UNIVERSIDADE D
COIMBRA

António Augusto Margalho Craveiro

UMA PLATAFORMA PARA MELHOR SEGURANÇA EM REDES DOMÉSTICAS
A FRAMEWORK FOR IMPROVED HOME NETWORK SECURITY

Dissertação no âmbito do Mestrado em Segurança Informática orientada pelo Professor Doutor Paulo Alexandre Ferreira Simões e Professor Doutor Tiago José dos Santos Martins da Cruz e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Julho de 2020

This page is intentionally left blank.

Acknowledgements

I would like to express my gratitude to Paulo Simões which, being my adviser for this thesis, had guide me to this challenging world of scientific research, another new journey in my life. Not only I had the chance, under his guidance, to learn how to work in the right way, but he has also allowing me to participate in international meetings, a remarkable experience. As for the thesis details, he was always available for counseling and advises, allowing me to improve my skills and produce better results. Thank you very much Paulo.

A special thanks to Tiago Cruz my co-adviser, for your contagious enthusiasm about all this work, and for your timely advises. It had helped me a lot.

Also, thanks to the folks at CISUC, who received me as a family, and make me feel comfortable and been part of the group. Special thanks to Luis Rosa for helping me to untie the wireshark knot, a fundamental tool in this thesis.

I want to thanks my family for their help and support. It made things a lot easier. Really!

Finally, to all my friends, who gave me a word of support which also helped making this journey more rewarding, thank you from the heart.

This page is intentionally left blank.

Resumo

Nesta tese propomo-nos a implementar uma infraestrutura que vise melhorar a segurança de redes domésticas. As redes domésticas têm sofrido, nos últimos anos, um surpreendente aumento em dimensão, variedade e complexidade. Todas estas mudanças levantam novas questões acerca da sua segurança, entre outras razões, pela miríade de novos dispositivos colocados na rede (p. ex. dispositivos IoT, televisões inteligentes, dispositivos inteligentes, câmaras de vigilância) que geram uma enorme quantidade de tráfego, que não é diretamente controlado pelo proprietário da rede, e é ainda pouco conhecido. Diversas situações de tráfego malicioso, gerado por estes dispositivos, têm sido recentemente identificadas. As causas prováveis deverão ser intenções maliciosas do fabricante do dispositivo, ou porque foram comprometidos face a uma fraca segurança. Estas ameaças ainda não foram devidamente abordadas provavelmente devido à ausência de informação detalhada sobre o comportamento do tráfego de rede gerado por cada um desses dispositivos.

Para resolver este problema, o primeiro passo deverá ser efetuar uma extensa análise do fluxo de tráfego, seguido pela definição de formas de vigiar esse fluxo e, finalmente, controlá-lo. Esta não é uma tarefa fácil, porque não é possível obter os detalhes do interior dos pacotes de rede, uma vez que na sua maioria estão encriptados. Como tal, ficamos reduzidos à análise estatística dos fluxos de pacotes, procurando detalhes como o tamanho, taxa de transmissão, frequência de pacotes, e percentagem de pacotes em função do tamanho. Pode não parecer muito, mas é fundamental para conseguir uma imagem clara do tráfego de rede numa rede doméstica típica, um primeiro passo se queremos proteger a nossa rede doméstica.

O principal objetivo desta tese é explorar a possibilidade de analisar a rede doméstica, e identificar o tráfego entre as aplicações da rede local e a internet. De seguida, definir as regras para monitorizar e controlar o fluxo utilizando um dispositivo apropriado. E, finalmente, integrar tudo numa rede de dispositivos locais controlados, comunicando entre si. Ao progredir para além do limite atual, entrando à porta do cliente, a solução proposta pretende fornecer monitorização de segurança compreensiva e contextualizada, com base no utilizador, e embutido no gateway residencial ou, eventualmente, delegando num equipamento especializado colocado na rede local.

Palavras-chave

Segurança de Redes Domésticas, Segurança Distribuída, Casas Inteligentes, Dispositivos Domésticos, Deteção de Intrusões.

This page is intentionally left blank.

Abstract

In this thesis we propose a framework for improving the security of home networks. Home networks have seen, in the last years, an amazing increase in size, variety, and complexity. All these changes raise new concerns about their security, among other reasons because the myriad of new devices placed in the networks (e.g. IoT devices, smart TVs, smart appliances, surveillance cameras) generate a lot of network traffic which is not directly controlled by the network owner and is still poorly known. Several situations of malicious traffic being generated by such devices have been recently identified. The probable reasons are either due to malicious intentions of device manufacturers or simply because they have been compromised due to poor security. These threats are not being fully addressed probably because of the absence of detailed information about behavior and network traffic generated by each of these devices.

To address this problem, the first step should be to perform a thoughtfully analysis of traffic flow, followed by finding the means to monitor that flow and, finally, to control it. This is not an easy task, because is not possible to get the inner details of the network packets, since most of then are encrypted. So, we are left with statistical analysis of packet flows, looking for aspects such as size, burst rate, packet frequency, and packet size percentage. It may not look much, but it is fundamental to get a clear picture of network traffic of a typical home network, a first step if we want to protect our home networks.

The main goal of this thesis is to explore the possibility of analyzing the home network and profiling the traffic between home appliances and the internet. Then, to define the rules to monitor and control the flow using a properly shaped device. And finally, to integrate all this in a network of home control devices communicating with each other. By moving beyond the last mile and into the customers' doorsteps, the proposed solution intends to provide comprehensive and contextual security monitoring on a per-user basis, embedded at the residential gateway or, eventually, resorting to a specialized appliance deployed in the home LAN.

Keywords

Home Network Security, Distributed Security, Smart Homes, Home Appliances, Intrusion Detection.

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

AP Access Point. 11, 15

BD Bluray Disk. 11, 40

DASH Dynamic Adaptive Streaming Over HTTP. 5

DHCP Dynamic Host Configuration Protocol. 11, 15, 55

DLNA Digital Living Network Alliance. 22, 31, 56

DNS Domain Name System. 5, 7, 11, 15, 21, 22, 31, 40, 49, 55, 56, 68

DVD Digital Video Disk. 11

GUI Graphical User Interface. 68

HTTP Hypertext Transfer Protocol. 5, 21, 22, 49, 56

ICS Industrial Control Systems. 7

IDE Integrated Development Environment. 64

IDS Intrusion Detection System. xviii, 1, 4–8

IoT Internet of Things. 1, 6–8, 22

IP Internet Protocol. 17, 49, 57

ISP Internet Service Provider. 5, 7, 9, 10, 12, 49, 51, 55

JSON JavaScrip Object Notation. 40

LAN Local Area Network. vii, 11

MANET Mobile Ad-Hoc Network. 6

OCSP Online Certificate Status Protocol. 40

OS Operating System. 12, 13, 20, 49, 64

SBC Single Board Computer. 13

SMP Security Management Platform. 10

SSH Secure Socket Shell. 4

TCP Transmission Control Protocol. 4, 17, 19

UDP User Datagram Protocol. 17, 19, 49, 55

USB Universal Serial Bus. 14

VoIP Voice Over Internet Protocol. 6

WSN Wireless Sensor Network. 6

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

In the last years the number of devices connected to the home networks has drastically grown, with the advent of products like smart TVs, smart speakers, smart meters, alarm systems, climate-control systems, gaming consoles, medical devices, smart-home devices and a plethora of other smart home Internet of Things (IoT) boxes. The proliferation of such devices, in an ecosystem which is often poorly managed but holds very sensitive personal information, raises considerable concerns from the security point-of-view. Recent incidents have shown that these devices cannot be trusted, since their manufacturers often abuse basic user privacy rights to begin with. In addition, even when manufacturers do behave correctly, the devices are designed with such poor security concerns that they easily become the target of malicious attacks from third-parties, providing a valuable entry point to the wealth of personal information available in the home network and becoming soldiers of large botnets used for high-profile internet attacks. Those risks are amplified by the fact that home networks are recklessly managed by the average owner, with little or no security concerns at all. Home networks also lack organization without internal segmentation or restrictions on device-to-device communications, and even without proper security monitoring mechanisms at internal level. Moreover, ownership and management of these devices is often fragmented between different parties (e.g. homeowner, other home users, energy utilities, local service providers such as telecommunications operators, and cloud service providers such as Google).

1.1 Motivation

The aforementioned scenario poses us with a new perspective of home network security. We no longer can think of this kind of security in a "protect from the wild outside world" way, but rather paying closer attention to what kind of traffic is generated inside the home network by our home appliances. To be able to deal with this kind of scenario, a completely new perspective is needed beyond traditional firewall and Intrusion Detection System (IDS) approaches. This perspective, although not neglecting the referred traditional functions of network security devices, must also include tools for defending the network from unwanted traffic originated from the inside. This new kind of threats is explained in [9], but has not been given sufficient attention so far. Nevertheless, the danger exists and is not too soon to start thinking in ways to protect our home networks from these new threats. Following this new security perspective, a possible approach is to define a new framework to manage all these new kinds of threats. This framework would integrate the existing IDS functions with the monitoring and control of outbound traffic. Taking into account the rapid evolution

of these new threats, and the enormous variety of smart home appliances, this framework needs to be cooperative. The idea is to have several home control devices linked together, through a security management platform. These devices are able to identify new threats through the traffic analysis, according to the smart device brand and model. Then, they send the information to the management platform for further analysis. After consolidation, the information is made available for all devices. This way, when a smart device of a known brand and model is added to some home network, if there is information about it in the management platform, that network security device can upload it for its won use. If there is no information the system can start monitoring the device traffic looking for patterns.

1.2 Objectives

This project intends to be a step towards the full implementation of a framework regarding this new home security paradigm. For now, the main concern is the analysis of the network traffic, the identification of patterns and all other information that can be extracted from them, using a real home network environment with commercially available smart home appliances. This will give us a more clear picture of the traffic flow in a home network, and also the clues to help deciding where to go in future developments. After that, it is possible to define the set of rules to monitor the flow of information originated in the home appliances and going to the internet. To achieve all these goals, it is needed to develop and implement a probe that will be capable of monitoring all the traffic, and act as a prototype of a future traffic monitoring device.

Also, it is intended the creation of a program for performing the aforementioned traffic analysis, based on the defined rules. This program should be light because it will run in a device with limited computing capability, but also powerful enough for performing the task without introducing lags in the network.

1.3 Contributions

This work intends to be a contribution to this new security paradigm giving a more clear picture of the traffic flow inside the network. The main goal is to help in identifying and profiling the data flow between home appliances and the internet, trying to figuring out if unwanted and uncontrolled data is leaving the home network towards an unknown server somewhere in the internet.

After that work is done, it will be possible to define and create a set of filtering rules to be used in future IDS or similar network monitoring systems, for the control of the home network outbound traffic. Those rules will be applied in an experimental probe to be implemented in a real home network environment.

Hopefully, it will also contribute to the awareness for these new kinds of threats, and for that we need to look to our home networks security in a new perspective with closer attention, as these networks are growing in size, diversity and complexity.

1.4 Organization

The organization of this thesis is as follows: Chapter 2 provides a literature review of the work that has been done about small sized networks, and the performance of hardware devices in these environments; Chapter 3 describes the concepts that supports the implementation of all the system; Chapter 4 explains how the test bed was implemented to do the tests; Chapter 5 is about the organization of the network data collection; Chapter 6 dives in the network data analysis; Chapter 7 details the code implementation and Chapter 8 makes the conclusions of the work.

Chapter 2

Literature Review

The subject of network intrusion detection has an extensive set of literature available. Many of these papers are about intrusion detection in general, while a few others specifically focus on intrusion detection in the scope of home networks. There is also some literature about concrete home IDS implementations, using both commercial and freely available products (whether software or hardware), usually testing different combinations. This chapter presents a review of some of the more significant studies made on this area, organized into four different sections: network traffic analysis; network intrusion detection techniques; security in the specific scenario of home networks; and practical implementations.

2.1 Network Traffic Analysis

Kidwell et al. [1] propose an algorithm to detect keystrokes in Secure Socket Shell (SSH) communications. It is based on the TCP packet exchange dynamics of the SSH protocol. By monitoring a series of communication details (like how many packets were sent, if it is a server or client communication, data size of server and client packets, among others), it allows detecting the packets that contain keystrokes. Authors claim the algorithm has good accuracy and is able to detect SSH communications over non-standard ports, permitting the detection of backdoors. This is an interesting approach, specially to detect already known attacks, but has the limitation of being protocol-specific and not useful for detecting unknown anomalies.

Mohamed et al. [2] analyze the use of neurofuzzy logic for short-period throughput prediction in network traffic, by applying three training-based models (AutoRegressive - AR; AutoRegressive Moving Average - ARMA; a-SNF). This work shows it is possible to have good prediction accuracy without large amounts of training data. Authors also show that large packets (>= 800 bytes) shape the traffic flow. It is an interesting work but has is, it is not very suitable for practical, large scale application, mainly because the hardware resources typically available at the home network are not powerful enough.

Yuantian et al. [3] analyze how to monitor traffic in big data environments. They draw some interesting conclusions, arguing that for real time analysis the flow traffic statistical analysis is better suited than port or payload based approaches. This is because ports can be changed by malicious attackers and payloads are hard to analyze (specially if they are

encrypted). This work also features a part dedicated to botnet detection and claims to be able to detect a presence of a botnet by analyzing DNS queries (specially the existence of many "Non-Existent Domain" responses). This is due to the pseudo-random generation of domains made by the botnet. This work also enforces the need of using machine learning techniques to perform the network statistical analysis.

Biernacki A. [4] proposes a way to identify adaptive video streaming. Adaptive video streaming using Dynamic Adaptive Streaming Over HTTP is an increasingly popular streaming protocol to transmit multimedia content. This puts overloading to the ISP when trying to optimize traffic. DASH works over HTTP, so port analysis is pointless, and most traffic is encrypted, so Deep Packet Inspection is also not possible. Another problem is that DASH traffic is usually mixed with other HTTP traffic. The best alternative is traffic analysis using machine learning techniques. However, the proposed algorithms needs to know the specificity of the different video streaming algorithms, to produce reliable results.

Wang et al. [5] propose yet another algorithm to detect malicious attacks using machine learning techniques. It is called "Seed Expansion", and is based on clustering schemes. This approach is supposed to have better performance in detecting the more recent attacks, that show polymorphic behavior and resource to more elaborated attacking techniques. The authors claim this algorithm has better performance then others, such as K-Means.

Miao et al. [6] dwell into the discovery of zero-day malicious traffic. They propose a distributed framework using machine learning techniques combining supervised and unsupervised learning mechanisms. The distributed nature avoids the problems of having a single server performing all the detection. Unsupervised learning used the K-means algorithm, while Random Forest is used for supervised learning.

Rossow et al. [7] present an approach to detect malware which consists in analyzing known malware attacks and creating a profile of the network flow of a malware attack. To do it, they use the "Sandnet" analysis environment, and the tests where over DNS and HTTP protocols, since these are the two more used protocols in malware attacks. Sandnet has more then 100,000 malware samples. Differently from other malware analysis that usually run for short periods of time, this experiment ran tests for about one hour.

2.2 Network Intrusion Detection Techniques

Hindy et al. [8] perform a thoughtfully analysis of network IDS taxonomies regarding security. Because the network is an increasingly complex and ever changing environment, it is of utmost importance to accurately catalog the different kinds of threats. According to the authors, there is no available taxonomy that reflects this reality, and the existing threat datasets are outdated. The work explores three main aspects:

1. Presenting an Intrusion Detection Systems survey and taxonomy.
2. Evaluating available datasets.
3. And proposing a threat taxonomy.

This paper helps to give a clear picture of the IDS landscape, and what to consider when selecting an IDS for a specific purpose. In this thesis the adopted taxonomy will be according to Table 2.1:

Table 2.1: IDS Taxonomy

Criteria	**Possibilities**
Type of analysis	Signature based Anomaly detection based
Positioning	Network based Host based
Action taken	Log results Trigger alerts Take counter-measure actions
Detection method	Isolated Distributed/Collaborative
Accuracy needed	Mandatory No network perceivable lag

Othman et al. in [9] provide another perspective of IDS classification, by doing also a comparative analysis of their characteristics, advantages and disadvantages. The paper goes through the entire IDS landscape, not focusing in a single type. It presents new types of IDS besides the traditional Network-based IDS and Host-based IDS, including the Hybrid IDS, Distributed and Collaborative IDS, Wireless IDS, and Hypervisor-based IDS.

Wanda et al. [10] also look over the types of IDS and perform an extensive analysis of the available literature for IDS detection techniques. they take a closer look at new types of IDS environments like IoT, MANET, VoIP and WSN. This paper is also be very useful as a reference for further readings focused on those specific areas.

Zhou et al. [11] look at the subject of data collection over a network. Three types of data collection are presented. Packet-based collection, considered the most widespread, analyzes each packet that flows in a network. It is a very effective method but with the growing volume of network traffic it may became impractical. Flow-based collection, which measures the number of packets of the same type that pass on a network point at a given time, is also very common, being particularly useful in distributed environments. Log-based data collection looks at the information saved in system log files to pick up the desired data. Not having a standard log file format, and considering the huge amount of information log files contain, this kind of analysis can only be performed with the help of automatic tools.

Latha et al. [12] analyze the different IDS techniques, giving special emphasis to the different attack profiles that attackers and malicious users may use when performing network attacks, including Denial of Service (DoS), Probing, Remote to User (R2L) and User to Root (U2R) techniques.

2.3 Security in the Specific Scenario of Home Networks

Alaba et al. [13] analyze the state-of-the-art of IoT and associated security threats. They also propose a taxonomy for those threats, along with the proposal of possible detection and mitigation solution.

Microsoft [14] explains why it is fundamental to have an IDS even in small networks such as home and small office networks. This is because those types of network are also targeted by distributed attacks, with the potential exposure of personal information and the risk of becoming instruments of attack to third-party networks (especially those of friends, co-workers and employers).

Cruz et al. [15] introduce a new framework for home security that intends to fill the gap currently existing between ISP security approaches and home network security approaches. These are traditionally different domains that do not cooperate with each other. This "no one's land" facilitates attacks on home networks, because the average user is unable to properly manage its network. To overcome this, the proposed framework provides a distributed IDS system combining residential gateways and ISP IDS systems for detecting and mitigating malicious attacks.

Mantere et al. [16] perform an analysis on the specificities of network traffic on ICS networks, and how that affects an IDS performance. Although this is a very different environment, the analysis on IDS performance impact can also be used for home network systems.

Ye et al. [17] present a new approach using emergent technology such as software defined networks to address the problem of monitoring encrypted network traffic. The proposed method uses machine learning algorithms.

Sperling et al. [18] propose another approach to IDS systems, based on analyzing DNS requests in order to find suspicious sites request. This is also an alternative approach that may help in protect a home network.

2.4 Practical Implementations

Amri et al. [19] propose an based on a Raspberry Pi in conjunction with a NodeMCU for controlling a network of sensors. This paper concerns only with usability, and it can provide a notion of what a smart home can be and how networked devices will be all around.

Ahmed et al. [20] propose a methodology for collecting data for further forensic analysis to find evidences of an attack. This is an important aspect for our work since the first step is collecting data and analyzing it to find traffic patterns.

Sforzin et al. [21] analyze the possibility of implementing a home network IDS using Raspberry Pi and Snort IDS program. The result is that the implementation performs well enough to be an option for a home network IDS.

Aspernäs [22] test RaspberryPi's ability of acting as an IDS. The test comprised two models: RaspberryPi model B+, and the latest model RaspberryPi 2 Model B. The used IDS software was Snort. The results show that the latest model, RaspberryPi 2 Model B, performs better (as expected) and in practice is the only one able to realistically perform this task.

Poonia et al. [23] discuss the necessity of IDS in home networks, and the capability of the RaspberryPi to perform IDS tasks in such an environment. The tests were made with a RaspberryPi Model 3, a RaspberryPi Model B+ and a RaspberryPi Model B, all with the same configuration. The test results show that using RaspberryPi Model 3 with Snort IDS provides a good compromise for a Home network IDS. They also show that Snort is the best IDS to use in such as environment.

Kyaw et al. [24] compare Snort and Bro IDS, considering their usage in a RaspberryPi 2 for home or small office scenarios. The capability of the RaspberryPi to accomplish this task was also tested. Results show that Snort IDS performs better, but the RaspberryPi2 may crash if there is intensive traffic in the network.

Zitta et al. [25] analyze another IDS, Suricata, and its performance is tested in Raspberry Pi 3 hardware. This is an alternative to the already aforementioned IDS Snort and Bro. This IDS also performs well on Raspberry Pi, although it needs custom rules to be added.

2.5 Summary

There is plenty of information about network security, design and implementation. There is also some related work addressing the application of those concepts to home networks and IoT. These are extensive topics, allowing for many different and diverse perspectives. But there are also many related fields of research still waiting for exploration. Regarding home networks, their rapid growth in volume and variety towards the massification of IoT and smart-home scenarios present new challenges in several aspects, including security. Due to their heterogeneous nature, home networks need different approaches from those adopted by traditional corporative networks. Some working implementations had already been proposed and tested to prove the concept.

Chapter 3

Proposed Concept

In recent years we have witnessed how the home network environment has become increasingly complex. What was started as a way to have more than one computer linked to the internet, eventually sharing a printer, has become a complex ecosystem with very different kinds of electronic devices, services and usage profiles. Quite often such devices get some kind of information from the environment they are in, but also send their own data in sometimes very complex interactions to servers outside the home network owner control. Moreover, it is expected that this trend will continue in the next years, as more and more devices are available for use at lower prices, and more activities and devices at home are been subject to some kind of automation and remote control.

All of this growing complexity demands a radically different approach regarding home network security. While the traditional threats still exist and are more and more complex, forcing the traditional security methods to rapidly adapt, in the last years we are facing new kinds of challenges. In fact, we are putting more home devices connected to the network, without really knowing how those devices interact with the surrounding environment. Failing to address this security flaw may lead our home networks to become exposed to different kinds of attacks. To make things worse the traditional perspective where the Internet Service Provider (ISP) security concerns ends at the perimeter of the access network, with the users being in charge of taking care of their own security needs, is not compatible with modern and fast changing threats. Is time to start doing some sort of cooperative work.

3.1 Proposed Approach and Basic Requirements

Although the security in the home network is not a new subject, the approach that has been used so far is mostly about intrusion detection systems, looking only for the threats that come from the outside, with home network in itself being considered a secure environment. This has been in the scope of virtually all research that have been done, as the Literature Review shows. But now we are facing a new kind of threat so it is mandatory for us to use a different approach. Controlling the information produced in our home network, by our devices, is now as important as to monitor what is coming from the outside. Introducing this new perspective, where intrusion or security threat may come from our own appliances, we hope to bring the attention to this new security reality, and doing a valuable contribution to this field of research, because sooner or later we will have to face these questions.

Being this a new network security perspective, where there are still more questions than answers, several different possible approaches may be considered in order to solve the aforementioned problems. In this thesis, it is proposed a solution in the form of a framework, whose concept is shown in Figure 3.1:

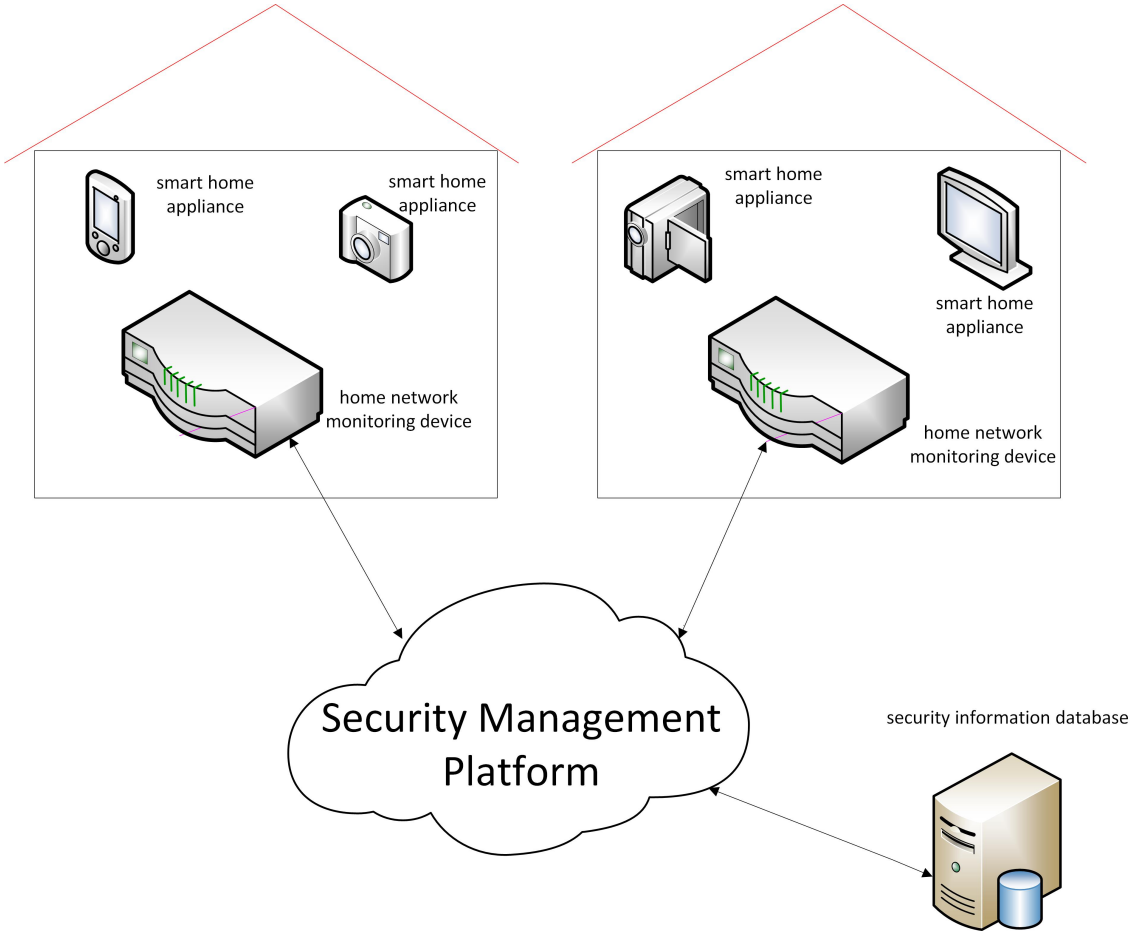


Figure 3.1: Home Security Architecture

The concept that is proposed and tested here is a possible solution that addresses most of the problems we previously referred to. Since we have two main actors, ISP's and home users, the basic idea is to share responsibility and surveillance duties between the two. The ISP will be responsible to create and maintain the Security Management Platform (SMP) which includes a database server, keeping all the security related information. The users will have a monitoring device on their own networks to monitor traffic flows and communicate with the ISP's SMP to receive updated information and upload their own.

This solution may raise other kinds of questions, because it implies exchange of personal information with third-parties. Those questions are not addressed here, because this is a first approach and the main focus is the technologically feasibility. A final complete solution will obviously have to take into account all those other requirements.

3.2 Appliance Design Considerations

To perform the capture and analysis of the network traffic, a probe will have to be set up in a real home network environment, because we need real traffic, characteristic of a typical home LAN. One of the main premises for this work has to do with avoiding the use of hypothetical or artificially generated traces, instead focusing on using real data. This approach has been deemed to be the more adequate to assess the feasibility of the concept. Having that done, in a second stage, the same home network will act as the monitored system, for the probe with the monitoring program.

3.2.1 Network Positioning

The proposed device has to be deployed like a network-based IDS, sitting between the home gateway and the rest of the network in a bridge configuration, ensuring that all the traffic will flow through it, also acting as DHCP and DNS server to be able to collect all DHCP and DNS requests for later analysis. To be able to analyze wireless network traffic, the device must have a wireless network interface, configured as a wireless AP. The implemented testbed is described in Fig 3.2.

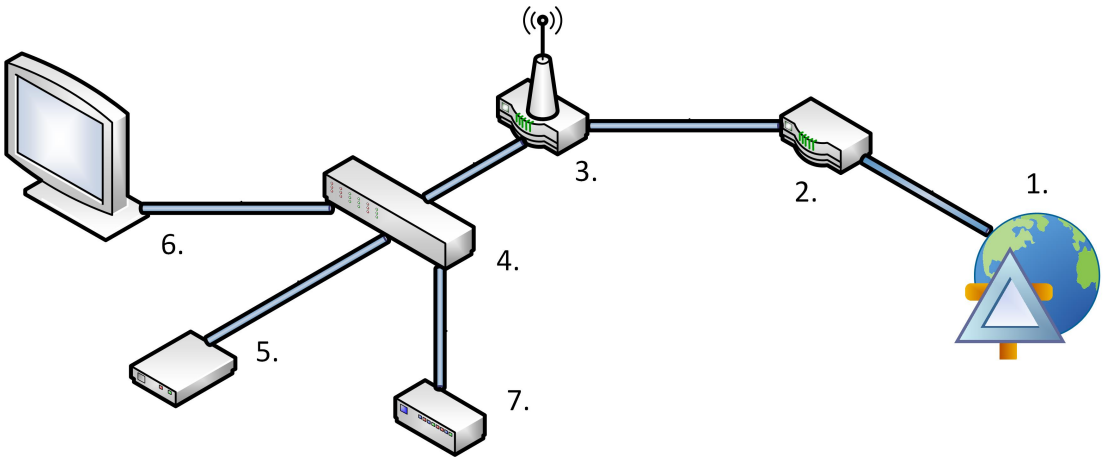


Figure 3.2: System configuration

Legend:

1. Server on the Internet
2. Home gateway
3. Home network monitoring device and wireless AP
4. Home switch
5. DVD/BD reader
6. Smart TV
7. TV Set Top Box

Note: Although the home network will likely also include computers, they are not shown here because they are not relevant for this work, nor its traffic will be analyzed.

3.2.2 Design of the Probe

To monitor and block outgoing traffic requires an approach somehow similar to a "reverse IDS". A device whose purpose is to analyze the traffic sent by the home network devices and going out to the internet. As this is a rather new concept, there are not known references about using a home router or some sort of appliance to perform such kind of monitoring tasks. Not only this is a new problem and, as such, the general public is not aware of it, but also it is not yet seriously addressed by the industry, because costumers are often only concerned about features, not security. On the other hand, the home routers which are usually supplied by the ISP are "black boxes" so it is not possible to add new features to them, nor know anything about how they work. In the future maybe this capability will be incorporated in home gateways, putting all the features in one single device but, by now, a specific device is needed to perform this task. This device must fulfill some characteristics, to be able to be used in a home network:

- √ It must not be expensive
- √ It cannot be much power consuming
- √ It cannot take much room
- √ It must be easily configurable
- √ It must be able to perform traffic monitoring without introducing perceivable lags in the network
- √ It must be able to run a capable IDS, or similar, application
- √ It must have (or be able to be added) two wired and one wireless network cards

It is easy to see that are conflicting requirements for this device. To find a cheap, low power device that performs well, is what all the hardware industry is looking for. But, as the device is to be deployed on a home network, we are not looking for a very demanding piece of hardware. In fact, the amount of traffic that usually flows in a home network and needs to be analyzed is not that much, so it is feasible to find such a device. Moreover, the tasks that the device will have to perform are not too intensive, from a computational standpoint.

Regarding software, the selected Operating System is to provide the execution environment for the device and needs to be lightweight and highly configurable. In fact, this is a device dedicated to a single task so it will not need to run some services usually found in most of the OS available and designed for general purpose use. For packet analysis what we need is something similar to an IDS, with a slightly different set of rules. In fact, what the software must do is to look at the packets going out of the network towards the internet, searching for specific predefined patterns. This is something that is currently supported by existing IDS with the only difference being the fact that such IDS look for incoming traffic rather than outgoing. What we will try to do is to apply IDS rules to the outgoing packets, monitoring the content that the home appliances will send to locations external to the LAN, somewhere on the internet.

Chapter 4

Implementation

This chapter will be focused on the probe development. Choosing the right hardware for the implementation presents several challenges related to the fulfillment of the requirements listed in the previous chapter. Due to the particular characteristics of the proposed device, we cannot expect to find something tailored for the job. Also, we have to face the possibility that the device we are looking for simply does not exist. If that happens, a decision must be made to select which requirements are mandatory, and which ones can be relaxed or even dropped. Also, it must be taken into consideration the supported OS and the development software the platform accepts, as well as the the existing software it supports. Another point to take into consideration is the amount of documentation and support about the device, whether being about software development, existing programs, or hardware capabilities.

4.1 Candidate devices

Doing an online searching gives us some possibilities: the ubiquitous Raspberry Pi [26] family of products, which almost came as a first choice; then the Raspberry Pi "clones" like the Banana Pi [27] and Orange Pi [28]; the Odroid family of Single Board Computer (SBC) [29]; and the Asus Tinker Board [30], which also looked promising. Of course, there is a plethora of other similar devices, but they are in general less powerful, or reliable, or even poorly documented. Furthermore, it is not feasible to analyze them all here.

4.2 Hardware implementation

Taking everything previously stated into account, the choice fell on a Raspberry Pi 3 B+, the most recent version of the Raspberry family, by the time this analysis was undertaken. Besides, this device suits the hardware requirements, it has the most comprehensive support options, whether is about literature, software libraries or online support groups. It has also been deeply tested in academic papers as the literature review as shown.

The Raspberry Pi 3 B+ model has the following characteristics [31]:

- Broadcom BCM2837B0, Cortex-A53 (ARMv8) 64-bit SoC @ 1.4GHz
- 1GB LPDDR2 SDRAM

- 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN, Bluetooth 4.2, BLE
- Gigabit Ethernet over USB 2.0 (maximum throughput 300 Mbps)
- Extended 40-pin GPIO header
- Full-size HDMI
- 4 USB 2.0 ports
- CSI camera port for connecting a Raspberry Pi camera
- DSI display port for connecting a Raspberry Pi touchscreen display
- 4-pole stereo output and composite video port
- Micro SD port for loading your operating system and storing data
- 5V/2.5A DC power input

This model of the Raspberry Pi has reasonably good computing capabilities, so it will probably be able to perform the required tasks. This assumption is supported be the number of papers which assessed the computational capabilities of the Raspberry Pi (cf. Section 2), some of them using older models. Besides that, it is a small, inexpensive device, with the bonus of being able to look neat on a house, thanks to the great variety of cover boxes available for purchase.

As the Raspberry Pi comes with only one Ethernet interface, a second interface needs to be added. The solution is to use an USB-Ethernet adapter which takes one of the available USB ports. For the rest of the setup, the Raspberry Pi comes with what it needs, specially the wireless network card. The prototype is depicted in Fig 4.1.

Figure 4.1: Used device

4.3 Software implementation

Regarding software (and more specifically the execution environment), the chosen OS was the Raspbian Linux distribution [32]. It was chosen because it is based on a Debian Linux distribution and has a lightweight version, having only the core OS structure and allowing the user to customize the system to its own needs, while avoiding overloading the hardware with non-relevant services.

Development environment

It is now time to choose the development environment to create our program. One thing that was taken for granted since the early stages of this work was that, regardless of the chosen option, support for the libpcap library would be a mandatory requirement. Libpcap can take care of all the work of dealing with the network interfaces at low level and do the captures, leaving to us the programming of the traffic analysis, the core feature of the program.

The first decision was to choose the programming language. There were two main candidates, among all available options: C and Python. C is arguably the fastest language, and not too demanding in terms of memory usage, which are very relevant characteristics for a program that has to run in a device with limited computing capabilities such as the Raspberry Pi. It is also very well documented, since it has been around for a long time, and has been used in a wide variety of programs. Python, on the other hand, is a modern programming language, with all the expected features, being much simpler to work with, with a huge community that provides ready-to-use libraries for all kinds of needs, including dealing with the libpcap library, and data structures for managing the data.

Eventually, the C language was chosen, due to the low computing overhead and lower memory usage. On the other hand, it is expected that the program will increase in functionality and associated complexity, while the device should remain the same, because of the need for stability and cost. The C programming language will better support an increase in size and complexity without compromising performance. Finally, there are a significant number of resources (books, articles, community sites, among others) related to network programming in C.

The structure of the program is depicted in Fig 4.2.

4.4 Testing network

The testing network consists of a home gateway with wireless AP, DHCP and DNS functions disabled. The Raspberry Pi, acting as monitoring device is configured as wireless AP access, DNS and DHCP servers. The other devices are the following:

- Smart TV 1: LG 32LN577S-ZK
- Smart TV 2: Sony KD-49XE7096
- Blu-ray Disc Player: Sony BDP-S3700

- Set Top Box: Altice Labs FiberGateway GR 241AG (MEO)
- Home ethernet hub: D-Link DES-1008D

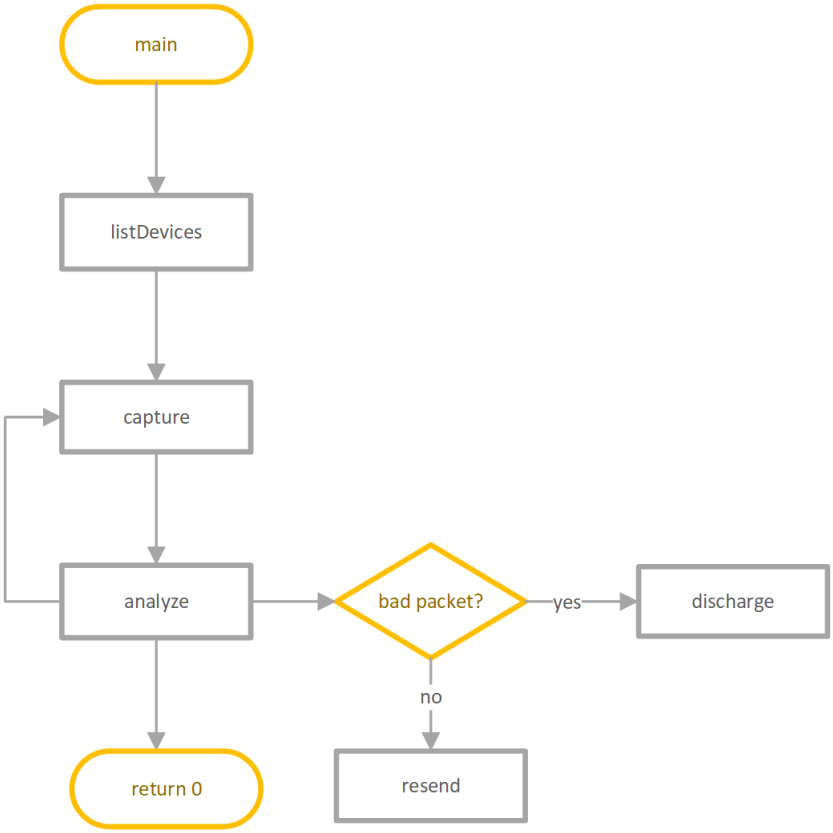


Figure 4.2: Program Structure

Chapter 5

Collection of Network Data

To start testing, the first thing to do is to find out how different devices communicate within the network, and to the outside. As such, a first scan has to be made in order to reveal relevant details about the devices, namely open ports, and also OS types and version. After that, is time to start the collection of data from the different devices. Since this work has to be done with applications that require a fair amount of computing capability, like wireshark[33], it was decided to use a computer running CentOS to do the captures, using the same bridge setup as the Raspberry Pi in the network.

5.1 Device details

To get the details of the different devices, the nmap [34] utility was used. The nmap is a free and open source utility program for network discovery and security auditing. To perform these tests the following commands were used:

- To find the network devices: nmap -–send-eth eth0 -sn -S 192.168.1.99 -n -–scan-delay 1 192.168.1.1-254
- To scan TCP ports: nmap -p 1-65535 -T4 -A -v <IP address>
- To scan UDP ports: nmap -sS -sU -T4 -A -v <IP address>

The meaning of the used parameters are [35]:

- -p 1-65535 – Port numbers to scan. In this case were all
- -sS – TCP SYN scan
- -sU – UDP scan
- -T4 – T<0-5>: Set timing template (higher is faster)
- -A – Enable OS detection, version detection, script scanning, and traceroute
- -v – Increase verbosity level
- <IP address>- Target IP address

The results of this analysis are summarized in the Tables 5.1, 5.2, 5.3, 5.4:

Table 5.1: Smart TV 1

Operating System		
Running: Linux 2.6.X|3.Xs OS CPE: cpe:/o:linux:linux_kernel:2.6 cpe:/o:linux:linux_kernel:3 OS details: Linux 2.6.32 - 3.5		
TCP		
Port	Service	Observations
1733	upnp	LG TV upnp (UPnP 1.0; DLNADOC 1.50; LGE_DLNA_SDK 04.28.17)
1868	upnp	Platinum upnpd (LG TV model: 32LN577S-ZK; Neptune 1.1.3)
8060	http	lighttpd 1.4.28
8080	tcpwrapped	
9955	alljoyn-stm?	
56789	tcpwrapped	
56790	tcpwrapped	
UDP		
Port	Service	Observations
1900	upnp	
6004	X11:4	
17282	unknown	
21524	unknown	
23531	unknown	
32528	unknown	
38037	landesk-cba	
41967	unknown	
49152	unknown	
49190	unknown	
49210	unknown	
49259	unknown	
57410	unknown	

Table 5.2: Smart TV 2

Operating System		
Running: Linux 3.X|4.X OS CPE: cpe:/o:linux:linux_kernel:3 cpe:/o:linux:linux_kernel:4 OS details: Linux 3.2 - 4.9		
TCP		
Port	Service	Observations
2870	upnp	IPI Media Renderer upnpd 1.0 (UPnP 1.0; DLNADOC 1.50)
13000	unknown	
56789	tcpwrapped	
56790	tcpwrapped	
UDP		
Port	Service	Observations
593	http-rpc-epmap	
1066	fpo-fns	
1900	upnp	
16972	unknown	
19154	unknown	
19273	unknown	
21333	unknown	
25240	unknown	
29243	unknown	
32931	unknown	
34038	unknown	
50612	unknown	
61322	unknown	

Table 5.3: Blu-ray Disk Reader

Operating System		
Linux 3.10; Device: media device; CPE: cpe:/h:sony:bdp-bx58 cpe:/o:linux:linux_kernel:3.10, cpe:/h:sony:bdp-s3700		
TCP		
Port	Service	Observations
50001	upnp	
50002	http	Sony BDP-BX58 TV http config 2.0
50201	http	Sony BDP-BX58 TV http config 2.0
50202	http	Sony BDP-BX58 TV http config 2.0
52323	upnp	
54400	http	Sony BDP-S3700 http media client (av=5.0; mv=2.0)
UDP		
Port	Service	Observations
113	auth	
137	netbios-ns	
512	biff	
664	secure-aux-bus	
997	maitrd	
1024	unknown	
1900	upnp	
3389	ms-wbt-server	
19956	unknown	
21320	unknown	
34358	unknown	
39632	unknown	
41638	unknown	
49204	unknown	
64727	unknown	

Table 5.4: Set Top Box

Operating System		
Running: Microsoft Windows PocketPC/CE OS CPE: cpe:/o:microsoft:windows_ce:5.0 OS details: Motorola VIP1200-series or Swisscom Bluewin TV digital set top box (Windows CE 5.0)		
TCP		
Port	Service	Observations
8080	http	T-Home Telekom Media Receiver httpd
8082	blackice-alerts?	
8086	http	Microsoft Mediaroom httpd (IPTV tuner)
53208	tcpwrapped	
UDP		
Port	Service	Observations
1025	blackjack	
1031	iad2	
1900	upnp	
5000	upnp	

From this first analysis it is possible to infer some information. First, the number of open UDP ports is greater than the TCP ports in all devices. Second, there are no standards regarding the usage of port numbers and respective protocols; the only exception to this is port 1900 UDP that is used in all devices for upnp broadcast. Most of the ports used are associated to unknown protocols, meaning that the manufacturers use their own protocols (or customized versions of existing services), therefore requiring a deeper analysis, encompassing features such as packet size, their frequency, or if particular protocols are

associated with specific action taken by the user on the device.

5.2 Packet collection

The captures will be done with the tshark tool, which is the command line version of the wireshark network protocol analyzer. This is partly due to the fact the computer we used for this purpose had a command line version of the OS, with no GUI environment. The wireshark tool allows for a more detailed analysis regarding some important aspects of network traffic, such as packet size, used protocols, source/destination addresses and ports, and packet frequency. Also, this also eases the search for more subtle behavioral patterns, related to the kind of messages that are sent out in response to specific actions, whether they are income communications or user actions on the device, or if the packet size is not consistent with known device solicitations. This will allow us to profile the different device behavior regarding network communications in the home network, establishing reference patterns.

The tshark command used for the captures is: sudo tshark -i br0 -F pcapng -W n -w <filename>.pcapng

where[36]:

- -i br0 – interface where tshark is sniffing, in this case the bridge "br0"
- -F pcapng – file output format, in this case the wireshark file format
- -W n – save host name resolution records along with captured packets
- <filename>.pcapng – path and filename of the capture file

To analyze the traffic we have chosen the interactions with the Netflix and YouTube content delivery sites. The purpose of this effort is not to analyze regular stream consumption, but rather to understand what other communication patterns are established by the Smart TV, and to where. Using this information to design a signature of the legitimate interactions is an important part of the analysis.

The test methodology starts by switching on the device, navigating to the desired application and start it. Once within the application, specific content (music or movie) is selected and reproduced. Each interaction (and corresponding network data collection) will take at least one hour, although some will be longer. To reduce the amount of useless traffic, during a capture the connections to the devices other than the one being analyzed will be disconnected from the switch, whenever possible (maximizing isolation). That way, the resulting capture file will be cleaner and easier to analyze. The tshark application is started before the device is switched on, and stopped after the device is switched off.

Chapter 6

Network Analysis

This chapter focuses on the analysis of the captured traffic. This is one of the fundamental parts of this work, because the conclusions of what we will find here will be the foundations of all the consequent work regarding the setup of a home network security environment. The aim of this chapter is to analyze the network patterns in order to create a profile for the specific smart device and application. Having done that part, it will be possible to start monitoring the network and be able to detect something that is out of the expected profile and will require deeper analysis to find a possible threat.

6.1 Methodology

The analysis process starts by doing a first division for each device, and on each device for every application. For the Smart TV's and BD Reader the chosen applications are Netflix and YouTube. Those applications were chosen because, being very popular, it is more likely that the TV manufacturer, or the site itself, may be tempted to gather information from users to sell it to third parties. On the other end, their popularity makes them more prone to attempts from malicious attackers to "poison" their TV applications. Also, the applications exists in the three smart devices - the two TV's and the BD reader - so it will be possible to do cross analysis. For the TV Set Top Box only normal television broadcasting channels will be analyzed, because it does not have applications.

The first thing to do is to "clean" the capture file, filtering it by the Ethernet address of the device to exclude out all other network communications. Then, we will look at the Domain Name System (DNS) requests made by the device, taking special attention to "no such name" responses. The DNS requests will give us a picture about to whom the application is talking to. The "no such name" response may indicate a device infected with a worm searching for a server. Next, we will look at the Hypertext Transfer Protocol (HTTP) POST requests because POST is supposed to send data to the receiver server so it is also a good place to look at for information. After that, it is time to look at the packet size percentage, and packet burst rate. These are two good indicators to profile a network stream. The first thing to analyze is the percentage of packages according to its size. First we will do it using the complete stream and comparing them all, and on a second step we will separate the streams in different parts. One with the first hour, another with the first half-hour, another with the first ten minutes, and a last one with one minute, in this case the fifth minute for every capture. The reason to do this is to check if the stream pattern keeps unchanged as we reduce the time frame of the capture. This is because we want to

be able to identify stream patterns as soon as possible, so we need to find the smallest part of a stream, in its beginning, that holds the general characteristics. The only exception is the one minute capture, because the first minute as different patterns as is the beginning of the communication. The fifth minute looks a good compromise between been in the beginning and having already the general stream pattern. Then it will be the time to look at the packet burst (packets/sec) using two different approaches. In the first one, we will analyze packet burst using bar graphics as before and also separated by packet size, and slicing the stream as in the precious analysis. In the second part, we will look at the packet burst graphics produced by wireshark showing the temporal behavior of every stream, so to have a visual idea of the general packet burst. Having this done, we will have a fairly good picture of a smart device network profile.

6.2 Smart TV 1 (LG 32LN577S-ZK)

We start by analyzing the Smart TV 1. This TV came with LG NetCast, the legacy platform for LG TV (until the year of 2014), now replaced by the LG webOS TV. The NetCast Platform has the usual facilities of this kind of device, such as content delivery of the most popular providers, an online store for LG applications, a graphic user-friendly interface, and the possibility for independent application development [37].

6.2.1 DNS Traffic

Looking at the DNS requests made by this device we can see that it starts by making requests to the manufacturer websites. This is a normal behavior since it has to search for updates. Then it starts to request servers from the chosen application (Netflix or YouTube). There is also some "no such name" DNS responses and we see a strange behavior on those. The application starts by doing pointless requests using an apparently random string generated with no domain construction rules, different each time. It does this for three times, in a very short period of time. Searching the web, we find that this is a normal behavior for the Google Chrome web browser[38], so we assume that the application web browser is somehow based on Google Chrome or follows similar rules. Other than that, there are requests to the "rdvs.alljoyn.org" domain, made at very regular intervals, two in a row every three minutes, during all the captured streams. The "Alljoyn" brand is an IoT framework [39], although the specific requested domain does not exist, according to [40]. This is clearly a vulnerability that should not occur (since someone can "buy" this domain and therefore become able to communicate with the TV).

6.2.2 HTTP POST

The analysis to the HTTP POST requests reveals that the television sends information about itself to some server on the internet. Also, the Netflix application sends information in a encoded format. There are also some communication from Windows 10 computers on the network regarding Digital Living Network Alliance (DLNA). DLNA is a certification for digital content distribution in a home network that uses a computer as a content server and HTTP as a media transport [41]. The YouTube application sends a POST requests to some server belonging to the manufacturer, and also some DLNA communications with a Windows 10 computer.

6.2.3 Netflix

The first application to be analyzed is from the Netflix multimedia content delivery company. Four captured streams are analyzed, in order to profile the regular streaming of a video. The characteristics of the stream captures are provided in Table 6.1:

Table 6.1: Smart TV 1 Netflix streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	8291	02:58:02	10682,691	8199000614	767	2711641	253,8
Stream 2	3784	01:51:38	6698,728	3742566057	558	1234228	184,2
Stream 3	9488	04:01:50	14510,878	9389098295	647	2937254	202,4
Stream 4	4781	01:47:25	6445,272	4730727772	733	1502579	233,1

Packet percentage analysis

The first thing to analyze is the percentage of packages according to its size in the entire stream. The results are in Fig 6.1:

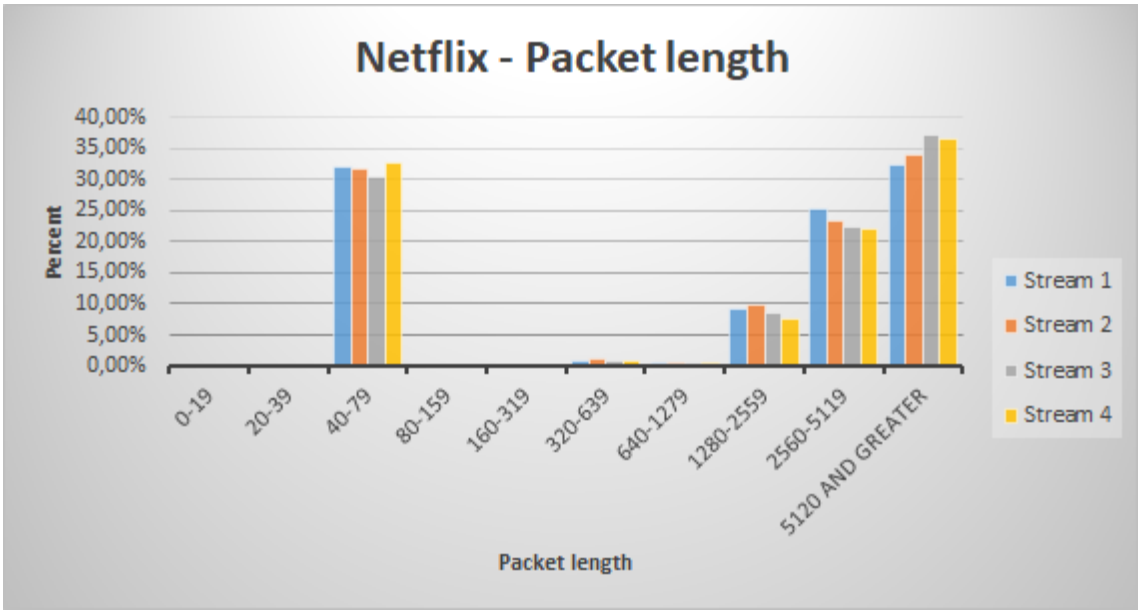
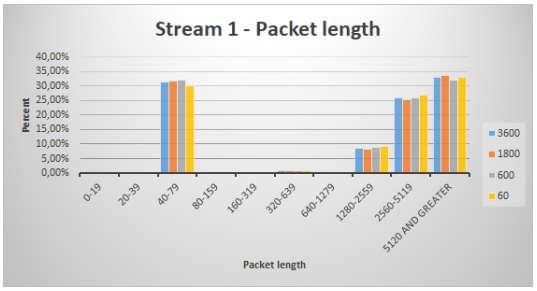


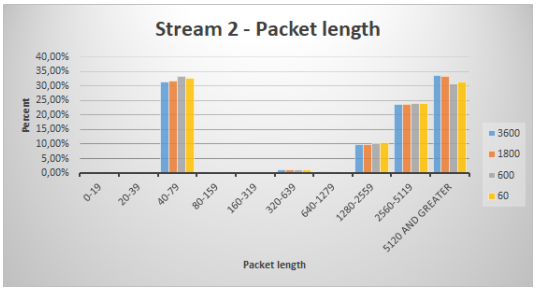
Figure 6.1: Netflix packet percentage

As can be easily seen, there is a common pattern: a visible percentage of packages of some sizes, namely 40-79, 1280-2559, 2560-5119 and 5120 and greater. The other sizes have residual representation or not at all. The percentage of each package group in every stream is very similar. While is not the subject of this thesis to identify the reasons for this behavior, it serves as a profiling tool of this application streams.

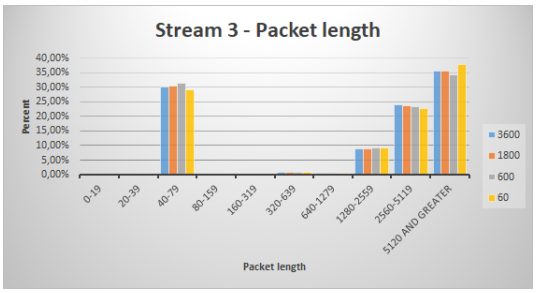
After the first analysis of the streams as a whole, let us make a more detailed analysis of each stream using the stream portion as stated previously. The results are provided in Fig 6.2 (a), (b), (c), (d):



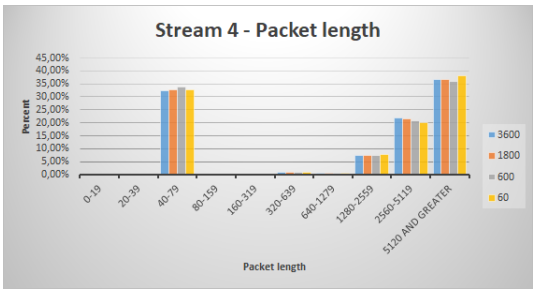
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.2: Smart TV 1 Netflix packet percentage in time frames

We can see from the graphics that the previous general pattern keeps the same as we reduce the time frame. Even with a one minute capture the packet percentage rate remains consistent which indicates this as a good profiling property.

Packet burst analysis

Let us now analyze the packet burst rate. First, let us see the packet burst graphics of the complete streams that are showed in Fig 6.3:

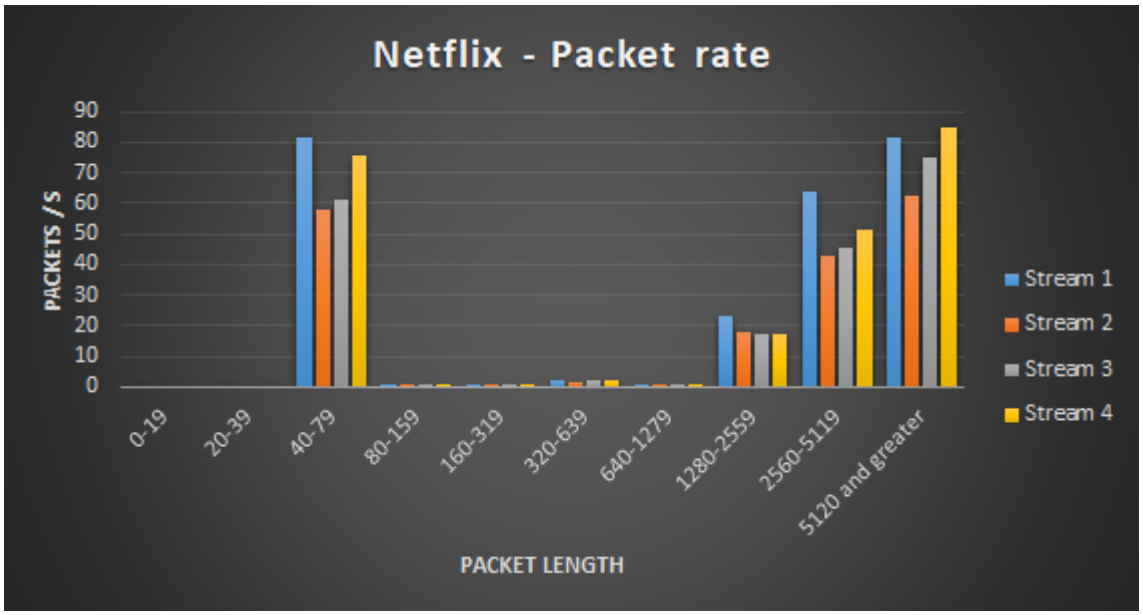


Figure 6.3: Netflix packet rate

The same packet sizes are the more prominent which is the expected result, since these are the sizes most used by the application.

We analyze now each stream in more detail using the defined time portions of the stream as showed in Fig 6.4 (a), (b), (c), (d):

(a) Stream 1 packet rate

(b) Stream 2 packet rate

(c) Stream 3 packet rate

(d) Stream 4 packet rate

Figure 6.4: Smart TV 1 Netflix packet rate in time frames

Here again we can see that the burst rate remains consistent in each stream making this also a good profiling property.

Let us look at the packet burst of each stream along the time. The graphics are in Figs 6.5, 6.6, 6.7 and 6.8:

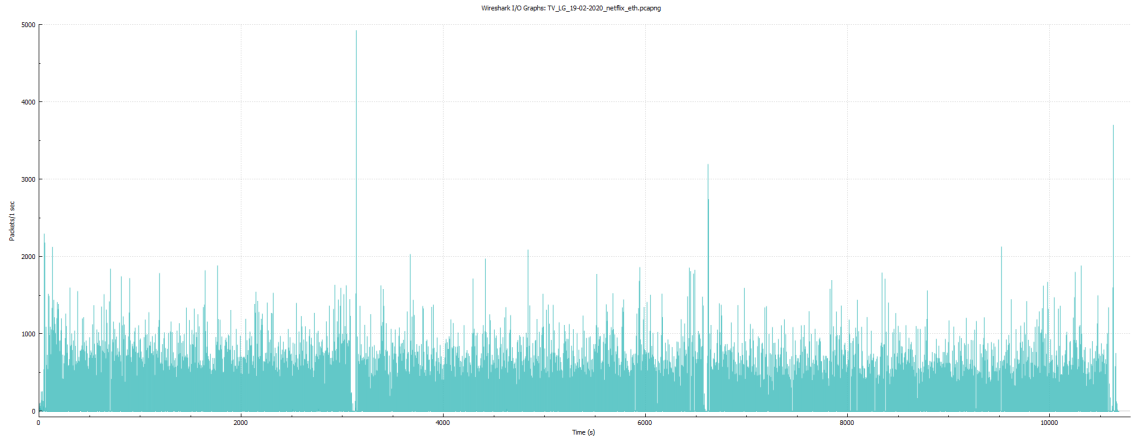


Figure 6.5: Stream 1 Netflix packet burst

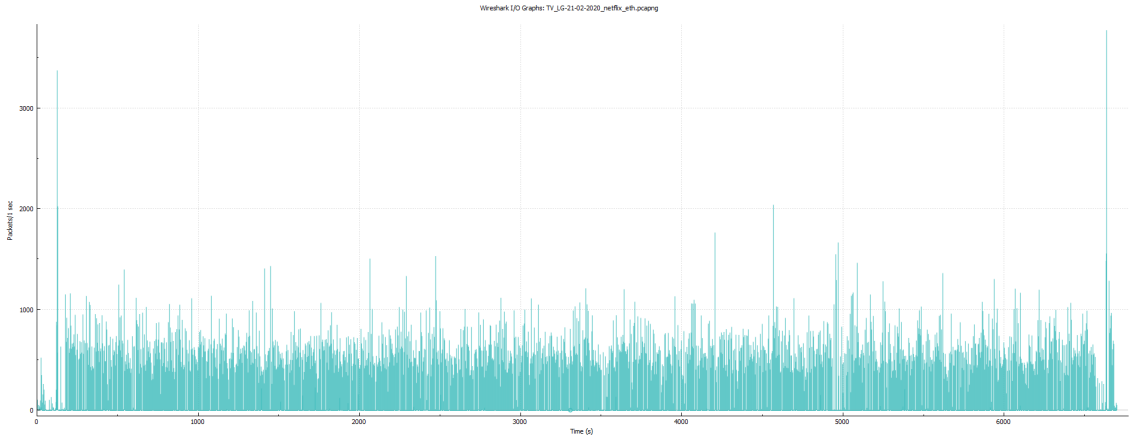


Figure 6.6: Stream 2 Netflix packet burst

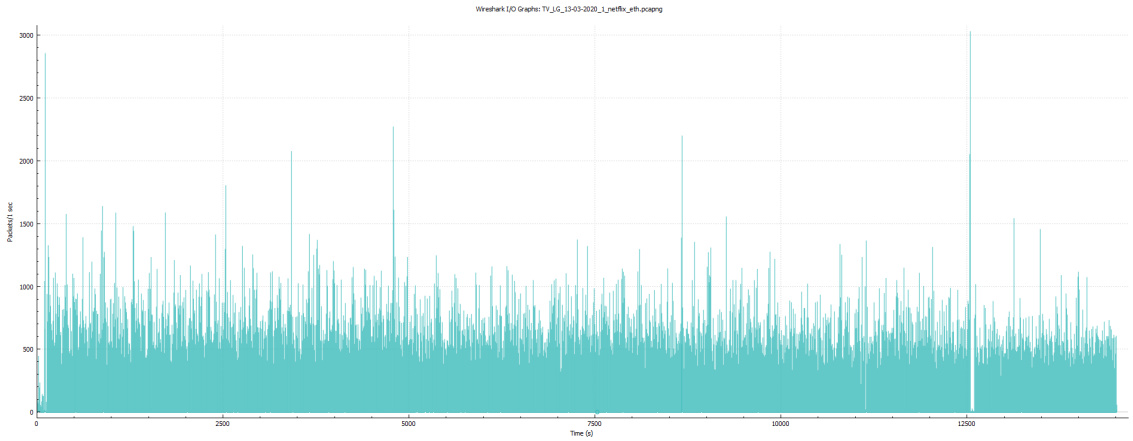


Figure 6.7: Stream 3 Netflix packet burst

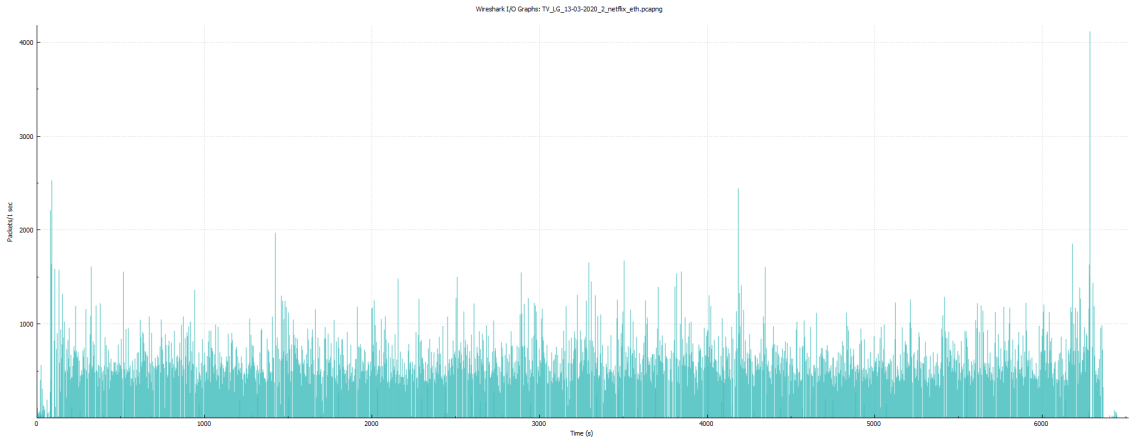


Figure 6.8: Stream 4 Netflix packet burst

We can see, by looking at the graphics that the four graphics have a regular burst rate during all the capture time.

6.2.4 YouTube

We now turn to the analysis of the YouTube content delivery site packet streams. The captured streams are of video sequences that played for more than one hour. The video sequence is decided by the application itself, after the selection of the first video. The main characteristics of the video streams follows in Table 6.2:

Table 6.2: Smart TV 1 YouTube streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	769	02:10:52	7852,592	756900899	96	357933	45,6
Stream 2	1209	02:43:39	9819,040	1190644499	121	566741	57,7
Stream 3	2802	02:43:47	9827,144	2760783894	280	1214266	123,6
Stream 4	1043	02:35:18	9318,079	1026687967	110	500068	53,7

Packet percentage analysis

Let start by analyzing the packet size distribution. The results are provided in the graphic showed in Fig 6.9:

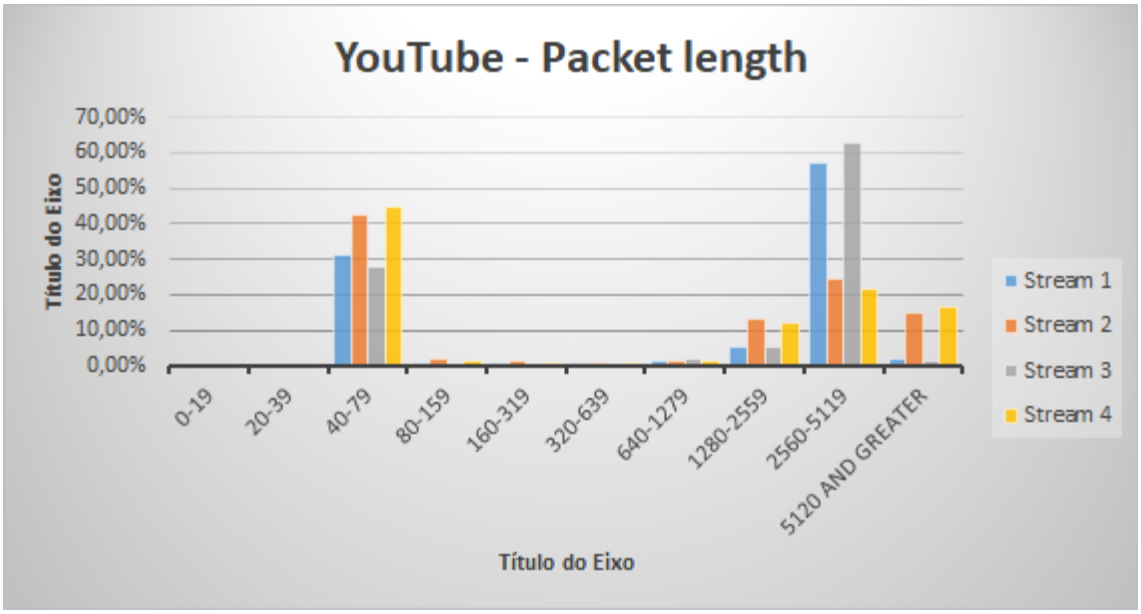
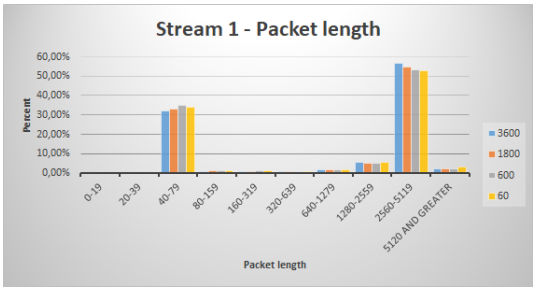


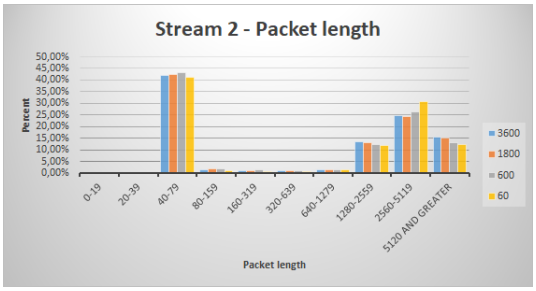
Figure 6.9: YouTube packet percentage

Here we have a somehow different pattern. Although the most percentage are from the some size groups 40-79, 1280-2559, 2560-5119, 5120 and greater, the relative proportion is different for each stream, making it more difficult to use as a stream pattern. One possible explanation for this is different video quality in opposition to the Netflix site where the video quality is more stable.

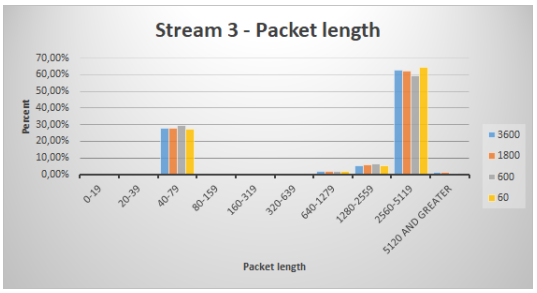
Next, is the analysis of each of the streams using the defined time frames. The results are showed in Fig 6.10 (a), (b), (c), (d):



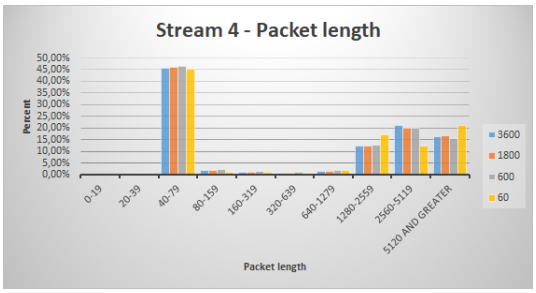
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.10: Smart TV 1 YouTube packet percentage in time frames

Looking at every stream in detail we can see a very regular pattern in each of the streams meaning a consistent behavior, but different from one another. It may be useful once the stream as been identified.

Packet burst analysis

As before, we look first at the packet burst of the whole streams according to the packet length, as provided in Fig 6.11:

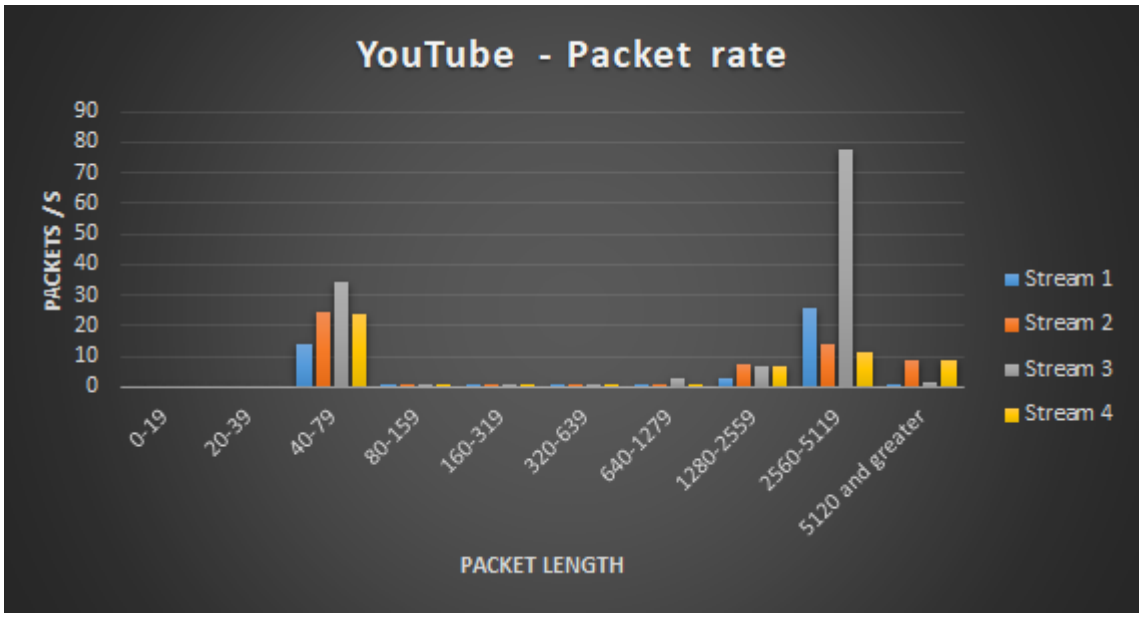
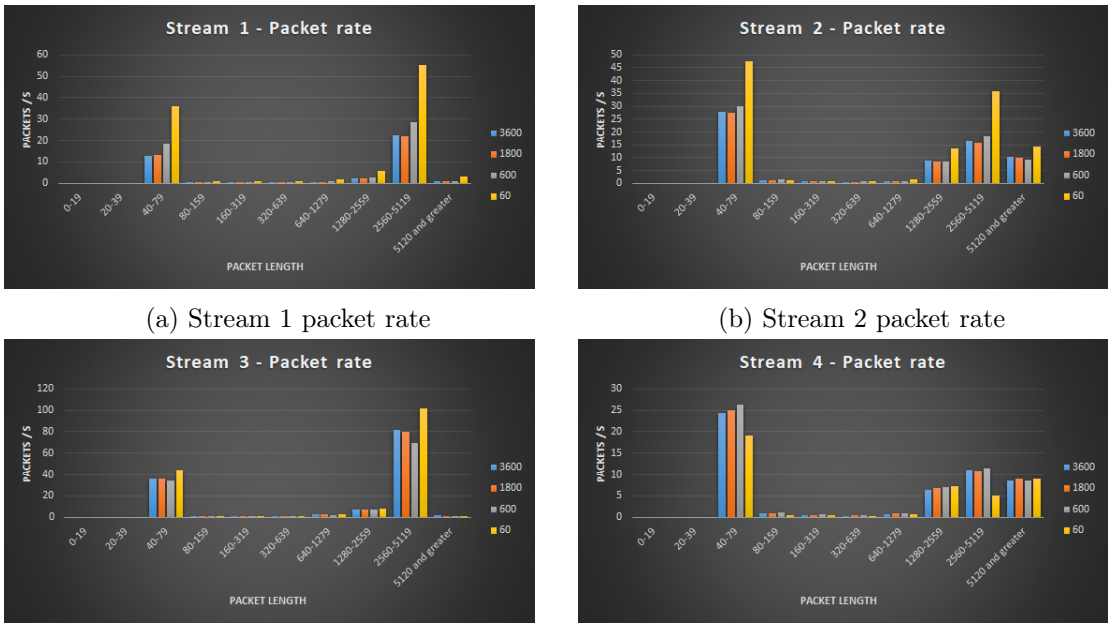


Figure 6.11: YouTube packet rate

Again, the pattern is very different from one stream to another, making it difficult to use as a pattern for the application.

Looking at each stream, we got the following graphics showed in Fig 6.12 (a), (b), (c), (d):



(a) Stream 1 packet rate

(b) Stream 2 packet rate

(c) Stream 3 packet rate

(d) Stream 4 packet rate

Figure 6.12: Smart TV 1 YouTube packet rate in time frames

Although there is some similarity in each of the streams and consistent with the previous graphic, the pattern as some dissimilarity specially in the one minute stream.

Following is the analysis of the temporal burst rate in Figs 6.13, 6.14, 6.15 and 6.16:

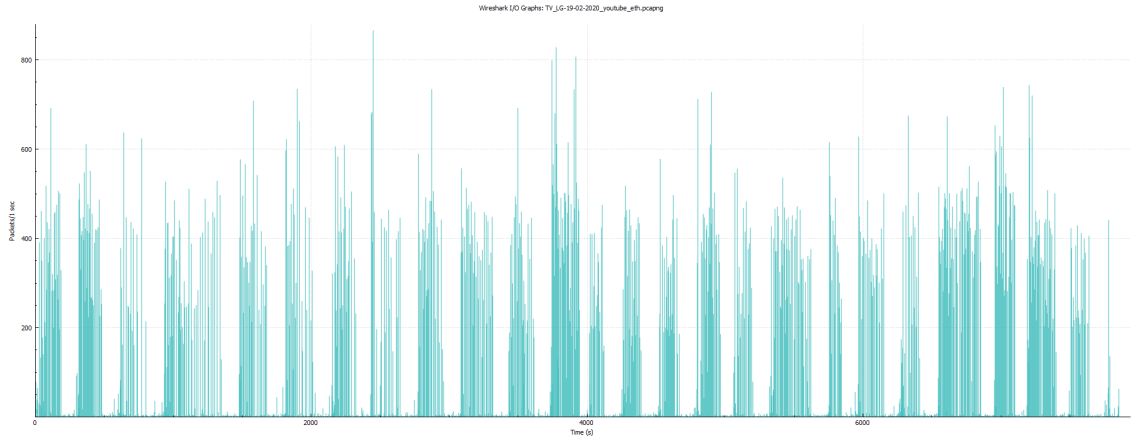


Figure 6.13: Stream 1 YouTube packet burst

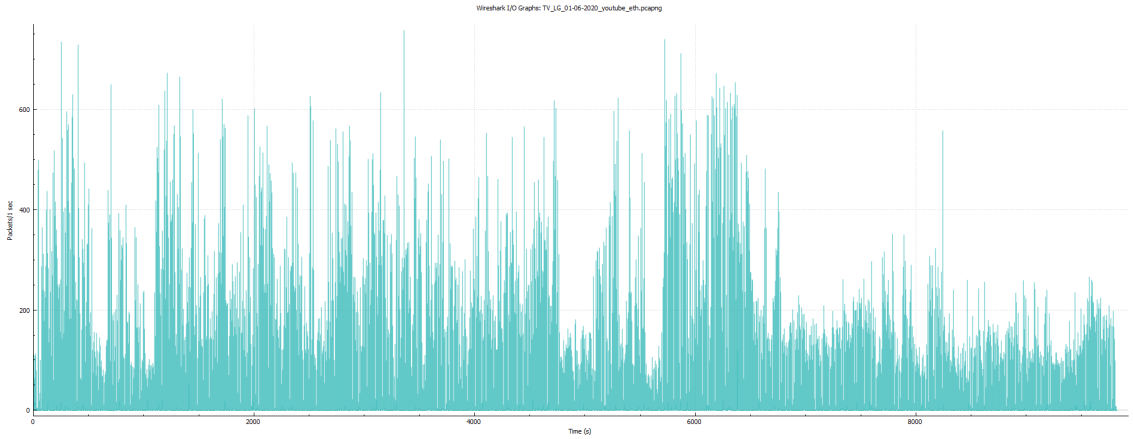


Figure 6.14: Stream 2 YouTube packet burst

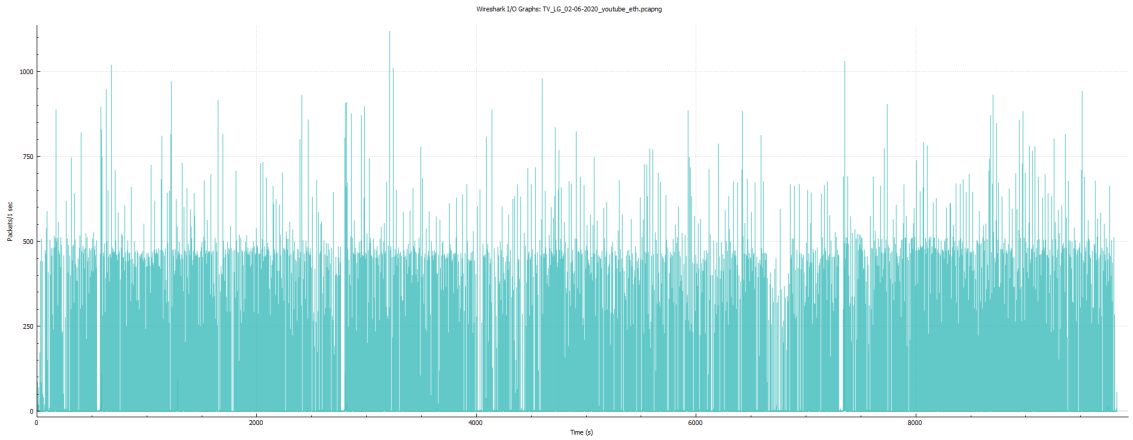


Figure 6.15: Stream 3 YouTube packet burst

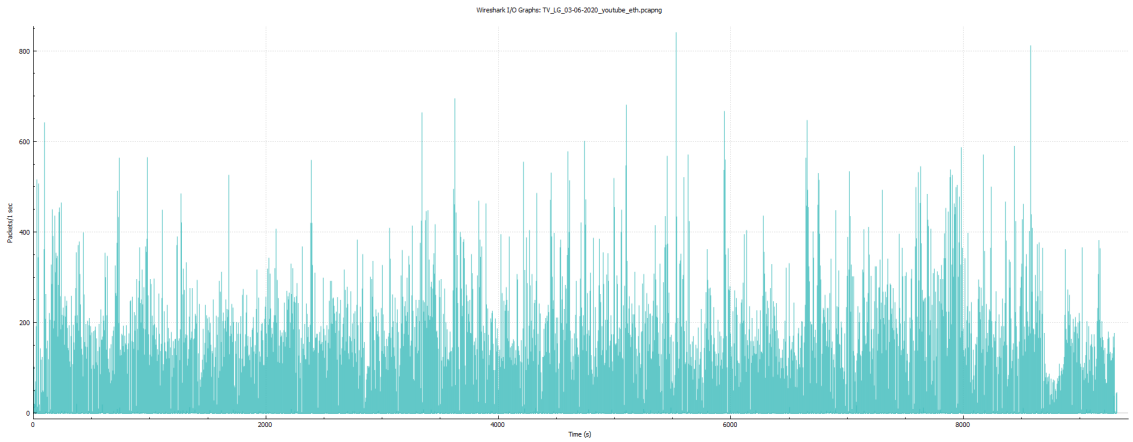


Figure 6.16: Stream 4 YouTube packet burst

Where we have different behavior from the different stream captures with no identifiable pattern. This is consistent with the previous analysis but makes it more difficult to detect in the network flow.

6.3 Smart TV 2 (Sony KD-49XE7096)

We now turn our attention to the other television. The Sony television came with Linux operating system and Opera as a web browser [42]. It has its own application store the "Opera TV" later becoming "VEWD TV Store" [43]. As is expected, this platform allows the download of applications, besides those that came from factory.

6.3.1 DNS

The DNS analysis shows no special interest. In some streams the DNS requests where the expected, for the manufacturer website and application website. There are a "no such name" response common to all streams, to the domain "ipv4-c002-opo001-meo-isp.1.oca.nflxvideo.net". These response happens at a very regular interval during all the stream and in all the streams, suggesting a behavior of the device and not from the application. Other then that, only a few "no such name" answers from the YouTube streams and related to YouTube or Google sites.

6.3.2 HTTP POST

This television also has posts from other devices regarding DLNA, and originated in a computer in the network. This behavior appears in all the streams.

6.3.3 Netflix

Here again, we start with the Netflix Smart TV application and the same analyzing pattern. As before we start showing the general characteristics of the streams in Table 6.3:

Table 6.3: Smart TV 2 Netflix streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	3521	02:20:21	8421,410	3487346327	414	1019787	121,1
Stream 2	13000	05:43:42	20622,517	12957267092	628	3610858	175,1
Stream 3	5790	04:00:10	14410,955	5734071568	397	1665899	115,6
Stream 4	3334	02:15:16	8116,226	3299346400	406	1035612	127,6

Packet percentage analysis

The packet size analysis according to their percentage, gave the following graphic results, as for Fig 6.17:

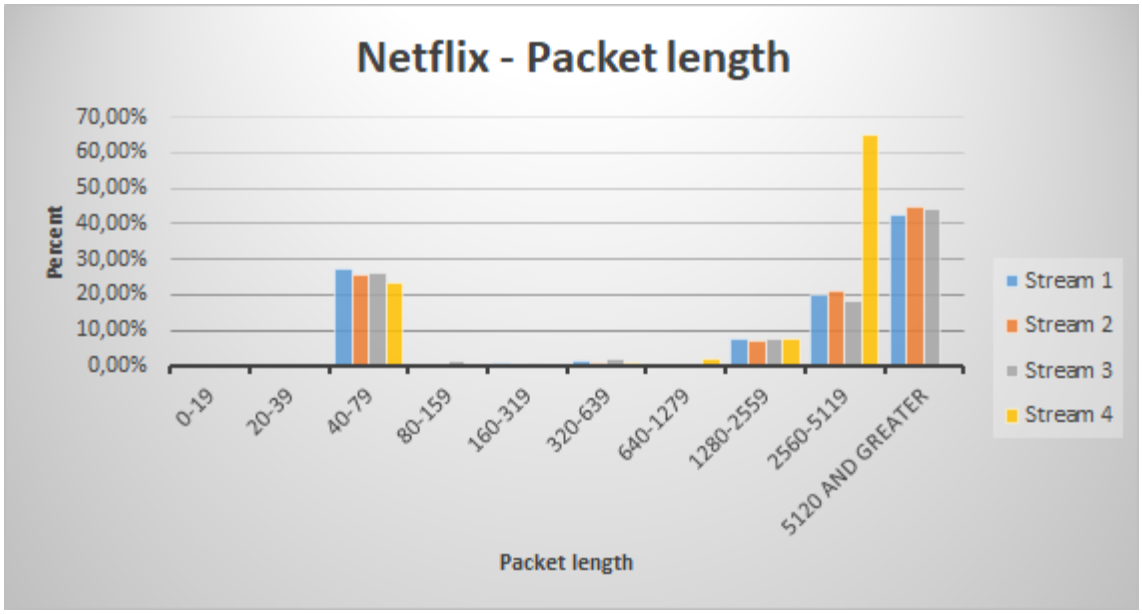
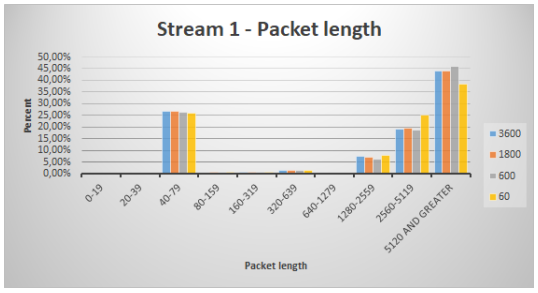


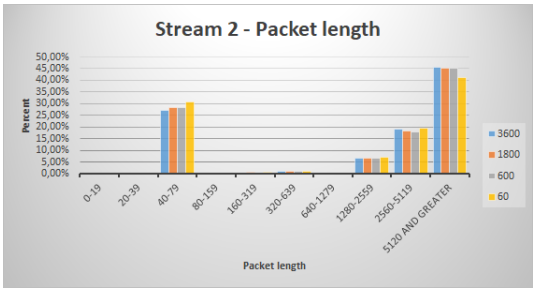
Figure 6.17: Netflix packet percentage

Here again the same pattern is observed. A high percentage, and with similar values, of packages of the same sizes, as from the other TV, and very few or no packages from the remaining ones. This may indicate a typical behavior of the application.

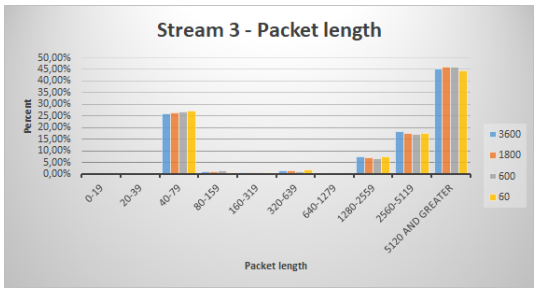
Comparing the different stream as for time slices, we get the following results, showed in Fig 6.18 (a), (b), (c), (d):



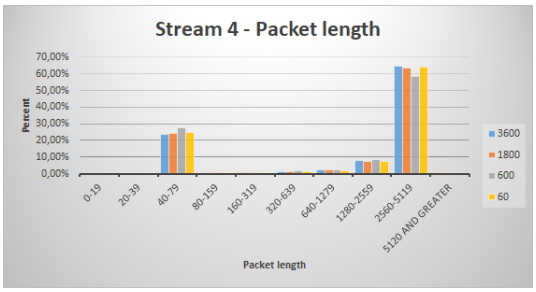
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.18: Smart TV 2 Netflix packet percentage in time frames

Again, what it seems to be a pattern of the Netflix application, we can see very similar behavior, whether in the same stream with different time frames, or across the different

streams. The same packet sizes with very close percentage.

Packet burst analysis

Now let us look at the packet burst of the Netflix application. Starting by the comparison of the different streams, according to Fig 6.19:

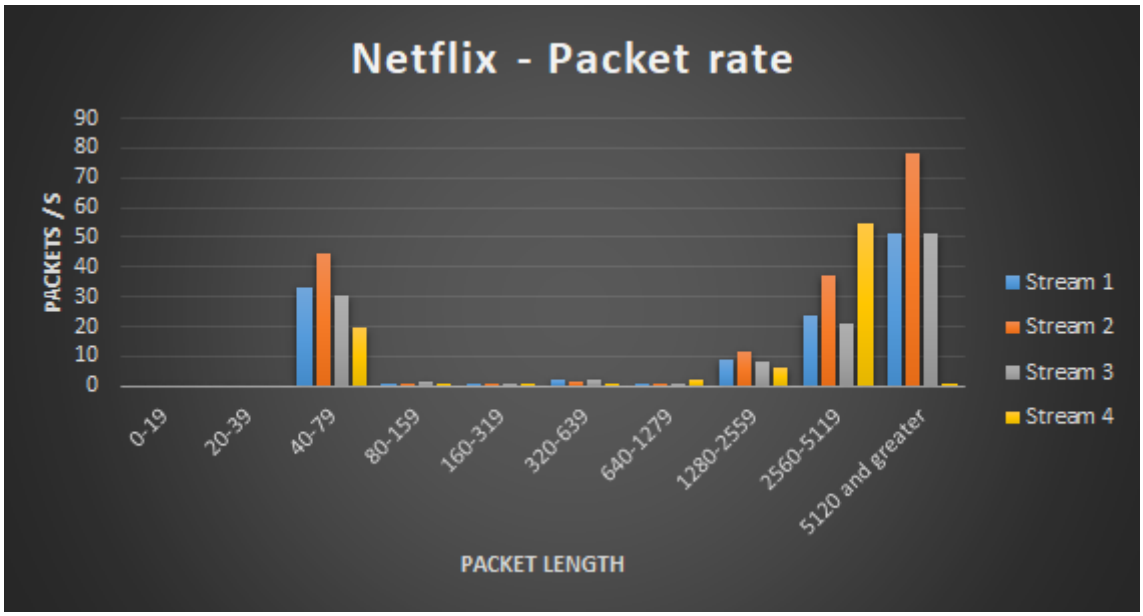
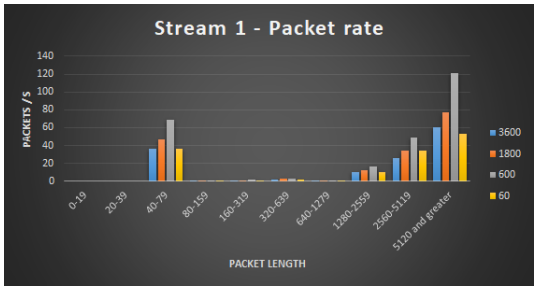


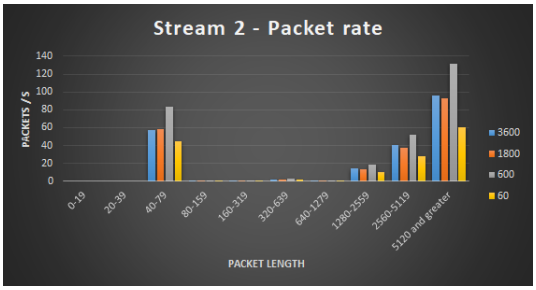
Figure 6.19: Netflix packet rate

Here we keep seeing the same behavior with similar packet rates for the same packet size groups.

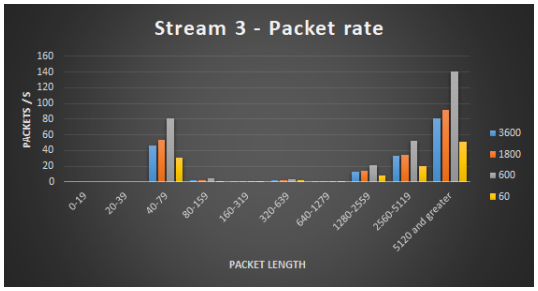
Keeping the same analysis sequence let us see the behavior of the different packets in time slices, showed in Fig 6.20 (a), (b), (c), (d):



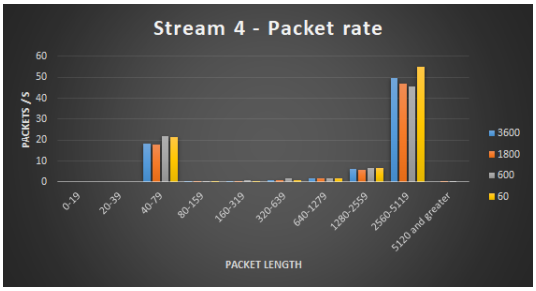
(a) Stream 1 packet rate



(b) Stream 2 packet rate



(c) Stream 3 packet rate



(d) Stream 4 packet rate

Figure 6.20: Smart TV 2 Netflix packet rate in time frames

Again the same pattern showing what is becoming the regular Netflix pattern. The same packet size groups and having similar burst rate in the same stream.

We turn our attention now to the temporal packet burst graphics of the Netflix streams. This is showed in Figs 6.21, 6.22, 6.23 and 6.24:

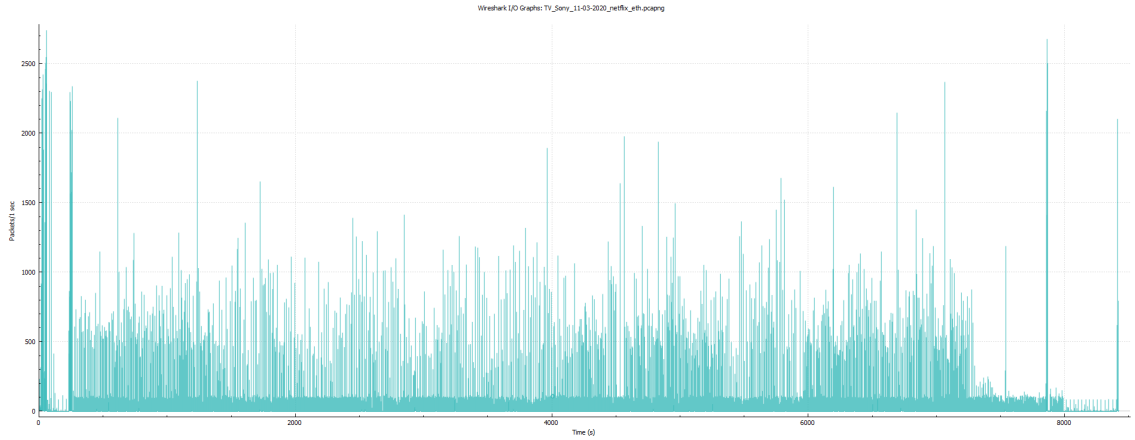


Figure 6.21: Stream 1 Netflix packet burst

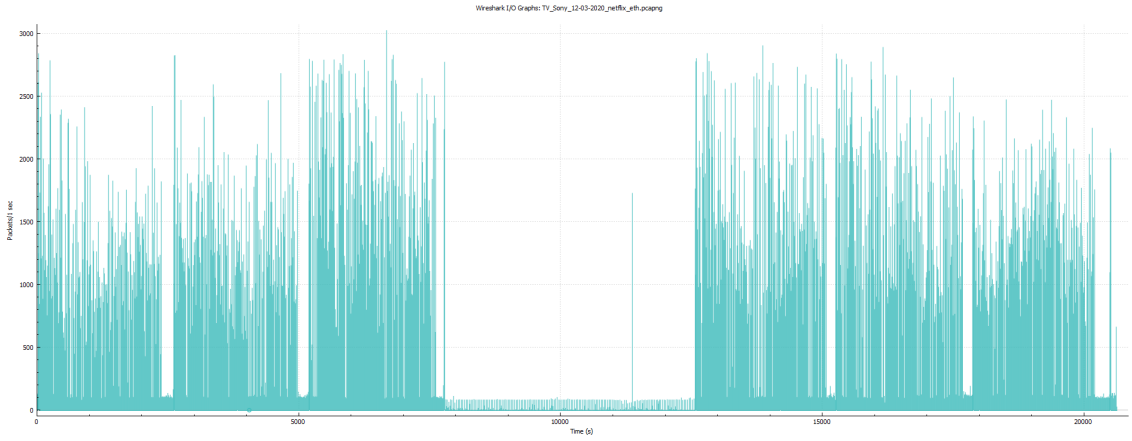


Figure 6.22: Stream 2 Netflix packet burst

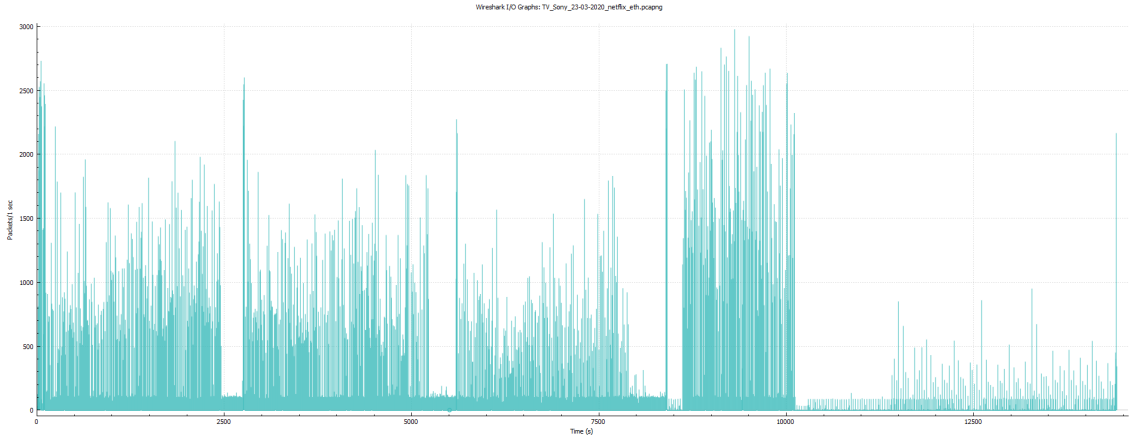


Figure 6.23: Stream 3 Netflix packet burst

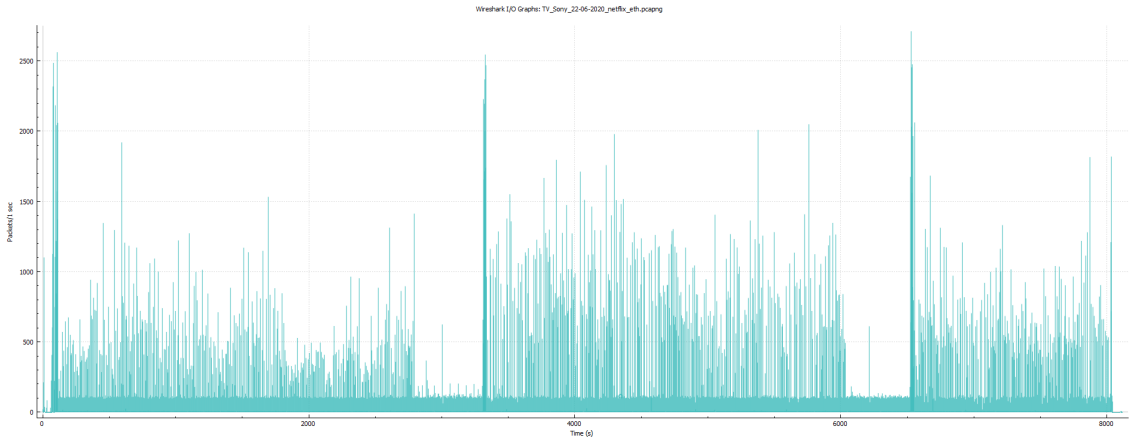


Figure 6.24: Stream 4 Netflix packet burst

This device shows different patterns for the different captures although the bit rate and packet rate are not very different among them.

6.3.4 YouTube

Now let us look at the YouTube packet streams. As usual, we start by showing the table with the stream general data, provided in Table 6.4:

Table 6.4: Smart TV 2 YouTube streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	2068	04:05:42	14742,300	2038526326	138	897159	60,9
Stream 2	1346	03:43:40	13420,217	1328056022	98	551419	41,1
Stream 3	1542	02:44:13	9853,866	1522572576	154	584891	59,4
Stream 4	1609	02:54:51	10491,850	1589209559	151	584457	55,7

Packet percentage analysis

Start by the analysis of the packet size percentage. The results are in the graphic presented in Fig 6.25:

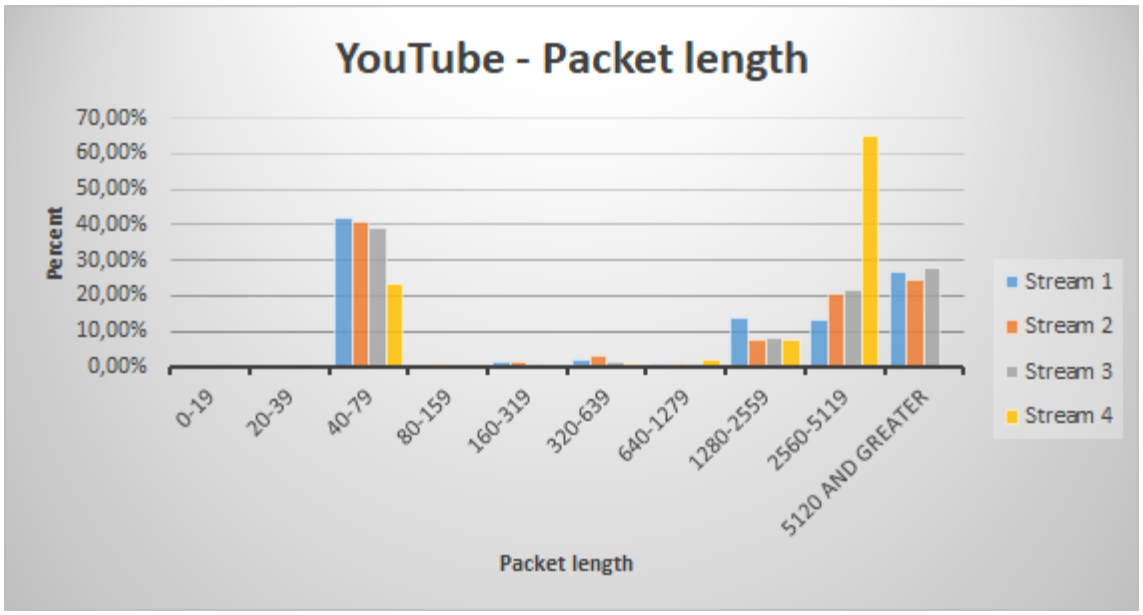
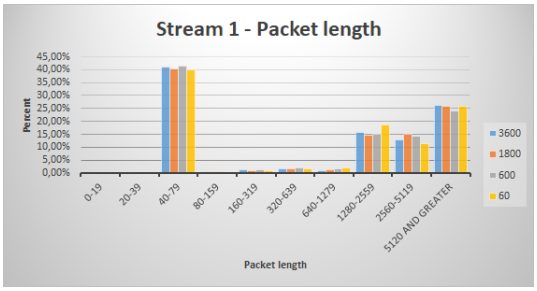


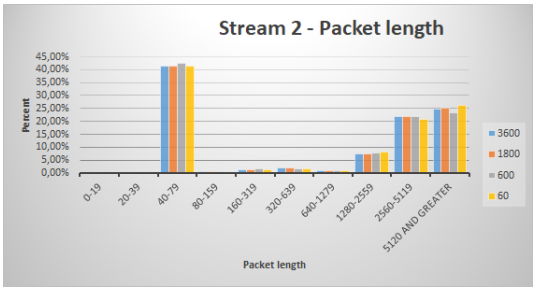
Figure 6.25: YouTube packet graphic

It is possible to see a more regular pattern then the previous YouTube application. Still the same groups of packet size, but with a more close percentage of each in the different streams.

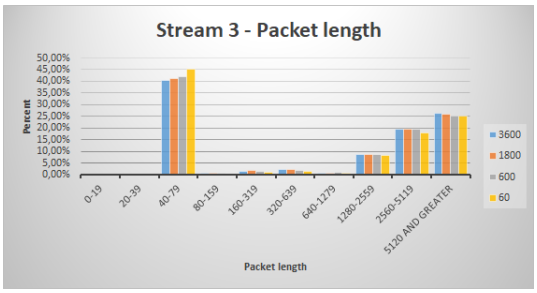
Following is the graphics of each stream in time frames, as been the practice in this work. It is showed in Fig 6.26 (a), (b), (c), (d):



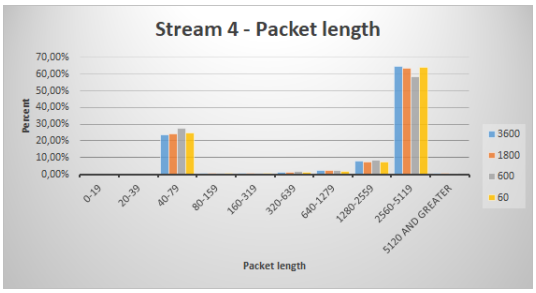
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.26: Smart TV 2 YouTube packet percentage in time frames

The YouTube application in this device as a much more consistent behavior then in the previous one. The packet percentage of each stream remains very close in the different time frames and also across the streams. This may help in defining a YouTube network profile.

Packet burst analysis

Continuing the analysis of the YouTube application, let us turn our attention to the packet rate starting with the comparison of the different streams. We can see the graphic in Fig 6.27:

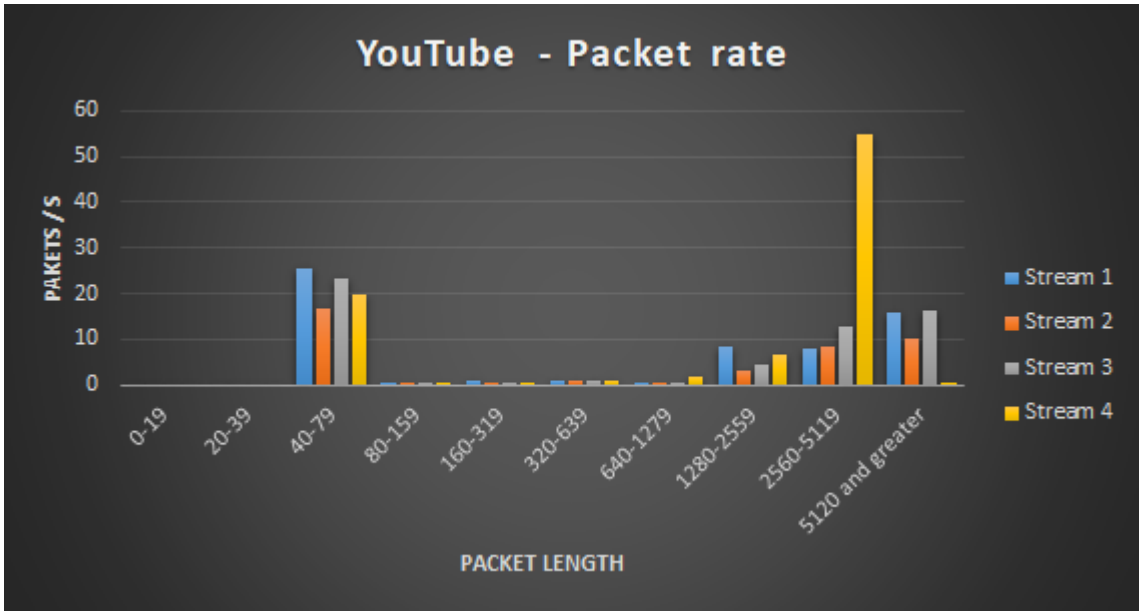
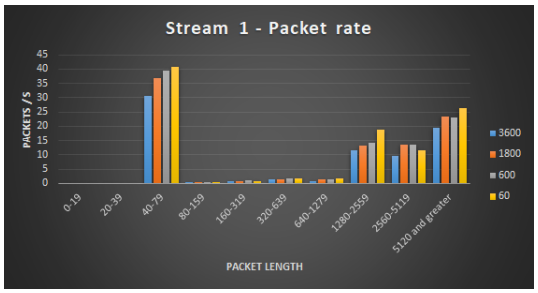


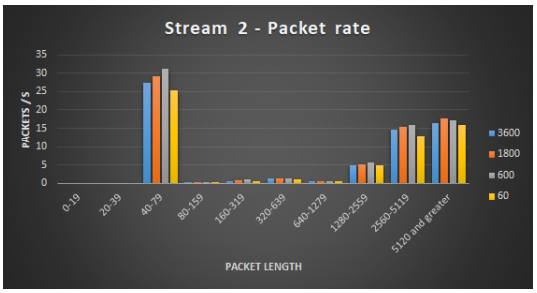
Figure 6.27: YouTube packet graphic

The already known pattern keeps showing in this graphic.

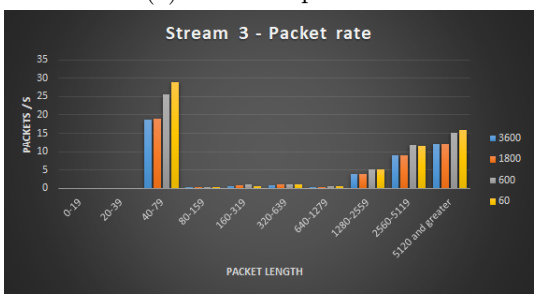
Looking now at the packet rate of each stream in the different time frames showed in Fig 6.28 (a), (b), (c), (d):



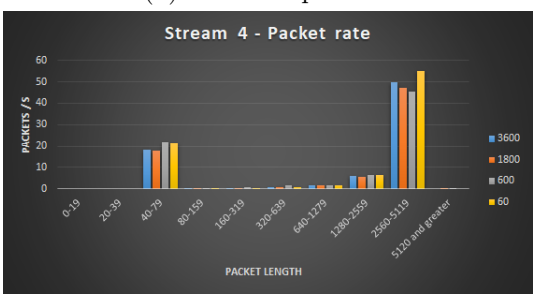
(a) Stream 1 packet rate



(b) Stream 2 packet rate



(c) Stream 3 packet rate



(d) Stream 4 packet rate

Figure 6.28: Smart TV 2 YouTube packet rate in time frames

As can be clearly seen, the pattern keeps consistent in the time frames, as expected.

Analyzing now the temporal burst rate presented in Figs 6.29, 6.30, 6.31 and 6.32:

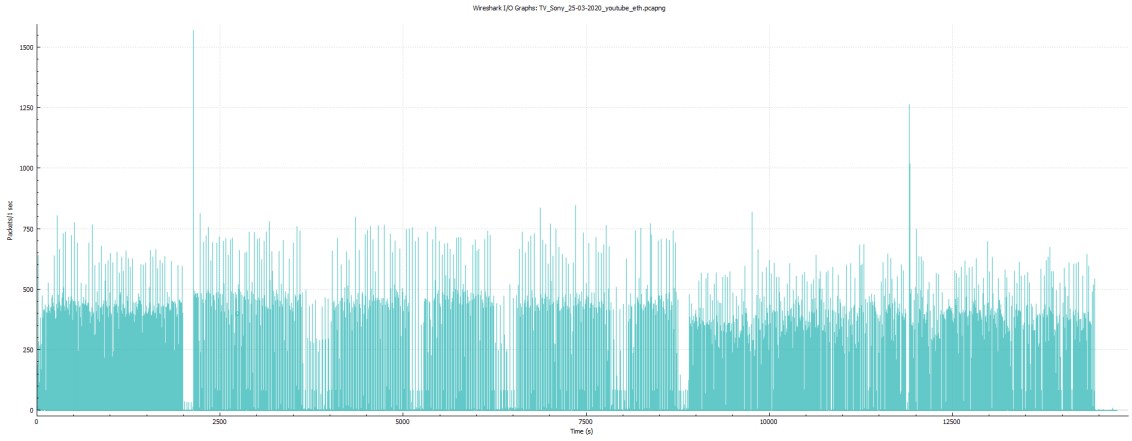


Figure 6.29: Stream 1 YouTube packet burst

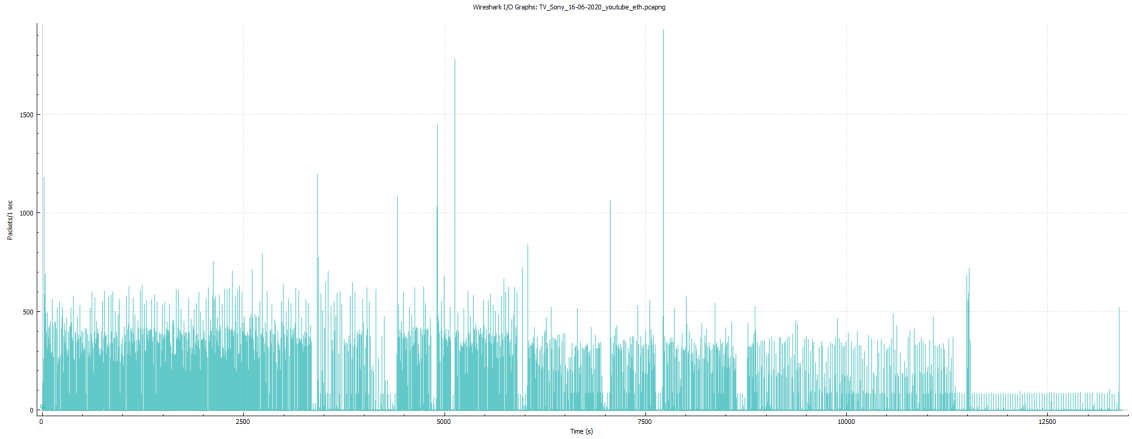


Figure 6.30: Stream 2 YouTube packet burst

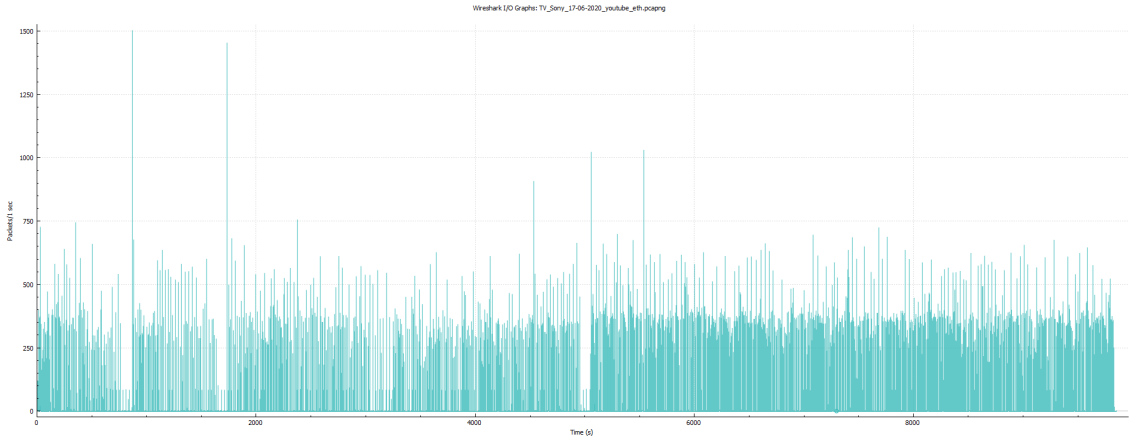


Figure 6.31: Stream 3 YouTube packet burst

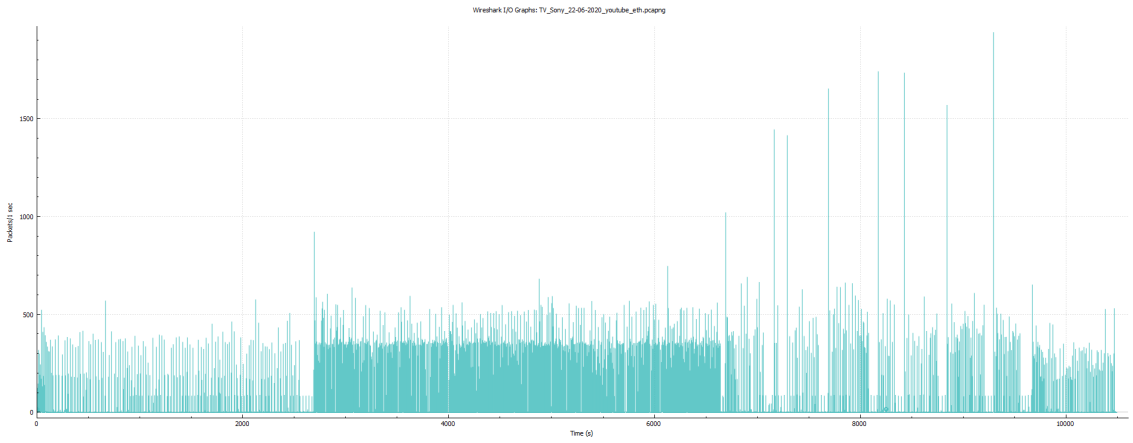


Figure 6.32: Stream 4 YouTube packet burst

The graphics shows a similar pattern along the time and compared with each other.

6.4 Blu-ray Disc Player (Sony BDP-S3700)

This time we are looking at a different device. This BD reader come with a big set of pre-installed applications, but the YouTube application had to be added.

6.4.1 DNS

In this device all DNS requests appears to be legitimate. It starts by doing a top domain query, followed by queries to brand sites, and finally to the chosen application sites. One of the YouTube streams has some "no such name" responses related with YouTube and Google sites, similar to the ones of the Smart TV 2. Both devices are from the same manufacturer so this is not an unusual behavior.

6.4.2 HTTP POST

Looking at the POST requests we can see different patterns for the two applications. The Netflix send a stream with what looks like user information because it includes a cookie with the same ID in all of the captures, and with encrypted data in JSON format. As for the YouTube, it sends a packet to a google domain with a Online Certificate Status Protocol (OCSP) request. The OCSP protocol is used to check revoked X.509 digital certificates. So, the YouTube application is checking for the validity of some X.509 digital certificate.

6.4.3 Netflix

Here again, we start with the Netflix smart device application and using the same analyzing pattern. First, let us look at the general stream data, showed in Table 6.5:

Table 6.5: Blu-Ray Disc Player Netflix streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	9924	05:13:19	18799,777	9819989400	522	3075939	163,6
Stream 2	2944	02:20:38	8438,202	2911481966	345	973333	115,3
Stream 3	5831	03:01:21	10881,193	5771229458	530	1758558	161,6
Stream 4	2058	02:04:18	7458,988	2034415659	272	701081	94,0

Packet percentage analysis

The packet percentage graphic is following in Fig 6.33:

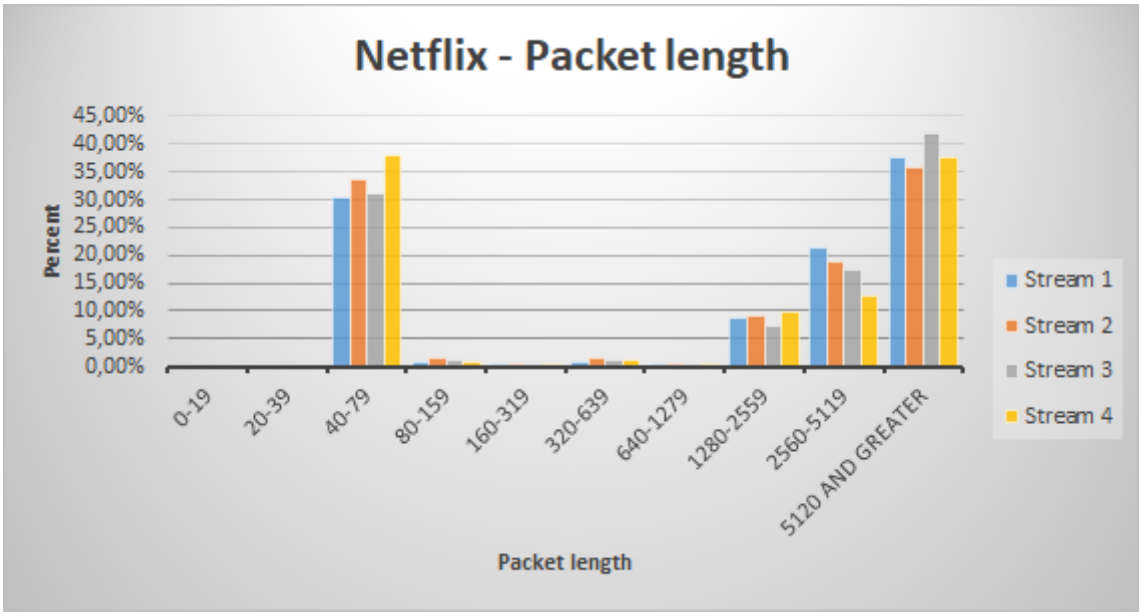
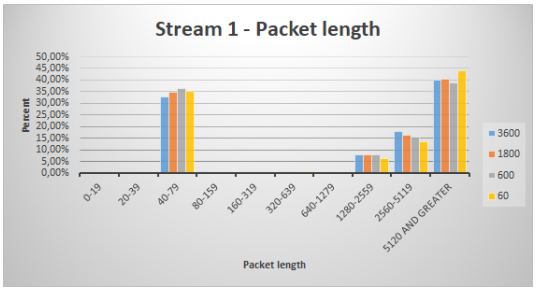


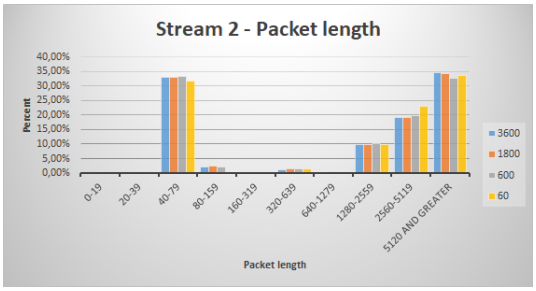
Figure 6.33: Netflix packet graphic

The graphic shows a already familiar pattern of the packet percentage according to packet size. Since this is the third device with similar pattern, it may be assumed that this is the typical Netflix behavior.

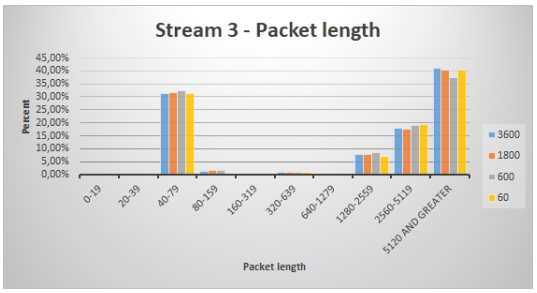
As been the rule, let us now see the streams time frames, presented in Fig 6.34 (a), (b), (c), (d):



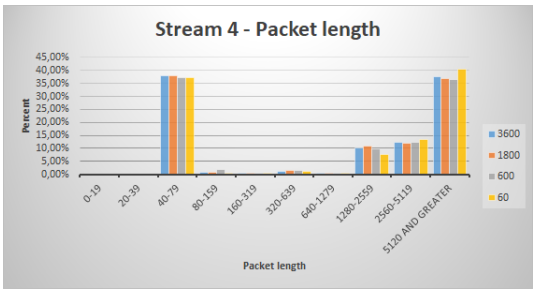
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.34: Blu-Ray Disc Player Netflix packet percentage in time frames

Not surprising, the patterns are consistent across the different time frames and the different streams, also supporting our claim that this is the typical Netflix smart device application.

Packet burst analysis

Now following with the packet burst analysis showed in Fig 6.35:

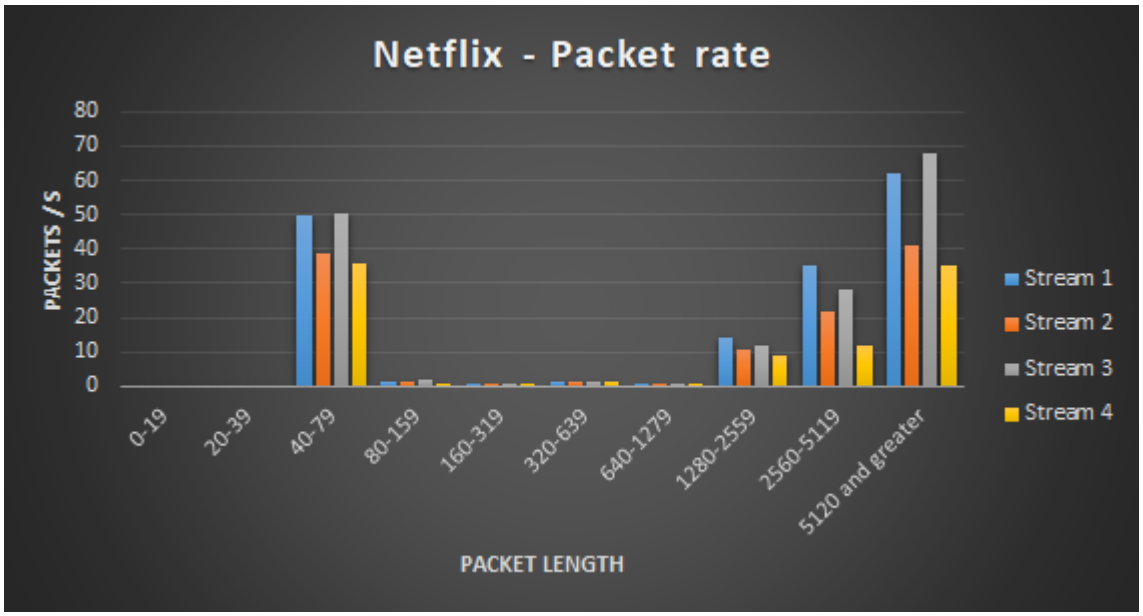
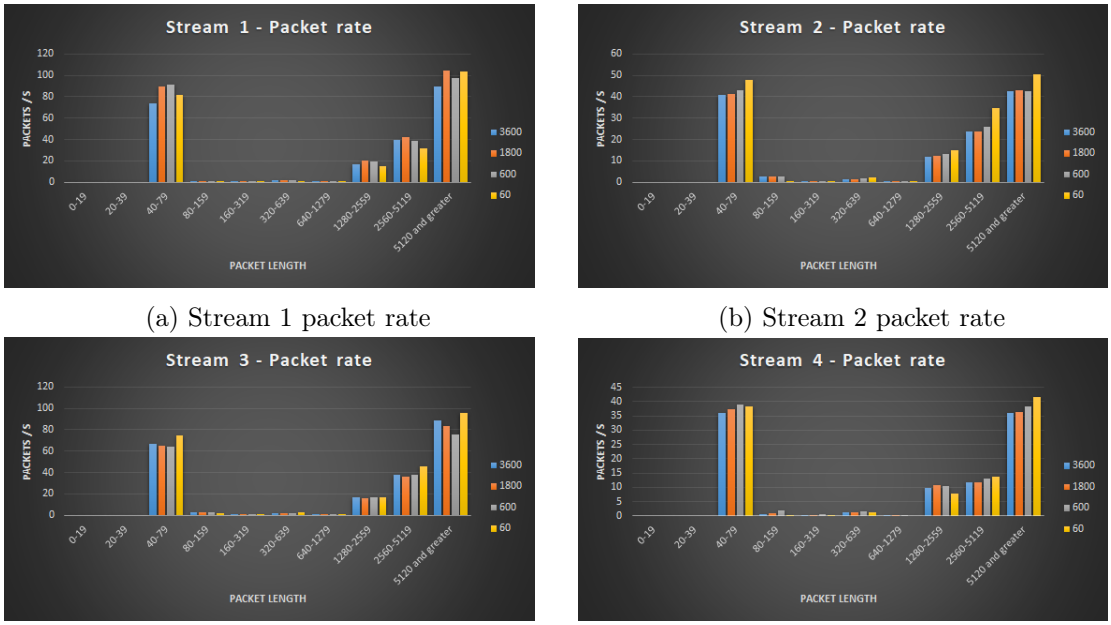


Figure 6.35: Netflix packet graphic

The already familiar pattern keeps appearing as expected.

Looking at the streams as per time frame in Fig 6.36 (a), (b), (c), (d):



(a) Stream 1 packet rate

(b) Stream 2 packet rate

(c) Stream 3 packet rate

(d) Stream 4 packet rate

Figure 6.36: Blu-Ray Disc Player Netflix packet rate in time frames

This last analysis of the Netflix packet stream confirms what we have found previously. There is a consistent pattern found in packet sizes of the streams the application produces.

Finally the packet burst along the complete stream time frame, are showed in Figs 6.37, 6.38, 6.39 and 6.40:

Figure 6.37: Stream 1 Netflix packet burst

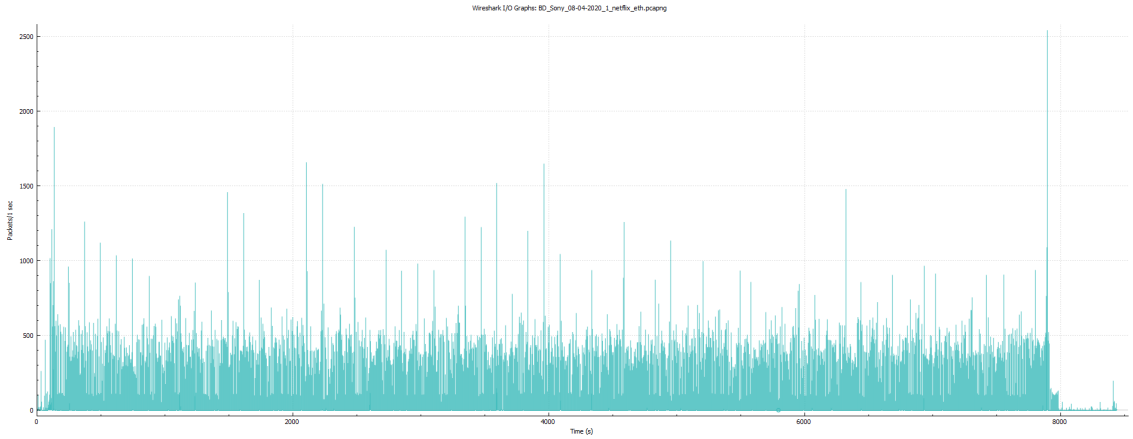


Figure 6.38: Stream 2 Netflix packet burst

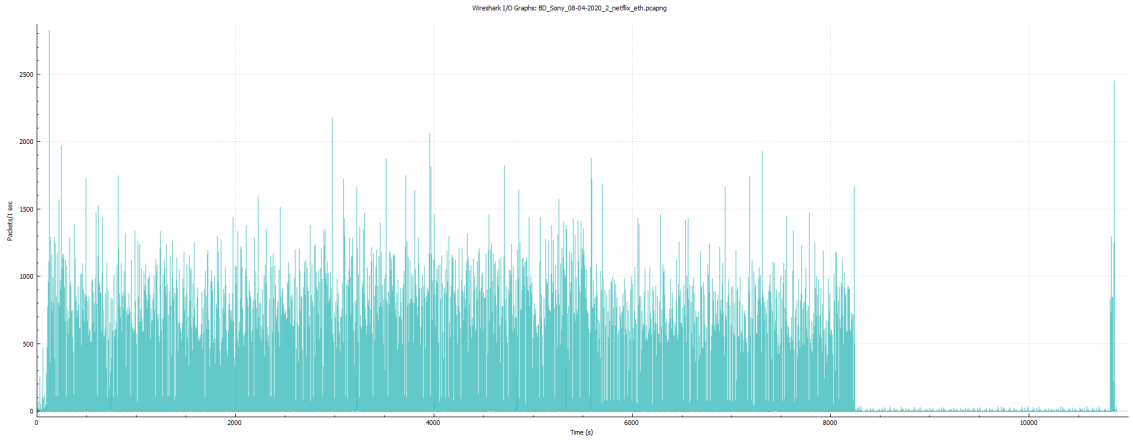


Figure 6.39: Stream 3 Netflix packet burst

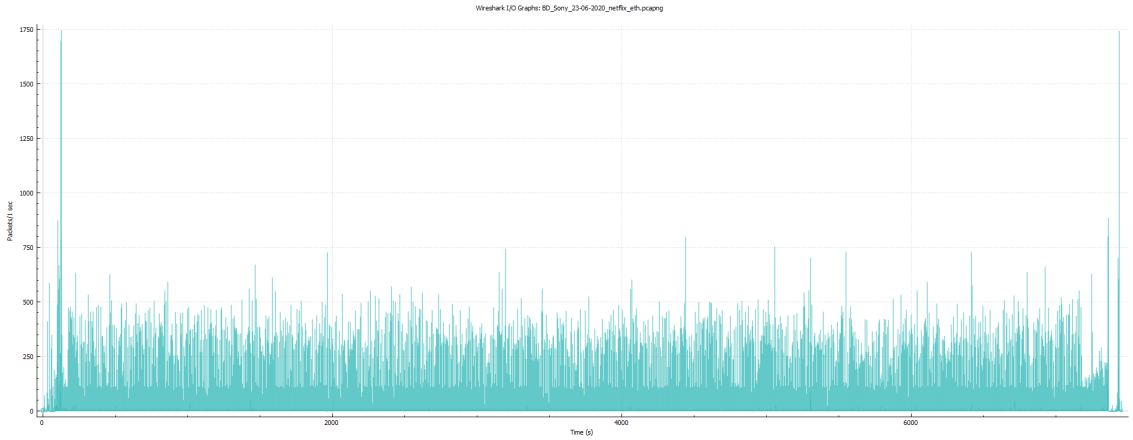


Figure 6.40: Stream 4 Netflix packet burst

These packet burst are very irregular not allowing for any conclusion.

6.4.4 YouTube

And now the analysis of the YouTube packet streams. First, the streams general characteristics presented in Table 6.6:

Table 6.6: Blu-Ray Disc Player YouTube streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	1113	03:39:04	13144,407	1095114156	83	537537	40,9
Stream 2	504	02:37:10	9430,811	495902655	52	246857	26,2
Stream 3	1029	02:20:07	8407,845	1012377711	120	489574	58,2
Stream 4	1371	02:04:09	7449,182	1350053074	181	629267	84,5

Packet percentage analysis

Doing the packet percentage analysis, the results are showed in Fig 6.41:

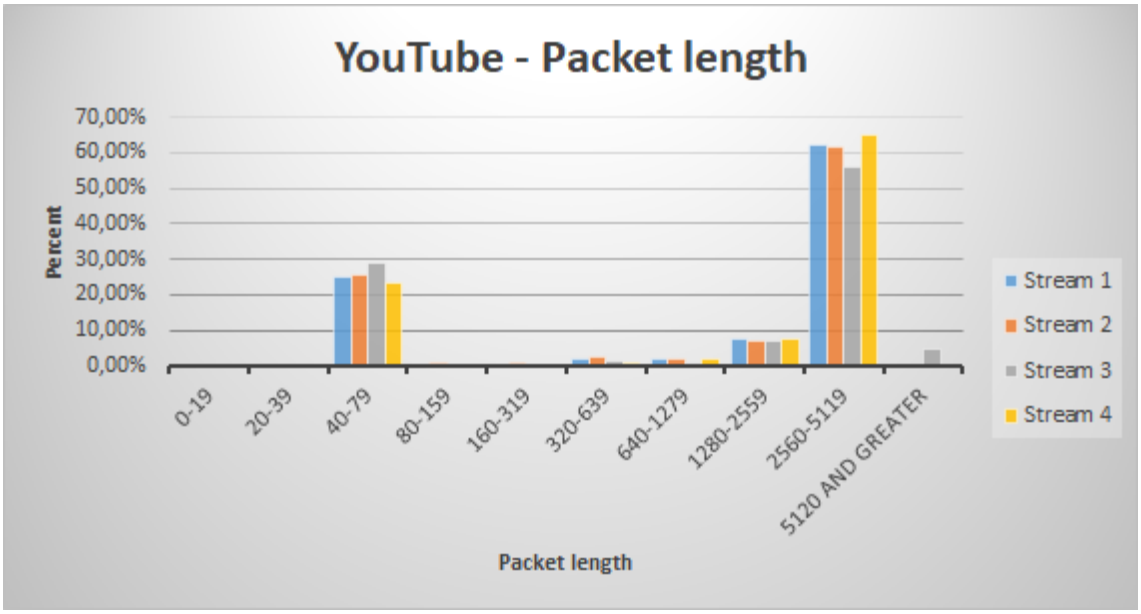
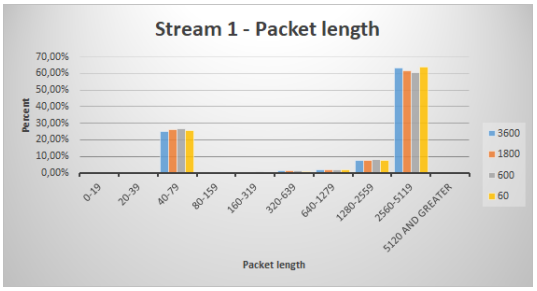


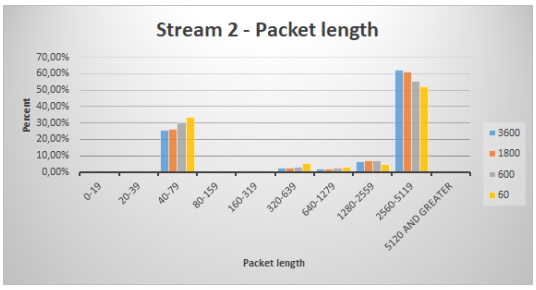
Figure 6.41: YouTube packet graphic

We can see a very regular pattern along all of the streams.

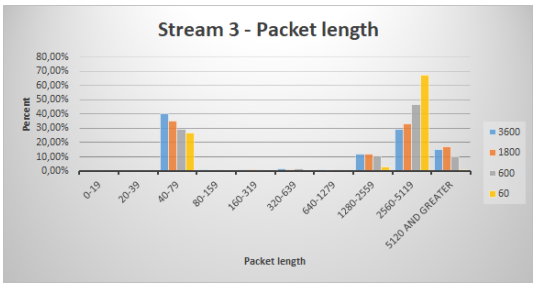
Next, the packet percentage according to the different time frames, are presented in Fig 6.42:



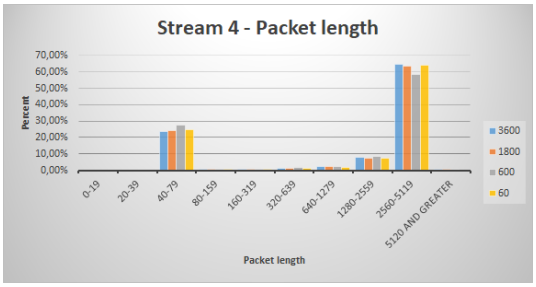
(a) Stream 1 packet percentage



(b) Stream 2 packet percentage



(c) Stream 3 packet percentage



(d) Stream 4 packet percentage

Figure 6.42: Blu-Ray Disc Player YouTube packet percentage in time frames

These graphics confirm the common pattern showed in the previous graphic.

Packet burst analysis

Doing now the analysis in the packet rate perspective, in Fig 6.43:

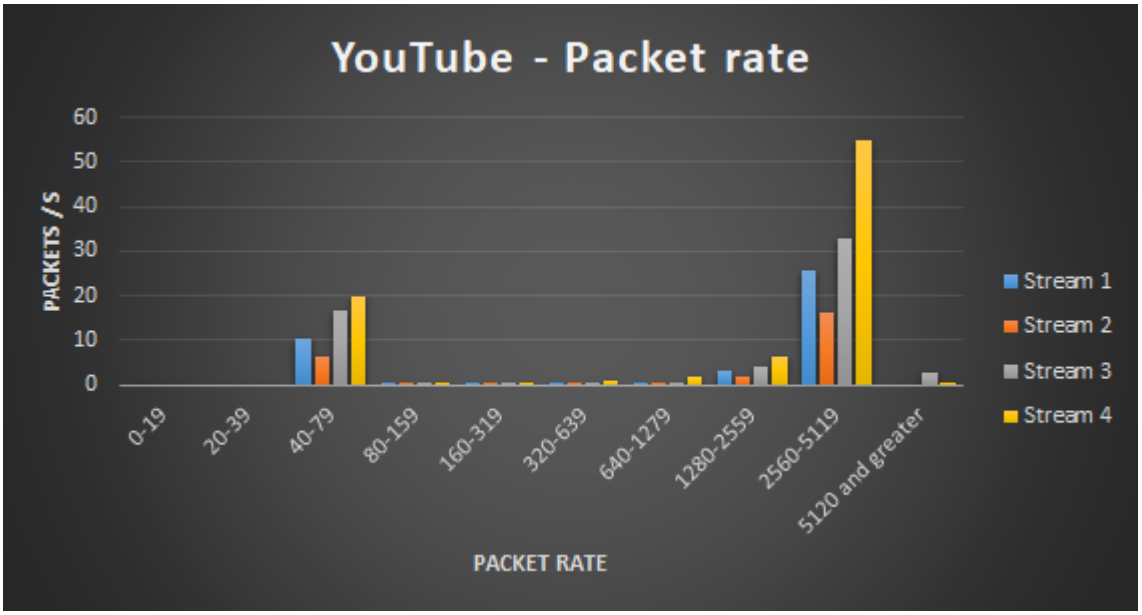


Figure 6.43: YouTube packet graphic

The streams have similar behavior, although with small differences from one another.

Lastly, the analysis of the streams according to time frames, showed in Fig 6.44 (a), (b), (c), (d):

(a) Stream 1 packet rate

(b) Stream 2 packet rate

(c) Stream 3 packet rate

(d) Stream 4 packet rate

Figure 6.44: Blu-Ray Disc Player YouTube packet rate in time frames

We can see the existence of a regular pattern in every stream, although with some variations.

Now seen the packet burst along the stream time line, presented in Figs 6.45, 6.46, 6.47 and 6.48:

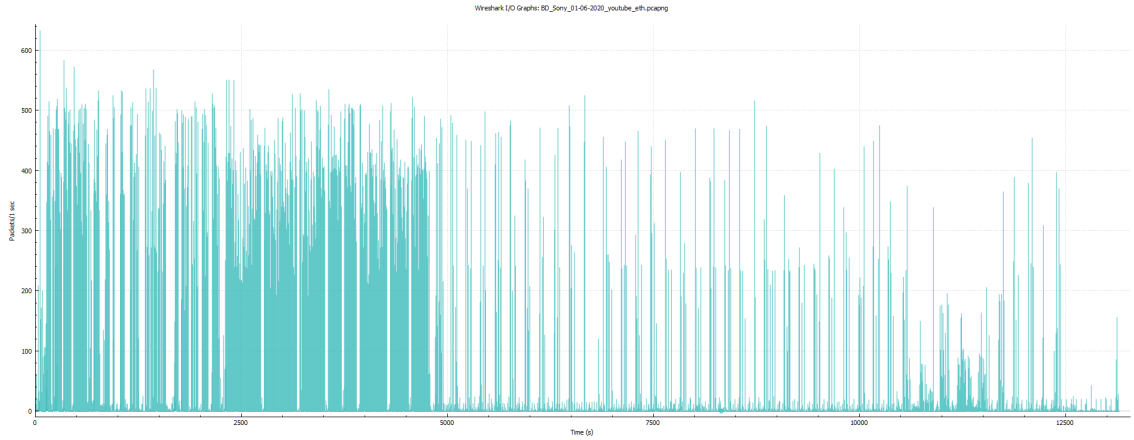


Figure 6.45: Stream 1 YouTube packet burst

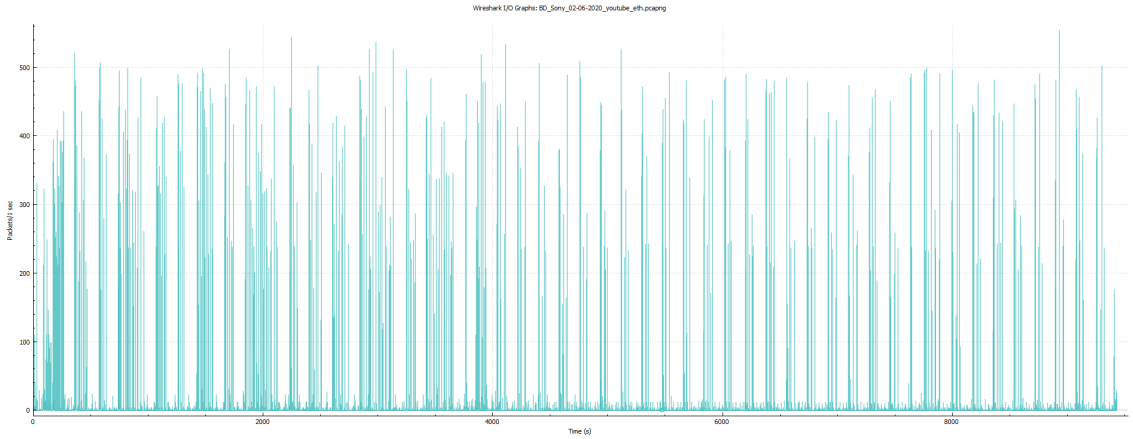


Figure 6.46: Stream 2 YouTube packet burst

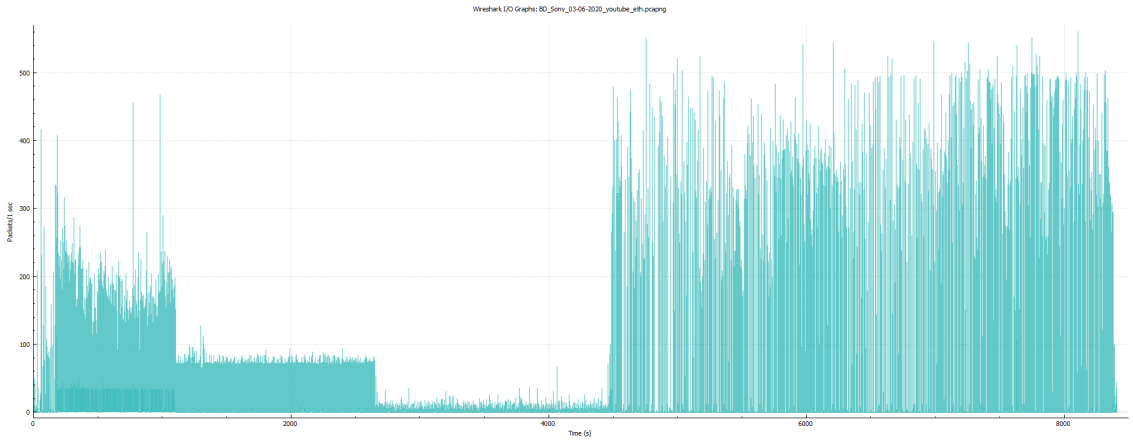


Figure 6.47: Stream 3 YouTube packet burst

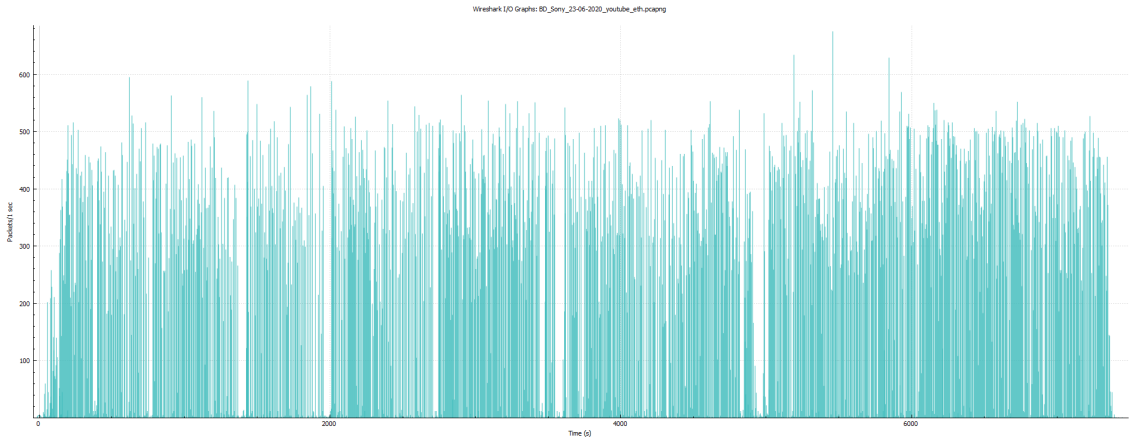


Figure 6.48: Stream 3 YouTube packet burst

These are very irregular packet burst so they are of no use for an application profiling. We will have to rely on the previous analysis.

6.5 Set Top Box (Cisco KMM3010-pt)

There is not much information about this Set Top Box. In fact, only looking at the labels in the device, and searching the internet, we are able to find the device name, and some details. Most of the information is in online forums, so the reliability is not the best. What we can know about it is that it has the Windows CE OS. As stated before, the analysis on this device will be different, since it has no applications but only TV channel streaming.

6.5.1 Preliminary analysis

The preliminary analysis show an interesting situation. Although the device has only one network interface, it communicates in two different networks, using different IP address ranges. One of this addresses is the expected address for the home network, 192.168.1.65, while the other is of the range "10.0.0.0".

Methodology: Although we are not interested in analyzing external applications, we will keep using the same analysis rules, only that they will be applied to the two IP addresses in separate.

6.5.2 10.0.0.0 Network

In this network the device appears to behave only as a receiver. The sender uses different addresses suggesting the existence of different servers. The used server addresses are 10.159.224.200 and the range 10.173.5.51-55. All the destination addresses are in the range 232.17.8.15/16/44, 232.17.9.7, 232.21.11.162, 232.0.7.7, been all source-specific multicast addresses [44]. There is no apparent rule for the use of any of the addresses. All communications are using UDP protocol. As this is a private address range, not allowed on the internet, this is probably a network of the ISP for television broadcast. The sender is some server on the network and the receivers are the Set Top Boxes of the home users.

DNS

There is no DNS communications because they are all multicast communications.

HTTP POST

There is no HTTP communications since there is only UDP protocol packets.

Data stream analysis

This is the general stream data table, in Fig 6.7:

Table 6.7: Set Top Box streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	8536	20:33:12	73992,435	8270896	111	7917	0,1
Stream 2	20000	08:07:33	29253,348	20059805400	685	14645603	500,6
Stream 3	479	09:14:37	33277,560	467823368	14	342384	10,3
Stream 4	7099	02:29:06	8946,519	6927352681	774	5057625	565,3
Stream 5	13000	07:13:40	26020,074	13275899498	510	9692992	372,5
Stream 6	13000	09:31:16	34276,573	13259114319	386	9680840	282,4

Remark: The streams 1 and 3 are overnight captures with the Set Top Box turned off which explains the low size comparing with the capture total time.

This is the first graphic showing the packet percentage according to their size, presented in Fig 6.49:

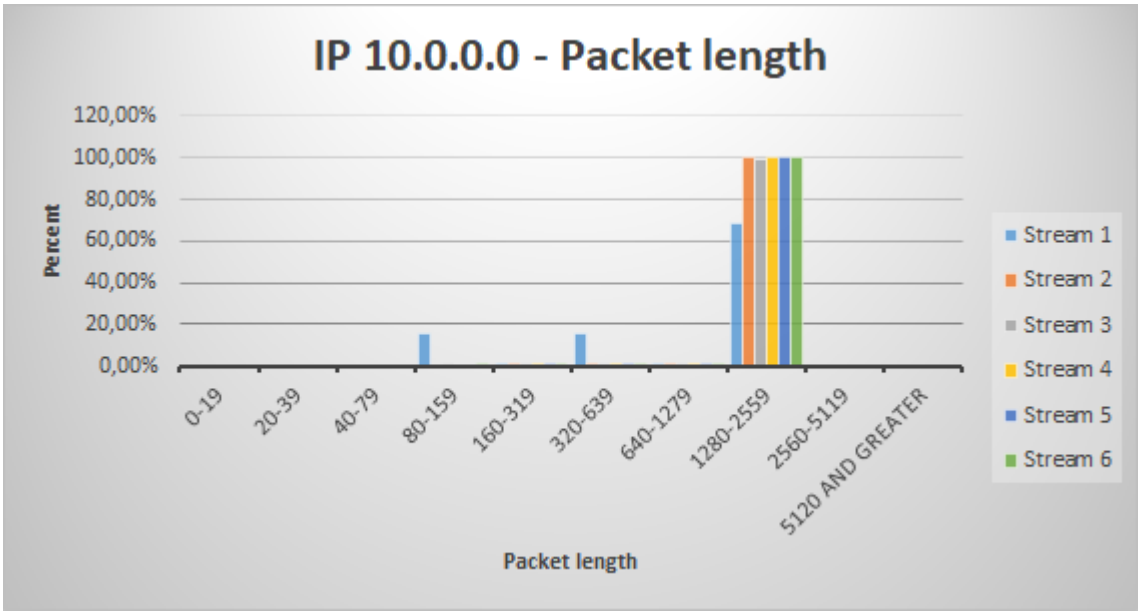
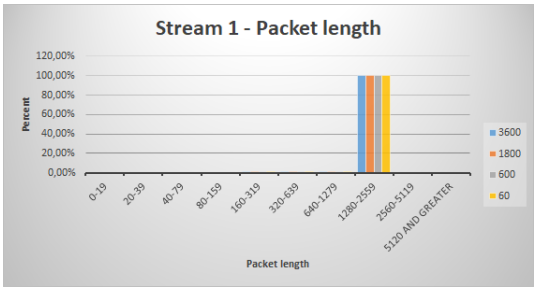


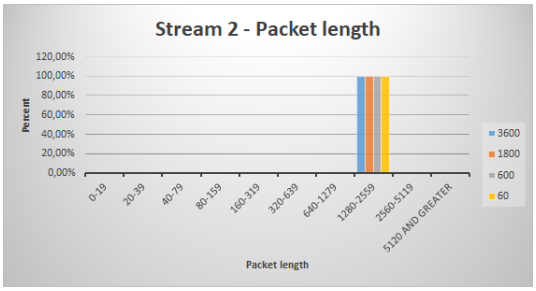
Figure 6.49: IP 10.0.0.0 packet percentage

We can see that, apart from stream 1 and 3, all the packets are of size 1280-2559. The stream 1 as also a bigger percentage of packets of this size but also some of 80-159 and 320-639. Stream 3 bar does not quit reach 100%, but we cannot spot where the remaining packets are.

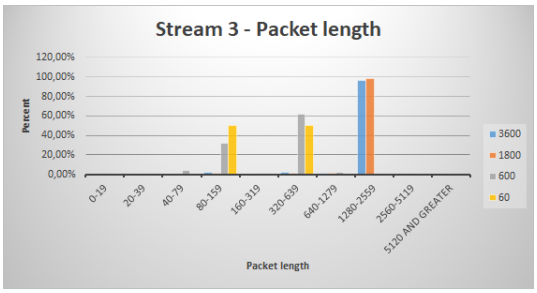
Let us look now at each stream in more detail using the defined time frames. This is showed in Fig 6.50 (a), (b), (c), (d), (e), (f):



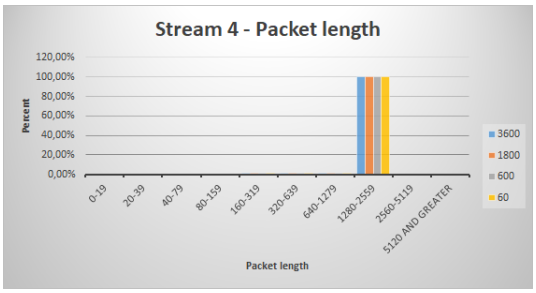
(a) Stream 1 packet percentage



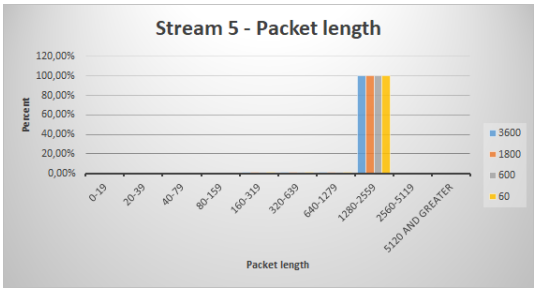
(b) Stream 2 packet percentage



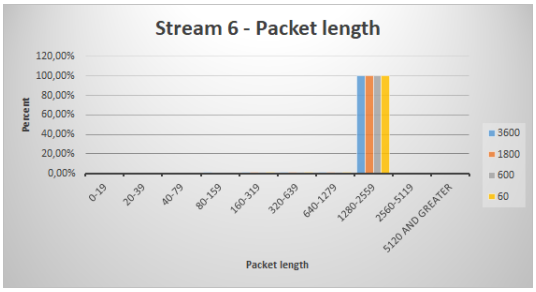
(c) Stream 3 packet percentage



(d) Stream 4 packet percentage



(e) Stream 5 packet percentage



(f) Stream 6 packet percentage

Figure 6.50: IP 10.0.0.0 stream graphics

The stream 1 as 100% in all time frames, which means that the packet sizes showed in the previous graphic are beyond 3600 sec (1 hour) of broadcasting. In stream 3 we can see more clearly where the packets not in the size 1280-2559 are and when they appears. Apart from the previously mentioned, these streams are very regular in packet size, almost are from the group 1280-2559. Been this a TV broadcasting streams in a ISP internal network, it is not a uncommon behavior;

Packet burst analysis

It is time now to look at the packet burst of the streams. This is showed in Fig 6.51:

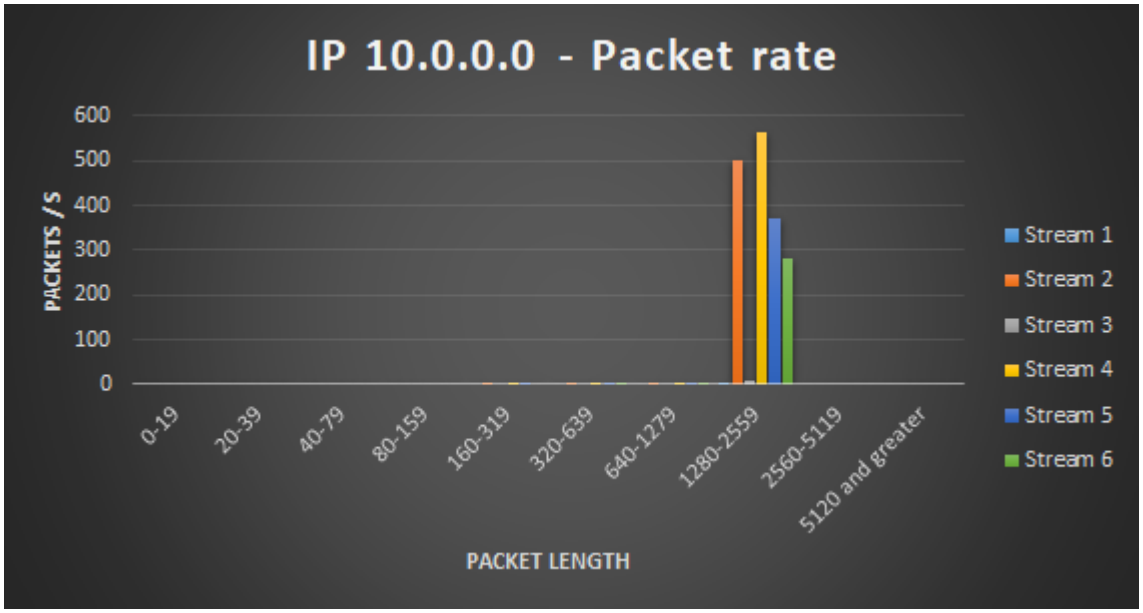
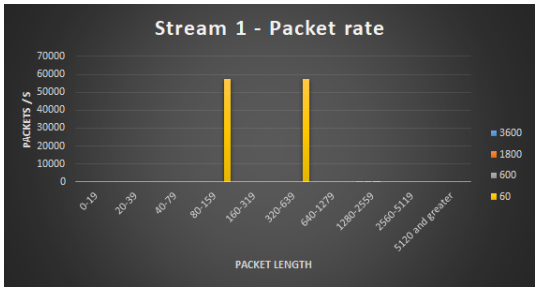


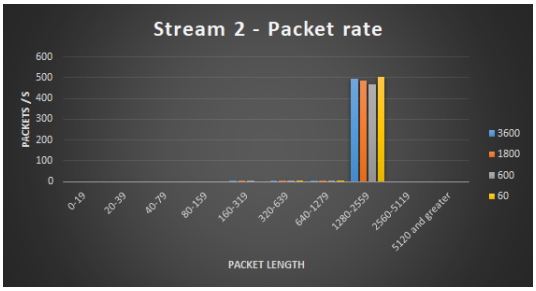
Figure 6.51: IP 10.0.0.0 packet graphic

The packet burst is very different from one stream to another as can be seen but all in the expected range.

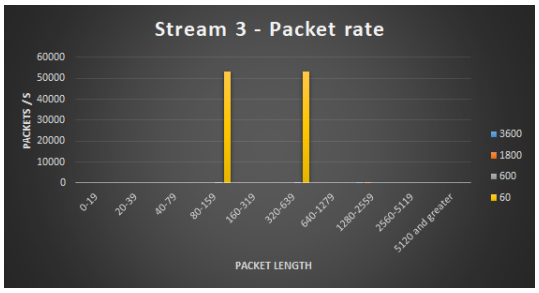
We turn now our attention to the different streams in the chosen time frames, in Fig 6.52 (a), (b), (c), (d), (e), (f):



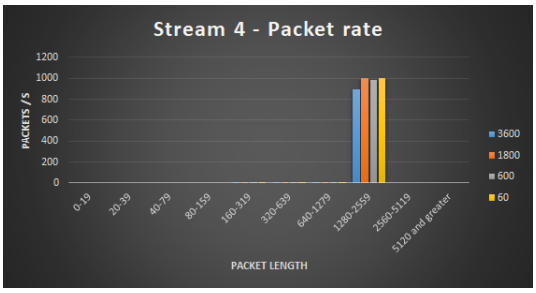
(a) Stream 1 packet rate



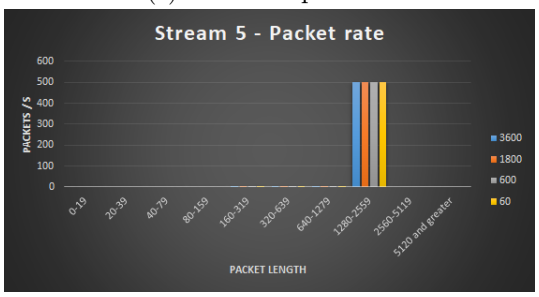
(b) Stream 2 packet rate



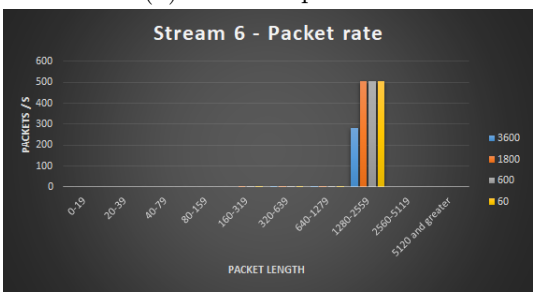
(c) Stream 3 packet rate



(d) Stream 4 packet rate



(e) Stream 5 packet rate



(f) Stream 6 packet rate

Figure 6.52: IP 10.0.0.0 stream graphics

This graphics shows some different behavior in some streams that deserves a deeper look. Namely the streams 1 and 3 shows only a visible stream rate in the one minute time frame. This is because the frames are sent in a regular interval with periods of no packets between them. As we enlarge the time frame these "no packet period" made the frame rate goes down to a very small number. The other streams have the expected behavior.

Next, it is showed the packet burst along the time, in Figs 6.53, 6.54, 6.55 and 6.56, 6.57 and 6.58:

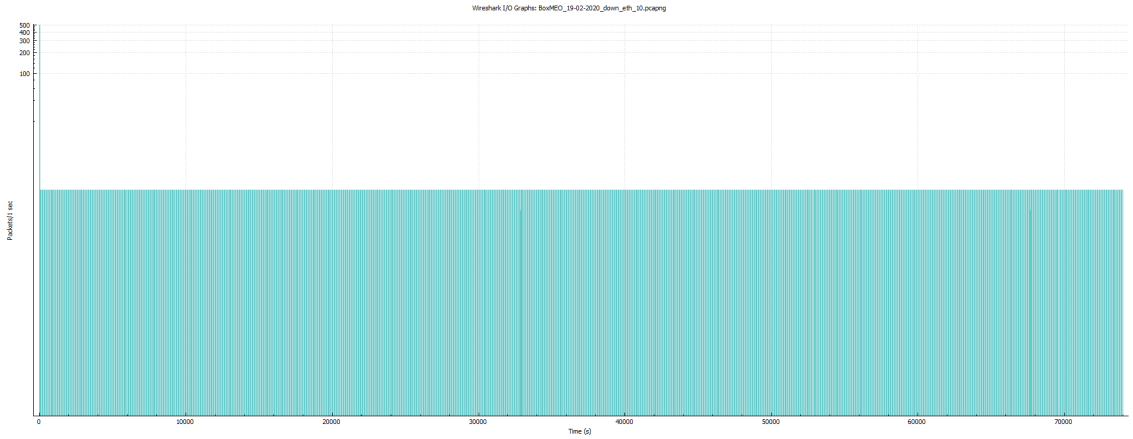


Figure 6.53: Stream 1 packet graphic

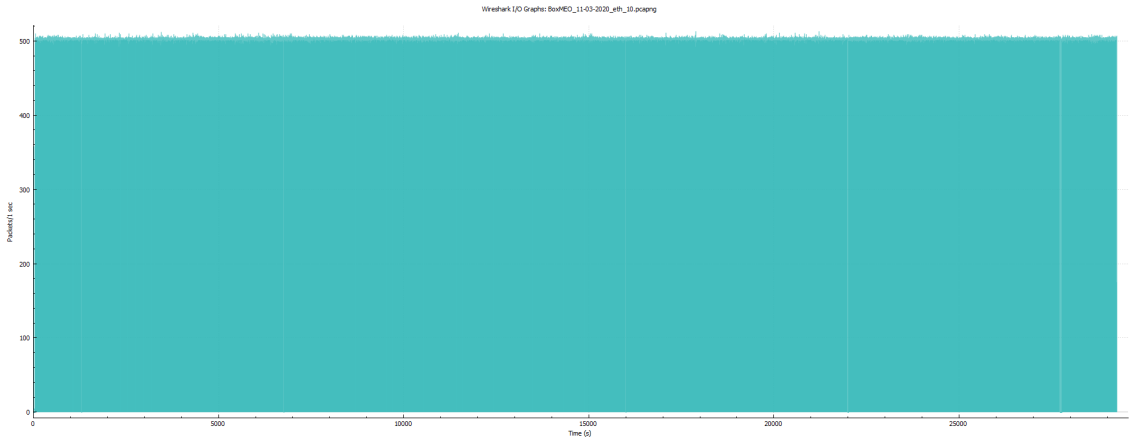


Figure 6.54: Stream 2 packet graphic

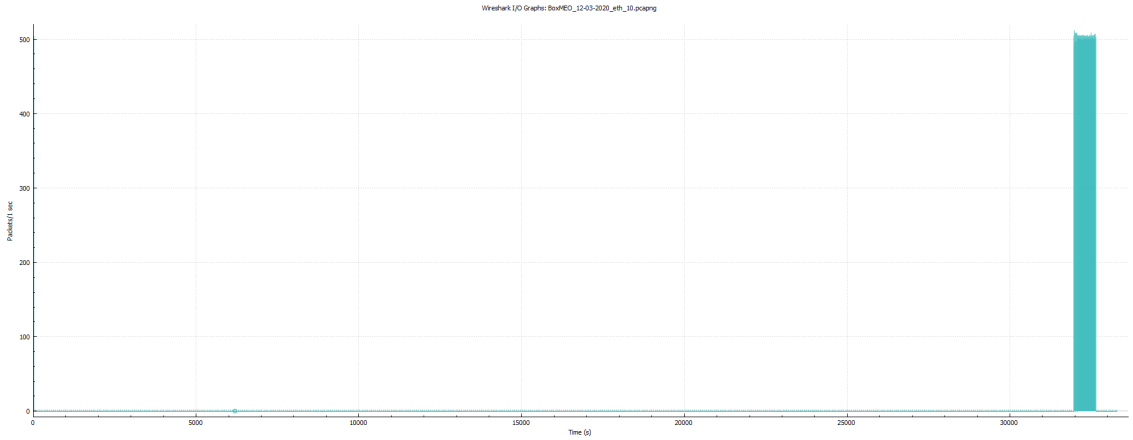


Figure 6.55: Stream 3 packet graphic

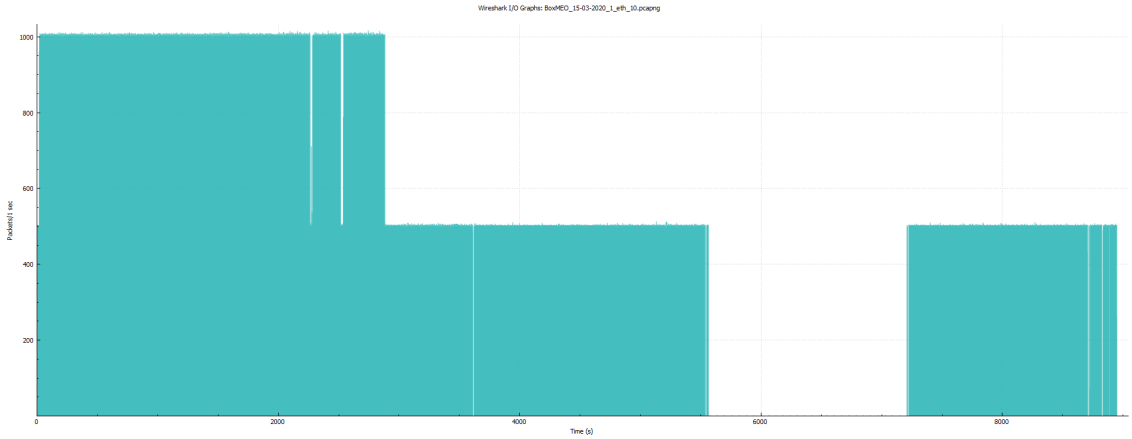


Figure 6.56: Stream 4 packet graphic

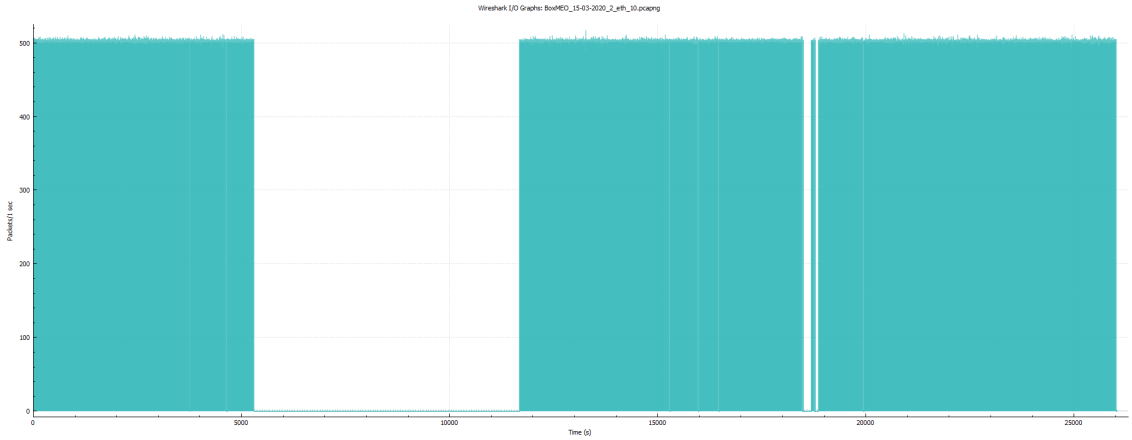


Figure 6.57: Stream 5 packet graphic

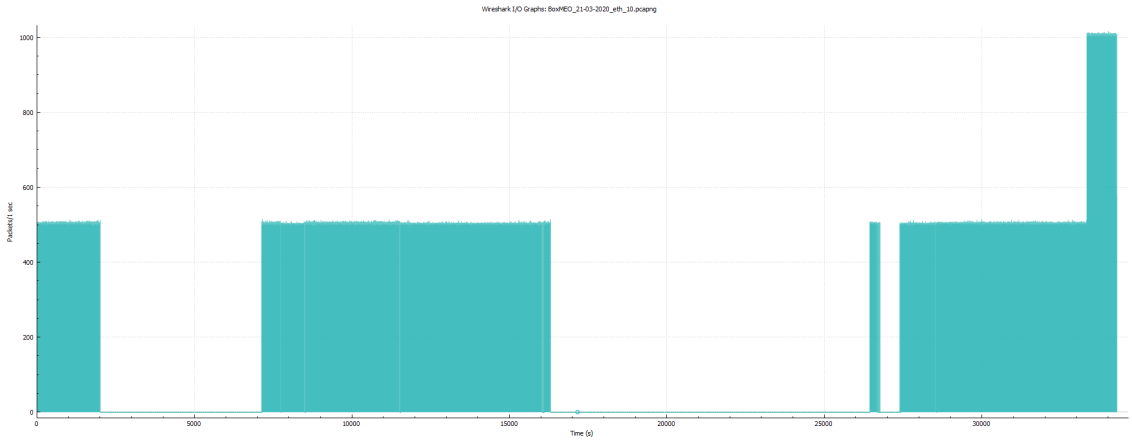


Figure 6.58: Stream 6 packet graphic

The extremely regular pattern of this network, and the absence of protocols like DHCP or DNS, together with the fact that only the UDP protocol is used, indicates that in this network everything is already setup. If this is the network used by ISP to television broadcast, and is configured as a private network, it is an acceptable behavior.

6.5.3 192.168.1.0 Network

Let us look now to the other network. Starting, as usual, with the analysis of the DNS and HTTP protocols, followed by the analysis of the stream figures.

DNS

All DNS requests are for the "iptv.telepac.pt" domain. There is no "no such name" in any of the response queries.

HTTP POST

The HTTP POST requests are for the DLNA protocol as with previous devices.

Data stream analysis

Looking at the data streams starting by looking at the streams general data, which is showed in Table 6.8:

Table 6.8: Set Top Box streams

Stream	Size (MB)	Time (h:m:s)	Time (s)	Bytes	KBytes/s	Packets	Packets/s
Stream 1	27	20:33:38	74018,484	25742702	347	44385	0,6
Stream 2	126	08:07:55	29275,864	122715031	4191	105044	3,6
Stream 3	37	09:15:05	33305,964	35973517	1080	38087	1,1
Stream 4	1241	02:29:04	8944,480	1229910117	137	339075	37,9
Stream 5	4664	07:13:56	26036,352	4625954560	177	1145897	44,0
Stream 6	579	09:31:14	34274,724	571511277	16	224341	6,5

Starting by looking at the streams as a whole, the graphic is showed in Fig 6.59:

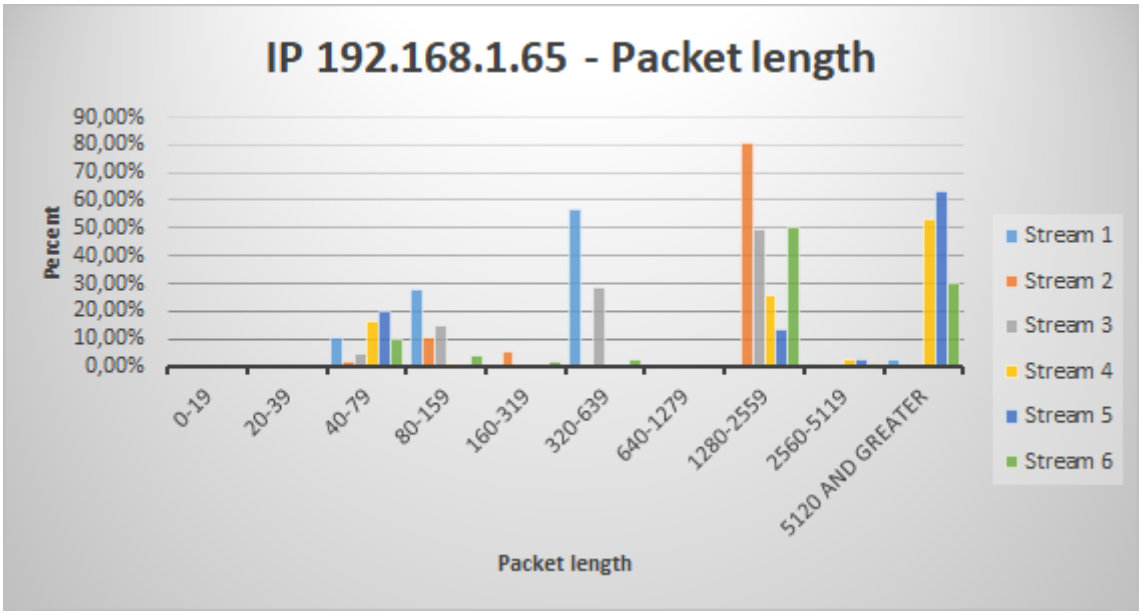
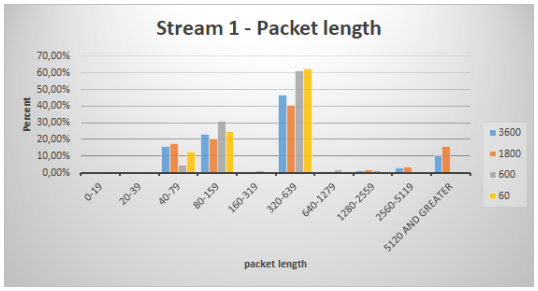


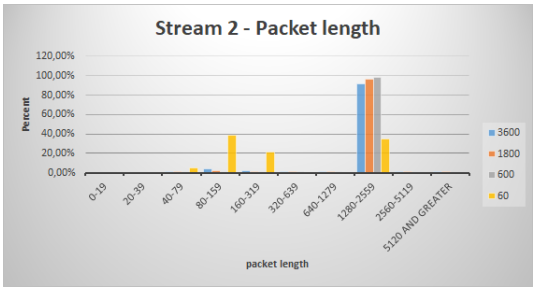
Figure 6.59: IP 192.168.1.65 packet percentage

Looking at the graphics we see that there is no visible pattern. The different streams have different packet percentage in varying packet size groups. Since this is the IP address visible to the home network, the protocols used and their frequency may vary in an unpredictable manner.

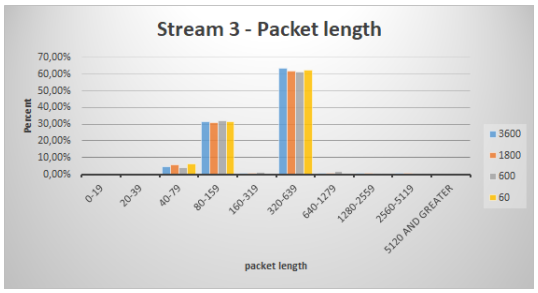
As before, it is time to look at each stream in the different time frames. The result is presented in Fig 6.60 (a), (b), (c), (d), (e), (f):



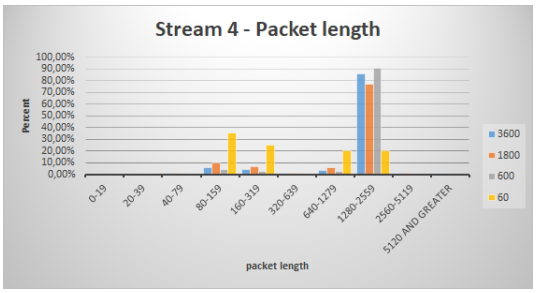
(a) Stream 1 packet percentage



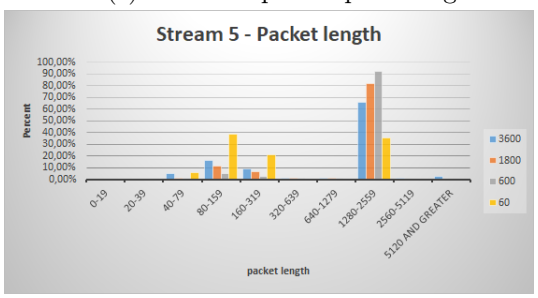
(b) Stream 2 packet percentage



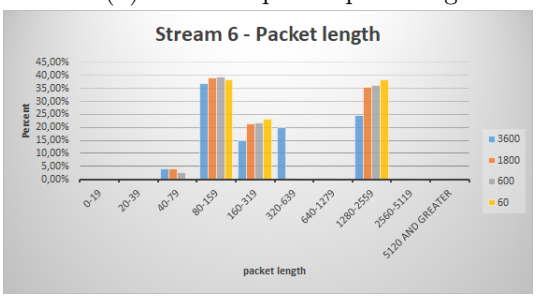
(c) Stream 3 packet percentage



(d) Stream 4 packet percentage



(e) Stream 5 packet percentage



(f) Stream 6 packet percentage

Figure 6.60: IP 192.168.1.0 stream graphics

As can be seen, although there is some consistency in each of the streams across the different time frames, they differ substantially from one another, rendering impossible to define a pattern for the device and network.

Packet burst analysis

Let us see the packet rate in this network, showed in Fig 6.61:

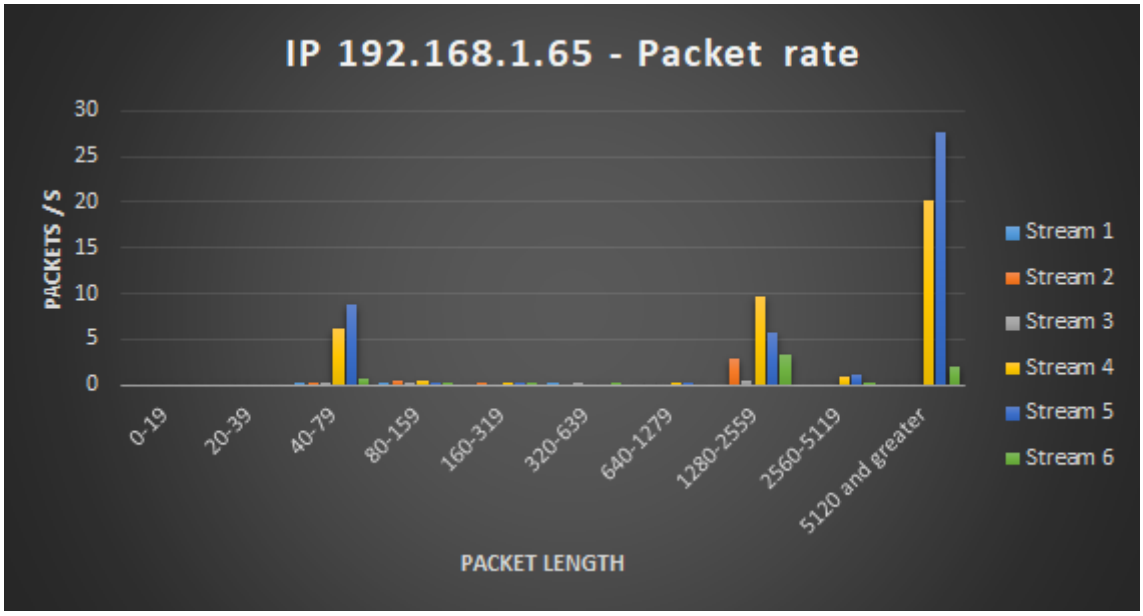
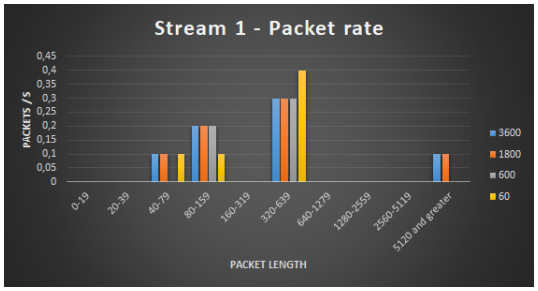


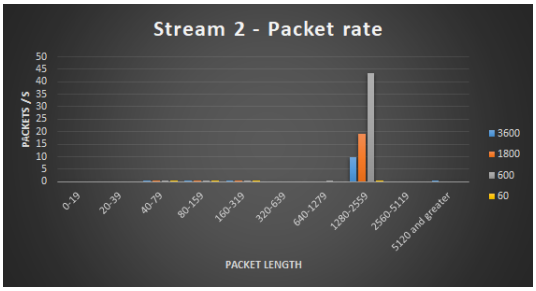
Figure 6.61: IP 192.168.1.0 packet graphic

As in the previous analysis, there is no identifiable stream pattern to characterize this network. The packets burst rate, vary significantly across streams and packet sizes.

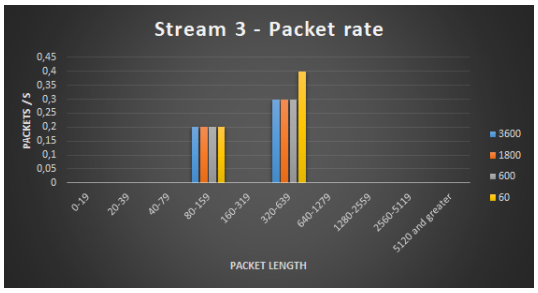
Now, let us analyze the packet burst rate in every stream across time frames, presented in Fig 6.2 (a), (b), (c), (d), (e), (f):



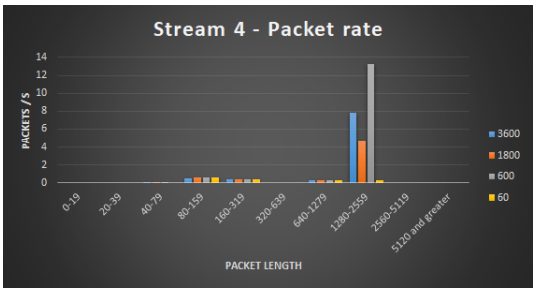
(a) Stream 1 packet rate



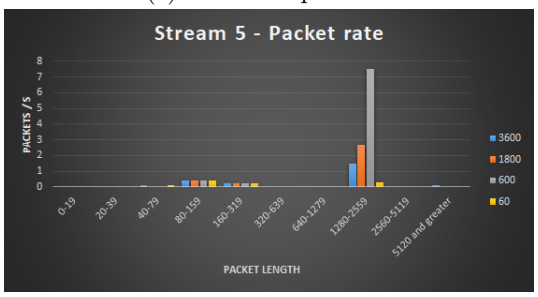
(b) Stream 2 packet rate



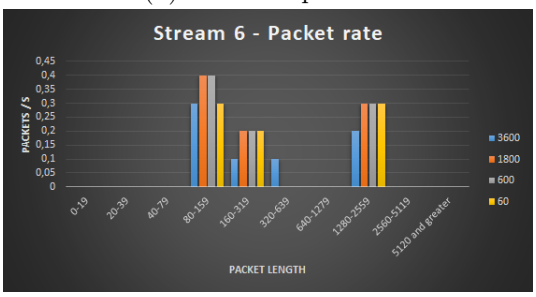
(c) Stream 3 packet rate



(d) Stream 4 packet rate



(e) Stream 5 packet rate



(f) Stream 6 packet rate

Figure 6.62: IP 192.168.1.0 stream graphics

Looking at packet burst rate, the differences are even greater then in the previous analysis. Here, even in the same stream, it is not always possible to find a pattern and across the different streams, there is no pattern at all.

Finally, we will look at the packet burst of the stream along all the capture in the Figs 6.63, 6.64, 6.65 and 6.66, 6.67 and 6.68:

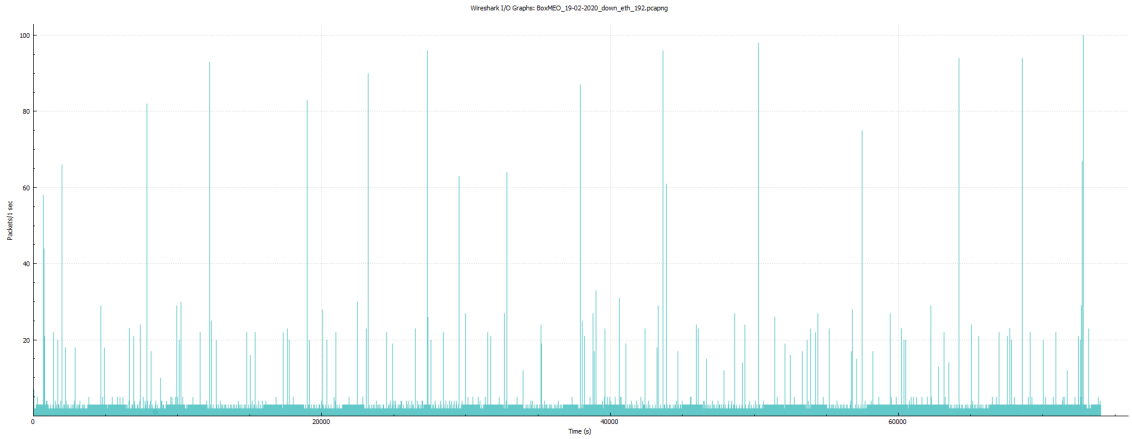


Figure 6.63: Stream 1 packet graphic

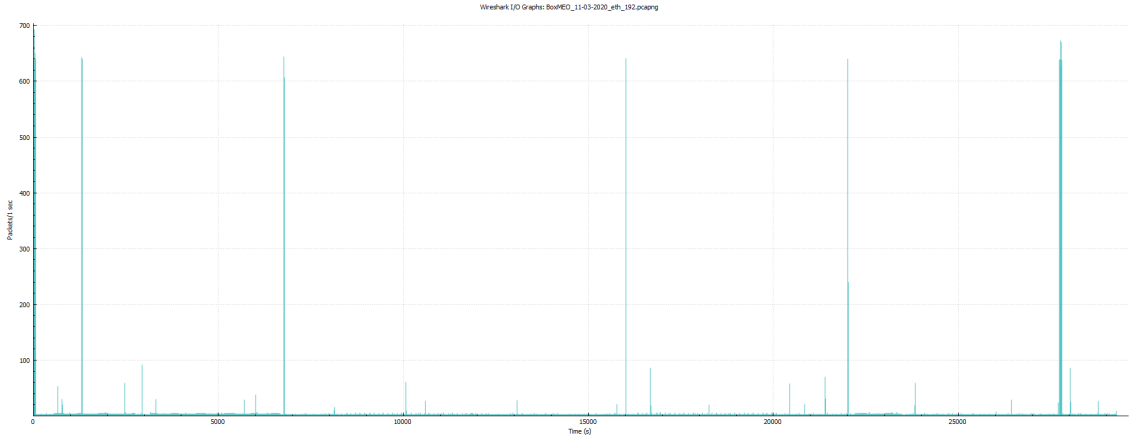


Figure 6.64: Stream 2 packet graphic

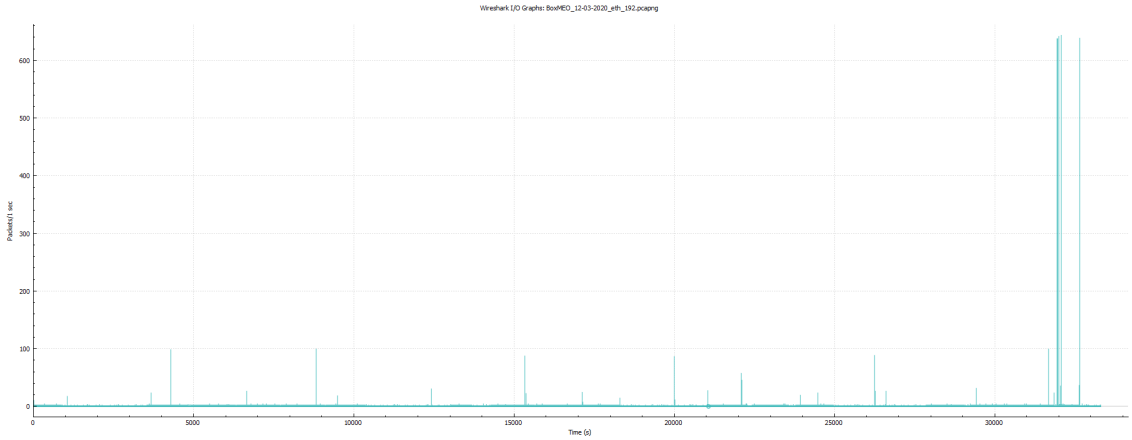


Figure 6.65: Stream 3 packet graphic

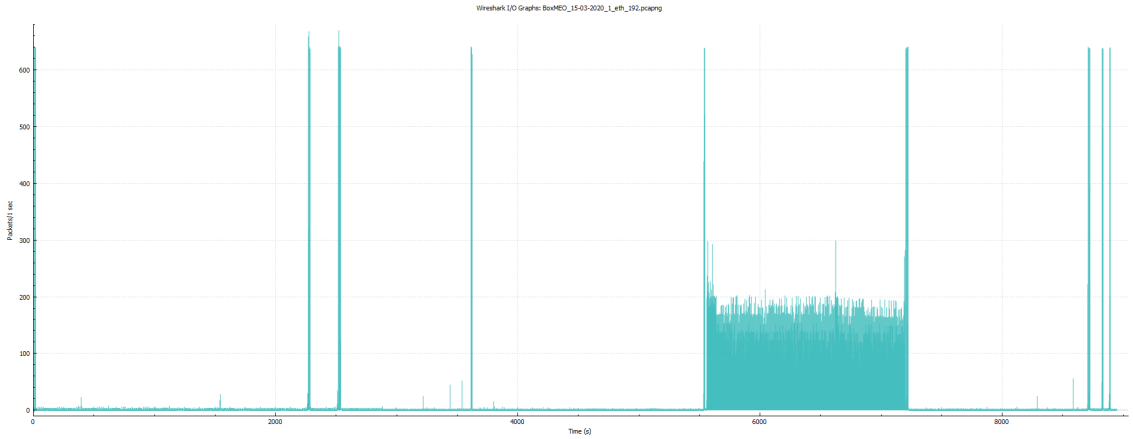


Figure 6.66: Stream 4 packet graphic

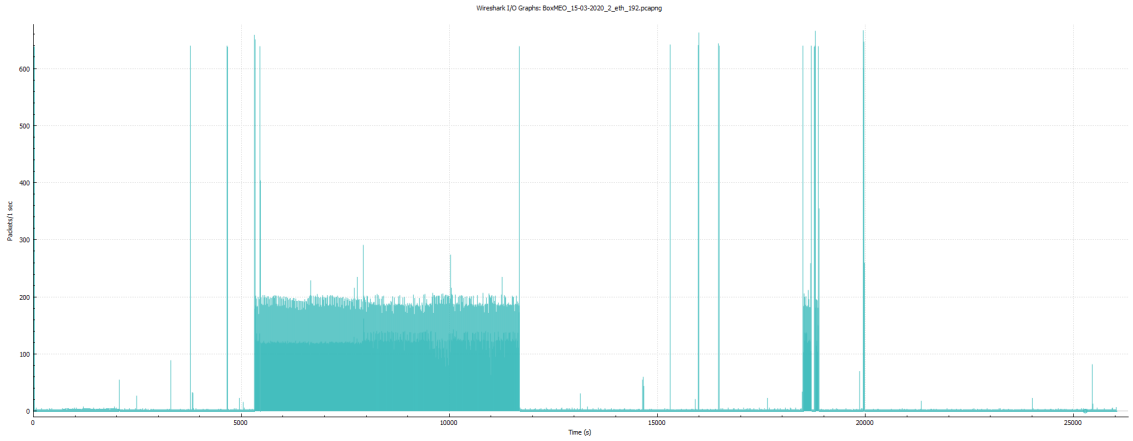


Figure 6.67: Stream 5 packet graphic

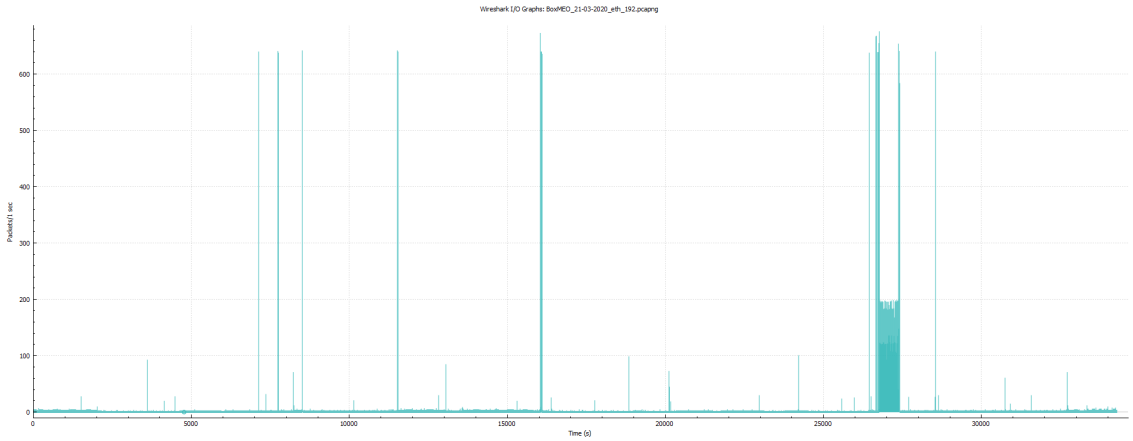


Figure 6.68: Stream 6 packet graphic

Looking at this graphics we cannot find any similarity between them, so this network as no possible pattern that we can find. One possible answer is that the device use this network only in management activities, and for communicating with others devices in the network.

6.6 Conclusions

In this chapter we tried to identify and define the stream patterns for the different smart devices and applications installed on them. Also, a first attempt has been made to find any irregular or suspicious communication. The stream pattern definition must be done using only network streaming data, since it is not possible to analyze packet content because it is encrypted. As so, the ability of profiling a network stream application largely depends on the way that application works, and the flow of data that is required. Also, the smart device hardware plays an important role on that matter. For Netflix, which sends high-definition video and audio, the amount of needed data leads to a constant stream and typical packet sizes, making it possible to create a profile. As for YouTube, many videos are of a much less quality, so the amount of data it requires is significantly reduced, leading to different stream profiles depending on the specific video content and used smart device. The Set Top Box works in a different way, there is no external applications, so we can only analyze is own network.

Chapter 7

Proof of Concept Implementation

The final part of this work consists on developing a program to monitor the outgoing traffic with rules given by the user, based on network traffic analysis. This program is a fundamental result because it is the most tangible outcome of this work, and represents the sum of all the previous stages.

7.1 Development environment

The monitoring program is written in the C programming language and takes advantage of the well known "libpcap" c library, part of the "tcpdump" packet analyzer program [45]. One may question why not to use a more modern programming language that includes a group of functional and security features, not present in C, which reduces the programmer workload. The answer lies in the performance and low memory usage requirements that were hinted at in Section 4.3 and were further reinforced by the evidence accumulated during the analysis and development stages. This is absolutely necessary because the host device has limited computing power and the program needs to perform real time traffic analysis in a eventually very loaded network, without introducing perceivable lags or losing packets. Moreover, besides abstracting several low-level details, the libpcap library also includes the possibility of creating filters to perform the desired monitoring actions with little effort, while forgoing the need for wrapper frameworks that add burden to the program.

Regarding the development environment, as the Raspberry Pi will host the Raspbian Operating System, which is based on Debian Linux, the development environment will be equally built on the Debian Operating System, used together with the Visual Studio Code programming IDE installed in a laptop computer. This is also because of the limited computing power of the Raspberry Pi, which advises the use of a more powerful computer for software development. Using the same based OS we hope to reduce the compiling problems that may arise when trying to do the final compilation on the Raspberry Pi. The VS Code is a freely available IDE [46] with a fairly good set of functionalities to help in programming, including the possibility to do a remote compiling on the Raspberry Pi from the development laptop.

7.2 Code structure

The program is divided in different code files, as is recommended by best practices, according to the desired functionality. As expected, each file performs a specific functionality, allowing for the flexibility and maintainability of the code. The code structure follows:

- arp_packet.c
  arp_pacekt.h
- common.c
  common.h
- device.c
  device.h
- filter.c
  filter.h
- icmp_packet.c
  icmp_packet.h
- ieee_vlan_packet.c
  ieee_vlan_packet.h
- igmp_packet.c
  igmp_packet.h
- ip_packet.c
  ip_packet.h
- ipv6_packet.c
  ipv6_packet.h
- list.c
  list.h
- local_packet.c
  local_packet.h
- netSniffer.c
  netSniffer.h
- reverse_arp_packet.c
  reverse_arp_packet.h
- tcp_packet.c
  tcp_packet.h
- ucp_packet.c
  ucp_packet.h
- unknown_packet.c
  unknown_packet.h

The rules for the file organization are as follows: there is a file for each of the different protocol types that can be identified by the "packet" word in the end of the name. Although some of the protocols may seems unnecessary, they are included in anticipation of future requirements. On the other hand, we never know which protocol will have to be blocked or processed. The chosen protocols are those that already exist and are supported on the libpcap library. There is a file ("list.c") for managing the data structure used to keep filters in memory, which is a linked list. This file is responsible for providing the operations required to add, delete, and change members of the list. There is another file for managing the filters, named "filter.c". This file deals with the actions needed for the filters, and has many connections with the "list" file. The main program is the "netSniffer.c" file.

The main program starts by presenting a menu to the user. The user must choose the network interface in which to do the captures, and the filter rules to use. Then the program opens the network interface and applies the chosen rules. When all the initialization is done, it starts an infinite loop for packet capture and analysis. During this analysis, whenever a packet who meets the filtering rules is detected, it will be analyzed. The program terminates when pressing the <CTRL C> key combination.

The main program flowchart is the following:

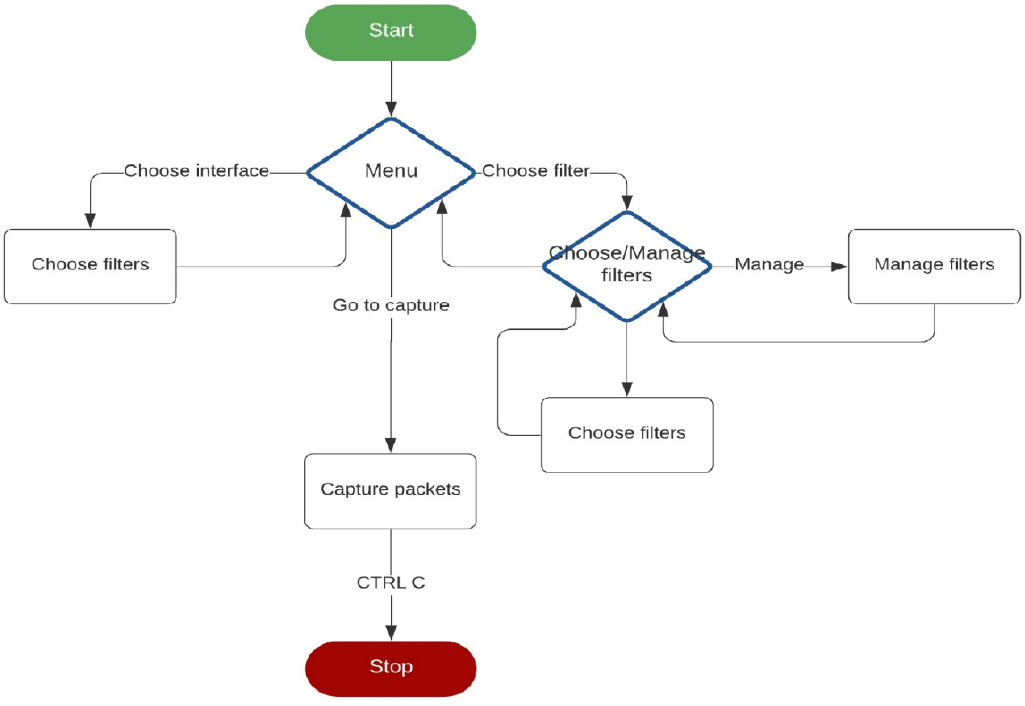


Figure 7.1: Main program flowchart

7.3 Code development

The initial part of the programming effort was to decide and define the required data structures and the general program flow. One of the first concerns was about the data set

of the filters, and how to store the structure used to keep the different pieces of data from each filter. This structure will have not only the filter rule string, but also other fields, like filter group and if the filter is active or not. This data may not seem relevant, but it has a more important role as the program grows larger, eventually getting more complex - for this reason special care was taken in choosing the most adequate approach to keep the filter data set in memory. The choice fell on a linked list because it is simple in structure, but flexible enough to accommodate the changes the user will do over time. The data flow is simple as the program has a single function to do. It consists on data initialization and then performing the loop.

The coding part started with the code to deal with the filters such as reading a new filter given by the user, saving filters to a file and read filters from a file, comparing them, and deleting a chosen filter data set. Afterwards, the code to manage the linked list operations, such as adding, removing and searching for filters was developed. This code has a connection with the filters code because is a list of filters. Finally, the main program, "netSniffer.c", acts only as a wrapper for the other code. After reading the network interface, and the filtering rules, it compiles the rules for the interface using the functions from the "libpcap" library, and starts the network analysis. During the capture, when a packet that meets the filtering rules is found, it is passed to the appropriated function to deal with it. By now, the code to deal with captured packets just gives a warning when is called.

Chapter 8

Conclusion

This thesis explores a new perspective regarding home network security. The work does not focus on trying to avoid attacks from the outside (although this is still a major concern), but rather on figuring out if and how our home appliances are having abnormal traffic patterns which might reveal security or privacy risks (e.g. giving user's personal information to third parties without their permission). To achieve this, a first round of network analysis was performed, in order to somehow infer the behavior of target devices/applications and to assess if their behavior can be easily profiled. In the future, this can be the basis for defining specific pattern behaviors for specific devices and applications, and integrate this ability into a traffic analyzer for detecting deviations and taking appropriate countermeasures - for instance to avoid having personal information ex-filtrated or allowing smart home devices to act as a zombies in botnets.

To implement all this structure it will be necessary to have a device able to perform this traffic profiling and analysis but still practical enough to be accepted for wide usage in the domestic environment - inexpensive, with low power consumption, and with a small footprint. This restricts our choice to devices with somehow limited but sufficient computing power. In the current devices landscape, it is possible to find such a device, even though it is necessary to find the right balance between price and performance. Our choice fell on a Raspberry Pi 3 B+, a well known device with reliable hardware, and able to deal with the imposed software demands. For the Operating System, the choice was the Raspberry Pi variant of the Debian Linux, the Raspbian without GUI. The GUI consumes power and resources, and is useless in a device remotely controlled that should be "invisible" in the network. Also, this OS is free for everybody to use.

Having defined the platform, the first thing to do is to analyze the network traffic, identifying behavioral patterns for the different smart devices and their applications, so to use it with a program to detect abnormal behavior. However, network content delivery companies disguise their online signature for commercial and security reasons, using traffic encryption (decoding data was not the aim of this work) and also a widespread network of servers, usually contracting with content distribution companies and changing the servers with no perceivable logic, making it virtually impossible to determine if a specific domain is legitimate or not. On the other hand, many smart devices communicate with what seems to be manufacturer's servers for what looks to be legitimate software updates. So, profiling through DNS analysis alone is not a reliable approach. Also, other possible approaches such as inspecting HTTP messages did not reveal any relevant information.

So, the alternative is to profile traffic, according to specific smart device and application, in order to detect abnormal behavior not revealed otherwise. Two applications for video streaming were analyzed: Netflix and YouTube. The results showed it is possible to profile the Netflix application, based on the percentage of the packet size used and, in a lesser degree, the packet rate. This behavior is persevering even in smaller time frames, so it will be possible to detect the signature shortly after the stream starts. As for YouTube, it was not possible to easily identify a similar pattern, because the streams have a more random behavior depending on content and user devices.

Having done this analysis, is was time to do some programming. This thesis did not intend to create a "ready-to-use" program, just a first prototype that will allow us to test the concept. Using a resource-efficient language was fundamental, since the host device has limited computing resources, as explained before. The C programming language was chosen, since it is fast and uses resources sparingly. Together with the "pcap" network library, which takes care of the low level network interface programming, this enabled the creation of a skeleton application for performing network analysis.

This thesis has shown it is possible to implement a device for network security that analyses the outgoing traffic looking for malicious activities. The device can be built at an affordable price for the average user, using already well established hardware and software components. The major problem is to identify the different network application patterns, as they are not always easy to profile. The application owners, whether manufacturers or content providers, must be called to help, revealing some data for allowing the correct profiling of their application's streams and for enabling the detection of malicious activities. While this seems unlikely for now, due to lack of regulation and lack of technological or business incentives, the expected increase of malicious attacks involving home network devices may lead to drastic scenarios where this collaboration among all the stakeholders becomes more consensual and/or imposed by regulation.

References

[1] S. Guha, P. Kidwell, A. Barthur, W. S. Cleveland, J. A. Gerth, and C. Bullard, "A streaming statistical algorithm for detection of ssh keystroke packets in tcp connections," *12th INFORMS Computing Society Conference, Monterey, California*, April 2011.

[2] M. F. Zhani and H. Elbiaze, "Analysis and prediction of real network traffic," *Journal of Networks, Vol 4, No 9*, November 2009.

[3] Y. Miao, Z. Ruan, L. Pan, Y. Wang, J. Zhang, and Y. Xiang, "Automated big traffic analytics for cyber security," April 2018.

[4] A. Biernacki, "Identification of adaptive video streams based on traffic correlation," *Springer link*, January 2019.

[5] J. Wang, L. Yang, J. Wu, and J. H. Abawajy, "Clustering analysis for malicious network traffic," *IEEE International Conference on Communications (ICC)*, 2017.

[6] Y. Miao, L. Pan, S. Rajasegarar, J. Zhang, C. Leckie, and Y. Xiang, "Distributed detection of zero-day network traffic flows," *Data Mining, Springer, 2017, pp 173-191*, April 2018.

[7] C. Rossow, C. J. Dietrich, H. Bos, L. Cavallaro, M. van Steen, F. C. Freiling, and N. Pohlmann, "Sandnet: Network traffic analysis of malicious software," *1st Workshop on Building Analysis Datasets and Gathering Experience Returns for Security, BADGERS 2011*, June 2011.

[8] H. Hindy, D. Brosset, E. Bayne, A. Seeam, C. Tachtatzis, R. Atkinson, and X. Bellekens, "A taxonomy and survey of intrusion detection system design techniques, network threats and datasets," *Division of Cyber Security, Abertay University, Scotland, Naval Academy Research Institute, France, Department of Computer Science, Middlesex University, Mauritius, EEE Department, University of StrathClyde, Scotland*, June 2018.

[9] S. M. Othman, N. T. Alsohybe, F. M. Ba-Alwi, and A. T. Zahary, "Survey on intrusion detection system types," *Faculty of Computer & Information Technology, Sana'a University, Yemen, International Journal of Cyber-Security and Digital Forensics (IJCSDF), The Society of Digital Information and Wireless Communications (SDIWC)*, December 2018.

[10] P. Wanda and H. J. Jie, "A survey of intrusion detection system," *School of Computer Science and Technology, Harbin University of Science and Technology, China, University of Respati Yogyakarta*, November 2018.

[11] D. Zhou, Z. Yan, Y. Fu, and Z. Yao, "A survey on network data collection," *State Key Laboratory of ISN, School of Cyber Engineering, Xiadian University, Xi'an, China,*

Department of Communications and Networking, Aalto University, Espoo, Finland, Journal of Network and Computer Applications 116 (2018) 9-23, 2018.

[12] S. Latha and S. Prakash, "A survey on network attacks and intrusion detection systems," *Cauvery College for Women, 2017 International Conference on Advanced Computing and Communication Systems (ICACCS-2017)*, 2017.

[13] F. A. Alaba, M. Othman, I. A. T. Hashem, and F. Alotaibi, "Internet of things security. a survey," *Faculty of Computer Science and Information Technology, University of Malaya, Kuala Lumpur, Malaysia, Faculty of Computer Science and Information Technology, University Putra Malaysia, Serdang, Selangor, Malaysia, Journal of Network and Computer Applications*, June 2017.

[14] Microsoft, "Noticing and responding to network-borne attacks." `https://docs.microsoft.com/en-us/previous-versions/tn-archive/cc723457(v=technet.10)`, 2014.

[15] T. Cruz, P. Simões, E. Monteiro, F. Bastos, and A. Laranjeira, "Cooperative security management for broadband network environments," *Wiley Security and Communication Networks*, vol. 8, no. 18, 2015.

[16] M. Mantere, M. Sailio, and S. Noponen, "Network traffic features for anomaly detection in specific industrial control system network," *VTT Technical Research Centre of Finland*, September 2013.

[17] F. Ye and Y. Qian, "Datanet: Deep learning based encrypted network traffic classification in SDN home gateway," *IEEE Access*, September 2018.

[18] T. L. von Sperling, F. L. C. Filho, R. T. S. Júnior, L. M. C. Martins, and R. L. Rocha, "Tracking intruders in iot networks by means of dns traffic analysis," *Cyber Security INCT Unit 6, Electrical Engineering Department, University of Brasília, Brazília, Brazil, 2017 Workshop on Communication Networks and Power Systems (WCNPS)*, November 2017.

[19] Y. Amri and M. A. Setiawan, "Improving smart home concept with the internet of things concept using raspberrypi and nodemcu," *Department of Informatics, Universitas Islam Indonesia, IOP Conf. Series: Materials Science and Engineering*, p. 325, 2018.

[20] A. A. Ahmed and Y. W. Kit, "Collecting and analyzing digital proof material to detect cybercrimes," *Faculty of Computer Systems & Software Engineering, Universiti Malaysia Pahang*, 2018.

[21] A. Sforzin†, M. Conti, F. G. Mármol, and J.-M. Bohli, "Rpids: Raspberry pi ids a fruitful intrusion detection system for iot," *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, July 2016.

[22] A. Aspernäs and T. Simonsson, "Ids on raspberry pi – a performance evaluation," *Diploma Thesis*, June 2015.

[23] P. Poonia, V. Kumar, and C. Nasa, "Performance evaluation of network based intrusion detection techniques with raspberry pi - a comparative analysis," *International Journal of Engineering Research & Technology (IJERT), ICCCS*, 2017.

[24] A. K. Kyaw, Y. Chen, and J. Joseph, "Pi-ids: Evaluation of open-source intrusion detection systems on raspberry pi 2," *IEEE*, November 2015.

[25] T. Zitta, M. Neruda, L. Vojtech, M. Matejkova, M. Jehlicka, L. Hach, and J. Moravec, "Penetration testing of intrusion detection and prevention system in low-performance embedded iot device," *18th International Conference on Mechatronics*, December 2018.

[26] R. P. Foundation, "Raspberry pi." https://www.raspberrypi.org/, 2020.

[27] C. Bio, "Banana pi open source project." http://www.banana-pi.org/, 2020.

[28] L. Shenzhen Xunlong Software CO., "Orange pi." http://www.orangepi.org/, 2020.

[29] H. C. Ltd, "Hardkernel." https://www.hardkernel.com/, 2020.

[30] Asus, "Tinker board s." https://www.asus.com/pt/Single-Board-Computer/Tinker-Board-S/, 2020.

[31] R. P. Foundation, "Raspberry pi specifications." https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/, 2014.

[32] Raspbian, "Raspbian home page." https://www.raspbian.org/, 2019.

[33] Wireshark, "Wireshark website." https://www.wireshark.org/, 2014.

[34] Nmap, "Nmap website." https://nmap.org/, 2014.

[35] Nmap, "Nmap options page." https://nmap.org/book/man-briefoptions.html, 2014.

[36] wireshark. https://www.wireshark.org/docs/man-pages/tshark.html, 2020.

[37] L. Electronics, "webos tv developer." http://webostv.developer.lge.com/discover/netcast/overview/, 2019.

[38] B. Zdrnja, "Google chrome and (weird) dns requests." https://isc.sans.edu/diary/Google+Chrome+and+weird+DNS+requests/10312/, 2011.

[39] openconnectivity foundation, "Alljoyn." https://openconnectivity.org/developer/reference-implementation/alljoyn/, 2020.

[40] urlvoid, "Urlvoid." https://www.urlvoid.com/scan/rdvs.alljoyn.org/, 2020.

[41] SpireSpark, "Dlna guidelines." https://spirespark.com/dlna/guidelines, 2020.

[42] S. E. B.V., "Kd-49xe7096 especificações | sony pt." https://www.sony.pt/electronics/support/televisions-projectors-lcd-tvs/kd-49xe7096/specifications, 2019.

[43] V. S. AS, "Opera tv is now vewd - vewd." https://www.vewd.com/news/opera-tv-is-now-vewd/, 2019.

[44] Wikipedia, "Multicast address." https://en.wikipedia.org/wiki/Multicast_address#Administratively_Scoped_IPv4_Multicast_addresses, 2020.

[45] Tcpdump, "Tcpdump website." https://www.tcpdump.org/, 2020.

[46] M. Corporation, "Visual studio code." https://code.visualstudio.com/, 2020.