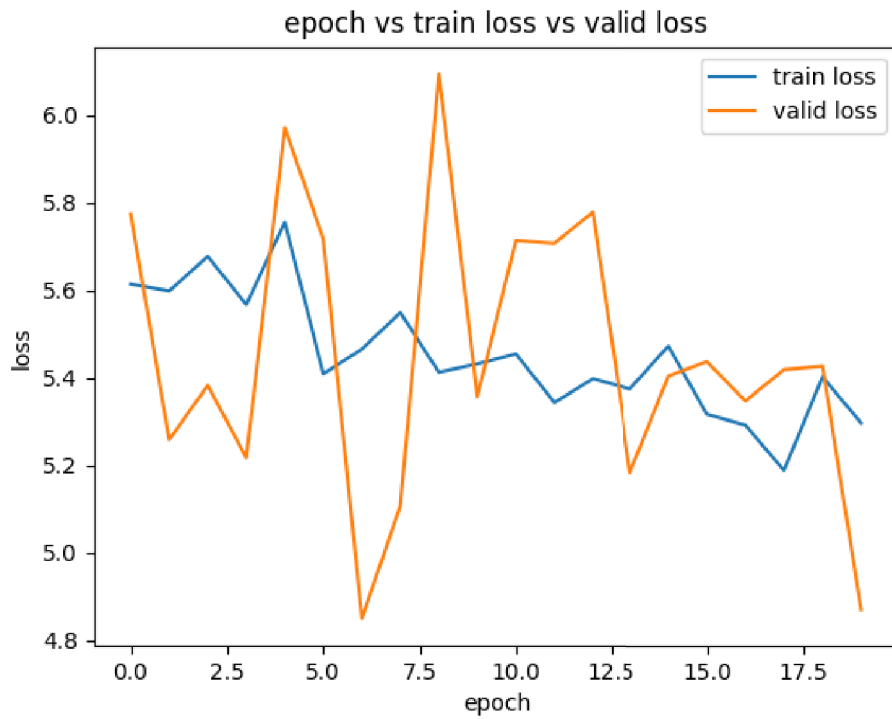


Figure 4.10: DVR minibatch train validation loss graph. (Normal 50:50 Abnormal)

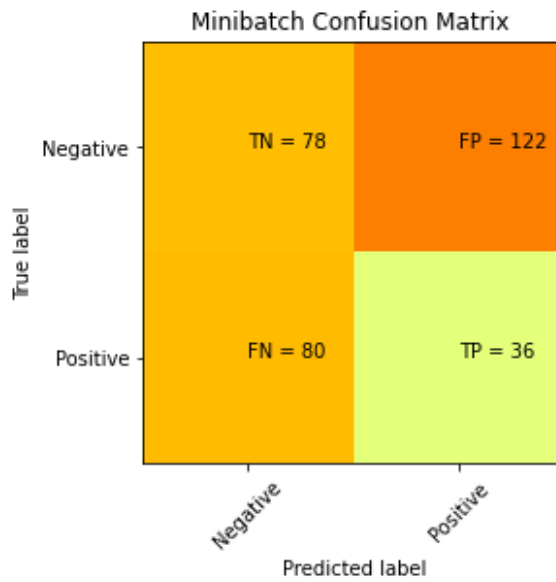


Figure 4.11: DVR Confusion Matrix.

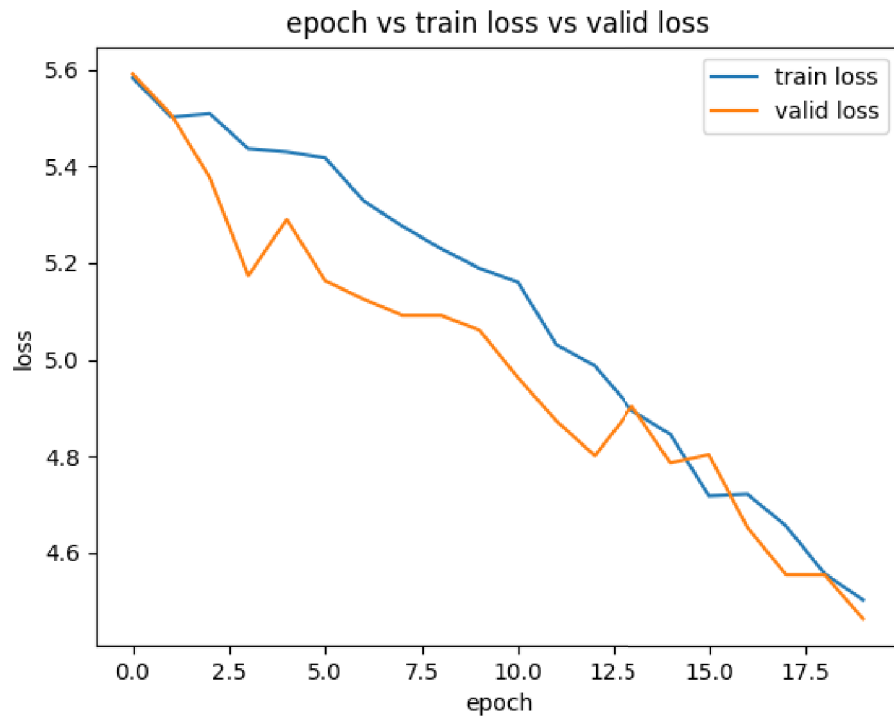


Figure 4.12: DVR Minibatch Train Validation Loss Graph. (Normal 70:30 Abnormal)

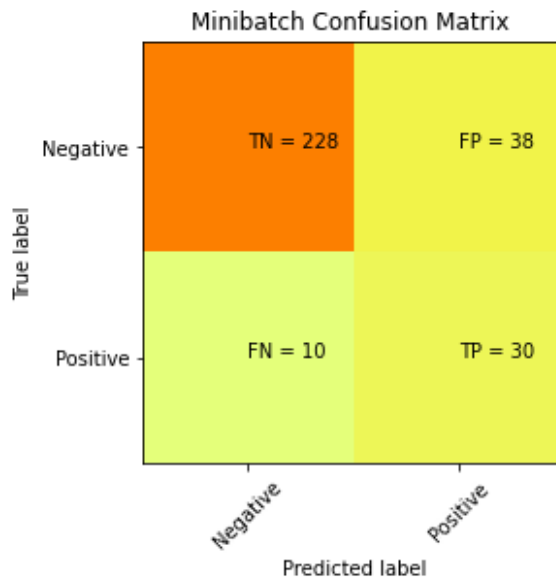


Figure 4.13: DVR Confusion Matrix

Considering the improvement of the F1 score when the data are split 70% normal and 30% abnormal and the fact that the abnormal data in LogBERT HDFS dataset [47] is approximately 6% of the overall dataset, we decided to create another minibatch with 90% normal data and 10 % abnormal ones. The F1-score increased further to 75.5%. The improvement of the LogBERT model is also reflected in the graph in Figure 4.14 where training and validation loss decreases gradually and reach a point of stability. Also, the gap between the validation and train loss is minimal, indicating a good fit learning curve.

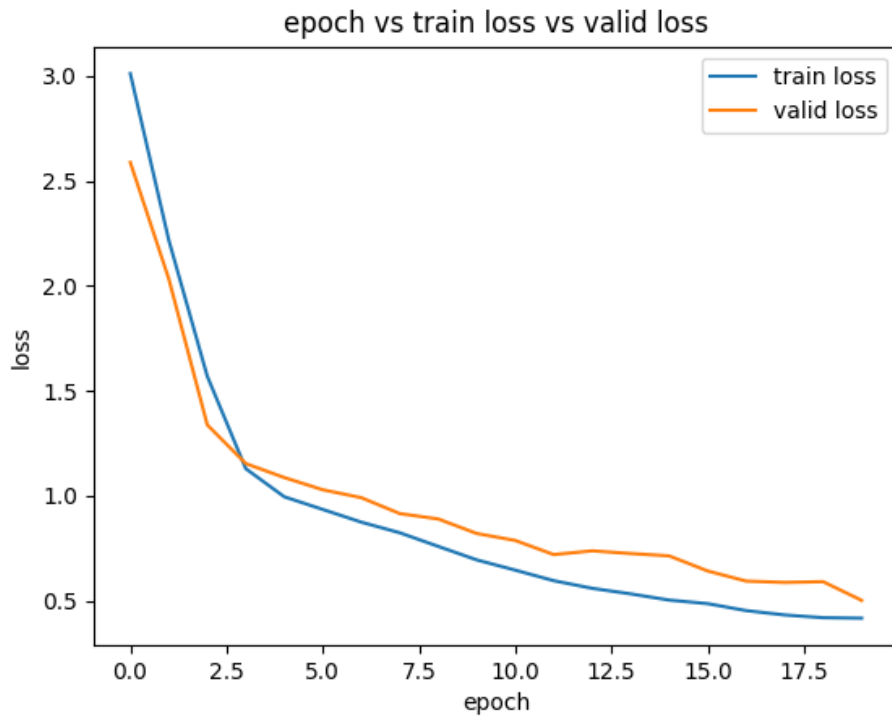


Figure 4.14: DVR Minibatch Train Validation Loss Graph.(Normal 90:10 Abnormal)

Case III - Predicting LogBERT Average of Mini Batch new normal behaviour data

The last mini-batch in case II where the dataset is populated with 90% normal and 10% abnormal data has the highest F1 score compared to the other mini-batches. Table 4.4 represents the 10% of abnormal data. There are several cases in RouterSploit and UFONet that we do

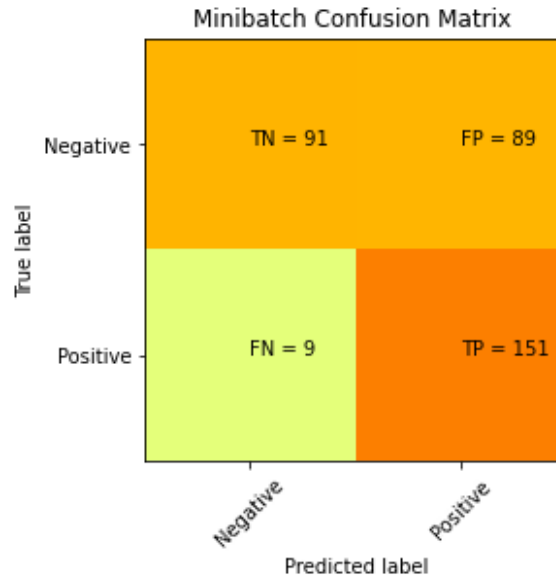


Figure 4.15: DVR Confusion Matrix

not have 10% of the abnormal data registered. In that case, we take the full percentage of the abnormal dataset, e.g., the 10% of the abnormal data is 15,112 but in RouterSploit we have a lower size dataset of 5,421; we keep the full RouterSploit size data of 5,421.

	IoT	ND-2	mini-MD	mini-RD	mini-UD
Indoor Smart Camera	2,203	1,101	245	245	-
Outdoor Camera	136,008	15,112	5,421	-	-
Home Router (1)	89,076	9,897	3,241	-	-
Home Router (2)	88,863	9,873	1,232	-	-
Smart Home Hub	101,261	11,251	55,641	-	-
DVR Digital Video Recorder	89,711	9,967	8,712	571	-

Table 4.4: Number of Abnormal Mini Batch Size **Legend** : Second set of Normal Data - **ND-2**, 10% Mirai Data - **mini-MD**, 10% RouterSploit Data - **mini-RD**, 10% UFONet Data- **mini-UD**

We calculate the macro average of F1 scores mini-batches for Mirai data since the weight in mini batches is equally distributed. For RouterSploit and UFONet we calculate the F1 score based on the dataset available. We encounter a slight increase in the F1 score for UFONet and RouterSploit compared to Mirai datasets (Refer to Table 4.5. This could be due to the similarities of anomalous packets exchange and normal packets exchange in UFONet and RouterSploit malware.

F1 Score			
IoT	M	R	U
Indoor Smart Camera	79.87	83.10	82.45
Outdoor Camera	72.45	77.42	-
Home Router (1)	72.89	73.15	-
Home Router (2)	69.23	61.34	-
Smart Home Hub	74.98	78.76	-
DVR Digital Video Recorder	78.32	81.11	77.89

Table 4.5: F1 Score with Second Set of Normal Data **Legend** : **M** - Second Set of Normal Data + Mirai data , **R** - Second Set of Normal Data + RouterSploit data, **U** - Second Set of Normal Data + UFONet,

4.2.5 Comparison of LogBERT with PCA, SVM, IsolationForest and LogClustering

RQ 3. What is the performance of LogBERT algorithm with real life data in IoT devices in comparison to other machine learning algorithms?

We compare the results of LogBERT with some of most used models which are also mentioned in our literature review; PCA, SVM, IsolationForest and LogClustering. The models are taken from the loglizer.models [81] machine learning library.

- **PCA:** detects anomalous sequences by building a counting matrix based on the frequency of log key sequences.
- **One Class SVM:** is widely used for log anomaly detection based only on normal data observation.
- **IsolationForest:** an unsupervised learning algorithm that uses tree structures to represent features for anomaly detection.
- **LogClustering:** a cluster-based approach detects anomalous log sequences by comparing them to existing clusters. results.

For the comparison of the LogBERT performance with other models, we randomly picked one minibatch from each combination (the split is done 90% normal data 10% abnormal data - refer to Section 4.2.4); normal data plus Mirai data, normal data plus RouterSploit and normal data plus UFONet data. We calculated the F1 score, precision which measures how many

positive class predictions are actually in the positive class, and recall which measures how many of the positive examples in a dataset were used in making positive predictions. The results are presented in Table.4.6, 4.7 and 4.8.

Indoor Smart Camera				Outdoor Smart Camera		
Model	F1	Recall	Precision	F1	Recall	Precision
LogBERT	80.21	83.12	77.5	72.72	65.12	82.34
PCA	16.32	100.00	8.889	21.65	99.54	12.15
OCSVM	2.06	5.26	1.28	1.97	14.15	1.06
Isolation Forest	56.65	67.34	48.9	29.66	17.43	99.50
LogClustering	53.01	36.4	97.54	73.82	61.01	93.46

Table 4.6: Indoor and Outdoor F1, Recall and Precision

Home Router (1)				Home Router (2)		
Model	F1	Recall	Precision	F1	Recall	Precision
LogBERT	72.64	59.01	94.46	69.23	68.17	70.34
PCA	28.73	22.23	40.64	17.13	13.24	24.29
OCSVM	3.08	2.34	4.51	3.27	2.3	5.7
Isolation Forest	64.75	62.45	67.23	55.95	57.32	54.65
LogClustering	74.76	72.45	77.23	64.01	61.76	66.45

Table 4.7: Home Router (1) and (2) Smart Camera F1, Recall and Precision

Smart Home Hub				DVR		
Model	F1	Recall	Precision	F1	Recall	Precision
LogBERT	75.52	70.37	81.5	80.11	75.37	85.5
PCA	19.10	23.45	16.12	17.44	22.54	14.23
OCSVM	5.43	6.72	4.56	4.41	5.89	3.53
Isolation Forest	53.59	65.76	45.23	47.99	48.43	47.56
LogClustering	76.08	68.45	85.64	69.64	60.76	81.56

Table 4.8: Smart Hub and DVR Camera F1, Recall and Precision

4.2.6 Comparison of LogBERT Performance on Computer Logs and IoT Packets

RQ 2. Can LogBERT [47] extend its usage in IoT network traffic data?

LogBERT algorithm was initially built to find anomalies in computer logs. The authors presented the results in this paper [47]. In this section we compare the results presented in this paper [47] with the results we achieved in this thesis in Section. 4.2.5. In Figure 4.16 we see that

LogBERT performs better in computer logs data with the highest F-1 score of 96.64. The best performance of LogBERT in IoT device was reach at Indoor Smart Camera with a F-1 score of 80.21. However the results vary based on the dataset and the balance of the dataset the LogBERT was trained and tested on. LogBERT was built with an intention of log anomaly detection, while in this thesis we used IoT data packets. Overall, LogBERT in both computer and IoT devices has the best performance comapred to other models like; PCA, OCSVM, IsolationForest, LogClustering.

Method	HDFS			BGL			Thunderbird		
	Precision	Recall	F-1 score	Precision	Recall	F-1 score	Precision	Recall	F-1 score
PCA	5.89	100.00	11.12	9.07	98.23	16.61	37.35	100.00	54.39
iForest	53.60	69.41	60.49	99.70	18.11	30.65	34.45	1.68	3.20
OCSVM	2.54	100.00	4.95	1.06	12.24	1.96	18.89	39.11	25.48
LogCluster	99.26	37.08	53.99	95.46	64.01	76.63	98.28	42.78	59.61
DeepLog	88.44	69.49	77.34	89.74	82.78	86.12	87.34	99.61	93.08
LogAnomaly	94.15	40.47	56.19	73.12	76.09	74.08	86.72	99.63	92.73
LogBERT	87.02	78.10	82.32	89.40	92.32	90.83	96.75	96.52	96.64

Figure 4.16: LogBERT Results on Computer Logs Presented in "LogBERT: Log Anomaly Detection via BERT" [47]

4.3 Energy Consumption Preliminary Work Results

RQ 4. Can energy measures give an indication of anomaly detection?

In this section, we present the results and the observations of our preliminary work on the energy consumption of IoT devices in normal and abnormal states. To avoid repetition we explain the observed results in this section focusing on the ANNKE DVR 8CH device. The observations and results for the rest of the IoT devices can be found in Appendix B.

As mentioned in Chapter 3, we initially collected the electric current data using an Arduino Uno microcontroller board and we used the Ohm's Law [68] to calculate the power. Energy is the quantifiable attribute required for a physical system to perform work and it is calculated power integrated over time. The four graphs in Figure 4.17 we present the power in the blue dots and the energy consumption in the red area (watts x time). The data are presented over a 30 min time. The average power of ANNKE DVR 8CH device in normal state is 0.6 watts. However, we observe an increase by three times when the device is exploited via Mirai Botnet and RouterSploit. A higher power is recorded during UFONet exploitation, where the device averages 1.92 watts.

Based on the four graphs Figure 4.17, we observe an increase in energy consumption three as much as the normal state, in the abnormal state. The energy consumption for ANNKE DVR 8CH is the highest during the UFONet malware. This is explained due to the nature of the malware which launches the numerous malicious packets through third party websites which act as botnets. The tendency for an increase in energy consumption on ANNKE DVR 8CH device can be an interesting topic to research in future work.

Power Formula based on Ohm's Law [68] is as follows:

$$P = I \times V, \tag{7}$$

where: **P** represents Power (Watts), **I** represents Current (ampere), and **V** represents the Voltage (volts).

$$E = P \times t, \tag{8}$$

where: E represents Energy (Watts second), P represents Power (Watts), and t represents the time (Seconds).

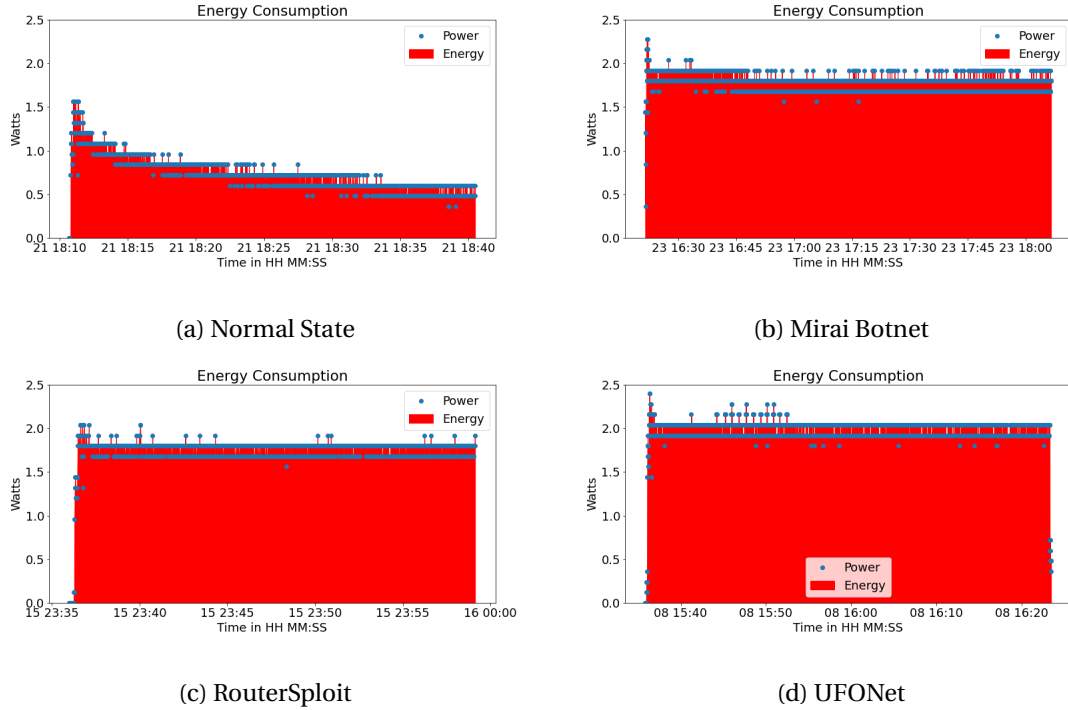


Figure 4.17: ANNKE DVR 8CH Energy Consumption Graph

Our preliminary work results show that there is a possibility for anomaly detection based on energy consumption of IoT devices under attack. The results of all IoT devices are presented in Appendix B. Compared to the normal state, all six IoT devices have a slight increase in energy consumption when they are compromised. These results suggest that energy consumption could be monitored as one metric for detecting suspicious activity on IoT devices. However, more work is needed to determine whether energy consumption is a reliable indicator of anomalies and whether different types of attacks result in different energy consumption patterns. energy consumption monitoring as a security metric warrant further exploration.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

The increase in the number of IoT devices has also increased the number of cyber attacks on such devices. Although preventing all the cyberattacks from IoT devices might be an ideal solution, it is, however, not a realistic approach considering the vast increase of IoT devices and the complex architecture of cyberattacks. Several approaches to cybersecurity tools in IoT devices have been proposed over the years, some of which are discussed in Chapter 2. Among the most interesting approaches proposed in recent years are those that integrate machine learning techniques into cyberattack prevention; e.g. PCA, IsolationForest, and SVM. As part of our thesis, we incorporate an unsupervised deep learning model based on BERT [29], LogBERT [47], to detect anomalies in IoT devices. This thesis shows the applicability of LogBERT also to IoT devices. Using six IoT datasets that we generated specifically for this research, we determine the best dataset balance and compare LogBERT's performance with existing state-of-the-art approaches in terms of precision, recall, and F1 score. Lastly, as part of this thesis work, we investigate the energy consumption of IoT devices during attacks. Our main contributions in this thesis are three-fold; 1) real IoT data generation, 2) anomaly detection using LogBERT algorithm, and 3) energy consumption analysis when IoT devices are exploited with different malware.

The implementation of such powerful ML and DL algorithms, like LogBERT, requires large data collection to gain high-performance results. Most of the ML and DL approaches mentioned

in Chapter 2 present the results of the proposed solutions based on synthetic data. Understanding the importance of the real-life data to give a bona fide solution, **1**) we generated our dataset. We exploited six IoT devices with three different malware, overcoming so one of the challenges while generating the real-life data mentioned in Section 1.4. A number of malware were researched and tested for this experiment, in which the three most adequate for the six IoT devices were Mirai [99], RouterSploit [112] and UFONet [36]. We collected two different dataset groups for each IoT device; the network traffic dataset and the energy consumption dataset. We use our real-life datasets in LogBERT to have as close to real-life results as possible. Understanding the importance of the data in the research field, the real-life IoT data generated for this thesis are submitted for public release in *Data in Brief - Journals Elsevier*. We hope this dataset will inspire further research in IoT cybersecurity approaches.

The main contribution in this thesis falls in **2**) anomaly detection using the LogBERT algorithm in IoT devices. The original use of LogBERT in computer logs presented in this paper [47] has prompted us to question its use in IoT devices throughout this study. In particular, we demonstrate that the usage of LogBERT [47] can be extended in IoT devices, despite the unstructured IoT network traffic data packets. In order to overcome the challenges identified in Section 1.4, we utilize LogParser[54] to interpret the unstructured IoT data. We present the best balance among datasets which leads to a better performance of LogBERT [47] in Chapter 4. According to the obtained results, LogBERT achieved superior performance supported by a higher recall, F-1 score, and precision. On numerous examples, LogBERT yields more accurate solutions than existing techniques. The usage of the LogBERT model in IoT devices has been found to produce better results compared with comparable state-of-the-art methods. The comparison of LogBERT performance with other machine learning algorithms like PCA, LogCluster, One SVM, and Isolation Forest concludes that the unsupervised LogBERT algorithm can be used to detect anomalies better among IoT devices. Due to its unsupervised nature and reliance on BERT, LogBERT is a versatile and easy-to-use algorithm.

Lastly, we present a work initiation on energy consumption when IoT devices are under attack. The energy consumption of an IoT device can be a useful indicator of whether or not the device has been compromised. To test this hypothesis, we carried out an experiment in which

we placed six IoT devices in a virtual environment and then exploited them with three kinds of malware. The results of the experiment showed that all six IoT devices had a slight increase in energy consumption when they were compromised. This suggests that energy consumption is a useful metric for detecting anomalies in IoT networks. However, further research is needed to confirm this result. According to the results presented in this thesis, a further investigation of energy consumption may be able to identify the compromised IoT device.

5.2 Limitation

The main limitation of this thesis is the number of IoT devices the LogBERT algorithm was tested on. IoT devices that are examined in this thesis represent only a fraction of the total market of IoT devices. There are currently ten billion IoT devices with different architecture and parameters worldwide [8]. Hence the results presented in this thesis are limited to the six chosen IoT devices and can not be used to draw a general conclusion about all available IoT devices worldwide.

Another limitation of this work is the quality of the data or lack thereof. To ensure reliable results, the data must meet certain criteria. The verification of these criteria can be a time-consuming process, but it is essential to the credibility of the work. In this work, we assumed that the data was of sufficient quality. In future work, we can utilize one of the quality evaluation metrics mentioned in Section 5.3. The data evaluation can help improve future work. This could involve plans for upcoming projects or improving current processes. Additionally, the data evaluation can help identify areas that need attention.

5.3 Future Work

The LogBERT algorithm has shown promising results in terms of accuracy for IoT data packet anomaly detection. However, the algorithm currently only takes the message content of a packet as an input feature. It may be interesting to explore other features of the data packets, such as the timestamp, IP address, source, and length of the packets. The improvement of LogBERT via the implementation of the timestamp feature can be made in the pre-training phase.

Similar to the Time Series BERT presented in this paper [26], LogBERT can incorporate the timestamp feature into two-time series fragments and determine the relationship between those fragments. Although, further investigation into the code implementation is needed to achieve the timestamp feature in the LogBERT algorithm.

One way to further understand LogBERT performance is to consider data quality. Data quality can impact many aspects of performance, so it is important to ensure that the data used for training and testing is as high-quality as possible. Data quality can help us get a better picture of how LogBERT is performing and what potential factors might be influencing its results. Multi-layered quality evaluation metrics are a way to check data quality. By looking at various factors, such as accuracy and precision, we can get a holistic view of the data. This technique can help us identify problems with the data. However, other data quality approaches such as overlap or consensus methods and auditing methods [115] can be a good use for further data investigation.

Another approach which can help increase the accuracy of anomaly detection when IoT devices are under exploitation is, incorporating energy consumption as an additional step toward anomaly detection. Furthermore, it would be interesting to deploy LogBERT in a real-time aggregation system to test its performance in a more realistic setting.

Appendix A

The code below is the mapping algorithm to check the number of Occurrences of the same masked logs. This code was initially published in [47] and it is modified for the cctv camera experiment.

Listing A.1: Checking occurrences of the same masked logs.

```
def mapping():
    log_temp = pd.read_csv(log_templates_file)
    log_temp.sort_values(by = ["Occurrences"], ascending=False,
                           inplace=True)

    log_temp_dict = {event: idx+1 for idx , event in

enumerate(list(log_temp["EventId"])) }
    print(log_temp_dict)

with open (output_dir + "camera_log_templates.json", "w")
as f:

        json.dump(log_temp_dict, f)
```

Appendix B

The Energy Consumption for Indoor camera - Wyze Cam WYZEC:

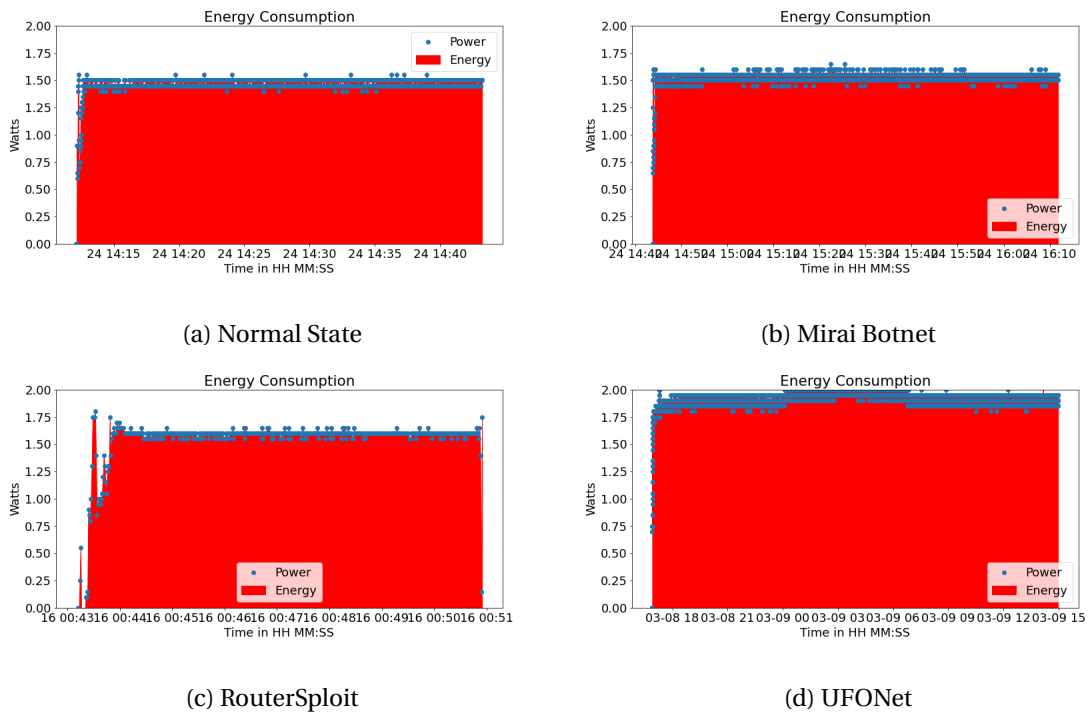


Figure B.1: Wyze Cam WYZEC Energy Consumption Graphs

During normal operation, Wyze Cam WYZEC devices consume 1.45 watts of power. However, we do not observe a significant increase in power for Mirai Botnet and RouterSploit. While exploitation of UFONet, the device averages a power of 2.0 watts.

Based on the four graphs Figure B.1, we observe a slight increase in energy consumption for Mirai Botnet and RouterSploit in comparison with the normal state. The energy consumption for ANNKE DVR 8CH is the highest during the UFONet malware. This is explained due to

the nature of the malware which launches the numerous malicious packets through third party websites which act as botnets.

The Energy Consumption for Outdoor camera - HOSAFE Outdoor Wifi Camera:

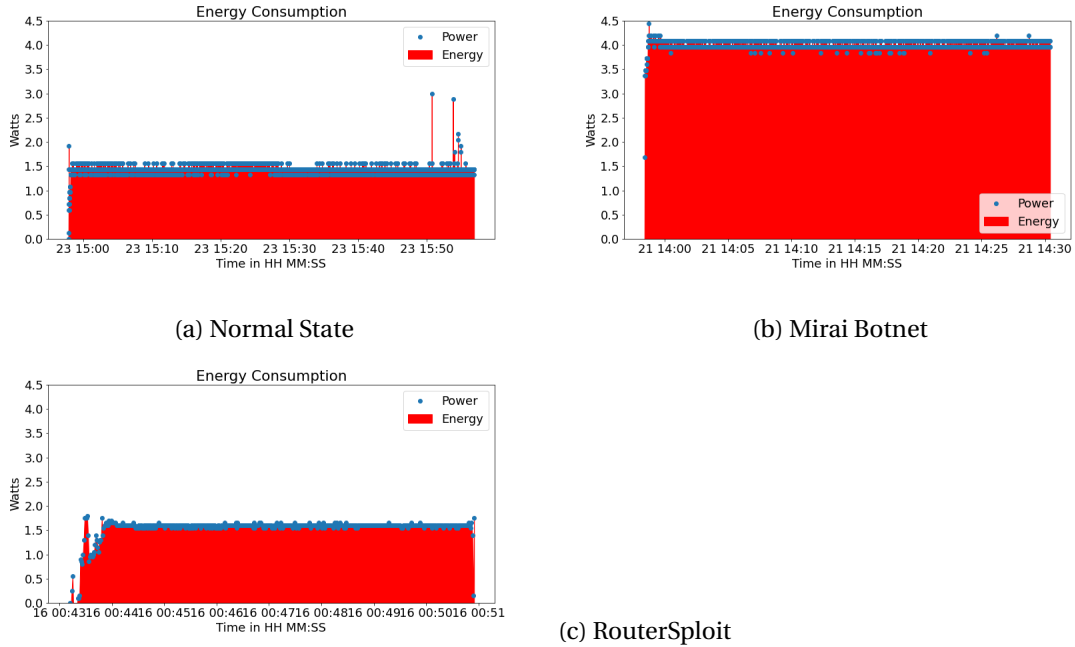


Figure B.2: HOSAFE Outdoor Wifi Camera Energy Consumption Graphs

In terms of power consumption, the HOSAFE Outdoor Wifi Camera device consume an average of 1.45 watts in their normal state. However, during exploitation via Mirai Botnet we observed a rapid increase in power, averaging 4.0 watts. This is likely due to the increased network activity associated with these attacks. Meanwhile, the RouterSploit malware resulted in the slight increase power consumption, averaging 1.55 watts.

Based on the four graphs in Figure B.2, we observe the highest energy consumption in Mirai Botnet. However, the energy consumption increases slightly in comparison with the normal state.

The Energy Consumption for Home router (1) - TP-Link ACS1750:

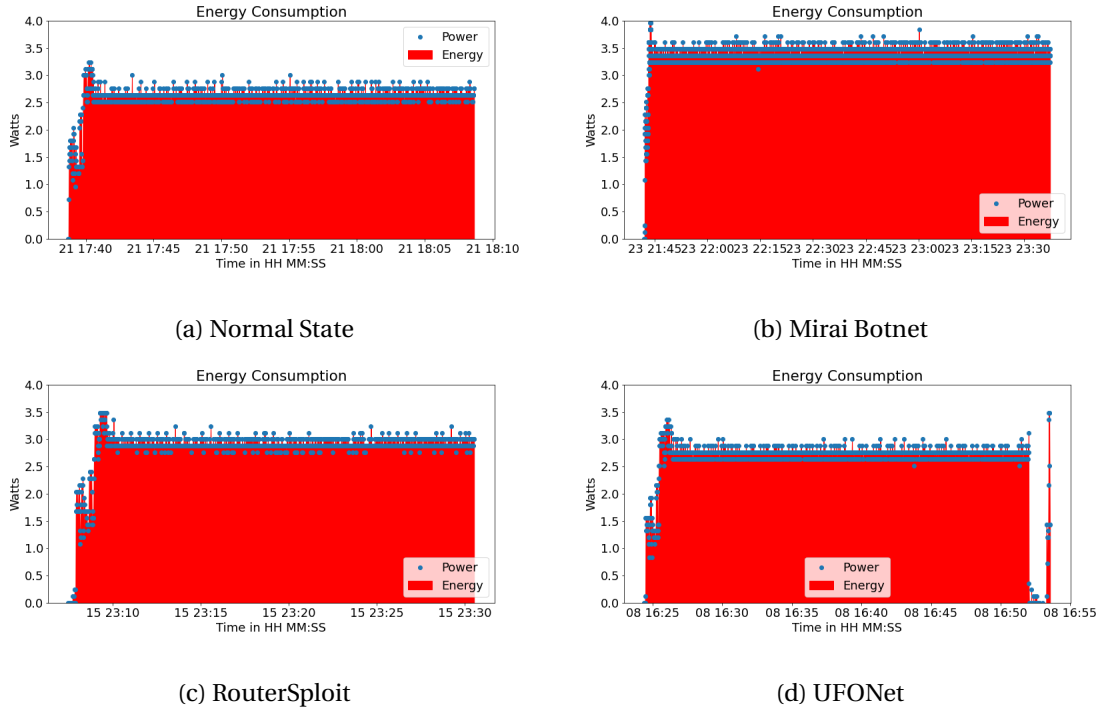


Figure B.3: TP-Link ACS1750 Energy Consumption Graphs

TP-Link ACS1750 Home Router consumes an average of 2.5 watts in its normal state. In abnormal state we observe an increase; under Mirai attack the power increases up to an average of 3.5 watts, under RouterSploit and UFONet and increase of 2.0 watts. The increase of the power is in direct proportion with the energy consumption. We observe an increase of the red area in graph B, C, and D in Figure B.3.

The Energy Consumption for Home router (2) - D-Link AC1200:

An interesting observation happens in D-Link AC1200 router. The power and energy consumption have a slight decrease in comparison to the normal state. However we observe an abnormal behaviour pattern of small rapid increase and decrease in power. The highest power consumption point is reached in RouterSploit with 2.5 watts. This case might need a further investigation in the future.

The Energy Consumption for Phillips Hue Smart Hub:

In terms of power consumption, the Phillips Hue Smart Hub device consume an average

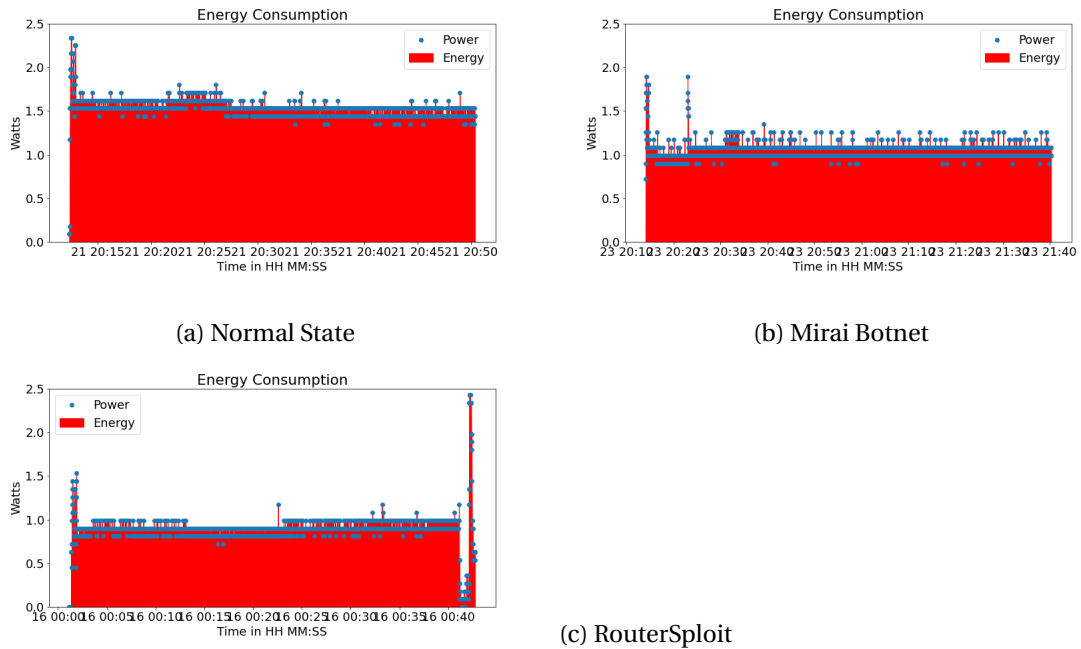
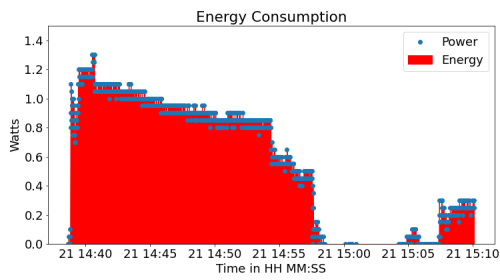


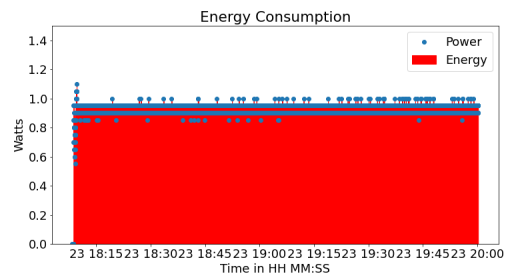
Figure B.4: D-Link AC1200 Energy Consumption Graphs

of 0.8 watts in their normal state. The graph A in Figure B.5 has an rapid abruption in normal state. This can be caused by an power abruption from the outlet the Smart HUB is plugged into. However, during exploitation via Mirai Botnet and RouterSploit we observed a slight increase in power, averaging 1.0 and 1.1 watts.

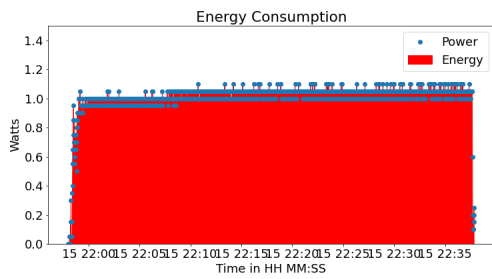
Based on the four graphs in Figure B.5, we observe a slight increase in energy consumption in Mirai Botnet and RouterSploit in comparison with the normal state. The change in the energy consumption between normal and abnormal state is not as rapid as in other IoT devices.



(a) Normal State



(b) Mirai Botnet



(c) RouterSploit

Figure B.5: Phillips Hue Smart Hub Energy Consumption Graphs

Bibliography

- [1] H. Abdi and L. J. Williams. Principal component analysis. Wiley Interdisciplinary Reviews: Computational Statistics, 2(4):433–459, 2010.
- [2] M. Abomhara and M. G. Køien. Cyber security and the internet of things: vulnerabilities, threats, intruders and attacks. Journal of Cyber Security and Mobility, 4(23):65–88, 2015.
- [3] F. A. Acheampong, H. Nunoo-Mensah, and W. Chen. Transformer models for text-based emotion detection: a review of BERT-based approaches. Artificial Intelligence Review, 54(8):5789–5829, 2021.
- [4] A. F. Agarap. Deep Learning using Rectified Linear Units (RELU). CoRR, 2:1–7, 2018.
- [5] R. Ahmad and I. Alsmadi. Machine learning approaches to IoT security: A systematic literature review. Internet of Things, 14:1–42, 2021.
- [6] M. Al-Garadi, A. Mohamed, A. Al-Ali, X. Du, I. Ali, and M. Guizani. A Survey of Machine and Deep Learning Methods for Internet of Things (IoT) Security. IEEE Communications Surveys & Tutorials, 22(3):1646–1685, 2020.
- [7] alanackart. DDOSIM, 2019. <https://github.com/alanackart/DDOSIM> Accessed on 23.05.2022.
- [8] I. Alrashdi, A. Alqazzaz, E. Aloufi, R. Alharthi, M. Zohdy, and H. Ming. AD-IoT: Anomaly Detection of IoT Cyberattacks in Smart City Using Machine Learning. In 2019 IEEE 9th Annual Computing and Communication Workshop and Conference (CCWC), pages 0305–0310. IEEE, 2019.

- [9] G. Alva, Y. Lin, and G. Fang. An overview of thermal energy storage systems. Energy, 144:341–378, 2018.
- [10] M. Antonakakis, T. April, M. Bailey, M. Bernhard, E. Bursztein, J. Cochran, Z. Durumeric, J. A. Halderman, L. Invernizzi, M. Kallitsis, et al. Understanding the Mirai botnet. In 26th USENIX security symposium (USENIX Security 17), pages 1093–1110, 2017.
- [11] J.-P. Arcangeli, A. Bouzeghoub, V. Camps, M.-F. Canut, S. Chabridon, D. Conan, T. Desprats, R. Laborde, E. Lavinal, and S. Leriche. INCOME – Multi-scale Context Management for the Internet of Things. In Ambient Intelligence, pages 338–347, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.
- [12] Arduino Team. Arduino Uno, 2010. <https://www.arduino.cc> Accessed on: 2022-03-15.
- [13] K. Ashton. That 'Internet of Things' thing. RFID Journal, 22(7):97–114, 2009.
- [14] ashxjain. Wiseg3ck0-AIM-DDOS,, 2014. <https://github.com/ashxjain/DDoS-Attack> Accessed on 26.03.2021.
- [15] ashxjain. Hacker News Way Back Machine, 2020. <https://hnwaybackmachine.aryan.app/2020/02/01/> Accessed on 14.03.2021.
- [16] A. Bahga and V. K. Madiseti. Blockchain Platform for Industrial Internet of Things. Journal of Software Engineering and Applications, 9(10):533–546, 2016.
- [17] A. Budzyn. The world's most valuable resource is no longer oil, but data. The Economist <https://www.economist.com/leaders/2017/05/06/the-worlds-most-valuable-res> May 6th, 2017.
- [18] C. Cervantes, D. Poplade, M. Nogueira, and A. Santos. Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. In 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM), pages 606–611. IEEE, 2015.

- [19] R. Chalapathy and S. Chawla. Deep learning for anomaly detection: A survey. CoRR, 2, 2019.
- [20] H. Chen. Applications of cyber-physical system: A literature review. Journal of Industrial Integration and Management, 2(03):1750012, 2017.
- [21] Y.-C. Chen, T. Giesecking, D. Campbell, V. Mooney, and S. Grijalva. A hybrid attack model for cyber-physical security assessment in electricity grid. In IEEE Texas Power and Energy Conference (TPEC), pages 1–6. IEEE, 2019.
- [22] A. Chesla. Multi Dimensional Attack Decision System and Method thereof, 2013. US Patent 8,566,936.
- [23] K. Christidis and M. Devetsikiotis. Blockchains and Smart Contracts for the Internet of Things. IEEE Access, 4(11):2292–2303, 2016.
- [24] T. Czauski, J. White, Y. Sun, H. Turner, and S. Eade. NERD —middleware for IoT human machine interfaces. Annals of Telecommunications, 71(3):109–119, 2016.
- [25] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. CoRR, 3:1–20, 2019.
- [26] W. Dang, B. Zhou, L. Wei, W. Zhang, Z. Yang, and S. Hu. Ts-bert: Time series anomaly detection via pre-training model bert. In International Conference on Computational Science, pages 209–223. Springer, 2021.
- [27] M. DeCesare. ForeScout IoT Enterprise Risk Report. Technical report, ForeScout, 2016. <https://www.forescout.com/wp-content/uploads/2016/10/iot-enterprise-risk-report.pdf> Accessed on: 2022-05-31.
- [28] A. M. Derrington, H. A. Allen, and L. S. Delicato. Visual mechanisms of motion analysis and motion perception. Annual review of psychology, 55(1):181–205, 2004.
- [29] J. Devlin, M. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. CoRR, 2:1–5, 2018.

- [30] R. Doshi, N. Apthorpe, and N. Feamster. Machine learning DDoS detection for consumer internet of things devices. In 2018 IEEE Security and Privacy Workshops (SPW), pages 29–35. IEEE, 2018.
- [31] M. Du and F. Li. Spell: Online Streaming Parsing of Large Unstructured System Logs. IEEE Transactions on Knowledge and Data Engineering, 31(11):2213–2227, 2019.
- [32] W. Du and Z. Zhan. Building decision tree classifier on private data. Electrical Engineering and Computer Science, 8, 2002.
- [33] P. DuBois. MySQL. Pearson Education, London, United Kingdom, 2008.
- [34] Z. Durumeric, E. Wustrow, and J. A. Halderman. {ZMap}: Fast internet-wide scanning and its security applications. In 22nd USENIX Security Symposium (USENIX Security 13), pages 605–620, 2013.
- [35] M. Elkhodr, S. Shahrestani, and H. Cheung. A Middleware for the Internet of Things. CoRR, 2016.
- [36] epsilon. UFONet, 2020. <https://ufonet.03c8.net/> Accessed on 13.01.2022.
- [37] M. Feily, A. Shahrestani, and S. Ramadass. A Survey of Botnet and Botnet Detection. In Third International Conference on Emerging Security Information, Systems and Technologies, pages 268–273, 2009.
- [38] N. Fenton and J. Bieman. Software metrics: a rigorous and practical approach. CRC press, 2014.
- [39] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. Hypertext transfer protocol–HTTP/1.1, 1999.
- [40] C. Frank, C. Nance, S. Jarocki, and W. E. Pauli. Protecting IoT from Mirai botnets; IoT device hardening. Journal of Information Systems Applied Research, 11(2):33, 2018.

- [41] A. García Vázquez, P. Soria-Rodríguez, P. Bisson, D. Gidoin, S. Trabelsi, and G. Serme. FI-WARE security: Future Internet Security Core. In Towards a Service-Based Internet, pages 144–152, Berlin, Heidelberg, 2011. Springer, Springer Berlin Heidelberg.
- [42] Gerald Combs. Wireshark, 1997. <https://www.wireshark.org> Accessed on: 2022-03-15.
- [43] A. Glikson. FI-WARE: Core Platform for Future Internet Applications. In Proceedings of the 4th Annual International Conference on Systems and Storage, 2011.
- [44] M. Goyal, I. Sahoo, and G. Geethakumari. HTTP Botnet Detection in IoT Devices using Network Traffic Analysis. In 2019 International Conference on Recent Advances in Energy-efficient Computing and Communication (ICRAECC), pages 1–6, 2019.
- [45] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. IEEE Transactions on Neural Networks and Learning Systems, 28(10):2222–2232, 2016.
- [46] H. Guntur, J. Ishii, and A. Satoh. Side-channel attack user reference architecture board SAKURA-G. In IEEE 3rd Global Conference on Consumer Electronics (GCCE), pages 271–274. IEEE, 2014.
- [47] H. Guo, S. Yuan, and X. Wu. Logbert: Log anomaly detection via BERT. In 2021 International Joint Conference on Neural Networks (IJCNN), pages 1–8. IEEE, 2021.
- [48] H.-S. Ham, H.-H. Kim, M.-S. Kim, and M.-J. Choi. Linear SVM-based android malware detection for reliable IoT services. Journal of Applied Mathematics, 2014, 2014.
- [49] P. Hanks. Collins dictionary of the English language. London: Collins, 1979.
- [50] W. H. Hassan et al. Current research on Internet of Things (IoT) security: A survey. Computer networks, 148:283–294, 2019.
- [51] Z. Hassan, R. Odarchenko, S. Gnatyuk, A. Zaman, and M. Shah. Detection of distributed denial of service attacks using snort rules in cloud computing & remote control systems.

- In 2018 IEEE 5th International Conference on Methods and Systems of Navigation and Motion Control (MSNMC), pages 283–288. IEEE, 2018.
- [52] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. IEEE, 7(22):82721–82743, 2019.
- [53] A. O. Hathaway, R. Crootof, P. Levitz, and H. Nix. The law of cyber-attack. Calif. L. Rev., 100:817, 2012.
- [54] P. He, J. Zhu, S. He, J. Li, and M. R. Lyu. Towards automated log parsing for large-scale log data analysis. IEEE Transactions on Dependable and Secure Computing, 15(6):931–944, 2017.
- [55] P. He, J. Zhu, Z. Zheng, and M. R. Lyu. Drain: An online log parsing approach with fixed depth tree. In 2017 IEEE international conference on web services (ICWS), pages 33–40. IEEE, 2017.
- [56] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. A Field Guide to Dynamical Recurrent Neural Networks, 2003.
- [57] J.-C. Hsueh and V. H.-C. Chen. An ultra-low voltage chaos-based true random number generator for IoT applications. Microelectronics Journal, 87:55–64, 2019.
- [58] W. Hu, Y. Liao, and V. R. Vemuri. Robust Support Vector Machines for Anomaly Detection in Computer Security. In ICMLA, pages 168–174, 2003.
- [59] T. K. Hui, R. S. Sherratt, and D. D. Sánchez. Major requirements for building Smart Homes in Smart Cities based on Internet of Things technologies. Future Generation Computer Systems, 76:358–369, 2017.
- [60] F. Hussain, S. G. Abbas, M. Husnain, U. U. Fayyaz, F. Shahzad, and G. A. Shah. IoT DoS and DDoS Attack Detection using ResNet. In 2020 IEEE 23rd International Multitopic Conference (INMIC), pages 1–6, 2020.

- [61] R.-H. Hwang, M.-C. Peng, V.-L. Nguyen, and Y.-L. Chang. An LSTM-based Deep Learning Approach for Classifying Malicious Traffic at the Packet Level. Applied Sciences, 9(16):3414, 2019.
- [62] C. Ioannou and V. Vassiliou. Classifying Security Attacks in IoT Networks Using Supervised Learning. In 2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS), pages 652–658, 2019.
- [63] K. Kalkan and S. Zeadally. Securing Internet of Things with Software Defined Networking. IEEE Communications Magazine, 56(9):186–192, 2018.
- [64] H. Keller and D. Massart. Peak purity control in liquid chromatography with photodiode-array detection by a fixed size moving window evolving factor analysis. Analytica chimica acta, 246(2):379–390, 1991.
- [65] M. A. Khan, M. Karim, Y. Kim, et al. A scalable and hybrid intrusion detection system based on the convolutional-LSTM network. Symmetry, 11(4):583, 2019.
- [66] M. A. Khan and K. Salah. IoT security: Review, blockchain solutions, and open challenges. Future Generation Computer Systems, 82:395–411, 2018.
- [67] J. Kim, N. Shin, S. Y. Jo, and S. H. Kim. Method of intrusion detection using deep neural network. In IEEE International Conference on Big Data and Smart Computing (BigComp), pages 313–316. IEEE, 2017.
- [68] N. Kipnis. A law of physics in the classroom: The case of ohm's law. Science & Education, 18(3):349–382, 2009.
- [69] H. I. Kobo, A. M. Abu-Mahfouz, and G. P. Hancke. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. IEEE access, 5:1872–1899, 2017.
- [70] C. K. Koç. Analysis of sliding window techniques for exponentiation. Computers & Mathematics with Applications, 30(10):17–24, 1995.

- [71] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, et al. Supervised Machine Learning: A review of classification techniques. Emerging Artificial Intelligence Applications in Computer Engineering, 160(1):3–24, 2007.
- [72] J. Kurose and K. Ross. Computer networks: A top down approach featuring the internet, 6th edition. Pearson Addison Wesley, NYC, New York, USA, 2012.
- [73] H. Li, K. Ota, and M. Dong. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing. IEEE Network, 32(1):96–101, 2018.
- [74] Y. Li, B. Fang, L. Guo, and Y. Chen. Network anomaly detection based on TCM-KNN algorithm. In Proceedings of the 2nd ACM Symposium on Information, Computer and Communications security, pages 13–19, 2007.
- [75] X. Liang and T. Znati. A long short-term memory enabled framework for DDoS detection. In IEEE Global Communications Conference (GLOBECOM), pages 1–6, 2019.
- [76] A. Likas, N. Vlassis, and J. J. Verbeek. The global k-means clustering algorithm. Pattern Recognition, 36(2):451–461, 2003.
- [77] S. Linic, P. Christopher, and D. B. Ingram. Plasmonic-metal nanostructures for efficient conversion of solar to chemical energy. Nature materials, 10(12):911–921, 2011.
- [78] D. Liu, Y. Li, Y. Hu, and Z. Liang. A P2P-botnet detection model and algorithms based on network streams analysis. In International Conference on Future Information Technology and Management Engineering (Acronym of the conf.), volume 1, pages 55–58, 2010.
- [79] F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In Eighth IEEE International Conference on Data Mining, pages 413–422. IEEE, 2008.
- [80] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach. CoRR, 1, 2019.
- [81] logpai. Deep-loglizer, 2021. <https://github.com/logpai/deep-loglizer> Accessed on 30.11.2021.

- [82] G. Lyon. NMAP, 2021. <https://nmap.org> Accessed on 14.01.2022.
- [83] Á. MacDermott, P. Kendrick, I. Idowu, M. Ashall, and Q. Shi. Securing things in the health-care internet of things. In 2019 Global IoT Summit (GloTS), pages 1–6. IEEE, 2019.
- [84] K. Mejri. Identification of infected IoT devices using network traffic and energy consumption analyses. Technical report, Mediterranean Institute of Technology, Tunisia, 2020.
- [85] A. Morton and I. Mareels. An efficient brute-force solution to the network reconfiguration problem. IEEE Transactions on Power Delivery, 15(3):996–1000, 2000.
- [86] H. Ning, H. Liu, and L. T. Yang. Cyberentity security in the Internet of Things. Computer, 46(4):46–53, 2013.
- [87] D. Oh, D. Kim, and W. W. Ro. A malicious pattern detection engine for embedded security systems in the Internet of Things. Sensors, 14(12):24188–24211, 2014.
- [88] A. Oliner and J. Stearley. What Supercomputers Say: A Study of Five System Logs. In 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN'07), pages 575–584. IEEE, 2007.
- [89] F. J. A. Padilla, E. Baccelli, T. Eichinger, and K. Schleiser. The future of IoT software must be updated. In IAB Workshop on Internet of Things Software Update (IoTSU), pages 1–4, Dublin, Ireland, 2016. Internet Architecture Board (IAB).
- [90] H. H. Pajouh, G. Dastghaibiyfard, and S. Hashemi. Two-tier network anomaly detection model: a machine learning approach. Journal of Intelligent Information Systems, 48(1):61–74, 2017.
- [91] H. H. Pajouh, R. Javidan, R. Khayami, A. Dehghantanha, and K.-K. R. Choo. A two-layer dimension reduction and two-tier classification model for anomaly-based intrusion detection in IoT backbone networks. IEEE Transactions on Emerging Topics in Computing, 7(2):314–323, 2016.

- [92] T.-S. Park and M.-S. Jun. User authentication protocol for blocking malicious user in network CCTV environment. In 2011 6th International Conference on Computer Sciences and Convergence Information Technology (ICCIT), pages 18–24. IEEE, 2011.
- [93] T. Pecorella, L. Brilli, and L. Mucchi. The role of physical layer security in IoT: A novel perspective. Information, 7(3):49, 2016.
- [94] G. Qu, S. Hariri, and M. Yousif. Multivariate statistical analysis for network attacks detection. In The 3rd ACS/IEEE International Conference on Computer Systems and Applications, 2005., page 9. IEEE, 2005.
- [95] J. R. Quinlan. Induction of decision trees. Machine learning, 1(1):81–106, 1986.
- [96] S. Raza, S. Duquennoy, T. Chung, D. Yazar, T. Voigt, and U. Roedig. Securing communication in 6LoWPAN with compressed IPsec. In 2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), pages 1–8. IEEE, 2011.
- [97] M. Roopak, G. Y. Tian, and J. Chambers. Deep learning models for cyber security in IoT networks. In IEEE 9th annual computing and communication workshop and conference (CCWC), pages 0452–0457. IEEE, 2019.
- [98] A. Rosay, E. Cheval, F. Carlier, and P. Leroux. Network Intrusion Detection: A Comprehensive Analysis of CIC-IDS2017. In 8th International Conference on Information Systems Security and Privacy, pages 25–36, Online Streaming, France, 2022. SCITEPRESS - Science and Technology Publications.
- [99] ruCyberPoison. Mirai IoT BotNet. <https://github.com/ruCyberPoison/-Mirai-Iot-BotNet> Accessed on 13.01.2022.
- [100] N. R. Sabar, X. Yi, and A. Song. A Bi-objective Hyper-Heuristic Support Vector Machines for Big Data Cyber-Security. IEEE Access, 6:10421–10431, 2018.
- [101] A. R. Sfarand, Z. Chtourou, and V. Challal. A systemic and cognitive vision for IoT security: A case study of military live simulation and security challenges. In 2017 International Conference on Smart, Monitored and Controlled Cities (SM2C), pages 101–105. IEEE, 2017.

- [102] S. Shirali-Shahreza and Y. Ganjali. Protecting Home User Devices with an SDN-Based Firewall. IEEE Transactions on Consumer Electronics, 64(1):92–100, 2018.
- [103] P. Shukla. ML-IDS: A machine learning approach to detect wormhole attacks in Internet of Things. In Intelligent Systems Conference (IntelliSys), pages 234–240, 2017.
- [104] P. Soucy and G. W. Mineau. A simple KNN algorithm for text categorization. In Proceedings 2001 IEEE International Conference on Data Mining, pages 647–648. IEEE, 2001.
- [105] B. L. R. Stojkoska and K. V. Trivodaliev. A review of Internet of Things for smart home: Challenges and solutions. Journal of Cleaner Production, 140:1454–1464, 2017.
- [106] S. M. Taghavinejad, M. Taghavinejad, L. Shahmiri, M. Zavvar, and M. H. Zavvar. Intrusion Detection in IoT-Based Smart Grid Using Hybrid Decision Tree. In 2020 6th International Conference on Web Research (ICWR), pages 152–156, 2020.
- [107] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pages 1–6. IEEE, 2009.
- [108] M. Tavallae, E. Bagheri, W. Lu, and A. A. Ghorbani. A detailed analysis of the KDD CUP 99 data set. In 2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications, pages 1–6. Ieee, 2009.
- [109] M. Tawfik, M. Nasser, B. Alsellami, A. M. Al-Hejri, and S. Nimbhore. Internet of Things-Based Middleware Against Cyber-Attacks on Smart Homes using Software-Defined Networking and Deep Learning. In 2021 2nd International Conference on Computational Methods in Science & Technology (ICCMST), pages 7–13, 2021.
- [110] I. R. R. Team. IoT security primer: Challenges and emerging practices. Gartner <https://www.gartner.com/en/doc/iot-security-primer-challenges-and-emerging-practices>, 2018.
- [111] N. Team. Netscout threat intelligence report. Gartner <https://www.netscout.com/threatreport/>, Accessed 06.07.2022, 2022.

- [112] thread9. RouterSplit, 2018. <https://github.com/threat9/routersplit> Accessed on 20.02.2021.
- [113] S. Tong and D. Koller. Support Vector Machine Active Learning with Applications to Text Classification. Journal of Machine learning research, 2:45–66, 2001.
- [114] X. Wang, D. Wang, Y. Zhang, L. Jin, and M. Song. Unsupervised learning for log data analysis based on behavior and attribute features. In Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, pages 510–518, 2019.
- [115] N. G. Weiskopf and C. Weng. Methods and dimensions of electronic health record data quality assessment: enabling reuse for clinical research. Journal of the American Medical Informatics Association, 20(1):144–151, 2013.
- [116] I. H. Witten, E. Frank, M. A. Hall, and C. J. Pal. Practical Machine Learning Tools and Techniques. In Data Mining, volume 2, page 4, 2005.
- [117] G. Wood. Ethereum: A secure decentralised generalised transaction ledger. Ethereum project yellow paper, 151:1–32, 2014.
- [118] W. Xu, L. Huang, A. Fox, D. Patterson, and M. Jordan. Online System Problem Detection by Mining Patterns of Console Logs. In Ninth IEEE International Conference on Data Mining, pages 588–597. IEEE, 2009.
- [119] R. B. Yadav, P. S. Kumar, and S. V. Dhavale. A survey on log anomaly detection using deep learning. In 2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO), pages 1215–1220. IEEE, 2020.
- [120] Z. Yan, P. Zhang, and A. V. Vasilakos. A survey on trust management for Internet of Things. Journal of Network and Computer Applications, 42:120–134, 2014.
- [121] S. Zeadally and N. Jabeur. Cyber-physical system design with sensor networking technologies. Institution of Engineering and Technology, United Kingdom, 2016.

- [122] Y. Zhang and X. Huang. Security and privacy techniques for the industrial internet of things. In Security and Privacy Trends in the Industrial Internet of Things, pages 245–268. Springer, 2019.
- [123] S. Zhao, W. Li, T. Zia, and A. Y. Zomaya. A dimension reduction model and classifier for anomaly-based intrusion detection in internet of things. In IEEE 15th International Conference on Dependable, Autonomic and Secure Computing, 15th International Conference on Pervasive Intelligence and Computing, 3rd International Conference on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), pages 836–843. IEEE, 2017.