DNS Firewall in Local Network

Marc Hernàndez Sànchez

Cybersecurity and Privacy

Enterprise security

Borja Guaita Perez

Victor Garcia Font

31/05/2022

FITXA DEL TREBALL FINAL

Títol del treball:	*DNS Firewall in Local Network*
Nom de l'autor:	*Marc Hernàndez Sànchez*
Nom del consultor/a:	*Borja Guaita Perez*
Nom del PRA:	*Victor Garcia Font*
Data de lliurament (mm/aaaa):	*31/05/2022*
Titulació o programa:	*Màster Universitari en Ciberseguretat y Privacitat*
Àrea del Treball Final:	*Seguretat empresarial*
Idioma del treball:	*Anglès*
Paraules clau	*Firewall, DNS, LAN*

Resum del Treball (màxim 250 paraules): *Amb la finalitat, context d'aplicació, metodologia, resultats i conclusions del treball*

El present treball exposa els conceptes bàsics que cal tenir per a poder desplegar de forma efectiva i eficient un Firewall DNS local a qualsevol xarxa domèstica, configurar-lo i mantenir-lo de forma sostinguda en el temps.

Durant el treball es repassen els principals conceptes que cal tenir en compte, així com la seva importància i impacte, tant a nivell domèstic com a nivell empresarial. S'exposa, amb diverses opcions i alternatives, quines son les eines, tant a nivell software com a nivell hardware, que cal tenir o adquirir per a poder realitzar el projecte i finalment es proposen diferents opcions de configuració per a poder realitzar el desplegament de forma efectiva, ja que es tracta d'una solució molt versàtil que varia en gran mesura segons les necessitats particulars de cada desplegament.

Finalment s'analitzen les conclusions de la feina realitzada, tant a nivell tècnic com a nivell pràctic, i s'exposen quins han estat els principals reptes alhora de realitzar aquest desplegament, quins son els següents passos a tenir en compte si es vol desplegar aquesta solució per a un temps indefinit de vida i quines son les solucions paral·leles que es poden desplegar en cas que el Firewall no respongui de la forma desitjada.

Abstract (in English, 250 words or less):

This thesis reviews the basics you need to be able to effectively and efficiently deploy a local DNS Firewall to any home network, configure it, and maintain it in a sustainable manner over time, taking into account the key concepts and configurations in their performance.

During the work, the main concepts that must be considered are reviewed, as well as their importance and impact, both at home and at business level. In this thesis is it explained, with options and alternatives, what are the tools, both at the software and hardware level, that you need to have or acquire to be able to carry out the project. Finally, different configuration options are proposed to be able to perform the deployment of effectively, as it is a very versatile solution that varies greatly depending on the particular needs of each deployment.

At the end, the conclusions of the work carried out are analysed, both at a technical and practical level, and the main challenges in carrying out this deployment are set out, what are the next steps to take into account if you want to deploy this solution for an indefinite period of life and what are the parallel solutions that can be deployed in case the Firewall does not respond in the desired way.

Content

List of figures

INTRODUCTION

Context and needs

It is widely known that nowadays the security of our networks, not only in the professional way, has become more and more important for our operations and in our lives. We all have a lot of personal information and important services running during all day in all of those infrastructures, so it's really important to be sure that all of these services are correctly securitised and working in the way that we expect.

It's been more than two years than the COVID pandemic began and there have been a lot of changes in the way that we interact with technology and the way that we communicate each other, not only in our personal life's but also in the professional way to do this. It is mainly for this reason that with the COVID pandemic we started working much more with technology and new ways of communication that become a challenge in terms of security of the information.

Furthermore, due to the pandemic, a lot of enterprises have implemented teleworking in a fast manner, without doing risk assessment and analysing all the challenges in terms of security of information that this type of work could have. Under this situation, those enterprise I have not only implemented different ways to work, but also different ways to share information and communicate each other. This can be done with new services that could have problems and can be a target for people outside the company to get some private information or punish the performance of itself.

Despite the fact that in the enterprise world all the processes and procedures seems to be a little bit organised end audited, so the workers could communicate each other and perform day tasks in a secure ecosystem, it is not happening the same in our homes, where the main infrastructure and security systems are still the same than years before.

The goal of this project is to design and deploy a custom Firewall completely integrated with our local network in order to provide security, control and monitoring of the LAN and all the activity the we can found inside it. This custom Firewall will be made with a recompilation of open-source software integrated with custom software in order to adapt it to the specific requirements of the network and provide some functions that could be useful for detection and specially for the monitoring and alerting, two fields really complicated to develop in a custom environment such as a local network.

Nowadays, the security in our homes is completely delegated to the endpoint equipment’s of the network, especially to the computers with dedicated software (antivirus, antimalware, add-blockers…). All of them have a general view of the platform and the processes running in the equipment, but once the equipment is not part of the communication, they lose this view and it’s the network and specially the router, the equipment’s that monitor all communications.

It is mainly for this reasons that some other tools like SIEMs (Security information and event management) have a general view of what is going on in all the equipment’s of the network.

In this thesis we aim to build a custom Firewall that provide us a secure environment and communications from our network to the internet and a general view about what is going on inside it. This will be possible because all the actions, traffic and monitoring will be derivate to this Firewall, so we will be able to have all the information in a single point, not only in an specific place, like I have already explained before.

COVID-19, cyberattacks and new challenges

As it has been mentioned before, the pandemic changed all the rules of the game in terms of security of information and the way that the enterprise an at least the people it's prepared to deal with the security of our infrastructure. The world saw an alarming 105% surge in ransomware cyberattacks last year. The attacks are designed to cripple people or businesses by making their computer systems unusable until they pay money or “ransom”.

Governments worldwide saw a 1,885% increase in ransomware attacks, and the health care industry faced a 755% increase in those attacks in 2021, according to the 2022 Cyber Threat Report released Thursday by SonicWall[1], an internet cybersecurity company. Ransomware also rose 104% in Europe, just under the 105% average increase worldwide, according to the report.

Cybersecurity has become one of the main challenges after the pandemic, because as we already know a lot of companies started working with some solutions or services that needs to be secure and controlled. It is quite easy to see it in numbers.

[1] https://www.sonicwall.com/2022-cyber-threat-report/

It is estimated that on average 30,000 websites are hacked every day[2]. In fact, a company falls victim to a cyberattack every 39 seconds and more than 60% of organizations globally have experienced at least one form of cyberattack.

According to the Ponemon Institute and IBM's Cost of a Data Breach Report 2021[3], the average total cost of a data breach increased from \$3.86 million to \$4.24 million in 2021. The report indicates a 10% year-over-year increase in average total cost, which is the highest ever recorded in the 17-year history of the report. Customer Personally Identifiable Information (PII) was the costliest record type with an average cost of \$161 per lost or stolen record.

The findings from the report mentioned before showed that the overall increase in average total cost was due to slower response time as a result of remote working. Organizations with more than 50% of their workforce working remotely took nearly 316 days to locate and contain the breach, compared to the regular average of 287 days. As per the report, data breaches with longer response time (more than 200 days) cost \$4.87 million on average while for breaches with less than 200 days response time cost \$3.61 million on average. The report also indicated that businesses could save up to 30% if they could contain a breach within 200 days.

The number of cyberattacks are growing rapidly and becoming more dangerous than ever before, not only direct to companies and enterprises, but also directly to people o some of certain services. We have to consider that the technology is evolving rapidly and then new ways of exploitation appear in a fast manner. Threat actors are constantly evolving and so are their tactics. Around 300,000 new pieces of malware are created daily to target individuals and organizations.

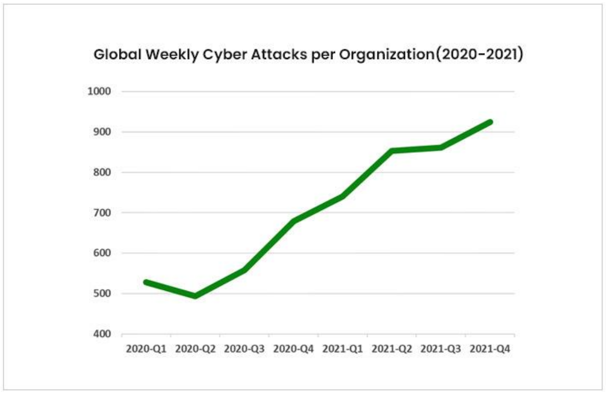


Figure 1. Global weekly cyber attacks

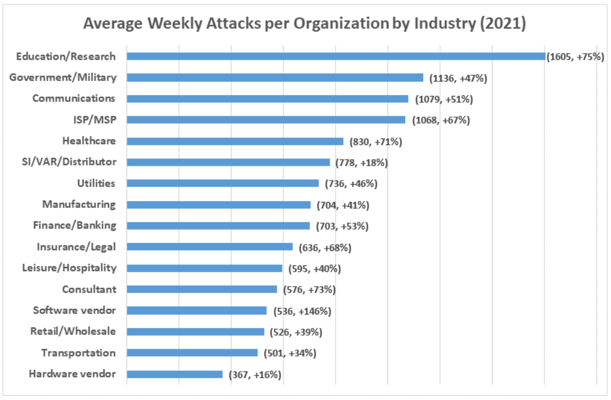


Figure 2. Average attacks per organization

[2] https://spanning.com/blog/cyberattacks-2021-phishing-ransomware-data-breach-statistics/
[3] https://www.ibm.com/security/data-breach

From exploiting human error to launching sophisticated assaults capable of bypassing even the strongest security systems, cyberattacks can come in various forms.

But ransomware was not only the main problem in terms of cyber security. According to the Identity Theft Resource Center's[4] (ITRC) data breach analysis, there were 1,291 data breaches through September 2021. This number indicates a 17% increase in data breaches in comparison to breaches in 2020, which was 1,108. The report also found a steep increase in the number of data compromise victims (281 million) during the first nine months of 2021.

Apart of that a bridge and ransomware there are several other stuff that has a direct impact in the operational system of a company. It is clear that this type of attacks and problems are growing up and they are not expected to end at least in the near future. The best thing that we can do against this type of problems is to try to understand our weaknesses and of course develop and strategy that allow us to prevent all of this to happen. Not only the huge companies, but also the small ones, are working in order to fight this type of attacks and intrusions, so the next step it's to take care of ourselves in our private networks and services.

Increasing network security of our homes

It is clear that the network security it's a priority for our organisations and enterprises in order to be sure that the information it's correctly securitised. The next step its to assume that the security of the information it's not only present in our works, but also inside our home. Despite the fact that the risk is not the same and the target could change if we compare it with an enterprise, we have to be really sure that we could also be a victim, so the first challenge it's to know exactly what we have inside our local area network and once we have this information, work in the same line as it's recommended in the enterprise world but adapting our risks and needs as we think that it has to be approached.

It is mainly for this reason that the security in our networks, and not only end user devices, is becoming more important and our societies are taking part with more attitude

[4] https://www.idtheftcenter.org/post/identity-theft-resource-center-to-share-latest-data-breach-analysis-with-u-s-senate-commerce-committee-number-of-data-breaches-in-2021-surpasses-all-of-2020/

than before. But there is still a lot of work to do, not only in a deployment way (like the one that it is exposed in this thesis) but also in awareness.

Thanks to this project we will be able to explore all the possibilities that we could have in our home LAN, we will be able to analyse what is going on inside it, and in the best of the futures we will be able to detect or monitor if there is strange movements or events inside our network.

Planning and objectives

As it has been explained in the previous chapters of the document, the work will be done in differentiated phases according to the development and the main challenges in the Firewall implementation. It is important to remark that this project is a full development project, so the main core of the work is done during the integration and development phase.

It is mainly for this reason that first is very important to make an information gathering to define the main software that will be running in the device in order to develop a stable bases to work over it in the following weeks.

The Gantt Diagram that is shown later on is divided in weeks from February to June. There are four main chapters (Planning, Development, Further development and Documentation).

The main idea is to divide the thesis in three main steps, the first one for learning and reading all the information that could be useful in order to define all the Firewall characteristics. The second one for the implementation, and if the timing allows it, develop a little bit more features. Finally, the last step will be the documentation of all the process and the system description in order to have all the information placed in the document.

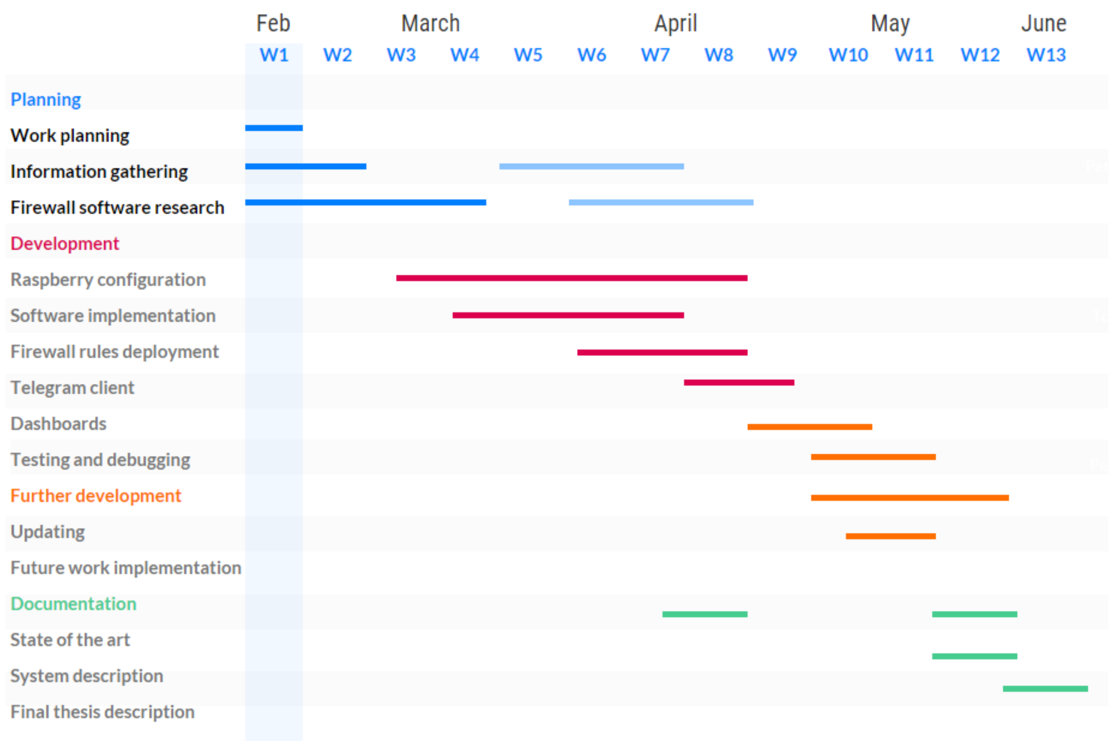


Figure 3. Initial planning and main tasks

Chasing this information with the one required in the UOC Portal, the first delivery will be a pure development one, specially related with configuration and Firewall characteristics. In this delivery the main implementations of the thesis will be at least presented and tested.

The second delivery will be devoted to the next steps of the development and in case the implementations are accomplished, the further implementations and work. Finally, in the last delivery previous to the final one, the main documentation with all the project description will be presented.

Once all documentation it's written and all the development is implemented and fully working, the final goal is to be able to deploy the same system with similar requirements in other networks or services without having too much trouble in terms of compatibility and performance. Thanks to this structure of work, I will be able to develop as long as it is working during time. This is specially important because there are some features that have been implemented that needs to be tested during time, an this testing east better and more reliable if the time of testing increases.

So, the main idea is to implement as much as features as we want in order to start testing them sooner, so we can obtain useful information in order to continue developing this features or try to use another software or solution to accomplish our objectives.

Once all of this information has been collected and we have a final firewall working without problems, the idea is to define the next steps in order to continue this development and of course allow the system to be updated automatically or manually. this is important because, as it will be shown in the thesis, there are some features that store information in the custom firewall. Furthermore, these features have to stay stable during that time without causing any problems. So it is for this reason that there has to be a good dimensioning off the stored data and the capacity of the Raspberry Pi to perform operations if the network increases and there is more CPU requirement.

Resources and material

To make this project possible there are some materials that must be present in order to accomplish the objective that we have defined. There are some of those materials that can be performed with other type off solutions, but the codes on the overall performance will have to be adapted in order to achieve our main objectives. The list of material that we will need in order to build a custom firewall that it is able to monitor all our local area network is the following:

Perimetral Router

The perimetral router of our homes it's always present if we have Internet connexion and some of our devices connected to them. It is very important to take this device into account because there are some activities end actions that this device is in charge of, that we will have to change in order to provide our general core of intelligence (Raspberry Pi) dissenter of those actions.

Thanks to this change of activities in our LAN, we will be able to monitor everything about what it's going on inside it and which will be the performance and the communications between the insides of our network on the Internet. At the end, we are changing the firewall role of the perimeter router two our Raspberry Pi. This is important to remark because it is really complicated (because of software incompatibilities) to have two entities with the same role in the same network, so the performance will not be the one expected. This is explained in much detail in the conclusion chapter.

Raspberry Pi model 4

Raspberry Pi is a series of small single-board computers (SBCs) developed in the United Kingdom by the Raspberry Pi Foundation in association with Broadcom[5]. The Raspberry Pi project originally leaned towards the promotion of teaching basic computer science in schools and in developing countries. The original model became more popular than anticipated, selling outside its target market for uses such as robotics. It is widely used in many areas, such as for weather monitoring, because of its low cost, modularity, and open design. It is typically used by computer and electronic hobbyists, due to its adoption of HDMI and USB devices.

This device acts as a small computer with a very good performance an low power consumption. Their main operating system, Raspbian, allows us to implement a lot of different features in order to achieve our main objectives. Instead of this device we are able to use other ones such as a dedicated computer or even some or other small devices like Arduino, but all of them have different problems that are solved by the use of the Raspberry. This is thanks to the low consumption and running an operating system based on Linux.

Swich with port mirroring

One of the most important devices that we have to consider if we want to monitor all of our network is a switch device that have the option of mirroring some of their ports. it is important to remark that not all the switching devices allows these features, so without the mirroring board we won't be able to monitor the devices without installing them agents or a third party software that provides us this feature.

This is only needed if we want to have some IDS behaviour in our network. The mirroring port allows us to replicate the traffic that we are filtering in other ports in one specific port, so we can gather all the information through one monitoring port of those switch in order to sum up all the traffic that it's going through our network. With this traffic we will be able to perform some actions in order to detect some malicious behaviour or even some compromise indicators that are inside of our network.

Despite the fact that it is not mandatory if we want to implement a firewall in our network that monitors the connexion from the Internet to the inside of it, it is strongly

[5] https://www.raspberrypi.com/products/raspberry-pi-4-model-b/

recommended just to be able to know what it's going on and not only end the handshake through the Internet.

Ethernet and other connections

the last part of our infrastructure are the Ethernet connexions and wired connexions. Although the best way of interconnecting network is using Ethernet, we don't have this opportunity always, so we have to consider other ways of communication in order to be able to provide Internet in all of our network. In my particular case I've used a device called PLC, that's provides Internet through the electricity infrastructure of our home.

we can also have wireless Connexions such as Wi-Fi or even other solutions such as an internal router or access point to provide better coverage an accessibility to the network. The more bigger the network the higher the challenge will be, but it is not a problem to have a complex distribution, because with our switch and some of the configurations that we will apply to our Raspberry Pi, we will be able to reach all the sources of information and monitor them without any problem.

STATE OF THE ART

Firewalls and NGFW

A firewall is a network security device that monitors incoming and outgoing network traffic and permits or blocks data packets based on a set of security rules. Its purpose is to establish a barrier between your internal network and incoming traffic from external sources (such as the internet) in order to block malicious traffic like viruses and hackers.

Firewalls carefully analyse incoming traffic based on pre-established rules and filter traffic coming from unsecured or suspicious sources to prevent attacks. This type of devices guard traffic at a computer's entry point, called ports, which is where information is exchanged with external devices.

This type of devices allows an exact control of what is going on between two networks. the best way to understand the performance of one of those devices it's imagine two rooms and one guard in the door between them. The quiet acts as a firewall, monitoring and controlling which one it's entering and going out of one of those rooms an even in the best of the situations trying to analyse the guys that are going inside an outside an which will be their main activities inside there.

As their name suggests, next generation firewalls (NGFW) are a more advanced version of the traditional firewall, and they offer the same benefits. Like regular firewalls, NGFW use both static and dynamic packet filtering and VPN support to ensure that all connections between the network, internet, and firewall are valid and secure. Both firewall types should also be able to translate network and port addresses in order to map IPs.

The main idea of a next generation firewall is that is devolution off the old firewall, so these new devices allows new features that can be applied to several services that are running inside our network. This type of devices are very useful for those companies that have huge demand of services and traffic an need to perform some activations when they are exchanging information from the inside networks to the outside.

In our particular case we don't need a next generation firewall, because our needs do not correspond to the services that advise of these characteristics can give us. Despite the fact that this device is not what we need it is important to mention that there are some features that this new devices are offering to the companies that can be interesting implement in our custom firewall. This is mainly for these reasons that next generation firewalls can be an inspiration to define new features and complement in the future development the firewall that we have implemented already.

SIEM, IPS and IDS for monitoring

As it has been explained in the previous chapters, one of the main objectives of this disease is to develop a custom firewall able to monitor all the data and all the events that are happening inside our network. To be able to do this we have to previously no what are we talking about. It is mainly for this reason that three main concepts have to be introduced in order to understand their main goals and functionalities to consider adding them to our solution.

First of all we have to introduce what is a SIEM and what it does. SIEM stands for security information and event management and provides organizations with next-generation detection, analytics and response. SIEM software combines security information management (SIM) and security event management (SEM) to provide real-time analysis of security alerts generated by applications and network hardware. SIEM software matches events against rules and analytics engines and indexes them for sub-second search to detect and analyze advanced threats using globally gathered intelligence. This gives security teams both insight into and a track record of the activities within their IT environment by providing data analysis, event correlation, aggregation, reporting and log management.

SIEM software works by collecting log and event data generated by an organizations application, security devices and host systems and bringing it together into a single centralized platform. SIEM gathers data from antivirus events, firewall logs and other locations; it sorts this data into categories, for example: malware activity and failed and successful logins. When SIEM identifies a threat through network security monitoring, it generates an alert and defines a threat level based on predetermined rules. For example, someone trying to log into an account 10 times in 10 minutes is ok, while 100 times in 10 minutes might be flagged as an attempted attack. In this way it detects threats and creates security alerts. SIEM's custom dashboards and event management system improves investigative efficiency and reduces time wasted on false-positives.

As it has been explained a SIEM solution is very useful for a company that has to monitor a lot of devices or even services that are working in their own network. For our needs, a

SIEM may be a tool that gives us too much functionality ease that we do not request. But this does not mean that we may capture some of those functionality and implement them in our solution.

In fact, we're going to monitor the main devices of our network without any agent and gathering their information directly from the activities that are doing. Despite the fact that we are going to introduce how the system has been developed, at that point it's important to mention that the mirroring function of our switch is a key point to monitor all the events that are happening in our network.

Without some of SIEM tools, specially the information gathering from one or more than one of our devices, we will be able to see what is going on inside our network (the communication between the internal devices of our LAN) and also the behaviour of our network communicating through the Internet. We will be able to see that information because we will be gathering all the events and adding them into several platforms of graphics and metrics.

Also, a SIEM is really important to monitor our network, we have to take into account that there are other tools that can be useful in this task. It's time to introduce what is an intrusion prevention system.

An intrusion prevention system (IPS) is a network security tool (which can be a hardware device or software) that continuously monitors a network for malicious activity and takes action to prevent it, including reporting, blocking, or dropping it, when it does occur. It is much more advanced than an intrusion detection system (IDS), which simply detects malicious activity but cannot take action against it beyond alerting an administrator. Intrusion prevention systems are sometimes included as part of a next-generation firewall (NGFW) or unified threat management (UTM) solution. Like many network security technologies, they must be powerful enough to scan a high volume of traffic without slowing down network performance.

An intrusion prevention system is placed inline, in the flow of network traffic between the source and destination, and usually sits just behind the firewall, that in our case will be the Raspberry Pi. There are several techniques that intrusion prevention systems use to identify threats:

- **Signature-based:** This method matches the activity to signatures of well-known threats. One drawback to this method is that it can only stop previously identified attacks and won’t be able to recognize new ones.

- **Anomaly-based:** This method monitors for abnormal behavior by comparing random samples of network activity against a baseline standard. It is more robust than signature-based monitoring, but it can sometimes produce false positives. Some newer and more advanced intrusion prevention systems use artificial intelligence and machine learning technology to support anomaly-based monitoring.
- **Policy-based:** This method is somewhat less common than signature-based or anomaly-based monitoring. It employs security policies defined by the enterprise and blocks activity that violates those policies. This requires an administrator to set up and configure security policies.

As we will be able to develop our own firewall that may contain an IPS solution, we will be able to implement those three strategies if we want to. But it is important to mention that these are three strategies won't be implemented as modern IPS system do. This is because we will be implementing an IPS solution and monitoring with the events from the SIEM recollection.

This means that the intelligence that we can provide to suspicious threats other activities will be made entirely for us, so we won't have indicators of compromise or strange behaviour previously defined in our network. This is a point that can be implemented in the future, but do too the huge amount of challenges and objectives that can be implemented as an IPS solution, the main goal is to bring the basis of this type of solution an in case there is more time or different ways to still develop this project doing this without any problem.

Finally, we will introduce one of the last tools that we will use as an inspiration to implement this project. We're talking about an intrusion detection system, that is a device or software application that monitors a network for malicious activity or policy violations. Any malicious activity or violation is typically reported or collected centrally using a security information and event management system. Some IDS's are capable of responding to detected intrusion upon discovery, as we have mentioned before with the IPS solution.

There is a wide array of IDS, ranging from antivirus software to tiered monitoring systems that follow the traffic of an entire network. The most common classifications are:

- **Network intrusion detection systems (NIDS):** A system that analyzes incoming network traffic. This will be the main feature of this project, as we are going to analyse entirely the communication from our local area network to the Internet.

- **Host-based intrusion detection systems (HIDS):** A system that monitors important operating system files.

There is also subset of IDS types. The most common variants are based on signature detection and anomaly detection.

- **Signature-based:** Signature-based IDS detects possible threats by looking for specific patterns, such as byte sequences in network traffic, or known malicious instruction sequences used by malware. This terminology originates from antivirus software, which refers to these detected patterns as signatures. Although signature-based IDS can easily detect known attacks, it is impossible to detect new attacks, for which no pattern is available.
- **Anomaly-based:** a newer technology designed to detect and adapt to unknown attacks, primarily due to the explosion of malware. This detection method uses machine learning to create a defined model of trustworthy activity, and then compare new behavior against this trust model. While this approach enables the detection of previously unknown attacks, it can suffer from false positives: previously unknown legitimate activity can accidentally be classified as malicious.

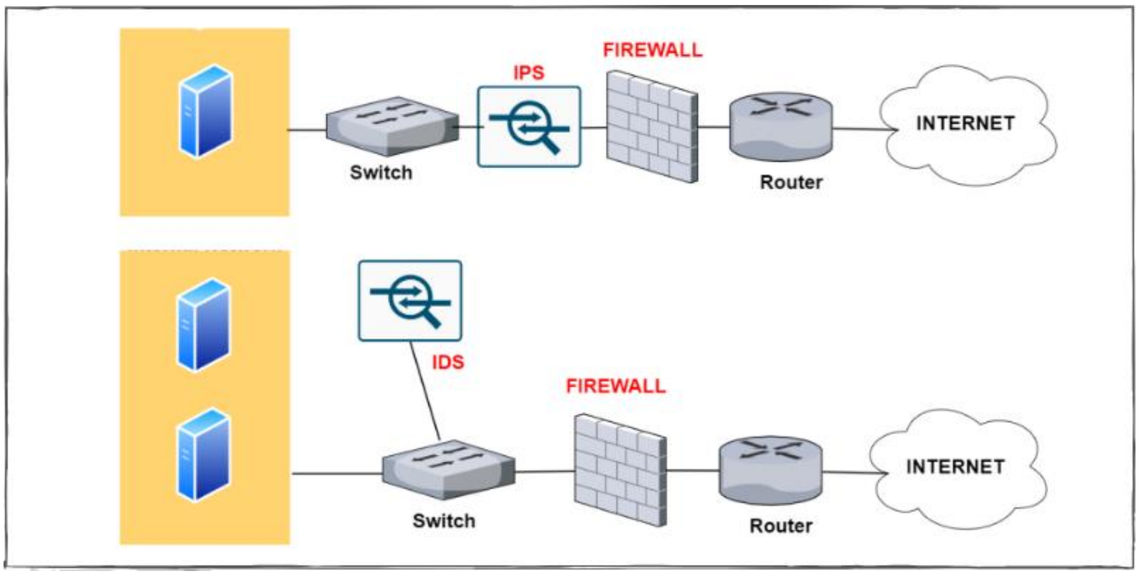


Figure 4. IPS & IDS structure

At this point is important to remark the main difference between an IPS and IDS. As we can see in the picture above, and IPS acts mainly as a firewall that implements blocking or not depending on the configuration and the rules that have been defined. despite the fact that we cannot consider a firewall the same as an IPS, date share a lot of functionality in terms of structure and network positioning, but an IPS provides much more intelligence than the firewall one. In the other side we can see that the idea is it's more devoted to monitoring and controlling what is going on in our network without blocking or applying any change in a reactive and preventive way.

In this project we are going to develop an IDS as its main concept but we will also try to implement some features of an IPS. So it is really important to understand both stroll's and try to implement the best features of each one trying to maximise the output of both solutions.

DNS RPZ

Domain Name Service Response Policy Zones (DNS RPZ) is a method that allows a nameserver administrator to overlay custom information on top of the global DNS to provide alternate responses to queries. It is currently implemented in the ISC BIND nameserver (9.8 or later). Another generic name for the DNS RPZ functionality is "DNS firewall". Even though DNS RPZ is not a final solution as the ones that I have previously mentioned, is really important to take into account this concept because we are going to implement this functionality directly in our firewall.

The prime motivation for creating this feature is to protect users from badness on the Internet related to known-malicious global identifiers such as host names, domain names, IP addresses, or nameservers. Criminals tend to keep using the same identifiers until they are taken away from them. Unfortunately, the Internet security industry's ability to take down criminal infrastructure at domain registries, hosting providers or ISPs is not timely enough to be effective. Using RPZ, a network or DNS administrator can implement their own protection policies base based on reputation feeds from security service providers on a near-real-time basis.

We have to consider that a local our network such as the one that we have on our houses is very similar in terms of infrastructure and of course services that the ones that we can find in on other organisation or enterprises. It is mainly for this reason that the general scope or the main objectives of our firewall could not be the same of the ones that we could find in a huge organisation. In terms of data protection and control, we have to consider that one of the main threats that we can find in our network behaviour is the interaction through the Internet. At this point, it is really important to have a tool that allows us to implement a DNS monitoring to block the traffic that we don't want to inside our network or that we want to monitor so we are able to cheque the destination of our search is in LAN.

Although we're going to talk about this later, at this point it is important to remark that, as I have said before, there are some functionality that will be disabled in the main router in order to provide the intelligence to our call monitoring that it will be placed in the Raspberry Pi. One of those functionality is that we are going to disable from the main router it's the DNS server, so we will implement this functionality from scratch in order to monitor or the traffic to the Internet and block some of it that good be dangerous or even annoying, for example some ads.

Packet filtering

Packet filtering is a firewall technique used to control network access by monitoring outgoing and incoming packets and allowing them to pass or halt based on the source and destination Internet Protocol (IP) addresses, protocols and ports. Network layer firewalls define packet filtering rule sets, which provide highly efficient security mechanisms.

During network communication, a node transmits a packet that is filtered and matched with predefined rules and policies. Once matched, a packet is either accepted or denied.

Packet filtering checks source and destination IP addresses. If both IP addresses match, the packet is considered secure and verified. Because the sender may use different applications and programs, packet filtering also checks source and destination protocols, such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP). Packet filters also verify source and destination port addresses.

Some packet filters are not intelligent and unable to memorize used packets. However, other packet filters can memorize previously used packet items, such as source and destination IP addresses.

Packet filtering is usually an effective defense against attacks from computers outside a local area network (LAN). As most routing devices have integrated filtering capabilities, packet filtering is considered a standard and cost-effective means of security.

Even though we are going to implement some features related with packet filtering in our DNS server, it is important to take into account that we will be implementing these features in our local network. This means that every feature that we implement that blocks traffic will have a direct impact in our user experience of the network. The result of implementing too much rules in terms of packet filtering could be not worth it because we will be dealing with a lot of errors and Internet consumption that is not accorded to our needs.

It is for this reason that we will be focused on the monitoring part an alerting than defining a lot of rules to block some traffic or services. This part is one of the most differential depending on the network we have and the requirements we want to define in order to deploy the solution.

IMPLEMENTATION

Previous work and analysis

As it has been mentioned in the previous chapters the technical implementation off this project has been made in order to establish the first important features at the very beginning of the development and be able to gather information about the performance of these new features. Once all of this information has been gathered, further development has been done an prepared in order to be able to continue working on the project once the main challenges are accomplished.

As we will explain in the general overview of the system, there are some software and hardware requirements that must be accomplished in order to develop all the required features an be able to continue developing them in the future. With this, it is important to remark that some previous study has been made in order to determine the best requirements and tools that can establish the basis of this custom Firewall.

During this chapter we are going to present all the options that have been taken into account to implement this custom firewall on why I have finally chosen between them the best ones. We will try to present at least two solutions for each challenge that we have, but in some cases there are several approaches, so the final target of our objectives good have been achieved in other ways.

Firewall with Raspberry Pi 4

The first and the most important thing that we have to take into account it's the hardware that we will need to implement all of the functionality is that we expect to have working as a custom firewall with an IPS system integrated and custom notifications.

This device has been chosen biggest it's cheap, it's based on Linux so it provides all the software tools to implement all the functionalities and it has power enough if we want to monitor eyes a LAN not bigger than 20 or 25 devices. Despite the fact that a Raspberry Pi model 4 it's good enough to perform this task, we can also implement this custom federal in older models of Raspberry Pi or even other devices such an Arduino.

If we expect to have a bigger network or don't want to implement this type of solution with a cheap device, we can also implement all of this in a dedicated server or even a dedicated computer with an instance running Linux. Although this will be much better in terms of processing power, we have to consider that the power consumption will be much higher.

There is a list of other devices that we could use to achieve this custom firewall considering what we have mentioned before:

- Raspberry Pi Model 2 or higher (with at least 1GB of RAM).
- Arduino UNO.
- Custom computer with at least 2GB of RAM.

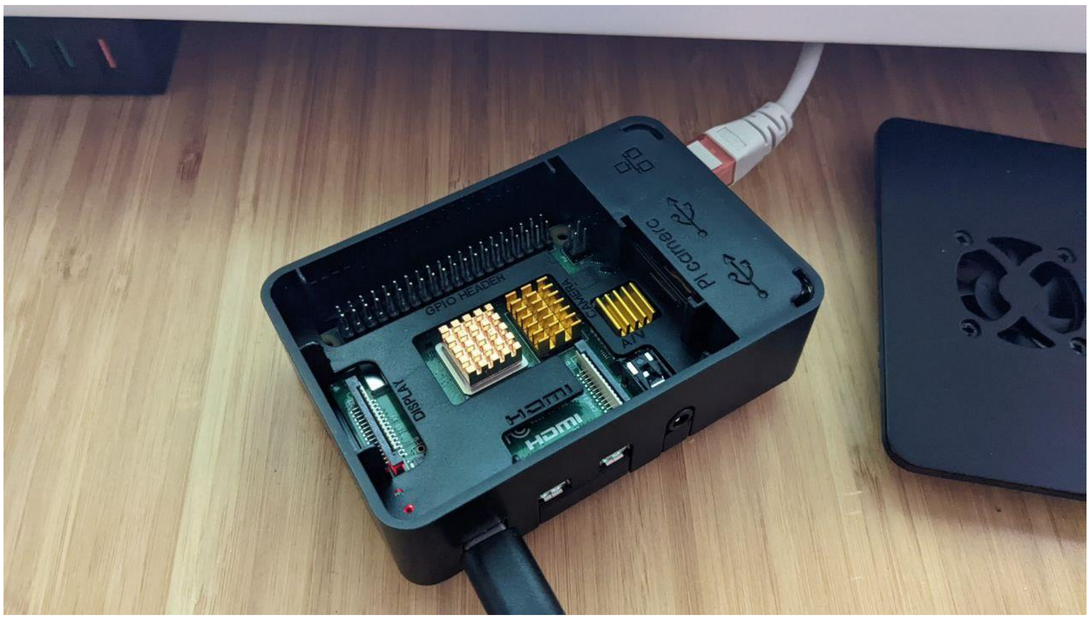


Figure 5. Raspberry Pi Model 4 presentation

As we can see in the previous image our system is implemented with Raspberry Pi Model 4. It also has a little box with a fan that can have their speed regulated Ann helps in controlling the temperature of our system. We also have three cooling pipes in the three main chips that are in our device. Despite the fact that the temperatures are not really high when we are requesting a high performance in packet filtering, it is also a good idea to keep this type of solution in production, because specially for long term development, we would want the lower temperatures as possible in order to maintain secure and stable our infrastructure.

Although the cooling fan worked very well, I had to slow down a little bit the rpm due to the noise that was generated in the room. With this fan in the maximum power the system did not reach the 60 degrees, and with 40% power in the device, the system work stable around 65 degrees. It was when we turn down this speed fan when the temperature went to more than 75 degrees. So I have finally decided to maintain the power of the rpm around 40% of the original one in order to maintain the temperatures as I expected.

Software for custom Firewall (Pi-Hole)

As we have mentioned before we can use different hardware tools to implement these custom Firewall. From now on we are going to consider that we are working with the Raspberry Pi model 4 or with other Raspberry models that at least have 1 GB off ram memory.

For this solution we will implement the Pi-Hole project for firewall monitoring in our network. Pi-Hole is a Linux network-level advertisement and Internet tracker blocking application which acts as a DNS sinkhole and optionally a DHCP server, intended for use on a private network. It is designed for low-power embedded devices with network capability, such as the Raspberry Pi, but can be installed on any Linux machine.

Pi-hole makes use of a modified dnsmasq called FTLDNS, cURL, lighttpd, PHP and the AdminLTE Dashboard to block DNS requests for known tracking and advertising domains. The application acts as a DNS server for a private network (replacing any pre-existing DNS server provided by another device or the ISP), with the ability to block advertisements and tracking domains for users' devices. It obtains lists of advertisement and tracking domains from a configurable list of predefined sources, and compares DNS queries against them. If a match is found within any of the lists, or a locally-configured blacklist, Pi-hole will refuse to resolve the requested domain and respond to the requesting device with a dummy address.

Because Pi-hole blocks domains at the network level, it is able to block advertisements, such as banner advertisements on a webpage, but it can also block advertisements in unconventional locations, such as on Android, iOS and smart TVs.

Using VPN services, Pi-Hole can block domains without using a DNS filter setup in a router. Any device that supports VPN can use Pi-Hole on a cellular network or a home network without having a DNS server configured.

The nature of Pi-hole allows it to also block website domains in general by manually adding the domain name to a blacklist. Likewise, domains can be manually added to a whitelist should a website's function be impaired by domains being blocked. Pi-hole can also function as a network monitoring tool, which can aid in troubleshooting DNS requests and network faults. Pi-hole can also be used to encourage the use of DNS over

HTTPS for devices using it as a DNS server with the cloudflared binary provided by Cloudflare.

The main reason to use be whole as our main software for firewall it is because it is plenty dedicated to a Raspberry model running raspbian and it's very easy to install, set up all the configurations and work with all the data that gives you. Very performance on their interface is very user friendly and there's a huge community that gives support to this tool.

Despite the fact that Pi-Hole it's the most used tool if we want to implement a custom firewall, there are other solutions that could be taken into account in case that the proposed is not what we expected to have. In the following list we will see other tools that provide similar performance and functionalities as Pi-Hole:

- Simplewall[6]
- Blokada[7]
- Adguard[8]
- OpenWRT[9]

It is important to consider that all of these solutions provide custom firewall options in order to deploy them in our network, but they do not offer the same type of options on in the same way as Pi-Hole does, so it is mainly for this reason that in this project we are not going to deploy them despite the fact that some of them are compatible with the solution that we are integrating here.

It is also important to take into account that some of the tools could have problems if they are both deployed at the same time in the same system. It is for this reason that the best approach is to decide which one fits better in our needs and try to adapt or modify the solution to be compliant with what we are expecting to have in the project.

A good approach will be to select a general list, that has a lot of information inside (and also a good timeline of updates) and start working on that to try to define the list in our interests. For this, the best solution is to start with Pi-Hole native or Simplewall, that are the most popular and versatile ones.

[6] https://www.simplewallsoftware.com/installation-guide
[7] https://www.slant.co/versus/24914/32130/~pi-hole_vs_blokada
[8] https://adguard.com/en/welcome.html
[9] https://openwrt.org/

IPS/IDS Software

Similar situation happens when we have to decide which type of IPS or IDS solution we want to implement into our local network. There are several solutions that could be achieved an implemented in Raspberry Pi and in general in a machine that is running Linux. the two main tools that we can consider to implement as an IDS and later on implement some IPS functionalities are Suricata and Snort.

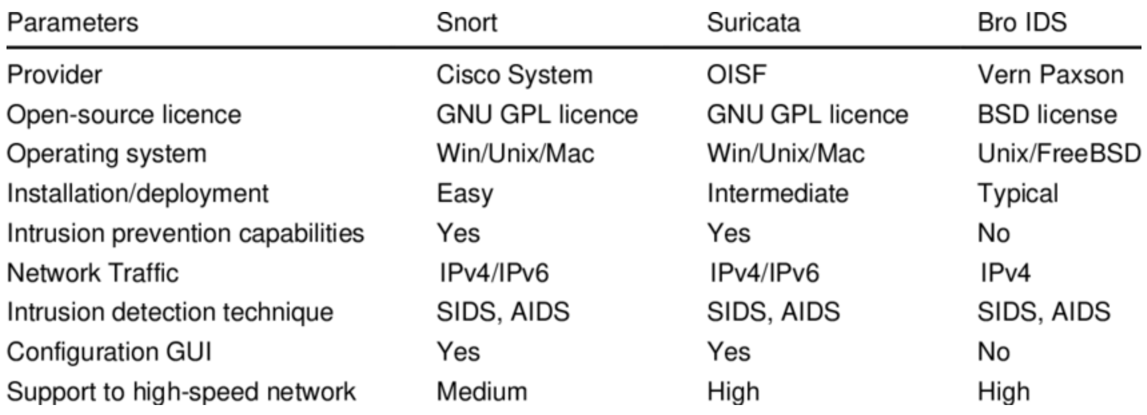
Suricata versus Snort

Since the early days of Snort's existence (Snort is the solution that have been under development much more time), it has been said that Snort is not "application-aware." It simply looks at traffic matching its rules and takes an action (alert, drop and so on) when there is a match. Pre-processors assist by shaping the traffic into a usable format for the rules to apply to: for instance, performing decompression and decoding, but there was no need for Snort to understand what application generated the data.

Suricata works slightly differently in this space. It supports Application-Layer detection rules and can, for instance, identify HTTP or SSH traffic on non-standard ports based on protocols. It will also then apply protocol specific log settings to these detections.

One of the main benefits of Suricata is that it was developed much more recently than Snort. This means it has many more features on board that are virtually unmissable these days. Some of those features are multithreading, file extraction and manual analysis.

While Snort and Suricata are certainly the most popular open-source intrusion detection systems, there are some alternatives. The earlier mentioned updated SNORT3 release looks very promising, with its support for multithreading, service identification and a more straightforward rule language. This has been in development for many years. However, the Alpha stage goes back to 2014, and a release date for a production version has not been set yet.



Parameters	Snort	Suricata	Bro IDS
Provider	Cisco System	OISF	Vern Paxson
Open-source licence	GNU GPL licence	GNU GPL licence	BSD license
Operating system	Win/Unix/Mac	Win/Unix/Mac	Unix/FreeBSD
Installation/deployment	Easy	Intermediate	Typical
Intrusion prevention capabilities	Yes	Yes	No
Network Traffic	IPv4/IPv6	IPv4/IPv6	IPv4
Intrusion detection technique	SIDS, AIDS	SIDS, AIDS	SIDS, AIDS
Configuration GUI	Yes	Yes	No
Support to high-speed network	Medium	High	High

Figure 6. Capture of comparison Snort&Suricata

There are alternatives to the traditional IDS/IPS solutions as well, but these can sometimes work slightly differently. The Bro Network Security Monitor[10] (now known as Zeek), for instance, is more of an anomaly detection system. Where Snort and Suricata work with traditional IDS signatures, Bro/Zeek utilizes scripts to analyze traffic.

It is mainly for this reason that in this project we're going to implement Suricata as the main IDS software running in our platform. It also have a good log management so we will be able, in the future, to process and analyse all of these logs in order to have information for a long time or even for a specific time ago.

This functionality will be very useful when we implement the Telegram bot notifier, that we will explain in the following chapters.

It is also important to remark that in order to deploy a solution like Suricata, we will need, as we have mentioned before, a device that provides the mirroring functionality in order to monitor all the events that are taking place in our network. Without this type of device (router or swich with mirroring) this solution will not work properly because the information will not be resent to the destination that will be in charge of processing this logs.

Despite different parts of devices can be chosen at this point, it is recommended that at least the swich selected is able to provide 100 MB/s of read and write access to the solution, so we will not have problems of dropping and congestion in our network, specially when we are downloading things or consuming high bandwidth.

[10] https://www.bro.org/

General overview of the system

As it has been mentioned during the previous chapters the system has been configured in order to monitor as much as devices as possible. It is for this reason that we will be working with a Raspberry running Raspbian, a switch with port mirroring and finally other types of connexions that will link all the devices (computers, smart phones…) to be able to monitor them.

It is important to remark that the structure that we are going to present here it's completely adapted to our particular case. It can be completely changed in order to be adapted to other types of local network that are smaller or bigger and with different configurations and devices. The most important thing in order to migrate this structure to another one is to maintain the mirroring of the switch and move all the intelligence that it is placed in the main router for default to our Raspberry Pi.

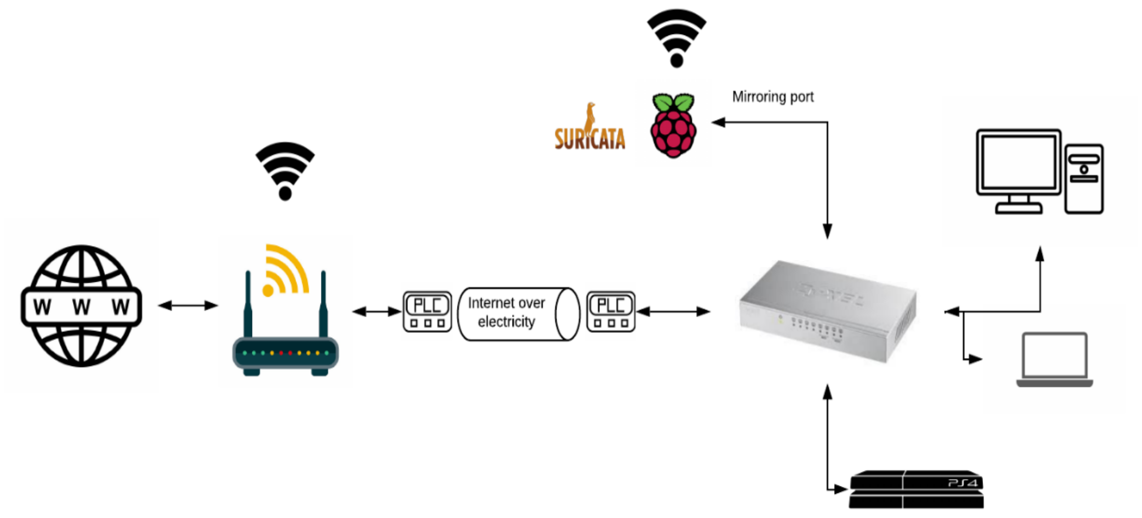


Figure 7. System schematic overview

As we can see in the previous diagram we have our main entrance from Internet through the custom router. This router it's the one that is provided by the company where we have the Internet contracted, and despite the fact that in our particular case it is not possible to make due to the software implemented in the router, one of the most interesting features that can be implemented in other structure it is the lock information sending to our Suricata.

following to the voter we have a connexion through a PLC inserted in my local area network that is connected directly to a switch that has mirroring port options. This switch will become the core part of the network, because all the devices will be connected directly to the switch and the future of mirroring will allow us to monitor all the traffic that it's inside it.

Connected to this switch we have our Raspberry Pi implementing our DNS server as well as IDS monitoring for our infrastructure. We also have several computers connected to the switch.

At this point it is important to remark that in my infrastructure the Wi-Fi of the network is provided by the main router of the company. This has been decided in this manner because for the operativity and necessities in the home, we will not have enough coverage if we place the Wi-Fi server into Raspberry Pi. It is mainly for this reason that the Raspberry does not provide Wi-Fi connexion. But in case this is an option, I will be much more interesting to deploy Wi-Fi service because in that case we will be able to monitor all the traffic through this interface. Furthermore, this will provide much more control with the features included in the Pi-Hole, because as we will see in the conclusions of this scope the performance of the solution is not good enough when we implemented it directly with our Raspberry instead of complementing it through the Wi-Fi.

It is also important to take into account if we analyse this specific infrastructure that, despite the initial idea of having the Raspberry Pi as the core and centre of the network, providing all the services needed to connect to the Internet an working as a major firewall an Internet provider, it has not been possible to deploy it as expected because the performance off the solutions was not as good as we thought initially, so in order to pressurise the correct performance of the network, the Raspberry was in passive mode and sometimes active, so it was able to block some traffic.

Another option that can be interesting to implement but it has not been possible in this project is to try to make a Wi-Fi access point from the Raspberry Pi and a link to another access point in other places of the network. With this, we will be able to deploy an entire wireless access to our network that will be directly managed from the raspberry, no matter which access point is getting the traffic. Thanks to this, we will be able to monitor our entire infrastructure and all of it from the same source.

Raspbian configuration

The first thing that we have to do in order to deploy the system is to configure the operating system that will be running in our Raspberry Pi. This system is called Raspbian and it can be downloaded from the original webpage of Raspberry[11]. The system is the most common and used for this device, and it is always supported by the community and new updates.

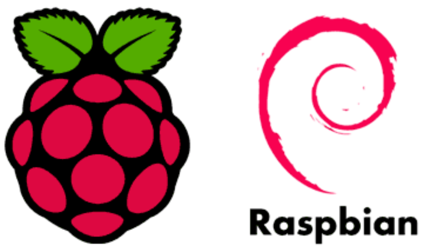


Figure 8. Raspbian Operative System

Raspbian is a free operating system based on Debian optimized for the Raspberry Pi hardware. An operating system is the set of basic programs and utilities that make your Raspberry Pi run. However, Raspbian provides more than a pure OS: it comes with over 35,000 packages, pre-compiled software bundled in a nice format for easy installation on your Raspberry Pi.

The initial build of over 35,000 Raspbian packages, optimized for best performance on the Raspberry Pi, was completed in June of 2012. However, Raspbian is still under active development with an emphasis on improving the stability and performance of as many Debian packages as possible.

It is important to consider that Raspbian is not affiliated with the Raspberry Pi Foundation. Raspbian was created by a small, dedicated team of developers that are fans of the Raspberry Pi hardware, the educational goals of the Raspberry Pi Foundation and, of course, the Debian Project.

Installing Raspbian on the Raspberry Pi is pretty straightforward. We'll be downloading Raspbian and writing the disc image to a microSD card, then booting the Raspberry Pi to that microSD card. For this project, you'll need a microSD card (go with at least 8 GB), a computer with a slot for it, and, of course, a Raspberry Pi and basic peripherals (a mouse, keyboard, screen, and power source). This isn't the only method for installing Raspbian, but it's a useful technique to learn because it can also be used to install so many other operating systems on the Raspberry Pi.

Once the disc image has been written to the microSD card, we will be ready to boot the system. We will have to put that SD into the Rasberry Pi, plug in the peripherals and power source, and connect the HDMI output to an screen. The current edition to

[11] https://www.raspberrypi.com/software/operating-systems/

Raspbian will boot directly to the desktop. The default credentials are username pi and password raspberry.

Once we have all the settings done and we are able to boot into the system, we will have to configure an static IP addres to be able to connect to the system trough a ssh client.

Setting up a Static IP Address on the Raspberry Pi

To begin setting up a static IP address on our Raspberry Pi, we will first need to retrieve some information about our current network setup.

Let's first retrieve the currently defined router for your network by running the following command. With this command we will be able to get out IP and the dimensioning of the network.

ip r | grep default

With the next command, we will be able to modify the config file in order to set the information that we want:

sudo nano /etc/dhcpcd.conf

Information:

interface NETWORK

static ip_address=STATIC_IP/24

static routers=ROUTER_IP

static domain_name_servers=DNS_IP

- NETWORK – your network connection type: eth0 (Ethernet) or wlan0 (wireless).
- STATIC_IP – the static IP address you want to set for the Raspberry Pi.
- ROUTER_IP – the gateway IP address for your router on the local network.
- DNS_IP – the DNS IP address (typically the same as your router's gateway address).

Finally, we only need to reboot the system and the configuration will be implemented. Setting it on the router helps ensure that your router doesn't assign the IP address to a different device before your Raspberry Pi connects.

Pi Hole setup

As it has been metioned in the previous chapters, Pi-hole is a Raspberry Pi based network wide ad blocker. We install the software to a Raspberry Pi running Raspberry Pi OS, run a short installation script and then point our machines to the Raspberry Pi's IP address for instant ad blocking. We can also add the sites you use frequently like tomshardware.com to a whitelist so you can help them keep the lights on, so it can act as a custom blocking firewall for specific urls and services, such as DNS or Web Server.

Despite the fact that there are several softwares that could achieve the same features, Pi-Hole has the following as the most important ones:

- Easy-to-install: our versatile installer walks you through the process and takes less than ten minutes
- Resolute: content is blocked in non-browser locations, such as ad-laden mobile apps and smart TVs
- Responsive: seamlessly speeds up the feel of everyday browsing by caching DNS queries
- Lightweight: runs smoothly with minimal hardware and software requirements
- Robust: a command-line interface that is quality assured for interoperability
- Insightful: a beautiful responsive Web Interface dashboard to view and control your Pi-hole
- Versatile: can optionally function as a DHCP server, ensuring all your devices are protected automatically
- Scalable: capable of handling hundreds of millions of queries when installed on server-grade hardware
- Modern: blocks ads over both IPv4 and IPv6
- Free: open-source software which helps ensure you are the sole person in control of your privacy

As it has been mentioned in the scope of the Rapsberry software, Pi-hole makes use of a modified dnsmasq called FTLDNScURL, lighttpd, PHP and the AdminLTE Dashboard to block DNS requests for known tracking and advertising domains.

The application acts as a custom DNS server for a private network (replacing any pre-existing DNS server provided by another device or the ISP), with the ability to block advertisements and tracking domains for users' devices. It is really versatile and can be modified easily by the user, through the graphic interface or directly from command line.

It obtains lists of advertisement and tracking domains from a configurable list of predefined sources, and compares DNS queries against them. If a match is found within any of the lists, or a locally-configured blacklist, Pi-hole will refuse to resolve the requested domain and respond to the requesting device with a dummy address.

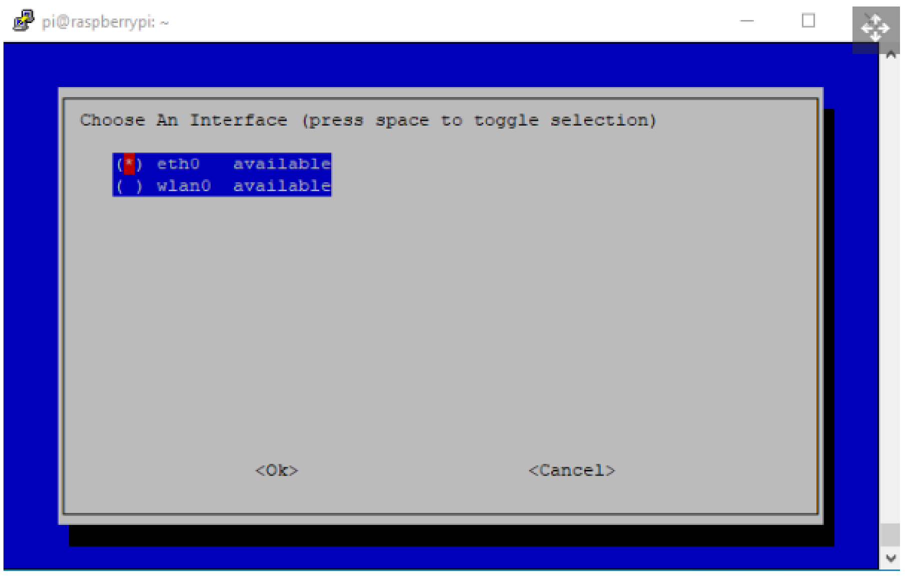
Pi-hole allows it to also block website domains in general by manually adding the domain name to a blacklist. Likewise, domains can be manually added to a whitelist should a website's function be impaired by domains being blocked. Pi-hole can also function as a network monitoring tool, which can aid in troubleshooting DNS requests and network faults. Pi-hole can also be used to encourage the use of DNS over HTTPS for devices using it as a DNS server with the cloudflared binary provided by Cloudflare.

Installing Pi-Hole in Raspberry Pi

The installation of the whole software it's really easy and its guided during all the process, so we are going to explain briefly the main ideas or facts that we have to take into account during the installation but we will not enter deeply in detail because it's really fast and easy to install and deploy this software in our system.

The following steps should be followed in order to install Pi-Hole. During the installation a custom window will appear (the assistant of the installation). Some screenshots will be added in order to clarify the process.

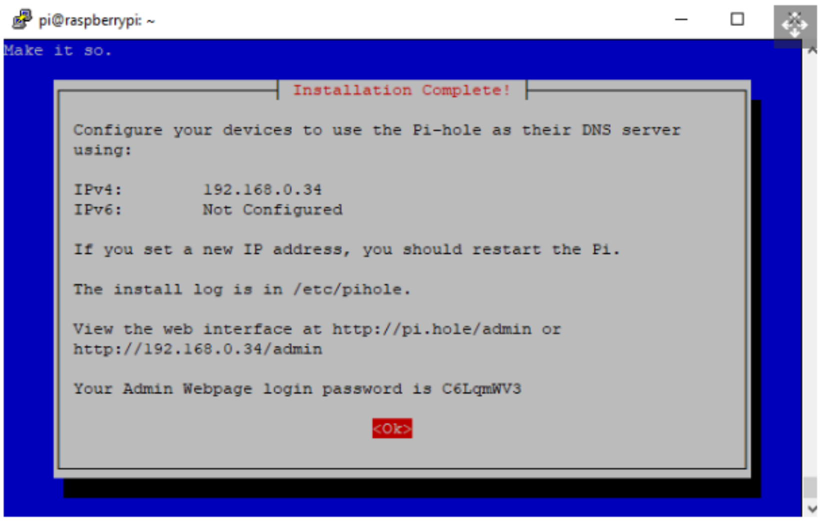
1. **Update your software repositories** and then download the latest updates for your Raspberry Pi. (sudo apt update, sudo apt upgrade -y)
2. **Install Pi-hole**. This command will download the script and then run the installer in the terminal (curl -sSL https://install.pi-hole.net | bash). This will open a window like the following.



```
Choose An Interface (press space to toggle selection)

  (*) eth0   available
  ( ) wlan0  available

            <Ok>                    <Cancel>
```

Figure 9. Configuration screenshot

3. The Pi-hole installer will start by updating the available software, and then a menu based installation wizard will start. **Press Enter** to progress through the installation.
4. **Choose eth0 as the interface** to use with Pi-hole. Press Tab to move the red highlight to Ok and then press Enter. This is our Gigabit Ethernet port which will provide the best possible connection.
5. **Select your upstream DNS provider.** We chose Google, but there are many others to choose from. Press Tab and then Enter.
6. **Accept the default list of blocked sites** by pressing tab and enter.
7. **Accept the default IPv4 and IPv6 protocols** by pressing tab and enter.
8. **Accept the current network settings, and set them as static**. Do make a note of the details as we will need them later.
9. **Install the web admin interface** by pressing tab and enter.
10. **Install the lightppd web server** used to serve the web admin pages by pressing tab and enter.
11. **Accept the default log** options.
12. **Accept the default privacy mode** by pressing tab and enter.
13. The installation is complete and the final page recaps the IP address of the Pi-hole device and provides an admin webpage login password.

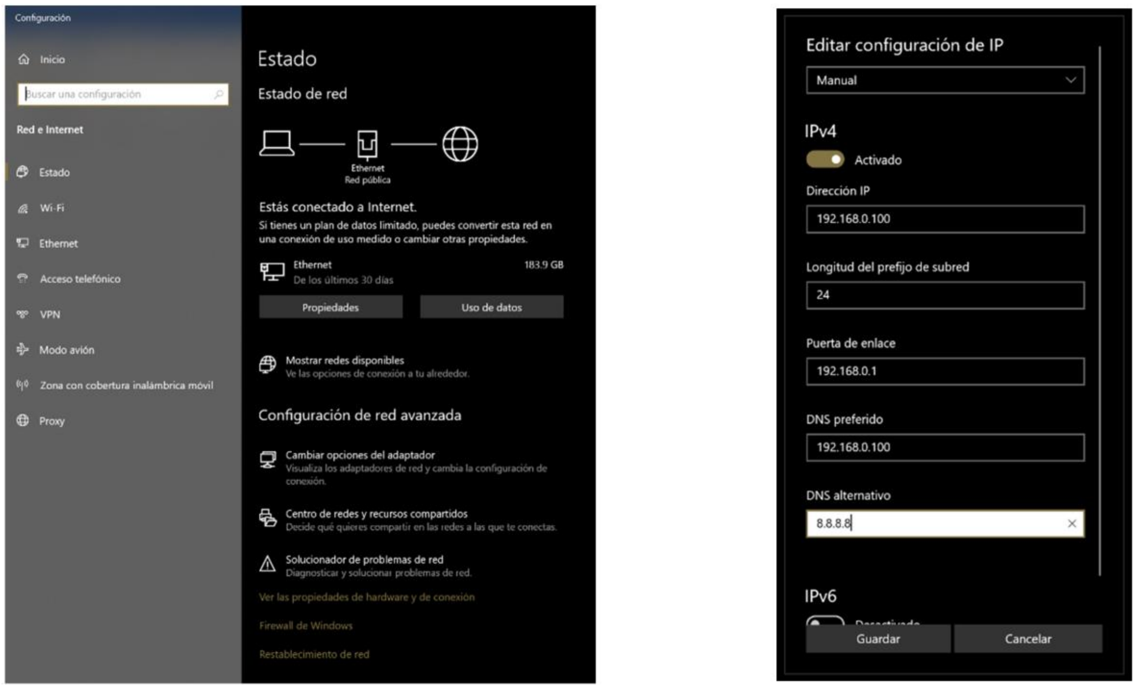


```
pi@raspberrypi: ~
Make it so.

              ┤ Installation Complete! ├
 Configure your devices to use the Pi-hole as their DNS server
 using:

 IPv4:        192.168.0.34
 IPv6:        Not Configured

 If you set a new IP address, you should restart the Pi.

 The install log is in /etc/pihole.

 View the web interface at http://pi.hole/admin or
 http://192.168.0.34/admin

 Your Admin Webpage login password is C6LqmWV3

                            <Ok>
```

Figure 10. Configuration screenshot

Once we have finished all the installation we will have our Pi-Hole so for running inside our Raspberry, but we have to take into account that this is not the end of the configuration. With our Raspberry Pi running Pi-hole setup and running we now need to point our devices to it so that Pi-hole's DNS servers can block unwanted advertisements. Here we are manually setting up a wired network connection with Windows 10. The steps will be similar for Wi-Fi. This is a particular configuration for Windows 10, but can also be implemented in other system such as Linux or Mac, the only difference is that the process will be a little bit different.

1. Right click on the Windows logo and select Network Connections.
2. Click on Properties.
3. Click on Edit to update the network configuration.
4. **Edit your IP address** to your desired address. We chose to stick with what the router's DHCP server issued. Set the Subnet prefix length to 24. The Gateway is the IP address of the router, in our case 192.168.0.1. Preferred DNS is our Pi-hole DNS server, 192.168.0.100. The Alternative DNS is used if our Pi-hole device goes offline, in this case it is Google's DNS server. Click Save to write the changes and restart the network interface.


Configuración

Inicio

Buscar una configuración

Red e Internet

Estado

Wi-Fi

Ethernet

Acceso telefónico

VPN

Modo avión

Zona con cobertura inalámbrica móvil

Proxy

Estado

Estado de red

Ethernet
Red pública

Estás conectado a Internet.
Si tienes un plan de datos limitado, puedes convertir esta red en una conexión de uso medido o cambiar otras propiedades.

Ethernet 183.9 GB
De los últimos 30 días

Propiedades | Uso de datos

Mostrar redes disponibles
Ve las opciones de conexión a tu alrededor.

Configuración de red avanzada

Cambiar opciones del adaptador
Visualiza los adaptadores de red y cambia la configuración de conexión.

Centro de redes y recursos compartidos
Decide qué quieres compartir en las redes a las que te conectas.

Solucionador de problemas de red
Diagnosticar y solucionar problemas de red.

Ver las propiedades de hardware y de conexión

Firewall de Windows

Restablecimiento de red

Editar configuración de IP

Manual

IPv4

Activado

Dirección IP: 192.168.0.100

Longitud del prefijo de subred: 24

Puerta de enlace: 192.168.0.1

DNS preferido: 192.168.0.100

DNS alternativo: 8.8.8.8

IPv6

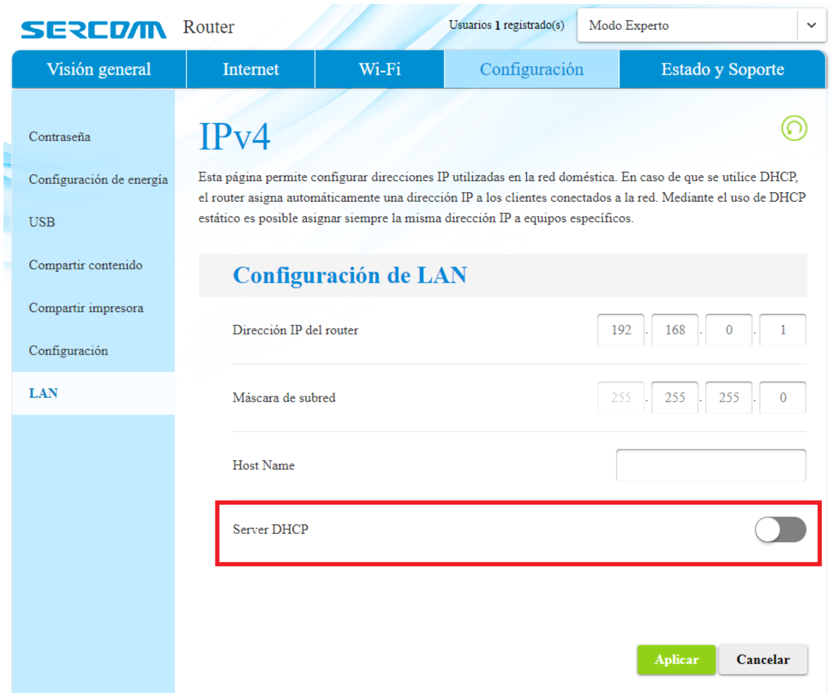
Guardar | Cancelar

Figure 11. Windows configuration screenshot

At this point it is important to remark that we have different options regarding the DHCP server. As we have explained before we can configure device by device which will be their main IP for this service. In the previous case the Windows 10, on boot, will communicate with our Raspberry Pi and will ask for an IP assignment. If the service of Pi-Hole is enable, it will provide connection and our device will be able to connect to our LAN.

But we have to consider that this process is not very scalable and depending on the device that we are working with, it will not be possible to configure this type of Connexions. Furthermore, this is the specific configuration for Windows 10, but we will have to know all the configurations depending on all the systems that we have, and in most of the cases they won't be able to establish this type of Connexions, so they won't have an IP provided by our server in Pi-Hole.

It is for this reason that there's another option where we can deal with this problem, but we have to have direct access to our main router in our network. By going to our main browser web page (usually in 192.168.0.0 or 192.168.0.1) and accessing to the advanced tools (once we log in), we will be able to disconnect the DHCP service from our main router.



IPv4

Esta página permite configurar direcciones IP utilizadas en la red doméstica. En caso de que se utilice DHCP, el router asigna automáticamente una dirección IP a los clientes conectados a la red. Mediante el uso de DHCP estático es posible asignar siempre la misma dirección IP a equipos específicos.

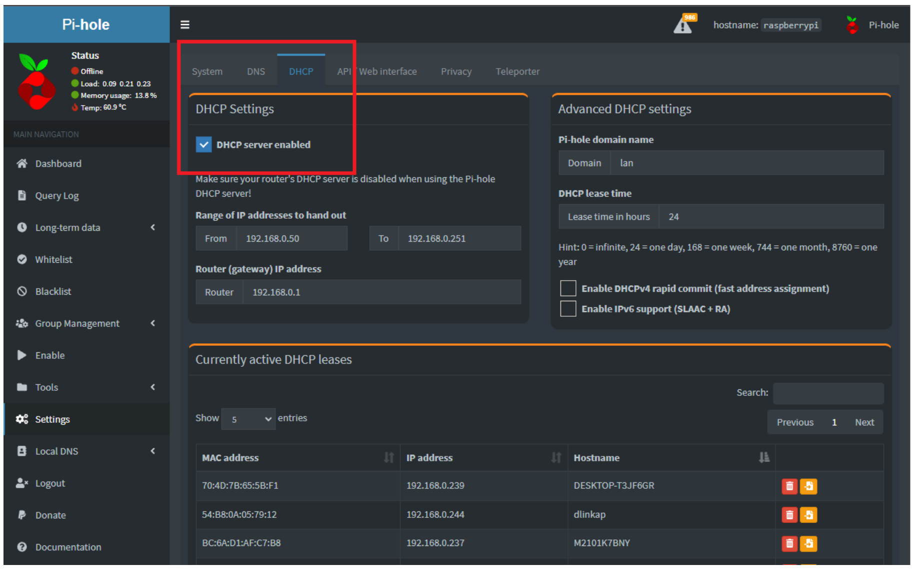
Configuración de LAN

Dirección IP del router	192 . 168 . 0 . 1
Máscara de subred	255 . 255 . 255 . 0
Host Name	
Server DHCP	off

Figure 12. Sercomm router configuration screenshot

In my particular case I have a SERCOMM router that has this options in the LAN configuration part. As we can see in the image the server option is not activated.

It is also important to take into account that if we don't activate the DHCP server in our main router, we have to activate this functionality in our Raspberry Pi Pi-Hole software. This is located in the setting parts, on DHCP menu as “DHCP server enabled”, as we can see in the image.



DHCP Settings

[✓] DHCP server enabled

Make sure your router's DHCP server is disabled when using the Pi-hole DHCP server!

Range of IP addresses to hand out
From 192.168.0.50 To 192.168.0.251

Router (gateway) IP address
Router 192.168.0.1

Advanced DHCP settings

Pi-hole domain name
Domain lan

DHCP lease time
Lease time in hours 24

Hint: 0 = infinite, 24 = one day, 168 = one week, 744 = one month, 8760 = one year

[ ] Enable DHCPv4 rapid commit (fast address assignment)
[ ] Enable IPv6 support (SLAAC + RA)

Currently active DHCP leases

Show 5 entries Search: Previous 1 Next

MAC address	IP address	Hostname
70:4D:7B:65:5B:F1	192.168.0.239	DESKTOP-T3JF6GR
54:B8:0A:05:79:12	192.168.0.244	dlinkap
BC:6A:D1:AF:C7:B8	192.168.0.237	M2101K7BNY

Figure 13. Pi-Hole configuration screenshot

It is important to consider that this configuration has to be done depending on our needs an on our network usage. If this type of service is enabled, all the devices in our network, not only the ones that are already connected but also the ones that aren't connected now, will communicate with the Raspberry in order to obtain a valid IP to connect to our network. So, in case our main DHCP server is not running, there will be no way to connect to our network.

To sum up all the options that we have in that particular case, from the one hand we have the option of configuring each device independently to point out to our main server (in that case, the Raspberry Pi) to obtain the configuration to be connected in our network. This is a long process but it gives you the certainty that only the devices that will be connected and configured independently will be able to communicate with our main server in order to obtain this configuration. On the other hand, we can make this configuration scalable by disabilitating the DHCP server from our main router and activate this option in our Pi-Hole software. This is a much faster and easy configuration but we have to consider that all the devices that want to connect to our network will have to communicate with our Raspberry Pi.

Despite the software provides you a correct IP to log in the network, I have experimented some delay when a new device, especially if this device have never been connected to the network, in the enrollment and the IP assignment. It is not a huge problem because it works almost always, but the connexion to the network it is a little bit slower than if the main server is located in the router. I have experimented more delay in those devices that are running Android or iOS as a main operating system. The other devices, running windows or Mac, are working fine and without any delay. This could be a problem related with the connexion between the Raspberry and the switch, because I have always had activated the logging through the mirror report an there are a lot of locks that are being collected for the software, so they might interfere in the connexion speed.

It is also important to take into account that if we reset the router for whatever reason, the default configuration of the router will be working again and the IP assignment will be performed by the router and the Raspberry. This means that will be collision between the two solutions, and despite I have been making different tests in this stuff, it is still working fine but with much more delay. This is because Pi-Hole software acts as a secondary one, so the IP assignment is always made by the router instead of by the Raspberry.

Pi Hole configuration

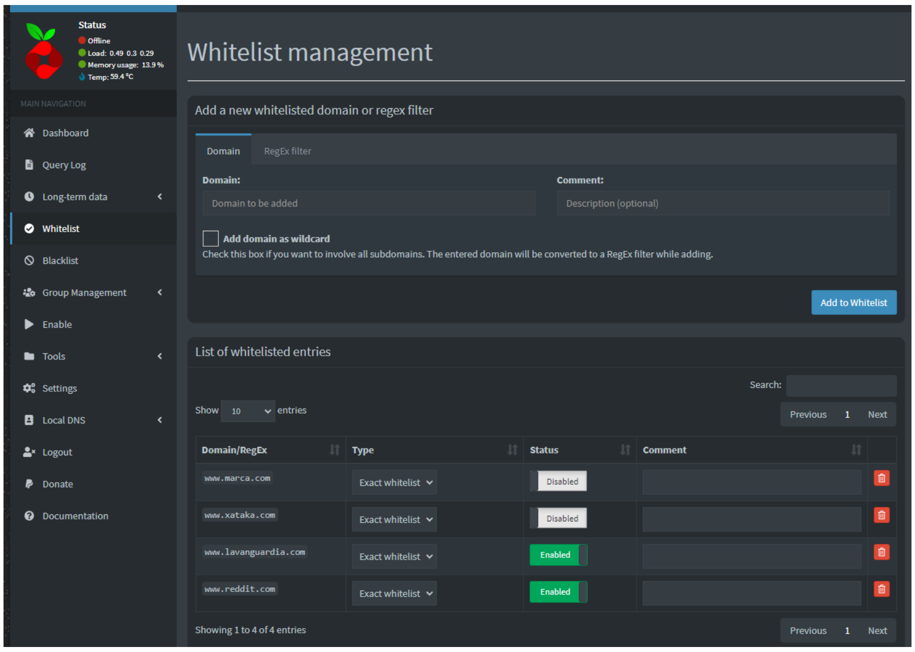
In this chapter we are going to present how the people software have been configured. It is important to consider that this solution has different many ways of configuration, because there are a lot of options to take into account depending in our needs and the way we want it to perform in our network. In this chapter I am going to introduce the basic configuration that half to be set up in the system and I am going to explore all the different options that we can introduce an modify in order to obtain Results that we want to have.

During the following chapter we are going to mention the most important configurations that we can take into account in Pi-Hole, and later we are going to mention those ones that despite they are not activated in my particular setup, it could be interesting if we want to have a major control of the network and what is going on inside it.

How to Whitelist a Site in Pi-Hole

Considering that many content sites, rely on advertising for a significant portion of their revenue, it makes sense to whitelist those you wish to support so they can serve you ads. Pi-hole has a whitelist menu where we can add specific domains and subdomains which will be added to Pi-hole’s lists.

1. Open a web browser to the IP address of your Raspberry Pi, and type /admin. For our setup we went to http://192.168.0.100/admin/
2. Log in to your Pi-Hole by entering the username and password. In the initial setup this information have been configured.
3. From the dashboard click Whitelist.
4. Under Domain, add the URL of the site that you wish to whitelist, then click Add to Whitelist. This site will now be able to serve adverts. Domains can be enabled / disabled in the whitelist by clicking on the button under Status. Whitelisted domains can be deleted by clicking on the trashcan icon.
5. Click logout to return to the default admin page.



Whitelist management

Add a new whitelisted domain or regex filter

Domain | RegEx filter

Domain: Domain to be added

Comment: Description (optional)

Add domain as wildcard
Check this box if you want to involve all subdomains. The entered domain will be converted to a RegEx filter while adding.

Add to Whitelist

List of whitelisted entries

Search:

Show 10 entries

Previous 1 Next

Domain/RegEx	Type	Status	Comment
www.marca.com	Exact whitelist	Disabled	
www.xataka.com	Exact whitelist	Disabled	
www.lavanguardia.com	Exact whitelist	Enabled	
www.reddit.com	Exact whitelist	Enabled	

Showing 1 to 4 of 4 entries

Previous 1 Next

Figure 14. Pi-Hole whitelist screenshot

As we can see in the previous figure it is really easy and intuitive to introduce a URL into our white list. Furthermore, we can enable or disable during the process, because it's easy and applied in real time. We can also delete all of this information or at it through config files, so if we if we have a huge list of URLs that we can download from the Internet or make by ourselves, we can upload all of the list through the config files an we don't need to add them one by one.

How to Blacklist a Site in Pi-Hole

In the same way that we have explained in the previous chapter, we have the option of blocking and specific webpage with its URL. This might be one of the most important features of this solution, specially if we have kids on our network or we don't want traffic in on specific content of the Internet.

Despite the fact that we can add, in the same way that the white list do, several URLs to be blocked and deny the traffic from there, the huge potential of this configuration is to obtain lists from the Internet that contain all the web pages for regarding one specific content or theme. As Pi-Hole is an open source solution we can find a lot of those lists in the Internet. I am going to mention some of the most important ones that I have enabled in the software:

- **The Firebog (WaLLy3k)[12]:** The lists found at The Firebog are separated several ways. First, the lists are separated into categories:
  - Suspicious
  - Advertising
  - Tracking & Telemetry
  - Malicious
  - Other

  Then, they're separated into green and blue. Green is the least likely for breakage, whereas blue lists are more likely to break things.
  I personally recommend using 1 to 2 blocklists from the Advertising, Tracking & Telemetry, and Malicious sections. You should avoid the crossed out lists. Feel free to experiment mixing the more aggressive "blue" lists with the less aggressive green ones.
  For many users, the categories and green/blue lists found here should cover what you need and/or want your PiHole to block.

- **Developer Dan (lightswitch05)[13]:** Most users will want to checkout the Ads & Tracking list and the Google AMP hosts list. You can experiment with the Tracking Aggressive as well. These lists are well maintained and updated very frequently.
  Personally, I use the Tracking Aggressive list and found it fits the bill for good blocking and functionality. A lot of users have noticed breakage when using this list, as it can be read in their official webpage, so we have to be prepared to remedy rectify if breakage occurs in your own use of this particular blocklist.
- **OISD Domain Blocklist[14]:** Basic primarily blocks advertisements whereas Full contains everything from advertisements, malware, scam/phishing, telemetry, tracking, etc. Additionally Full includes everything from the Basic and NSFW lists. The Full list is massive and incorporates a lot of smaller blocklists. If you run this one, chances are you won't need to run any other lists as there will be a lot of needless overlap.
  The NSFW list blocks domains that are known to host pornographic content not limited to known porn streaming/downloading sites.
  However, this results in you having to place a lot of trust in a single party. You also will not be able to assign different lists, which negates the "Group

[12] https://firebog.net/
[13] https://github.com/lightswitch05/hosts
[14] https://oisd.nl/?p=dl

management" feature of PiHole. Group management has the capability of applying different blocking rules to different user-defined "groups."

- **RegEx Blocklist[15]:** PiHole features RegEx (regular expression), which can create more complex filter rules for your PiHole set up. This is often described as an "advanced" function, but any user can take the time to learn how to properly write RegEx entries.
  RegExes are actually used in a variety of applications -- not just Pi-Hole. Perhaps the main purpose for RegEx is for filtering, most notably while performing a search. The search function (CTRL + F) in your browser is an excellent example of RegEx filtering as a search function; the page gets "filtered" based on what you input into this search function.

As we can see in the previous list there are a lot of options that we can implement in order to block specific traffic in our network. In my particular case I started blocking a lot of this traffic with the Firebog block list, because as we can read in the main forums of Raspberry, it seems one of the best solutions regarding the performance on the options that we have in the configuration.

Despite that it is important to consider that if we block a lot of traffic we can have strange performance in our network because nowadays a lot of web pages do not work if we block the advertising at all. So firstly, I have implemented an strategy of blocking as much as URLs as possible, but later on I realised that almost all the directions and web pages that I usually visit didn't work as expected, so I finally decided to shut down this feature priorizing the user experience in the network.

This does not mean that we are not able to implement this type of solution, this means that we have to be very cautious off what type of configuration are we implementing in the system. The best approach will be to block proportionally during time, so we can add knew block list if we are experimenting a good performance in the network. I strongly do not recommend to block everything and start unblocking.

Other Pi-Hole tools

Despite the fact that Pi-Hole have a lot of features that we can tune depending on our needs, there isn't a specific place in the software overview where we can get a lot of

15 https://github.com/mmotti/pihole-regex/blob/master/regex.list

information of our network and the behaviour of the traffic that is going on there. I am talking about the tools sector. In this specific part of the solution we can get information about the following topics that are very interesting if we want to have a monitoring of our network.

- **Pi-hole diagnosis:** This is one of the most interesting parts of the solution. In this chapter we are going to obtain real time information about the alerts or messages considered important by the software that we have to take into account in order to be sure that the traffic inside our network is performing as we expect. In this chapter we are going to obtain warnings regarding “dnsmasq” performance and related with this feature. It will be very useful in case we have an specific problem with one of the urls or webpages that is not working well despite it is not included in our white or black list.
- **Update Gravity:** As we have explained before in this chapter we will be able to update our list of blocked domains. By clicking the button update our Pi-Hole software will be automatically updated with the last features that we can implement. It is important to remark that this blocked domains are a part of the software, so if we want to add more we have to follow the previous chapter.
- **Query Lists**: This is a great list where we can find blocked domains in our network. It is really useful to complement the searches with the first pi-hole diagnosis chapter.
- **Audit log:** Despite the fact that we have a lot of log files in the system, with this functionality it will be very intuitive and easy to compare which queries are allowed and which ones are blocked in our system. It is really interesting to see that if we take our Internet browsing as a reference the major part of allowed unblocked queries will be related to the ones in Google, because all the advertising comes from this specific web pages. If we want to work with this data in a deeper way we will need to go to the lock files that are inside the Raspbian system.
- **Tail pihole.log, Tail pihole-FTL.log and debug log:** As we can deduce from the name of this two chapters, they provide us a list of the locks that are contained in the system. Thanks to this real time logs we can see how the system is performing, blocking specific traffic and making some warnings regarding our activity in the network. Despite that, the user interface is not really easy to use, so if we want to get more information or all the logs to perform statistics or calculations, we will need to go to the logs of the system.

- **Network**: From my point of view this chapter is the most UN useful one of all the tools that are in the software. This chapter gives us information about which devices are connected in our network and some of their specifications (not only the IPS or Mac addresses, but also the number of queries that they have made and a host name in case our software can get this information). This chapter is really useful to get a list of all the devices that are connected in our network or all the devices that once up on a time were connected inside it. Thanks to this we can get information about if there is any device that it is not expected to be in our network or if there is any device that it's making a lot of queries in our infrastructure.

  To make an interesting example, I've had in my local other network an smart device that controls the power consumption of one of my computers. Thanks to this network tool I have been able to detect that this device is performing, in some cases, more queries than even my smartphone. this is really impressive because this device was saturating my network with queries to work.

Pi-Hole I have several other tools that can be useful in order to monitor our infrastructure and configuration of this custom firewall. We are going to enter more into the detail of all the dashboard part in order do describe which are the most interesting parts to monitor our network thanks to the graphics.

We also have a group management but that allow us to load more information or lists in the system, introduce particular information of our client detection and make groups of interest of those clients.

Suricata configuration

As it has been mentioned during the thesis, Suricata is an open source network threat detection engine that provides capabilities including intrusion detection (IDS), intrusion prevention (IPS) and network security monitoring. It does extremely well with deep packet inspection and pattern matching which makes it incredibly useful for threat and attack detection.

While many of the features and functionalities are similar to Snort – Suricata is different in several important ways:

- It’s multi-threaded so a single instance can perform at much higher traffic volumes, so it is scalable.
- There is more support available for application layer protocols;
- It supports hashing and file extraction; and
- It has hooks for the Lua scripting language, which can be used to modify outputs and even create complex and detailed signature detection logic.

One of the distinguishing traits of Suricata, especially in comparison to Snort that we have mentioned in the previous chapters, is that it has a dynamic protocol protection capability that is port agnostic. This means it can identify some of the more common application layer protocols, like HTTP, DNS, TLS, when these are communicating over non-standard ports. The rule language allows you to construct matching conditions in the application layer protocol to a much greater extent than comparable IDS tools.

For example, you can match HTTP header fields and values, or write rules to look at the HTTP post body. This gives you an awareness of the context for that network transaction, which can influence that matching logic that you’re using. By comparison, with other IDS tools, you’d write a rule that looks for a content match – a certain string inside a packet payload – without that context.

To be clear, it is possible to understand this context without the additional application layer protocol support – it just requires a deeper level of understanding around the packet and protocol structure. In those cases, context is applied to content by matching the byte values at predefined offsets – the distances from one another – that represent demarcations in the packet structure. This is complicated and easy to get wrong.

The application layer support Suricata provides simplifies this dramatically. Instead of having to know specific byte values and field lengths, if you want to match on a value in an HTTP host header you simply use the rule option keyword: http_host. This is much easier to get right.

So it is clear that thanks to this solution we are able to monitor, in a very tuned way, the different events that are taking place in our network and specially those ones that are interesting to monitor for us, like specific traffic or events that are not normal in our particular case.

Regarding intrusion prevention mode, in the Bricata platform, the administrator chooses whether they want to configure the system in detection or prevention mode. If our sensors are deployed in-line, you choose prevention mode and the Suricata engine drops the packets in-flight if it determines a rule matches those packets. It’s worth pointing out that this isn’t a point of differentiation because Snort works this way too.

Port mirroring configuration

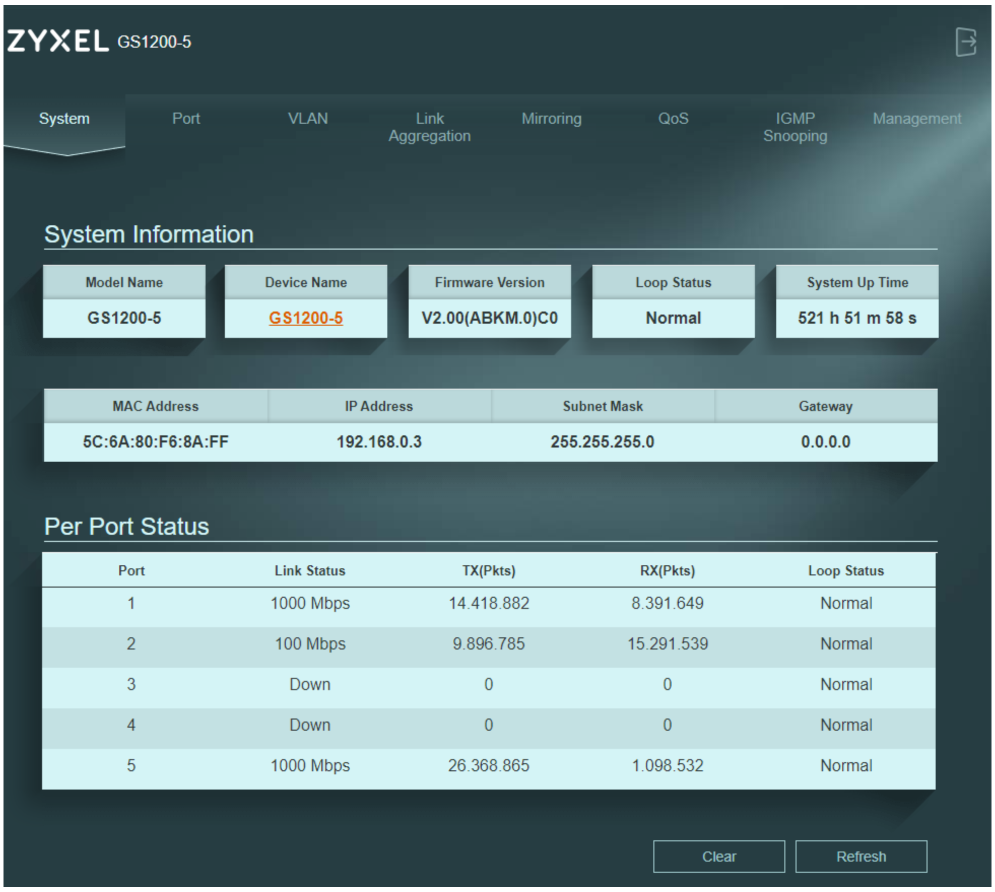
As we have mentioned during the thesis, one of the most important things to implement our IDS tool, is to have a switch with a mirroring function activated. This will allow us to gather all the events of our network an pass them to the Suricata software. Without this solution, we won't be able to gather all this information an hour Suricata we will not apply that intelligence that we are expecting. It is also important to mention that the switch has to be placed in on specific position, so our Raspberry Pi can receive all these logs.

In order to clarify how this configuration has to be made, I am going to present my particular case with my Swich Zyxel GS1200. Despite the fact that all the switches are different and specially the software that they are running, we only have to consider that they have port mirroring, so we will have to enter to the configuration page of the router an activate this function.

Figure 15. Zyxel swich

In my case my switch is placed in the IP 192.168.0.3. It is important to remark that the default configuration placed the switch in the 192.168.1.3 IP, so previously I had to modify it through the configuration page to place this switch in the same network turn all the system.

As we will see in the following image, we have five ports with different specifications and traffic information. We will have to go to mirroring configuration page and specify all the characteristics. In my particular case the port that is monitoring all the mirroring traffic it’s the number 5, and this port is mirroring the rest of the switch despite the fact that not all of them have traffic.



System Information

Model Name	Device Name	Firmware Version	Loop Status	System Up Time
GS1200-5	GS1200-5	V2.00(ABKM.0)C0	Normal	521 h 51 m 58 s

MAC Address	IP Address	Subnet Mask	Gateway
5C:6A:80:F6:8A:FF	192.168.0.3	255.255.255.0	0.0.0.0

Per Port Status

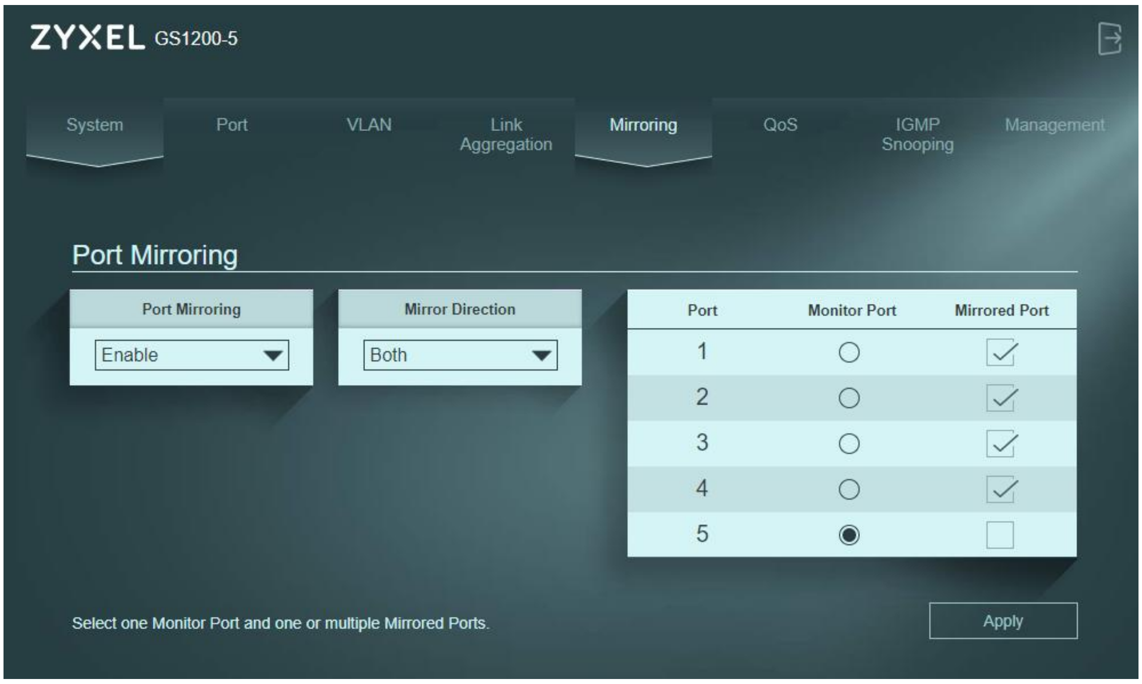
Port	Link Status	TX(Pkts)	RX(Pkts)	Loop Status
1	1000 Mbps	14.418.882	8.391.649	Normal
2	100 Mbps	9.896.785	15.291.539	Normal
3	Down	0	0	Normal
4	Down	0	0	Normal
5	1000 Mbps	26.368.865	1.098.532	Normal

Figure 16. Zyxel swich configuration screenshot

This is an interesting way to establish this configuration because in the future I won't be able to remember which port is making the mirroring for the rest of the ports that are configured, so if we want to monitor all the network and as we have seen the solution is scalable, it is recommended to mirror all ports so if in the future we use one of the ports that now is free we will be gathering this information too automatically.

In case that we are interested only in the monitoring an specific part of the network, the best approach will be to set only one or two ports mirrored (that will be the ones that are being used to monitor all the events with Suricata), and the rest use them as are default ones without any specific configuration.

It is also recommended that the port that has the major part of the traffic is the one that is performing the monitoring, because although it is quite difficult to achieve a traffic higher than the swich limitations, in case that we are performing some activities in the network like TV streaming or downloading, we might have problems if we set the connexion of monitoring to a 100 Mbps limit. It is mainly for this reason that my monitoring port is the number 5, because it is the one (like number 1) that has a higher traffic possibility.



Port Mirroring

Port Mirroring	Mirror Direction
Enable	Both

Port	Monitor Port	Mirrored Port
1	○	✓
2	○	✓
3	○	✓
4	○	✓
5	●	☐

Select one Monitor Port and one or multiple Mirrored Ports.

Figure 17. Zyxel swich port configuration

With all of this configuration we should be able to send all the events that are being generated in our network directly to our Raspbery Pi. We have to consider that all of those devices that are not connected to our switch will not be monitorized by Suricata. It is for this reason, as it has been mentioned during the thesis, that if we want to monitor the Wi-Fi network, we will have to connect an access point or a router in our switch or in our Raspberry Pi directly.

In that case, once we have this configuration working, it will be as easy as connect an access point directly to one of the mirrored ports that are not used, for example in my case it will be a good option if we use the number 3 or 4 ports. By connecting a router or an access point to this sports we will be able to send all the information directly to Suricata.

In case that we have said the configuration with only those ports that are being used with the mirror function activated and we set an access point directly connected to this switch we will have to configure the port that will be used in the configuration page in order to be able to send the events to Suricata.

Installation process

To install Suricata in our Raspberry Pi we only need to follow the next steps. The way to install it is very easy, but we have to take into account several steps in order to be able to configure the tool as we want later.

1. Prepare the installation, by installing the necessary dependencies:

   sudo apt install libpcre3 libpcre3-dbg libpcre3-dev build-essential libpcap-dev libyaml-0-2 libyaml-dev pkg-config zlib1g zlib1g-dev make libmagic-dev libjansson-dev rustc cargo python-yaml python3-yaml liblua5.1-dev

2. Download Suricata source code. In this step we will have to change the version number if newer.

   wget https://www.openinfosecfoundation.org/download/suricata-6.0.1.tar.gz
3. Untar and install the package downloaded.

   tar -xvf suricata-6.0.1.tar.gz

   cd $HOME/suricata-6.0.1/

   ./configure --prefix=/usr --sysconfdir=/etc --localstatedir=/var --enable-nfqueue --enable-lua
4. Compile and install the Suricata

   make

   sudo make install

   cd $HOME/suricata-6.0.1/suricata-update/
5. Build suricata-update and install suricata-update

   sudo python setup.py build

   sudo python setup.py install

   sudo make install-full

   sudo suricata-update

Once we finish all this steps we will have the solution installed in our system (Raspbian for Raspberry Pi Model 4). Now, we will need to set up all the configuration for our particular network and then launch the process to make it run.

The configuration file that has to be modified is: */etc/suricata/suricata.yaml.* We will have to change HOME_NET value to our lan configuration, in my particular case 192.168.0.0/24. So the variable will be: *HOME_NET: "[192.168.0.0/24]".*

Finally, we will have to launch the solution with the following command:

sudo suricata -c /etc/suricata/suricata.yaml -i eth0 -S var/lib/suricata/rules/suricata.rules

Remember that the mirroring port of our swich, that we configured previously, must be working.

Testing Suricata performance

In order to test whether Suricata is running properly, it is useful to add a rule that displays a warning each time an ICMP Echo (ping) is received. This rule should be removed after this test. It is important to mention that all the rules that are implemented in Suricata will be placed in the following directory: *"/var/lib/suricata/rules/suricata.rules".*

To make this test, we will add: *alert icmp any any -> any any (msg: "ICMP Packet found"; sid: 1; rev: 1;)*

Then, we will have to wait some minutes in order to by able to capture some of this traffic and we will have to go to the "fast.log" file, placed in the directory: */var/log/suricata/fast.log.* We can "tail" this log in order to see the last events regarding this configuration. At this point we should be able to read, in this configuration file, an alert like the following:

04/10/2022-19:42:23.757598 [**] [1:1:1] ICMP Packet found [**] [Classification: (null)] [Priority: 3] {ICMP} 192.168.0.100:8 -> 192.168.1.25:0

Once we have detected an alert like the previous one, we will have to delete the rule implemented and do some network usage. We can wait several days in order to detect some interesting traffic, because the default configuration of Suricata will only notify the user in case that some traffic is suspicious according to their intelligence, so we can force this to happen or wait until this happen naturally. This will allow us to see some interesting logs generated by Suricata, as the following capture:

04/19/2022-09:57:40.853554 [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.100:40272 -> 8.8.8.8:53

04/19/2022-09:58:08.568521 [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.151:62273 -> 192.168.0.100:53

```
04/19/2022-09:58:08.569437  [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.100:37842 -> 8.8.8.8:53

04/19/2022-09:58:08.569498  [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.100:37842 -> 8.8.8.8:53

04/19/2022-10:01:12.182409  [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.151:40728 -> 192.168.0.100:53

04/19/2022-10:01:12.183373  [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.100:53088 -> 8.8.8.8:53

04/19/2022-10:01:12.183434  [**] [1:2027863:4] ET INFO Observed DNS Query to .biz TLD [**] [Classification: Potentially Bad Traffic] [Priority: 2] {UDP} 192.168.0.100:53088 -> 8.8.8.8:53
```

As we can see in the previous logs, we can detect that there is some traffic going from 192.168.0.100 (that is our Raspberry Pi Model 4) to other IP, like internet ones (8.8.8.8) or even other network devices, like 192.168.0.151, that is one of my computers connected to the network.

Once we arrive at that point, we will be able to monitor all the traffic of our network, but there are still some configurations to do in order to maintain the service and make all the performance run as expected and in a reliable manner. This is why we will have to set Suricata as a service in our Rasbian system and then make some tuninng in order to avoid the packet loss.

Suricata as a service and packet tunning

To be able to use Suricata as a service, the file "/etc/systemd/system/suricata.service" must be created: /etc/systemd/system/suricata.service.

In this file we will need to add the following lines:

```
# Sample Suricata systemd unit file.
```

[Unit]

Description=Suricata Intrusion Detection Service

After=network.target syslog.target

[Service]

ExecStart=/usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth0 -S /var/lib/suricata/rules/suricata.rules

ExecReload=/bin/kill -HUP $MAINPID

ExecStop=/bin/kill $MAINPID

[Install]

WantedBy=multi-user.target

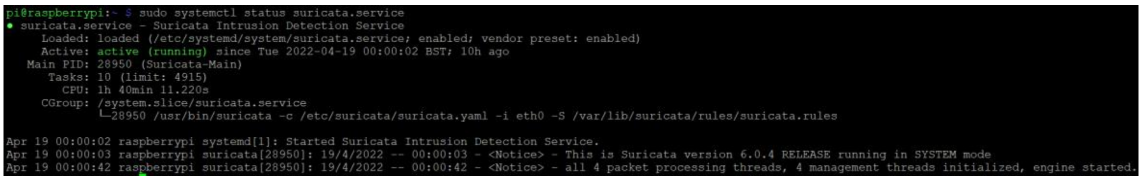
Then, once we have this file defined, we will have to launch the service with the following command:

sudo systemctl enable suricata.service

sudo systemctl start suricata.service

sudo systemctl status suricata.service

We will be getting a message similar to the following one:



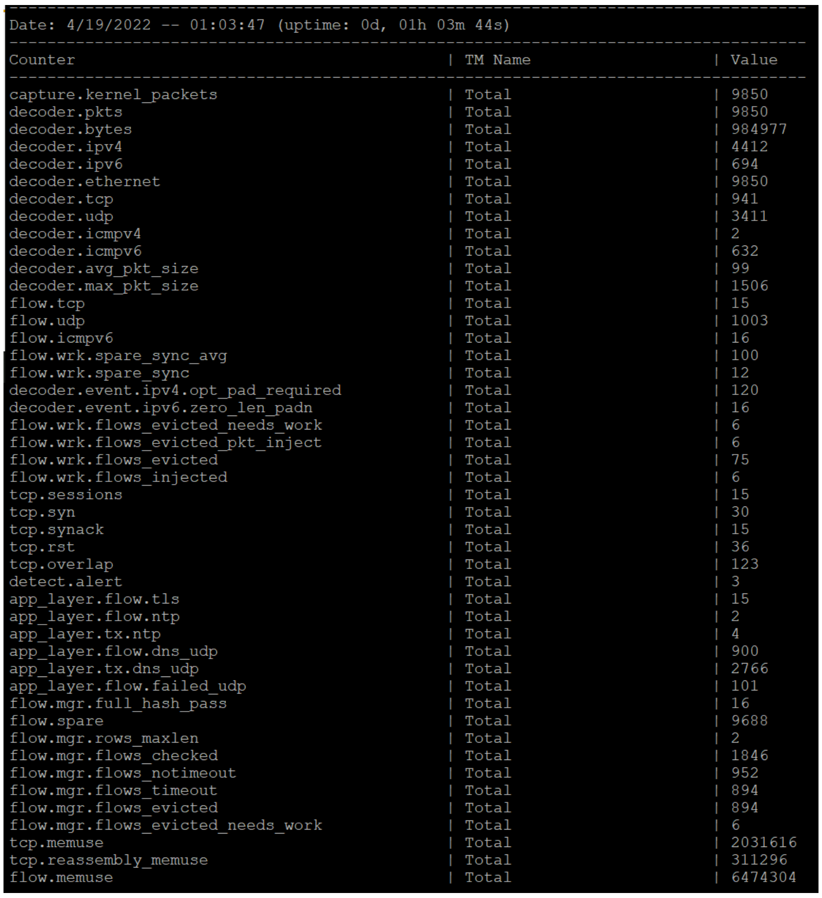
```
pi@raspberrypi:~ $ sudo systemctl status suricata.service
● suricata.service - Suricata Intrusion Detection Service
     Loaded: loaded (/etc/systemd/system/suricata.service; enabled; vendor preset: enabled)
     Active: active (running) since Tue 2022-04-19 00:00:02 BST; 10h ago
   Main PID: 28950 (Suricata-Main)
      Tasks: 10 (limit: 4915)
        CPU: 1h 40min 11.220s
     CGroup: /system.slice/suricata.service
             └─28950 /usr/bin/suricata -c /etc/suricata/suricata.yaml -i eth0 -S /var/lib/suricata/rules/suricata.rules

Apr 19 00:00:02 raspberrypi systemd[1]: Started Suricata Intrusion Detection Service.
Apr 19 00:00:03 raspberrypi suricata[28950]: 19/4/2022 -- 00:00:03 - <Notice> - This is Suricata version 6.0.4 RELEASE running in SYSTEM mode
Apr 19 00:00:42 raspberrypi suricata[28950]: 19/4/2022 -- 00:00:42 - <Notice> - all 4 packet processing threads, 4 management threads initialized, engine started.
```

Figure 18. Suricata service screenshot

This will allow us, in case our Raspberry Pi is shut down or has any problem that need to reboot the system, launch automatically Suricata on boot.

In the following steps, we are going to deal with the packet loss of our network. Suricata may require some optimizations in order to work optimally on your network. This depends of course on the number of devices to be monitored and the traffic generated and the amount of traffic that is being generated. If our network is not big or the activities of our devices is not extremely high, we may not have any packet loss from the beginning, but as this depends on many factors, it is really recommended to check this.

Check that the “capture.kernel_drops” variable in the “/var/log/suricata/stats.log” file is not too high. Ideally, it should remain at 0 and therefore not be displayed in the counter list.



```
------------------------------------------------------------------------------------
Date: 4/19/2022 -- 01:03:47 (uptime: 0d, 01h 03m 44s)
------------------------------------------------------------------------------------
Counter                                       | TM Name                   | Value
------------------------------------------------------------------------------------
capture.kernel_packets                        | Total                     | 9850
decoder.pkts                                  | Total                     | 9850
decoder.bytes                                 | Total                     | 984977
decoder.ipv4                                  | Total                     | 4412
decoder.ipv6                                  | Total                     | 694
decoder.ethernet                              | Total                     | 9850
decoder.tcp                                   | Total                     | 941
decoder.udp                                   | Total                     | 3411
decoder.icmpv4                                | Total                     | 2
decoder.icmpv6                                | Total                     | 632
decoder.avg_pkt_size                          | Total                     | 99
decoder.max_pkt_size                          | Total                     | 1506
flow.tcp                                      | Total                     | 15
flow.udp                                      | Total                     | 1003
flow.icmpv6                                   | Total                     | 16
flow.wrk.spare_sync_avg                       | Total                     | 100
flow.wrk.spare_sync                           | Total                     | 12
decoder.event.ipv4.opt_pad_required           | Total                     | 120
decoder.event.ipv6.zero_len_padn              | Total                     | 16
flow.wrk.flows_evicted_needs_work             | Total                     | 6
flow.wrk.flows_evicted_pkt_inject             | Total                     | 6
flow.wrk.flows_evicted                        | Total                     | 75
flow.wrk.flows_injected                       | Total                     | 6
tcp.sessions                                  | Total                     | 15
tcp.syn                                       | Total                     | 30
tcp.synack                                    | Total                     | 15
tcp.rst                                       | Total                     | 36
tcp.overlap                                   | Total                     | 123
detect.alert                                  | Total                     | 3
app_layer.flow.tls                            | Total                     | 15
app_layer.flow.ntp                            | Total                     | 2
app_layer.tx.ntp                              | Total                     | 4
app_layer.flow.dns_udp                        | Total                     | 900
app_layer.tx.dns_udp                          | Total                     | 2766
app_layer.flow.failed_udp                     | Total                     | 101
flow.mgr.full_hash_pass                       | Total                     | 16
flow.spare                                    | Total                     | 9688
flow.mgr.rows_maxlen                          | Total                     | 2
flow.mgr.flows_checked                        | Total                     | 1846
flow.mgr.flows_notimeout                      | Total                     | 952
flow.mgr.flows_timeout                        | Total                     | 894
flow.mgr.flows_evicted                        | Total                     | 894
flow.mgr.flows_evicted_needs_work             | Total                     | 6
tcp.memuse                                    | Total                     | 2031616
tcp.reassembly_memuse                         | Total                     | 311296
flow.memuse                                   | Total                     | 6474304
```

Figure 19. Log "cat" file

As we can see in my capture, there is no “capture.kernel_drops”, so that means that there is no packet loss. This is because my network is not big enough to deal with this type of problems, but in case this loss appears in the configuration file of Suricata, we can modify the variable “ring-size” in the configuration file “/etc/suricata/suricata.yaml“ by increasing its value.

This tunning has to be done depending on the characteristics of each network. We can find more information regarding packet loss in the Suricata Documentation[16]. The best way to achieve a good configuration on this setup is to perform different tests with different configurations in order to find the optimal one.

[16] https://suricata.readthedocs.io/en/suricata-6.0.1/configuration/index.html

Other useful information and configurations

Despite the fact that the documentation of Suricata is very extensive and well written, there are some other configuration files that we must know in order to establish at least a final configuration for our solution. The four main files that we have to take into account in order to learn a little bit more about this software are:

- **suricata.log:** Suricata startup messages
- **stats.log:** statistics on your network traffic
- **fast.log:** suspicious activities detected by Suricata
- **eve.json:** your local network traffic and suspicious activities in JSON format

As we can see, we can obtain a lot of information from Suricata regarding our network, but the most useful one is fast.log, because in our case the rest of the information can be obtained with Pi-Hole.

So, by tailing some of those log files, we will be able to know what is going on inside our network. Despite this, it is really recommended to change some of the configurations for this logs, specially if our network has a huge amount of traffic, because if we don't apply any modifications to how this logs are regenerated, in the following months we will deal with huge files on they will be very difficult to cheque and process. So, it is recommended to establish a file rotation configuration.

The size of the “eve.json” log file can quickly become very large and take up all the space on the SD card. The use of the “logrotate” mechanism integrated in Linux is very useful to avoid this to happen, specially in long term monitoring. We will configure it to log Suricata logs for 10 days, while limiting the size of each file to 1 GB. So the logs will not be able to take up more than 10 GB on the SD card while having a history over the last 10 days. Older histories are automatically deleted.

To make this possible, we have to create a file “/etc/logrotate.d/suricata” with the following content:

```
/var/log/suricata/*.log /var/log/suricata/*.json

{

   daily

   maxsize 1G

   rotate 10
```

missingok

nocompress

create

sharedscripts

postrotate

systemctl restart suricata.service

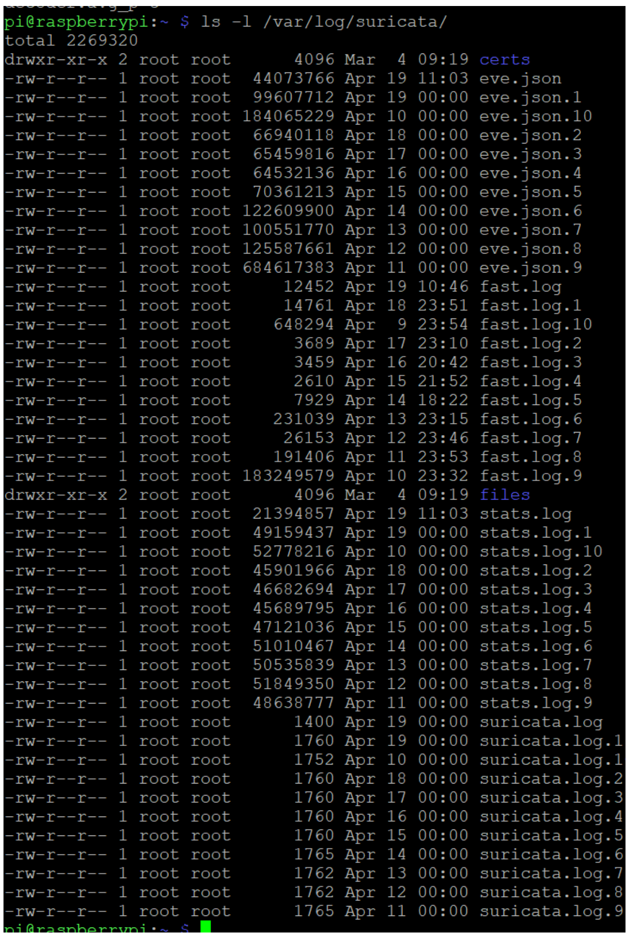
endscript

}

Once we have this configuration stablished, in order to check that the rotation is correctly configured and that it works, you can force it manually with the command:

sudo logrotate -f /etc/logrotate.conf

Then, by applying ls -l /var/log/suricata/ command, we will get the following output:



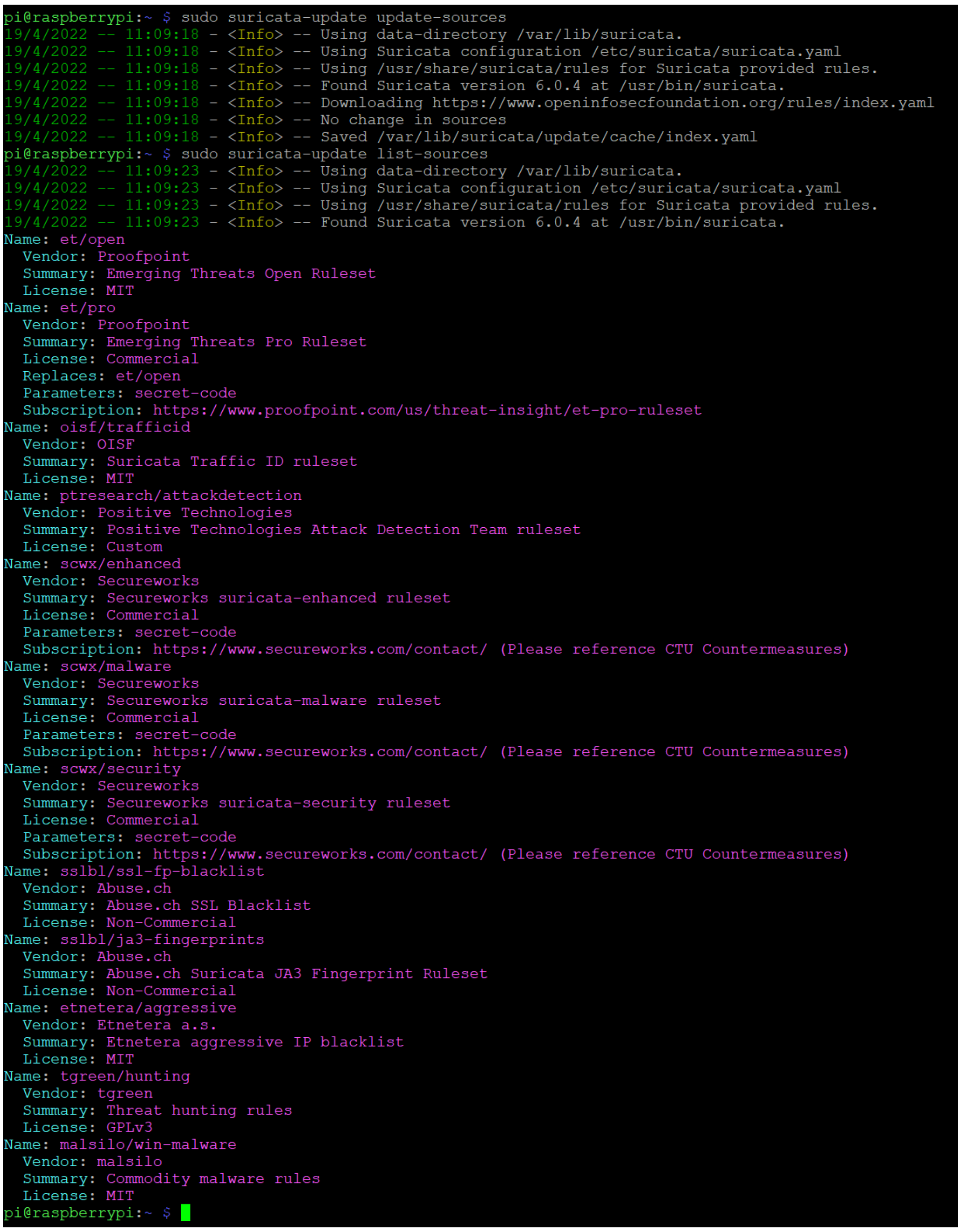
```
pi@raspberrypi:~ $ ls -l /var/log/suricata/
total 2269320
drwxr-xr-x 2 root root      4096 Mar  4 09:19 certs
-rw-r--r-- 1 root root  44073766 Apr 19 11:03 eve.json
-rw-r--r-- 1 root root  99607712 Apr 19 00:00 eve.json.1
-rw-r--r-- 1 root root 184065229 Apr 10 00:00 eve.json.10
-rw-r--r-- 1 root root  66940118 Apr 18 00:00 eve.json.2
-rw-r--r-- 1 root root  65459816 Apr 17 00:00 eve.json.3
-rw-r--r-- 1 root root  64532136 Apr 16 00:00 eve.json.4
-rw-r--r-- 1 root root  70361213 Apr 15 00:00 eve.json.5
-rw-r--r-- 1 root root 122609900 Apr 14 00:00 eve.json.6
-rw-r--r-- 1 root root 100551770 Apr 13 00:00 eve.json.7
-rw-r--r-- 1 root root 125587661 Apr 12 00:00 eve.json.8
-rw-r--r-- 1 root root 684617383 Apr 11 00:00 eve.json.9
-rw-r--r-- 1 root root     12452 Apr 19 10:46 fast.log
-rw-r--r-- 1 root root     14761 Apr 18 23:51 fast.log.1
-rw-r--r-- 1 root root    648294 Apr  9 23:54 fast.log.10
-rw-r--r-- 1 root root      3689 Apr 17 23:10 fast.log.2
-rw-r--r-- 1 root root      3459 Apr 16 20:42 fast.log.3
-rw-r--r-- 1 root root      2610 Apr 15 21:52 fast.log.4
-rw-r--r-- 1 root root      7929 Apr 14 18:22 fast.log.5
-rw-r--r-- 1 root root    231039 Apr 13 23:15 fast.log.6
-rw-r--r-- 1 root root     26153 Apr 12 23:46 fast.log.7
-rw-r--r-- 1 root root    191406 Apr 11 23:53 fast.log.8
-rw-r--r-- 1 root root 183249579 Apr 10 23:32 fast.log.9
drwxr-xr-x 2 root root      4096 Mar  4 09:19 files
-rw-r--r-- 1 root root  21394857 Apr 19 11:03 stats.log
-rw-r--r-- 1 root root  49159437 Apr 19 00:00 stats.log.1
-rw-r--r-- 1 root root  52778216 Apr 10 00:00 stats.log.10
-rw-r--r-- 1 root root  45901966 Apr 18 00:00 stats.log.2
-rw-r--r-- 1 root root  46682694 Apr 17 00:00 stats.log.3
-rw-r--r-- 1 root root  45689795 Apr 16 00:00 stats.log.4
-rw-r--r-- 1 root root  47121036 Apr 15 00:00 stats.log.5
-rw-r--r-- 1 root root  51010467 Apr 14 00:00 stats.log.6
-rw-r--r-- 1 root root  50535839 Apr 13 00:00 stats.log.7
-rw-r--r-- 1 root root  51849350 Apr 12 00:00 stats.log.8
-rw-r--r-- 1 root root  48638777 Apr 11 00:00 stats.log.9
-rw-r--r-- 1 root root      1400 Apr 19 00:00 suricata.log
-rw-r--r-- 1 root root      1760 Apr 19 00:00 suricata.log.1
-rw-r--r-- 1 root root      1752 Apr 10 00:00 suricata.log.1
-rw-r--r-- 1 root root      1760 Apr 18 00:00 suricata.log.2
-rw-r--r-- 1 root root      1760 Apr 17 00:00 suricata.log.3
-rw-r--r-- 1 root root      1760 Apr 16 00:00 suricata.log.4
-rw-r--r-- 1 root root      1760 Apr 15 00:00 suricata.log.5
-rw-r--r-- 1 root root      1765 Apr 14 00:00 suricata.log.6
-rw-r--r-- 1 root root      1762 Apr 13 00:00 suricata.log.7
-rw-r--r-- 1 root root      1762 Apr 12 00:00 suricata.log.8
-rw-r--r-- 1 root root      1765 Apr 11 00:00 suricata.log.9
pi@raspberrypi:~ $
```

Figure 20. Suricata log rotation

As we can see here, as I have been running Suricata for more than one month, my rotation policy has been working properly as we expected, generating 10 log files per each configuration. It could also be important, if those log files are not very big and we have enough space in our SD card, to maintain all the locks in order to make some calculations in long term network monitoring.

Finally, it is important to mention that Suricata has a lot of options in rules and other alert monitoring, so we are able to disable or enable some default rules or even at newer ones depending on our requirements. In my particular case, as this was not the goal of this thesis, I have maintained the default configuration of Suricata regarding the alerts an rules, but a future work could be done in order to modify all of this configuration files to get information that can be more useful than the one that we are getting now.

To be able to maintain all of this intelligent through the time, it is very useful to add in a Crontab task a comment that updates the Suricata software. In my particular case those are the following rules that are enabled:



```
pi@raspberrypi:~ $ sudo suricata-update update-sources
19/4/2022 -- 11:09:18 - <Info> -- Using data-directory /var/lib/suricata.
19/4/2022 -- 11:09:18 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
19/4/2022 -- 11:09:18 - <Info> -- Using /usr/share/suricata/rules for Suricata provided rules.
19/4/2022 -- 11:09:18 - <Info> -- Found Suricata version 6.0.4 at /usr/bin/suricata.
19/4/2022 -- 11:09:18 - <Info> -- Downloading https://www.openinfosecfoundation.org/rules/index.yaml
19/4/2022 -- 11:09:18 - <Info> -- No change in sources
19/4/2022 -- 11:09:18 - <Info> -- Saved /var/lib/suricata/update/cache/index.yaml
pi@raspberrypi:~ $ sudo suricata-update list-sources
19/4/2022 -- 11:09:23 - <Info> -- Using data-directory /var/lib/suricata.
19/4/2022 -- 11:09:23 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
19/4/2022 -- 11:09:23 - <Info> -- Using /usr/share/suricata/rules for Suricata provided rules.
19/4/2022 -- 11:09:23 - <Info> -- Found Suricata version 6.0.4 at /usr/bin/suricata.
Name: et/open
  Vendor: Proofpoint
  Summary: Emerging Threats Open Ruleset
  License: MIT
Name: et/pro
  Vendor: Proofpoint
  Summary: Emerging Threats Pro Ruleset
  License: Commercial
  Replaces: et/open
  Parameters: secret-code
  Subscription: https://www.proofpoint.com/us/threat-insight/et-pro-ruleset
Name: oisf/trafficid
  Vendor: OISF
  Summary: Suricata Traffic ID ruleset
  License: MIT
Name: ptresearch/attackdetection
  Vendor: Positive Technologies
  Summary: Positive Technologies Attack Detection Team ruleset
  License: Custom
Name: scwx/enhanced
  Vendor: Secureworks
  Summary: Secureworks suricata-enhanced ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: scwx/malware
  Vendor: Secureworks
  Summary: Secureworks suricata-malware ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: scwx/security
  Vendor: Secureworks
  Summary: Secureworks suricata-security ruleset
  License: Commercial
  Parameters: secret-code
  Subscription: https://www.secureworks.com/contact/ (Please reference CTU Countermeasures)
Name: sslbl/ssl-fp-blacklist
  Vendor: Abuse.ch
  Summary: Abuse.ch SSL Blacklist
  License: Non-Commercial
Name: sslbl/ja3-fingerprints
  Vendor: Abuse.ch
  Summary: Abuse.ch Suricata JA3 Fingerprint Ruleset
  License: Non-Commercial
Name: etnetera/aggressive
  Vendor: Etnetera a.s.
  Summary: Etnetera aggressive IP blacklist
  License: MIT
Name: tgreen/hunting
  Vendor: tgreen
  Summary: Threat hunting rules
  License: GPLv3
Name: malsilo/win-malware
  Vendor: malsilo
  Summary: Commodity malware rules
  License: MIT
pi@raspberrypi:~ $
```

Figure 21. Suricata updates

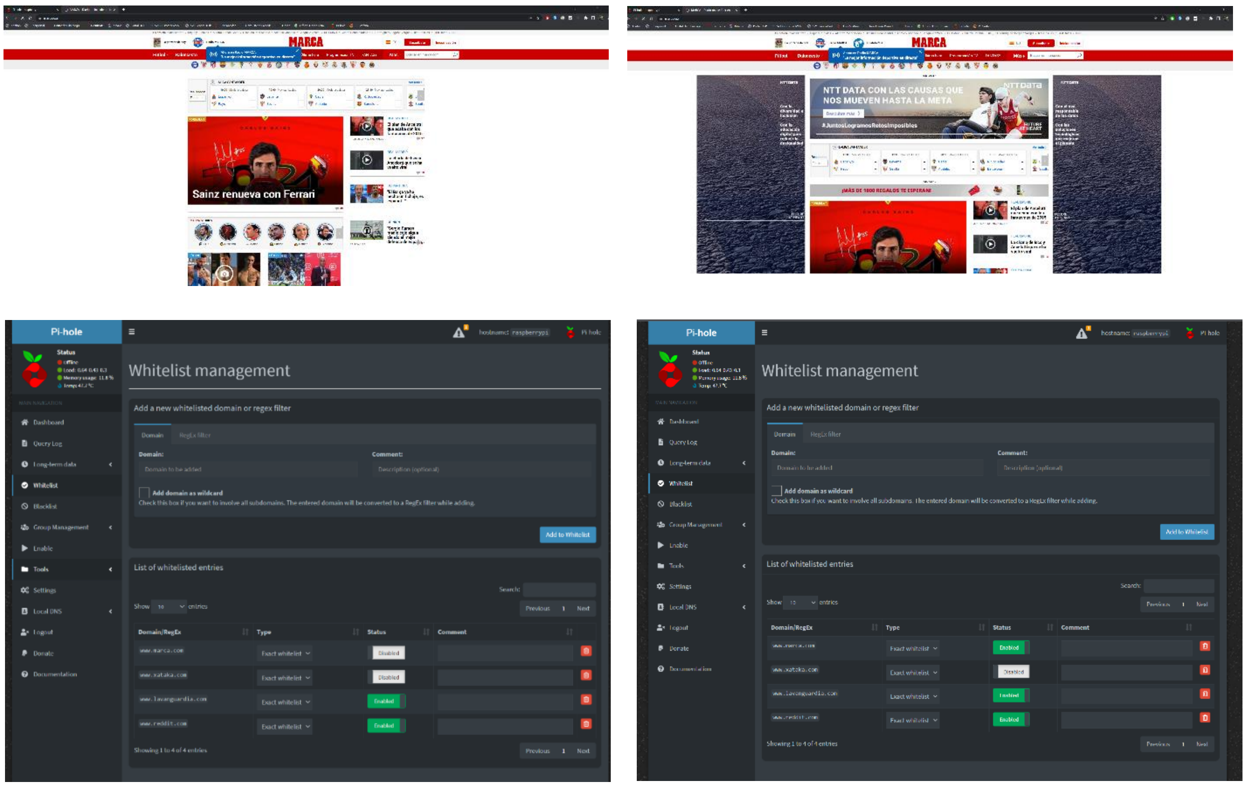
As we can see, all of them are default ones that came with the solution. As I have mentioned, depending on your usage, some rules may need to be disabled. For example, if you allow the use of Dropbox within your network, SID 2012647 rule must be disabled. This rules have to be disabled in the “disable.conf” file.

It is important to remark, as the Suricata documentation of reference says, that this rules are updated in newer versions of the solution, so the more time we have this system deployed, the new rules will appear with new updates in the system.

TESTING

As we have been mentioning during the thesis, the testing has to be done during a long term analysis because sometimes the software does not work as expected, not only Pi-Hole but also Suricata. The best way to test if the system is working as expected is to implement the white list an the blacklist in Pi-Hole, deactivate other advertising tools that we could have by default in our system and make some testing checking if this advertisement is appearing or not in our browser.

This is an example of one of the testing that has been done, but I'm going to present some other ones that can be interesting an has also be done in the system.



Whitelist management

Add a new whitelisted domain or regex filter

Domain: Domain to be added
Comment: Description (optional)
Add domain as wildcard
Check this box if you want to involve all subdomains. The entered domain will be converted to a RegEx filter while adding.

List of whitelisted entries

Domain/RegEx	Type	Status	Comment
www.marca.com	Exact whitelist	Disabled	
www.xataka.com	Exact whitelist	Disabled	
www.lavanguardia.com	Exact whitelist	Enabled	
www.reddit.com	Exact whitelist	Enabled	

Showing 1 to 4 of 4 entries

Domain/RegEx	Type	Status	Comment
www.marca.com	Exact whitelist	Enabled	
www.xataka.com	Exact whitelist	Disabled	
www.lavanguardia.com	Exact whitelist	Enabled	
www.reddit.com	Exact whitelist	Enabled	

Showing 1 to 4 of 4 entries

Figure 22. AD Blocking comparison

As we can see in the previous images, I have deactivated the ad block that I have installed in my default browser and I have looked for a web page that has a lot of advertising. First of all we can see that if we don't enable this web page in our white list we won't have any advertising, as we can see in the first image. All the news appear correctly and there is no other information that can be suspected as advertising despite the fact that all the blocking in the browser is deactivated. This means that our software is working properly and it is filtering this advertising previously on the loading.

Once we configure our Pi-Hole software with the white list activate it, it will not filter anymore those packets from this website. It is easy to see that at this time, as we can see in the second image of the web page, the advertising is appearing along the website despite the fact that we have all the ad block advertising for the browser deactivated. It is mainly for this reason that we can conclude that the filtering of the packets applied previously then the web page loading is working properly.

As I have mentioned before, a lot of other tests have been done in order to control an check that the system is working as expected, and those are the following conclusions that I have extracted.

URL filtering test

One of the most interesting an ambitious test that we can perform in the system is the one that allows us to block certain URLs or web pages that we can define previously in the software. The is there it is quite similar as the previous one that we have presented with the advertising. The main goal is to define our web page (their main name) and test if the software it is blocking as we expect the traffic from this webpage or not.

To perform this test I have downloaded a blacklist from different web pages in the Internet and implement it in the blacklist control part of Pi-Hole. The main idea is to implement a parental control in the network blocking those websites that are not expected to be available, like the ones that have content for adults, video games or other type of stuff.

the main result has been very good, because the list was very long and it has a lot of web pages, but he main problem was that as the software is not devoted to work as a parental control, if you don’t have the webpage listed in this control part, it will not block it, so you must have all the webpages and domains that are suitable to be checked.

This is not scalable, and the best way to approach this solution is to block URL or some content by their categorization. This configuration is not possible, because as I have commented, this is not the goal of Pi-Hole. But it is interesting to conclude that this functionality is very good when we want to define, one by one, the content that has to be blocked in the network.

From my point of view another good approach could have been the possibility to block this content by some time frames. For example, in case that we have someone at home and we don't want to allow them to be distracted when it is working, we can define an schedule at timeline to allow the consumption of videos in YouTube or Netflix, but this content by default it's blocked.

Another good idea for this type of management would have been a timing control for this type of consumption. For instance, and following the previous example, it could have been a good idea to allow some traffic in the network until an specific time. In that case, we will be able to navigate in Youtube for, at least, one hour in a day, and once we have reached this time, the content is blocked.

Group management and client test

Another interesting tool that we can find in the software it's the possibility to identify all the devices that are connected in the network in real time an all of the devices that once in a time have been connected there. This is a really useful tool in order to learn on be aware of which devices are connected and if we have them identified. This can be really useful if we have suspect that someone is navigating in our network and we cannot identify him because it behaviour is really similar to the one that we are having commonly.

In the group management and client configuration part we have a list of all the devices that are connected in the network. The software is able to detect without any problem all those Connexions, but sometimes it only gets the mac direction an not more information such as the host name or the device information that could be useful in order to identify easily the characteristics of this device.

Some of the test that has been done were devoted to analyse if the software was able to identify this host name or not. In my particular case I have a Google pixel 5 smartphone device that it is almost always identified without any doubt in the network. Every time I connect this device to the network I am not only having information about the Mac and the IP that has assigned but also have information about the name “Pixel 5”.

Despite this, there is sometimes that the software is not able to identify this device, so I am only having this mac address. It would have been interesting to have the possibility to insert a tag or a comment depending on the mac address of each device, so once we have detected for the first time this mac (that will not change during the time), we will now know all the devices that are connected by the tag that we have defined. Thanks to this, we won't have anymore the problem off detecting devices that have already been enrolled in the network by its direction but only by its name that we have previously defined.

Taking into account what I have mentioned before and supposing that we have this functionality, it is really useful and easy to see if we have other device that it is not registered previously in the network. We can also process the logs of Suricata, where this mac appears, to make some warnings. But in that case we will be coding this with an script to be able to detect what is going on.

Network test

Despite the fact that all of the previous tests are really important in order to be sure that all the functionalities that we have previously implemented in the system are working as we expect, one of the most important tests is the one devoted to the network performance and activity.

It is clear to see that if we add a packet filtering in a small device as the Rasperry Pi, and this device has to perform a lot of operations to make this blocking properly, we could deal with some problems of stability or user experience when we are navigating through the Internet.

At this point the conclusions are that, although there is no problems when we are navigating through the Internet and we are able to reach and make the same type of operations as before, it is important to mention that the speed of the connectivity, the latency of this connectivity and especially the stability (understanding the stability of the system like the capacity of performing these operations that we are expecting at the very first time and not waiting until this blocking happens in the second or even 10th time) what not as high as the previous one without this device.

A clear example has been extracted when playing some video games that need an stable connectivity with a good ping and latency. I have found that, despite in most of the times this connectivity is working fine and there is no problem when playing with video games, there is sometimes a cut in the connectivity longer than five seconds that does not allow the person that is playing to continue his activity, and this is always very frustrating.

This does not happen, for instance, when we are consuming other type of content, like videos in YouTube, because this type of cutting in the connexion does not have a high impact in the user experience, because this type of service is have a buffer where all the information is stored if we have a problem like this.

But as I have mentioned before the high impact of this instability applies in those activities or those consumption's that need a very good latency and ping. Despite the fact that I have tried to deal with this problems adding some white list in the system and deleting some alerts in Suricata that could apply in the particular case of playing video games, I have not been able to solve these problems and I have also have not been able to identify why this problems were happening.

Although I don't have any information regarding this problem, it seems that it could happen because the Raspberry does not have power enough to manage all the traffic that is coming from the mirroring port an have to be filtered for the software and send it to the Suricata intelligence. By deactivating both solutions, the problem seems to be solved, as we can easily expect if we don't interfere in the connexion from the devices on the Internet.

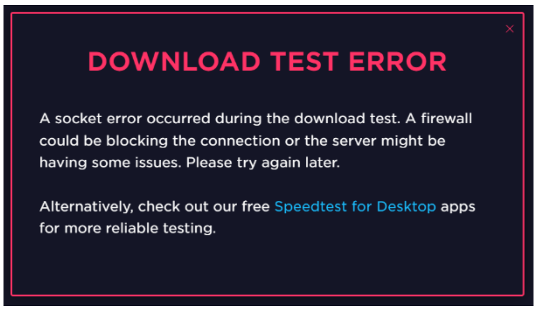
I have also made the test with Suricata enabled and Pi-Hole disabled and vice versa, but in both cases the problem still appears and no solution have been found.

One of the last Test that has been made In order to monitor and understand the impact that this custom firewall could have in the network, is to make an speed test. As as we can see in the following image is the performance of the system is really good if it is working with stability and with any cut.

The maximum speed that I can achieve in a download manner is around 60 Mbps, that is the same that I have if all the device is disconnected an even if the land link it's directly connected to my device. So, as we can see, if the system is working correctly we don't have any problem in terms of ping, download speed and upload speed.

We have to take into account that this metrics have been taken in an specific time-slot, where some other problems could appear that do not involve our Raspberry Pi. Furthermore, as it was introduced in the very beginning of the thesis, there are two PLC devices in the network that route all the information through the electric system of the house. This can cause some interferences in the internet stability that could be contained in the test but are not Firewall fault. So, it is important to remark that this results are quite interesting to analyse but they can have an small distortion from external factors.

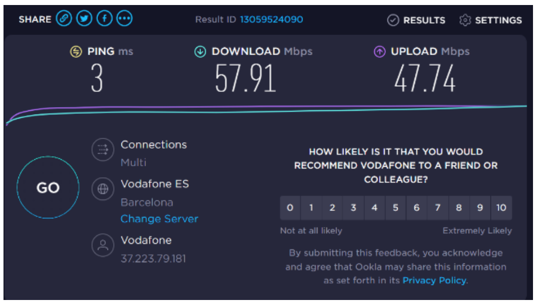
Regarding the network interferences, the problem appears when we have these random cuts in the network that I have previously mentioned. As we can see in the picture the system it is not able to connect through the server to perform the speed test in order to analyse all the data that we are expecting. This is maybe due to a blocking in the custom firewall. I have been not able to identify the cause and the software that it is making this blocking.



DOWNLOAD TEST ERROR

A socket error occurred during the download test. A firewall could be blocking the connection or the server might be having some issues. Please try again later.

Alternatively, check out our free Speedtest for Desktop apps for more reliable testing.



SHARE Result ID 13059524090 RESULTS SETTINGS

PING ms 3 DOWNLOAD Mbps 57.91 UPLOAD Mbps 47.74

GO

Connections Multi

Vodafone ES Barcelona Change Server

Vodafone 37.223.79.181

HOW LIKELY IS IT THAT YOU WOULD RECOMMEND VODAFONE TO A FRIEND OR COLLEAGUE?

0 1 2 3 4 5 6 7 8 9 10

Not at all likely Extremely Likely

By submitting this feedback, you acknowledge and agree that Ookla may share this information as set forth in its Privacy Policy.

Figure 23. Network speed analysis

DASHBOARDS

Another interesting way to understand what it's going on in our network an getting as much as information in an easy way is to cheque the dashboards that we can find in the different solutions that we have implemented in the system.

Despite the fact that Suricata does not have any user interface that can display dashboards, Pi-Hole have a really interesting user interface that gives a lot off information where we can cheque how the system is working an the performance in the real time and with historic. It is one of the best ways to get, in a quick manner, all the information and have all the knowledge of what is happening inside our local area network.

In this chapter we are going to analyse the most important and interesting dashboards and we're going to present another way of checking this information through one of the most common and useful tools of dashboarding called Grafana.

First of all we're going to introduce how this solution could be implemented in our system and how we can gather as much as information as we want in order to display this dashboards in a custom interface. Secondly, we are going to tune all of the data displayed in the dashboards to select the most important one and the one that should be analysed in more detail to get conclusions of the performance of our system, specially when we are talking about firewall and packet filtering.

Grafana implementation

Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources. A licensed Grafana Enterprise version with additional capabilities is also available as a self-hosted installation or an account on the Grafana Labs cloud service. It is expandable through a plug-in system. End users can create complex monitoring dashboards using interactive query builders. Grafana is divided into a front end and back end, written in TypeScript and Go, respectively.

This tool is really interesting because it allows us to gather all the information that we consider important to be checked in our system and put it in a dashboard in a way that we want to cheque this. If we compare it with other dashboards that are integrated in the solutions that we have installed, in this case it is everything custom and developed according to our expectations.

Furthermore, there is a huge community that develops dashboards and connectors to the systems, so there is no need to build from scratch a dashboard and the connectors from the source of the information to the Grafana dashboard web page.

In our particular case for this custom firewall development, despite the fact that we already have the dashboards provided by Pi-Hole, it is really interesting to investigate an try to develop different waves of showing the information because, in the future, this could help us on improving the performance and the way that we extract the information from the system.

As I have said, one of the key points of this solution is the possibility to implement the dashboards that are most interesting for each one an that provides the information that you are requiring, not all the information available or only a part of this information that we have.

from my point of view the two men parts that must be monitorized in the system is the status of the main hardware that it is performing all the actions (that is, all the Raspberry Pi metrics), and on the other hand the way that this custom firewall is performing all the actions that we have defined previously (similar to the ones that are provided in Pi-Hole).

Defining dashboard for Raspberry Pi metrics

In this chapter we are going to see how to monitor all of the metrics of the Raspberry Pi model 4. We will be able to check the temperature, the CPU consumption, all the operations that are being made in real time and with some historic and of course the way that the system is performing an if it has some pics of usage depending on the time.

To be able to implement this type of solution we will be using Telegraph as our main data collector in the source. Later we are going to use influxDB as the data structure and standardisation in order to be able to be read from the graph on a web page.

There are two ways to install Telegraf on Raspberry Pi 4, which we remember is an ARM architecture, the easiest one is using the official InfluxData package repository, and the other is to download the package from the InfluxData website and install it by hand. In our case we will follow the steps for the first option. For the installation, we will need to follow the different commands:

- *curl -sL https://repos.influxdata.com/influxdb.key | sudo apt-key add -*
- *echo "deb https://repos.influxdata.com/debian buster stable" | sudo tee /etc/apt/sources.list.d/influxdb.list*
- *apt-get update && apt-get upgrade*
- *sudo apt install Telegraf -y*
- *sudo systemctl start Telegraf*
- *sudo systemctl enable Telegraf*

Once we have executed all those comments we will need to configure Telegraph to be able to send the information to the InfluxDB server, that will be the entity responsible off gathering this information send it from Telegraph an standard said it to be able to be reat bye Grafana. We will need to modify the file "*/etc/Telegraf/Telegraf.d/rpiservices.conf*" and the "*/etc/Telegraf/Telegraf.d/Raspberry Pitemp.conf*" as the following ones and later on restart the Telegraf service with the command "*sudo systemctl restart Telegraf*":

rpiservices.conf file:

```
# Output Plugin InfluxDB
[[outputs.influxdb]]
  database = "Telegraf"
  urls = [ "http://YOURINFLUXDBIP:8086" ]
  username = "YOURUSER"
  password = "YOURPASS"
  skip_database_creation = true

# Input Plugins
[[inputs.cpu]]
    percpu = true
    totalcpu = true
    collect_cpu_time = false
    report_active = false
[[inputs.disk]]
    ignore_fs = ["tmpfs", "devtmpfs", "devfs"]
[[inputs.io]]
[[inputs.mem]]
[[inputs.net]]
[[inputs.system]]
[[inputs.swap]]
```

```
[[inputs.netstat]]
[[inputs.processes]]
[[inputs.kernel]]
```

Raspberry Pitemp.conf file:

```
[[inputs.file]]
  files = ["/sys/class/thermal/thermal_zone0/temp"]
  name_override = "cpu_temperature"
  data_format = "value"
  data_type = "integer"



[[inputs.exec]]
  commands = ["/opt/vc/bin/vcgencmd measure_temp"]
  name_override = "gpu_temperature"
  data_format = "grok"
  grok_patterns = ["%{NUMBER:value:float}"]
```

Once we have arrived at this point we will be able to gather all the information from the Raspberry (thanks to the Telegraph service), put all of them in there influxDB service and store them in the device. Once here, the only thing that we need to do is understand how the data is stored and perform a dashboard that will get the information from this database and place it in the way that we are expecting.

In case that you want to create the particular dashboard for your system, you will have to create one from scratch from the Grafana menu, but if you are not familiarised with this solution or it is the first time that you are making one of these implementations from scratch the recommendation is to import a dashboard already implemented that gets the information that we are expecting.

In this case we have imported the dashboard number 10578[17] from the official page of Grafana, that collect this information that we are storing in the influxDB. As we can see in the following capture, we are able to read the following metrics:

[17] https://grafana.com/grafana/dashboards/10578

- CPUs (defaults to all)
- Disks (per-disk IOPS)
- Network interfaces (packets, bandwidth, errors/drops)
- Mountpoints (space / inodes)
- Temperature.

There is much more information in the dashboard regarding usage discs and network interfaces but the resume in the first part of the dashboard is the one that provides the most part of the useful information that we can extract from those metrics.

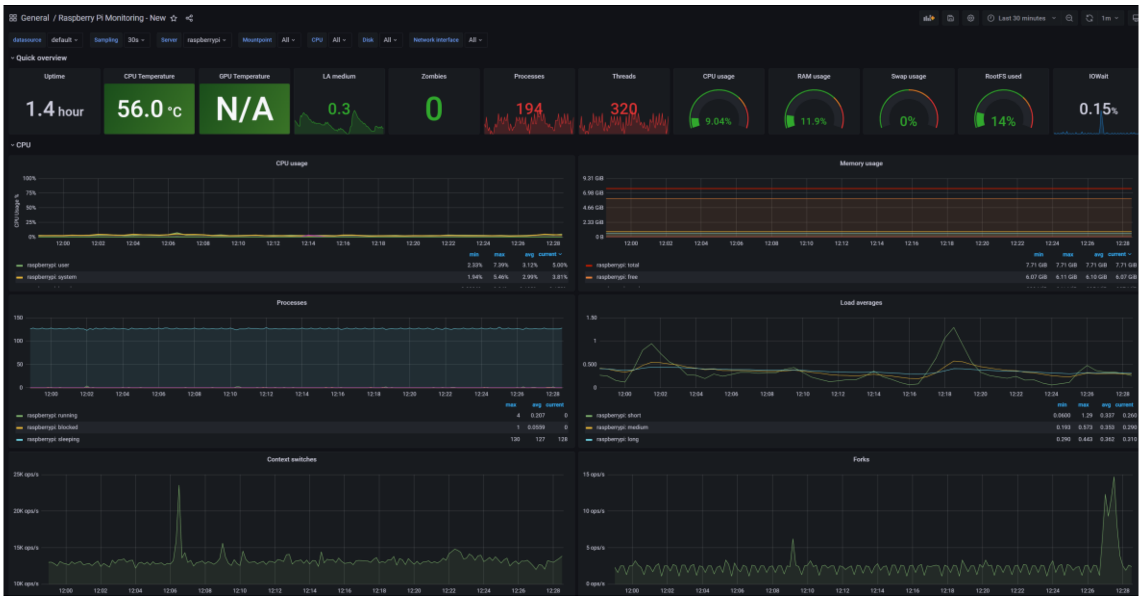


Figure 24. Monitoring custom graphics

In my particular case this dashboard has been very useful for two main reasons. The first one half been the monitoring of the CPU temperature to adjust the speed off the fan that is in the box of the Raspberry. As we can see in the dashboard the temperature is around 56 degrees with the system working perfectly. There is a lot of information regarding the historic off the data, the course and its temperature on workload and a lot of information that it's gathered from Telegraph.

Another important thing that has been very useful when we consider this type of dashboards is checking if in some of the network cuts that I have experimented in the system, as we have explained in the previous chapter with the speed test performed. Knowing the CPU usage is very helpful to see if there is a problem of memory or maybe workload in the system or if the problem is in other place.

As I have previously explained, with some of the network failures in terms of stability and ping, I have not seen any peak of workload in the CPU or GPU, so that means that the system is working as expected without more work that it can support and the problem of this cuts comes from another place. This will have the constrain of each network and the devices that we can find inside. In my particular case, as the network is not congested and the devices are mainly in “sleep” or “idle” mode, the stability of the system is the one expected.

Pi-Hole dashboard

As we have bene talking during all the thesis, one of the main advantages of Pi-Hole is the possibility to check and monitor all the information gathered in the Raspberry Pi and its main OS. Pi-hole uses the well-known relational database management system SQLite3 both for its long-term storage of query data and for its domain management. In contrast to many other database management solutions, Pi-hole does not need a server database engine as the database engine is directly embedded in FTLDNS.

It seems an obvious choice as it is probably the most widely deployed database engine - it is used today by several widespread web browsers, operating systems, and embedded systems (such as mobile phones), among others. Hence, it is rich in supported platforms and offered features. SQLite implements most of the SQL-92 standard for SQL and can be used for high-level queries.

This solution allow us to store a huge amount of data that could be use to check long term metrics as well as specific peaks of traffic blocking, which will provide reliable information about what is going on inside our network.

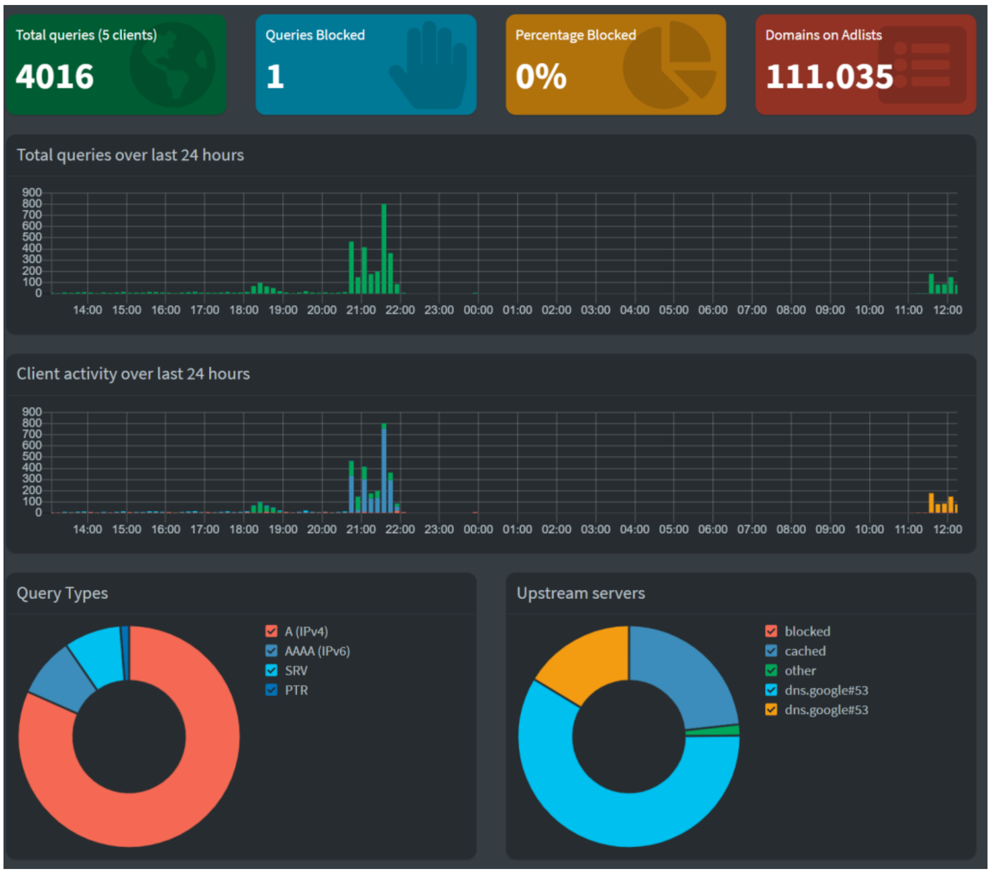
Once the Pi-Hole software is running and we have already deployed the server with the custom credentials, we will be able to check it by resolving the IP configured in the system and login in with the administrator default credentials. Once there, we will be able to set up a summary of information or a long term graphic that will display all the data stored in the databases.

It is important to take into account that the information automatically stored in the SQL database should be consistent in order to be able to read long term observations with sense. Cutting the performance of the device could cause this graphics show strange patterns.

As we can see in the following figure, we have a first overview of most relevant information about what is going on Inside our network and later on we have different groups in different manners that show especially the long term information.

If we focus on the first part we will be able to see the total queries and the total queries blocked in our network in the past one hour. This information is really useful if we want to be able to read and see the real time information. Depending on our configuration about the add blocking, we will see this information very different and with different amount of percentage block or queries blocked. One important thing to cheque in this first part is the amount of clients that are reporting this type of information, because if we configure our switch with port mirroring but we have problems inside, we will be able to detect this type of problems.

For instance, in my particular case, there were one device that was connected in one of the normal ports of the switch so it was not reporting information through the Raspberry. I was able to see this phenomenon because it only appeared for clients that were reporting information and I had five inside my network.


Total queries (5 clients) 4016

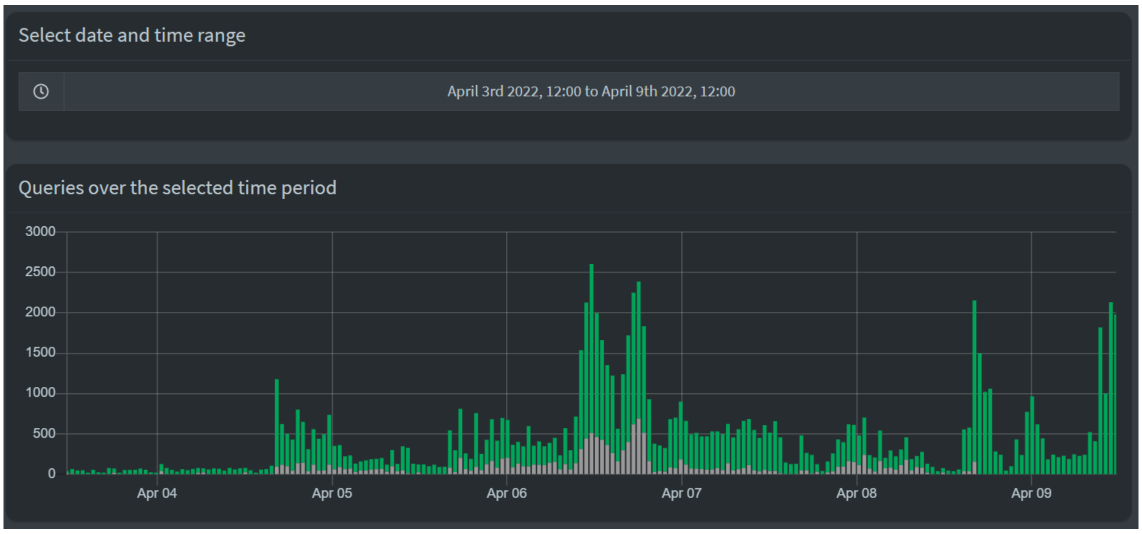
Queries Blocked 1

Percentage Blocked 0%

Domains on Adlists 111.035

Figure 25. Overview of Pi-Hole dashboard

In the next graphs from figure 25 we are able to see the total queries of the past 24 hours (that in my particular case as we can see in the graph during the night this Raspberry what's not on so it was not reporting) and then the same information but classified in the type of traffic. This part of the graphic is quite interesting to cheque in absolute terms, because it will be very easy to see if we are generating much more traffic than normal because we are downloading something or because there are some devices that are having strange behaviour.


Select date and time range

April 3rd 2022, 12:00 to April 9th 2022, 12:00

Queries over the selected time period

Figure 26. Historical query data

As we can see in the previous capture, the figure 26, we also have the capacity to track all the traffic a long time. As we can see in the image the information it's displayed with a temporal axis and queries overtime. This information is similar to the previous one but we have the possibility to cheque this type of data for all the time period that we had the Raspberry working in our network.

This particular chart is really useful if we want to analyse trends or behaviour of our network when we are not using it directly or when eat is performing some particular activities, such as downloading information (like we can see between 6 and 7 of April) or when we have a lot of devices that are being connected inside the network.

In the annex off the thesis we will be able to see more screenshots with different configurations that show more timeline of performance an different types of metrics regarding the ad block off the Raspberry Pi and the queries that are appearing inside.

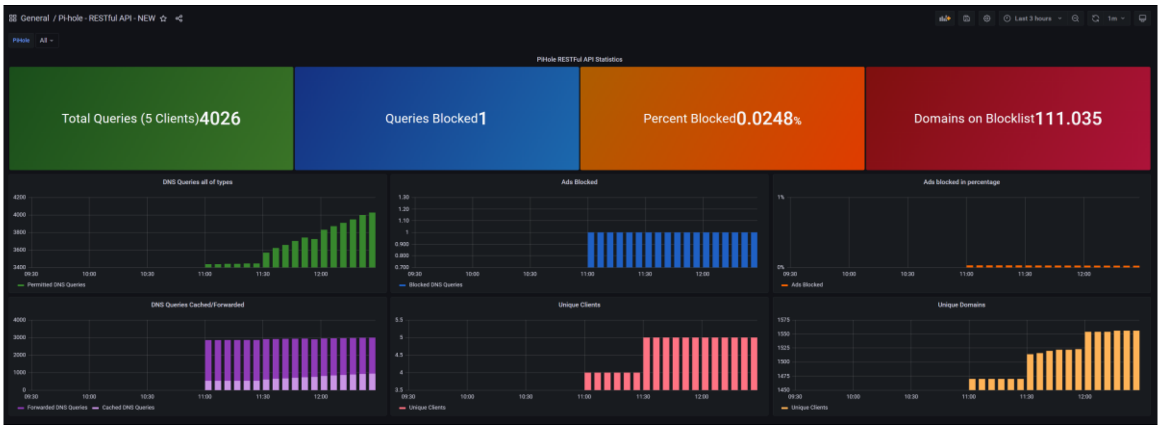

General / Pi-hole - RESTful API - NEW
PiHole RESTFul API Statistics
Total Queries (5 Clients) 4026
Queries Blocked 1
Percent Blocked 0.0248%
Domains on Blocklist 111.035
DNS Queries all of types
Ads Blocked
Ads blocked in percentage
DNS Queries Cached/Forwarded
Unique Clients
Unique Domains

Figure 27. Grafana overview

As we have seen in the previous figures there's a lot of information that can be displayed directly from Pi-Hole software. Despite this, as we have configured Grafana connectors to our systems, it is quite easy to connect the Grafana dashboard to the one that has information from the Pi-Hole.

This will allow us to see the same information that we have already seen in Pi-Hole. But the real advantage that this provides is that Grafana is a complete custom environment so we can modify how the data is displayed or the time range. This will allow us not only to get the information in a dashboard but to get this information in a custom dashboard so we can cheque all the things that we want in our particular way and not depending on the software installed an their limitations.

As we have presented in the previous chapters we will only need to configure Telegraf to display the information that we want. In my particular case I have only linked Pi-Hole with Telegraph and later on to Grafana[18], and I have imported one custom dashboard. But in case that you want to create the custom dashboard by yourself you can do it by the Grafana creator.

Once we have the Telegraph connector working with the quote that will be presented we only need to set the information to InfluxDB, in the same way that we have explained before in the metrics chapter, and then Grafana will be able to read all of this information and post it as a dashboard in the way that we imported (in case that it is a custom dashboard already defined) or in the way that we defined particularly through the dashboard creation of the solution.

Telegraf code needed:

[18] https://grafana.com/grafana/dashboards/13565

```
[[inputs.http]]
#PiHole URL for data in JSON format
urls = ["http://192.168.1.100/admin/api.php"]
method = "GET"

#Overwrite measurement name from default `http` to `pihole_stats`
name_override = "pihole_stats"

#Exclude url and host items from tags
tagexclude = ["host"]

#Data from HTTP in JSON format
data_format = "json"

#JSON values to set as string fields
json_string_fields = ["url", "status"]

insecure_skip_verify = true
```

To check that the previous code is working as expected, is recommended to verify its performance and the process execution with the following commands:

```
Telegraf --config /etc/Telegraf/Telegraf.d/pihole.conf –test

systemctl restart Telegraf
```

CONCLUSIONS

As we have seen during all the thesis, we have explained how can we deploy a custom firewall in our local area network. Furthermore, we have also introduced the main concepts that we have to take into account in order to make this deployment effective an adapted to our needs an our infrastructure.

It is mainly for this reason that the conclusions of all the work done can be explain it in different ways. To so up all the work done and evaluate how we have accomplished all the objectives off this custom firewall, first of all we are going to evaluate the most important thing that we have to consider in this type of deployments, that is if we have been able to block certain traffic of our network through the Internet. Secondly, we are going to evaluate how all the configurations of the deployment have been set up, and finally if we have accomplished the rest of the objectives, specially regarding the future work, of the thesis.

Main objectives

As we have mentioned before, the main objective of the thesis was to deploy a custom firewall that was able to block certain traffic of our network to the Internet. The key point of the deployment was that this traffic was able to be monitor eyes and selected depending on our needs and on our infrastructure.

This specific and general objective has been accomplished without doubts and complications, thanks mainly to the Pi-Hole software and the capacity of the Raspberry Pi to analyse all the traffic that was inside the network. This custom software was general enough to be able to modify the rules of the firewall depending on our needs and the way that we want to perform this blocking from the Internet.

In addition, we have been able to complement this blocking with new lists from the Internet that are updated with high frequency. With this lists we are able to block specific traffic that we have already defined manually in the system end traffic that should be blocked according to the community or the entity that it is providing this list.

Despite the fact that the main objective was accomplished, some other objectives defined were accomplished too. The IDS functionality provided by Suricata was an extra to have a complete network monitoring tool configure it from scratch by us.

At this point Suricata was the perfect complement to hour custom firewall in order to be able to have a real time picture of our network and the traffic that was travelling inside it. Furthermore, the complete integration with Pi-Hole and other solutions like Grafana, allowed us not only to be able to monitor our network but also to be able to do it with a lot of different dashboards and ways to deploy the custom data in the way that we expect.

Secondary objectives

Despite the fact that we already had a tool that gave us the information of all the network and the option to read this information in several ways, there are some other objectives that were proposed at the very beginning of the work without knowing the facility of the integration with the other solutions and the stability in terms of monitoring an memory consumption of the device.

It is mainly for this reason that we can conclude that almost all the first objectives proposed at the very beginning of the thesis as a secondary ones, have been accomplished at least from their base.

The integration between InfluxDB, Telegraf and Grafana was executed without any problem even taking into account that device had an small amount of ram memory and CPU power (those where one of the main concerns). Thanks the disintegration we have been able to monitor, not only the network thanks to the data reported from Pi-Hole and Suricata, but also the main device that is performing all the operations.

Despite this, further work can be done, because all the deployment and the monitoring has been imported from dashboards already developed. At this point, we can still work on deploying more dashboards or modifying the ones that we have already important in order to improve the user experience of the information consumption and making the CPU consumption of the Raspberry less by deleting those services that are not needed in our case.

One of the last conclusions that we can extract regarding the main expectations of the system what's the client and devices management detection of the network. This is a particular conclusion that can only be extracted because of my particular network configuration and the devices that are connected in it. It is important to mention that Pi-Hole software has been able to monitor all the devices that were connected to the network not only at the same time but during several weeks.

This type of performance is not expected in the software, and it is one of the most criticised solutions that it is not working as the users expect. In my particular case this solution has been working perfectly, it has been very useful in terms of device detection with Suricata, and it provided reliable information in all the dashboards, not only the ones displayed in Pi-Hole but also the ones that are in Grafana.

On the other hand, we can obtain the conclusion that the Pi-Hole software in charge of blocking some of the queries or packages that are going to the Internet is not working as expected, especially when there is a huge workload of traffic and more than one device consuming bandwidth.

This is a phenomenon that is already known not only by the creators of the solution but also by all the community that is using it. We have to take into account that this is an open source solution and it may present some strange behaviour if we are performing activities in the network with huge cost of resources.

Improvements in the system

As it has been mentioned during the thesis, there is still some margin for improving, not only the performance of the solution but also how it can be done in order to provide more reliable results.

First, we must take into account that the device selected to make all the monitoring has not the maximum ram allowed. By selecting otherwise with more ram memory or a newer device that could be launched in the future with more CPU capacity, the system will perform in a better way, not only in terms of stability during time but specially with packet loss in Suricata.

Another improvement that could be taken into account is the selection of other professional switch in order to be able to monitor more devices through the Internet portal. Also, if the idea is to monitor as much as network as possible, some modifications can be implemented to reach all the network available, like connecting another router working as an access point directly to the switch in mirroring configuration.

But the things mentioned before are completely hardware. We can also make some improvement in the software part by delegating the different lists that are blocking the traffic and the different rules that Suricata is analysing. For instance, in my particular configuration the rules from Suricata were the one recommended, and if we consider this with the small number of devices of my network, the packet loss was really low (almost all the time was zero). But in case that the network increases, some improvements should be done in this scope by adding or deleting Suricata rules according to the processing capacity of the system.

Finally, in terms of memory consumption a huge work can be done in the dashboard that is devoted to monitor the Raspberry metrics. This dashboard is providing a lot of information to the InfluxDB that later on is traduced to Grafana. Most part of these metrics are imported by default and are not useful to check.

Future work

One of the first point that was consider it at the very beginning of this work and the definition of all the challenges that should be accomplished in this custom firewall development, what's to set the premise that this configuration should be able to be managed and defined in the easiest way possible. This means not only the deployment but also the way in which the information is checked directly to the system or even notified tweet.

It is mainly for this reason that at the beginning one of the main ideas was to establish an notification centre for each user that has this firewall developed. Thanks to this, the user that was in charge of this firewall (in fact, the firewall administrator) would have the opportunity to have alerts in real time that gave him some information regarding di firewall performance and of course if there is some problems in the network.

This notification system has not been developed to time scheduling problems, but it would be a great feature to implement in the future. The main idea is explained in the following chapter.

Telegram Bot notifier

As we have seen during the thesis, there are some models and software installed in our Raspberry Pi that allow us lock all the information in configuration files that are placed somewhere inside the Linux system. For example, Suricata log is a place in each directory, so we can obtain information directly from those files that are ingested in a real time manner.

This will provide the option, for instance, to detect if there have been some strange traffic inside the network and from which device this traffic has been detected. So, by scrapping these files that we already know their directory and structure, we will be able to detect not only strange behaviour inside our network, but also if there are some stats of the Raspberry Pi that are not working as expected.

This can be easily implemented with a Cron task[19] and a Python script. By developing an easy script which main function is to parse and read all the information from the locks that we want, we will be able to know in a real time (or in the time and defined in the Cron task), all this stuff that we want to notify through this telegram bot[20].

Finally, by opening a telegram client and bought through the application, all the information pass it from this script, will be easily sent through this bought to our phone or do a channel that we can define if we want to publish this information for more people.

Despite the fact that this is quite an easy structure because telegram provides a lot of information and a lot of scripts that already do this task, the challenge at this point is to be able to understand and analyse all the different types of logs that are set by Pi-Hole and Suricata, be able to parse and finally understand which ones we request.

Further integrations

Another interesting development that could be deployed in the system is more software related to IDS and monitoring functionalities. In this thesis we have selected Suricata as our main intelligence in the system, but there are such a lot of other tools that could be even better depending on our needs and the way that we want to monitor our network.

It could be an interesting work to do to try to develop, if the system is capable in terms of CPU and ram memory, both systems and compare their performance in order to conclude which one east working better in this scenario. Furthermore, the possibility to implement other type of monitoring software could be very interesting in order to inject new information in Grafana dashboard's, because there are some functionalities that are not deployed in Suricata that we can find in other solutions.

A part off this, it could be a great idea to deploy other services inside the Raspberry that are not directly related with firewalls but could be interesting to have in our network under

[19] https://www.redeszone.net/tutoriales/servidores/cron-crontab-linux-programar-tareas/
[20] https://core.telegram.org/bots

these security controls. One example of services could be a private VPN to be able to access directly to our network and monitor directly the devices that we have inside it. With this we can also add other type of stuff like a private cloud or NAS to assert the information that we want to have, so we will be able to access this information remotely thanks to the VPN and everything under a safety environment.

To sum up, there are a lot of alternatives and options that we can implement in a Linux system like the one that we have used to implement a custom firewall. As we have been using open source tools really easy to deploy and configure, there are a lot of options to merge the best part of each solution in order to obtain all the stuff that we require. It is for this reason that the further work may not end never, but this is one of the best things of this type of developments.

GLOSSARY

Blocklist: A list of entities that are blocked or denied privileges or access.

Allow list: A list of entities that are considered trustworthy and are granted access or privileges.

Antispyware software: A program that specializes in detecting and blocking or removing forms of spyware.

Asset: A person, structure, facility, information, and records, information technology systems and resources, material, process, relationships, or reputation that has value.

Access control list (ACL): A list of individual users and groups of users associated with an object, and the rights that the user or group has when accessing that object.

Attack: An attempt to gain unauthorized access to system services, resources, or information, or an attempt to compromise system integrity.

Attack surface: The set of ways in which an adversary can enter a system and potentially cause damage.

Authentication: The process of verifying the identity or other attributes of an entity (user, process, or device).

Availability: The property of being accessible and usable upon demand.

Botnet: A collection of computers compromised by malicious code and controlled across a network.

Confidentiality: A property that information is not disclosed to users, processes, or devices unless they have been authorized to access the information.

Cryptography: The use of mathematical techniques to provide security services, such as confidentiality, data integrity, entity authentication, and data origin authentication.

Cybersecurity: The activity or process, ability or capability, or state whereby information and communications systems and the information contained therein are protected from and/or defended against damage, unauthorized use or modification, or exploitation.

Data loss prevention: A set of procedures and mechanisms to stop sensitive data from leaving a security boundary.

Demilitarized zone (DMZ): Networks that are between a company's internal network and the external network. A DMZ is used as an additional buffer to further separate the public network from your internal private network.

Decryption: The process of transforming ciphertext into its original plaintext.

Event: An observable occurrence in an information system or network.

Exploit: A technique to breach the security of a network or information system in violation of security policy.

Firewall: A capability to limit network traffic between networks and/or information systems.

Hacker: An unauthorized user who attempts to or gains access to an information system.

Incident: An occurrence that actually or potentially results in adverse consequences to (adverse effects on) (poses a threat to) an information system or the information that the system processes, stores, or transmits and that may require a response action to mitigate the consequences.

Indicator: An occurrence or sign that an incident may have occurred or may be in progress.

Integrity: The property whereby information, an information system, or a component of a system has not been modified or destroyed in an unauthorized manner.

Intrusion detection: Intrusion detection is a relatively new technology used with firewalls. It allows firewalls to perform specified actions when suspicious activity occurs.

IP: An Internet protocol or IP address is a number that is used to uniquely identify computers connected to the Internet.

Malicious code: Program code intended to perform an unauthorized function or process that will have adverse impact on the confidentiality, integrity, or availability of an information system.

Network services: In the NICE Framework, cybersecurity work where a person: Installs, configures, tests, operates, maintains, and manages networks and their firewalls, including hardware (e.g., hubs, bridges, switches, multiplexers, routers, cables, proxy servers, and protective distributor systems) and software that permit the sharing and transmission of all spectrum transmissions of information to support the security of information and information systems.

Packet: In general usage, a packet is a unit of information transmitted as a whole from one device to another on a network. In packet-switching networks, a packet is defined more specifically as a transmission unit of fixed maximum size that consists of binary digits representing data, a header containing an identification number, source, and destination addresses, and sometimes error-control data.

Packet filter: A type of firewall devices that process network traffic on a packet-by-packet basis. Packet filter devices allow or block packets, and are typically implemented through standard routers.

Packet sniffer: A device or program that is used to monitor traffic on a network, can be installed anywhere in a networked system, and is virtually undetectable. Sniffers are used for legitimate network management functions or for stealing information off a network.

Penetration testing: An evaluation methodology whereby assessors search for vulnerabilities and attempt to circumvent the security features of a network and/or information system.

Plaintext: Unencrypted information.

Response: The activities that address the short-term, direct effects of an incident and may also support short-term recovery.

Risk analysis: The systematic examination of the components and characteristics of risk.

Threat: A circumstance or event that has or indicates the potential to exploit vulnerabilities and to adversely impact (create adverse consequences for) organizational operations, organizational assets (including information and information systems), individuals, other organizations, or society.

Virus: A computer program that can replicate itself, infect a computer without permission or knowledge of the user, and then spread or propagate to another computer.

Vulnerability: A characteristic or specific weakness that renders an organization or asset (such as information or an information system) open to exploitation by a given threat or susceptible to a given hazard.

BIBLIOGRAPHY

- Ref 1. Sonic Wall cyber thread report
  https://www.sonicwall.com/2022-cyber-threat-report/

- Ref 2. Statistics of ransomware and cyberattacks
  https://spanning.com/blog/cyberattacks-2021-phishing-ransomware-data-breach-statistics/

- Ref 3. Leak from IBM of a security data breach
  https://www.ibm.com/security/data-breach

- Ref 4. Data breach extracted from antivirus committee
  https://www.idtheftcenter.org/post/identity-theft-resource-center-to-share-latest-data-breach-analysis-with-u-s-senate-commerce-committee-number-of-data-breaches-in-2021-surpasses-all-of-2020/

- Ref 5. Reference information for Raspberry Pi Model 4
  https://www.Raspberry Pi.com/products/raspberry-pi-4-model-b/

- Ref 6. Simple wall software reference
  https://www.simplewallsoftware.com/installation-guide

- Ref 7. Blokada software reference
  https://www.slant.co/versus/24914/32130/~pi-hole_vs_blokada

- Ref 8. Adguard software reference and description
  https://adguard.com/en/welcome.html

- Ref 9. OpenWRT software reference
  https://openwrt.org/

- Ref 10. Bro network security monitor reference
  https://www.bro.org/

- Ref 11. Different operating systems that can be deployed in Raspberry Pi
  https://www.Raspberry Pi.com/software/operating-systems/

- Ref 12. Firebog list for Pi-Hole
  https://firebog.net/

- Ref 13. Lightswitch list for Pi-Hole
  https://github.com/lightswitch05/hosts

- Ref 14. OISD Domain blocklist for Raspberry Pi
  https://oisd.nl/?p=dl

- Ref 15. Regex list for Raspberry Pi
  https://github.com/mmotti/pihole-regex/blob/master/regex.list

- Ref 16. Suricata configuration proposal
  https://suricata.readthedocs.io/en/suricata-6.0.1/configuration/index.html

- Ref 17. Grafana dashboard proposal for Pi-Hole
  https://Grafana.com/Grafana/dashboards/10578

- Ref 18. Grafana dashboard proposal for monitoring
  https://Grafana.com/Grafana/dashboards/13565

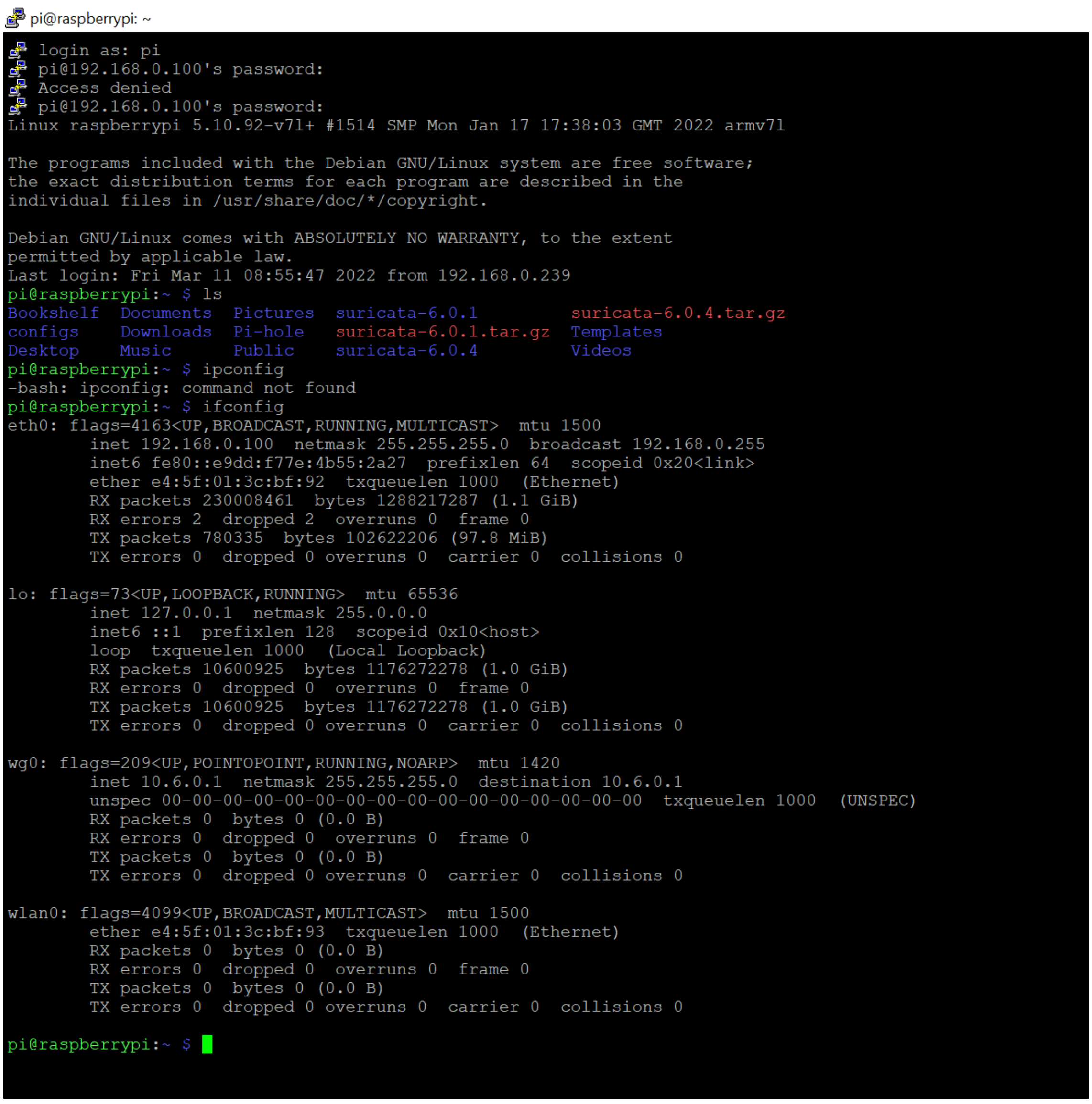
- Ref 19. Crontab reference for Linux operating systems
  https://www.redeszone.net/tutoriales/servidores/cron-crontab-linux-programar-tareas/

- Ref 20. Telegram bot development
  https://core.telegram.org/bots

ANNEX

In this part of the thesis we are going to present similar configurations of screenshots taken directly by the terminal of the Raspberry or even directly from the Grafana or Pi-Hole stats. The main goal of the annex is to show different types of configurations and a clear picture of how the system is implemented with their own definitions.

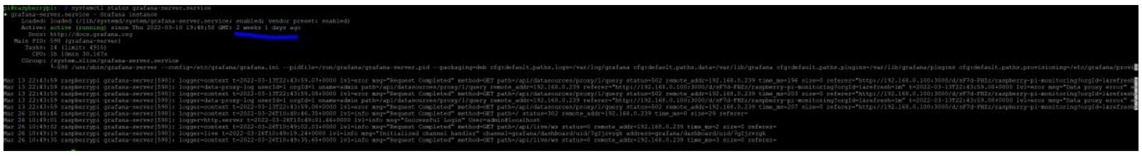
In the first part of the annex you will be able to see how is the LAN configuration of the Raspberry done an later on I am going to present other captures of the dashboard that provides more information in the long term size.



pi@raspberrypi: ~

```
login as: pi
pi@192.168.0.100's password:
Access denied
pi@192.168.0.100's password:
Linux raspberrypi 5.10.92-v71+ #1514 SMP Mon Jan 17 17:38:03 GMT 2022 armv7l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Fri Mar 11 08:55:47 2022 from 192.168.0.239
pi@raspberrypi:~ $ ls
Bookshelf  Documents  Pictures  suricata-6.0.1         suricata-6.0.4.tar.gz
configs    Downloads  Pi-hole   suricata-6.0.1.tar.gz  Templates
Desktop    Music      Public    suricata-6.0.4         Videos
pi@raspberrypi:~ $ ipconfig
-bash: ipconfig: command not found
pi@raspberrypi:~ $ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.100  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::e9dd:f77e:4b55:2a27  prefixlen 64  scopeid 0x20<link>
        ether e4:5f:01:3c:bf:92  txqueuelen 1000  (Ethernet)
        RX packets 230008461  bytes 1288217287 (1.1 GiB)
        RX errors 2  dropped 2  overruns 0  frame 0
        TX packets 780335  bytes 102622206 (97.8 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 10600925  bytes 1176272278 (1.0 GiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 10600925  bytes 1176272278 (1.0 GiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wg0: flags=209<UP,POINTOPOINT,RUNNING,NOARP>  mtu 1420
        inet 10.6.0.1  netmask 255.255.255.0  destination 10.6.0.1
        unspec 00-00-00-00-00-00-00-00-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

wlan0: flags=4099<UP,BROADCAST,MULTICAST>  mtu 1500
        ether e4:5f:01:3c:bf:93  txqueuelen 1000  (Ethernet)
        RX packets 0  bytes 0 (0.0 B)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

pi@raspberrypi:~ $
```

Figure 28. Raspberry Pi LAN configuration



```
pi@raspberrypi:~ $ systemctl status grafana-server.service
● grafana-server.service - Grafana instance
     Loaded: loaded (/lib/systemd/system/grafana-server.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2022-03-10 19:48:58 GMT; 2 weeks 1 days ago
       Docs: http://docs.grafana.org
   Main PID: 598 (grafana-server)
      Tasks: 14 (limit: 4915)
        CPU: 1h 10min 30.167s
     CGroup: /system.slice/grafana-server.service
             └─598 /usr/sbin/grafana-server --config=/etc/grafana/grafana.ini --pidfile=/run/grafana/grafana-server.pid --packaging=deb cfg:default.paths.logs=/var/log/grafana cfg:default.paths.data=/var/lib/grafana cfg:default.paths.plugins=/var/lib/grafana/plugins cfg:default.paths.provisioning=/etc/grafana/provi

Mar 13 22:43:59 raspberrypi grafana-server[598]: logger=context t=2022-03-13T22:43:59.07+0000 lvl=eror msg="Request Completed" method=GET path=/api/datasources/proxy/1/query status=502 remote_addr=192.168.0.239 time_ms=196 size=0 referer="http://192.168.0.100:3000/d/sF7d-FHZz/raspberry-pi-monitoring?orgId=1&refresh
Mar 13 22:43:59 raspberrypi grafana-server[598]: logger=data-proxy-log userId=1 orgId=1 uname=admin path=/api/datasources/proxy/1/query remote_addr=192.168.0.239 referer="http://192.168.0.100:3000/d/sF7d-FHZz/raspberry-pi-monitoring?orgId=1&refresh=1m" t=2022-03-13T22:43:59.08+0000 lvl=eror msg="Data proxy error" e
Mar 13 22:43:59 raspberrypi grafana-server[598]: logger=context t=2022-03-13T22:43:59.08+0000 lvl=eror msg="Request Completed" method=GET path=/api/datasources/proxy/1/query status=502 remote_addr=192.168.0.239 time_ms=203 size=0 referer="http://192.168.0.100:3000/d/sF7d-FHZz/raspberry-pi-monitoring?orgId=1&refresh
Mar 13 22:43:59 raspberrypi grafana-server[598]: logger=data-proxy-log userId=1 orgId=1 uname=admin path=/api/datasources/proxy/1/query remote_addr=192.168.0.239 referer="http://192.168.0.100:3000/d/sF7d-FHZz/raspberry-pi-monitoring?orgId=1&refresh=1m" t=2022-03-13T22:43:59.08+0000 lvl=eror msg="Data proxy error" e
Mar 13 22:43:59 raspberrypi grafana-server[598]: logger=context t=2022-03-13T22:43:59.08+0000 lvl=eror msg="Request Completed" method=GET path=/api/datasources/proxy/1/query status=502 remote_addr=192.168.0.239 time_ms=207 size=0 referer="http://192.168.0.100:3000/d/sF7d-FHZz/raspberry-pi-monitoring?orgId=1&refresh
Mar 26 10:48:46 raspberrypi grafana-server[598]: logger=context t=2022-03-26T10:48:46.35+0000 lvl=info msg="Request Completed" method=GET path=/ status=302 remote_addr=192.168.0.239 time_ms=0 size=29 referer=
Mar 26 10:49:01 raspberrypi grafana-server[598]: logger=http.server t=2022-03-26T10:49:01.44+0000 lvl=info msg="Successful Login" User=admin@localhost
Mar 26 10:49:02 raspberrypi grafana-server[598]: logger=context t=2022-03-26T10:49:02.03+0000 lvl=info msg="Request Completed" method=GET path=/api/live/ws status=0 remote_addr=192.168.0.239 time_ms=2 size=0 referer=
Mar 26 10:49:19 raspberrypi grafana-server[598]: logger=live t=2022-03-26T10:49:19.24+0000 lvl=info msg="Initialized channel handler" channel=grafana/dashboard/uid/7gIjrvzgk address=grafana/dashboard/uid/7gIjrvzgk
Mar 26 10:49:35 raspberrypi grafana-server[598]: logger=context t=2022-03-26T10:49:35.65+0000 lvl=info msg="Request Completed" method=GET path=/api/live/ws status=0 remote_addr=192.168.0.239 time_ms=3 size=0 referer=
```

Figure 29. Grafana service running check

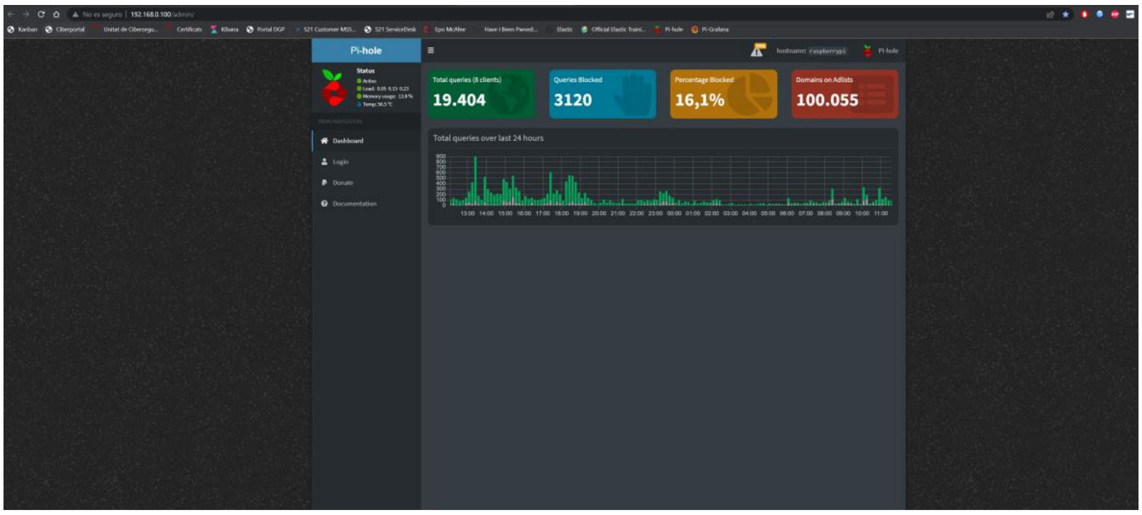
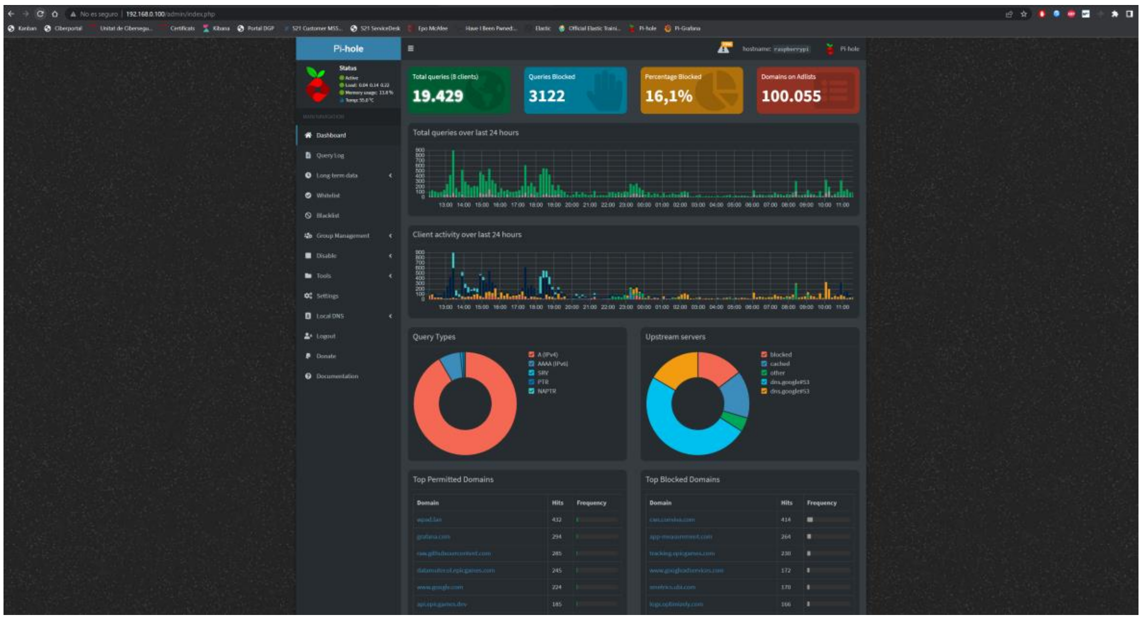


Figure 30. Pi-Hole queries activity



Top Permitted Domains

Domain	Hits	Frequency
wpad.lan	432	
grafana.com	294	
raw.githubusercontent.com	285	
datarouter.ol.epicgames.com	245	
www.google.com	224	
api.epicgames.dev	185	

Top Blocked Domains

Domain	Hits	Frequency
cws.conviva.com	414	
app-measurement.com	264	
tracking.epicgames.com	230	
www.googleadservices.com	172	
metrics.icloud.com	170	
logx.optimizely.com	166	

Figure 31. Pi-Hole general overview

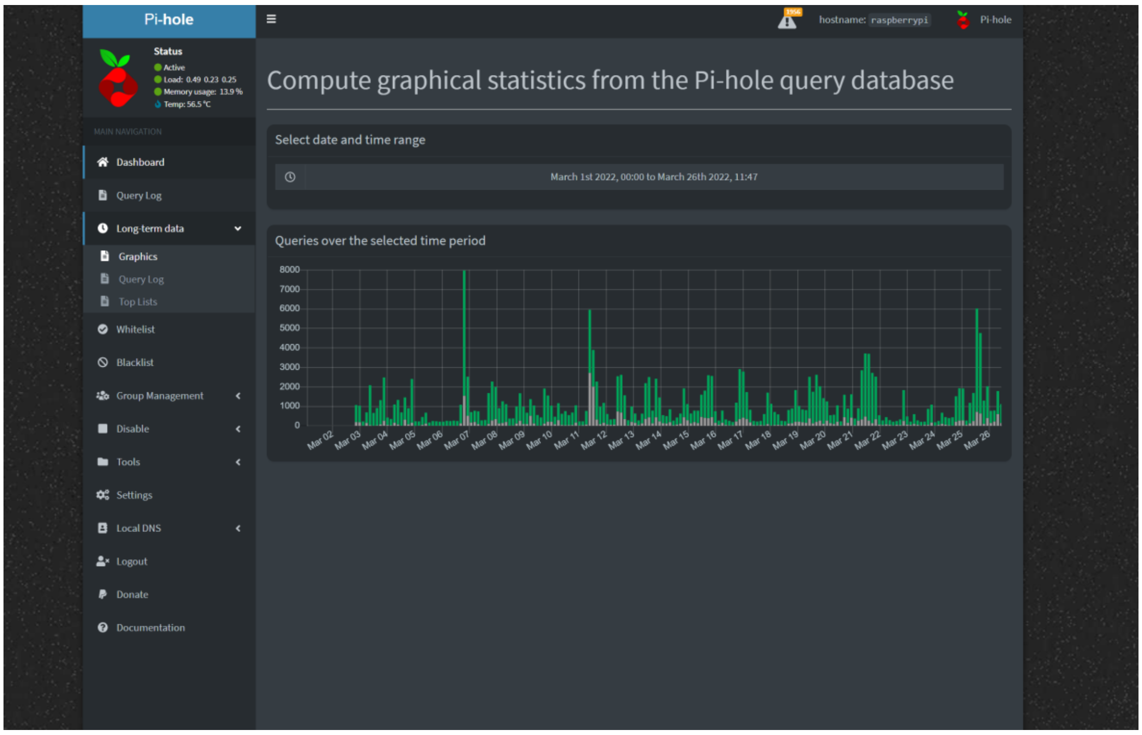


Figure 32. Pi-Hole one month traffic

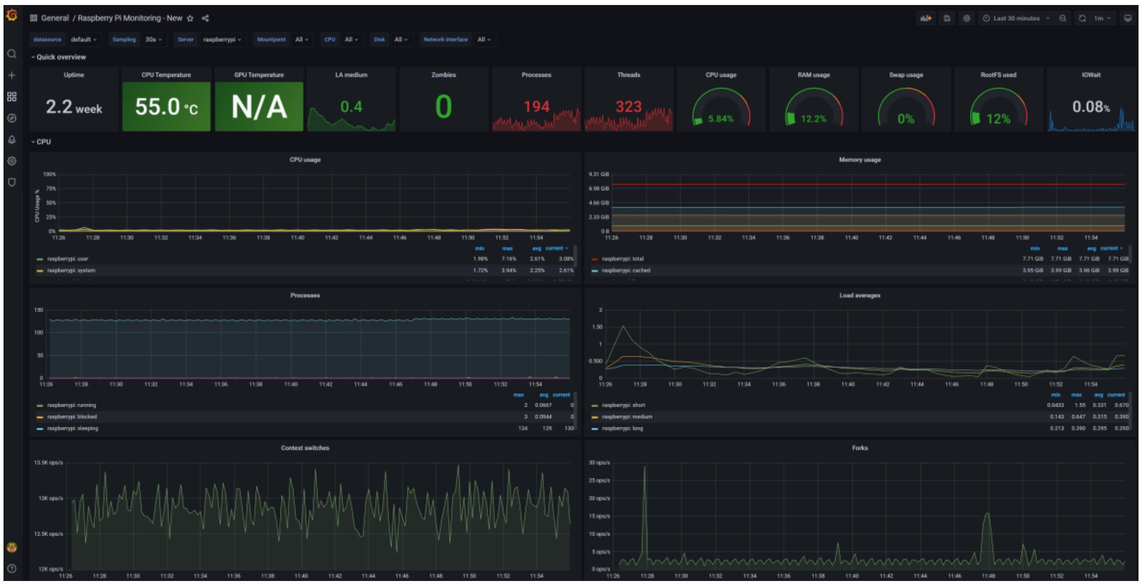


Figure 33. Pi-Hole Grafana metrics (real-time)

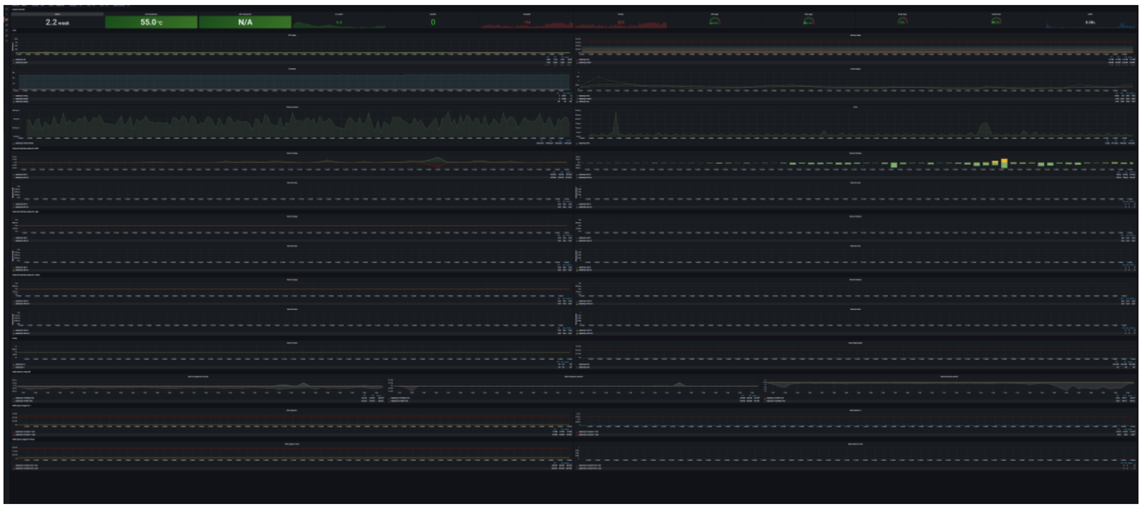
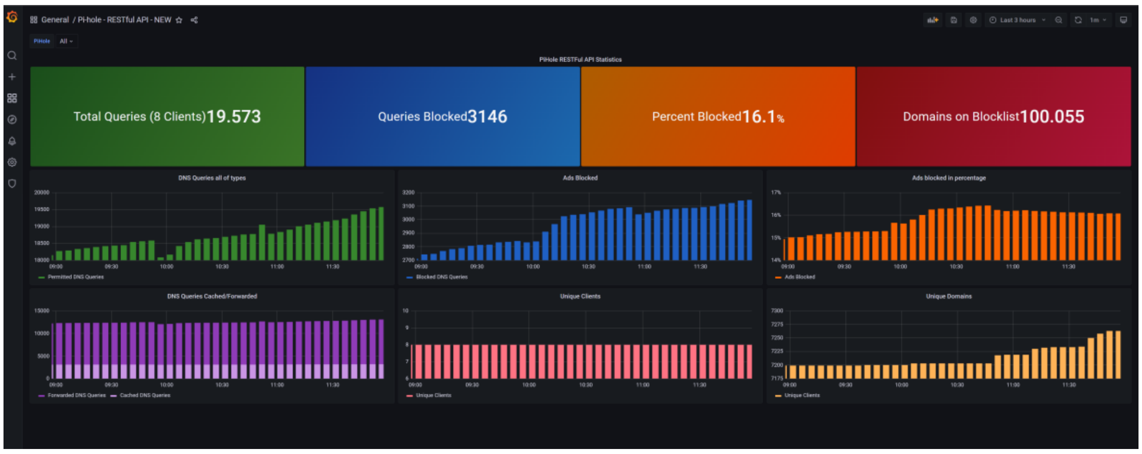


Figure 34. Grafana metrics overview


General / Pi-hole - RESTful API - NEW
PiHole RESTFul API Statistics
Total Queries (8 Clients) 19.573
Queries Blocked 3146
Percent Blocked 16.1%
Domains on Blocklist 100.055
DNS Queries all of types
Ads Blocked
Ads blocked in percentage
DNS Queries Cached/Forwarded
Unique Clients
Unique Domains
Figure 35. Pi-Hole metrics in Grafana