ASHESI UNIVERSITY

BUILDING A CHEAP SECURITY SOLUTION FOR A SMART HOME NETWORK USING A RASPBERRY PI AND SNORT 3

UNDERGRADUATE THESIS

B.Sc. Computer Science

Sharon Adelaide Asomani-Wiafe

2020

ASHESI UNIVERSITY

Supervisor: Dr Stephane Nwolley

BUILDING A CHEAP SECURITY SOLUTION FOR A SMART HOME NETWORK USING A RASPBERRY PI AND SNORT 3

UNDERGRADUATE THESIS

Undergraduate thesis submitted to the Department of Computer Science, Ashesi University in partial fulfilment of the requirements for the award of

Bachelor of Science Degree in Computer Science

Sharon Adelaide Asomani-Wiafe

2020

Declaration

I hereby declare that this undergraduate thesis is the result of my own original work and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature:

……

Candidate's Name:

……
Date:

……

I hereby declare that preparation and presentation of this undergraduate thesis were supervised in accordance with the guidelines on supervision of undergraduate thesis laid down by Ashesi University.

Supervisor's Signature:

……

Supervisor's Name:

……
Date:

……

Acknowledgements

I would first like to thank my supervisor, whose encouragement, support and academic advice helped me undertake this project. I am grateful to Claude Noel Tamakloe for his help with brainstorming to arrive at this topic. I would like to express my most profound appreciation to all my friends for their support and motivation when things got difficult, especially Yaw Botwe and Nutifafa Cudjoe. Finally, I am thankful to Ebo Adjepon-Yamoah for always being ready to listen and give constructive advice.

Abstract

Digitisation is moving at breakneck speed, and soon almost all devices will be interconnected via a network. The goal of the Internet of Things (IoT) is to extend internet connectivity to such devices. In a smart home, examples of such devices are a toaster or a refrigerator that could be linked to the Internet and accessed remotely. This predicted future promises to improve the standard of living; however, it brings with it a new set of security challenges, such as a denial of service attack and ARP spoofing, among others. This paper therefore, seeks to discover if using Snort 3, a popular intrusion detection system deployed on a Raspberry Pi would be able to protect these devices.

Table of Contents

List of Tables

List of Figures

Chapter 1: Introduction

1.1 Introduction

The term, Internet of Things (IoT), has become very popular and widely used recently to signify the connection of everyday traditional devices and systems such as ventilation systems, refrigerators, factory equipment and medical devices to a network such as the Internet. There are numerous IoT devices in airports, filling stations, offices and warehouses that perform specific tasks to improve efficiency and generally improve the quality of life. For instance, for IoT systems centred around people, specifically elderly and disabled people, these systems help to increase their autonomy [1]. IoT solutions centred around safety seek to reduce the number of precarious situations people may face. For example, interconnected vehicles on a highway would be able to communicate with each other, thus helping the driver to avoid driving into other cars; and autonomous mining equipment helps to reduce the percentage of human life lost to mining incidents [2]. With the above-stated examples, there is evidence that connecting these devices would improve overall efficiency and increase the quality of life. Furthermore, in the future, it would be possible to predict the amount of alcohol party goers would consume and deploy more taxis to such locations to reduce alcohol-related road accidents.

1.2 Brief History

The first proof of the Internet of Things concept can be referenced to a project undertaken by the Carnegie Melon University in 1982 where the team improved a Coke dispensing machine. It became the first machine connected to the Internet with it being capable of reporting its inventory. It also reported on the temperature of the drinks in the device [3]. However, the term Internet of Things (IoT) was popularised by Kevin Ashton in 1999, during a presentation highlighting the value of Radio Frequency Identification (RFID) technology in the supply chain industry [4].

1.3 Future of IoT

Since then, the number of connected devices is exponentially increasing, and the term being used to describe the environment where such devices can be found is a smart environment. The market for the emergence of such environments is so lucrative that the market of smart homes was expected to peak at $26 billion by the end of 2019 [5]. It has likewise been predicted that by 2020 the number of IT devices in circulation would surpass fifty (50) billion [6].

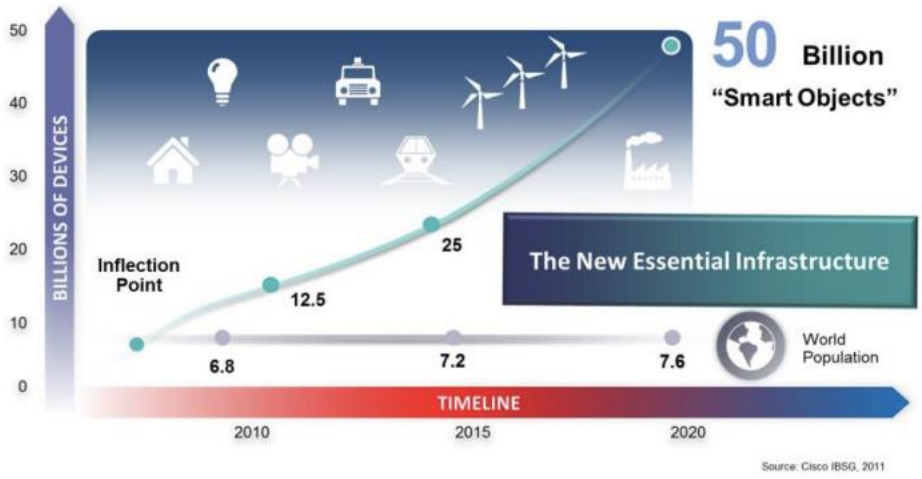


Figure 1.1: The predicted number of smart objects (taken from [6])

These smart environments provide a plethora of opportunities to individuals and businesses alike. The connection of these devices to the Internet allows for remote access to the sensors and devices which make up the IoT systems, thus efficiently collecting data from these devices and sensors for monitoring, forecasting and improving the intelligence of the devices themselves. The connection of these devices to the Internet allows for enhanced interaction among people and the smart environment; and provides a cost-effective method for saving energy [38]. As mentioned, there is a myriad of smart environments, but the focus of this thesis is on the security of the most private of all smart environments: the smart home.

1.4 Security of IoT Networks

Many IoT devices and systems produced are geared towards consumers, and a study conducted in the U.S. estimates two-thirds of U.S. households are expected to own some form of smart device by the end of 2019 [8]. As a result of the wide range of business opportunities provided by such devices, manufacturers are continually churning out new IoT devices. However, security mechanisms have not been produced at the same pace as the IoT systems and devices themselves, causing a massive gap in the security and protection of these IoT devices and systems [7]. For instance, one point of vulnerability due to the protection gap was discovered in 2016. This vulnerability of the system was exploited by a malware called *Mirai* [9]. This malware particularly hijacks poorly secured closed-circuit devices like digital video cameras and routers [10]. According to Angrishi, in just under two weeks, Mirai gained control of about 200,000 devices, which were later used as botnets to mount large-scale Distributed Denial of Service attacks (DDoS) [9]. This example is just one of the consequences of IoT devices lacking sufficient security mechanisms.

As mentioned earlier, the scope of this thesis is limited to the security of IoT networks in a smart home. The home is a very private space for individuals hence the lack of adequate protection would leave the members of smart homes open to attacks by malicious hackers who tend to harm. For instance, the CCTV cameras in a household may be connected to the Internet to allow the members of the family to remotely access live feed when they are away from home. It could also alert security services of any intrusions. All these services provided are meant to ensure that the members of the household enjoy a more comfortable standard of living. However, due to the lack of security, hackers with malicious intent may exploit this weakness and intrude on the privacy of the individuals in the household. They may also turn off the functionality to alert security services of any intrusions or break-ins to the homes causing physical and psychological harm to the members of the household. All these attacks can be

prevented or at least detected if there are robust security mechanisms ingrained in the architecture of IoT systems to ensure consumer protection.

1.5 Problem Statement

The problem of having vulnerabilities in a smart home environment would grow to a much bigger scale if left unchecked. It would be every hacker's dream considering the estimated number of devices in circulation. All data and devices would be left vulnerable to attacks, and people would not feel safe in their homes. Despite the advancements in IoT systems, there have not been comparatively significant advancements in terms of the security of the network linking these devices. There is a considerable gap in terms of the security and protection of IoT devices in smart homes. This thesis seeks to find a highly portable and extremely scalable security mechanism for IoT systems in a smart home environment.

There are various alternatives to protecting IoT systems from malicious actors. Some of these alternatives include creating a gateway to serve as a central point of control between the IoT devices and the outside world (i.e. the Internet) [11]. However, this solution does not secure the system from malicious attacks but only secures the system based on user configuration. Another solution such as public-key cryptography is not feasible because the IoT devices are resource-constrained and public-key cryptography is resource-intensive [12]. Furthermore, there is research being done into lightweight cryptography solutions [13,14]. However, they are not robust enough to protect the network from internal attackers (i.e., nodes that have been granted more access than required), or from external attacks like DDoS. One other possible solution to protect the IoT system is using an intrusion detection system on a Raspberry Pi. This low-powered resource-constrained device is the security mechanism being investigated in this thesis.

An intrusion detection system (IDS) is a software tool that monitors and scans traffic on a network for suspicious activity and, based on configuration, sends alerts when any suspicious activity is detected. IDSs installed at a point within the network to examine traffic

from all devices on the network are known as Network Intrusion Detection Systems (NIDS). NIDS are mainly developed to be run on laptops and hence the existing software may not be used on IoT systems because of the limited resources available in the network. One such example is Snort 3, a well-known open-source system that uses both signature-based and anomaly-based methods to detect anomalies.

1.6 Research Questions

The research questions for this thesis, based on the problem identified, are as follows:

1. Can a Raspberry Pi provide the computational requirements of Snort 3?
2. Would Snort 3 installed on a Raspberry Pi be able to identify malicious behaviour?

1.7 Objective of the Thesis

This thesis seeks to propose a cheap way to ensure the security of IoT networks by successfully mounting Snort 3 on a Raspberry Pi. This resource-constrained low powered device is typical of most devices that form part of the network.

1.8 Organization of the Paper

This thesis is structured as follows: Chapter 1 introduces the background and expected future of IoT systems. It also describes the scope of the thesis and mentions the main objective of this thesis. Chapter 2 describes related work in the field of security mechanisms in IoT systems, specifically those that utilise NIDSs. Chapter 3 describes the methodology used in this research. Chapter 4 discloses and analyses the results of this research and Chapter 5 summarises the research conducted and serves as the conclusion for this thesis.

Chapter 2: Background and Related Work

2.1 Introduction

Generally, the IoT concept is a new and fresh concept that offers so many opportunities in terms of improving the efficiency of systems and generally making life easier and more comfortable [15]. An example would be using your smartphone to turn the lights off when no one is home [16]. Before IoT became as popular as it is now, it started as Machine to Machine (M2M) communication. M2M primarily signifies two machines communicating with each other without human control. The mode of communication could be wired, i.e. with cables or wireless, or over a network such as the Internet. However, with this rapid increase in the technological advancement of IoT systems, there has been a lag in the development of security protocols administered to these systems [17]. The various new features added to these systems have dramatically increased the attack surface areas of these systems, making these systems more vulnerable to attacks.

2.2 Security Issues

This section describes the various security issues related to an IoT system, specifically, a smart home automation system. Since there is no standardised architecture for IoT systems, the architecture referenced to in this thesis is that propounded by Hassija, Chamola, Saxena, Jain, Goyal, and Sikdar, [18].

The authors of this paper propose that any IoT application, whether it be for home automation, smart cities or smart retail, can be divided into four layers, which are:

- Sensing layer – the primary purpose of this layer is to collect data from the smart environment where these devices can be found. As the name suggests, the sensors that form part of the IoT system can be found in this layer.
- Network layer – the main purpose of this layer is to transmit collected data to the appropriate endpoints

- Middleware layer - the middleware layer helps to create some form of abstraction between the network layer and the application layer. This layer provides application programming interfaces (APIs) to the application layer. Hence, the application layer does not interact directly with the network layer.
- Application layer– the main purpose of the application layer is to provide services requested to the user. The user interacts with the entire system through this layer[18]
- Gateway –this part of the architecture does not constitute a layer. However, it is used to support administrative services, such as the addition of new interfaces.

The authors go ahead to report on the various types of attacks that can be encountered at the layers mentioned above. The different types of attacks that can be leveraged against the multiple layers have been summarised in Figure 2.1 below.

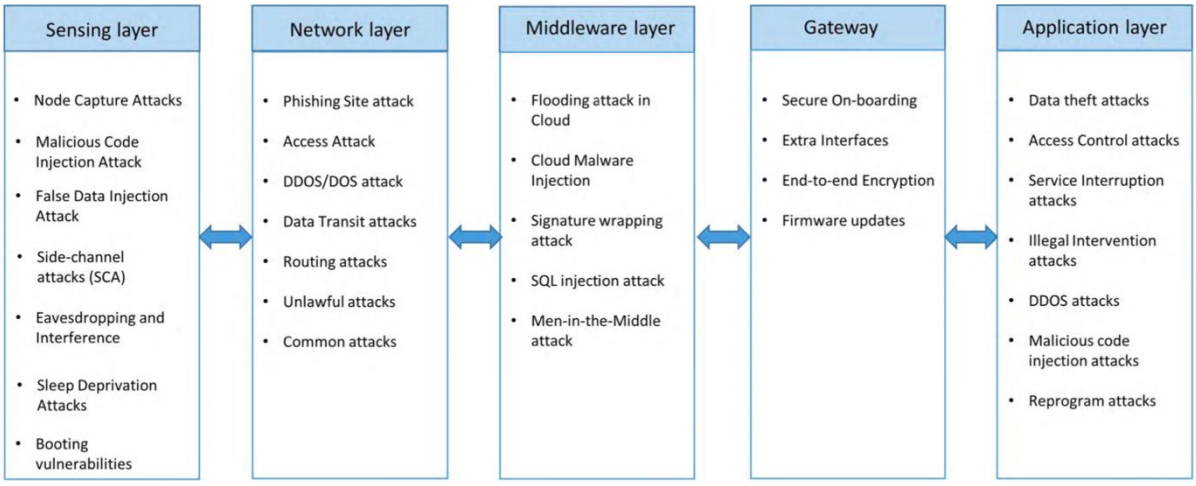


Fig. 2.1: Types of attacks on IoT systems (taken from [18])

2.3 Related Work

This section presents the works done by other researchers into using IDSs as a security mechanism for IoT networks.

Li et al. [19] present an IDS that makes use of signature-based approaches and employs Artificial Immune System Mechanism. The authors propose that attack signatures are modelled as immune cells that can be classified as malicious or normal. The authors do not recommend which placement strategy should be adopted. According to them, the computational

requirements needed to run the algorithms cannot be handled by the IoT network due to the resource constraint.

Misra et al. [20] propose a specification-based detection method as a solution to prevent DDoS attacks over the middleware layer of IoT systems. The limitation of this solution is that there is no standardised architecture for IoT. Hence, some IoT systems may lack the middleware layer. Therefore, this solution is not compatible with all IoT systems.

Gupta et al. [21] present an architecture for an IDS that constructs standard behaviour profiles for each device on the network with an assigned IP address. The authors propose that profiling would be done using Computational Intelligence algorithms. Aside from the resource constraints of IoT devices, no type of attack could be detected by the proposed solution.

Cho et al. [22] propose an IDS whereby packets that pass through the border router (the router connecting physical devices and the network) are examined to detect botnet attacks. Their proposed solution utilises anomaly-based approaches and assumes that botnets caused unexcepted changes in traffic. The limitation of this solution is that there may be a device that has unusual traffic simply because of increased usage. Hence, the proposed IDS would classify the device as an intruder.

Khanum and Usman [23] propose an IDS mechanism that makes use of wireless authentication and encryption systems. The result of the study showed that the proposed mechanism resulted in numerous false alarms detected and failed to adjust to discover new types of attacks.

Hugelshofer et al. [24] propose a lightweight IDS that majorly decrease memory consumptions. However, their solution does not identify the majority of the primary attacks and only detects a few, such as IP spoofing.

Farooqi and Khan [25] propose a distributed IDS to monitor nodes that are neighbours to the other. The authors assume that a malicious actor cannot take over existing nodes or

introduce a new node into the system. The proposed solution creates a form of the trust relationship between neighbour nodes, and hence the solution is not plausible if one of the trusted nodes is being attacked.

Krimmling and Peter [26] propose an IDS that employs a hybrid detection method by merging signature-based approaches as well as anomaly-based approaches. The tests they conducted showed that the proposed system failed to detect some frequent attacks such as Man-In-The-Middle attacks.

Cervantes et al. [27] propose an IDS where the role of each node in the system is to monitor a superior node, estimating the traffic patterns of that node. This approach also utilises the concept of trust, just like what was employed by Jonckers[11]. Since all the nodes are monitoring the superior nodes, one of the nodes can be under the control of malicious actors, and the other nodes would not detect this.

Thanigaivelan et al. [28] propose a hybrid IDS where network nodes and the border router are assigned different tasks. Each node monitors the other and sends notifications of possible attacks to the border router. The border router then utilises anomaly-based detection based on normal behaviour learned to determine if there was an actual intrusion or not. This solution as at now is not plausible because the authors did not define the scope of normal behaviour.

Based on the literature review conducted, I identified some gaps that my proposed solution would attend to. One gap identified is that some authors failed to offer a placement strategy along with their proposed solution. Hence, the placement strategy I am adopting would be to connect the Raspberry Pi with Snort 3 installed directly into the router in the smart home. I chose this because the router is the entry point for data packets into that environment. Hence, the device is placed right after the entry point and, therefore, can start to detect anomalies in the network.

Another such gap identified is that the previous solutions failed to detect certain anomalies and intrusions. This is mainly because the authors proposed new IDSs from scratch. However, these IDSs were not robust enough and the IoT networks were still susceptible to attacks. One way to solve this issue is to use a tried and tested intrusion detection software. Thus, the IDS chosen for this Thesis is Snort 3, which is an open-source NIDS. It was selected because it is the de facto standard for IDS, and its modular design allows for flexibility.

Table 2.1 below outlines a comparative analysis undertaken to identify the Network Intrusion Detection Systems (NIDS) with the lowest resource intensity among the top four NIDS. This analysis seeks to reinforce the use of Snort 3 as the best NIDS to undertake this research because of its low resource intensity.

Table 2.1: Summary of comparative analysis of top four NIDS

NIDS	Features	Deployment Platform	Benefits	Weaknesses
Snort [29]	• Support multiple packet processing threads • Use a shared configuration and attribute table • Autodetect services for port less configuration • Modular design • Plugin framework with over 200 plugins • More scalable memory profile	Windows NT and 2000, Unix (Solaris, HP-UX, IRIX, OpenBSD, NetBSD, Free BSD, and Mac OS X), Linux,	-A vast community of users, many support resources available online [29] - It has low CPU usage [35]	-Snort has no user interface or easy-to-use administrative console [35]

	• LuaJIT configuration, loggers, and rule options • Hyperscan support • Rewritten TCP handling • New rule parser and syntax [29]	and even on PowerPCs.		
Suricata [30]	• High performance - multi-threaded, scalable codebase • Multipurpose Engine - NIDS, NIPS, NSM, offline analysis, etc. • Cross-platform support - Linux, Windows, macOS, OpenBSD, etc. • Modern TCP/IP support including a scalable flow engine, full IPv4/IPv6, TCP streams, and IP packet defragmentation • Protocol parsers - packet decoding, application layer decoding • HTTP engine - HTTP parser, request logger, keyword match, etc.	• Linux • FreeBSD •OpenBSD •macOS / Mac OS X •Windows [30]	- Suricata supports multithreading, which means it can use multiple cores at once. Hence, it can process large quantities of traffic without having to cut back on rules. [30]	- Suricata is prone to false positives. - System and network resource-intensive

	•Autodetect services for portless configuration • Lua scripting (LuaJIT) • Application-layer logging and analysis, including TLS/SSL certs, HTTP requests, DNS requests, and more [30]			
Zeek [31]	• Fully passive traffic analysis off a network tap or monitoring port • Standard libpcap interface for capturing packets • Real-time and offline analysis • Cluster-support for large-scale deployments • Unified management framework for operating both standalone and cluster setups. • Open source under a BSD license • Support for many application-layer protocols	Runs on commodity hardware on standard UNIX-style systems (including Linux, FreeBSD, and macOS) [31]	- It can be tailored for a variety of network use cases in addition to NIDS (Network Intrusion Detection System) -It is different from other tools in that it doesn't depend on a specific	- Zeek is aimed at providing security solutions for high-performance networks. This is listed as a weakness because it makes the system resource-intensive

<table>
<tr>
<td></td>
<td>(including DNS, FTP, HTTP, IRC, SMTP, SSH, SSL)
• Analysis of file content exchanged over application-layer protocols, including MD5/SHA1 computation for fingerprinting [31]</td>
<td></td>
<td>detection approach Another critical point is that it is not dependent on traditional signatures [31]</td>
<td>- It is primarily a wireless solution [31]</td>
</tr>
<tr>
<td>Sguil [32]</td>
<td>This software is a collection of free software components for Network Security Monitoring (NSM) and event-driven analysis of IDS alerts. Below are the various tools which form part of Sguil as well as their multiple functions. [33]
<table>
<tr><td>MySQL 4.x or 5.x</td><td>Data storage and retrieval</td></tr>
<tr><td>Snort 2.x / Suricata</td><td>Intrusion detection alerts, scan</td></tr>
</table></td>
<td>It can run on any operating system that supports tcl/tk (including Linux, *BSD, Solaris, macOS, and Win32) [33]</td>
<td>- It provides a GUI, which makes it easier to use</td>
<td>- It is a resource-intensive application as the specification of recommended server hardware is as follows.
CPU = 3.0 GHz
RAM = 2GB
Disk Storage = 150 [32]</td>
</tr>
</table>

		detection, packet logging			
		Barnyard / Barnyard2	Decodes IDS alerts and sends them to sguil		
		SANCP	TCP/IP session records		
		Tcpflow	Extract an ASCII dump of a given TCP session		
		p0f	Operating system fingerprinting		

	<table><tr><td>tcpdump</td><td>Extracts individual sessions from packet logs</td></tr><tr><td>Wireshark</td><td>Packet analysis tool</td></tr></table> [34]			

Chapter 3: Methodology/Approach

The primary purpose of this study is to demonstrate that a Raspberry Pi device has the computational power to run Snort 3 continuously. The study also seeks to show that a Raspberry Pi equipped with Snort 3 can be used to protect a smart home from malicious actors by being able to detect malicious activity such as the different types of attacks summarised in Fig. 2.1. This chapter outlines the steps taken to find answers to the above research questions and thoroughly describes the tools and devices used to conduct this research.

3.1 Description of Research Design

The research is conducted in two phases. The first phase employs the use of build methodology. Build methodology is a type of research approach that consists of the building of an artefact to demonstrate the possibility of the existence of such a system. This methodology is appropriate because it establishes that it is indeed possible to deploy Snort 3 on a Raspberry Pi.

The second phase employs experimentation. Using the device, it would be used to assess how the Raspberry Pi performs as the Snort 3's host based on:

- CPU performance; and

- The number of alerts generated.

As a way to ensure the credibility of the experiment results and to determine that Snort is functioning as correctly as it would on an ideal device, a control experiment was conducted by assessing the number of alerts generated by Snort running on an ideal device (a laptop).

3.2 Research Method

3.2.1 Phase One

As described above, this phase involves installing and configuring Snort 3 on a Raspberry Pi.

Snort 3: The IDS chosen for this research is Snort 3. It is a robust network intrusion detection system, which is capable of analysing network traffic in real-time. This was the IDS chosen for the project because:

- It is open-source; hence other developers and members of the Snort 3 community identify loopholes that can be exploited and report back to the Snort 3 development team to add new rules to fix gaps identified. This feature of Snort 3 helps to ensure that the system is robust because it is always updated to match new security threats identified.
- It is widely used by private users and institutions.
- Additional reasons can be found in the comparative analysis conducted and reported in the previous chapter.

Raspberry Pi: It is a small-sized computer with relatively significant computing power. One alternative to the Raspberry Pi I considered was Arduino. However, a Raspberry Pi is utilised in the research because, among the two devices, the Raspberry Pi is the more suitable candidate in terms of storage and computing speed [36].

Installation and configuration were done following Noah Dietrich's guide to installing Snort 3 on Ubuntu. This guide was useful because Ubuntu is based on Debian, which is the OS running on the Raspberry Pi.

3.2.2 Phase Two

The next phase of the research involved experimentation to assess the performance of the Raspberry Pi as Snort 3's host. The two primary devices used to conduct this experiment were:

1. The Raspberry Pi 2 Model B is a Quad-Core ARM Cortex-A7 with 1 GB of memory. We installed the operative system Raspbian (Debian Wheezy) and Snort 3.0.0. We equipped the Raspberry Pi with a 32 GB class 10 MicroSD card.
2. The computer's hardware configuration was a 3.3 GHz Intel Core i5 with 8 GB of memory and equipped with Ubuntu 18.04.
3. An Ethernet crossover cable to serve as a link between both devices. It is worth mentioning that the maximum speed on the link between the two devices was 100 Mbit/s.

3.2.2.1 Network traffic simulation

To simulate network traffic, both malicious and normal, a pcap file downloaded from https://www.netresec.com/index.ashx?page=PcapFiles, an online public repository was used. The 654401 packets in this file were divided into three categories based on the sizes of the packets present in the pcap file. This pcap file was chosen because it contains packets that are likely to be present in a smart home network e.g.

Small – packets of length less than 150 bytes

Long – packets of length of at least 1000 bytes

Mixed– packets of all sizes.

The categorisation of the files is necessary because packet length influences CPU consumption. To simulate network traffic, both devices were connected via the crossover cable. A python script mainly utilising the python library Scapy, a packet manipulation tool was used to categorise packets and sent to the Raspberry Pi mimicking normal network conditions.

3.2.2.2 Recorded statistics

Sysstat, a performance monitoring tool on Linux systems was installed on the Raspberry Pi to record the average CPU usage in percentages. This was necessary to observe the CPU utilisation as the sizes of the packets on the testing network were changing as well as to observe whether a Raspberry Pi could provide the necessary computational resources required by Snort 3.

3.2.2.3 IDS Rules

Snort 3 employs the use of rules to perform its detection duties as these rules are what provide the desired security. Thus enabling more rules does not translate into increased security. Rather, it translates into more CPU consumption as would be described in Chapter 4. The ruleset utilised in this Thesis is Snort 3 subscriber ruleset.

3.3 Experiments Procedure

3.3.1 CPU Performance.

This part of the experiment was carried out in three waves. The python script below was used to facilitate the first wave, which involved sending the packets in the small category to the Raspberry Pi with Snort 3 listening for packets on the Raspberry Pi's ethernet interface.

```
from scapy.all import *
packets = rdpcap("/home/sharon/Documents/2018-12-22-15-50-15-192.168.1.195.pcap")
small = []

#grouping small packets
for packet in packets:
  try:
    if packet.len <= 150:
      small.append(packet)
  except Exception:
    continue

smallPackets = PacketList(res = large)

print("Commence sending")
sendp(smallPacket, iface="enp2s0")
```

Figure 3.1: Python Code snippet

Simultaneously, sysstat was monitoring the CPU usage twice every second. For the second and third wave, the above procedure was repeated but with the packets from the other two categories: large and mixed. The number of packets in each category was approximately 70000.

3.3.2 Number of alerts generated

For this part of the experiment, Snort 3 was installed and configured on a computer. The computer's hardware configuration was a 1.60 GHz Intel Core i5 with 8 GB of memory and equipped with Ubuntu 18.04. The same network traffic was simulated on both the Raspberry Pi and on the laptop with Snort 3 listening for packets on the ethernet interfaces of both devices. The alerts generated from both devices were stored in a text file for analysis.

Table 3.1: Statistics monitored during the experiments

Name	Description
Avg. CPU	Average Raspberry Pi CPU usage for a given experiment
Alerts	No. of alerts generated by Snort for a given experiment

Chapter 4: Results

This chapter discusses the results from the experiments described in the previous chapter. This thesis sought to answer two research questions: Can a Raspberry Pi provide the computational requirements of Snort 3? Would Snort 3 installed on a Raspberry Pi be able to identify malicious behaviour?

In the assessment of the computational abilities, the CPU consumption of the Raspberry Pi was recorded while packets of different sizes were transmitted from the computer to the Raspberry Pi. The graph below summarises the research findings.

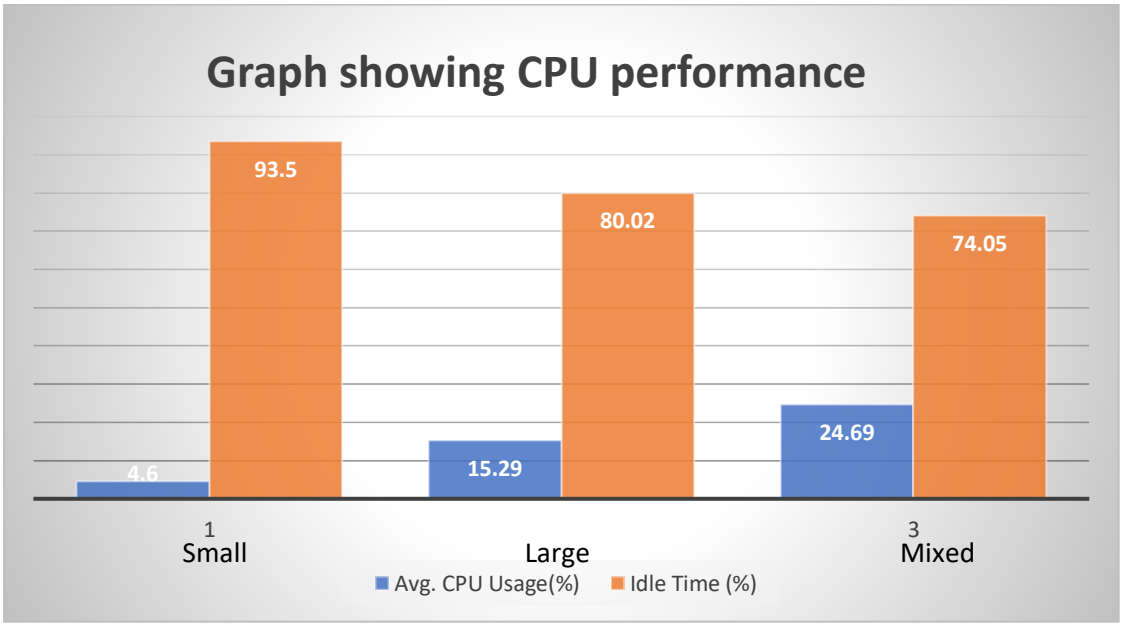


Figure 4.1: Average CPU performance per packet category

Figure 4.1 shows that, for the packets in the small category, the average CPU consumption is 4.6% with 93.5% idle time and for the large category, the average CPU consumption is 15.29% with 80.02 idle time. For the mixed category, the average CPU consumption is 24.69% with 74.05% idle time. This trend asserts the logic that CPU consumption increases as it is subjected to higher processing power. However, it is worth noting that even with the packets in the large and mixed category, the CPU consumption is not up to 50%. This proves that a Raspberry Pi can provide the necessary computational requirements Snort 3 requires thereby answering the first research question in the affirmative.

It is also worth noting that the process of loading the rule set in Snort 3 requires an average of 25% CPU consumption. However, this happens only once on start up of Snort 3.

To answer the second research question, the same network activity was simulated on both the Raspberry Pi and a laptop, which served as a control experiment. The objective of this experiment was to determine whether Snort 3 on a Raspberry Pi would be able to correctly detect malicious packets as it would on an ideal device (laptop). All the alerts generated were stored in a .csv file for analysis. From the pcap file containing 654401 packets, 76119 of those packets were flagged as malicious by Snort 3 running on the laptop. There was a 100% match between the alerts generated by Snort 3 on the laptop and Snort 3 on the Raspberry Pi. Figure 4.2 below summarises the number of alerts generated per protocol. This graph is representative of the results from both the Raspberry Pi and the laptop.

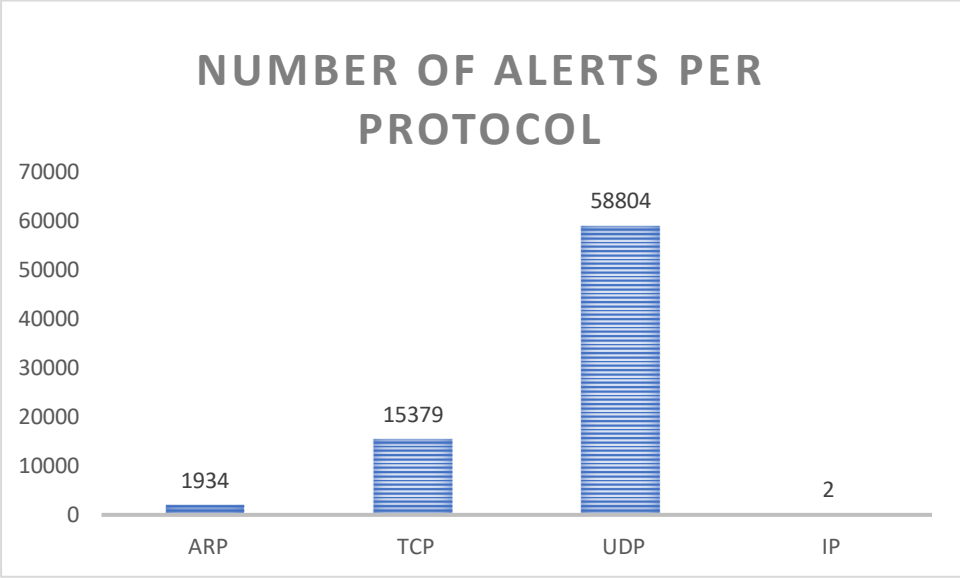


Figure 4.2: The number of alerts generated by Snort 3 per protocol

The 100% match between the results of both devices proves that Snort 3 can function as effectively on a Raspberry Pi as it would on a laptop.

Chapter 5: Conclusion

In conclusion, this study sought to prove that a Raspberry Pi can serve as an effective and efficient host of Snort 3. This study mainly monitored the CPU consumption and the number of alerts generated by Snort 3 running on a Raspberry Pi, connected to an apparent smart home network. The results demonstrated that a Raspberry Pi has the computational resources necessary to host Snort 3. It is worth mentioning that Snort 3 resource demands did not overwhelm the Raspberry Pi. As such, it would be a very useful tool in smart homes. By utilising this placement strategy, all the packets entering the network are scanned and analysed before they are allowed into the network. This thesis highlights one of the affordable ways to ensure the security of IoT networks by successfully mounting Snort 3 on a Raspberry Pi.

One limitation of this study is that tests were carried out with network traffic recorded in trace files rather than real live traffic in a smart home. Future research may focus on running experiments with intrusion detection over wireless protocols such as MQTT, WiFi and Bluetooth. Additionally, further research can be conducted by investigating some alternatives to Snort 3, such as Suricata and Zeek. Additional examinations can be directed at uncovering the suitability of an Arduino Uno as a host for Snort 3.

References

[1] Domingo, M.C. 2012. An overview of the Internet of Things for people with disabilities. *Journal of Network and Computer Applications*. 35, 2 (Mar. 2012), 584–596. DOI:https://doi.org/10.1016/j.jnca.2011.10.015.

[2] IoT In Mining: Benefits, Challenges And Latest Case Studies: 2019. *http://www.fitchsolutions.com/corporates/metals-mining/IoT-mining-benefits-challenges-and-latest-case-studies-20-03-2019*. Accessed: 2019-10-11.

[3] The little-known story of the first IoT device: 2019. *https://www.ibm.com/blogs/industries/little-known-story-first-IoT-device/*. Accessed: 2019- 10- 11.

[4] Ashton, K. 2009. That "Internet of Things" Thing. 1.

[5] Wilson, C., Hargreaves, T. and Hauxwell-Baldwin, R. 2015. Smart homes and their users: a systematic analysis and key challenges. *Personal and Ubiquitous Computing*. 19, 2 (Feb. 2015), 463–476. DOI:https://doi.org/10.1007/s00779-014-0813-0.

[6] Vermesan, O. and Friess, P. 2013. *Internet of things: converging technologies for smart environments and integrated ecosystems*. River Publishers.

[7] Khera, M. 2017. Think Like a Hacker: Insights on the Latest Attack Vectors (and Security Controls) for Medical Device Applications. *Journal of Diabetes Science and Technology*. 11, 2 (Mar. 2017), 207–212. DOI:https://doi.org/10.1177/1932296816677576.

[8] "2015 U.S. Digital Future in Focus," comScore, 2015.

[9] Angrishi, K. 2017. Turning Internet of Things (IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets. *arXiv:1702.03681 [cs]*. (Feb. 2017).

[10] Hackers release source code for a powerful DDoS app called Mirai: 2019. *https://techcrunch.com/2016/10/10/hackers-release-source-code-for-a-powerful-ddos-app-called-mirai/*. Accessed: 2019- 10- 13.

[11] Jonckers, D. 2016. A security mechanism for the Internet of Things in a smart home context. KU Leuven. 108.

[12] Atzori, L., Iera, A. and Morabito, G. 2010. The Internet of Things: A Survey. *Comput. Netw.* 54, 15 (Oct. 2010), 2787–2805. DOI:https://doi.org/10.1016/j.comnet.2010.05.010.

[13] Feldhofer, M., Dominikus, S. and Wolkerstorfer, J. 2004. Strong Authentication for RFID Systems Using the AES Algorithm. *Cryptographic Hardware and Embedded Systems - CHES 2004* (2004), 357–370.

[14] Eschenauer, L. and Gligor, V.D. 2002. A Key-management Scheme for Distributed Sensor Networks. *Proceedings of the 9th ACM Conference on Computer and Communications Security* (New York, NY, USA, 2002), 41–47.

[15] Lanzisera, S., Weber, A.R., Liao, A., Pajak, D. and Meier, A.K. 2014. Communicating power supplies: Bringing the internet to the ubiquitous energy gateways of electronic devices. *IEEE Internet of Things Journal*. 1, 2 (Apr. 2014), 153–160. DOI:https://doi.org/10.1109/JIOT.2014.2307077.

[16] Atzori, L., Iera, A. and Morabito, G. 2010. The Internet of Things: A survey. *Computer Networks*. 54, 15 (Oct. 2010), 2787–2805. DOI:https://doi.org/10.1016/j.comnet.2010.05.010.

[17] Mandeep Khera. 2017. Think Like a Hacker: Insights on the Latest Attack Vectors (and Security Controls) for Medical Device Applications. *J Diabetes Sci Technol* 11, 2 (March 2017), 207–212. DOI:https://doi.org/10.1177/1932296816677576

[18] Hassija, V., Chamola, V., Saxena, V., Jain, D., Goyal, P. and Sikdar, B. 2019. A Survey on IoT Security: Application Areas, Security Threats, and Solution Architectures. *IEEE Access*. 7, (2019), 82721–82743. DOI:https://doi.org/10.1109/ACCESS.2019.2924045.

[19] Liu, C., Yang, J., Chen, R., Zhang, Y. and Zeng, J. 2011. Research on immunity-based intrusion detection technology for the Internet of Things. *2011 Seventh International Conference on Natural Computation* (Jul. 2011), 212–216.

[20] Misra, S., Krishna, P.V., Agarwal, H., Saxena, A. and Obaidat, M.S. 2011. A Learning Automata Based Solution for Preventing Distributed Denial of Service in Internet of Things. *Proceedings of the 2011 International Conference on Internet of Things and 4th International Conference on Cyber, Physical and Social Computing* (Washington, DC, USA, 2011), 114–122.

[21] Gupta, A., Pandey, O.J., Shukla, M., Dadhich, A., Mathur, S. and Ingle, A. 2013. Computational intelligence based intrusion detection systems for wireless communication and pervasive computing networks. *2013 IEEE International Conference on Computational Intelligence and Computing Research* (Dec. 2013), 1–7.

[22] Cho, E.J., Kim, J.H. and Hong, C.S. 2009. Attack Model and Detection Scheme for Botnet on 6LoWPAN. *Proceedings of the 12th Asia-Pacific Network Operations and Management Conference on Management Enabling the Future Internet for Changing Business and New Computing Services* (Berlin, Heidelberg, 2009), 515–518.

[23] Khanum, S. and Usman, M. 2012. Mobile Agent Based Hierarchical Intrusion Detection System. *in Wireless Sensor Networks." International Journal of Computer Science Issues, IJCSI* (2012).

[24] Hugelshofer, F., Smith, P., Hutchison, D. and Race, N.J.P. 2009. OpenLIDS: a lightweight intrusion detection system for wireless mesh networks. *Proceedings of the 15th annual international conference on Mobile computing and networking - MobiCom '09* (Beijing, China, 2009), 309.

[25] Farooqi, A.H. and Khan, F.A. 2009. Intrusion Detection Systems for Wireless Sensor Networks: A Survey. *Communication and Networking* (2009), 234–241.

[26] Krimmling, J. and Peter, S. 2014. Integration and evaluation of intrusion detection for CoAP in smart city applications. *2014 IEEE Conference on Communications and Network Security* (San Francisco, CA, USA, Oct. 2014), 73–78.

[27] Cervantes, C., Poplade, D., Nogueira, M. and Santos, A. 2015. Detection of sinkhole attacks for supporting secure routing on 6LoWPAN for Internet of Things. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)* (Ottawa, ON, Canada, May 2015), 606–611.

[28] Thanigaivelan, N.K., Nigussie, E., Kanth, R.K., Virtanen, S. and Isoaho, J. 2016. Distributed internal anomaly detection system for Internet-of-Things. *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)* (Las Vegas, NV, USA, Jan. 2016), 319–320.

[29] Snort - Network Intrusion Detection & Prevention System: 2020. *https://www.snort.org/snort3*. Accessed: 2020- 04- 02.

[30] Suricata: 2020. *https://suricata-ids.org/*. Accessed: 2020- 04- 02.

[31] The Zeek Network Security Monitor: 2020. *https://zeek.org/*. Accessed: 2020- 04- 02.

[32] Sguil - Open Source Network Security Monitoring: 2020. *https://bammv.github.io/sguil/docs.html*. Accessed: 2020- 04- 02.

[33] Lockhart, A. 2006. *Network Security Hacks. Hack 108 - Monitor Your IDS in Real Time - Use Sguil's advanced GUI to monitor and analyse IDS events in a timely manner*. O'Reilly Media.

[34] Cox, K. and Gerg, C. 2004. *"13: Strategies for High-Bandwidth Implementations of Snort". Managing security with Snort and IDS tools*. O'Reilly.

[35] Pukkawanna, S., Pongpaibool, P. and Visoottiviseth, V. 2008. LD2: A system for lightweight detection of denial-of-service attacks. (Dec. 2008), 1–7.

[36] Patnaikuni, D.R.P. 2017. A Comparative Study of Arduino, Raspberry Pi and ESP8266 as IoT Development Board. *International Journal of Advanced Research in Computer Science*. 8, 5 (Jun. 2017), 2350–2352. DOI:https://doi.org/10.26483/ijarcs.v8i5.3959.

[37] Snort 3 on Ubuntu 18 & 19: 2020. *https://snort-org-site.s3.amazonaws.com/production/document_files/files/000/000/211/original/Snort3.pdf?X-Amz-Algorithm=AWS4-HMAC-SHA256&X-Amz-Credential=AKIAIXACIED2SPMSC7GA%2F20200410%2Fus-east-1%2Fs3%2Faws4_request&X-Amz-Date=20200410T180355Z&X-Amz-Expires=172800&X-Amz-SignedHeaders=host&X-Amz-Signature=8047df5fb4445db46f091cf72ac039d587152f99f6d91b5b7088cc53fd62bed1*. Accessed: 2020- 04- 10.

[38] Zouinkhi, A., Ayadi, H., Val, T., Boussaid, B. and Abdelkrim, M.N. 2020. Auto-management of energy in IoT networks. *International Journal of Communication Systems*. 33, 1 (Jan. 2020), e4168. DOI:https://doi.org/10.1002/dac.4168.

Appendix

Statistics after Snort 3 processes packets

```
--------------------------------------------------
Packet Statistics
--------------------------------------------------
daq
                   pcaps: 1
                received: 654401
                analyzed: 654401
                   allow: 654401
                rx_bytes: 1266212769
--------------------------------------------------
codec
                   total: 713201       (100.000%)
                discards: 174215       ( 24.427%)
                     arp: 3810         (  0.534%)
                     eth: 713201       (100.000%)
                    ipv4: 709391       ( 99.466%)
                     tcp: 242682       ( 34.027%)
                     udp: 62829        (  8.809%)
--------------------------------------------------
Module Statistics
--------------------------------------------------
appid
                 packets: 535176
       processed_packets: 448655
         ignored_packets: 86521
          total_sessions: 77380
           appid_unknown: 3316
      service_cache_adds: 3657
--------------------------------------------------
arp_spoof
                 packets: 3810
--------------------------------------------------
back_orifice
                 packets: 62829
--------------------------------------------------
binder
                 packets: 75902
                inspects: 75902
--------------------------------------------------
detection
                analyzed: 654401
              hard_evals: 4
            raw_searches: 9394
         cooked_searches: 67216
            pkt_searches: 76610
            total_alerts: 76119
                  logged: 76119
```

```
-------------------------------------------------
dns
                packets: 1429
               requests: 1322
              responses: 107
-------------------------------------------------
normalizer
         test_tcp_options: 15970
        test_tcp_trim_win: 893
          test_tcp_ts_nop: 30413
          test_tcp_block: 793
-------------------------------------------------
perf_monitor
                packets: 717409
-------------------------------------------------
port_scan
                packets: 709391
-------------------------------------------------
search_engine
              max_queued: 44
            total_flushed: 3179
            total_inserts: 3179
             total_unique: 3179
     non_qualified_events: 3183
          searched_bytes: 484308361
-------------------------------------------------
stream
                  flows: 75902
             total_prunes: 15256
              idle_prunes: 15256
-------------------------------------------------
stream_ip
               sessions: 64682
                   max: 43393
                created: 64682
               released: 64682
             total_frags: 403880
            current_frags: 58800
              reassembled: 58800
           trackers_added: 64682
           trackers_freed: 64682
         trackers_cleared: 64682
       trackers_completed: 58800
           nodes_inserted: 403880
            nodes_deleted: 403880
        reassembled_bytes: 485553180
         fragmented_bytes: 589027452
-------------------------------------------------
stream_tcp
               sessions: 6420
```

```
                max: 2052
            created: 6420
           released: 5705
           timeouts: 1776
       instantiated: 2001
             setups: 6420
           discards: 112
             events: 52
       syn_trackers: 4091
   syn_ack_trackers: 1
      data_trackers: 1613
        segs_queued: 5465
      segs_released: 5465
          segs_used: 5433
    rebuilt_packets: 4208
      rebuilt_bytes: 388184
    client_cleanups: 2606
    server_cleanups: 1600
               syns: 14715
           syn_acks: 1039
             resets: 6
               fins: 8172
-------------------------------------------------
stream_udp
           sessions: 4800
                max: 2238
            created: 6278
           released: 6278
           timeouts: 1478
-------------------------------------------------
tcp
   bad_tcp4_checksum: 174215
-------------------------------------------------
wizard
          tcp_scans: 5433
          udp_scans: 61400
-------------------------------------------------
Appid Statistics
-------------------------------------------------
detected apps and services
        Application: Flows   Clients  Users   Payloads  Misc   Incompat.  Failed
          unknown: 893       4747     0       0         0      0          0
-------------------------------------------------
Summary Statistics
-------------------------------------------------
timing
           runtime: 00:03:22
           seconds: 202.244956
           packets: 654401
          pkts/sec: 3239
```

o")~ Snort exiting

Statistics after Snort 3 processes packets (on laptop)

```
Packet Statistics
-------------------------------------------------
daq
                   pcaps: 1
                received: 654401
                analyzed: 654401
                   allow: 654401
                rx_bytes: 1266212769
-------------------------------------------------
codec
                   total: 713201       (100.000%)
                discards: 174215       ( 24.427%)
                     arp: 3810         (  0.534%)
                     eth: 713201       (100.000%)
                    ipv4: 709391       ( 99.466%)
                     tcp: 242682       ( 34.027%)
                     udp: 62829        (  8.809%)
-------------------------------------------------
Module Statistics
-------------------------------------------------
appid
                packets: 535176
      processed_packets: 448655
        ignored_packets: 86521
         total_sessions: 77380
          appid_unknown: 3316
     service_cache_adds: 3657
-------------------------------------------------
arp_spoof
                packets: 3810
-------------------------------------------------
back_orifice
                packets: 62829
-------------------------------------------------
binder
                packets: 75902
                inspects: 75902
-------------------------------------------------
detection
                analyzed: 654401
              hard_evals: 4
             raw_searches: 9394
          cooked_searches: 67216
             pkt_searches: 76610
             total_alerts: 76119
                   logged: 76119
-------------------------------------------------
dns
```

packets: 1429
requests: 1322
responses: 107
---
normalizer
test_tcp_trim_win: 81
test_tcp_ts_nop: 2489
test_tcp_block: 52
---
pcre
pcre_rules: 2091
pcre_native: 2091
---
port_scan
packets: 709391
trackers: 76
---
search_engine
max_queued: 2
total_flushed: 3179
total_inserts: 3179
total_unique: 3179
non_qualified_events: 3183
searched_bytes: 484308361
---
stream
flows: 75902
total_prunes: 15256
idle_prunes: 15256
---
stream_ip
sessions: 64682
max: 64682
created: 64682
released: 64682
total_bytes: 573680012
total_frags: 403880
current_frags: 58800
reassembled: 58800
trackers_added: 64682
trackers_freed: 64682
trackers_cleared: 64682
trackers_completed: 58800
nodes_inserted: 403880
nodes_deleted: 403880
reassembled_bytes: 485553180
fragmented_bytes: 589027452
---
stream_tcp
sessions: 6420

max: 6420
created: 6420
released: 5705
timeouts: 1776
instantiated: 2001
setups: 6420
discards: 112
events: 52
syn_trackers: 4091
syn_ack_trackers: 1
data_trackers: 1613
segs_queued: 5465
segs_released: 5465
segs_used: 5433
rebuilt_packets: 4208
rebuilt_bytes: 388184
client_cleanups: 2606
server_cleanups: 1600
syns: 14715
syn_acks: 1039
resets: 6
fins: 8172
---
stream_udp
sessions: 4800
max: 4800
created: 6278
released: 6278
timeouts: 1478
total_bytes: 483270676
---
tcp
bad_tcp4_checksum: 174215
---
wizard
tcp_scans: 5433
udp_scans: 61400

Appid Statistics
---
detected apps and services
Application: Flows Clients Users Payloads Misc Incompat. Failed
unknown: 66 190 0 0 0 0 0

Summary Statistics
---
timing
runtime: 00:00:19
seconds: 19.037240
pkts/sec: 34442

Mbits/sec: 508
o")~ Snort exiting

Image showing CPU usage when small packets are transmitted from Snort 3 on the raspberry Pi.

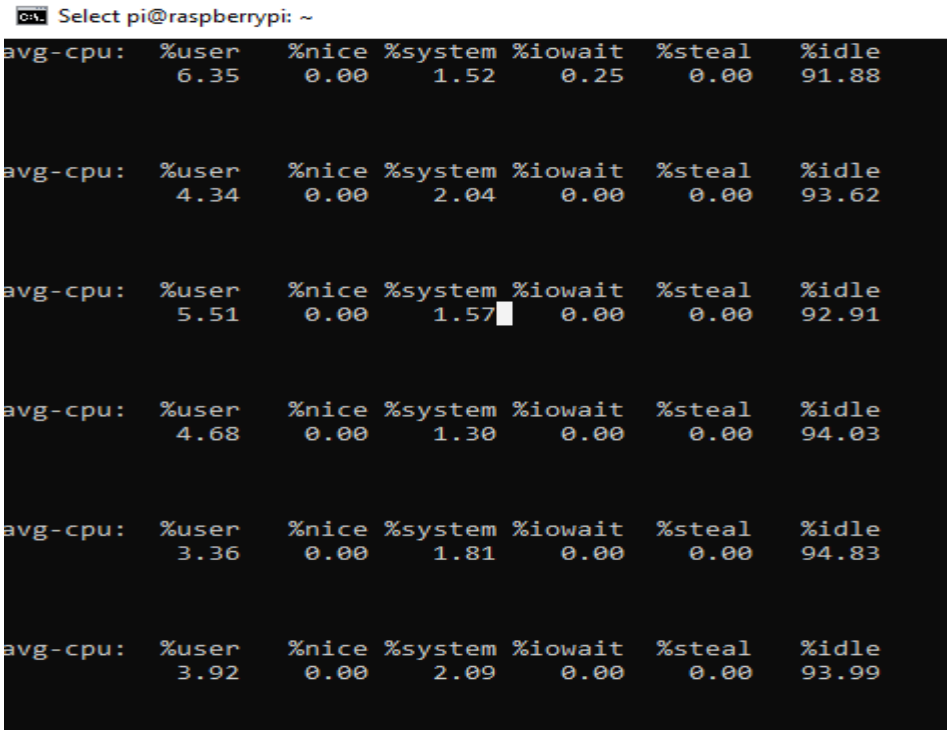


Image showing CPU usage large packets are transmitted from the computer to Raspberry Pi

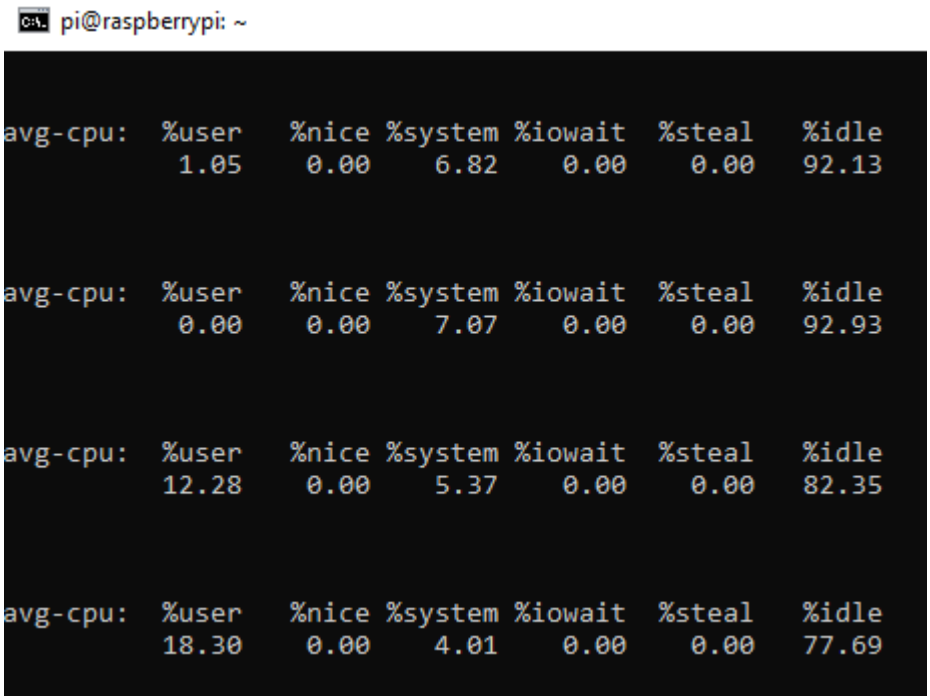


Image showing CPU usage when mixed packets are transmitted from the computer to Raspberry Pi

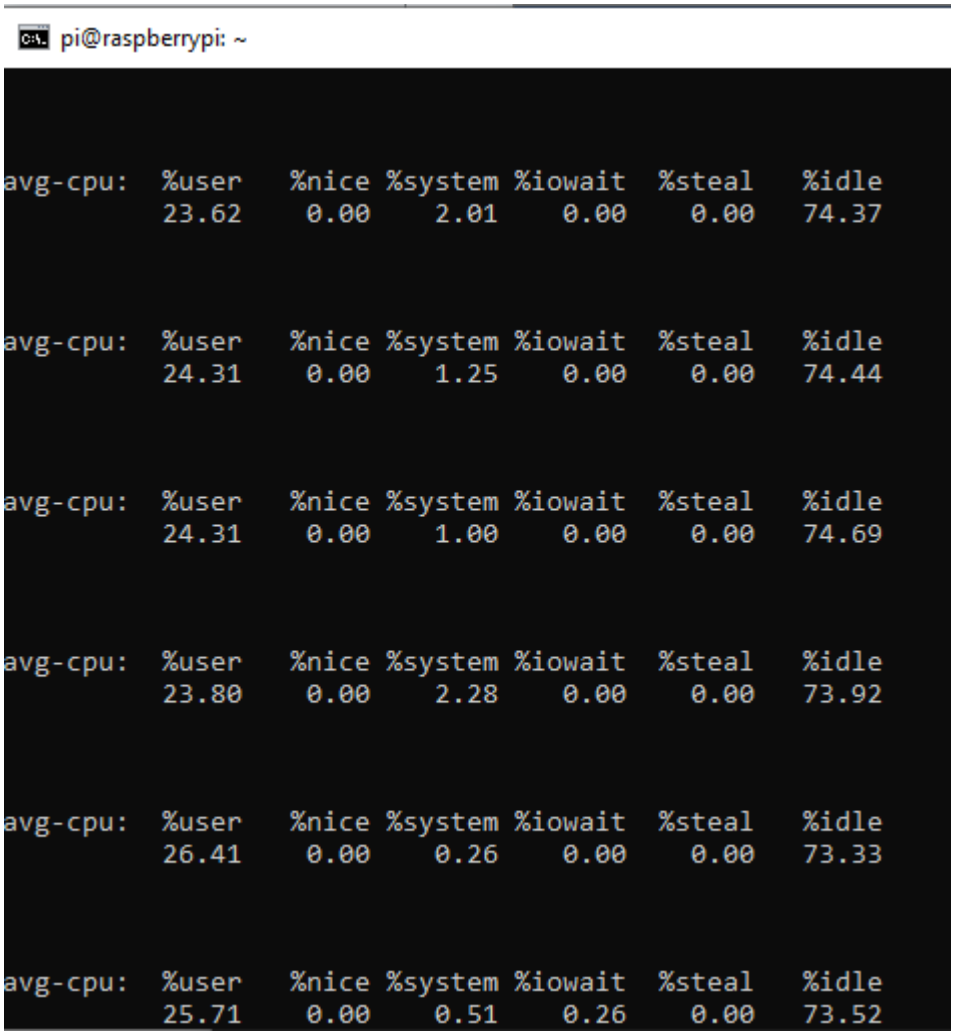
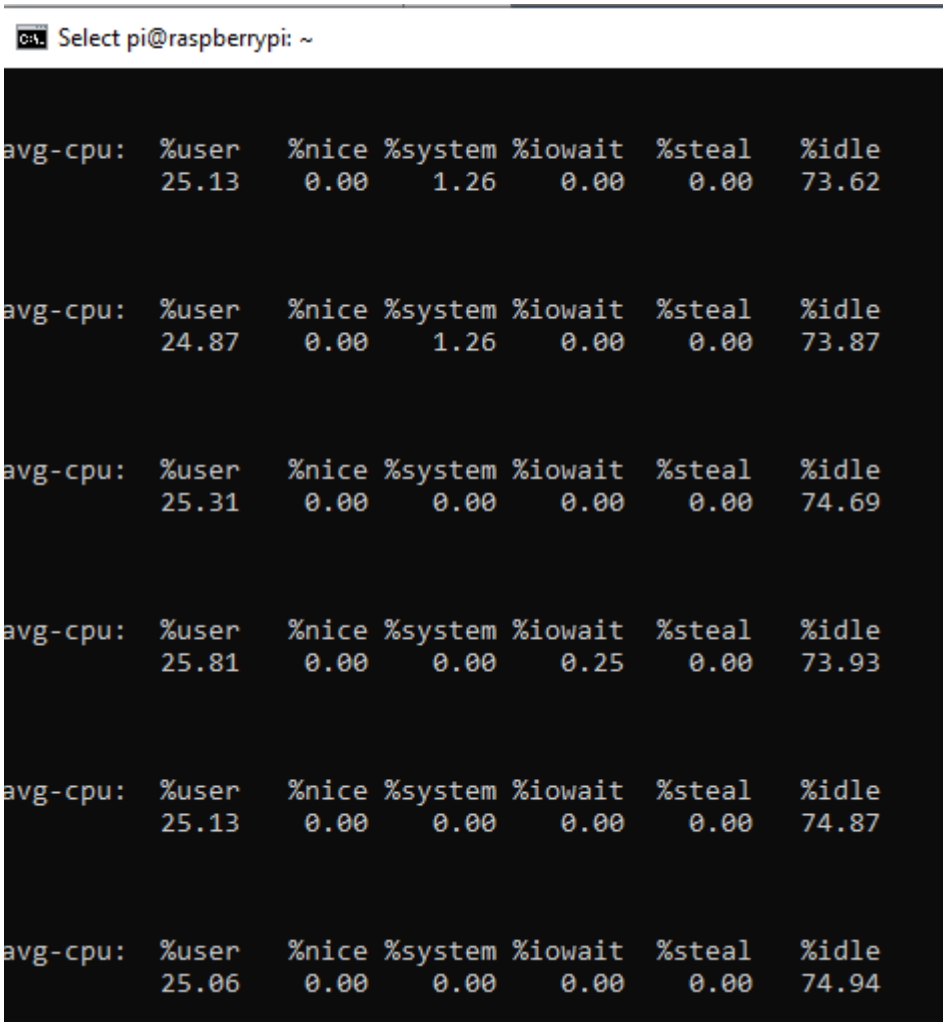


Image showing CPU consumption when rules are being loaded into Snort 3 on start-up



Snippet from csv file showing some logged alerts

	A	B	C	D	E	F	G	H	I	J
75537	12/23-02:00:43.868038	646538	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75538	12/23-02:00:43.868529	646541	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75539	12/23-02:00:43.869014	646544	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75540	12/23-02:00:43.869278	646547	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75541	12/23-02:00:43.869776	646550	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75542	12/23-02:00:43.870263	646553	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75543	12/23-02:00:43.890759	646556	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75544	12/23-02:00:43.891258	646559	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75545	12/23-02:00:43.891751	646562	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75546	12/23-02:00:43.892014	646565	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75547	12/23-02:00:43.892515	646568	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75548	12/23-02:00:43.893000	646571	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75549	12/23-02:00:43.893258	646574	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75550	12/23-02:00:43.893751	646577	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75551	12/23-02:00:43.894016	646580	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75552	12/23-02:00:43.894507	646583	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75553	12/23-02:00:43.894999	646586	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75554	12/23-02:00:43.895498	646589	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75555	12/23-02:00:43.895756	646592	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75556	12/23-02:00:43.896248	646595	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow
75557	12/23-02:00:43.896506	646598	UDP	stream_ip	4124	C2S	192.168.1.195:60840	85.190.152.140:34665	116:445:1	allow