FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Proxy-based solution for legacy IoT security and privacy

Rodrigo Monteiro da Cunha Costa Caldas

MASTER DISSERTATION

Mestrado Integrado em Engenharia Eletrotécnica e de Computadores

Supervisor: Ricardo Santos Morla

Co-supervisor: Carlos Novo

July 23, 2021

Abstract

The lack of communication security is a common vulnerability in legacy IoT devices, with some of them being vulnerable to trivial attacks that can easily compromise the entire system security and its users. As a result, securing legacy device traffic becomes a necessity when redesigning the device itself is not feasible.

Using TLS is a common approach to securing network communications, since it provides privacy based on encryption. However, detection based on machine learning has made possible the detection of traffic from IoT devices and inferring the behavior of devices even when network traffic is fully encrypted. This detection can compromise the privacy of the system as well as be a first step for other attacks. On the other hand, it has already been shown in multiple domains that machine learning detectors can also be misled when using traffic obfuscation techniques, which are popular security solutions used to proactively protect IoT systems from possible traffic analysis attacks performed by a network eavesdropper.

The objective of this dissertation is to analyze and mitigate inference threats on encrypted traffic. After establishing an existing layer 2 tunnel over TLS on a multi-node fire detection and alarm system, we address the possibility of device behavior inference using a machine learning-based approach.

To mitigate these threats, we implement traffic obfuscation methods such as packet padding and dummy traffic in the tunnel, which, as intended, resulted in a decrease in the eavesdropper's ability to infer legacy device behavior. These mitigations remain effective even if the eavesdropper retrains its model considering the padding and dummy traffic.

Resumo

A falta de segurança das comunicações é uma vulnerabilidade comum nos dispositivos IoT herdados, sendo alguns deles vulneráveis a ataques triviais que podem facilmente comprometer toda a segurança do sistema e dos seus utilizadores. Como resultado, a segurança do tráfego de dispositivos herdados torna-se uma necessidade quando o redesenho do próprio dispositivo não é viável.

A utilização de TLS é uma abordagem comum à segurança das comunicações de rede, uma vez que proporciona privacidade baseada na encriptação. No entanto, a detecção baseada em aprendizagem computacional tornou possível a detecção de tráfego de dispositivos IoT e inferir o comportamento dos dispositivos, mesmo quando o tráfego de rede está totalmente encriptado. Esta detecção pode comprometer a privacidade do sistema, bem como ser um primeiro passo para outros ataques. Por outro lado, já foi demonstrado em múltiplos domínios que os detectores de aprendizagem computacional também podem ser enganados quando se utilizam técnicas de ofuscação do tráfego, que são soluções populares de segurança utilizadas para proteger proactivamente os sistemas IoT de possíveis ataques de análise de tráfego realizados por um espião de rede.

O objectivo desta dissertação é analisar e mitigar as ameaças de inferência ao tráfego encriptado. Depois de estabelecer um túnel de camada 2 existente sobre TLS num sistema de detecção e alarme de incêndio de múltiplos nós, abordamos a possibilidade de inferência do comportamento do dispositivo utilizando uma abordagem baseada em aprendizagem computacional.

Para mitigar estes ataques, implementamos métodos de ofuscação de tráfego como o *padding* de pacotes e tráfego fictício no túnel, o que, como pretendido, resultou numa diminuição da capacidade do espião para inferir o comportamento do dispositivo legado. Estas mitigações continuam a ser eficazes mesmo que o espião volte a treinar o seu modelo considerando o *padding* e o tráfego fictício.

Agradecimentos

I would like to thank my supervisor, Ricardo Morla, for his helpful orientation, support and advice throughout the last months. To my co-supervisor Carlos Novo, I also thank for all the discussions and support during my dissertation.

Last but not least, I would also like to thank my parents and brother for always encouraging and helping me, and my girlfriend Patrícia and her family for all the unconditional support throughout all these years.

Rodrigo Caldas

"It's fine to celebrate success but it is more important to heed the lessons of failure."

Bill Gates

Contents

1 Introduction **1**
- 1.1 Context and Motivation . . . 1
- 1.2 Problem Description . . . 2
- 1.3 Objectives . . . 2
- 1.4 Contributions . . . 3
- 1.5 Document Structure . . . 3

2 Literature Review **5**
- 2.1 IoT Security . . . 5
  - 2.1.1 Exponential growth of smart devices . . . 5
  - 2.1.2 IoT vulnerabilities . . . 6
  - 2.1.3 IoT security mechanisms . . . 6
  - 2.1.4 Privacy in IoT . . . 7
- 2.2 Traffic analysis and fingerprinting . . . 7
  - 2.2.1 Website . . . 7
  - 2.2.2 Tor . . . 8
  - 2.2.3 IoT traffic . . . 9
- 2.3 Traffic detection avoidance . . . 9
- 2.4 Conclusion . . . 10

3 IoT system and layer 2 TLS Tunnel **13**
- 3.1 IoT System . . . 13
- 3.2 Existing L2 TLS Tunnel . . . 14
- 3.3 Improvements for detection avoidance . . . 14
  - 3.3.1 Avoidance goals . . . 15
  - 3.3.2 Avoidance approach . . . 15

4 Tunnel and behavior detection **17**
- 4.1 Devices and traffic behavior . . . 17
- 4.2 Statistical features . . . 18
- 4.3 Machine Learning model . . . 19

5 Results **25**
- 5.1 Detection results . . . 25
- 5.2 Avoidance results . . . 26

6 Conclusion **31**
- 6.1 Future Work . . . 31

References **33**

List of Figures

1.1 Inference threats on legacy IoT devices traffic. 2

2.1 Publications in IoT security from 2016 to 2018 6

3.1 IoT system setup. 14
3.2 Schematic of the IoT system and TLS tunnel architecture and components. 14
3.3 Packet format. 16

4.1 Setup for IoT tunnel traffic generation. 18
4.2 Simplified illustration of data processing. 20
4.3 Sliding Time Window. 21
4.4 Classifier input for the different behaviors 22
4.5 Process to build a ML model 23

5.1 Targeted approach results 27
5.2 Non-targeted approach results: accuracy while discriminating different behaviors. 28
5.3 Non-targeted approach results: bitrate CDF for each behavior: top figure representing original traffic (no obfuscation), bottom figure with normal range obfuscation. 28
5.4 Non-targeted approach results: accuracy while differentiating ACK and INI behaviors for different obfuscation ranges 29
5.5 Non-targeted approach results: bitrate CDF for ACK and INI mixed traffic. . . . 29

List of Tables

4.1 IP packet and TLS record features 19
4.2 Confusion matrix for binary classification 23

5.1 Detection accuracy from UNSW dataset and tunnel background dataset 25
5.2 Device behavior detection results 26

Acronyms and Abbreviations

IoT	Internet of Things
DoS	Denial-of-Service
OWASP	Open Web Application Security Project
ML	Machine Learning
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
VPN	Virtual Private Network
TLS	Transport Layer Security
IP	Internet Protocol
CSV	Comma-separated values
TCP	Transmission Control Protocol
DTLS	Datagram Transport Layer Security

Chapter 1

Introduction

The first chapter of this thesis includes a concise introduction to the topic of encrypted traffic detection on legacy devices and also traffic inference threats by a network eavesdropper. The context and motivation supporting this dissertation are described and briefly summarized in a problem description, and then the objectives, contributions and document structure are presented.

1.1 Context and Motivation

The Internet of Things (IoT) is increasingly a popular technology in progressive development due to the spread of communication systems and the availability of smart devices and computer systems that have revolutionized the way we interact with the world.

Nowadays, devices are not only expected to be connected to the Internet and other local devices, but also to communicate directly with other devices over the Internet. Despite the many benefits that can be obtained from all interactions established between different types of intelligent devices, these benefits are eclipsed by concerns about privacy and vulnerabilities. Traffic detection based on machine learning (ML) has made possible to identify IoT traffic through patterns that define it, and has been extensively applied to infer device behavior [1].

Despite providing unprecedented accessibility and efficiency, legacy IoT systems – that are devices for which the architecture cannot be modified, and which have not had network security and privacy considerations incorporated into their design – have caused serious security and privacy threats to users in recent years [2] and as such measures should be taken to mitigate these risks. The research community in the domain of IoT Security has been working hard to build efficient methods for identifying traffic, intrusion and cyber attacks using machine learning algorithms for IoT security analysis. Although privacy in legacy IoT systems is fundamental to ensure the protection of users and their personal information, sometimes it doesn't exist.

Network eavesdropping is a network layer attack that is based on the violation of confidentiality, where an attacker taking advantage of vulnerabilities in the communications of possible victims, performs unauthorized monitoring on this communication. This technique is also linked to metadata collection. Figure 1.1 represents in a simplified way the steps an eavesdropper follows

to perform an inference threat on legacy IoT devices using a machine learning approach. Traffic analysis attacks allow an attacker to infer sensitive information by analyzing network traffic from legacy devices even if the device traffic is encrypted. These attacks are sometimes passive in nature and are difficult to detect.

Therefore, legacy systems require custom levels of privacy and security to be considered, leading to the need to develop effective traffic morphing techniques to make it difficult for a network observer to effectively perform traffic analysis attacks.

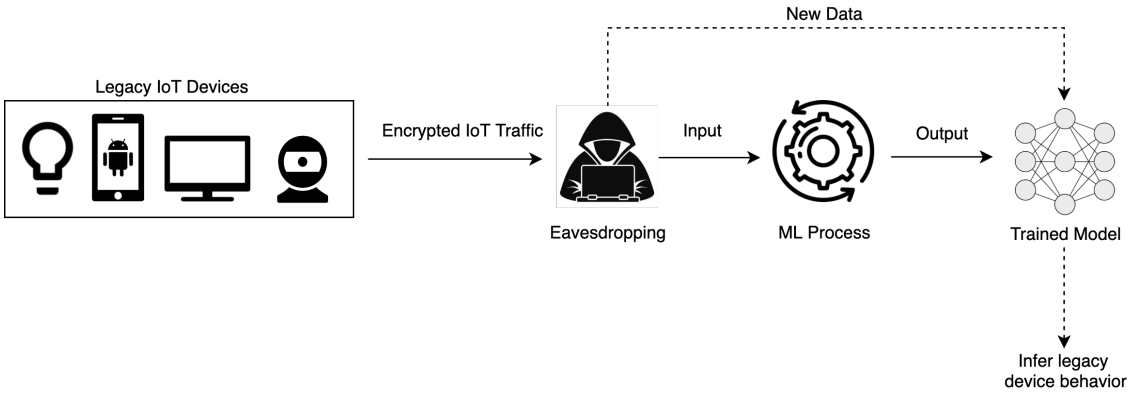

Figure 1.1: Inference threats on legacy IoT devices traffic.

1.2 Problem Description

Legacy IoT devices are becoming more and more targeted by threats related to the inference of legacy device behavior by machine learning classifiers [3][4][5]. On the one hand, it becomes necessary to understand in which conditions the traffic and behavior of legacy devices can be identified. On the other hand, it is important to determine how legacy traffic can be shaped so that it is not detected by a classifier (lowering the ability of an eavesdropper to infer legacy device behavior), and also what impact this change will have on the quality of the traffic.

By applying this to our legacy devices, it may be possible to understand whether it is possible to distinguish legacy devices from other IoT devices and distinguish different behaviors generated by the same legacy device, assessing the impact of mitigation strategies against inference threats on encrypted traffic.

1.3 Objectives

The main goal of this thesis is focused on the detection of legacy device behavior and how the obfuscation of traffic generated by legacy devices can avoid being detected by an attacker eavesdropping the network traffic.

For the purpose of studying the performance of behavior detection techniques and privacy preservation in legacy IoT devices through a layer 2 TLS tunnel, a ML classifier shall be developed that classifies legacy traffic behavior, as well as methods to shape all traffic to avoid being detected.

1.4 Contributions

The main contributions of this dissertation are:

- Explore the detection of TLS traffic and device behavior generated by legacy IoT devices using Machine Learning-based detectors.
- Implementation of traffic obfuscation techniques such as packet padding and dummy traffic generation in order to avoid device behavior detection.

This work has been included in a conference paper – *5th Cyber Security in Networking Conference* [6] – which is under review at the time of writing.

1.5 Document Structure

Apart from this introduction, this document is divided into 5 more chapters. In the following Chapter 2, *Literature Review*, we overview the essential concepts, as well as an exploration of related research in the domains of IoT security and also on traffic analysis and traffic detection avoidance. This is followed by Chapter 3, *IoT system and layer 2 TLS Tunnel*, where all the initial system of our legacy devices and the TLS tunnel are presented, as well as the improvements implemented in the tunnel with the goal of avoiding traffic detection. Chapter 4, *Tunnel and behavior detection*, gives a detailed description of our legacy devices and the different traffic behaviors, as well as the extracted traffic features and the development of our ML model. Chapter 5, *Results*, presents the results obtained for device behavior detection without obfuscation mechanisms and later using traffic obfuscation. Finally, on the last Chapter 6, *Conclusion*, we discuss the results obtained on previous chapters, ending this document with some conclusion remarks and proposals of future work.

Chapter 2

Literature Review

This Literature Review starts by reviewing work focusing on IoT security particularly the challenges and concerns that this topic has brought in recent years. Then it looks at traffic analysis and fingerprinting, including for IoT traffic, and at mechanisms to avoid such detection.

2.1 IoT Security

Internet of Things (IoT) is a technology that is becoming more and more present in our daily lives that allows different physical devices, vehicles, or even home appliances to communicate and even inter operate with each another [2][7]. These services provided by IoT applications offer a great benefit to human life, but can come at a huge price considering the protection of privacy and security of the user [8]. Given that the data shared between devices over the Internet contains a significant amount of private information, preserving the security of that information is an important issue that cannot be disregarded [9][10].

Authors in [11] reported that there are still several IoT security challenges, such as jamming and spoofing attacks, which compromise the integrity and privacy of the user's data. Thus despite bringing unprecedented accessibility and efficiency, IoT technologies have caused severe threats in recent years. More and more research is being done to mitigate these threats, but many problems remain open.

2.1.1 Exponential growth of smart devices

Globally, it is estimated that 127 new devices connect to the Internet every second. A McKinsey Global Institute report [12] estimates that IoT technologies could have an annual economic impact of $3.9 trillion to $11.1 trillion by 2025 in several different scenarios, including factories, cities, retail environments and the human body.

Given these encouraging numbers, a huge technological wave is expected in the coming years, and with it not only smart devices will bring various challenges, but also communications between

them must be as secure as possible, which in itself is already a very difficult challenge.

2.1.2 IoT vulnerabilities

An increasing number of IoT devices are being implemented in a wide range of environments, progressively becoming common objects of daily life. In such a scenario, cyber-security becomes essential to prevent threats such as information leaks, denial of service (DoS) attacks, unauthorized network access and attacks on user's privacy.

The OWASP Internet of Things Project [13] has indicated the most common vulnerabilities of IoT systems and has shown that many of these risks emerge due to the lack of adoption of well-known security techniques such as encryption, authentication and access control [14].

Zhou et al. [2] highlighted that along with the rapid growth of IoT applications and devices, cyber attacks will also be improved and will pose a more serious threat than ever to security and privacy. On the other hand, Apthorpe et al. [15] focuses on how network traffic content, patterns and metadata can reveal sensitive information about a user's activity. These authors introduced some ways of mitigating these risks such as obfuscating all traffic to mask variations that encode user's behavior or also using VPN tunnels to mask packet headers would also make device identification more difficult.

2.1.3 IoT security mechanisms

The main purpose of the security mitigation application according to Mohamad Noor et al. [16] is to preserve privacy, confidentiality, ensuring the security of users, data and IoT devices.
Figure 2.1 shows tendencies in techniques and methods that have been used in 2016-2018. Authentication is still the most popular technique for security, while encryption research is being focused on lightweight and limited devices.

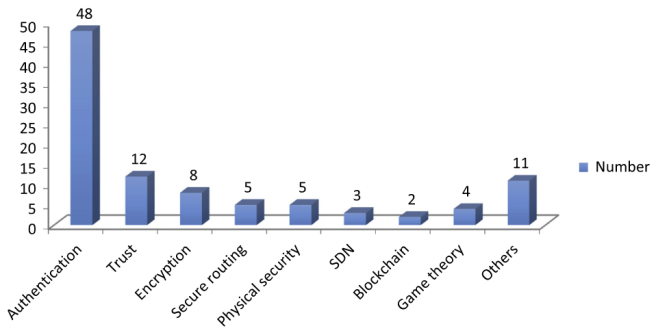

Figure 2.1: Publications in IoT security from 2016 to 2018

Source: [16]

Mohamad Noor et al. [16] also states that authentication and encryption can be effective solutions to mitigate IoT security problems. However, for low-resource and low-power computing devices, the implementation of effective authentication and encryption is still in its first phase and as such cannot prevent malicious nodes on the network. As a result, other more effective solutions for risk mitigation must be found.

According to Tiburski et al. [17], security protocols, such as *Transport Layer Security* (TLS), are increasingly chosen by most existing intelligent systems in order to protect their communications. The authors performed experiments that demonstrated that this implemented protocol is able to provide security to IoT communication channels without affecting system performance.

2.1.4 Privacy in IoT

While IoT systems expose us to a variety of different types of security threats in our daily life, privacy remains a huge challenge because a lot of user's personal information is delivered and shared between connected devices. However, these privacy considerations are still being poorly addressed, as seen in [18].

In [19], Hwang debates how the Internet of things is affecting our lives mentioning that IoT gives us many benefits, but these benefits have a cost. Hwang highlighted that some security and privacy issues are caused by poor user configuration. Therefore, providing solutions to easily configure or automatically enforce security and privacy policies is highly critical.

As a result, it is evident that security solutions capable of ensuring the protection of IoT systems and users represent a priority need. Researchers are working to propose and find solutions to mitigate these privacy challenges.

2.2 Traffic analysis and fingerprinting

Several encryption methods and anonymous communication systems are developed to protect the communication content and hide the user's identity. However, both of these technologies are vulnerable to traffic analysis, where communication patterns, such as packet sizes, timings and counts, are used to infer sensitive information [20].

Recent work has shown that network eavesdroppers can use traffic rate metadata to infer user's activity. In order to mitigate these risks, Hafeez et al. presented a functional traffic morphing mechanism that makes hard for network eavesdropper to perform traffic analysis attacks, as seen in [1]. For the same purpose, Pinheiro et al. [21] provided a lightweight padding-based mechanism to obfuscate only one type of information, the packet length, to increase the user privacy, showing that this method lowers the classifier's accuracy by at least 75%.

2.2.1 Website

Encryption is frequently proposed as a privacy protection tool for browsing the World Wide Web. However, this process does not hide all information about the encrypted text since it can expose

sensitive information from users, such as their browsing behavior.

HTTP object counts and sizes are frequently exposed. Sun et al. [22] investigated the identifiability of traffic based on this information on a large sample of webpages, and proved that it is possible to identify a significant part of it with a high accuracy. In addition, the information revealed seems to be sufficient to be useful to an attacker monitoring such encrypted traffic in an attempt to compromise the user privacy of web browsing. Similarly, Shen et al. [23] proposed a new webpage fingerprinting method by observing the length of the packets information in bi-directional interaction between clients and servers and concluded that this could be a differentiating feature in webpage fingerprinting. They then extract the cumulative length from a sequence of packets to represent the fingerprint of a specific webpage.

From a defense perspective against fingerprint attacks on web pages, Wright et al. [24] proposed a technique to make one traffic pattern similar to another. However, their morphing algorithm only mapped a distribution of the size of one packet to another, not changing the packet sequencing. The authors also proposed a variant of their defense that would only extend the packets, it never split or re-ordered the packets. Lu et al. [25] analyzed the traffic morphing and noticed that traces of the request information remain even under traffic morphing and can be extracted for identification. Underlining that an effective countermeasure to website fingerprinting should not only hide the packet size distribution, but also remove the request information.

The successful results obtained by these authors prove that it is relatively simple to identify the traffic generated by the web pages and, consequently, improvements in the existing countermeasures are necessary in order to avoid the exposure of this type of information.

2.2.2 Tor

Tor is likely the best known anonymous communication system on the Internet, based on a second generation onion forwarding network protocol. It is mainly used due to the difficulty in tracking the Tor user's activity.

This system implements layered routing, that is, using the project's voluntary network of servers around the world, creates circuits to route user's data. All traffic on the Tor network is encrypted and even a network participant cannot observe the traffic passing through the server, as everything is encrypted. Tor provides anonymous TCP connections through source-routed paths that originate at the Tor client and go through three relays: guard, middle and exit relays. Traffic enters the network through guard relays. Then, middle relays carry network traffic from guard relays to exit relays where the traffic exits the anonymity network [26].

More recent studies show that it is possible to analyze and detect Tor traffic. Cuzzocrea et al. [27] proposed a machine learning technique capable of identifying whether a user is using the Tor network. In addition, the proposed method is capable of recognizing the type of activity (e.g. email or P2P applications) that the end-user is performing on the Tor network. According to Almubayed et al. [28], Tor can be classified amongst HTTPS encrypted traffic. Tor has implemented different defenses in order to avoid automatic identification of traffic such as TLS encryption, padding and packet retransmission. However, as the authors have evidenced, Tor does not seem to be able to

properly obfuscate network packets, which makes it possible for a local observer to identify Tor traffic on the network and fingerprints at most of the top sites on Alexa.

2.2.3 IoT traffic

Msadek et al. [3] demonstrated how an attacker would be able to discover sensitive information about the IoT device, such as its type, by identifying specific patterns in IoT traffic. To perform the fingerprint attack, the authors implemented machine learning models based on selected features extracted from the encrypted IoT traffic. Once the type of device is identified, the result will be that the devices and user data will be at risk. To bring IoT security policymakers and manufacturers of IoT devices to the attention, the authors demonstrated in this article the high precision of machine learning techniques in fingerprinting encrypted traffic on IoT devices. Through the analysis of four commercially available smart home devices (a Sense sleep monitor, a Nest Cam Indoor security camera, an WeMo switch and an Amazon Echo), Feamster et al. [15] found that the network traffic rates of these devices exposed user activities, highlighting that encryption alone does not properly protect the privacy of smart homes. Given the general traffic analysis strategy adopted and the limited nature of most IoT devices, the authors stated that many other currently available smart devices may also suffer from similar privacy vulnerabilities. Similarly, Skowron et al. [29] observed that IoT traffic allows relatively easy recognition of IoT devices and have also proven that an eavesdropper using machine learning-based methods can successfully learn the activities made by a system user by following changes in the status of the smart devices.

As a result, traffic analysis attacks allow an attacker to infer sensitive information about users by analyzing the network traffic of user's devices, for example, the Sense sleep monitor allows that from a peak of traffic at the end of the night it probably corresponds to when the user went to sleep. The achieved results showed that IoT traffic is strongly distinguishable and allows accurate identification of the devices only using traffic features (such as packet sizes, distribution or different statistical measures) without requiring deep packet inspection. In order to mitigate the privacy implications of traffic analysis attacks, techniques to obfuscate or shape all traffic are required to make it more difficult to identify IoT devices and their activities, and preventing the leakage of user's private data.

2.3 Traffic detection avoidance

Different studies have proposed mechanisms to avoid traffic detection in non-legacy IoT domains, particularly focusing on traffic tunnels and traffic shaping. According to Skowron et al. [29], the use of traffic tunnels allows a user to reduce the risk of exposure of his private data by routing all traffic through a VPN server or a proxy [30] that would obscure the destination IP address of network traffic. The risk of attack is considerably reduced as an adversary using simple techniques would be unable to split traffic flows that correspond to specific individual IoT devices. However this method only does make the attack more difficult, it does not provide any guarantee of privacy

since VPN can reveal traffic rate patterns that make the metadata privacy attack possible. An adversary could still use the existing traffic rate information from the VPN in machine learning.

Traffic shaping can be implemented by device producers or by a third party on a gateway router. Several methods already exist for interfering with communication models that allow hiding their context such as packet padding [21] that consists of attaching some irrelevant information at the end of the packets in order to obfuscate the traffic flow characteristics.

For this purpose, Apthorpe et al. [31], demonstrate that traffic shaping by independent link padding (ILP) prevents the metadata attack while preserving device functionality. This technique involves shaping traffic rates to match a predetermined rate or schedule, thus not exposing any information about device behavior to a possible attacker. The authors found that performing constant rate ILP on the home gateway router required approximately 40KB/s of overhead traffic in order to provide a sufficiently low latency to preserve the device's minimal functionality resulting in approximately 104GB of overhead data per month, which is excessive for intelligent homes. Similarly, a few years later, Apthorpe et al. [32] have developed a new and more complex defense, "stochastic traffic padding" (STP), which is a traffic modelling algorithm that uses intermittent traffic padding periods to limit the information revealed about user activities through traffic rate metadata, making it hard for the passive network opponent to differentiate authentic user interactions from generated traffic patterns designed to look like user activities. STP shapes traffic to fixed patterns chosen to cover all traffic flows of possible user activities and as such has limitations on bandwidth overhead costs. This method is also focused on protecting traffic rate metadata from user activity inference but cannot shape categorical metadata (protocols, DNS hostnames and IP addresses). In order to completely protect from activity inference, STP must be combined with a method to remove or obfuscate categorical metadata related to user activities.

2.4 Conclusion

Security and privacy remain huge challenges for IoT devices, bringing a whole new level of privacy concern for users. This is because these devices not only collect private data from users, but can also monitor their activity.

While network traffic identification has been well discussed, it is still immature in IoT due to the differences in traffic characteristics between IoT and non-IoT devices. However, despite reviews of IoT and network traffic in existing studies, it still represents a considerable privacy problem for network users, in particular with the common presence of sensing devices even when communications are ensured by security protocols such as TLS.

Several methods have been developed to ensure user privacy. However, there are still restrictions, particularly because of the general bandwidth overhead costs that some of these mechanisms require, and as a result, further research and development is needed to minimize the impact of these limitations.

All in all, IoT security and privacy research has recently gained much impetus with the help of available simulation tools, computing and analysis platforms. Although IoT attack models and ML

based IoT security techniques are already identified [33][34], there are still certain implications particularly on how to avoid legacy traffic behavior detection in order to protect the system and user's privacy, and as such should remain a critical point requiring further investigation.

Chapter 3

IoT system and layer 2 TLS Tunnel

The development of an IoT system demands that adequate security - namely confidentiality, integrity and availability (CIA) - are required to guarantee the security of communications, which implies solutions to ensure efficient and reliable end-to-end communication. Using a layer 2 TLS tunnel developed in the scope of the project in which this dissertation is inserted, it is possible to establish a secure TLS connection between two endpoints. However, despite the security of the communications, there are still challenges such as inference threats on encrypted traffic which requires new features to be introduced in the tunnel in order to mitigate the threats. The following sections describe in more detail the IoT system and the layer 2 TLS tunnel used as well as new enhancements aimed to modify the traffic that flows through the tunnel, making it more difficult for an eavesdropper to infer legacy device behavior.

3.1 IoT System

Security from IoT systems doesn't just have to keep us informed – it can play an active role in keeping us safe. The integration of IoT into industry has increasingly provided great benefits for safety and protection of human lives that should not be dismissed. The IoT system used in this thesis consists of two legacy devices and a legacy control application, presented in Figure 3.1 belonging to a legacy commercial, multi-node, embedded system that is aimed at lifesafety. The legacy control application issues commands and listens to the device responses. All communication between legacy devices and between devices and control application is based on a proprietary protocol over UDP. Since this system was originally designed to operate on an isolated network, i.e. a network that is supposed to be secure, this would not require major communications security concerns, thus the traffic being generated in the network would not have any kind of security. Only plaintext messages would be exchanged and as such an intruder would easily have access to the information. In a more advanced stage, where the plan of this dissertation fits, the setup in Figure 4.1 is considered in order to operate on a shared (potentially unsecured) network, which brings a higher set of challenges to ensure security and provide the same system performance.

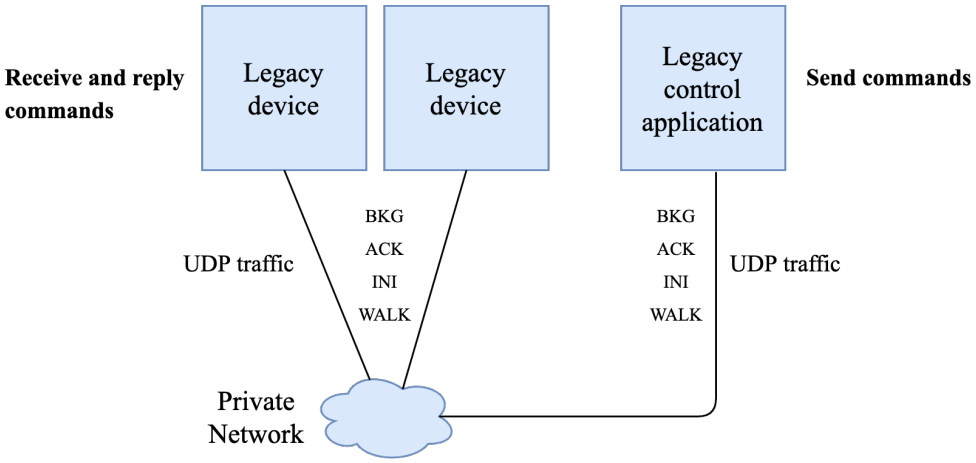

Figure 3.1: IoT system setup.

3.2 Existing L2 TLS Tunnel

A TLS tunnel is intended to send IP packets over a secure connection. The existing TLS tunnel used in this thesis presents a typical client-server architecture. Figure 3.2 shows the different components of the TLS tunnel, namely the internal components on the TLS tunnel server and client and how they connect to the legacy devices through the private network and public networks. The tunnel server supports one or more TLS tunnel clients, allowing inbound and outbound device traffic flowing through the private network. It also has packet-level security controls such as packet filtering and rate limiting, and for access control through client and server certificates.

Although the tunnel provides secure communications through traffic encryption, it does not address inference attacks on encrypted tunnel traffic. The encapsulation block, presented on Figure 3.2, is where padding and other privacy-preserving solutions may be implemented in order to avoid the identification of legacy application behavior, which are presented in the following subsection 3.3.

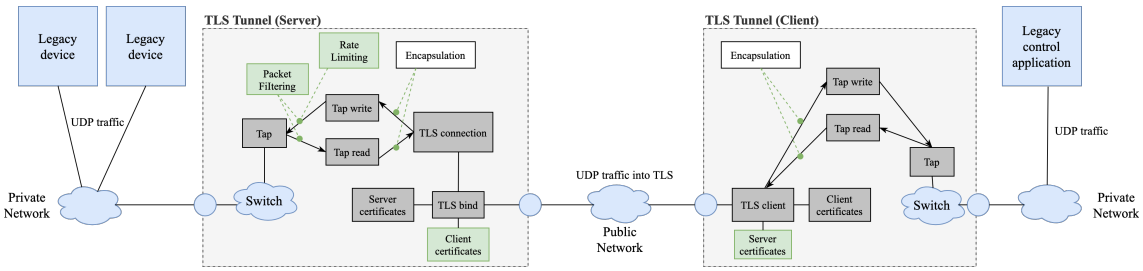

Figure 3.2: Schematic of the IoT system and TLS tunnel architecture and components.

3.3 Improvements for detection avoidance

When it comes to preventing inference threats traffic encryption alone is not enough. While the TLS tunnel only makes the traffic attack a slightly more difficult, it offers no guarantee of privacy

[29]. A network eavesdropper could still use sensitive information from the encrypted traffic in machine learning to infer legacy device behavior. Traffic shaping is a feasible solution that can avoid detecting device behavior through the generated TLS traffic and therefore make the attack practically impossible.

For this purpose, we included a module on the tunnel consisting of basic constant and random padding mechanisms and dummy traffic generation. By deploying this to the tunnel it may modifies the tunnel traffic with the goal of avoiding the detection of legacy application behavior. The following subsections describe the goals and the approach that was implemented with the purpose of avoiding the behavior detection.

3.3.1 Avoidance goals

In order to avoid inference threats, two different goals were considered: a) introducing dummy packets which are fake generated packets injected into the network to make it difficult to perceive real traffic and deterministic padding that consists of padding a message payload of a given length to form a message of a particular corresponding output length, making the attacker falsely believe a certain behavior is occurring or b) masking all the network traffic with dummy traffic and padding in a randomized way, which makes an eavesdropper unable to accurately infer the behaviors that are actually taking place.

For the first goal, a *targeted approach* is followed: when IoT devices are performing a certain behavior, dummy packets and padding are added with the same length as the responses from a different behavior, ideally fooling the classifier's accuracy by 100% and making the attacker always believe that it is dealing with the second behavior.

For the second goal, a *non-targeted approach* is used that unlike the previous methodology does not require detailed information about the behaviors to imitate. This approach introduces randomness to the dummy packets and padding and therefore being applicable to all types of behavior. As a consequence of the added randomness, its expect that all valuable traffic information is hidden behind non-existent patterns. Leading the classifiers to predict in the dark and resulting in a 50% classification accuracy for each binary classifier, as good as coin-flipping.

3.3.2 Avoidance approach

The threat model consists of learning the behaviors of a given system from the traffic generated by its devices. In order to mitigate the privacy implications of traffic analysis attacks we consider two methods to hide predictable traffic patterns: packet padding and dummy traffic generation.

Even if end-to-end encrypted communications are used, the attacker can still gain knowledge of the amount of traffic that has been generated. One way to obfuscate the amount of traffic is to modify the length of sensitive data, designated as true payload, that is contained in the messages that are being sent. Adding padding to an encrypted message can make traffic analysis more difficult by obscuring the true payload length. The padded length can be a deterministic or random value that prevents an attacker from knowing the exact length of the message.

Considering T_i as the true payload size of packet i and λ_i as the random value set for the size of packet i,

$$\lambda_i = randomInt(\lambda_{min}, \lambda_{max}),$$

then δ_i corresponds to the number of bytes to be added to packet i,

$$\delta_i = \lambda_i - T_i.$$

Dummy traffic generation consists of introducing fake packets into the network to make it harder for an adversary to detect the real behaviors. To produce the payload of the dummy packets we used the python **Secrets** [35] module, which generates a random secure byte string, giving access to the most secure source of randomness. The length of the dummy payload can be a deterministic or random value. We prioritize real traffic over dummy traffic to ensure that there is no extra latency and that IoT devices can perform regularly. Also, dummy traffic follows the same path as real traffic and when it reaches the destination host (which can be the client or the tunnel server), dummy traffic is silently dropped. Sending dummy packets is determined based on a function that returns 0 or 1 randomly. If the function returns a 1 then a dummy packet is sent.

Figure 3.3 represents format we established for TLS packets flowing within the tunnel. The packet header consists of 3 mandatory bytes, the first byte **SIG** represents the type of the packet:

- **SIG =** $0000\ 0000_2$ representing a real packet without padding;
- **SIG =** $0000\ 0001_2$ representing a dummy packet without padding;
- **SIG =** $0000\ 0010_2$ representing a real packet with padding;
- **SIG =** $0000\ 0011_2$ representing a dummy packet with padding.

Then, the next two bytes (Payload length) correspond to the true size of the payload. For packets with padding, they require an extra two bytes in the header (Padding length) which correspond to the length of the padding added at the end of the packet. Similarly to how dummy packets are dropped, when the padding packets reach the destination address, the receiver only handles the true payload content. The padding bytes are also dropped, since these bytes are only used to mask the packets passing in the tunnel and do not represent any useful information.

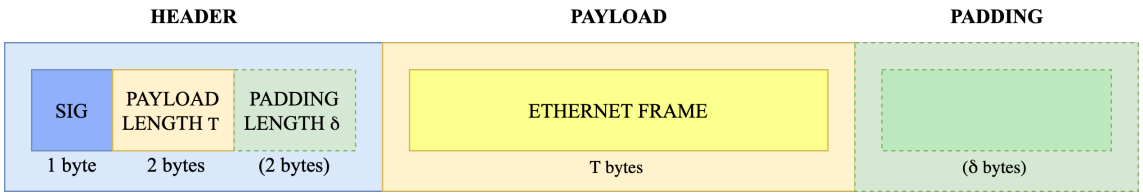

Figure 3.3: Packet format.

Chapter 4

Tunnel and behavior detection

Once the TLS tunnel is up and running, it is essential to understand how it can be more or less detectable by an attacker eavesdropping the network traffic. In this case, the nature of the TLS traffic can be used to a) distinguish tunnel traffic from other IoT traffic and b) to distinguish tunnel traffic with different legacy application behaviors.

In the following sections, we describe the different types of device behavior as well as the processing of the collected data that is used to evaluate the performance of the implemented detection model based on machine learning.

4.1 Devices and traffic behavior

The classification model was developed in order to perform an inference threat on legacy devices traffic, collecting traffic data from local devices – specifically a life safety detection system – and also using IoT traffic datasets publicly available from the University of New South Wales (UNSW)[36].

Figure 4.1 shows the setup used to generate legacy IoT tunnel traffic. On the top-left corner, Figure 4.1 represents the legacy devices that generate background traffic in the private network, which is consistently sent through the tunnel. The various commands that the application sends have distinct traffic profiles and correspond to the different behaviors that are intended to be identified.

This work focuses on a set of four device behaviors: one (**BKG**) corresponds to the background traffic generated by the devices in the lack of commands from the legacy control application, and the other three are the following device-specific commands:

- **ACK** (*Acknowledge*) – command used to recognize events or alarms previously registered by the system;
- **INI** (*Inhibit*) – command that forces sensors or actuators to be ignored or bypassed, preventing them from triggering or responding to alarms or events;

- **WALK** (*Walktest*) – command that makes the system enter a testing mode, in which sensors may be tested and activated without triggering false alarms.

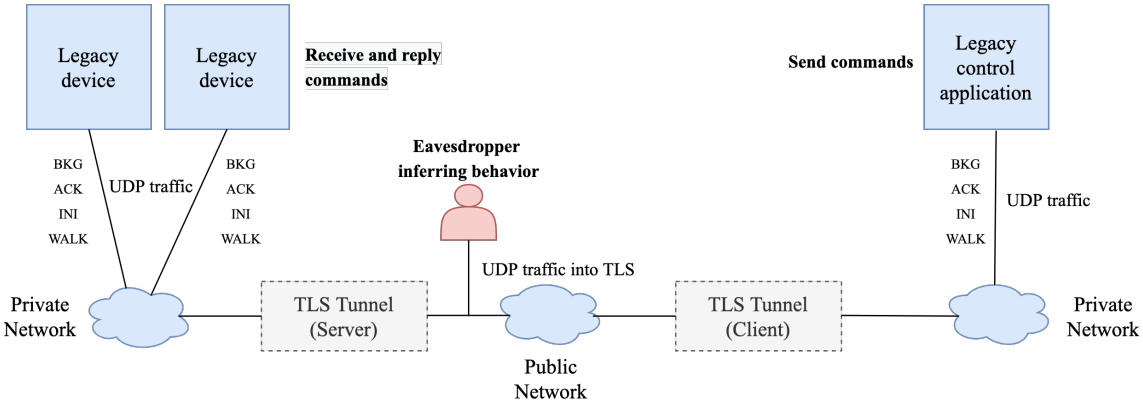

Figure 4.1: Setup for IoT tunnel traffic generation.

4.2 Statistical features

The feature extraction focuses on statistical properties of network traffic since only encrypted communications are addressed which makes payload analysis unfeasible. All traffic capture processing is done using Tshark [37] which is the command line version of Wireshark.

A Python script is built to run Tshark (see code 4.1 and 4.2) for every pcap file obtained. We record packet sizes, inter-arrival delay and direction – inbound or outbound – for each IP packet and record sizes, inter-arrival delay, record type and direction for each TLS record (Table 4.1) for each traffic flow converting it into a CSV (comma-separated value) file. Tshark does not return the packet and record directions and as such it was necessary to implement another script that identifies the source and destination address and assigns a label 0 or 1 to the packet or record according to the direction in the flow.

```
tshark -r 'data.pcap' -Y "ip and tcp" -T fields -E header=y -e frame.time_epoch
    -e tcp.stream -E separator=, -e ip.src -e ip.dst -e ip.len -e tcp.
    time_delta
```

Listing 4.1: Capture filters for IP packets

```
tshark -r 'data.pcap' -Y "ssl" -T fields -e frame.time_epoch -E header=y -E
    separator=',' -E occurrence=a -E aggregator=";" -e tcp.stream -e ip.src -e
    ip.dst -e tls.record.length -e tcp.time_delta -e tls.record.content_type
```

Listing 4.2: Capture filters for TLS records

Figure 4.2 presents a simplified scheme of the implemented data pre-processing techniques. When the traffic is collected using Wireshark, the data pre-processing starts with the feature extraction using Tshark. For data segmentation, we adopted a sliding time window approach that slides across all the data stream according to a predefined time interval.

IP packet features	
IP.bytes	IP packet length
IP.ipt	IP packet inter-arrival delay
IP.dir	IP packet direction

TLS record features	
TLS.bytes	TLS record length
TLS.ipt	TLS record inter-arrival delay
TLS.type	TLS record content type
TLS.dir	TLS record direction

Table 4.1: IP packet and TLS record features

The sliding window analysis was adopted because the TLS tunnel traffic, unlike other types of traffic, consists of a single and extremely long TCP flow. There are several advantages of using sliding time windows such as a) being able to find different behaviors in the same TCP flow or even b) classify a TCP connection for which the beginning has not been seen. All analyses used $\omega = 500$ms as window size for traffic behavior identification. Figure 4.3 illustrates the traffic segmentation using a sliding time window approach. Each window of 500ms consists of the IP packets and TLS records features captured in each time interval. To segment the flow by time intervals, we define a temporal cumulative variable for the packets and a second one for the TLS records. Both temporal cumulative variables are updated based on the values collected from the delta time of the IP's and TLS records.

We designed the legacy control application such that it issues the three commands – *ACK*, *INI* and *WALK* – periodically every 400ms. Figure 4.4 represents a portion of each feature of the traffic behaviors being distinguished, after applying the pre-processing techniques. Each row in Figure 4.4 represents a window with a maximum of 50 IP packets and 38 TLS records, resulting in 150+152 features which are then fed as input for classification.

The length and direction of IP packets (*IP.bytes* and *IP.dir*) seem to exhibit an alternating behavior, due to the TCP acknowledgments. When ignoring these TCP artifacts and focusing on tunnel's TLS records (*TLS.bytes* and *TLS.dir*), it is possible to see that the background behavior has an almost constant server-to-client traffic, while other behaviors exhibit different patterns in length and direction. These patterns are visible even while using tunneling and encryption, and thus may allow behaviors to be differentiated by traffic classifiers.

4.3 Machine Learning model

Before starting to build a machine learning model, its crucial to know what the problem that is going to be solved. Machine learning models can be used to discover the types and behaviors of IoT devices transmitting traffic in the network. Device behavior fingerprinting is performed using only statistical features extracted from encrypted network traffic. Therefore, when accessing network traffic, an adversary can use traces to train their machine learning models and infer what behaviors are being generated by the devices on the network.

Listing 4.3 shows the python code for the model we use in this work. The model is a dense neural network with 3 hidden layers – the first, second and third hidden layers have 256, 128 and 64 nodes, respectively – these three layers use the relu activation function. The output layer

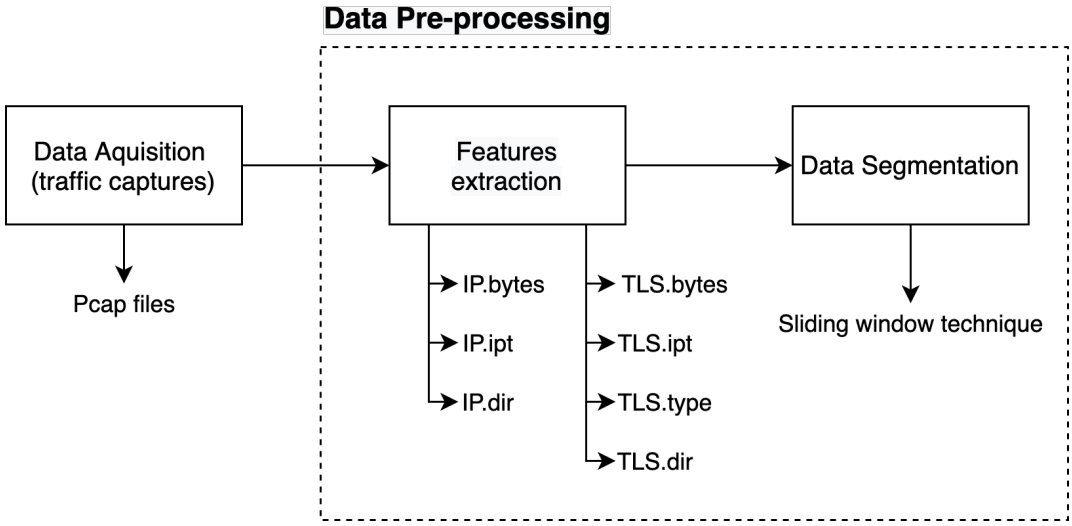

Figure 4.2: Simplified illustration of data processing.

has only one node and uses the sigmoid activation function. Using a sigmoid on the output layer guarantee that the network output is between 0 and 1. The dropout layer existing between each of the above layers randomly sets input units to 0 with a rate frequency at each step during the training time, helping to avoid overfitting. It randomly sets inputs to 0 with a frequency of 0.2.

We trained the model with an Adam optimization algorithm which has the particularity of tuning itself automatically, allowing good results on a wide range of problems. To calculate the loss we used the binary cross-entropy function.

```
def get_compiled_model():
  model = tf.keras.Sequential([
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(1, activation='sigmoid')
  ])
  model.compile(optimizer='adam',
                loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
                metrics=['accuracy'])
  return model
```

Listing 4.3: Machine Learning model used in this work

Figure 4.5 outlines the entire Machine Learning process. The first step in the ML process is to obtain the data from multiple sources followed by a process of fine tuning the data and also labelling. Since this is a binary classification (only 2 classes), the data labeling process assigns 0 for one of the classes and 1 for the other class. Then, this data is randomly split into two sets: 70% of the data is used for training and 30% for testing. Later the ML model is fed with the training

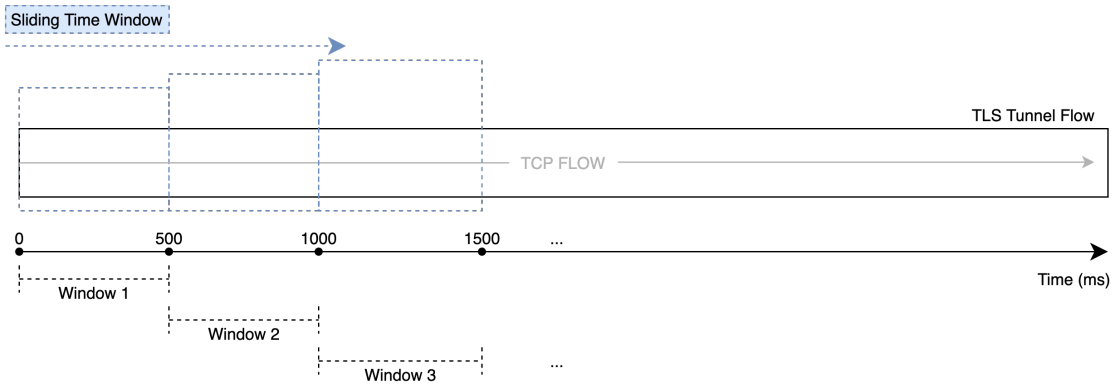

Figure 4.3: Sliding Time Window.

dataset in order for it to be learned. Once the ML model is learned, it can be saved and tested with any test dataset.

The ML classifier is built in a docker container with the following applications and libraries installed:

- Docker base image: tensorflow/tensorflow:latest-jupyter
- Python version: 3.6.9
- Jupyter Notebook version: 6.2.0
- Tensorflow version: 2.4.1
- Pandas version: 1.1.5
- Numpy version: 1.19.5
- Tshark version: 3.2.3

Multiple measurements can be used to evaluate the performance of a given model. Assuming a binary classifier such as the one trained to detect tunnel traffic, four possible combinations of predicted label and true label are possible, usually presented through a confusion matrix (Table 4.2), with a window of tunnel traffic considered "Positive" and a window of other type of traffic "Negative":

- True Positive (TP): a tunnel traffic window that is correctly identified by the detector;
- True Negative (TN): a non-tunnel traffic window correctly identified by the detector;
- False Positive (FP): a non-tunnel traffic window that the detector predicts as being a tunnel traffic window;
- False Negative (FN): a tunnel traffic window that the detector predicts as being a non-tunnel traffic window.

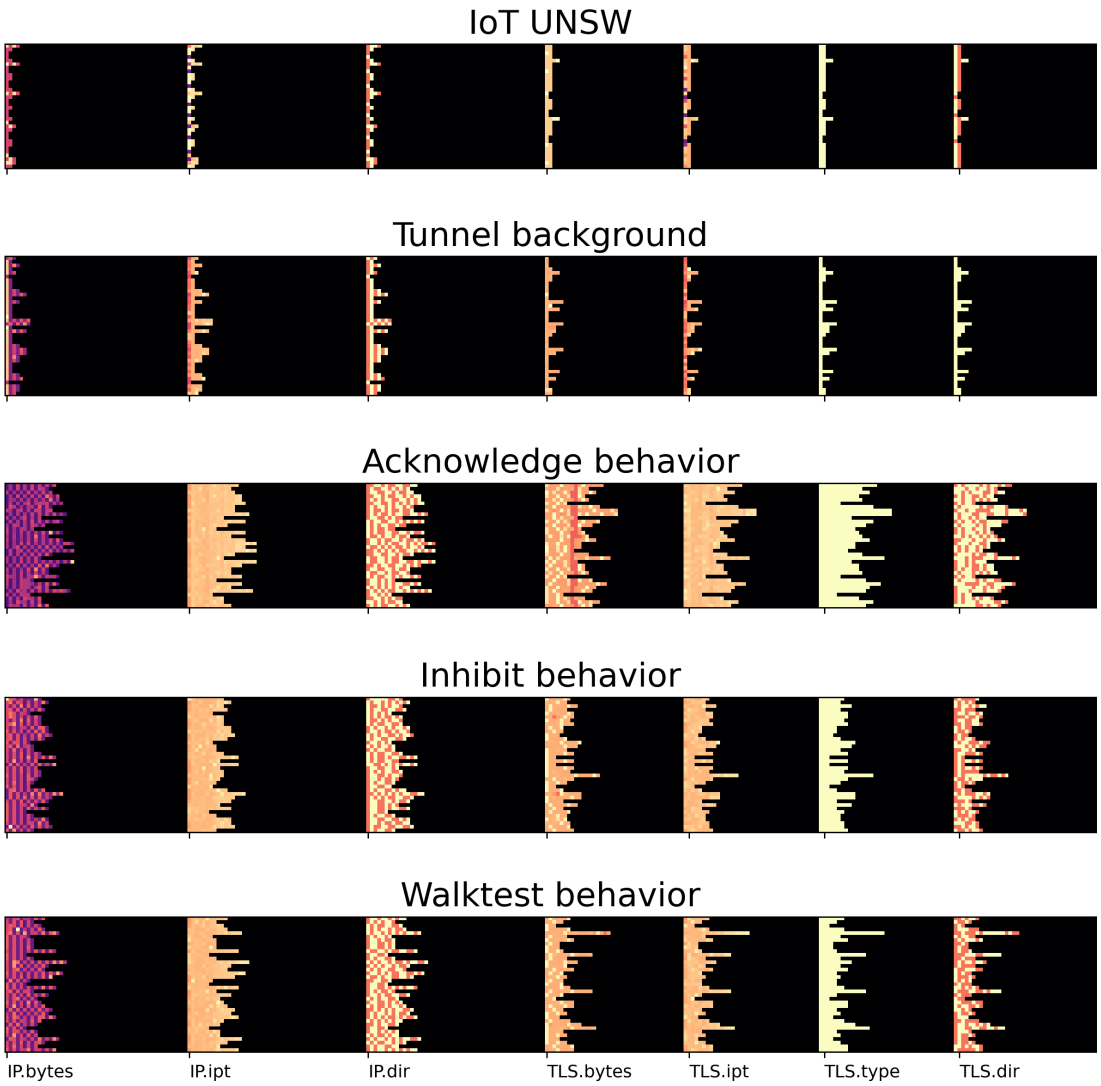

Figure 4.4: Classifier input for the different behaviors

This can also be applied to detect different device behaviors. Assuming that the model is trained to detect device *behavior A* and considering *behavior A* and *behavior B*, four possible combinations of predicted label and true label are possible, with a window of *behavior A* considered "Positive" and a window of *behavior B* "Negative":

- True Positive (TP): a *behavior A* window that is correctly identified by the detector;
- True Negative (TN): a *behavior B* window correctly identified by the detector;
- False Positive (FP): a *behavior B* window that the detector predicts as being a *behavior A* window;
- False Negative (FN): a *behavior A* window that the detector predicts as being a *behavior B* window.

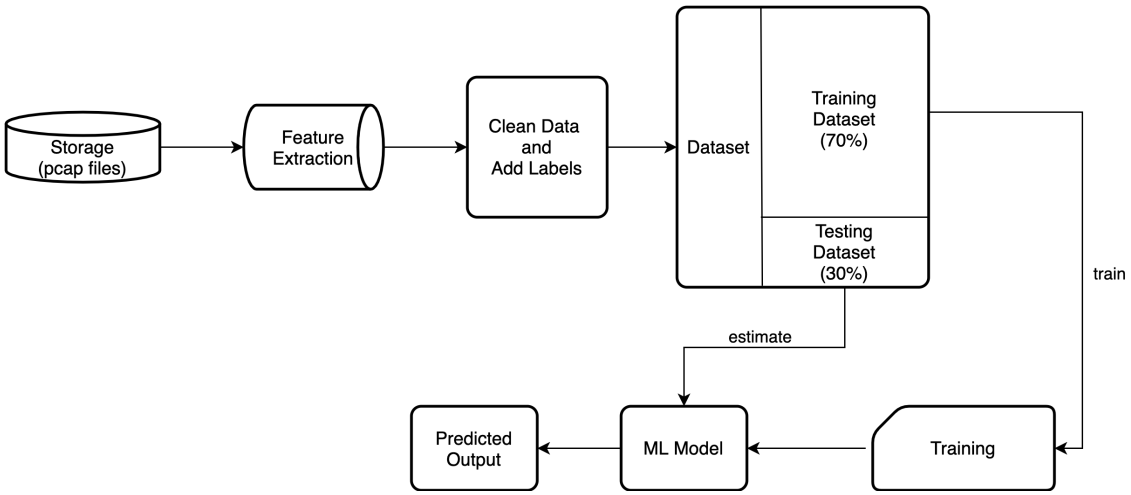

Figure 4.5: Process to build a ML model

True lable / Predition	**Negative**	**Positive**
Negative	True Negative (TN)	False Negative (FN)
Positive	False Positive (FP)	True Positive (TP)

Table 4.2: Confusion matrix for binary classification

One of the most commonly used metric in classification problems is *Accuracy* which is defined as the percentage of correct predictions for the test data. It is the most intuitive performance measure and is simply a ratio of the correctly predicted observation to the total observations, being computed as:

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

The detection results of the implemented model are covered and discussed in more detail in the next Section 5.

Chapter 5

Results

Since the goal of this work is to decrease the ability of an eavesdropper to infer the behavior of legacy devices through traffic obfuscation techniques such as padding and dummy packets it is then necessary to realize and evaluate the performance of the developed traffic classifier before and after of applying these techniques. In Section 5.1 we evaluate the performance of the classifier in identifying background tunnel traffic versus IoT UNSW traffic and in distinguishing the various behaviors generated by the legacy device. In Section 5.2, the performance of the classifier is evaluated again, this time with the mitigation mechanisms implemented in the *targeted* and *non-targeted* approaches described in Section 3.3.

5.1 Detection results

The classification of traffic belonging to IoT devices from UNSW datasets versus locally generated tunnel background traffic yielded an accuracy of 97% (Table 5.1) on almost 4k UNSW and tunnel window samples each.
These results demonstrate that it is possible to build a classifier that can fairly accurately identify UNSW IoT device traffic from the background traffic that our legacy devices generate over the tunnel. The existence of specific patterns in the features of each traffic type (e.g., dissimilar values for the inter-arrival delay in the UNSW dataset compared to the tunnel) help differentiating traffic belonging to the tunnel from other IoT devices (UNSW).

	Accuracy	**Number of Samples per class**
UNSW vs. Tunnel	0.97	3973

Table 5.1: Detection accuracy from UNSW dataset and tunnel background dataset

A further analysis performed was regarding the different behaviors produced by our legacy devices. Table 5.2 shows the accuracy results of behavior detection experiments. The different behaviors generated by the local IoT devices are identified successfully by the developed classifier

with extremely high accuracy based on the statistical features of the network traffic. In one particular case – WALK vs. INI – the created model yield poor classification results due to the high similarity in the collected features, particularly the length of the requests and responses.

Device behaviors	Accuracy	Number of samples per class
BKG vs. INI	0.97	4964
BKG vs. ACK	0.99	5013
BKG vs. WALK	0.98	5064
ACK vs. INI	0.91	5080
ACK vs. WALK	0.93	4926
WALK vs. INI	0.63	4732

Table 5.2: Device behavior detection results

5.2 Avoidance results

Avoidance results comprise the evaluation of both implemented approaches, which have been covered thoroughly in Section 3.3. Starting by the *targeted approach*, we trained a model to distinguish the traffic originally generated by ACK and INI behaviors. Recall that the original classification results from Table 5.2 list an accuracy of 91%, showing that the classifier is able to differentiate and identify traffic from either one of these two behaviors without padding or dummy traffic on the tunnel.

By observing the traffic related to the ACK behavior, we developed a strategy using dummy packets with exactly the same length as the ACK traffic, which are inserted into the INI behavior traffic. Having both types of traffic – true INI and fake ACK – the classifier used in Table 5.2 has proven to be mislead and its accuracy degrades to 34%. We then compared this approach with only padding and verified that the classifier is even more mislead, resulting in an accuracy of 9%. These two strategies combined – inserting dummy packets and padding into the INI traffic to imitate ACK traffic – induces the classifier in error even more, yielding a classification accuracy of 6%. This shows that it is possible to mask a behavior with both padding and dummy packets. These classification results obtained for the *targeted approach* are shown in figure 5.1.

Behavior classification results employing the *non-targeted approach* are presented in Figure 5.2. In order to avoid detection by a classifier, we introduce randomness in both padding and dummy packets inserted into the tunnel traffic. In a first approach (designated as 'Normal range obfuscation') thresholds of $\lambda_{min} = 100$ and $\lambda_{max} = 192$ bytes are set as limits. In this case, the classifier behaves like a coin flipping experiment and fails to detect the legacy device behavior,

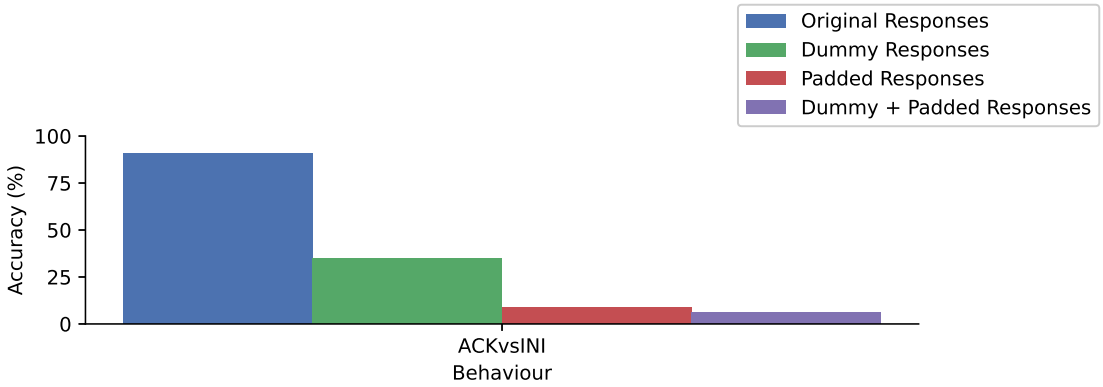

Figure 5.1: Targeted approach results

approaching 50% accuracy. In the same figure it is possible to verify that even if the eavesdropper retrains its model considering the padding and dummy traffic the classification results do not improve. Training the model with datasets that already considers traffic obfuscation results is a weak and not useful learning, demonstrating that traffic obfuscation prevents the detection of patterns that allow the identification of different behaviors. Figure 5.3 compares traffic tunnel with and without padding and obfuscation for the 4 device behaviors. 3 in 4 behaviors without padding or dummy packets are clearly distinguishable (with INI and WALK overlapping in the top figure), but no longer when introducing padding and dummy packets (bottom figure).

Finally, we evaluated the performance of the classifier (results in Figure 5.4) using the *non-targeted approach* differentiating ACK and INI behaviors for different obfuscation ranges λ_{min} and λ_{max}, as can be seen in Figure 5.5. The low obfuscation range corresponds to a $\lambda_{min} = 100$ and $\lambda_{max} = 128$ bytes and for which the classifier yields an accuracy close to 70%. This level of obfuscation also corresponds to a larger number of bytes used when compared to the original traffic. High obfuscation range corresponds to a $\lambda_{min} = 100$ and $\lambda_{max} = 256$ bytes limit which induces the classifier into significantly more error lowering its accuracy to close to 50%. This level of obfuscation corresponds to the largest number of bytes used, when compared to the original traffic and to the low obfuscation traffic. Results of *Normal range obfs.* are also show in the same image for comparison.

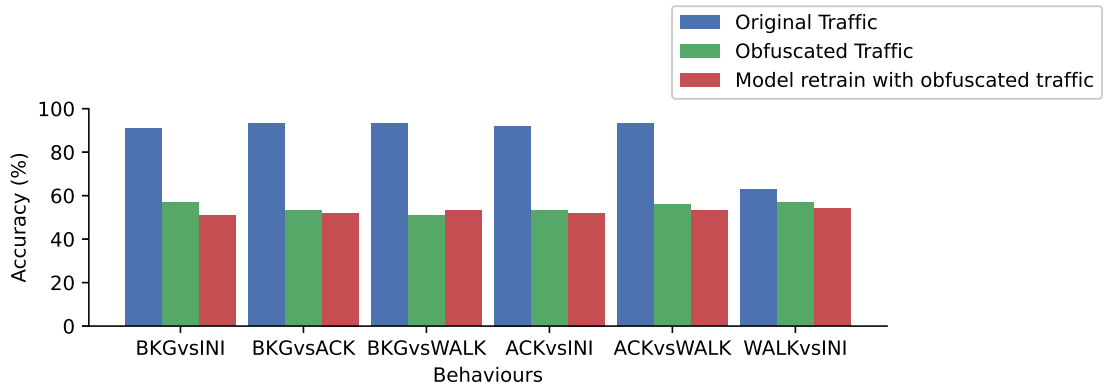

Figure 5.2: Non-targeted approach results: accuracy while discriminating different behaviors.

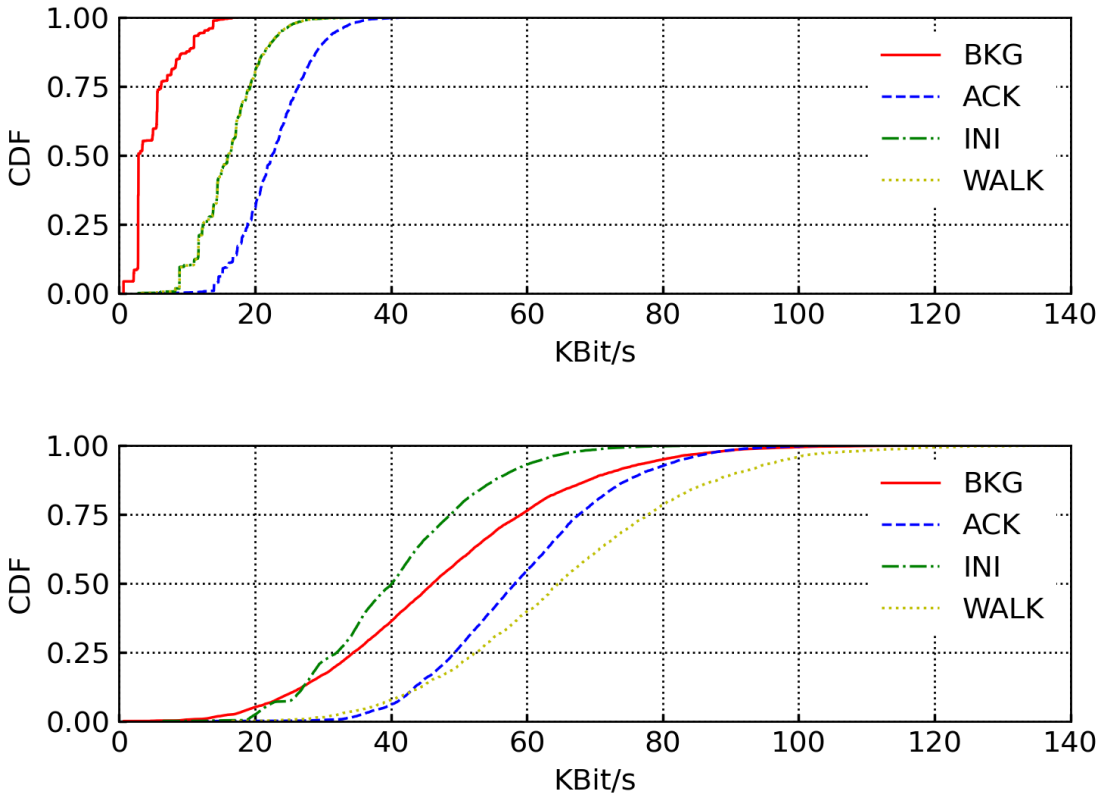

Figure 5.3: Non-targeted approach results: bitrate CDF for each behavior: top figure representing original traffic (no obfuscation), bottom figure with normal range obfuscation.

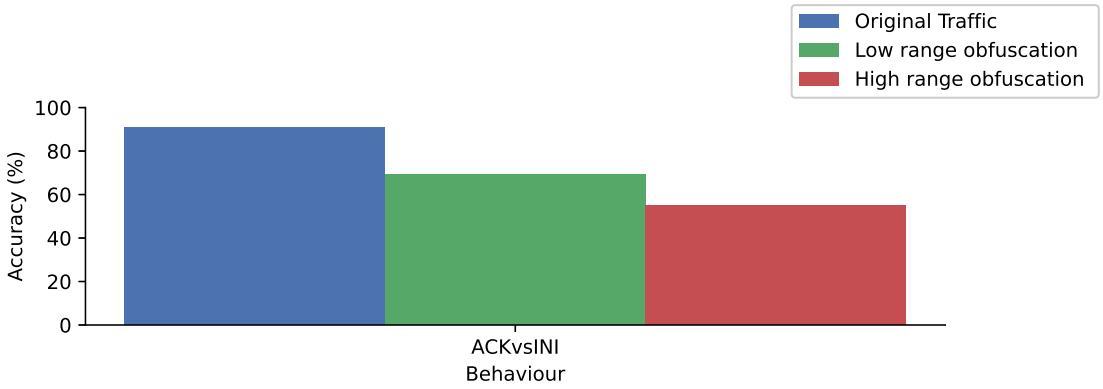

Figure 5.4: Non-targeted approach results: accuracy while differentiating ACK and INI behaviors for different obfuscation ranges

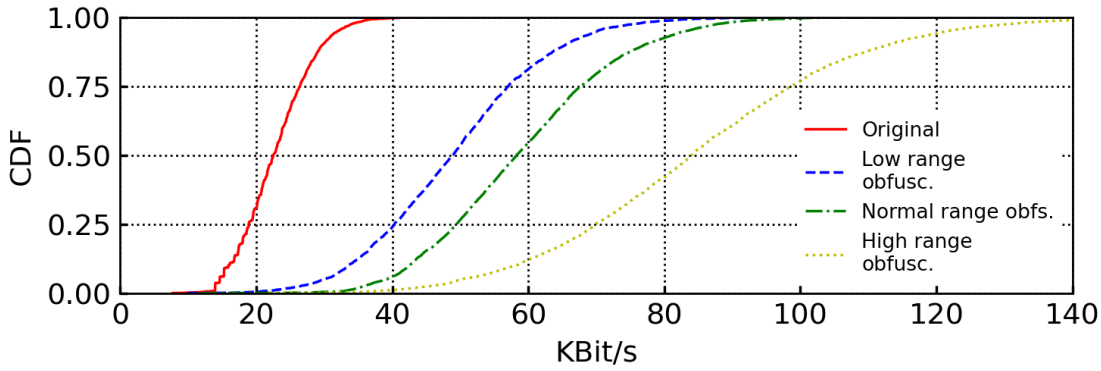

Figure 5.5: Non-targeted approach results: bitrate CDF for ACK and INI mixed traffic.

Chapter 6

Conclusion

The work presented in the previous chapters aimed to determine whether it is possible to detect legacy device traffic and to infer legacy device behavior even if the network traffic is fully encrypted, as is the case with the tunnel. To illustrate how these inference threats can be mitigated we developed traffic morphing techniques with the goal of avoiding the detection of legacy device behavior.

Using a existing layer 2 tunnel over TLS for securing legacy IoT device traffic has shown that encrypted traffic alone does not prevent device behavior detection. In order to mitigate this threat, in Chapter 3 we presented our improvements that changes tunnel traffic to avoid behavior detection. We showed that detection avoidance is feasible with constant and random padding and dummy packet generation, and also characterized the increase in tunnel traffic when these detection avoidance techniques are used to avoid detection of life safety system behavior.

The detection results obtained in Chapter 5 show that it is, in fact, possible to distinguish tunnel traffic from non-tunnel traffic like native secure IoT traffic (UNSW IoT dataset) as well as distinguish different behaviors of our legacy IoT system using a Machine Learning-based classifier described in Chapter 4. When it comes to detection avoidance, in Chapter 5 we also present the avoidance results showing that introducing traffic morphing techniques, as intended, lowers the ability of an eavesdropper to infer legacy device behavior. We have proved that this is true even if the eavesdropper retrains its model with padding and dummy traffic.

All things considered, securing existing legacy device traffic is a necessity when the assumptions used in device security design no longer hold. When it is not possible to redesign the legacy device, combining a TLS proxy with traffic morphing techniques such as those proposed in this dissertation can be used to avoid the detection of device behaviors, while preventing the entire life safety system from being compromised as well as the security of its users.

6.1 Future Work

There is still much to be explored in the field of inference threats on encrypted traffic and traffic morphing techniques in order to make it harder for the tunnel traffic to be distinguished from

other that is likely to coexist in the network where the legacy devices are deployed. We intend to perform and study inference attacks using other behaviors generated by our legacy IoT devices, as well as using different IoT devices. In addition, our aim is to test different time window sizes and different time values to the application's command sending periodicity observing the impact on the behavior classification. We also expect to explore the impact of newer TLS version 1.3 on detection avoidance and the impact of using datagram TLS (DTLS) on the tunnel instead of normal TLS on tunnel round trip times.

References

[1] I. Hafeez, M. Antikainen, and S. Tarkoma. Protecting IoT-environments against traffic analysis attacks with traffic morphing. In *2019 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)*, pages 196–201, 2019. doi:10.1109/PERCOMW.2019.8730787.

[2] W. Zhou, Y. Jia, A. Peng, Y. Zhang, and P. Liu. The effect of IoT new features on security and privacy: New threats, existing solutions, and challenges yet to be solved. *IEEE Internet of Things Journal*, 6(2):1606–1616, 2019. doi:10.1109/JIOT.2018.2847733.

[3] N. Msadek, R. Soua, and T. Engel. IoT device fingerprinting: Machine learning based encrypted traffic analysis. In *2019 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–8, 2019. doi:10.1109/WCNC.2019.8885429.

[4] N. Zainuddin, M. Daud, S. Ahmad, M. Maslizan, and S. A. L. Abdullah. A Study on Privacy Issues in Internet of Things (IoT). In *2021 IEEE 5th International Conference on Cryptography, Security and Privacy (CSP)*, pages 96–100, 2021. doi:10.1109/CSP51677.2021.9357592.

[5] M. R. Shahid, G. Blanc, Z. Zhang, and H. Debar. IoT Devices Recognition Through Network Traffic Analysis. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 5187–5192, 2018. doi:10.1109/BigData.2018.8622243.

[6] 5th Cyber Security in Networking Conference - CSNet, 2021. URL: https://csnet-conference.org/2021/index.php/call-for-papers/.

[7] V. Hassija, V. Chamola, V. Saxena, D. Jain, P. Goyal, and B. Sikdar. A survey on IoT security: Application areas, security threats, and solution architectures. *IEEE Access*, 7:82721–82743, 2019. doi:10.1109/ACCESS.2019.2924045.

[8] Y. Yang, L. Wu, G. Yin, L. Li, and H. Zhao. A survey on security and privacy issues in Internet-of-Things. *IEEE Internet of Things Journal*, 4(5):1250–1258, 2017. doi:10.1109/JIOT.2017.2694844.

[9] Z. 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*, pages 230–234, 2014. doi:10.1109/SOCA.2014.58.

[10] M. M. Hossain, M. Fotouhi, and R. Hasan. Towards an analysis of security issues, challenges, and open problems in the Internet of Things. In *2015 IEEE World Congress on Services*, pages 21–28, 2015. doi:10.1109/SERVICES.2015.12.

[11] Y. Meng, W. Zhang, H. Zhu, and X. S. Shen. Securing consumer IoT in the smart home: Architecture, challenges, and countermeasures. *IEEE Wireless Communications*, 25(6):53–59, 2018.

[12] J. Manyika, M. Chui, P. Bisson, J. Woetzel, R. Dobbs, J. Bughin, and D. Aharon. Unlocking the potential of the Internet of Things. *McKinsey Global Institute*, 2015.

[13] Who is the OWASP® foundation? URL: https://owasp.org/.

[14] S. Choi, C. Yang, and J. Kwak. System hardening and security monitoring for IoT devices to mitigate IoT security vulnerabilities and threats. *KSII Transactions on Internet & Information Systems*, 12(2), 2018.

[15] N. Apthorpe, D. Reisman, and N. Feamster. A smart home is no castle: Privacy vulnerabilities of encrypted IoT traffic, 2017.

[16] M. M. Noor and W. H. Hassan. Current research on Internet of Things (IoT) security: A survey. *Computer Networks*, 148:283 – 294, 2019. doi:https://doi.org/10.1016/j.comnet.2018.11.025.

[17] R. T. Tiburski, L. A. Amaral, E. de Matos, D. F. G. de Azevedo, and F. Hessel. Evaluating the use of TLS and DTLS protocols in IoT middleware systems applied to E-health. In *2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC)*, pages 480–485, 2017. doi:10.1109/CCNC.2017.7983155.

[18] S. Zheng, N. Apthorpe, M. Chetty, and N. Feamster. User perceptions of smart home IoT privacy. 2(CSCW), 2018. doi:https://doi.org/10.1145/3274469.

[19] Y. H. Hwang. IoT security privacy: Threats and challenges. New York, NY, USA, 2015. Association for Computing Machinery. doi:https://doi.org/10.1145/2732209.2732216.

[20] X. Gong, N. Kiyavash, and N. Borisov. Fingerprinting websites using remote traffic analysis. New York, NY, USA, 2010. Association for Computing Machinery. doi:https://doi.org/10.1145/1866307.1866397.

[21] A. J. Pinheiro, J. M. Bezerra, and D. R. Campelo. Packet padding for improving privacy in consumer IoT. In *2018 IEEE Symposium on Computers and Communications (ISCC)*, pages 00925–00929, 2018. doi:10.1109/ISCC.2018.8538744.

[22] Q. Sun, D. R. Simon, Yi-Min Wang, W. Russell, V. N. Padmanabhan, and Lili Qiu. Statistical identification of encrypted web browsing traffic. In *Proceedings 2002 IEEE Symposium on Security and Privacy*, pages 19–30, 2002. doi:10.1109/SECPRI.2002.1004359.

[23] M. Shen, Y. Liu, S. Chen, L. Zhu, and Y. Zhang. Webpage fingerprinting using only packet length information. In *ICC 2019 - 2019 IEEE International Conference on Communications (ICC)*, pages 1–6, 2019. doi:10.1109/ICC.2019.8761167.

[24] C. V. Wright, S. E. Coull, and F. Monrose. Traffic morphing: An efficient defense against statistical traffic analysis. In *NDSS*, volume 9. Citeseer, 2009.

[25] L. Lu, E. Chang, and M. C. Chan. Website fingerprinting and identification using ordered feature sequences. In Dimitris Gritzalis, Bart Preneel, and Marianthi Theoharidou, editors, *Computer Security – ESORICS 2010*, pages 199–214, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[26] A. Mani, T. Wilson-Brown, R. Jansen, A. Johnson, and M. Sherr. Understanding Tor usage with privacy-preserving measurement. New York, NY, USA, 2018. Association for Computing Machinery. doi:https://doi.org/10.1145/3278532.3278549.

[27] A. Cuzzocrea, F. Martinelli, F. Mercaldo, and G. Vercelli. Tor traffic analysis and detection via machine learning techniques. In *2017 IEEE International Conference on Big Data (Big Data)*, pages 4474–4480, 2017. doi:10.1109/BigData.2017.8258487.

[28] A. Almubayed, H. Ali, and J. Atoum. A model for detecting Tor encrypted traffic using supervised machine learning. *International Journal of Computer Network and Information Security*, 7(7):10, 2015.

[29] M. Skowron, A. Janicki, and W. Mazurczyk. Traffic fingerprinting attacks on Internet of Things using machine learning. *IEEE Access*, 8:20386–20400, 2020. doi:10.1109/ACCESS.2020.2969015.

[30] J. Chen, F. Miao, and Q. Wang. SSL/TLS-based secure tunnel gateway system design and implementation. In *2007 International Workshop on Anti-Counterfeiting, Security and Identification (ASID)*, pages 258–261, 2007. doi:10.1109/IWASID.2007.373739.

[31] N. Apthorpe, D. Reisman, S. Sundaresan, A. Narayanan, and N. Feamster. Spying on the smart home: Privacy attacks and defenses on encrypted IoT traffic, 2017.

[32] N. Apthorpe, D. Yuxing Huang, D. Reisman, A. Narayanan, and N. Feamster. Keeping the smart home private with smart(er) IoT traffic shaping. *Proceedings on Privacy Enhancing Technologies*, 2019(3):128 – 148, 01 Jul. 2019. doi:https://doi.org/10.2478/popets-2019-0040.

[33] O. Salman, I. H. Elhajj, A. Chehab, and A. Kayssi. A machine learning based framework for IoT device identification and abnormal traffic detection. *Transactions on Emerging Telecommunications Technologies*.

[34] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu. IoT security techniques based on machine learning: How do IoT devices use ai to enhance security? *IEEE Signal Processing Magazine*, 35(5):41–49, 2018. doi:10.1109/MSP.2018.2825478.

[35] Secrets — generate secure random numbers for managing secrets. URL: https://docs.python.org/3/library/secrets.html.

[36] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics. *IEEE Transactions on Mobile Computing*, 18(8):1745–1759, August 2018. URL: https://ieeexplore.ieee.org/document/8440758/, doi:10.1109/TMC.2018.2866249.

[37] Tshark - dump and analyze network traffic. URL: https://www.wireshark.org/docs/man-pages/tshark.html.