DETECTION AND CLASSIFICATION OF DDOS FLOODING ATTACKS IN SMART HOME NETWORKS USING MACHINE LEARNING TECHNIQUES AND RULE-BASED ALGORITHM.

ASMAU WALI KAZAURE

SCHOOL OF SCIENCE AND TECHNOLOGY

NOTTINGHAM TRENT UNIVERSITY

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF THE REQUIREMENTS OF

NOTTINGHAM TRENT UNIVERSITY

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

MARCH 2023

Dedication

I confirm that this is my own work and the use of all materials from other sources has been properly and fully acknowledged.

No part of this thesis has already been, or is being currently submitted for any such degree, diploma, or other qualification.

Signature: ASMAU WALI

Email: N0825492@ntu.ac.uk

Date: March 31st, 2023

Acknowledgement

I would like to express my immense gratitude to my supervisor Dr. Xiaoqi Ma for his expert guidance, support and understanding throughout my PhD journey. I would also like to thank my second supervisor Dr. Jun He for his relentless professional support all through. I would like to thank Petroleum Technology Development Fund, Nigeria for making this possible. To my colleague Michael, thank you for the energy you brought during our collaborative research works. Special thanks to Amir for the insightful guidance all along.

My heartfelt gratitude goes to my parents for their unwavering support, sacrifices and encouragement throughout this journey. The support from my siblings translated into a source of strength during challenging times. I will forever be grateful to my husband for the sincere support, motivation and care. This means the world to me. To my in-laws, your trust and belief in my aspirations coupled with your warm support and encouragement has really contributed to my success. I thank God for brining all these people together in my life as their collated support has resulted in this successful journey. Thank you, God, for giving me the strength, knowledge and wisdom all through.

Abstract

Smart homes are gaining more popularity by the day due to the ease they provide in terms of running our homes. However, the energy and resource constrained nature of the smart home devices make security integration challenging, thus making them prone to cyber-attacks. DDoS remains one of the most threatening attacks to this network and IoT in general. To curb this issue, there is a need to study the behavioural pattern of this attack and smart home devices at a low level. This will aid in designing a timely and more effective DDoS detection and attack type classification system, which is what this thesis presents.

This research collects DDoS and benign traffic in a real smart home environment and performs an Exploratory Data Analysis (EDA), visualizing the behavioural pattern of DDoS flooding attacks when targeted at smart home networks in comparison to the benign smart home traffic pattern. Specific smart home traffic properties were selected, correlated, and visualized showing their reversed behaviour during an attack compared to their normal benign nature. To further validate the findings, public IoT datasets were analysed in the same manner and the same results were achieved. The results and observations from the findings are used to propose and implement a novel hybrid anomaly and feature-based DDoS detection and attack type classification system.

The implemented system detects and classifies a wide range of DDoS flooding attacks at the very onset including unfamiliar, amplification, and protocol-based attacks. To validate this system, it is tested rigorously on both private and public sourced benign and infiltrated smart home traffic. An excellent performance was recorded making it not user, device or attack centric among other benefits.

Due to the excellent performance recorded, the attack type classification approach was further applied to a supervised machine learning model, Random Forest. This was tested to find out the performance of the Random Forest model in attack type classification compared to when it is coupled with the classification module from the hybrid anomaly and feature-based solution. The performance clearly showed the latter outperforming the Random Forest model on its own by far in terms of attack type classification, thus proving that domain knowledge is very important when it comes to security design and implementation even when using Machine Leaning models.

Contents

List of Publications

- Wali, A., Apejoye, O., He, J. and Ma, X., 2021, December. An exploratory data analysis of the network behavior of hive home devices. In 2021 8th International Conference on Internet of Things: Systems, Management and Security (IOTSMS) (pp. 1-8). IEEE.
- Wali, A., Apejoye, O., Raja, T., He, J. and Ma, X., 2022, September. A Novel Approach to Identifying DDoS Traffic in the Smart Home Network via Exploratory Data Analysis. In International Conference on Applied Intelligence and Informatics (pp. 478-498). Cham: Springer Nature Switzerland.
- Raja, T.V., Ezziane, Z., He, J., Ma, X. and Kazaure, A.W.Z., 2022, October. Detection of DDoS Attack on Smart Home Infrastructure Using Artificial Intelligence Models. In 2022 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC) (pp. 12-18). IEEE.

Chapter 1

Introduction

1.1 Background

IoT devices are becoming more popular in our daily lives due to the advantageous services they render to users. These devices cover a broad surface in terms of connectivity ranging from but not limited to, healthcare, home automation, weather forecast, transport, agriculture, security, and a variety of other dimensions. IoT further gives us the ability to have more control over our IoT ecosystem. By tailoring these devices to run exactly when we need them, this improves our energy conservation plans. We also get to monitor devices' usage in real time, which paves way for accountability when the need arises. It is estimated that 150,000 IoT devices join the global network every minute [1].

However, the energy and resource constrained nature of these devices make them prone to cyber-attacks [2] [3] [4] [5] [6]. This, in addition to their heterogeneous nature, makes security implementation challenging [7] [8]. The device vendors are not helping matters too as their focus is more aligned to device functionality and features rather than security [9]. This poses risks in terms of security as the lives of individuals are directly affected [10]. DDoS flooding attacks remain a big threat to the IoT network. During the first quarter of 2020 there has been a significant rise in DDoS attacks witnessing an 80 percent increase from 2019 [11]. This attack tends to flood a targeted server with voluminous unnecessary traffic, in the process over saturating its capacity causing service to be denied or halted to legitimate devices. Several of the server's resources get negatively affected like processing power and memory capabilities as it is preoccupied in dealing with more traffic than it is designed to handle.

The traditional approaches used in DDoS attack detection will not suffice in the smart home network due its heterogeneous and resource constrained nature at device level. This brings about the urgency to develop more efficient, centralized, and scalable solutions to deal with attacks of this form. The main goal of this thesis is to address these contemporary issues by designing and implementing a DDoS detection and attack type identification system based on the general behavioural properties of the smart home network, thereby making it not user, attack or device centric.

1.2 Research Questions

For this persistent attack to be detected and mitigated, there is a prerequisite need to study the individual attack traffic properties in relation to the benign corresponding properties of the smart network. This will help in identifying the affected traffic properties and what to look out for during a DDoS flooding attack. The best way to do this is by using data visualization techniques, as the network traffic is vast and multidimensional. This visualization method will also be beneficial to network security operators as the human brain tends to better process images than text [12]. It can also give network operators the advance notice needed in case of a sensed attack. Although there have been immense contributions in DDoS visualization, detection and mitigation areas, there are still open challenges on the best way to identify and visualize DDoS attack patterns which will pave way for better detection and mitigation approaches [13] [14].

This led to the first 3 research questions which are as follows:

RQ1 - What is the normal traffic pattern of a smart home network when visualized?

RQ2 - What is the traffic pattern of the smart home network during a DDoS attack when visualized?

RQ3 - What general smart home traffic properties get affected during a DDoS flooding attack?

Several works have addressed DDoS detection and classification in the smart home network, however, there remain open challenges in this avenue [13] [14] [15] [16]. Some solutions make use of outdated or simulated data, which might hinder the accuracy when deployed in real life scenarios. In addition, some of the approaches used are not very practical or feasible in some scenarios. For example, using the single packet inspection method to determine if it is malicious or benign [17] [18]. This not only is time and resource consuming, but a less effective way of identifying DDoS patterns. This is due to DDoS flooding attacks being volume based, thus will need a volume based or cumulative approach to determine an attack pattern as opposed to the single packet approach. The approach of employing sequential user behaviour [19] or Device Usage Description (DUD) model [20] is not very practical as the former is user centric which will raise false positives when there is slight change in user pattern while the latter is not practical in large scale scenarios as it’s a device centric solution and not a generalized one. The issue of a solution being attack centric also comes into play [21] [22]. Using too much metrics for attack detection is another issue as it results in high resource consumption and detection time [23].

This led to the last 4 research questions which are as follows:

RQ4 – How can the properties identified in RQ3 be used to develop a light weight, practical, centralized, and counter spoof DDoS detection and attack type identification system that is not user, attack nor device centric covering unfamiliar attack?

RQ5 – Can a single network feature be used to detect all DDoS flooding attacks?

RQ6 – Can feature absence or feature range be used in DDoS attack detection?

RQ7 – Can the approach used in RQ4 be applied to a Supervised Machine Learning model leading to better performance while maintaining the benefits attained in answering RQ4?

RQ8 – Are the following metrics* (**Is attack present, type of attack present, is attack detected, packet number attack started, packet number attack detected, packet number attack type indicted, attack type indicated, window attack started, window attack detected**)** ***more relevant than using confusion matrix in assessing the performance of a DDoS detection and classification system?

1.3 Aim and Objectives

The main aim of this research is to develop a light weight, practical, centralized, and counter spoof DDoS detection and attack type indication system that detects and indicates attack type at the very onset, while not being user, attack nor device centric covering unfamiliar attacks based on the general behaviour exhibited by smart home devices when under attack. To achieve this aim, certain objectives must be outlined and adhered to. These are as follows:

- ✓ Identify gaps in literature relating to DDoS attack detection and classification in the smart home network.
- ✓ Setup a real smart home network to collect benign and attack data.
- ✓ Identify the smart home network properties that get affected during the attack.
- ✓ Design and implement a DDoS detection and attack type indication system.
- ✓ Test and validate the system on both private and public attack and benign datasets.
- ✓ Evaluate the system's performance based on the following factors: Is attack present, is attack detected, packet number attack started, packet number attack detected, packet number attack type indicated, attack type indicated, window attack started, window attack detected.
- ✓ Apply attack type indication module to a supervised machine learning model and compare the attack type indication performance with and without it.

1.4 High level methodology

Due to the broad and multiple research phases involved in this research, a high-level methodology covering the entire research journey is presented here. More detailed and elaborate methodology relating to each phase or chapter is presented at the beginning of each chapter. However, this high-level methodology will give a general and comprehensive overview of the main methodological processes involved and adhered to. As this research is technically inclined, a quantitative research method is used because the research processes are heavily reliant on numerical analysis and empirical data. Figure 1.1 presents this high-level methodology.

This methodology comprises of 5 phases. Each phase and what it entails are as follows:

- ➢ Literature review: The adapted methodology for this phase is "Literature review and focusing the research" [24]. It involves reviewing literature in the field of interest which is detection and classification of DDoS attacks in the smart home network. This includes anomaly, signature and supervised machine learning based solutions. Literature surveys around these topics are consulted to know the current state of the art and gaps. In addition to that, individual research papers, conference proceedings and technical reports are also consulted paying special attention to approach used, detection rates, practicality, light weightiness, validation process, attacks covered and issues the respective approaches and solutions solved. This will help provide a comprehensive and solid understanding of the state of the art relating to the topics in question. This will also help come up with the respective research questions that need answering in this thesis.

- Data collection: This phase involves setting up the smart home network and DDoS attack experiments. Purely benign smart home traffic and a mixture of benign and attack traffic is collected by designing use cases that will provide the required data. The collected data includes several DDoS attacks and mixed attacks. Mixed attacks are a combination of different individual DDoS attacks like TCP SYN, ICMP and UDP attacks launched at the same time targeting the smart home network. Public attack and benign data from reputable sources are also gathered. These will be used in the validation phase.
- Exploratory Data Analysis (EDA): The collected data is analysed using EDA [25] which is a statistical method used to analyse datasets summarizing their main characteristics using data visualization techniques [26]. Both private and public attack and benign data are analysed using this approach. Observations relating to how the smart home network properties are affected during a targeted attack are noted. These will be used to propose, design, and implement the detection and attack type indication system.
- Iterative system development model [27] to design and implement detection and attack type indication system: The iterative model is adapted due to its recursive nature especially at the testing and tuning stage for this system. It is found to be most suitable as lots of testing and tunings are carried out to continuously improve the system after each iteration. The results and observations derived from the data analysis stage are used to design and implement a hybrid anomaly and feature-based detection and attack type indication system. The research questions to be answered and gaps to be bridged are taken into consideration in this phase as the solution will be modelled to successfully answer the research questions.

  The attack type indication approach used in the hybrid anomaly and feature-based detection and attack type indication system is applied to a supervised machine learning model (Random Forest). The ability of the Random Forest (GB) model in classifying attack type will be compared to when it is coupled with the algorithm-based approach which is based on domain knowledge.

The systems performances are evaluated based on quantitative factors which are: Is attack present, type of attack present, is attack detected, packet number attack started, packet number attack detected, packet number attack type indicated, attack type indicated, window attack started, window attack detected. These factors will provide very precise and accurate details as to how early and accurately the attack is detected and type indicated.

The system is further validated by testing on public datasets spanning known attacks, Unfamiliar attacks (attacks not tested or exposed to system previously), mixed attacks and normal traffic using the same performance metrics listed in the performance evaluation phase. This will prove the system’s ability in:

1) Not being user, device or attack centric,
2) Eliminating bias with regards to the private dataset it was initially evaluated on
3) Detecting and attack type identification of a wide range of DDoS attacks including mixed and unfamiliar attacks.

- Compare with state of the art: This phase deals with comparing both the hybrid anomaly and feature-based detection and attack type indication system and the hybrid ML based classification model to existing methods and solutions in the same field. The comparison is based on 12 rigorous factors that provide clear and holistic justification that is scientifically sound.

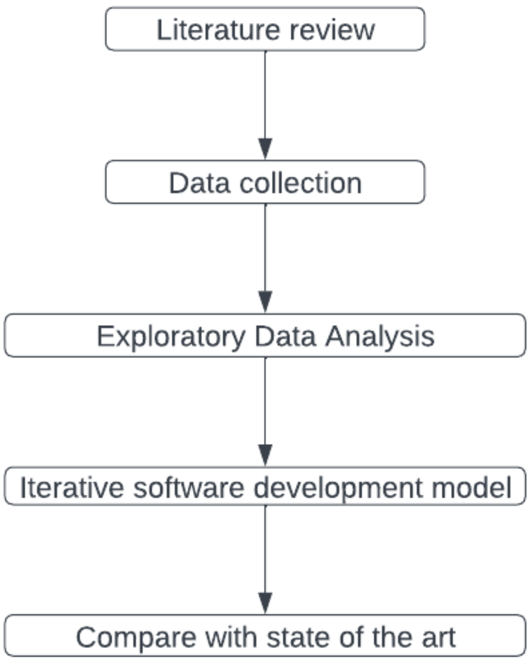

Literature review
Data collection
Exploratory Data Analysis
Iterative software development model
Compare with state of the art

Figure 1.1 High level methodology

1.5 Contributions

This research has 6 main contributions which are as follows:

- ❖ **Contribution 1**: In the event of studying the smart home network behaviour and traffic patterns, unique correlated traffic patterns attributed to each mode of device control was visualized. This correlation and visualization are new with regards to deriving a unique signature for each method or mode used to control the smart devices. The explored modes include manually operating the devices, automated/ scheduled, using Hive app, using Home kit app and using Google home app. The protocol and packet length sequence of each mode of control was found to be unique and uniform regardless of the platform (iPhone, iPad, Samsung smart phone) used to control it. These correlated patterns can be used in forensic investigations to prove how someone controlled a particular device or devices and whether they were present at the scene during some specified times. For instance, if the evidence shows proof of manual mode of operation, then this ties one to physically being at the premises. Furthermore, as each operation mode has a unique traffic pattern, these patterns could be part of the allowed list on the smart home network to detect certain attacks relating to unauthorized control of device which might have a deviating pattern from the allowed listed ones. **This is addressed in chapter 3.**
- ❖ **Contribution 2**: Normal smart home traffic pattern in comparison to when DDoS flooding attacks infiltrate the network are visualized using Exploratory Data Analysis. This visualization is new as it clearly visualizes the benign and attack patterns based on smart home network features that get simultaneously affected during an attack. The visualized network features can be incorporated into data visualisation tools and Intrusion Detection Systems. This will provide clearer low-level statistics as to how the network is deviating from its normal pattern during an attack. The visualised EDA [25] [26] images can also be trained on a Convolutional Neural Network (CNN) [28] using ResNet [29]. It is well known that deep learning models especially CNN achieved high significance due to their outstanding performance in the image processing field. The potential of CNN can be used to detect DDoS attacks by converting the network traffic data into images. **This is addressed in chapter 4**.
- ❖ **Contribution 3**: A new approach to DDoS attack detection has been presented. The approach uses feature absence and feature range in attack detection from the very onset. Some prominent network features (Sequence numbers and TCP flags) were found to be absent for the duration of certain attacks. The narrative needs to be changed from only focusing on present network feature statistics to detect attacks, rather features that are normally present but tend to be absent for a prolonged period also contribute to rapid attack detection as seen in this research. In addition to that, the sequence number range in normal traffic tend to be

very wide, starting with a 0 or 1 at the beginning of a session and keeps incrementing to very high values. However, during an attack, the sequence numbers were found to stall at 0 or 1 all through. This new detection technique led to contribution 4. **This is addressed in chapter 4.**

- ❖ **Contribution 4**: A hybrid anomaly and feature-based DDoS detection and attack type indication algorithm has been implemented and tested. This algorithm is based on the findings in contribution 3. The feature absence (sequence numbers & TCP flags) and feature range (Sequence numbers) phenomena are used as baseline as to what is considered anomaly in the traffic. Thus, this is incorporated as conditions in the algorithm to flag the anomalies as attack packets. After a grouped series of packets are flagged and labelled as attack, the protocol with the highest count among those flagged packets is used as the attack type classification label for each of the attack labelled packets. Both detection and attack type identification modules of the system performed excellently with only 1 wrong prediction due to traffic from a new device joining the network. The system was able to detect and indicate the attack type at the very onset. In addition to that the solution is light weight, practical, centralized, and counter spoof that is not user, attack nor device centric covering unfamiliar and mixed attacks. **This is addressed in chapter 5.**
- ❖ **Contribution 5**: A hybrid Machine learning detection and attack type identification model is developed. Random Forest model is trained based on the same network features used in contribution 4. The model was able to accurately detect the attack at the very onset and to some extent classify the attack type. The model was able to predict the presence/absence of an attack in all testcases within the first 5 packets of an attack except for the attack type prediction which it failed to correctly predict in 5 testcases out of 20. However, the novel attack type identification approach used in contribution 4 which is based on highest protocol count among the attack labelled packets was applied to the Random Forest model. After the RF model detects the attack, the attack type indication module applies the highest protocol count check and labels the attack type using that. This hybrid model outperformed the RF's ability to classify the attack type correctly including unfamiliar attacks with 99% accuracy. In all the testing and validation cases, the hybrid model predicted all testcases correctly except for one instance due to the same reason of new device joining the network as in contribution 4 in Indicating the attack type while the RF model on its own misclassified the attack type 6 out of 20 testcases. This proves that the hybrid model is more effective in terms of attack type identification**. This is addressed in chapter 6**.

- ❖ **Contribution 6**: A new approach to assessing the performance of a DDoS attack detection and attack type identification system is presented. This new approach is proven to be more relevant in terms of precisely measuring the system's ability to detect and identify attack types at the very onset and how accurate the prediction is. Currently the conventional method is the use of confusion matrix [30]. However, confusion matrix does not specify how early the attack is detected or classified rather it gives statistics on how much the solution was able to predict right. This proposed approach is used in this research and has proven to provide more relevant performance details. The metrics used to gauge the performance of the detection and attack type indication system on each data source in this new approach are: Is attack present, type of attack present, is attack detected, packet number attack started, packet number attack detected, packet number attack type indicated, attack type indicated, window attack started, window attack detected. **This is addressed in chapter 5.**

1.6 Scope and limitations

The scope of this research is limited to the following areas:

- ➢ Smart home network: The focus of this research is on smart home devices and the network they form; thus, the solution is tailored to detect and indicate attack types in this network.
- ➢ TCP/HTTP based protocols: This research is also limited to devices that use TCP/HTTP communication protocols; thus, the solution is only intended for TCP/HTTP based traffic with TLS/SSH based encryption. However, the solution still accommodates other protocols like UDP, ICMP and the like used by these TCP/HTTP based devices.
- ➢ DDoS flooding attacks: The attacks covered in this research are limited to DDoS flooding attacks like TCP SYN, UDP, ICMP, HTTP, DNS, NTP, ARP and the like thus slow stealth DDoS attacks are not catered for by the solutions.
- ➢ Detection and attack type indication: The implemented and tested system is limited to detecting and indicating attack types as no mitigation is involved at this point.

1.7 Thesis structure

The rest of this thesis is structures as follows:

- ❖ **Chapter 2 Literature review**: This chapter deals with reviewing the current state of the art in the field of DDoS detection and classification in the smart home network. The challenges, gaps and open research questions relating to the solutions are identified.
- ❖ **Chapter 3 Smart Home network behaviour**: This chapter delves into setting up a real-life smart home network and collecting data from the network. The collected data is analysed to understand the behavioural pattern of the network.

- ❖ **Chapter 4 Exploratory Data Analysis comparing attack and benign smart home traffic properties**: This chapter is concerned with attacking the smart home network with DDoS flooding attacks and comparing the attack and benign patterns of the network using EDA.
- ❖ **Chapter 5 A novel algorithm-based DDoS attack detection and attack type indication in the smart home network**: This chapter implements the DDoS detection and attack type indication algorithm based on the findings from the EDA in chapter 4.
- ❖ **Chapter 6 A novel hybrid Machine Learning detection and attack type classification model using domain knowledge**: This chapter presents the hybrid supervised ML based attack detection and attack type indication model using the features and attack type indication approach from chapter 5.
- ❖ **Chapter 7 Research contributions**: The overall research contributions and how they were achieved are discussed here.
- ❖ **Chapter 8 Conclusion**: This chapter concludes the thesis. The limitations are discussed as well as future work.

Chapter 2

Literature review

2.1 Introduction

The fast evolving and disruptive nature of DDoS attacks has become a great concern for smart homeowners. Due to this, security experts and researchers are continuously conducting investigations and research studies with regards to better and more efficient methods of dealing with this attack. The main goal of this chapter is to harness knowledge of the current state of the art, thereby identifying existing solutions, the gaps involved and how to bridge these gaps in ways that will address both the attack and the constrained and heterogenic nature of the network.

The following methodology [24] was adhered to for a comprehensive and holistic understanding of the current state of the art and opportunities to build on.

- ✓ Identify research topic: Topics with the following key words are the points of interest: *Anomaly/signature/ML/hybrid-based DDoS detection and classification systems in IoT, IDS in smart home networks, DDoS attack identification and traffic analysis in smart homes, smart home network characteristics.*
- ✓ Database query: Reputable academic search engines and databases will be queried with the key words of interest. These include Google Scholar, IEEE Xplore, ScienceDirect.
- ✓ Review secondary sources to get an overview of the topic: Literature surveys/ reviews of the current state of the art relating to the topic of interest are consulted.
- ✓ Develop a search strategy and use appropriate preliminary sources and primary research: An inclusion and exclusion criteria will be outlined to avoid going out of scope. Relevant information from selected sources will be extracted like detection and classification approaches, performance metrics used, detection time, practicality and overall strengths and limitations.
- ✓ Prepare bibliographic information and notes on each article: Relevant details derived from each source will be stitched together.
- ✓ Evaluate the research reports: The potential gaps identified based on the consulted sources will be identified. This will give rise to research questions of interest and what approach/ methodology to employ while bridging these gaps.

The rest of the chapter is structured as follows: Section 2.2 delves into the smart home architecture and network characteristics. 2.3 discusses the security challenges and risks associated with smart home networks due to their nature. 2.4 delves into an overview of DDoS attacks and their impact on smart home networks. 2.5 is concerned with the existing solutions in DDoS detection and classification.

2.6 discusses the main research gaps identified and how this thesis will contribute to bridging some of these gaps. 2.7 summarises the chapter.

2.2 Smart home eco-system

Advancements on the Internet of Things (IoT) has created a sharp rise in the popularity of smart homes over the years. The ease, convenience and efficiency that come with smart home devices keeps necessitating their presence in our lives today. Very drastic increase in the use of smart devices has been witnessed with statistics predicting that over 75 billion IoT devices will be connected around the world [31]. Figure 2.1 shows a projection of the sharp increase in the use of these interconnected devices from 2015-2025.

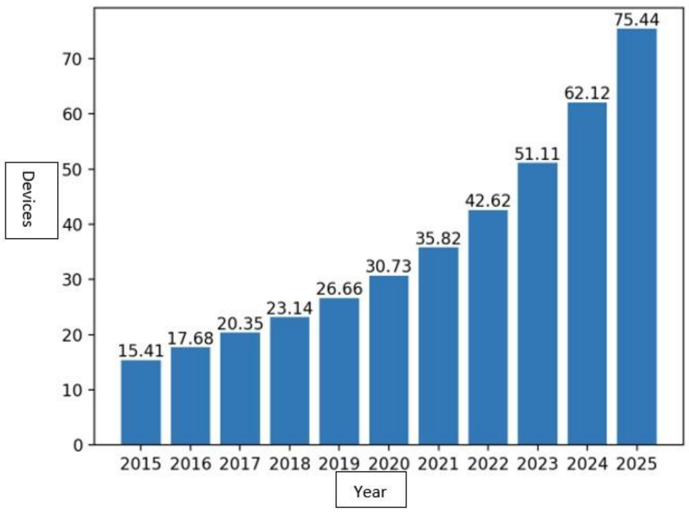


Figure 2.1 Projection of increase in IoT use, 2015-2025 [31]

The smart home eco system is made up of interconnected devices, controllers, sensors, and actuators. These exchange signals and communicate via numerous protocols [32]. The popular communication protocols include Zigbee, Z-wave, Wi-Fi, and Bluetooth [33].

The smart home eco system can be broadly categorized into 3 namely centralized, decentralized and hybrid [34]. Centralized infrastructure is managed by a single hub that handles device management and data processing duties. As this type of connectivity comes with good coordination, it is prone to single point of failure. On the other hand, the decentralized version shares the decision-making process among several other devices that intercommunicate. This is deemed safer but can be difficult to maintain [35]. The hybrid infrastructure is a combination of both centralized and decentralized that brings about more flexibility and control [36].

The smart home can also be looked at from a layered perspective. There are four main layers which are physical, communications, information, and decision [37]. The physical layer consists of hardware components like routers. The communication layer handles the connection between servers, routers, and other networked devices [38]. The information layer handles data storage, processing and analysis received from devices and sensors in the network [39]. The decision layer determines what action or data is to be stored in the information layer [40].

From this we can see how complex and heterogeneous the smart home eco system can be.

2.3 Challenges and risks concerning smart home network

Since smart home networks are becoming more and more interconnected with and very much internet dependant, this broadens the attack surface in addition to their not so security accommodating nature. This section delves into the various smart home network properties that make them susceptible to cyber-attacks.

Among the smart home characteristics that make them highly prone and exposed to the fast-evolving cyber-attacks are:

- Resource constrained: Smart home devices tend to have very limited memory, battery resource and processing power. This brings about the challenge in incorporating security protocols, thus leaving them exposed [41].
- Protocol diversity: The heterogeneous nature of the devices under a single roof without standardized and unified method of consolidation makes it difficult to implement seamless security measures [41].
- Wide attack surface: The ever-increasing smart home connections widens the attack surface thereby making provision for multiple entry and routing points for cyber attackers [42].
- Update challenges: Updating firmware and applying patches is difficult with the devices frequent updates to fix vulnerabilities [43].

The smart home device vendors are not making things easier as they tend to prioritize functionality over security. This results in having vulnerable devices in the market due to several security flaws like weak authentication credentials and weak or no encryption. This makes them prone to numerous attacks like unauthorized access, Man-In-The-Middle attack, reverse engineering attacks and the like [44] [45] [46]. Some popular cyberattacks on smart devices over the years include:

- The baby monitor hack: In 2015, there were several reports of baby monitors being hacked as the attackers gained unauthorized access. The BBC reported a specific case of a hacker that shouted obscenities after taking control of a couple’s baby monitor [47].
- Ring camera hack: In 2019, unauthorized access to Ring cameras was reported. This event really put emphasis on privacy risks related to smart home devices. Weak and reused

credentials were the main point of exploit in this attack for unauthorized access. The attackers after gaining access were able to communicate with the legitimate owners and view live video feeds [48].

- Mirai Botnet: In 2016, a malware called Mirai infiltrated smart home devices among other IoT. This was done by exploiting weak and default authentication credentials. The infected devices were used as Bots to spread DDoS attacks across high profile organizations leading to serious outages. More than 600,000 devices were affected [49].
- Dyn attack: In 2016, a company called Dyn had its servers targeted with DDoS. This company provides DNS related services. This resulted in significant disruptions to popular sites like CNN, Reddit, Netflix and Twitter among others. The botnets used in this attack were largely made up of IoT devices like digital cameras and DVR players [50].
- GitHub attack: In 2018, GitHub was hit by DDoS attack that lasted for about 20 minutes. This was traced back to over a thousand autonomous systems with tens of thousands of unique end points which greatly overwhelmed their defence measures [50].

2.4 DDoS flooding attacks in the smart home network

DDoS flooding attacks remain a big threat to the IoT network. During the first quarter of 2020 there has been a significant rise in DDoS attacks witnessing an 80 percent increase from 2019 [11]. This attack tends to flood a targeted server with voluminous unnecessary traffic, in the process over saturating its capacity causing service to be denied or halted to legitimate devices. Several of the server's resources get negatively affected like processing power and memory capabilities.

DDoS attacks are well-known attacks of major concern violating the "Availability" principle of security. The attack vectors exploit several internet protocols features built decades ago when security was not a concern as it is today [51]. This attack is majorly categorised in two namely bandwidth and resource depletion attacks [52]. In the bandwidth depletion attack, legitimate looking traffic that is highly voluminous gets directed towards the target device or network while the resource depletion attack directs bogus service request that appear legitimate to the target device, thereby tricking it to respond continuously to the level that it can't respond to legitimate devices requiring its service. In both scenarios service is denied to those that require it as the name implies Denial of Service attack. When multiple sources or agents are used to direct this attack to a target device, this becomes Distributed Denial of Service (DDoS). Figure 2.2 shows a taxonomy of these attacks [53]. A breakdown of these attacks and how they operate is as follows:

- Amplification attack: In this type of attack, a request is sent by the attacker to several DNS or NTP servers using a spoofed address which is the victims IP address. The servers respond to this with much larger bytes thereby overwhelming the victims' network.
- Protocol exploit attack: In this type of attack, the communication protocol is taken advantage of. An example is the TCP SYN flood attack where the attacker takes advantage of the 3-way TCP handshake, initiating the process without completing it. It floods the target server with SYN requests which arrive faster than the target server can process them, thus leaving it saturated. This results in the connection being half open as it is never acknowledged or ended [53]
- Zero-day attack: This attack tends to exploit an unknown vulnerability in a target device, software, or network. Dealing with this sort of attack is challenging as traditional signature or anomaly-based IDS still miss it [54].
- ICMP: This attack tends to overwhelm the target server with ICMP echo requests (pings). The server tries to process each incoming packet and responds to it and in the process failing to process legitimate packets as it is already saturated.

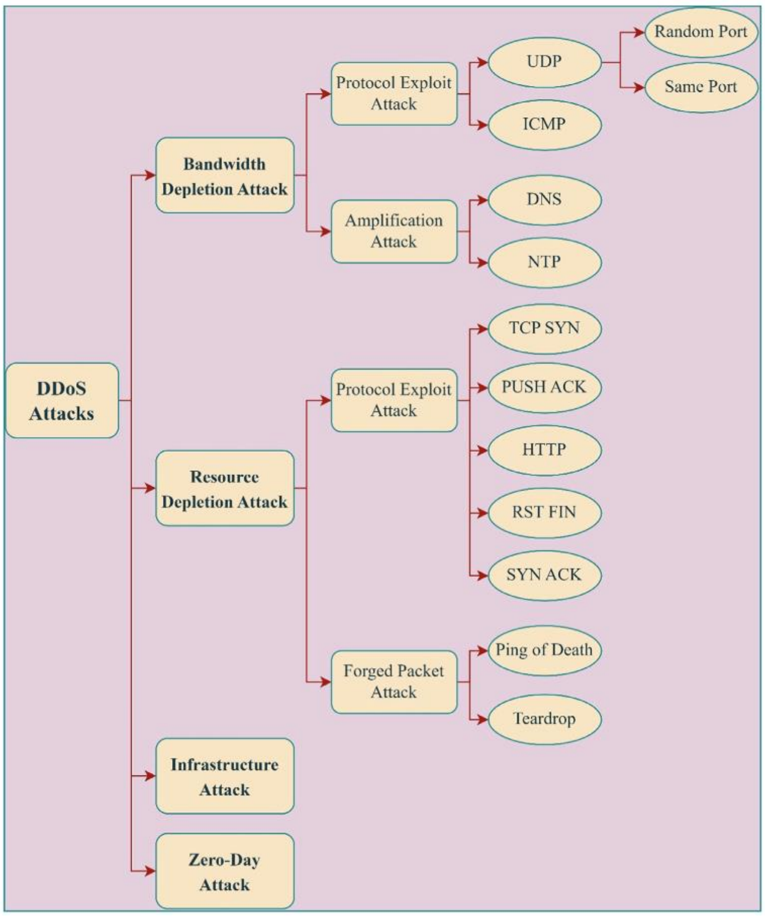


Figure 2.2 DDoS attack taxonomy [53]

2.5 DDoS detection and classification in smart home networks

Several works have addressed DDoS detection in the smart home network, however, there remain open challenges in this avenue [13] [14] [15] [16]. For the attack to be efficiently and effectively detected and mitigated, analysis of the smart home network characteristics in relation to the attack patterns remain crucial. This section delves into the use of data visualization in cyber security and some DDoS detection and classification tools and approaches. The approaches of interest here are rule/signature based, supervised machine learning based, and hybrid-based Intrusion Detection and (Prevention) Systems (ID(P)S). However, the scope of this research is limited to detection and not prevention or mitigation.

2.5.1 Data visualization in cyber security

Data visualization is very important when it comes to understanding normal device behaviour and attack patterns. This can help security analysts in spotting outliers, patterns and trends that separate a normal behaviour from a malicious one [14]. When it comes to DDoS attacks, data visualization aids in anomaly detection in a network traffic [55].

Smart home, being one of the most popular and relatable IoT to users, has gained a lot of attention in the research community. This has led to growth in research relating to smart home behaviour and security. Several works have addressed smart home device identification or fingerprinting [56] [57] [58] [59] [60] [61] [62] [63] [64] [65] [66]. Privacy attacks from the adversary angle have also gained much attention within the research community, thereby being able to profile a smart home users' behaviour [67] [68] [69] [70] [71] [72] [73] [74] from unencrypted logs.

However, with all these developments, it is observed that there is lack of visualization for low-level network features in relation to their behaviour during a DDoS flooding attack. Studies mainly focus on smart home behaviour profiling and device fingerprinting. For this persistent attack to be detected and mitigated, there is a prerequisite need to study the individual attack traffic properties in relation to the benign corresponding properties of the smart network. This will help in identifying the affected traffic properties and what to look out for during a DDoS flooding attack. The best way to do this is by using data visualization techniques, as the network traffic is vast and multidimensional. This visualization method will also be beneficial to network security operators as the human brain tends to better process images than text [12]. It can also give network operators the advance notice needed in case of a sensed attack. Although there have been immense contributions in DDoS detection techniques, there are still open challenges on the best way to identify and visualize DDoS attack patterns which will pave way for better detection and mitigation approaches [13] [14].

Exploratory Data Analysis is a statistical method of analysing data to summarize the main characteristics of the dataset by using data visualization tools and techniques to represent the derived results for ease of understanding. Visualization features like scatter plots, heat maps and time series graphs will help give insights to attack patterns [61].

2.5.2 Rule/ signature/ hybrid -based ID(P)S

The signature-based ID(P)S monitors network traffic for known or prestored signatures relating to numerous known attacks. This allows it to precisely identify an attack for what exactly is as it already has knowledge of how it looks or behaves [54] [75] [76]. However, this ID(P)S cannot recognize unfamiliar attacks because it has not learned its signature or behavioural pattern in the past. Due to this, the signature-based ID(P)S requires a frequently updated database to keep track of the most recent attack signatures so it can flag them right away.

The anomaly-based ID(P)S monitors a traffic for deviating patterns from the normal traffic pattern and uses this as a basis for intrusion detection. The challenging part of this ID(P)S is the very thin line between normal and abnormal traffic patterns which in turn leads to False Positive alarms [75] [76] [77].

The hybrid-based ID(P)S is a combination of both signature and anomaly-based versions. This combines both advantages of the anomaly and signature-based systems, providing a more robust solution. Nonetheless it can be challenging to implement and consolidate [54]. The working process as well as the up and down sides of the anomaly and hybrid- based ID(P)S are shown in figure 2.3 and table 2.1.

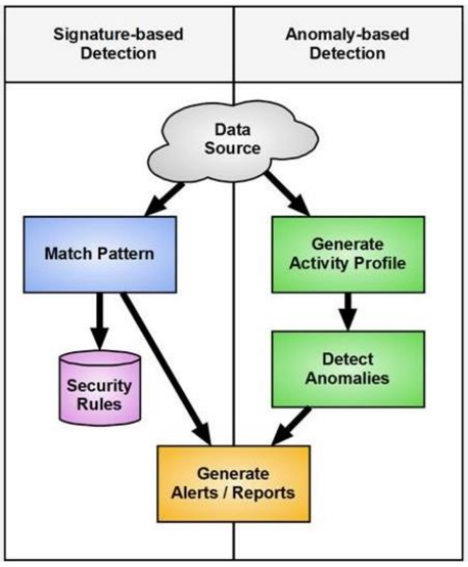
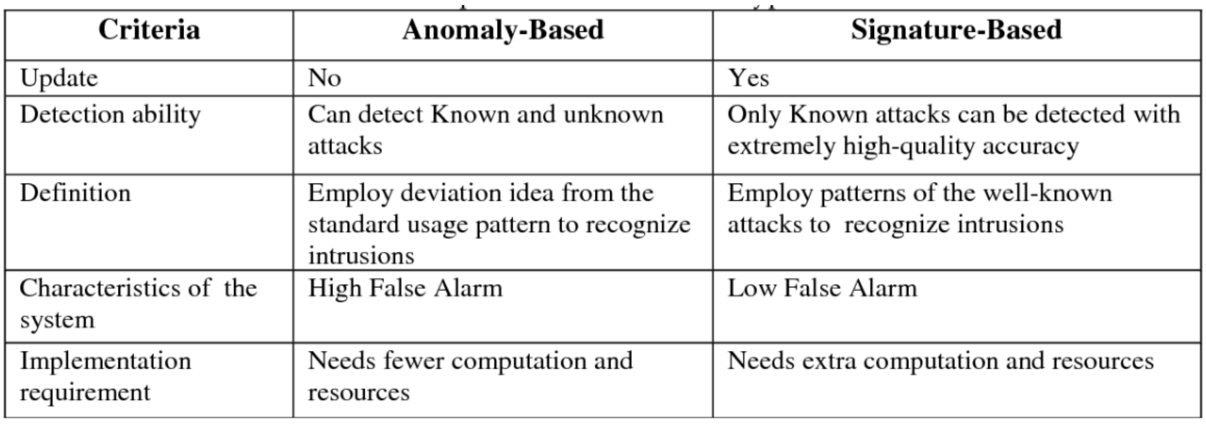


Figure 2.3 Signature & anomaly-based monitoring process [76]



Criteria	Anomaly-Based	Signature-Based
Update	No	Yes
Detection ability	Can detect Known and unknown attacks	Only Known attacks can be detected with extremely high-quality accuracy
Definition	Employ deviation idea from the standard usage pattern to recognize intrusions	Employ patterns of the well-known attacks to recognize intrusions
Characteristics of the system	High False Alarm	Low False Alarm
Implementation requirement	Needs fewer computation and resources	Needs extra computation and resources

Table 2.1 Upsides and downsides of signature and anomaly-based ID(P)S [77]

Snort and Suricata are two popular opensource ID(P)S's. Snort is an IPS that can run in IDS mode purchased by Cisco from Sourcefire in 2013. It detects malicious activities like DoS and unauthorised access attacks. It carries out real time traffic analysis and packet logging. Snort has features to support anomaly-based detection. Similarly, Suricata is another IDPS tool that supports both signature and anomaly-based features. It was developed by the Open Information Security Foundation (OSIF). Both Snort and Suricata have proven effective in traditional networks, however certain limitations arise when they are deployed in smart home networks or IoT in general due to the nature of these networks in terms of resources, scalability, and high false alarm rates due to how heterogenic the network is [78].

Researchers in [20] propose the use of Device Usage Description (DUD) model for device behaviour and flow rules extraction to detect DDoS attacks in a smart home network. Nevertheless, this method tends to be device specific and may be problematic in large-scale networks as extracting each device DUD may not be feasible and bring about significant overheads. The paper also states traffic properties considered in generating flow rules for DDoS detection but without any visual representation or comparison. User behavioural pattern was also used in [19] to develop anomaly detection model in IoT device operations. The sequence of activities performed by the user is learned and any deviation from this sequence is classified as an anomaly. However, this could be problematic as any change in user behaviour can raise a false positive alarm.

A rule based SIEM detection model is presented in [79]. It analyses a series of packets to find out if certain rules are breached based on a fixed threshold. Its main detection metric is SYN flags to detect TCPSYN attacks. This was tested on a simulated network. An SDN based detection system is presented in [23]. Various metrics like number of flow entries, similar payload packet count, number of sent and received packets on each node, power ratio of each node, in/out traffic load and session IP counter among others. Due to the intricacy of processing these metrics, it results in high detection time and consumes a lot of processing power, thus the need to investigate on a light weight and intelligent method to detect attacks. The use of a packet counter can also be problematic in diverse environments as each device tends to behave differently in terms of number of packets transmitted over time. The use of similar payload in attack detection raises the rate of false positive alarms. The paper mentions early attack detection but without how this was evaluated.

A Networked Smart Object (NOS) at edge is presented in [43]. The NOS acts as the middleware which approves and acknowledges every request on the network before a connection is established. However, this results in delays and the volume of attacks it can handle is determined by the number of NOS. It is also resource and memory consuming due to the TCP protocol involved in the request and approval or acknowledgement aspect by the NOS. Performance metrics used were latency, computing effort and attack recovery time.

A whitelist and blacklist IP method is used in [80] to allow or deny IP addresses. The main features of detection are IP address and packet interarrival time which can raise false positive alarms as each device is different in terms of packet transmission frequency.

Researchers in [81] present a signature-based IDS based on wired connection. It is limited to detecting HELLO flooding and version number modification attacks. Packet sending rate and signal power are the features used in detection.

A device level signature-based botnet detection system is presented in [82]. Snort and Suricata are used to test this using 3 public datasets which are ISOT [83], IoT23 [84], and BOTIoT [85]. Only known attacks are detected and the performance metrics used are number of alerts, accuracy, detection time, CPU, and memory usage. The attacks covered are TCP, UDP, ICMP and IRC.

Researchers in [86] proposed a detection mechanism that uses a predefined threshold of packets to determine attack traffic. If the total number of packets sent to an IP address exceeds the set threshold for a specific duration, this is blocked. Only TCPSYN attacks are covered in this work. This is not practical for large scale environments as false alarm rates will be high.

In [87] an entropy-based DoS/DDoS detection and mitigation system In IoT is presented. The detection metrics are source/ destination IP addresses coupled with their respective port numbers and protocols. This experiment was simulated and when the window size is increased, the switches stop responding. Another downside to this approach was the fact that the entropy is calculated in real time at the beginning of the set window or threshold. If an attack starts at the very beginning of this window, then no entropy is calculated as no prior variation to compare with is present, thus the failure to detect the attack. In [88] another entropy-based DDoS detection mechanism is brought forward with the same detection metrics and downsides as [87].

An anomaly detection model for fog empowered networks is proposed in [89]. Continuous Ranked Probability Score (CPRS) is used as the forecasting model. The number of packets passing through the network within a specified time window coupled with packet inter arrival times are the main detection metrics used. However, no clear performance metrics were discussed.

Firewalls have also been known to play a crucial role in network protection. They control network traffic based on specified rules set by an administrator or security expert [90]. Firewalls have evolved and are categorised into several generations. First generation firewalls filter packets at the network layer of the OSI model [91] considering the source and destination IP addresses and ports. The second generation extends to carrying out stateful packet inspection by having the ability to keep track of state or status of connections which in turn is used to differentiate between legitimate and malicious traffic. The third-generation firewalls go beyond stateful inspection as they can have integrated features like IPS coupled with application layer filtering. Next Generation Firewalls (NGFW) further extend into deep packet inspection coupled with the previous generations' abilities [92]. However, all these firewall generations require an expert's intervention in deployment, defining rules and checking them for correctness [90] which raises and issue for the average smart home user.

A firewall appliance for the smart home network called FANE is proposed in [90]. It operates using a Wi-Fi bridge that connects to the IoT network segment to the internet. It learns firewall rules by observing network packets of the devices at installation and whenever a new device joins, it has to go through this process. Standard firewall rules using IP tables and rate limiting are applied. However, no clear performance evaluation is provided.

A Network Based IPS is proposed in [93]. This is located behind a firewall with network-based IP sensors installed to block malicious traffic. The countermeasure mechanisms include blocking activity from the source address, dropping malicious packets and resetting the connection. However, no clear performance evaluation is provided.

In [94] a comparison between a NGFW and a traditional firewall is made. These are tested on DDoS, SQL injection and phishing attacks. TCP SYN and UDP attacks are focused on in the DDoS category. The UDP attack traffic was dropped by both firewalls due to a stateful rule that prevents UDP packets from going through a particular server and port. On the other hand, the TCP SYN traffic was allowed to pass by the traditional firewall while the IPS embedded in the NGWF blocked the traffic due to the 3-way handshake being ignored. However how subsequent legitimate traffic of the same protocol is handled was not discussed. No clear performance evaluation was provided here as well.

Another firewall-based DDoS attack mitigation is proposed in [95]. It used Manufacturer Usage description [MUD] to determine whether traffic is malicious or legitimate. The user is required to set the MUD rules for traffic flow generated by each device as well as access control lists. When the set rate limit is exceeded, the firewall drops the incoming packets. The same downsides to the work in [94] are noted here as well.

Machine learning based IDPS is presented in [96]. The models tested and their respective accuracies are: Random Forest (85.9), Decision Trees (83.8), K Nearest Neighbour (84.3), Bagging (85.8), Ada Boost (86.6) and Voting (84.8). An IPS is further proposed to take an action after the detection by dropping the malicious packets and blocking the attackers IP and MAC addresses. Another IPS alternative presented is to redirect the intruder's connection to a honeypot.

An ML embedded IDPS is presented in [97]. Three models were tested with regards to DoS detection. The models and their respective accuracies are: Random Forest (99.68), Decision Trees (99.68), Gradient Boosting (99.59). The IPS embedded then drops the malicious packets.

However, as these solutions might work well in handling certain attacks, it is not easily the case in real life smart home scenarios. Firewall and IDPS technologies are mostly suited for enterprise networks with mostly traditional IT devices [90]. The nature of the smart home network coupled with an average user's knowledge and financial constraints give rise to several challenges in making use of or deploying these technologies. These challenges include:

Knowledge and expertise constraints: An average user does not have the technical know-how when it comes to setting up firewalls and ID(P)S's. Having to define firewall rules based on the homes network traffic flow will be challenging. Another dimension is from cases where the firewall with combined ID(P)S or Machine learning features requires the smart home user to install and train each device joining the network as seen from the literature reviewed. Furthermore, having a segmented network for both smart home devices and traditional devices will be ideal [90] but an average user lacks the expertise. There is a need for usability with these security mechanisms in the average user home network without expert knowledge.

Financial constraints: Having to deploy a NGFW with combined ID(P)S will add to the cost of the smart home devices already purchased, which will discourage users due to the financial implication.

Continuous user intervention: These security mechanisms like firewalls need continuous monitoring and updates which an average home user will prefer to have this automated.

2.5.3 Supervised Machine learning based solutions for attack detection and classification

Supervised Machine Learning solutions involves training an algorithm using a labelled dataset of attack and normal traffic. The Machine Learning model then uses this to make predictions on unseen data by classifying it as attack or normal data.

Labelled datasets are required to train ML models for DDoS detection. These datasets should contain several DDoS attacks and benign traffic as well. There are numerous publicly available datasets for this purpose even though some are outdated which makes them unfit for training robust DDoS detection models which cannot handle recent attacks due to the fast-evolving nature of these attacks [54]. Table 2.2 shows some public datasets and what they comprise of.

Researchers in [98] use ML models to detect attack traffic in smart home network. Support Vector Machine (SVM), Random Forest (RF), Decision Tree (DT), Logistics Regression (LR), K-nearest neighbours (KNN), and Naive Bayes (NB) algorithm were tested. Random Forest classifier was the best performing model for detection. However specific attack type classification was not achieved. Confusion matrix was used to assess the performance.

A framework for DDoS attack detection in smart home network using ML models is presented in [99]. The attacks covered are TCPSYN, ICMP and UDP attacks. Features used in detection are TCP, UDP and ICMP distribution, packet size and count and IP diversity ratio. Only the accuracy (98%) and average latency (1.18 milli secs) were provided for performance assessment. Random Forest model was also reported to have performed well in the detection.

In [100] several ML classifiers were used to detect DDoS attacks in the smart home network. The attacks covered are TCPSYN, UDP and HTTP flood attacks. The Random Forest classifier was one of the best performing models in attack detection. Features used in detection include packet size, packet interarrival time, protocol, bandwidth, and IP addresses. Specific attack type classification has not been achieved. Confusion matrix was used to assess the performance. A similar research was carried out in [101] achieving similar results.

In [102] a DDoS defence mechanism is proposed called FLOWGUARD. It consists of an identification and classification module. Long Short-Term Memory (LSTM) was used for identification while CNN was used for classification. Both achieved high accuracy and confusion matrix was used to assess the performance. LSTM achieved 98.9% while CNN achieved 99.9% accuracy. 83 features were also used for the training.

Researchers in [103] propose a hybrid IDS that combines multiple ML models. An average accuracy of 95% was achieved. Confusion matrix was used to assess the performance. However, attack type classification was not carried out.

An IDS is presented in [104] which detects and classifies into 4 categories which are Man-In-The-Middle, DoS, replay and reconnaissance attacks. On the side of DDoS attacks, three attacks were covered which are TCP, UDP and HELLO flood attacks. Confusion matrix was used to assess the performance.

In [105] an ensemble feature selection algorithm was used to select features to detect DDoS attacks using various classifiers. Specific attack type classification was not achieved, and confusion matrix were used to assess the performance.

A clustering based semi supervised ML model for DDoS attack detection is presented in [106]. Attack type classification was not achieved, and confusion matrix was used to assess the performance.

A supervised IDS is presented in [107] which classifies attack types into 4 main categories which are DoS, evil twin, MITM and scanning attacks. Specific DDoS attack types were not classified, and confusion matrix is used in performance assessment.

From the reviewed existing works, certain gaps are evident. Specific DDoS attack type classification is not carried out which tends to be highly relevant in terms of what mitigation measure to take. In addition to that most if not all current solutions base their performance assessment on confusion matrix which does not provide relevant details like onset attack detection. This is very relevant when it comes to assessing the performance of DDoS attack solutions as how early the attack is detected matters most. Another observation is that solutions tend to be attack centric covering between 1-4 attacks (TCPSYN, UDP, ICMP, HTTP) at a time. There is a need for more robust solutions that can cover a wider range of flooding attacks at a go like DNS, NTP, ARP, TCP, HTTP, UDP, fragmentation and ICMP flooding attacks and the like.

2.6 Gaps identified and contributions

From reviewed literature, there have been numerous contributions in terms of DDoS detection in smart home networks and IoT at large. Nevertheless, there are still gaps in the approach relating to better and improved methods of DDoS traffic identification. There is lack of detailed analysis and visual comparison of attack and benign traffic patterns.

Some solutions make use of simulated data [108] [109] [110] [111] [112] [113], which might hinder the accuracy when deployed in real life scenarios. In addition, some of the approaches used are not very practical or feasible in some scenarios. For example, using the single packet inspection method to determine if it’s malicious or benign. This not only is time and resource consuming, but a less effective way of identifying DDoS patterns. This is due to DDoS flooding attacks being volume based, thus will need a volume based or cumulative approach to determine an attack pattern as opposed to the single packet approach. The approach of employing sequential user behaviour or DUD model is not very practical as the former will raise false positives when there is slight change in user pattern while the latter is not practical in large scale scenarios as it’s a device centric solution.

Name	**Format**	**Size**	**No. of Records**	**Attack Types**	**Features**	**Data Types**	**Environment**	**Publisher**	**Year**
ISOT	PCAP	1.74 GB	1.67+M unique flows	HTTP Botnet	49	Network (App layer)	Testbed	University of Victoria	2017
Bot-IoT (Used in this research)	Pcap, argus, CSV	PCAP (69.3GB) CSV (16.7GB)	72M	DoS, DDOS (TCP, UDP, HTTP), Services scan, OS Scan, Keylogging, Data ex-filtration attacks	46	Network	Testbed	UNSW Canberra Cyber	2018
N_BaloT	CSV	_	7062606	Mirai and BASHLITE (10 115 attack classes, 1 benign Class)	115	Network	Real (9 Commercial IoT Devices)	Maiden et al.	2018
Anthi Dataset	Arff	977MB	2M Malicious-Benign Ration 50-50%)	DoS,DDoS,MITM, Spoofing, Insecure Firmware, Data leakage	135	Network	Real (8 Devices)	Anthi et al.	2019
IoTID20	CSV	294MB	625784	DoS, Mirai, MITM, Scan	12	Network	_	Ontario Tech University	2020
IoT-23 (Used in this research)	Pcap, CSV	21GB 8.8GB (Lighter Ver)	_	Mirai, Tori, Gagfyt, Kenjiro, Hakai, IRCBot, Linux.Mirai, Linux.Hajmi, Muhsitk, Hide and Seek, Trojan, Okiru	21	Network (Application Layer Protocols)	Real (23 Devices)	Avast, AIC group, CTU	2020

Table 2.2 Public datasets [82]

Furthermore, some existing works tend to be unfit for large scale or diverse environments as they require training the behavioural patterns of different devices which will be time and resource consuming [8]. This also raises the issue of having user, attack, or device centric solutions due to the type of network features used in detection which are very user or device dependent. More uniform and generalized approaches are needed which will cater for all regardless of the user or device.

Light weight solutions are also needed. From literature we can see that some solutions use too many features for detection which leads to delayed detection times. A reduced number of features which at the same time are effective in attack detection need to be investigated.

The issue of assessing a systems performance based on confusion matrices and other conventional statistics also come into play. Confusion matrix does not specify how early the attack is detected or classified rather it gives statistics on how much the solution was able to predict right. The narrative from using conventional methods like confusion matrices and other statistics in terms of performance assessment as they do not provide relevant information as to how early and accurate the system was able to detect or classify an attack which is very crucial in the field of DDoS attacks needs to be changed. The effect is what you want to mitigate as early as possible not how much the attack is guessed right down the line when much of the damage has been done.

Another challenging issue is dealing with unfamiliar attacks. Existing works don't seem to have gotten around detecting and classifying unfamiliar attacks due to their unpredictive nature. An Intelligent way to handle this kind of attack is needed.

To design an efficient and effective DDoS detection and classification system, there is a need to have an in-depth understanding with regards to the attack pattern and network changes that occur during the attack and in the process monitoring the most affected network properties. These can be used as a baseline for attack identification. To bridge the above-mentioned gaps, this research will carry out the following:

- EDA on the normal and attack network characteristics in a real smart home environment and propose a better way to identify a DDoS pattern based on the analysed network features.
- Identify the most affected network features during a DDoS flooding attack and capitalize on them for a timely and more effective solution
- Design and implement a detection and classification solution that is light weight, not user, attack, or device centric and practical. This will cover all DDoS flooding attacks including unfamiliar attacks.
- A better way to assess DDoS detection and classification systems performance which will provide details of how early and accurate a system is able to detect and classify an attack.
- A more robust supervised machine learning model that covers a wide range of attacks including unfamiliar ones.

2.7 Summary

This chapter has reviewed the current state of the art in terms of DDoS detection and classification in smart home networks. The different approaches used to combat this attack has been researched as well as the existing gaps that need to be bridged. Areas have been identified which this research aims to contribute to in terms of bridging the gaps identified.

Chapter 3

Smart home network behaviour

3.1 Introduction

This chapter delves into the network behaviour of the smart home devices used in this research. The chosen brand is Hive home [114] due to the lack of attention from the research community despite being one of the most patronized smart home brands in the UK. An in-depth study of the Hive home devices is carried out in this chapter to have a clear understanding of how these devices behave normally, as this will be used as a baseline for attack detection in subsequent chapters. The behavioural pattern of the devices both independently and collectively as an IoT network in connection to user behaviour is studied and analysed as these observations and analyses will be used to tailor security protocols that will suit the IoT network in question. Exploratory Data Analysis (EDA) is the technique used to analyse and visualize the collected hive home dataset, which led to some interesting novel findings.

In terms of device behaviours and characteristics, this chapter covers the following:

- Collection of data from a real-life Hive home network due to unavailability of Hive dataset in existing works.
- EDA on the collected logs visualizing the behaviour of these devices covering the following aspects: flow volume, flow duration, protocols, traffic categorization, device identification, varying flow volumes and duration based on device mode of control (manual, automated, Hive app, home kit app, Google home app) and the distinct traffic pattern that applies to each of the mentioned device modes of control.
- Discussion of new findings based on results derived from the EDA with regards to device behaviour when certain triggers are applied.

The sub-sections in this chapter covers the methodology used, network setup, data collection process, EDA on the collected data, new findings, and comparison with literature and finally a summary of the chapter.

3.2 Methodology

The methodology used in this chapter is broken down and explained in this section. It has four phases, which are network setup, data collection, Exploratory Data Analysis, and reporting.

The network setup phase involves getting the required tools, designing the network topology, and making the smart home network connection. The next phase is data collection where the use cases for data collection were outlined. This includes use cases like collecting data from each individual device and together as a collective, triggering the devices at certain times using specific modes (manual, scheduled, application) and monitoring the devices in different states like idle and active. Traffic generated from the various use cases are sniffed using Wireshark. The EDA phase entails data pre-processing, network feature selection, normalization, and visualization of results. The last phase is the reporting phase where the findings are documented, compared with literature and new findings discussed.

3.3 Network setup

Hive home smart devices were used for this study. The devices include a smart hub (to integrate the smart devices), a motion sensor, a smart plug, and a smart bulb. The network communication that takes place when these devices are both idle and active is the main point of interest, thus a setup to collect this network data for further analysis was carried out. Hardware and software tools used are listed as follows:

- Samsung A12 smart phone
- Netgear GS308E – 100NAS switch [115]
- Mac book air OS X El Capitan 10.11.6
- Jupyter Notebook [116]
- TL-WR940N Router [117]
- iPhone SE
- iPad
- LAN cable
- Wireshark 2.6.0
- Hive starter pack (motion sensor, plug, Bulb, hub) [114]
- Hive home app v.10.44.0 (6)
- Google home app v.2.42.120
- Home kit app 14.4.2

Traffic generated from/to each of the mentioned devices was captured separately to know the type of network traffic that relates to a particular device. To get very detailed network traffic, the capture setup was made to collect traffic at layer 2 (datalink). This was done by connecting the hub to port 1 of the switch. Port 8 of the switch was then connected to the router (for internet connection). To capture all that flowed in and out of the hub and all devices paired to it, port 1 was mirrored on port 4. Port 4 was connected to the laptop using a Local Area Network (LAN) cable and Wireshark was used to capture this traffic. This capture setup is depicted in Figure. 3.1.

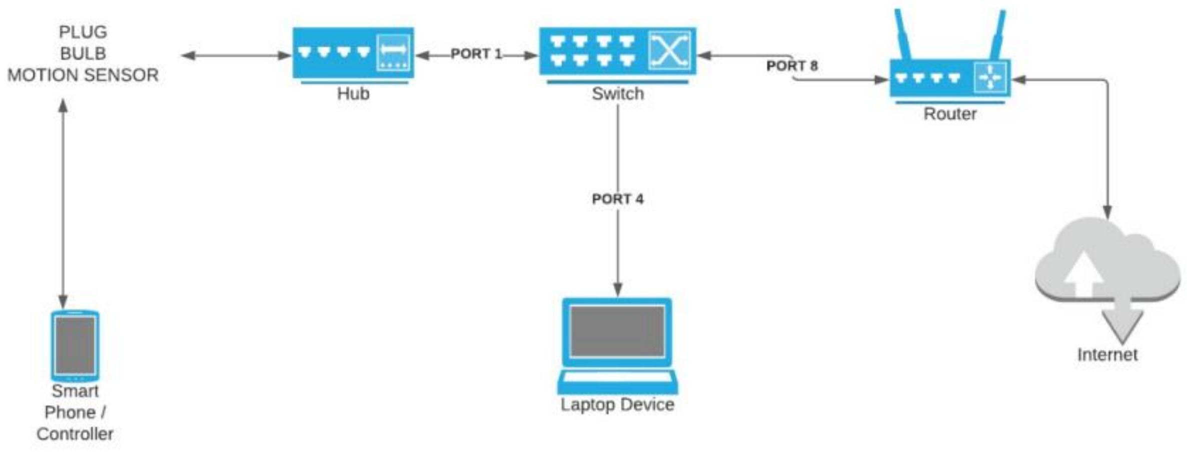


Figure 3.1 Data collection connectivity

3.4 Data collection

Before the commencement of collecting traffic, use cases were drafted for specific scenarios. This will give a clear understanding of the kind of traffic that gets generated by these devices for every scenario. The parameters that come with each use case include:

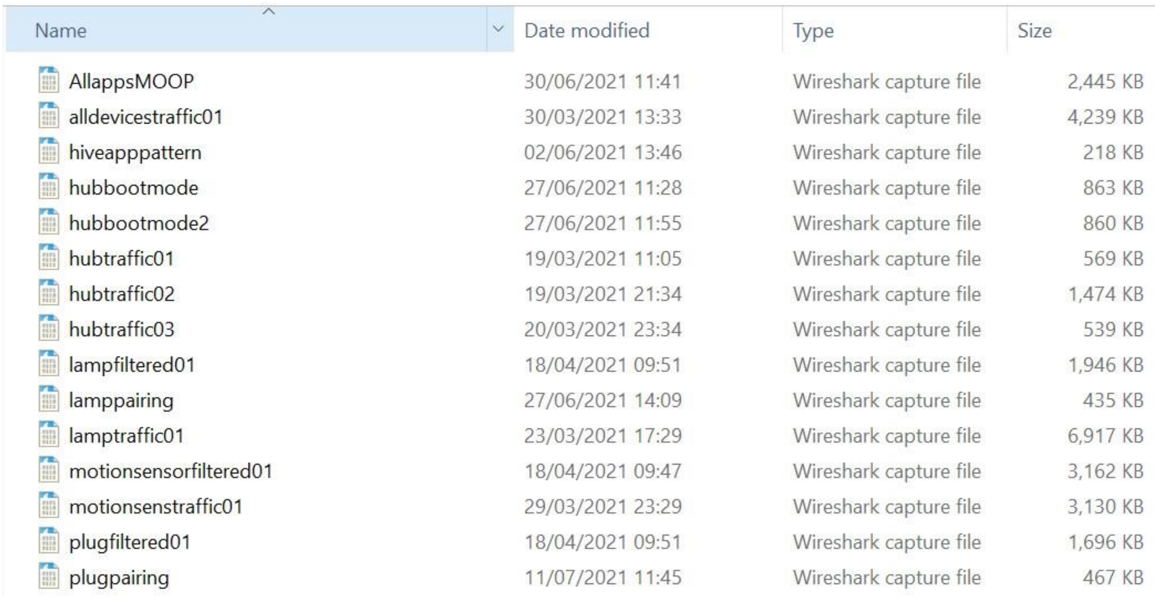
- Device: This indicates what device or devices are being monitored. This can be a single device isolated or connected to other devices. For instance, there is a use case which monitors only the smart hub to know how this device behaves in solo and there is also a use case which monitors the smart hub, motion sensor, smart plug and bulb connected to get the type of traffic generated when the devices are working together.
- Mode of control: This refers to the means used to operate or control the device(s). Five different modes were studied which include:
  - Using the Hive proprietary app: The Hive app was used to operate these devices after downloading it on the control devices (iPhone, iPad, Samsung smart phone).
  - Using Google home app: The Google home app was used to operate these devices after downloading it on control devices.
  - Using home kit: This app comes preloaded on apple devices (iPhone, iPad). This app is compatible with Hive devices just like the Google home app. This was also used to operate the hive devices.
  - Manually: The devices (plug and smart bulb) were controlled by physical means by turning their switch ON and OFF.
  - Scheduled: Using the Hive app, times when the devices should automatically go ON and OFF were set on the app. This automated the triggers without any manual intervention either physically or via the apps. In this auto mode the hive devices go ON/OFF at the desired pre-set times.

- Trigger: This refers to the trigger action used for each particular use case. This can be turning the device ON, OFF, brightness UP/DOWN, physically triggering motion sensor and the like.
- State: This indicates whether the device is in active or idle state. Active state is when the device is ON from the switch and triggered like plug ON or bulb brightness UP/DOWN while idle state is when the device is in the OFF state but still ON from the switch.
- Window duration: This refers to how long a particular use case is being monitored as well as capturing the traffic relating to that particular use case.
- Mode: This indicates whether the device is pairing, unpairing or at boot stage.
- Count: This indicates the number of times a particular trigger like ON/OFF has been executed within a particular window duration.

Table 3.1 shows some of these use cases. Figure 3.2 shows some of the Wireshark file captures. Figure 3.3 shows some of the MOOP Wireshark captures.

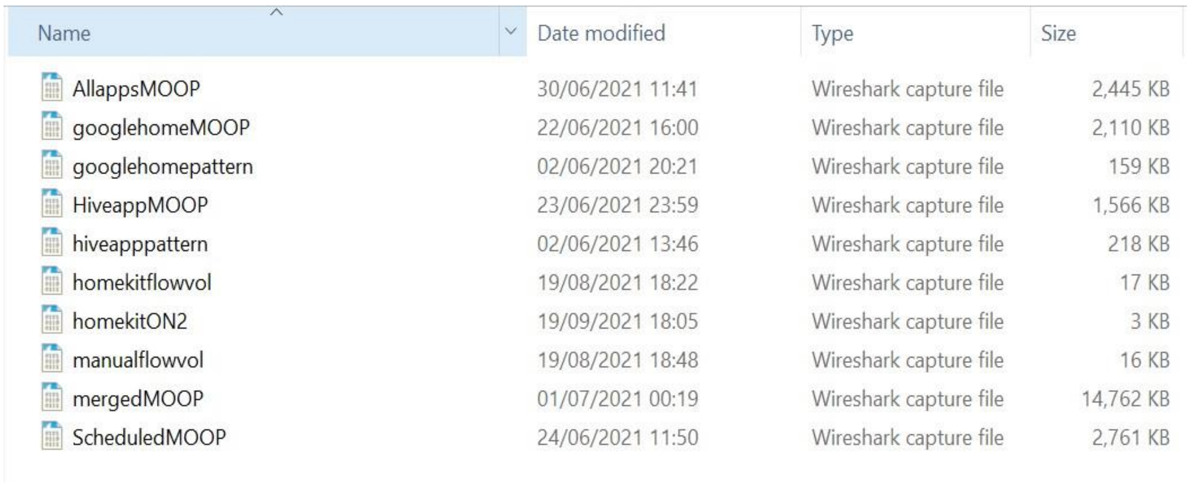
Table 3.1 Use cases scenarios

ID	Device(s)	Mode of operation	Trigger	State	Duration	Mode	Count
1	Hub	Manual	ON	Idle	2 hours	Boot	3
2	Hub			Idle	8 hours		
3	Hub + plug	Hive app		Idle	30 minutes	Pairing	1
4	Hub + plug	Hive app		Idle	30 minutes	Unpairing	1
5	Hub + plug			Idle	2 hours	Paired	
6	Hub + plug	Manual	ON	Active	2 hours	Paired	4
7	Hub + plug	Manual	OFF	Active-idle	2 hours	Paired	4
8	Hub + plug	Hive app	ON	Active	2 hours	Paired	4
9	Hub + plug	Hive app	OFF	Active-idle	2 hours	paired	4
10	Hub + plug	Home kit	ON	Active	2 hours	Paired	4
11	Hub + plug	Home kit	OFF	Active-idle	2 hours	Paired	4
12	Hub + plug	Google home	ON	Active	2 hours	Paired	4
13	Hub + plug	Google home	OFF	Active-idle	2 hours	Paired	4
14	Hub + plug	Scheduled	ON from 6-8pm	Active	2 hours	Paired	1
15	Hub + plug	Scheduled	OFF from 8:30-9pm	Active-idle	1 hour	Paired	1
16	Hub + Motion Sensor + bulb + plug	Scheduled	If motion detected, ON bulb for 10 minutes and plug for 5 minutes.	Active	5 hours	Paired	4
17	Hub + bulb	Hive app	Brightness UP	Active	1 hour	Paired	3
18	Hub + bulb	Hive app	Brightness DOWN	Active	1 hour	Paired	3
19	Hub + motion sensor	Scheduled	Move to perimeter	Active	5 hours	Paired	10
20	Hub + Motion Sensor + bulb + plug	Scheduled	If motion detected, ON bulb and plug for 1 hour		8 hours		



Name	Date modified	Type	Size
AllappsMOOP	30/06/2021 11:41	Wireshark capture file	2,445 KB
alldevicestraffic01	30/03/2021 13:33	Wireshark capture file	4,239 KB
hiveapppattern	02/06/2021 13:46	Wireshark capture file	218 KB
hubbootmode	27/06/2021 11:28	Wireshark capture file	863 KB
hubbootmode2	27/06/2021 11:55	Wireshark capture file	860 KB
hubtraffic01	19/03/2021 11:05	Wireshark capture file	569 KB
hubtraffic02	19/03/2021 21:34	Wireshark capture file	1,474 KB
hubtraffic03	20/03/2021 23:34	Wireshark capture file	539 KB
lampfiltered01	18/04/2021 09:51	Wireshark capture file	1,946 KB
lamppairing	27/06/2021 14:09	Wireshark capture file	435 KB
lamptraffic01	23/03/2021 17:29	Wireshark capture file	6,917 KB
motionsensorfiltered01	18/04/2021 09:47	Wireshark capture file	3,162 KB
motionsenstraffic01	29/03/2021 23:29	Wireshark capture file	3,130 KB
plugfiltered01	18/04/2021 09:51	Wireshark capture file	1,696 KB
plugpairing	11/07/2021 11:45	Wireshark capture file	467 KB

Figure 3.2 Wireshark captures for separate devices

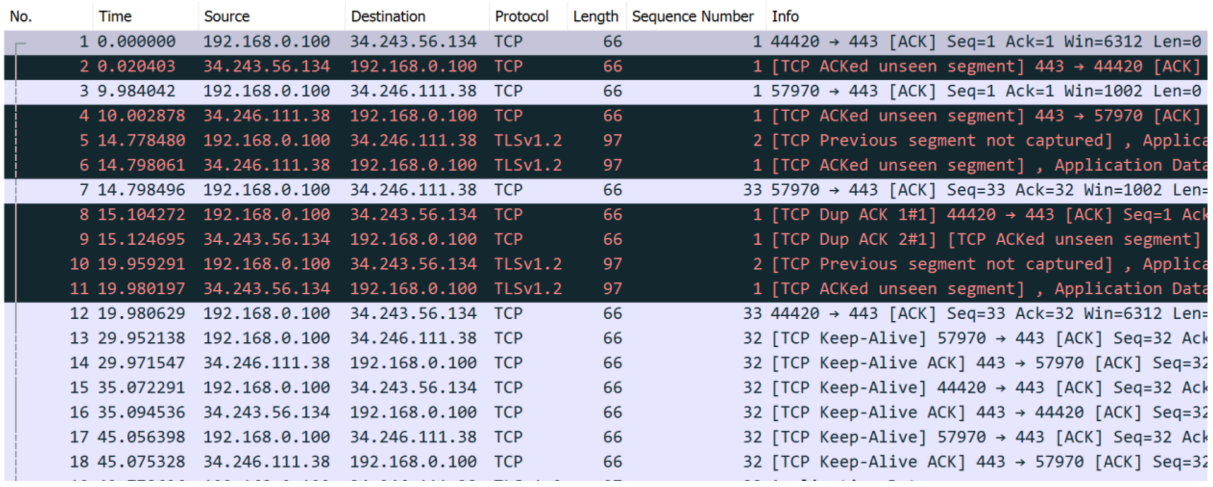


Name	Date modified	Type	Size
AllappsMOOP	30/06/2021 11:41	Wireshark capture file	2,445 KB
googlehomeMOOP	22/06/2021 16:00	Wireshark capture file	2,110 KB
googlehomepattern	02/06/2021 20:21	Wireshark capture file	159 KB
HiveappMOOP	23/06/2021 23:59	Wireshark capture file	1,566 KB
hiveapppattern	02/06/2021 13:46	Wireshark capture file	218 KB
homekitflowvol	19/08/2021 18:22	Wireshark capture file	17 KB
homekitON2	19/09/2021 18:05	Wireshark capture file	3 KB
manualflowvol	19/08/2021 18:48	Wireshark capture file	16 KB
mergedMOOP	01/07/2021 00:19	Wireshark capture file	14,762 KB
ScheduledMOOP	24/06/2021 11:50	Wireshark capture file	2,761 KB

Figure 3.3 MOOP Wireshark captures

Before the execution of each use case, Wireshark is connected to sniff the traffic. This lasts until the window duration has elapsed. This is then saved in the default Wireshark extension as a pcap file which is later converted to csv for further analysis if needed.

Figure 3.4 shows a sample of the raw data capture. This shows the packet header details like the source and destination IP addresses between the smart devices and the external servers they communicate with. The time stamps, protocols utilized, packet lengths and sequence number utilized by each packet are shown as well. The info column shows additional details like the TCP flags.



No.	Time	Source	Destination	Protocol	Length	Sequence Number	Info
1	0.000000	192.168.0.100	34.243.56.134	TCP	66	1	44420 → 443 [ACK] Seq=1 Ack=1 Win=6312 Len=0
2	0.020403	34.243.56.134	192.168.0.100	TCP	66	1	[TCP ACKed unseen segment] 443 → 44420 [ACK]
3	9.984042	192.168.0.100	34.246.111.38	TCP	66	1	57970 → 443 [ACK] Seq=1 Ack=1 Win=1002 Len=0
4	10.002878	34.246.111.38	192.168.0.100	TCP	66	1	[TCP ACKed unseen segment] 443 → 57970 [ACK]
5	14.778480	192.168.0.100	34.246.111.38	TLSv1.2	97	2	[TCP Previous segment not captured] , Applica
6	14.798061	34.246.111.38	192.168.0.100	TLSv1.2	97	1	[TCP ACKed unseen segment] , Application Data
7	14.798496	192.168.0.100	34.246.111.38	TCP	66	33	57970 → 443 [ACK] Seq=33 Ack=32 Win=1002 Len=
8	15.104272	192.168.0.100	34.243.56.134	TCP	66	1	[TCP Dup ACK 1#1] 44420 → 443 [ACK] Seq=1 Ack
9	15.124695	34.243.56.134	192.168.0.100	TCP	66	1	[TCP Dup ACK 2#1] [TCP ACKed unseen segment]
10	19.959291	192.168.0.100	34.243.56.134	TLSv1.2	97	2	[TCP Previous segment not captured] , Applica
11	19.980197	34.243.56.134	192.168.0.100	TLSv1.2	97	1	[TCP ACKed unseen segment] , Application Data
12	19.980629	192.168.0.100	34.243.56.134	TCP	66	33	44420 → 443 [ACK] Seq=33 Ack=32 Win=6312 Len=
13	29.952138	192.168.0.100	34.246.111.38	TCP	66	32	[TCP Keep-Alive] 57970 → 443 [ACK] Seq=32 Ack
14	29.971547	34.246.111.38	192.168.0.100	TCP	66	32	[TCP Keep-Alive ACK] 443 → 57970 [ACK] Seq=32
15	35.072291	192.168.0.100	34.243.56.134	TCP	66	32	[TCP Keep-Alive] 44420 → 443 [ACK] Seq=32 Ack
16	35.094536	34.243.56.134	192.168.0.100	TCP	66	32	[TCP Keep-Alive ACK] 443 → 44420 [ACK] Seq=32
17	45.056398	192.168.0.100	34.246.111.38	TCP	66	32	[TCP Keep-Alive] 57970 → 443 [ACK] Seq=32 Ack
18	45.075328	34.246.111.38	192.168.0.100	TCP	66	32	[TCP Keep-Alive ACK] 443 → 57970 [ACK] Seq=32

Figure 3.4 Raw data capture sample

3.5 Exploratory Data Analysis

This section delves into the EDA of the unencrypted collected logs. EDA is a statistical method of analysing data to summarize the main characteristics of the dataset by using data visualization tools and techniques to represent the results derived for ease of understanding. Jupyter notebook is the tool used for this purpose [116]. The areas covered were chosen for specific reasons. Traffic categorization was covered to have a holistic view of the kind of traffic Hive devices exchange. This will help in addressing strange and malicious traffic. Device identification was carried out to know the fingerprint of each device to be able to identify Hive devices in a pool of other IoT. Protocols and flow volume and duration were studied as these aspects are used in propagating DDoS attacks. Studying them will help in knowing the normal Hive traffic pattern when it comes to protocol sequence, flow volume and duration in comparison to malicious use of them in flooding attacks as we will see in subsequent chapters. This section is further divided into subsections addressing traffic categorization, device identification, protocols in both Idle and active states, flow volume (total number of incoming and outgoing bytes in one cycle) and flow duration (time it takes from the beginning of a flow to the end) and traffic pattern distinct to each mode of operation.

3.5.1 Traffic categorization

Traffic collected from this network was broadly categorized into 3 after analysis of the executed use cases. These categories are as follows:

- Periodic queries: These queries were found to take place automatically regardless of an event trigger ranging from every few minutes to some hours depending on the protocol or device. The hub was studied without pairing any device to it to capture the network activity that takes place in its lone state. This was repeated with devices (plug, lamp, motion sensor) paired to the hub to identify what happens differently in this scenario. Figure 3.5 shows this periodic activity originating from the hub without any device paired to it compared to when a device is paired to the hub over a period of 2 hours. As seen from Figure 3.5 there is more frequent DNS, TCP and TLS activity happening when a device is paired to the hub as opposed to when the hub is on its own. Figure 3.6 – 3.10 shows a raw data capture of this extended protocol activity comparing the hubs protocol activity to when it is connected to a motion sensor in one minute window. We can see that the protocols in figure 3.6 are much less than those in figure 3.7 – 3.10 due to the motion sensors presence.

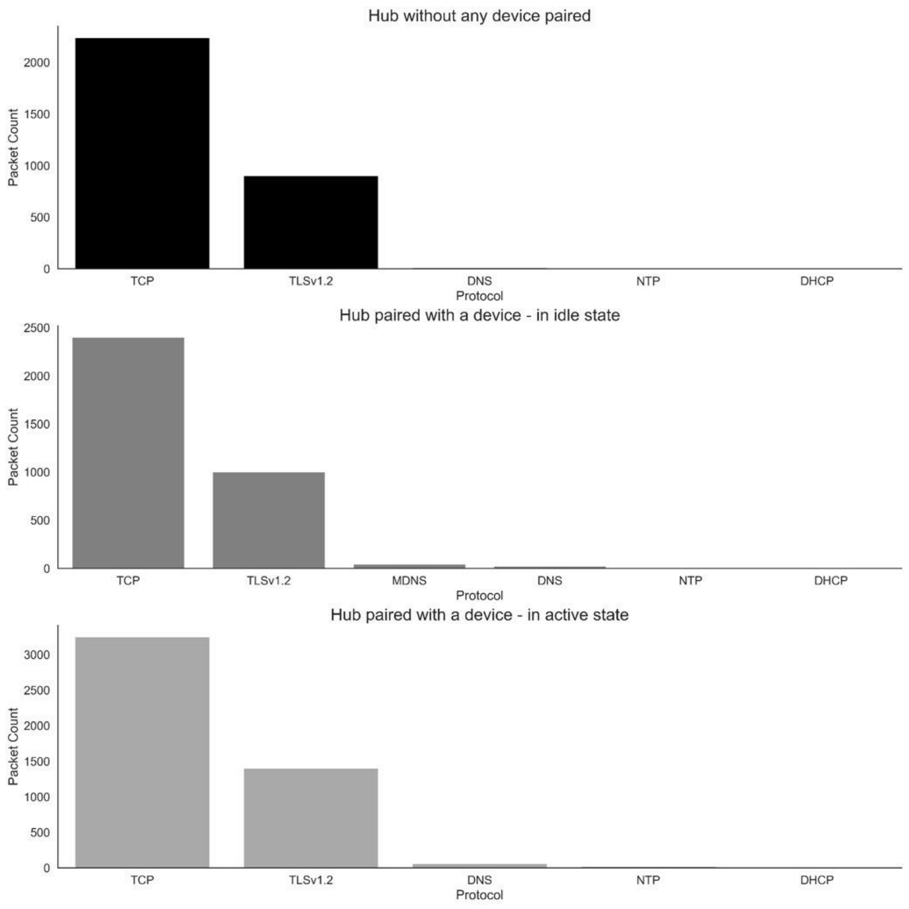
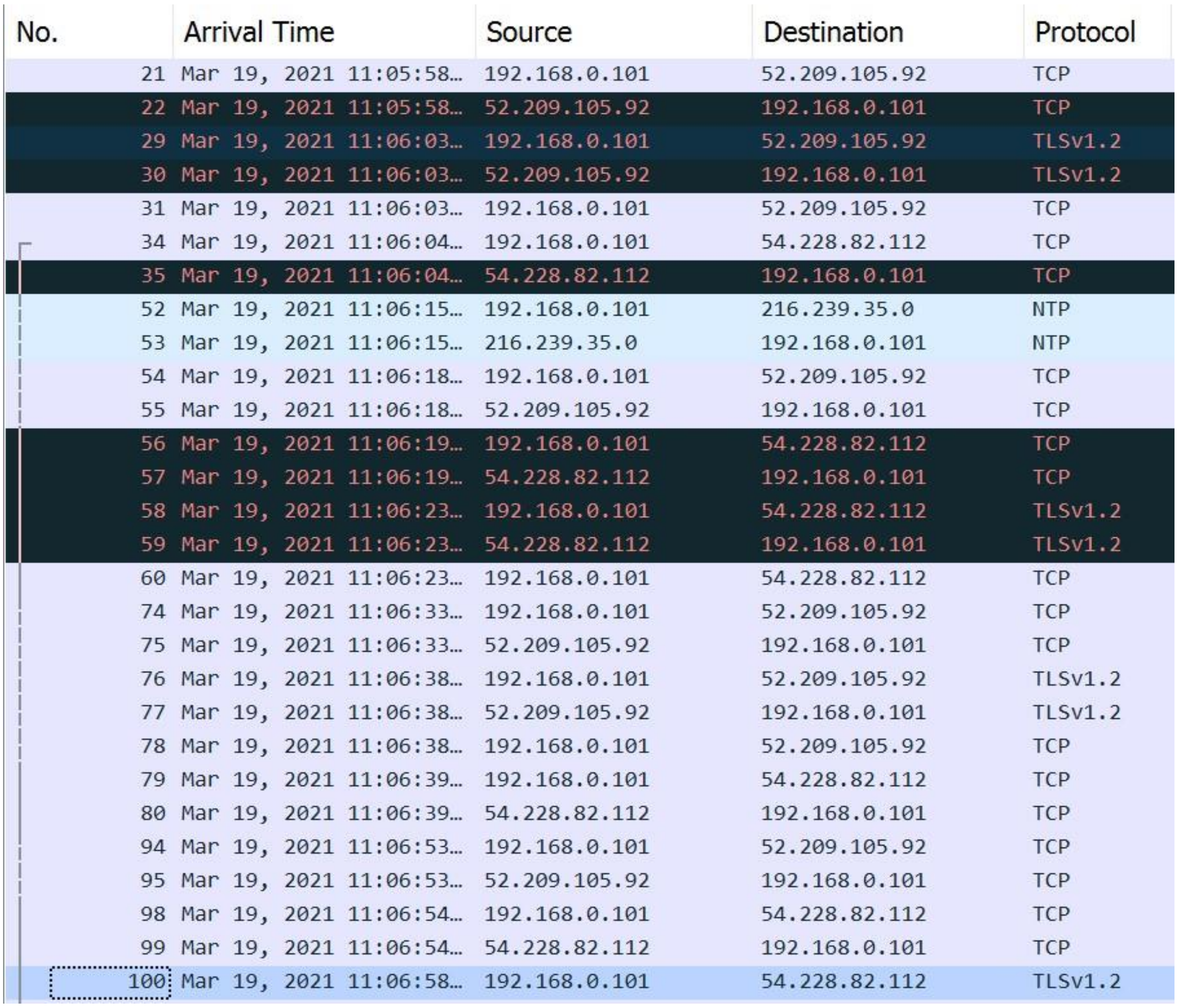


Figure 3.5 Protocol count compared by device state

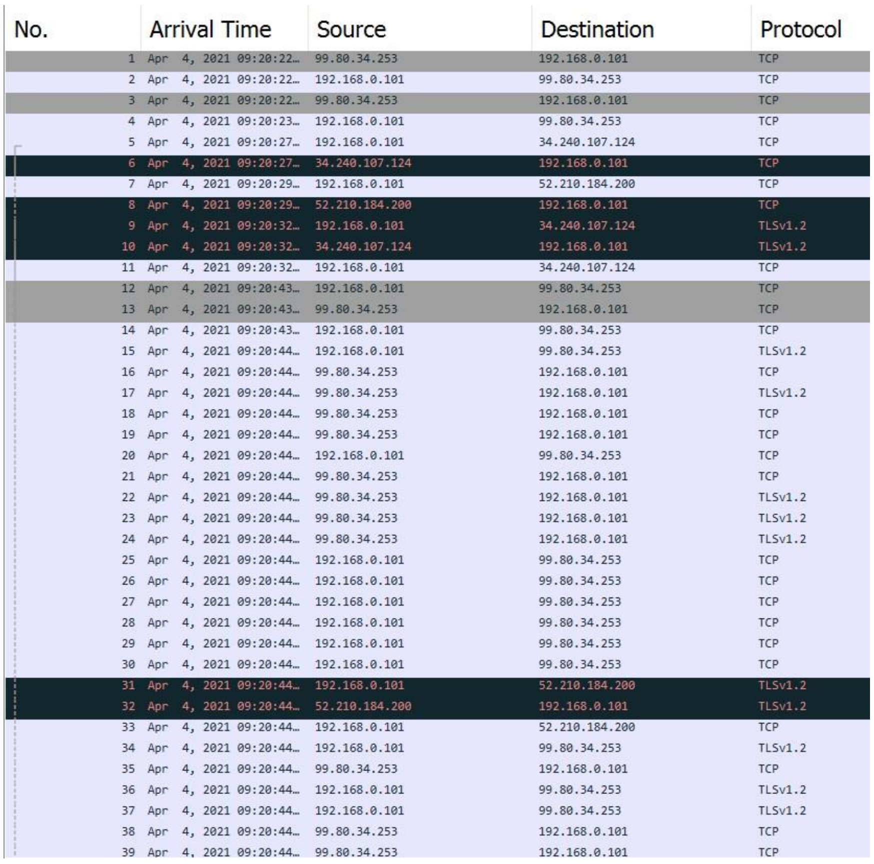
- Event trigger: This kind of traffic gets generated whenever an event is triggered. For instance, when the plug or lamp goes ON or OFF or the motion sensor detects movement. This results in generation of DNS, TCP, and TLS packets. In some cases, mDNS traffic is also generated depending on the mode of operation used to trigger the event. An example is shown in figure 3.11 when the mode of operation is from a mobile phone via an app as compared to a manual trigger shown in figure 3.12 where no mDNS protocol appears. We can see mDNS protocol as the very first under the protocol column with the mobile phones IP address in the source address column as 192.168.0.102 with the DNS query appearing at the very bottom of the protocol column showing the smart hub source address as 192.168.0.101. However, with the manual trigger, only the DNS query appears with the same hub IP source address.

- Boot, pairing and firmware updates: This traffic is generated whenever the hub is in boot mode, when paring with the devices or a firmware update takes place. A certain number of DNS servers are communicated with when these take place. Figure 3.13 shows some DNS servers that are being queried during boot mode in the info column with the hub IP address being 192.168.0.103 in source address column while figure 3.14 shows some DNS servers being queried during pairing mode with the hub/plug IP address being 192.168.0.101 as the source.



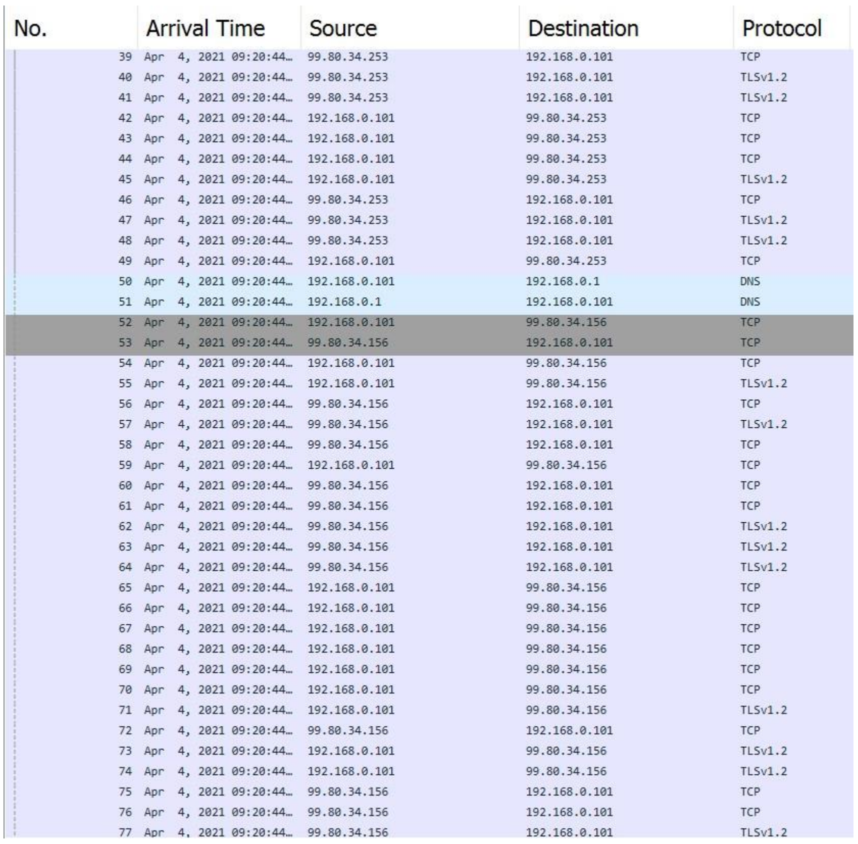
No.	Arrival Time	Source	Destination	Protocol
21	Mar 19, 2021 11:05:58…	192.168.0.101	52.209.105.92	TCP
22	Mar 19, 2021 11:05:58…	52.209.105.92	192.168.0.101	TCP
29	Mar 19, 2021 11:06:03…	192.168.0.101	52.209.105.92	TLSv1.2
30	Mar 19, 2021 11:06:03…	52.209.105.92	192.168.0.101	TLSv1.2
31	Mar 19, 2021 11:06:03…	192.168.0.101	52.209.105.92	TCP
34	Mar 19, 2021 11:06:04…	192.168.0.101	54.228.82.112	TCP
35	Mar 19, 2021 11:06:04…	54.228.82.112	192.168.0.101	TCP
52	Mar 19, 2021 11:06:15…	192.168.0.101	216.239.35.0	NTP
53	Mar 19, 2021 11:06:15…	216.239.35.0	192.168.0.101	NTP
54	Mar 19, 2021 11:06:18…	192.168.0.101	52.209.105.92	TCP
55	Mar 19, 2021 11:06:18…	52.209.105.92	192.168.0.101	TCP
56	Mar 19, 2021 11:06:19…	192.168.0.101	54.228.82.112	TCP
57	Mar 19, 2021 11:06:19…	54.228.82.112	192.168.0.101	TCP
58	Mar 19, 2021 11:06:23…	192.168.0.101	54.228.82.112	TLSv1.2
59	Mar 19, 2021 11:06:23…	54.228.82.112	192.168.0.101	TLSv1.2
60	Mar 19, 2021 11:06:23…	192.168.0.101	54.228.82.112	TCP
74	Mar 19, 2021 11:06:33…	192.168.0.101	52.209.105.92	TCP
75	Mar 19, 2021 11:06:33…	52.209.105.92	192.168.0.101	TCP
76	Mar 19, 2021 11:06:38…	192.168.0.101	52.209.105.92	TLSv1.2
77	Mar 19, 2021 11:06:38…	52.209.105.92	192.168.0.101	TLSv1.2
78	Mar 19, 2021 11:06:38…	192.168.0.101	52.209.105.92	TCP
79	Mar 19, 2021 11:06:39…	192.168.0.101	54.228.82.112	TCP
80	Mar 19, 2021 11:06:39…	54.228.82.112	192.168.0.101	TCP
94	Mar 19, 2021 11:06:53…	192.168.0.101	52.209.105.92	TCP
95	Mar 19, 2021 11:06:53…	52.209.105.92	192.168.0.101	TCP
98	Mar 19, 2021 11:06:54…	192.168.0.101	54.228.82.112	TCP
99	Mar 19, 2021 11:06:54…	54.228.82.112	192.168.0.101	TCP
100	Mar 19, 2021 11:06:58…	192.168.0.101	54.228.82.112	TLSv1.2

Figure 3.6 Hub protocol frequency



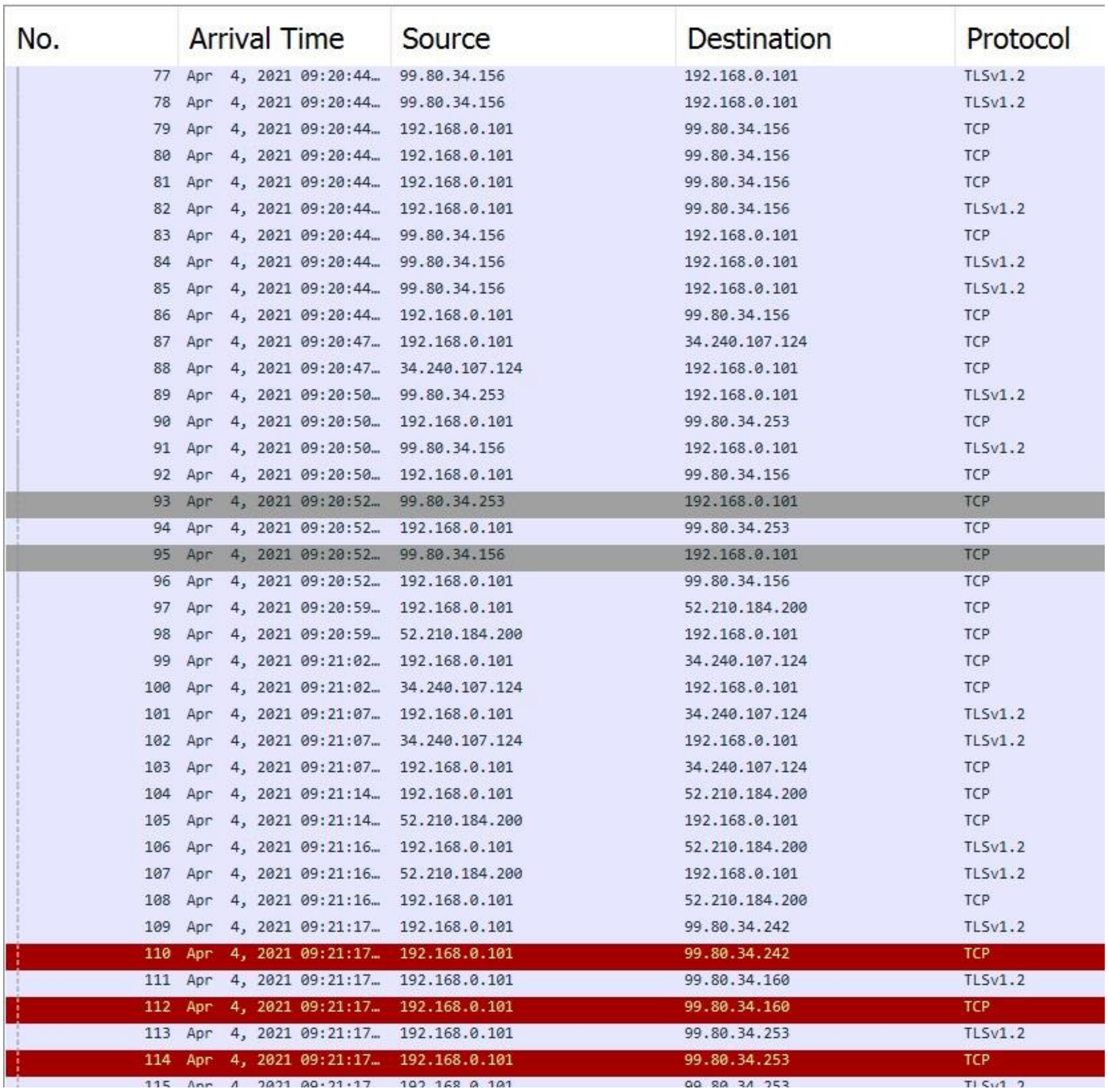
No.	Arrival Time	Source	Destination	Protocol
1	Apr 4, 2021 09:20:22…	99.80.34.253	192.168.0.101	TCP
2	Apr 4, 2021 09:20:22…	192.168.0.101	99.80.34.253	TCP
3	Apr 4, 2021 09:20:22…	99.80.34.253	192.168.0.101	TCP
4	Apr 4, 2021 09:20:23…	192.168.0.101	99.80.34.253	TCP
5	Apr 4, 2021 09:20:27…	192.168.0.101	34.240.107.124	TCP
6	Apr 4, 2021 09:20:27…	34.240.107.124	192.168.0.101	TCP
7	Apr 4, 2021 09:20:29…	192.168.0.101	52.210.184.200	TCP
8	Apr 4, 2021 09:20:29…	52.210.184.200	192.168.0.101	TCP
9	Apr 4, 2021 09:20:32…	192.168.0.101	34.240.107.124	TLSv1.2
10	Apr 4, 2021 09:20:32…	34.240.107.124	192.168.0.101	TLSv1.2
11	Apr 4, 2021 09:20:32…	192.168.0.101	34.240.107.124	TCP
12	Apr 4, 2021 09:20:43…	192.168.0.101	99.80.34.253	TCP
13	Apr 4, 2021 09:20:43…	99.80.34.253	192.168.0.101	TCP
14	Apr 4, 2021 09:20:43…	192.168.0.101	99.80.34.253	TCP
15	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TLSv1.2
16	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
17	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
18	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
19	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
20	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
21	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
22	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
23	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
24	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
25	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
26	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
27	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
28	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
29	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
30	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
31	Apr 4, 2021 09:20:44…	192.168.0.101	52.210.184.200	TLSv1.2
32	Apr 4, 2021 09:20:44…	52.210.184.200	192.168.0.101	TLSv1.2
33	Apr 4, 2021 09:20:44…	192.168.0.101	52.210.184.200	TCP
34	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TLSv1.2
35	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
36	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TLSv1.2
37	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TLSv1.2
38	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
39	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP

Figure 3.7 Hub + motion sensor protocol frequency



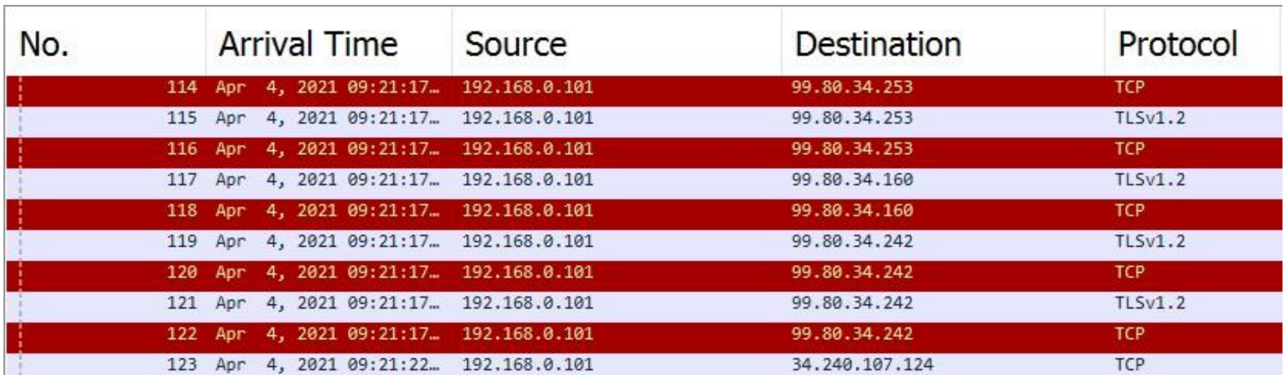
No.	Arrival Time	Source	Destination	Protocol
39	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
40	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
41	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
42	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
43	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
44	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
45	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TLSv1.2
46	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TCP
47	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
48	Apr 4, 2021 09:20:44…	99.80.34.253	192.168.0.101	TLSv1.2
49	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.253	TCP
50	Apr 4, 2021 09:20:44…	192.168.0.101	192.168.0.1	DNS
51	Apr 4, 2021 09:20:44…	192.168.0.1	192.168.0.101	DNS
52	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
53	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
54	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
55	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TLSv1.2
56	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
57	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
58	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
59	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
60	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
61	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
62	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
63	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
64	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
65	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
66	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
67	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
68	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
69	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
70	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
71	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TLSv1.2
72	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
73	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TLSv1.2
74	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TLSv1.2
75	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
76	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
77	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2

Figure 3.8 Hub + motion sensor protocol frequency



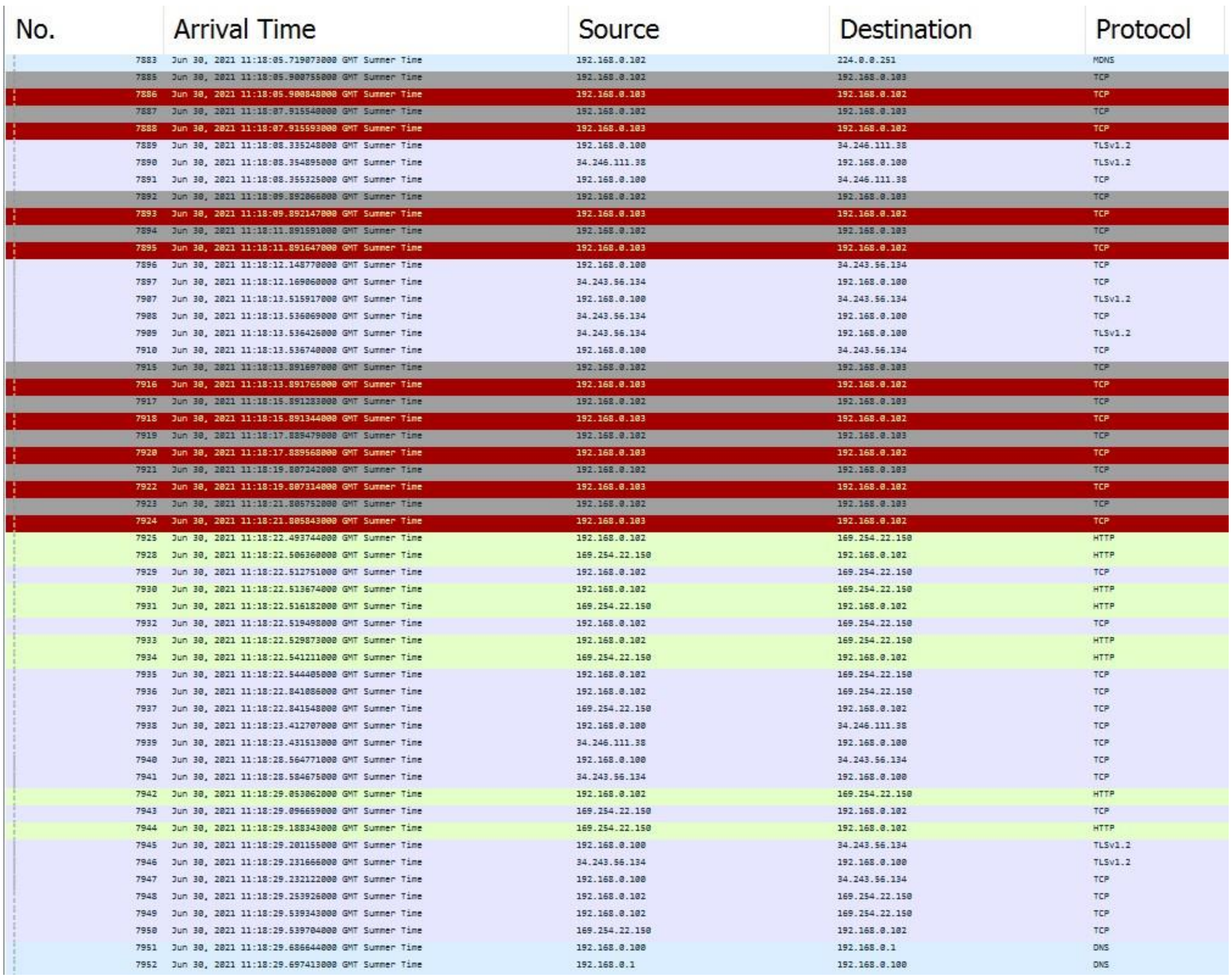
No.	Arrival Time	Source	Destination	Protocol
77	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
78	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
79	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
80	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
81	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
82	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TLSv1.2
83	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TCP
84	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
85	Apr 4, 2021 09:20:44…	99.80.34.156	192.168.0.101	TLSv1.2
86	Apr 4, 2021 09:20:44…	192.168.0.101	99.80.34.156	TCP
87	Apr 4, 2021 09:20:47…	192.168.0.101	34.240.107.124	TCP
88	Apr 4, 2021 09:20:47…	34.240.107.124	192.168.0.101	TCP
89	Apr 4, 2021 09:20:50…	99.80.34.253	192.168.0.101	TLSv1.2
90	Apr 4, 2021 09:20:50…	192.168.0.101	99.80.34.253	TCP
91	Apr 4, 2021 09:20:50…	99.80.34.156	192.168.0.101	TLSv1.2
92	Apr 4, 2021 09:20:50…	192.168.0.101	99.80.34.156	TCP
93	Apr 4, 2021 09:20:52…	99.80.34.253	192.168.0.101	TCP
94	Apr 4, 2021 09:20:52…	192.168.0.101	99.80.34.253	TCP
95	Apr 4, 2021 09:20:52…	99.80.34.156	192.168.0.101	TCP
96	Apr 4, 2021 09:20:52…	192.168.0.101	99.80.34.156	TCP
97	Apr 4, 2021 09:20:59…	192.168.0.101	52.210.184.200	TCP
98	Apr 4, 2021 09:20:59…	52.210.184.200	192.168.0.101	TCP
99	Apr 4, 2021 09:21:02…	192.168.0.101	34.240.107.124	TCP
100	Apr 4, 2021 09:21:02…	34.240.107.124	192.168.0.101	TCP
101	Apr 4, 2021 09:21:07…	192.168.0.101	34.240.107.124	TLSv1.2
102	Apr 4, 2021 09:21:07…	34.240.107.124	192.168.0.101	TLSv1.2
103	Apr 4, 2021 09:21:07…	192.168.0.101	34.240.107.124	TCP
104	Apr 4, 2021 09:21:14…	192.168.0.101	52.210.184.200	TCP
105	Apr 4, 2021 09:21:14…	52.210.184.200	192.168.0.101	TCP
106	Apr 4, 2021 09:21:16…	192.168.0.101	52.210.184.200	TLSv1.2
107	Apr 4, 2021 09:21:16…	52.210.184.200	192.168.0.101	TLSv1.2
108	Apr 4, 2021 09:21:16…	192.168.0.101	52.210.184.200	TCP
109	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TLSv1.2
110	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TCP
111	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.160	TLSv1.2
112	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.160	TCP
113	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.253	TLSv1.2
114	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.253	TCP
115	Apr 4, 2021 09:21:17	192.168.0.101	99.80.34.253	TLSv1.2

Figure 3.9 Hub + motion sensor protocol frequency



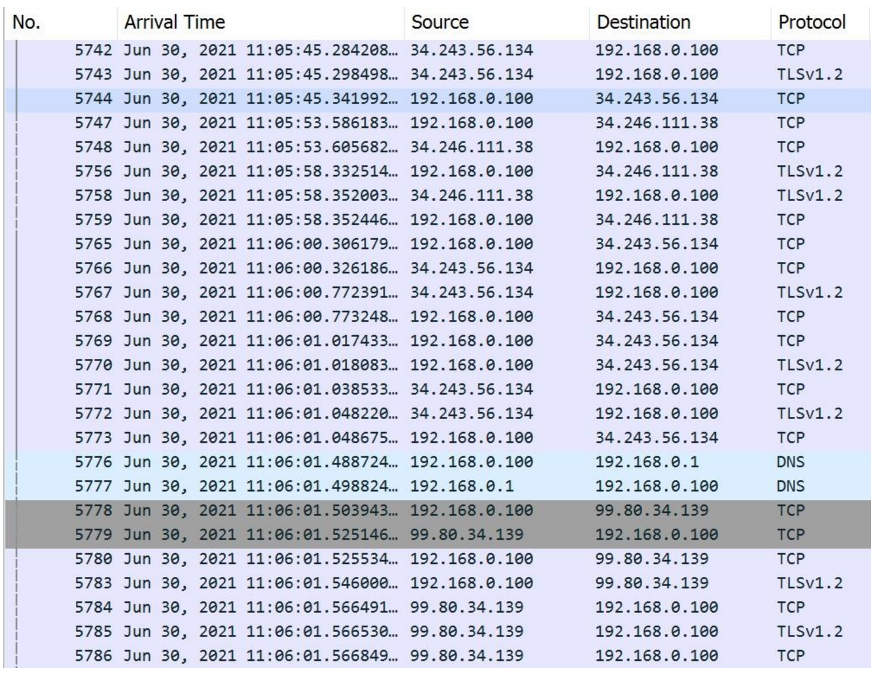
No.	Arrival Time	Source	Destination	Protocol
114	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.253	TCP
115	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.253	TLSv1.2
116	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.253	TCP
117	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.160	TLSv1.2
118	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.160	TCP
119	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TLSv1.2
120	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TCP
121	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TLSv1.2
122	Apr 4, 2021 09:21:17…	192.168.0.101	99.80.34.242	TCP
123	Apr 4, 2021 09:21:22…	192.168.0.101	34.240.107.124	TCP

Figure 3.10 Hub + motion sensor protocol frequency



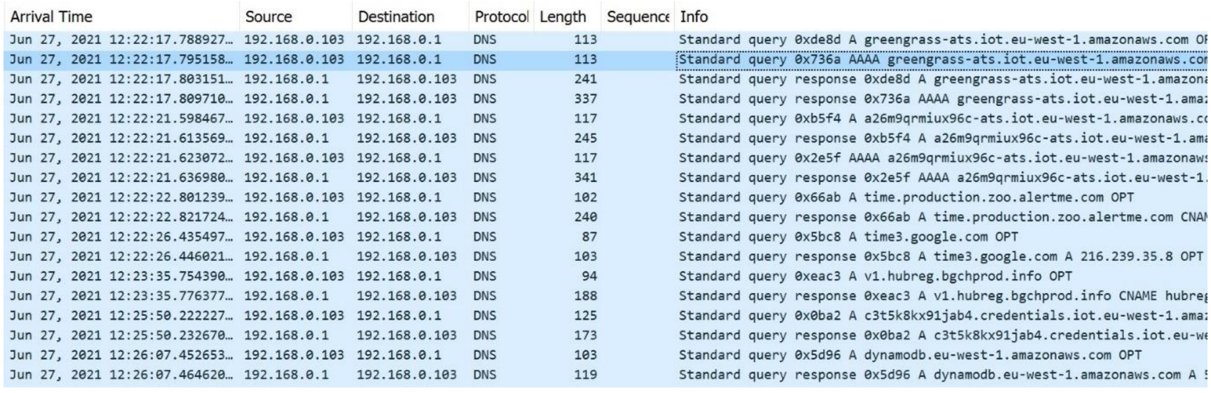
No.	Arrival Time	Source	Destination	Protocol
7883	Jun 30, 2021 11:18:05.719073000 GMT Summer Time	192.168.0.102	224.0.0.251	MDNS
7885	Jun 30, 2021 11:18:05.900755000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7886	Jun 30, 2021 11:18:05.900848000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7887	Jun 30, 2021 11:18:07.915540000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7888	Jun 30, 2021 11:18:07.915593000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7889	Jun 30, 2021 11:18:08.335248000 GMT Summer Time	192.168.0.100	34.246.111.38	TLSv1.2
7890	Jun 30, 2021 11:18:08.354895000 GMT Summer Time	34.246.111.38	192.168.0.100	TLSv1.2
7891	Jun 30, 2021 11:18:08.355325000 GMT Summer Time	192.168.0.100	34.246.111.38	TCP
7892	Jun 30, 2021 11:18:09.892066000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7893	Jun 30, 2021 11:18:09.892147000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7894	Jun 30, 2021 11:18:11.891591000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7895	Jun 30, 2021 11:18:11.891647000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7896	Jun 30, 2021 11:18:12.148770000 GMT Summer Time	192.168.0.100	34.243.56.134	TCP
7897	Jun 30, 2021 11:18:12.169060000 GMT Summer Time	34.243.56.134	192.168.0.100	TCP
7907	Jun 30, 2021 11:18:13.515917000 GMT Summer Time	192.168.0.100	34.243.56.134	TLSv1.2
7908	Jun 30, 2021 11:18:13.536069000 GMT Summer Time	34.243.56.134	192.168.0.100	TCP
7909	Jun 30, 2021 11:18:13.536426000 GMT Summer Time	34.243.56.134	192.168.0.100	TLSv1.2
7910	Jun 30, 2021 11:18:13.536740000 GMT Summer Time	192.168.0.100	34.243.56.134	TCP
7915	Jun 30, 2021 11:18:13.891697000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7916	Jun 30, 2021 11:18:13.891765000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7917	Jun 30, 2021 11:18:15.891283000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7918	Jun 30, 2021 11:18:15.891344000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7919	Jun 30, 2021 11:18:17.889479000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7920	Jun 30, 2021 11:18:17.889568000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7921	Jun 30, 2021 11:18:19.807242000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7922	Jun 30, 2021 11:18:19.807314000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7923	Jun 30, 2021 11:18:21.805752000 GMT Summer Time	192.168.0.102	192.168.0.103	TCP
7924	Jun 30, 2021 11:18:21.805843000 GMT Summer Time	192.168.0.103	192.168.0.102	TCP
7925	Jun 30, 2021 11:18:22.493744000 GMT Summer Time	192.168.0.102	169.254.22.150	HTTP
7928	Jun 30, 2021 11:18:22.506360000 GMT Summer Time	169.254.22.150	192.168.0.102	HTTP
7929	Jun 30, 2021 11:18:22.512751000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7930	Jun 30, 2021 11:18:22.513674000 GMT Summer Time	192.168.0.102	169.254.22.150	HTTP
7931	Jun 30, 2021 11:18:22.516182000 GMT Summer Time	169.254.22.150	192.168.0.102	HTTP
7932	Jun 30, 2021 11:18:22.519498000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7933	Jun 30, 2021 11:18:22.529873000 GMT Summer Time	192.168.0.102	169.254.22.150	HTTP
7934	Jun 30, 2021 11:18:22.541211000 GMT Summer Time	169.254.22.150	192.168.0.102	HTTP
7935	Jun 30, 2021 11:18:22.544405000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7936	Jun 30, 2021 11:18:22.841086000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7937	Jun 30, 2021 11:18:22.841548000 GMT Summer Time	169.254.22.150	192.168.0.102	TCP
7938	Jun 30, 2021 11:18:23.412707000 GMT Summer Time	192.168.0.100	34.246.111.38	TCP
7939	Jun 30, 2021 11:18:23.431513000 GMT Summer Time	34.246.111.38	192.168.0.100	TCP
7940	Jun 30, 2021 11:18:28.564771000 GMT Summer Time	192.168.0.100	34.243.56.134	TCP
7941	Jun 30, 2021 11:18:28.584675000 GMT Summer Time	34.243.56.134	192.168.0.100	TCP
7942	Jun 30, 2021 11:18:29.053062000 GMT Summer Time	192.168.0.102	169.254.22.150	HTTP
7943	Jun 30, 2021 11:18:29.096659000 GMT Summer Time	169.254.22.150	192.168.0.102	TCP
7944	Jun 30, 2021 11:18:29.188343000 GMT Summer Time	169.254.22.150	192.168.0.102	HTTP
7945	Jun 30, 2021 11:18:29.201155000 GMT Summer Time	192.168.0.100	34.243.56.134	TLSv1.2
7946	Jun 30, 2021 11:18:29.231666000 GMT Summer Time	34.243.56.134	192.168.0.100	TLSv1.2
7947	Jun 30, 2021 11:18:29.232122000 GMT Summer Time	192.168.0.100	34.243.56.134	TCP
7948	Jun 30, 2021 11:18:29.253926000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7949	Jun 30, 2021 11:18:29.539343000 GMT Summer Time	192.168.0.102	169.254.22.150	TCP
7950	Jun 30, 2021 11:18:29.539704000 GMT Summer Time	169.254.22.150	192.168.0.102	TCP
7951	Jun 30, 2021 11:18:29.686644000 GMT Summer Time	192.168.0.100	192.168.0.1	DNS
7952	Jun 30, 2021 11:18:29.697413000 GMT Summer Time	192.168.0.1	192.168.0.100	DNS

Figure 3.11 mDNS protocol in appearance when using a mobile phone



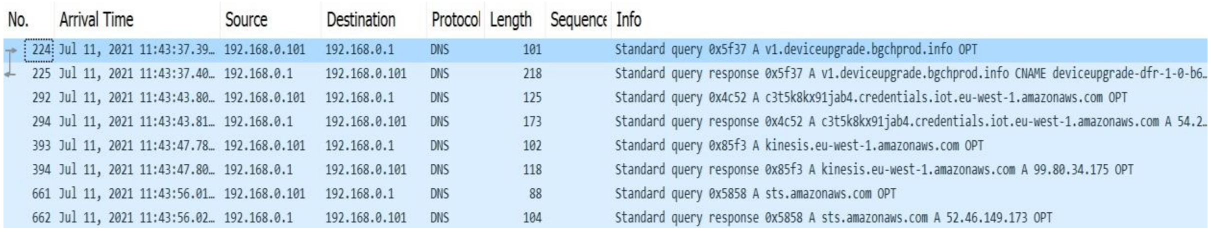
No.	Arrival Time	Source	Destination	Protocol
5742	Jun 30, 2021 11:05:45.284208…	34.243.56.134	192.168.0.100	TCP
5743	Jun 30, 2021 11:05:45.298498…	34.243.56.134	192.168.0.100	TLSv1.2
5744	Jun 30, 2021 11:05:45.341992…	192.168.0.100	34.243.56.134	TCP
5747	Jun 30, 2021 11:05:53.586183…	192.168.0.100	34.246.111.38	TCP
5748	Jun 30, 2021 11:05:53.605682…	34.246.111.38	192.168.0.100	TCP
5756	Jun 30, 2021 11:05:58.332514…	192.168.0.100	34.246.111.38	TLSv1.2
5758	Jun 30, 2021 11:05:58.352003…	34.246.111.38	192.168.0.100	TLSv1.2
5759	Jun 30, 2021 11:05:58.352446…	192.168.0.100	34.246.111.38	TCP
5765	Jun 30, 2021 11:06:00.306179…	192.168.0.100	34.243.56.134	TCP
5766	Jun 30, 2021 11:06:00.326186…	34.243.56.134	192.168.0.100	TCP
5767	Jun 30, 2021 11:06:00.772391…	34.243.56.134	192.168.0.100	TLSv1.2
5768	Jun 30, 2021 11:06:00.773248…	192.168.0.100	34.243.56.134	TCP
5769	Jun 30, 2021 11:06:01.017433…	192.168.0.100	34.243.56.134	TCP
5770	Jun 30, 2021 11:06:01.018083…	192.168.0.100	34.243.56.134	TLSv1.2
5771	Jun 30, 2021 11:06:01.038533…	34.243.56.134	192.168.0.100	TCP
5772	Jun 30, 2021 11:06:01.048220…	34.243.56.134	192.168.0.100	TLSv1.2
5773	Jun 30, 2021 11:06:01.048675…	192.168.0.100	34.243.56.134	TCP
5776	Jun 30, 2021 11:06:01.488724…	192.168.0.100	192.168.0.1	DNS
5777	Jun 30, 2021 11:06:01.498824…	192.168.0.1	192.168.0.100	DNS
5778	Jun 30, 2021 11:06:01.503943…	192.168.0.100	99.80.34.139	TCP
5779	Jun 30, 2021 11:06:01.525146…	99.80.34.139	192.168.0.100	TCP
5780	Jun 30, 2021 11:06:01.525534…	192.168.0.100	99.80.34.139	TCP
5783	Jun 30, 2021 11:06:01.546000…	192.168.0.100	99.80.34.139	TLSv1.2
5784	Jun 30, 2021 11:06:01.566491…	99.80.34.139	192.168.0.100	TCP
5785	Jun 30, 2021 11:06:01.566530…	99.80.34.139	192.168.0.100	TLSv1.2
5786	Jun 30, 2021 11:06:01.566849…	99.80.34.139	192.168.0.100	TCP

Figure 3.12 Absence of mDNS when operating using manual mode



Arrival Time	Source	Destination	Protocol	Length	Sequence	Info
Jun 27, 2021 12:22:17.788927…	192.168.0.103	192.168.0.1	DNS	113		Standard query 0xde8d A greengrass-ats.iot.eu-west-1.amazonaws.com OP
Jun 27, 2021 12:22:17.795158…	192.168.0.103	192.168.0.1	DNS	113		Standard query 0x736a AAAA greengrass-ats.iot.eu-west-1.amazonaws.com
Jun 27, 2021 12:22:17.803151…	192.168.0.1	192.168.0.103	DNS	241		Standard query response 0xde8d A greengrass-ats.iot.eu-west-1.amazona
Jun 27, 2021 12:22:17.809710…	192.168.0.1	192.168.0.103	DNS	337		Standard query response 0x736a AAAA greengrass-ats.iot.eu-west-1.ama
Jun 27, 2021 12:22:21.598467…	192.168.0.103	192.168.0.1	DNS	117		Standard query 0xb5f4 A a26m9qrmiux96c-ats.iot.eu-west-1.amazonaws.c
Jun 27, 2021 12:22:21.613569…	192.168.0.1	192.168.0.103	DNS	245		Standard query response 0xb5f4 A a26m9qrmiux96c-ats.iot.eu-west-1.ama
Jun 27, 2021 12:22:21.623072…	192.168.0.103	192.168.0.1	DNS	117		Standard query 0x2e5f AAAA a26m9qrmiux96c-ats.iot.eu-west-1.amazonaws
Jun 27, 2021 12:22:21.636980…	192.168.0.1	192.168.0.103	DNS	341		Standard query response 0x2e5f AAAA a26m9qrmiux96c-ats.iot.eu-west-1.
Jun 27, 2021 12:22:22.801239…	192.168.0.103	192.168.0.1	DNS	102		Standard query 0x66ab A time.production.zoo.alertme.com OPT
Jun 27, 2021 12:22:22.821724…	192.168.0.1	192.168.0.103	DNS	240		Standard query response 0x66ab A time.production.zoo.alertme.com CNA
Jun 27, 2021 12:22:26.435497…	192.168.0.103	192.168.0.1	DNS	87		Standard query 0x5bc8 A time3.google.com OPT
Jun 27, 2021 12:22:26.446021…	192.168.0.1	192.168.0.103	DNS	103		Standard query response 0x5bc8 A time3.google.com A 216.239.35.8 OPT
Jun 27, 2021 12:23:35.754390…	192.168.0.103	192.168.0.1	DNS	94		Standard query 0xeac3 A v1.hubreg.bgchprod.info OPT
Jun 27, 2021 12:23:35.776377…	192.168.0.1	192.168.0.103	DNS	188		Standard query response 0xeac3 A v1.hubreg.bgchprod.info CNAME hubre
Jun 27, 2021 12:25:50.222227…	192.168.0.103	192.168.0.1	DNS	125		Standard query 0x0ba2 A c3t5k8kx91jab4.credentials.iot.eu-west-1.ama
Jun 27, 2021 12:25:50.232670…	192.168.0.1	192.168.0.103	DNS	173		Standard query response 0x0ba2 A c3t5k8kx91jab4.credentials.iot.eu-w
Jun 27, 2021 12:26:07.452653…	192.168.0.103	192.168.0.1	DNS	103		Standard query 0x5d96 A dynamodb.eu-west-1.amazonaws.com OPT
Jun 27, 2021 12:26:07.464620…	192.168.0.1	192.168.0.103	DNS	119		Standard query response 0x5d96 A dynamodb.eu-west-1.amazonaws.com A

Figure 3.13 DNS queries during boot mode



No.	Arrival Time	Source	Destination	Protocol	Length	Sequence	Info
224	Jul 11, 2021 11:43:37.39…	192.168.0.101	192.168.0.1	DNS	101		Standard query 0x5f37 A v1.deviceupgrade.bgchprod.info OPT
225	Jul 11, 2021 11:43:37.40…	192.168.0.1	192.168.0.101	DNS	218		Standard query response 0x5f37 A v1.deviceupgrade.bgchprod.info CNAME deviceupgrade-dfr-1-0-b6…
292	Jul 11, 2021 11:43:43.80…	192.168.0.101	192.168.0.1	DNS	125		Standard query 0x4c52 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
294	Jul 11, 2021 11:43:43.81…	192.168.0.1	192.168.0.101	DNS	173		Standard query response 0x4c52 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 54.2…
393	Jul 11, 2021 11:43:47.78…	192.168.0.101	192.168.0.1	DNS	102		Standard query 0x85f3 A kinesis.eu-west-1.amazonaws.com OPT
394	Jul 11, 2021 11:43:47.80…	192.168.0.1	192.168.0.101	DNS	118		Standard query response 0x85f3 A kinesis.eu-west-1.amazonaws.com A 99.80.34.175 OPT
661	Jul 11, 2021 11:43:56.01…	192.168.0.101	192.168.0.1	DNS	88		Standard query 0x5858 A sts.amazonaws.com OPT
662	Jul 11, 2021 11:43:56.02…	192.168.0.1	192.168.0.101	DNS	104		Standard query response 0x5858 A sts.amazonaws.com A 52.46.149.173 OPT

Figure 3.14 DNS queries during pairing mode

3.5.2 Device Identification

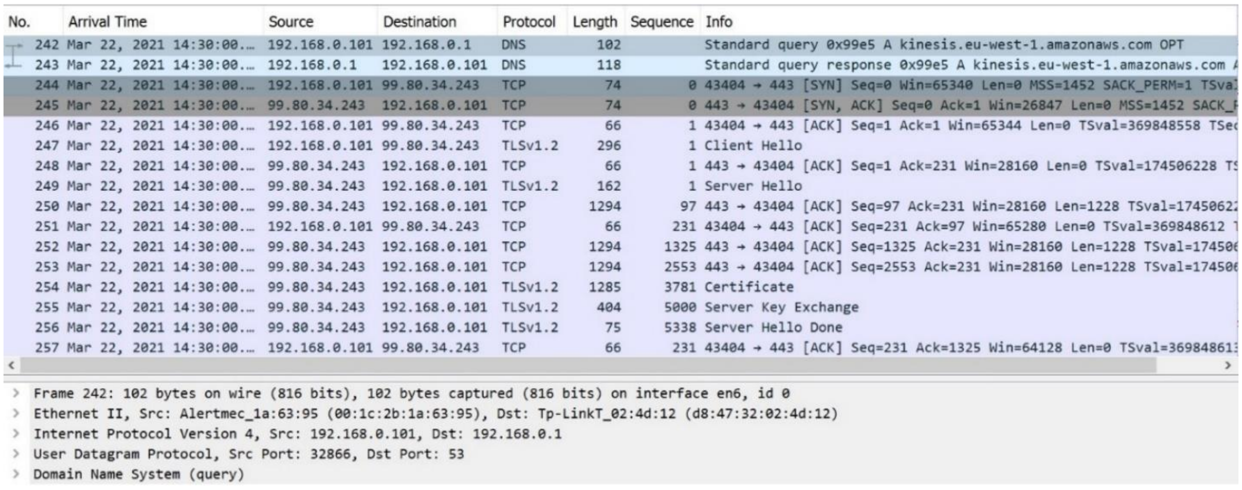
Each device was paired with the hub individually to get a unique fingerprint of the device. This method will help in identifying each device from the type of traffic it generates when all devices are paired to the hub. However, it was discovered that all 3 devices have a similar pattern. All devices utilized the same source MAC and IP Address, which is that of the hub. They also utilized the same protocols and server-side port numbers. Furthermore, when an event is triggered, all devices have the same traffic pattern of contacting the same DNS servers, having the same flow volume and duration as will be seen in subsequent sections of this chapter. Table 3.2 shows a list of these DNS servers, their respective lengths and purpose. Whenever an event is triggered like the smart plug or lamp going ON, a DNS request is made to **kinesis.eu-west-1.amazonaws.com,** which contains requests and reply packets. The same thing happens when the plug or lamp goes OFF, as this DNS is queried. An interesting observation is, whenever the interval between one triggered event (ON/OFF) and the next is less than an hour then a DNS request is made to only **kinesis.eu-west-1.amazonaws.com**, however if the interval is more than an hour then a DNS request is made to **c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com**, **sts.amazonaws.com** and **kinesis.eu-west-1.amazonaws.com**. The reason behind this is **c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com** and **sts.amazonaws.com** are responsible for providing temporary authentication keys whenever a request or a triggered event takes place. These keys are short lived and by default expire in one hour, thus the need for newly generated authentication keys when a new query is made an hour after the last one as seen in table 3.2.

However, one slightly different behaviour was observed which differentiated the motion sensor from the plug and lamp. When it detects motion, a DNS query and response is established as mentioned earlier which is the same with the plug and lamp. However, the motion sensor always establishes another DNS query 5 minutes later, which is not the case for the plug and lamp. Figures 3.15, 3.16 and 3.17 show these commonalities shared by the plug, lamp, and motion sensor. Figure 3.18 shows event triggered DNS activity from these devices, independently. The packet length was plotted against the time stamp. The reason for using the packet length is because each length corresponds to a particular DNS address. For instance, length **102** is **kinesis.eu-west-1.amazonaws.com** and **125** is **c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com**. This method was found to present a neater and less cluttered labels on the graph, as they are shorter and easier to read. The lamp and bulb have a single peak of contacting the **kinesis.eu-west-1.amazonaws.com** DNS server of **length 102**, when an event is triggered while the motion sensor has a distinct pattern of having 2 peaks for every event trigger to this same server. In Figure 3.18 the lamp was triggered at 12:00, 1:00, 2:30 and at 3:30, all having a single peak. The plug was triggered at 2:30, 3:30 and 4:30 with single peaks as well for each trigger. The motion sensor detected motion 3 times between 8:30 and 12:00 each time having double peaks when motion was detected.

As observed from the section above in periodic queries, the hub has a unique pattern of generating periodic traffic like TCP Keep Alive, DNS, ARP, and DHCP. This unique traffic makes it easier to identify the hub in a pool of other IoT devices. This can be seen in figure 3.19, 3.20, 3.21 and 3.22 respectively with the hubs IP address being 192.168.0.101.

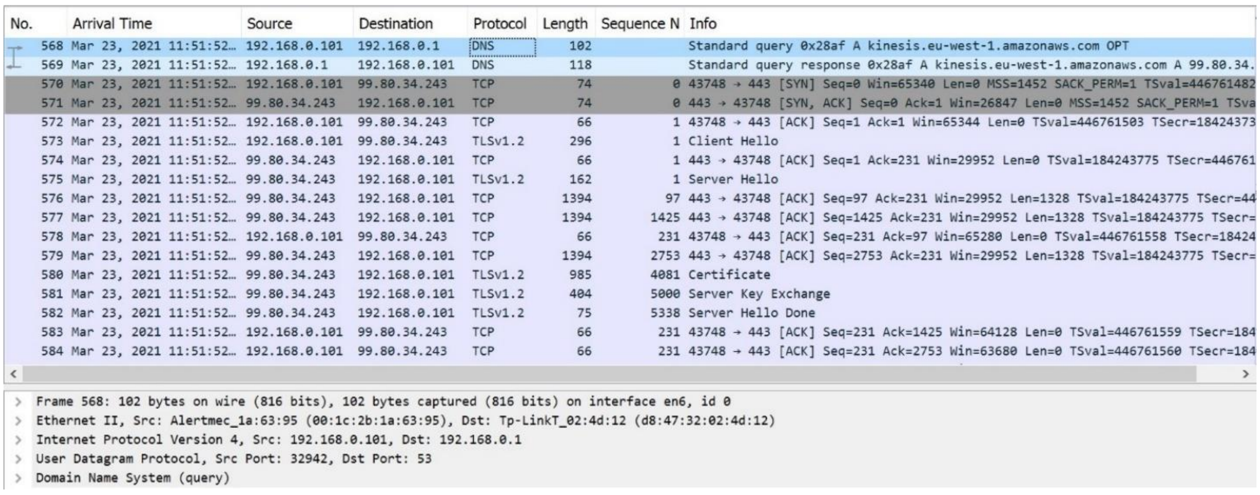
Table 3.2 DNS servers queried and their purpose

Sts.amazonaws.com (packet length 88)	For provision of temporary limited privilege credentials for AWS identity and access management. These temporary credentials last for one hour by default thus, why the sts.amazonaws.com DNS query when an event is an hour apart from the last.
Dynamodb.eu-west-1.amazonaws.com (packet length 103)	Proprietary NoSQL database service that supports key value and documents data structure.
Kinesis.eu-west-1.amazonaws.com (packet length 102)	Managed scalable cloud-based service allowing real-time processing of streaming large amount of data. Designed for real-time applications (in this case hive app).
Credentials.iot.eu-west-1.amazonaws.com (packet length 125)	This credentials provider authenticates a client in this case the smart device making request, and issues temporary limited privilege security token. This token can be used to sign and authenticate any AWS request. Therefore, the same one-hour expiration rule applies.
Ec2.00.00.00.00.eu-west-1.amazonaws.com	Elastic compute cloud that provides secure, resizable compute capacity on the cloud.



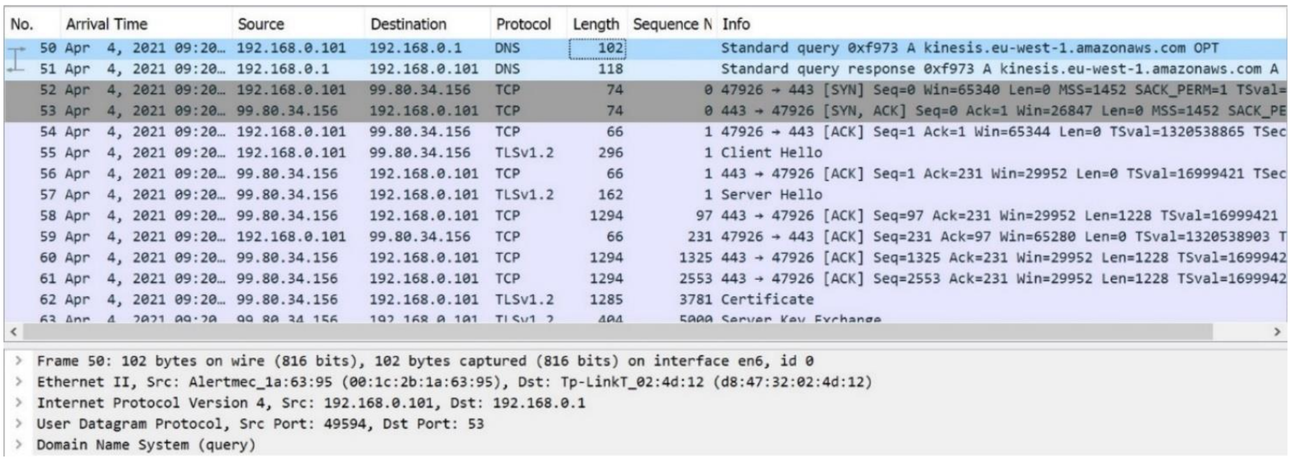
```
No.  Arrival Time                Source        Destination   Protocol Length Sequence Info
 242 Mar 22, 2021 14:30:00.…    192.168.0.101 192.168.0.1   DNS         102          Standard query 0x99e5 A kinesis.eu-west-1.amazonaws.com OPT
 243 Mar 22, 2021 14:30:00.…    192.168.0.1   192.168.0.101 DNS         118          Standard query response 0x99e5 A kinesis.eu-west-1.amazonaws.com A
 244 Mar 22, 2021 14:30:00.…    192.168.0.101 99.80.34.243  TCP          74        0 43404 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval
 245 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TCP          74        0 443 → 43404 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_F
 246 Mar 22, 2021 14:30:00.…    192.168.0.101 99.80.34.243  TCP          66        1 43404 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=369848558 TSec
 247 Mar 22, 2021 14:30:00.…    192.168.0.101 99.80.34.243  TLSv1.2     296        1 Client Hello
 248 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TCP          66        1 443 → 43404 [ACK] Seq=1 Ack=231 Win=28160 Len=0 TSval=174506228 TS
 249 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TLSv1.2     162        1 Server Hello
 250 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TCP        1294       97 443 → 43404 [ACK] Seq=97 Ack=231 Win=28160 Len=1228 TSval=17450622
 251 Mar 22, 2021 14:30:00.…    192.168.0.101 99.80.34.243  TCP          66      231 43404 → 443 [ACK] Seq=231 Ack=97 Win=65280 Len=0 TSval=369848612 T
 252 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TCP        1294     1325 443 → 43404 [ACK] Seq=1325 Ack=231 Win=28160 Len=1228 TSval=174506
 253 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TCP        1294     2553 443 → 43404 [ACK] Seq=2553 Ack=231 Win=28160 Len=1228 TSval=174506
 254 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TLSv1.2    1285     3781 Certificate
 255 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TLSv1.2     404     5000 Server Key Exchange
 256 Mar 22, 2021 14:30:00.…    99.80.34.243  192.168.0.101 TLSv1.2      75     5338 Server Hello Done
 257 Mar 22, 2021 14:30:00.…    192.168.0.101 99.80.34.243  TCP          66      231 43404 → 443 [ACK] Seq=231 Ack=1325 Win=64128 Len=0 TSval=36984861

> Frame 242: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface en6, id 0
> Ethernet II, Src: Alertmec_1a:63:95 (00:1c:2b:1a:63:95), Dst: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
> Internet Protocol Version 4, Src: 192.168.0.101, Dst: 192.168.0.1
> User Datagram Protocol, Src Port: 32866, Dst Port: 53
> Domain Name System (query)
```

Figure 3.15 Plug traffic



```
No.  Arrival Time             Source          Destination     Protocol Length Sequence N Info
 568 Mar 23, 2021 11:51:52…  192.168.0.101   192.168.0.1     DNS         102            Standard query 0x28af A kinesis.eu-west-1.amazonaws.com OPT
 569 Mar 23, 2021 11:51:52…  192.168.0.1     192.168.0.101   DNS         118            Standard query response 0x28af A kinesis.eu-west-1.amazonaws.com A 99.80.34.
 570 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TCP          74          0 43748 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=446761482
 571 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TCP          74          0 443 → 43748 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSva
 572 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TCP          66          1 43748 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=446761503 TSecr=18424373
 573 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TLSv1.2     296          1 Client Hello
 574 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TCP          66          1 443 → 43748 [ACK] Seq=1 Ack=231 Win=29952 Len=0 TSval=184243775 TSecr=446761
 575 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TLSv1.2     162          1 Server Hello
 576 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TCP        1394         97 443 → 43748 [ACK] Seq=97 Ack=231 Win=29952 Len=1328 TSval=184243775 TSecr=44
 577 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TCP        1394       1425 443 → 43748 [ACK] Seq=1425 Ack=231 Win=29952 Len=1328 TSval=184243775 TSecr=
 578 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TCP          66        231 43748 → 443 [ACK] Seq=231 Ack=97 Win=65280 Len=0 TSval=446761558 TSecr=18424
 579 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TCP        1394       2753 443 → 43748 [ACK] Seq=2753 Ack=231 Win=29952 Len=1328 TSval=184243775 TSecr=
 580 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TLSv1.2     985       4081 Certificate
 581 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TLSv1.2     404       5000 Server Key Exchange
 582 Mar 23, 2021 11:51:52…  99.80.34.243    192.168.0.101   TLSv1.2      75       5338 Server Hello Done
 583 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TCP          66        231 43748 → 443 [ACK] Seq=231 Ack=1425 Win=64128 Len=0 TSval=446761559 TSecr=184
 584 Mar 23, 2021 11:51:52…  192.168.0.101   99.80.34.243    TCP          66        231 43748 → 443 [ACK] Seq=231 Ack=2753 Win=63680 Len=0 TSval=446761560 TSecr=184

> Frame 568: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface en6, id 0
> Ethernet II, Src: Alertmec_1a:63:95 (00:1c:2b:1a:63:95), Dst: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
> Internet Protocol Version 4, Src: 192.168.0.101, Dst: 192.168.0.1
> User Datagram Protocol, Src Port: 32942, Dst Port: 53
> Domain Name System (query)
```

Figure 3.16 lamp traffic



```
No.  Arrival Time            Source          Destination     Protocol Length Sequence N Info
 50 Apr  4, 2021 09:20…     192.168.0.101   192.168.0.1     DNS         102            Standard query 0xf973 A kinesis.eu-west-1.amazonaws.com OPT
 51 Apr  4, 2021 09:20…     192.168.0.1     192.168.0.101   DNS         118            Standard query response 0xf973 A kinesis.eu-west-1.amazonaws.com A
 52 Apr  4, 2021 09:20…     192.168.0.101   99.80.34.156    TCP          74          0 47926 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=
 53 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TCP          74          0 443 → 47926 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PE
 54 Apr  4, 2021 09:20…     192.168.0.101   99.80.34.156    TCP          66          1 47926 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=1320538865 TSec
 55 Apr  4, 2021 09:20…     192.168.0.101   99.80.34.156    TLSv1.2     296          1 Client Hello
 56 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TCP          66          1 443 → 47926 [ACK] Seq=1 Ack=231 Win=29952 Len=0 TSval=16999421 TSec
 57 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TLSv1.2     162          1 Server Hello
 58 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TCP        1294         97 443 → 47926 [ACK] Seq=97 Ack=231 Win=29952 Len=1228 TSval=16999421
 59 Apr  4, 2021 09:20…     192.168.0.101   99.80.34.156    TCP          66        231 47926 → 443 [ACK] Seq=231 Ack=97 Win=65280 Len=0 TSval=1320538903 T
 60 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TCP        1294       1325 443 → 47926 [ACK] Seq=1325 Ack=231 Win=29952 Len=1228 TSval=1699942
 61 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TCP        1294       2553 443 → 47926 [ACK] Seq=2553 Ack=231 Win=29952 Len=1228 TSval=1699942
 62 Apr  4, 2021 09:20…     99.80.34.156    192.168.0.101   TLSv1.2    1285       3781 Certificate
 63 Apr  4, 2021 09:20      99.80.34.156    192.168.0.101   TLSv1.2     404       5000 Server Key Exchange

> Frame 50: 102 bytes on wire (816 bits), 102 bytes captured (816 bits) on interface en6, id 0
> Ethernet II, Src: Alertmec_1a:63:95 (00:1c:2b:1a:63:95), Dst: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
> Internet Protocol Version 4, Src: 192.168.0.101, Dst: 192.168.0.1
> User Datagram Protocol, Src Port: 49594, Dst Port: 53
> Domain Name System (query)
```

Figure 3.17 Motion sensor traffic

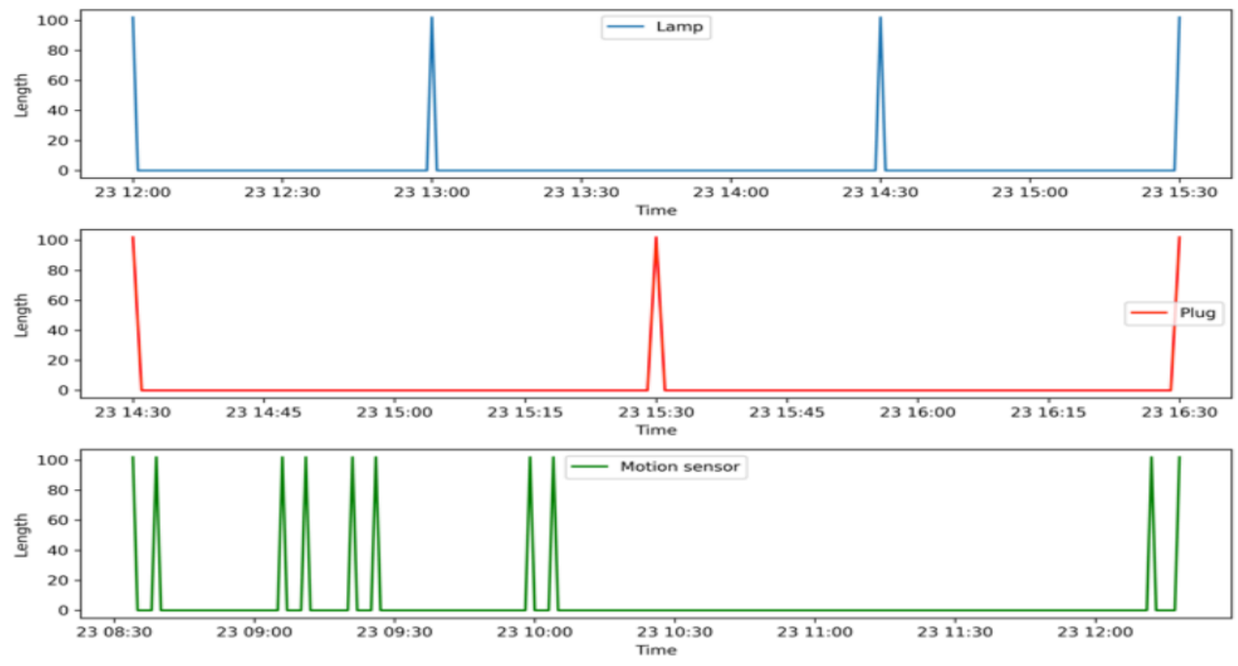
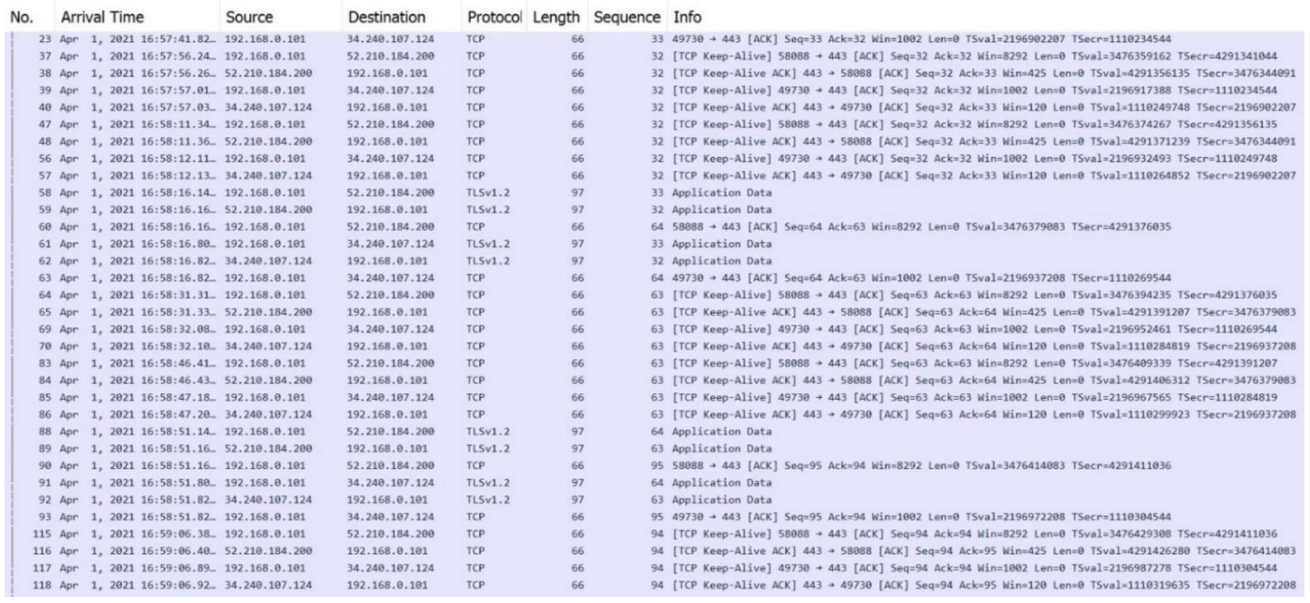
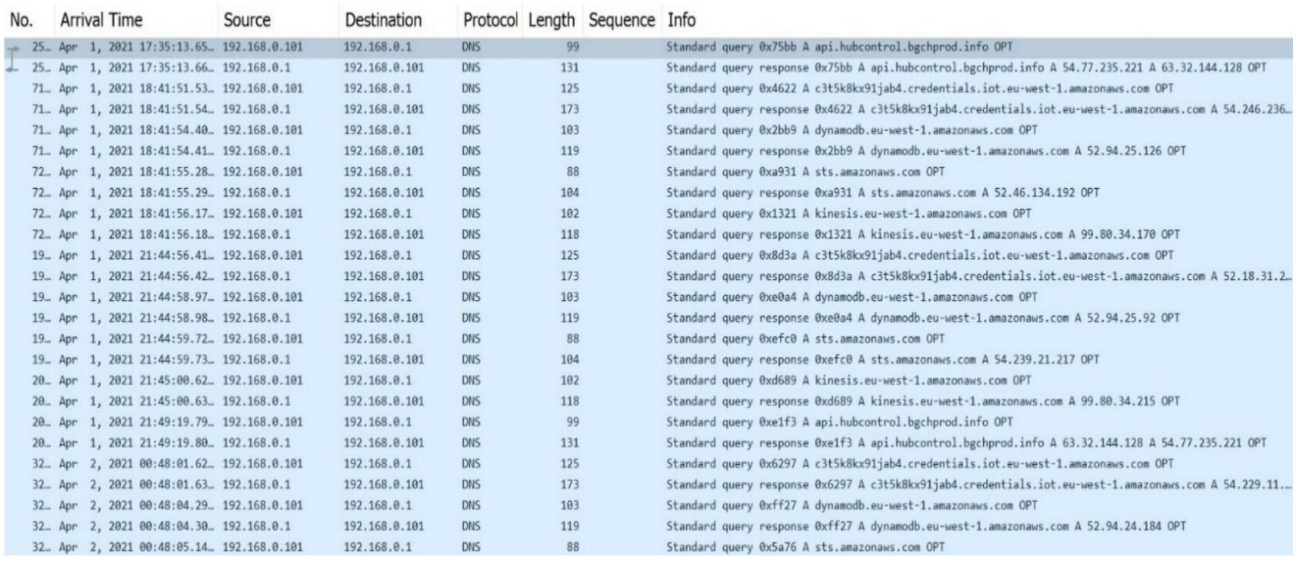


Figure 3.18 Device identification by trigger pattern using DNS query length



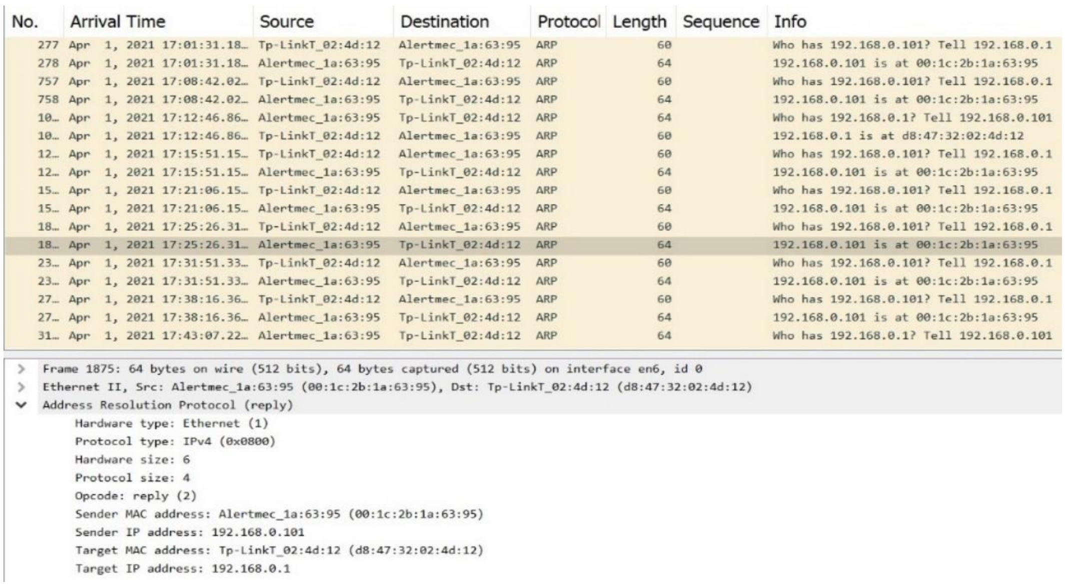
No.	Arrival Time	Source	Destination	Protocol	Length	Sequence	Info
23	Apr 1, 2021 16:57:41.82…	192.168.0.101	34.240.107.124	TCP	66	33	49730 → 443 [ACK] Seq=33 Ack=32 Win=1002 Len=0 TSval=2196902207 TSecr=1110234544
37	Apr 1, 2021 16:57:56.24…	192.168.0.101	52.210.184.200	TCP	66	32	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=32 Ack=32 Win=8292 Len=0 TSval=3476359162 TSecr=4291341044
38	Apr 1, 2021 16:57:56.26…	52.210.184.200	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=32 Ack=33 Win=425 Len=0 TSval=4291356135 TSecr=3476344091
39	Apr 1, 2021 16:57:57.01…	192.168.0.101	34.240.107.124	TCP	66	32	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=32 Ack=32 Win=1002 Len=0 TSval=2196917388 TSecr=1110234544
40	Apr 1, 2021 16:57:57.03…	34.240.107.124	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=32 Ack=33 Win=120 Len=0 TSval=1110249748 TSecr=2196902207
47	Apr 1, 2021 16:58:11.34…	192.168.0.101	52.210.184.200	TCP	66	32	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=32 Ack=32 Win=8292 Len=0 TSval=3476374267 TSecr=4291356135
48	Apr 1, 2021 16:58:11.36…	52.210.184.200	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=32 Ack=33 Win=425 Len=0 TSval=4291371239 TSecr=3476344091
56	Apr 1, 2021 16:58:12.11…	192.168.0.101	34.240.107.124	TCP	66	32	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=32 Ack=32 Win=1002 Len=0 TSval=2196932493 TSecr=1110249748
57	Apr 1, 2021 16:58:12.13…	34.240.107.124	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=32 Ack=33 Win=120 Len=0 TSval=1110264852 TSecr=2196902207
58	Apr 1, 2021 16:58:16.14…	192.168.0.101	52.210.184.200	TLSv1.2	97	33	Application Data
59	Apr 1, 2021 16:58:16.16…	52.210.184.200	192.168.0.101	TLSv1.2	97	32	Application Data
60	Apr 1, 2021 16:58:16.16…	192.168.0.101	52.210.184.200	TCP	66	64	58088 → 443 [ACK] Seq=64 Ack=63 Win=8292 Len=0 TSval=3476379083 TSecr=4291376035
61	Apr 1, 2021 16:58:16.80…	192.168.0.101	34.240.107.124	TLSv1.2	97	33	Application Data
62	Apr 1, 2021 16:58:16.82…	34.240.107.124	192.168.0.101	TLSv1.2	97	32	Application Data
63	Apr 1, 2021 16:58:16.82…	192.168.0.101	34.240.107.124	TCP	66	64	49730 → 443 [ACK] Seq=64 Ack=63 Win=1002 Len=0 TSval=2196937208 TSecr=1110269544
64	Apr 1, 2021 16:58:31.31…	192.168.0.101	52.210.184.200	TCP	66	63	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=63 Ack=63 Win=8292 Len=0 TSval=3476394235 TSecr=4291376035
65	Apr 1, 2021 16:58:31.33…	52.210.184.200	192.168.0.101	TCP	66	63	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=63 Ack=64 Win=425 Len=0 TSval=4291391207 TSecr=3476379083
69	Apr 1, 2021 16:58:32.08…	192.168.0.101	34.240.107.124	TCP	66	63	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=63 Ack=63 Win=1002 Len=0 TSval=2196952461 TSecr=1110269544
70	Apr 1, 2021 16:58:32.10…	34.240.107.124	192.168.0.101	TCP	66	63	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=63 Ack=64 Win=120 Len=0 TSval=1110284819 TSecr=2196937208
83	Apr 1, 2021 16:58:46.41…	192.168.0.101	52.210.184.200	TCP	66	63	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=63 Ack=63 Win=8292 Len=0 TSval=3476409339 TSecr=4291391207
84	Apr 1, 2021 16:58:46.43…	52.210.184.200	192.168.0.101	TCP	66	63	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=63 Ack=64 Win=425 Len=0 TSval=4291406312 TSecr=3476379083
85	Apr 1, 2021 16:58:47.18…	192.168.0.101	34.240.107.124	TCP	66	63	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=63 Ack=63 Win=1002 Len=0 TSval=2196967565 TSecr=1110284819
86	Apr 1, 2021 16:58:47.20…	34.240.107.124	192.168.0.101	TCP	66	63	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=63 Ack=64 Win=120 Len=0 TSval=1110299923 TSecr=2196937208
88	Apr 1, 2021 16:58:51.14…	192.168.0.101	52.210.184.200	TLSv1.2	97	64	Application Data
89	Apr 1, 2021 16:58:51.16…	52.210.184.200	192.168.0.101	TLSv1.2	97	63	Application Data
90	Apr 1, 2021 16:58:51.16…	192.168.0.101	52.210.184.200	TCP	66	95	58088 → 443 [ACK] Seq=95 Ack=94 Win=8292 Len=0 TSval=3476414083 TSecr=4291411036
91	Apr 1, 2021 16:58:51.80…	192.168.0.101	34.240.107.124	TLSv1.2	97	64	Application Data
92	Apr 1, 2021 16:58:51.82…	34.240.107.124	192.168.0.101	TLSv1.2	97	63	Application Data
93	Apr 1, 2021 16:58:51.82…	192.168.0.101	34.240.107.124	TCP	66	95	49730 → 443 [ACK] Seq=95 Ack=94 Win=1002 Len=0 TSval=2196972208 TSecr=1110304544
115	Apr 1, 2021 16:59:06.38…	192.168.0.101	52.210.184.200	TCP	66	94	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=94 Ack=94 Win=8292 Len=0 TSval=3476429308 TSecr=4291411036
116	Apr 1, 2021 16:59:06.40…	52.210.184.200	192.168.0.101	TCP	66	94	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=94 Ack=95 Win=425 Len=0 TSval=4291426280 TSecr=3476414083
117	Apr 1, 2021 16:59:06.89…	192.168.0.101	34.240.107.124	TCP	66	94	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=94 Ack=94 Win=1002 Len=0 TSval=2196987278 TSecr=1110304544
118	Apr 1, 2021 16:59:06.92…	34.240.107.124	192.168.0.101	TCP	66	94	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=94 Ack=95 Win=120 Len=0 TSval=1110319635 TSecr=2196972208

Figure 3.19 Hub TCP keep-alive packets



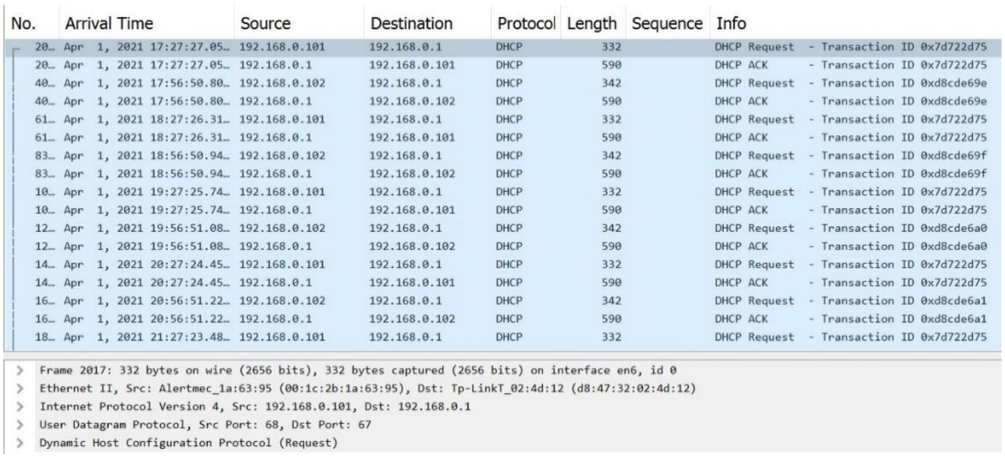
```
No.   Arrival Time                 Source          Destination     Protocol  Length  Sequence  Info
25…   Apr  1, 2021 17:35:13.65…   192.168.0.101   192.168.0.1     DNS           99            Standard query 0x75bb A api.hubcontrol.bgchprod.info OPT
25…   Apr  1, 2021 17:35:13.66…   192.168.0.1     192.168.0.101   DNS          131            Standard query response 0x75bb A api.hubcontrol.bgchprod.info A 54.77.235.221 A 63.32.144.128 OPT
71…   Apr  1, 2021 18:41:51.53…   192.168.0.101   192.168.0.1     DNS          125            Standard query 0x4622 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
71…   Apr  1, 2021 18:41:51.54…   192.168.0.1     192.168.0.101   DNS          173            Standard query response 0x4622 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 54.246.236…
71…   Apr  1, 2021 18:41:54.40…   192.168.0.101   192.168.0.1     DNS          103            Standard query 0x2bb9 A dynamodb.eu-west-1.amazonaws.com OPT
71…   Apr  1, 2021 18:41:54.41…   192.168.0.1     192.168.0.101   DNS          119            Standard query response 0x2bb9 A dynamodb.eu-west-1.amazonaws.com A 52.94.25.126 OPT
72…   Apr  1, 2021 18:41:55.28…   192.168.0.101   192.168.0.1     DNS           88            Standard query 0xa931 A sts.amazonaws.com OPT
72…   Apr  1, 2021 18:41:55.29…   192.168.0.1     192.168.0.101   DNS          104            Standard query response 0xa931 A sts.amazonaws.com A 52.46.134.192 OPT
72…   Apr  1, 2021 18:41:56.17…   192.168.0.101   192.168.0.1     DNS          102            Standard query 0x1321 A kinesis.eu-west-1.amazonaws.com OPT
72…   Apr  1, 2021 18:41:56.18…   192.168.0.1     192.168.0.101   DNS          118            Standard query response 0x1321 A kinesis.eu-west-1.amazonaws.com A 99.80.34.170 OPT
19…   Apr  1, 2021 21:44:56.41…   192.168.0.101   192.168.0.1     DNS          125            Standard query 0x8d3a A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
19…   Apr  1, 2021 21:44:56.42…   192.168.0.1     192.168.0.101   DNS          173            Standard query response 0x8d3a A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 52.18.31.2…
19…   Apr  1, 2021 21:44:58.97…   192.168.0.101   192.168.0.1     DNS          103            Standard query 0xe0a4 A dynamodb.eu-west-1.amazonaws.com OPT
19…   Apr  1, 2021 21:44:58.98…   192.168.0.1     192.168.0.101   DNS          119            Standard query response 0xe0a4 A dynamodb.eu-west-1.amazonaws.com A 52.94.25.92 OPT
19…   Apr  1, 2021 21:44:59.72…   192.168.0.101   192.168.0.1     DNS           88            Standard query 0xefc0 A sts.amazonaws.com OPT
19…   Apr  1, 2021 21:44:59.73…   192.168.0.1     192.168.0.101   DNS          104            Standard query response 0xefc0 A sts.amazonaws.com A 54.239.21.217 OPT
20…   Apr  1, 2021 21:45:00.62…   192.168.0.101   192.168.0.1     DNS          102            Standard query 0xd689 A kinesis.eu-west-1.amazonaws.com OPT
20…   Apr  1, 2021 21:45:00.63…   192.168.0.1     192.168.0.101   DNS          118            Standard query response 0xd689 A kinesis.eu-west-1.amazonaws.com A 99.80.34.215 OPT
20…   Apr  1, 2021 21:49:19.79…   192.168.0.101   192.168.0.1     DNS           99            Standard query 0xe1f3 A api.hubcontrol.bgchprod.info OPT
20…   Apr  1, 2021 21:49:19.80…   192.168.0.1     192.168.0.101   DNS          131            Standard query response 0xe1f3 A api.hubcontrol.bgchprod.info A 63.32.144.128 A 54.77.235.221 OPT
32…   Apr  2, 2021 00:48:01.62…   192.168.0.101   192.168.0.1     DNS          125            Standard query 0x6297 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
32…   Apr  2, 2021 00:48:01.63…   192.168.0.1     192.168.0.101   DNS          173            Standard query response 0x6297 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 54.229.11.…
32…   Apr  2, 2021 00:48:04.29…   192.168.0.101   192.168.0.1     DNS          103            Standard query 0xff27 A dynamodb.eu-west-1.amazonaws.com OPT
32…   Apr  2, 2021 00:48:04.30…   192.168.0.1     192.168.0.101   DNS          119            Standard query response 0xff27 A dynamodb.eu-west-1.amazonaws.com A 52.94.24.184 OPT
32…   Apr  2, 2021 00:48:05.14…   192.168.0.101   192.168.0.1     DNS           88            Standard query 0x5a76 A sts.amazonaws.com OPT
```

Figure 3.20 Hub DNS packets



```
No.   Arrival Time                 Source               Destination          Protocol  Length  Sequence  Info
277   Apr  1, 2021 17:01:31.18…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
278   Apr  1, 2021 17:01:31.18…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
757   Apr  1, 2021 17:08:42.02…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
758   Apr  1, 2021 17:08:42.02…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
10…   Apr  1, 2021 17:12:46.86…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            Who has 192.168.0.1? Tell 192.168.0.101
10…   Apr  1, 2021 17:12:46.86…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            192.168.0.1 is at d8:47:32:02:4d:12
12…   Apr  1, 2021 17:15:51.15…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
12…   Apr  1, 2021 17:15:51.15…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
15…   Apr  1, 2021 17:21:06.15…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
15…   Apr  1, 2021 17:21:06.15…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
18…   Apr  1, 2021 17:25:26.31…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
18…   Apr  1, 2021 17:25:26.31…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
23…   Apr  1, 2021 17:31:51.33…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
23…   Apr  1, 2021 17:31:51.33…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
27…   Apr  1, 2021 17:38:16.36…   Tp-LinkT_02:4d:12    Alertmec_1a:63:95    ARP           60            Who has 192.168.0.101? Tell 192.168.0.1
27…   Apr  1, 2021 17:38:16.36…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            192.168.0.101 is at 00:1c:2b:1a:63:95
31…   Apr  1, 2021 17:43:07.22…   Alertmec_1a:63:95    Tp-LinkT_02:4d:12    ARP           64            Who has 192.168.0.1? Tell 192.168.0.101

>  Frame 1875: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface en6, id 0
>  Ethernet II, Src: Alertmec_1a:63:95 (00:1c:2b:1a:63:95), Dst: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
v  Address Resolution Protocol (reply)
       Hardware type: Ethernet (1)
       Protocol type: IPv4 (0x0800)
       Hardware size: 6
       Protocol size: 4
       Opcode: reply (2)
       Sender MAC address: Alertmec_1a:63:95 (00:1c:2b:1a:63:95)
       Sender IP address: 192.168.0.101
       Target MAC address: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
       Target IP address: 192.168.0.1
```

Figure 3.21 Hub ARP packets



```
No.   Arrival Time                 Source          Destination     Protocol  Length  Sequence  Info
20…   Apr  1, 2021 17:27:27.05…   192.168.0.101   192.168.0.1     DHCP         332            DHCP Request  - Transaction ID 0x7d722d75
20…   Apr  1, 2021 17:27:27.05…   192.168.0.1     192.168.0.101   DHCP         590            DHCP ACK      - Transaction ID 0x7d722d75
40…   Apr  1, 2021 17:56:50.80…   192.168.0.102   192.168.0.1     DHCP         342            DHCP Request  - Transaction ID 0xd8cde69e
40…   Apr  1, 2021 17:56:50.80…   192.168.0.1     192.168.0.102   DHCP         590            DHCP ACK      - Transaction ID 0xd8cde69e
61…   Apr  1, 2021 18:27:26.31…   192.168.0.101   192.168.0.1     DHCP         332            DHCP Request  - Transaction ID 0x7d722d75
61…   Apr  1, 2021 18:27:26.31…   192.168.0.1     192.168.0.101   DHCP         590            DHCP ACK      - Transaction ID 0x7d722d75
83…   Apr  1, 2021 18:56:50.94…   192.168.0.102   192.168.0.1     DHCP         342            DHCP Request  - Transaction ID 0xd8cde69f
83…   Apr  1, 2021 18:56:50.94…   192.168.0.1     192.168.0.102   DHCP         590            DHCP ACK      - Transaction ID 0xd8cde69f
10…   Apr  1, 2021 19:27:25.74…   192.168.0.101   192.168.0.1     DHCP         332            DHCP Request  - Transaction ID 0x7d722d75
10…   Apr  1, 2021 19:27:25.74…   192.168.0.1     192.168.0.101   DHCP         590            DHCP ACK      - Transaction ID 0x7d722d75
12…   Apr  1, 2021 19:56:51.08…   192.168.0.102   192.168.0.1     DHCP         342            DHCP Request  - Transaction ID 0xd8cde6a0
12…   Apr  1, 2021 19:56:51.08…   192.168.0.1     192.168.0.102   DHCP         590            DHCP ACK      - Transaction ID 0xd8cde6a0
14…   Apr  1, 2021 20:27:24.45…   192.168.0.101   192.168.0.1     DHCP         332            DHCP Request  - Transaction ID 0x7d722d75
14…   Apr  1, 2021 20:27:24.45…   192.168.0.1     192.168.0.101   DHCP         590            DHCP ACK      - Transaction ID 0x7d722d75
16…   Apr  1, 2021 20:56:51.22…   192.168.0.102   192.168.0.1     DHCP         342            DHCP Request  - Transaction ID 0xd8cde6a1
16…   Apr  1, 2021 20:56:51.22…   192.168.0.1     192.168.0.102   DHCP         590            DHCP ACK      - Transaction ID 0xd8cde6a1
18…   Apr  1, 2021 21:27:23.48…   192.168.0.101   192.168.0.1     DHCP         332            DHCP Request  - Transaction ID 0x7d722d75

>  Frame 2017: 332 bytes on wire (2656 bits), 332 bytes captured (2656 bits) on interface en6, id 0
>  Ethernet II, Src: Alertmec_1a:63:95 (00:1c:2b:1a:63:95), Dst: Tp-LinkT_02:4d:12 (d8:47:32:02:4d:12)
>  Internet Protocol Version 4, Src: 192.168.0.101, Dst: 192.168.0.1
>  User Datagram Protocol, Src Port: 68, Dst Port: 67
>  Dynamic Host Configuration Protocol (Request)
```

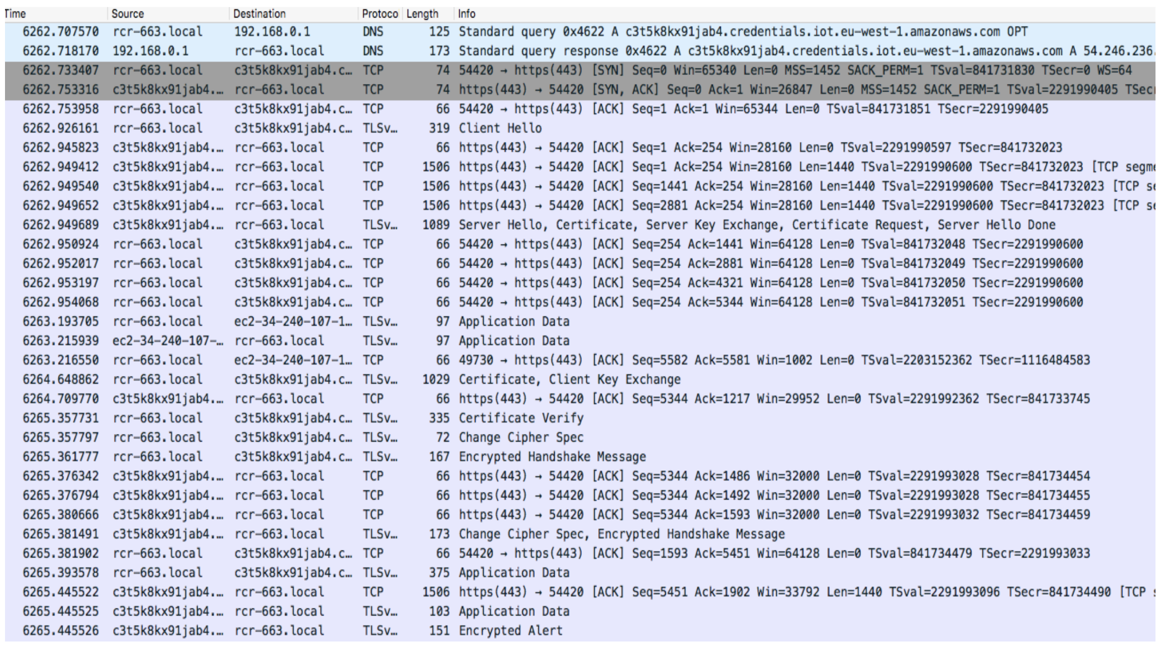
Figure 3.22 Hub DHCP packets

3.5.3 Protocols (idle & active states)

These devices both in idle and active state utilize several protocols. Idle state refers to when a device is connected to the hub but without any triggered activity like ON or OFF. Active state refers to when a device is connected to the hub and triggers take place, which results in extra traffic generation, thus the spike in certain protocols. The hub sends TCP Keep Alive messages every 14 and then every 19 seconds. This message tends to keep the hub awake to prevent the connection between the client (hub) and the server from breaking, which is why this takes place frequently. Another protocol is NTP (Network Time Protocol) taking place every 34 minutes. NTP is a protocol utilized by IoT devices, as very accurate timings are highly important in IoT communication. This happens periodically to synchronize their time with publicly available NTP servers. DNS requests are also made to four addresses every 3 to 4 hours. Other protocols observed were TLS, ARP, ICMPv6, DHCP and mDNS. This shows that the hub regardless of a device paired to it, or an event being triggered generates this traffic intermittently. By pairing a device to the hub and triggering events, the frequency of some of these protocols increase. The ratios of these protocols are compared over 2 hours when the hub is not paired with any device, when the hub is paired with devices but are in idle state and when there is activity (event triggers). This is shown in Figure 3.5. It is observed that the packet count of these protocols like DNS (52), TCP (3143) AND TLS (1423) drastically increase due to the presence of activity while NTP (8) and DHCP (4) remain fairly the same. The hub without devices paired has the least count of these protocols DNS (10), TCP (2239) TLS (900), NTP (8) and DHCP (4) followed by when devices are idle which shows a slight increase in DNS (22), TCP (2401) and TLS (1003) while NTP (8) and DHCP (4) remain the same. This protocol data gives us an idea on the frequency or count of the several protocols utilized by these devices over time. We can see that there is a limit or cap to how frequent these get generated which will be useful in DDoS attack mitigation strategies like rate limiting.

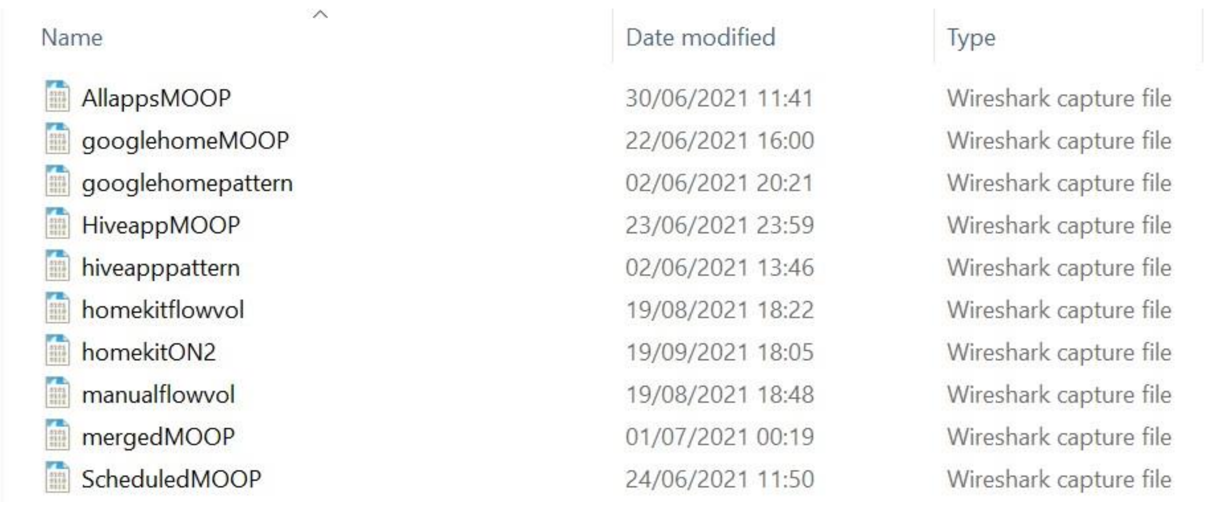
3.5.4 Flow volume and duration

Whenever an event is triggered, or boot and pairing modes are taking place or some periodic updates take place, a DNS query and response happens. A TCP connection is then established which involves a client and server handshake and a change of cipher spec between the smart devices and the DNS servers as shown in figure 3.23 under the info column. This entire process is referred to as a flow. The total number of bytes exchanged in this entire flow is known as the flow volume while the total time it takes for one complete flow is the flow duration. This was computed by getting the time difference between the first and last packet in that flow. Packets in a flow come in pairs consisting of a request and reply packet. Each packet also has a fixed length. Furthermore, a single flow comprises of several combination of protocols as we have seen. This can include DNS, TCP, TLS and mDNS. The flow volume and flow duration differ for each mode of operation and also when an event is triggered by one device compared to when multiple devices are triggered like a motion sensor detecting movement and triggering the light bulb to go ON. The Hive devices were tested in 5 different operation modes (manual, scheduled, Hive app, Google home app, Home kit app). This was carried out using several control devices, which are Samsung A12 phone, iPhone SE, and an iPad. This was done to make sure these discovered distinct patterns for each mode are uniform across a variety of control devices. Traffic was captured from all the above-mentioned control modes to identify a pattern for each mode and also the flow volume and duration as seen from figure 3.24. The trigger times for each mode of operation was noted to use these for cross referencing during analysis. Several packet header details were captured including source, destination, time, packet length, sequence number, protocol, and info (a column that gives extra details like packet sequence and labels). Figure 3.25 shows the varying flow volumes and duration for the different modes of operation.



```
Time          Source              Destination         Protoco Length Info
 6262.707570  rcr-663.local       192.168.0.1         DNS      125  Standard query 0x4622 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
 6262.718170  192.168.0.1         rcr-663.local       DNS      173  Standard query response 0x4622 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 54.246.236.
 6262.733407  rcr-663.local       c3t5k8kx91jab4.c…   TCP       74  54420 → https(443) [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=841731830 TSecr=0 WS=64
 6262.753316  c3t5k8kx91jab4.…    rcr-663.local       TCP       74  https(443) → 54420 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSval=2291990405 TSec
 6262.753958  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=841731851 TSecr=2291990405
 6262.926161  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…    319  Client Hello
 6262.945823  c3t5k8kx91jab4.…    rcr-663.local       TCP       66  https(443) → 54420 [ACK] Seq=1 Ack=254 Win=28160 Len=0 TSval=2291990597 TSecr=841732023
 6262.949412  c3t5k8kx91jab4.…    rcr-663.local       TCP     1506  https(443) → 54420 [ACK] Seq=1 Ack=254 Win=28160 Len=1440 TSval=2291990600 TSecr=841732023 [TCP segm
 6262.949540  c3t5k8kx91jab4.…    rcr-663.local       TCP     1506  https(443) → 54420 [ACK] Seq=1441 Ack=254 Win=28160 Len=1440 TSval=2291990600 TSecr=841732023 [TCP s
 6262.949652  c3t5k8kx91jab4.…    rcr-663.local       TCP     1506  https(443) → 54420 [ACK] Seq=2881 Ack=254 Win=28160 Len=1440 TSval=2291990600 TSecr=841732023 [TCP s
 6262.949689  c3t5k8kx91jab4.…    rcr-663.local       TLSv…   1089  Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
 6262.950924  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=254 Ack=1441 Win=64128 Len=0 TSval=841732048 TSecr=2291990600
 6262.952017  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=254 Ack=2881 Win=64128 Len=0 TSval=841732049 TSecr=2291990600
 6262.953197  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=254 Ack=4321 Win=64128 Len=0 TSval=841732050 TSecr=2291990600
 6262.954068  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=254 Ack=5344 Win=64128 Len=0 TSval=841732051 TSecr=2291990600
 6263.193705  rcr-663.local       ec2-34-240-107-1…   TLSv…     97  Application Data
 6263.215939  ec2-34-240-107-…    rcr-663.local       TLSv…     97  Application Data
 6263.216550  rcr-663.local       ec2-34-240-107-1…   TCP       66  49730 → https(443) [ACK] Seq=5582 Ack=5581 Win=1002 Len=0 TSval=2203152362 TSecr=1116484583
 6264.648862  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…   1029  Certificate, Client Key Exchange
 6264.709770  c3t5k8kx91jab4.…    rcr-663.local       TCP       66  https(443) → 54420 [ACK] Seq=5344 Ack=1217 Win=29952 Len=0 TSval=2291992362 TSecr=841733745
 6265.357731  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…    335  Certificate Verify
 6265.357797  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…     72  Change Cipher Spec
 6265.361777  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…    167  Encrypted Handshake Message
 6265.376342  c3t5k8kx91jab4.…    rcr-663.local       TCP       66  https(443) → 54420 [ACK] Seq=5344 Ack=1486 Win=32000 Len=0 TSval=2291993028 TSecr=841734454
 6265.376794  c3t5k8kx91jab4.…    rcr-663.local       TCP       66  https(443) → 54420 [ACK] Seq=5344 Ack=1492 Win=32000 Len=0 TSval=2291993028 TSecr=841734455
 6265.380666  c3t5k8kx91jab4.…    rcr-663.local       TCP       66  https(443) → 54420 [ACK] Seq=5344 Ack=1593 Win=32000 Len=0 TSval=2291993032 TSecr=841734459
 6265.381491  c3t5k8kx91jab4.…    rcr-663.local       TLSv…    173  Change Cipher Spec, Encrypted Handshake Message
 6265.381902  rcr-663.local       c3t5k8kx91jab4.c…   TCP       66  54420 → https(443) [ACK] Seq=1593 Ack=5451 Win=64128 Len=0 TSval=841734479 TSecr=2291993033
 6265.393578  rcr-663.local       c3t5k8kx91jab4.c…   TLSv…    375  Application Data
 6265.445522  c3t5k8kx91jab4.…    rcr-663.local       TCP     1506  https(443) → 54420 [ACK] Seq=5451 Ack=1902 Win=33792 Len=1440 TSval=2291993096 TSecr=841734490 [TCP
 6265.445525  c3t5k8kx91jab4.…    rcr-663.local       TLSv…    103  Application Data
 6265.445526  c3t5k8kx91jab4.…    rcr-663.local       TLSv…    151  Encrypted Alert
```

Figure 3.23 Client server handshake and cipher spec change.



Name	Date modified	Type
AllappsMOOP	30/06/2021 11:41	Wireshark capture file
googlehomeMOOP	22/06/2021 16:00	Wireshark capture file
googlehomepattern	02/06/2021 20:21	Wireshark capture file
HiveappMOOP	23/06/2021 23:59	Wireshark capture file
hiveapppattern	02/06/2021 13:46	Wireshark capture file
homekitflowvol	19/08/2021 18:22	Wireshark capture file
homekitON2	19/09/2021 18:05	Wireshark capture file
manualflowvol	19/08/2021 18:48	Wireshark capture file
mergedMOOP	01/07/2021 00:19	Wireshark capture file
ScheduledMOOP	24/06/2021 11:50	Wireshark capture file

Figure 3.24 Wireshark files for each MOOP

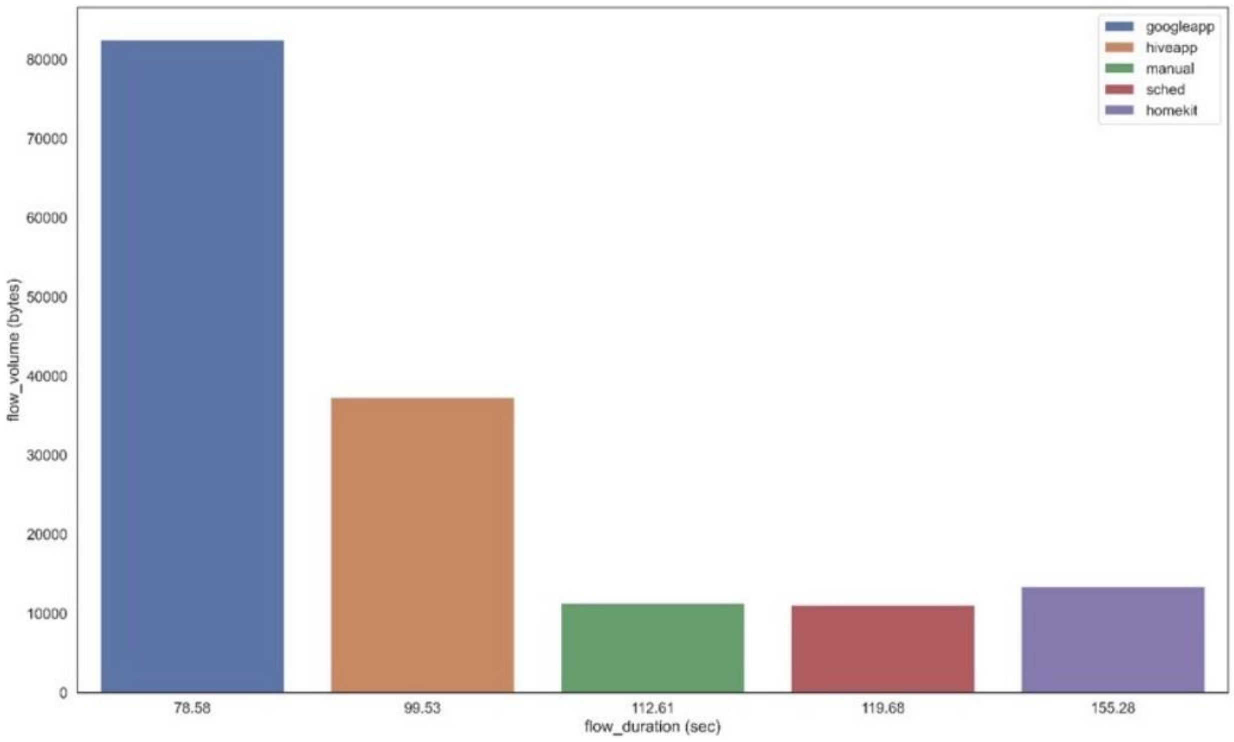
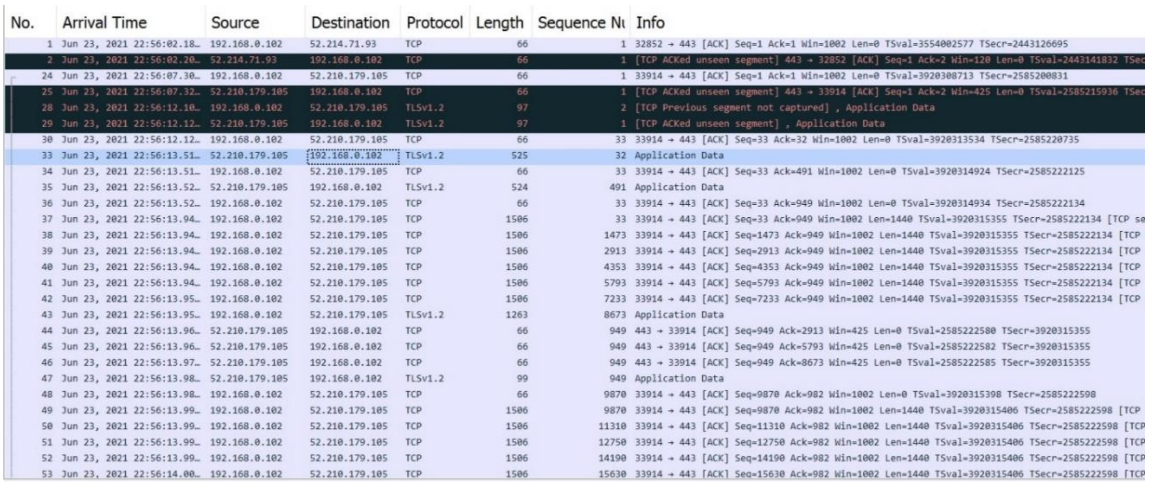


Figure 3.25 Device mode of operation by flow volume & duration

It is observed that when we use any of the smart phone apps (Hive, home kit, Google home) to control the devices, the flow volumes of these tend to be higher compared to when we operate the devices manually or the scheduled way, which have the same flow volume and duration. This increase in volume is due to the extra traffic generated because of using an application to control the devices. Mere opening any of the apps generates bytes of traffic without triggering an event. The duration can also vary because of extra time taken by the user to launch the app and further trigger an event as opposed to the scheduled and manual modes that has no delay involved during the flow, as there is less human intervention. Figure 3.26 shows a capture from the Hive app pattern.



No.	Arrival Time	Source	Destination	Protocol	Length	Sequence Nu	Info
1	Jun 23, 2021 22:56:02.18…	192.168.0.102	52.214.71.93	TCP	66	1	32852 → 443 [ACK] Seq=1 Ack=1 Win=1002 Len=0 TSval=3554002577 TSecr=2443126695
2	Jun 23, 2021 22:56:02.20…	52.214.71.93	192.168.0.102	TCP	66	1	[TCP ACKed unseen segment] 443 → 32852 [ACK] Seq=1 Ack=2 Win=120 Len=0 TSval=2443141832 TSec
24	Jun 23, 2021 22:56:07.30…	192.168.0.102	52.210.179.105	TCP	66	1	33914 → 443 [ACK] Seq=1 Ack=1 Win=1002 Len=0 TSval=3920308713 TSecr=2585200831
25	Jun 23, 2021 22:56:07.32…	52.210.179.105	192.168.0.102	TCP	66	1	[TCP ACKed unseen segment] 443 → 33914 [ACK] Seq=1 Ack=2 Win=425 Len=0 TSval=2585215936 TSec
28	Jun 23, 2021 22:56:12.10…	192.168.0.102	52.210.179.105	TLSv1.2	97	2	[TCP Previous segment not captured] , Application Data
29	Jun 23, 2021 22:56:12.12…	52.210.179.105	192.168.0.102	TLSv1.2	97	1	[TCP ACKed unseen segment] , Application Data
30	Jun 23, 2021 22:56:12.12…	192.168.0.102	52.210.179.105	TCP	66	33	33914 → 443 [ACK] Seq=33 Ack=32 Win=1002 Len=0 TSval=3920313534 TSecr=2585220735
33	Jun 23, 2021 22:56:13.51…	52.210.179.105	192.168.0.102	TLSv1.2	525	32	Application Data
34	Jun 23, 2021 22:56:13.51…	192.168.0.102	52.210.179.105	TCP	66	33	33914 → 443 [ACK] Seq=33 Ack=491 Win=1002 Len=0 TSval=3920314924 TSecr=2585222125
35	Jun 23, 2021 22:56:13.52…	52.210.179.105	192.168.0.102	TLSv1.2	524	491	Application Data
36	Jun 23, 2021 22:56:13.52…	192.168.0.102	52.210.179.105	TCP	66	33	33914 → 443 [ACK] Seq=33 Ack=949 Win=1002 Len=0 TSval=3920314934 TSecr=2585222134
37	Jun 23, 2021 22:56:13.94…	192.168.0.102	52.210.179.105	TCP	1506	33	33914 → 443 [ACK] Seq=33 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP se
38	Jun 23, 2021 22:56:13.94…	192.168.0.102	52.210.179.105	TCP	1506	1473	33914 → 443 [ACK] Seq=1473 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP
39	Jun 23, 2021 22:56:13.94…	192.168.0.102	52.210.179.105	TCP	1506	2913	33914 → 443 [ACK] Seq=2913 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP
40	Jun 23, 2021 22:56:13.94…	192.168.0.102	52.210.179.105	TCP	1506	4353	33914 → 443 [ACK] Seq=4353 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP
41	Jun 23, 2021 22:56:13.94…	192.168.0.102	52.210.179.105	TCP	1506	5793	33914 → 443 [ACK] Seq=5793 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP
42	Jun 23, 2021 22:56:13.95…	192.168.0.102	52.210.179.105	TCP	1506	7233	33914 → 443 [ACK] Seq=7233 Ack=949 Win=1002 Len=1440 TSval=3920315355 TSecr=2585222134 [TCP
43	Jun 23, 2021 22:56:13.95…	192.168.0.102	52.210.179.105	TLSv1.2	1263	8673	Application Data
44	Jun 23, 2021 22:56:13.96…	52.210.179.105	192.168.0.102	TCP	66	949	443 → 33914 [ACK] Seq=949 Ack=2913 Win=425 Len=0 TSval=2585222580 TSecr=3920315355
45	Jun 23, 2021 22:56:13.96…	52.210.179.105	192.168.0.102	TCP	66	949	443 → 33914 [ACK] Seq=949 Ack=5793 Win=425 Len=0 TSval=2585222582 TSecr=3920315355
46	Jun 23, 2021 22:56:13.97…	52.210.179.105	192.168.0.102	TCP	66	949	443 → 33914 [ACK] Seq=949 Ack=8673 Win=425 Len=0 TSval=2585222585 TSecr=3920315355
47	Jun 23, 2021 22:56:13.98…	52.210.179.105	192.168.0.102	TLSv1.2	99	949	Application Data
48	Jun 23, 2021 22:56:13.98…	192.168.0.102	52.210.179.105	TCP	66	9870	33914 → 443 [ACK] Seq=9870 Ack=982 Win=1002 Len=0 TSval=3920315398 TSecr=2585222598
49	Jun 23, 2021 22:56:13.99…	192.168.0.102	52.210.179.105	TCP	1506	9870	33914 → 443 [ACK] Seq=9870 Ack=982 Win=1002 Len=1440 TSval=3920315406 TSecr=2585222598 [TCP
50	Jun 23, 2021 22:56:13.99…	192.168.0.102	52.210.179.105	TCP	1506	11310	33914 → 443 [ACK] Seq=11310 Ack=982 Win=1002 Len=1440 TSval=3920315406 TSecr=2585222598 [TCP
51	Jun 23, 2021 22:56:13.99…	192.168.0.102	52.210.179.105	TCP	1506	12750	33914 → 443 [ACK] Seq=12750 Ack=982 Win=1002 Len=1440 TSval=3920315406 TSecr=2585222598 [TCP
52	Jun 23, 2021 22:56:13.99…	192.168.0.102	52.210.179.105	TCP	1506	14190	33914 → 443 [ACK] Seq=14190 Ack=982 Win=1002 Len=1440 TSval=3920315406 TSecr=2585222598 [TCP
53	Jun 23, 2021 22:56:14.00…	192.168.0.102	52.210.179.105	TCP	1506	15630	33914 → 443 [ACK] Seq=15630 Ack=982 Win=1002 Len=1440 TSval=3920315406 TSecr=2585222598 [TCP

Figure 3.26 Hive app flow volume and duration traffic

The flow volume is calculated by taking the total bytes using the individual packet lengths of each packet in the flow from the beginning which starts at packet no 24 in figure 3.26. The flow duration is arrived at by comparing the arrival times of the first and last packet in the flow to get the time difference. This was done for each mode of operation thus arriving at figure 3.25 showing the respective flow volumes and duration of each MOOP.

Figure 3.27 also shows the varying flow volumes and duration of a triggered event originating from one device compared to when the motion sensor triggers the plug and bulb to go ON (integrated form). Both the flow volume and duration of the traffic that originated because of multiple devices being active at the same time is higher. This helps give an idea of the maximum flow volume to expect at times of operating multiple devices making the volume to peak which can be useful in terms of traffic rate limiting as part of DDoS attack mitigation.

3.5.5 Traffic pattern based on mode of operation

As mentioned earlier, controlling the smart home devices was done using 5 different modes, with each mode monitored and analysed in isolation. This led to some interesting observations whereby each mode exhibited a distinct pattern in terms of the protocol and packet length traffic sequence. This was tested using three different devices (iPhone, iPad, Samsung smart phone) to confirm this exhibited unique sequence for each mode is uniform regardless of the platform used to control it. After several repetition of the test cases this uniformity was validated.

Whenever an event is triggered like ON/OFF, a DNS query and response takes place between the device and a kinesis DNS server which is responsible for triggered events of this sort. This became a

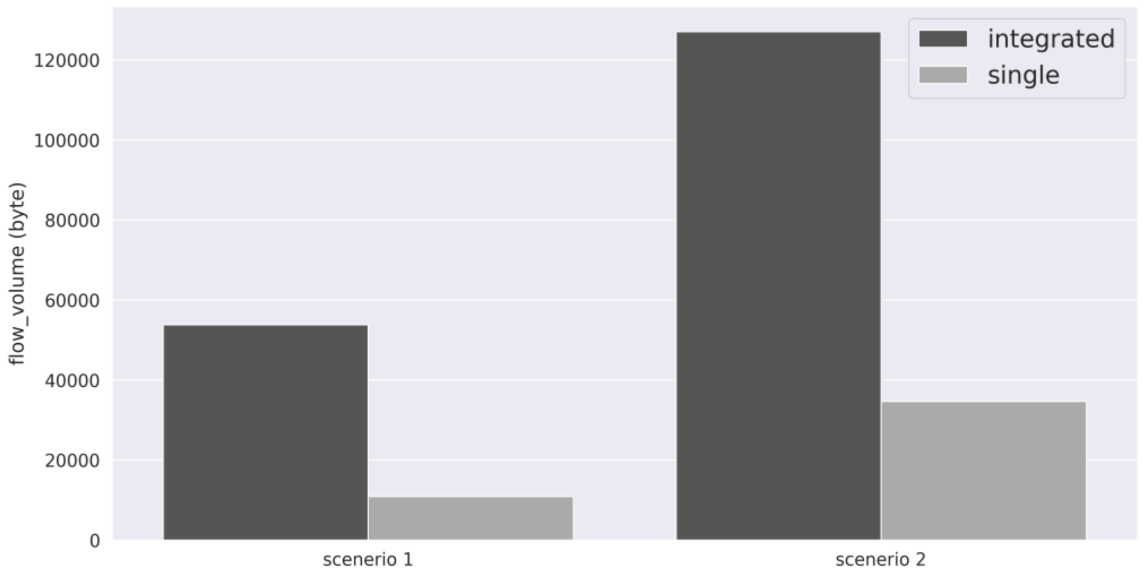


Figure 3.27 Single & integrated device traffic compared by flow volume

baseline for identifying triggered events as this server is always queried when such happens. This was found to be uniform across all control devices. However, the preceding packets right before this DNS packet were observed to have a unique sequential combination in terms of protocol and packet length for each mode of operation. Each packet is paired with a corresponding protocol and packet length, among other network details. Table 3.3 shows the unique pattern attributed to each mode of operation. We can see that each mode has a different combination of protocols and packet lengths except the manual and scheduled modes which have an identical pattern, thus categorized together. This order for both protocol and packet length were found to be consistent for each MOOP when there is a trigger.

Table 3.3 MOOP protocol & packet length sequence

Mode of operation (MOOP)	Protocol + Packet length order
Hive app	TCP+1506 TLSv1.2+1019 TCP+66 TLSv1.2+99 TCP+66 DNS+102 DNS+118
Google home app	MDNS+103 TCP+1506 TLSv1.2+1003 TCP+66 TLSv1.2+99 TCP+66 DNS+102 DNS+118
Home kit app	TCP+275 TCP+66 TCP+111 TCP+66 TCP+60 TCP+66 DNS+102 DNS+118
Manual and scheduled	TLSv1.2+97 TLSv1.2+97 TCP+66 DNS+102 DNS+118

3.6 Comparison to literature and new findings

This chapter has analysed several behavioural aspects of Hive home devices. An EDA was performed on the unencrypted features from the captured logs addressing traffic categorization, device identification, protocols in both Idle and active states, flow volume (total number of incoming and outgoing bytes in one cycle), flow duration (time it takes from the beginning of a flow to the end) and traffic sequence based on the mode of operation. Based on these logs it conforms to [118] about generally categorizing M2M generated traffic into 3, which are periodic update, event driven and payload exchange. Furthermore, this research also agrees with [119] where it states that IoT communicates with a number of fixed DNS servers. Several protocols were found to be utilized by these devices, which vary in volume and duration depending on the device state (active or idle). On the aspect of device identification, the motion sensor and the hub have their unique patterns, but this was not the case with the plug and bulb as they had an identical pattern. This could be due to their similar basic functionalities of ON and OFF. Other similarities shared by all the devices are communication with the same DNS servers, server port numbers and protocols among others. This conforms to the findings in [57] that devices from the same vendor behave in a very similar manner. The idle and active moments of these devices can also be identified based on the drastic increase in the volume or count of certain protocols when active as we have seen in this study.

During analysis, some observations were made which led to new findings. These are as follows:

- Distinct flow volumes and duration for each mode of operation as shown in figure 3.25. It was observed that while using the apps the flow volume in bytes was higher than when operating the devices in the manual or scheduled mode. This is due to more traffic being generated from mere opening of the app. The flow duration also differed for each mode. This provides a signature relating to how these devices are operated which can be used to detect an unauthorised user.
- Unique traffic patterns based on protocol and packet length are also exhibited for each of the modes as shown in table 3.3. We can see that the packets before the DNS query during event triggers have a unique sequence for each MOOP. However, the manual and scheduled modes have identical patterns which is also the case for flow volume and duration. This can also be used as a signature to know what mode is used for operating these devices.

These new findings can be used in forensic investigations to prove how someone controlled a particular device or devices and whether they were present at the scene during some specified times. For instance, if the evidence shows proof of manual mode of operation, then this ties one to physically being at the premises. Furthermore, as each operation mode has a unique traffic pattern, these patterns could be whitelisted on the smart home network to detect certain attacks relating to unauthorized control of device which might have a deviating pattern from the whitelisted ones. For instance, if a user normally operates their device manually between 8am and 10am, and for some reason the device gets operated using another mode like one of the apps, then this should raise a flag for unauthorised control.

3.7 Summary

This chapter has covered several aspects relating to the behaviour of Hive home devices both in their lone state and as a collective. Traffic was collected from a real-life hive home network with an EDA performed on it. The areas covered are traffic categorization, device identification, protocols in both Idle and active states, flow volume (total number of incoming and outgoing bytes in one cycle), flow duration (time it takes from the beginning of a flow to the end) and traffic sequence based on the mode of operation. Several observations have been made which agrees with existing literature with regards to the behaviour of IoT devices in general. New findings were also derived from the EDA results which can be applied in forensic investigations and detection of unauthorised control of smart home device by an attacker.

Chapter 4

Exploratory Data Analysis comparing attack and benign smart home traffic properties

4.1 Introduction

As the previous chapter has dealt with the study of normal smart home network behaviour, this chapter extends to study how these devices behave under the influence of DDoS flooding attacks in real time. Both behavioural properties (attack and benign) are compared to see what aspects differ. This will aid in designing a timely and more effective DDoS visualization and detection models as we will see in subsequent sections. The findings from this EDA led to a proposed novel DDoS detection approach as covered in this chapter. This chapter covers the following areas in general:

- It collects DDoS and benign traffic in a real smart home environment and performs an Exploratory Data Analysis (EDA), visualizing the behavioural pattern of 3 types of DDoS flooding attacks when targeted at smart home networks in comparison to the benign smart home traffic pattern. The attacks covered are TCP SYN, ICMP and UDP flooding attacks.
- For each of the covered attacks, specific smart home traffic properties were selected, correlated, and visualized showing their reversed behaviour during an attack compared to their normal benign nature.
- To further validate the findings, public IoT datasets were analysed in the same manner and the same results were achieved.
- It presents 3 principles derived from the EDA based on which DDoS flooding attack traffic can be identified.
- Finally, it presents a DDoS detection approach by integrating the 3 principles derived from the EDA which are feature variance, absent features, and feature range. It discusses the significance of this approach in comparison to present approaches.

The sub-sections in this chapter covers the methodology used, attack propagation, EDA on attack traffic, comparison of attack and benign traffic EDA, new findings, comparison with literature and finally a summary of the chapter.

4.2 Methodology

The various processes and sub processes followed in this phase are broken down and explained in this section as shown in figure 4.1. It has three main phases, which are data collection, Exploratory Data Analysis and proposed novel method of DDoS detection based on the EDA results. This method is designed in such a way that one phase and its sub phases are needed to be completed before moving to the next. This ensures that a complete understanding and outcome of each phase is clearly drawn out before proceeding. As this chapter is aimed at studying attack behaviours in comparison to normal network behaviours and further proposing a better method to detect these attacks, this methodology provides the suitable execution points to achieve this in the following ways:

- ✓ The data collection phase anticipates for an unbiased EDA and evaluation of the proposed detection approach thereby making provision for credible public data. This was used to showcase that the same observations were arrived at in terms of traffic properties and patterns for both the public and private data covering attack and benign traffic.
- ✓ Using EDA as a method for the data analysis provided clearer and more efficient way to represent the multidimensional nature of the datasets. The graphical or visual aspects of the EDA also makes it easier to understand as the human brain tends to better process images than texts, thus giving a clear picture of what both an attack and normal traffic looks like with very visible differences.
- ✓ To design an effective DDoS detection system, there is a need to have an in-depth understanding with regards to the attack pattern and network changes that occur during the attack and in the process monitoring the most affected network properties. These can be used as a baseline for attack identification. This methodology paved way for the prerequisite of having the in-depth understanding from the EDA thus, giving rise to the proposed detection approach with ease and better clarity referencing facts and observations derived from the EDA.

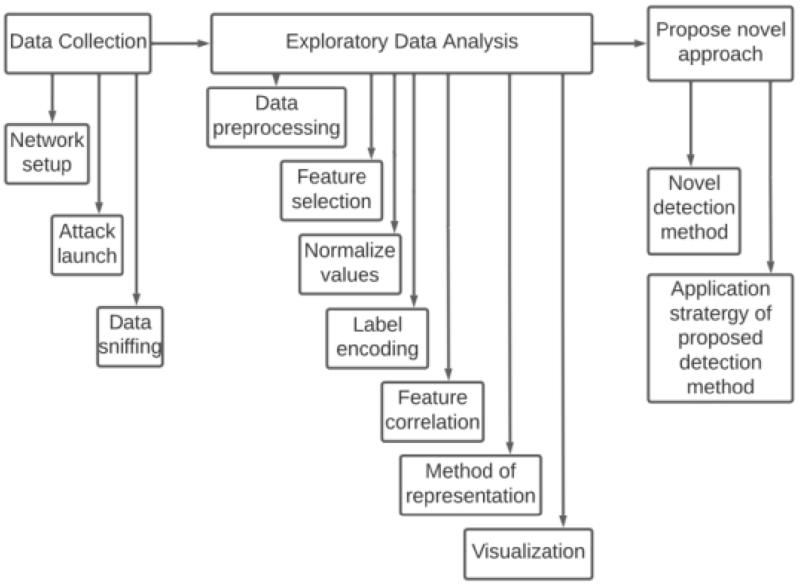


Figure 4.1 Research methodology

4.2.1 Data Collection

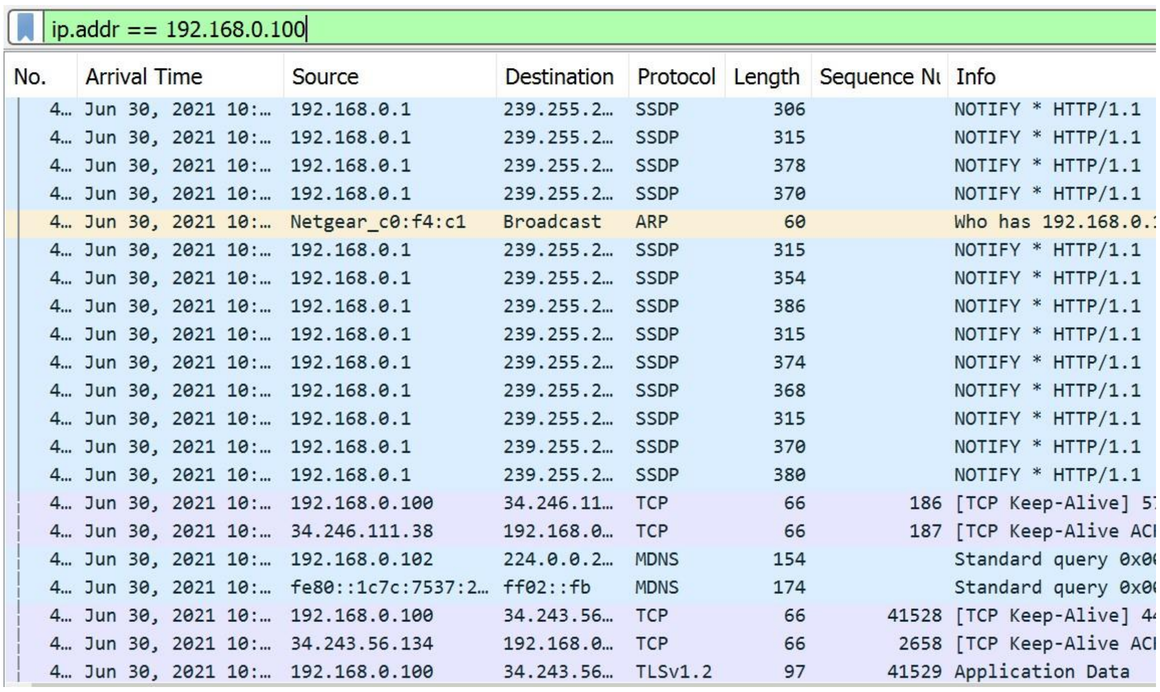
- Setup: The network topology to be used for data generation and collection is the same one from figure 3.1 in the previous chapter.
- Benign data: Normal smart home traffic was collected here, which was generated from using the smart home devices. Public datasets were also sourced for validation purposes.
- Attack data: 6 types of DDoS flooding attacks (TCP, UDP, ICMP, HTTP, SLOW LOIC, RECOIL) were launched on the smart home network and the traffic from this attack was collected. Public datasets were also sourced for validation. HTTP, SLOW LOIC and RECOIL attacks are collected for testing purposes in the next chapter.

4.2.2 Exploratory Data Analysis

- Data pre-processing: Each dataset was filtered to have only the relevant traffic flows from the target devices needed for analysis. This involves eliminating background traffic generated by other devices on the network. An instance is shown in figure 4.2 where several other IP addresses and protocols appear in the traffic. However only IP address 192.168.0.100 is relevant in this case, which is the smart hub's address, thus the filter syntax applied at the very top of the figure to get only this relevant traffic.
- Feature selection: The corresponding attack and benign network traffic properties to be analysed were selected and extracted. The most affected benign traffic properties during an attack were chosen and filtered out. This was done for each attack. These properties include protocol, packet length, sequence number and TCP flags bearing in mind the time stamps and packet ID or frame number for each packet. Figure 4.3 shows the selected features which will be converted to csv and exported for further analysis. Figure 4.4 shows this converted and exported file.
- Data normalization: Among the selected network features, those with a wide range of numeric values were normalized using the min-max scalar to have a more befitting range during visual representation. These features are packet length and packet sequence. Figure 4.5 shows the packet length scaling for TCP SYN dataset and a benign dataset.
- Label encoding: Non numerical values like the various protocols were encoded using a numerical value. This means each protocol corresponds to a number on the plotted figures. This is shown in figure 4.6 where encoding labels are applied to TCP SYN and benign dataset.
- Feature correlation: The selected benign features were analysed side by side to find out if they get affected simultaneously during an attack. This was tested for each of the 3 attacks covered.

➢ Method of representation: Various methods of visual representation were used like bar charts, pie charts, frequency polygons and scatter plots. The most befitting method of representation was chosen based on the network feature(s) being visualized.

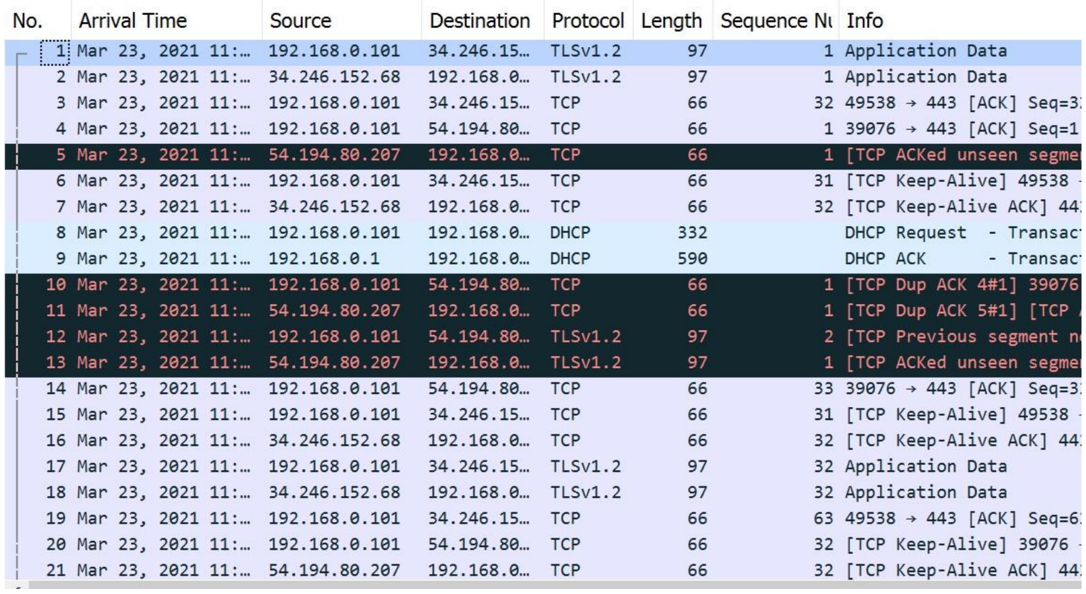
➢ Visualization: Python programming was used on Google Colab [120] to plot the charts and graphs. Corresponding network features for each attack and benign scenario were compared. Each of the analysed network feature (protocol, packet length, sequence number, TCP flag) was plotted against the respective frame number of the corresponding packet.



ip.addr == 192.168.0.100

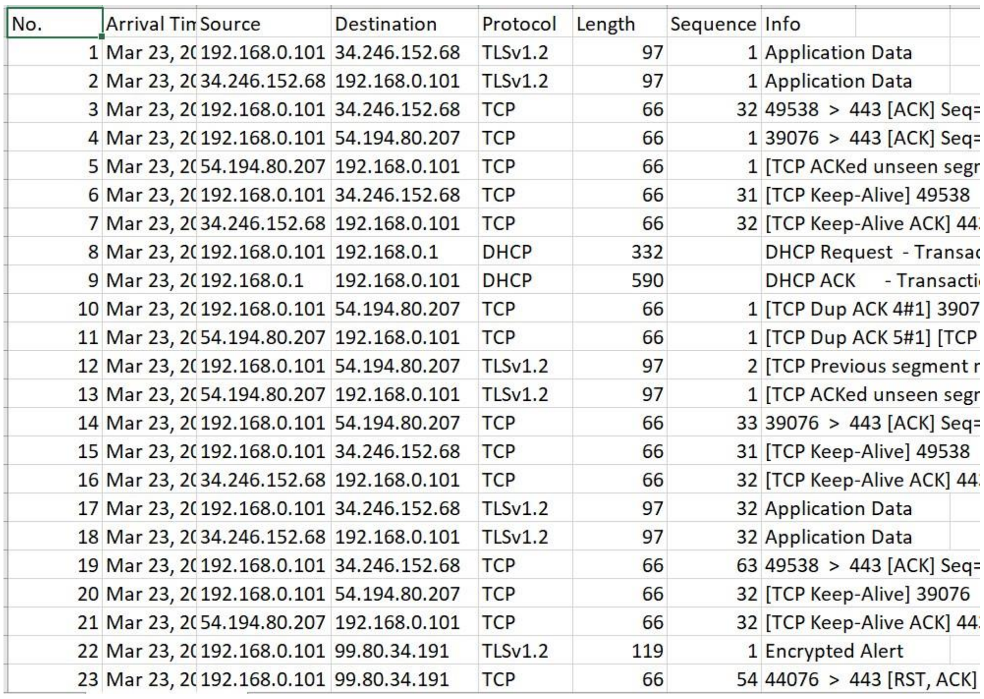
No.	Arrival Time	Source	Destination	Protocol	Length	Sequence Nu	Info
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	306		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	315		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	378		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	370		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	Netgear_c0:f4:c1	Broadcast	ARP	60		Who has 192.168.0.:
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	315		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	354		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	386		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	315		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	374		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	368		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	315		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	370		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.1	239.255.2…	SSDP	380		NOTIFY * HTTP/1.1
4…	Jun 30, 2021 10:…	192.168.0.100	34.246.11…	TCP	66	186	[TCP Keep-Alive] 5
4…	Jun 30, 2021 10:…	34.246.111.38	192.168.0…	TCP	66	187	[TCP Keep-Alive ACI
4…	Jun 30, 2021 10:…	192.168.0.102	224.0.0.2…	MDNS	154		Standard query 0x0
4…	Jun 30, 2021 10:…	fe80::1c7c:7537:2…	ff02::fb	MDNS	174		Standard query 0x0
4…	Jun 30, 2021 10:…	192.168.0.100	34.243.56…	TCP	66	41528	[TCP Keep-Alive] 4
4…	Jun 30, 2021 10:…	34.243.56.134	192.168.0…	TCP	66	2658	[TCP Keep-Alive ACI
4…	Jun 30, 2021 10:…	192.168.0.100	34.243.56…	TLSv1.2	97	41529	Application Data

Figure 4.2 Filtering traffic



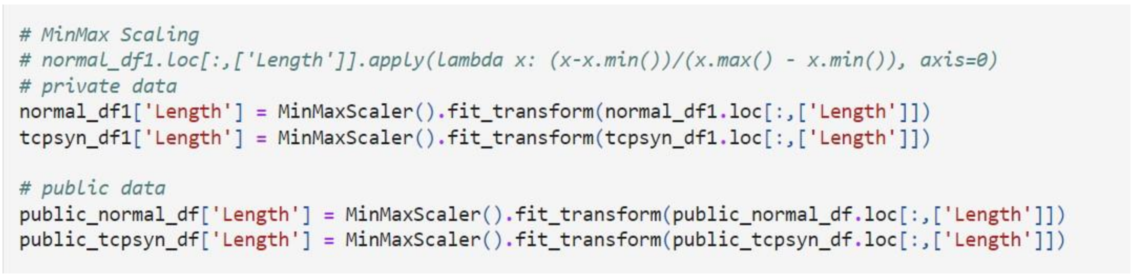
No.	Arrival Time	Source	Destination	Protocol	Length	Sequence Nu	Info
1	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TLSv1.2	97	1	Application Data
2	Mar 23, 2021 11:…	34.246.152.68	192.168.0…	TLSv1.2	97	1	Application Data
3	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TCP	66	32	49538 → 443 [ACK] Seq=3
4	Mar 23, 2021 11:…	192.168.0.101	54.194.80…	TCP	66	1	39076 → 443 [ACK] Seq=1
5	Mar 23, 2021 11:…	54.194.80.207	192.168.0…	TCP	66	1	[TCP ACKed unseen segme
6	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TCP	66	31	[TCP Keep-Alive] 49538
7	Mar 23, 2021 11:…	34.246.152.68	192.168.0…	TCP	66	32	[TCP Keep-Alive ACK] 44
8	Mar 23, 2021 11:…	192.168.0.101	192.168.0…	DHCP	332		DHCP Request - Transac
9	Mar 23, 2021 11:…	192.168.0.1	192.168.0…	DHCP	590		DHCP ACK - Transac
10	Mar 23, 2021 11:…	192.168.0.101	54.194.80…	TCP	66	1	[TCP Dup ACK 4#1] 39076
11	Mar 23, 2021 11:…	54.194.80.207	192.168.0…	TCP	66	1	[TCP Dup ACK 5#1] [TCP
12	Mar 23, 2021 11:…	192.168.0.101	54.194.80…	TLSv1.2	97	2	[TCP Previous segment n
13	Mar 23, 2021 11:…	54.194.80.207	192.168.0…	TLSv1.2	97	1	[TCP ACKed unseen segme
14	Mar 23, 2021 11:…	192.168.0.101	54.194.80…	TCP	66	33	39076 → 443 [ACK] Seq=3
15	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TCP	66	31	[TCP Keep-Alive] 49538
16	Mar 23, 2021 11:…	34.246.152.68	192.168.0…	TCP	66	32	[TCP Keep-Alive ACK] 44
17	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TLSv1.2	97	32	Application Data
18	Mar 23, 2021 11:…	34.246.152.68	192.168.0…	TLSv1.2	97	32	Application Data
19	Mar 23, 2021 11:…	192.168.0.101	34.246.15…	TCP	66	63	49538 → 443 [ACK] Seq=6
20	Mar 23, 2021 11:…	192.168.0.101	54.194.80…	TCP	66	32	[TCP Keep-Alive] 39076
21	Mar 23, 2021 11:…	54.194.80.207	192.168.0…	TCP	66	32	[TCP Keep-Alive ACK] 44

Figure 4.3 Selected features



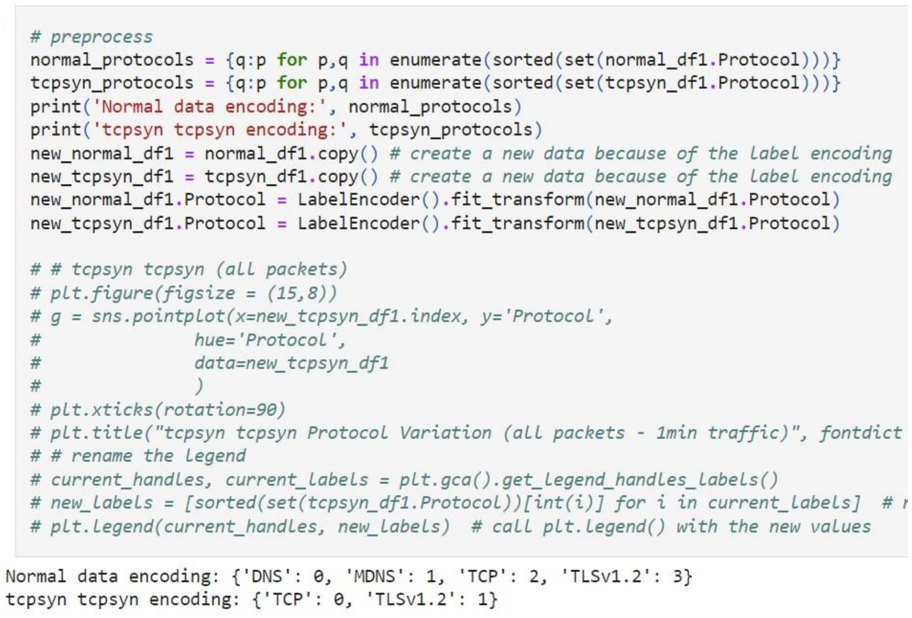
No.	Arrival Tin	Source	Destination	Protocol	Length	Sequence	Info
1	Mar 23, 2(	192.168.0.101	34.246.152.68	TLSv1.2	97	1	Application Data
2	Mar 23, 2(	34.246.152.68	192.168.0.101	TLSv1.2	97	1	Application Data
3	Mar 23, 2(	192.168.0.101	34.246.152.68	TCP	66	32	49538 > 443 [ACK] Seq=
4	Mar 23, 2(	192.168.0.101	54.194.80.207	TCP	66	1	39076 > 443 [ACK] Seq=
5	Mar 23, 2(	54.194.80.207	192.168.0.101	TCP	66	1	[TCP ACKed unseen segr
6	Mar 23, 2(	192.168.0.101	34.246.152.68	TCP	66	31	[TCP Keep-Alive] 49538
7	Mar 23, 2(	34.246.152.68	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 44
8	Mar 23, 2(	192.168.0.101	192.168.0.1	DHCP	332		DHCP Request - Transac
9	Mar 23, 2(	192.168.0.1	192.168.0.101	DHCP	590		DHCP ACK - Transacti
10	Mar 23, 2(	192.168.0.101	54.194.80.207	TCP	66	1	[TCP Dup ACK 4#1] 3907
11	Mar 23, 2(	54.194.80.207	192.168.0.101	TCP	66	1	[TCP Dup ACK 5#1] [TCP
12	Mar 23, 2(	192.168.0.101	54.194.80.207	TLSv1.2	97	2	[TCP Previous segment r
13	Mar 23, 2(	54.194.80.207	192.168.0.101	TLSv1.2	97	1	[TCP ACKed unseen segr
14	Mar 23, 2(	192.168.0.101	54.194.80.207	TCP	66	33	39076 > 443 [ACK] Seq=
15	Mar 23, 2(	192.168.0.101	34.246.152.68	TCP	66	31	[TCP Keep-Alive] 49538
16	Mar 23, 2(	34.246.152.68	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 44
17	Mar 23, 2(	192.168.0.101	34.246.152.68	TLSv1.2	97	32	Application Data
18	Mar 23, 2(	34.246.152.68	192.168.0.101	TLSv1.2	97	32	Application Data
19	Mar 23, 2(	192.168.0.101	34.246.152.68	TCP	66	63	49538 > 443 [ACK] Seq=
20	Mar 23, 2(	192.168.0.101	54.194.80.207	TCP	66	32	[TCP Keep-Alive] 39076
21	Mar 23, 2(	54.194.80.207	192.168.0.101	TCP	66	32	[TCP Keep-Alive ACK] 44
22	Mar 23, 2(	192.168.0.101	99.80.34.191	TLSv1.2	119	1	Encrypted Alert
23	Mar 23, 2(	192.168.0.101	99.80.34.191	TCP	66	54	44076 > 443 [RST, ACK]

Figure 4.4 PCAP converted to CSV



```
# MinMax Scaling
# normal_df1.loc[:,['Length']].apply(lambda x: (x-x.min())/(x.max() - x.min()), axis=0)
# private data
normal_df1['Length'] = MinMaxScaler().fit_transform(normal_df1.loc[:,['Length']])
tcpsyn_df1['Length'] = MinMaxScaler().fit_transform(tcpsyn_df1.loc[:,['Length']])

# public data
public_normal_df['Length'] = MinMaxScaler().fit_transform(public_normal_df.loc[:,['Length']])
public_tcpsyn_df['Length'] = MinMaxScaler().fit_transform(public_tcpsyn_df.loc[:,['Length']])
```

Figure 4.5 Packet length scaling



```
# preprocess
normal_protocols = {q:p for p,q in enumerate(sorted(set(normal_df1.Protocol)))}
tcpsyn_protocols = {q:p for p,q in enumerate(sorted(set(tcpsyn_df1.Protocol)))}
print('Normal data encoding:', normal_protocols)
print('tcpsyn tcpsyn encoding:', tcpsyn_protocols)
new_normal_df1 = normal_df1.copy() # create a new data because of the label encoding
new_tcpsyn_df1 = tcpsyn_df1.copy() # create a new data because of the label encoding
new_normal_df1.Protocol = LabelEncoder().fit_transform(new_normal_df1.Protocol)
new_tcpsyn_df1.Protocol = LabelEncoder().fit_transform(new_tcpsyn_df1.Protocol)

# # tcpsyn tcpsyn (all packets)
# plt.figure(figsize = (15,8))
# g = sns.pointplot(x=new_tcpsyn_df1.index, y='Protocol',
#                 hue='Protocol',
#                 data=new_tcpsyn_df1
#                 )
# plt.xticks(rotation=90)
# plt.title("tcpsyn tcpsyn Protocol Variation (all packets - 1min traffic)", fontdict
# # rename the legend
# current_handles, current_labels = plt.gca().get_legend_handles_labels()
# new_labels = [sorted(set(tcpsyn_df1.Protocol))[int(i)] for i in current_labels]  # r
# plt.legend(current_handles, new_labels)  # call plt.legend() with the new values
```

```
Normal data encoding: {'DNS': 0, 'MDNS': 1, 'TCP': 2, 'TLSv1.2': 3}
tcpsyn tcpsyn encoding: {'TCP': 0, 'TLSv1.2': 1}
```

Figure 4.6 Label Encoding

4.2.3 Proposed novel approach

- Propose novel detection method: A novel DDoS identification approach is proposed based on the derived EDA results.
- Novel method application strategies: Ways by which the proposed novel method can be incorporated into the smart home network for better DDoS detection are outlined.

4.3 Attack data collection

The smart home devices used in this study include a smart hub (to integrate the smart devices), motion sensor, smart plug and bulb. The network communication that takes place when these devices are being flooded with DDoS traffic is the main point of interest, thus a setup to collect this traffic for further analysis. Six types of DDoS flooding attacks were launched on the smart devices with the smart hub serving as the main gateway. These are TCPSYN, UDP, ICMP, HTTP, RECOIL, SLOWLOIC and mixed attacks. Low Orbit Ion Cannon (LOIC) [121] and Hping3 [122] of Kali Linux suit [123] coupled with Wireshark [124] as packet analyser are the tools used for these attacks. For each attack, the target IP address and flooding rate of packets was specified and then launched. Each attack was directed to the target device using 4 different machines on the same private network to make it a distributed attack. Traffic generated from/to the target smart home device (hub) was captured separately for each attack to know the network changes that relate to each attack. To get very detailed network traffic, the capture setup was made to collect traffic at layer 2 (datalink). This was done by connecting the hub to port 1 of the switch. Port 8 of the switch was then connected to the router (for internet connection). To capture all that flowed in and out of the hub and all devices paired to it, port 1 was mirrored on port 4. Port 4 was connected to the laptop using a Local Area Network (LAN) cable and Wireshark was used to capture this traffic. This connection is shown in figure 4.7.

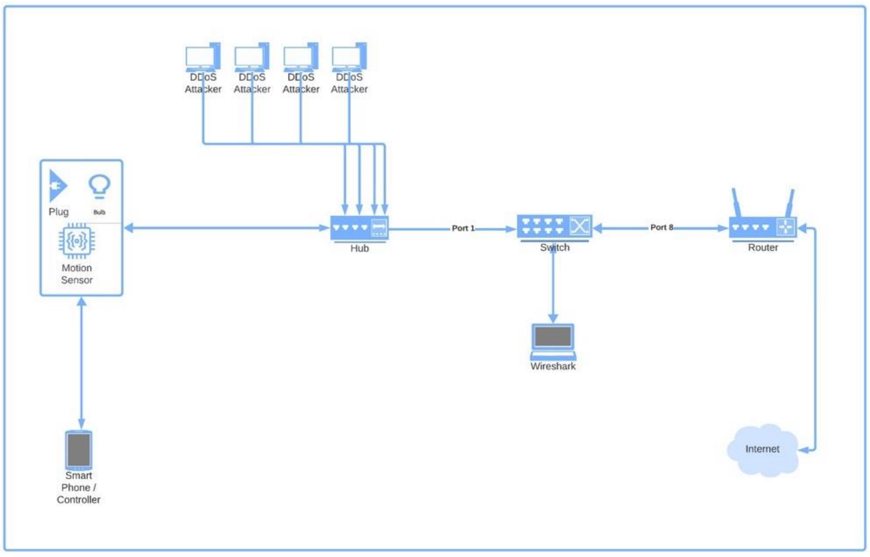
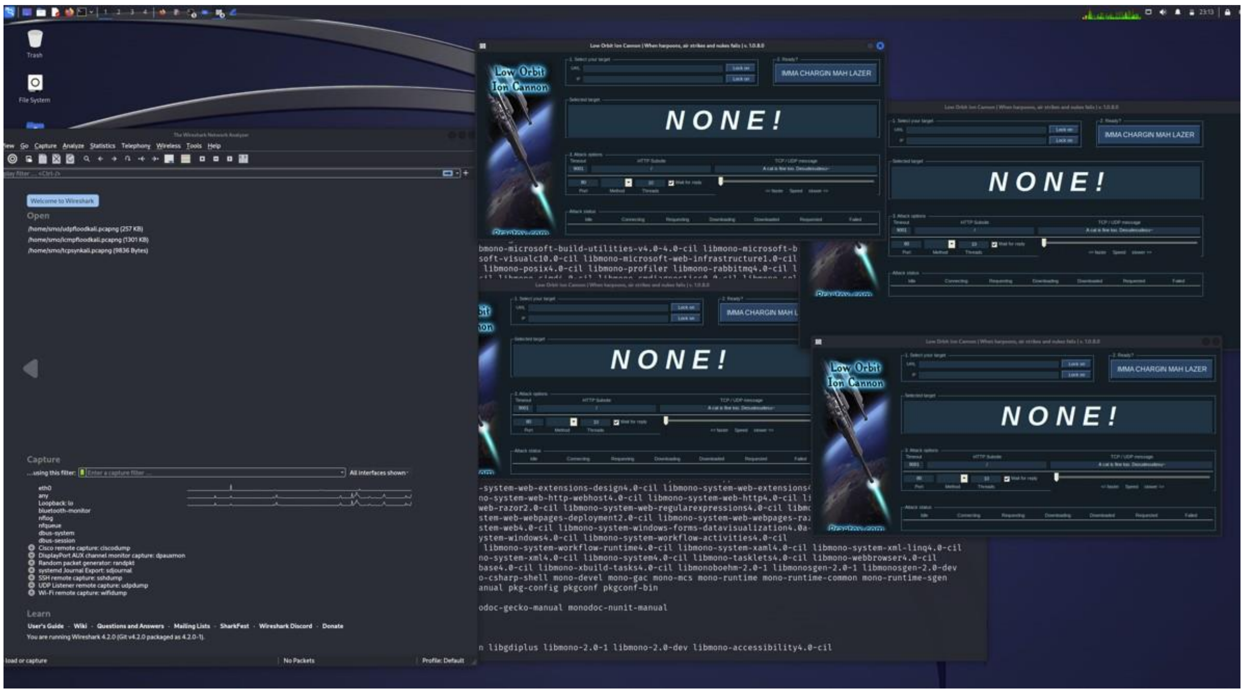


Figure 4.7 Attack data collection

Figure 4.8 shows the launching of LOIC on 4 virtual machines and Wireshark. Each of these LOIC windows will be used to launch the same attack in parallel with others. Figure 4.9 shows the launch of TCP flood attack. The IP address of the smart hub which is 192.168.0.103 is typed in the box as specified target IP. The attack thread is set to 1000 and the type of attack is chosen from the drop-down menu in this case TCP. The port number is specified as port 80. After populating these details, the **immachargingmalazer** button is clicked to initialize the attack. Figure 4.9 shows the attack traffic being captured by in the Wireshark window. The same process is repeated to launch HTTP, UDP, ICMP, RECOIL, SLOWLOIC and mixed attack. Figures 4.10 and 4.11 show the launch of HTTP and UDP attacks.

As mentioned, Hping3 on Kali Linux is also used for attack propagation. Figure 4.12 shows the command to launch the TCP flood attack. The type of attack is specified as **-s** which is TCP, the attack is set to flood using **--flood**, the port number of hub is specified as **-p49808** and the hubs IP address as target which is **192.168.0.103.** On the left side in the Wireshark window, we can see the flood packets to the hubs IP address. Figure 4.13 shows the UDP flood attack. A similar syntax is used to the TCP flood however UDP is specified here by using **--2**. The UDP flood packets can be seen on the Wireshark window. Figure 4.14 shows the ICMP flood attack. This is signified by --**1** in the syntax. The flood packets are seen in the Wireshark window as well.



Low Orbit Ion Cannon
IMMA CHARGIN MAH LAZER
NONE!
NONE!
NONE!
NONE!
Welcome to Wireshark
Open
/home/smo/udpfloodkali.pcapng (257 KB)
/home/smo/icmpfloodkali.pcapng (1301 KB)
/home/smo/tcpsynkali.pcapng (9836 Bytes)
Capture
Learn

```
bmono-microsoft-build-utilities-v4.0-4.0-cil libmono-microsoft-b
soft-visualc10.0-cil libmono-microsoft-web-infrastructure1.0-cil
 libmono-posix4.0-cil libmono-profiler libmono-rabbitmq4.0-cil l
```

```
-system-web-extensions-design4.0-cil libmono-system-web-extensions
no-system-web-http-webhost4.0-cil libmono-system-web-http4.0-cil li
web-razor2.0-cil libmono-system-web-regularexpressions4.0-cil libmo
stem-web-webpages-deployment2.0-cil libmono-system-web-webpages-ra
stem-web4.0-cil libmono-system-windows-forms-datavisualization4.0a-
ystem-windows4.0-cil libmono-system-workflow-activities4.0-cil
 libmono-system-workflow-runtime4.0-cil libmono-system-xaml4.0-cil libmono-system-xml-linq4.0-cil
no-system-xml4.0-cil libmono-system4.0-cil libmono-tasklets4.0-cil libmono-webbrowser4.0-cil
base4.0-cil libmono-xbuild-tasks4.0-cil libmonoboehm-2.0-1 libmonosgen-2.0-1 libmonosgen-2.0-dev
o-csharp-shell mono-devel mono-gac mono-mcs mono-runtime mono-runtime-common mono-runtime-sgen
anual pkg-config pkgconf pkgconf-bin

odoc-gecko-manual monodoc-nunit-manual


n libgdiplus libmono-2.0-1 libmono-2.0-dev libmono-accessibility4.0-cil
```

Figure 4.8 LOIC windows and Wireshark

During the collection of attack traffic, each capture session starts off with normal traffic and then the attack traffic comes through after launching the attack. That way, a real-life scenario of having normal traffic flow before an attack is launched at the target gets mimicked. The attack source IP address is used to pinpoint the very moment the attack traffic starts coming through. Table 4.1 shows the attack traffic capture details. Each attack is launched several times and collected as a separate dataset file as indicated under file count column. Each data source is properly labelled according to the category it falls in. These categories include:

- Single attacks: Individually launched attacks. This comprises of one exploited protocol.
- Mixed attacks: A combination of the single attacks considered are launched at once and this traffic of mixed attacks is collected.
- Unfamiliar attacks: Attacks not exposed to detection system in the first batch of testing. This will be used to confirm if the system correctly detects and classifies new attacks in the next chapter.
- Public data: Both attack and normal data are publicly sourced to be used in the validation phase.

The data source being public or private is also indicated. The source and target IP addresses are also specified as well as the composition of the dataset whether it is purely normal or a mixture of attack and normal.

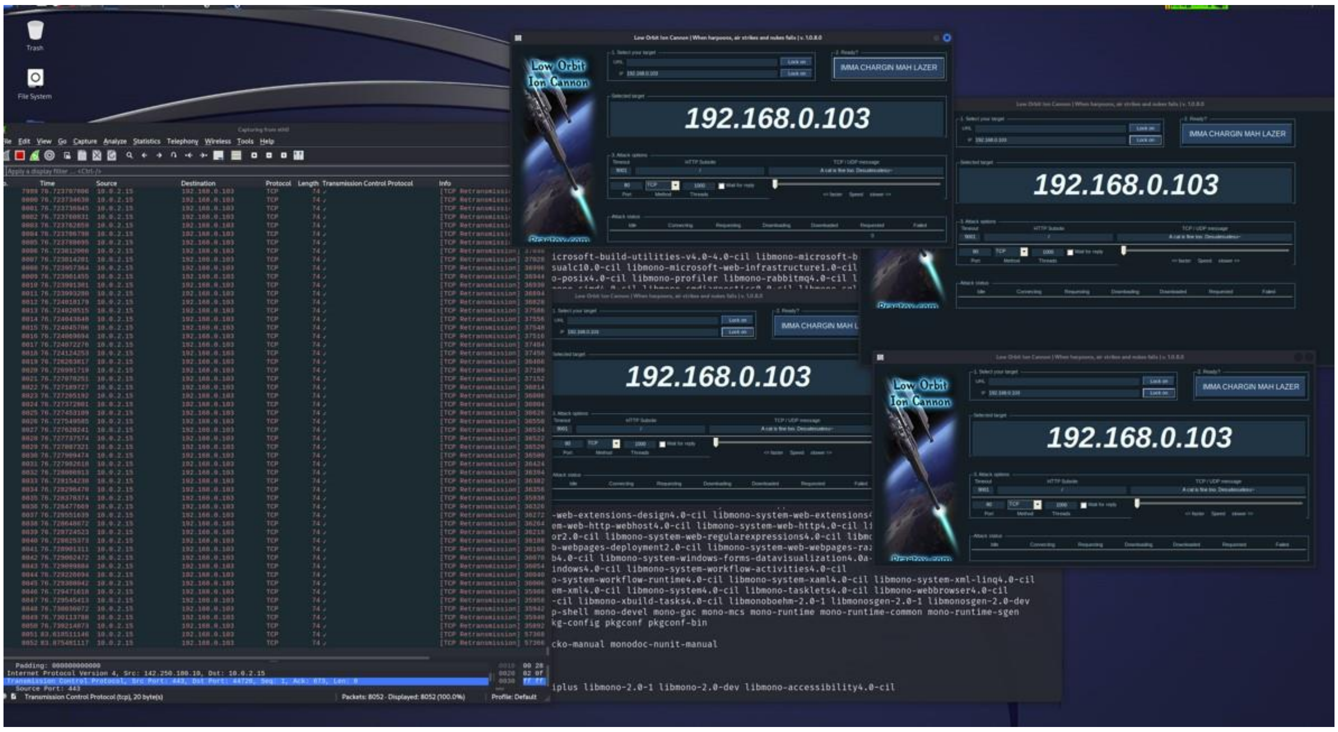
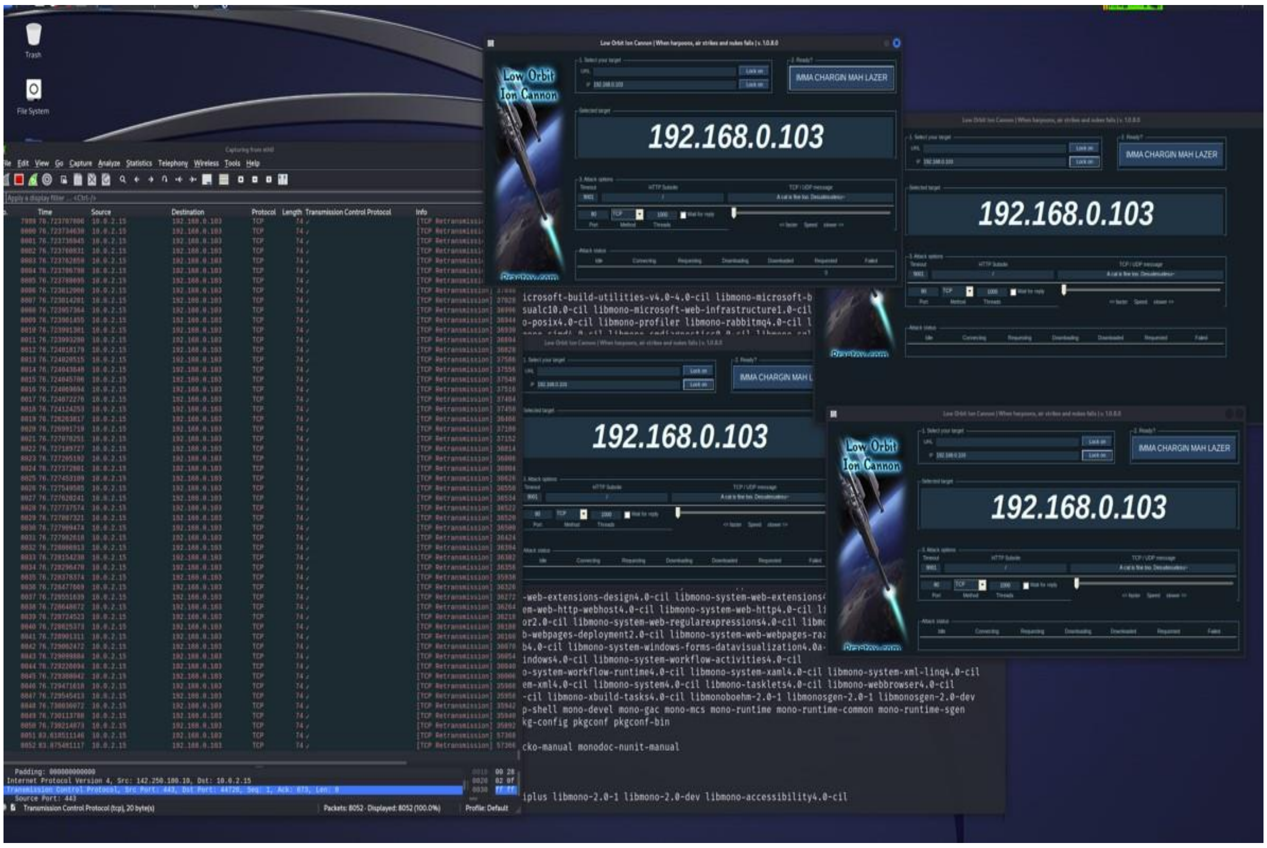


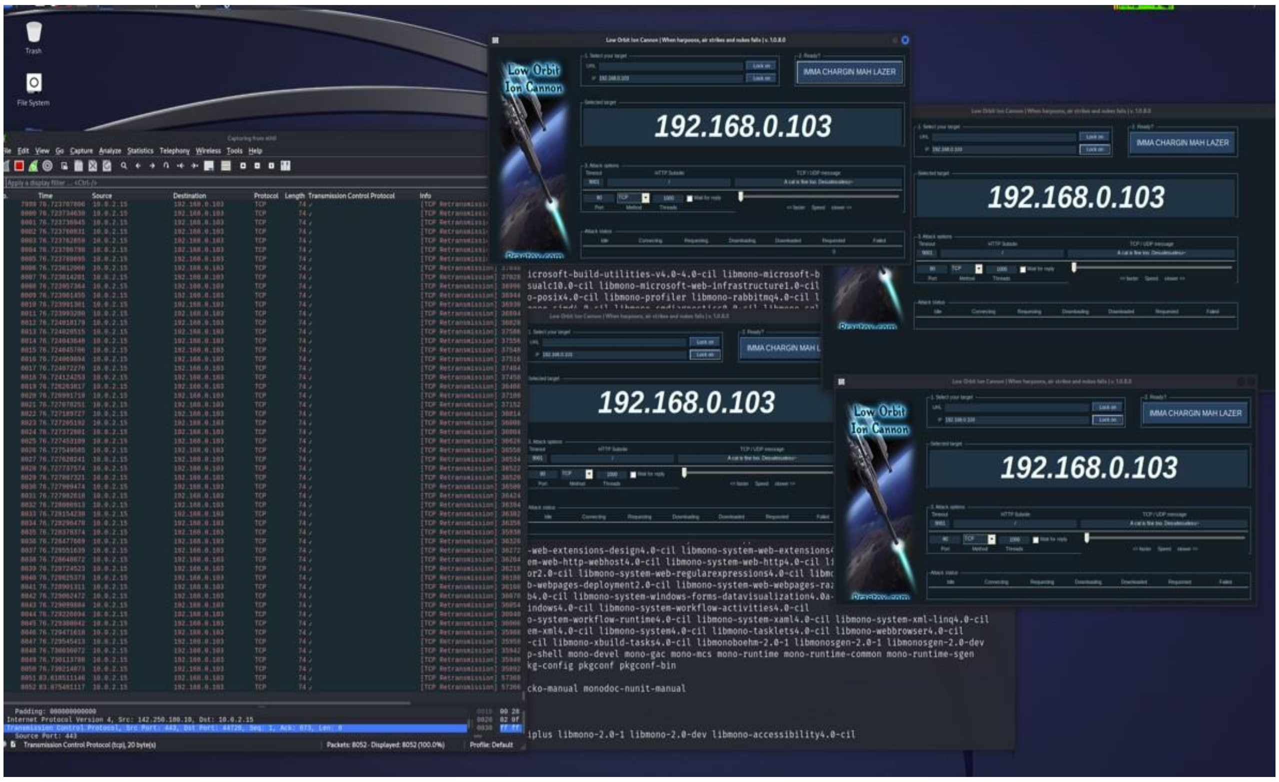
Figure 4.9 TCP flood using LOIC



```
icrosoft-build-utilities-v4.0-4.0-cil libmono-microsoft-b
sualc10.0-cil libmono-microsoft-web-infrastructure1.0-cil
o-posix4.0-cil libmono-profiler libmono-rabbitmq4.0-cil l

-web-extensions-design4.0-cil libmono-system-web-extensions
em-web-http-webhost4.0-cil libmono-system-web-http4.0-cil li
or2.0-cil libmono-system-web-regularexpressions4.0-cil libmo
b-webpages-deployment2.0-cil libmono-system-web-webpages-ra
b4.0-cil libmono-system-windows-forms-datavisualization4.0a-
indows4.0-cil libmono-system-workflow-activities4.0-cil
o-system-workflow-runtime4.0-cil libmono-system-xaml4.0-cil libmono-system-xml-linq4.0-cil
em-xml4.0-cil libmono-system4.0-cil libmono-tasklets4.0-cil libmono-webbrowser4.0-cil
-cil libmono-xbuild-tasks4.0-cil libmonoboehm-2.0-1 libmonosgen-2.0-1 libmonosgen-2.0-dev
p-shell mono-devel mono-gac mono-mcs mono-runtime mono-runtime-common mono-runtime-sgen
kg-config pkgconf pkgconf-bin

cko-manual monodoc-nunit-manual

iplus libmono-2.0-1 libmono-2.0-dev libmono-accessibility4.0-cil
```

Low Orbit Ion Cannon | When harpoons, air strikes and nukes fails | v. 1.0.8.0
IMMA CHARGIN MAH LAZER
192.168.0.103
Padding: 000000000000
Internet Protocol Version 4, Src: 142.250.180.10, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 44720, Seq: 1, Ack: 673, Len: 0
Source Port: 443
Packets: 8052 · Displayed: 8052 (100.0%)
Profile: Default

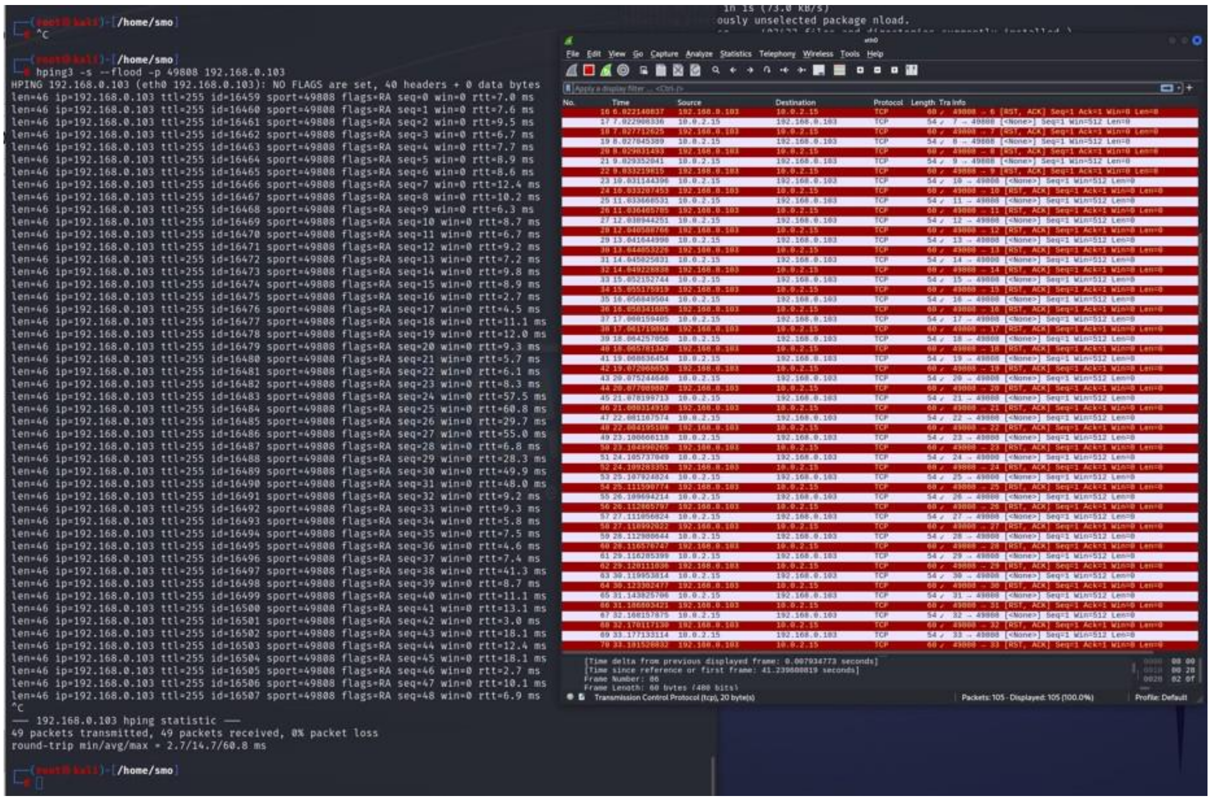
Figure 4.10 HTTP flood using LOIC



```
icrosoft-build-utilities-v4.0-4.0-cil libmono-microsoft-b
sualc10.0-cil libmono-microsoft-web-infrastructure1.0-cil
o-posix4.0-cil libmono-profiler libmono-rabbitmq4.0-cil l

-web-extensions-design4.0-cil libmono-system-web-extensions
em-web-http-webhost4.0-cil libmono-system-web-http4.0-cil li
or2.0-cil libmono-system-web-regularexpressions4.0-cil libmo
b-webpages-deployment2.0-cil libmono-system-web-webpages-ra
b4.0-cil libmono-system-windows-forms-datavisualization4.0a-
indows4.0-cil libmono-system-workflow-activities4.0-cil
o-system-workflow-runtime4.0-cil libmono-system-xaml4.0-cil libmono-system-xml-linq4.0-cil
em-xml4.0-cil libmono-system4.0-cil libmono-tasklets4.0-cil libmono-webbrowser4.0-cil
-cil libmono-xbuild-tasks4.0-cil libmonoboehm-2.0-1 libmonosgen-2.0-1 libmonosgen-2.0-dev
p-shell mono-devel mono-gac mono-mcs mono-runtime mono-runtime-common mono-runtime-sgen
kg-config pkgconf pkgconf-bin

cko-manual monodoc-nunit-manual

iplus libmono-2.0-1 libmono-2.0-dev libmono-accessibility4.0-cil
```

Low Orbit Ion Cannon | When harpoons, air strikes and nukes fails | v. 1.0.8.0
IMMA CHARGIN MAH LAZER
192.168.0.103
Padding: 000000000000
Internet Protocol Version 4, Src: 142.250.180.10, Dst: 10.0.2.15
Transmission Control Protocol, Src Port: 443, Dst Port: 44720, Seq: 1, Ack: 673, Len: 0
Source Port: 443
Packets: 8052 · Displayed: 8052 (100.0%)
Profile: Default

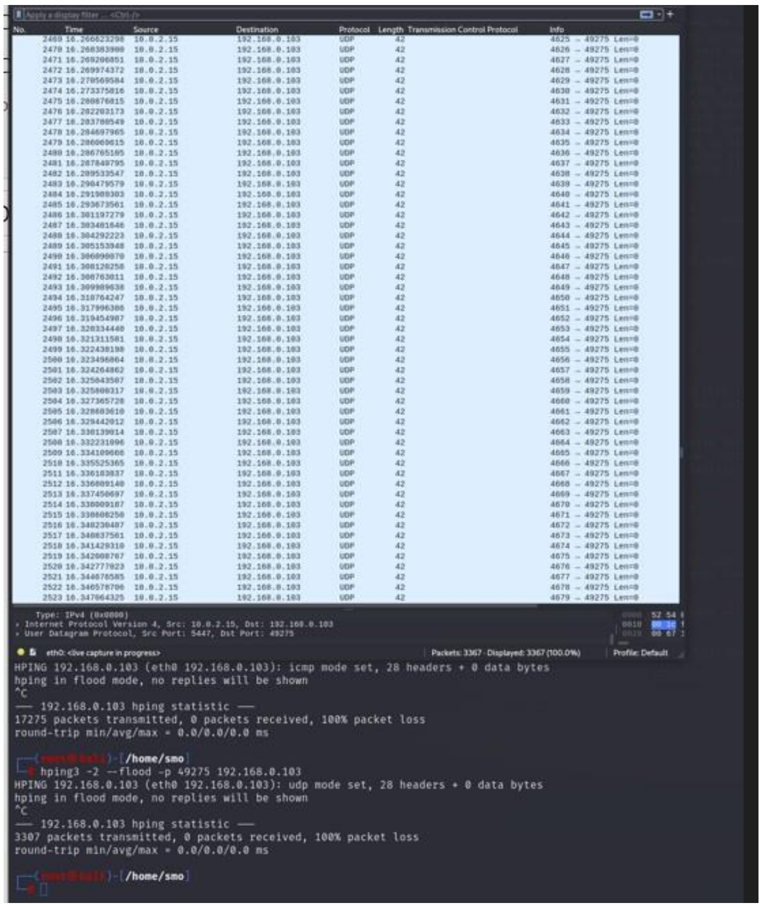
Figure 4.11 UDP flood using LOIC



```
┌──([redacted])-[/home/smo]
└─# ^C

┌──([redacted])-[/home/smo]
└─# hping3 -s --flood -p 49808 192.168.0.103
HPING 192.168.0.103 (eth0 192.168.0.103): NO FLAGS are set, 40 headers + 0 data bytes
len=46 ip=192.168.0.103 ttl=255 id=16459 sport=49808 flags=RA seq=0 win=0 rtt=7.0 ms
len=46 ip=192.168.0.103 ttl=255 id=16460 sport=49808 flags=RA seq=1 win=0 rtt=7.6 ms
len=46 ip=192.168.0.103 ttl=255 id=16461 sport=49808 flags=RA seq=2 win=0 rtt=9.5 ms
len=46 ip=192.168.0.103 ttl=255 id=16462 sport=49808 flags=RA seq=3 win=0 rtt=6.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16463 sport=49808 flags=RA seq=4 win=0 rtt=7.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16464 sport=49808 flags=RA seq=5 win=0 rtt=8.9 ms
len=46 ip=192.168.0.103 ttl=255 id=16465 sport=49808 flags=RA seq=6 win=0 rtt=8.6 ms
len=46 ip=192.168.0.103 ttl=255 id=16466 sport=49808 flags=RA seq=7 win=0 rtt=12.4 ms
len=46 ip=192.168.0.103 ttl=255 id=16467 sport=49808 flags=RA seq=8 win=0 rtt=10.2 ms
len=46 ip=192.168.0.103 ttl=255 id=16468 sport=49808 flags=RA seq=9 win=0 rtt=6.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16469 sport=49808 flags=RA seq=10 win=0 rtt=8.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16470 sport=49808 flags=RA seq=11 win=0 rtt=6.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16471 sport=49808 flags=RA seq=12 win=0 rtt=9.2 ms
len=46 ip=192.168.0.103 ttl=255 id=16472 sport=49808 flags=RA seq=13 win=0 rtt=7.2 ms
len=46 ip=192.168.0.103 ttl=255 id=16473 sport=49808 flags=RA seq=14 win=0 rtt=9.8 ms
len=46 ip=192.168.0.103 ttl=255 id=16474 sport=49808 flags=RA seq=15 win=0 rtt=8.9 ms
len=46 ip=192.168.0.103 ttl=255 id=16475 sport=49808 flags=RA seq=16 win=0 rtt=2.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16476 sport=49808 flags=RA seq=17 win=0 rtt=4.5 ms
len=46 ip=192.168.0.103 ttl=255 id=16477 sport=49808 flags=RA seq=18 win=0 rtt=11.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16478 sport=49808 flags=RA seq=19 win=0 rtt=12.0 ms
len=46 ip=192.168.0.103 ttl=255 id=16479 sport=49808 flags=RA seq=20 win=0 rtt=9.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16480 sport=49808 flags=RA seq=21 win=0 rtt=5.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16481 sport=49808 flags=RA seq=22 win=0 rtt=6.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16482 sport=49808 flags=RA seq=23 win=0 rtt=8.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16483 sport=49808 flags=RA seq=24 win=0 rtt=57.5 ms
len=46 ip=192.168.0.103 ttl=255 id=16484 sport=49808 flags=RA seq=25 win=0 rtt=60.8 ms
len=46 ip=192.168.0.103 ttl=255 id=16485 sport=49808 flags=RA seq=26 win=0 rtt=29.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16486 sport=49808 flags=RA seq=27 win=0 rtt=55.0 ms
len=46 ip=192.168.0.103 ttl=255 id=16487 sport=49808 flags=RA seq=28 win=0 rtt=6.8 ms
len=46 ip=192.168.0.103 ttl=255 id=16488 sport=49808 flags=RA seq=29 win=0 rtt=28.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16489 sport=49808 flags=RA seq=30 win=0 rtt=49.9 ms
len=46 ip=192.168.0.103 ttl=255 id=16490 sport=49808 flags=RA seq=31 win=0 rtt=48.0 ms
len=46 ip=192.168.0.103 ttl=255 id=16491 sport=49808 flags=RA seq=32 win=0 rtt=9.2 ms
len=46 ip=192.168.0.103 ttl=255 id=16492 sport=49808 flags=RA seq=33 win=0 rtt=9.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16493 sport=49808 flags=RA seq=34 win=0 rtt=5.8 ms
len=46 ip=192.168.0.103 ttl=255 id=16494 sport=49808 flags=RA seq=35 win=0 rtt=7.5 ms
len=46 ip=192.168.0.103 ttl=255 id=16495 sport=49808 flags=RA seq=36 win=0 rtt=4.6 ms
len=46 ip=192.168.0.103 ttl=255 id=16496 sport=49808 flags=RA seq=37 win=0 rtt=7.4 ms
len=46 ip=192.168.0.103 ttl=255 id=16497 sport=49808 flags=RA seq=38 win=0 rtt=41.3 ms
len=46 ip=192.168.0.103 ttl=255 id=16498 sport=49808 flags=RA seq=39 win=0 rtt=8.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16499 sport=49808 flags=RA seq=40 win=0 rtt=11.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16500 sport=49808 flags=RA seq=41 win=0 rtt=13.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16501 sport=49808 flags=RA seq=42 win=0 rtt=3.0 ms
len=46 ip=192.168.0.103 ttl=255 id=16502 sport=49808 flags=RA seq=43 win=0 rtt=18.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16503 sport=49808 flags=RA seq=44 win=0 rtt=12.4 ms
len=46 ip=192.168.0.103 ttl=255 id=16504 sport=49808 flags=RA seq=45 win=0 rtt=18.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16505 sport=49808 flags=RA seq=46 win=0 rtt=2.7 ms
len=46 ip=192.168.0.103 ttl=255 id=16506 sport=49808 flags=RA seq=47 win=0 rtt=10.1 ms
len=46 ip=192.168.0.103 ttl=255 id=16507 sport=49808 flags=RA seq=48 win=0 rtt=6.9 ms
^C
--- 192.168.0.103 hping statistic ---
49 packets transmitted, 49 packets received, 0% packet loss
round-trip min/avg/max = 2.7/14.7/60.8 ms

┌──([redacted])-[/home/smo]
└─# 
```

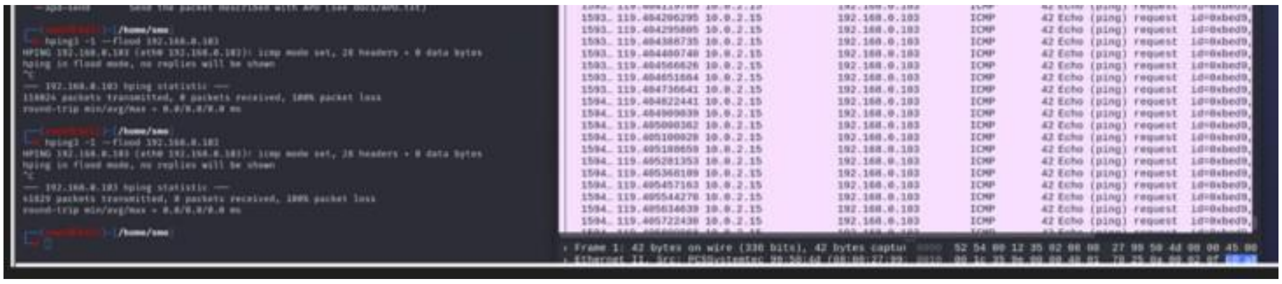
```
[Time delta from previous displayed frame: 0.007934773 seconds]
[Time since reference or first frame: 41.239608819 seconds]
Frame Number: 86
Frame Length: 60 bytes (480 bits)
Transmission Control Protocol (tcp), 20 byte(s)
Packets: 105 · Displayed: 105 (100.0%)    Profile: Default
```

Figure 4.12 TCP flood using Hping3



```
2513 16.337450697  10.0.2.15   192.168.0.103   UDP   42   4669 → 49275 Len=0
2514 16.338009187  10.0.2.15   192.168.0.103   UDP   42   4670 → 49275 Len=0
2515 16.338608250  10.0.2.15   192.168.0.103   UDP   42   4671 → 49275 Len=0
2516 16.340230487  10.0.2.15   192.168.0.103   UDP   42   4672 → 49275 Len=0
2517 16.340837561  10.0.2.15   192.168.0.103   UDP   42   4673 → 49275 Len=0
2518 16.341429310  10.0.2.15   192.168.0.103   UDP   42   4674 → 49275 Len=0
2519 16.342008707  10.0.2.15   192.168.0.103   UDP   42   4675 → 49275 Len=0
2520 16.342777023  10.0.2.15   192.168.0.103   UDP   42   4676 → 49275 Len=0
2521 16.344078585  10.0.2.15   192.168.0.103   UDP   42   4677 → 49275 Len=0
2522 16.346578706  10.0.2.15   192.168.0.103   UDP   42   4678 → 49275 Len=0
2523 16.347064325  10.0.2.15   192.168.0.103   UDP   42   4679 → 49275 Len=0
    Type: IPv4 (0x0800)
› Internet Protocol Version 4, Src: 10.0.2.15, Dst: 192.168.0.103
› User Datagram Protocol, Src Port: 5447, Dst Port: 49275
eth0: <live capture in progress>    Packets: 3367 · Displayed: 3367 (100.0%)    Profile: Default
HPING 192.168.0.103 (eth0 192.168.0.103): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.103 hping statistic ---
17275 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([redacted])-[/home/smo]
└─# hping3 -2 --flood -p 49275 192.168.0.103
HPING 192.168.0.103 (eth0 192.168.0.103): udp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.103 hping statistic ---
3307 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([redacted])-[/home/smo]
└─# 
```

Figure 4.13 UDP flood using Hping3



```
┌──([illegible])-[/home/sms]
└─$ hping3 -1 --flood 192.168.0.103
HPING 192.168.0.103 (eth0 192.168.0.103): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.103 hping statistic ---
118824 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([illegible])-[/home/sms]
└─$ hping3 -1 --flood 192.168.0.103
HPING 192.168.0.103 (eth0 192.168.0.103): icmp mode set, 28 headers + 0 data bytes
hping in flood mode, no replies will be shown
^C
--- 192.168.0.103 hping statistic ---
61829 packets transmitted, 0 packets received, 100% packet loss
round-trip min/avg/max = 0.0/0.0/0.0 ms

┌──([illegible])-[/home/sms]
└─$
```

```
1593… 119.404206295 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404295805 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404388735 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404480740 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404566626 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404651664 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1593… 119.404736641 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.404822441 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.404909039 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405000362 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405100028 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405188659 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405281353 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405368109 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405457163 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405544270 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405634039 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
1594… 119.405722430 10.0.2.15    192.168.0.103   ICMP   42 Echo (ping) request  id=0xbed9,
› Frame 1: 42 bytes on wire (336 bits), 42 bytes captu   0000  52 54 00 12 35 02 08 00  27 98 50 4d 00 00 45 00
› Ethernet II, Src: PCSSystemtec [illegible]
```

Figure 4.14 ICMP using Hping3

Table 4.1 Data collection

ID	Label (Protocol)	Source IP	Target IP	Public/Private	Known/Unfamiliar	Count	Composition
4	HTTP	192.168.0.103	192.168.0.102	Private	Known	3	Attack+benign
5	SLOWLOIC	192.168.0.103	192.168.0.102	Private	Unfamiliar	2	Attack+benign
6	RECOIL	192.168.0.103	192.168.0.102	Private	Unfamiliar	2	Attack+benign
7	Mixed	192.168.0.103	192.168.0.102	Private	Unfamiliar	3	Attack+benign
8	Normal		192.168.0.101	Private		34	Benign
	TCPSYN	192.168.0.101	192.168.0.102 192.168.0.103	Private	familiar	8	Attack+benign
	UDP	192.168.0.101	192.168.0.102 192.168.0.103	Private	Familiar	8	Attack+benign
	ICMP		192.168.0.101	Private	Familiar	5	Attack+benign
9	TCPSYN	192.168.100.147-150	192.168.100.3	Public	Known	12	Attack+benign
10	UDP	192.168.100.147-150	192.168.100.3	Public	Known	10	Attack+benign
11	HTTP	192.168.100.147-150	192.168.100.3	Public	Known	3	Attack+benign
12	FRAGMENTED	192.168.1.195	74.91.117.248	Public	Unfamiliar	3	Attack+benign
13	Mixed	192.168.100.147-150	192.168.100.3	Public	Unfamiliar	3	Attack+benign
	ICMP			Public	Familiar	1	Attack
14	Normal		192.168.1.158 192.168.1.132	Public	Unfamiliar	5	Benign

The benign data used in this research is from the previous chapter which addresses EDA on benign smart home traffic. This consists of the same network devices and topology as mentioned in the attack data collection above just excluding the DDoS attacking points. The IoT-23 dataset [84] is the public dataset used for validation of the EDA on benign smart home traffic. This dataset consists of IoT network traffic from real devices. It has 20 malware captures executed in IoT devices, and 3 captures for benign IoT devices traffic. These traffic flows were captured in the Stratosphere Laboratory, AIC group, FEL, CTU University, Czech Republic. The benign traffic flows were extracted from this dataset and used in this research. The BoT-IoT dataset [85] is one of the public attack datasets used for the

EDA validation. It was created at Cyber Range Lab of UNSW Canberra by designing a realistic network environment. The network environment incorporated a combination of normal and botnet traffic. The dataset includes DDoS, DoS, OS and service scan, keylogging and data exfiltration attacks, with the DDoS and DoS attacks further organized, based on the protocol used. TCPSYN and UDP DDoS attack flows were extracted from the dataset and used in this research. Lastly, public ICMP DDoS attack flow was extracted from the BUET-DDoS2020 dataset [125]. It consists of several DDoS flooding attack flows including TCPSYN, DNS, HTTP and UDP.

4.4 Exploratory Data Analysis

Exploratory Data Analysis is a statistical method of analysing data to summarize the main characteristics of the dataset by using data visualization tools and techniques to represent the derived results for ease of understanding. Google Colab [120] has been used for this purpose in this research. As outlined in the methodology, the EDA process has several sub processes of data pre-processing, feature selection, data normalization, label encoding, feature correlation, method representation and lastly visualization. After collecting the data, it was filtered to have only the relevant flows relating to the smart home devices. The data had several columns carrying different packet details. Certain network features were found to be greatly affected during a DDoS flood. These features were filtered out and focused on for further analysis. They are as follows:

- Protocol: These are the various communication entities necessary for different types of data transmission between devices. They include TCP, UDP, NTP, ICMP, DNS, MDNS, TLS, SSH. Each transmitted data packet is associated with a protocol.
- TCP Sequence number: This is a counter-based mechanism attributed to TCP packets to keep track of transmitted and received packets. Packets in the same flow carry an incremental value of sequence numbers, thus you can identify what packet comes after which by computing these numbers.
- Packet length: This is the payload size carried by each packet
- TCP flags: These are the labels carried by each TCP packet to indicate the state of connection. These are also present during the 3-way TCP handshake. For a complete TCP handshake to take place, these flags must be present in their sequential manner depending on the type of communication.
- Encryption: A secure communication channel is bound by encryption protocols. These include TLS and SSH.

As each packet is attached to some or all the above listed network properties, they were observed to be simultaneously affected during an attack, thus the reason they were correlated as a baseline for attack detection.

4.4.1 Benign traffic

The benign smart home traffic was analysed based on the listed features. One minute traffic was filtered out and visualized to show how these properties behave. The ID of each frame or packet which corresponds to the time series column in Wireshark is plotted against its corresponding protocol. This shows the dynamic pattern of the visualised packet headers in the smart home traffic.

- Protocols: The protocols in a benign flow are dynamic. They tend to change from one packet to the next or every few packets. These varying protocols can include DNS, TCP, MDNS, TLS, NTP, ICMP, and the like. This is shown in figure 4.15 and 4.16 for both the private and public benign traffic flows respectively. Both figures show the alternating pattern of the protocols having various combinations. Encryption protocols (TLS.v1.3) also tend to be predominant in a flow as most exchanged packets are securely encrypted. This results in encryption protocols being one of the frequent ones among the pool of protocols utilized by the devices. Message Queuing Telemetry Protocol (MQTT) is the application layer protocol used by the private dataset which are Hive devices. The significance of these figures is to show that smart home traffic comprises of a variation of several protocols in a short period of time. This will be compared with the absence of this variation in the DDoS attacks covered in this research in the next section.

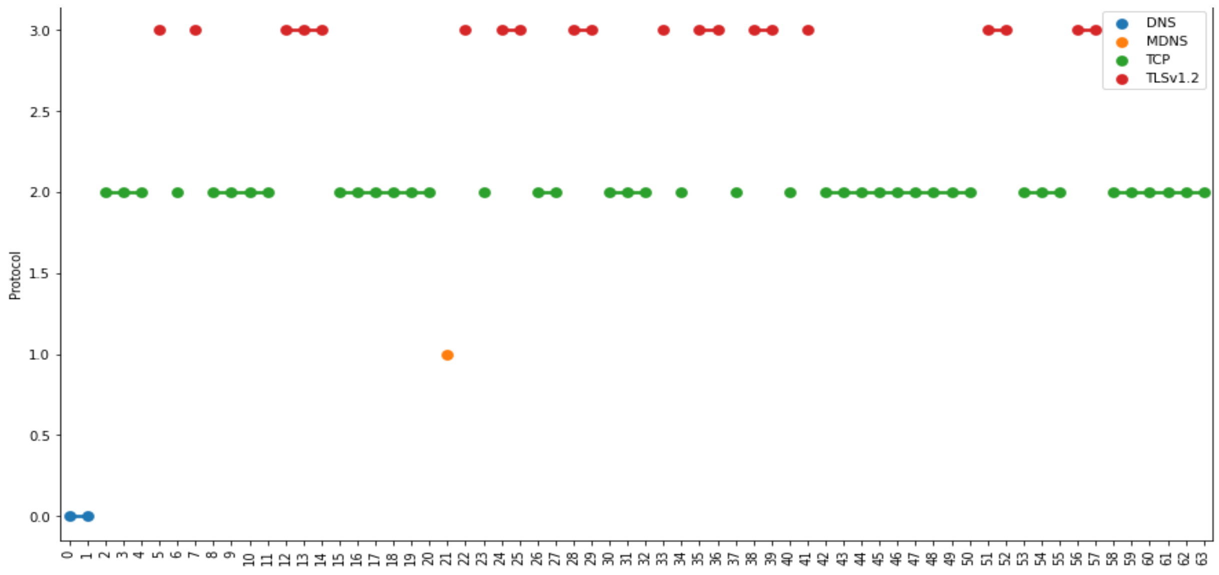


Figure 4.15 Varying protocol (private data)

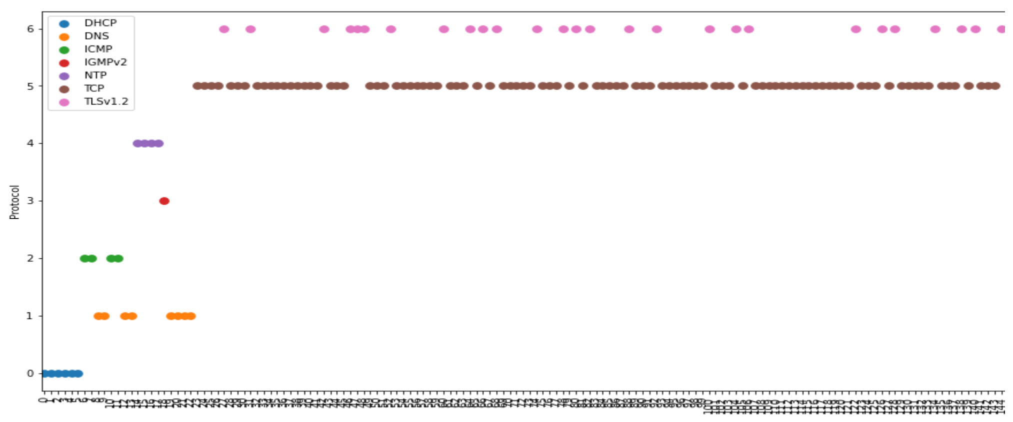


Figure 4.16 Varying protocol (public data)

- Packet length: The packet length of the benign flow had the same varying property as the protocol. Figures 4.17 and 4.18 show the private and public packet length dynamic nature. We can see that the lengths fall in different ranges from one packet to the next.

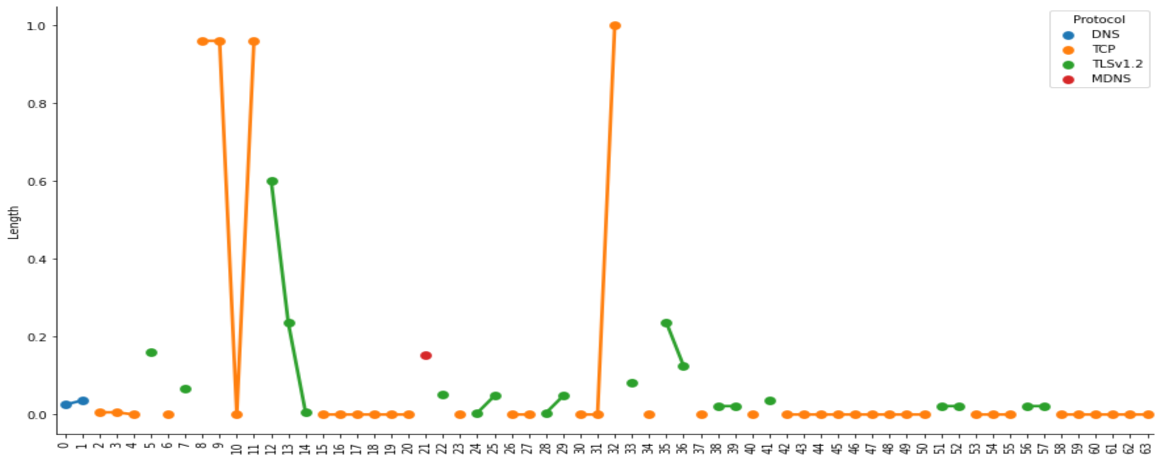


Figure 4.17 Varying packet length (private data)

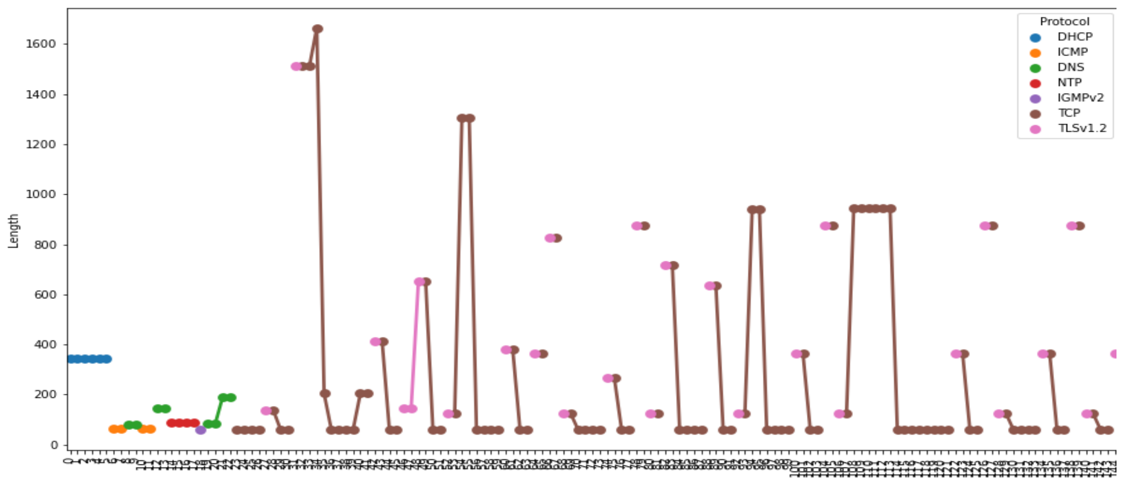


Figure 4.18 Varying packet length (public data)

- TCP Sequence numbers: TCP Sequence numbers also had the same varying nature. This is shown in figures 4.19 and 4.20. However, not all protocols carry TCP sequence numbers like

DNS, NTP, ICMP and the like. Nonetheless TCP sequence numbers were found to be persistent as the scope of this research is limited to TCP/ HTTP based traffic. This makes the appearance of TCP sequence numbers frequent as the predominant traffic protocols are of this type. TCP sequence numbers were observed to be incremental from one packet to the next in a flow. This starts from a 0, 1 and shoots up to very high values as shown in figures 4.19 and 4.20. This shows that apart from being dynamic in nature, the TCP sequence numbers have a wide range that they fall within. The relevance of this TCP sequence number visualisation is to compare the dominance of these TCP sequence numbers to exploited attack protocols that do not carry this feature like UDP and ICMP attacks among others. This will aid in attack detection in TCP based traffic which the scope of this research is limited to.

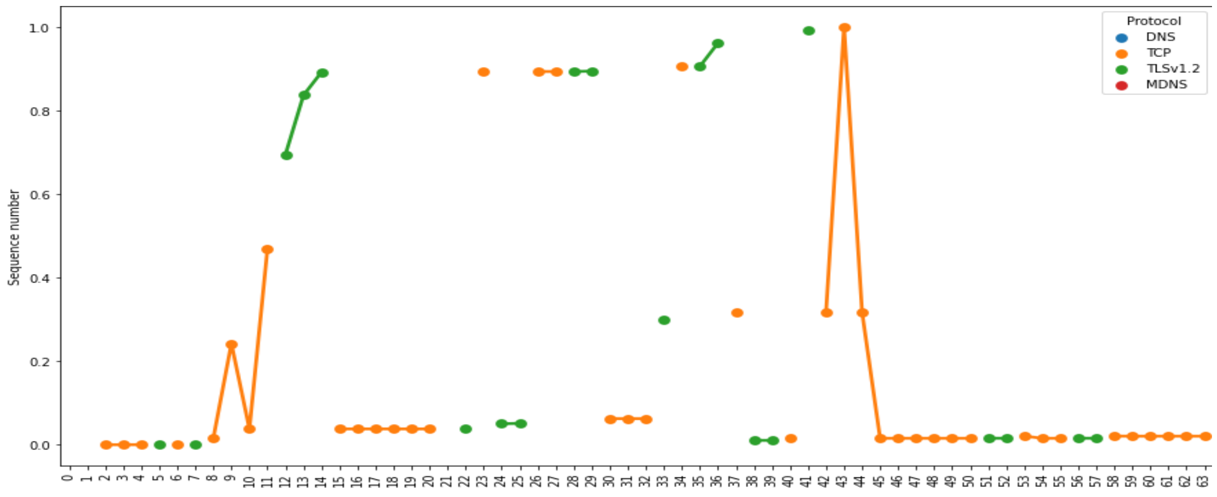


Figure 4.19 Varying sequence number (private data)

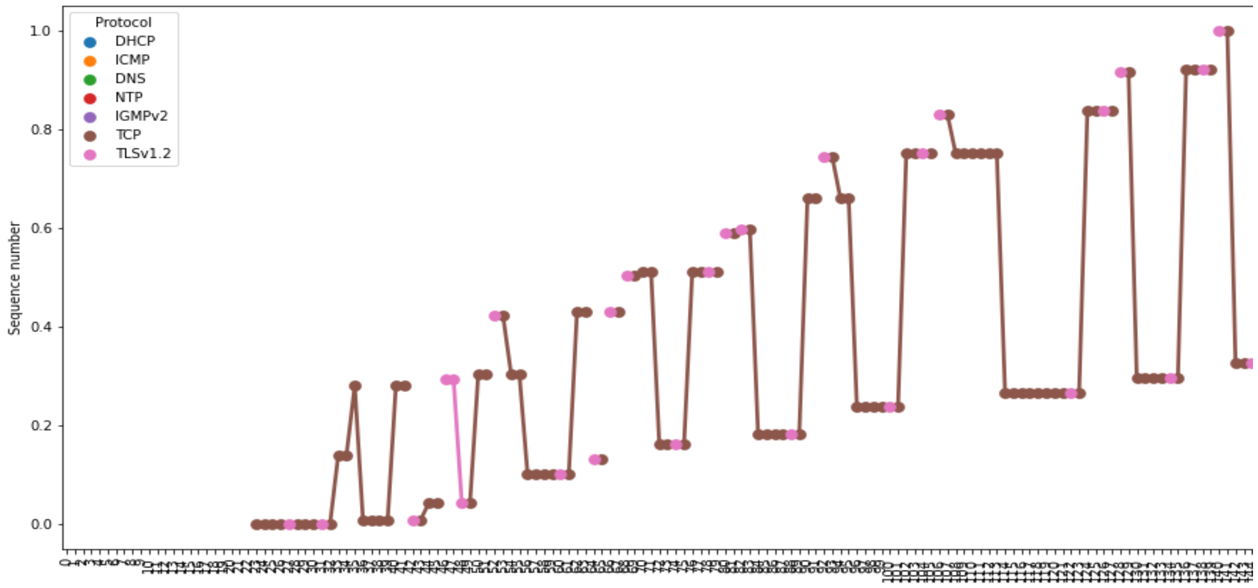


Figure 4.20 Varying sequence number (public data)

- TCP flags: Due to the 3-way handshake that takes place for TCP flows, varying TCP flags were observed to be present in a flow. Figures 4.21 and 4.22 show the various proportions of these flags contained in a one-minute window for both private and public datasets respectively. This observation also validates TCP flags having several combinations in a flow, thus also having a dynamic pattern like the previous features. The TCP flags from both public and private dataset are extracted from a one-minute window of the traffic. The public dataset tends to have more generated packets in a one-minute window compared to the private one as seen across all visualizations in this section. The significance of this is to show how TCP flags vary in a traffic flow compared to some attack traffic that lack this variation which can be used for attack detection in TCP based traffic.

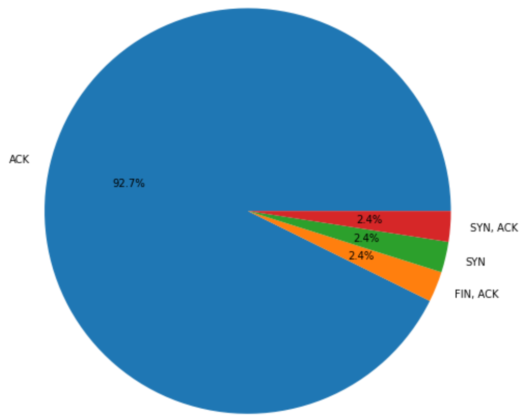


Figure 4.21 Varying TCP flags (Private)

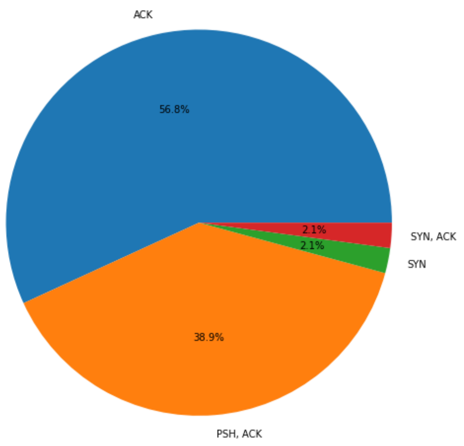
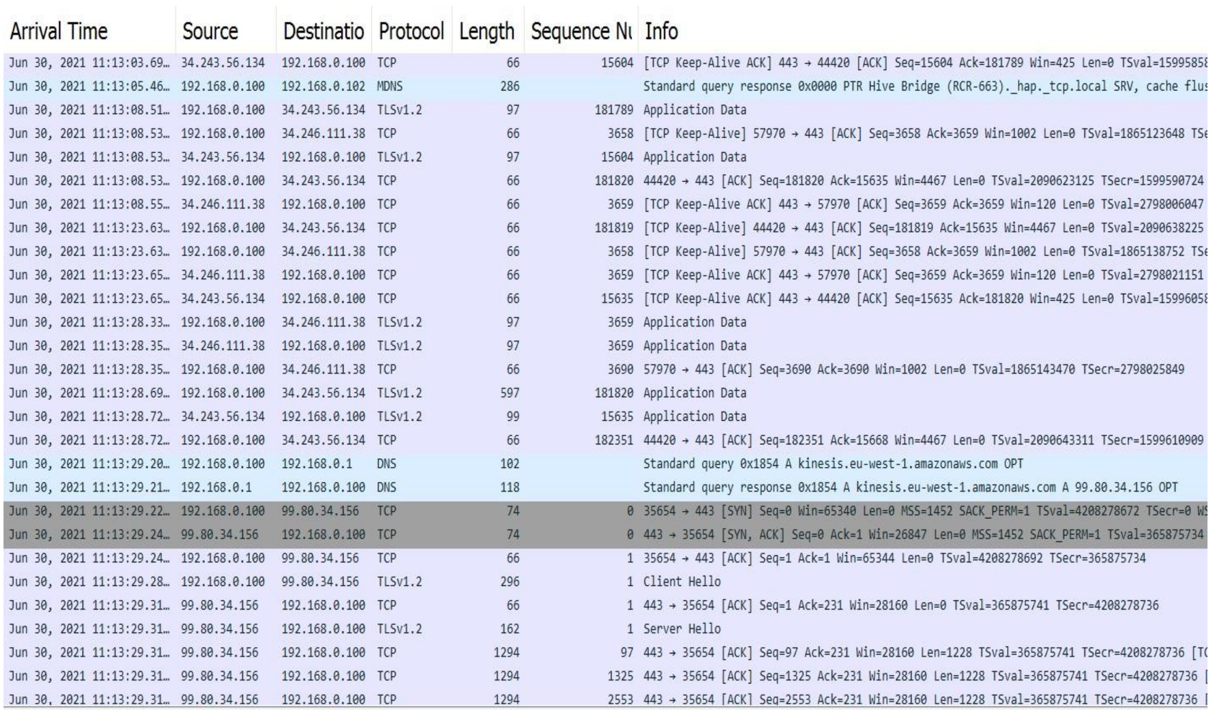


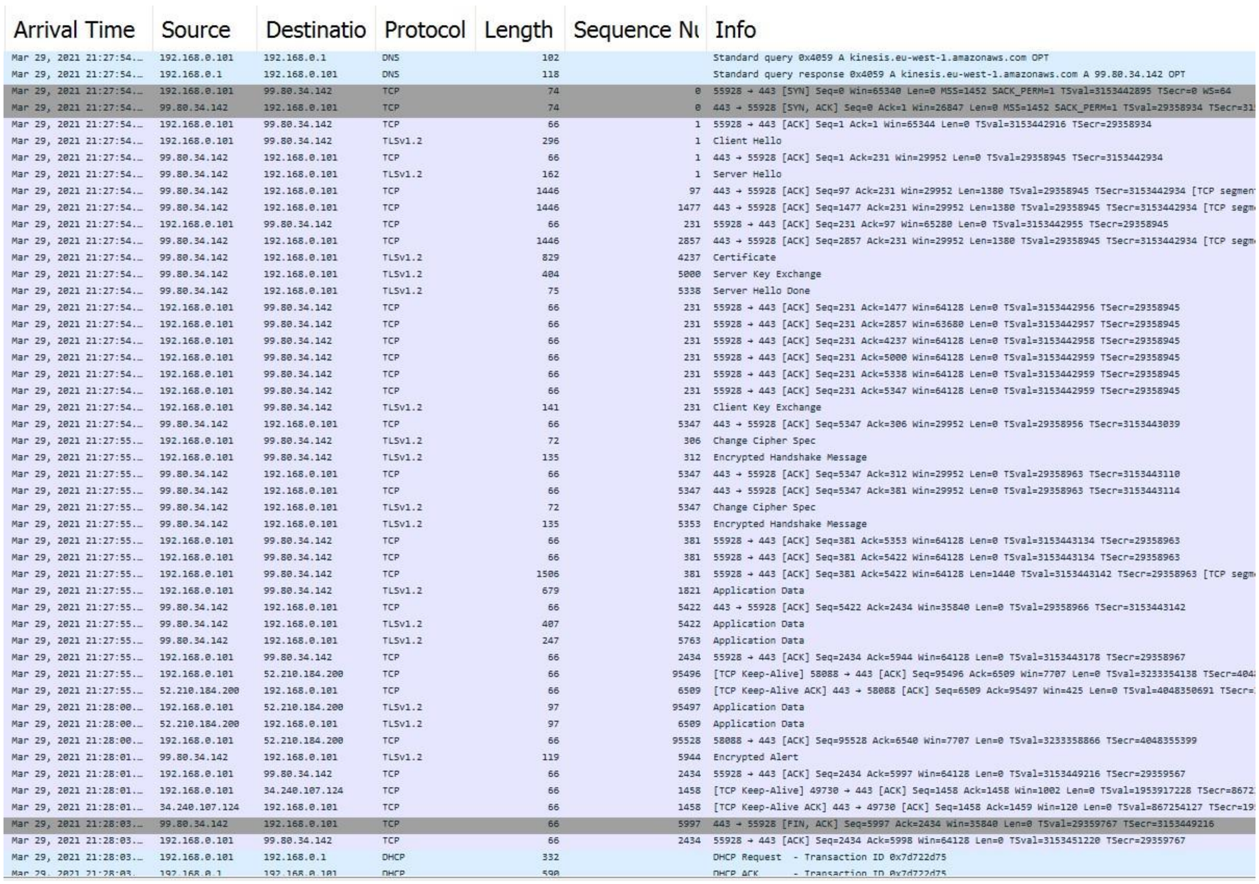
Figure 4.22 Varying TCP flags (public)

Figures 4.23 to 4.32 show the various packet header patterns visualised at packet level. Figure 4.23 to 4.26 are from the private dataset while figure 4.27 to 4.32 are captures from the public dataset. In all the figures under the protocol column we can see the dynamic nature of these protocols over time as explained in the EDA. The packet length columns exhibit the same pattern of having dynamic lengths. From the TCP sequence number columns (Sequence number) we can see these incremental numbers. As mentioned earlier they start with 0,1 at the beginning of a flow which can be seen to correspond with the first SYN packet under the info column. These can be seen in figures 4.23, 4.24, 4.25, 4.26, 4.30 and 4.31 right after the DNS query and response. The next packet after this query tends to have a SYN flag and a sequence number of 0 which keeps incrementing as the flow generates. We can also see the various TCP flags present in this traffic under the info column including SYN, ACK, PSH and FIN as also visualised in figures 4.23, 4.24 and 4.25.



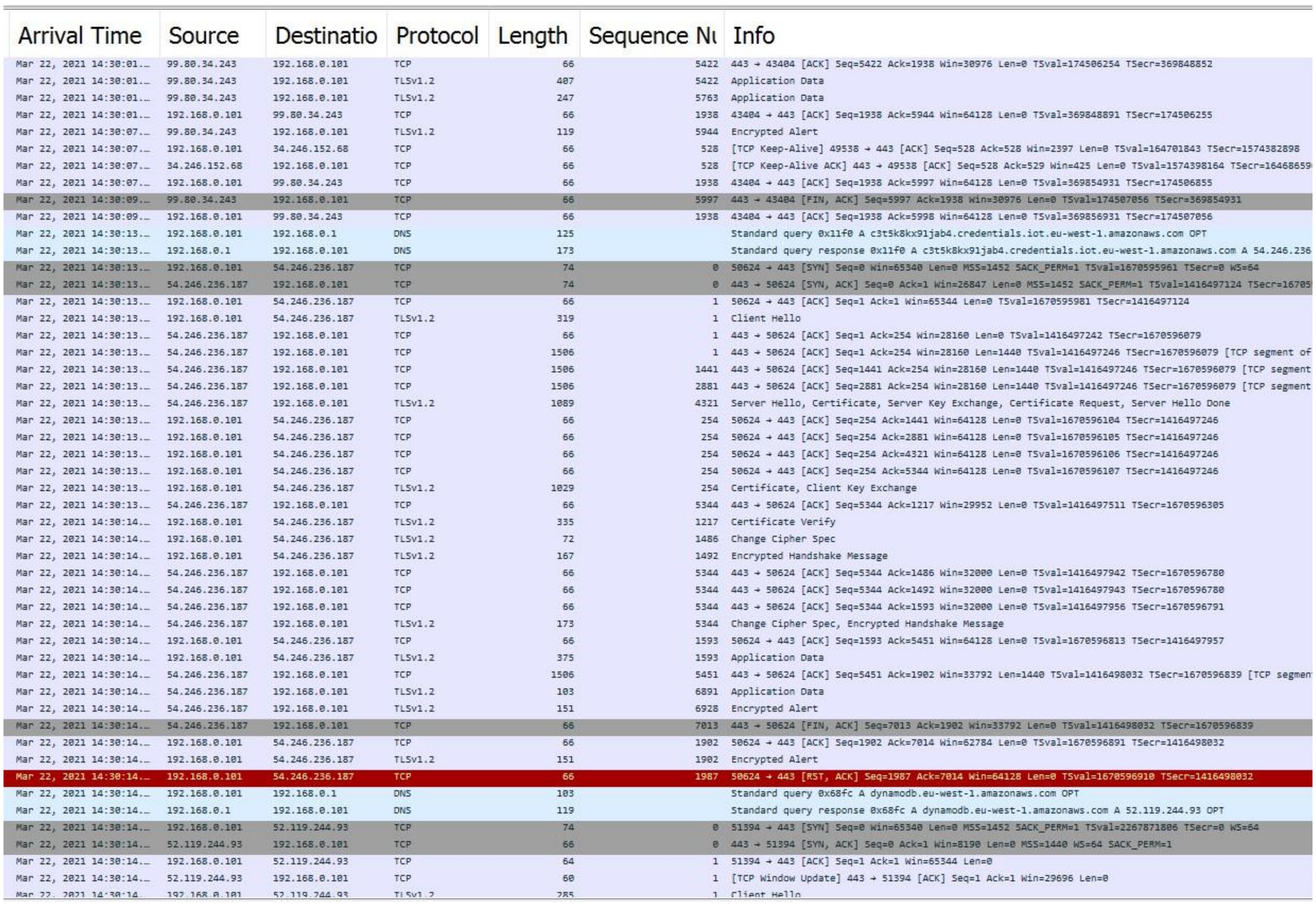
Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nu	Info
Jun 30, 2021 11:13:03.69…	34.243.56.134	192.168.0.100	TCP	66	15604	[TCP Keep-Alive ACK] 443 → 44420 [ACK] Seq=15604 Ack=181789 Win=425 Len=0 TSval=1599585
Jun 30, 2021 11:13:05.46…	192.168.0.100	192.168.0.102	MDNS	286		Standard query response 0x0000 PTR Hive Bridge (RCR-663)._hap._tcp.local SRV, cache flus
Jun 30, 2021 11:13:08.51…	192.168.0.100	34.243.56.134	TLSv1.2	97	181789	Application Data
Jun 30, 2021 11:13:08.53…	192.168.0.100	34.246.111.38	TCP	66	3658	[TCP Keep-Alive] 57970 → 443 [ACK] Seq=3658 Ack=3659 Win=1002 Len=0 TSval=1865123648 TSe
Jun 30, 2021 11:13:08.53…	34.243.56.134	192.168.0.100	TLSv1.2	97	15604	Application Data
Jun 30, 2021 11:13:08.53…	192.168.0.100	34.243.56.134	TCP	66	181820	44420 → 443 [ACK] Seq=181820 Ack=15635 Win=4467 Len=0 TSval=2090623125 TSecr=1599590724
Jun 30, 2021 11:13:08.55…	34.246.111.38	192.168.0.100	TCP	66	3659	[TCP Keep-Alive ACK] 443 → 57970 [ACK] Seq=3659 Ack=3659 Win=120 Len=0 TSval=2798006047
Jun 30, 2021 11:13:23.63…	192.168.0.100	34.243.56.134	TCP	66	181819	[TCP Keep-Alive] 44420 → 443 [ACK] Seq=181819 Ack=15635 Win=4467 Len=0 TSval=2090638225
Jun 30, 2021 11:13:23.63…	192.168.0.100	34.246.111.38	TCP	66	3658	[TCP Keep-Alive] 57970 → 443 [ACK] Seq=3658 Ack=3659 Win=1002 Len=0 TSval=1865138752 TSe
Jun 30, 2021 11:13:23.65…	34.246.111.38	192.168.0.100	TCP	66	3659	[TCP Keep-Alive ACK] 443 → 57970 [ACK] Seq=3659 Ack=3659 Win=120 Len=0 TSval=2798021151
Jun 30, 2021 11:13:23.65…	34.243.56.134	192.168.0.100	TCP	66	15635	[TCP Keep-Alive ACK] 443 → 44420 [ACK] Seq=15635 Ack=181820 Win=425 Len=0 TSval=1599605
Jun 30, 2021 11:13:28.33…	192.168.0.100	34.246.111.38	TLSv1.2	97	3659	Application Data
Jun 30, 2021 11:13:28.35…	34.246.111.38	192.168.0.100	TLSv1.2	97	3659	Application Data
Jun 30, 2021 11:13:28.35…	192.168.0.100	34.246.111.38	TCP	66	3690	57970 → 443 [ACK] Seq=3690 Ack=3690 Win=1002 Len=0 TSval=1865143470 TSecr=2798025849
Jun 30, 2021 11:13:28.69…	192.168.0.100	34.243.56.134	TLSv1.2	597	181820	Application Data
Jun 30, 2021 11:13:28.72…	34.243.56.134	192.168.0.100	TLSv1.2	99	15635	Application Data
Jun 30, 2021 11:13:28.72…	192.168.0.100	34.243.56.134	TCP	66	182351	44420 → 443 [ACK] Seq=182351 Ack=15668 Win=4467 Len=0 TSval=2090643311 TSecr=1599610909
Jun 30, 2021 11:13:29.20…	192.168.0.100	192.168.0.1	DNS	102		Standard query 0x1854 A kinesis.eu-west-1.amazonaws.com OPT
Jun 30, 2021 11:13:29.21…	192.168.0.1	192.168.0.100	DNS	118		Standard query response 0x1854 A kinesis.eu-west-1.amazonaws.com A 99.80.34.156 OPT
Jun 30, 2021 11:13:29.22…	192.168.0.100	99.80.34.156	TCP	74	0	35654 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=4208278672 TSecr=0 WS
Jun 30, 2021 11:13:29.24…	99.80.34.156	192.168.0.100	TCP	74	0	443 → 35654 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSval=365875734
Jun 30, 2021 11:13:29.24…	192.168.0.100	99.80.34.156	TCP	66	1	35654 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=4208278692 TSecr=365875734
Jun 30, 2021 11:13:29.28…	192.168.0.100	99.80.34.156	TLSv1.2	296	1	Client Hello
Jun 30, 2021 11:13:29.31…	99.80.34.156	192.168.0.100	TCP	66	1	443 → 35654 [ACK] Seq=1 Ack=231 Win=28160 Len=0 TSval=365875741 TSecr=4208278736
Jun 30, 2021 11:13:29.31…	99.80.34.156	192.168.0.100	TLSv1.2	162	1	Server Hello
Jun 30, 2021 11:13:29.31…	99.80.34.156	192.168.0.100	TCP	1294	97	443 → 35654 [ACK] Seq=97 Ack=231 Win=28160 Len=1228 TSval=365875741 TSecr=4208278736 [TC
Jun 30, 2021 11:13:29.31…	99.80.34.156	192.168.0.100	TCP	1294	1325	443 → 35654 [ACK] Seq=1325 Ack=231 Win=28160 Len=1228 TSval=365875741 TSecr=4208278736 [
Jun 30, 2021 11:13:29.31…	99.80.34.156	192.168.0.100	TCP	1294	2553	443 → 35654 [ACK] Seq=2553 Ack=231 Win=28160 Len=1228 TSval=365875741 TSecr=4208278736 [

Figure 4.23 Private dataset benign pattern



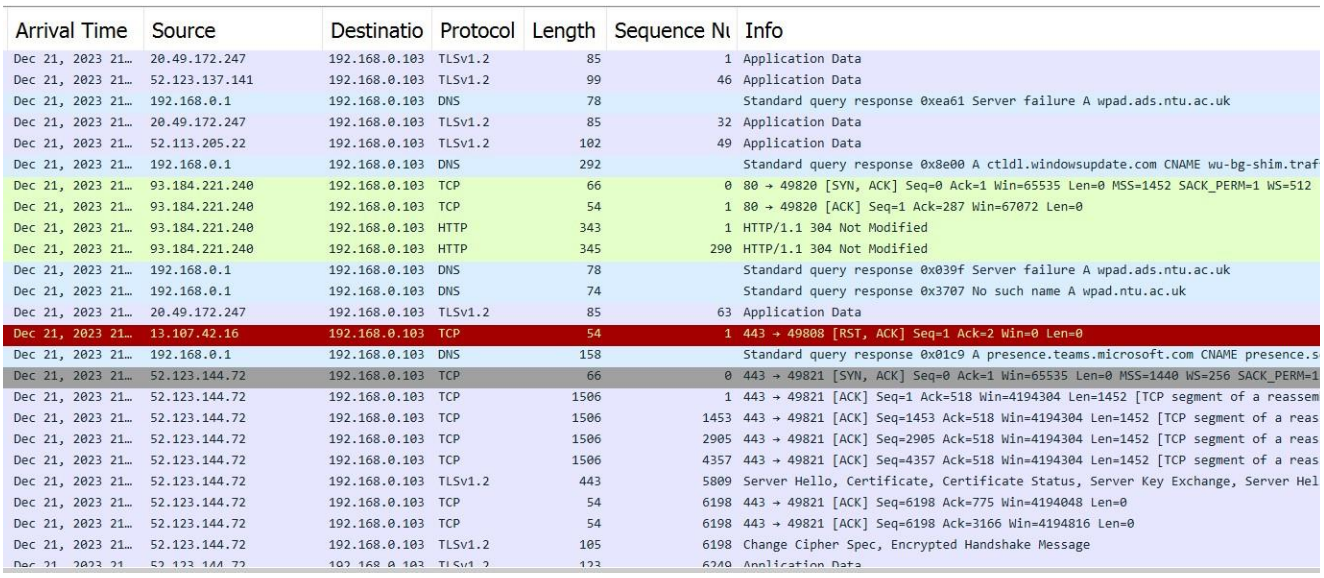
Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nu	Info
Mar 29, 2021 21:27:54…	192.168.0.101	192.168.0.1	DNS	102		Standard query 0x4059 A kinesis.eu-west-1.amazonaws.com OPT
Mar 29, 2021 21:27:54…	192.168.0.1	192.168.0.101	DNS	118		Standard query response 0x4059 A kinesis.eu-west-1.amazonaws.com A 99.80.34.142 OPT
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	74	0	55928 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=3153442895 TSecr=0 WS=64
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	74	0	443 → 55928 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSval=29358934 TSecr=31
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	1	55928 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=3153442916 TSecr=29358934
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TLSv1.2	296	1	Client Hello
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	66	1	443 → 55928 [ACK] Seq=1 Ack=231 Win=29952 Len=0 TSval=29358945 TSecr=3153442934
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TLSv1.2	162	1	Server Hello
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	1446	97	443 → 55928 [ACK] Seq=97 Ack=231 Win=29952 Len=1380 TSval=29358945 TSecr=3153442934 [TCP segmen
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	1446	1477	443 → 55928 [ACK] Seq=1477 Ack=231 Win=29952 Len=1380 TSval=29358945 TSecr=3153442934 [TCP segm
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=97 Win=65280 Len=0 TSval=3153442955 TSecr=29358945
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	1446	2857	443 → 55928 [ACK] Seq=2857 Ack=231 Win=29952 Len=1380 TSval=29358945 TSecr=3153442934 [TCP segm
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TLSv1.2	829	4237	Certificate
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TLSv1.2	404	5000	Server Key Exchange
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TLSv1.2	75	5338	Server Hello Done
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=1477 Win=64128 Len=0 TSval=3153442956 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=2857 Win=63680 Len=0 TSval=3153442957 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=4237 Win=64128 Len=0 TSval=3153442958 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=5000 Win=64128 Len=0 TSval=3153442959 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=5338 Win=64128 Len=0 TSval=3153442959 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TCP	66	231	55928 → 443 [ACK] Seq=231 Ack=5347 Win=64128 Len=0 TSval=3153442959 TSecr=29358945
Mar 29, 2021 21:27:54…	192.168.0.101	99.80.34.142	TLSv1.2	141	231	Client Key Exchange
Mar 29, 2021 21:27:54…	99.80.34.142	192.168.0.101	TCP	66	5347	443 → 55928 [ACK] Seq=5347 Ack=306 Win=29952 Len=0 TSval=29358956 TSecr=3153443039
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TLSv1.2	72	306	Change Cipher Spec
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TLSv1.2	135	312	Encrypted Handshake Message
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TCP	66	5347	443 → 55928 [ACK] Seq=5347 Ack=312 Win=29952 Len=0 TSval=29358963 TSecr=3153443110
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TCP	66	5347	443 → 55928 [ACK] Seq=5347 Ack=381 Win=29952 Len=0 TSval=29358963 TSecr=3153443114
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TLSv1.2	72	5347	Change Cipher Spec
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TLSv1.2	135	5353	Encrypted Handshake Message
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TCP	66	381	55928 → 443 [ACK] Seq=381 Ack=5353 Win=64128 Len=0 TSval=3153443134 TSecr=29358963
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TCP	66	381	55928 → 443 [ACK] Seq=381 Ack=5422 Win=64128 Len=0 TSval=3153443134 TSecr=29358963
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TCP	1506	381	55928 → 443 [ACK] Seq=381 Ack=5422 Win=64128 Len=1440 TSval=3153443142 TSecr=29358963 [TCP segm
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TLSv1.2	679	1821	Application Data
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TCP	66	5422	443 → 55928 [ACK] Seq=5422 Ack=2434 Win=35840 Len=0 TSval=29358966 TSecr=3153443142
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TLSv1.2	407	5422	Application Data
Mar 29, 2021 21:27:55…	99.80.34.142	192.168.0.101	TLSv1.2	247	5763	Application Data
Mar 29, 2021 21:27:55…	192.168.0.101	99.80.34.142	TCP	66	2434	55928 → 443 [ACK] Seq=2434 Ack=5944 Win=64128 Len=0 TSval=3153443178 TSecr=29358967
Mar 29, 2021 21:27:55…	192.168.0.101	52.210.184.200	TCP	66	95496	[TCP Keep-Alive] 58088 → 443 [ACK] Seq=95496 Ack=6509 Win=7707 Len=0 TSval=3233354138 TSecr=404
Mar 29, 2021 21:27:55…	52.210.184.200	192.168.0.101	TCP	66	6509	[TCP Keep-Alive ACK] 443 → 58088 [ACK] Seq=6509 Ack=95497 Win=425 Len=0 TSval=4048350691 TSecr=
Mar 29, 2021 21:28:00…	192.168.0.101	52.210.184.200	TLSv1.2	97	95497	Application Data
Mar 29, 2021 21:28:00…	52.210.184.200	192.168.0.101	TLSv1.2	97	6509	Application Data
Mar 29, 2021 21:28:00…	192.168.0.101	52.210.184.200	TCP	66	95528	58088 → 443 [ACK] Seq=95528 Ack=6540 Win=7707 Len=0 TSval=3233358866 TSecr=4048355399
Mar 29, 2021 21:28:01…	99.80.34.142	192.168.0.101	TLSv1.2	119	5944	Encrypted Alert
Mar 29, 2021 21:28:01…	192.168.0.101	99.80.34.142	TCP	66	2434	55928 → 443 [ACK] Seq=2434 Ack=5997 Win=64128 Len=0 TSval=3153449216 TSecr=29359567
Mar 29, 2021 21:28:01…	192.168.0.101	34.240.107.124	TCP	66	1458	[TCP Keep-Alive] 49730 → 443 [ACK] Seq=1458 Ack=1458 Win=1002 Len=0 TSval=1953917228 TSecr=8672
Mar 29, 2021 21:28:01…	34.240.107.124	192.168.0.101	TCP	66	1458	[TCP Keep-Alive ACK] 443 → 49730 [ACK] Seq=1458 Ack=1459 Win=120 Len=0 TSval=867254127 TSecr=19
Mar 29, 2021 21:28:03…	99.80.34.142	192.168.0.101	TCP	66	5997	443 → 55928 [FIN, ACK] Seq=5997 Ack=2434 Win=35840 Len=0 TSval=29359767 TSecr=3153449216
Mar 29, 2021 21:28:03…	192.168.0.101	99.80.34.142	TCP	66	2434	55928 → 443 [ACK] Seq=2434 Ack=5998 Win=64128 Len=0 TSval=3153451220 TSecr=29359767
Mar 29, 2021 21:28:03…	192.168.0.101	192.168.0.1	DHCP	332		DHCP Request - Transaction ID 0x7d722d75
Mar 29, 2021 21:28:03…	192.168.0.1	192.168.0.101	DHCP	590		DHCP ACK - Transaction ID 0x7d722d75

Figure 4.24 Private dataset benign pattern



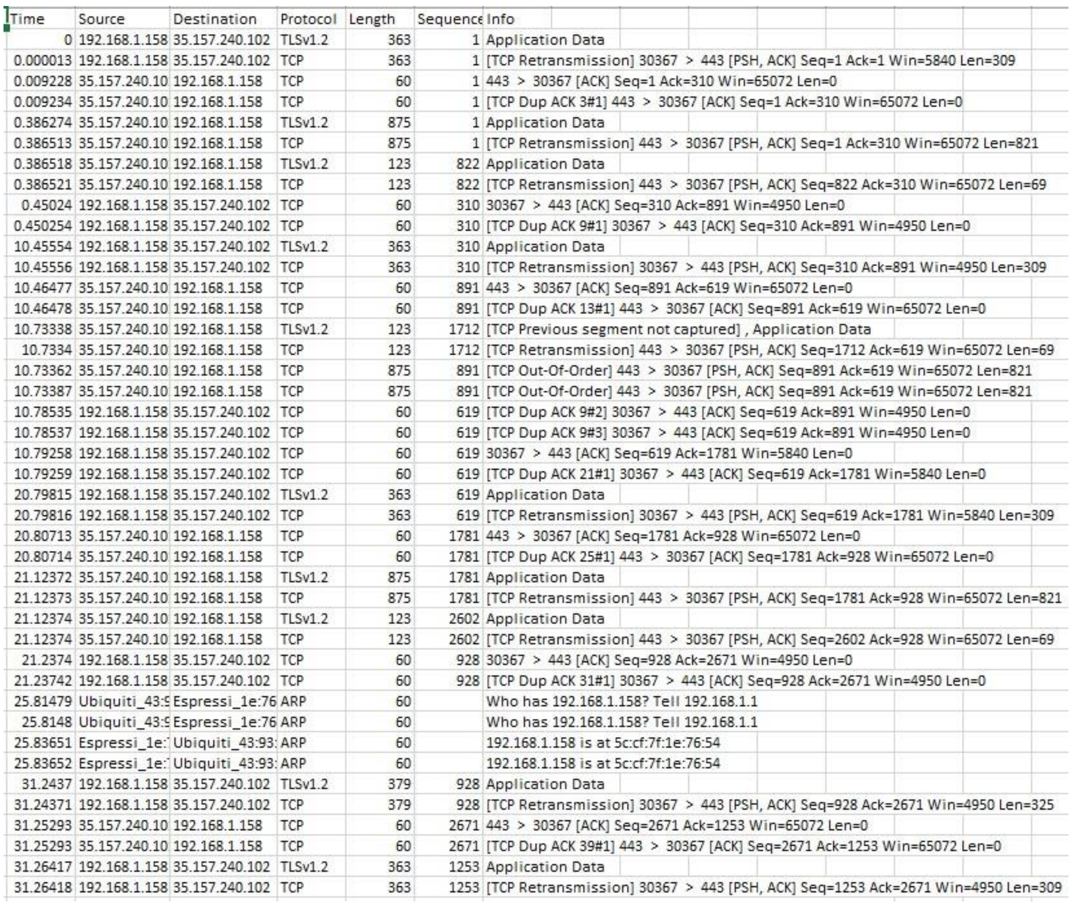
```
Arrival Time            Source           Destinatio       Protocol   Length   Sequence Nu  Info
Mar 22, 2021 14:30:01…  99.80.34.243     192.168.0.101    TCP            66          5422  443 → 43404 [ACK] Seq=5422 Ack=1938 Win=30976 Len=0 TSval=174506254 TSecr=369848852
Mar 22, 2021 14:30:01…  99.80.34.243     192.168.0.101    TLSv1.2       407          5422  Application Data
Mar 22, 2021 14:30:01…  99.80.34.243     192.168.0.101    TLSv1.2       247          5763  Application Data
Mar 22, 2021 14:30:01…  192.168.0.101    99.80.34.243     TCP            66          1938  43404 → 443 [ACK] Seq=1938 Ack=5944 Win=64128 Len=0 TSval=369848891 TSecr=174506255
Mar 22, 2021 14:30:07…  99.80.34.243     192.168.0.101    TLSv1.2       119          5944  Encrypted Alert
Mar 22, 2021 14:30:07…  192.168.0.101    34.246.152.68    TCP            66           528  [TCP Keep-Alive] 49538 → 443 [ACK] Seq=528 Ack=528 Win=2397 Len=0 TSval=164701843 TSecr=1574382898
Mar 22, 2021 14:30:07…  34.246.152.68    192.168.0.101    TCP            66           528  [TCP Keep-Alive ACK] 443 → 49538 [ACK] Seq=528 Ack=529 Win=425 Len=0 TSval=1574398164 TSecr=16468659
Mar 22, 2021 14:30:07…  192.168.0.101    99.80.34.243     TCP            66          1938  43404 → 443 [ACK] Seq=1938 Ack=5997 Win=64128 Len=0 TSval=369854931 TSecr=174506855
Mar 22, 2021 14:30:09…  99.80.34.243     192.168.0.101    TCP            66          5997  443 → 43404 [FIN, ACK] Seq=5997 Ack=1938 Win=30976 Len=0 TSval=174507056 TSecr=369854931
Mar 22, 2021 14:30:09…  192.168.0.101    99.80.34.243     TCP            66          1938  43404 → 443 [ACK] Seq=1938 Ack=5998 Win=64128 Len=0 TSval=369856931 TSecr=174507056
Mar 22, 2021 14:30:13…  192.168.0.101    192.168.0.1      DNS           125                Standard query 0x11f0 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com OPT
Mar 22, 2021 14:30:13…  192.168.0.1      192.168.0.101    DNS           173                Standard query response 0x11f0 A c3t5k8kx91jab4.credentials.iot.eu-west-1.amazonaws.com A 54.246.236
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            74             0  50624 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=1670595961 TSecr=0 WS=64
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP            74             0  443 → 50624 [SYN, ACK] Seq=0 Ack=1 Win=26847 Len=0 MSS=1452 SACK_PERM=1 TSval=1416497124 TSecr=16705
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            66             1  50624 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0 TSval=1670595981 TSecr=1416497124
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TLSv1.2       319             1  Client Hello
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP            66             1  443 → 50624 [ACK] Seq=1 Ack=254 Win=28160 Len=0 TSval=1416497242 TSecr=1670596079
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP          1506             1  443 → 50624 [ACK] Seq=1 Ack=254 Win=28160 Len=1440 TSval=1416497246 TSecr=1670596079 [TCP segment of
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP          1506          1441  443 → 50624 [ACK] Seq=1441 Ack=254 Win=28160 Len=1440 TSval=1416497246 TSecr=1670596079 [TCP segment
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP          1506          2881  443 → 50624 [ACK] Seq=2881 Ack=254 Win=28160 Len=1440 TSval=1416497246 TSecr=1670596079 [TCP segment
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TLSv1.2      1089          4321  Server Hello, Certificate, Server Key Exchange, Certificate Request, Server Hello Done
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            66           254  50624 → 443 [ACK] Seq=254 Ack=1441 Win=64128 Len=0 TSval=1670596104 TSecr=1416497246
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            66           254  50624 → 443 [ACK] Seq=254 Ack=2881 Win=64128 Len=0 TSval=1670596105 TSecr=1416497246
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            66           254  50624 → 443 [ACK] Seq=254 Ack=4321 Win=64128 Len=0 TSval=1670596106 TSecr=1416497246
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TCP            66           254  50624 → 443 [ACK] Seq=254 Ack=5344 Win=64128 Len=0 TSval=1670596107 TSecr=1416497246
Mar 22, 2021 14:30:13…  192.168.0.101    54.246.236.187   TLSv1.2      1029           254  Certificate, Client Key Exchange
Mar 22, 2021 14:30:13…  54.246.236.187   192.168.0.101    TCP            66          5344  443 → 50624 [ACK] Seq=5344 Ack=1217 Win=29952 Len=0 TSval=1416497511 TSecr=1670596305
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TLSv1.2       335          1217  Certificate Verify
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TLSv1.2        72          1486  Change Cipher Spec
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TLSv1.2       167          1492  Encrypted Handshake Message
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TCP            66          5344  443 → 50624 [ACK] Seq=5344 Ack=1486 Win=32000 Len=0 TSval=1416497942 TSecr=1670596780
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TCP            66          5344  443 → 50624 [ACK] Seq=5344 Ack=1492 Win=32000 Len=0 TSval=1416497943 TSecr=1670596780
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TCP            66          5344  443 → 50624 [ACK] Seq=5344 Ack=1593 Win=32000 Len=0 TSval=1416497956 TSecr=1670596791
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TLSv1.2       173          5344  Change Cipher Spec, Encrypted Handshake Message
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TCP            66          1593  50624 → 443 [ACK] Seq=1593 Ack=5451 Win=64128 Len=0 TSval=1670596813 TSecr=1416497957
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TLSv1.2       375          1593  Application Data
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TCP          1506          5451  443 → 50624 [ACK] Seq=5451 Ack=1902 Win=33792 Len=1440 TSval=1416498032 TSecr=1670596839 [TCP segmen
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TLSv1.2       103          6891  Application Data
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TLSv1.2       151          6928  Encrypted Alert
Mar 22, 2021 14:30:14…  54.246.236.187   192.168.0.101    TCP            66          7013  443 → 50624 [FIN, ACK] Seq=7013 Ack=1902 Win=33792 Len=0 TSval=1416498032 TSecr=1670596839
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TCP            66          1902  50624 → 443 [ACK] Seq=1902 Ack=7014 Win=62784 Len=0 TSval=1670596891 TSecr=1416498032
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TLSv1.2       151          1902  Encrypted Alert
Mar 22, 2021 14:30:14…  192.168.0.101    54.246.236.187   TCP            66          1987  50624 → 443 [RST, ACK] Seq=1987 Ack=7014 Win=64128 Len=0 TSval=1670596910 TSecr=1416498032
Mar 22, 2021 14:30:14…  192.168.0.101    192.168.0.1      DNS           103                Standard query 0x68fc A dynamodb.eu-west-1.amazonaws.com OPT
Mar 22, 2021 14:30:14…  192.168.0.1      192.168.0.101    DNS           119                Standard query response 0x68fc A dynamodb.eu-west-1.amazonaws.com A 52.119.244.93 OPT
Mar 22, 2021 14:30:14…  192.168.0.101    52.119.244.93    TCP            74             0  51394 → 443 [SYN] Seq=0 Win=65340 Len=0 MSS=1452 SACK_PERM=1 TSval=2267871806 TSecr=0 WS=64
Mar 22, 2021 14:30:14…  52.119.244.93    192.168.0.101    TCP            66             0  443 → 51394 [SYN, ACK] Seq=0 Ack=1 Win=8190 Len=0 MSS=1440 WS=64 SACK_PERM=1
Mar 22, 2021 14:30:14…  192.168.0.101    52.119.244.93    TCP            64             1  51394 → 443 [ACK] Seq=1 Ack=1 Win=65344 Len=0
Mar 22, 2021 14:30:14…  52.119.244.93    192.168.0.101    TCP            60             1  [TCP Window Update] 443 → 51394 [ACK] Seq=1 Ack=1 Win=29696 Len=0
Mar 22, 2021 14:30:14…  192.168.0.101    52.119.244.93    TLSv1.2       285             1  Client Hello
```

Figure 4.25 Private dataset benign pattern



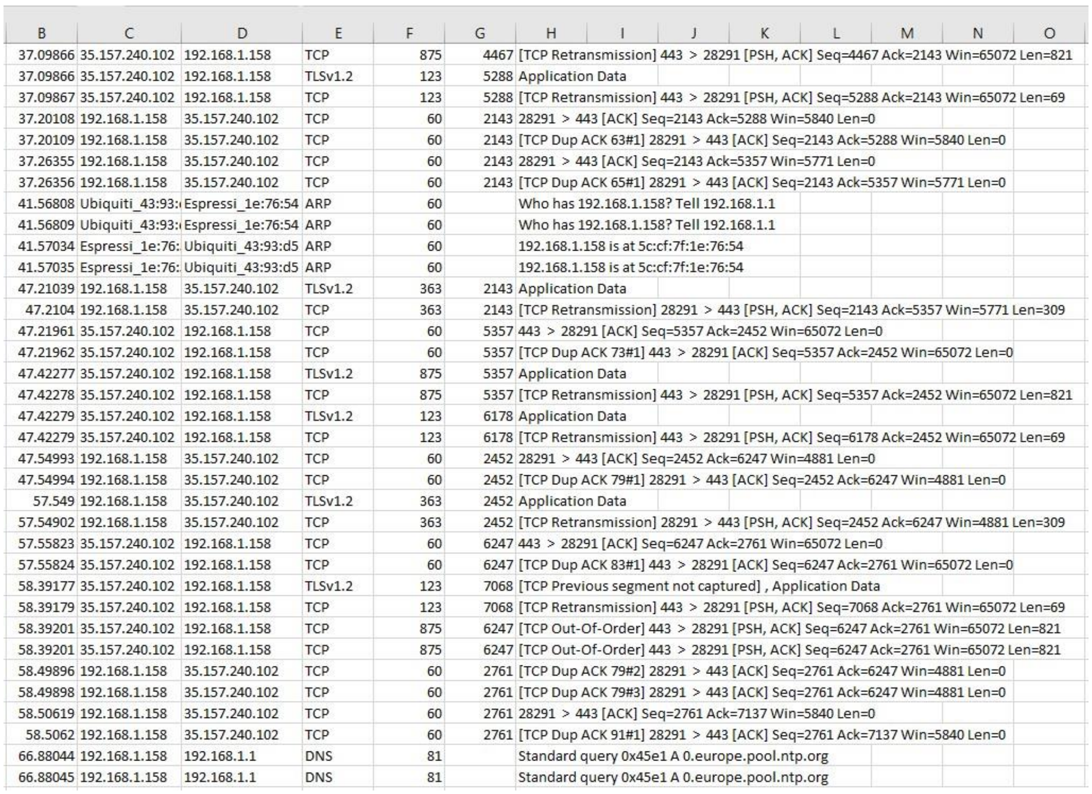
```
Arrival Time      Source           Destinatio     Protocol  Length  Sequence Nu  Info
Dec 21, 2023 21…  20.49.172.247    192.168.0.103  TLSv1.2       85            1  Application Data
Dec 21, 2023 21…  52.123.137.141   192.168.0.103  TLSv1.2       99           46  Application Data
Dec 21, 2023 21…  192.168.0.1      192.168.0.103  DNS           78               Standard query response 0xea61 Server failure A wpad.ads.ntu.ac.uk
Dec 21, 2023 21…  20.49.172.247    192.168.0.103  TLSv1.2       85           32  Application Data
Dec 21, 2023 21…  52.113.205.22    192.168.0.103  TLSv1.2      102           49  Application Data
Dec 21, 2023 21…  192.168.0.1      192.168.0.103  DNS          292               Standard query response 0x8e00 A ctldl.windowsupdate.com CNAME wu-bg-shim.traf
Dec 21, 2023 21…  93.184.221.240   192.168.0.103  TCP           66            0  80 → 49820 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1452 SACK_PERM=1 WS=512
Dec 21, 2023 21…  93.184.221.240   192.168.0.103  TCP           54            1  80 → 49820 [ACK] Seq=1 Ack=287 Win=67072 Len=0
Dec 21, 2023 21…  93.184.221.240   192.168.0.103  HTTP         343            1  HTTP/1.1 304 Not Modified
Dec 21, 2023 21…  93.184.221.240   192.168.0.103  HTTP         345          290  HTTP/1.1 304 Not Modified
Dec 21, 2023 21…  192.168.0.1      192.168.0.103  DNS           78               Standard query response 0x039f Server failure A wpad.ads.ntu.ac.uk
Dec 21, 2023 21…  192.168.0.1      192.168.0.103  DNS           74               Standard query response 0x3707 No such name A wpad.ntu.ac.uk
Dec 21, 2023 21…  20.49.172.247    192.168.0.103  TLSv1.2       85           63  Application Data
Dec 21, 2023 21…  13.107.42.16     192.168.0.103  TCP           54            1  443 → 49808 [RST, ACK] Seq=1 Ack=2 Win=0 Len=0
Dec 21, 2023 21…  192.168.0.1      192.168.0.103  DNS          158               Standard query response 0x01c9 A presence.teams.microsoft.com CNAME presence.s
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP           66            0  443 → 49821 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1440 WS=256 SACK_PERM=1
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP         1506            1  443 → 49821 [ACK] Seq=1 Ack=518 Win=4194304 Len=1452 [TCP segment of a reassem
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP         1506         1453  443 → 49821 [ACK] Seq=1453 Ack=518 Win=4194304 Len=1452 [TCP segment of a reas
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP         1506         2905  443 → 49821 [ACK] Seq=2905 Ack=518 Win=4194304 Len=1452 [TCP segment of a reas
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP         1506         4357  443 → 49821 [ACK] Seq=4357 Ack=518 Win=4194304 Len=1452 [TCP segment of a reas
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TLSv1.2      443         5809  Server Hello, Certificate, Certificate Status, Server Key Exchange, Server Hel
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP           54         6198  443 → 49821 [ACK] Seq=6198 Ack=775 Win=4194048 Len=0
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TCP           54         6198  443 → 49821 [ACK] Seq=6198 Ack=3166 Win=4194816 Len=0
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TLSv1.2      105         6198  Change Cipher Spec, Encrypted Handshake Message
Dec 21, 2023 21…  52.123.144.72    192.168.0.103  TLSv1.2      123         6249  Application Data
```

Figure 4.26 Private dataset benign pattern



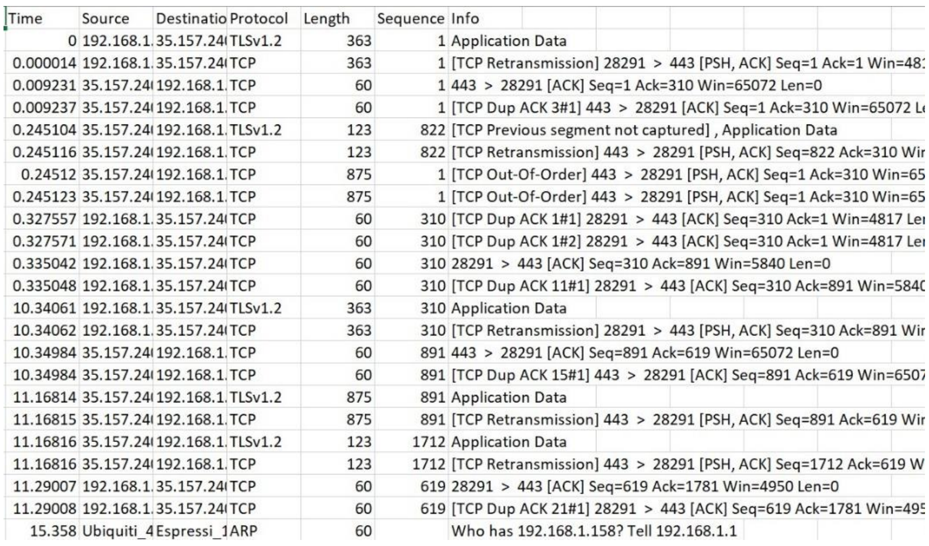
Time	Source	Destination	Protocol	Length	Sequence	Info
0	192.168.1.158	35.157.240.102	TLSv1.2	363	1	Application Data
0.000013	192.168.1.158	35.157.240.102	TCP	363	1	[TCP Retransmission] 30367 > 443 [PSH, ACK] Seq=1 Ack=1 Win=5840 Len=309
0.009228	35.157.240.10	192.168.1.158	TCP	60	1	443 > 30367 [ACK] Seq=1 Ack=310 Win=65072 Len=0
0.009234	35.157.240.10	192.168.1.158	TCP	60	1	[TCP Dup ACK 3#1] 443 > 30367 [ACK] Seq=1 Ack=310 Win=65072 Len=0
0.386274	35.157.240.10	192.168.1.158	TLSv1.2	875	1	Application Data
0.386513	35.157.240.10	192.168.1.158	TCP	875	1	[TCP Retransmission] 443 > 30367 [PSH, ACK] Seq=1 Ack=310 Win=65072 Len=821
0.386518	35.157.240.10	192.168.1.158	TLSv1.2	123	822	Application Data
0.386521	35.157.240.10	192.168.1.158	TCP	123	822	[TCP Retransmission] 443 > 30367 [PSH, ACK] Seq=822 Ack=310 Win=65072 Len=69
0.45024	192.168.1.158	35.157.240.102	TCP	60	310	30367 > 443 [ACK] Seq=310 Ack=891 Win=4950 Len=0
0.450254	192.168.1.158	35.157.240.102	TCP	60	310	[TCP Dup ACK 9#1] 30367 > 443 [ACK] Seq=310 Ack=891 Win=4950 Len=0
10.45554	192.168.1.158	35.157.240.102	TLSv1.2	363	310	Application Data
10.45556	192.168.1.158	35.157.240.102	TCP	363	310	[TCP Retransmission] 30367 > 443 [PSH, ACK] Seq=310 Ack=891 Win=4950 Len=309
10.46477	35.157.240.10	192.168.1.158	TCP	60	891	443 > 30367 [ACK] Seq=891 Ack=619 Win=65072 Len=0
10.46478	35.157.240.10	192.168.1.158	TCP	60	891	[TCP Dup ACK 13#1] 443 > 30367 [ACK] Seq=891 Ack=619 Win=65072 Len=0
10.73338	35.157.240.10	192.168.1.158	TLSv1.2	123	1712	[TCP Previous segment not captured] , Application Data
10.7334	35.157.240.10	192.168.1.158	TCP	123	1712	[TCP Retransmission] 443 > 30367 [PSH, ACK] Seq=1712 Ack=619 Win=65072 Len=69
10.73362	35.157.240.10	192.168.1.158	TCP	875	891	[TCP Out-Of-Order] 443 > 30367 [PSH, ACK] Seq=891 Ack=619 Win=65072 Len=821
10.73387	35.157.240.10	192.168.1.158	TCP	875	891	[TCP Out-Of-Order] 443 > 30367 [PSH, ACK] Seq=891 Ack=619 Win=65072 Len=821
10.78535	192.168.1.158	35.157.240.102	TCP	60	619	[TCP Dup ACK 9#2] 30367 > 443 [ACK] Seq=619 Ack=891 Win=4950 Len=0
10.78537	192.168.1.158	35.157.240.102	TCP	60	619	[TCP Dup ACK 9#3] 30367 > 443 [ACK] Seq=619 Ack=891 Win=4950 Len=0
10.79258	192.168.1.158	35.157.240.102	TCP	60	619	30367 > 443 [ACK] Seq=619 Ack=1781 Win=5840 Len=0
10.79259	192.168.1.158	35.157.240.102	TCP	60	619	[TCP Dup ACK 21#1] 30367 > 443 [ACK] Seq=619 Ack=1781 Win=5840 Len=0
20.79815	192.168.1.158	35.157.240.102	TLSv1.2	363	619	Application Data
20.79816	192.168.1.158	35.157.240.102	TCP	363	619	[TCP Retransmission] 30367 > 443 [PSH, ACK] Seq=619 Ack=1781 Win=5840 Len=309
20.80713	35.157.240.10	192.168.1.158	TCP	60	1781	443 > 30367 [ACK] Seq=1781 Ack=928 Win=65072 Len=0
20.80714	35.157.240.10	192.168.1.158	TCP	60	1781	[TCP Dup ACK 25#1] 443 > 30367 [ACK] Seq=1781 Ack=928 Win=65072 Len=0
21.12372	35.157.240.10	192.168.1.158	TLSv1.2	875	1781	Application Data
21.12373	35.157.240.10	192.168.1.158	TCP	875	1781	[TCP Retransmission] 443 > 30367 [PSH, ACK] Seq=1781 Ack=928 Win=65072 Len=821
21.12374	35.157.240.10	192.168.1.158	TLSv1.2	123	2602	Application Data
21.12374	35.157.240.10	192.168.1.158	TCP	123	2602	[TCP Retransmission] 443 > 30367 [PSH, ACK] Seq=2602 Ack=928 Win=65072 Len=69
21.2374	192.168.1.158	35.157.240.102	TCP	60	928	30367 > 443 [ACK] Seq=928 Ack=2671 Win=4950 Len=0
21.23742	192.168.1.158	35.157.240.102	TCP	60	928	[TCP Dup ACK 31#1] 30367 > 443 [ACK] Seq=928 Ack=2671 Win=4950 Len=0
25.81479	Ubiquiti_43:9	Espressi_1e:76	ARP	60		Who has 192.168.1.158? Tell 192.168.1.1
25.8148	Ubiquiti_43:9	Espressi_1e:76	ARP	60		Who has 192.168.1.158? Tell 192.168.1.1
25.83651	Espressi_1e:	Ubiquiti_43:93:	ARP	60		192.168.1.158 is at 5c:cf:7f:1e:76:54
25.83652	Espressi_1e:	Ubiquiti_43:93:	ARP	60		192.168.1.158 is at 5c:cf:7f:1e:76:54
31.2437	192.168.1.158	35.157.240.102	TLSv1.2	379	928	Application Data
31.24371	192.168.1.158	35.157.240.102	TCP	379	928	[TCP Retransmission] 30367 > 443 [PSH, ACK] Seq=928 Ack=2671 Win=4950 Len=325
31.25293	35.157.240.10	192.168.1.158	TCP	60	2671	443 > 30367 [ACK] Seq=2671 Ack=1253 Win=65072 Len=0
31.25293	35.157.240.10	192.168.1.158	TCP	60	2671	[TCP Dup ACK 39#1] 443 > 30367 [ACK] Seq=2671 Ack=1253 Win=65072 Len=0
31.26417	192.168.1.158	35.157.240.102	TLSv1.2	363	1253	Application Data
31.26418	192.168.1.158	35.157.240.102	TCP	363	1253	[TCP Retransmission] 30367 > 443 [PSH, ACK] Seq=1253 Ack=2671 Win=4950 Len=309

Figure 4.27 Public dataset benign pattern



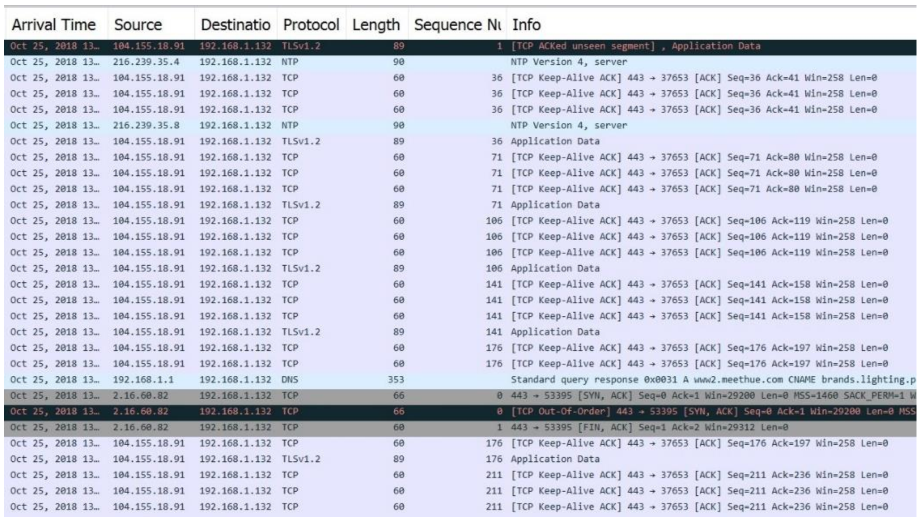
B	C	D	E	F	G	H	I	J	K	L	M	N	O
37.09866	35.157.240.102	192.168.1.158	TCP	875	4467	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=4467 Ack=2143 Win=65072 Len=821							
37.09866	35.157.240.102	192.168.1.158	TLSv1.2	123	5288	Application Data							
37.09867	35.157.240.102	192.168.1.158	TCP	123	5288	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=5288 Ack=2143 Win=65072 Len=69							
37.20108	192.168.1.158	35.157.240.102	TCP	60	2143	28291 > 443 [ACK] Seq=2143 Ack=5288 Win=5840 Len=0							
37.20109	192.168.1.158	35.157.240.102	TCP	60	2143	[TCP Dup ACK 63#1] 28291 > 443 [ACK] Seq=2143 Ack=5288 Win=5840 Len=0							
37.26355	192.168.1.158	35.157.240.102	TCP	60	2143	28291 > 443 [ACK] Seq=2143 Ack=5357 Win=5771 Len=0							
37.26356	192.168.1.158	35.157.240.102	TCP	60	2143	[TCP Dup ACK 65#1] 28291 > 443 [ACK] Seq=2143 Ack=5357 Win=5771 Len=0							
41.56808	Ubiquiti_43:93:	Espressi_1e:76:54	ARP	60		Who has 192.168.1.158? Tell 192.168.1.1							
41.56809	Ubiquiti_43:93:	Espressi_1e:76:54	ARP	60		Who has 192.168.1.158? Tell 192.168.1.1							
41.57034	Espressi_1e:76:	Ubiquiti_43:93:d5	ARP	60		192.168.1.158 is at 5c:cf:7f:1e:76:54							
41.57035	Espressi_1e:76:	Ubiquiti_43:93:d5	ARP	60		192.168.1.158 is at 5c:cf:7f:1e:76:54							
47.21039	192.168.1.158	35.157.240.102	TLSv1.2	363	2143	Application Data							
47.2104	192.168.1.158	35.157.240.102	TCP	363	2143	[TCP Retransmission] 28291 > 443 [PSH, ACK] Seq=2143 Ack=5357 Win=5771 Len=309							
47.21961	35.157.240.102	192.168.1.158	TCP	60	5357	443 > 28291 [ACK] Seq=5357 Ack=2452 Win=65072 Len=0							
47.21962	35.157.240.102	192.168.1.158	TCP	60	5357	[TCP Dup ACK 73#1] 443 > 28291 [ACK] Seq=5357 Ack=2452 Win=65072 Len=0							
47.42277	35.157.240.102	192.168.1.158	TLSv1.2	875	5357	Application Data							
47.42278	35.157.240.102	192.168.1.158	TCP	875	5357	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=5357 Ack=2452 Win=65072 Len=821							
47.42279	35.157.240.102	192.168.1.158	TLSv1.2	123	6178	Application Data							
47.42279	35.157.240.102	192.168.1.158	TCP	123	6178	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=6178 Ack=2452 Win=65072 Len=69							
47.54993	192.168.1.158	35.157.240.102	TCP	60	2452	28291 > 443 [ACK] Seq=2452 Ack=6247 Win=4881 Len=0							
47.54994	192.168.1.158	35.157.240.102	TCP	60	2452	[TCP Dup ACK 79#1] 28291 > 443 [ACK] Seq=2452 Ack=6247 Win=4881 Len=0							
57.549	192.168.1.158	35.157.240.102	TLSv1.2	363	2452	Application Data							
57.54902	192.168.1.158	35.157.240.102	TCP	363	2452	[TCP Retransmission] 28291 > 443 [PSH, ACK] Seq=2452 Ack=6247 Win=4881 Len=309							
57.55823	35.157.240.102	192.168.1.158	TCP	60	6247	443 > 28291 [ACK] Seq=6247 Ack=2761 Win=65072 Len=0							
57.55824	35.157.240.102	192.168.1.158	TCP	60	6247	[TCP Dup ACK 83#1] 443 > 28291 [ACK] Seq=6247 Ack=2761 Win=65072 Len=0							
58.39177	35.157.240.102	192.168.1.158	TLSv1.2	123	7068	[TCP Previous segment not captured] , Application Data							
58.39179	35.157.240.102	192.168.1.158	TCP	123	7068	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=7068 Ack=2761 Win=65072 Len=69							
58.39201	35.157.240.102	192.168.1.158	TCP	875	6247	[TCP Out-Of-Order] 443 > 28291 [PSH, ACK] Seq=6247 Ack=2761 Win=65072 Len=821							
58.39201	35.157.240.102	192.168.1.158	TCP	875	6247	[TCP Out-Of-Order] 443 > 28291 [PSH, ACK] Seq=6247 Ack=2761 Win=65072 Len=821							
58.49896	192.168.1.158	35.157.240.102	TCP	60	2761	[TCP Dup ACK 79#2] 28291 > 443 [ACK] Seq=2761 Ack=6247 Win=4881 Len=0							
58.49898	192.168.1.158	35.157.240.102	TCP	60	2761	[TCP Dup ACK 79#3] 28291 > 443 [ACK] Seq=2761 Ack=6247 Win=4881 Len=0							
58.50619	192.168.1.158	35.157.240.102	TCP	60	2761	28291 > 443 [ACK] Seq=2761 Ack=7137 Win=5840 Len=0							
58.5062	192.168.1.158	35.157.240.102	TCP	60	2761	[TCP Dup ACK 91#1] 28291 > 443 [ACK] Seq=2761 Ack=7137 Win=5840 Len=0							
66.88044	192.168.1.158	192.168.1.1	DNS	81		Standard query 0x45e1 A 0.europe.pool.ntp.org							
66.88045	192.168.1.158	192.168.1.1	DNS	81		Standard query 0x45e1 A 0.europe.pool.ntp.org							

Figure 4.28 Public dataset benign pattern



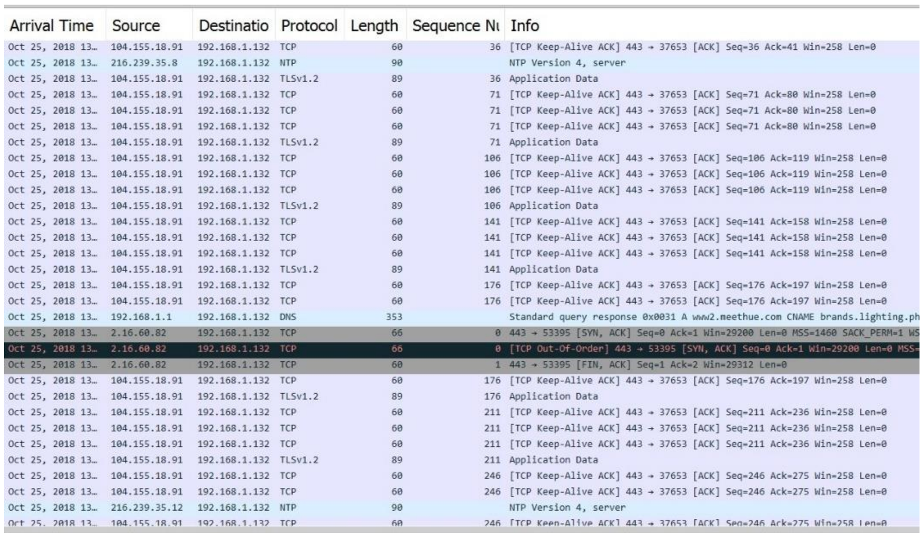
Time	Source	Destinatio	Protocol	Length	Sequence	Info
0	192.168.1.	35.157.24(	TLSv1.2	363	1	Application Data
0.000014	192.168.1.	35.157.24(	TCP	363	1	[TCP Retransmission] 28291 > 443 [PSH, ACK] Seq=1 Ack=1 Win=48:
0.009231	35.157.24(	192.168.1.	TCP	60	1	443 > 28291 [ACK] Seq=1 Ack=310 Win=65072 Len=0
0.009237	35.157.24(	192.168.1.	TCP	60	1	[TCP Dup ACK 3#1] 443 > 28291 [ACK] Seq=1 Ack=310 Win=65072 L
0.245104	35.157.24(	192.168.1.	TLSv1.2	123	822	[TCP Previous segment not captured] , Application Data
0.245116	35.157.24(	192.168.1.	TCP	123	822	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=822 Ack=310 Wir
0.24512	35.157.24(	192.168.1.	TCP	875	1	[TCP Out-Of-Order] 443 > 28291 [PSH, ACK] Seq=1 Ack=310 Win=65
0.245123	35.157.24(	192.168.1.	TCP	875	1	[TCP Out-Of-Order] 443 > 28291 [PSH, ACK] Seq=1 Ack=310 Win=65
0.327557	192.168.1.	35.157.24(	TCP	60	310	[TCP Dup ACK 1#1] 28291 > 443 [ACK] Seq=310 Ack=1 Win=4817 Le
0.327571	192.168.1.	35.157.24(	TCP	60	310	[TCP Dup ACK 1#2] 28291 > 443 [ACK] Seq=310 Ack=1 Win=4817 Le
0.335042	192.168.1.	35.157.24(	TCP	60	310	28291 > 443 [ACK] Seq=310 Ack=891 Win=5840 Len=0
0.335048	192.168.1.	35.157.24(	TCP	60	310	[TCP Dup ACK 11#1] 28291 > 443 [ACK] Seq=310 Ack=891 Win=584C
10.34061	192.168.1.	35.157.24(	TLSv1.2	363	310	Application Data
10.34062	192.168.1.	35.157.24(	TCP	363	310	[TCP Retransmission] 28291 > 443 [PSH, ACK] Seq=310 Ack=891 Wir
10.34984	35.157.24(	192.168.1.	TCP	60	891	443 > 28291 [ACK] Seq=891 Ack=619 Win=65072 Len=0
10.34984	35.157.24(	192.168.1.	TCP	60	891	[TCP Dup ACK 15#1] 443 > 28291 [ACK] Seq=891 Ack=619 Win=6507
11.16814	35.157.24(	192.168.1.	TLSv1.2	875	891	Application Data
11.16815	35.157.24(	192.168.1.	TCP	875	891	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=891 Ack=619 Wir
11.16816	35.157.24(	192.168.1.	TLSv1.2	123	1712	Application Data
11.16816	35.157.24(	192.168.1.	TCP	123	1712	[TCP Retransmission] 443 > 28291 [PSH, ACK] Seq=1712 Ack=619 W
11.29007	192.168.1.	35.157.24(	TCP	60	619	28291 > 443 [ACK] Seq=619 Ack=1781 Win=4950 Len=0
11.29008	192.168.1.	35.157.24(	TCP	60	619	[TCP Dup ACK 21#1] 28291 > 443 [ACK] Seq=619 Ack=1781 Win=495
15.358	Ubiquiti_4	Espressi_1	ARP	60		Who has 192.168.1.158? Tell 192.168.1.1

Figure 4.29 Public dataset benign pattern



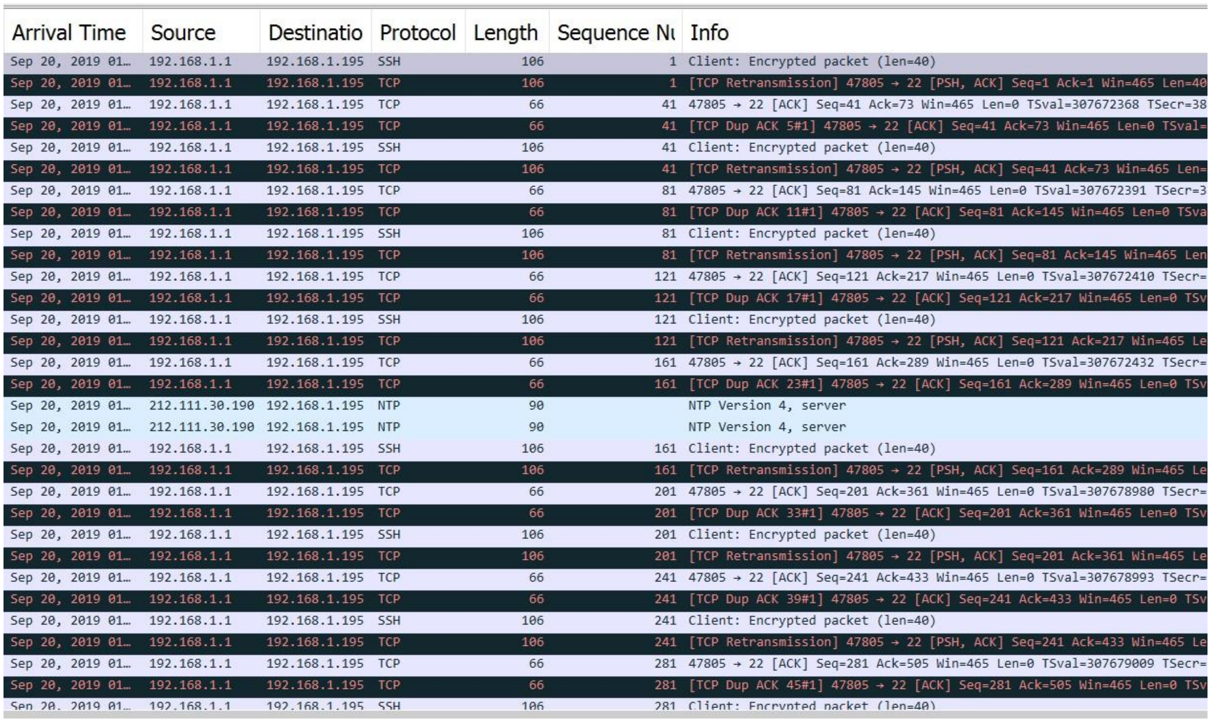
Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nu	Info
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	1	[TCP ACKed unseen segment] , Application Data
Oct 25, 2018 13…	216.239.35.4	192.168.1.132	NTP	90		NTP Version 4, server
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	36	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=36 Ack=41 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	36	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=36 Ack=41 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	36	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=36 Ack=41 Win=258 Len=0
Oct 25, 2018 13…	216.239.35.8	192.168.1.132	NTP	90		NTP Version 4, server
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	36	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	71	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	106	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	141	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	192.168.1.1	192.168.1.132	DNS	353		Standard query response 0x0031 A www2.meethue.com CNAME brands.lighting.p
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	66	0	443 → 53395 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 W
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	66	0	[TCP Out-Of-Order] 443 → 53395 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	60	1	443 → 53395 [FIN, ACK] Seq=1 Ack=2 Win=29312 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	176	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0

Figure 4.30 Public dataset benign pattern



Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nu	Info
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	36	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=36 Ack=41 Win=258 Len=0
Oct 25, 2018 13…	216.239.35.8	192.168.1.132	NTP	90		NTP Version 4, server
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	36	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	71	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=71 Ack=80 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	71	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	106	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=106 Ack=119 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	106	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	141	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=141 Ack=158 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	141	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	192.168.1.1	192.168.1.132	DNS	353		Standard query response 0x0031 A www2.meethue.com CNAME brands.lighting.ph
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	66	0	443 → 53395 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=1460 SACK_PERM=1 WS
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	66	0	[TCP Out-Of-Order] 443 → 53395 [SYN, ACK] Seq=0 Ack=1 Win=29200 Len=0 MSS=
Oct 25, 2018 13…	2.16.60.82	192.168.1.132	TCP	60	1	443 → 53395 [FIN, ACK] Seq=1 Ack=2 Win=29312 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	176	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=176 Ack=197 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	176	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	211	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=211 Ack=236 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TLSv1.2	89	211	Application Data
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	246	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=246 Ack=275 Win=258 Len=0
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	246	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=246 Ack=275 Win=258 Len=0
Oct 25, 2018 13…	216.239.35.12	192.168.1.132	NTP	90		NTP Version 4, server
Oct 25, 2018 13…	104.155.18.91	192.168.1.132	TCP	60	246	[TCP Keep-Alive ACK] 443 → 37653 [ACK] Seq=246 Ack=275 Win=258 Len=0

Figure 4.31 Public dataset benign pattern



Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nı	Info
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	1	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	1	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=1 Ack=1 Win=465 Len=40
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	41	47805 → 22 [ACK] Seq=41 Ack=73 Win=465 Len=0 TSval=307672368 TSecr=38
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	41	[TCP Dup ACK 5#1] 47805 → 22 [ACK] Seq=41 Ack=73 Win=465 Len=0 TSval=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	41	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	41	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=41 Ack=73 Win=465 Len=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	81	47805 → 22 [ACK] Seq=81 Ack=145 Win=465 Len=0 TSval=307672391 TSecr=3
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	81	[TCP Dup ACK 11#1] 47805 → 22 [ACK] Seq=81 Ack=145 Win=465 Len=0 TSva
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	81	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	81	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=81 Ack=145 Win=465 Len
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	121	47805 → 22 [ACK] Seq=121 Ack=217 Win=465 Len=0 TSval=307672410 TSecr=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	121	[TCP Dup ACK 17#1] 47805 → 22 [ACK] Seq=121 Ack=217 Win=465 Len=0 TSv
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	121	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	121	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=121 Ack=217 Win=465 Le
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	161	47805 → 22 [ACK] Seq=161 Ack=289 Win=465 Len=0 TSval=307672432 TSecr=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	161	[TCP Dup ACK 23#1] 47805 → 22 [ACK] Seq=161 Ack=289 Win=465 Len=0 TSv
Sep 20, 2019 01…	212.111.30.190	192.168.1.195	NTP	90		NTP Version 4, server
Sep 20, 2019 01…	212.111.30.190	192.168.1.195	NTP	90		NTP Version 4, server
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	161	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	161	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=161 Ack=289 Win=465 Le
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	201	47805 → 22 [ACK] Seq=201 Ack=361 Win=465 Len=0 TSval=307678980 TSecr=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	201	[TCP Dup ACK 33#1] 47805 → 22 [ACK] Seq=201 Ack=361 Win=465 Len=0 TSv
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	201	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	201	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=201 Ack=361 Win=465 Le
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	241	47805 → 22 [ACK] Seq=241 Ack=433 Win=465 Len=0 TSval=307678993 TSecr=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	241	[TCP Dup ACK 39#1] 47805 → 22 [ACK] Seq=241 Ack=433 Win=465 Len=0 TSv
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	241	Client: Encrypted packet (len=40)
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	106	241	[TCP Retransmission] 47805 → 22 [PSH, ACK] Seq=241 Ack=433 Win=465 Le
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	281	47805 → 22 [ACK] Seq=281 Ack=505 Win=465 Len=0 TSval=307679009 TSecr=
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	TCP	66	281	[TCP Dup ACK 45#1] 47805 → 22 [ACK] Seq=281 Ack=505 Win=465 Len=0 TSv
Sep 20, 2019 01…	192.168.1.1	192.168.1.195	SSH	106	281	Client: Encrypted packet (len=40)

Figure 4.32 Public dataset benign pattern

4.4.2 Attack traffic

The results from the attack flow EDA for the 3 considered attacks (TCP, UDP and ICMP) are visualized and discussed here.

- TCPSYN: This attack takes advantage of the 3-way TCP handshake, initiating the process without completing it. It floods the target server with SYN requests which arrive faster than the target server can process them, thus leaving it saturated. This results in the connection being half open as it is never acknowledged or ended. After some time of waiting without acknowledgement from the malicious source, the target server sends a bulk of TCP reset packets to the malicious source with the aim to wake it up to respond to the half open requests. During this flooding process, certain network features lose their dynamic nature and get stalled at a single state making the network pattern static. The protocol in the traffic flow gets stalled at TCP for however long the attack runs. Figure 4.33 shows the 1st 50 packets of both the private and public TCP attack traffic focusing on the protocol. Comparing these to figures 4.15 and 4.16 shows the static pattern exhibited by the protocols during the attack. The frame number of each packet is plotted against its corresponding protocol which shows the attack pattern of the packets over time protocol wise. Moreover, encryption protocols (TLsv.2) that tend to be persistent in the benign traffic are nowhere to be found in the attack traffic. The same was observed for the packet length in figure 4.34 in comparison to figures 4.17 and 4.18 where the lengths vary. The frame number of each packet is plotted against its corresponding packet length to show the pattern of attack traffic in terms of packet length. The sequence numbers also maintained a static pattern of getting stalled at 0 or 1 for all the attack packets. The respective frame numbers are also plotted against their corresponding sequence numbers to show the attack traffic pattern in terms of TCP sequence numbers. These are shown in figures 4.35 as compared to figures 4.19 and 4.20 where they exhibit a wide range of varying values. The frame numbers in the public dataset plots do not start from the first 50 as the traffic began with benign traffic. The TCP flags were also affected as the SYN flag became predominant as shown in figure 4.36 when compared to figures 4.21 and 4.22 where the flags vary. In figure 4.36 we can see a proportion of the TCP flags are RST because of the target server trying to reconnect with the malicious source.

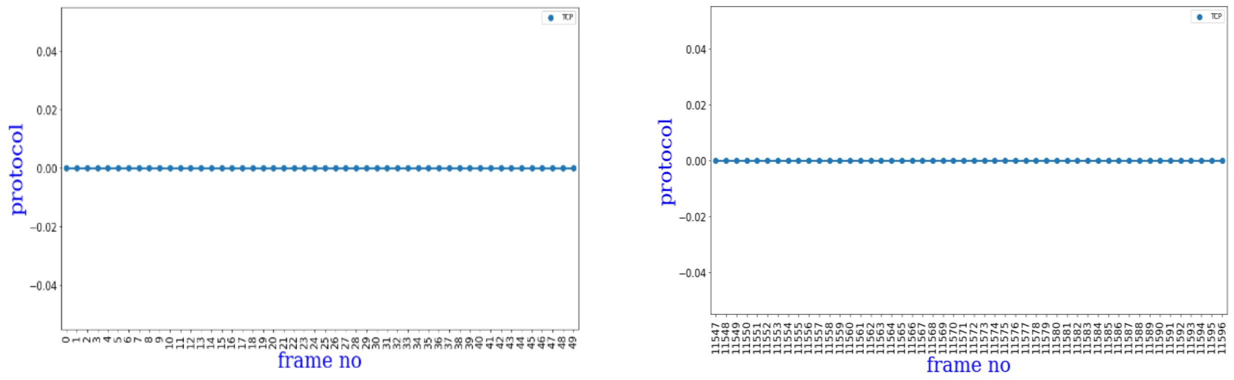


Figure 4.33 Protocol pattern, 1st 50 packets, private dataset (left) and public dataset (right)

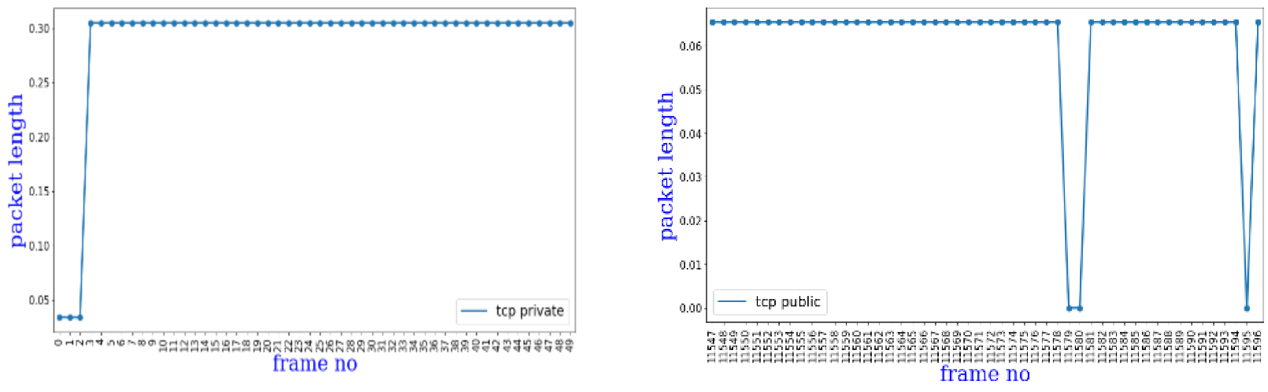


Figure 4.34 Packet length pattern, 1st 50 packets, private dataset (left) and public dataset (right)

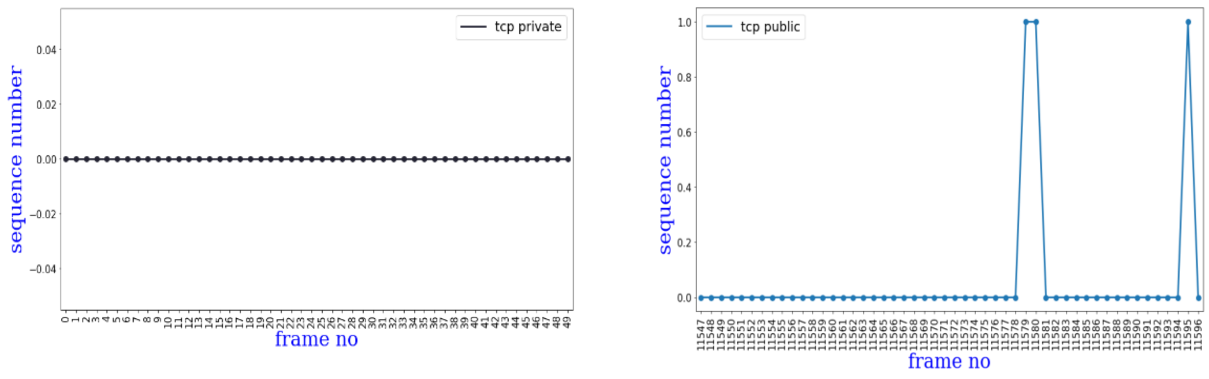


Figure 4.35 Sequence number pattern, 1st 50 packets, private dataset (left), public dataset (right)

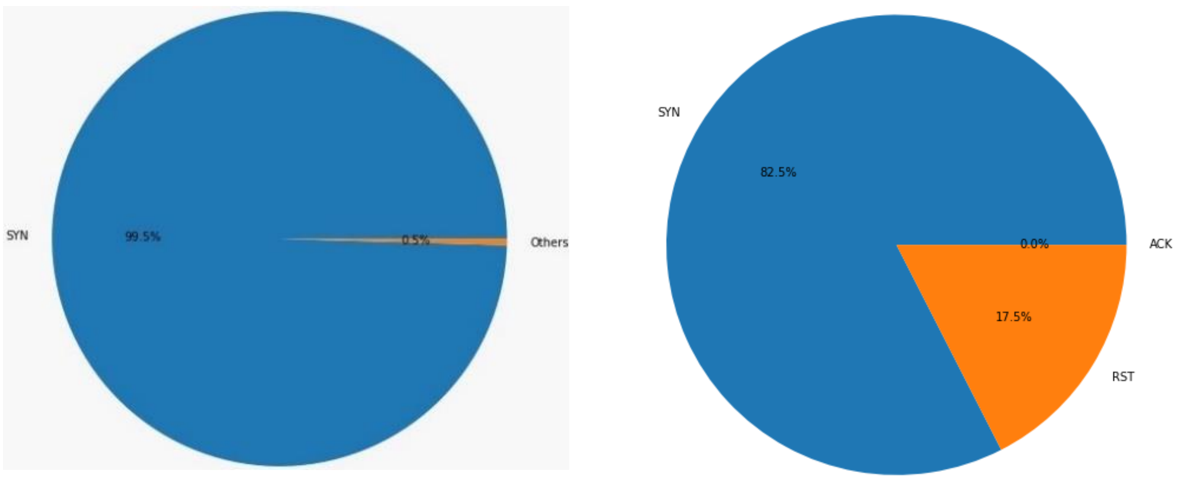


Figure 4.36 TCP flags proportion (private data on left and public data on right)

Figure 4.36 shows the dominance exhibited by the SYN flag during this attack. This unusual pattern can help in attack detection as the normal traffic pattern has far less proportion of the SYN flag. In figures 4.34 and the public dataset of 4.35 we can see that 3 packets are not uniformly aligned as the rest of the packets. This is due to normal smart home traffic making its way into the attack traffic, thus disruption the static pattern slightly.

- UDP: This attack floods a target server with UDP packets. This in turn overwhelms the server's ability to process and respond to the packets and in the process denying service to legitimate packets. The protocol and packet lengths were found to lose their varying nature as seen in the TCPSYN attack. The packet length and protocol are plotted against their corresponding frame numbers in both figures 4.37 and 4.38. Figure 4.37 shows the static protocol pattern during this attack when compared to figures 4.15 and 4.16. The same is seen in figure 4.38 for the packet lengths when compared to figures 4.17 and 4.18. The sequence numbers are absent deviating from the normal traffic pattern of having sequence numbers frequently as observed in figures 4.19 and 4.20 of the TCP based traffic. Another unusual pattern observed during this attack was the absence of encryption protocols (TLsv.2) as compared to the normal traffic pattern in figures 4.15 and 4.16 where TLsv.2 is one of the predominant protocols.

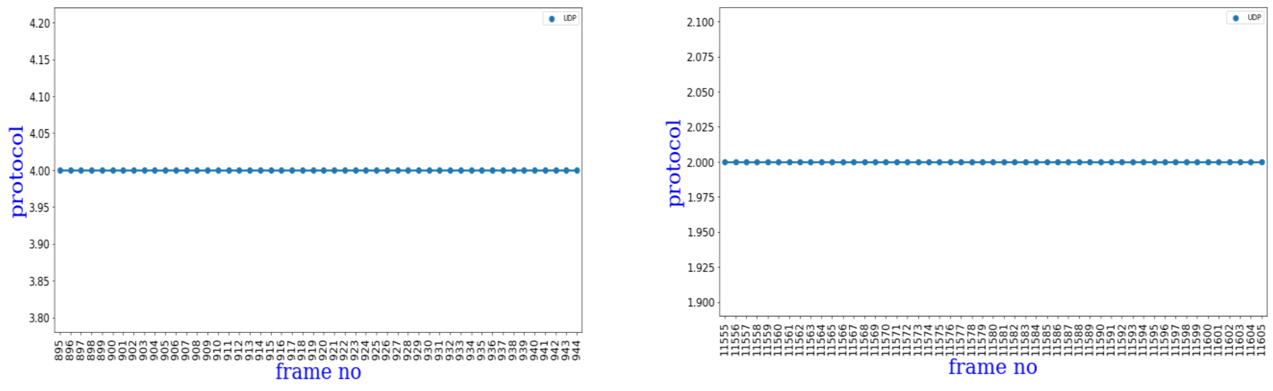


Figure 4.37 Protocol pattern, 1st 50 packets, private dataset (left) and public dataset (right)

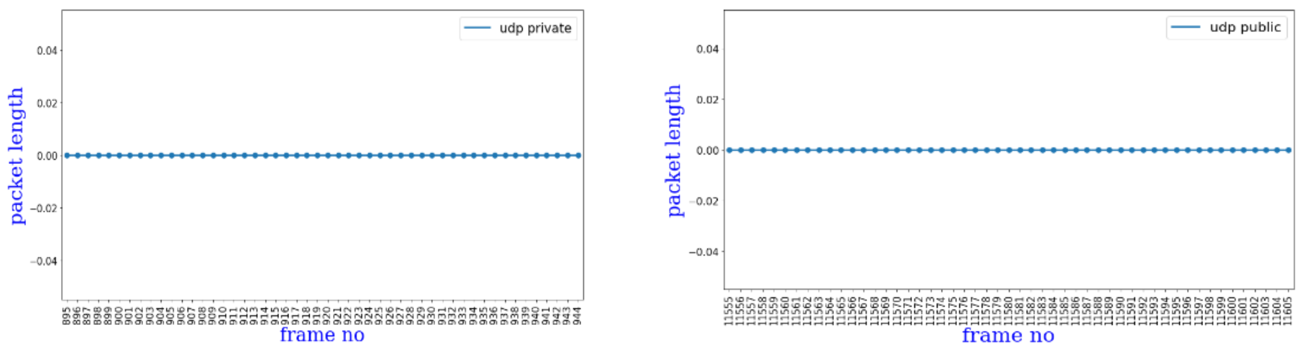


Figure 4.38 Packet length pattern, 1st 50 packets, private dataset (left) and public dataset (right)

The normal TCP based traffic as we have seen tends to have frequent TCP flags like SYN, ACK, and FIN flags. However, these flags are absent due to the UDP flood packets which deviates from a normal TCP traffic. This can also help in attack detection.

- ICMP: This attack overwhelms the target server with ICMP echo requests (pings). The server tries to process each incoming packet and responds to it and in the process failing to process legitimate packets as it is already saturated. This attack pattern is very similar to UDP attack as sequence numbers, encryption protocols and TCP flags are also absent for the duration of the attack. The packet length and protocol are plotted against their corresponding frame numbers in both figures 4.39 and 4.40. The protocols in figure 4.39 and packet lengths in figure 4.40 exhibit a static pattern. Figure 4.39 shows the protocol stalled at ICMP. In figure 4.40 we can see the packet length alternating between two lengths strictly. The echo request has a packet length of 0.000 and a reply of 0.020 in the private dataset of figure 4.40 while it is 0.000 and 0.012 for the public data. This is due to the reply packets directed to each ping. The incoming pings have the same length while the reply packets have the same length. This still opposes the varying packet length nature of the normal traffic flow from figures 4.17 and 4.18.

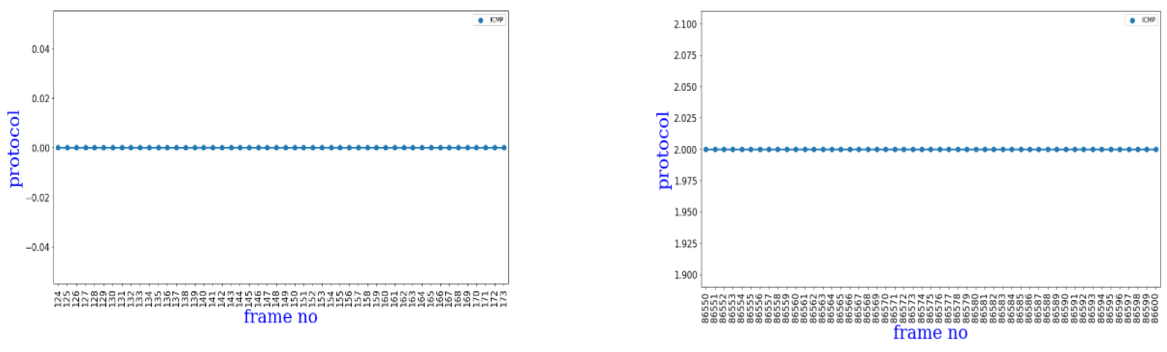


Figure 4.39 Protocol pattern, 1st 50 packets, private dataset (left) and public dataset (right)

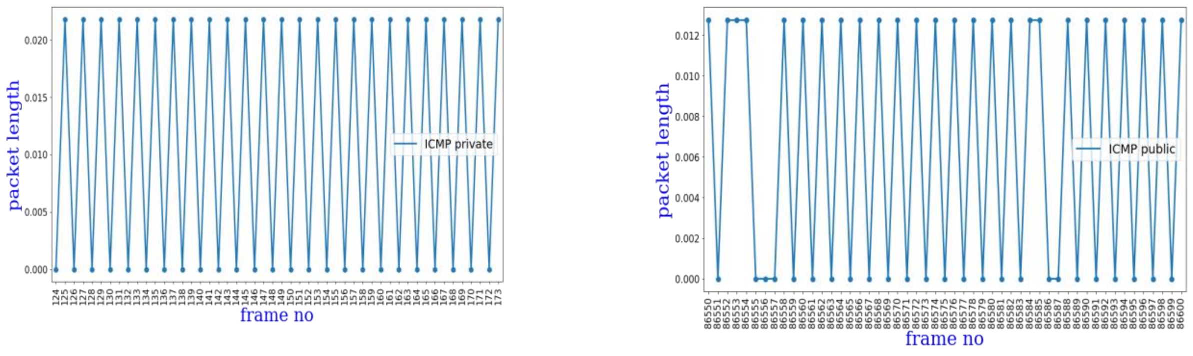
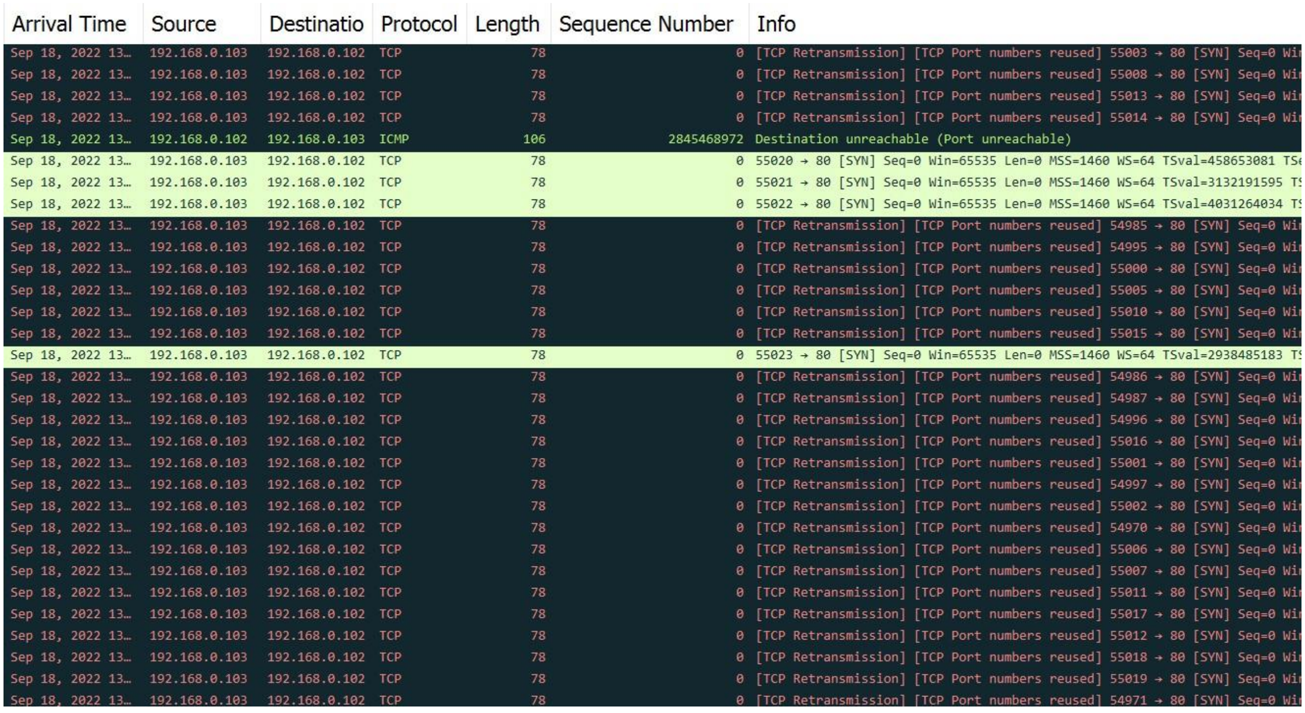


Figure 4.40 Packet length pattern, 1st 50 packets, private dataset (left) and public dataset (right)

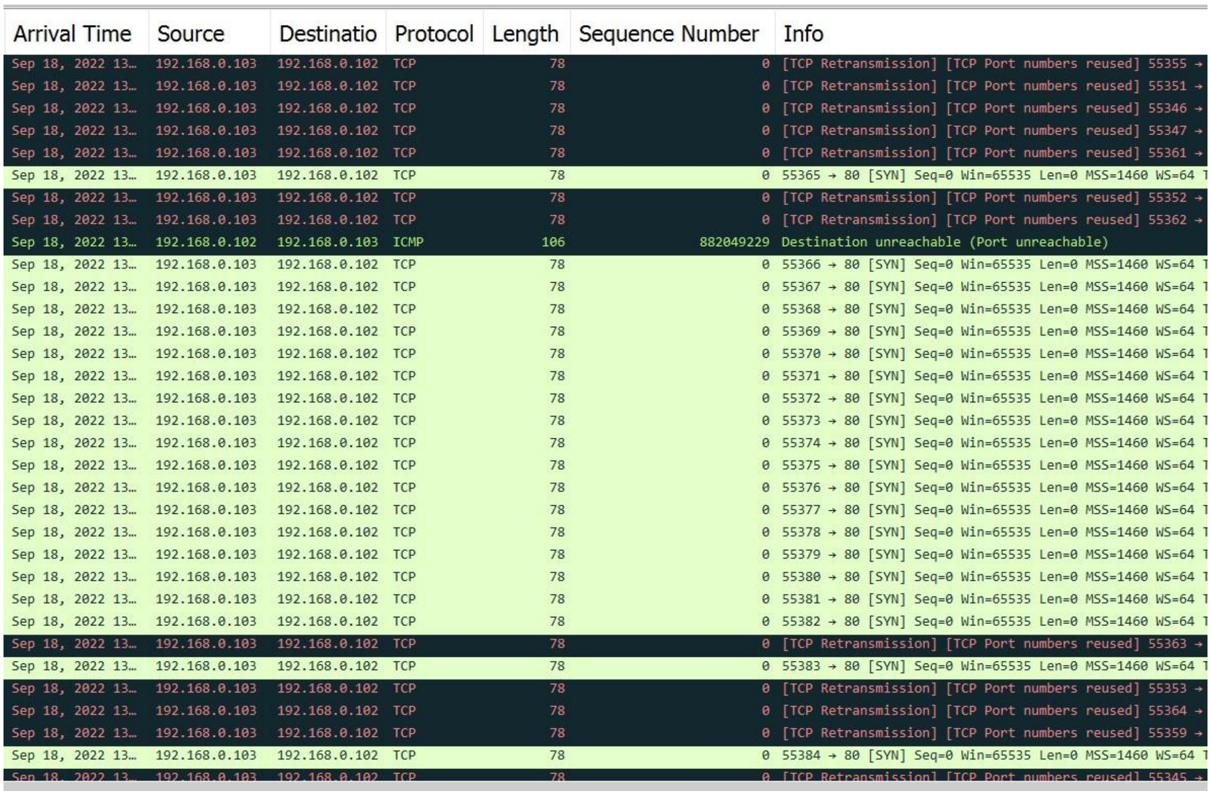
Even with the slight appearance of benign traffic amid the attack flow as observed in figures 4.34 and public dataset of 4.35, this still does not affect the predominant static effect the attack causes to the network pattern as seen from figure 4.33 to 4.40. The EDA has clearly shown the difference in pattern between a benign and attack traffic flow regardless of the dataset being analyzed.

Figures 4.41 to 4.52 show Wireshark captures of the attacks visualized. Figures 4.41 to 4.47 show captures of TCP SYN attack. The protocols under the protocol column are all TCP due to the flood. This conforms to the static pattern of the protocol in figure 4.33. The packet length under the packet length column of figures 4.41 to 4.47 are also stalled at the same length as visualized in figure 4.34. The TCP sequence numbers are also stalled at 0 or 1 as presented in figure 4.35. TCP flags which can be found in figures 4.41 to 4.47 under the info column are also stalled at the SYN flag as visualized in figure 4.36.



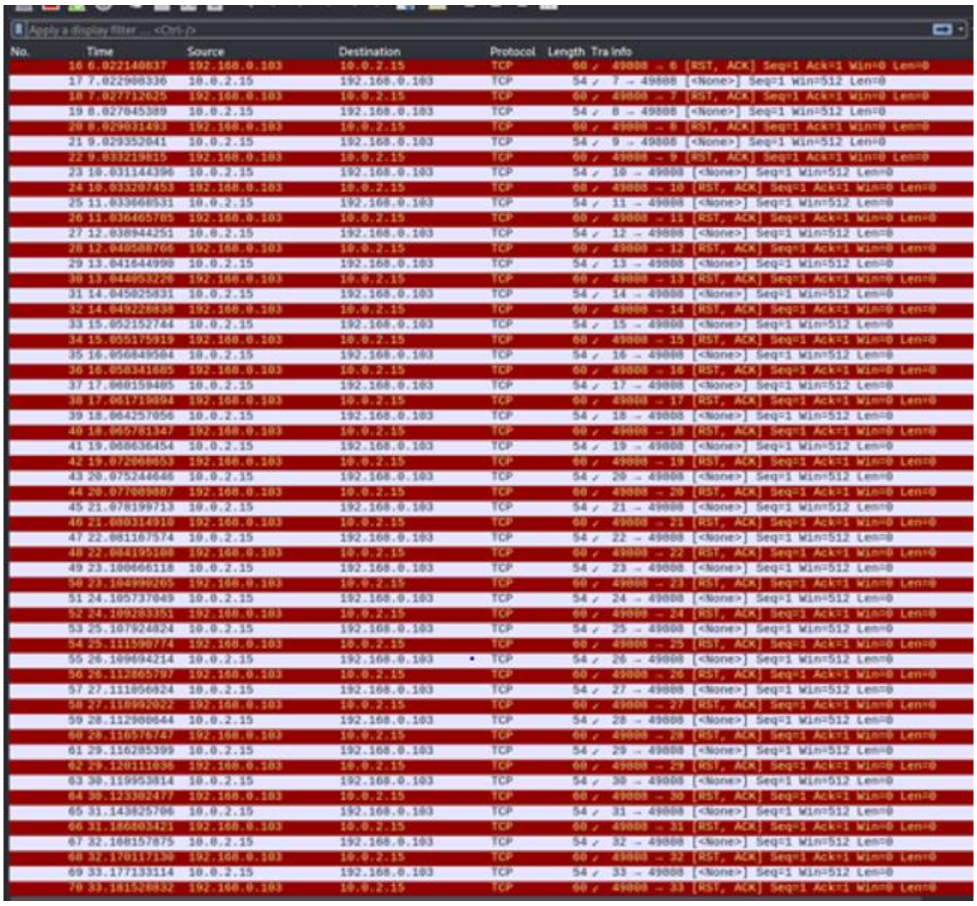
Arrival Time	Source	Destinatio	Protocol	Length	Sequence Number	Info
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55003 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55008 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55013 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55014 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.102	192.168.0.103	ICMP	106	2845468972	Destination unreachable (Port unreachable)
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55020 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=458653081 TS
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55021 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=3132191595 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55022 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=4031264034 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54985 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54995 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55000 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55005 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55010 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55015 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55023 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 TSval=2938485183 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54986 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54987 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54996 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55016 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55001 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54997 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55002 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54970 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55006 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55007 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55011 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55017 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55012 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55018 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55019 → 80 [SYN] Seq=0 Wi
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 54971 → 80 [SYN] Seq=0 Wi

Figure 4.41 TCP SYN attack using LOIC



Arrival Time	Source	Destinatio	Protocol	Length	Sequence Number	Info
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55355 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55351 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55346 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55347 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55361 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55365 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55352 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55362 →
Sep 18, 2022 13…	192.168.0.102	192.168.0.103	ICMP	106	882049229	Destination unreachable (Port unreachable)
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55366 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55367 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55368 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55369 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55370 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55371 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55372 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55373 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55374 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55375 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55376 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55377 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55378 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55379 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55380 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55381 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55382 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55363 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55383 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55353 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55364 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55359 →
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	55384 → 80 [SYN] Seq=0 Win=65535 Len=0 MSS=1460 WS=64 T
Sep 18, 2022 13…	192.168.0.103	192.168.0.102	TCP	78	0	[TCP Retransmission] [TCP Port numbers reused] 55345 →

Figure 4.42 TCP SYN attack using LOIC



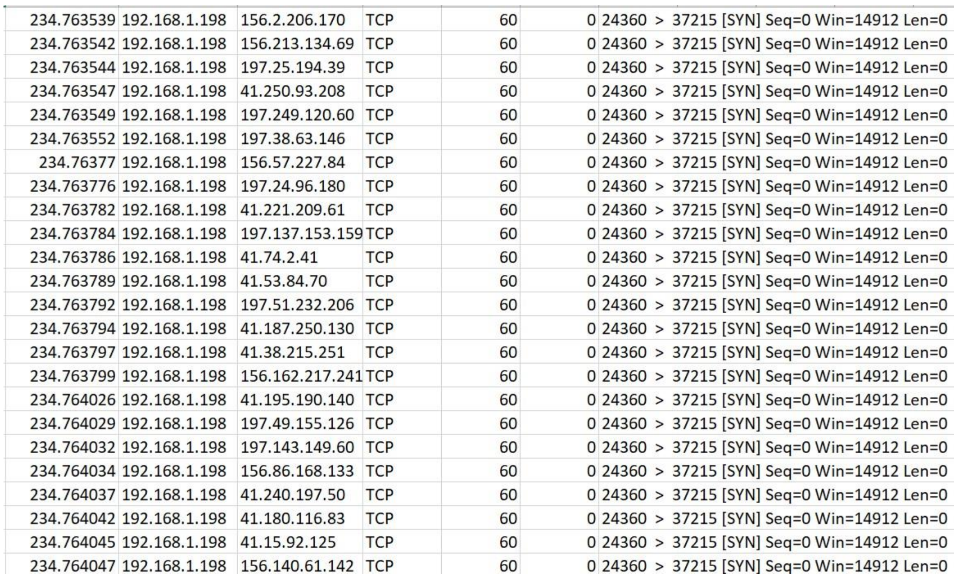
Apply a display filter … <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Tra	Info
16	6.022140837	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 6 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
17	7.022908336	10.0.2.15	192.168.0.103	TCP	54	✓	7 → 49808 [<None>] Seq=1 Win=512 Len=0
18	7.027712625	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 7 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
19	8.027845389	10.0.2.15	192.168.0.103	TCP	54	✓	8 → 49808 [<None>] Seq=1 Win=512 Len=0
20	8.029031493	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 8 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
21	9.029352041	10.0.2.15	192.168.0.103	TCP	54	✓	9 → 49808 [<None>] Seq=1 Win=512 Len=0
22	9.033219815	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 9 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
23	10.031144396	10.0.2.15	192.168.0.103	TCP	54	✓	10 → 49808 [<None>] Seq=1 Win=512 Len=0
24	10.033207453	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 10 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
25	11.033668531	10.0.2.15	192.168.0.103	TCP	54	✓	11 → 49808 [<None>] Seq=1 Win=512 Len=0
26	11.036465785	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 11 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
27	12.038944251	10.0.2.15	192.168.0.103	TCP	54	✓	12 → 49808 [<None>] Seq=1 Win=512 Len=0
28	12.040588766	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 12 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
29	13.041644990	10.0.2.15	192.168.0.103	TCP	54	✓	13 → 49808 [<None>] Seq=1 Win=512 Len=0
30	13.044053226	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 13 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
31	14.045025831	10.0.2.15	192.168.0.103	TCP	54	✓	14 → 49808 [<None>] Seq=1 Win=512 Len=0
32	14.049228838	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 14 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
33	15.052152744	10.0.2.15	192.168.0.103	TCP	54	✓	15 → 49808 [<None>] Seq=1 Win=512 Len=0
34	15.055175919	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 15 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
35	16.056849504	10.0.2.15	192.168.0.103	TCP	54	✓	16 → 49808 [<None>] Seq=1 Win=512 Len=0
36	16.058341685	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 16 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
37	17.060159405	10.0.2.15	192.168.0.103	TCP	54	✓	17 → 49808 [<None>] Seq=1 Win=512 Len=0
38	17.061719894	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 17 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
39	18.064257056	10.0.2.15	192.168.0.103	TCP	54	✓	18 → 49808 [<None>] Seq=1 Win=512 Len=0
40	18.065781347	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 18 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
41	19.068636454	10.0.2.15	192.168.0.103	TCP	54	✓	19 → 49808 [<None>] Seq=1 Win=512 Len=0
42	19.072068653	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 19 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
43	20.075244646	10.0.2.15	192.168.0.103	TCP	54	✓	20 → 49808 [<None>] Seq=1 Win=512 Len=0
44	20.077009887	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 20 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
45	21.078199713	10.0.2.15	192.168.0.103	TCP	54	✓	21 → 49808 [<None>] Seq=1 Win=512 Len=0
46	21.080314910	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 21 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
47	22.081167574	10.0.2.15	192.168.0.103	TCP	54	✓	22 → 49808 [<None>] Seq=1 Win=512 Len=0
48	22.084195108	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 22 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
49	23.100666118	10.0.2.15	192.168.0.103	TCP	54	✓	23 → 49808 [<None>] Seq=1 Win=512 Len=0
50	23.104990265	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 23 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
51	24.105737049	10.0.2.15	192.168.0.103	TCP	54	✓	24 → 49808 [<None>] Seq=1 Win=512 Len=0
52	24.109283351	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 24 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
53	25.107924824	10.0.2.15	192.168.0.103	TCP	54	✓	25 → 49808 [<None>] Seq=1 Win=512 Len=0
54	25.111590774	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 25 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
55	26.109694214	10.0.2.15	192.168.0.103	TCP	54	✓	26 → 49808 [<None>] Seq=1 Win=512 Len=0
56	26.112865797	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 26 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
57	27.111056824	10.0.2.15	192.168.0.103	TCP	54	✓	27 → 49808 [<None>] Seq=1 Win=512 Len=0
58	27.118992022	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 27 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
59	28.112908644	10.0.2.15	192.168.0.103	TCP	54	✓	28 → 49808 [<None>] Seq=1 Win=512 Len=0
60	28.116576747	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 28 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
61	29.116285399	10.0.2.15	192.168.0.103	TCP	54	✓	29 → 49808 [<None>] Seq=1 Win=512 Len=0
62	29.120111036	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 29 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
63	30.119953814	10.0.2.15	192.168.0.103	TCP	54	✓	30 → 49808 [<None>] Seq=1 Win=512 Len=0
64	30.123302477	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 30 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
65	31.143825706	10.0.2.15	192.168.0.103	TCP	54	✓	31 → 49808 [<None>] Seq=1 Win=512 Len=0
66	31.186803421	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 31 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
67	32.168157875	10.0.2.15	192.168.0.103	TCP	54	✓	32 → 49808 [<None>] Seq=1 Win=512 Len=0
68	32.170117130	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 32 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0
69	33.177133114	10.0.2.15	192.168.0.103	TCP	54	✓	33 → 49808 [<None>] Seq=1 Win=512 Len=0
70	33.181528832	192.168.0.103	10.0.2.15	TCP	60	✓	49808 → 33 [RST, ACK] Seq=1 Ack=1 Win=0 Len=0

Figure 4.43 TCP SYN attack using Hping3

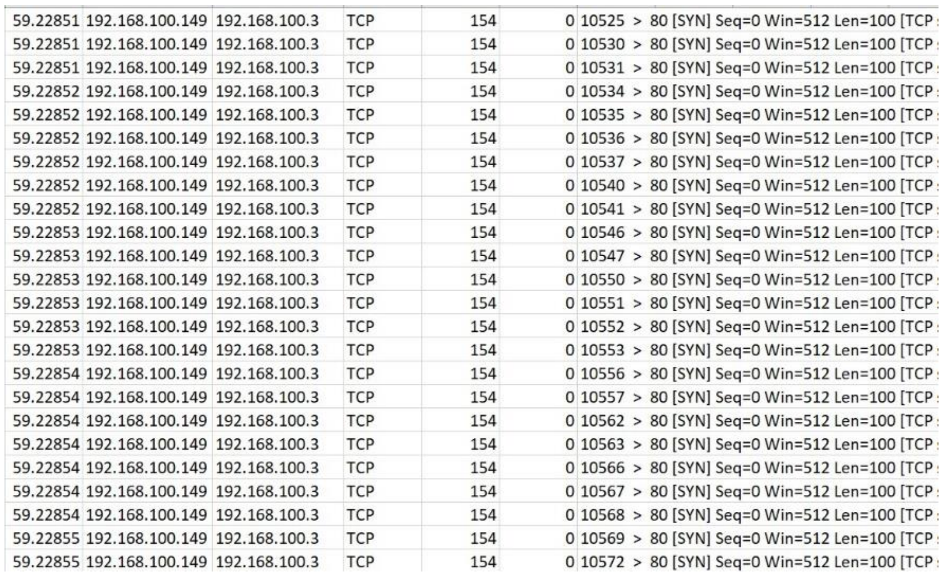
Figure 4.48 and 4.49 show Wireshark captures of UDP flood attack. The protocols under the protocol column are stalled at UDP as shown in figure 4.37. The packet length is also stalled on the same number all through as presented in figure 4.38.

Figures 4.50 to 4.52 show the Wireshark captures of ICMP flood attack. The protocols under the protocol columns are all stalled at ICMP as presented in figure 4.39. The packet length in figures 4.50 and 4.51 are stalled at 2 lengths as one is for the request while the other is for reply as shown in figure 4.40. However, the packet length in figure 4.52 is just a single length of 42 which is the echo request length as no replies are received.



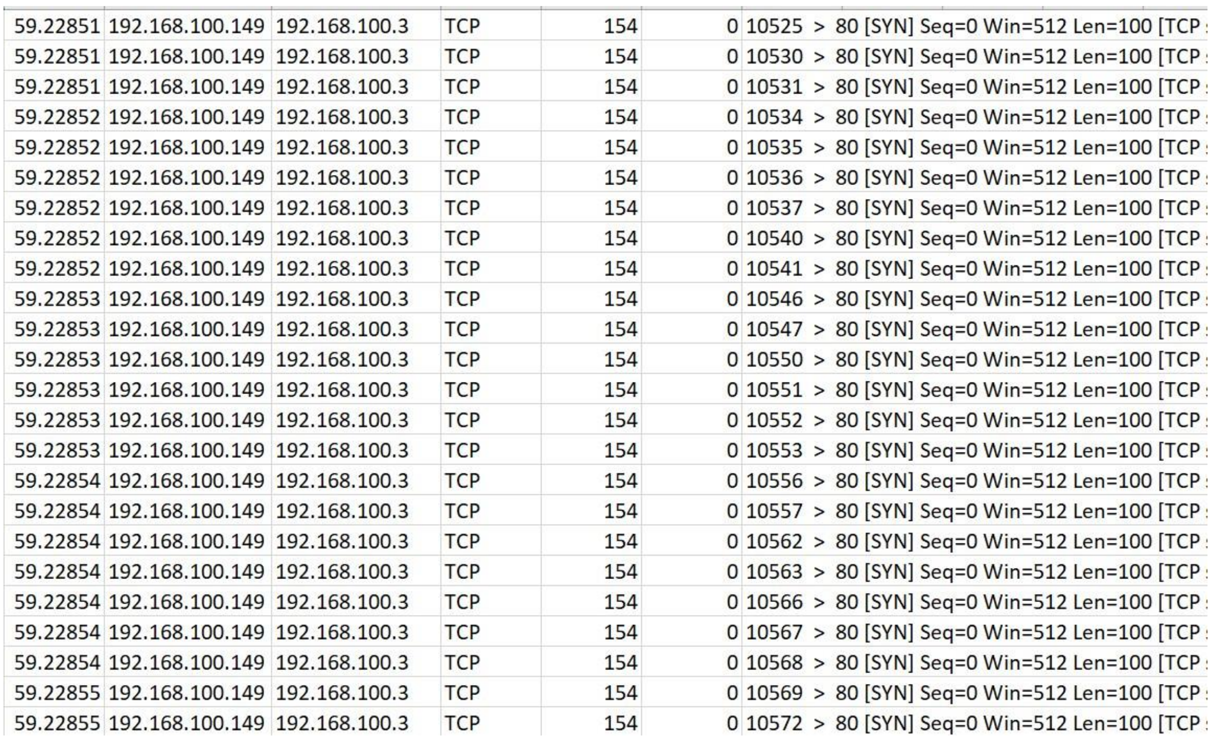
234.763539	192.168.1.198	156.2.206.170	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763542	192.168.1.198	156.213.134.69	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763544	192.168.1.198	197.25.194.39	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763547	192.168.1.198	41.250.93.208	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763549	192.168.1.198	197.249.120.60	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763552	192.168.1.198	197.38.63.146	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.76377	192.168.1.198	156.57.227.84	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763776	192.168.1.198	197.24.96.180	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763782	192.168.1.198	41.221.209.61	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763784	192.168.1.198	197.137.153.159	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763786	192.168.1.198	41.74.2.41	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763789	192.168.1.198	41.53.84.70	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763792	192.168.1.198	197.51.232.206	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763794	192.168.1.198	41.187.250.130	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763797	192.168.1.198	41.38.215.251	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.763799	192.168.1.198	156.162.217.241	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764026	192.168.1.198	41.195.190.140	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764029	192.168.1.198	197.49.155.126	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764032	192.168.1.198	197.143.149.60	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764034	192.168.1.198	156.86.168.133	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764037	192.168.1.198	41.240.197.50	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764042	192.168.1.198	41.180.116.83	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764045	192.168.1.198	41.15.92.125	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0
234.764047	192.168.1.198	156.140.61.142	TCP	60	0	24360 > 37215 [SYN] Seq=0 Win=14912 Len=0

Figure 4.44 TCPSYN Public dataset using Ostinato



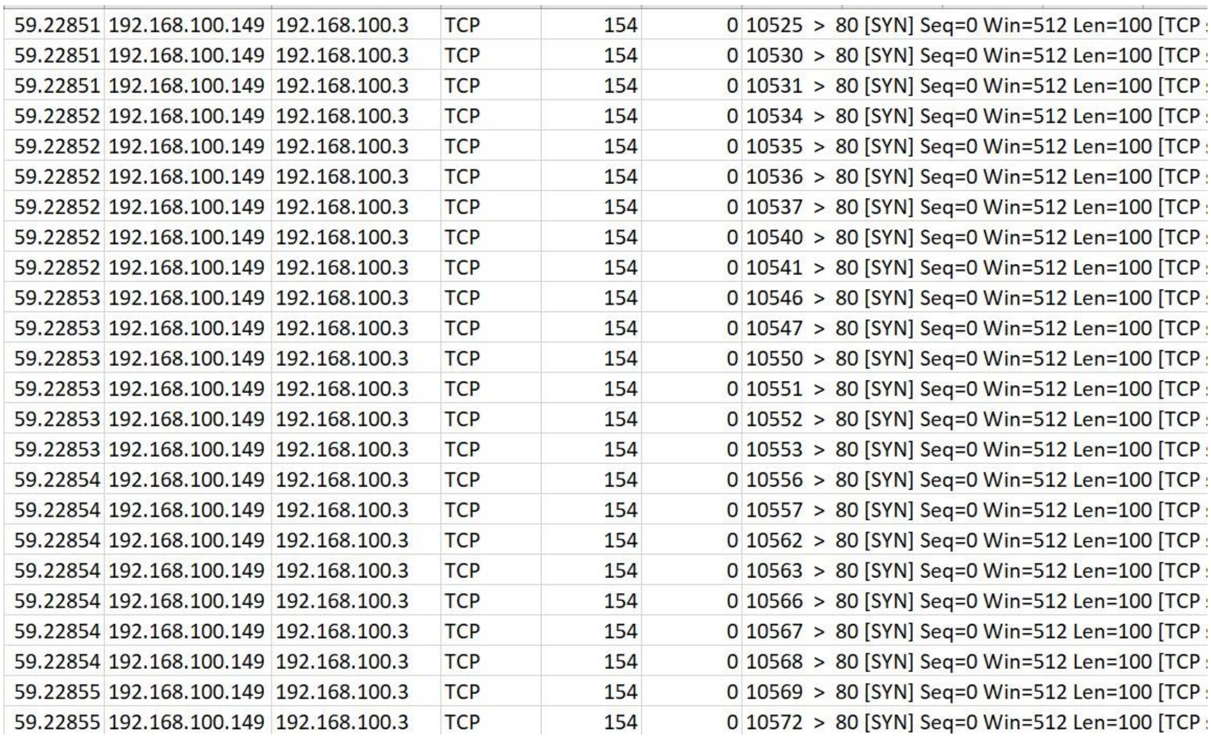
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10525 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10530 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10531 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10534 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10535 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10536 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10537 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10540 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10541 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10546 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10547 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10550 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10551 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10552 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10553 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10556 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10557 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10562 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10563 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10566 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10567 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10568 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10569 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10572 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP

Figure 4.45 TCP SYN public dataset using Ostinato



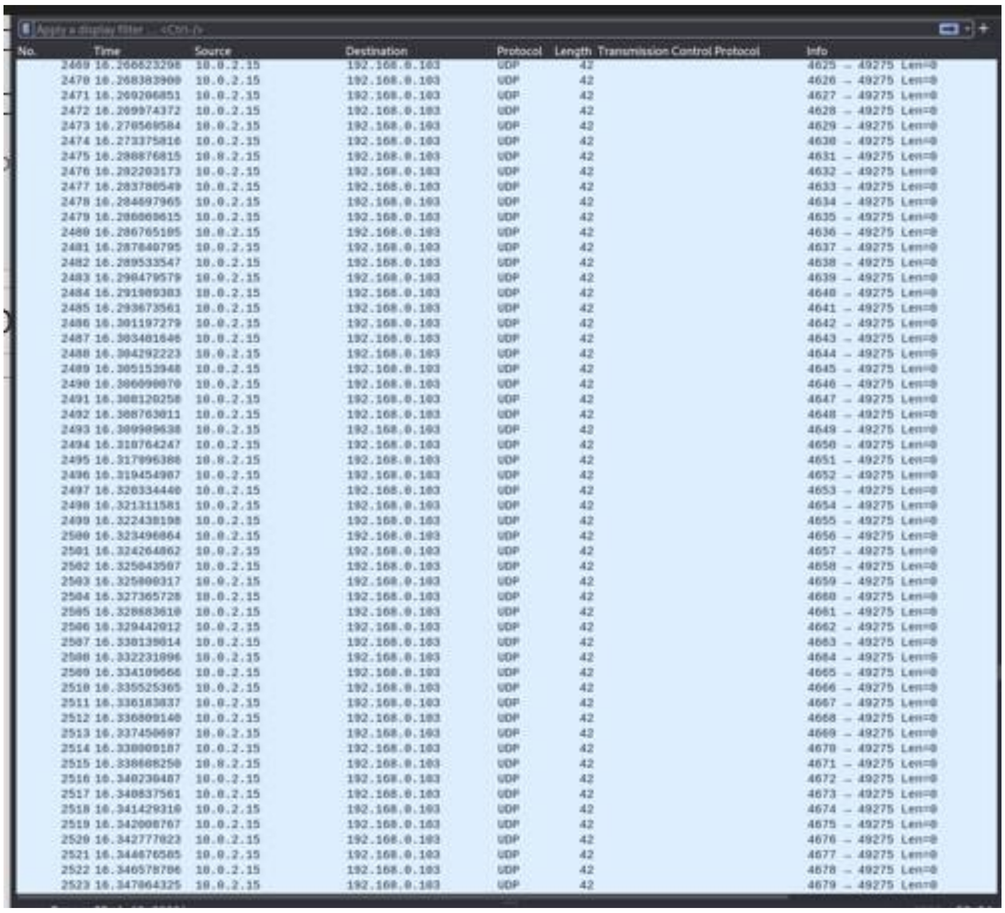
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10525 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10530 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10531 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10534 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10535 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10536 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10537 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10540 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10541 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10546 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10547 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10550 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10551 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10552 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10553 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10556 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10557 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10562 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10563 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10566 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10567 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10568 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10569 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10572 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP

Figure 4.46 TCP SYN public dataset using Ostinato



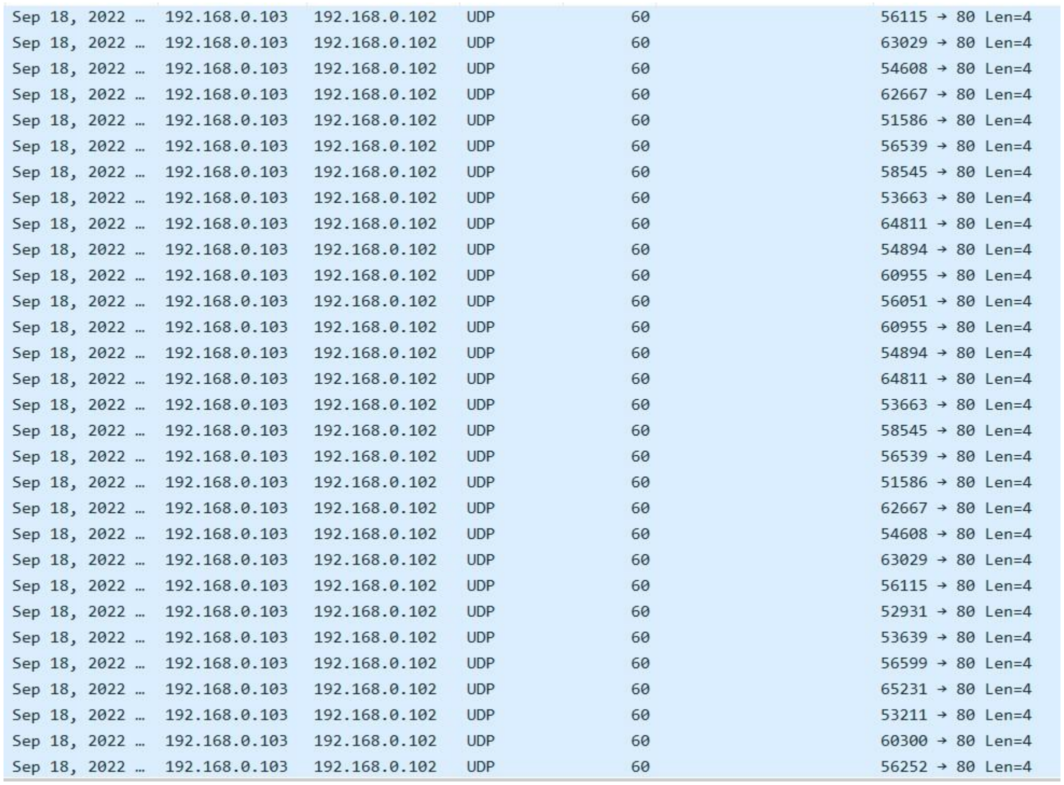
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10525 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10530 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22851	192.168.100.149	192.168.100.3	TCP	154	0	10531 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10534 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10535 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10536 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10537 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10540 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22852	192.168.100.149	192.168.100.3	TCP	154	0	10541 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10546 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10547 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10550 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10551 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10552 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22853	192.168.100.149	192.168.100.3	TCP	154	0	10553 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10556 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10557 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10562 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10563 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10566 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10567 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22854	192.168.100.149	192.168.100.3	TCP	154	0	10568 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10569 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP
59.22855	192.168.100.149	192.168.100.3	TCP	154	0	10572 > 80 [SYN] Seq=0 Win=512 Len=100 [TCP

Figure 4.47 TCP SYN public dataset using Ostinato



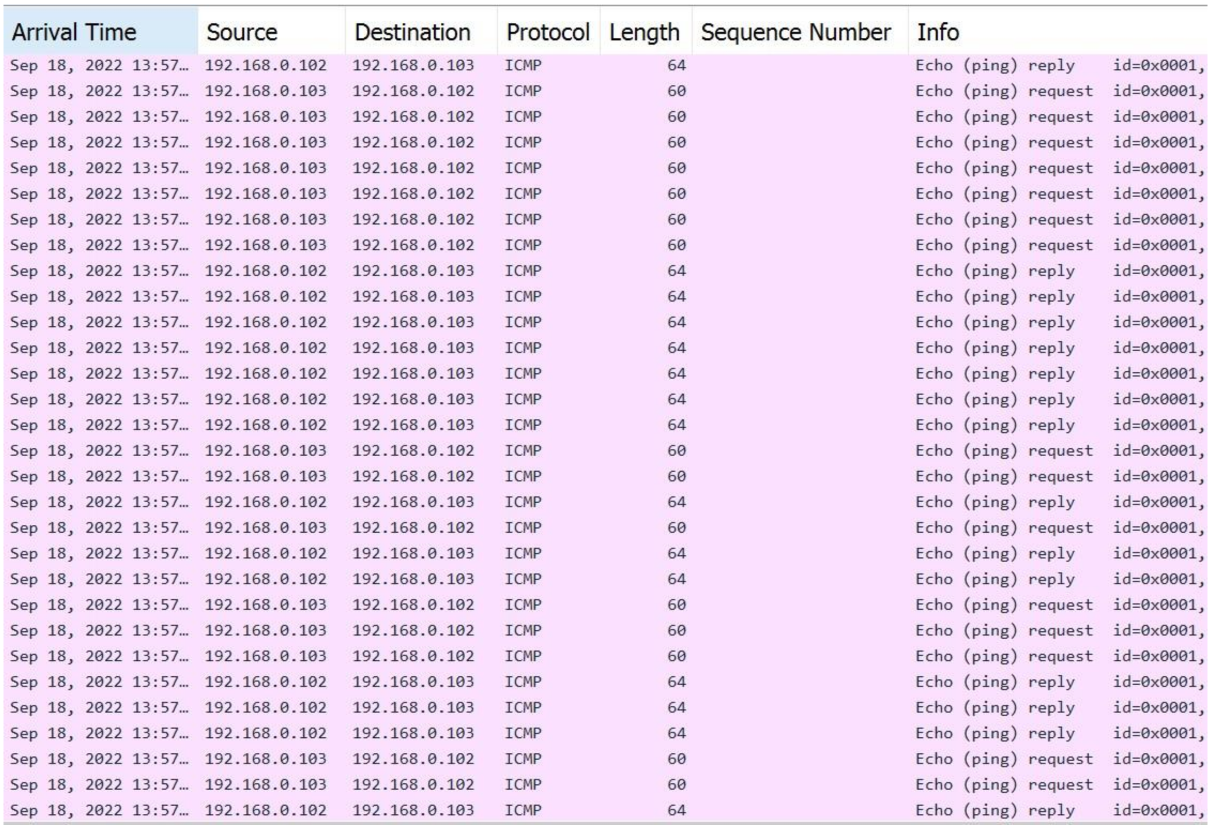
```
Apply a display filter ... <Ctrl-/>
No.    Time               Source       Destination      Protocol  Length  Transmission Control Protocol  Info
  2469 16.266623296  10.0.2.15    192.168.0.103    UDP       42                                      4625 → 49275 Len=0
  2470 16.268383900  10.0.2.15    192.168.0.103    UDP       42                                      4626 → 49275 Len=0
  2471 16.269206851  10.0.2.15    192.168.0.103    UDP       42                                      4627 → 49275 Len=0
  2472 16.269974372  10.0.2.15    192.168.0.103    UDP       42                                      4628 → 49275 Len=0
  2473 16.270569584  10.0.2.15    192.168.0.103    UDP       42                                      4629 → 49275 Len=0
  2474 16.273375816  10.0.2.15    192.168.0.103    UDP       42                                      4630 → 49275 Len=0
  2475 16.280876815  10.0.2.15    192.168.0.103    UDP       42                                      4631 → 49275 Len=0
  2476 16.282203173  10.0.2.15    192.168.0.103    UDP       42                                      4632 → 49275 Len=0
  2477 16.283780549  10.0.2.15    192.168.0.103    UDP       42                                      4633 → 49275 Len=0
  2478 16.284697965  10.0.2.15    192.168.0.103    UDP       42                                      4634 → 49275 Len=0
  2479 16.286069615  10.0.2.15    192.168.0.103    UDP       42                                      4635 → 49275 Len=0
  2480 16.286765105  10.0.2.15    192.168.0.103    UDP       42                                      4636 → 49275 Len=0
  2481 16.287840795  10.0.2.15    192.168.0.103    UDP       42                                      4637 → 49275 Len=0
  2482 16.289533547  10.0.2.15    192.168.0.103    UDP       42                                      4638 → 49275 Len=0
  2483 16.290479579  10.0.2.15    192.168.0.103    UDP       42                                      4639 → 49275 Len=0
  2484 16.291909383  10.0.2.15    192.168.0.103    UDP       42                                      4640 → 49275 Len=0
  2485 16.293673561  10.0.2.15    192.168.0.103    UDP       42                                      4641 → 49275 Len=0
  2486 16.301197279  10.0.2.15    192.168.0.103    UDP       42                                      4642 → 49275 Len=0
  2487 16.303401646  10.0.2.15    192.168.0.103    UDP       42                                      4643 → 49275 Len=0
  2488 16.304292223  10.0.2.15    192.168.0.103    UDP       42                                      4644 → 49275 Len=0
  2489 16.305153948  10.0.2.15    192.168.0.103    UDP       42                                      4645 → 49275 Len=0
  2490 16.306090070  10.0.2.15    192.168.0.103    UDP       42                                      4646 → 49275 Len=0
  2491 16.308120258  10.0.2.15    192.168.0.103    UDP       42                                      4647 → 49275 Len=0
  2492 16.308763011  10.0.2.15    192.168.0.103    UDP       42                                      4648 → 49275 Len=0
  2493 16.309909638  10.0.2.15    192.168.0.103    UDP       42                                      4649 → 49275 Len=0
  2494 16.310764247  10.0.2.15    192.168.0.103    UDP       42                                      4650 → 49275 Len=0
  2495 16.317096386  10.0.2.15    192.168.0.103    UDP       42                                      4651 → 49275 Len=0
  2496 16.319454907  10.0.2.15    192.168.0.103    UDP       42                                      4652 → 49275 Len=0
  2497 16.320334440  10.0.2.15    192.168.0.103    UDP       42                                      4653 → 49275 Len=0
  2498 16.321311581  10.0.2.15    192.168.0.103    UDP       42                                      4654 → 49275 Len=0
  2499 16.322438198  10.0.2.15    192.168.0.103    UDP       42                                      4655 → 49275 Len=0
  2500 16.323496064  10.0.2.15    192.168.0.103    UDP       42                                      4656 → 49275 Len=0
  2501 16.324264862  10.0.2.15    192.168.0.103    UDP       42                                      4657 → 49275 Len=0
  2502 16.325043507  10.0.2.15    192.168.0.103    UDP       42                                      4658 → 49275 Len=0
  2503 16.325800317  10.0.2.15    192.168.0.103    UDP       42                                      4659 → 49275 Len=0
  2504 16.327365728  10.0.2.15    192.168.0.103    UDP       42                                      4660 → 49275 Len=0
  2505 16.328683610  10.0.2.15    192.168.0.103    UDP       42                                      4661 → 49275 Len=0
  2506 16.329442012  10.0.2.15    192.168.0.103    UDP       42                                      4662 → 49275 Len=0
  2507 16.330139014  10.0.2.15    192.168.0.103    UDP       42                                      4663 → 49275 Len=0
  2508 16.332231096  10.0.2.15    192.168.0.103    UDP       42                                      4664 → 49275 Len=0
  2509 16.334109666  10.0.2.15    192.168.0.103    UDP       42                                      4665 → 49275 Len=0
  2510 16.335525365  10.0.2.15    192.168.0.103    UDP       42                                      4666 → 49275 Len=0
  2511 16.336183837  10.0.2.15    192.168.0.103    UDP       42                                      4667 → 49275 Len=0
  2512 16.336809140  10.0.2.15    192.168.0.103    UDP       42                                      4668 → 49275 Len=0
  2513 16.337450697  10.0.2.15    192.168.0.103    UDP       42                                      4669 → 49275 Len=0
  2514 16.338009187  10.0.2.15    192.168.0.103    UDP       42                                      4670 → 49275 Len=0
  2515 16.338608250  10.0.2.15    192.168.0.103    UDP       42                                      4671 → 49275 Len=0
  2516 16.340230487  10.0.2.15    192.168.0.103    UDP       42                                      4672 → 49275 Len=0
  2517 16.340837561  10.0.2.15    192.168.0.103    UDP       42                                      4673 → 49275 Len=0
  2518 16.341429310  10.0.2.15    192.168.0.103    UDP       42                                      4674 → 49275 Len=0
  2519 16.342008767  10.0.2.15    192.168.0.103    UDP       42                                      4675 → 49275 Len=0
  2520 16.342777023  10.0.2.15    192.168.0.103    UDP       42                                      4676 → 49275 Len=0
  2521 16.344676585  10.0.2.15    192.168.0.103    UDP       42                                      4677 → 49275 Len=0
  2522 16.346578706  10.0.2.15    192.168.0.103    UDP       42                                      4678 → 49275 Len=0
  2523 16.347064325  10.0.2.15    192.168.0.103    UDP       42                                      4679 → 49275 Len=0
```

Figure 4.48 UDP attack using Hping3



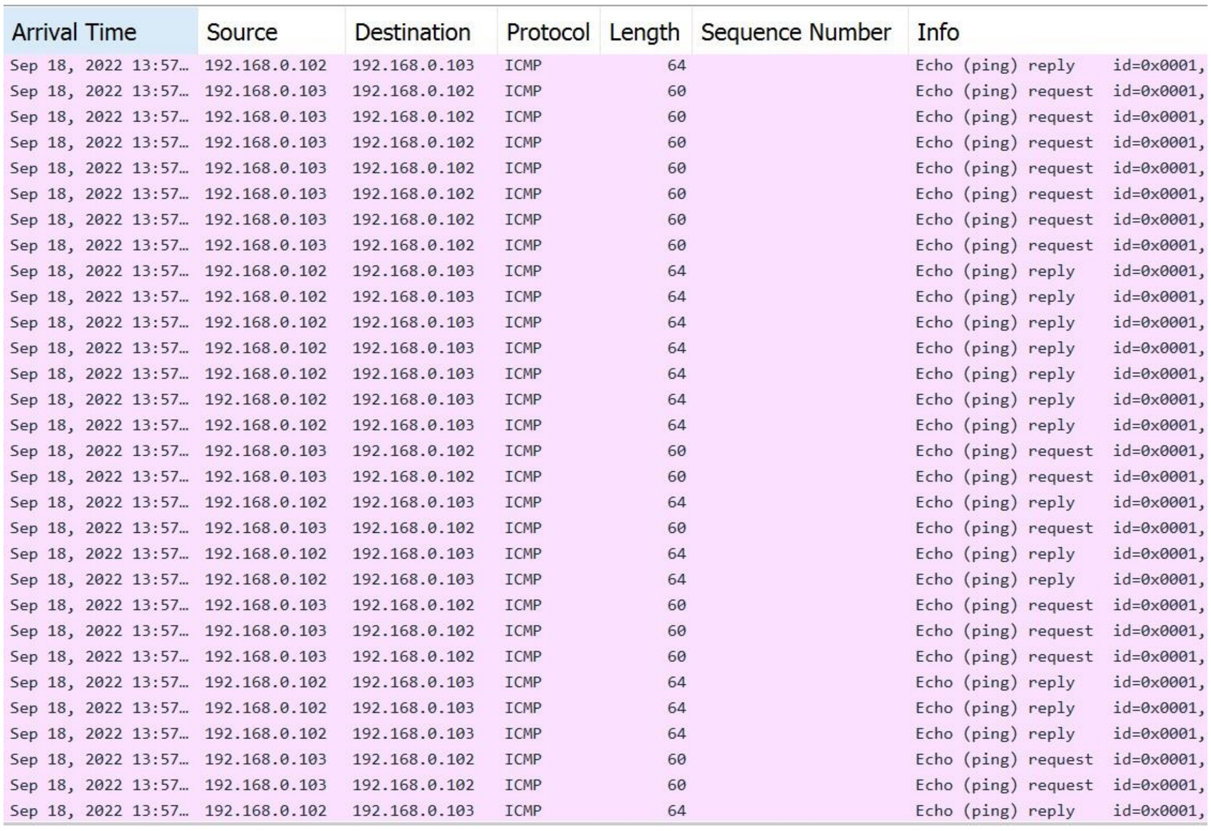
```
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56115 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    63029 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    54608 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    62667 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    51586 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56539 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    58545 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    53663 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    64811 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    54894 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    60955 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56051 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    60955 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    54894 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    64811 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    53663 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    58545 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56539 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    51586 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    62667 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    54608 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    63029 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56115 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    52931 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    53639 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56599 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    65231 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    53211 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    60300 → 80 Len=4
Sep 18, 2022 …  192.168.0.103  192.168.0.102  UDP    60    56252 → 80 Len=4
```

Figure 4.49 UDP attack using Hping3



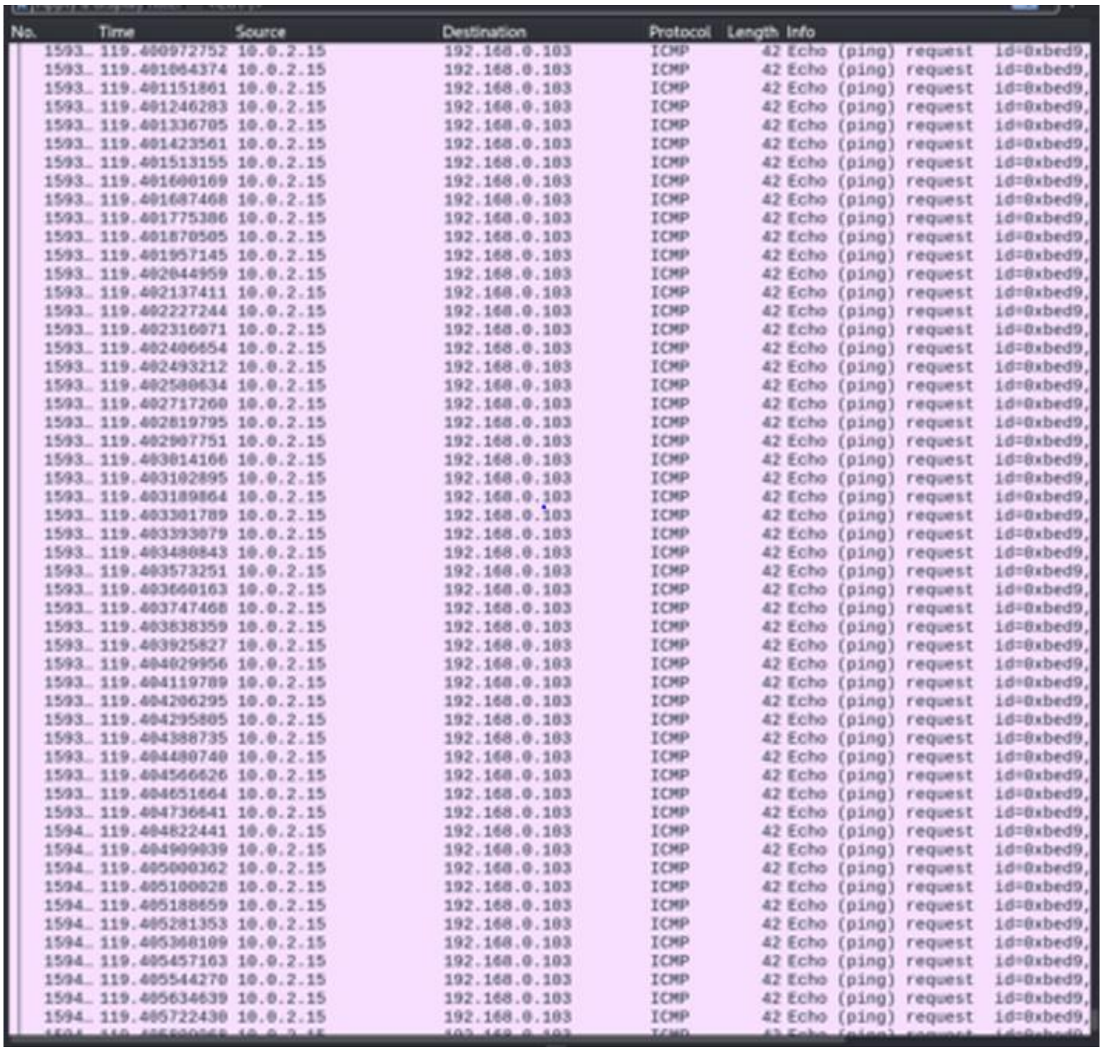
Arrival Time	Source	Destination	Protocol	Length	Sequence Number	Info
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,

Figure 4.50 ICMP attack using LOIC



Arrival Time	Source	Destination	Protocol	Length	Sequence Number	Info
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.103	192.168.0.102	ICMP	60		Echo (ping) request id=0x0001,
Sep 18, 2022 13:57…	192.168.0.102	192.168.0.103	ICMP	64		Echo (ping) reply id=0x0001,

Figure 4.51 ICMP attack using LOIC



```
No.      Time           Source          Destination      Protocol  Length Info
1593...  119.400972752  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401064374  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401151861  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401246283  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401336705  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401423561  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401513155  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401600169  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401687468  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401775386  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401870505  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.401957145  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402044959  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402137411  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402227244  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402316071  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402406654  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402493212  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402580634  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402717260  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402819795  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.402907751  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403014166  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403102895  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403189864  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403301789  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403393079  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403480843  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403573251  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403660163  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403747468  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403838359  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.403925827  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404029956  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404119789  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404206295  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404295805  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404388735  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404480740  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404566626  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404651664  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1593...  119.404736641  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.404822441  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.404909039  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405000362  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405100028  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405188659  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405281353  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405368109  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405457163  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405544270  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405634639  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
1594...  119.405722430  10.0.2.15       192.168.0.103    ICMP      42 Echo (ping) request  id=0xbed9,
```

Figure 4.52 ICMP attack using Hping3

4.5 Proposed novel detection method

The proposed detection mechanism takes into consideration three characteristics of IoT traffic as a benchmark for DDos traffic identification as derived from the EDA. These characteristics are often neglected when it comes to DDoS detection. They are:

- Feature randomness: From the EDA we can observe that certain network features like protocol, packet length and sequence numbers have a dynamic property. They tend to be randomized as opposed to the static nature they exhibit during a DDoS attack as seen from the attack EDA and Wireshark captures. For instance, when you take a window of 20 packets, you sometimes find the protocols changing after every three to four packets. The sequence number also varies in most cases for each packet having an incremental value. The packet length exhibits the same property of varying lengths every few packets. TCP flags also fall in this category as a normal TCP flow contains varying flags (SYN, SYN+ACK, ACK, FIN+ACK) as opposed to the single flag exhibited during attacks like TCPSYN flood. Lack of this dynamic pattern exhibited by these features in the smart home traffic flow should raise flags.

- ➢ Absent features: To identify or detect an attack, we should change the narrative from only focusing on the statistics or behaviour of present network features. Rather, features that are normally meant to be present but for some reason are absent for a prolonged period or certain threshold of packets should be a point of concern. For example, the normal network flow we have observed in the EDA tends to have a continuous flow of sequence numbers, as they are required for majority of the packets. However, during a UDP or ICMP attack, the sequence numbers are absent as these two protocols are not sequence number carriers naturally. A prolonged exhibition of this absence should raise a flag in a TCP based network or traffic. The same rule applies to missing encryption protocols in the traffic flow. For smart home devices that make use of encryption protocols like the TLSv.2 from the benign EDA, this protocol is persistent whenever a TCP connection is established. However, in the attacks within the scope of this research, this encryption protocol appears to be absent as attacks of this sort do not carry encryption protocols normally. As mentioned earlier, a complete established TCP flow carries several TCP flags. Having completely absent or some missing flags in a flow should raise concern as this is a common characteristic of flooding attacks. This shows that focusing on the missing features can also aid in identifying a malicious traffic flow. This should be an important avenue to consider for a more effective approach of attack detection.
- ➢ Feature range: Another important neglected point in DDoS identification is the range in which some features normally fall into. If we look at the sequence number range for the benign smart home traffic, we see the values are mostly double, triple, or quadruple digits with a few single digits of 0's and 1's at the beginning of a flow. This pattern contrasts with an attack flow within the scope of this research which carries only single digits of sequence numbers of just 0's or1's for all the packets as seen in the Wireshark captures provided.

After considering the points above, a DDoS traffic detection approach is presented. The network features considered in this proposed approach were found to be simultaneously affected during an attack. This means the packet lengths, sequence numbers, protocols and TCP flags all lost their variance at the very onset of an attack by exhibiting a static pattern as well as missing encryption protocols or sequence numbers in some cases. This new approach is composed of the following processes and sub processes to be implemented at the gateway of the network:

- ▪ Define target address: The target IP/MAC address of the device is set here.
- ▪ Specify flow direction: The desired flow direction of packets is specified. For instance, incoming/outgoing or bidirectional. However, for a lighter weight detection module with clearer variance and less redundancy during the flow, a single direction of flow is advised being incoming traffic to the specified device

- Set threshold/ time window: A cumulative sum of packets on which the subsequent checks will be carried out is set. This can be capped at every 20 packets or what best suits the type of network. Alternatively, a time window of some seconds or minutes can be used here for instance the n number of packets after every 60 seconds.
- Network feature checks: This is where feature presence, absence and variance are inspected, after which a decision is made of the pattern being malicious or not. This is done for the following network components:
- Protocol & Packet length: The protocol and packet length variance for the specified number of packets are checked. If the ratio of the various protocols and lengths present are balanced, then this is passed. However, if the ratio is found to be imbalanced like one protocol being present thorough out or the same value of length for all packets or majority of the packets, then is also logged. However, this is not objective as this must be tested first to confirm what ratios work.
- TCP Sequence number: The sequence number is first checked for presence or absence. If absent, this is logged. If present, variance, and range are checked. If the variance ratio is imbalanced or majority of packets have the same value of sequence number as are attacks within the scope of this research, this is logged. The range is also checked. If the values are all single digits of either 0's or 1's or both, this is also logged. Alternatively, if the range of values is incremental or maintains values with multiple digits, then this is passed. This is limited to TCP based network as mentioned in the scope of this research.
- Encryption & TCP flags: The presence or absence of encryption protocols and TCP flags are checked. If found absent for all packets, this is logged. On the other hand, if found present, this goes through a balance check. If found to be evenly distributed, this is passed. On the contrary if it is found to be highly uneven, this is logged. For example, with the SYN flag being predominant or completely dominant.

For a malicious pattern to be detected a minimum of 3 logged features must take place to avoid false positives. Moreover, the fact that these listed components get affected simultaneously during an attack makes it more likely for all components to be logged during an attack. For instance, a TCPSYN attack will log absence of encryption, imbalanced sequence number, imbalanced packet length, imbalanced protocol, and imbalanced TCP flags with the SYN flag predominant. On the other hand, a UDP or ICMP attack will log an absence of sequence numbers, absence of TCP flags, absence of encryption, imbalanced protocol, and an imbalanced packet length.

4.6 Comparison to literature and new contributions

The previous section has proposed an approach for DDoS flooding attack detection. This section moves to compare the proposed approach to literature based on several factors. It also highlights the contributions made in this chapter. It discusses how some aspects in literature have been improved and the novel contributions made.

As this the proposed approach is theoretical, which has not been implemented and tested, the factors used in this comparison are limited to technique, methodology and practicality among others rather than performance related factors. After implementation and testing performance related comparisons are made in the next chapter. The factors considered as a baseline for comparison here are five in number. These are as follows:

- Technique: The detection technique used like machine learning or algorithm based.
- Features: Network features used and biased features that could favour performance.
- Data analysis: The method used to analyse the traffic or dataset.
- Centric: Is the solution attack, user, or device centric?
- Practicality: How feasible and light weight is the solution. What are the down sides?

Table 4.2 shows this comparison citing each paper and evaluating it against the above listed factors.

Table 4.2 Comparison details

Paper	Technique	Features	Data analysis	Centric	Practicality	**Data source**	**Attacks covered**
[19]	Algorithm based. Learns sequential user behaviour to detect anomalies	Conditions (temperature, humidity, noise), User behaviours (operating devices like opening fridge, Tv On)	No comparison or analysis of data in normal and anomalous moments. Lack of visual dataset representations.	Solution tends to be user centric as sequential user behaviour is being learned prior to anomaly detection.	Having to learn each user behaviour does not seem practical in bigger environments. Having multiple users in the same place can also raise performance problems.	Real Iot devices	Anomalous behaviours or events
[20]	Rule based algorithm. Uses Device Usage Description (DUD) model for device behaviour and flow rules extraction to detect attacks.	Communication direction, Destination IP, port, protocol, interarrival time of packets, number of packets over time, packet sequence (7)	Lack of sufficient graphs. Only one visual showing number of exchanged packets. Presented what normal data looks like, however nothing about what attack data looks like or visual comparison.	Solution tends to be device and user centric. It's limited to Samsung Smartthings. Solution depends on extracting device behaviours which is also dependant on user behaviours in some cases.	Due to the need for having to extract and train or learn each devices pattern this is not practical in large scale environments. Training time too will be a hassle as each device will have to go through training for a period.	Private network.	UDP, SYN, ACK, HTTP, DNS Flood
[22]	Machine learning based DNS	DNS packets (1)	Data analysed and visualized showing	Solution is attack centric as	Solution attack centric.	Private network	DNS spoofing

	botnet detection.		differences in normal and attack data.	it only covers DNS attacks.			
[23]	Counter based DDoS attack detection using SDN.	number of flow entries, similar payload packet count, number of sent and received packets on each node, power ratio of each node, in/out traffic load and session IP counter	Attack and normal data patterns analysed visualizing differences	Attacks covered and their collection details not provided.	Due to the intricacy of processing these metrics, it results in high detection time and processing power.	Simulation using SDN.	DDoS (not specified)
[62]	Machine learning based DDoS detection using stateful and stateless network features.	Packet size, protocol, interarrival time, bandwidth, IP destination address (5)	Dataset used analysed visualizing differences in attack and normal data.	Only 3 attacks covered.	Public data has not been used for validation.		
[79]	Using Wireshark to identify TCPSYN attack	Request/reply packet count, SYN & ACK flag count, average packet per second, average packet size, average bytes of captured traffic, number of packets, time span (7)	Raw attack data Wireshark screen shots provided. However, no visual comparison with normal traffic.	TCPSYN attack centric as only this attack is analysed.	No detection module in place	Simulation	TCP SYN
[87]	Entropy-based DoS/DDoS detection system	source/ destination IP addresses coupled with their respective port numbers and protocols	Only entropy for attack data is visualized leaving out the normal data pattern.	Attacks covered limited to TCPSYN and UDP attacks.	If an attack starts at the very beginning of this window, then no entropy is calculated as no prior variation to compare with is present, thus the failure to detect the attack	Simulation using SDN	TCP, UDP
[100]	ML based DDoS detection through network traffic analysis.	Selected network features not mentioned.	Dataset used not analysed with no visual comparison between normal and attack patterns.	Attacks covered not mentioned.	Lack of sufficient data analysis and some important details missing like features selected and attacks covered.	Simulation using Hping3, Golden eye.	TCP SYN, UDP, HTTP
[104]	A supervised based IDS for anomaly detection.	121 network features.	Dataset used not analysed to show attack and benign patterns.	Covers reconnaissance, spoofing, replay, MITM and DoS/DDoS (TCP, UDP, Hello flood) attacks	Single packet inspection approach used which is time and resource consuming as dealing with each packet for feature extraction and classification takes significant time and processing power.	Real IoT devices. Kali used.	TCP, UDP, HELLO flood.
[126]	Algorithm based solution.	Data rate, packet length, average time	Data used has not been analysed or	No mention of dataset used or how it was	The proposed algorithm was not presented as well as	Simulation. Wireshark used.	DDoS (not specified)

	Applies a limit to network parameters which if exceeded flags an attack.	between requests and responses, port, protocol (6)	compared visually.	collected. Attacks covered also not mentioned.	the type of attacks covered in the paper which makes it difficult to reconstruct and validate.		
[127]	Machine learning based solution to detect botnets.	29 features of BoTIoT dataset. 8 features were selected to train model.	Statistical data provided showing difference in attack and benign data. However, no visualization provided	Device centric due to metrics used for detection.	Metrics used were limited to packet count over time, which might hinder the accuracy of detection	BoTIoT dataset. Python libraries used.	Botnet attacks.

From reviewed literature, there have been numerous contributions in terms of DDoS detection in smart home networks and IoT at large. Nevertheless, there are still gaps in the approach relating to better and improved methods of DDoS traffic identification. There is lack of detailed analysis and visual comparison of attack and benign traffic patterns. Some solutions make use of simulated data [108] [109] [110] [111] [112] [113], which might hinder the accuracy when deployed in real life scenarios. In addition, some of the approaches used are not very practical or feasible in some scenarios. For example, using the single packet inspection method to determine if it’s malicious or benign. This not only is time and resource consuming, but a less effective way of identifying DDoS patterns. This is due to DDoS flooding attacks being volume based, thus will need a volume based or cumulative approach to determine an attack pattern as opposed to the single packet approach. The approach of employing sequential user behaviour or Device Usage Description model (DUD) is not very practical as the former will raise false positives when there is slight change in user pattern while the latter is not practical in large scale scenarios as it’s a device centric solution and not a generalized one. There is also the issue of using biased network features in detection like IP addresses and port numbers which are device specific. This tends to favour the performance of the detection approach to the very scenario in question, thus making the solution user or device centric. Some detection approaches also use too many features which is time and resource consuming and leads to very significant overheads. This is due to the intricacy of processing these metrics as it results in high detection time and consumes a lot of processing power.

The network features used as a basis for detection gives this approach a generalized edge in terms of not being device, user or attack centric as seen in related works. This is due to all detection metrics being derived from general network characteristics shared by IoT devices. The various checks incorporated in this approach gives it the robust edge when dealing with the detection of several kinds of attacks. This is not only limited to the attacks covered here, but other attacks with similar propagation pattern to the ones covered like NTP, ARP and DNS flooding attacks.

The entropy or variance of the detection metrics is not calculated in real time based on the network statistics. Rather, a set of rules and conditions are used to identify an attack traffic. This will handle the failure in detection experienced by some approaches as seen in the related works when an attack starts at the very beginning of a set window. This is because no prior variance or entropy statistics are available to compare the current attacks entropy to. As a result, this approach does not rely on prior real time network statistics.

This approach is also light weight in terms of time and resource consumption as the flexibility to monitor a single flow direction (incoming traffic to target device) is possible. This results in clearer and less redundant variance statistics as opposed to a bidirectional flow (incoming/outgoing) which might hinder the clarity of the variance and carry redundant statistics like protocols, lengths, Flags, and sequence numbers.

This approach also eliminated the use of certain network features that can result in higher rates of false positives/negatives. For instance, the use of incoming/ outgoing distribution will raise a false negative for attacks like ICMP flooding. This is due to the nature of the attack traffic whereby majority of the echo requests have a paired reply packet from the target, thus making the incoming/ outgoing ratio evenly distributed. Same applies to the use of source IP addresses and port numbers in highly distributed attack scenarios where the source IP's and ports are spoofed. Moreover, IP addresses and ports tend to bring about biases especially in Machine Learning domains as they are specific to each scenario.

This approach can be incorporated into areas like *Data visualization tools, Intrusion Detection System (IDS), Software Defined Network (SDN) & Machine Learning* for better DDoS detection. The proposed network features used as detection metrics can be integrated into data visualisation tools and IDS. This will provide clearer low-level statistics as to how the network is deviating from its normal pattern during an attack. These features can also be defined as flow rules and conditions at the SDN gateway coupled with the appropriate mitigation measure in the case of a detected attack. The visualised EDA images can also be trained on a Convolutional Neural Network (CNN) using ResNet, as deep learning models especially CNN achieved high significance due to their outstanding performance in the image processing field.

4.7 Summary

This paper has carried out an EDA on collected smart home data, comparing the behavioral pattern of certain network features in a benign and DDoS attack flow. The same results have been derived for both privately collected data and public datasets. Based on the EDA results, a novel DDoS detection approach has been proposed which is neither user, attack nor device centric as it is applicable to IoT devices in general and a variety of DDoS flooding attacks. The detection model is based on the observed general network behavior of the smart home devices. These include feature variance, absent features and feature range which tend to be neglected in attack detection as observed from related works. The narrative needs to be changed from only focusing on present network feature statistics to detect attacks, rather features that are normally present but tend to be absent for a prolonged period also contribute to rapid attack detection as seen in this paper. However, this proposed approach relies heavily on the static nature of DDoS attack traffic and as such, low stealth DDoS attacks that exhibit a dynamic nature are not detected by this approach. The approach also detects a DDoS pattern at a certain threshold of packets which if not reached will fail to detect an attack. However, the overwhelming nature of the attack traffic always tends to go way above the threshold. As this detection method has not been implemented and tested, the next chapter covers this.

Chapter 5

A Novel Hybrid DDoS attack Detection and attack type indication system in the Smart Home Network

5.1 Introduction

The previous chapter proposed a novel approach to DDoS identification in the smart home traffic based on some observations derived from the Exploratory Data Analysis (EDA). This chapter implements a detection and attack type indication system based on the EDA observations which were majorly categorized into three conditions (Feature absence, Feature range, Feature randomness). The main goal is to have a detection and attack type indication solution that is light weight, practical, not attack, user nor device centric which works on several flooding attacks. In addition to that this solution is required to flag the covered attacks at the very onset to avoid the intended damage as the attack will be contained or thwarted in time. These conditions which the solution is based on are translated into an algorithm that will flag attack traffic in a pool of benign traffic and further indicate the type of attack. This solution is first tested on 3 flooding attacks (TCP SYN, UDP, ICMP). The performance of the system is evaluated, and improvements are made to the algorithm after which it is further tested on unfamiliar flooding attacks (HTTP, Slow LOIC, RECOIL [128]) and a mixture of all the attacks (TCP SYN, UDP, ICMP, HTTP, RECOIL, slow LOIC). This detection and attack type indication algorithm is also validated using public attack and benign data to eliminate biases and verify that it is not user, attack nor device centric. Finally, the implemented and tested detection and attack type indication system is compared to other state of the art solutions based on 12 critical factors. This chapter covers and achieves the following points:

- A better and more efficient methodology is designed and employed in terms of robustness, relevance, and clarity when it comes to the testing procedure as it redefines performance metrics using more relevant factors.
- Novel detection and attack type indication approaches are presented. It achieved better performance when compared to state of the art in terms of light weightiness, attacks covered, accuracy, practicality in terms of feasibility and scalability and not being attack, user nor device centric.
- The detection and attack type indication algorithms cover a broader surface in terms of attack variation. It works on known attacks, unfamiliar attacks, and mixed attacks. It is also validated successfully on Public normal and attack data.

The remaining sections in this chapter cover methodology used, how the novel detection and attack type indication system works, an implementation of the system, its performance and results achieved, a comparison of the systems performance to literature, new findings, and finally a summary of the chapter.

5.2 Methodology

The various processes and sub-processes followed to achieve a successful implementation of the proposed detection and attack type indication system is broken down in this section as shown in figure 5.1.

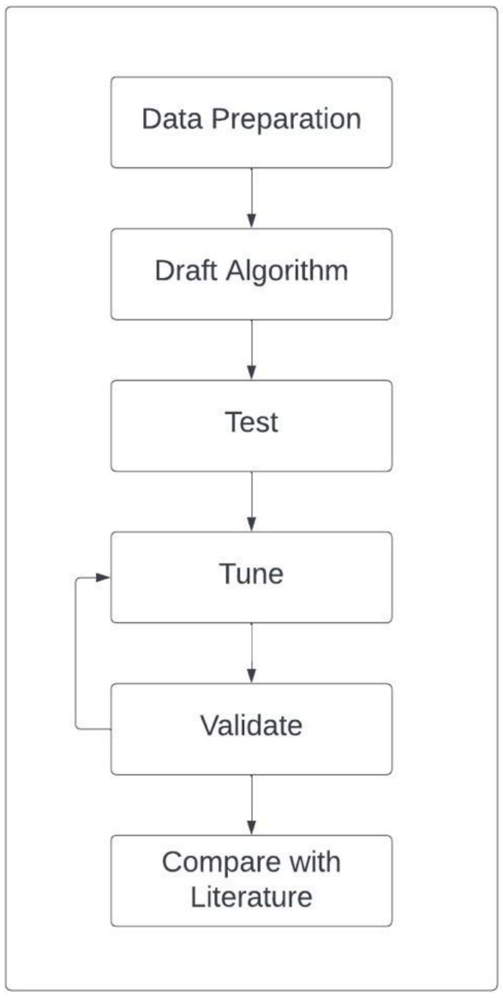


Figure 5.1 Methodology

This methodology consists of 6 main phases which are data preparation, algorithm scripting, testing, tuning, validation, and comparison to literature. The methodology is designed to be smooth and seamless in terms of transitioning from one phase to the next. It is manageable due to its segregated phases with having to complete one phase before moving onto the next as the results from the previous phase are used to commence working on the next phase. A recursive point is also included in the methodology comprising of testing, tuning and validation. This gives a smooth and clear process in terms of accommodating observations during testing that will be used in tuning the system for better performance. More attacks (unfamiliar attacks, mixed attacks, public data) are tested on the system at the validation phase for robustness and elimination of biases in terms of data source used. As the main goal is to have a solution that is light weight, practical, not user, device, nor attack centric, which works on unfamiliar and mixed attacks, this methodology tends to provide a plan to achieve each of these goals. A concise way to measure the systems performance in relation to how early the attack is flagged and robustness is also presented using more relevant metrics and factors. The methodology also lays out a rigorous comparison of the systems performance to other state of the art solutions based on several clear and critical factors. The factors based on which the systems performance is measured tends to be clearer and more scientifically sound which makes this methodology stand out when compared with state-of-the-art methods. The various phases and what they entail are outlined in the subsequent sections.

5.2.1 Data preparation

This phase is concerned with making sure the collected data is in the most suitable form for use in testing the proposed system. This involves the following steps:

- Data conversion: The collected raw data on table 4.1 in chapter 4 is converted from the Wireshark format (.pcap) to csv. This csv format is better in terms of compatibility with Google Colab that will be used as a platform to draft and test the algorithm.
- Filtering features: The relevant network features to be used are filtered. These features are source address, destination address, protocol, packet length, TCP sequence number and TCP flags. Although these features are for a TCP based traffic, using them does not completely deny the necessary UDP and ICMP traffic in the network as the algorithm is designed to accommodate them.
- Labelling: Each data source is properly labelled according to the category it falls in. These categories include the various attacks, whether it is a mixed or single attack, known or unfamiliar attack, public or private data and finally purely normal data or a mixture of normal and attack. This will help in giving a clear distinction of how the detection and attack type indication algorithm performs on each category.

5.2.2 Algorithm drafting

This phase delves into the steps taken to develop the detection and attack type indication algorithm in the most suitable environment. This includes the following:

- Language choice: A language that is easy to comprehend, light weight and compatible across several simulation platforms is considered for scripting the algorithm. The chosen language that fits all the considered qualities is python.
- Required conditions: Before scripting the algorithm, each considered network feature is translated into a condition that if met or breached, will result in a predefined decision. This will help in coupling up the various segments of the detection and attack type indication algorithm and the action taken after each condition.
- Window size: A specified number of packets to be inspected at a time is set here. This is intended to be set at the gateway. However, Google Colab is used to test this.
- IP address filter: The relevant IP addresses to be used are noted for each data source. This will be specified in the algorithm for each data source file to be tested. The flow direction considered is to the target device to avoid redundancy.
- Relevant protocols: Relevant protocols to be used in the conditions are noted from each data source. This will later be incorporated into the algorithm so the protocols from each data source are recognized during testing.
- Data source labels: The data source files, and column headers are labelled using a uniform naming convention so that each data source as well as the various columns are recognized without errors during testing.
- Result presentation: This involves coming up with how the detected and indicated attacks and normal traffic are flagged, labelled, and presented in the output file. The chosen method for this is to create 2 additional columns in the output file. One column will present traffic as either “attack “or “not attack” corresponding to each packet while the second column will present the attack type corresponding to each detected attack. Normal packets will carry the label “no type” in the classification column as they are already labelled “not attack” in the first column.
- Software choice: The preferred testing platform is Google Colab as it is open source, no downloads/installation needed, allows writing code and collaborating on it with team members and lastly being able to save directly to Git hub.
- Develop script: After fulfilling all the above-mentioned steps, a script is developed in python using Google Colab to test on the various data sources.
- Output location: The output location for each tested data source is set.

5.2.3 Testing

The testing phase is concerned with running several variations of the attacks, recording the performance, and making observations which will be used to improve the performance. The following steps are followed to achieve this:

- Test case metrics: Several factors used to distinguish between the various testcases, and the performance of each test case are outlined. This helps to give precise and scientifically sound results in relation to the performance of the algorithm. These factors include Dataset category (private/public), attack variation (single/mixed/unfamiliar), attack type (TCP, UDP, ICMP, HTTP), detection column label, indication column label, packet number attack started, packet number attack detected, packet number attack indicated.
- Note observations: Observations made relating to the performance of the algorithm are noted and these are used in the tuning phase to improve the weak points of the system.
- Unfamiliar attacks: These attacks (HTTP, SLOW LOIC, RECOIL) are tested on the algorithm to confirm if it precisely detects and indicate the attack type in unfamiliar attacks. Results from this is recorded in the test case table.
- Mixed attacks: These attacks are tested on the algorithm to confirm if it precisely detects and indicates the attack types in mixed attacks (a combination of several unfamiliar and known attacks). Results from this is recorded in the test case table.

5.2.4 Tuning

This phase requires making improvement to the detection and attack type indication algorithm based on the observations made during the testing phase. The steps in this phase include:

- Eliminate redundancy: Network features that pose redundant in the algorithm are removed to increase the light weightiness of the system. Only the best performing features in terms of detection trigger will be used.
- Alter number of consecutive packets to be flagged: Based on the performance of the algorithm on the various test cases in relation to number of consecutive packets to be flagged, this number is altered to the one which gives better results in terms of accuracy.
- False positive/negative reduction: One of the major aims of this system is to detect and indicate the attack type at the very onset (within the first 10-15 attack packets targeting the device) to minimize or totally avoid damage. Based on the performance results from the test cases, possible ways to improve the algorithm to achieve this precise detection and attack type indication at the onset or as close to the onset of the attack as possible are applied.

5.2.5 Validation

This phase is concerned with testing the detection and attack type indication algorithm on public data. This data is from a different environment and from a variety of smart home device brands. This is done to make the system bias free in terms of the data used. Furthermore, it will prove that the detection and attack type indication system is not user or device centric as it works on other smart home device brands from other users. The steps involved in this phase are:

- Normal data: Sourcing purely normal data.
- Normal + attack: Sourcing a mixture of normal and attack data.
- Record performance: Testing both publicly sourced data on the algorithm and recording its performance on the test case metric table. This will provide a clear result and conformation on whether the detection and attack type indication system are not user nor device centric.

5.2.6 Comparison to literature

This phase is about comparing the performance of the implemented and tested detection and attack type indication algorithm to state-of-the-art solutions. This will help to provide a clear-cut evaluation based on the performance of the implemented solution against current solutions. This comparison is based on 12 factors which are:

1) Features used: This refers to the network features used like protocol and port numbers and their total number. Are biased features used in attack detection? The traffic flow direction is also identified.
2) Attacks covered: The types and variations of attacks the solution covers.
3) Problem solved: What the solution solves like detection or classification or both.
4) Focus: Is the solution user, device or attack centric?
5) Practicality: Is the solution feasible in large scale environment, with reasonable resources and deployment time?
6) Data source: Is data source simulated or is it from a real smart home network? Is data composition realistic for use in testing?
7) Performance metrics: Are the metrics used to measure solution performance relevant?
8) Onset detection: Is the attack detected/ type indicated at the very onset?
9) Validation: How is the solution validated?
10) Counter spoof: Is the solution resistant to IP/port spoofing?
11) Approach: Relevance/downsides of approach to solving problem.
12) Coverage: Does the traffic inspection point cover all devices on the network?

5.3 How the detection and attack type indication algorithm work

The detection and attack type indication algorithm has two main functions which are to detect DDoS attacks and indicate the attack type in the infiltrated smart home traffic. It is built on 3 smart home traffic properties derived from the EDA in the previous chapter which are feature absence, feature range and feature variance. These properties are based on the general characteristics of smart home device traffic which the DDoS attack traffic violates. They are based on some smart home network features which were discovered to be simultaneously affected during a flooding attack. These affected features are protocol, packet length, sequence numbers, TCP flags and encryption. A raw data comparison of the state of these features in normal traffic to when attack sets in has been addressed in chapter 4 in figures 4.29 to 4.40. Each feature is categorized into one of the 3 properties due to its behavioural pattern during the attack flood. The features and properties are linked in the following ways:

- Feature absence: This refers to network features that are normally present in the smart home traffic but disappear during certain DDoS flooding attacks like ICMP, UDP and DNS among others. The affected network features that fall in this category are TCP flags, encryption protocols, and TCP sequence numbers. However, this is not tagging all ICMP, UDP or DNS traffic as malicious because the TCP based traffic also uses these at some point. These will be only tagged malicious if they go above a certain threshold.
- Feature range: This refers to network features that have numbers falling in wide ranges as they get incremental from one packet to the next in a short period of time but get limited to 0's/1's during an attack. The affected network feature in this category are the sequence numbers. This was discovered to be true for attacks like TCP SYN, HTTP and RECOIL among others. The sequence numbers that usually start from 0 or 1 and shoots up to thousands within seconds was found to be limited to just 0's or 1's or a combination of both for the packets during the duration of these attacks.
- Feature variance: This refers to the randomness or dynamics exhibited by some network features in normal smart home traffic but tend to get static during an attack. The features affected here are protocol, packet length, TCP flags and TCP sequence numbers. These features tend to have different compositions within a short period of time but become static to a single value during an attack like in TCP SYN, HTTP, UDP among others.

The above-mentioned properties were translated into conditions which if met or violated results in a specific action which in turn were embedded in an algorithm that detects and classifies the incoming attacks at the very onset. However, these conditions will be improved after the initial testing stage, thus removing some aspects of it. These conditions are as follows for each network feature for the detection aspect:

- **Condition A (Protocol)**: If 10 consecutive packets out of 20 have the same protocol, in combination with either sequence number or TCP flag triggers then flag all 20 packets as “Attack”. The TCP flag triggers are mentioned in condition D. A 20-packet window is chosen as it doubles the number of the consecutive packets needed to flag an attack. Moreover the 20-packet window is useful in the attack type indication stage where the highest protocol count is taken out of the 20 attack packets to label the attack type. This reduces the chances of a false negative due to window not being wide enough.
- **Condition B (Packet length)**: If 10 consecutive packets out of 20 have the same packet length in combination with TCP sequence number or TCP flag triggers, then flag all 20 packets as “Attack”.
- **Condition C (TCP Sequence numbers)**: If 10 consecutive packets out of 20 have absent TCP sequence numbers, then flag all 20 packets as “Attack”. If 10 consecutive packets out of 20 have their sequence numbers as only 0’s or 1’s or a combination of 0’s and 1’s, then flag all 20 packets as “Attack”. However, this does not mean all packets that do not carry these sequence numbers like UDP and ICMP will be automatically flagged as attack. The 10 consecutive packet rule takes care of this.
- **Condition D (TCP flags)**: If 10 consecutive packets out of 20 have absent TCP flags, then flag all 20 packets as “Attack”. If 10 consecutive packets out of 20 have their TCP flags set to SYN or RST or a combination of SYN and RST, then flag all 20 packets as “Attack”.
- **Condition E (Encryption)**: If encryption protocol (tlsv1.2) is missing for 10 consecutive packets out of 20 in combination with either TCP sequence number or TCP flag alarm, then flag all 20 packets as “Attack”. However, this depends on whether the devices use encryption protocols or not and the type of encryption used.

Attack type indication comes after the flagged packets have been labelled as “Attack”. The condition relating to classification is as follows:

- If 20 packets are flagged and labelled as “Attack” based on the aforementioned conditions, then focus on the protocol column and take the protocol with the highest count out of the 20 packets flagged as “Attack” and label all 20 packets as the protocol name like “UDP” or “TCP”. This provides the information on the protocol used in the attack.

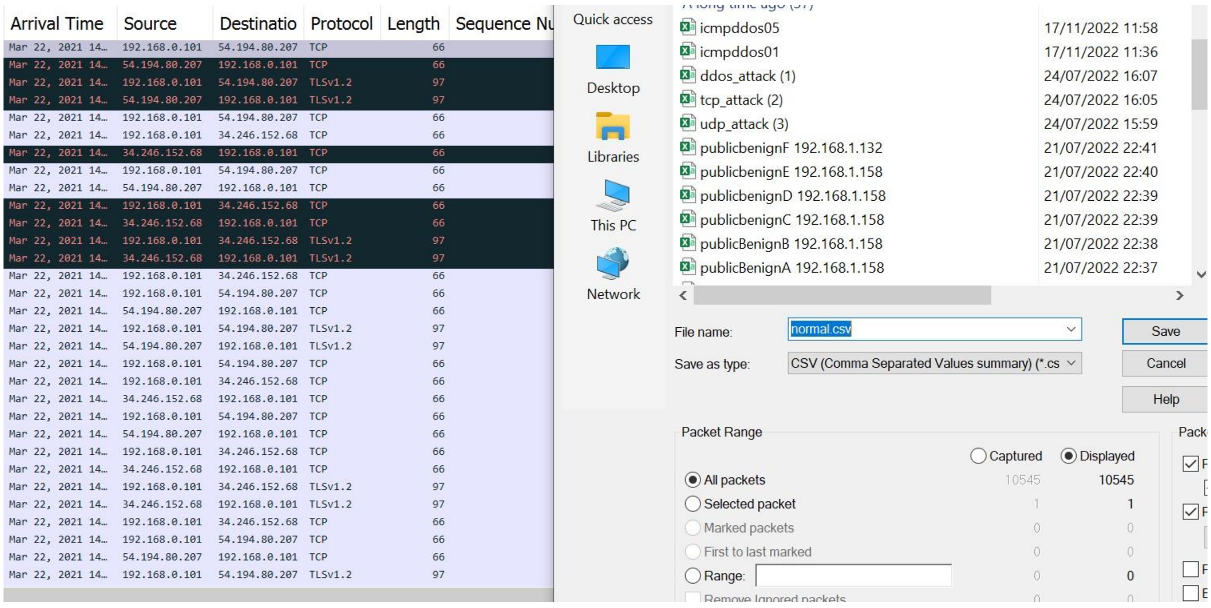
The network feature(s) in the conditions above that lead(s) to packets being labelled as “Attack” and type indicated using the highest protocol count from the attack labelled packets is subject to changes based on the performance of the feature(s) at the testing stage. This depends on the result of how well a condition independently or in combination with others flag an attack. An accurate truth table will be provided after testing confirming the most appropriate conditions.

5.4 Implementation

This section delves into the series of steps and processes carried out during the implementation of the detection and attack type indication system. Several phases of the previously presented methodology are laid out here with details of the actual work done to successfully achieve the overall intended system.

5.4.1 Data preparation

The dataset files were converted from the Wireshark format (.pcap) to csv. Figure 5.2 shows the process of this conversion. The pcap file is exported as csv in Wireshark and saved. Figure 5.3 shows the converted csv. The relevant network features to be tested are filtered. These features are source and destination IP addresses, protocol, packet length, sequence number and TCP flags. The required attack source and destination IP addresses are also isolated. Each data source is properly labelled according to the category it falls in. These categories include the various attacks, whether it is a mixed or single attack, known or unfamiliar attack, public or private data and finally purely normal data or a mixture of normal and attack. This will help in giving a clear distinction of how the detection and attack type indication algorithm performs on each category. Table 5.1 shows the details of the dataset labels with their file count, source and destination IP addresses and categories they fall into whether public/private or known/Unfamiliar.



Arrival Time	Source	Destinatio	Protocol	Length	Sequence Nu
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TLSv1.2	97	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TLSv1.2	97	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TLSv1.2	97	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TLSv1.2	97	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TLSv1.2	97	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TLSv1.2	97	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TLSv1.2	97	
Mar 22, 2021 14…	34.246.152.68	192.168.0.101	TLSv1.2	97	
Mar 22, 2021 14…	192.168.0.101	34.246.152.68	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TCP	66	
Mar 22, 2021 14…	54.194.80.207	192.168.0.101	TCP	66	
Mar 22, 2021 14…	192.168.0.101	54.194.80.207	TLSv1.2	97	

Quick access

Desktop

Libraries

This PC

Network

icmpddos05	17/11/2022 11:58
icmpddos01	17/11/2022 11:36
ddos_attack (1)	24/07/2022 16:07
tcp_attack (2)	24/07/2022 16:05
udp_attack (3)	24/07/2022 15:59
publicbenignF 192.168.1.132	21/07/2022 22:41
publicbenignE 192.168.1.158	21/07/2022 22:40
publicbenignD 192.168.1.158	21/07/2022 22:39
publicbenignC 192.168.1.158	21/07/2022 22:39
publicBenignB 192.168.1.158	21/07/2022 22:38
publicBenignA 192.168.1.158	21/07/2022 22:37

File name: normal.csv

Save as type: CSV (Comma Separated Values summary) (*.cs

Save

Cancel

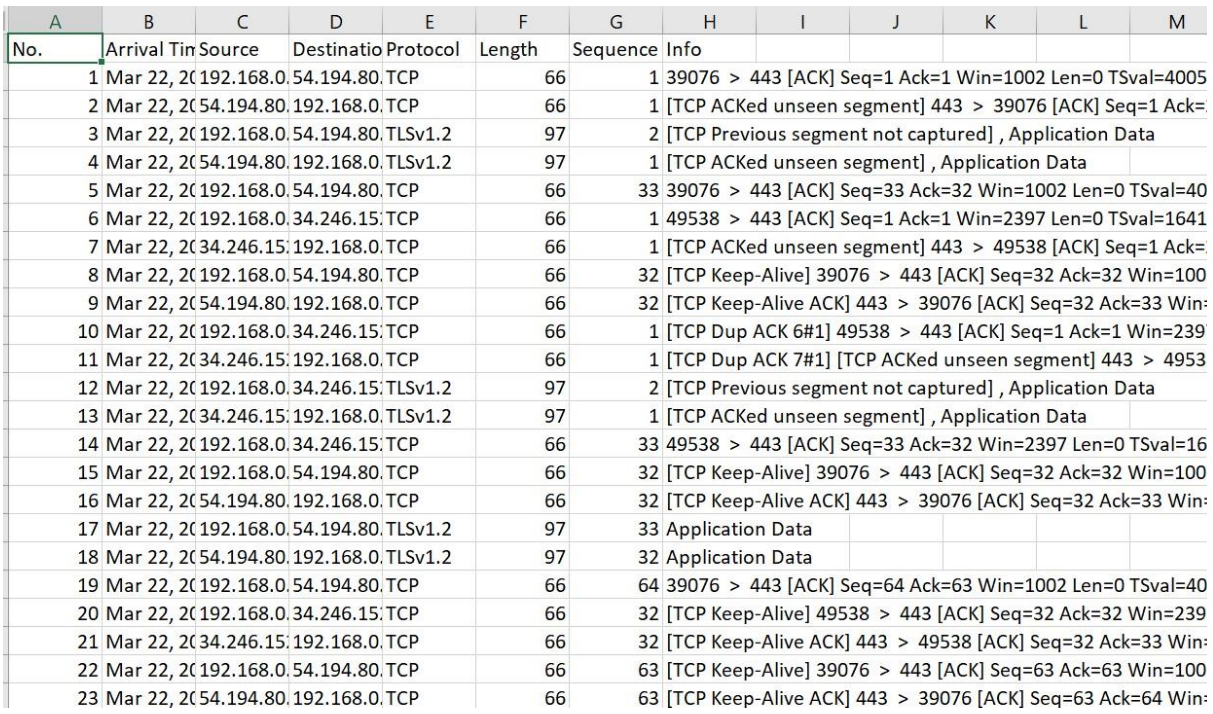
Help

Packet Range

	Captured	[✓ Displayed]
[✓ All packets]	10545	10545
Selected packet	1	1
Marked packets	0	0
First to last marked	0	0
Range:	0	0
Remove Ignored packets	0	0

Pack

Figure 5.2 CSV conversion



No.	Arrival Tim	Source	Destinatio	Protocol	Length	Sequence	Info
1	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	1	39076 > 443 [ACK] Seq=1 Ack=1 Win=1002 Len=0 TSval=4005
2	Mar 22, 20	54.194.80.	192.168.0.	TCP	66	1	[TCP ACKed unseen segment] 443 > 39076 [ACK] Seq=1 Ack=
3	Mar 22, 20	192.168.0.	54.194.80.	TLSv1.2	97	2	[TCP Previous segment not captured] , Application Data
4	Mar 22, 20	54.194.80.	192.168.0.	TLSv1.2	97	1	[TCP ACKed unseen segment] , Application Data
5	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	33	39076 > 443 [ACK] Seq=33 Ack=32 Win=1002 Len=0 TSval=40
6	Mar 22, 20	192.168.0.	34.246.15	TCP	66	1	49538 > 443 [ACK] Seq=1 Ack=1 Win=2397 Len=0 TSval=1641
7	Mar 22, 20	34.246.15	192.168.0.	TCP	66	1	[TCP ACKed unseen segment] 443 > 49538 [ACK] Seq=1 Ack=
8	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	32	[TCP Keep-Alive] 39076 > 443 [ACK] Seq=32 Ack=32 Win=100
9	Mar 22, 20	54.194.80.	192.168.0.	TCP	66	32	[TCP Keep-Alive ACK] 443 > 39076 [ACK] Seq=32 Ack=33 Win=
10	Mar 22, 20	192.168.0.	34.246.15	TCP	66	1	[TCP Dup ACK 6#1] 49538 > 443 [ACK] Seq=1 Ack=1 Win=239
11	Mar 22, 20	34.246.15	192.168.0.	TCP	66	1	[TCP Dup ACK 7#1] [TCP ACKed unseen segment] 443 > 4953
12	Mar 22, 20	192.168.0.	34.246.15	TLSv1.2	97	2	[TCP Previous segment not captured] , Application Data
13	Mar 22, 20	34.246.15	192.168.0.	TLSv1.2	97	1	[TCP ACKed unseen segment] , Application Data
14	Mar 22, 20	192.168.0.	34.246.15	TCP	66	33	49538 > 443 [ACK] Seq=33 Ack=32 Win=2397 Len=0 TSval=16
15	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	32	[TCP Keep-Alive] 39076 > 443 [ACK] Seq=32 Ack=32 Win=100
16	Mar 22, 20	54.194.80.	192.168.0.	TCP	66	32	[TCP Keep-Alive ACK] 443 > 39076 [ACK] Seq=32 Ack=33 Win=
17	Mar 22, 20	192.168.0.	54.194.80.	TLSv1.2	97	33	Application Data
18	Mar 22, 20	54.194.80.	192.168.0.	TLSv1.2	97	32	Application Data
19	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	64	39076 > 443 [ACK] Seq=64 Ack=63 Win=1002 Len=0 TSval=40
20	Mar 22, 20	192.168.0.	34.246.15	TCP	66	32	[TCP Keep-Alive] 49538 > 443 [ACK] Seq=32 Ack=32 Win=239
21	Mar 22, 20	34.246.15	192.168.0.	TCP	66	32	[TCP Keep-Alive ACK] 443 > 49538 [ACK] Seq=32 Ack=33 Win=
22	Mar 22, 20	192.168.0.	54.194.80.	TCP	66	63	[TCP Keep-Alive] 39076 > 443 [ACK] Seq=63 Ack=63 Win=100
23	Mar 22, 20	54.194.80.	192.168.0.	TCP	66	63	[TCP Keep-Alive ACK] 443 > 39076 [ACK] Seq=63 Ack=64 Win=

Figure 5.3 Converted pcap to csv

Table 5.1 Dataset categorization details

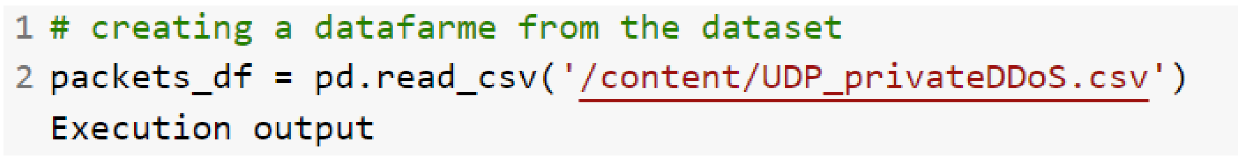
ID	Dataset	Source IP	Destination IP	Public/Private	Known/Unfamiliar	Composition	Protocol
1	TCPSYN (5 files)	192.168.0.103	192.168.0.102	Private	Known	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP
2	UDP (5 files)	192.168.0.103	192.168.0.102	Private	Known	Attack+benign	UDP,TCP,DNS,ICMP,TLSv1.2, NTP
3	ICMP (5 files)	192.168.0.103	192.168.0.102	Private	Known	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP
4	HTTP (3 files)	192.168.0.103	192.168.0.102	Private	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP,HTTP
5	SLOWLOIC (2 files)	192.168.0.103	192.168.0.102	Private	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP
6	RECOIL (2 files)	192.168.0.103	192.168.0.102	Private	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP
7	Mixed (3 files)	192.168.0.103	192.168.0.102	Private	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,NTP,HTTP,UDP
8	Normal (5 files)		192.168.0.101	Private		Benign	TCP,DNS,ICMP,TLSv1.2,NTP, MDNS,ARP
9	TCPSYN (2 files)	192.168.100.147-150	192.168.100.3	Public	Known	Attack+benign	TCP,DNS,ICMP,TLSv1.2,
10	UDP (2 files)	192.168.100.147-150	192.168.100.3	Public	Known	Attack+benign	TCP,DNS,ICMP,TLSv1.2

11	HTTP (2 files)	192.168.100.147-150	192.168.100.3	Public	Unfamiliar	Attack+benign	TCP,DNS,ICMP,SSH,HTTP
12	FRAGMENTED (2 files)	192.168.1.195	74.91.117.248	Public	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,UDP
13	Mixed (2 files)	192.168.100.147-150	192.168.100.3	Public	Unfamiliar	Attack+benign	TCP,DNS,ICMP,TLSv1.2,UDP
14	Normal (5 files)		192.168.1.158	Public		Benign	TCP,DNS,ICMP,SSH,NTP,HTTP,ARP
	Normal (1 file)		192.168.1.132	Public		Benign	TCP,DNS,ICMP,SSH,NTP,HTTP,ARP

5.4.2 Algorithm Drafting

The listed conditions from section 5.3 are translated into a python script. The window size is fixed to 20. The number of consecutive packets required to flag an attack pattern is tested by setting it to 5 and 10 packets respectively. Both will be tested to see how they perform. The relevant destination IP addresses for each dataset will be incorporated into the script as well as the correct dataset label so it can be identified as the data source. The flow direction is set to only focus on incoming packets to the target smart home devices. This means a one-way traffic direction will be monitored which contributes to the solutions light weight nature. The encryption protocol present in the dataset is also defined in the script so it can identify the absence of such protocols. Pandas and NumPy are the libraries imported at the beginning of the script as some functions from them will be utilized.

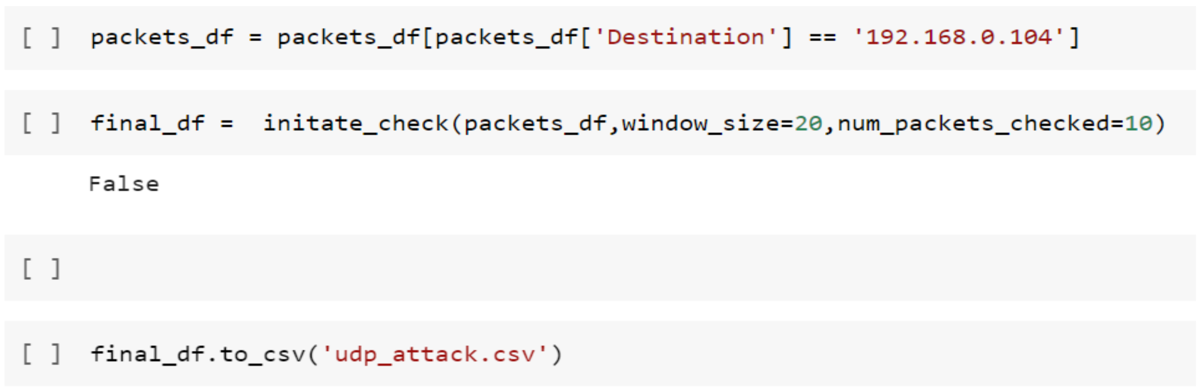
Figure 5.4 shows the code responsible for creating the data frame with the respective relevant columns. This is read from the dataset file it is pointed to.



```
# creating a datafarme from the dataset
packets_df = pd.read_csv('/content/UDP_privateDDoS.csv')
Execution output
```

Figure 5.4 Data frame creation

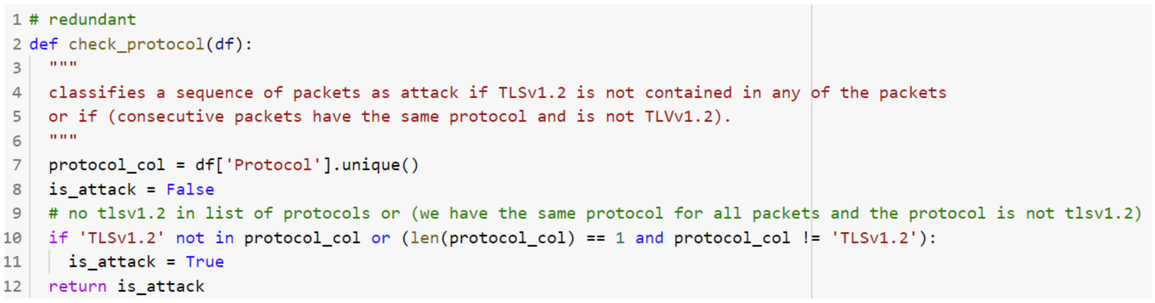
Figure 5.5 is where the destination IP is set, which is that of the smart home device. The “window size” refers to the number of packets checked at a time. The “number of packets checked” is what determines how many consecutive packets make an attack pattern if the conditions in figures 5.6 to 5.9 are true. The output file path is also set here.



```
[ ]  packets_df = packets_df[packets_df['Destination'] == '192.168.0.104']

[ ]  final_df =  initate_check(packets_df,window_size=20,num_packets_checked=10)

     False

[ ]

[ ]  final_df.to_csv('udp_attack.csv')
```

Figure 5.5 Setting IP, window size, packets checked and output path

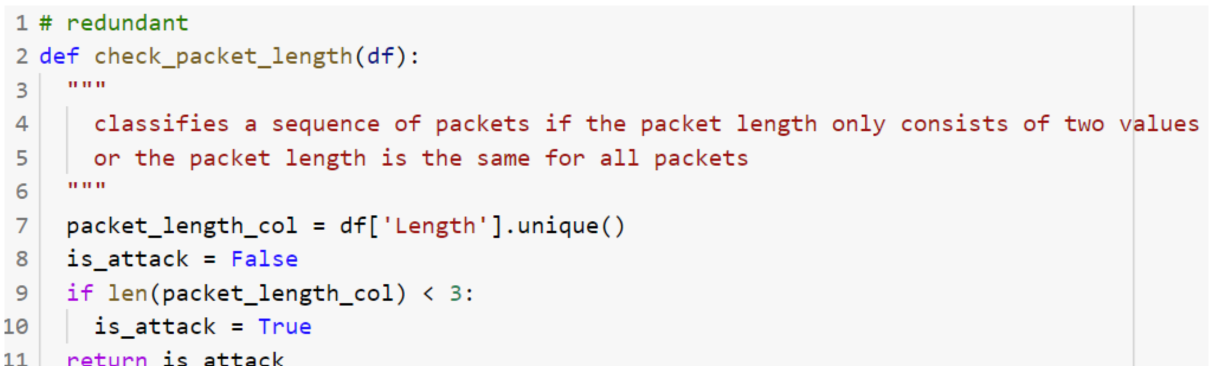
Figure 5.6 shows the code in charge of checking the protocol conditions. The number of unique elements present in the protocol column are checked for. If any of those unique elements appear consecutively for ten or more packets at a go, then this is flagged as attack. Secondly, if the encryption protocol TLSv1.2 is missing for ten consecutive packets, then the 10 packets are flagged as attack.



```
# redundant
def check_protocol(df):
  """
  classifies a sequence of packets as attack if TLSv1.2 is not contained in any of the packets
  or if (consecutive packets have the same protocol and is not TLVv1.2).
  """
  protocol_col = df['Protocol'].unique()
  is_attack = False
  # no tlsv1.2 in list of protocols or (we have the same protocol for all packets and the protocol is not tlsv1.2)
  if 'TLSv1.2' not in protocol_col or (len(protocol_col) == 1 and protocol_col != 'TLSv1.2'):
    is_attack = True
  return is_attack
```

Figure 5.6 Protocol condition check

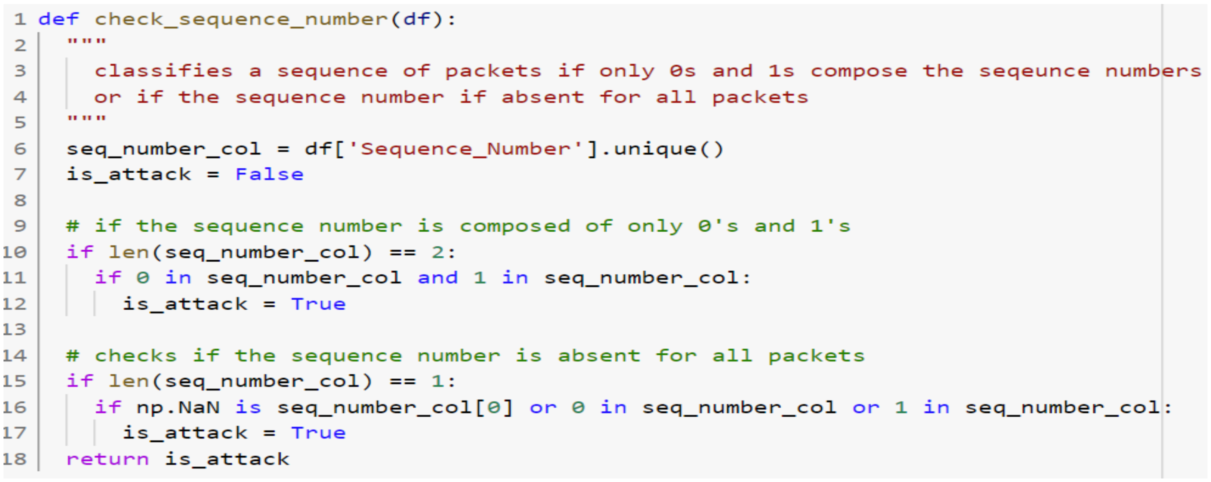
Figure 5.7 shows the code responsible for checking the packet length conditions. It checks the number of unique elements in the packet length data frame. If only one or two unique elements are found for 10 consecutive packets, then these 10 packets are flagged as attack.



```
# redundant
def check_packet_length(df):
  """
    classifies a sequence of packets if the packet length only consists of two values
    or the packet length is the same for all packets
  """
  packet_length_col = df['Length'].unique()
  is_attack = False
  if len(packet_length_col) < 3:
    is_attack = True
  return is_attack
```

Figure 5.7 Packet length condition check

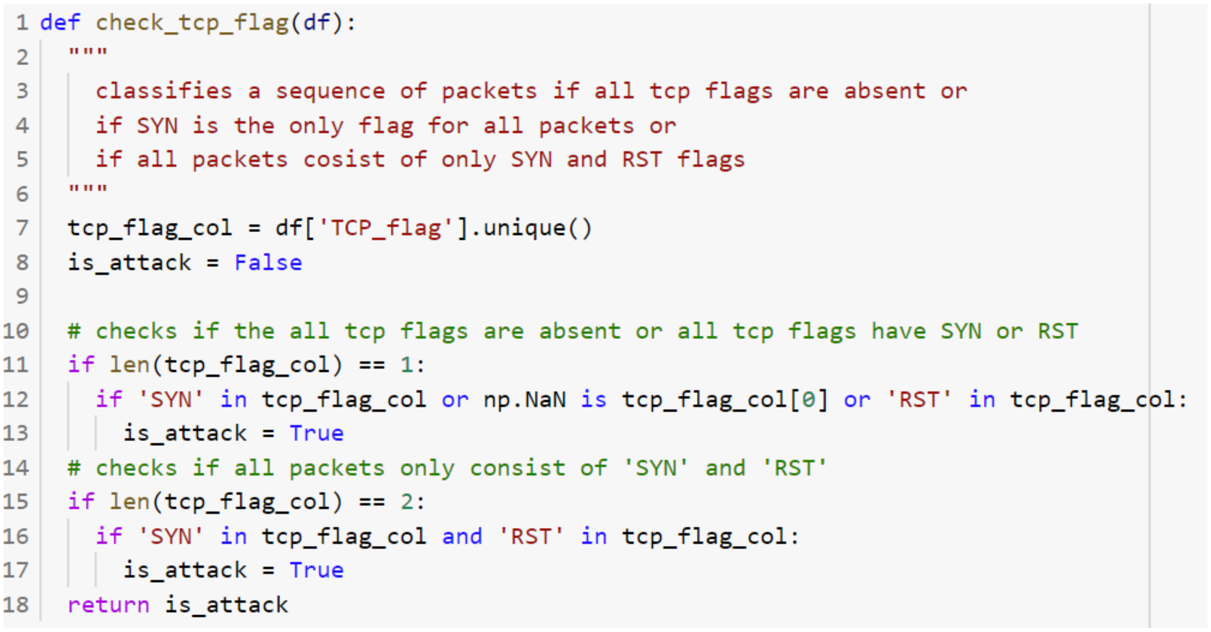
Figure 5.8 shows the code responsible for checking the sequence number conditions. If the sequence number column of the data frame is found to have only 0's or 1's or a combination of both for 10 consecutive packets, then these 10 packets are flagged as attack, otherwise normal. Secondly, if 10 consecutive packets have absented or null sequence numbers, these are also flagged as attack, otherwise normal.



```python
def check_sequence_number(df):
    """
      classifies a sequence of packets if only 0s and 1s compose the seqeunce numbers
      or if the sequence number if absent for all packets
    """
    seq_number_col = df['Sequence_Number'].unique()
    is_attack = False

    # if the sequence number is composed of only 0's and 1's
    if len(seq_number_col) == 2:
      if 0 in seq_number_col and 1 in seq_number_col:
        is_attack = True

    # checks if the sequence number is absent for all packets
    if len(seq_number_col) == 1:
      if np.NaN is seq_number_col[0] or 0 in seq_number_col or 1 in seq_number_col:
        is_attack = True
    return is_attack
```

Figure 5.8 Sequence number condition check

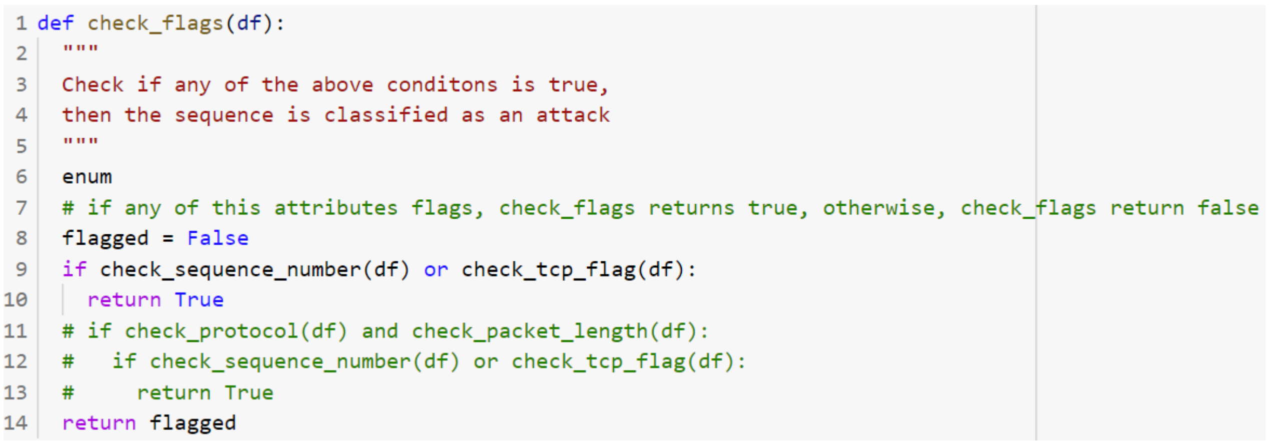
Figure 5.9 shows the code responsible for checking the TCP flag conditions. This checks for the number of unique elements in the TCP flag column data frame. It further checks if only SYN or RST flags or a combination of both are found for 10 consecutive packets then these are flagged as attack, otherwise normal. If TCP flags are found to be absent for ten consecutive packets, these are also flagged as attack.



```python
def check_tcp_flag(df):
    """
      classifies a sequence of packets if all tcp flags are absent or
      if SYN is the only flag for all packets or
      if all packets cosist of only SYN and RST flags
    """
    tcp_flag_col = df['TCP_flag'].unique()
    is_attack = False

    # checks if the all tcp flags are absent or all tcp flags have SYN or RST
    if len(tcp_flag_col) == 1:
      if 'SYN' in tcp_flag_col or np.NaN is tcp_flag_col[0] or 'RST' in tcp_flag_col:
        is_attack = True
    # checks if all packets only consist of 'SYN' and 'RST'
    if len(tcp_flag_col) == 2:
      if 'SYN' in tcp_flag_col and 'RST' in tcp_flag_col:
        is_attack = True
    return is_attack
```

Figure 5.9 TCP flag condition check

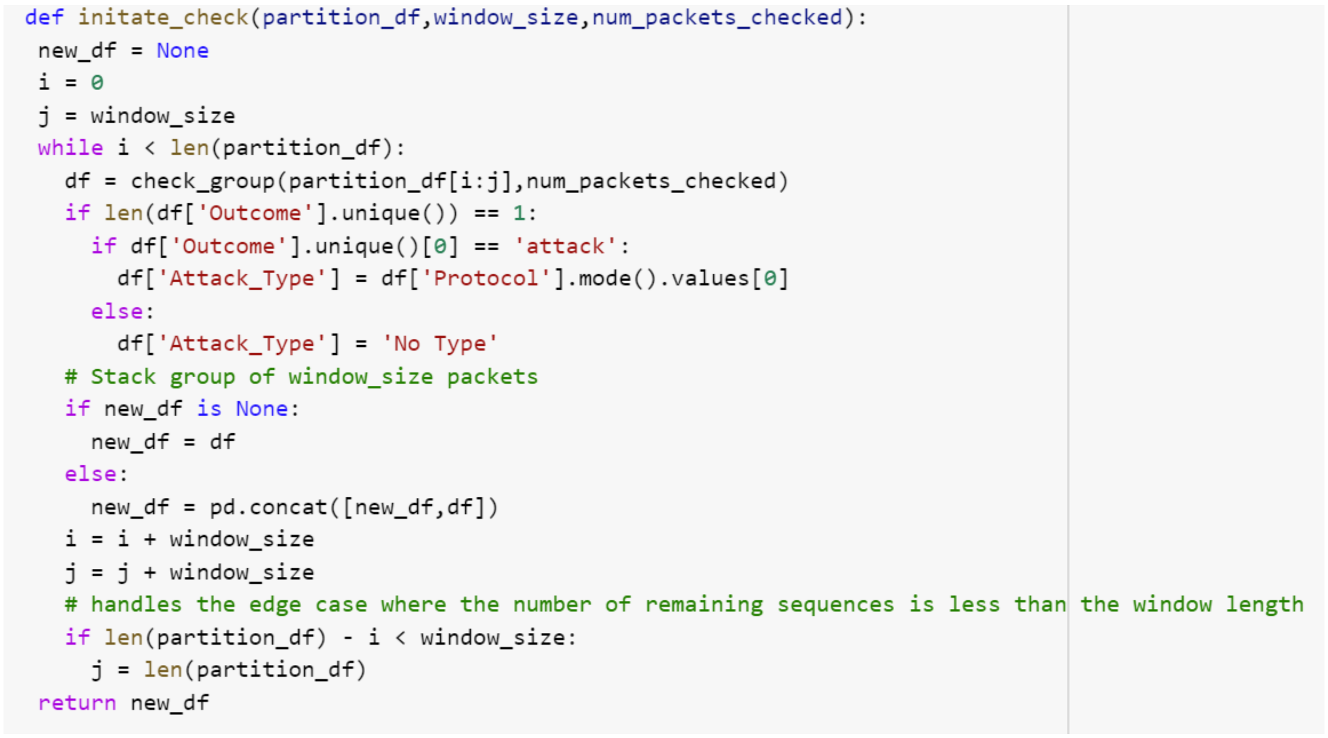
Figure 5.10 shows the code responsible for checking if any of the conditions in figures 5.6 – 5.9 are true, then return the sequence of packets with attack label. This code triggers the actual labelling of a sequence of packets as an attack or not attack depending on what has been flagged from figure 5.6 – 5.9.



```python
def check_flags(df):
  """
  Check if any of the above conditons is true,
  then the sequence is classified as an attack
  """
  enum
  # if any of this attributes flags, check_flags returns true, otherwise, check_flags return false
  flagged = False
  if check_sequence_number(df) or check_tcp_flag(df):
    return True
  # if check_protocol(df) and check_packet_length(df):
  #   if check_sequence_number(df) or check_tcp_flag(df):
  #     return True
  return flagged
```

Figure 5.10 Flagging and labelling true conditions

Figure 5.11 shows the code responsible for stacking the labelled packets of each iterated window. The protocol column of the packets labelled “attack” is checked for the most appearing protocol. The protocol with the highest count is then labelled as the attack type, thereby indicating the attack type detected. However if packets have a label of no attack, then the corresponding attack type column carries the label “ no type ”.



```python
def initate_check(partition_df,window_size,num_packets_checked):
 new_df = None
 i = 0
 j = window_size
 while i < len(partition_df):
   df = check_group(partition_df[i:j],num_packets_checked)
   if len(df['Outcome'].unique()) == 1:
     if df['Outcome'].unique()[0] == 'attack':
       df['Attack_Type'] = df['Protocol'].mode().values[0]
     else:
       df['Attack_Type'] = 'No Type'
   # Stack group of window_size packets
   if new_df is None:
     new_df = df
   else:
     new_df = pd.concat([new_df,df])
   i = i + window_size
   j = j + window_size
   # handles the edge case where the number of remaining sequences is less than the window length
   if len(partition_df) - i < window_size:
     j = len(partition_df)
 return new_df
```

Figure 5.11 Stacking labelled packets and indicating attack type

5.4.3 Initial Testing

Some of the dataset files from table 5.1 are run over the scripted detection and attack type indication algorithm from section 5.4.2. This initial testing stage focuses only on testing and recording the performance of known private attacks which comprises of a mixture of benign and attack traffic (ID 1-3) as well as benign private traffic (ID 8) from table 5.1. The number of “packets checked” will be tested by setting it to “5” and “10”. All algorithm conditions A – E from section 5.3. will be applied. Table 5.2 shows when the number of packets check is set to 5 while 5.3 shows when it is set to 10. Observations based on the performance are also noted which will be used in the tuning phase. The test case metric table is being populated with the details of the tested datasets based on several factors mentioned in section 5.2.3. This is presented in table 5.2 and 5.3. For each dataset it states the following:

- The name of the dataset, specifying if its attack or normal.
- Whether attack traffic is present in the flow.
- If the algorithm detected the attack.
- Attack type indicated.
- The packet number the attack started in the traffic flow.
- The packet number at which the algorithm detected the attack.
- The packet number at which the attack type is indicated.
- The feature(s) that triggered the attack detection. This will help to gauge the best performing feature in terms of detection trigger. The keys to this column are A = Protocol, B = Packet length, C = Sequence number, D = TCP flags, E = Encryption.
- The window where the attack started. For instance, if there are 100 packets and the window size is 20 packets. If an attack starts between packet 1 and 20 then this is the first window. The packet number the attack started is divided by 20 to get the actual window.
- The window at which the attack is detected and classified.

The number of “packets checked” will be tested by setting it to “5” and “10”. All algorithm conditions A – E from section 5.3. will be applied. This means that if any of the conditions is flagged in 5 or 10 consecutive packets, then this will be flagged as “attack”. Table 5.2 shows when the number of packets checked is set to 5 while 5.3 shows when it is set to 10.

From table 5.2 we can see that more false alarms (in red) have been raised in comparison to table 5.3 where the packets checked is set to 10. This indicates that using a threshold of 10 packets yields better results. Nevertheless, the algorithm still must be improved for better performance to reduce the false alarms. From the trigger feature column of both tables, we can see that Protocol (A), Packet Length (B) and Encryption (E) are those that led to all the false alarms. However, Sequence Number (C) and TCP flags (D) resulted in all the true cases. These factors will be taken into consideration in the tuning stage where the algorithm is improved. Figure 5.12 shows an instance where setting the packets checked to 5, labelled normal traffic as “attack” while figure 5.13 shows the 10-threshold labelling the same set of packets as “not attack” which is true.

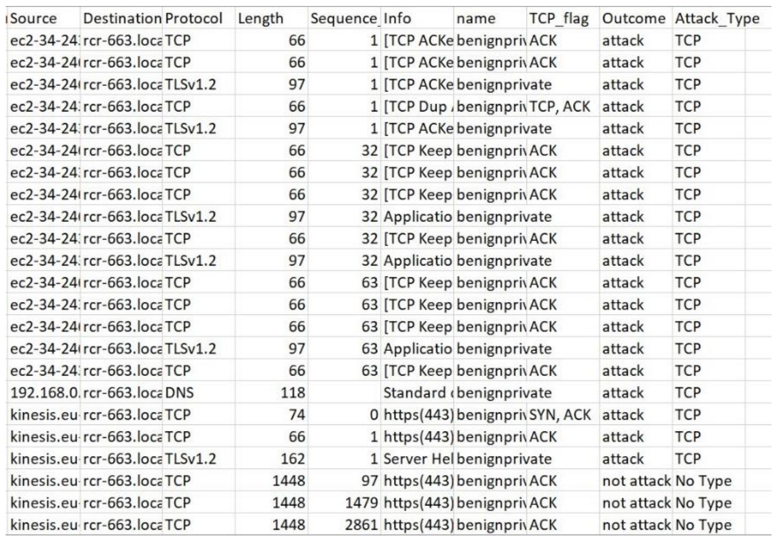
Table 5.2 Initial Testing results, packets checked set to 5

ID	Dataset	Attack present?	Attack detected?	Attack type indicated?	Pkt no attack started	Pkt no attack detected	Pkt no attack type indicated & and attack type	Trigger feature(s)	Window attack started	Window attack detected
1	TCPSYN01.csv	Yes	Yes	Yes	2402	232	232,TCP	A,E	120th	10th
2	TCPSYN02.csv	Yes	Yes	Yes	272	15	15,TCP	A,B,E	13th	1ST
3	TCPSYN03.csv	Yes	Yes	Yes	38	38	38,TCP	A,B,C,D,E	2nd	2ND
4	TCPSYN04.csv	Yes	Yes	Yes	19	19	19,TCP	A,B,C,D, E	1st	1ST
5	TCPSYN05.csv	Yes	Yes	Yes	279	53	53,TCP	A,B,E	13th	3rd
6	UDP01.csv	Yes	Yes	Yes	196	75	75,TCP	A,B,C,E	9th	4TH
7	UDP02.csv	Yes	Yes	Yes	304	118	118,TCP	A	15th	6th
8	UDP03.csv	Yes	Yes	Yes	325	46	46,TCP	A,B,E	16th	3rd
9	UDP04.csv	Yes	Yes	Yes	202	43	43,TCP	A,B,E	10th	3rd
10	UDP05.csv	Yes	Yes	Yes	297	297	297,UDP	A,B,C,D,E	14th	14th
11	ICMP01.csv	Yes	Yes	Yes	598	370	370,TCP	A,B,E	29th	19th
12	ICMP02.csv	Yes	Yes	Yes	509	96	96,TCP	A,B,E	25th	5th
13	ICMP03.csv	Yes	Yes	Yes	191	191	191,ICMP	A,B,C,D,E	9th	9th
14	ICMP04.csv	Yes	Yes	Yes	181	181	181,ICMP	A,B,C,D,E	9th	9th
15	ICMP05.csv	Yes	Yes	Yes	293	62	62,TCP	A,B,E	14th	4th
16	Normal01.csv	No	Yes	Yes	Null	12	12,TCP	A,B,E	Null	1ST
17	Normal02.csv	No	Yes	Yes	Null	6	6,TCP	B	Null	1st
18	Normal03.csv	No	Yes	Yes	Null	56	56,Tlsv1.2	A,B,E	Null	3rd
19	Normal04.csv	No	Yes	Yes	Null	93	93,TCP	A,E	Null	4th
20	Normal05.csv	No	Yes	Yes	Null	47	47,Tlsv1.2	A,E	Null	3rd

Table 5.3 Initial Testing results, packets checked set to 10

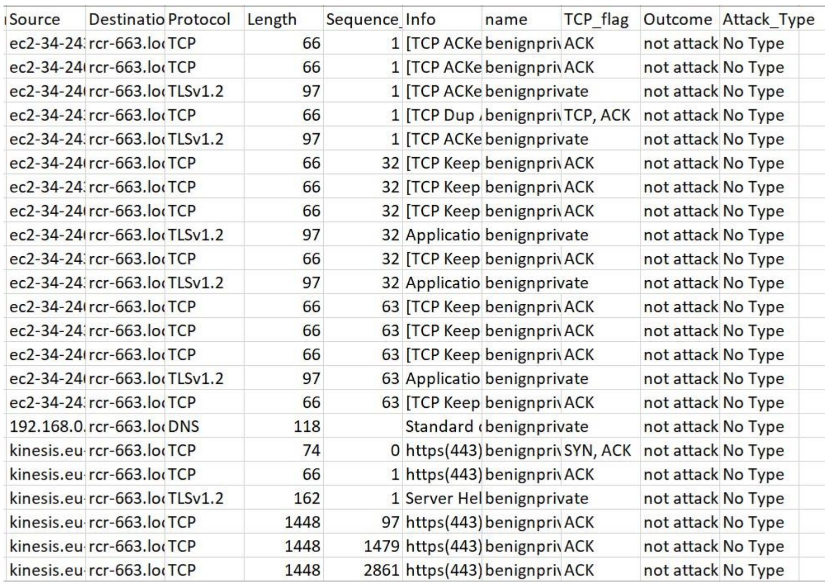
ID	Dataset	Attack present?	Attack detected?	Attack type indicated?	Pkt no attack started	Pkt no attack detected	Pkt no attack type indicated & and attack type	Trigger feature(s)	Window attack started	Window attack detected
1	TCPSYN01.csv	Yes	Yes	Yes	2402	496	496,TCP	A,E	120th	24th
2	TCPSYN02.csv	Yes	Yes	Yes	272	167	167,TCP	A,B,E	13th	8th
3	TCPSYN03.csv	Yes	Yes	Yes	38	38	38,TCP	A,B,C,D	2nd	2ND
4	TCPSYN04.csv	Yes	Yes	Yes	19	19	19,TCP	A,B,C,D, E	1st	1ST
5	TCPSYN05.csv	Yes	Yes	Yes	279	294	294,TCP	A,B,C,D,E	13th	14TH
6	UDP01.csv	Yes	Yes	Yes	196	196	196,UDP	A,B,C,D,E	9th	9TH
7	UDP02.csv	Yes	Yes	Yes	304	304	304,UDP	A,B,C,D,E	15th	15th
8	UDP03.csv	Yes	Yes	Yes	325	325	325,UDP	A,B,C,D, E	16th	16th

9	UDP04.csv	Yes	Yes	Yes	202	202	202,UDP	A,B,C,D,E	10th	10th
10	UDP05.csv	Yes	Yes	Yes	297	297	297,UDP	A,B,C,D,E	14th	14th
11	ICMP01.csv	Yes	Yes	Yes	598	598	598,ICMP	A,B,C,D,E	29th	29th
12	ICMP02.csv	Yes	Yes	Yes	509	509	509,ICMP	A,B,C,D,E	25th	25th
13	ICMP03.csv	Yes	Yes	Yes	191	191	191,ICMP	A,B,C,D,E	9th	9th
14	ICMP04.csv	Yes	Yes	Yes	181	181	181,ICMP	A,B,C,D,E	9th	9th
15	ICMP05.csv	Yes	Yes	Yes	293	293	293,ICMP	A,B,C,D,E	14th	14th
16	Normal01.csv	No	Yes	Yes	Null	81	81,TCP	B	Null	4th
17	Normal02.csv	No	Yes	Yes	Null	6	6,TCP	B	Null	1st
18	Normal03.csv	No	Yes	Yes	Null	56	56,Tlsv1.2	B	Null	3rd
19	Normal04.csv	No	Yes	Yes	Null	93	93,TCP	B	Null	4th
20	Normal05.csv	No	Yes	Yes	Null	47	47,Tlsv1.2	B	Null	3rd



Source	Destination	Protocol	Length	Sequence_	Info	name	TCP_flag	Outcome	Attack_Type
ec2-34-24:	rcr-663.loca	TCP	66	1	[TCP ACKe	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TCP	66	1	[TCP ACKe	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TLSv1.2	97	1	[TCP ACKe	benignprivate		attack	TCP
ec2-34-24:	rcr-663.loca	TCP	66	1	[TCP Dup /	benignpriv	TCP, ACK	attack	TCP
ec2-34-24:	rcr-663.loca	TLSv1.2	97	1	[TCP ACKe	benignprivate		attack	TCP
ec2-34-24(	rcr-663.loca	TCP	66	32	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24:	rcr-663.loca	TCP	66	32	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TCP	66	32	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TLSv1.2	97	32	Applicatio	benignprivate		attack	TCP
ec2-34-24:	rcr-663.loca	TCP	66	32	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24:	rcr-663.loca	TLSv1.2	97	32	Applicatio	benignprivate		attack	TCP
ec2-34-24(	rcr-663.loca	TCP	66	63	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24:	rcr-663.loca	TCP	66	63	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TCP	66	63	[TCP Keep	benignpriv	ACK	attack	TCP
ec2-34-24(	rcr-663.loca	TLSv1.2	97	63	Applicatio	benignprivate		attack	TCP
ec2-34-24:	rcr-663.loca	TCP	66	63	[TCP Keep	benignpriv	ACK	attack	TCP
192.168.0.	rcr-663.loca	DNS	118		Standard (	benignprivate		attack	TCP
kinesis.eu	rcr-663.loca	TCP	74	0	https(443)	benignpriv	SYN, ACK	attack	TCP
kinesis.eu	rcr-663.loca	TCP	66	1	https(443)	benignpriv	ACK	attack	TCP
kinesis.eu	rcr-663.loca	TLSv1.2	162	1	Server Hel	benignprivate		attack	TCP
kinesis.eu	rcr-663.loca	TCP	1448	97	https(443)	benignpriv	ACK	not attack	No Type
kinesis.eu	rcr-663.loca	TCP	1448	1479	https(443)	benignpriv	ACK	not attack	No Type
kinesis.eu	rcr-663.loca	TCP	1448	2861	https(443)	benignpriv	ACK	not attack	No Type

Figure 5.12 False positive by setting packets checked to 5



Source	Destinatio	Protocol	Length	Sequence_	Info	name	TCP_flag	Outcome	Attack_Type
ec2-34-24:	rcr-663.loc	TCP	66	1	[TCP ACKe	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TCP	66	1	[TCP ACKe	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TLSv1.2	97	1	[TCP ACKe	benignprivate		not attack	No Type
ec2-34-24:	rcr-663.loc	TCP	66	1	[TCP Dup /	benignpriv	TCP, ACK	not attack	No Type
ec2-34-24:	rcr-663.loc	TLSv1.2	97	1	[TCP ACKe	benignprivate		not attack	No Type
ec2-34-24(	rcr-663.loc	TCP	66	32	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24:	rcr-663.loc	TCP	66	32	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TCP	66	32	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TLSv1.2	97	32	Applicatio	benignprivate		not attack	No Type
ec2-34-24:	rcr-663.loc	TCP	66	32	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24:	rcr-663.loc	TLSv1.2	97	32	Applicatio	benignprivate		not attack	No Type
ec2-34-24(	rcr-663.loc	TCP	66	63	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24:	rcr-663.loc	TCP	66	63	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TCP	66	63	[TCP Keep	benignpriv	ACK	not attack	No Type
ec2-34-24(	rcr-663.loc	TLSv1.2	97	63	Applicatio	benignprivate		not attack	No Type
ec2-34-24:	rcr-663.loc	TCP	66	63	[TCP Keep	benignpriv	ACK	not attack	No Type
192.168.0.	rcr-663.loc	DNS	118		Standard (	benignprivate		not attack	No Type
kinesis.eu	rcr-663.loc	TCP	74	0	https(443)	benignpriv	SYN, ACK	not attack	No Type
kinesis.eu	rcr-663.loc	TCP	66	1	https(443)	benignpriv	ACK	not attack	No Type
kinesis.eu	rcr-663.loc	TLSv1.2	162	1	Server Hel	benignprivate		not attack	No Type
kinesis.eu	rcr-663.loc	TCP	1448	97	https(443)	benignpriv	ACK	not attack	No Type
kinesis.eu	rcr-663.loc	TCP	1448	1479	https(443)	benignpriv	ACK	not attack	No Type
kinesis.eu	rcr-663.loc	TCP	1448	2861	https(443)	benignpriv	ACK	not attack	No Type

Figure 5.13 True negative by setting the packets checked to 10

5.4.3.1 Observations

The relevant observations made which will be taken into consideration at the tuning stage are presented in table 5.4.

Table 5.4 Observation details

Observation	ID on table 5.2	ID on table 5.3	Reason
Trigger feature (E) which is the encryption condition leads to false positives.	1, 2	1,2,5,6,8,9,11,12,15,16,18,19,20	Encryption protocols tend to be absent in normal traffic for 10 packets sometimes, thus the false positives raised.
Trigger feature (E), the encryption condition leads to wrong attack type classification.	18, 20	18, 20	The classification method takes the protocol with the highest count in the attack labelled packet and labels the attack type using that protocol name thus labelling attack types using encryption labels.
Trigger features A (Protocol) and B (Packet length) lead to false positives.	1, 2, 16, 17, 18, 19, 20	1,2,5,6,8,9,11,12,15,16,18,19,20	Protocol and packet length tend to be the same for 10 consecutive packets in normal traffic sometimes.
Trigger features C (Sequence no) and D (TCP flags) lead to true positives.	3-15	3,4,10,13,14	Sequence no and TCP flags tend to always be part of the trigger features during true positive attacks. They never appear as trigger features in false positive attacks.
Packets checked set to 5 leads to more false alarms			5 consecutive packets are not sufficient to declare and attack, thus the higher false alarms observed.
Packets checked set to 10 leads to less false alarms			10 consecutive packets sufficient to declare an attack, thus the more positive alarms.

5.4.4 Tuning

The detection and attack type indication algorithm will be tuned and improved based on the noted observations in table 5.4. The respective approaches taken to improve the algorithm are presented in table 5.5.

Table 5.5 Tuning approaches

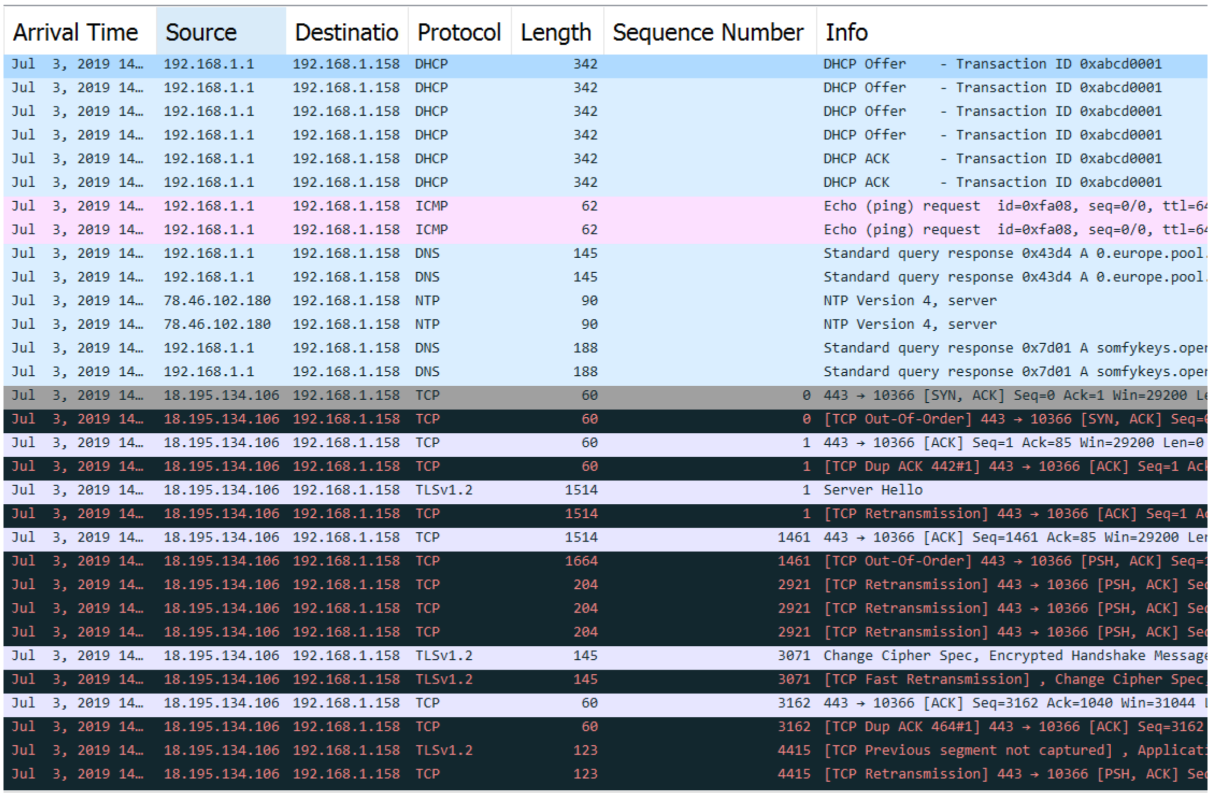
ID	Observation	Tuning approach	Location in algorithm/ Reason for false alarm
1	Trigger feature (E) which is the encryption condition leads to false positives.	Remove condition from detection algorithm.	Figure 5.6. Condition E tends to raise false positives due to the number of packets checked set to 5 or 10. For this to work, the number of packets checked has to be increased. This either calls for the condition to be dropped or have a separate number for packets checked. The former will be used as conditions C and D tend to detect an attack with packets checked set to 10 even without condition E, thus eliminating redundant conditions.
2	Trigger features A (Protocol) and B (Packet length) lead to false positives.	Remove conditions from detection algorithm.	Figure 5.6 and 5.7. Conditions A and B tend to raise False positives due to the number of packets checked set to 5 or 10. This number needs to be increased or entirely drop the conditions to solve the issue. The latter will be done to reduce redundancy in the algorithm as conditions C and D tend to detect the attack without conditions A and B.
3	Trigger features C (Sequence no) and D (TCP flags) lead to true positives.	Use them as trigger features in detection algorithm.	Figure 5.8 and 5.9
4	Packets checked “5” leads to false alarms	Set packets checked to 10	Figure 5.5. Setting the packets to 5 tends to raise false positives which indicates that its

			too low in comparison to when it is set to 10 which yields better results.

5.4.5 Final Testing and Validation

The tuned detection and attack type indication algorithm is tested on the respective collected datasets from table 4.1 section 4. This will provide clear results in terms of the algorithms performance before and after tuning. Conditions A, B and E have been dropped from the algorithm as they are responsible for generating false positives as seen from the trigger feature column of tables 5.2 and 5.3. They were also found to be redundant as conditions C and D can flag the attack pattern even without A, B and E. Table 5.6 provides the results to this. The same performance metrics from table 5.2 and 5.3 are used to measure the performance for clear comparison. All datasets from table 4.1 in chapter 4 are tested in this stage to see the performance of the algorithm after tuning it.

The results in table 5.6 clearly shows the respective tuned components have very much improved the performance of the detection and attack type indication system. We can see the accuracy achieved is much higher in terms of onset attack detection and attack type indication including unfamiliar (UDP fragmentation, RECOIL, SLOWLOIC) and mixed attacks. Normal data is also recognized as completely normal traffic except for one case**, ID 21**. The reason for this false positive is pointed out in the public dataset used [85]. The device that emanated this traffic was newly deployed to the network, thus it generated unusual DHCP traffic for connectivity establishment purposes. This is shown in figure 5.14. However, after that, the traffic flow became normal. The absence of TCP sequence numbers and TCP flags for more than 10 consecutive packets triggered this as seen from the figure. In the case of mixed attacks, only the predominant attack type in the first window of the attack is indicated as the system is designed to detect only at onset, which is a very crucial aspect in DDoS attack detection. Furthermore, the performance metrics used here gives much more precise and useful details compared to the traditional confusion matrix. The fact that early detection one of the most important aspects in dealing with DDoS attacks, the conventional performance metrics used do not provide this detail. Literature shows a system can provide high accuracy but not able to detect at onset or even provide details about the point at which attack was detected, which questions the efficiency of the system. The trigger features also show how effective the use of Sequence numbers and TCP flags are in attack traffic detection as their pattern deviates and reverses completely from the usual one. Furthermore, the normal traffic carries other protocols like DNS, ICMP, UDP among others as identified on table 4.1 in chapter 4. However, this has not identified these legitimate protocols as attack traffic but accommodates them. Even though the algorithm is designed for TCP/ HTTP based traffic, it still accommodates these legitimate protocols. This is due to the number of "packets checked" threshold which does not flag any protocol as attack unless it exceeds that number. Figure 5.15 shows a flow process of how the final version of the algorithm works.



```
Arrival Time      Source          Destinatio    Protocol Length Sequence Number Info
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP Offer    - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP Offer    - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP Offer    - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP Offer    - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP ACK      - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DHCP     342                    DHCP ACK      - Transaction ID 0xabcd0001
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 ICMP      62                    Echo (ping) request  id=0xfa08, seq=0/0, ttl=6
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 ICMP      62                    Echo (ping) request  id=0xfa08, seq=0/0, ttl=6
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DNS      145                    Standard query response 0x43d4 A 0.europe.pool
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DNS      145                    Standard query response 0x43d4 A 0.europe.pool
Jul  3, 2019 14…  78.46.102.180   192.168.1.158 NTP       90                    NTP Version 4, server
Jul  3, 2019 14…  78.46.102.180   192.168.1.158 NTP       90                    NTP Version 4, server
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DNS      188                    Standard query response 0x7d01 A somfykeys.ope
Jul  3, 2019 14…  192.168.1.1     192.168.1.158 DNS      188                    Standard query response 0x7d01 A somfykeys.ope
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60                  0 443 → 10366 [SYN, ACK] Seq=0 Ack=1 Win=29200 L
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60                  0 [TCP Out-Of-Order] 443 → 10366 [SYN, ACK] Seq=
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60                  1 443 → 10366 [ACK] Seq=1 Ack=85 Win=29200 Len=0
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60                  1 [TCP Dup ACK 442#1] 443 → 10366 [ACK] Seq=1 Ac
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TLSv1.2 1514                  1 Server Hello
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP     1514                  1 [TCP Retransmission] 443 → 10366 [ACK] Seq=1 A
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP     1514               1461 443 → 10366 [ACK] Seq=1461 Ack=85 Win=29200 Le
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP     1664               1461 [TCP Out-Of-Order] 443 → 10366 [PSH, ACK] Seq=
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP      204               2921 [TCP Retransmission] 443 → 10366 [PSH, ACK] Se
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP      204               2921 [TCP Retransmission] 443 → 10366 [PSH, ACK] Se
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP      204               2921 [TCP Retransmission] 443 → 10366 [PSH, ACK] Se
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TLSv1.2  145               3071 Change Cipher Spec, Encrypted Handshake Messag
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TLSv1.2  145               3071 [TCP Fast Retransmission] , Change Cipher Spec
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60               3162 443 → 10366 [ACK] Seq=3162 Ack=1040 Win=31044
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP       60               3162 [TCP Dup ACK 464#1] 443 → 10366 [ACK] Seq=3162
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TLSv1.2  123               4415 [TCP Previous segment not captured] , Applicat
Jul  3, 2019 14…  18.195.134.106  192.168.1.158 TCP      123               4415 [TCP Retransmission] 443 → 10366 [PSH, ACK] Se
```

Figure 5.14. New device joining network traffic

Table 5.6 Final testing and validation results

ID	Dataset	Attack present?	Attack detected ?	Attack classified?	Pkt no attack started	Pkt no attack detected	Pkt no attack classified and type	Trigger feature(s)	Window attack started	Window attack detected
1	HTTP01.csv	Yes	Yes	Yes	47	47	47,TCP	C,D	2ND	2ND
2	HTTP02.csv	Yes	Yes	Yes	124	124	124,TCP	C,D	6TH	6TH
3	HTTP03.csv	Yes	Yes	Yes	189	189	189,TCP	C,D	9TH	9TH
4	SLOWLOIC01.csv	Yes	Yes	Yes	209	209	209,TCP	C,D	10TH	10TH
5	SLOWLOIC02.csv	Yes	Yes	Yes	312	312	312,TCP	C,D	15TH	15TH
6	RECOIL.csv	Yes	Yes	Yes	192	192	192,TCP	C,D	9TH	9TH
7	UDP01.csv	Yes	Yes	Yes	11462	11462	11462,UDP	C,D	573RD	573RD
8	TCPSYN01.csv	Yes	Yes	Yes	2618	2618	2618,TCP	C,D	130TH	130TH
9	TCPSYN02.csv	Yes	Yes	Yes	25295	25295	25295,TCP	C,D	1264TH	1264TH
10	Mixed001.csv	Yes	Yes	Yes	204	204	204,TCP	C,D	10TH	10TH
11	Mixed002.csv	Yes	Yes	Yes	120	120	120,UDP	C,D	6TH	6TH
12	Mixed003.csv	Yes	Yes	Yes	283	283	283,ICMP	C,D	14TH	14TH
13	FRAG001.csv	Yes	Yes	Yes	129249	129249	129249,UDP	C,D	6462ND	6462ND

14	FRAG002.csv	Yes	Yes	Yes	163740	163740	163740,UDP	C,D	8187TH	8187TH
15	Normal001.csv	No	No	No	Null	Null	Null	Null	Null	Null
16	Normal002.csv	No	No	No	Null	Null	Null	Null	Null	Null
17	Normal003.csv	No	No	No	Null	Null	Null	Null	Null	Null
18	Normal004.csv	No	No	No	Null	Null	Null	Null	Null	Null
19	Normal005.csv	No	No	No	Null	Null	Null	Null	Null	Null
20	Normal006.csv	No	No	No	Null	Null	Null	Null	Null	Null
21	Normal007.csv	No	Yes	Yes	1	1	1,DHCP	C,D	1st	1st
22	TCPSYN01.csv	Yes	Yes	Yes	2402	2402	2402,TCP	C,D	120th	120th
23	TCPSYN02.csv	Yes	Yes	Yes	272	272	272,TCP	C,D	13th	13th
24	TCPSYN03.csv	Yes	Yes	Yes	38	38	38,TCP	C,D	2nd	2ND
25	TCPSYN04.csv	Yes	Yes	Yes	19	19	19,TCP	C,D	1st	1ST
26	TCPSYN05.csv	Yes	Yes	Yes	279	279	279,TCP	C,D	13th	13th
27	UDP01.csv	Yes	Yes	Yes	196	196	196,UDP	C,D	9th	9TH
28	UDP02.csv	Yes	Yes	Yes	304	304	304,UDP	C,D	15th	15th
29	UDP03.csv	Yes	Yes	Yes	325	325	325,UDP	C,D	16th	16th
30	UDP04.csv	Yes	Yes	Yes	202	202	202,UDP	C,D	10th	10th
31	UDP05.csv	Yes	Yes	Yes	297	297	297,UDP	C,D	14th	14th
32	ICMP01.csv	Yes	Yes	Yes	598	598	598,ICMP	C,D	29th	29th
33	ICMP02.csv	Yes	Yes	Yes	509	509	509,ICMP	C,D	25th	25th
34	ICMP03.csv	Yes	Yes	Yes	191	191	191,ICMP	C,D	9th	9th
35	ICMP04.csv	Yes	Yes	Yes	181	181	181,ICMP	C,D	9th	9th
36	ICMP05.csv	Yes	Yes	Yes	293	293	293,ICMP	C,D	14th	14th
37	Normal01.csv	No	No	No	Null	Null	Null	Null	Null	Null
38	Normal02.csv	No	No	No	Null	Null	Null	Null	Null	Null
39	Normal03.csv	No	No	No	Null	Null	Null	Null	Null	Null
40	Normal04.csv	No	No	No	Null	Null	Null	Null	Null	Null
41	Normal05.csv	No	No	No	Null	Null	Null	Null	Null	Null

The following are the results achieved from table 5.6.

- Total number of test cases: 41
- Actual True positives: 29, Attained True positives: 29
- Actual True Negatives: 12, Attained True Negatives: 11
- False positives: 1
- False Negatives: 0

Table 5.7 shows which conditions (A TO E) from section 5.3 result in false positives and true positives. No false negatives have been attained, thus not addressed here.

Table 5.7 Conditions and their outcomes

Condition(s)	**Outcome**
A	False positive
B	False positive
C	True positive
D	True positive
E	False positive
A and B	False positive
A and E	False positive
A and B and E	False positive
C and D	True positive

Figure 5.15 shows a flow process of how the final version of the algorithm works. After capturing 20 packets, the sequence number is checked, and condition C (section 5.3) is applied. If found true, the packets are labelled as "attack" and if found false they are labelled as "not attack". The TCP flags are checked next and condition D (section 5.3) is applied. If found true, the packets are labelled as "attack" and if found false they are labelled as "not attack". The highest protocol count of those labelled "attack" is taken, and the protocol name is used to further indicate the attack type.

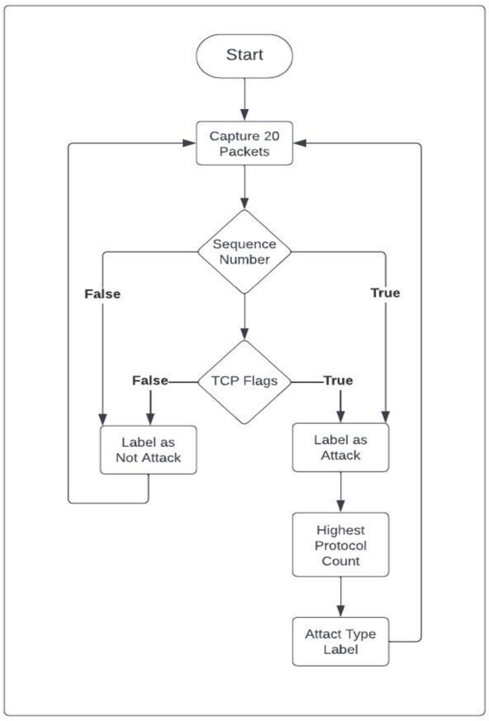


Figure 2.15 Algorithm Flow process

5.5 Comparison to literature and new findings

This section compares the systems performance to existing solutions in DDoS detection in the smart home network. It is based on 12 comparison factors as presented on table 5.8 and 5.9 which are:

- ✓ Features used: The respective features used in attack detection are listed and compared to the ones used in this research. We can see that majority of the features used are device dependent like IP address, ports and packet frequency which tend to make the solution device centric. However, this research uses generalized features applicable to all smart home devices thereby making the solution not device or user centric. In addition to that, we can see that existing solutions monitor 2-way traffic while our solution monitors one way which contributes to reduction of redundant traffic thereby making the system light weight.
- ✓ Attacks covered: The various attacks the existing systems can detect are identified. We can see that existing solutions tend to be attack centric detecting between 1 to 4 attacks overall. However, our solution can detect and indicate the attack type of a wider range of DDoS attacks like volume, protocol and amplification based including unfamiliar attacks. This is due to the nature of the network features used which are generalized smart home features and not device specific. Furthermore, some generalized attack features (absence and range) are also used which makes the solution a hybrid one and robust enough to cater for all attacks and devices.
- ✓ Problem solved: The problem solved by the existing solutions are compared to ours. We can see that current solutions focus on attack detection while ours both detects and indicates the attack type including unfamiliar and mixed attacks.
- ✓ Data source: The various datasets or data sources used by existing solutions for testing/validation are compared. We can see that a reasonable number of existing works used only private simulated data. In addition to that some have used public datasets which are outdated like the DARPA99 [129]. Our solution stands out as it uses both private data generated from real smart home environment, not simulated plus making use of recent reputable public datasets for validation.
- ✓ Focus: Areas where the existing solutions focus on are identified. Several existing works tend to be either attack, device, or user centric due to the detection features used. Our solution stands out as it is not user, attack, or device centric as detection features used a general smart home and attack traffic properties, thus catering for both device and attack perspectives.
- ✓ Practicality: This indicates how feasible or practical deploying the system is in real life environments. We can see that some solutions tend to consume a lot of resources, while some are not fit for diverse environments. Some also require extraction of device behaviour which

doesn’t seem practical in a large or diverse network as this will be time consuming and requires device behaviour extraction each time a new device is added to the network. Our solution stands out in this aspect as it is not device or user pattern specific, so it doesn’t require any sort of training or extraction. This makes it ready to deploy at any environment. Our solution also tends to be suitable for diverse settings due to its accommodating features.

- ✓ Performance metrics: The metrics used to assess a systems performance are identified here. From the table we can see that existing solutions use confusion matrix, memory utilization, CPU utilization, rejected packets, blacklisted addresses and the like. However, none of these metrics gives clear and precise details on whether the system was able to detect the attack at the very onset, which should be one of the most relevant details when it comes to DDoS detection performance measure. Some works have not assessed their performance at all as seen form table 5.8 and 5.9. Our work stands out as the performance metrics used clearly provides information on how early and accurately the system was able to detect and classify.
- ✓ Onset detection: How early the existing solutions have been able to detect or classify the attacks are outlined. We can see that none of the reviewed works have provided this detail which should be at the top of the list when it comes to DDoS attack detection. Our solution has provided these details and each time it is able to detect and indicate attack type accurately from the very onset.
- ✓ Validation: This indicates how a system has been validated to prove that it is reliable outside the private network or initial data source it is tested on. We can see that based on the papers reviewed in this work, solutions have not been validated on reputable public data. Our solution has been validated using recent and reputable public data as well as unfamiliar attacks on which it performed excellently well. This proves that the solution is applicable to a diverse range of devices and networks as well as attacks.
- ✓ Approach: The detection approach used by existing works and how it affects their performance is compared. We can see that most of the approaches lead to the solutions being either device, user or attack centric due to the detection approach (mainly features) used while some lead to high False Positive alarms. Approaches that use real time entropy tend to miss the attack traffic when it starts at the very beginning of a window or at the very end of a window, so the attack is only detected if it starts at the middle of window. Our approach stands out as it uses feature absence, variance, and range to detect the attacks thereby not being centric and having no false positive alarms. Our approach has no issue missing any attack that starts at the very beginning of a window as a set of predefined rules and features are used for detection as opposed to the real time entropy method.

- ✓ Coverage: This shows the area a solution covers at a time during monitoring. Majority of the solutions monitor at device level while our work monitors at network level (router). This provides a central point of detection covering the entire network at once.
- ✓ Counter spoof: This indicates if a solution is counter spoof in terms of port and IP spoofing. Majority of the existing works are not counter spoof as they use IP addresses and sometimes port numbers as detection features which can mistake a spoofed feature for a legitimate device and let it pass through as part of the whitelisted devices. Furthermore, if it gets blacklisted this can block legitimate IP's. Our solution avoids this by not using features that can be spoofed as a basis for detection.

Table 5.8 Comparison to literature

ID	Factor	[20]	[23]	[43]	[55]	[79]	Our solution
1	Features and flow direction	Communication direction,destination IP,port,protocol,time interval of packets,number of packets,packet sequence, DUD. 2 way	Packets/sec , payload size. 2 way	Networked smart object(NOS). 2 way	Attack signatures 2-way	TCP SYN flags. 2 way	Protocol(for attack type identification),seq no,TCP flags
2	Attacks covered	UDP,SYN,ACK ,DNS,HTTP	Not mentioned	Not mentioned	TCP,UDP,ICMP,IRC	TCP SYN, ICMP, DNS	All DDoS flooding attacks and unfamiliar
3	Problem solved	Detection	Detection	Detection	Detection	Detection	Detection & attack type identification
4	Data source	Private	Private(simulated)	Private	ISOT,BOTIoT,IoT 23	Private(simulated)	Private(real network), IoT23,BOTIoT,BUETDDoS2020
5	Focus	Device, user centric	User/ device centric	Not clear	Not clear	Attack centric	Covers all
6	Practicality	Not feasible in large scale as you must extract each device behaviour.	Features used will lead to FP in diverse environments as they vary between devices	High resource consumption	Resource consuming	Analyses traffic from each device independently which can cause delays.	Suitable for large scale and diverse networks. No false positive alarms.
7	Performance metrics and rates	Not clear Detection rate: 97-99%	Memory and CPU utilization	Latency, computing effort, attack recovery time	Accuracy, detection time, CPU &memory usage	Not used	Packet attack started, detected,type indicated, Window attack started,detcted
8	Onset detection	Not mentioned	No evidence	Not mentioned	Not mentioned	Not mentioned	Detects and indicates attack type at onset

9	Validation	Not mentioned	Not validated	Validated using 1 source	Validated	Not validated	Validated
10	Approach	Change in device or user behaviour will cause FP.	Detection approach problematic as features used will raise FP	Not scalable as number of NOS determines how much attack it can handle	Only known attacks are detected.	Doesn't provide detection approach	Features used covers all devices and attacks and detects at very onset.
11	Coverage	Device level	Device level	Device level	Device level	Network level	Network level
12	Counter spoof	No	No	No	No	No	No

Table 5.9 Comparison to literature

ID	**Factor**	**[80]**	**[81]**	**[86]**	**[87]**	**[89]**	**Our solution**
1	Features and flow direction	IP address, interval between packets. 2 way	Packet sending rate, signal power. 2 way	Number of packet sent over a duration. 2 way	Source/destination IP & port. 2 way		Protocol(for attack type indication),seq no,TCP flags
2	Attacks covered	Not mentioned	Hello flooding, version number modification	TCPSYN	TCPSYN,UDP	TCP SYN,Smurf	All DDoS flooding attacks and unfamiliar
3	Problem solved	Detection	Detection	Detection	Detection	Detection	Detection & attack type indication
4	Data source	Private(Simulation)	Private	Private	Private(simulated) and BOTIoT	DARPA99	Private(real network), IoT23,BOTIoT,BUETDDoS 2020
5	Focus	Not clear	Attack, user centric	Attack,user,device centric	Attack centric	Attack centric	Covers all
6	Practicality	Not feasible in diverse environments as features used can raise false alarms	Features used will raise FP in diverse environment	Feature used will lead to FP.	Not practical for large scale as system stops detecting when window gets too large.		Suitable for large scale and diverse networks. No false positive alarms.
7	Performance metrics and rates	No of whitelisted and black listed IP's Accuracy: 99.1	Not tested	Rejected packets	Detection rate,FP,mean,standard deviation Detection rate: 80%, 20%FPR	Not used	Packet attack started, detected,attack type indicated, Window attack started, and indicated
8	Onset detection	Not mentioned	Not mentioned	Not mentioned	Not mentioned		Detects and indicates attack type at onset
9	Validation	Not validated	Not validated	Not validated	Validated	Not validated	Validated
10	Approach	Features used can lead to blocking legitimate IP's	Countermeasure can lead to blocking legitimate IP's	Not practical for diverse or large-scale environments.	Attack starting at the beginning of a window or end of a previous window is not detected		Features used covers all devices and attacks and detects at very onset.

11	Coverage	Network level	Network and device	Device level	Network		Network level
12	Counter spoof	No	No	No	No	No	Yes

Several new findings have surfaced from this chapter. These include:

- The use of absent features as a basis for attack detection including unfamiliar attacks. We have seen how absence of TCP flags and Sequence numbers gives a strong base for attack detection as well as encryption protocols in heavily encrypted networks. As these are predominant and frequent in normal traffic, their absence creates a deviating pattern.
- The use of feature range as a basis for attack detection including unfamiliar attacks. We have seen how the sequence number range can be used to determine the very onset of attack traffic. Sequence numbers in normal traffic tend to be incremental and very dynamic while they get stalled at 0 or 1 during an attack.
- The use of highest protocol count to indicate the attack type of both known and unfamiliar attacks. We have seen how effective using the predominant protocol from attack labelled packets lead to straightforward and accurate attack type indication, more interestingly for unfamiliar attacks. This provides timely information as to the protocol exploited during the attack which in turn can be used to decide what countermeasure to take.
- Presentation of more relevant metrics in terms of evaluating the performance of a DDoS detection and classification system. We have seen how the performance metrics used in the testing phase of this chapter has provided more relevant and precise details as opposed to the conventional means of using confusion matrices and other statistics that don't point out how early a proposed system is able to detect attacks.

5.6 Summary

This chapter has implemented and tested a hybrid anomaly and feature-based DDoS detection and attack type indication algorithm. This is based on general smart home traffic properties and attack signatures derived from the EDA in the previous chapter. The system has been tested on private attack and normal data and on public data including unfamiliar attacks. Rigorous testing has been carried out and the system performed well. The system has also proven the effectiveness and importance of using feature absence as a basis for attack detection. Performance metrics for DDoS detection and classification systems has also been presented which have proven to be more effective and relevant in terms of providing precise and accurate performance details as opposed to the traditional confusion matrix and other statistics that don't say if a system can spot an attack at the very onset.

Chapter 6

A Novel Hybrid Machine Learning Attack Type identification model using Domain Knowledge

6.1 Introduction

The previous chapter implemented a hybrid anomaly and feature-based DDoS detection and attack type identification system using a set of properties derived from the EDA. This chapter aims to apply the approach used in the previous chapter for attack type identification using a Supervised Machine Learning model. The network features used in detection as well as the highest protocol count attack type indication approach will be integrated into the ML model with the aim of achieving better performance in terms of attack type indication. Random Forest is the chosen model due to its promising performance after testing and comparing it to other ensemble models. It has also shown promising results on smart home traffic from literature [97] [98] [99] [100]. First, the model is trained and tested to see its performance in attack type indication. It is then coupled with the indication approach used in the previous chapter to compare which model (RF OR hybrid RF) performs better at attack type indication. This is trained and tested on 3 flooding attacks (TCP SYN, UDP, ICMP). The performance of the system is evaluated, and improvements are made to the classification algorithm after which it is further tested on unfamiliar flooding attacks (HTTP, Slow LOIC, RECOIL) and a mixture of all the attacks (TCP SYN, UDP, ICMP, HTTP, RECOIL, slow LOIC). It is also validated using public attack and benign data to eliminate biases and verify that it is not user, attack nor device centric. Finally, the trained and tested hybrid Random Forest detection and attack type identification model (Hybrid RF) is compared to other state of the art supervised DDoS attack type classification models like CNN [28] Gradient Boosting [133] and Ada Boost [136]. This chapter covers and achieves the following points:

- A novel detection supervised learning model capitalizing on feature absence is presented. This is tested on both private and public data including unfamiliar attacks.
- The supervised learning detection model is coupled with an algorithm-based attack type indication approach based on domain knowledge for more accurate attack type indication. This is also evaluated using both private and public data including unfamiliar attacks.

The remaining sections in this chapter cover methodology used, how the proposed hybrid detection model works, an implementation of the model, its performance and results achieved, a comparison of the systems performance to literature, new findings, and finally a summary of the chapter.

6.2 Methodology

The various processes and sub-processes followed to achieve a successful implementation of the proposed hybrid detection and attack type indication model is broken down in this section as shown in figure 6.1.

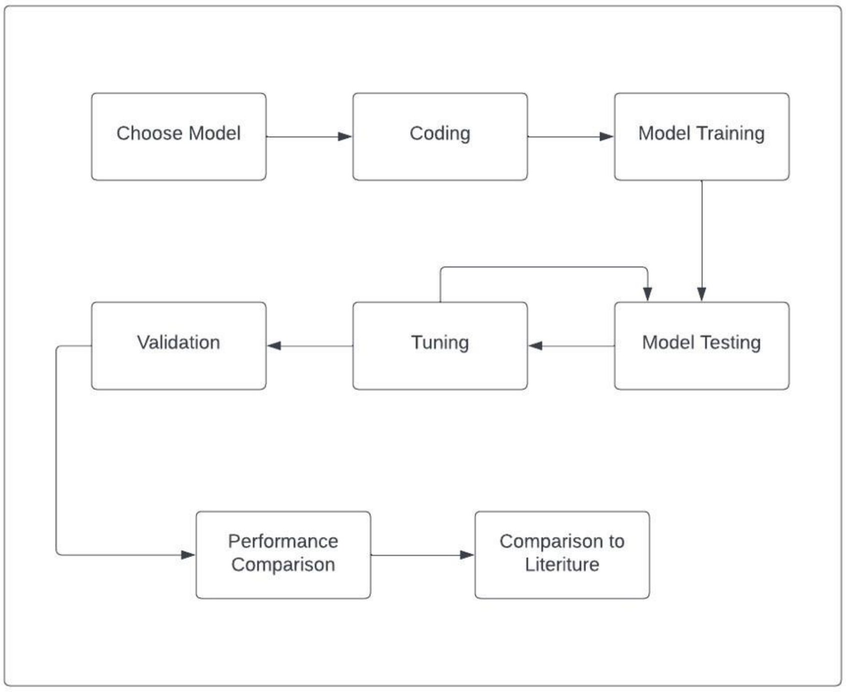


Figure 6.1 Methodology

This methodology consists of 8 main phases which are choosing model, coding, model training, model testing, tuning, validation, performance comparison and comparison to literature. The methodology is designed to be smooth and seamless in terms of transitioning from one phase to the next. It is manageable due to its segregated phases with having to complete one phase before moving onto the next as the results from the previous phase are used to commence working on the next phase. The model accommodates observations from the testing phase which will be used in tuning the system for better performance at the validation stage. Validation data includes unfamiliar attacks, mixed attacks, and public data for elimination of biases in terms of data source used and prove that the hybrid model is not user, device, nor attack centric, which works on zero day and mixed attacks which this methodology accommodates. A concise way to measure the systems performance in relation to how early and accurate the attack is detected and classified is also presented using more relevant metrics and factors. The methodology also lays out a comparison of the systems performance to other state of the art solutions based on several clear and critical factors. The factors based on which the systems performance is measured tends to be clearer and more scientifically sound which makes this methodology stand out when compared with state-of-the-art methods. The various phases and what they entail are as follows:

- ✓ **Choose model**: This phase involves reviewing literature to find the most suitable model (supervised ML model) in terms of attack detection and classification using smart home network traffic. Other ensemble models will also be tested and compared with the most suitable model found from literature.
- ✓ **Coding**: This phase involves importing the required python libraries to be used during training and testing. These libraries will provide pre-built functions which will simplify the algorithm scripting process. This means pre-existing, tested and working code will be used for some tasks as this will reduce errors and inconsistencies. Helper functions will also be created to have an organized and modular code. The created functions will also be reusable in other parts of the code, thus eliminating the need for repetitive code. Updates made where these functions are used will also reflect throughout thereby saving time and avoiding inconsistencies.
- ✓ **Model training**: Several stages are involved in this phase to ensure an effective model is achieved at the end. It consists of data collection, data pre-processing, splitting the datasets, and finally training the model using the datasets.
- ✓ **Model testing**: The performance of the trained models is evaluated here. The testing bit of the split dataset is used for evaluating the model's performance. The models will be tested based on attack detection accuracy. Observations will be made at this stage which will be used in the tuning stage. At this stage the best performing model will be coupled with the attack type indication module and the performance will be compared when it is hybrid and on its own.
- ✓ **Tuning**: This phase involves making improvements to the hybrid model based on its performance at the testing stage. Potential issues to look out for include trying different sliding windows and tuning the attack type indication algorithm.
- ✓ **Validation**: This phase involves using additional public datasets consisting of mixed (attack and benign) and purely benign traffic. Private data will also be used including unfamiliar and mixed attacks. This will provide bias free performance details at the same time ensuring rigorous testing and validation.
- ✓ **Performance comparison**: The performance of the chosen model in attack type classification will be compared to its performance when coupled with the domain knowledge-based attack type indication approach. The important factors are how accurate and early the attack type is detected and classified.

- ✓ **Comparison to literature**: The performance of the hybrid detection and attack type identification model will be compared to existing solutions. This will prove or disprove how important domain knowledge is when dealing with attack detection and attack type indication when designing machine learning models.

6.3 Proposed hybrid Machine Learning detection and attack type identification model

The proposed hybrid detection and attack type identification model has two main functions. First it detects the attack based on what it learned from the training data and secondly it indicates the attack type from attack labelled packets based on the most predominant protocol. Figure 6.2 shows the flow process involved.

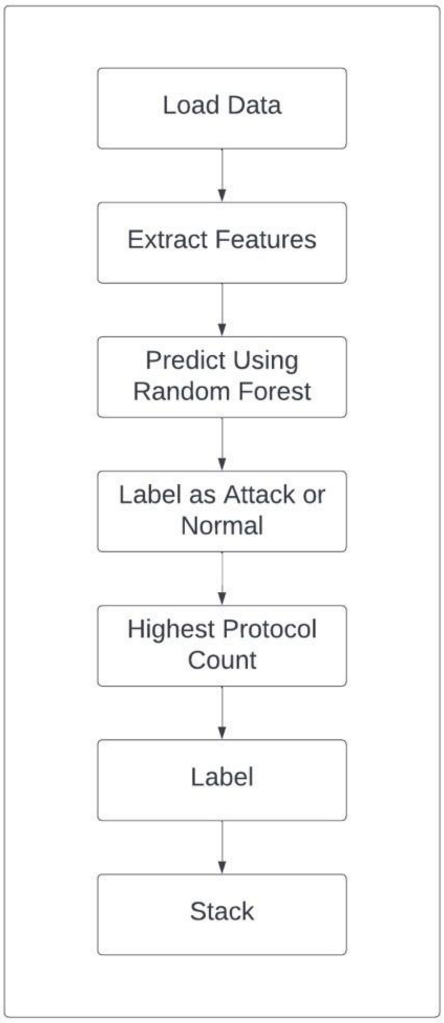


Figure 6.2 Hybrid model flow process

After loading the dataset, the required features are then extracted with the appropriate flow direction (destination IP address). The features used in the previous chapter for the hybrid anomaly and feature-based solution are used here, including the absent features. These include protocol, packet length, TCP flags and sequence numbers in addition to packet inter-arrival times. After the extraction, the chosen model predicts what the traffic is, based on its learning knowledge of what differentiates an attack from normal traffic. The traffic is then labelled as either the predicted attack type or normal. The labelled packets both attack and normal are then stacked in an output location. The protocol column of the attack type labelled packets is inspected and the protocol with the highest count is used as the attack type label for those packets. This is also stacked, and the process starts over. Stacking refers to placing a window of labelled packet on top of the previously labelled ones.

6.4 Implementation

This section delves into the series of steps and processes carried out in selecting the best model which will be used in the implementation of the hybrid detection and attack type identification model. Several models were tested to get the best performing model.

6.4.1 Coding

As mentioned earlier several libraries were imported in Google Colab to start with. These include:

- Pandas [130]: This is for data manipulation. It makes provision for structures like DataFrame which helps in analysing and cleaning organized data.
- NumPy [130]: This is for mathematical operations like matrix operations and handling of large data sources that require complex computations.
- Matplotlib [130]: This is for data visualization as it provides plotting and presentation tools.
- Seaborn [130]: This is for more complex visualizations. It is an extension of Matplotlib.
- Pickle [130]: This is for serializing and deserializing objects so they can be saved and loaded at later times like models that have been already trained.
- Re [130]: This allows for use of regular expressions to search and manipulate text output in the data.
- Scikit-learn [130]: This is for model development and evaluation.

The next step is the creation of helper functions. The created functions are:

- read_multifiles: For reading in several data sources at a time.
- make_modeling_data: For data pre-processing steps like feature extraction and scaling and splitting into training and testing.
- make_attack_model: For fitting pre-processed and split data into Random Forest model for training and testing.
- attack_detector: For unseen data source attack type classification.
- extract_flow_metrics: For extracting flow features from each window.
- get_tcp_flag: For extraction of TCP flags from the info column.

6.4.2 Model Selection

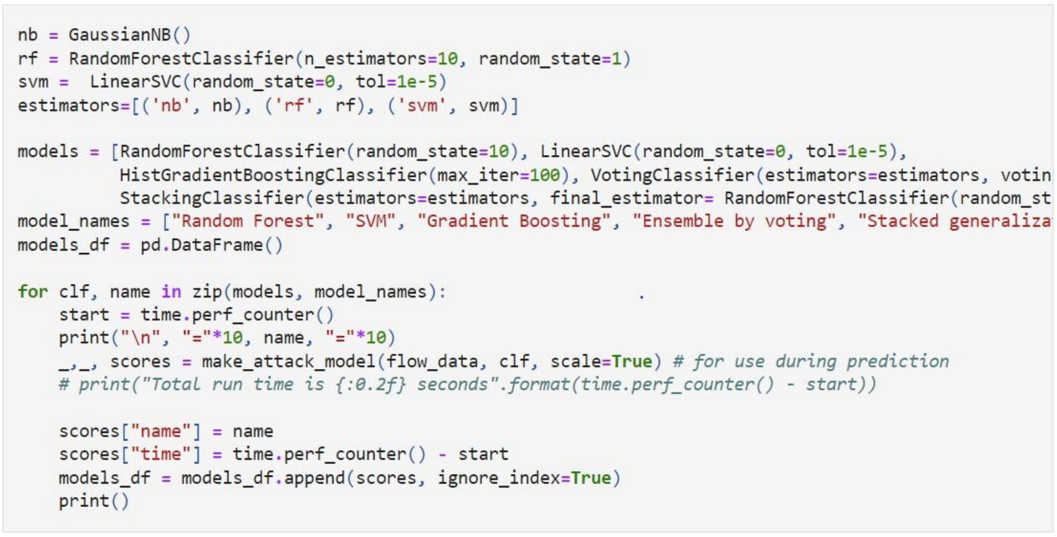
The following ensemble models are trained and tested to find out the best performing one in terms of attack detection:

- Random Forest (RF) [131]: This is an ensemble learning method used in regression and classification. It combines several decision trees at the training stage and takes the mode as output for classification tasks while using the mean for regression tasks.
- Support Vector Machines (SVM) [132]: This is another supervised machine learning algorithm that does classification and regression. It works by finding a hyperplane which differentiates the data points there by assigning them to their respective classes.
- Gradient Boosting [133]: This is another ensemble model that is used for classification and regression tasks. By combining predictions from multiple weaker models, it builds its own strong prediction model.
- Ensemble by voting [134]: This is another ensemble model that uses several machine learning models that have been trained independently and using their combined predictions to arrive at a final prediction. In classification scenarios a majority vote is used for final prediction while the average is used for regression cases.
- Stacked generalization [135]: This is another ensemble learning technique that uses predictions from several other base models by using a meta model. It combines the strengths of different models thus improving the overall performance.
- Ada boost classifier [136]: This is another ensemble model used for classification tasks. It combines weak classifiers and improves an overall accuracy by merging their respective predictions.

These models were trained and tested using the “make attack model function” as shown in figure 6.3. The private datasets from table 6.1 are used in training the models. Figure 6.4 shows the performance of these models compared. We can see that Random Forest scored highest in terms of Balance accuracy and F1 score. It is also the second fastest in terms of training time after Gradient Boosting. This makes Random Forest the selected model that will be coupled with the attack type identification module from the algorithm in chapter 5.

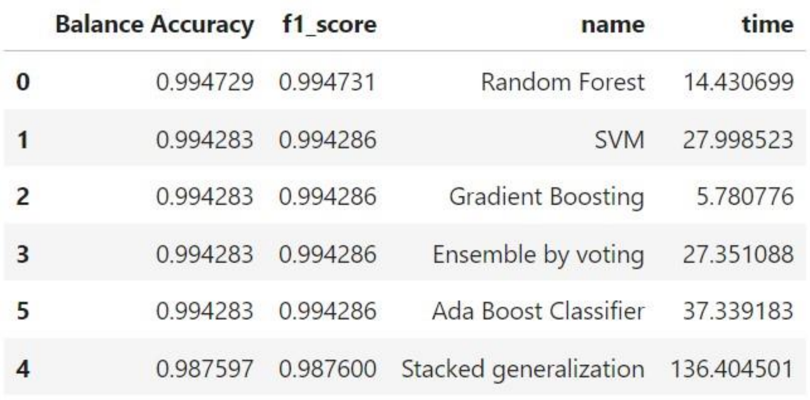
Table 6.1 Datasets

ID	Label	Source IP	Target IP	Public/Private	Known/Unknown	Count	Composition
1	TCPSYN01.csv	192.168.0.103	192.168.0.102	Private	Known	5	Attack+benign
2	UDP01.csv	192.168.0.103	192.168.0.102	Private	Known	5	Attack+benign
3	ICMP01.csv	192.168.0.103	192.168.0.102	Private	Known	5	Attack+benign
4	HTTP01.csv	192.168.0.103	192.168.0.102	Private	Unknown	3	Attack+benign
5	SLOWLOIC01.csv	192.168.0.103	192.168.0.102	Private	Unknown	2	Attack+benign
6	RECOIL01.csv	192.168.0.103	192.168.0.102	Private	Unknown	1	Attack+benign
7	Mixed01.csv	192.168.0.103	192.168.0.102	Private	Unknown	3	Attack+benign
8	Normal01.csv		192.168.0.101	Private		1	Benign
9	TCPSYN001.csv	192.168.100.147-150	192.168.100.3	Public	Known	1	Attack+benign
10	UDP001.csv	192.168.100.147-150	192.168.100.3	Public	Known	1	Attack+benign
14	Normal001.csv		192.168.1.158	Public		5	Benign



```
nb = GaussianNB()
rf = RandomForestClassifier(n_estimators=10, random_state=1)
svm =  LinearSVC(random_state=0, tol=1e-5)
estimators=[('nb', nb), ('rf', rf), ('svm', svm)]

models = [RandomForestClassifier(random_state=10), LinearSVC(random_state=0, tol=1e-5),
          HistGradientBoostingClassifier(max_iter=100), VotingClassifier(estimators=estimators, votin
          StackingClassifier(estimators=estimators, final_estimator= RandomForestClassifier(random_st
model_names = ["Random Forest", "SVM", "Gradient Boosting", "Ensemble by voting", "Stacked generaliza
models_df = pd.DataFrame()

for clf, name in zip(models, model_names):
    start = time.perf_counter()
    print("\n", "="*10, name, "="*10)
    _,_, scores = make_attack_model(flow_data, clf, scale=True) # for use during prediction
    # print("Total run time is {:0.2f} seconds".format(time.perf_counter() - start))

    scores["name"] = name
    scores["time"] = time.perf_counter() - start
    models_df = models_df.append(scores, ignore_index=True)
    print()
```

Figure 3.3 Trained models



	Balance Accuracy	f1_score	name	time
0	0.994729	0.994731	Random Forest	14.430699
1	0.994283	0.994286	SVM	27.998523
2	0.994283	0.994286	Gradient Boosting	5.780776
3	0.994283	0.994286	Ensemble by voting	27.351088
5	0.994283	0.994286	Ada Boost Classifier	37.339183
4	0.987597	0.987600	Stacked generalization	136.404501

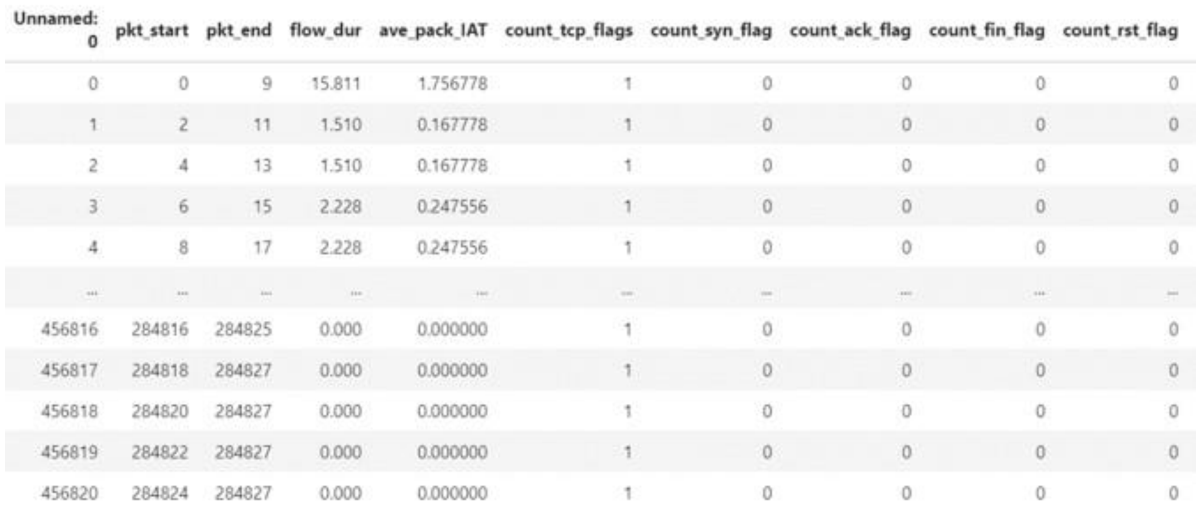
Figure 6.4 Comparing model performance

6.4.3 Chosen Model

Random Forest was selected based on its performance among the tested models in section 6.4.2 and from reviewed literature. It was found to perform well on DDoS detection and classification using smart home traffic. It is an ensemble learning method comprising of several decision trees and collectively harnessing their prediction capabilities. This gives it a robust nature paving way for more accurate results. Studies have shown the Random Forest model to have performed well in DDoS attack detection in smart home networks and IoT in general [97] [98] [99] [100]. This is retrained using the same private datasets from table 6.1.

The stages involved in this phase are as follows:

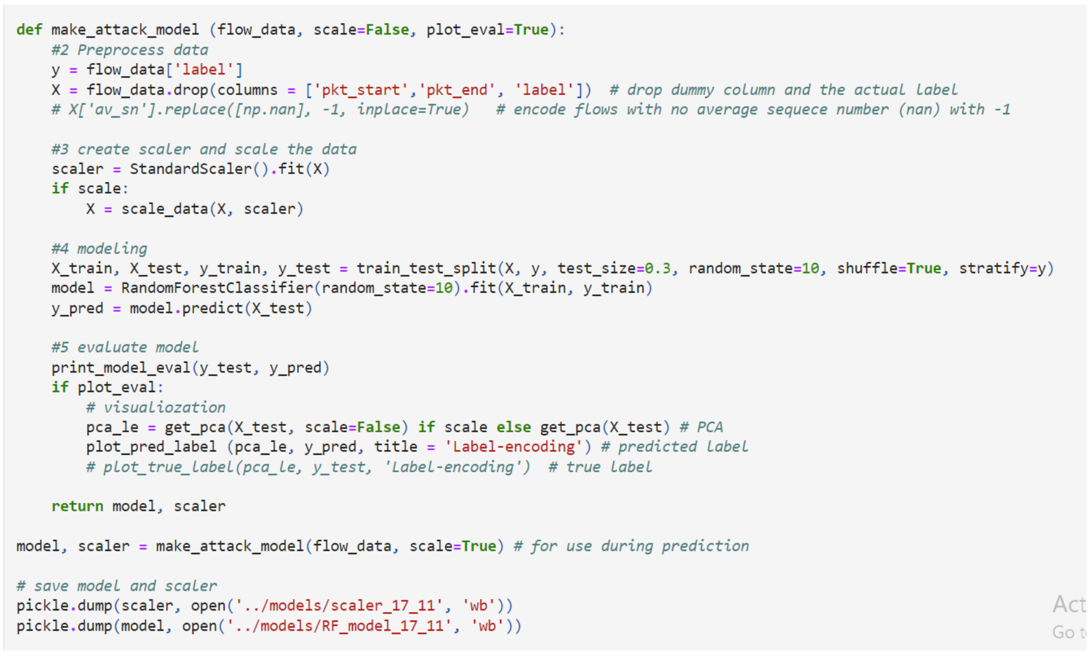
- Create training data: The read_multiplefiles function is used to load in the numerous datasets to train the model. Background noises are filtered out. For the benign dataset the smart hubs IP address is used as destination address filter. This will leave only data flowing to the smart home device. For the attack dataset, the attack IP is used as source and smart home device IP as destination. This way only traffic flowing from the attack source to the smart device is left. The flow metrics are extracted next using the extract_flow_metrics function. Null values are replaced with empty strings. The considered features are then extracted which are TCP flags, protocol, packet interarrival time, time interval between first and last packet in a flow, packet length and sequence numbers. The window is set to 10 packets and after these flow metrics are extracted, the window slides by 2 packets to continue extraction until it exhausts the entire dataset. The flow data is then created which will be used in the training phase. Figure 6.5 shows the output of the created flow data. The rest of the columns carry the dataset label and protocol counts.



Unnamed: 0	pkt_start	pkt_end	flow_dur	ave_pack_IAT	count_tcp_flags	count_syn_flag	count_ack_flag	count_fin_flag	count_rst_flag
0	0	9	15.811	1.756778	1	0	0	0	0
1	2	11	1.510	0.167778	1	0	0	0	0
2	4	13	1.510	0.167778	1	0	0	0	0
3	6	15	2.228	0.247556	1	0	0	0	0
4	8	17	2.228	0.247556	1	0	0	0	0
...	...	...	...	...	...	...	...	...	...
456816	284816	284825	0.000	0.000000	1	0	0	0	0
456817	284818	284827	0.000	0.000000	1	0	0	0	0
456818	284820	284827	0.000	0.000000	1	0	0	0	0
456819	284822	284827	0.000	0.000000	1	0	0	0	0
456820	284824	284827	0.000	0.000000	1	0	0	0	0

Figure 6.5 Flow data output

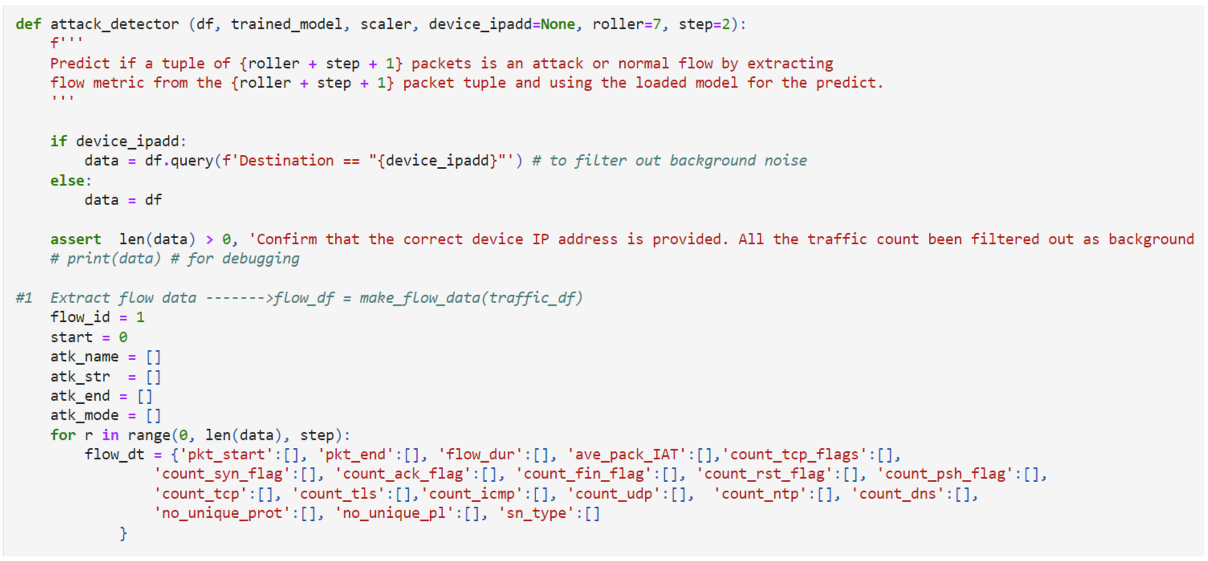
- Train model: This phase involves scaling the feature variables using the standard scaler from scikit-learn, splitting the data using train_test_split function, training the Random Forest classifier, testing, and saving the model for future use. Figure 6.6 shows the code responsible for these steps.



```python
def make_attack_model (flow_data, scale=False, plot_eval=True):
    #2 Preprocess data
    y = flow_data['label']
    X = flow_data.drop(columns = ['pkt_start','pkt_end', 'label'])  # drop dummy column and the actual label
    # X['av_sn'].replace([np.nan], -1, inplace=True)   # encode flows with no average sequece number (nan) with -1

    #3 create scaler and scale the data
    scaler = StandardScaler().fit(X)
    if scale:
        X = scale_data(X, scaler)

    #4 modeling
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=10, shuffle=True, stratify=y)
    model = RandomForestClassifier(random_state=10).fit(X_train, y_train)
    y_pred = model.predict(X_test)

    #5 evaluate model
    print_model_eval(y_test, y_pred)
    if plot_eval:
        # visualiozation
        pca_le = get_pca(X_test, scale=False) if scale else get_pca(X_test) # PCA
        plot_pred_label (pca_le, y_pred, title = 'Label-encoding') # predicted label
        # plot_true_label(pca_le, y_test, 'Label-encoding')  # true label

    return model, scaler

model, scaler = make_attack_model(flow_data, scale=True) # for use during prediction

# save model and scaler
pickle.dump(scaler, open('../models/scaler_17_11', 'wb'))
pickle.dump(model, open('../models/RF_model_17_11', 'wb'))
```

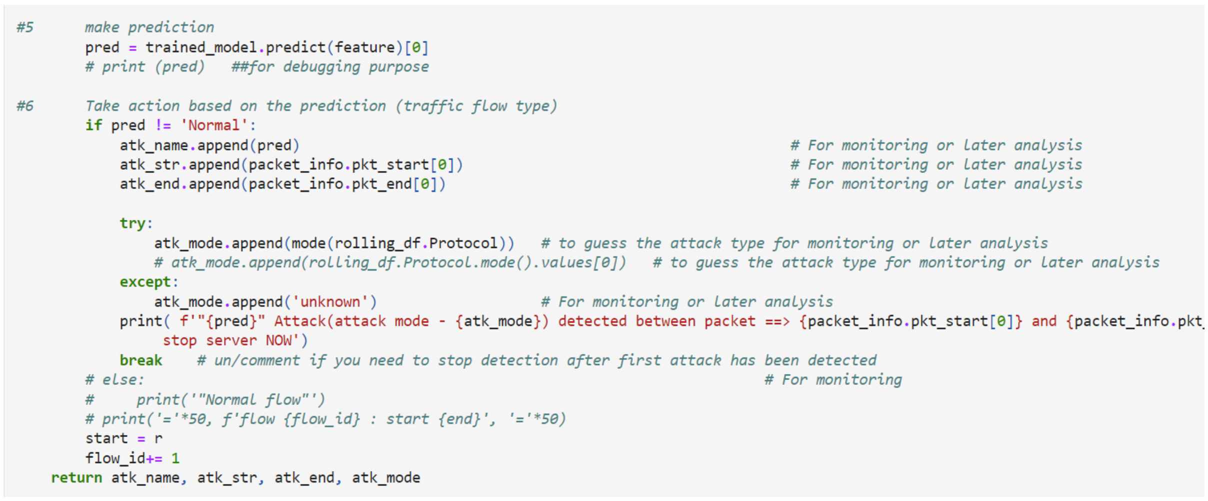
Figure 6.6 Model training

- Prediction: This phase involves filtering out the background noise, extract features from a rolling chunk of the data, and predict if that chuck is an attack or not. Once an attack is detected, the packet number of the start and end of the chunk is returned together with the attack type. Figure 6.7 shows the code responsible for background noise filter and feature extraction while figure 6.8 shows the code handling attack prediction, labelling and attack type indication using the highest protocol count (mode function) among the attack labelled packets. The output shows what the Random Forest model predicted as attack type and what the secondary classifier using the protocol mode predicts.
- Evaluation: The trained Random Forest model is evaluated using confusion matrix. This shows the precision, recall and f1score. This is shown in figure 6.9. Looking at the confusion matrix in all scenarios the model correctly predicted all the attacks as attack. It also correctly identified all the normal data as normal. So, in terms of binary classification, it works well. However, it misclassified some of the attacks like where it identified 23 UDP attacks as ICMP. I202 ICMP as TCPSYN. This is where the strength of the Hybrid model will come in as it will help the RF classify the attack type correctly.



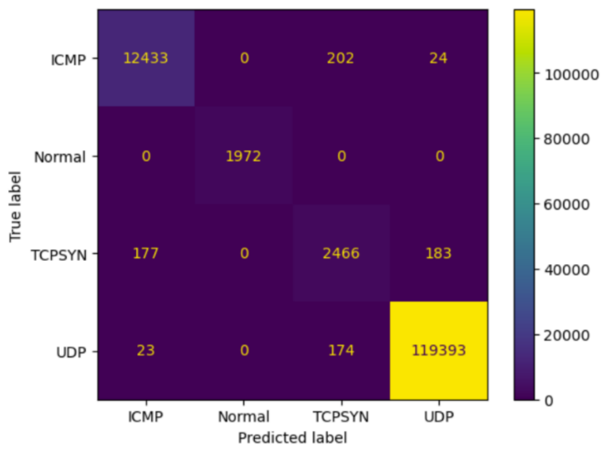
```
def attack_detector (df, trained_model, scaler, device_ipadd=None, roller=7, step=2):
    f'''
    Predict if a tuple of {roller + step + 1} packets is an attack or normal flow by extracting
    flow metric from the {roller + step + 1} packet tuple and using the loaded model for the predict.
    '''

    if device_ipadd:
        data = df.query(f'Destination == "{device_ipadd}"') # to filter out background noise
    else:
        data = df

    assert  len(data) > 0, 'Confirm that the correct device IP address is provided. All the traffic count been filtered out as background
    # print(data) # for debugging

#1  Extract flow data ------->flow_df = make_flow_data(traffic_df)
    flow_id = 1
    start = 0
    atk_name = []
    atk_str  = []
    atk_end = []
    atk_mode = []
    for r in range(0, len(data), step):
        flow_dt = {'pkt_start':[], 'pkt_end':[], 'flow_dur':[], 'ave_pack_IAT':[],'count_tcp_flags':[],
                'count_syn_flag':[], 'count_ack_flag':[], 'count_fin_flag':[], 'count_rst_flag':[], 'count_psh_flag':[],
                'count_tcp':[], 'count_tls':[],'count_icmp':[], 'count_udp':[],  'count_ntp':[], 'count_dns':[],
                'no_unique_prot':[], 'no_unique_pl':[], 'sn_type':[]
            }
```

Figure 6.7 Filtration and feature extraction



```
#5      make prediction
        pred = trained_model.predict(feature)[0]
        # print (pred)   ##for debugging purpose

#6      Take action based on the prediction (traffic flow type)
        if pred != 'Normal':
            atk_name.append(pred)                                              # For monitoring or later analysis
            atk_str.append(packet_info.pkt_start[0])                           # For monitoring or later analysis
            atk_end.append(packet_info.pkt_end[0])                             # For monitoring or later analysis

            try:
                atk_mode.append(mode(rolling_df.Protocol))   # to guess the attack type for monitoring or later analysis
                # atk_mode.append(rolling_df.Protocol.mode().values[0])   # to guess the attack type for monitoring or later analysis
            except:
                atk_mode.append('unknown')                   # For monitoring or later analysis
            print( f'"{pred}" Attack(attack mode - {atk_mode}) detected between packet ==> {packet_info.pkt_start[0]} and {packet_info.pkt
                 stop server NOW')
            break    # un/comment if you need to stop detection after first attack has been detected
        # else:                                                              # For monitoring
        #     print('"Normal flow"')
        # print('='*50, f'flow {flow_id} : start {end}', '='*50)
        start = r
        flow_id+= 1
    return atk_name, atk_str, atk_end, atk_mode
```

Figure 6.8 Prediction and labelling



```
             ICMP  Normal  TCPSYN        UDP
precision   0.984     1.0   0.868      0.998
recall      0.982     1.0   0.873      0.998
fscore      0.983     1.0   0.870      0.998
support  12659.000 1972.0 2826.000 119590.000
```

Figure 6.9 Random Forest confusion matrix

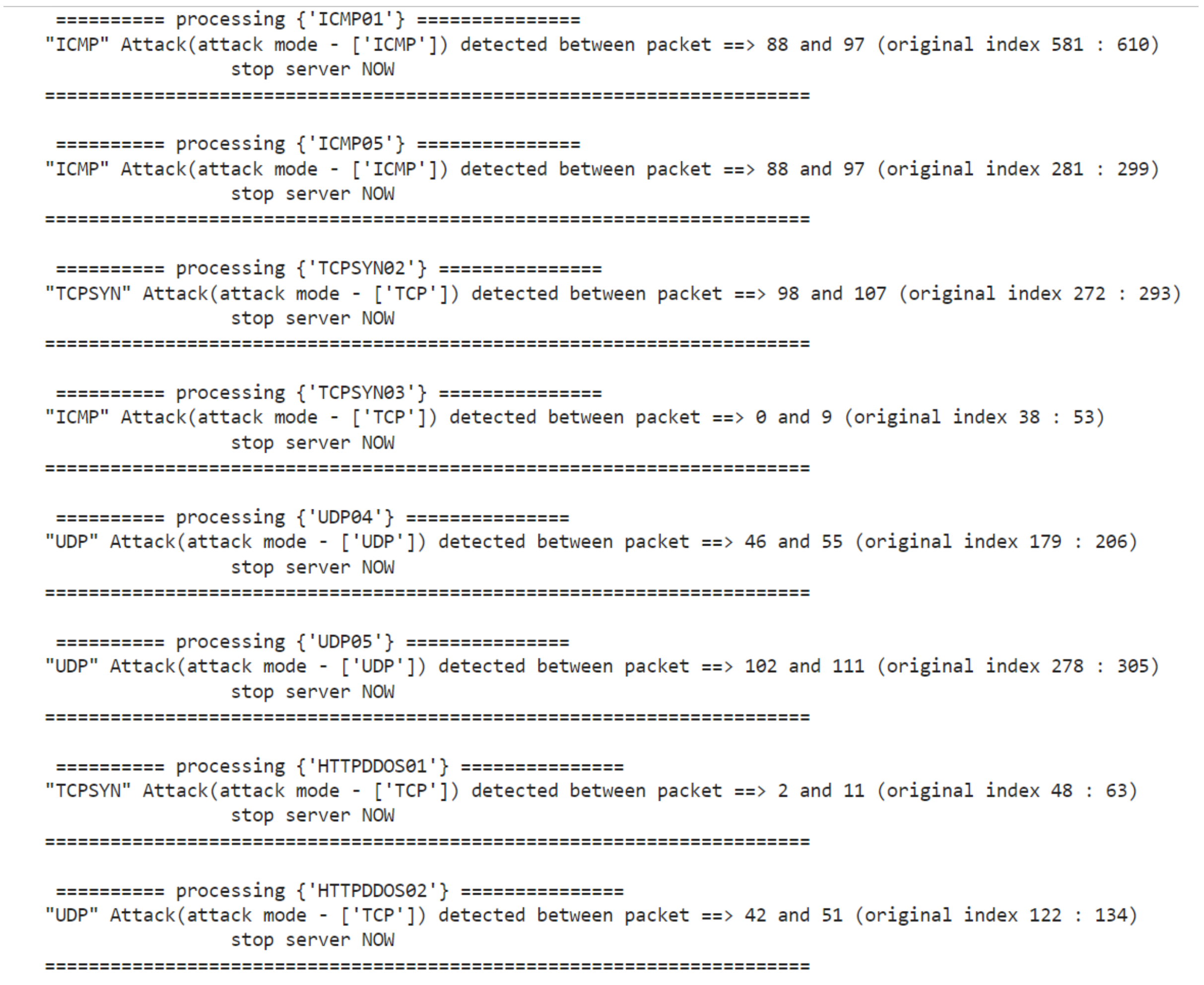
6.4.4 Model testing

This phase involves testing the trained model on the test partition of the initially split dataset. This data is unseen by the model and comprises of unfamiliar attacks and mixed attacks. Table 6.2 shows the testing results achieved from both the Random Forest classifier and the Hybrid Random Forest model.

Table 6.2 Testing results

ID	Dataset	Attack present?	Attack detected ?	Attack classified by RF	Attack classified by hybrid RF	Pkt no attack started	Window range attack detected
1	HTTP01.csv	Yes	Yes	TCPSYN	TCP	47	48-63
2	HTTP02.csv	Yes	Yes	UDP	TCP	124	122-134
3	HTTP03.csv	Yes	Yes	TCPSYN	TCP	189	189-198
4	SLOWLOIC01.csv	Yes	Yes	TCPSYN	TCP	209	209-228
5	SLOWLOIC02.csv	Yes	Yes	UDP	TCP	312	309-322
6	RECOIL01.csv	Yes	Yes	TCPSYN	TCP	192	211-232
7	UDP04.csv	Yes	Yes	UDP	UDP	202	179-206
8	UDP05.csv	Yes	Yes	UDP	UDP	297	278-305
9	TCPSYN02.csv	Yes	Yes	TCPSYN	TCP	272	272-293
10	TCPSYN03.csv	Yes	Yes	ICMP	TCP	38	38-53
11	Mixed001.csv	Yes	Yes	UDP	TCP	283	265-324
12	Mixed002.csv	Yes	Yes	ICMP	ICMP	120	72-130
13	Mixed003.csv	Yes	Yes	UDP	TCP	204	197-237
14	ICMP01.csv	Yes	Yes	ICMP	ICMP	598	581-610
	ICMP05.csv	Yes	Yes	ICMP	ICMP	293	281-299
14	Normal001.csv	No	No	No	Null	Null	Null

From table 6.2 we can see that the hybrid Random Forest model has outperformed the Random Forest model in terms of attack type classification. The hybrid RF has classified all attack types correctly with no false positives. However, the Random Forest on its own classified the attacks wrong in 4 instances which have been highlighted in red. For the mixed attacks, the Hybrid RF takes the first attack protocol that appears in the mixed attack traffic as it is the predominant one at that point. The hybrid RF has classified even the unfamiliar attacks with high accuracy. On the other hand, the Random Forest was able to detect all attacks at the very onset as seen from the table. It detects at the very first window and sometimes at the very first packet too. This shows its excellent performance in detection. This proves that a hybrid model is strong in terms of covering the detection and classification aspects as each has where its strength lies. Figure 6.10 shows the raw result output.



```
 ========== processing {'ICMP01'} ===============
"ICMP" Attack(attack mode - ['ICMP']) detected between packet ==> 88 and 97 (original index 581 : 610)
                 stop server NOW
======================================================================

 ========== processing {'ICMP05'} ===============
"ICMP" Attack(attack mode - ['ICMP']) detected between packet ==> 88 and 97 (original index 281 : 299)
                 stop server NOW
======================================================================

 ========== processing {'TCPSYN02'} ===============
"TCPSYN" Attack(attack mode - ['TCP']) detected between packet ==> 98 and 107 (original index 272 : 293)
                 stop server NOW
======================================================================

 ========== processing {'TCPSYN03'} ===============
"ICMP" Attack(attack mode - ['TCP']) detected between packet ==> 0 and 9 (original index 38 : 53)
                 stop server NOW
======================================================================

 ========== processing {'UDP04'} ===============
"UDP" Attack(attack mode - ['UDP']) detected between packet ==> 46 and 55 (original index 179 : 206)
                 stop server NOW
======================================================================

 ========== processing {'UDP05'} ===============
"UDP" Attack(attack mode - ['UDP']) detected between packet ==> 102 and 111 (original index 278 : 305)
                 stop server NOW
======================================================================

 ========== processing {'HTTPDDOS01'} ===============
"TCPSYN" Attack(attack mode - ['TCP']) detected between packet ==> 2 and 11 (original index 48 : 63)
                 stop server NOW
======================================================================

 ========== processing {'HTTPDDOS02'} ===============
"UDP" Attack(attack mode - ['TCP']) detected between packet ==> 42 and 51 (original index 122 : 134)
                 stop server NOW
======================================================================
```

Figure 6.10 Raw result output

6.4.5 Tuning

No tuning was carried out as the intended result was achieved. The Random Forest proved strong in detection while the hybrid Random Forest proved even stronger in classification. This dropped the need for tuning as there was no observation of concern.

6.4.6 Validation

This phase tests the Random Forest and the hybrid Random Forest on Public data to eliminate biases and prove that the solution is not user, device or attack centric. Table 6.3 presents the results.

Table 6.3 Validation results

ID	**Dataset**	**Attack present?**	**Attack detected ?**	**Attack classified by RF**	**Attack classified by hybrid RF**	**Pkt no attack started**	**Window range attack detected**
1	Normal002.csv	No	No	No	No	None	None
2	Normal002.csv	No	No	No	No	None	None

3	Normal002.csv	No	No	No	No	None	None
4	Normal002.csv	No	No	No	No	None	None
5	SLOWLOIC02.csv	No	No	No	No	None	None
6	Normal002.csv	No	No	No	No	None	None
7	Normal002.csv	No	Yes	ICMP	ICMP	None	1-10
8	UDP005.csv	Yes	Yes	UDP	UDP	11462	11463-11472
9	TCPSYN001.csv	Yes	Yes	ICMP	TCP	11549	11550-11559

The validation results from table 6.3 also show similar performance to the preceding test results. The hybrid Random Forest has outperformed the Random Forest in attack type classification. We can see that the Random Forest misclassified the TCPSYN attack as ICMP while the hybrid Random Forest classified it correctly as TCP. Nevertheless, the Random Forest showed excellent performance in attack detection at onset. There is a case of false positive highlighted in red. The same False Positive alarm was experienced on the same Normal dataset in the section 5.4.5 of chapter 5. This is due to the same reason as explained in the Final testing section (5.4.5) of chapter 5. The device in question was newly deployed and was establishing connection with its surroundings thereby sending and receiving pings which got mistook as ICMP attack. This tends to be a limitation in the system as accommodation for new devices has not been catered for. Nevertheless, this proves the hybrid Random Forest model to be effective regardless of the environment it is tested in as it has done well on public data as well as private.

6.5 Comparison to literature and new findings

Several literature sources have been consulted for performance comparison. However, this turned out to be difficult due to the nature of the Proposed hybrid RF model. These reasons are as follows:

- Most existing Supervised ML solutions applied in the smart home network domain in DDoS detection [96] [97] [98] [99] are purely ML based and so this cannot be fairly compared with the proposed hybrid based Random Forest model.
- The existing hybrid models [103] [105] tend to combine supervised and unsupervised Machine learning techniques as opposed to this proposed hybrid model that combines Supervised ML model and domain knowledge-based classifier.
- The method of performance assessment in existing works [92] [93] [94] [95] [96] is different from the one used for the proposed approach. This is because the proposed approach is more

concerned with attack detection and attack type identification at the very onset and maintaining accuracy while existing works assess performance based on how many times their system detected correctly without looking into how early it detects.

- Some existing works tend to carry out general attack type classification, like categorizing DDoS, MITM, and replay attacks [104] [107]. On the other hand, this work is concerned with DDoS attack type classification and no other attacks outside its scope.

Due to the above-mentioned reasons a fair and straight forward comparison was not achieved. Nevertheless, the proposed hybrid RF model stands out in several aspects as follows:

- It can detect and classify unfamiliar attacks which existing models have not been able to.
- It covers a wider range of DDoS flooding attacks in terms of detection and classification as opposed to existing models that cover about 3-4 at a time as seen from chapter 2.
- It has proven to detect and identify attack types at the very onset using the right metrics.
- It has no False Positive alarms which is the first of its kind among existing works.
- It uses metrics that point out how early and accurate it can detect and classify as opposed to existing works that give other statistics which don't say much with regards to onset detection and classification.
- It capitalizes on using absent or missing values as detection features as opposed to some existing solutions that drop or replace missing values with the most common surrounding value. A common practice in ML pre-processing is to drop or replace absent values.

Table 6.4 compares this hybrid RF with some works from literature.

Works	**Model and accuracy**
[96]	RF (85.9) Ada Boost (86.6) DT (83.8)
[97]	RF (99.68) DT (99.68) GB (99.59)
[99]	RF (98)
[102]	LSTM (98.9) CNN (99.9)
This work	RF (99)

6.6 Summary

This chapter has trained and tested the performance of two models when it comes to attack detection and attack type indication. A Random Forest model and a hybrid Random Forest model using domain knowledge for attack type indication were compared. The Hybrid version outperformed the independent Random Forest model in attack type indication. However, the independent Random Forest has proven to perform excellent with high accuracy in attack detection at the very onset. On the other hand, the hybrid model had the same accuracy in attack type indication. This has given rise to a more robust version due to the application of domain knowledge as each has where its strength lies. This proposed hybrid model has also proven to work perfectly on unfamiliar attacks and has been validated using public data achieving 99 % accuracy.

Chapter 7

Research Contributions

7.1 Overall contributions

The respective contributions made by this research are presented here. These contributions will be linked to the various research questions raised at the beginning of this work. The respective gaps bridged will be discussed and the strategy applied that resulted in each of the systems benefits will be outlined.

- ✓ **Contribution 1**: In the event of studying the smart home network behaviour and traffic patterns, unique traffic patterns attributed to each mode of device control was discovered. This discovery is new with regards to deriving a unique signature for each method or mode used to control the smart devices. The explored modes include manually operating the devices, automated/ scheduled, using Hive app, using Home kit app and using Google home app. The protocol and packet length sequence of each mode of control was found to be unique and uniform regardless of the platform (iPhone, iPad, Samsung smart phone) used to control it. These new findings can be used in forensic investigations to prove how someone controlled a particular device or devices and whether they were present at the scene during some specified times. For instance, if the evidence shows proof of manual mode of operation, then this ties one to physically being at the premises. Furthermore, as each operation mode has a unique traffic pattern (section 3.5.4 and 3.5.5), these patterns could be whitelisted on the smart home network to detect certain attacks relating to unauthorized control of device which might have a deviating pattern from the whitelisted ones. This contribution does not answer any of the research questions raised in chapter 1 as it was an unplanned discovery. Table 7.1 shows how the contribution was achieved. **This contribution is addressed in chapter 3 section 3.5.4 and 3.5.5**

ID	Discovery	How it was achieved
1	Distinct mode of operation for devices	Distinct protocol & packet length sequence (3.5.5)
2	Distinct mode of operation for devices	Distinct flow volume & duration (3.5.4)

Table 7.1 How contribution 1 was achieved

- ✓ **Contribution 2**: Normal smart home traffic pattern in comparison to when DDoS flooding attacks infiltrate the network are visualized using Exploratory Data Analysis in section 4.4 of chapter 4. This visualization is new as it clearly visualizes the benign and attack patterns based on smart home network features that get simultaneously affected during an attack. The visualized network features can be incorporated into data visualisation tools and Intrusion Detection Systems. This will provide clearer low-level statistics as to how the network is deviating from its normal pattern during an attack. Table 7.2 presents how this contribution was achieved. This contribution covers **RQ1, RQ2 AND RQ3**.**This is addressed in chapter 4 (section 4.4)**.

ID	Discovery	How it was achieved
1	Attack traffic pattern	EDA on protocol, packet length, sequence numbers, TCP flags (section 4.4)
2	Benign traffic pattern	EDA on protocol, packet length, sequence numbers, TCP flags (section 4.4)

Table 7.2 How contribution 2 was achieved

- ✓ **Contribution 3**: A new approach to DDoS attack detection has been presented. The approach uses feature absence and feature range in attack detection from the very onset. Some prominent network features (Sequence numbers and TCP flags) were found to be absent for the duration of certain attacks. The narrative needs to be changed from only focusing on present network feature statistics to detect attacks, rather features that are normally present but tend to be absent for a prolonged period also contribute to rapid attack detection as seen in this research. In addition to that, the sequence number range in normal traffic tends to be very wide, starting with a 0 or 1 at the beginning of a session and keeps incrementing to very high values. However, during an attack, the sequence numbers were found to stall at 0 or 1 all through. This new finding led to contribution 4. Table 7.3 shows how this contribution was achieved. This covers **RQ3**. **This is addressed in chapter 4 (section 4.5).**

ID	Discovery	How it was achieved
1	Proposed detection approach	Feature variance, absence, and range (Section 4.5)

Table 7.3 How contribution 3 was achieved

- ✓ **Contribution 4**: A hybrid anomaly and feature-based DDoS detection and attack type indication algorithm has been implemented and tested. This algorithm is based on the findings in contribution 3. After a grouped series of packets are flagged and labelled as attack, the protocol with the highest count among those flagged packets is used as the attack type indication label for each of the attack labelled packets. Both detection and attack type indication modules of the system performed excellently while always detecting and indicating the attack type at the very onset. In addition to that the solution is light weight, practical, centralized, and counter spoof that is not user, attack nor device centric covering unfamiliar and mixed attacks. Table 7.4 shows how this contribution was achieved. This covers **RQ4, RQ5, RQ6. This is addressed in chapter 5.**

ID	Discovery	How it was achieved
1	Light weight	2 features (sequence number & TCP flags), one way traffic monitoring
2	Not user, device or attack centric	Using smart home general characteristics and general attack signatures
3	Network level coverage	Monitoring at gateway (router)

4	Unfamiliar attacks	General DDoS attack signatures used
5	Onset detection	Setting detection threshold to 10 consecutive packets
6	Covers all DDoS attacks	General DDoS attack signatures used
7	Counter spoof	Avoided using spoof prone features like IP address and port numbers
8	Practical	No training needed, no single packet inspection,
9	Not biased	Validated using public data
10	Reliable	Tested using relevant metrics
11	Accurate attack type indication	Using highest protocol count among attack labelled packets

Table 7.4 How contribution 4 was achieved

- ✓ **Contribution 5**: A hybrid Machine learning detection and attack type indication model is developed. The Random Forest model is trained based on the same network features used in contribution 4. The model was able to accurately detect the attack at the very onset and to some extent classify the attack type. However, the novel attack type indication approach used in contribution 4 which is based on highest protocol count among the attack labelled packets was applied to the RF model. After the RF model detects the attack, the indication module applies the highest protocol count check and labels the attack type using that. This hybrid model outperformed the RF's ability to classify the attack type correctly including unfamiliar attacks. In all the testing and validation cases, the hybrid model performed better in indicating the attack type while the RF model on its own misclassified the attack type a couple of times. This proves that the hybrid model is more effective in terms of attack type indication. Table 7.5 shows how this contribution was achieved. This covers **RQ7. This is addressed in chapter 6**.

ID	**Discovery**	**How it was achieved**
1	Light weight	4 features (seq no, TCP flags, protocol, packet length),1 way traffic monitoring
2	Not user, device or attack centric	Using smart home general characteristics and general attack signatures
3	Network level coverage	Monitoring at gateway (router)
4	Unfamiliar attacks	Used features that are highly affected during attack. (Domain knowledge)
5	Onset detection	Feature choice (domain knowledge), sliding window by 2 packets, RF model
6	Covers all DDoS attacks	Used features that are highly affected during attack. (Domain knowledge)
7	Counter spoof	Avoided using spoof prone features like IP address and port numbers
8	High accuracy in detection and attack type indication	Hybrid model and domain knowledge
9	Not biased	Validated using public data
10	Reliable	Tested using relevant metrics
11	Accurate attack type indication	Highest protocol count among attack labelled packets (domain knowledge)

Table 7.5 How contribution 5 was achieved

- ✓ **Contribution 6**: A new approach to assessing the performance of a DDoS attack detection and attack type indication system is presented. This new approach is proven to be more relevant in terms of precisely measuring the system's ability to detect and classify attacks at the very onset and how accurate the prediction is. Currently the conventional method is the use of confusion matrix and other statistics. However, confusion matrix does not specify how early the attack is detected or classified rather it gives statistics on how much the solution was able to predict right. This proposed approach is used in this research and has proven to provide more relevant performance details. The metrics used to gauge the performance of the detection and attack type indication system on each data source in this new approach are: Is attack present, type of attack present, is attack detected, packet number attack started, packet number attack detected, packet number attack classified, attack type classified, window attack started, window attack detected. Table 7.6 shows how this contribution was achieved. This covers **RQ8. This is addressed in chapter 5 section 5.4.3 and 6 section 6.4.5.**

ID	Discovery	How it was achieved
1	Onset attack detection	Packet no attack started, packet no attack detected
2	Onset attack type indication	Packet no attack started, packet no attack type indicated
3	Accurate attack indicated	Attack type present, Attack type indicated

Table 7.6 How contribution 6 was achieved

7.4 Summary

This chapter has laid out the various contributions made by this research and linked them to the respective research questions they address. Furthermore, it has identified the various approaches used to achieve the gaps bridged by the contributions.

Chapter 8

Conclusion

8.1 Introduction

This research has shown how effective data visualization is in terms of studying and identifying attack patterns. By employing this method via EDA, smart home traffic properties that get highly affected during DDoS attacks have been identified. This was used to produce a robust, light weight detection and attack type indication solution that is not user, attack, or device centric. The implemented and tested system has achieved excellent results even on unfamiliar attacks which have proven to be difficult to tackle by existing solutions. This is due to using generalized smart home traffic properties and generalized attack signatures which gave rise to a hybrid anomaly and feature-based detection and attack type indication system.

The relevance of using feature absence cannot be underestimated as this has been discovered to highly contribute to attack detection at the very onset. The same goes for feature range. The narrative needs to be changed from only focusing on present network feature statistics to detect attacks, rather features that are normally present but tend to be absent for a prolonged period also contribute to rapid attack detection as seen in this research.

This research has also proven how powerful some network features are, on their own in terms of attack detection. On the other hand, we have also seen how some conventionally used features by existing works lead to high false positive rate. This brings us back to the importance of data visualization to understand traffic patterns.

A hybrid Supervised Machine Learning model has also been developed that has performed excellently well with no false positive rates, which is rare among exiting solutions. Furthermore, it can detect and classify unfamiliar attacks from the very onset which says a lot about its robustness. This is due to its hybrid nature of using a Supervised ML model coupled with a domain knowledge-based attack type indicator. In addition to that the features used in training the model were selected purposely due to the prospects they showed right from the EDA phase. From this, we can see how relevant the application of domain knowledge is when designing Machine Learning models for attack detection.

A more effective method of assessing the performance of DDoS attack detection systems has also been presented which tends to give more useful details in terms of a system's ability to detect and classify attacks at the very onset accurately. This should change the narrative from using conventional methods like confusion matrices and other statistics in terms of performance assessment as they do not provide relevant information as to how early and accurate the system was able to detect or classify an attack which is very crucial in the field of DDoS attacks. The effect is what you want to mitigate as early as possible not how much the attack is guessed right down the line when much of the damage has been done. This proposed method of assessment will help give birth to more reliable and robust solutions.

8.2 Areas of improvement and future work

The hybrid signature and anomaly-based system has some limitations, which will be part of the future work for improvement. The system is not able to detect or classify low stealth attacks as it mainly capitalizes on the static nature of most DDoS attack traffic, which low stealth attacks deviate from. Future work is looking at using the EDA technique to study the attack pattern of these low stealth attacks better and come up with a way to integrate a detection mechanism for them in the current system.

The system also uses an attack threshold of 10 packets to determine an attack pattern. If this threshold is not reached the attack will be missed. However, this highly unlikely as the nature of DDoS flooding attacks is too overwhelming for it to be missed due to this fixed threshold.

The visualised EDA images can be trained on a convolutional Neural Network (CNN) using ResNet. More so, attack type binary classification can be further carried out based on the predominant protocol derived from the statistics. It is well known that deep learning models especially CNN achieved high significance due to their outstanding performance in the image processing field. The potential of CNN can be used to detect DDoS attacks by converting the network traffic data into images. This an area of interest for future work.

Bibliography

[1] K. Kostas, M. Just and M. A. Lones, "IoTDevID: A Behaviour-Based Fingerprinting Method for Device Identification in the IoT," arXiv Preprint arXiv: 2102.08866, 2021.

[2] S. Notra, M. Siddiqi, H. H. Gharakheili, V. Sivaraman and R. Boreli, "An experimental study of security and privacy risks with emerging household appliances," in 2014 IEEE Conference on Communications and Network Security, pp. 79-84, 2014.

[3] F. Loi, A. Sivanathan, H. H. Gharakheili, A. Radford and V. Sivaraman, "Systematically evaluating security and privacy for consumer IoT devices," in Proceedings of the 2017 Workshop on Internet of Things Security and Privacy, pp. 1-6, 2017.

[4] L. Andrea, C. Chrysostomou and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in 2015 IEEE Symposium on Computers and Communication (ISCC), pp. 180-187, 2015

[5] K. Moskvitch, "Securing IoT: In your smart home and your connected enterprise," Engineering & Technology, 12(3), pp. 40-42, 2017.

[6] N. Dhanjani, "Abusing the Internet of Things: Blackouts, Freakouts, and Stakeouts. " O'Reilly Media, Inc.", 2015.

[7] J. Fernandes and A. Prakash, "Security analysis of emerging smart home applications," in 2016 IEEE Symposium on Security and Privacy (SP), pp. 636-654, 2016

[8] N. Chaabouni, M. Mosbah, A. Zemmari, C. Sauvignac and P. Faruki, "Network intrusion detection for IoT security based on learning techniques," IEEE Communications Surveys & Tutorials, 21(3), pp. 2671-2701, 2019.

[9] F. Hussain, R. Hussain, S. A. Hassan and E. Hossain, "Machine learning in IoT security: Current solutions and future challenges," IEEE Communications Surveys & Tutorials, 22(3), pp. 1686-1721, 2020.

[10] B. Bezawada, M. Bachani, J. Peterson, H. Shirazi, I. Ray and I. Ray, "Behavioral fingerprinting of iot devices," in Proceedings of the 2018 Workshop on Attacks and Solutions in Hardware Security, pp. 41-50, 2018.

[11] O. Kupreev, E. Badovskaya, A. Gutnikov, "DDoS attacks in Q1 2020", Kaspersky DDoS reports, 2020 [Online]. https://securelist.com/ddos-attacks-in-q1-2020/96837/ [Accessed: 01- Jan- 2022].

[12] K. Brush, E. Burns, "Data Visualization", Techtarget Business Analytics, 2022 [Online]. https://www.techtarget.com/searchbusinessanalytics/definition/data-visualization [Accessed 03-Nov-2023].

[13] H. Huang, J. Chu and X. Cheng, "Trend analysis and countermeasure research of DDoS attack under 5G network", In 2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP), pp. 153-160, 2021.

[14] C. Wu, S. Sheng and X. Dong, "Research on visualization systems for DDoS attack detection", In 2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 2986-2991, 2018.

[15] M. Cinque, D. Cotroneo, and A. Pecchia, “Challenges and Directions in Security Information and Event Management (SIEM),” In 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), Memphis, USA, pp. 95–99, 2018.

[16] J. Miranda-Calle, V. Reddy, P. Dhawan and P. Churi, “Exploratory data analysis for cybersecurity”. World Journal of Engineering, 2021.

[17] E. Anthi, L. Williams, M. Słowińska, G. Theodorakopoulos and P. Burnap, “A supervised intrusion detection system for smart home IoT devices” IEEE Internet of Things Journal, 6(5), pp.9042-9053, 2019.

[18] Q. Niyaz, W. Sun and A. Javaid, “A deep learning based DDoS detection system in software-defined networking (SDN)”. arXiv preprint arXiv:1611.07400, 2016.

[19] M. Yamauchi, Y. Ohsita, M. Murata, K. Ueda and Y. Kato, “Anomaly detection in smart home operation from user behaviors and home conditions” IEEE Transactions on Consumer Electronics, 66(2), pp.183-192, 2020.

[20] J. Wang, S. Hao, R. Wen, B. Zhang, L. Zhang, H. Hu and R. Lu, "IoT-praetor: Undesired behaviors detection for IoT devices," IEEE Internet of Things Journal, vol. 8, no. 2, pp. 927-940, Feb. 2021.

[21] Y. Mon, M. Soe, C. Su, T. Tun and M. Mie, “IoT Security: “Simulation and Analysis of TCP SYN Flooded DDOS Attack using WireShark”, Transactions on Networks and Communications, 8(3), 2020.

[22] R. Vinayakumar, M. Alazab, S. Srinivasan, Q. Pham, S. Padannayil and K. Simran, “A visualized botnet detection system based deep learning for the Internet of Things networks of smart cities”, IEEE Transactions on Industry Applications, 56(4), pp.4436-4456, 2020.

[23] J. Bhayo, S. Hameed and S. Shah, “An efficient counter-based ddos attack detection framework leveraging software defined iot (sd-iot)”, IEEE Access, 8, pp. 221612-221631, 2020

[24] Literature review and focusing the research. [Online]. https://www.sagepub.com/ sites/default/files/upm-binaries/29986_Chapter3.pdf [Accessed: 10- Jan- 2022].

[25] T. Jebb, S. Parrigon, and E. Woo, “Exploratory data analysis as a foundation of inductive research”, Human Resource Management Review, Vol. 27 No. 2, available at: https://doi.org/10.1016/j.hrmr.2016.08.003, 2017.

[26] D. Miranda-Calle, V. Reddy, P. Dhawan and P. Churi, "Exploratory data analysis for cybersecurity", *World Journal of Engineering*, Vol. 18 No. 5, pp. 734-749. https://doi.org/10.1108/WJE-11-2020-0560, 2021.

[27] J. Okesola, A. Ayodele, A. Owoade, O. Adeaga, A. Oluseyi, and I. Odun-Ayo. "Software requirement in iterative SDLC model." In *Intelligent Algorithms in Software Engineering: Proceedings of the 9th Computer Science On-line Conference 2020, Volume 1 9*, pp. 26-34. Springer International Publishing, 2020.

[28] F. Hussain, S. Abbas, M. Husnain, U. Fayyaz, F. Shahzad and G. Shah, “IoT DoS and DDoS attack detection using ResNet”, In 2020 *IEEE 23rd International Multitopic Conference (INMIC)* (pp. 1-6). IEEE, 2020.

[29] R. Khan, X. Zhang, R. Kumar and E. Aboagye, “Evaluating the performance of resnet model based on image recognition”, In *Proceedings of the 2018 International Conference on Computing and Artificial Intelligence* (pp. 86-90), 2018.

[30] A. Salih and A. Abdulazeez, “Evaluation of classification algorithms for intrusion detection system: A review”, *Journal of Soft Computing and Data Mining*, *2*(1), pp.31-40, 2021.

[31] Statista internet of things: The number of connected devices worldwide 2012-2025. [Online]. https://www.statista.com/statistics/471264/ iot-number-of-connected-devices-worldwide/. [Accessed: 25-June- 2022].

[32] M.R. Alam, M. St-Hilaire, and T. Kunz, "Peer-to-peer energy trading among smart homes," Applied energy, vol. 238, pp. 1434-1443, 2019.

[33] S.S. Gill, P. Garraghan, and R. Buyya, "ROUTER: Fog enabled cloud based intelligent resource management approach for smart home IoT devices," Journal of Systems and Software, vol. 154, pp. 125-138, 2019.

[34] B. Alsinglawi, M. Elkhodr, Q.V. Nguyen, U. Gunawardana, A. Maeder, and S. Simoff, "RFID localisation for Internet of Things smart homes: a survey," arXiv preprint arXiv:1702.02311, 2017.

[35] N. Balta-Ozkan, R. Davidson, M. Bicket, and L. Whitmarsh, "The development of smart homes market in the UK," Energy, vol. 60, pp. 361-372, 2013.

[36] D. Marikyan, S. Papagiannidis, and E. Alamanos, "A systematic review of the smart home literature: A user perspective," Technological Forecasting and Social Change, vol. 138, pp. 139-154, 2019.

[37] D. Geneiatakis, I. Kounelis, R. Neisse, I. Nai-Fovino, G. Steri, and G. Baldini, “Security and privacy issues for an iot based smart home,” in 2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO). IEEE, 2017, pp. 1292–1297

[38] M. Bilal, "A review of internet of things architecture, technologies and analysis smartphone-based attacks against 3d printers," arXiv preprint arXiv:1708.04560, 2017.

[39] M. Wu, T.-J. Lu, F.-Y. Ling, J. Sun, and H.-Y. Du, "Research on the architecture of internet of things," in 2010 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE), vol. 5, pp. V5-484, IEEE, 2010.

[40] F. Alghayadh and D. Debnath, "A hybrid intrusion detection system for smart home security," in 2020 IEEE International Conference on Electro Information Technology (EIT), pp. 319-323, IEEE, July 2020.

[41] Z. K. Zhang, M. C. Y. Cho, C. Wang, C. Hsu, C. Chen, and S. Shieh, "IoT security: ongoing challenges and research opportunities," in 2014 IEEE 7th International Conference on Service-Oriented Computing and Applications, pp. 230-234, IEEE, 2014.

[42] A. Acar, H. Fereidooni, T. Abera, A.K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.R. Sadeghi, and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!," in Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, pp. 207-218, July 2020.

[43] S. Sicari, A. Rizzardi, D. Miorandi, and A. Coen-Porisini, "Securing the smart home: A real case study," Internet Technology Letters, vol. 1, no. 3, pp. e22, 2018.

[44] P. Pongle and G. Chavan, "Real time intrusion and wormhole attack detection in internet of things," International Journal of Computer Applications, vol. 121, no. 9, 2015.

[45] I. Andrea, C. Chrysostomou, and G. Hadjichristofi, "Internet of things: Security vulnerabilities and challenges," in Computers and Communication (ISCC), 2015 IEEE Symposium on, pp. 180-187, IEEE, 2015.

[46] D. Midi, A. Rullo, A. Mudgerikar, and E. Bertino, "Kalisaa system for knowledge-driven adaptable intrusion detection for the internet of things," in Distributed Computing Systems (ICDCS), 2017 IEEE 37th International Conference on, pp. 656-666, IEEE, 2017.

[47] Web baby- monitoring cameras open to hacking. [Online] https://www.bbc.co.uk/news/technology-34138480 [Accessed: 05- May- 2020].

[48] Santa hacker speaks to girl via Ring camera [Online] https://www.bbc.co.uk/news/technology-50760103 [Accessed: 05- May- 2020].

[49] Mirai Botnet: Three admit creating and running attack tool [Online] https://www.bbc.co.uk/news/technology-42342221 [Accessed: 05- May- 2020].

[50] DDoS attack that disrupted internet was largest of its kind in history, experts say [Online] https://www.theguardian.com/technology/2016/oct/26/ddos-attack-dyn-mirai-botnet [Accessed : 01-Nov-2023].

[51] M. S. Al-Abri, A. Al-Badi, and A. Al-Badi, "A Review of Blockchain-Based DDoS Mitigation Solutions," in IEEE Access, vol. 9, pp. 107925-107942, 2021, doi: 10.1109/ACCESS.2021.3107645.

[52] C. Douligeris, A. Mitrokotsa, DDoS attacks and defense mechanisms: A classification, in: Proceedings of the 3rd IEEE International Symposium on Signal Processing and Information Technology, ISSPIT 2003, ISBN: 0780382927, 2003, pp. 190–193.

[53] P. Kumari, and A. Kumar: "A comprehensive study of DDoS attacks over IoT network and their countermeasures, *Computers & Security* p.103096, 2023.

[54] D. Javaheri, S. Gorgin, J.A. Lee and M. Masdari, "Fuzzy Logic-Based DDoS Attacks and Network Traffic Anomaly Detection Methods: Classification, Overview, and Future Perspectives," Information Sciences, 2023.

[55] V. Bulavas, "Investigation of network intrusion detection using data visualization methods," *2018 59th International Scientific Conference on Information Technology and Management Science of Riga Technical University (ITMS)*, Riga, Latvia, 2018, pp. 1-6, doi: 10.1109/ITMS.2018.8552977.

[56] S. Marchal, M. Miettinen, T. D. Nguyen, A. Sadeghi and N. Asokan, "Audi "Toward autonomous iot device-type identification using periodic communication," IEEE J. Select. Areas Commun. , vol. 37, (6), pp. 1402-1412, 2019

[57] A. Sivanathan, H. H. Gharakheili, F. Loi, A. Radford, C. Wijenayake, A. Vishwanath and V. Sivaraman, "Classifying IoT devices in smart environments using network traffic characteristics," IEEE Transactions on Mobile Computing, vol. 18, (8), pp. 1745-1759, 2018.

[58] M. Mazhar and Z. Shafiq, "Characterizing smart home iot traffic in the wild," in 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation (IoTDI), 2020, pp. 203-215.

[59] Y. Amar, H. Haddadi, R. Mortier, A. Brown, J. Colley and A. Crabtree, "An analysis of home IoT network traffic and behaviour," arXiv Preprint arXiv: 1803.05368, 2018.

[60] K. Xu, Y. Wan, G. Xue and F. Wang, "Multidimensional behavioral profiling of internet-of-things in edge networks," in 2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS), 2019, pp. 1-10

[61] S. Dong, Z. Li, D. Tang, J. Chen, M. Sun and K. Zhang, "Your smart home can't keep a secret: Towards automated fingerprinting of iot traffic," in Proceedings of the 15th ACM Asia Conference on Computer and Communications Security, 2020, pp. 47-59.

[62] Y. Meidan, M. Bohadana, A. Shabtai, J. D. Guarnizo, M. Ochoa, N. O. Tippenhauer and Y. Elovici, "ProfilIoT: A machine learning approach for IoT device identification based on network traffic analysis," in Proceedings of the Symposium on Applied Computing, 2017, pp. 506-509

[63] M. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas and J. Lloret, "Network traffic classifier with convolutional and recurrent neuralnetworks for Internet of Things," IEEE Access, vol. 5, pp. 18042- 18050, 2017

[64] A. Sivanathan, D. Sherratt, H. H. Gharakheili, A. Radford, C. Wijenayake, A. Vishwanath and V. Sivaraman, "Characterizing and classifying IoT traffic in smart cities and campuses," in 2017 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2017, pp. 559-564

[65] B. Copos, K. Levitt, M. Bishop and J. Rowe, "Is anybody home? Inferring activity from smart home network traffic," in 2016 IEEE Security and Privacy Workshops (SPW), 2016, pp. 245-251.

[66] R. Trimananda, J. Varmarken, A. Markopoulou and B. Demsky, "PingPong: Packet-level signatures for smart home device events," arXiv Preprint arXiv: 1907.11797, 2019.

[67] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A. Sadeghi and S. Uluagac, "Peek-a-boo: I see your smart home activities, even encrypted!" in Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2020, pp. 207-218

[68] N. Apthorpe, D. Reisman and N. Feamster, "A smart home is no castle: Privacy vulnerabilities of encrypted iot traffic," arXiv Preprint arXiv: 1705.06805, 2017.

[69] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan and N. Feamster, "Spying on the smart home: Privacy attacks and defenses on encrypted iot traffic," arXiv Preprint arXiv: 1708.05044, 2017.

[70] T. OConnor, R. Mohamed, M. Miettinen, W. Enck, B. Reaves and A. Sadeghi, "HomeSnitch: Behavior transparency and control for smart home IoT devices," in Proceedings of the 12th Conference on Security and Privacy in Wireless and Mobile Networks, 2019, pp. 128-138.

[71] W. Zhang, Y. Meng, Y. Liu, X. Zhang, Y. Zhang and H. Zhu, "Homonit: Monitoring smart home apps from encrypted traffic," in Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, 2018, pp. 1074-1088.

[72] K. Xu, F. Wang, S. Jimenez, A. Lamontagne, J. Cummings and M. Hoikka, "Characterizing DNS Behaviors of Internet of Things in Edge Networks," IEEE Internet of Things Journal, vol. 7, (9), pp. 7991-7998, 2020.

[73] I. Đuric, B. Dusan, B. Zorica, L. Aleksandra and R. Bozidar, "Model of an intelligent smart home system based on ambient intelligence and user profiling." *Journal of Ambient Intelligence and Humanized Computing* 14, no. 5 pp. 5137-5149, 2023.

[74] B. Hammi, Z. Sherali, K. Rida and N. Jamel, "Survey on smart homes: Vulnerabilities, risks, and countermeasures." *Computers & Security* 117 (2022): 102677, 2022.

[75] R. Fekolkin, "Intrusion detection & prevention system: overview of snort & suricata." *Internet Security, A7011N, Lulea University of Technology* (2015): 1-4. 2015.

[76] H. Shaikha and W. Abdullah, "A Review of Intrusion Detection Systems," Academic Journal of Nawroz University, vol. 6, no. 3, pp. 101–105, 2017, doi: https://doi.org/10.25007/ajnu.v6n3a90.

[77] M. A. Alsheikh, M. A. Al-Qutayri, A. A. Alghamdi, and M. S. Al-Rodhaan, "Fuzzy Gaussian Mixture-based Correntropy for Host Anomaly Detection System," IEEE Access, vol. 8, pp. 174, 2020.

[78] Y. Jia, F. Zhong, A. Alrawais, B. Gong and X. Cheng, "FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks," in *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552-9562, Oct. 2020, doi: 10.1109/JIOT.2020.2993782.

[79] B. Al-Duwairi, W. Al-Kahla, M. A. AlRefai, Y. Abdelqader, A. Rawash, and R. Fahmawi, "SIEM-based detection and mitigation of IoT-botnet DDoS attacks," International Journal of Electrical and Computer Engineering, vol. 10, no. 2, pp. 2182–2191, 2020, doi: 10.11591/ijece.v10i2.pp2182-2191.

[80] M. Shurman, R. Khrais, and A. Yateem, "DoS and DDoS attack detection using deep learning and IDS," International Arab Journal of Information Technology, vol. 17, no. 4A Special Issue, pp. 655–661, 2020, doi: 10.34028/IAJIT/17/4A/10.

[81] P. Ioulianou, V. Vasileios, M. Ioannis, and L. Michael. "A signature-based intrusion detection system for the internet of things." *Information and Communication Technology,* 2018.

[82] M. H. Nasir, J. Arshad, and M. M. Khan, "Collaborative device-level botnet detection for internet of things," Computer Security, vol. 129, p. 103172, Jun. 2023, doi: 10.1016/J.COSE.2023.103172.

[83] University of Victoria, Isot botnet dataset 2010 [Online]. http://www.uvic.ca/ engineering/ece/isot/datasets/ [Accessed: 03- Mar- 2022]

[84] S. Garcia, A. Parmisano and M. Erquiaga, ''IoT-23: A labeled dataset with malicious and benign IoT network traffic (version 1.0.0),'' Zenodo, vol. 20 pp.15, doi: 10.5281/zenodo.4743746, 2020

[85] N. Koroniotis, N. Moustafa, E. Sitnikova and B. Turnbull, "Towards the development of realistic botnet dataset in the internet of things for network forensic analytics: Bot-iot dataset", Future Generation Computer Systems, vol. 100, pp. 779-796, 2019.

[86] K. Al-Begain, M. Khan, B. Alothman, C. Joumaa, and E. Alrashed, "A DDoS Detection and Prevention System for IoT Devices and Its Application to Smart Home Environment," Applied Sciences 2022, Vol. 12, Page 11853, vol. 12, no. 22, p. 11853, Nov. 2022, doi: 10.3390/APP122211853.

[87] J. Galeano, J. Carmona, J. Valenzuela and F. Luna, "Detection and mitigation of dos and ddos attacks in iot-based stateful sdn: An experimental approach", Sensors, 20(3), p.816, 2020

[88] J. Li, M. Liu, Z. Xue, X. Fan and X. He, "RTVD: A real-time volumetric detection scheme for DDoS in the Internet of Things" IEEE Access, 8, pp. 36191-36201, 2020.

[89] D. K. Sharma et al., "Anomaly detection framework to prevent DDoS attack in fog empowered IoT networks," Ad Hoc Networks, vol. 121, p. 102603, Oct. 2021, doi: 10.1016/J.ADHOC.2021.102603.

[90] H. Christoph and E. Buchmann, "Fane: a firewall appliance for the smart home." In *2019 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pp. 449-458. IEEE, 2019.

[91] S. Kumar, S. Dalal and V. Dixit, “The OSI model: Overview on the seven layers of computer networks”, *International Journal of Computer Science and Information Technology Research*, *2*(3), pp.461-466, 2014.

[92] J. Liang and K. Yoohwan, "Evolution of firewalls: Toward securer network using next generation firewall." In *2022 IEEE 12th Annual Computing and Communication Workshop and Conference (CCWC)*, pp. 0752-0759. IEEE, 2022.

[93] A. Kumar, A. Kumar, R. Muhammad, S. Achyut and C. Xiaochun, "Intrusion detection and prevention system for an IoT environment." *Digital Communications and Networks* 8, no. 4 pp. 540-551, 2022.

[94] B. Soewito and A. Charlie, "Next generation firewall for improving security in company and iot network." In *2019 International Seminar on Intelligent Technology and Its Applications (ISITIA)*, pp. 205-209. IEEE, 2019.

[95] A. Feraudo, A. Diana, Y. Poonam, M. Richard and B. Paolo, "Mitigating IoT Botnet DDos Attacks through MUD and eBPF based Traffic Filtering." *arXiv preprint arXiv:2305.02186*, 2023.

[96] P. Illy, K. Georges, K. Kuljeet and G. Sahil, "ML-based IDPS enhancement with complementary features for home IoT networks." *IEEE Transactions on Network and Service Management* 19, no. 2 pp. 772-783, 2022.

[97] L. Qaddoori and A. Qutaiba, "An embedded intrusion detection and prevention system for home area networks in advanced metering infrastructure." *IET Information Security*, 2023.

[98] B. B. Gupta, P. Chaudhary, X. Chang, and N. Nedjah, “Smart defense against distributed Denial of service attack in IoT networks using supervised learning classifiers,” *Computers & Electrical Engineering*, vol. 98, p. 107726, Mar. 2022, doi: https://doi.org/10.1016/j.compeleceng.2022.107726, 2022.

[99] H. Gordon, C. Batula, B. Tushir, B. Dezfouli and Y. Liu, "Securing Smart Homes via Software-Defined Networking and Low-Cost Traffic Classification," *2021 IEEE 45th Annual Computers, Software, and Applications Conference (COMPSAC)*, Madrid, Spain, 2021, pp. 1049-1057, doi: 10.1109/COMPSAC51774.2021.00143, 2021.

[100] R. Doshi, N. Apthorpe and N. Feamster, "Machine Learning DDoS Detection for Consumer Internet of Things Devices," *2018 IEEE Security and Privacy Workshops (SPW)*, San Francisco, CA, USA, 2018, pp. 29-35, doi: 10.1109/SPW.2018.00013, 2018.

[101] A. A. Sallam, M. N. Kabir, Y. M. Alginahi, A. Jamal and T. K. Esmeel, "IDS for Improving DDoS Attack Recognition Based on Attack Profiles and Network Traffic Features," *2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA)*, Langkawi, Malaysia, 2020, pp. 255-260, doi: 10.1109/CSPA48992.2020.9068679, 2020.

[102] Y. Jia, F. Zhong, A. Alrawais, B. Gong, and X. Cheng, “FlowGuard: An Intelligent Edge Defense Mechanism Against IoT DDoS Attacks,” *IEEE Internet of Things Journal*, vol. 7, no. 10, pp. 9552–9562, Oct. 2020, doi: https://doi.org/10.1109/jiot.2020.2993782, 2020.

[103] F. Alghayadh and D. Debnath, “A Hybrid Intrusion Detection System for Smart Home Security Based on Machine Learning and User Behavior,” *Advances in Internet of Things*, vol. 11, no. 01, pp. 10–25, 2021, doi: https://doi.org/10.4236/ait.2021.111002, 2021.

[104] E. Anthi, L. Williams, M. Slowinska, G. Theodorakopoulos, and P. Burnap, “A Supervised Intrusion Detection System for Smart Home IoT Devices,” *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9042–9053, Oct. 2019, doi: https://doi.org/10.1109/jiot.2019.2926365,2019.

[105] K. J. Singh and T. De, “Efficient Classification of DDoS Attacks Using an Ensemble Feature Selection Algorithm,” *Journal of Intelligent Systems*, vol. 0, no. 0, Dec. 2017, doi: https://doi.org/10.1515/jisys-2017-0472, 2017.

[106] M. Aamir and S. M. A. Zaidi, “Clustering based semi-supervised machine learning for DDoS attack classification,” *Journal of King Saud University - Computer and Information Sciences*, Feb. 2019, doi: https://doi.org/10.1016/j.jksuci.2019.02.003, 2019.

[107] T. Li, Z. Hong and L. Yu, "Machine Learning-based Intrusion Detection for IoT Devices in Smart Home," *2020 IEEE 16th International Conference on Control & Automation (ICCA)*, Singapore, 2020, pp. 277-282, doi: 10.1109/ICCA51439.2020.9264406, 2020.

[108] W. Li, S. Tug, W. Meng, Y. Wang Designing collaborative block chained signature-based intrusion detection in IoT environments Future Generation Computer Systems, 96 (2019), pp. 481-489, 2019.

[109] A. Qureshi, L. Hadi, M. Nhamoinesu, J. Abbas and A. Jawad, "RNN-ABC: A new swarm optimization-based technique for anomaly detection." *Computers* 8, no. 3 pp. 59, 2019.

[110] M. Ge, X. Fu, N. Syed, Z. Baig, G. Teo, A. Robles-Kelly, “Deep learning-based intrusion detection for IoT networks” 2019 IEEE 24th Pacific Rim International Symposium on Dependable Computing (PRDC), IEEE (2019), pp. 256-25609, 2019.

[111] S. Ahn, H. Yi, Y. Lee, W.R. Ha, G. Kim, Y. Paek, “Hawkware: network intrusion detection based on behaviour analysis with ANN’s on an IoT device” 57th ACM/IEEE Design Automation Conference (DAC), IEEE (2020), pp. 1-6, 2020.

[112] G. Raja, A. Ganapathisubramaniyan, G. Anand, “Intrusion detector for blockchain based IoT networks” 2018 Tenth International Conference on Advanced Computing (ICoAC), IEEE (2018), pp. 328-332, 2018.

[113] A. Diro, N. Chilamkurti, “Distributed attack detection scheme using deep learning approach for internet of things” Future Generation Computer Systems, 82 (2018), pp. 761-768, 2018.

[114] Hive Home [Online] https://www.hivehome.com/ [Accessed: 05- July- 2020].

[115] GS308E -8- Port Gigabit Ethernet Plus Switch [Online] https://www.netgear.com/support/product/gs308e [Accessed: 05- July- 2020].

[116] Jupyter Notebook [Online] https://jupyter.org/ [Accessed: 05- July- 2020].

[117] TL-WR940N [Online] https://www.tp-link.com/uk/home-networking/wifi-router/tl-wr940n/ [Accessed: 05- July- 2020].

[118] M. Laner, N. Nikaein, P. Svoboda, M. Popovic, D. Drajic and S. Krco, "Traffic models for machine-to-machine (M2M) communications: Types and applications," in Machine-to-Machine (M2M) Communications Anonymous Elsevier, pp. 133-154, 2015.

[119] K. Xu, F. Wang, S. Jimenez, A. Lamontagne, J. Cummings and M. Hoikka, "Characterizing DNS Behaviors of Internet of Things in Edge Networks," IEEE Internet of Things Journal, vol. 7, (9), pp. 7991-7998, 2020.

[120] Google Colab [Online] https://www.bing.com/ck/a?!&&p=8ad2fa96d733084eJmltdHM9MTcwMzExNjgwMCZpZ3VpZD0yMTg2YTM2ZC0xZTUxLTZkZjktMzkwZi1iMjJkMWY3NjZjNWUmaW5zaWQ9NTIwOA&ptn=3&ver=2&hsh=3&fclid=21

86a36d-1e51-6df9-390fb22d1f766c5e&psq=google+colab&u=a1aHR0cHM6Ly9jb2xhYi5y ZXNlYXJjaC5nb29nbGUuY29tLw&ntb=1 [Accessed: 02- Jan- 2021].

[121] LOIC download [Online] https://sourceforge.net/projects/loic/ [Accessed: 03- Mar- 2021].

[122] Hping3 – Network auditing, DoS and DDoS [Online] https://www.kalilinux.in/2021/03/hping3-kali-linux-dos-ddos-network.html [Accessed: 03- Nov- 2023].

[123] Kali Linux Penetration testing and ethical hacking Linux distribution [Online] https://www.kali.org/ [Accessed: 03- Nov- 2023].

[124] Wireshark [Online] https://www.wireshark.org/ [Accessed: 03- June- 20219].

[125] M. Hassan (via Mendeley Data) BUET-DDoS2020 2021 [Online]. https://doi.org/10.17632/bzgf9r36kp.2 [Accessed: 03- Mar- 2022]

[126] U. Saxena, J. Sodhi and Y. Singh, "An analysis of ddos attacks in a smart home networks", In 2020 IEEE 10th International Conference on Cloud Computing, Data Science & Engineering, pp. 272-276, 2020

[127] S. Pokhrel, R. Abbas and B. Aryal, "IoT Security: Botnet detection in IoT using Machine learning", arXiv preprint arXiv:2104.02231, 2021.

[128] LOIC's new flag help [Online] https://documentation.help/LOIC/recoil.html#:~:text=The%20ReCoil%20attack%20focuses%20on,to%20keep%20the%20socket%20alive. [Accessed: 03- Mar- 2022]

[129] MIT Lincoln Laboratory 1999 Intrusion Detection evaluation dataset [Online] https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset . [Accessed: 05- Jan- 2022]

[130] Python standard library [Online] https://data-flair.training/blogs/python-libraries/ [Accessed: 03- Mar- 2022]

[131] Sklearn.ensemble.RandomForest classifier [Online] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]

[132] Support Vector Machines [Online] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]

[133] Gradient Boosting classifier [Online] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]

[134] Sklearn.ensemble.Voting classifier [Online] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]

[135] Sklearn.ensemble.Stacking classifier [Online] http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]

[136] Sklearn.ensemble.Adaboost classifier [Online] http://scikit learn.org/stable/ modules/generated/sklearn.ensemble.RandomForestClassifier.html [Accessed: 07- Mar- 2023]