Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

Benjamin Andreas Ulsmåg

Private Information Exposed by the Use of Robot Vacuum Cleaner in Smart Environments

Master's thesis in MISEB
Supervisor: Jia-Chun Lin
Co-supervisor: Ming-Chang Lee
June 2023

NTNU
Norwegian University of
Science and Technology

Benjamin Andreas Ulsmåg

Private Information Exposed by the Use of Robot Vacuum Cleaner in Smart Environments

Master's thesis in MISEB
Supervisor: Jia-Chun Lin
Co-supervisor: Ming-Chang Lee
June 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Dept. of Information Security and Communication Technology

NTNU
Norwegian University of
Science and Technology

Private Information Exposed by the Use of Robot Vacuum Cleaner in Smart Environments

Benjamin Andreas Ulsmåg

01.06.2023

Problem Description

Title: Private Information Exposed by the Use of Robot Vacuum Cleaner in Smart Environments
Student: Benjamin Andreas Ulsmåg

The use of robot vacuum cleaners is rapidly increasing in all kinds of smart environments. Vendors are developing new smart features and APIs to allow integration of third party systems and expand functionality, smart phone applications are used to make it easier for users to personalize their robot vacuum cleaner experience. These applications are delivered through cloud services where commando and control is communicated between the smart environments and cloud services. This communication is generating network traffic in local wireless and cabled networks as well as on the Internet. The traffic generated by the robot vacuum cleaner reflects the actions made by users and can potentially expose user private information if eavesdropped.

Smart phone applications use encrypted end-to-end communication to mitigate the risk of exposing private information. This kind of security measure is implemented by the application itself and not the network infrastructure. Information about IP-addresses, packet lengths, ports and low level protocols will still be available for attackers carrying out network eavesdropping. The metadata and header information can reveal IoT actions and potentially expose user private information. This thesis aims to address and determine which kind of private information that can be exposed by carrying out passive eavesdropping attack on network traffic generated by a robot vacuum cleaner.

Supervisor: Jia-Chun Lin
Co-supervisor: Ming-Chang Lee

Abstract

Robot vacuum cleaners are popular IoT devices and are deployed in all kinds of smart environments. Integration with IoT systems introduce more security and privacy issues related to the operation of these devices. Vendors have developed smart phone applications where users can personalize cleaning or view information about the vacuum cleaner. This increase the integration between user's life and the robot vacuum cleaner, which potentially exposes private information. Industry standards include end-to-end encryption between the application, cloud service and robot vacuum cleaner to secure the private information exchanged. Regardless of encryption, network header metadata is still available through network eavesdropping attacks. In this project we investigated the potential private information exposed by this metadata. An Irobot Roomba i7 was deployed in two different smart environments where passive network eavesdropping was conducted during smart feature triggering. Analysis revealed that it was possible to attribute different events triggered on the Irobot Roomba i7, only based on metadata in the Internet traffic capture. Different signature-based detection algorithms are proposed, with a high detection rate. Wi-Fi and Internet capturing metadata were compared and similar patterns were identified, making the detection method applicable for Wi-Fi eavesdropping as well. This thesis covers the implementation, capturing and analysis of network traffic and proposes event detection algorithms.

Sammendrag

Robotstøvsugere er blitt populære IoT enheter og er mye brukt i ulike smarte miljøer. Integrasjon med andre IoT systemer skaper flere sikkerhets og personverns utfordringer ved bruken av disse. Produsenter har utviklet applikasjoner hvor brukere kan konfigurere rengjøring og se informasjon om robotstøvsugeren etter eget ønske. Dette øker integrasjonen mellom brukernes liv og robotstøvsugeren, noe som kan eksponere mer privat informasjon. Industristandarder bruker ende-til-ende kryptering av kommunikasjon mellom applikasjonen, skytjenester og robotstøvsugere for å sikre den private informasjonen som sendes. Selv om denne informasjonen er kryptert, vil metadata i nettverkspakker fortsatt være tilgjengelig gjennom nettverksavlytningsangrep. I dette prosjektet skal vi undersøke hva slags privat informasjon som potensielt kan bli eksponert av denne dataen. En Irobot Roomba i7 ble installert i to forskjellige smarte miljøer hvor et passivt nettverksavlytningsangrep ble gjort mens ulike robotstøvsuger funksjonaliteter ble utført. Analyse av denne dataen avslørte at det var mulig å attribuere flere ulike smarte funksjonaliteter som ble utført av robotstøvsugeren, bare ved å se på Internett trafikken. Ulike signatur-baserte identifiserings algoritmer ble laget og viste en høy deteksjonsrate. Wi-Fi og Internett trafikken til robotstøvsugeren ble sammenlignet og like trafikkmønstre ble funnet, noe som gjør at deteksjonsmetodene også kan brukes for Wi-Fi trafikk. Denne oppgaven tar for seg implementasjon, konfigurasjon og analyse av nettverkstrafikk og presenterer en deteksjonsalgoritme for Irobot Roomba i7 hendelser.

Preface

This thesis is written as the final subject of *Experience-based Master in Information Security* (MISEB) at the Norwegian University of Science and Technology (NTNU), at Gjøvik.

I would like to thank my supervisor Jia-Chun Lin and co-supervisor Ming-Chang Lee for all the support and guidance in this project, and for including me in the IoT research group where good discussions and knowledge transfer took place. I would also like to thank the persons within the IoT research group: Helene Knudsen, Mathias Hedberg, Kevin Nordnes, Lloyd Nicolay Gustavsson and David Sintayehu Solum, for all the support. Additionally, I would like to thank my partner Helene for the support during these three years at NTNU, helping me balance work, school and personal live, this would not be possible without you, thank you!

Contents

Figures

Tables

Acronyms

AP Access Point. xiii, 17, 18, 28, 29, 32, 56, 67

ARP Address Resolution Protocol. 27, 28, 30, 32

AWS Amazon Web Services. 29

DHCP Dynamic Host Configuration Protocol. 10, 18, 27, 28, 32

DNS Domain Name System. xiii, xiv, 27–32, 34, 35, 37–39, 42, 44, 46, 47, 50, 51, 55, 56, 61–63, 65–67

FQDN Fully Qualified Domain Name. 28, 29, 31, 34, 37, 39, 40, 55, 56

IGMP Internet Group Management Protocol. 67, 68

IoT Internet Of Things. 1–3, 5–7, 9–11, 15, 17, 20, 21, 31, 56, 60, 66, 68

IP Internet Protocol. xiv, 7, 11, 18, 19, 28, 31, 32, 34, 39, 61–63, 67

ISP Internet Service Provider. 2, 16, 18, 19, 28, 56

LAN Local Area Network. xiv, 2, 7, 10, 17–20, 25, 28, 32, 56, 57, 65–67, 69

MAC Media Access Control. 6–8, 19, 28, 32, 57, 65–67

NAT Network Address Translation. 18, 31

NIC Network Interface Card. 7, 17–19, 56

NTP Network Time Protocol. 30, 32

OSI Open Systems Interconnection. 13

RVC Robot Vacuum Cleaner. 1, 13, 14, 29

SD Standard Deviation. 35, 44

SSID Service Set Identifier. 17, 56, 60

TCP Transmission Control Protocol. xiii, 27, 30–32, 34, 39, 47, 50, 61–63

TLS Transport Layer Security. 7, 27, 30, 34, 35, 39

UDP User Datagram Protocol. 27

VPN Virtual Private Network. 7

WAN Wide Area Network. xiii, 2, 11, 18–20, 27, 28, 32, 56, 65–67, 69

Wi-Fi Wireless Fidelity. 6, 8, 10, 11, 13, 15, 18, 19, 56, 65, 67, 68

WLAN Wireless Local Area Network. xiv, 2, 19, 20, 25, 27, 32, 56, 57, 65–69

Chapter 1

Introduction

This chapter introduces the problem domain of the master project topic. Research objectives is presented with the associated research questions which the thesis aims to address. Further, the research delimitation and contribution is presented, before the overall structure of the thesis is introduced.

1.1 Problem Domain

The increase of Internet Of Things (IoT) and deployment of smart environments are rapidly increasing and are expected to increase further [1]. IoT devices are designed to automate and streamline users daily activities and chores. Robot vacuum cleaners, smart lighting, smart garage ports and smart door locks are becoming a part of every smart home environment. The close integration between IoT devices and users lifes, introduces new security and privacy challenges.

Robot vacuum cleaners have become a popular smart environment device. These robots can automate floor cleaning based on users preferences and customization [2]. Integration with other IoT devices allows cleaning to be triggered based on human action, which can potentially expose user behavior and routines.

Other researches have addressed security challenges on robot vacuum cleaners with penetration testing, vulnerability assessments and active network eavesdropping and interception. It has also been conducted research about passive eavesdropping in smart home environments including robot vacuum cleaners, but without detailed analysis of the device. Event attribution and privacy challenges associated with this is therefore not addressed.

1.2 Research Objectives

The goal of this thesis is to identify private information exposed in a smart environment, only based on network traffic generated by a Robot Vacuum Cleaner (RVC). We want to address this from an attackers perspective, and only use passive eavesdropping in the different phases of network communication. To be able

to extract user private information, we analyze network traffic and attempt to identify traffic pattern signatures. These three research questions were created to address this topic and guide the research.

1. Which private information can be gathered from a robot vacuum cleaner by carrying out a passive network eavesdropping attack in a smart environment?
2. How can the information exposed by the eavesdropping attack be misused by an attacker?
3. Which security measures can be implemented to limit the exposed data and decrease the risk of misuse?

1.3 Scope and Delimitation

The scope of this thesis is passive eavesdropping of WLAN and WAN traffic, this excludes actions that will effect the traffic such as traffic shaping, man-in-the-middle-attacks, traffic injection and similar actions. All traffic capturing and analysis are from the perspective of an attacker. Only information that is available in the capturing files are therefore included in the thesis' analysis. This excludes decryption of traffic or knowledge about other local configurations and passwords within the environments or devices.

Irobot Roomba i7 is the only robot vacuum cleaner considered in this thesis. This robot vacuum cleaner is connected to a separate WLAN during the entire data capturing process, allowing only cloud based communication. Local IoT communication and influence is therefore not included. Environment and network infrastructure is delimited to only basic Internet access, this excludes security implementations of for example firewalls, access-lists, identity management and multicast addressing, which could affect the communication.

The complexity of eavesdropping is also not included, due to the large variety of solutions in different smart environments and Internet access. WAN interfaces are delivered by Internet Service Provider (ISP), access to this traffic flow will not be considered and a simulated WAN Interface is created within the LAN of the environments.

Analysis is done using Wireshark and basic python scripting. Signature is therefore only identified by human manual analysis through these tools. This limits the analysis to only look at overall characteristics or initial traffic and not machine learning.

1.4 Contribution

This project contributes with research on Irobot Roomba i7 robot vacuum cleaner, including detailed network traffic analysis and successful identification and attribution of different events. Previous researches have addressed the same security

and privacy challenges with IoT smart environment including robot vacuum cleaners, but focusing on attributing different smart environment events not specifically on a robot vacuum cleaner. Other projects have focused on robot vacuum cleaner, but actively attacked the devices to evaluate the security and privacy issues by exploiting different vulnerabilities detected. This thesis therefore add knowledge about level of event attribution which is possible on an Irobot Roomba i7 and proposes detection and signature for different events.

1.5 Thesis Structure

The rest of this thesis is divided into the chapters *Background, Related work, Method, Analysis and Results, Evaluation, Discussion and Conclusion*. First the Background chapter will present relevant information needed to understand the topic. Related work will cover existing research on this area. The Method will present the different processes of selection, configuration, processing and analysis. Further the analysis and results will be presented. This is followed by an evaluation chapter to evaluate the research results. Lastly a discussion and conclusion chapter will summarize the thesis' challenges, decisions and answer the thesis' research questions.

Chapter 2

Background

This chapter introduce fundamental concepts and background of Internet of Things and smart environments. It will also present robot vacuum cleaners how they operates and which communication protocols they use. Furthermore, it defines and present an overview of traffic eavesdropping and potential defense mechanisms which is important in regards to the research questions.

2.1 Internet of Things

IoT is a system of interconnected physical and virtual devices communicating and sharing information, using the Internet or private networks. Autonomous IoT devices are available for information sharing and event triggering continuously and can act based on inputs, status or triggers from other IoT devices [3]. IoT systems take advantage of the large scale information sharing. Intelligence software enables the devices to become smarter and more advanced based on information shared among IoT devices. The devices includes a number of different hardware components and software versions, while standardized communication protocols and system architecture makes the integration between IoT platforms possible [3].

Small devices like video cameras, smart door locks or air quality monitors have limited local processors and storage. Complexity and the need for data processing have made vendors integrate their systems to centralized cloud infrastructures. Data from the IoT devices is therefore sent to cloud services, where it is processed. Algorithms communicate commando and control traffic back to the devices based on user configuration [4]. However, the use of cloud introduce latency because sensor data needs to be transferred to the cloud server where it is processed and actions are decided. All this extra transmission latency is not applicable in for example a smart car breaking system because it requires fast decision making. Local computing is therefore distributed closer to the sensor providing low latency decision making, this is referred to as fog or edge computing [5].

2.2 Smart Environments

Smart environments are identified by their seamless connectivity between the sensors, edge devices and a centralized control system. Data is continuously collected from the smart environment providing the centralized controller with live data [6]. This data can be analyzed, and trigger actions from the controller to other devices in the environment. In a smart home environment the controller can be notified by a garage port opener or sensor that the car has left and trigger a sequence of events such as locking the door, turning of all the lights or starting the robot vacuum cleaner. In smart industry environments the IoT sensors can communicate that temperature or other air quality measurements are outside off the threshold values and then trigger systems to carry out actions to stabilize this. Due to availability of this data, users can remotely monitor, automate and control the environments based on their needs and requirements. This can give a personalized user experience and value [7]. Several centralized smart home applications are developed to make the user experienced and device integration as easy as possible, Home assistant [8] is an example of this. The application enables integration between IoT systems, based on application programmable interfaces. These interactions aims to include as many IoT systems as possible, introducing security and privacy challenges across different IoT platforms.

2.3 Robot Vacuum Cleaners

Robot vacuum cleaners are popular smart home IoT devices. These robots can clean the smart environment autonomously, and can be configured to clean based on scheduled cleaning tasks or automatic cleaning based on integration with other IoT systems. Newer models have advanced cleaning and navigation technology and are able to map their surroundings, avoid obstacles and suggest cleaning routines based on season or the level of dust in the environment. Popular robot vacuum cleaner vendors are Irobot [9], Neato [10], Ecovacs [11] and Roborock [12].

2.3.1 Robot Vacuum Cleaner Communication Protocols

The newest models from all the vendors Irobot [9], Neato [10], Ecovacs[11], Roborock[12] and Neatsvor [13], use Wireless Fidelity (Wi-Fi) as their main communication protocol. Wi-Fi is used to communicate with the cloud service, and present live data in the associated smart home application.

IEEE 802.11 is a media access control specification used in modern Wi-Fi communication [14]. Wi-Fi is used to connect wireless IoT devices to the wired smart environment network infrastructure communicating with cloud services. Traffic can therefore be eavesdropped both during wireless and wired communication [15]. Wi-Fi transmission uses Media Access Control (MAC)-addresses [16] to determine the packet origin and destination. A MAC address includes 48 bits, where

the first 24 are used as an organization identifier. The last 24 bits are then used as a unique identifier within an organization MAC range. Registers of organizations global MAC identifier are available online, and can be used to identify which devices that are connected to wireless or wired networks [17].

2.4 Traffic Eavesdropping

Traffic eavesdropping is a technique used to collect network traffic, not addressed to the collecting device [18]. Eavesdropping can be separated into two categories, passive and active. In active eavesdropping an attacker will interfere with the traffic flow. This could be packet injection, modification or disruption. Passive eavesdropping will only collect traffic without any interference. To conduct wireless eavesdropping an attacker will only need to be in wireless range of the targeted devices [15] and in wired eavesdropping physical or remote access to network devices in the smart environment is required and increases the complexity.

Wireshark [19], Tshark [20], tcpdump [21] and Microsoft message analyzer [22] are some tools that can be used for network eavesdropping. All these tools can monitor traffic received on a specific Network Interface Card (NIC). During wireless eavesdropping the wireless NIC needs to be in monitoring mode and process all packets received. For wired eavesdropping an attacker can configure a Switch Port Analyzer (SPAN) port on a LAN switch duplicating and forwarding specified traffic to the interface connected to the capturing device. This functionality also have legitimate use cases with implementation of Intrusion Detection Systems in networks as an example.

2.5 Eavesdropping Defense Mechanisms

Traffic shaping is a technique used to shape the network traffic based on policies. This is used to optimize data networks, prioritizing traffic and limiting transmission of irrelevant data, but also proposed as a defense mechanism in IoT smart environments [23]. Authors in [24] proposes a method to shape smart environment traffic to defend against traffic flow analysis attacks, mitigating the risk and increase the complexity of such attacks.

Encryption is another popular defense mechanism against network eavesdropping. Traffic can be encrypted in different network layers simultaneously creating multi layer encryption. Applications create end-to-end encryption with for example Transport Layer Security (TLS) [25] or Secure Shell (SSH) [26]. This can also be done on the network layer by using different types of Virtual Private Network (VPN)s. Two categories are tunneled or transport mode VPNs where tunnel mode encrypts the original IP-header and add a new, hiding the original source and detonation address [27] [28]. Both Internet Protocol Security (IPSec) [27] and Layer Two Tunneling Protocol [28] are examples of popular VPN protocols.

As for Wi-Fi communication it has become industry standard to include encryption between the wireless devices and Access Points (AP), authentication can be done with pre-shared keys, certificates or integration with identity access management solution for example Windows active directory or Kerberos.

The use of global administrated MAC addresses introduce privacy and security issues as a MAC address can be tracked or identified easily, especially in wireless eavesdropping. Local MAC randomization [29] can be used to hide the original global MAC address associated with an organization and use a new local MAC address within a wireless network hiding device information for potential misuse.

Chapter 3

Related Work

This chapter introduce existing research and literature relevant for this thesis' topic. Presented work are focused around security and privacy issues, related to the topics: IoT, smart environment and robot vacuum cleaners. Further, related work on different eavesdropping attacks and possible countermeasures, as well as IoT event detection are described.

3.1 Smart Home Security and Privacy

The use of IoT devices in smart environments have increased the security issues within these environments. According to Alferidah and Jhanjhi [30], and Swessi and Idoudi [31] these issues are presented in all layers of the IoT systems hardware, software and communication. The nature of information sharing also introduces privacy issues in smart environments [30]. Alferidah and Jhanjhi [30] have created an overview of the most critical vulnerabilities and possible counter measures in an IoT environment.

IoT smart integration enables controllers to trigger actions based on sensor data, without user interaction. Gu et al. [32] did a research on wireless Zigbee traffic mining in a smart office environment. They were able to identify and attribute 35 different events only by passively eavesdropping the wireless traffic. With further analysis they were able to expose private information about the office routines based on this traffic.

3.2 Security and Privacy Challenges of Robot Vacuum Cleaners

The popularity of robot vacuum cleaners raises the concern for information security and privacy issues. Sundström and Nilsson [33] looked at the security implementation and vulnerabilities on a Roborock S7. They discovered that the robot vacuum cleaner was reasonably secure. Due to ethical concerns, the cloud service security was not in scope. During the setup stage they discovered that all

devices within wireless coverage of the vacuum cleaner, could add initial configuration, regardless of application support. According to Sundström and Nilsson [33] the Roborock S7 was vulnerable against Dynamic Host Configuration Protocol (DHCP) starvation attack from rouge devices on the same network. The authors suggested that networks used to control a Roborock should at least have basic authentication requirements, to avoid rouge devices. A similar research is done by Ullrich et al. [10] where the robot vacuum cleaner was produced by Neato. In this research the authors evaluated communication and security towards the cloud service and application. They discovered that week cryptography and shared private keys among the devices resulted in a huge privacy risk. The collected data revealed personal information about the customers routines, apartment size, pets and number of residents.

Sami et al. [34] did a research on private information eavesdropping, based on laser sensor data of a robot vacuum cleaner. This sensor data was extracted through a side-channel on the targeted robot vacuum cleaner. Through the research they were able to sense vibrations in objects like pager bags and detect words said by humans in the environment. By sensing vibrations on objects from television or music speakers they were able to identify songs and tv shows, with high precision. They suggested that manufactures have to make security implementations, limiting high precision private data to be extracted from the devices.

Nguyen [35], Kaminski et al. [36] and Torgilsman and Bröndum [37] all address security and privacy concerns with the deployment off different robot vacuum cleaners. They use the STRIDE threat analysis framework to identify and categorize the different vulnerabilities. They executed attacks towards the robot vacuum cleaners to expose information. In addition several security and privacy issues related to setup, LAN and cloud communication for these vacuum cleaners was discovered. All of them proposed security improvements that should be implemented by the vendors.

3.3 Eavesdropping and Event Detection

Alyami et al. [38] establish a method to capture out-of-network encrypted Wi-Fi traffic, and attribute different IoT devices within a smart environment. The research had a 95 percent accuracy of identifying these devices, and in some cases also their working state. Acar et al. [39] also conducted a similar research on smart environments, using machine learning to identify devices and their actions. These devices used Wi-Fi, Zigbee and Bluetooth. They also suggested countermeasurements that can be implemented to defend against passive eavesdropping attribution. Xiong et al. [40] proposes a network traffic flow mechanism to limit the possibilities to attribute IoT devices and events with eavesdropping. They inject dummy traffic, and delay random traffic packets to mix the network traffic sequence. This defence mechanism creates more delay and latency within the environment, and disrupted some devices and functionalities.

Trimananda et al. [41] have created a tool to learn and create detection rules

for IoT devices based on Wi-Fi and WAN traffic. The traffic in the research is encrypted, and they have only used packet lengths and IP address as attributes. Events are triggered through the tool, and corresponding traffic capturing is initiated. Timestamp and event capturing files are then used to train a machine learning algorithm to create event signatures. They were able to identify user behaviour within the smart home using this tool.

Chapter 4

Method

This chapter covers the method used in this research, including choices made in the smart environment architecture, capturing, filtering and analysis processes. Further the logical structure of the detection algorithms are presented. The entire method is broken down into several stages to better structure the research, as illustrated in Figure 4.1.

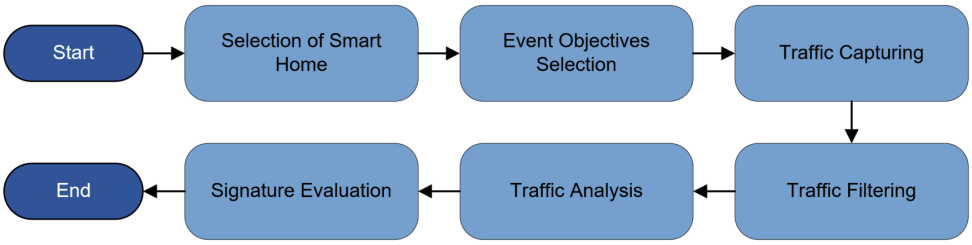


Figure 4.1: Overview of the stages within the thesis methodology

4.1 Selection of Smart Environment

This research used only one robot vacuum cleaner which was selected based on a set of requirements, a survey was therefore conducted at the start of the research. A selection process for the associated smart home environments was also conducted trying to simulate a general smart home.

Requirements within three different categories were used to select the RVC: *Communication protocols, smart home features* and *popularity* were used in the selection process. These requirements are described below:

- Communication protocol: Wi-Fi is the most wide spread communication protocol in today's smart environments [42]. Eavesdropping devices and analysis tools are available for IEEE.802.11 and abstraction layers higher in the Open Systems Interconnection (OSI) model [43]. Therefore the first requirement for the RVC will be that it communicates over Wi-Fi.

- Smart Home Features: The number features available, could increase attribution and potentially more information could be exposed. The robot vacuum cleaner needs to have several smart home features to test [44].
- Popularity: The prevalence of different vendors and models will always vary, based on the number of sold and used devices. Overall usage and good reviews will increase the relevance of this thesis. Therefore popularity is included in the evaluation of RVC.

Data and ratings from three robot vacuum cleaner review sites [45], [46] and [47] were used in the first phase of the selection. A summary of all these reviews was used to determine the most reliable robot vacuum cleaner vendors. To determine the popularity of the different vendors, downloadings and ratings from Google Play were compared [48].

Results from the review sites are presented in Table 4.1, and shows that the two best rated robot vacuum cleaner vendors are Irobot and Roborock.

Table 4.1: Robot vacuum selection review-site comparison

(a) Results from review-site [45]

Vendor	Number on top ten
Irobot	3
Roborock	3
Neatsvor	0
Ecovacs	0
iLife	2

(b) Results from review-site [46]

Vendor	Number on top ten
Irobot	2
Roborock	2
Neatsvor	0
Ecovacs	2
iLife	1

(c) Results from review-site [47]

Vendor	Number on top ten
Irobot	2
Roborock	2
Neatsvor	3
Ecovacs	1
iLife	0

(d) Summary of all review-sites

Vendor	Number on top ten
Irobot	7
Roborock	7
Neatsvor	3
Ecovacs	3
iLife	3

Applications for different vendors' robot vacuum cleaners are presented in Table 4.2. It is worth mentioning that the "Smart Life" application is used to control the Neatsvor vacuum cleaner, but is primarily a smart home integration application.

Table 4.2: Robot vacuum cleaner applications download and rating statistics

Vendor	Application	Downloads	Rating
Irobot	Irobot Home	5 million +	4,0/5,0
Roborock	Roborock	1 million +	4,6/5,0
Neatsvor	Smart Life	10 million +	4,5/5,0
Ecovacs	Ecovacs Home	1 million +	2,5/5,0
iLife	iLifehome	50 thousand +	NA

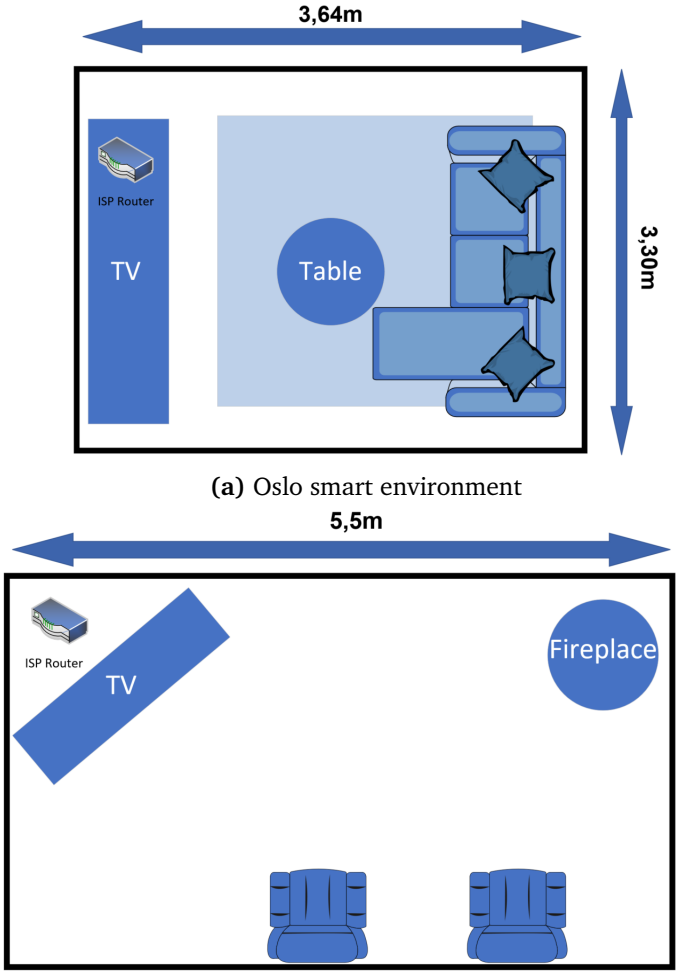
Both Irobot and Roborock received seven recommendations on the review websites when the results from all sites were added. This is significantly higher than the other three vendors on the list, each of which only had three representations. Additionally, both vendors were referenced in all three review sites which strengthens their credibility.

In the application download and rating analysis, the Neatsvor application has over 10 million downloads. However, this application is more focused on smart home integration, and the high number of downloads is likely not solely due to the robot vacuum cleaner. Meanwhile, Ecovacs home received a 2.5/5.0 rating, and despite having a similar number of downloads as Roborock, it fell short in the selection process. Hence, Irobot and Roborock emerged as the two most relevant vendors. In a comparison of their products, it was found that the Irobot Roomba i7 and Roborock S6 were the most suitable models, they include a wide range of smart home features, uses Wi-Fi as their main communication protocol and are the most popular [49] [50].

The final comparison was conducted using *bestcordlessvacuumsite.com* [51]. Irobot Roomba i7 and Roborock S6 have similar reviews and rating all over, and they both have a sufficient number of smart home features, such as IoT smart home integration, application, different cleaning types and detailed environment discovery. The fact that the Irobot application is downloaded five times more than the Roborock was the decisive factor. Irobot Roomba i7 is the selected robot vacuum cleaner for this master project and is presented in Figure 4.2.

Figure 4.2: Irobot Roomba i7 [52]

To ensure the validity of the results across diverse settings, the testing was conducted in two different smart environments, now called **Oslo** and **Drammen**. The robot vacuum cleaner was configured from factory defaults for each of the environment. Both **Oslo** and **Drammen** had independent Internet access, provided by an external ISP. To control the duration of each test, the available test area was restricted to one room. This decreased the duration of each test, making the research more efficient. Illustrations of both **Oslo** and **Drammen** smart environments are presented in Figure 4.3.



(a) Oslo smart environment

(b) Drammen smart environment

Figure 4.3: Oslo and Drammen smart environments

The rest of the research environment had to support traffic eavesdropping and analysis of the Irobot Roomba i7. This sets some requirements for the selection of smart environment. It has to include an AP providing a separate Service Set Identifier (SSID) only to be used by the Irobot Roomba i7. This way the identification of relevant and irrelevant traffic is possible without interference by other connected IoT devices. In addition to this, a wired network infrastructure needs to be available providing LAN access to the traffic generated from the Irobot Roomba, this is where the LAN eavesdropping is conducted. Lastly the environment would need Internet connectivity to be able to connect to Irobot cloud services and be controlled by a smart home application.

A Raspberry PI 3b+ was chosen as the capturing platform for this research. These computers are designed to run autonomously and has build in Ethernet and WiFi NICs. The Raspberry PI was installed with a Kali Linux operating system

from [53]. Several network traffic analysis and capturing tools are included in the Kali Linux distribution. During wireless capturing, the wireless NIC had to be configured in *monitor mode*. A separate Wi-Fi adapter *TP-LINK TL-WN722N V2/V3* was acquired and used as the monitoring wireless NIC.

Both **Oslo** and **Drammen** only had a single ISP modem, providing both WLAN and LAN. To enable WAN interface simulation and eavesdropping, an additional Access Point (AP) and LAN switch were installed. The AP translated all Wi-Fi traffic generated form the Irobot Roomba with Network Address Translation (NAT) to a single IP address in the smart environment LAN, simulating a WAN interface. The AP then forwarded traffic to the ISP router through the switch. All traffic forwarded on the interface connected to the AP were duplicated and forwarded to the Raspberry PI, through a configured SPAN port. The network infrastructure is shown in Figure 4.4, where the monitored and SPAN configured interfaces show how the eavesdropping is carried out.

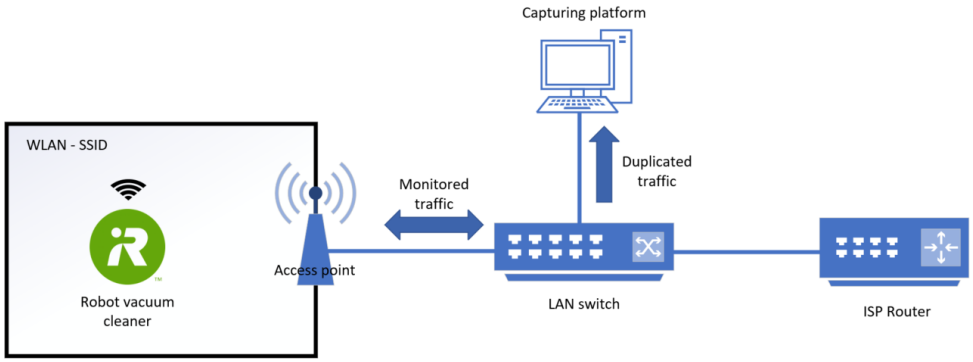


Figure 4.4: WAN simulating and eavesdropping in the smart environment infrastructure

Wireshark [19] is a widely used network protocol analyser which allows network capturing and analysis in real-time. It can be used for deep header inspection in all network layers and perform basic identification of a wide range of different protocols [19]. This software is integrated on Kali Linux and is available for Windows at [54]. Tshark is a subprocess of Wireshark, and can be used to capture traffic through command-line . In this process it is possible to define capturing filters which only store traffic that is interesting for the analysis phase. Tshark was therefore used to capture traffic, and the manual analysis was done with Wireshark.

The network infrastructure in both environments consists of the devices presented in Table 4.3 and are connected in a smart environment infrastructure shown in Figure 4.5. Telia was the ISP in both **Oslo** and **Drammen** and the same Sagemcom router was used, this did not affect the internal Irobot Roomba communication since the only function of Sagemcom is DHCP and Internet connectivity.

Table 4.3: Smart environment device inventory

Device	Type
Capturing platform	Raspberry PI 3b+, with Kali Linux
Analysis platform	HP Elitebook, with Windows 11
Access point	TP-Link archer MR200, ver5.30
LAN switch	Cisco catalyst 2960 series, 8 port
ISP router	Sagemcom, Telia

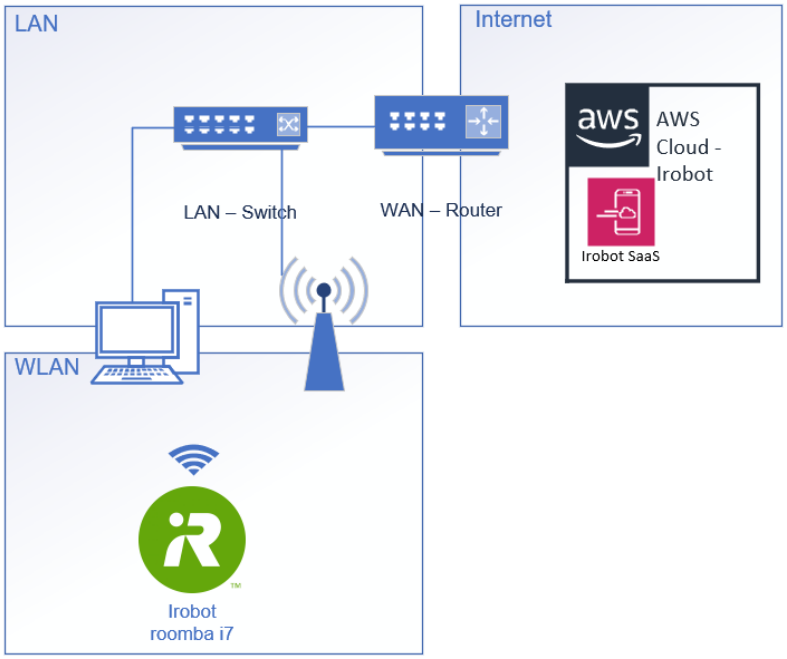


Figure 4.5: Smart home setup [9]

4.2 Traffic Capturing

Two Tshark processes were executed simultaneously on the Raspberry PI, one instance captured WAN traffic on the Ethernet NIC (eth0) and the other captured WLAN traffic on the Wi-Fi adapter NIC (wlan1). Both instances had a capturing filter argument, capturing only traffic relevant for the analysis phase. The syntax for the Tshark filter is described in [20]. The Tshark arguments used in these captures were interface specification *-i*, traffic filter *-f* and output file name *-w*. Filtering was based on the Irobot Roomba's Wi-Fi MAC address for WLAN, and the simulated WAN IP-address for LAN. Due to local user restrictions, the Tshark commands were executed in sudo mode. Tshark filter syntax as well as WLAN and LAN specific syntax are listed below.

- tshark [-i <capture interface>] [-f <capture filter>] [-w <outfile>]

- sudo tshark -i wlan1 -f 'eth.host MAC address' -w output.pcap
- sudo tshark -i eth0 -f 'ip.host WAN address' -w output.pcap

A fixed capturing process was used the entire research, this created the best foundation for event comparison. The entire capturing process is illustrated in a flow diagram in Figure 4.6. First both WLAN and LAN Tshark capturing were started. While these captures were ongoing, events were triggered according to a test matrix. When all events had been triggered, the capturing was stopped and files were extracted with WinSCP to a Windows machine for further analysis [55].

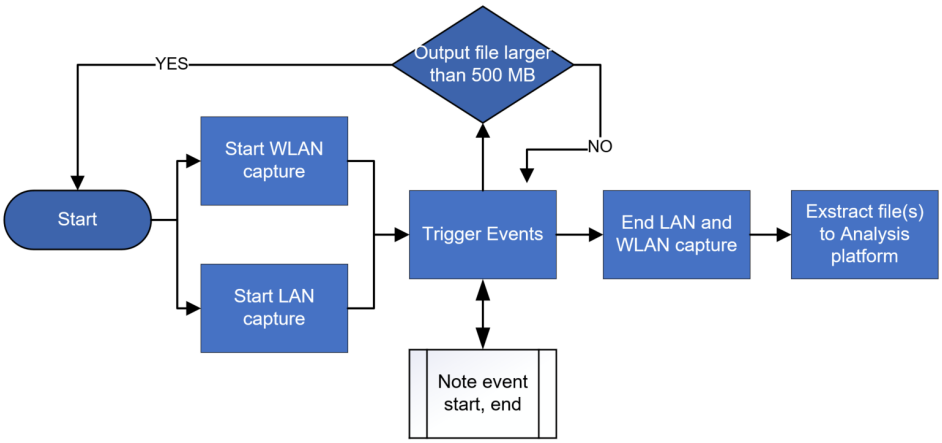


Figure 4.6: Event capturing process

4.3 Event Objectives

This section introduces the selection and triggering of different smart home features on the Irobot Roomba i7. For all selected event objectives the justification, functionality and triggering process are described. All event objectives, except *Standby traffic*, are triggered 10 times in each of the smart environments **Oslo** and **Drammen**.

Standby traffic capture is an event selected and is providing information about the continuous traffic flow generated by the Irobot Roomba when no event is triggered. This event is therefore used to identify network traffic which is not relevant in the event detection process and can be excluded for event analysis. Observed traffic is also used to identify the ongoing network session between the Irobot Roomba and the cloud server and exclude traffic generated from other IoT devices within the same environment. If an attacker can identify the present of an Irobot Roomba it can launch targeted attack such as spare phishing and increase the success rate.

The capturing started after the Irobot Roomba had been operational in **Oslo** smart environment for one month, ensuring that the captured traffic is generated when the vacuum cleaner is in operating state and not set-up phase. A continuous

capture was conducted over 14 days. During this time no physical or application interaction was done by the user. The capture was then extracted to the analysis platform.

All other event objectives aims to expose different user behaviours and each of them are described further in detail. *Automated cleaning*, is a cleaning event triggered by integration with third party IoT systems. Through the Irobot application it is possible to configure integration with IFTTT location services [56], triggering a cleaning when the user's phone is observed outside a configured radius of the smart environment. It is also possible to integrate with Gust smart lock system, Ecobee thermostat system, My Leviton smart home integration and MyQ garage system [9]. IFTTT was selected as the preferred trigger integration due to the availability. If this event is attributed an attacker will know when the user has left the smart environment and can potentially conduct robbery without the risk of the user being home or map the users routines such as working hours.

The location service is configured to trigger a cleaning event including the entire smart home map when the user's phone is more then 200 meters away from the address of the smart environment. Event start is defined at the time when a notification was received stating that cleaning is triggered, and finished when a "finished cleaning" notification is received. *Automated cleaning* events are only triggered once per day, but due to time constrains the configured cleaning job was deleted and reconfigured for each event, allowing the event to be triggered several times per day.

Application triggered cleaning, is triggered through the Irobot smart phone application. Users can configure customized cleaning events only including parts of the smart environment.The event is triggered by opening the Irobot application and manually triggering a customized cleaning event defining all of the smart environment area. It is started when the application is opened and finished when a "finished cleaning" notification is received. Attribution of this event will expose information about user phone activity and if detected during a longer time period expose user routines.

Scheduled cleaning can be configured through the smart phone application. Users can schedule a cleaning by specifying an area in the smart environment and time when the cleaning should start. This event is integrated into user's routines as it most likely is configured when the user is away from the smart environment.

The event is triggered by configuring scheduled cleaning jobs including the entire smart map through the application. It is not possible to schedule cleanings with less than three hours in between, due to time constrains only one predefined cleaning job was used and the configured time was changed, allowing to trigger the event more frequently.

Physical triggered cleaning can be triggered by pushing a physical button on the Irobot Roomba marked with "Clean". This triggers a cleaning job of the entire environment area that the robot vacuum cleaner can navigate in. This can potentially trigger a map discovery if the surroundings are not recognized. Identification of this event will expose information about user present inside the smart

environment. The cleaning event is finished when a notification is received.

Application start is an event to identify if the Irobot application is opened on a user's phone. Whenever the application is started it pulls live information from the Irobot Roomba and displays it in the application. Exposure of this information will reveal user interaction with the Irobot application. Triggering starts when the application is started and finished when the application is closed. No specific action is conducted in the application during this event.

Remove bin is when the bin on the Irobot Roomba is ejected from the vacuum cleaner. This is typically done after a cleaning event or when a notification is sent to the user. Identification of this event will place the user within the smart environment together with the Irobot Roomba exposing user position.

The event is triggered when the user ejects the bin from the Irobot Roomba and they are separated for at least 40 seconds, simulating the time it would take to empty the bin. It is finished when the bin is reentered into the Irobot Roomba.

4.4 Traffic Filtering

This section introduce the method used to identify irrelevant traffic in the *Standby traffic* and creating filters that excluded this traffic in further event analysis. It also include the process of creating separate event files from the continuous environment captures. The overall process steps are shown in Figure 4.7, where there is an iterative process between identification and filter creation.

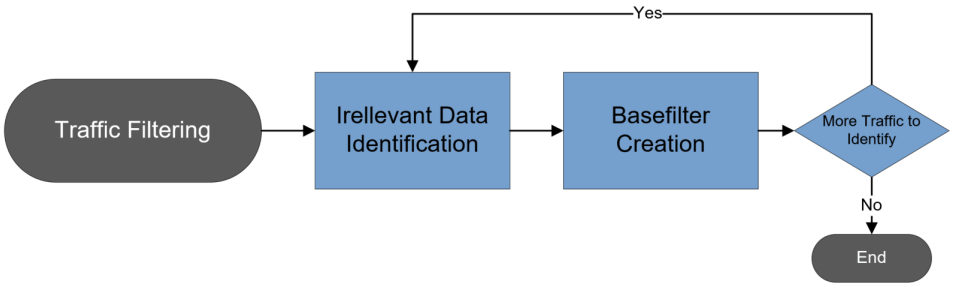


Figure 4.7: Traffic filtering process

The *Standby traffic* capture is used to identify traffic patterns and protocols that are irrelevant to the event triggering conducted in this project. Since there was no interaction with the Irobot Roomba during the capturing period, all the identified traffic patterns and protocols are irrelevant for the actual event objectives.

The capturing files are imported and opened in Wireshark and analyzed with the use of **Protocol hierarchy tool**. This tools presents the different protocols and their distribution within the capturing file. An example of this analysis is shown in Figure 4.8. Traffic from each of the identified protocols are analysed and irrelevant

traffic observed is excluded by adding logical expressions to the baseline filter in Wireshark. Resulting in a complete basefilter excluding all irrelevant traffic identified.

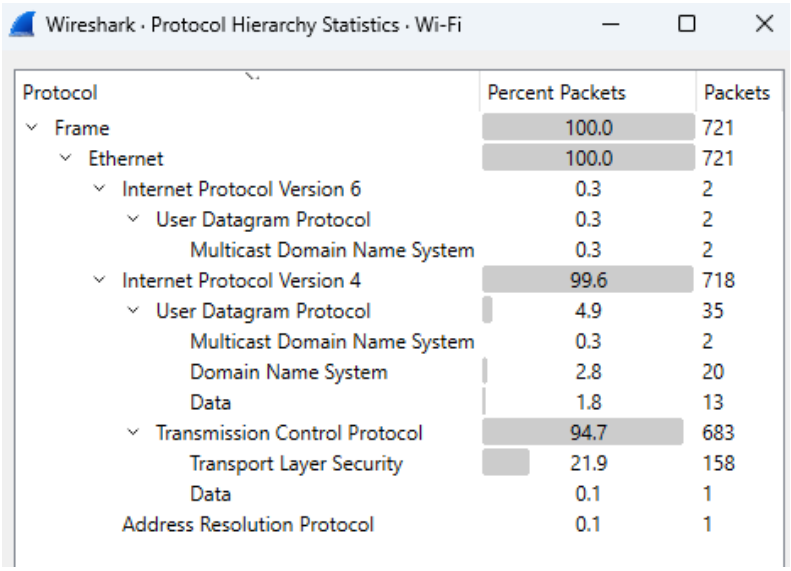


Figure 4.8: Wireshark protocol hierarchy tool

The basefilter is applied to the event capturing files and in addition a Wireshark time filter is added to only display traffic generated during an ongoing event, creating one capture file per event. The Wireshark filter syntax is found in [19] and the time filter syntax is presented below.

- **(frame.len >= "Year Month day, start-time") && (frame.len <= "Year Month day, end-time")**

4.5 Traffic Analysis

The traffic analysis workflow is presented in Figure 4.9, and includes three subprocesses: *Protocol and event relation, Traffic sequence identification* and *Overall event characteristics*. Results from all these subprocesses were used to create event signatures. These signatures are implemented into python detection algorithms to evaluate the success rate of the signature detection.

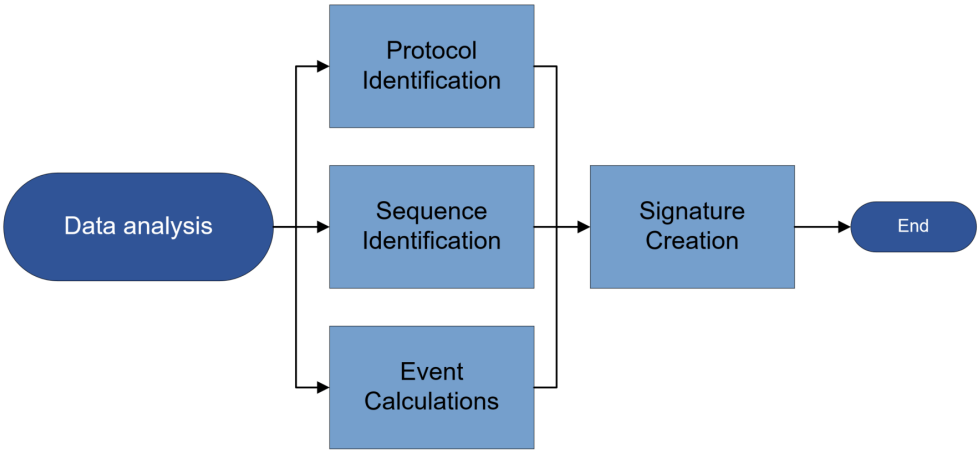


Figure 4.9: Traffic analysis process

Protocol and event relation are analyzed with **Protocol hierarchy tools** and the identified protocol and their attributes are collected. If specific protocols occur within the majority of the event files it could be used as a identification signature.

Traffic sequence analysis used same attributes as Trimananda et al. [41], without the proposed machine learning algorithm. Packet length sequences were extracted with the use of a python script using a Pyshark library, and analyzed manually. This analysis also includes the sequence of protocols, enabling signatures based on more than just packet lengths as attributes. Traffic flow directions were also taken in to account during this process. This included the traffic flows listed below:

- Traffic flow with Irobot Roomba as source address.
- Traffic flow with Irobot Roomba as destination address.
- Traffic flow both directions.

Overall characteristics of each event file were analyzed, and used to determine if the number of 20 events were sufficient. Extracted information about *number of packets, number of bytes* and *protocols* were used in this process. Standard deviation of this data indicated the consistency of each event objective.

Results of the three subprocesses were used to propose an event signature. Attributes in the different signatures were implemented as search conditions in a python function. A separate function was created for each event signature, returning a confidence variable to the main function. This enables the detection algorithm to add new signatures to increase the detection confidence of an event. All these functions followed the same logical flow as presented in Figure 4.10

Figure 4.10: Pseudo code for signature detection function

```
    def Identify_event(event_file, event_confidence_variable):
        event_signature = ['signature']
        if event_signature is in eventfile:
            event_confident_variabel =+ level_of_confidece
        else
            None
        return event_confidence_variabel
```

4.6 Signature Evaluation

All signature functions were integrated in a main python script. This script imports selected pcap files and run through all the signature functions. True positive and False positive detection increases the confident variable for each event. A comparison of the confidence variables is executed at the end, to determine which event is most likely to have been triggered. The main function follows the logic presented in Figure 4.11. If a signature is identified in the majority of other events it can not be used to unlikely identify an event and will be rejected by this project.

Figure 4.11: Pseudo code for event detection algorithm

```
        Main()
             #import event pcap file
             Capture = import(Event_file_x)
             #run detection functions, and create confidence variables
             #eventX_confident_variable = eX_cv
             e1_cv = identify_event1(Capture, event1_confidence_variable)
             e2_cv = identify_event2(Capture, event1_confidence_variable)
             e3_cv = identify_event3(Capture, event1_confidence_variable)
             e4_cv = identify_event4(Capture, event1_confidence_variable)
             e5_cv = identify_event5(Capture, event1_confidence_variable)
             e6_cv = identify_event6(Capture, event1_confidence_variable)
             #compare event_confidence_variables highest is event
             confidence_variables = [list of all variables]
             for event in range(1,6)
                  if confidence_variabels[event] is larger than last
                      number
                      largest = confidence_variabels[event]
```

Capture analysis is conducted on the LAN traffic due to time constrains within the project. To determine if the same method and type of signatures can be used to identify events in WLAN, a comparison is done between corresponding LAN and WLAN events. If the level of encryption is different for LAN and WLAN it is not possible to identify the same protocol distribution, and other signatures needs to

be utilized. Packet length sequences is visible in all levels of network communication and differences are determined of included header lengths. A comparison is conducted to determine if the same sequences are present and if observed, the same method and type of filters can be used.

Chapter 5

Analysis and Results

This chapter presents the analysis and results of standby traffic and event captures. First the standby traffic is analysed to identify traffic irrelevant to the event objectives triggered. This is followed by a section for each event including analysis and results of overall event characteristics, protocol detection and packet sequences resulting in a signature detection algorithm. WAN traffic is compared with the corresponding WLAN traffic to identify if these signatures are applicable in both transmissions domains.

5.1 Standby Traffic

This section presents the analysis of the captured *Standby traffic* to identify traffic patterns or protocols to exclude in further event analysis. First an analysis of the relevance of the different identified protocols is conducted. All occurring traffic patterns within the standby event can be found in the event captures as well, but are irrelevant for the actual event triggering.

The standby event capturing was conducted from 8th of January to 22th of January 2023 in **Oslo** environment. During this time period there was no physical or application interacting with the Irobot Roomba, all traffic captured was therefore generated by the Irobot Roomba or the connected Irobot cloud service. This traffic is not created by any human interaction and not exposing any private information. Beforehand, the Irobot Roomba had been installed and operated for one month, ensuring that it was in an operating and not installation state. Smart home map, room dividers and customized cleaning jobs were configured.

Wireshark protocol hierarchy analysis tool was used to display protocol statistics and Table 5.1 presents the protocols and distribution of them. Approximately 50% of the captured traffic was identified as User Datagram Protocol (UDP) traffic, this is mainly DNS but also some NTP and DHCP traffic. Another 26.2% was TCP where the majority of the captured traffic is TLS which is used by the Irobot Roomba to ensure end-to-end encryption with the cloud server. The last protocol identified was Address Resolution Protocol (ARP) representing 24,6% of the standby traffic.

Table 5.1: Protocol hierarchy and statistics in standby traffic capture

Transport protocol	Percentage	Service protocol	Percentage
UDP	49,2	DHCP	0,1
		DNS	48,6
		NTP	0,4
TCP	26,2	NA	NA
ARP	24,6	NA	NA

The network architecture in **Oslo** and **Drammen** creates a single broadcast domain between the AP, capturing platform and the ISP router. As devices only use physical layer MAC addresses to communicate on the LAN they need ARP to create an IP to MAC forwarding table. ARP traffic is therefore broadcasted between these devices requesting updates on IP to MAC information. The captured ARP traffic is not generated by the vacuum cleaner in another broadcast domain and is added as a part of the basefilter with the logical expression *!arp*.

The ISP modem is by default configured as the local DHCP server allocating, reserving and leasing IP addresses to devices connected to the LAN. To ease the detection of the simulated WAN traffic a DHCP reservation was configured on the ISP routers for the AP's LAN MAC address in both **Oslo** and **Drammen**. This kept the simulated WAN address from changing during capturing and potentially lose traffic. DHCP traffic was still needed to be exchanged between the AP and the ISP modems requesting and verifying that the reserved DHCP leases still were active during capturing. DHCP traffic in the LAN capture is therefore only generated by the AP and the ISP modem and was added to the basefilter with the logical expression *!dhcp*.

The most dominant protocol in the standby capture was DNS with 49.2% of the packets. 98.3% of these DNS packets were requests and responses for the DNS A record for Fully Qualified Domain Name (FQDN) *a.root-servers.net*, this traffic flow is shown through Wireshark in Figure 5.1. This is one of the DNS top level domain servers in the DNS hierarchy and will only point to top level domain server such as .com, .org and .no. By analyzing the management console of the AP, it was observed that the AP requested the DNS record for *a.root-servers.net* to determine if it had Internet connectivity or not. A successful response will indicate Internet connectivity. These DNS packets are therefore irrelevant in the analysis of Irobot Roomba traffic and are specifically excluded in a Wireshark filter for further DNS analysis with the following filter *((dns) && !(frame.len ==78 or frame.len ==94))*. Frame lengths of 78 and 94 bytes are used to identify the DNS request and response in Figure 5.1

```
01-12 04:03:25,… 84.208.20.110 192.168.0.56  DNS   94 Standard query response 0x478e A a.root-servers.net A 198.41.0.4
01-12 04:03:35,… 192.168.0.56  84.208.20.110 DNS   78 Standard query 0x8d0b A a.root-servers.net
01-12 04:03:35,… 84.208.20.110 192.168.0.56  DNS   94 Standard query response 0x8d0b A a.root-servers.net A 198.41.0.4
01-12 04:03:50,… 192.168.0.56  84.208.20.110 DNS   78 Standard query 0x8d0c A a.root-servers.net
01-12 04:03:50,… 84.208.20.110 192.168.0.56  DNS   94 Standard query response 0x8d0c A a.root-servers.net A 198.41.0.4
01-12 04:03:55,… 192.168.0.56  84.208.20.110 DNS   78 Standard query 0x478f A a.root-servers.net
```

Figure 5.1: Reoccurring DNS traffic associated with a.root-server.net

When the filter was applied only 174 DNS packets were left. In the remaining DNS traffic it was observed several DNS requests and responses generated by the AP towards the TP-Link cloud services, these FQDNs are listed below and shown in Figure 5.2.

- n-devs-gw.tplinkcloud.com
- n-deventry-gw.tplinkcloud.com

```
192.168.0.56   84.208.20.110 DNS   89 Standard query 0xfad3 A n-deventry-gw.tplinkcloud.com
84.208.20.110  192.168.0.56  DNS  151 Standard query response 0xfad3 A n-deventry-gw.tplinkcloud.com CNAME n-euw1-deventry.tplinkcloud.com
192.168.0.56   84.208.20.110 DNS   85 Standard query 0xfad4 A n-devs-gw.tplinkcloud.com
84.208.20.110  192.168.0.56  DNS  242 Standard query response 0xfad4 A n-devs-gw.tplinkcloud.com CNAME n-euw1-devs-gw.tplinkcloud.com A 52
```

Figure 5.2: DNS traffic generated by the AP towards TP-Link cloud services

These DNS packets are excluded because they are related to the AP and not the RVC. After exclusion only four different FQDN requests generated by the Irobot Roomba were displayed. Three of the FQDN are towards the *.irobot* domain and the last one is a part of *.amazoneaws*. Amazone Amazon Web Services (AWS) is a large provider of cloud services and bought Irobot cooperation in 2022 [57] and is therefore using Amazone AWS for their cloud services. These FQDNs are listed below.

- 0.irobot.pool.ntp.org
- disc-prod.iot.irobotapi.com
- unauth1.prod.iot.irobotapi.com
- a2uowfjvhio0fa.iot.us-east-1.amazonaws.com

All DNS traffic generated by the Irobot Roomba are occurring regularly throughout the entire standby traffic time period, this is presented in Figure 5.3. Irobot's public NTP server *0.irobot.pool.ntp.org* is requested each 12th hour, in the project's standby traffic this is at 03:36 and 15:36. This is used to synchronize the local clock on the Irobot Roomba to a global time zone, allowing all actions or smart features using time to operate correct. *disc-prod.iot.irobotapi.com*, *unauth1.prod.iot.irobotapi.com* and *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com* are all requested once a day at the same time. The function of these requests are not identified, but when trying to access the FQDNs through Google Chrome it prompts "*Missing Authentication Token*", based on the naming convention of the FQDNs it is easy to believe that they are used for re-authentication of the Irobot Roomba.

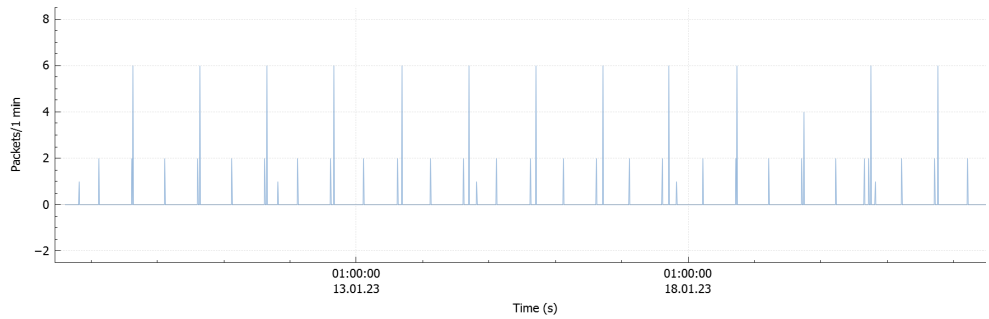


Figure 5.3: Irobot's reoccurring DNS traffic presented in graphical view

The DNS observations identifies that the Irobot Roomba is depending on DNS to be able to access Irobot's cloud services. This protocol is a potential indication of cloud functionality, used or access by the Irobot Roomba and is included in further analysis of events. DNS request for *a.root-servers.net* had to be excluded due to the large amount. All Irobot generated DNS requests had a packet size larger then 78 bytes as shown in Figure 5.1 and all responses were larger then the response for *a.root-server.net*. As the DNS response is the most interesting part of the communication it is safe to exclude all DNS packets smaller then 94 bytes without missing any DNS responses towards the Irobot domains.

Network Time Protocol (NTP) server-client sessions are identified every 30 minutes generating a large volume of NTP traffic. When observing DNS requests to Irobot's public NTP service *0.irobot.pool.ntp.org* and NTP traffic in Wireshark it is possible to identify that the Irobot Roomba is changing the corresponding NTP server every time the DNS response is received, this is shown in Figure 5.4. Since the NTP traffic is continuous it is not related to events triggered, and can be excluded from further analysis by adding the logical expression *!ntp* to the basefilter.

```
192.168.0.56     79.160.225.150   NTP    90 NTP Version 4, client
79.160.225.150   192.168.0.56     NTP    90 NTP Version 4, server
84.208.20.110    192.168.0.56     DNS   145 Standard query response 0xf368 A 0.irobot.pool.ntp.org
192.168.0.56     80.203.110.169   NTP    90 NTP Version 4, client
80.203.110.169   192.168.0.56     NTP    90 NTP Version 4, server
```

Figure 5.4: Irobot NTP client-server traffic

TCP traffic was analyzed in the same manner as the network service protocols ARP, DHCP, DNS and NTP. Remaining TCP traffic is then displayed in Wireshark and is shown in Figure 5.5. Wireshark's protocol hierarchy statistic tools identified that 64.4% of the remaining TCP packets uses TLS to secure the connection, this aligns with the observations in Figure 5.5 where the reoccurring TCP pattern consists of two TLS packets including payload information and one empty TCP acknowledge packet confirming that the last packet was received. Continuous TCP traffic is generated to keep the TCP connection between the Irobot Roomba and the cloud service open as TCP has a timeout value on all session. The majority of

smart environments are installed behind NAT and the TCP session must therefore be initiated from the distributed IoT devices. Keep-alive traffic includes the least amount of data needed to keep the connection alive or continuously updated. All event specific packets are therefore larger than 97 bytes. This filter is combined with the DNS packet length filter and is applied with the logical expression *(frame.len > 97)* excluding all packets less than 97 bytes.

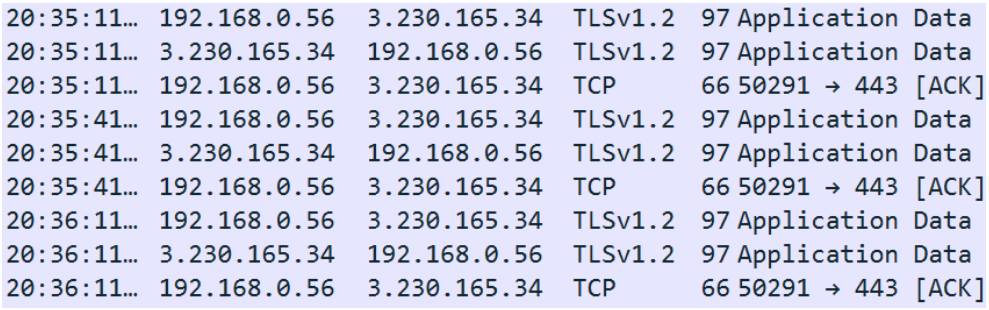


Figure 5.5: Roccuring TCP traffic from the standby capturing displayed in Wireshark

Commando and control traffic from the Irobot's cloud services is generated from the same corresponding host as the TCP-keep-alive traffic. To identify the FQDN used in this session establishment, a combined filter with DNS and TCP is applied, as shown Figure 5.6. As presented the Irobot Roomba terminated the TCP-keep-alive session right before a DNS request is sent for *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com*, and a new session is established with one of the IP addresses in the DNS response. After the establishment of the new session there is uploaded data to the new host, probably synchronising and updating current status on the Irobot Roomba.

An attacker will have to eavesdrop traffic for less than 24 hours to be able to identify which TCP session belongs to the Irobot Roomba. In a large scale eavesdropping attack attackers can trigger actions based on identification of this DNS response, knowing that a Irobot vacuum cleaner is establishing a new session with one of the responded IP addresses.

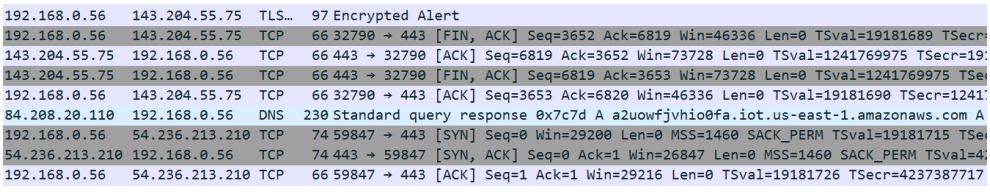


Figure 5.6: Identification of Irobot commando and control traffic displayed in Wireshark

A base filter is created based on the observations described earlier in this section to exclude irrelevant standby traffic from further analysis of events. DNS on-

line verification generated by the AP and TCP-keep-alive traffic is excluded with the same logical expression excluding all packets smaller then 98 bytes, *(frame.len > 97)*. This leaves the DNS responses to other Irobot cloud services which can be used for service attribution. ARP, DHCP and NTP are excluded due to functionality. The base filter created and used in following analysis of event captures are presented below. When this filter was applied to the *Standby traffic* capture the total number of packets was reduced from 5,052,284 to 4,010 displaying only 0.8% of the captured traffic.

- *(!ntp && !dhcp && !arp && frame.len > 97)*

5.2 General Event Analysis

This section introduces the general capturing process conducted in both **Oslo** and **Drammen** environment. Considerations and observations used during the event triggering process are also presented.

Capturing is started on the Raspberry PI for both WLAN and LAN, filters used on WLAN captures are the same in both environments due to static WLAN MAC address for the Irobot Roomba. This MAC address is found through the Irobot home application. However the simulated WAN address is different, in Oslo environment is was 192.168.0.56 and in Drammen is was 192.168.0.91, the reason for this was that the IP address reserved in Oslo was already in use within the Drammen LAN. Tshark commands for WLAN, Oslo LAN and Drammen LAN is listed below.

- WLAN: *sudo tshark -i wlan1 -f 'eth.host 50:6F:0C:2F:EB:A2' -w 'output.pcap'*
- Oslo: *sudo tshark -i eth0 -f 'ip.host "192.168.0.51' -w 'output.pcap'*
- Drammen: *sudo tshark -i eth0 -f 'ip.host "192.168.0.91' -w 'output.pcap'*

The initiated capture was continuous running in both environments during the entire event triggering, since the output files never exceeded 500MB. During the capturing all event objectives were triggered 10 times per environment, resulting in 20 captures per event in total. All cleaning events were triggered and identified as finished when the smart home application prompted a "finished cleaning" notification. Both start and end time for all events was noted and used in the event file extraction.

When all event triggering was conducted, the capturing was stopped and the environment capturing file, including all triggered events, were extracted to another computer via WinSCP file transfer application. The pcap files were opened in Wireshark and a time filter was applied to extract individual files for each of the different triggered events, resulting in 20 different files for each of the event objectives. Timefilter syntax and example is presented below.

- *(frame.time >= "Month day, year hh:mm:ss") && (frame.time <= "Month day, year hh:mm:ss")*

- *(*frame.time >= "Jan 1, 2023 01:00:00") && (frame.time <= "Jan 2, 2023 20:59:59")

The basefilter is applied to all event files excluding the irrelevant traffic identified in *Standby traffic* analysis, traffic left is then associated to actions or events triggered on the vacuum cleaner. Wireshark's protocol hierarchy tool was used to identify the protocol distribution of the events, further calculations about packets and bytes sent during the events were performed with a python scripts. Packet length sequences were extracted with python and manual analysis is used to identify common traffic patterns within each event. All observed patterns or characteristic were compared and used to create a event signature and detection algorithm.

These detection algorithms are applied to all 120 event files to evaluate the level of detection and to compare the different signatures with each other. If an algorithm have more then 90% detection rate and not identified in any other event objectives, it was possible to attribute.

5.3 Automated Cleaning

This section introduce specific configuration and decisions during *Automated cleaning* event and analysis. The results from the analysis will be presented at the end of the section.

Automated cleaning is integrated with IFTTT location service and a cleaning event is triggered when the user's phone is observed more than 200 meters away from the smart environment postal address. Start time is noted when a notification is received and event end is noted when a notification for finished cleaning is received. Due to time constrains during this master project several *Automated cleaning* events were triggered on the same day. Irobot restrictions only allow one triggering of these events each day, so the executed customized cleaning was deleted and then reconfigured after event end. This allowed more then one event per day.

Triggering date and time for all *Automated cleaning* events are presented for **Oslo** in Table 5.2 and **Drammen** in Table 5.3. The triggering of these events in **Drammen** follows a unrealistic time schedule due to limited availability of the smart environment. However the triggering in **Oslo** environment was triggered without recreating the cleaning configuration, executing one event per day. Attribution of this event exposes detailed information about the user's routines, and when the environment was empty.

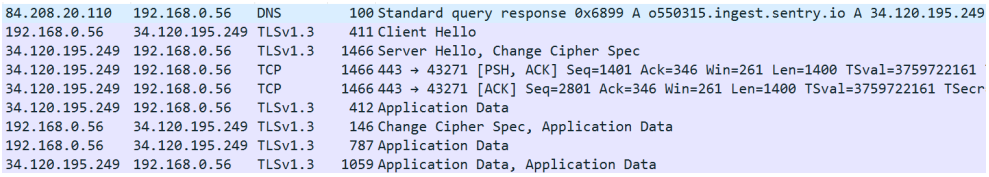
Table 5.2: Automated cleaning triggering date and time overview for Oslo

Event	Date	Start time	End time
1	21.02.2023	21:06	21:14
2	22.02.2023	07:37	07:45
3	23.02.2023	10:10	10:16
4	01.03.2023	07:42	07:47
5	02.03.2023	11:05	11:10
6	03.03.2023	07:03	07:08
7	06.03.2023	07:04	07:09
8	07.03.2023	08:42	08:47
9	08.03.2023	07:49	07:54
10	09.03.2023	07:22	07:29

Table 5.3: Automated cleaning triggering date and time overview for Drammen

Event	Date	Start time	End time
1	25.02.2023	21:32	21:50
2	26.02.2023	01:53	02:10
3	26.02.2023	15:43	15:55
4	26.02.2023	17:00	17:12
5	26.02.2023	22:11	22:23
6	27.02.2023	07:57	08:10
7	27.02.2023	08:51	09:02
8	27.02.2023	11:03	11:13
9	27.02.2023	12:04	12:16
10	27.02.2023	13:36	13:48

Protocol distribution shows that there are two DNS response packets in each of the event files, the rest of the traffic is TCP where the majority are TLS encrypted. TCP packets without TLS is either TCP acknowledgement packets without payload or a part of a TCP three-way-handshake or tare-down. When the event is triggered it is observed an increase in packets between the Irobot Roomba and the corresponding Irobot cloud server, this communication is at a consistent level until a cleaning is done. Then a DNS response for FQDN *0550315.ingest.sentry.io* is received and a new TCP session to one of the responded IP addresses is established. The entire corresponding TCP session from Oslo environment event 5 is shown in Figure 5.7, this is consistent for all *Automated cleaning* events. After this session is finished, a new DNS response for FQDN *s3.amazoneaws.com* is received and a new TCP session is established to one of the responded IP addresses.



```
84.208.20.110   192.168.0.56   DNS         100 Standard query response 0x6899 A o550315.ingest.sentry.io A 34.120.195.249
192.168.0.56    34.120.195.249 TLSv1.3     411 Client Hello
34.120.195.249  192.168.0.56   TLSv1.3    1466 Server Hello, Change Cipher Spec
34.120.195.249  192.168.0.56   TCP        1466 443 → 43271 [PSH, ACK] Seq=1401 Ack=346 Win=261 Len=1400 TSval=3759722161
34.120.195.249  192.168.0.56   TCP        1466 443 → 43271 [ACK] Seq=2801 Ack=346 Win=261 Len=1400 TSval=3759722161 TSecr
34.120.195.249  192.168.0.56   TLSv1.3     412 Application Data
192.168.0.56    34.120.195.249 TLSv1.3     146 Change Cipher Spec, Application Data
192.168.0.56    34.120.195.249 TLSv1.3     787 Application Data
34.120.195.249  192.168.0.56   TLSv1.3    1059 Application Data, Application Data
```

Figure 5.7: TCP session between Irobot Roomba and *0550315.ingest.sentry.io*, initiated after a cleaning is finished

Calculations of number of bytes and number of packets with associated average and Standard Deviation (SD) are presented in Table 5.4 and Table 5.5. As presented, the average and standard deviation in Oslo environment is significantly higher than in Drammen, regardless of that the Drammen cleaning had a longer duration overall. This might be due to the number of obstacles or carpet in the environments. An interesting calculation is that more than 90% of all packets captured are sent between the Irobot Roomba and *s3.amazoneaws.com*, due to large packet sizes in the session this includes more than 95% of the transferred bytes. TLS encryption hides the information, but this is most likely an upload of all the cleaning data collected during the event, and if encryption is broken a lot of private information can be exposed. Event 6 in Oslo environment has a low number of bytes and packets compared to the other events, but included both DNS responses and upload traffic associated with them. This could be due to lack of updates or disruption on transmission.

Table 5.4: Overall statistics for Automated Cleaning in Oslo

Event	Packet number	Total bytes sent
Event 1	2,703	3,882,164
Event 2	2,736	3,811,206
Event 3	2,659	3,818,287
Event 4	2,681	3,747,788
Event 5	2,589	3,704,329
Event 6	236	237,163
Event 7	2,701	3,860,634
Event 8	2,631	3,770,236
Event 9	2,609	3,738,406
Event 10	2,609	3,799,896
Average	2,415.4	3,437,010.9
SD	767.27	1,125,643

Table 5.5: Overall statistics for Automated Cleaning in Drammen

Event	Packet number	Total bytes sent
Event 1	1,074	1,443,851
Event 2	1,131	1,524,076
Event 3	1,209	1,644,223
Event 4	1,207	1,641,145
Event 5	1,013	1,422,457
Event 6	1,013	1,422,457
Event 7	1,220	1,726,862
Event 8	1,248	1,715,015
Event 9	1,227	1,669,154
Event 10	1,456	1,838,832
Average	1,179.8	1,604,807.2
SD	131.48	144,434.9

The first 20 packet lengths in all event captures was extracted with a python script presented in Appendix B and are analyzed to find common sequences that can be used to identify the event. These packet lengths are shown in Figure 5.8. D and S in front of the packet lengths indicates if the Irobot Roomba is destination or source of the packet. The yellow marked fields are the common identified sequence pattern in all event captures. The sequence starts with two packets sent from the Irobot cloud server to the Irobot Roomba with the lengths of *315 or 316* and *288 or 289* bytes, these two packets can be received in mixed order, but always appear as a packet pair. The Irobot Roomba then responded with a packet pair with lengths of *176* and *186 or 187* bytes. This is followed by a packet from the Irobot cloud server with the length of *408 or 409* bytes. The entire identified packet sequence is therefore *[315, 288, 176, 186, 408]* or *[288, 315, 176, 186, 408]*, with a offset of 1 byte due to the variation of packet size within the sequence. Oslo event 2 and Drammen event 8 does not include this sequence, and it looks like one of the packet pairs are merged.

Oslo
Event 1: **D 289, D 316, D 316, S 176, S 187, D 409**, D 404, S 175, S 480, D 1140, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475, S 179, S 253
Event 2: D 510, **S 176, S 187, D 409**, D 271, S 179, S 440, D 1100, S 175, S 405, D 988, S 179, S 446, D 1106, S 176, S 342, S 179, S 253, D 626, S 179
Event 3: **D 315, D 288, S 176, S 186, D 408**, D 271, D 271, S 175, S 405, D 988, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 342, S 179, S 253
Event 4: **D 315, D 288, S 176, S 186, D 408**, D 404, S 175, S 480, D 1140, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475, S 179, S 253, D 626
Event 5: **D 316, D 289, S 176, S 187, D 409**, D 271, S 175, S 405, D 988, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 342, S 179, S 253, D 626
Event 6: **D 316, D 289, S 176, S 187, D 409**, D 271, S 175, S 405, D 988, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 342, S 179, S 253, D 626
Event 7: S 172, S 179, D 392, **D 315, D 288, S 176, S 186, D 408**, D 271, S 179, S 440, D 1100, S 175, S 405, D 988, S 179, S 446, D 1106, S 176, S 342
Event 8: S 172, S 179, D 392, **D 315, D 288, S 176, S 186, D 408**, D 271, S 179, S 440, D 1100, S 175, S 405, D 988, S 179, S 446, D 1106, S 176, S 342
Event 9: **D 289, D 316, S 176, S 187, D 409**, D 271, S 179, S 440, D 1100, S 175, S 405, D 988, S 179, S 446, D 1106, S 176, S 342, S 176, S 483, S 179
Event 10: S 172, S 291, D 859, **D 288, D 315, S 176, S 186, D 408**, D 271, S 175, S 405, D 988, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 342
Drammen
Event 1: **D 315, D 288, S 176, S 186, D 408**, D 404, S 175, S 425, D 982, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176, S 474, S 176, S 615, S 179
Event 2: **D 289, D 316, S 176, S 187, D 409**, D 404, S 175, S 449, D 1046, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475, S 179, S 253, D 626
Event 3: S 172, S 288, D 714, **D 316, D 289, S 176, S 187, D 409**, D 404, S 175, S 425, D 982, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176, S 474
Event 4: **D 315, D 288, S 176, S 186, D 408**, D 404, S 179, S 439, D 1099, S 175, S 425, D 982, S 179, S 445, D 1105, S 176, S 615, S 179, S 253, D 626
Event 5: **D 289, D 316, S 176, S 187, D 409**, D 404, S 175, S 410, D 936, S 179, S 446, D 1106, S 176, S 475, S 176, S 616, S 179, S 253, D 626, S 179
Event 6: **D 289, D 316, D 316, S 176, S 187, D 409**, D 404, S 175, S 449, D 1046, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475, S 176, S 616
Event 7: **D 288, D 315, S 176, S 186, D 408**, D 404, S 175, S 449, D 1046, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475, S 176, S 616, S 179
Event 8: **D 289, D 316**, S 297, **D 409**, D 404, S 179, S 440, D 1100, S 175, S 449, D 1046, S 179, S 446, D 1106, S 176, S 475, S 176, S 616, S 179, S 253
Event 9: S 172, S 233, D 551, **D 315, D 288, S 176, S 186, D 408**, D 404, S 175, S 449, D 1046, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 475
Event 10: **D 289, D 316, S 176, S 187, D 409**, D 404, S 179, S 440, D 1100, S 175, S 449, D 1046, S 179, S 446, D 1106, S 176, S 475, S 176, S 616, S 179

Figure 5.8: Automated Cleaning extracted packet length sequences

Both packet length sequences and the presence of two FQDN are identified in all *Automated cleaning* event captures. These observations forms the signature for this event. Two python functions were created: one for the detection of the DNS responses for *0550315.ingest.sentry.io* and *s3.amazoneaws.com* and one to identify the packet sequences *[315, 288, 176, 186, 408]* or *[288, 315, 176, 186, 408]*. The pseudo code for DNS detection is presented in Figure 5.9 and sequence detection in Figure 5.10.

```
1     function cleaning_event_detection(event_capture)
2          if 'o550315.ingest.sentry.io' in event_capture
3               dns1 == True
4          elsif
5               dns1 == False
6          if 's3.amazonaws.com' in event_capture
7               dns2 == True
8          elsif
9               dns2 == False
10         if dns1 and dns2 == True
11              cleaning_confidence = + 10
12         return cleaning_confidence
```

Figure 5.9: The algorithm for identifying DNS responses for FQDNs *0550315.ingest.sentry.io* and *s3.amazoneaws.com*

```
1     function detect_application_start(packet_lengths_src)
2          signature1 = [316, 289, 176, 187, 409]
3          signature2 = [289, 316, 176, 187, 409]
4          if signature1 in packet_lengths
5               automated_cleaning_confidence = + 10
6          elsif signature1 in packet_lengths
7               automated_cleaning_confidence = + 10
```

Figure 5.10: The algorithm for identifying the *Automated cleaning* packet length sequence

To evaluate the detection algorithm it was used on all the different *Automated cleaning* event files. The DNS signature detection algorithm was able to identify the DNS responses for *0550315.ingest.sentry.io* and *s3.amazoneaws.com* in all 20 event files. The sequence signature detection algorithm was able to identify the signature in 18 of 20 event files, resulting in a successful detection rate of 90%. The two events that did not include this signature were Oslo event 2 and Drammen event 8, they had the sequences *510, 176, 187, 409* and *289, 316, 297, 409*. As mentioned in the sequence analysis the sequence detection will not be able to identify these.

5.4 Application Triggered Cleaning

This section introduce specific configuration and decisions during the *Application triggered cleaning* event triggering and analysis. The results from the analysis will be presented at the end of the section.

Application triggered cleaning is an event triggered manually by the Irobot Roomba's users through the Irobot home application's predefined or customized cleaning jobs. The event used a customized cleaning job defining the entire area of Oslo or Drammen smart map. This ensured that the cleaning area is constant for all the triggered events, creating the best foundation for comparison. These events can be triggered as many times as the user would like and is therefore the same during the entire capturing phase.

Triggering date and time for all *Application triggered cleaning* events are presented in Table 5.6 for Oslo and Table 5.7 for Drammen. These events are triggered in a non-realistic manner due to time constrains, this is especially for the Drammen triggering where a new event was triggered every 30 minutes. In a realistic smart environment these triggerings would occur less and not in a structured manner. *Application triggered cleaning* is triggered when the user needs additional cleaning outside of the scheduled one, preferably triggering this when leaving home. Identification of this event can therefore expose private information about user location.

Table 5.6: Application triggered cleaning triggering date and time overview for Oslo environment.

Event	Date	Start time	End time
1	28.02.2023	18:20	18:27
2	28.02.2023	18:35	18:42
3	01.03.2023	18:53	19:00
4	09.03.2023	07:44	07:49
5	09.03.2023	08:03	08:10
6	09.03.2023	08:25	08:31
7	09.03.2023	08:57	09:04
8	09.03.2023	09:18	09:26
9	12.03.2023	12:20	12:35
10	12.03.2023	12:54	13:09

Table 5.7: Application triggered cleaning triggering date and time overview for Drammen environment.

Event	Date	Start time	End time
1	25.02.2023	14:30	14:45
2	25.02.2023	15:00	15:15
3	25.02.2023	15:30	15:45
4	25.02.2023	16:00	16:15
5	25.02.2023	16:30	16:45
6	25.02.2023	17:00	17:15
7	25.02.2023	17:30	17:45
8	25.02.2023	19:00	19:15
9	25.02.2023	19:30	19:45
10	25.02.2023	20:00	20:15

Protocol distribution shows that there are two DNS response packets in each of the event files, the rest of the traffic is TCP where the majority are TLS encrypted. TCP packets without TLS are either TCP acknowledgement packets without payload or a part of a TCP three-way-handshake or tare-down. When the event is triggered it is observed an increase in packets between the Irobot Roomba and the corresponding Irobot cloud server, this communication is at a consistent level until the cleaning is done. Then a DNS response FQDN *0550315.ingest.sentry.io* is received and a new TCP session to one of the responded IP addresses is established. The entire corresponding TCP session from Oslo environment event 5 is shown in Figure 5.11, this is consistent for all *Application triggered cleaning* events. After this session is finished a new DNS response for FQDN *s3.amazoneaws.com* is received and a new TCP session is established to one of the responded IP addresses.

```
84.208.20.110   192.168.0.56   DNS          100 Standard query response 0x3a1c A o550315.ingest.sentry.io A 34.120.195.249
192.168.0.56    34.120.195.249 TLSv1.3      726 Client Hello
34.120.195.249  192.168.0.56   TLSv1.3      300 Server Hello, Change Cipher Spec, Application Data
192.168.0.56    34.120.195.249 TLSv1.3      146 Change Cipher Spec, Application Data
192.168.0.56    34.120.195.249 TLSv1.3      787 Application Data
34.120.195.249  192.168.0.56   TLSv1.3     1059 Application Data, Application Data
```

Figure 5.11: TCP session between Irobot Roomba and *0550315.ingest.sentry.io*, initiated after application triggered cleaning is finished

Calculation of number of bytes and packets sent with associated average and standard deviation for all *Application triggered cleaning* events are presented in Table 5.8 and Table 5.9. Both number of packets and bytes sent are higher in Oslo despite shorter duration of cleaning, this could be environment specific due to floor, windows and furniture within each environment. More then 90% of the packets are sent from the Irobot Roomba to FQDN *s3.amazoneaws.com*, as the majority of these packets are close to maximum packet size of 1,500 bytes they account for more then 95% of the bytes transferred. This is most likely an upload of all the collected information form the cleaning. In Drammen environment both number of packets and bytes sent are increased for each consecutive event, based on similar duration and encrypted traffic it is not possible to determine why this occurred in Drammen.

Table 5.8: Application triggered cleaning, overall statistics Oslo

Event	Number of packets	Total number of bytes
Event 1	2,667	3,732,218
Event 2	2,686	3,730,748
Event 3	2,656	3,710,958
Event 4	3,065	4,365,510
Event 5	2,880	4,076,534
Event 6	2,633	3,771,269
Event 7	2,661	3,786,648
Event 8	2,647	3,798,184
Event 9	2,729	3,798,630
Event 10	2,639	3,786,194
Average	2,726.3	3,855,689.3
SD	139.57	206,499.4

Table 5.9: Application triggered cleaning, overall statistics Drammen

Event	Number of packets	Number of bytes
Event 1	708	802,488
Event 2	724	945,100
Event 3	740	956,470
Event 4	809	1,071,613
Event 5	824	1,088,836
Event 6	869	1,157,323
Event 7	855	1,144,086
Event 8	929	1,248,783
Event 9	988	1,321,061
Event 10	983	1,339,631
Average	842.9	1,107,539.1
SD	101.85	172,281.2

The first 20 packet lengths of each of the *Application triggered cleaning* events are extracted with the python script presented in Appendix B. These sequences are manually analyzed to identify common packet length sequence to be used in event attribution. Packet lengths for all events are presented in Figure 5.12, D and S indicate if the Irobot Roomba is destination or source of the packet. The yellow marked fields are the common sequences identified in the majority of event captures. Irobot cloud server is initiating the event with three packets with the length of *[208, 288, 315]*, the last two packets can occur in mixed order, and all length can vary with one byte. Three packets are used to keep the complexity of the signature low. Identified sequence signature for *Application triggered cleaning* is *[208, 288, 315]* with the offset of 1 bytes for all packet sizes. Oslo event 9 is the only event capture wich does not include the identified signature, expected success rate of evaluation is therefore 95%.

Oslo
Event 1: **D 208, D 288, D 315**, S 176, S 186, D 408, S 176, S 1285, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176, S 625
Event 2: S 179, S 160, D 346, **D 209, D 289, D 316**, S 176, S 187, D 409, S 176, S 1201, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445
Event 3: **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 1064, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176
Event 4: S 179, S 160, D 346, **D 208, D 288, D 315**, S 176, S 186, D 408, S 176, S 1200, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445
Event 5: S 179, S 160, D 346, **D 208, D 288, D 315**, S 176, S 186, D 408, S 176, S 1200, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445
Event 6: S 179, S 160, D 346, S 172, S 233, D 551, **D 209, D 289, D 316**, S 176, S 187, D 409, S 176, S 1201, D 555, S 175, S 561, D 1239, S 179, S 439
Event 7: **D 209, D 315, D 288**, S 176, S 186, D 408, S 176, S 1201, S 179, S 160, D 346, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445
Event 8: S 179, S 160, D 346, **D 209, D 316, D 289**, S 176, S 187, D 409, S 176, S 1201, D 555, S 179, S 439, D 1099, S 175, S 561, D 1239, S 179, S 445
Event 9: D 209, D 510, S 176, S 1201, S 176, S 187, D 409, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176, S 625, S 179
Event 10: **D 209, D 288, D 315**, S 176, S 1201, S 176, S 186, D 408, D 555, S 175, S 561, D 1239, S 179, S 439, D 1099, S 179, S 445, D 1105, S 176, S 625
Drammen
Event 1: **D 209, D 315, D 288**, S 176, S 186, D 408, S 176, S 949, D 503, S 175, S 540, D 1200, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 715
Event 2: **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 227, D 503, S 175, S 509, D 1106, S 179, S 440, D 1100, S 179, S 446, D 1106, S 574
Event 3: **D 209, D 316, D 289**, S 176, S 187, D 409, S 176, S 1514, S 1514, S 787, D 503, D 503, S 175, S 524, D 1152, S 179, S 440, D 1100, S 179, S 446
Event 4: **D 209, D 288, D 315**, S 176, S 1085, S 176, S 186, D 408, D 503, S 175, S 524, D 1152, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176, S 574
Event 5: S 179, S 160, D 346, **D 209, D 316, D 289**, S 176, S 1514, S 256, S 176, S 187, D 409, D 503, S 175, S 540, D 1200, S 179, S 440, D 1100, S 179
Event 6: S 172, S 179, D 392, **D 208, D 315, D 288**, S 176, S 186, D 408, S 176, S 1359, D 503, S 175, S 509, D 1106, S 179, S 440, D 1100, S 179, S 446
Event 7: **D 207, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 1514, S 1514, S 1514, S 1514, S 205, D 318, S 175, S 398, D 869, S 179, S 440, D 1100
Event 8: **D 207, D 315, D 288**, S 176, S 186, D 408, S 176, S 1514, S 1514, S 926, D 318, S 175, S 398, D 869, S 179, S 440, D 1100, S 179, S 446, D 1106
Event 9: **D 208, D 315, D 288**, S 176, S 985, S 176, S 186, D 408, D 208, S 176, D 316, D 289, S 985, S 176, S 187, D 409, D 503, S 175, S 509, D 1106
Event 10: **D 208, D 315, D 288**, S 176, S 186, D 408, S 176, S 1514, S 255, D 503, S 175, S 509, D 1106, S 179, S 440, D 1100, S 179, S 446, D 1106, S 176

Figure 5.12: *Application triggered* cleaning packet length sequence

To identify *Application triggered cleaning* two detection algorithms are made, one identifying the DNS responses *0550315.ingest.sentry.io* and *s3.amazoneaws.com* and one to identify the sequence signature specified above. Pseudo code for both algorithms are presented in Figure 5.13 and Figure 5.14.

```
1   function cleaning_event_detection(event_capture)
2       if 'o550315.ingest.sentry.io' in event_capture
3           dns1 == True
4       elsif
5           dns1 == False
6       if 's3.amazonaws.com' in event_capture
7           dns2 == True
8       elsif
9           dns2 == False
10      if dns1 and dns2 == True
11          cleaning_confidence = + 10
12      return cleaning_confidence
```

Figure 5.13: The algorithm for DNS detection in Application triggered cleaning events

```
1   function detect_ATC(packet_lengths)
2       signature1 = [209,289,316]
3       signature2 = [209.316.289]
4       if signature1 in packet_lengths
5           ATC_confidence = + 10
6       elsif signature1 in packet_lengths
7           ATC_confidence = + 10
8       return ATC_confidence
```

Figure 5.14: The algorithm for identifying the Application triggered cleaning packet sequence signature

To evaluate the detection algorithms they are tested on all the capturing files for *Application triggered cleaning*, trying to identify both signatures. The results of the identification are as expected, DNS detection had a success rate of 100% and the sequence signature detection had a success rate of 95%. The isolated event can therefore be identified with a high confidence, based on these results.

5.5 Scheduled Cleaning

This section introduce specific configuration and decisions during *Scheduled cleaning* event and analysis. The results from the analysis will be presented at the end of the section.

Scheduled cleaning is triggered through configuration of customized cleaning jobs in the Irobot home application. A scheduled cleaning job was configured specifying the entire smart map as the cleaning area. Irobot restricts its users to not configure two scheduled cleanings with less then 3 hours space, cleaning jobs with less then 3 hours in between creates an error message. During this project only one scheduled clean was configured, but after the cleaning was finished the user entered the application changing the configured time. This enabled scheduled cleanings to be executed with less then 3 hours in between. It was observed that whenever a scheduled cleaning job was changed, the Irobot Roomba made a sound indicating that it got an update. The scheduled cleaning job is most likely sent to the vacuum cleaner at the time of configuration and then triggered locally. This way the Irobot Roomba is able to clean without Internet connection at the triggering time.

Triggering dates and time for *Scheduled cleaning* are presented in Table 5.10 and 5.11. These events are triggered in a non-realistic way due to time constrains, this is supported by Irobot's own restrictions mentioned above. Timings from *Scheduled cleaning* can therefore not be used in the attribution of this specific event, however it would be a good indication if collected by an attack over a longer period of time. If a cleaning event is detected every Monday, Wednesday and Friday at 09:00 it is most likely due to scheduled cleaning, because a human triggered event would differ more in time. An observation during triggering was that the Irobot Roomba always started within 30 seconds before the scheduled time, this is also observed in the actual packet capture. In Figure 5.15 event 2 in Oslo is shown, and there the traffic starts right before the scheduled timestamp. This is applicable for all events in the rage of 30-0 seconds before scheduled time.

Table 5.10: Scheduled cleaning triggering date and time overview for Oslo

Event	Date	Start time	End time
1	10.03.2023	10:45	10:54
2	10.03.2023	11:15	11:24
3	10.03.2023	12:15	12:24
4	10.03.2023	13:30	13:39
5	10.03.2023	15:00	15:09
6	10.03.2023	15:30	15:39
7	10.03.2023	15:55	16:04
8	10.03.2023	16:10	16:19
9	11.03.2023	10:10	10:19
10	11.03.2023	10:30	10:39

Table 5.11: Scheduled cleaning triggering date and time overview for Drammen

Event	Date	Start time	End time
1	25.02.2023	20:30	20:45
2	25.02.2023	21:00	21:15
3	26.02.2023	11:20	11:35
4	26.02.2023	11:50	12:05
5	26.02.2023	12:20	12:35
6	26.02.2023	12:50	13:05
7	26.02.2023	13:20	13:35
8	26.02.2023	13:50	14:05
9	26.02.2023	14:20	14:35
10	26.02.2023	15:00	15:15

	Time	Source	Destination	Protocol	Length	Info
1	2023-03-10 11:14:58,349776	192.168.0.56	54.158.18.46	TLSv1.2	175	Application Data
2	2023-03-10 11:14:58,492115	192.168.0.56	54.158.18.46	TLSv1.2	466	Application Data
3	2023-03-10 11:14:58,613240	54.158.18.46	192.168.0.56	TLSv1.2	1094	Application Data
4	2023-03-10 11:14:58,615823	192.168.0.56	54.158.18.46	TLSv1.2	179	Application Data
5	2023-03-10 11:14:58,754391	192.168.0.56	54.158.18.46	TLSv1.2	442	Application Data
6	2023-03-10 11:14:58,870833	54.158.18.46	192.168.0.56	TLSv1.2	1102	Application Data
7	2023-03-10 11:14:58,883248	192.168.0.56	54.158.18.46	TLSv1.2	179	Application Data

Figure 5.15: Start time for Scheduled cleaning in Oslo event 2, cleaning was scheduled 11:15

Protocol distribution and overall traffic flow are similar as presented for *Automated cleaning* and *Application triggered cleaning*. The two same DNS responses listed below are observed.

- 0550315.ingest.sentry.io
- s3.amazoneaws.com

Calculations of number of packets and bytes sent and SD are presented in Table 5.12 and Table 5.13. Average and standard deviation in both Oslo and Drammen environment is small indicating small differences in the triggered events, this could be due to locally triggered events on the Irobot Roomba.

Table 5.12: Scheduled cleaning, overall statistics Oslo smart home

Event	Number of packets	Number of bytes
Event 1	2,541	3,669,941
Event 2	2,633	3,678,959
Event 3	2,622	3,660,004
Event 4	2,524	3,629,262
Event 5	2,627	3,658,515
Event 6	2,608	3,729,110
Event 7	2,596	3,645,238
Event 8	2,655	3,685,536
Event 9	2,573	3,713,211
Event 10	2,636	3,768,883
Average	2,601.5	3,683,865.9
SD	43.03	42,219.79

Table 5.13: Scheduled cleaning, overall statistics Drammen

Event	Number of pkt	Number of bytes
Event 1	996	1,354,755
Event 2	1,052	1,422,052
Event 3	1,150	1,592,566
Event 4	1,317	1,650,499
Event 5	1,166	1,570,805
Event 6	1,179	1,612,050
Event 7	1,160	1,582,275
Event 8	1,177	1,610,090
Event 9	1,205	1,634,883
Event 10	1,170	1,590,726
Average	1,157.2	1,562,070.1
SD	85,66	95,885,11

The 20 first packet lengths were extracted with a python script presented in Appendix B. The manual analysis of the packet lengths presented in Figure 5.16, resulted in two packet sequences of outbound traffic from the Irobot Roomba. The packet lengths varied more than previously and had a byte offset of 15 and 5 bytes depending on which signature sequence that was used. Both signature sequences are listed below with the associated byte offset.

- 176, 173, 179, 443, 177, byte offset 15 bytes
- 176, 443, 179, 443, 177, byte offset 5 bytes

Oslo
Event 1: **S 179, S 160**, D 346, S 175, S 482, D 1142, **S 179, S 441**, D 1101, **S 179**, S 448, D 1108, S 176, S 477, S 179, S 253, D 626, S 179, S 448, D 1108
Event 2: S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179, S 448**, D 1108, **S 176**, S 477, S 179, S 253, D 626, S 179, S 448, D 1108, S 179, S 448, D 1108
Event 3: S 179, S 160, D 346, S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179, S 448**, D 1108, **S 176**, S 477, S 179, S 253, D 626, S 179, S 448, D 1108
Event 4: S 179, S 442, D 1102, S 175, S 482, D 1142, **S 179, S 448**, D 1108, **S 176, S 477**, **S 179**, S 253, D 626, S 179, S 448, D 1108, S 179, S 448, D 1108
Event 5: S 179, S 253, D 626, **S 179, S 448**, D 1108, **S 179, S 448**, D 1108, **S 176**, S 674, S 176, S 812, D 151, S 583, D 1494, D 1494, D 1494, D 1210, S 192
Event 6: **S 179, S 442**, D 1102, S 175, S 482, D 1142, **S 179, S 448**, D 1108, **S 176**, S 477, S 179, S 253, D 626, S 179, S 448, D 1108, S 179, S 448, D 1108
Event 7: **S 179, S 160**, D 346, S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 179, S 253, D 626, S 179, S 448, D 1108
Event 8: **S 179, S 160**, D 346, S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 179, S 253, D 626, S 179, S 448, D 1108
Event 9: **S 179, S 160**, D 346, S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 179, S 253, D 626, S 179, S 448, D 1108
Event 10: S 175, S 482, D 1142, **S 179, S 442**, D 1102, **S 179, S 448**, D 1108, **S 176**, S 477, S 179, S 253, D 626, S 179, S 448, D 1108, S 179, S 448, D 1108
Drammen
Event 1: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 179, S 253, D 626, S 179, S 448, D 1108, S 179, S 448, D 1108
Event 2: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 618, S 179, S 253, D 626, S 176, S 835, S 179, S 448, D 1108, S 179
Event 3: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 176, S 618, S 179, S 253, D 626, S 176, S 835, S 179, S 448
Event 4: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 176, S 618, S 179, S 253, D 626, S 179, S 448, D 1108, S 176
Event 5: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 618, S 179, S 253, D 626, S 179, S 448, D 1108, S 176, S 835, S 179
Event 6: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 176, S 618, S 179, S 253, D 626, S 176, S 835, S 179, S 448
Event 7: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 176, S 618, S 179, S 253, D 626, S 179, S 448, D 1108, S 179
Event 8: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 477, S 176, S 618, S 179, S 253, D 626, S 179, S 448, D 1108, S 179
Event 9: S 172, S 234, D 552, **S 175, S 466**, D 1094, **S 179, S 448**, D 1108, **S 176**, S 477, S 176, S 618, S 179, S 253, D 626, S 179, S 448, D 1108, S 179
Event 10: **S 175, S 466**, D 1094, **S 179, S 442**, D 1102, **S 179**, S 448, D 1108, S 176, S 618, S 179, S 253, D 626, S 176, S 835, S 179, S 448, D 1108, S 179

Figure 5.16: *Scheduled cleaning* packet length sequences

As the DNS signature is the same as for the two previous cleaning events analyzed, the same detection algorithm is used to evaluate this. Further, a new detection algorithm for the packet length sequence is presented in Figure 5.17. Both DNS and sequence detection had a success rate of 100% in all capturing files. Making this a good indication of the event if it is not detected in other events as well.

```
1    function detect_application_start(packet_lengths_src)
2        signature1 = [176,173,179,443,177]
3        signature2 = = [176,443,179,443,177]
4        if signature1 in packet_lengths
5            SC_PTC_confidence = + 10
6        elsif signature1 in packet_lengths
7            SC_PTC_confidence = + 10
8        return SC_PTC_confidence
```

Figure 5.17: Algorithm for scheduled cleaning packet sequence signature detection

5.6 Physical Triggered Cleaning

This section introduce specific configuration and decisions during *Physical triggered cleaning* event and analysis. The results from the analysis will be presented at the end of the section.

Physical triggered cleaning events are only triggered by pressing the physical button on the Irobot Roomba. During triggering the user had to press the button twice to start the cleaning. Event triggering dates and time are presented in Table 5.14 and Table 5.15. The triggering in Oslo appears more realistic due to Drammen's strict triggering plan, these timestamps can therefore not be used to

differentiate different cleaning events, but could have been a good indication if captured in a real-life smart environment.

Table 5.14: Physical triggered cleaning date and time overview for Oslo

Event	Date	Start time	End time
1	23.02.2023	18:08	18:24
2	23.02.2023	18:36	19:05
3	23.02.2023	19:14	19:34
4	23.02.2023	20:13	20:35
5	23.02.2023	20:44	21:06
6	09.03.2023	09:43	10:02
7	09.03.2023	10:30	10:50
8	09.03.2023	12:32	12:50
9	09.03.2023	13:16	14:05
10	09.03.2023	17:44	18:05

Table 5.15: Physical triggered cleaning date and time overview for Drammen

Event	Date	Start time	End time
1	25.02.2023	22:00	22:12
2	25.02.2023	22:30	22:45
3	25.02.2023	23:00	23:15
4	25.02.2023	23:20	23:35
5	25.02.2023	23:40	23:55
6	26.02.2023	00:00	00:15
7	26.02.2023	00:20	00:35
8	26.02.2023	00:40	00:55
9	26.02.2023	01:06	01:20
10	26.02.2023	01:31	01:45

Protocol distribution and traffic flow are similar as in the other three cleanings events presented above. The two same DNS responses are identified as well as their corresponding TCP sessions. Standard deviation for both environments are small, indicating that the events are similar and supporting that the hypothesis on 20 events are sufficient.

Table 5.16: Physical triggered cleaning, overall statistics Oslo

Event	Number of packets	Number of bytes
Event 1	2,791	3,965,147
Event 2	3,180	4,357,126
Event 3	2,926	4,033,946
Event 4	2,872	4,112,516
Event 5	2,944	4,209,443
Event 6	2,984	4,122,412
Event 7	2,925	4,160,462
Event 8	2,869	4,100,166
Event 9	2,918	4,227,626
Event 10	2,768	3,957,559
Average	2,917.7	4,124,640.3
SD	113.98	122,683.4

Table 5.17: Physical triggered cleaning, overall statistics Drammen

Event	Number of packets	NUmber of bytes
Event 1	1,078	1,481,677
Event 2	1,087	1,484,930
Event 3	1,125	1,520,872
Event 4	1,088	1,517,429
Event 5	1,150	1,555,605
Event 6	1,086	1,507,084
Event 7	1,092	1,501,894
Event 8	1,115	1,510,971
Event 9	1,098	1,485,863
Event 10	1,100	1,492,331
Average	1,101.9	1,505,865.6
SD	22.05	22,318.19

The first 20 packet lengths are extracted with the same python script as for the other events, presented in Appendix B. Analysis resulted in the identification of the same packet sequence as for *Scheduled cleaning* event, this could be because these two events are both triggered on the Irobot Roomba locally and not initiated from the Irobot cloud server. If this is the reason, the identified sequence could be present in all cleaning events. The extracted sequences are presented in Figure 5.18, yellow marked are the identified sequence and D and S represents if the Irobot Roomba is destination or source of the packet.

Oslo
Event 1: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 176**, S 475, S 175, S 369, D 903, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 2: **S 179, S 159**, D 345, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 369, D 903, S 176, S 290, S 179, S 253, D 626, S 172, S 179, D 392, S 179
Event 3: **S 179, S 160**, D 346, **S 179, S 440**, D 1100, **S 179**, S 446, D 1106, S 175, S 369, D 903, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179
Event 4: **S 179, S 160**, D 346, **S 179, S 440**, D 1100, **S 179**, S 446, D 1106, S 175, S 369, D 903, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179
Event 5: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 369, D 903, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 6: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 175**, S 369, D 903, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 7: **S 172, S 179**, D 392, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 369, D 903, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 8: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 369, D 903, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 9: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 369, D 903, S 172, S 233, D 551, S 176, S 290, S 179, S 253, D 626, S 179
Event 10: **S 179, S 159**, D 345, **S 179, S 440**, D 1100, **S 179**, S 446, D 1106, S 175, S 369, D 903, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179
Drammen
Event 1: S 175, S 314, D 745, **S 179, S 447**, D 1107, **S 179, S 447**, D 1107, **S 179**, S 445, D 1105, S 179, S 253, D 626, S 179, S 445, D 1105, S 179, S 445
Event 2: **S 179, S 446**, D 1106, S 176, S 290, S 175, S 338, D 809, S 176, S 290, S 179, S 253, D 626, **S 179, S 446**, D 1106, **S 179, S 446**, D 1106, **S 176**
Event 3: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 176**, S 290, S 175, S 338, D 809, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179
Event 4: **S 179, S 159**, D 345, **S 179, S 440**, D 1100, **S 179**, S 446, D 1106, S 175, S 338, D 809, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179
Event 5: **S 179, S 439**, D 1099, **S 179, S 445**, D 1105, **S 175**, S 314, D 745, S 176, S 289, S 176, S 289, S 179, S 253, D 626, S 179, S 445, D 1105, S 179
Event 6: **S 179, S 439**, D 1099, **S 179, S 445**, D 1105, **S 175**, S 314, D 745, S 176, S 430, S 179, S 253, D 626, S 179, S 445, D 1105, S 179, S 445, D 1105
Event 7: **S 179, S 160**, D 346, **S 179, S 439**, D 1099, **S 179**, S 445, D 1105, S 175, S 314, D 745, S 176, S 289, S 176, S 289, S 179, S 253, D 626, S 179
Event 8: S 172, S 233, D 551, **S 179, S 439**, D 1099, **S 179**, S 445, D 1105, S 175, S 314, D 745, S 176, S 289, S 176, S 289, S 179, S 253, D 626, S 179
Event 9: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 175**, S 338, D 809, S 176, S 574, S 176, S 431, S 179, S 253, D 626, S 176, S 648, S 179, S 446
Event 10: **S 179, S 440**, D 1100, **S 179, S 446**, D 1106, **S 175**, S 338, D 809, S 176, S 290, S 176, S 290, S 179, S 253, D 626, S 179, S 446, D 1106, S 179

Figure 5.18: Physical triggered cleaning packet length sequences

The identified signatures are identical as for *Scheduled cleaning*, the two same signature detection algorithms were used to evaluate the findings. Both algorithms had a 100% success rate for identifying *Physical triggered cleaning*, this means that neither of the events can be distinguished with these signatures. This attribution could been done by using timestamps, but the events in this research have too unrealistic triggering.

5.7 Application Start

This section introduce specific configuration and decisions during *Application start* event and analysis. The results from the analysis will be presented at the end of the section.

Application start events are triggered by the user opening the Irobot application. Triggering dates and times are presented in Table 5.18 and Table 5.19. No specific action was defined during the event, so several different actions was executed such as, changing scheduled cleaning time, watch the dashboard and display configuration. Only the initial event traffic will therefore be included in the application open event.

Table 5.18: Application start event date and time overview for Oslo

Event	Date	Start time	End time
1	10.03.2023	10:26	10:27
2	10.03.2023	11:06	11:07
3	10.03.2023	11:56	11:57
4	10.03.2023	13:22	13:23
5	10.03.2023	14:58	14:59
6	10.03.2023	15:27	15:28
7	10.03.2023	15:51	15:52
8	10.03.2023	16:07	16:08
9	11.03.2023	10:06	10:07
10	11.03.2023	10:22	10:23

Table 5.19: Application start event date and time overview for Drammen

Event	Date	Start time	End time
1	25.02.2023	20:50	20:52
2	25.02.2023	21:20	21:21
3	25.02.2023	22:20	22:22
4	25.02.2023	22:50	22:52
5	26.02.2023	11:10	11:11
6	26.02.2023	11:40	11:41
7	26.02.2023	12:10	12:11
8	26.02.2023	12:40	12:41
9	26.02.2023	13:10	13:11
10	26.02.2023	13:40	13:41

The protocol distribution analysis identified only TCP packets, no DNS packets is sent during the event, indication that the requested information pulled from the Irobot Roomba when the application is started is initiated from *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com*. If the smart phone had been located in the same smart environment, it could be possible to identify a DNS request to this service. No standard action was performed in the application during this event, the standard deviation from the calculations presented in Table 5.20 and Table 5.21 is therefore large, and only the initiating traffic is relevant to this analysis. During event 5-10 in Drammen, the user performed a scheduled clean configuration resulting in a high number of bytes sent compared to some of the other events.

Table 5.20: Application start event overall statistics Oslo

Event	Number of packets	Number of bytes
Event 1	11	5,202
Event 2	22	8,698
Event 3	20	8,644
Event 4	17	7,135
Event 5	20	8,580
Event 6	20	8,608
Event 7	20	9,213
Event 8	23	9,527
Event 9	20	8,730
Event 10	19	8,877
Average	19.2	8,321.4
SD	3.29	1,258.62

Table 5.21: Application start, overall statistics Drammen

Event	Packet number	Total bytes sent
Event 1	30	20,875
Event 2	26	19,568
Event 3	8	2,655
Event 4	8	2,659
Event 5	26	19,561
Event 6	34	21,897
Event 7	29	20,222
Event 8	25	19,468
Event 9	33	21,744
Event 10	26	19,570
Average	24,5	16,821.9
SD	9.22	7.519.30

The first 20 packet lengths were extracted with the python script presented in Appendix B, and the result is presented in Figure 5.19. The yellow marked fields are a part of the identified packet length sequence. The identified sequences are *[209, 288, 315]* and *[209, 315, 288]*, where both sequences have an offset of 1 byte. This is the same packet sequence signature as identified in *Application triggered cleaning*, and is therefore used to identify *Application start*. These sequences are therefore generated whenever the Irobot home application is started. To differentiate these two events the DNS signature found in all cleaning events have to be identified.

Oslo
Event 1: **D 209, D 315, D 288**, S 296, D 408, S 176, S 1053, D 1514, D 1514, D 1084, D 1514, D 1514, D 1111, S 174, S 140, D 333, S 175, S 1514, S 569, D 1514
Event 2: **D 208, D 316, D 289**, S 176, S 187, D 409, S 176, S 1052, D 1514, D 1514, D 1112, D 1514, D 1514, D 1085, S 174, S 140, D 333, S 175, S 1514, S 570
Event 3: **D 208**, D 537, S 176, S 186, D 408, S 176, S 1052, D 1514, D 1514, D 1085, D 1514, D 1514, D 1112, S 174, S 140, D 333, S 175, S 1514, S 570, D 1514
Event 4: S 179, S 160, D 346, **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1052, D 1514, D 1514, D 1111, D 1514, D 1514, D 1084, S 174, S 140, D 333
Event 5: **D 209, D 315, D 288**, S 176, S 186, D 408, S 176, S 1053, D 1514, D 1514, D 1085, D 1514, D 1514, D 1112, S 174, S 140, D 333, S 175, S 1514, S 570
Event 6: **D 205, D 289, D 316**, S 176, S 1046, S 176, S 187, D 409, D 1514, D 1514, D 1112, D 1514, D 1514, D 1085, S 174, S 140, D 333, S 175, S 1514, S 570
Event 7: **D 207, D 315, D 288**, S 176, S 186, D 408, S 176, S 1051, D 1514, D 1514, D 1085, D 1514, D 1514, D 1112, S 174, S 140, D 333, S 175, S 1514, S 570
Event 8: S 179, S 159, D 345, **D 207, D 289, D 316**, S 176, S 187, D 409, S 176, S 1051, D 1514, D 1514, D 1514, D 1514, D 1514, D 654, S 174, S 140, D 333
Event 9: **D 207**, D 508, S 176, S 1050, S 176, S 186, D 408, D 1514, D 1514, D 1112, D 1514, D 1514, D 1085, S 174, S 140, D 333, S 175, S 1514, S 570, D 1514
Event 10: **D 208, D 316, D 289**, S 176, S 1052, S 176, S 187, D 409, S 172, S 219, D 505, D 1514, D 1514, D 1085, D 1514, D 1514, D 1112, S 174, S 140, D 333
Drammen
Event 1: **D 209, D 288, D 315**, S 296, D 408, S 176, S 1514, S 376, D 879, D 852, S 174, S 140, D 333, S 175, S 469, D 904, S 175, S 346, D 848
Event 2: **D 209, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 131, D 852, D 879, S 174, S 140, D 333, S 175, S 469, D 904, S 175, S 346, D 848
Event 3: **D 209, D 315, D 288**, S 176, S 186, D 408, S 176, S 1514, S 131, D 879, D 852, S 174, S 140, D 333, S 175, S 469, D 904, S 175, S 346, D 848
Event 4: **D 209, D 289, D 316**, S 176, S 1514, S 159, S 176, S 187, D 409, D 852, S 174, D 879, S 140, D 333, S 175, S 469, D 904, S 175, S 346, D 848
Event 5: S 172, S 179, D 392, **D 208, D 315, D 288**, S 176, S 186, D 408, S 176, S 987, D 825, D 852, S 174, S 140, D 333, S 175, S 466, D 877, S 175
Event 6: **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 158, D 852, D 825, S 174, S 140, D 333, S 175, S 466, D 877, S 175, S 346, D 848
Event 7: **D 208, D 288, D 315**, S 176, S 1361, S 176, S 186, D 408, D 852, D 825, S 174, S 140, D 333, S 175, S 466, D 877, S 175
Event 8: **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1514, S 98, D 851, D 824, S 174, S 140, D 333, S 175, S 465, D 876, S 175, S 346, D 848
Event 9: **D 208, D 288, D 315**, S 176, S 1514, S 126, S 176, S 186, D 408, D 825, D 852, S 174, S 140, D 333, S 175, S 466, D 877, S 175, S 346, D 848
Event 10: **D 208, D 289, D 316**, S 176, S 187, D 409, S 176, S 1389, D 825, D 852, S 174, D 852, S 140, D 333, S 175, S 466, D 877, S 175, S 346, D 848

Figure 5.19: Application start packet length sequences

Evaluation of *Application start* detection is determined with the same packet length sequence detection algorithm presented in Figure 5.14. This detection had a success rate of 90% detection of the event, and it is therefore possible to determine that the application is started with high confidence.

5.8 Bin Remove

This section introduce specific configuration and decisions during *Bin remove* event and analysis. The results from the analysis will be presented at the end of the section.

Bin remove is triggered when the physical bin eject button is pressed on the Irobot Roomba i7 and the bin is then released. It is injected by pushing the bin back into place. Triggering dates and times are presented in Table 5.22 and 5.23, intervals between the triggering is small, but in analysis it was possible to differentiate when the different traffic occurred. During event triggering the response from the Irobot Roomba was variable, sometimes it flashed and sometimes it did not respond at all. The flashing is due to the Irobot Roomba loosing connection to the charger and not ejection of the bin. The amount of packets and bytes are also variable most likely due to the inconsistent in the event triggering process, resulting in a high standard deviation. These calculations are presented in Table 5.24 and Table 5.25.

Table 5.22: Bin remove date and time overview for Oslo

Event	Date	Start time	End time
1	11.03.2023	17:30	17:31
2	11.03.2023	17:35	17:36
3	11.03.2023	17:40	17:41
4	11.03.2023	17:44	17:45
5	11.03.2023	17:47	17:48
6	11.03.2023	17:49	17:50
7	11.03.2023	17:51	17:52
8	11.03.2023	17:53	17:54
9	11.03.2023	17:55	17:56
10	11.03.2023	18:01	18:02

Table 5.23: Bin remove date and time overview for Drammen

Event	Date	Start time	End time
1	26.02	15:22	15:23
2	26.02	15:30	15:31
3	26.02	15:35	15:36
4	27.02	15:05	15:06
5	27.02	15:10	15:11
6	27.02	15:15	15:16
7	27.02	15:20	15:21
8	27.02	15:25	15:26
9	27.02	15:30	15:31
10	27.02	15:35	15:36

Table 5.24: Bin remove, overall statistics Oslo

Event	Packet number	Total bytes sent
Event 1	6	2,512
Event 2	15	5,765
Event 3	12	4,366
Event 4	15	7,869
Event 5	12	5,022
Event 6	17	8,426
Event 7	18	8,492
Event 8	12	5,022
Event 9	11	4,956
Event 10	12	5,022
Average	13	5,745.2
SD	3.43	1,937.64

Table 5.25: Bin remove, overall statistics Drammen

Event	Packet number	Total bytes sent
Event 1	23	8,610
Event 2	24	10,149
Event 3	9	4,407
Event 4	12	5,022
Event 5	8	4,333
Event 6	9	4,399
Event 7	17	6,595
Event 8	9	4,399
Event 9	15	7,869
Event 10	12	5,022
Average	13,8	6,080.5
SD	5.87	2,112.52

Packet length sequences were extracted with the python script in Appendix B, and the results are presented in Figure 5.20. The identified sequences are marked in yellow, packets with the length of *410 or 411* bytes were observed, these lengths are not observed in other events and is defined as the sequence signature for *Bin remove*. The detection algorithm created is presented in pseudo code in Figure 5.21 and is used to identify the signature in all event capture files. The evaluation result of testing is 100% success rate of signature detection.

Oslo
Event 1: D 208, D 288, D 315, S 176, S 186, D 408, S 176, S 1052, S 179, S 450, D 1110, S 179, S 186, **D 410**, S 179, S 450, D 1110, S 179, S 185, D 409
Event 2: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 3: S 179, S 160, D 346, S 172, S 233, D 551, S 179, S 450, D 1110, S 179, S 187, **D 411**, S 179, S 450, D 1110, S 179, S 450, D 1110, S 179, S 450
Event 4: S 179, S 492, D 1222, S 179, S 450, D 1110, S 179, S 186, **D 410**
Event 5: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 6: S 603, D 1220, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 7: S 179, S 490, D 1220, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 8: S 325, D 505, S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**, S 172, S 179, D 392
Event 9: S 179, S 490, D 1220, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 10: S 179, S 490, D 1220, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 186, **D 410**
Drammen
Event 1: S 179, S 448, D 1108, S 179, S 187, **D 411**
Event 2: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 3: S 172, S 293, D 861, S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 4: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 172, S 179, D 392, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 5: S 172, S 219, D 505, S 179, S 448, D 1108, S 179, S 187, **D 411**, S 172, S 234, D 552
Event 6: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 489, D 1219
Event 7: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 8: S 561, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 9: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 448, D 1108, S 179, S 186, **D 410**
Event 10: S 179, S 448, D 1108, S 179, S 187, **D 411**, S 179, S 448, D 1108, S 179, S 186, **D 410**

Figure 5.20: Bin remove packet length sequences

```
1     function bin_remove(br_confidence)
2         if packet.length == 410 or 411 in capture.file
3             br_confident = + 10
4         return br_confident
```

Figure 5.21: Pseudo code for detection algorithm for *Physical triggered cleaning* packet length sequence

5.9 Signature Comparison

This section will present the comparison of the different signatures used for event attribution in the sections above. First an evaluation test where all signatures are tested on all events to identify if the signatures are unique. This is followed by an analysis of the evaluation results.

Remove bin signature seems to be detected in all events except *Application start*, this signature is therefore removed from further analysis.

All the different cleaning events, *Automated cleaning, Scheduled cleaning, Application triggered cleaning* and *Physical triggered cleaning* had the same DNS responses present in the packet capturing. First a DNS response for FQDN *0550315.ingest.sentry.io*, followed by a response fors*3.amasoneaws.com*. These DNS packets can therefore not be used as signature for any of the specific cleaning events, but can increase confidence in a cleaning detection. However there are several similarities between the identified sequences and this will be discussed further.

Signature sequence can not differentiate between *Scheduled cleaning* or *Physical triggered cleaning*. However the traffic sequence on all the scheduled cleaning events is started within a minute before the scheduled cleaning's configured triggering time. It is safe to assume that scheduled cleaning is configured every whole hour or half hour. A normal user would not configure a cleaning at 15:17, but more

likely at 15:00. If an attacker is monitoring a smart environment during a longer time period, the two different cleanings can be separated based on reoccurring identification. A identified cleaning every Monday at 07:59 is most likely a scheduled cleaning, since humans are unable to provide the same level of consistency as IoT devices.

We can assume that the sequence used to identify *Application triggered cleaning* and *Application start* is due to the fact that the application is started, and not the actual triggering of the cleaning event. It is still possible to differentiate between these two events with the use of cleaning DNS signature. As all other cleaning events, it includes a DNS response for FQDN *0550315.ingest.sentry.io* and then *s3.amazoneaws.com*. If the *Application start* sequences and the cleaning DNS responses are detected, an application triggered cleaning is most likely executed. An element of uncertainty occurs if the user opens the application before or during another cleaning event, then this could create a false positive.

The only difference between *Automated cleaning* and *Application triggered cleaning* is the first package in the sequence of *Application start*. This packet has the length of 209 or 208 bytes and occurs every time the application is started. The identification of this is therefore a good attribute to differentiate for these events.

5.10 Wireless and Wired Traffic Capture Comparison

This section will compare the corresponding LAN and WLAN captures, and determine if the same method and identification can be applicable to identify events only based on wireless traffic as well.

WLAN and LAN traffic were captured for all triggered events, but due to the thesis' time constrains only identification of signatures and detection algorithms on LAN captures was conducted. The simulated WAN traffic had more available attributes, due to the Wi-Fi's encryption. In [41] they have already proposed a method to identify actions based on packet lengths. This research therefore focused on the WAN traffic, to be able to include DNS as an identifier. To evaluate if the same method and algorithms are applicable to WLAN traffic a comparison of two corresponding captures was done. Through analysis we identified that the added Wi-Fi header was 79 bytes and the base filter of 97 bytes in LAN captures was therefore converted to 176 bytes in WLAN. When this filter was applied, the same packet sequences as in the simulated WAN traffic were observed. These findings are presented in Figure 5.22. With the basefilter applied, it was observed that the WLAN capture included less packets than the corresponding LAN capture. This could be the result of retransmission, dual Wi-Fi channels, signal disruption or packet collision on the NIC in monitoring mode. When the NIC was configured in monitoring mode, it collected all Wi-Fi traffic in the area. This includes traffic from other SSIDs within wireless coverage. The original ISP modem was also broadcasting it's SSIDs, causing high Signal to Noise Ratio (SnR) for more than one SSID. Without control traffic between the AP and the NIC it could potentially lose traffic. Regardless of the packet loss it is still possible to identify similar patterns in WLAN

traffic as in LAN traffic.

Time	Source	Destination	Length
2023-02-26 17:00:09,519611	TP-Link_f6:01:74	iRobot_93:24:28	385
2023-02-26 17:00:09,519628	TP-Link_f6:01:74	iRobot_93:24:28	368
2023-02-26 17:00:10,036786	iRobot_93:24:28	TP-Link_f6:01:74	255
2023-02-26 17:00:10,182996	iRobot_93:24:28	TP-Link_f6:01:74	266

(a) WLAN

Time	Source	Destination	Length
2023-02-26 17:00:09,471142	52.204.51.243	192.168.0.91	316
2023-02-26 17:00:09,471765	52.204.51.243	192.168.0.91	289
2023-02-26 17:00:10,036901	192.168.0.91	52.204.51.243	176
2023-02-26 17:00:10,183079	192.168.0.91	52.204.51.243	187

(b) LAN

Figure 5.22: WLAN and LAN comparison

WLAN captures need more analysis before they can be implemented in the detection algorithm. An advantage of WLAN traffic is the availability of MAC addresses, an attacker can therefore easily identify the robot vacuum cleaner and filter traffic based on this information. Compared to LAN where an attacker will have to eavesdrop for up to 24 hours before the Irobot traffic can be identified.

Number of packets or bytes transmitted could also be used as an attribute to identify that there has been triggered a cleaning within the smart environment. This detection will be applicable for both LAN and WLAN eavesdropping.

Chapter 6

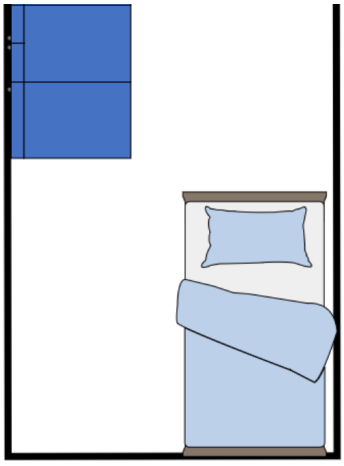
Evaluation

This chapter presents the evaluation of the identified signatures and algorithms in a live smart environment. First the evaluation method and live smart environments are presented, this is followed by the manual data processing conduced on the captured files. Then evaluation and results are presented with the detection success rate. During the evaluation the events *Scheduled cleaning* and *Physical triggered cleaning* are merged into one event due to identical signatures. The rest of the events *Automated cleaning, Application triggered cleaning, Application start* and *Remove bin* are included.

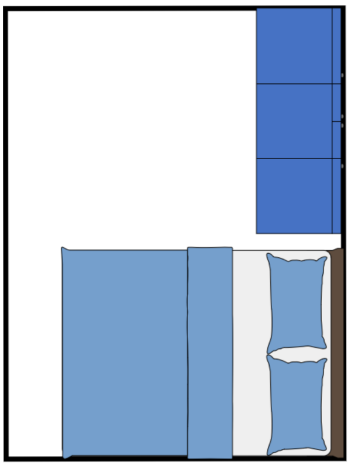
6.1 Evaluation Method

This section describes the method used to evaluate the thesis results. The evaluation process reuse network infrastructure, capturing process and data filtering as in the original research. Some new aspects are included in the evaluation to make the environments more representative for general smart environments. These new aspects are describes in detail further in this section.

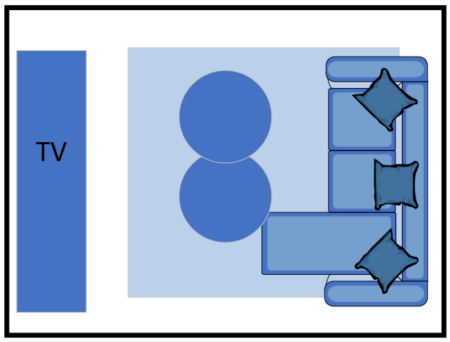
Event triggering was conducted in three different environments, now called **Guestroom, Bedroom** and **Living room**. The environment layouts are shown in Figure 6.1. The Irobot Roomba i7 was reverted to factory default for each of the evaluation environments, this mitigates the chance of any interference between the environments. User input such as robot, floor and room names were configured differently for all environments. A map discovery process was executed as part of the initial set-up phase.



(a) Guestroom



(b) Bedroom



(c) Living room

Figure 6.1: Evaluation environments

For each of the evaluation environments there were connected an additional IoT device to the same SSID as the Irobot Roomba. These devices generated traffic simulating a real-life smart environment. The additional IoT devices and associated evaluation environments are listed below.

- **Guestroom**: IPAD connected
- **Bedroom**: Laptop connected
- **Living room**: Smart phone connected

All events were triggered once in each of the evaluation environments. Each event had a 30 minute time window where the event was triggered and finished within. An example of an overall testing schedule is presented in the list below, where the first event is triggered between 08:00 and 08:30.

- First event: between 08:00 and 08:30
- Second event: between 08:30 and 09:00
- Third event: between 09:00 and 09:30
- Fourth event: between 09:30 and 10:00
- Fifth event: between 10:00 and 10:30

- Sixth event: between 11:30 and 11:00

To ensure that the order of events is not affecting the results it was decided by a python script, using the library *random*. Script logic is presented in Figure 6.2. This function was executed three times, ensuring that the order of events was random and had minimum influence cross events.

Figure 6.2: Pseudo code for event order randomize function

```
1          event_list = [scheduled_cleaning, Automated_cleaning,
               Application_triggered_cleaning, Application_start,
               Physical_triggered_cleaning, Bin_remove]
2          for three rounds do:
3               shuffle event_list
4               print shuffeled list
```

The basefilter created during baseline analysis in Chapter 5 is applied to all the capturing files. This excluded traffic not relevant to the actual event triggered. One additional processing step was included to be able to identify only the relevant corresponding Irobot cloud server. During the restart of the Irobot Roomba it had to request a DNS record for *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com* before establishing a TCP connection, this DNS response was extracted with a python script presented in Appendix C, an pseudo code is presented in Figure 6.3. This was further identified in Wireshark where the TCP hand-shake towards one of the responded IP addresses was found. The IP observed in the TCP handshake was added to the Wireshark filter. All traffic towards *50315.ingest.sentry.io* and *s3.amazoneaws.com* was therefore also excluded, but since only the DNS responses are used, all DNS traffic is also included in the Wireshark filter and is possible to identify.

Figure 6.3: Pseudo code for IP extraction from DNS response

```
1        Fuction find_dns_response(event_capture)
2          if a2uowfjvhio0fa.iot.us-east-1.amazonaws.com in event_capture
3              filter = Responded Ip-addresses and dns
```

The new basefilter was applied together with a time filter extracting one capturing file for each 30 minutes, resulting in one event per file. Then event detection algorithm were used on all the event files to evaluate the level of detection in the general smart environment.

6.2 Evaluation Results

This subsection presents the data processing and rule detection results. First the event order generated by the randomized script is presented. This is followed by

the DNS extraction and identification of the corresponding Irobot cloud server traffic. Evaluation results are presented and commented. Results from the randomize event order function, are presented in Table 6.1. The randomization of the event triggering order mitigates the influence cross events.

Table 6.1: Evaluation environments' event triggering order

Order	**Guestroom**	**Bedroom**	**Living room**
1	Automated clean	Application start	Remove bin
2	App triggered clean	Scheduled cleaning	App triggered clean
3	Scheduled cleaning	App triggered clean	Physical triggered
4	Physical triggered	Physical triggered	Application start
5	Application start	Automated clean	Scheduled cleaning
6	Bin remove	Bin remove	Automated cleaning

All capture files got processed by the DNS extraction algorithm identifying DNS responses for *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com* and extract information about the packet enabling identification in Wireshark. DNS responses for packet captures in the three environments are shown in Figure 6.4, several IP addresses are responded and the Irobot Roomba choose one of these to and establish a TCP connection to.

```
Answers
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.237.86.141
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.209.54.66
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.160.103.221
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.210.43.199
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 52.5.159.245
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 34.233.211.108
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.85.140.119
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 52.5.40.41
[Request In: 28562]
[Time: 0.011935043 seconds]
```

(a) Guestroom DNS extraction

```
Answers
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.210.235.55
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 44.208.195.70
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 23.23.90.38
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 3.93.155.217
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 34.232.18.249
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 3.234.13.159
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 18.233.12.97
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 18.211.44.186
[Request In: 86950]
[Time: 0.025594753 seconds]
```

(b) Bedroom DNS extraction

```
Answers
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 3.219.113.226
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 34.232.186.251
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 3.229.212.229
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 50.16.156.93
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 52.21.108.21
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.221.214.148
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 100.25.90.84
  a2uowfjvhio0fa.iot.us-east-1.amazonaws.com: type A, class IN, addr 54.84.55.93
[Request In: 11580]
[Time: 0.017775750 seconds]
```

(c) Living room DNS extraction

Figure 6.4: Evaluation environment DNS extraction

A TCP handshake towards one of the IPs in the DNS response was identified right after the DNS response in Wireshark. The identification of these are presented in Figure 6.5. This is a easy way for any attacker to identify corresponding traffic based on DNS requests. The used basefilters are listed below, and are identical for all environments except the IP address used to identify the Irobot cloud server.

- ((frame.time >= "Apr <day>, 2023 XX:00:00") && (frame.time <= "Apr <day>, 2023 XX:30:00")) AND
- (frame.len > 97) AND
- ((ip.addr == <DNS response IP>) or (dns && ip.dst == 192.168.0.56))
- DNS response IPs:
  - Guestroom: 54.237.86.141
  - Bedroom: 3.93.155.217
  - Living room: 3.219.113.226

Time	Source	Destination	Protocol	Length	Info
2023-04-11 08:36:46,…	84.208.20.110	192.168.0.56	DNS	230	Standard query re
2023-04-11 08:36:46,…	192.168.0.56	54.237.86.141	TCP	74	39280 → 443 [SYN]
2023-04-11 08:36:46,…	54.237.86.141	192.168.0.56	TCP	74	443 → 39280 [SYN,
2023-04-11 08:36:46,…	192.168.0.56	54.237.86.141	TCP	66	39280 → 443 [ACK]
2023-04-11 08:36:46,…	192.168.0.56	54.237.86.141	TLSv1.2	445	Client Hello

(a) Guestroom corresponding Irobot cloud detection

Time	Source	Destination	Protocol	Length	Info
2023-04-11 18:41:58,…	84.208.20.110	192.168.0.56	DNS	230	Standard que
2023-04-11 18:41:58,…	192.168.0.56	3.93.155.217	TCP	74	45291 → 443
2023-04-11 18:41:58,…	3.93.155.217	192.168.0.56	TCP	74	443 → 45291
2023-04-11 18:41:58,…	192.168.0.56	3.93.155.217	TCP	66	45291 → 443
2023-04-11 18:41:58,…	192.168.0.56	3.93.155.217	TLSv1.2	445	Client Hello

(b) Bedroom corresponding Irobot cloud detection

Time	Source	Destination	Protocol	Length	Info
2023-04-13 08:14:51,…	84.208.20.110	192.168.0.56	DNS	230	Standard que
2023-04-13 08:14:51,…	192.168.0.56	3.219.113.226	TCP	74	40739 → 443
2023-04-13 08:14:51,…	3.219.113.226	192.168.0.56	TCP	74	443 → 40739
2023-04-13 08:14:51,…	192.168.0.56	3.219.113.226	TCP	66	40739 → 443
2023-04-13 08:14:51,…	192.168.0.56	3.219.113.226	TLSv1.2	445	Client Hello

(c) Living room corresponding Irobot cloud detection

Figure 6.5: Evaluation environments corresponding Irobot cloud detection

All 18 filtered event capturing files were processed by the detection algorithm in Appendix A. No manual analysis was done to the files before hand and detection results are presented in Table 6.2. *Scheduled cleaning* and *Physical triggered cleaning* are merged to *Cleaning* but no further signature identification is done. The event is either *Scheduled cleaning* or *Physical triggered cleaning*.

Table 6.2: Evaluation results

Event	Auto clean	App clean	Cleaning	App start	Bin removed
True positive	100%	100%	100%	100%	0%
False positive	0%	0%	0%	0%	66%

The rules and detection algorithm were able to detect all events, except *Bin remove*, with 100% accuracy. Cleaning detection resulted in True positive for all cleaning events. *Bin remove* detection gave False negative for all the bin remove events. This might be because the ejection of the bin was executed without causing the Irobot Roomba to lose connection to the charging connectors. The detection algorithm also had False positive identification of *Bin remove* event for 66% of the events. The bin remove signature is therefore not able to detect bin removal.

Chapter 7

Discussion

This chapter discusses the challenges, processes and decisions throughout the research. The main topics is the thesis' answer to the research questions. Further, the limitation of Wi-Fi scope, how the events were triggered, why human analysis was selected and the reason to exclude the complexity of eavesdropping.

7.1 Our Answer to Research Question 1

Which private information can be gathered from a robot vacuum cleaner by carrying out a passive sniffing attack in a smart environment?

It is possible to identify the presence of an Irobot Roomba i7 vacuum cleaner inside a smart environment based on the WLAN or LAN capture itself. In WLAN, an attacker can eavesdrop and lookup all MAC addresses against open source registers. For LAN capture the presence of DNS requests to any Irobot owned domain will place a device behind the WAN address.

The signature detection algorithm proposed and evaluated in this project was able to identify and attribute different events conducted on the Irobot Roomba i7. Detection of *Automated cleaning* exposed information of when the user left the location, revealing private information. By observing the last five *Automated cleaning* events in **Oslo** is was possible to identify when the user left work, collecting and analysis of events triggering over a longer time period can expose user behaviour. *Application triggered cleaning* and *Application start* was also identified exposing user interaction with the Irobot Roomba.

If implementations to differentiate physical triggered and scheduled cleaning are added, the detection of physical triggered cleaning will reveal user activity inside the smart environment. Schedule cleaning on the other hand can give away user routines. We can assume that users usually configure scheduled cleaning when there is a high probability that there is no one at the location. This can reveal environment routines.

For *Application triggered cleaning* and *Application start* it is harder to identify the actual privacy exposure. The identification of these events can reveal more

information if observed during longer time periods. Then user patterns and behaviors can be exposed. Users could trigger cleaning every time they are leaving for work or the gym.

As mentioned for WLAN capturing, an attacker can extract private information as soon as the capturing is started because identification of the traffic is based in MAC addresses. For LAN, an attacker will have to eavesdrop WAN traffic for up to 24 hours before the DNS request to *a2uowfjvhio0fa.iot.us-east-1.amazonaws.com* is sent, and the corresponding Irobot cloud service is identified.

7.2 Our Answer to Research Question 2

How can information exposed by the eavesdropping be misused by an attacker?

Private information exposed in an attack can be utilized to identify user behavior and routines. This could potentially reveal habits of when the user is leaving the environment, and identify user presence with high confidence. This information can be used to target user environment during empty hours, or address the environment when the user is present. Such information can also be sold to other actors.

The identification of devices can be used to target attacks, based on IoT inventory. Spear phishing [58] will be more effective. They can also target attacks to exploit known or unknown vulnerabilities for the identified devices. This will increase the success rate of an attack. Exposed privacy information will threaten the security of any smart environment.

7.3 Our Answer to Research Question 3

Which security measures can be implemented to limit the exposed data and decrease the risk of misuse?

The most efficient way to defend against the detection algorithm, would be to implement traffic shaping. This could disrupt the predicted network traffic flows, pad existing packets or inject packets to break the patterns. This will be an effective way to defend against this in LAN or WLAN eavesdropping. A disadvantage with traffic shaping, will be higher latency and more data processing on local equipment. Implementation of traffic shaping could be on the robot vacuum cleaner itself, this would secure the communication regardless of the smart home environment infrastructure. Another approach is to implement this as a service within the smart environment, then the overall security of the smart environment would increase.

Irobot should implement random MAC addressing [59] for WLAN communication. This would allow the Irobot vacuum cleaner to use a random MAC address each time is connects to a new network, or change randomly to mislead an attacker.

As for WAN or LAN traffic, DNS is an easy identifier due to the use of Wi-Fi channel encryption. The initial session needs to be established based on a DNS request revealing the the IP address to the initial corresponding host. However, the daily change of corresponding host could be communicated through the secure connection hiding the change in cloud server host. This would make the detection of Irobot Roomba traffic harder to conduct by an attacker.

7.4 Collection of Wi-Fi Traffic

The selected TP-Link AP had Internet Group Management Protocol (IGMP) [60] default enabled. This protocol enables devices on a local network to subscribe to different multicast groups. Return traffic will then be addressed to the multicast group and not the device's MAC address. Due to this functionality, only the outbound traffic generated form the robot vacuum cleaner was captured during the standby event. Initial analysis of the standby traffic verified this when WLAN basefilter was applied. These findings are shown in Figure 7.1.

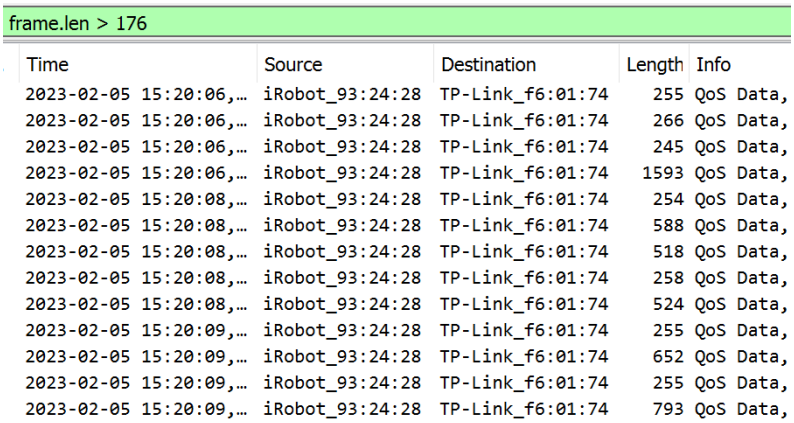


Figure 7.1: Wireshark WLAN capture, included basefilter and enabled IGMP

If IGMP enabled Wi-Fi would to be in the scope of this thesis, a process of filtering based on multicast group addresses should have been implemented. A 20 minutes *Application triggered cleaning* capturing was conducted in **Oslo** capturing only traffic including the MAC address of the AP. The capture included 9,342 packets, which is 340% more than LAN traffic average for the same event. By applying a Wireshark filter, excluding all traffic except Irobot and multicast MAC addressed, we identified the new IGMP traffic flow, shown in Figure 7.2.

(wlan.addr == 50:14:79:93:24:28 or wlan.addr == 01:00:5e:7f:ff:fa) && (frame.len > 176)

Time	Source	Destination	Length	Info
2023-02-07 20:35:12,…	iRobot_93:24:28	TP-Link_f6:01:74	258	QoS Da
2023-02-07 20:35:13,…	iRobot_93:24:28	TP-Link_f6:01:74	524	QoS Da
2023-02-07 20:35:20,…	iRobot_93:24:28	TP-Link_f6:01:74	265	QoS Da
2023-02-07 20:36:20,…	iRobot_93:24:28	TP-Link_f6:01:74	375	QoS Da
2023-02-07 20:36:21,…	TP-Link_f6:01:74	IPv4mcast_7f:ff:fa	578	Data,
2023-02-07 20:36:21,…	TP-Link_f6:01:74	IPv4mcast_7f:ff:fa	598	Data,
2023-02-07 20:36:21,…	iRobot_93:24:28	TP-Link_f6:01:74	297	QoS Da
2023-02-07 20:36:21,…	TP-Link_f6:01:74	IPv4mcast_7f:ff:fa	530	Data,
2023-02-07 20:36:21,…	TP-Link_f6:01:74	IPv4mcast_7f:ff:fa	539	Data,
2023-02-07 20:36:21,…	TP-Link_f6:01:74	IPv4mcast_7f:ff:fa	602	Data,

Figure 7.2: WLAN IGMP application triggered cleaning test in Oslo

This increases the complexity of the filtering mechanisms and identification of multicast traffic in **Oslo** and **Drammen**. The complexity occurs when more then one IoT device is connected to the same network. Devices can subscribe to the same or a new multicast group making the wireless environment more complex and hard to navigate in. IGMP is not used in all Wi-Fi networks [14] and it is therefor disabled on the AP.

7.5 Event Triggering

As shown in Chapter 5, several events were triggered during the same day and with limited time between them. The reason for this is the time constraint of this master project and the availability of smart environments. This increases the possibility of cross event influence, especially in the end of cleaning when the Irobot Roomba uploads cleaning data to the cloud service at *s3.amazonsaws.com*. To mitigate the influence, it was decided to only focus on the event initiation and not the end of cleaning reporting. Event triggering traffic is assumed to be the same regardless of previous events.

Real-life simulation of a smart environment is hard to recreate as there will not be triggered 5 cleaning events, within 2 hours. Event triggering timestamps in this project will appear unrealistic due to structured triggering in a short time period. Timestamps captured and identified in this research can therefore not be used in attributing events, but as mentioned in the analysis it could be a good attribute to include in the analysis and extraction of user private information.

7.6 Method of Analysis

Both human-based analysis and machine learning were discussed as the traffic analysis method in this project. Several other researches have used machine learning to extract information from network traffic based on various attributes. During

the literature review no documentation describing Irobot Roomba's communication pattern or protocols was found, Irobot did also not reply with information upon requests for this information. Either way, the data would have to be preprocessed by a human to identify which attributes that could be used in further analysis. The analysis method was therefore decided to be manual human-learn rule-based learning.

7.7 The Complexity of Eavesdropping

The level of complexity of conducting an eavesdropping attack for WLAN, LAN or WAN is not addressed in this thesis. This is a topic which should be addressed in separate research, due to the variety of devices and configurations in different smart environments. Smart environments used in this project only serve Internet access to exclude possible local configuration.

In wireless eavesdropping, an attacker would only need to be in wireless range of the targeted devices to collect corresponding network traffic. Eavesdropping devices can be placed in the vicinity of the smart environment or installed inside an environment, collected data can be stored locally or exported to a online service through Internet connection. There are pros and cons with the different approaches which is not addressed. For LAN and WAN eavesdropping the attacker would need physical access to the local network, or exploit remote access to network devices. These operations challenge both physical and technical security and are therefore out of scope for this thesis.

Chapter 8

Conclusions

The primary objective of this project is to evaluate if private information is exposed by conducting a passive network eavesdropping attack on a smart environment installed with a robot vacuum cleaner. In addition it addresses the potential risks and countermeasures to defend against such attacks.

In order to meet the project requirements, a robot vacuum cleaner survey was conducted to choose the most relevant vacuum cleaner available. The decision was based on popularity and open-source review sites where *Irobot Roomba i7* was identified as the most suitable. In order to determine if private information is exposed, two smart environments were configured to conduct testing and collection of data generated by the robot vacuum cleaner. A series of event objectives were defined based on the potential private information they could reveal if detected. Captures from the different events were analyzed to identify irrelevant traffic to be removed and signatures to be used in event detection.

To ensure validity of the identified signatures, three evaluation environments were configured presenting as live smart environments. All the events were triggered within these environments and used as input in a signature detection algorithm to determine if the signatures are consistent and possible to detect.

In conclusion, the identified signatures and detection algorithms were able to identify *Automated cleaning, Application triggered cleaning* and *Application start*, conducted on the Irobot Roomba only based on data from passive network eavesdropping attacks. The thesis also propose different defense mechanisms that would make the proposed signatures and detection to fail. As the time constrains and resources were limited, only one robot vacuum cleaner was used in the project, and the complexity of eavesdropping is not addressed.

8.1 Future Work

Future research should look into development of an automated tool, which can capture, process and analysis network traffic automatically based on the attributes used in this research. This would ease the process and enable researches to extract

similar results from a series of robot vacuum cleaners. In addition it could be valuable to compare different vendors and privacy differences cross these vendors. This would contribute to better security awareness and design for all users of robot vacuum cleaners.

Further analysis of more events and new robot vacuum cleaners would be interesting. Live smart environments can be designed with continuous packet capturing and event triggering based on normal user behaviour. Live detection of such events should also be developed which would decrease the detection time and amount of storage acquired.

Bibliography

[1] M. Dachyar, T. Y. M. Zagloel and L. R. Saragih, 'Knowledge growth and development: Internet of things (iot) research, 2006–2018,' *Heliyon*, vol. 5, no. 8, e02264, 2019.

[2] F. B. Insights, *Robotic vacuum cleaner market size, share | global report, 2028*, https://www.fortunebusinessinsights.com/industry-reports/robotic-vacuum-cleaners-market-100645, Accessed on May 10, 2023, 2021.

[3] H. F. Atlam and G. B. Wills, 'Iot security, privacy, safety and ethics,' in *Digital twin technologies and smart cities*, Springer, 2020, pp. 123–149.

[4] M. Pavelić, Z. Lončarić, M. Vuković and M. Kušek, 'Internet of things cyber security: Smart door lock system,' in *2018 international conference on smart systems and technologies (SST)*, IEEE, 2018, pp. 227–232.

[5] D. Mocrii, Y. Chen and P. Musilek, 'Iot-based smart homes: A review of system architecture, software, communications, privacy and security,' *Internet of Things*, vol. 1, pp. 81–98, 2018.

[6] H. Lin and N. W. Bergmann, 'Iot privacy and security challenges for smart home environments,' *Information*, vol. 7, no. 3, p. 44, 2016.

[7] G. Mantas, D. Lymberopoulos and N. Komninos, 'Security in smart home environment,' in *Wireless Technologies for Ambient Assisted Living and Healthcare: Systems and Applications*, IGI global, 2011, pp. 170–191.

[8] H. Assistant, *Home assistant*, 2023. [Online]. Available: https://www.home-assistant.io/.

[9] *Irobot® roomba® i7 medium grey*, 2022. [Online]. Available: https://www.irobot.no/roomba/i-serien/irobot-roomba-i7-medium-grey.

[10] F. Ullrich, J. Classen, J. Eger and M. Hollick, 'Vacuums in the cloud: Analyzing security in a hardened {iot} ecosystem,' in *13th USENIX Workshop on Offensive Technologies (WOOT 19)*, 2019.

[11] *Ecovacs intelligente automatiske rengjøringsroboter*, https://www.ecovacs.com/no, 2022.

[12] *S7, ta rengjøringen til et nytt nivå med sonisk mopping*, 2021. [Online]. Available: https://no.roborock.com/pages/roborock-s7.

[13] *Neatsvor x600 robotstøvsuger*, Aug. 2022. [Online]. Available: https://www.neatsvor.no/produkt/neatsvor-x600.

[14] 'Ieee standard for information technology—telecommunications and information exchange between systems local and metropolitan area networks — specific requirements - part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications,' *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pp. 1–3534, 2016. DOI: 10.1109/IEEESTD.2016.7786995.

[15] Wikipedia Contributors, *Network eavesdropping*, https://en.wikipedia.org/wiki/Network_eavesdropping, Wikipedia, Feb. 2023.

[16] I. S. Association *et al.*, *Guidelines for use of extended unique identifier (eui), organizationally unique identifier (oui), and company id (cid)*, 2018.

[17] DNS Checker, *Mac address lookup - mac lookup online*, https://dnschecker.org/mac-lookup.php, 2023.

[18] Fortinet, *What are eavesdropping attacks?* https://www.fortinet.com/resources/cyberglossary/eavesdropping, 2022.

[19] A. Orebaugh, G. Ramirez and J. Beale, *Wireshark and Ethereal network protocol analyzer toolkit*. Elsevier, 2006.

[20] Wireshark.org, *Tshark(1)*, 2023. [Online]. Available: https://www.wireshark.org/docs/man-pages/tshark.html.

[21] Tcpdump.org, *Tcpdump & libpcap*, https://www.tcpdump.org/, Tcpdump.org, 2023.

[22] greggigwg, *Microsoft message analyzer operating guide*, https://learn.microsoft.com/en-us/message-analyzer/microsoft-message-analyzer-operating-guide, Microsoft.com, Oct. 2022.

[23] A. Saeed, N. Dukkipati, V. Valancius, V. The Lam, C. Contavalli and A. Vahdat, 'Carousel: Scalable traffic shaping at end hosts,' in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, 2017, pp. 404–417.

[24] S. Xiong, A. D. Sarwate and N. B. Mandayam, 'Network traffic shaping for enhancing privacy in iot systems,' *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1162–1177, 2022.

[25] E. Rescorla, 'The transport layer security (tls) protocol version 1.3,' Tech. Rep., 2018.

[26] T. Ylonen and C. Lonvick, 'The secure shell (ssh) transport layer protocol,' Tech. Rep., 2006.

[27] R. Atkinson, *Rfc1825: Security architecture for the internet protocol*, 1995.

[28] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn and B. Palter, 'Layer two tunneling protocol" l2tp",' Tech. Rep., 1999.

[29] J.-C. Zúñiga, C. J. Bernardos and A. Andersdotter, 'Randomized and Changing MAC Address,' Internet Engineering Task Force, Internet-Draft draft-ietf-madinas-mac-address-randomization-06, Mar. 2023, Work in Progress, 16 pp. [Online]. Available: `https://datatracker.ietf.org/doc/draft-ietf-madinas-mac-address-randomization/06/`.

[30] D. K. Alferidah and N. Jhanjhi, 'A review on security and privacy issues and challenges in internet of things,' *International Journal of Computer Science and Network Security IJCSNS*, vol. 20, no. 4, pp. 263–286, 2020.

[31] D. Swessi and H. Idoudi, 'A survey on internet-of-things security: Threats and emerging countermeasures,' *Wireless Personal Communications*, vol. 124, no. 2, pp. 1557–1592, 2022.

[32] T. Gu, Z. Fang, A. Abhishek and P. Mohapatra, 'Iotspy: Uncovering human privacy leakage in iot networks via mining wireless context,' in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, IEEE, 2020, pp. 1–7.

[33] T. Dahlberg Sundström and J. Nilsson, *Ethical hacking of a premium robot vacuum: Penetration testing of the roborock s7 robot vacuum cleaner*, 2022.

[34] S. Sami, Y. Dai, S. R. X. Tan, N. Roy and J. Han, 'Spying with your robot vacuum cleaner: Eavesdropping via lidar sensors,' in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, 2020, pp. 354–367.

[35] T. Nguyen, 'A deep look into privacy and security of vacuum robot,'

[36] M. E. Kaminski, M. Rueben, W. D. Smart and C. M. Grimm, 'Averting robot eyes,' *Md. L. Rev.*, vol. 76, p. 983, 2016.

[37] C. Torgilsman and E. Bröndum, *Ethical hacking of a robot vacuum cleaner*, 2020.

[38] M. Alyami, I. Alharbi, C. Zou, Y. Solihin and K. Ackerman, 'Wifi-based iot devices profiling attack based on eavesdropping of encrypted wifi traffic,' in *2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC)*, IEEE, 2022, pp. 385–392.

[39] A. Acar, H. Fereidooni, T. Abera, A. K. Sikder, M. Miettinen, H. Aksu, M. Conti, A.-R. Sadeghi and S. Uluagac, 'Peek-a-boo: I see your smart home activities, even encrypted!' In *Proceedings of the 13th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2020, pp. 207–218.

[40] S. Xiong, A. D. Sarwate and N. B. Mandayam, 'Network traffic shaping for enhancing privacy in iot systems,' *IEEE/ACM Transactions on Networking*, vol. 30, no. 3, pp. 1162–1177, 2022.

[41] R. Trimananda, J. Varmarken, A. Markopoulou and B. Demsky, 'Packet-level signatures for smart home devices,' in *Network and Distributed Systems Security (NDSS) Symposium*, vol. 2020, 2020.

[42] I. Heđi, I. Špeh and A. Šarabok, 'Iot network protocols comparison for the purpose of iot constrained networks,' in *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, IEEE, 2017, pp. 501–505.

[43] Y. Li, D. Li, W. Cui and R. Zhang, 'Research based on osi model,' in *2011 IEEE 3rd International Conference on Communication Software and Networks*, IEEE, 2011, pp. 554–557.

[44] M. Contigiani, R. Pollini, M. Sturari, A. Mancini and E. Frontoni, 'Iot architecture for the processing of data collected by a central vacuum cleaner,' in *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, American Society of Mechanical Engineers, vol. 58233, 2017, V009T07A044.

[45] C. Lawrence and K. Mortram, Nov. 2022. [Online]. Available: https://www.tomsguide.com/us/best-robot-vacuums,review-2000.html.

[46] A. Moscaritolo, *The best robot vacuums for 2023*, Dec. 2022. [Online]. Available: https://uk.pcmag.com/vacuums/74630/the-best-robot-vacuums.

[47] Aug. 2021. [Online]. Available: https://www.hjemoghage.no/robotstovsuger-test/?tid=robotstovsuger-ad&gclid=Cj0KCQjwguGYBhDRARIsAHgRm4-1N_SChYVvvL2gNk6Zd7sJ1-CKhCW-VOfcdOJIsheRGWfOQqWdPv4aAgr4EALw_wcB.

[48] 2021. [Online]. Available: https://play.google.com/store/games?pli=1.

[49] 2023. [Online]. Available: https://no.roborock.com/pages/robot-vacuum-cleaner-compare.

[50] 2018. [Online]. Available: https://smartrobotreviews.com/g/rv/irobot-roomba-i7-7150/.

[51] Jan. 2017. [Online]. Available: https://www.bestcordlessvacuumguide.com/roomba-comparison/.

[52] Elkjop.no, *Irobot roomba i7 robotstøvsuger i715040*, 2023. [Online]. Available: https://www.elkjop.no/product/hjem-rengjoring-og-kjokkenutstyr/stovsugere-og-rengjoring/robotstovsuger/irobot-roomba-i7-robotstovsuger-i715040/273857.

[53] Kali, Jul. 2022. [Online]. Available: https://www.kali.org/get-kali/#kali-arm.

[54] 2016. [Online]. Available: https://www.wireshark.org/download.html.

[55] Winscp.net, *Winscp :: Official site :: Download*, 2023. [Online]. Available: https://winscp.net/eng/download.php.

[56] eliaswirth, *Ifttt*, 2023. [Online]. Available: https://ifttt.com/applets/hJMKghVa-if-someone-calls-start-maps-and-show-his-location.

[57] Wikipedia Contributors, *Irobot*, May 2023. [Online]. Available: `https://en.wikipedia.org/wiki/IRobot`.

[58] Fortinet, *What is spear phishing? definition, risks and more*, `https://www.fortinet.com/resources/cyberglossary/spear-phishing`, 2022.

[59] C. Bernardos and A. Mourad, 'Rfc 8948 structured local address plan (slap) quadrant selection option for dhcpv6,' 2020.

[60] W. Fenner, 'Internet group management protocol, version 2,' Tech. Rep., 1997.

Appendices

Appendix A

Event Detection Algorithm

```
# Python code, Signeture detection
import pyshark
from pyshark.packet import consts
from pyshark.packet.common import Pickleable
import matplotlib
import numpy
import os


def cleaning_conf(c_conf):

    dns_o_i_s_i = False
    dns_aws = False
    clean_end = None

    #Loop all packets in capture
    for packet in cap:

        #find DNS response lager then 100 bytes
        if packet.highest_layer == 'DNS' and packet.ip.dst_host == wan_addr
            :
            #print('dns')
            #find cleaning dns response
            if packet.dns.resp_name == 'o550315.ingest.sentry.io':
                dns_o_i_s_i = True
                #print('dns1')
            if packet.dns.resp_name == 's3.amazonaws.com':
                dns_aws = True
                #print('dns2')

        if dns_o_i_s_i and dns_aws == True:
            c_conf =+ 10
            return c_conf

    return c_conf
```

```
35
36 def get_type_of_cleaning():
37     indication_sc = 0
38     indication_tc = 0
39     cleaning_type = None
40     clean_start = None
41
42     #Loop through all packets in capture
43     for packet in cap:
44         #Add indication for sc
45         if  packet.ip.dst_host == '192.168.0.56' and packet.length == ('
               1101' or '1107'):
46             indication_sc = indication_sc + 1
47
48         #Add indication for tc
49         if packet.ip.dst_host == '192.168.0.56' and packet.length == ('1105
               ' or '1106' or '1099'):
50             indication_tc = indication_tc + 1
51
52     if indication_sc > indication_tc:
53         cleaning_type = 'scheduled cleaning'
54     if indication_sc < indication_tc:
55         cleaning_type = 'triggered cleaning'
56
57
58     return cleaning_type
59
60 def open_application_conf(oa_conf):
61     #open application True/False
62     open_application = False
63     ao_time = 'Opening time not identified'
64     oa_initiator = [209, 289, 316]
65     oa_initiator_1 = [209, 315, 289]
66     for packet in cap:
67         if packet.length == '209':
68             ao_time = packet.sniff_time
69             #print(ao_time)
70
71         if 209 in packet_length[0:20]:
72             oa_index = packet_length_dst.index(209)
73             oa_start_compare = packet_length_dst[oa_index:oa_index + len(
                   oa_initiator)]
74             open_application = numpy.allclose(oa_initiator,
                   oa_start_compare, atol= 3)
75             open_application_1 = numpy.allclose(oa_initiator_1,
                   oa_start_compare, atol= 3)
76             if (open_application or open_application_1) == True:
77                 oa_conf =+ 10
78                 return oa_conf
79
```

```
80          if 208 in packet_length[0:20]:
81              oa_index = packet_length_dst.index(208)
82              oa_start_compare = packet_length_dst[oa_index:oa_index + len(
                    oa_initiator)]
83              open_application = numpy.allclose(oa_initiator,
                    oa_start_compare, atol= 3)
84              open_application_1 = numpy.allclose(oa_initiator_1,
                    oa_start_compare, atol= 3)
85              if (open_application or open_application_1) == True:
86                  oa_conf =+ 10
87                  return oa_conf
88
89          if 207 in packet_length[0:20]:
90              oa_index = packet_length_dst.index(207)
91              oa_start_compare = packet_length_dst[oa_index:oa_index + len(
                    oa_initiator)]
92              open_application = numpy.allclose(oa_initiator,
                    oa_start_compare, atol= 3)
93              open_application_1 = numpy.allclose(oa_initiator_1,
                    oa_start_compare, atol= 3)
94              if (open_application or open_application_1) == True:
95                  oa_conf =+ 10
96                  return oa_conf
97
98
99
100     return oa_conf
101
102 def find_packet_seq(tc_confident):
103     triggered_cleaning = [503, 175, 509, 1106, 179, 439, 1099, 179, 445,
            1105, 176]
104     scheduled_cleaning = [179, 447, 1107, 176, 476, 176, 617, 179, 253,
            626, 179, 447, 1107]
105     oopen_application = [209, 289, 316, 176, 187, 409]
106     bin_entered = [179, 186, 410]
107     auto_clean = [316, 289, 176, 187, 409]
108
109     if 503 in packet_length:
110         tc_index = packet_length.index(503)
111         tc_compare = packet_length[tc_index:tc_index + len(
                triggered_cleaning) ]
112         tc_test = numpy.allclose(triggered_cleaning, tc_compare, atol= 1)
113
114         if tc_test == True:
115             tc_confident = tc_confident + 10
116
117     return tc_confident
118
119 def auto_clean_conf(ac_conf, packet_length):
120     auto_clean = [316, 289, 176, 187, 409]
```

```
121     auto_clean_1 = [289, 316, 176, 187, 409]
122
123     if 207 in packet_length[0:30]:
124         return ac_conf
125     if 208 in packet_length[0:30]:
126         return ac_conf
127     if 209 in packet_length[0:30]:
128         return ac_conf
129
130
131     if 316 in packet_length:
132         ac_index = packet_length.index(316)
133         ac_start_compare = packet_length[ac_index:ac_index + len(auto_clean
                )]
134         ac_indication = numpy.allclose(auto_clean, ac_start_compare, atol=
                1)
135         if ac_indication == True:
136             ac_conf =+ 10
137             return ac_conf
138
139     if 315 in packet_length:
140         ac_index = packet_length.index(315)
141         ac_start_compare = packet_length[ac_index:ac_index + len(auto_clean
                )]
142         ac_indication = numpy.allclose(auto_clean, ac_start_compare, atol=
                1)
143         if ac_indication == True:
144             ac_conf =+ 11
145             return ac_conf
146
147     if 288 in packet_length:
148         ac_index = packet_length.index(288)
149         ac_start_compare = packet_length[ac_index:ac_index + len(
                auto_clean_1)]
150         ac_indication = numpy.allclose(auto_clean_1, ac_start_compare, atol
                = 1)
151         if ac_indication == True:
152             ac_conf =+ 12
153             return ac_conf
154
155     if 289 in packet_length:
156         ac_index = packet_length.index(289)
157         ac_start_compare = packet_length[ac_index:ac_index + len(
                auto_clean_1)]
158         ac_indication = numpy.allclose(auto_clean_1, ac_start_compare, atol
                = 1)
159         if ac_indication == True:
160             ac_conf =+ 13
161             return ac_conf
162
```

```
163     return ac_conf
164
165 def trigger_clean_conf(tc_conf, oa_conf, c_conf):
166
167     #If there is a cleaning and tha application is opened, we can say that
            it is likely that is has been a triggered clean
168     if oa_conf and c_conf == 10:
169         tc_conf =+ 10
170
171
172     return tc_conf
173
174 def physical_cleaning_conf(pc_conf, packet_length_src):
175     physical_clean = [176, 173, 179, 443, 177]
176     physical_clean_1 = [176, 443, 179, 443, 177]
177     count = 0
178     #The value can be 172, 176, 175 and 179
179     for packet in packet_length_src:
180
181         if 172 <= packet <= 179:
182           pc_compare = packet_length_src[count:count + len(physical_clean)]
183           pc_indicator = numpy.allclose(physical_clean, pc_compare, atol=
                  15)
184           pc_indicator_1 = numpy.allclose(physical_clean_1, pc_compare,
                  atol= 5)
185
186           if pc_indicator == True:
187               pc_conf =+ 10
188               return pc_conf
189
190           if pc_indicator_1 == True:
191               pc_conf =+ 11
192               return pc_conf
193
194         count = count + 1
195
196
197     return pc_conf
198
199 def remove_bin(br_conf):
200
201     removebin_value = '410'
202     for packet in cap:
203         if packet.length ==  '410':
204             return 10
205         if packet.length == "411":
206             return 10
207     return 0
208
209 def dns_cleaning(c_conf, dns_names):
```

```
210
211     if 'o550315.ingest.sentry.io' and 's3.amazonaws.com' in dns_names:
212         return 10
213     else:
214         return 0
215
216 # MAIN Function is starts
217
218
219 folder = [r'C:\Users\benja\Documents\Mater test results\LAN\Live\Env3_dns']
220 files = os.listdir(folder[0])
221 #print(folder[0] + '\\' + files[0])
222 wan_addr = '192.168.0.56'
223
224 for file in files:
225     ac_conf = 0
226     oa_conf = 0
227     sc_conf = 0
228     tc_conf = 0
229     rb_conf = 0
230     pc_conf = 0
231     c_conf = 0
232
233
234     file_path = str(folder[0] + '\\' + file)
235     cap = pyshark.FileCapture(file_path)
236     cap.load_packets()
237     packet_length = [2000]
238     packet_length_dst = [2000]
239     packet_length_src = [2000]
240     packet_time = []
241     dns_names = []
242
243
244
245     for packet in cap:
246         if packet.highest_layer != 'DNS':
247
248             if int(packet.length) != packet_length[-1]:
249                 packet_length.append(int(packet.length))
250                 #packet_time.append(int(packet.sniff_time))
251
252                 if packet.ip.dst_host == wan_addr:
253                     packet_length_dst.append(int(packet.length))
254
255                 if packet.ip.src_host == wan_addr:
256                     packet_length_src.append(int(packet.length))
257         else:
258             dns_names.append(packet.dns.resp_name)
259
```

```
260
261
262
263     print(file)
264     #print(packet_length_dst)
265     ac_conf =+ auto_clean_conf(ac_conf, packet_length)
266     oa_conf =+ open_application_conf(oa_conf)
267     #c_conf =+ cleaning_conf(c_conf)
268     c_conf =+ dns_cleaning(c_conf, dns_names)
269     tc_conf =+ trigger_clean_conf(tc_conf, oa_conf, c_conf)
270     #pc_conf =+ physical_cleaning_conf(pc_conf, packet_length_src)
271     rb_conf =+ remove_bin(rb_conf)
272     #print(packet_length_dst[0:20])
273     #print(dns_names)
274
275     #print("Auto clean  " + str(ac_conf) + '  Open application  '+ str(
            oa_conf) + '   Cleaning  ' + str(c_conf) + '  triggered  ' + str(
            tc_conf) + '  Pysical  ' + str(pc_conf))
276     if ac_conf > 0:
277         print('Auto Clean')
278     if tc_conf > 0:
279         print('Triggered Cleaning')
280     if c_conf > 0 and ac_conf == 0 and tc_conf == 0:
281         print('Scheduled or Physical cleaning')
282     if rb_conf > 0:
283         print('Bin is removed')
284     if oa_conf > 0 and tc_conf == 0:
285         print('application is opened')
```

Appendix B

Packet Lengths Extraction Algorithm

```
     import pyshark
from pyshark.packet import consts
from pyshark.packet.common import Pickleable
import matplotlib
import numpy
import os

folder = [r'filepath']
files = os.listdir(folder[0])
#print(folder[0] + '\\' + files[0])
sum_lenght=  []
sum_nr = []
print(files)
for file in files:

    file_path = str(folder[0] + '\\' + file)
    #print(file)
    cap = pyshark.FileCapture(file_path)
    cap.load_packets()
    packet_lengths = []
    packet_count = 0



    for packet in cap:

       # if packet.ip.dst_host == '192.168.0.56':
       #     packet_lengths.append('D ' + packet.length)
       # else:
       #     packet_lengths.append('S ' + packet.length)
        packet_lengths.append(int(packet.length))
        packet_count = packet_count + 1
```

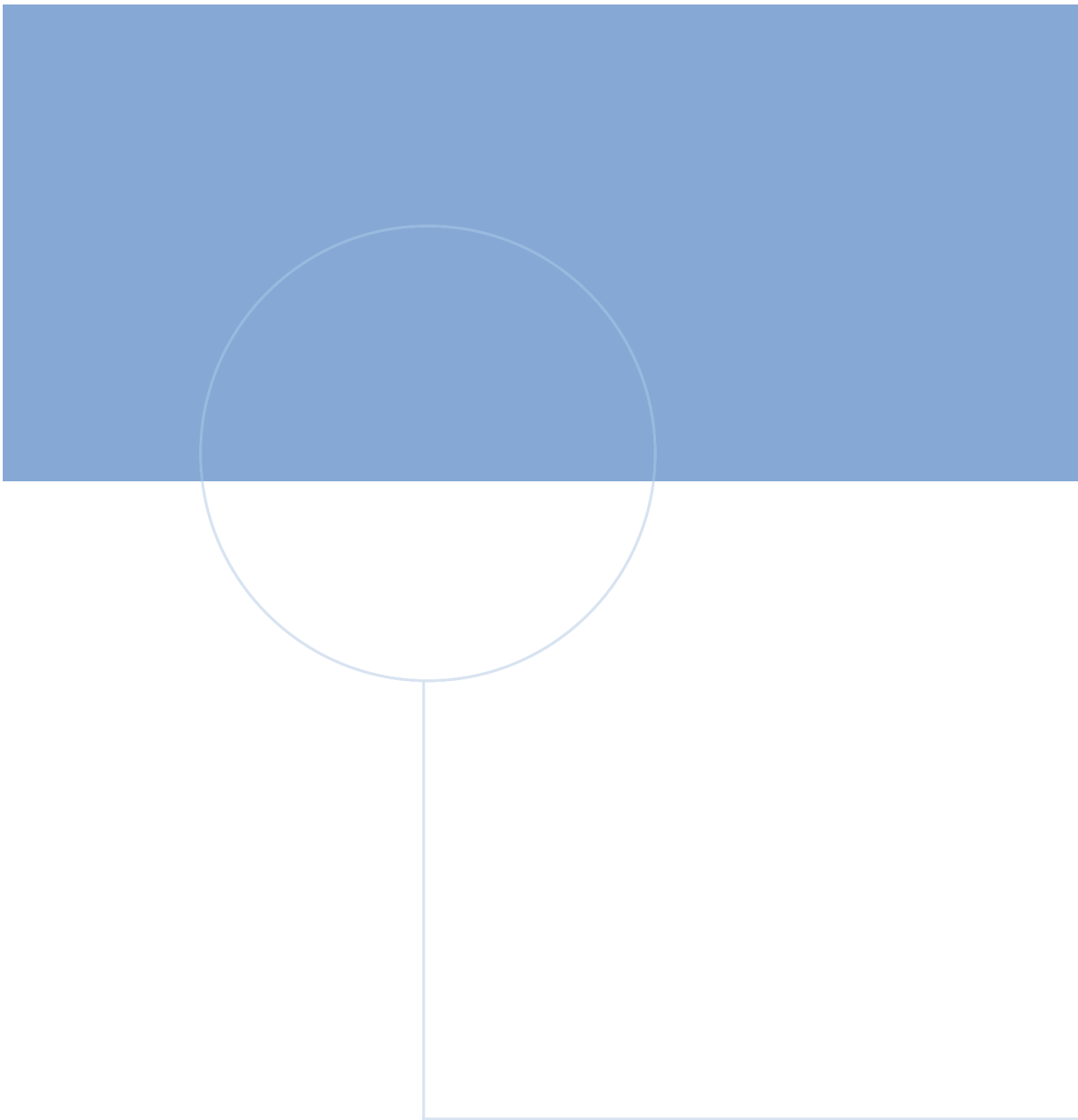
```
33
34
35      print(file)
36      print(packet_count)
37      print(sum(packet_lengths))
38      sum_lenght.append(sum(packet_lengths))
39      sum_nr.append(packet_count)
40      #if len(packet_lengths) >= 20:
41      #    print(packet_lengths[0:20])
42      #else:
43      #    print(packet_lengths[0:len(packet_lengths)])
44
45
46  print(sum(sum_lenght)/10)
47  print(sum(sum_nr)/10)
48  print(sum_lenght)
49  print(sum_nr)
```

Appendix C

DNS Extraction Algorithm

```
import pyshark
import os


def dns_ip_find ():


    #Loop all packets in capture
    for packet in cap:

        #find DNS response lager then 100 bytes
        if packet.highest_layer == 'DNS' and packet.ip.dst_host == '
            192.168.0.56':
            #print('dns')
            #find cleaning dns response
            if packet.dns.resp_name == 'a2uowfjvhio0fa.iot.us-east-1.
                amazonaws.com':
                print(packet.dns.pretty_print())
                print(packet.sniff_time)
                print(packet.number)
                print('dns1')

            if packet.dns.resp_name == 'unauth1.prod.iot.irobotapi.com':
                print(packet.dns.pretty_print())
                print(packet.sniff_time)
                print('dns1')

            if packet.dns.resp_name == 'disc-prod.iot.irobotapi.com':
                print(packet.dns.pretty_print())
                print(packet.sniff_time)
                print('dns1')


            if packet.dns.resp_name == '0.irobot.pool.ntp.org':
                print(packet.dns.pretty_print())
```

```
                print(packet.sniff_time)
                print('dns1')
            if packet.dns.resp_name == 'o550315.ingest.sentry.io':
                print(packet.number)




    return None


folder = [r'C:\Users\benja\Documents\Mater test results\LAN\Live\Env1_dns']
files = os.listdir(folder[0])

for file in files:

    print(file)
    file_path = str(folder[0] + '\\' + file)
    cap = pyshark.FileCapture(file_path, display_filter='dns')
    cap.load_packets()

    dns_ip_find()
```



NTNU

Norwegian University of
Science and Technology