Santa Clara University

Department of Computer Science and Engineering

Date: June 8, 2023

I HEREBY RECOMMEND THAT THE THESIS PREPARED

UNDER MY SUPERVISION BY

Paul Boldyrev, Timothy Hradil, and Matthew Ding

ENTITLED

AI-Powered IoT Monitoring and Security for Smart Homes

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

FOR

THE DEGREE OF

BACHELOR OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

DocuSigned by:

Behnam Dezfouli

6ED175D6B91A4FF...

Thesis Advisor
Dr. Behnam Dezfouli

N. Ling

N. Ling (Jun 14, 2023 19:08 PDT)

Chairman of Department
Dr. Nam Ling

AI-Powered IoT Monitoring and Security for Smart Homes

By

Paul Boldyrev, Timothy Hradil, and Matthew Ding

SENIOR DESIGN PROJECT REPORT

Submitted to
the Department of Computer Science and Engineering
of

SANTA CLARA UNIVERSITY

in Partial Fulfillment of the Requirements
for the degree of
Bachelor of Science in Computer Science and Engineering

Santa Clara, California

June 8, 2023

Acknowledgments

We would like to thank our advisor, Dr. Behnam Dezfouli for his support over the past year. Your availability and willingness to help made a significant positive impact on our experience during this project.

Additionally, we would like to thank Ted Kummert for his guidance and for sharing his industry experience during the development of the project. Your emphasis on planning and execution helped us to successfully implement our project ahead of the expected timeline.

Of course, the support and encouragement of our friends and family helped tremendously in our journey towards completing this thesis. Thank you for proofreading our papers, debugging our code, and listening to us practice our presentations. In particular, we would like to thank Jordan Randleman for his mentorship during our time at Santa Clara University.

AI-Powered IoT Monitoring and Security for Smart Homes

Department of Computer Science and Engineering
Santa Clara University
2023

ABSTRACT

Smart home devices are becoming increasingly popular across the United States. Unfortunately, in order to keep the cost of these devices down, manufacturers tend to place cybersecurity measures lower on their list of priorities. As a result, these devices have become targets for security breaches like the Mirai Botnet and distributed denial of service (DDoS) attacks. The cybersecurity solutions currently available on the market are either expensive, difficult to set up, or follow a static set of rules for attack detection. In this thesis, we propose a novel approach to detecting attacks on Internet of Things (IoT) devices through the use of a forward proxy powered by a machine learning model running on a Linux machine. The proposed solution is more affordable, requires minimal configuration, and dynamically adjusts detection rules based on prior attacks. The use of a machine learning model in the solution allows us to take advantage of the regularity in IoT traffic patterns to identify and isolate potentially malicious packets. The model was trained using the IoT-23 data set, which provides a labeled set of malicious and benign packets from IoT devices. We verified the effectiveness of the approach using a testbed of various smart home devices set up in an isolated home network. These devices range in price and usage context, allowing us to ensure that the solution will be effective with the majority of smart home devices. After testing, we have determined that the model is able to successfully identify and isolate of malicious packets sent from a compromised IoT device if the device matches a device that was used in training data. Moving forward, we hope to implement the solution directly onto the router hardware using the P4 language for widespread adoption.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

C&C	Command and Control
CSV	Comma Separated Value
DDoS	Distributed Denial of Service
DNS	Domain Name System
DoS	Denial of Service
E-DDoS	Energy-oriented DDoS
IoT	Internet of Things
ISP	Internet Service Provider
MitM	Man-in-the-Middle
NTP	Network Time Protocol
TCP	Transmission Control Protocol
UDP	User Datagram Protocol

Chapter 1
Introduction

1.1 Motivation

With the rise of affordable Internet of Things (IoT) devices, concerns about device security have become increasingly common. Rapid development of IoT infrastructure has resulted in a general lack of security for the devices and networks in use. The result is a system of networks that are vulnerable to Distributed Denial-of-Service (DDoS) attacks. Without a solution to the present lack of security for IoT networks, other networks are also at risk [1].

A common type of DDoS attack is a flood attack. A well-known example of this is the Mirai Botnet attack in 2016. At its peak, the Mirai botnet controlled over 770,000 IoT devices and launched a flood attack of over 1 Tbps [2]. Since 2016, the Mirai source code has become available as open source code and many similar attacks have been launched.

SAM Seamless Network is a company that provides network security statistics on attacks in IoT networks. They tracked IoT devices and networks to discover the commonality of DoS, DDoS, brute force, DPI signature-based, phishing, and other attacks. They found that over one billion attacks took place in 2021. Nearly 900 million of these attacks were phishing attacks performed via IoT devices [3]. Even though awareness of IoT network security concerns is rising, attacks are still dangerously common and successful. SAM's primary suggestion for better IoT security is to have better general network security. This is a straightforward solution except for the lack of network security software that sufficiently compensates for IoT shortcomings. Therefore, there is a need for software that is dedicated to monitoring IoT networks and dynamically respond to incoming attacks.

1.2 Attack types on smart home devices

To further highlight the need for an effective security solution, we will examine and analyze some of the most popular types of cyberattacks. The analysis will include the underlying protocols used, how access was gained, and notable real-world occurrences of the attack type.

The DDoS attack and its energy-oriented counterpart are particularly effective in disrupting the daily service operations of a device or system with the latter attack type increasing the cost of electricity for its target [4]. These attacks are executed by routing excessive network traffic towards a particular device or system. This may be done on an individual scale, such as a laptop or a smart home device, or on a larger scale, such as a cloud service provider. Ultimately, the result is the same: high amounts of malicious traffic either overwhelm a system to cripple its performance or inflict exceedingly high energy costs upon the system's owner [4].

One popular type of DDoS attack vector is a reflection or amplification attack, also known as DRDoS attack [5]. In this scenario, the attacker spoofs a request in order to appear as if it is coming from the intended victim of the attack: the reflection part of the attack. This request is then sent to public servers - the amplification part - which return a response. These responses flood the victim's bandwidth and make it impossible for real traffic to proceed. The common protocol for the reflection part of the attack is the User Datagram Protocol (UDP) as it lacks a proper handshake between parties [5]. Transmission Control Protocol (TCP) does not allow for easy amplification as the TCP acknowledge (ACK) packet is not larger than the TCP synchronize (SYN) packets. This attack has a significant impact on the Internet Service Provider's (ISP) upstream infrastructure as it may not be designed to handle an overwhelming amount of traffic. As a result, a mitigation strategy that ISPs currently use is to simply void all incoming traffic once an attack is detected. This strategy, while beneficial to the ISP, is clearly not the ideal solution for the customer as valid traffic is also voided. In a way, this mitigation strategy accomplishes what the attacker originally intended: a complete disruption of all network traffic for a service.

One notable example of a DDoS attack vector to disrupt service is the use of the Mirai botnet to target OVHcloud, a French internet service company [6]. This was the first use of the botnet and the publication of its code has led to a spike in similar DDoS attacks. The Mirai botnet was built by attempting to brute force access into IoT devices by exploiting weak, default, and commonly used security credentials [7]. By scanning through large ranges of IP addresses and using a predefined list of 62 security credentials [8], Mirai quickly amassed between 200,000 and 300,000 infected devices [9]. In particular, Mirai spread by sending TCP SYN probes to potential victims through pseudo-random IP addresses. Upon a successful login, Mirai reported the IP and associated credentials to a report server. Importantly, Mirai also killed other processes bound to the ports TCP/22 or TCP/23 on the infected devices. This is an important pattern of behavior which can be used for detection of similar attacks [9]. The botnet was used to carry out volumetric, TCP state exhaustion, and application-layer attacks [9]. In these attacks, the smart home devices were used as amplifiers to overwhelm the targets of attacks.

The Man-in-the-Middle (MitM) attack is effective at intercepting sensitive or otherwise valuable data while it is in transit. The attacker sits between the client and the server and is able to monitor and modify the data that is exchanged [10]. This type of attack vector is particularly dangerous for IoT devices in personal homes. By gaining access to communication channels between the controller and the device, an attacker may set a thermostat to the maximum temperature, gain access to a live camera feed, or otherwise abuse a smart home at its owner's expense.

One common vector for a MitM attack is by routing traffic through a malicious server and altering the packet data. This is different from passive eavesdropping on packets in that it requires an increased response time for the packet processing to occur [11]. This feature of the attack vector is important as it allows for a routing protocol focused on detecting anomalies in round trip time of packet exchanges. The most frequently targeted communication layer of IoT devices for MitM attacks is the transport layer, which relies on TCP. Currently, one of the most novel ways to detect this attack is by enforcing routing between IoT devices and detecting the anomalies in the trusted time server [11]. This allows for active monitoring and voiding of malicious packets.

Since MitM attacks are usually done on a small scale to a targeted victim, there is little information about large scale MitM attacks on IoT devices. Certainly, none have occurred at the scale of the Mirai botnet. However, one notable event has been the MitM attack on Samsung smart fridges. Due to a lack of SSL certificate validation by the Samsung device, it was possible to spy on the unsuspecting users of the fridge using MitM attacks [12]. The connection provided access to the victim's Google services, such as Gmail calendars, which were accessible via the smart fridge [12].

1.3 Existing solutions and their shortcomings

Attempting to protect IoT devices from external attacks is not a novel idea, and solutions that accomplish similar goals are already present on the market. However, all of the currently available solutions lack the combination of features that we believe are essential to properly securing smart home devices. We will examine two of the most relevant services and highlight their shortcomings in this section.

Firewalla [13] is currently the closest competitor to the project. The actual product is a small box that is directly connected to a router, serving as a forward proxy for the home network. This box then allows the customer to set a variety of rule-based filters. In addition to simple filters, Firewalla also actively monitors the network for new device connections, scans open network ports, and detects if a user on the network has clicked a phishing/malware link. Importantly, Firewalla is a consumer device, and it requires the consumers to have extensive knowledge of their home networks for it to be utilized effectively.

Although a powerful solution for some use cases, Firewalla lacks the dynamic threat detection and focus on IoT devices that we intend to implement in our forward proxy. In particular, Firewalla uses static rules for monitoring network traffic. If an endpoint has not been registered as malicious by either the user or Firewalla, it will be allowed as a valid connection [13]. This is a significant issue when considering outgoing IoT device traffic. Malicious attacks may send data to unauthorized control servers that are unique for each attack [14]. As a result, Firewalla would not be able to prevent novel future attacks. The use of machine learning for dynamic threat detection would allow a solution to detect attacks from servers that have never been seen before (and therefore have not been identified as malicious). Lastly, Firewalla is not focused on IoT devices, as seen in their emphasis on detecting if a user interacts with a phishing link or malware online. Creating an IoT-focused solution is critical due to the unique connection patterns of IoT devices, such as repeating calls to the same endpoint [15], which are advantageous for monitoring for anomalies.

Another competitor is Glasswire [16]. Rather than a physical box that connects to a home router, Glasswire is a network monitoring software with a firewall. This method of monitoring enables Glasswire to be more thorough in securing the network traffic of the device it is running on. In addition to the same protection offered by Firewalla, Glasswire runs malware detection on files and the device OS itself [16]. Additionally, the user interface incorporates graphics to visualize network traffic history. This particular feature empowers the user to detect anomalies themselves.

Despite the thorough analysis that Glasswire provides for its host device, the service lacks monitoring features for the entire home network and IoT devices in particular. Glasswire shows what devices are connected but provides no further information about the potential malicious interactions of these devices [16]. Of course, this also means that no machine learning models are used to classify traffic as healthy or unhealthy. The lack of a direct connection to the router is also a significant downside for this service as it means that Glasswire is unable to restrict traffic from malicious or compromised devices.

Chapter 2
Related Work

In this chapter, we will examine the proposed solutions for identifying malicious traffic and the impact these methods have on the smart device industry. As each proposed solution examines a slightly different angle of examining traffic from smart devices, at the end of the review we describe how we synthesized these solutions to create our own unique approach.

2.1 Methodology

To ensure a thorough and comprehensive analysis of the available literature, we must first establish a guiding question. For this review, the guiding question that was used to analyze available literature was "what are the current solutions for traffic fingerprinting?" This guiding question is particularly helpful as it ensures that we are looking at the most recent papers, and, in particular, studies on traffic fingerprinting rather than general smart home security.

While a breadth of resources are available on this topic, it was important to ensure that all sources used were credible. To do so, we restricted our sources to articles published in peer reviewed journals. Additionally, conference articles were also included if the conference could be verified as credible by multiple other sources. Some examples of the conferences include the Annual Conference on Information Sciences and Systems and the International Conference on Trends in Electronics and Informatics. The combination of the reputable journal articles and conference articles ensured that only credible sources with reliable information were used.

The individual papers were selected if they included mentions of traffic fingerprinting in their abstract. This was done through the manual screening of articles using the IEEE Xplore database with search terms such as traffic fingerprinting, IoT network security, and IoT device in network. These search terms ensured that we found relevant articles to IoT network security and traffic fingerprinting. We then selected sources that directly acknowledged traffic fingerprinting as a method for managing IoT network security. Many of these sources included discussions of the strengths and weaknesses of the particular approach being analyzed.

2.2 History and background

The earliest identification methods for traffic were network ports. These were a fixed transmission layer which was registered by the Internet Assigned Numbers Authority (IANA) and allowed traffic to be identified by a simple port number [17]. Of course, as networks expanded, using port numbers became unfeasible. The identification protocol quickly evolved to keep up with the expanding number of devices.

The next iteration of traffic fingerprinting technology, called "deep packet inspection," focused on including short strings in the packet payload itself [18]. Those strings were isolated and compared to a known list used for identification. Although the precision of this method improved from the port signature method, it had a few shortcomings.

When used in high speed connections, deep packet inspection became inefficient and expensive. Another issue of greater significance was that the analysis of the entire payload meant that non-fingerprint-related data would be read [18]. This created a significant legal issue as the method could illegally infringe on privacy. This technique, although flawed, paved the way for packet identification and moved the technology forward from static address analysis to more dynamic interpretation of the packet itself.

A novel approach to traffic fingerprinting emerged when the application side was considered for identifying traffic. Rather than the receiver being solely responsible for the identification of traffic, the new solution proposed an identification plugin [19] responsible for identifying traffic from a particular application. This framework was particularly innovative as it suggested that each plugin would have a unique application it was responsible for. This method yielded much greater accuracy [19] in traffic identification when compared to more general solutions. However, this solution was not as flexible to implement in everyday home networking solutions as it required independent plugins for each application. While useful for certain industrial applications and certainly having a novel approach, this solution is not viable for smart home security.

2.3 Current innovations and research

This brings us to some of the modern approaches to traffic fingerprinting as it pertains to smart home devices. These approaches focus on the scalability of the solution and its impact in preventing cyberattacks. The majority of these approaches use some combination of machine learning and packet features to create effective models [20]. Since machine learning requires a large number of traffic features to be accurate, modern approaches use almost every aspect of a packet to create these models. Importantly, these approaches do not look at the packet contents, eliminating the privacy issues of earlier solutions.

A unique, effective approach to selecting features used autonomously generated smart home traffic and traffic generated after direct human interaction [21]. These two pathways for traffic generation have a drastically different packet signature, making it important to have a substantial amount of data on both. This approach defined four key attributes of the traffic: flow volume, flow duration, average flow rate, and device sleep time [21]. A probability distribution was then created to help analyze traffic patterns visually. This approach also used the port numbers, Domain Name System (DNS) queries, and Network Time Protocol (NTP) queries. Using machine learning to classify test data using this approach was 99.88 percent accurate in detecting benign IoT devices.

While the above approach focuses on traffic fingerprinting from an individual's perspective, it is important to also analyze the situation from the ISP's perspective. By shifting the security issue upstream, it may be easier to mitigate a large number of attacks using a single solution as opposed to individual solutions in every home.

Another solution accomplishes that by conducting traffic fingerprinting between the ISP and local networks [22]. This approach focused more closely on the timing of packets from individual devices, one of the most distinguishing characteristics of an IoT device. After the packet timing and other various features are extracted, they are converted into vectors for the machine learning algorithms to learn from. In addition to successfully distinguishing IoT traffic from other traffic, this solution was also able to identify the number of active IoT devices on a private network while only accessing outgoing network traffic. This is particularly significant as it may allow the ISP to isolate IoT traffic in an ongoing attack rather than simply voiding all traffic from a private network.

2.4 Problems and opportunities for improvement

From the earliest iterations of traffic fingerprinting technology, packet privacy has been a primary concern. It is difficult, if not impossible, to accurately determine a packet's origin without being able to examine the packet itself. As a result, any time a discussion on packet sniffing or traffic fingerprinting arises, privacy issues are also discussed. While current innovations focus heavily on distinguishing certain devices from others, we believe that the opportunity for improvement lies within the metrics that are used to conduct traffic fingerprinting.

A clear opportunity for improvement is a solution that allows security systems and firewalls to filter traffic without needing to examine packet payloads. As previously mentioned, one potential solution that researchers used relied on the timing of a packet [23]. Unfortunately, this solution was not as effective when compared to examining the actual payload. This was due to the fact that traffic timing could be impacted by external factors regardless of the type of traffic.

Another group of researchers proposed an evolution to the policy system that is currently used by smart home devices. In addition to using TCP or UDP for communication, a device would also carry an identity signature [24]. This identity information was encoded as a set of key-value entities that described the physical characteristics of the device. Although successful in identifying devices, this approach is not scalable. Manufacturers typically look to minimize the number of features on IoT devices and it is unlikely that they would accept the addition of another feature for identification.

We believe that the solution lies in the middle ground between device-agnostic traffic features and privacy-infringing methods.

One method for traffic fingerprinting is inspecting the time delay between the sender and the receiver. It is important to note that this method is not popular. The time delay is largely dependent on the size of a payload and the delay due to wireless communication. If the traffic fingerprinting solution is deployed on the same network it attempts to analyze, we can assume that the delay due to wireless communication is a known factor. As a result, we are left with a correlation between the payload size and the network statistics about the transaction or the payload. With this knowledge, a device identification solution [24] can be employed for future reference. Additionally, as previous research pointed out, machine learning could also be extremely useful in classifying the traffic delay to the device.

The solution described above has no concrete plan for implementation and testing. However, the unification between payload-agnostic data and a known database is the next step in device fingerprinting. Importantly, this solution to device fingerprinting ensures that the process is done in an ethical manner that makes consumer privacy a priority. The development and evolution of the traffic fingerprinting process would be the first step to creating more secure private firewalls that protect users from unwanted or malicious traffic on their network.

2.5 Ethical issues

When working with smart home systems, it is important to consider both the traditional network packets and the nature of the smart device. Unlike a laptop, a smart home device facilitates a kind of social environment within a home [25]. This becomes particularly important given that smart home devices have been proven to facilitate companionship for those living alone [25]. As a result, successfully traffic fingerprinting must ensure that the social interconnection facilitated by the device is not disrupted.

Another consideration we must keep in mind is the influence that smart home devices have on purchasing behaviors and information consumption. Consumers have been proven to use smart home devices as a significant source of information on a product and even complete their purchases using the device [26]. This makes attacks like MitM attacks even more significant as they can be used to harvest data linked to the consumer's purchasing behavior. As a result, traffic fingerprinting for smart home devices must maintain anonymity of the data within the packet so that it cannot be harvested.

2.6 Discussion

The analysis of traffic fingerprinting methodologies is crucial to the success of this project as fingerprinting is at the core of the anti-cyberattack solution for smart home devices. The final goal is to create a forward proxy that is able to identify, report, and void malicious traffic that originates from a set of smart home devices. The first step in this process is differentiating benign traffic and malicious traffic. This requires the ability to distinguish between healthy traffic as well as malicious traffic that is attempting to mask as normal traffic.

Methods of fingerprinting that combine multiple identification features will serve as the base for our overall identification process. An example of this is combining traffic signatures with data-agnostic features such as flow rate. The research done to understand the behavior of devices from the ISP's perspective [22] is critical as the proposed solution is intended to be developed for and marketed to ISPs rather than individual consumers. This approach ensures that we maximize the use and, therefore, the benefit of our solution.

While the previously discussed approaches focus on broad cybersecurity solutions, we are aiming to isolate smart home devices in order to take advantage of their recurring and standardized traffic patterns. Accordingly, the research that focused on the packet timing as the main feature for identification [21] is most important for our solution. The machine learning techniques that were mentioned by the articles will also ensure that our solution is on the cutting edge of traffic fingerprinting technology.

Chapter 3
System Architecture

This chapter will explore the architecture of our solution, both hardware and software. The solution consists of three key elements: the smart home devices, the router, and the Raspberry Pi. This section describes each of these elements and their roles in the proposed approach for securing smart home devices.

3.1 Design Rationale

The primary objective for this project is to create a low-cost solution that can be widely used to significantly reduce the risk of attacks that employ smart home devices. The design for the capture and analysis of packets stemmed from this objective.

We perform network packet analysis at the router level rather than at each smart home device since smart home devices have minimal processing power. We also made this decision because creating universal software for all smart home devices is unfeasible due to the numerous different types of IoT devices. By moving the analysis upstream of the network traffic flow, we are able to generalize the solution and make it more adaptable. Moving the solution upstream also allows us to have a more holistic understanding of the network patterns rather than the patterns of a specific smart home device. This allows the machine learning model to be better suited for a variety of attacks without being overfit to a particular device type.

Although we expect the solution to be used at scale and installed on routers, for the purposes of the senior design project we opted to use a Raspberry Pi to process network traffic. This allows us to focus on the solution rather than how to access and modify an existing router's firmware.

3.2 Conceptual Model

The conceptual model of the final design is presented in Figure 3.1.

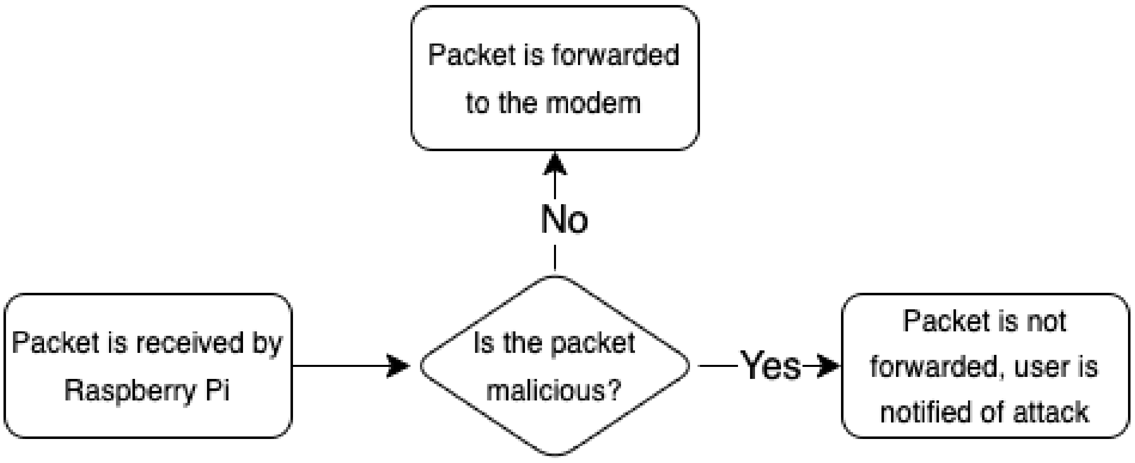
Packet is forwarded to the modem
No
Packet is received by Raspberry Pi
Is the packet malicious?
Yes
Packet is not forwarded, user is notified of attack

Fig. 3.1: Conceptual Model

The Raspberry Pi is set up as a wireless access point between the smart home devices and the router. As shown, the packets that are sent from a smart home device are received by the Raspberry Pi. When any given smart home device on the network sends a message to the router, the software running on the Raspberry Pi will scan the packet, determine if it is benign (i.e. safe/normal traffic) or malicious (i.e. unsafe traffic), and then forward to the router or void the packet.

Under normal use, the Pi will simply forward packets to the router. However, if an attacker attempts to use the smart home device as a component to a botnet, the software should recognize the traffic as malicious and void packets to prevent any attacks the device may be attempting to launch. This functionality will be provided via software on the Pi with no setup or intervention required by the smart home device user.

3.3 System Sequence Diagram

To better illustrate the functionality of the solution, Figure 3.2 shows the system sequence diagram, which highlights the approach to packet capture and analysis.

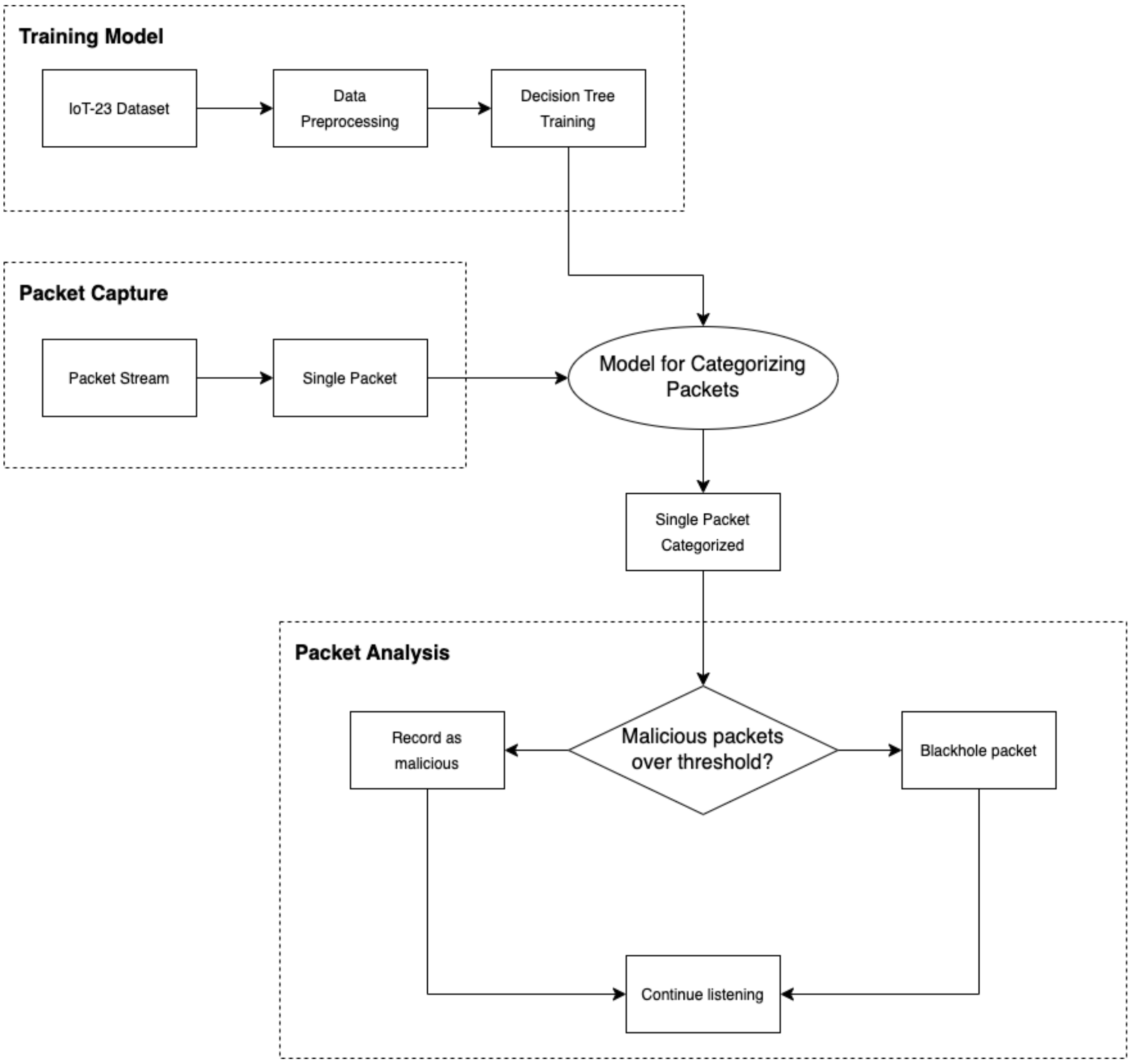

Fig. 3.2: System Sequence Diagram

The solution to securing smart home devices has two distinct steps: training the machine learning model on existing data and using the trained model on packets in a live stream of network traffic.

The analysis of live traffic relies on this preliminary step to ensure efficiency and the highest possible throughput speed. Using the IoT-23 data set [27] that is publicly available for research use, we were able to train a machine learning model that would then be used to classify real network packets. In the preliminary experimentation, we identified the Decision Tree model as the most accurate in identifying the nature of the network packet. In particular, the success rate when tested with approximately 2,000,000 packets was 99.04%, which was significantly higher than other methods such as Linear Discriminant Analysis, which yielded an accuracy of 80.62%.

As shown in Figure 3.2, the model for categorizing network packets is first trained using the predefined data set then applied to live packet capture. As live network traffic is captured, each packet is processed to determine if it is benign or malicious. If the threshold that has been set by the user or the ISP for malicious traffic is reached, the rest of the malicious packets are voided to reduce the load on the victim's network. If the threshold is yet to be reached, traffic that has been labeled as malicious is allowed to reach the internet and the user or the ISP is alerted.

3.4 Architectural Diagram

The hardware requirements for the system are minimal. Figure 3.3 depicts the hardware architecture with the packet analysis object abstracting the system sequence diagram in Figure 3.2.

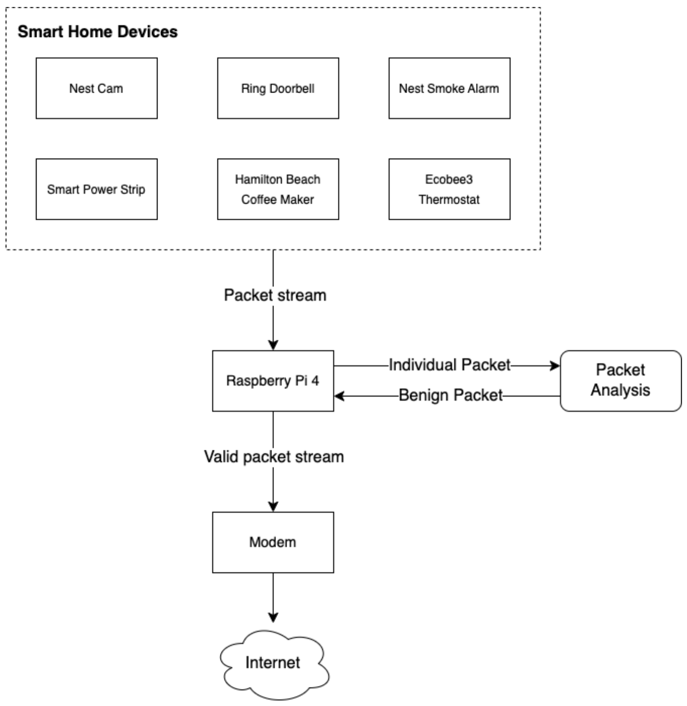

Fig. 3.3: Hardware Diagram

While we envision the solution as a feature in router software, for the purpose of the senior design project we opted for using a standalone Raspberry Pi 4 with a gigabit Ethernet connection. When making this decision, we were considering two important factors. First, we wanted to be able to process traffic at or above the average household's internet speed. Second, we wanted to be able to implement a functional and scalable version of the solution.

The average household traffic rate in the United States is 127.19 Mbps [28]. Given this information, we chose to use a Raspberry Pi 4 with an Ethernet cable directly attached to the router. This provided us with the ability to maintain speeds of up to 943 Mbps [29].

3.5 Machine Learning Model

Since one of the main goals for this project was to dynamically respond to malicious traffic to and from IoT devices, we chose to use a machine learning model for classification. In order to avoid overfitting, we trained the model on a diverse set of traffic from multiple sources. This also helped us create a model that could successfully detect many types of malicious traffic.

3.5.1 Training data

We did not have the capacity to generate a substantial amount of benign and malicious data from real IoT devices. Instead, we decided to use the IoT-23 data set [27]. This data set was selected because it provided almost 21GB of data on data flow from real IoT devices with both malicious and benign data.

For the initial machine learning model, we used 2,000,000 benign packets and 2,000,000 malicious packets. Table 3.1 below shows the distribution of different packet types in the training data set. The Command and Control (C&C) malicious packets demonstrated a connection to an external command and control server.

Packet Type	Packet Count	Percentage
Benign	2,000,000	50%
Malicious: Port Scan	664,000	16.6%
Malicious: Okiru Botnet	540,000	13.5%
Malicious: DDoS	480,000	12.0%
Malicious: C&C	316,000	7.9%

Table 3.1: Packet type distribution within training data.

3.5.2 Feature selection

The .pcap files from the IoT-23 data set provided us with numerous features that we could have used for the model. However, one of the primary concerns was overfitting the model, which would cause poor performance in real world use. To mitigate this, we reduced the number of features down to six. These features were selected based on past research [30] [31] as well as testing on a subset of the IoT-23 data set.

We wanted the implementation to remain stateless to minimize overhead and increase throughput speed so that the software would not have an impact on internet speeds. Thus we restricted the solution to packet header values from different packet layers. In particular, we focused on the network header and the transport layer header. Additionally, we did not do any feature normalization in order to maintain statelessness. Table 3.2 summarizes the features we used in the model

Feature Name	Layer	Description
ip_v	Network	IPv4 or IPv6
ip_length	Network	Length of IP header
isTCP	Transport	Transport layer protocol
isUDP	Transport	Transport layer protocol
src_port	Transport	Source port of packet
dst_port	Transport	Destination port of packet

Table 3.2: Model features and descriptions.

3.5.3 Model selection

The primary concern with using the machine learning approach was processing speed. We needed to ensure that the model was able to process packets at a rate high enough for an average American household's network speed. Accuracy of the model's decision was also critical. Table 3.3 summarizes the performance of multiple machine learning models we used as well as their test time and the theoretical throughput speed based on 100-byte packets.

Model	Precision	Recall	F-1	Time (ns)	Throughput
LDA	0.8145	0.7893	0.7697	219.52	455 Mbps
Naive Bayes	0.5848	0.5765	0.5799	44.63	2240 Mbps
Random Decision Tree	0.9928	0.9927	0.9927	42.93	2329 Mbps
Random Forest	0.9978	0.9978	0.9978	2736.89	36 Mbps

Table 3.3: Machine learning model performances.

Using the above data, we chose the Random Decision Tree model. Although the model could be overfit to the training data, we chose the Random Decision Tree approach due to its speed and the accuracy of the results. To address overfitting, we restricted the tree depth to 10 nodes. Throughout testing, we modified the model parameters to ensure ideal performance.

Below is a snippet of the code for the decision tree model in C++ that was used on the Raspberry Pi.

```cpp
char *predict(int src_port, ... , int service_ssl) {
    if (dst_port <= 73.5)
        if (size_bytes <= 4839312514.0)
            return ("7435440.0,0.0");
        else
            return ("0.0,59161.0");
    else if (protocol_tcp <= 0.5)
        if (size_bytes <= 36.0)
            return ("2101.0,5.0");
        else
            return ("1847.0,23.0");
    else if (src_port <= 62335.5)
        if (dst_port <= 717.5)
            if (src_port <= 32811.5)
                if (src_port <= 10366.5)
                    if (src_port <= 10301.5)
```

```
                        return ("0.0,156859.0");
                    else return ("1.0,1002.0");
                else return ("0.0,342631.0");
                ...
```

3.6 Using the Raspberry Pi

Before we began testing, we first routed all packets through the Pi and ran the machine learning algorithm on the Pi. This simulated a standalone access point for the test bed devices to connect to, which in turn permitted us to have packet-level insight for data collection.

3.6.1 Parsing .pcap files

Before using packets generated in real time, we first used a subset of the IoT-23 data set and ingested the data directly from a .pcap file into the Pi. For this, we used PcapPlusPlus [32]. This C++ library allowed us to rapidly iterate on the implementation by abstracting lower level network logic in the switch. Using the `PcapFileReaderDevice` class provided by PcapPlusPlus, opening a .pcap file was similar to ingesting a file with `IO Stream`. Using the acquired `RawPacket` and `Packet` objects, we extracted the necessary features covered in Section 3.5.2.

Once we were able to read and display information from .pcap files as a proof of concept, we proceeded to implement live packet capture. PcapPlusPlus also provided the `PcapLiveDevice` class, which could ingest incoming packet flow to an Ethernet interface. The packets were then processed with the same `RawPacket` and `Packet` objects.

PcapPlusPlus also made it easy to implement the Decision Tree machine learning model. After parsing the rules and generating effective C++ code for the model, we were able to directly insert the decision tree model into the PcapPlusPlus packet processing function. This enabled us to successfully validate the results of the Decision Tree machine learning model.

3.6.2 Pi as an access point

Utilizing the Raspberry Pi 4 as an access point allowed us to focus development on creating an effective machine learning solution rather then spending a time configuring router hardware.

Following the architecture diagram from Section 3.4, we connected the Pi using an Ethernet cable directly to a household modem. We ran the Raspberry Pi OS Lite firmware on the board and configured the network manager to create a wireless hotspot. It was important for us that packet flow was entirely managed by the machine learning model and PcapPlusPlus. To do so, PcapPlusPlus wrapped the Raspberry Pi's DPDK-enabled network interface which permitted us to have full control over the packet flow.

Chapter 4

Evaluation

In this chapter, the solution using a Raspberry Pi running PcapPlusPlus and the Decision Tree machine learning model is evaluated.

4.1 Testbed Set-up

In order to fully evaluate our program's capabilities, we first created two separate versions of the program with two different physical testbeds. The first version was tailored for functional evaluation. This installation wrote the packet data and model result to a Comma Separated Value (CSV) file without affecting the flow of traffic. This data could then be displayed on a web based dashboard created in Python using Streamlit. Using this CSV data we evaluated the effectiveness of the system. The second version was tailored for the performance evaluation. This installation controlled the flow of traffic without writing any data. The confidence level was set to zero so that any packet would pass the filter so that max throughput could be measured.

4.2 Functional Evaluation

For the functional evaluation, we used 4 different smart home devices and collected 100 packets worth of traffic from each one as well as 100 packets worth of traffic from a simulated DoS attack using TCPreplay. For each of these collections we performed a two sample z test with the significance level being .05, meaning a critical value of 1.645, and the null hypothesis being a system that randomly guesses will be correct half the time. We evaluated each model at confidence levels of .5, .7, and .9 for determining the maliciousness of a packet.

4.3 Performance Evaluation

For performance evaluation we used TCPreplay to send Amazon Echo data into a dummy network interface which fed into the program at various rates. We then utilized Htop to monitor the memory and CPU usage and Iftop to monitor the data rate through the Ethernet interface while the program was under load. This allowed us to see the maximum data rate the system was capable of along with its utilization. For each data rate, we recorded the 10 second averages after they had stabilized.

4.4 Optimization

In order to optimize the machine learning model we ran the functional evaluation and then did analysis on the various features. These changes had no influence on the performance of the system since the only the model is changing not the program.

4.5 Challenges

Many of the devices we had originally planned to test on refused to connect to the Pi's hot spot. We are unsure if this was due to the Pi's configuration or the devices themselves, although we did attempt multiple configurations of the hot spot with no success which suggests it was the devices. The devices included the Apple HomePod Mini, Google Home Mini, Nest Wireless Camera, Hamilton Beach Smart Coffee Maker, Nest Smoke Detector, and Blink Camera. The thermostat proved difficult to setup as we could not physically install it, but we were able to power it on using a 24V AC adapter and skip through the setup.

4.6 Outcomes

4.6.1 Pre-optimization functional evaluation

The first attempt went fairly well for the devices; however, this model did not perform well when it came to the DoS attack. A pattern we quickly noticed was that packets were either entirely blocked or entirely unblocked. After analysis of the results, we determined this was due to the machine learning model's heavy usage of each packet's port as a key feature for determining chance of maliciousness. For the optimization we decided to focus on the DoS attack since this was a necessary functionality for the solution.

Confidence Level	**.5**		**.7**		**.9**	
Device Name	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**
August Lock	0	10	0	10	0	10
Ecobee Thermostat	0	10	0	10	0	10
Ring Doorbell	100	-10	100	-10	100	-10
Echo Show 3	2	9.6	0	10	0	10

Table 4.1: Functional Evaluation of IoT Devices Before Optimization.

Confidence Level	**.5**		**.7**		**.9**	
Attack Name	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**
DoS	0	10	0	10	0	10

Table 4.2: Functional Evaluation of Simulated Attacks Before Optimization.

4.6.2 Post-optimization functional evaluation

Interestingly, after optimization, every result flipped from the pre-optimization evaluation. This initially seemed like a bug, but investigation revealed it to be correct. Like the pre-optimization evaluation, each device or attack was, for the most part, entirely blocked or entirely unblocked. The new model did successfully block the attack. The reversal of results from the old model to the new model suggests that it is difficult to build a single general purpose model for all devices and attacks.

Confidence Level	**.5**		**.7**		**.9**	
Device Name	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**
August Lock	100	-10	100	-10	100	-10
Ecobee Thermostat	100	-10	100	-10	100	-10
Ring Doorbell	0	10	0	10	0	10
Echo Show 3	99	-9.8	99	-9.8	99	-9.8

Table 4.3: Functional Evaluation of IoT Devices After Optimization.

Confidence Level	**.5**		**.7**		**.9**	
Attack Name	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**	**% Blocked**	**Z-Score**
DoS	100	-10	100	-10	100	-10

Table 4.4: Functional Evaluation of Simulated Attacks After Optimization.

4.6.3 Performance evaluation

Although the maximum throughput of 430 Mbps is much less than the model's theoretical maximum of 2329 Mbps and the Raspberry Pi 4's theoretical maximum of 943 Mbps [29], we are happy with this performance as it is much higher than the average US traffic rate of 127.19 Mbps [28]. Based on the CPU usage being at the maximum while the memory usage is minimal, we can conclude that the system is bottle-necked by the Raspberry Pi 4's CPU.

Data Rate In	**Data Rate Out**	**CPU Usage**	**Memory Usage**
1Mbps	1Mbps	100%	0.2%
10Mbps	10Mbps	102%	0.2%
100Mbps	100Mbps	104%	0.2%
250Mbps	250Mbps	104%	0.2%
500Mbps	430Mbps	106%	0.2%

Table 4.5: Performance Evaluation.

Chapter 5
Conclusion and Future Work

5.1 Design Improvements

Based on the findings of the model making the majority of its decision by inspecting the ports used by the devices, which resulted in it either blocking all or none of the packets, it is clear that a one size fits all model paradigm does not work. A future approach would be to fingerprint individual devices then create different models for each different device. Fingerprinting could be done by the user, but this increases friction and potential complications. Instead, the ideal system would actively learn from a devices traffic over time to create the model on the fly. However, this could lead to problems if a device is updated and its traffic patterns drastically change.

5.2 Commercial Viability

The system, utilizing a synthetic workload, was able to reach 430 Mbps throughput at maximum CPU utilization. Considering the average US traffic rate of 127.19 Mbps a less powerful CPU could be used in the vast majority of cases [28]. Further testing on various devices would be required to determine the best CPU for various use cases. If this system is to be built directly into routers then testing may be complicated by the routers need to perform other functions while the Pi 4 can dedicate an entire core to the program. Furthermore, integrating a C++ program into a router may prove difficult, costly, or impossible. The program could be ported to P4 which would increase ease of integration; however, this would take substantial work. Additionally, work would need to be done to monitor and push updates to the system if it were put into production.

5.3 Project Takeaways

Although the final results were not exactly what we envisioned when we started, the discovery that individual models need to be created for individual devices is valuable. Furthermore, we laid out the ground work for what a system like this would look like by demonstrating an implementation of a general model. The proliferation of insecure IoT devices will only grow in scale. We have taken one step towards protecting the World Wide Web from infected devices, but more must be done or else attacks will grow in scale equal to the devices themselves.

5.4 Personal Takeaways

For all of us, this was the longest continuous project we have worked on. Besides learning technical skills in building machine learning in Python, packet capture tooling through C++, and data analysis through purpose built dashboards, we gained a wealth of knowledge in planning and executing a long term project. Throughout research, design, and implementation we constantly collaborated to build a project we are proud of. In the end, we built a project we can point to with pride as a testament to our hard work and determination.

Bibliography

[1] Y. Al-Hadhrami and F. K. Hussain, "Ddos attacks in iot networks: a comprehensive systematic literature review," *World Wide Web*, Jan 2021.

[2] "Inside the infamous mirai iot botnet: a retrospective analysis," Dec 2017. [Online]. Available: https://blog.cloudflare.com/inside-mirai-the-infamous-iot-botnet-a-retrospective-analysis/#toc-2

[3] N. Liebermann, *2021 IoT Security Landscape*, Apr 2022.

[4] B. Tushir, Y. Dalal, B. Dezfouli, and Y. Liu, "A quantitative study of ddos and e-ddos attacks on wifi smart home devices," *IEEE Internet of Things Journal*, 2020.

[5] C. Rossow, "Amplification hell: Revisiting network protocols for ddos abuse," *Proceedings 2014 Network and Distributed System Security Symposium*, 2014.

[6] OVHcloud, "Ovhcloud." [Online]. Available: https://us.ovhcloud.com/

[7] G. Gallopeni, B. Rodrigues, M. Franco, and B. Stiller, "A practical analysis on mirai botnet traffic," *IEEE Xplore*, p. 667 to 668, Jun 2020. [Online]. Available: https://ieeexplore.ieee.org/document/9142798/

[8] J. Fruhlinger, "The mirai botnet explained: How iot devices almost brought down the internet," Mar 2018.

[9] L. Xu, J. Huang, S. Hong, J. Zhang, and G. Gu, "Attacking the brain: Races in the sdn control plane," ser. SEC'17. USA: USENIX Association, 2017, p. 451 to 468.

[10] S. K, S. V, A. Singh, A. R, H. Saxena, and S. S. S, "Detection and mitigation of man-in-the-middle attack in iot through alternate routing," p. 341 to 345, Mar 2022. [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9753832

[11] J. J. Kang, K. Fahd, S. Venkatraman, R. Trujillo-Rasua, and P. Haskell-Dowland, "Hybrid routing for man-in-the-middle (mitm) attack detection in iot networks," *2019 29th International Telecommunication Networks and Applications Conference (ITNAC)*, Nov 2019.

[12] C. Osborne, "Samsung connected home fridge becomes weapon in mitm attacks," Sep 2015. [Online]. Available: https://www.zdnet.com/article/samsung-connected-home-fridge-becomes-weapon-in-mitm-attacks/

[13] Firewalla, "Cyber security," *Firewalla*, Apr 2022. [Online]. Available: https://help.firewalla.com/hc/en-us/articles/360026357333

[14] D. Puri, "Ddos attacks using iot devices follow the manchurian candidate model," *Network World*, Oct 2016. [Online]. Available: https://www.networkworld.com/article/3128372/ddos-attacks-using-iot-devices-follow-the-manchurian-candidate-model.html

[15] M. Mainuddin, Z. Duan, and Y. Dong, "Network traffic characteristics of iot devices in smart homes," *2021 International Conference on Computer Communications and Networks (ICCCN)*, Jul 2021.

[16] "Glasswire network security monitor and firewall tool features." [Online]. Available: https://www.glasswire.com/features/

[17] W. Zhang and H. Wang, "Identification of peer-to-peer traffic based on process fingerprint," *2011 International Conference on Mechatronic Science, Electric Engineering and Computer (MEC)*, Aug 2011.

[18] A. W. Moore and K. Papagiannaki, "Toward the accurate identification of network applications," *Lecture Notes in Computer Science*, p. 41 to 54, 2005. [Online]. Available: https://link.springer.com/chapter/10.1007/978-3-540-31966-5_4

[19] Z. Shuzhuang, F. Binxing, and L. Hao, "A scalable framework of network traffic identification," *2008 The Ninth International Conference on Web-Age Information Management*, Jul 2008.

[20] R. Kumar, M. Swarnkar, G. Singal, and N. Kumar, "Iot network traffic classification using machine learning algorithms: An experimental analysis," *IEEE Internet of Things Journal*, 2021.

[21] 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, no. 8, p. 1745 to 1759, Aug 2019.

[22] X. Ma, J. Qu, J. Li, J. C. S. Lui, Z. Li, W. Liu, and X. Guan, "Inferring hidden iot devices and user interactions via spatial-temporal traffic fingerprinting," *IEEE/ACM Transactions on Networking*, vol. 30, no. 1, p. 394 to 408, Feb 2022. [Online]. Available: https://ieeexplore.ieee.org/document/9548663

[23] 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, no. 8, p. 1745 to 1759, Aug 2019.

[24] P. Foremski, S. Nowak, P. Frohlich, J. Hernandez-Ramos, and G. Baldini, "Autopolicy: Automated traffic policing for improved iot network security," *Sensors*, vol. 20, no. 15, p. 4265, Jul 2020.

[25] B. Lee, O. Kwon, I. Lee, and J. Kim, "Companionship with smart home devices: The impact of social connectedness and interaction types on perceived social support and companionship in smart homes," *Computers in Human Behavior*, vol. 75, p. 922 to 934, Oct 2017.

[26] B. Canziani and S. MacSween, "Consumer acceptance of voice-activated smart home devices for product information seeking and online ordering," *Computers in Human Behavior*, vol. 119, p. 106714, Jun 2021.

[27] S. Garcia, A. Parmisano, M. J. Erquiaga, V. Valeros, and M. Rigaki, "Iot-23: a labeled dataset with malicious and benign iot network traffic (2020)," *More details here https://www. stratosphereips. org/datasets-iot23*, 2021.

[28] T. Cooper and J. Tanberk, "Best and worst states for internet coverage, prices and speeds, 2021," Sep 2021. [Online]. Available: https://broadbandnow.com/research/best-states-with-internet-coverage-and-speed

[29] M. Zolnierczyk, "Raspberry pi network speed test: Rpi2, rpi3, zero, zerow (lanwifi)," Apr 2017. [Online]. Available: https://notenoughtech.com/raspberry-pi/raspberry-pi-internet-speed

[30] M. Austin, "Iot malicious traffic classification using machine learning," *Graduate Theses, Dissertations, and Problem Reports*, Jan 2021. [Online]. Available: https://researchrepository.wvu.edu/etd/8024

[31] C. V. Oha, F. S. Farouk, P. P. Patel, P. Meka, S. Nekkanti, B. Nayini, S. X. Carvalho, N. Desai, M. Patel, and S. Butakov, "Machine learning models for malicious traffic detection in iot networks /iot-23 dataset/," *Machine Learning for Networking*, p. 69 to 84, 2022.

[32] [Online]. Available: https://pcapplusplus.github.io