

CYBERSECURITY AUTOMATION FOR AN INDUSTRY 4.0 GARMENT MANUFACTURING SYSTEM

A.D.H Jinadasa

(IT18133410)

B.Sc. (Hons) Degree in Information Technology Specializing in Cyber Security

Department of Computer Systems Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

October 2021

CYBERSECURITY AUTOMATION FOR AN INDUSTRY 4.0 GARMENT MANUFACTURING SYSTEM

A.D.H Jinadasa

(IT18132410)

Dissertation submitted in partial fulfilment of the requirement for the Bachelor of
Information Technology Specializing in Cyber Security

Department of Computer Systems Engineering


Sri Lanka Institute of Information Technology

Sri Lanka

October 2021

DECLARATION

“I declare that this is our own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text. Also, I hereby grant to Sri Lanka Institute of Information Technology, the nonexclusive right to reproduce and distribute my dissertation, in whole or in part in print, electronic or other medium. I retain the right to use this content in whole or part in future works (such as articles or books).”

Name	Student ID	Signature
A.D.H Jinadasa	IT18132410	

The above candidates are carrying out research for the undergraduate

Dissertation under my supervision.

Signature of the supervisor: Prof. Pradeep Abeygunawardhana Date:

Signature of the co-supervisor: Ms. Wellalage Sasini Nuwanthika Date:

ABSTRACT

Digital transformation towards Industry 4.0 and Internet of Things (IoT) focuses on productivity rather than security, therefore often face cybersecurity challenges. Even though security should be prioritized, organizations neglect security due to various reasons. To overcome network security challenges intrusion detection system was proposed. Due to high number of interconnected devices having a proper network security solution is a must. However, solutions available in the current market are high in cost and excessive for small to medium Industry 4.0 environments. This solution based on Raspberry Pi 4 running Snort open-source IDS software with following requirements in mind, ease of use, minimum configurations towards user, portability and affordability. This solution can be easily tailored in for many different occasions & requirements. Project thesis focused on installation and configuration of Snort IDS software and additional modules for added features including log management server running Elastic stack. Moreover, extensive testing was carried out in order to evaluate the performance and effectiveness of proposed solution. These test results prove the effectiveness of proposed IDS solution. We believe this IDS is ideal cost-effective solution for small to medium Industry 4.0 environments.

Keywords—*Industry 4.0, Internet of Things (IoT), cybersecurity, Intrusion Detection System (IDS), Raspberry Pi.*

ACKNOWLEDGEMENT

I wish to acknowledge the help provided by Dr. Asela Kulatunga, Head of Department of Manufacturing and industrial engineering, University of Peradeniya, Sri Lanka for giving us the opportunity to visit Peradeniya University to study CNC machine and robotic applications workflow which was a good initialization point for our research project. I wish to show my gratitude to Mr. Wijeweera, owner of Wijeweera Knit Wear (Private) Limited for letting us visit the garment factory to analyze requirements for our project in the initialization stage.

I would like to thank our project supervisor Prof. Pradeep Abeygunawardhana and co-supervisor Ms. Sasini Nuwanthika all the support and time dedicated to our research project, as well as for the direction and guidance through challenging questions and while leading us to address gaps in the research.

Last but not least, I would like to express our gratitude to our external supervisor, Chartered Eng. P.A. Gamini De Alwis, for guiding us in the right direction throughout our research project by sharing his experience in the manufacturing field, and Dr. Darshi De Saram for sharing his knowledge and experience, as well as providing valuable feedback, which helped us make our project a success.

TABLE OF CONTENTS

DECLARATION	3
ABSTRACT.....	4
ACKNOWLEDGEMENT	5
LIST OF FIGURES	8
LIST OF TABLES	9
LIST OF ABBREVIATIONS	9
1. INTRODUCTION	10
1.1 Background.....	10
1.2 Literature Review.....	12
1.2.1 CPS and IOT	12
1.2.2 Threats and vulnerabilities in CPS.....	13
1.2.3 Threats and vulnerabilities in IOT	13
1.2.4 Firewall & intrusion detection system (IDS)	14
1.2 Research Gap	17
1.3 Research Problem	17
1.3.1 Collaboration between different systems.....	18
1.3.2 Centralized security management	18
1.3.3 Secure communication.....	18
1.3.4 Insecure data	18
1.3.5 Initial cost.....	18
1.3.6 Absence of methodology to industry 4.0	18
1.4 Research Objective	19
1.4.1 Main objectives.....	19
1.4.2 specific objectives.....	19
2. METHODOLOGY	20
2.1 Planning	20
2.2 Requirement Analysis	20
2.2.1 Hardware requirements.....	21
2.2.2 Software requirements.	22
2.3 Design Phase.....	26
2.4 Implementation	27
2.4.1 Operating system installation.....	27
2.4.2 Wireless access point configurations.	28
2.4.3 Snort Installation and configuration.....	29

2.4.4 Barnyard2 installation and configuration.....	31
2.4.5 Pulledpork module installation and configuration	32
2.4.6. Configuring the Iptables firewall	33
2.4.7 Filebeat installation and configuration.....	33
2.4.8 Logstash install and configuration	34
2.4.9 ElasticSearch installation and configuration	35
2.4.10 Kibana installation and configuration	35
2.5 Testing.....	37
2.5.1 Intrusion detection evaluation dataset.....	37
2.5.2 Simulation of attacks.....	38
2.6 Commercialization Aspects	38
3. RESULTS & DISCUSSION.....	39
3.1 Results.....	39
3.1.1 CPU usage.....	39
3.1.2 RAM usage	41
3.1.3 Generated alerts.....	41
3.1.4 Real-world attack simulation	42
3.2 Research Findings.....	45
3.3 Discussion	46
4. CONCLUSION.....	47
REFERENCES	49
APPENDICES	51
Appendix A: Turnitin similarity score	51

LIST OF FIGURES

Figure 1: Industrial revolution	10
Figure 2: CPS and IOT connection	12
Figure 3: CPS challenges	13
Figure 4: IOT threats.....	14
Figure 5:Gantt chart	20
Figure 6: Raspberry Pi model 4B.....	21
Figure 7: Components of Snort.....	23
Figure 8: Elastic stack architecture	24
Figure 9: Network Diagram	26
Figure 10: Flashing the OS	27
Figure 11: DNS configurations	28
Figure 12: Hostapd configurations.....	28
Figure 13: Snort Version.....	29
Figure 14: Custom rules	30
Figure 15: Generated alerts	30
Figure 16: SSH attempt denied	30
Figure 17: Barnyard2 installation	31
Figure 18: Stored alerts in database	31
Figure 19: Stored signatures	31
Figure 20: Pulledpork configuration.....	32
Figure 21: Pulledpork installing rules.....	32
Figure 22: IPtables Firewall policy rules	33
Figure 23: Filebeat configuration.....	34
Figure 24: Input section Logstash configuration	34
Figure 25: Filter and Output section	34
Figure 26: The curl request and Elasticsearch response	35
Figure 27: Unorganized Data in Kibana	36
Figure 28: Visualize data using graphs and charts.....	36
Figure 29: Testing network diagram.....	38
Figure 30: Snort idle resource consumption	39
Figure 31: TCPReplay replaying at 20Mbps	39
Figure 32: CPU Usage of Raspberry Pi IDS.....	40
Figure 33: Packet loss and CPU usage.....	40
Figure 34: Highest recorded packet drop	40
Figure 35: RAM usage.....	41
Figure 36: Generated alert	41
Figure 37: Port scan alerts.....	42
Figure 38: LOIC DDOS simulation	42
Figure 39: Snort DDOS alerts.....	43
Figure 40: Slowloris attack response	43
Figure 41: SSH brute force attack using Medusa	44
Figure 42: SSH brute force attack alerts	44
Figure 43: Detection of MITM attacks	44

LIST OF TABLES

Table 1: Signature-based detection	15
Table 2: Anomaly-based detection	15
Table 3: Products comparison.....	17
Table 4: Containing attack packets	38
Table 5: Simulated attack findings.....	45

LIST OF ABBREVIATIONS

CPS	Cyber physical system
IoT	Internet of things
IDS	Intrusion detection system
HIDS	Host based intrusion detection system
NIDS	Network based intrusion detection system
OS	Operating system
DNS	Domain name service
SSH	Secure shell
TCP	Transmission Control Protocol
CPU	Central processing unit
RAM	Random access memory
LOIC	Low Orbit Ion Cannon
DDOS	Distributed denial of service
MITM	Man in the middle

1. INTRODUCTION

1.1 Background

Discovery of steam power was the greatest human productivity breakthrough. It initiated the first industrial revolution in 1760s. Since its inception innovations such as spinning mule, power loom was introduced. The first ever sawing machine marked the beginning of the worldwide apparel industry. As the first industrial revolution was driven by coal, water and steam the second revolved around electricity. Gas and oil. The impact of the revolution in apparel sector is the sewing machine began to be produced in a serial manner. Development of computer technologies such as in microprocessors, software, fiber optic cables, and telecommunication domains facilitate the digital revolution also referred as third industrial revolution.

In 2011 German government commenced the beginning of fourth industrial revolution. Real world and virtual environment united in a system also called as cyber physical systems (CPS). CPS are connected and communicating with each other or with humans, ultimately making decisions without human intervention. Targeted to develop new internet services and business models providing efficiency, transparency, fault detection, flexibility, monitoring and most importantly productivity while reducing costs.

Value creation and business models will emerge from the technical integration of Cyber Physical Systems and the usage of IoT in industrial processes, as well as modular architectures that adapt to rapidly changing requirements. Industry 4.0 garment manufacturing systems rely on IoT and other technologies like Cyber Physical Systems, wireless sensor networks, Machine Learning, Data analytics, augmented reality, cloud computing, 3D printing, system integration, and cyber security to build a bridge between the digital and physical worlds.

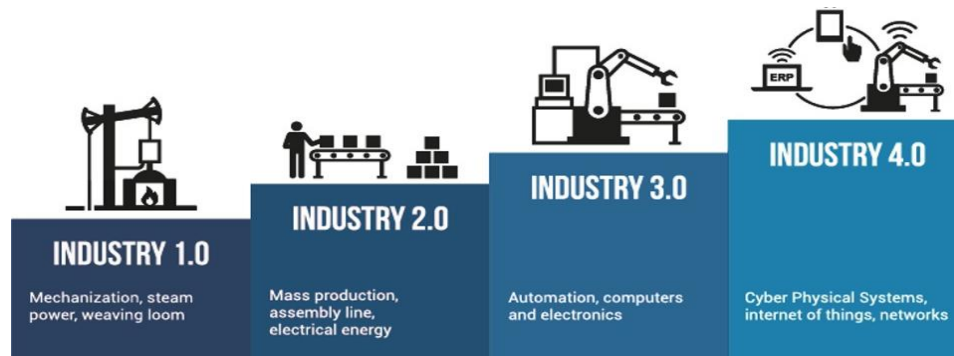


Figure 1: Industrial revolution

Novel changes in Industry 4.0 lead to a passion for manufacturing plants with productivity, customization features, flexibility, operational efficiency [1] and SAM/SMV (Standard Allowed Minute/Standard Minute Value) rather than security. The integration of complex smart manufacturing technologies lead to increased intercommunication and data density, thus massively expand the scope of attacks pointing at industrial espionage and sabotage [1], because Industrial 4.0 are implemented targeting the functionality than security [2].

CPS are used to gain higher productivity in manufacturing [3], and to usher innovation due to their potential to integrate technology from different sectors for the implementation of real world processes [1][2]. Manufacturing automation is becoming a part of critical infrastructure in the present and future context. CPS is designed and implemented to be easily extendable and scalable as the structure includes heterogeneous communication technologies. The integration of IoT devices combined with various technologies is a complex task and implementing security for those systems is more complex task. Therefore, cybersecurity has evolved into a major concern.

Objective is to design and implement an automated system for garment manufacturing system focusing on four major cyber security aspects for garment manufacturing systems including:

- Frameworks and standardization
- Centralized security configurations with update management
- Authentication and Physical Access Control
- Intrusion detection

A comprehensive, cost effective and efficient security solution is proposed with a centralized security approach to overcome the potential challenges of the smart system. This thesis will solely be focusing on network security and intrusion detection.

1.2 Literature Review

Industrial revolutions are key turning points in humanity's history. Following the discovery of steam power, revelations have accelerated, and the fourth industrial revolution is already underway. Manufacturing plants are very interested in new changes in Industry 4.0. It offers a method for dealing with large amounts of data, as well as the development of cyber physical systems (CPS) and improved communication between digital and physical components. [1]

From security point of view, with the vast growth of interconnected devices and the data density produces whole heap of new challenges. Cyber security should become a major objective that requires the highest level of attention. With the advancement of network capabilities, cyber-attacks have become more frequent. Motivated by financial & strategic reasons. Almost every stakeholder who uses IoT technologies is impacted by this issue. Large enterprises are frequently targeted by cyber-attacks, which result in major financial losses as well as many other losses such as data corruption, system crashes, privacy breaches, prestige, trust, and reputation. [2].

1.2.1 CPS and IOT

CPS and IoT shares the same core architecture. The cyber-physical system is presented a high combination and coordination between physical components and computational components on IoT.

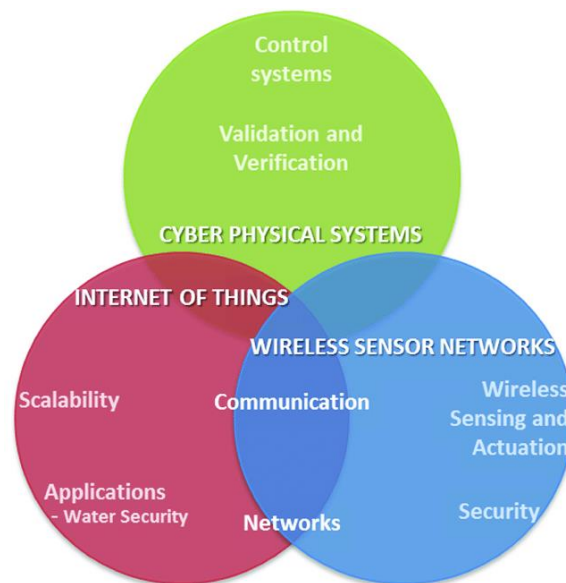


Figure 2: CPS and IOT connection

1.2.2 Threats and vulnerabilities in CPS

Out of the challenges mentioned below figure 3. Security can be categorized as a major challenge that often neglected or go unnoticed. Security challenges can be further divided into data security and control security. Data security focuses on protecting data at rest and data at storage. A unique characteristic of cyber physical threats are they mostly originate in cyberspace but impact on the physical system.

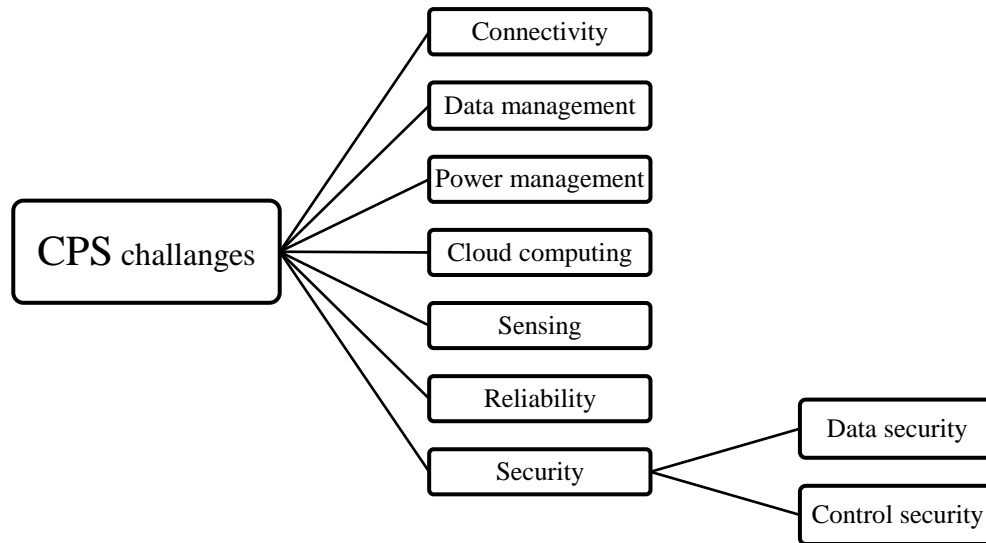


Figure 3: CPS challenges

Security policy violation can be defined as a vulnerability. These vulnerabilities can occur due to lack of security rules, weak system design. Vulnerabilities can be found in many different levels such as in hardware level software level design and policy level and even in user level. Hardware vulnerabilities, software vulnerabilities, network vulnerabilities, platform vulnerabilities, and management vulnerabilities are all well-known categories of cyber-physical system vulnerabilities. CPS threats are classified as denial of service (DOS), spoofing, tampering, disclosure, and repudiation.

1.2.3 Threats and vulnerabilities in IOT

IoT devices spread in an exponential rate. General IoT is a combination of four levels as shown in figure 4. The number of threats towards IoT growing every day. this created the need for a reliable protection. IoT to achieve its full potential, protection against threats and vulnerabilities is a necessary. [2].

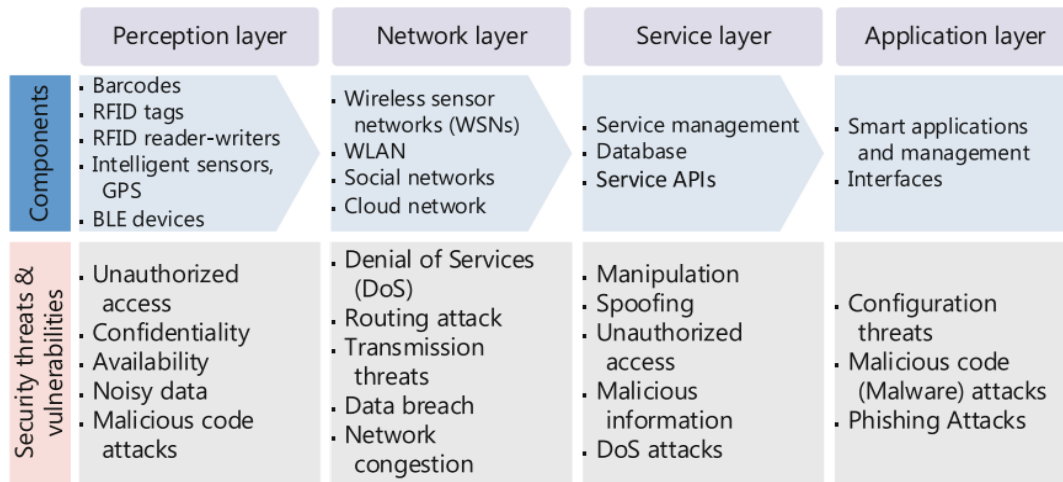


Figure 4: IOT threats

1.2.4 Firewall & intrusion detection system (IDS)

CPS components and the “things” in IOT such as sensors, smart devices are commonly having less processing power and have the ability to exchange data via the internet. Even though benefits outweigh the disadvantages, Industry 4.0 security and privacy challenges cannot be ignored. Even with proper security configurations it is important to prevent intruders from accessing the CPS components via the network. There are various methods to provide security against network intruders [4].

- Firewall.

One of these measures is utilizing a firewall. It protects devices within a network against intruders and controls the network traffic. According to predefined rule set, firewall decides whether or not the packets that arrive on the network can pass through.

- Intrusion detection system.

IDS is a software or hardware tool to detect and prevent incoming and outgoing malicious packets in a network. It works by recognizing the signs of a possible attack and sending an alert regarding the malicious packet. In some instances, it can trigger a response to combat the threat [5].

There are two different types of intrusion detection systems. First type is host-based IDS (HIDS), which installed in host machines and monitors one host at a time, HIDSs are lighter and use less processing power. The second type is network-based IDS (NIDS), which have the ability to

monitors the entire network for malicious activities, NIDSs are more accurate at the cost of extra usage of system resources.

There are two different detection mechanisms, they signature-based and anomaly-based detection, some occasions these two methods are combined for more reliable security [6]. Signature-based systems has predetermined set of rules and compares with the patterns in the traffic to detect attacks similar to any signature rules that has been stored in a database.

Table 1: Signature-based detection

<i>Signature-based detection</i>	
Pro	Con
High accuracy (lower rate of false positive)	Require database updates
Easy to configure and maintain	Unable to counter Zero-day attacks [14]

Anomaly-based detection systems first gather knowledge of normal traffic activities and then alerting user if there is any malicious activity occurs.

Table 2: Anomaly-based detection

<i>Anomaly-based detection</i>	
Pro	Con
Capable of detecting Zero-day attacks	High amount of false positive & false negative
Better protection against DDOS attacks	Required to define threat profiles

Whitepaper written by Michael Brennan, SANS Institute explains organizations with little to no network monitoring, opting for an expensive solution is generally not plausible. This paper focuses on the need for an IDS for companies that cannot afford one of the more quality commercial Intrusion Detection Systems available in the market. In addition, paper focuses on the IDS placement within the network “In order for Snort to be most effective, it needs to be positioned where it will see the most traffic possible.” [7]

Many research projects have been done to evaluate and compare the performance of opensource intrusion detection systems.

One interesting study shows IDS's detection techniques such as deep packet inspection are too are too much resource consuming for a Wireless Mesh Network (WMN), resulting unstable nodes. In order to address this issue authors proposed an IDS solution that was lightweight and less resource consuming. This solution however, able to provide protection only for few common attack types.

Another similar [8] study also points out the infeasibility of deploying IDS in WMNs. Both studies came to the same results and conclusion. using the full capabilities of Snort on WMNs nodes is impractical. This study proposed PRIDE (Practical Intrusion Detection in resource constrained wireless mesh network) solution which have the ability to distribute IDS functionalities to WMNs across the network, each node runs a custom version of Snort IDS with different set of rules: different nodes will be responsible for detecting different types of attack. This IDS function distribution was optimized to provide coverage to entire network.

Even though many research projects have been done to evaluate open-source Intrusion Detection Systems, but only handful of these emphasis of assessing its impact on resource constrained single board computers such as raspberry pi.

One interesting work is [9] which the authors evaluate the capability of raspberry pi 2 of running IDS and the performance of Snort and Bro (Zeek) IDS software. As for the result of these tests Raspberry pi 2 was able to run IDS and detect attacks and handled considerable number of packets, and they demonstrate that Snort has better performance than Bro IDS. However, these tests were based on older raspberry pi version 2 but still showed the potential of single board computers.

Another similar study [10] discuss the feasibility of utilizing Raspberry Pi while running Snort IDS in a distributed system. Authors specifically evaluate Raspberry Pi performance, packet drops and steady Snort configurations, and they claim Snort resource consumption did not overwhelm raspberry pi apart from high CPU usage when filtering high sets of packets.

However, these tests were done with old hardware but still showed the potential of single board computers. This thesis is focus on integrating raspberry pi 4b as a IDPS sensor to a medium size network that contains IOT devises, and evaluate resource usage and effectiveness by performing various type of network attacks.

1.2 Research Gap

Industry 4.0 creates a massive hunting ground for malware, botnets and other malicious activity, as the threats towards CPS and IoT continue to rise. In order to combat these network threats security systems such as IDS, Firewalls must also evolve. Even though highly capable network security solutions are available for cooperate networks shown in table 3, these solutions are expensive and unattainable and excessive for small to medium industries who are migrating into industry 4.0. This thesis is focus on designing a cost effective, lightweight yet reliable IDS/IPS system with a dashboard visualize alert logs. This IDS solution is small, portable, pre-packaged with Snort IDS and have the ability to deploy anywhere seamlessly [11].

Table 3: Products comparison

Product	HIDS/NIDS	Signature based	Anomaly based	Base Price
McAfee NSP	NIDS	✓	✓	\$ 10,995
Cisco Firepower	HIDS	✓	✓	\$ 100,000
CrowdStrike	HIDS	✓	✓	\$ 4,000
Tipping Point	HIDS	✓	X	\$ 6,000
Proposed Solution	NIDS	✓	✓	Less than \$ 200

Source: www.esecurityplanet.com

1.3 Research Problem

When manual systems migrating towards industry 4.0 systems, technologies such as IoT, machine learning, cognitive computing will be utilized. Many of the system designers and developers are failed to recognize cyber security challenges. This research focused on identifying cyber security requirements which are not been identified before and providing a comprehensive security solution for industry 4.0.

The main research problem is **“How can we develop and implement an intrusion detection system in the industry 4.0 environment that is accurate and efficient as well as easy to operate by an average user, while overcoming below mentioned challenges.”**

- Challenges:

Developing a secure Industry 4.0 system has become a strenuous process due to challenges listed below.

1.3.1 Collaboration between different systems

A Collaborative model between computer systems and physical devices is essential for exchanging information, [2] store information, documentation, decision making, corrective and preventive action.

1.3.2 Centralized security management

Using a centralized control system such as Supervisory control and data acquisition (SCADA) to apply security configurations/updates to physical devices and monitor physical devices to maximize efficiency [3]. In addition to the CPS modeling language, software, physical device environments, hardware platforms, and other functional and non-functional factors must be included in a typical CPS model [2].

1.3.3 Secure communication

CPS that uses the SCADA systems is connected to the internet over TCP/IP protocols without additional protection [12], which have known vulnerabilities.

1.3.4 Insecure data

Manufacturing companies neglect cyber security when moving towards Industry 4.0. The IoT-based CPSs that are connected to many of embedded sensors and communication devices present a significant risk linked with the growth of data usage and the much higher risks of system breaches [13].

1.3.5 Initial cost

Migration to industry 4.0 is not an instantaneous process, in a manufacturing company will result in end-to-end integration to plan and execute the engineering according to the business needs impressive introductory interest in the matter of cost and time is required [14].

1.3.6 Absence of methodology to industry 4.0

Absence of dynamic vital arrangement to help the movement to Industry 4.0.

1.4 Research Objective

1.4.1 Main objectives

This research component targeted towards implementing a lightweight, reliable accurate, and easy to implement and operate IDS/IPS system targeted towards small to medium industry 4.0 environments. In order to monitor the network for anomalies, malicious activities, policy violations and alert the user. This IDS will be implemented with following requirements in mind

Portability: the device should be portable enough to carry with users, must be able to relocate to another location without an effort.

Minimum configuration: the device should be pre-configured. However, user have full power to add or remove configuration to suit their needs.

Ease of use: user will have the ability to deploy the IDS seamlessly without any complications

Versatility: the device could be used anywhere from home to medium size industrial environments.

Affordable: the device initial and operational cost should not exceed 1000 USD per year.

1.4.2 specific objectives

- Provide easy access dashboard to the user.
- Visualize network behavior to user.
- Automate the IDS rule update process.
- Enable firewall rules as an extra layer of protection.
- Alert user when an anomaly occurs.

2. METHODOLOGY

In order to scientifically answer above research question, IDS was implemented using a readily available single board computer. At the core IDS sensor was based on Raspberry Pi 4B 4GB RAM version with Snort IDS software. The phases of this project are as follows; Planning, Requirement Analysis, Design, Implementation and Testing. The implemented IDS will evaluate by measuring resource usage and alert count when exposing network to various sizes of bandwidth loads and launching different types of network attacks against connected devices.

2.1 Planning

Planning is the primary step of carrying out this project. As the first step the main project idea was discussed with the supervisor and co-supervisor. The topic assessment document including individual sub-components was submitted to the panel. Once the project was approved, the next phase of planning initiated, it includes information gathering, creating Gantt chart and work breakdown structure. The planning phase gives a clear imagination of the end goal and a better understanding of how IDS internal process.



Figure 5:Gantt chart

2.2 Requirement Analysis

Extensive studies were carried out in order to identify requirements. Initially, outlined the hardware requirements of the base device; second, selected the IDS software and assessed the compatibility with the base device. At this stage required key components for the IDS had been identified. Required components can be divided into two categories, which is software and hardware components.

2.2.1 Hardware requirements.

- Raspberry pi 4 model B

Raspberry Pi is a series of small single board computer developed in United Kingdom by Raspberry Pi Foundation. Raspberry Pi is compatible with several different OS such as Raspberry OS, Ubuntu, Arch Linux, Windows 10 IOT and Android Things. Versatility and cost effectiveness makes it a popular choice among developer community.



Figure 6: Raspberry Pi model 4B

The Raspberry Pi 4 Model B as shown in figure 2 is their latest iteration similar in size to a credit card (85x56x17mm) but still contains powerful hardware. It has built-in gigabit Ethernet, 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless connectivity, additionally two USB 3.0 ports and two micro-HDMI ports, which makes this perfect candidate for this project [15]. We selected the variant with 4GB of RAM, owing to the fact it is adequate for Snort IDS. Following is the specification List of Raspberry Pi 4B.

- Broadcom BCM2711, Quad core Cortex-A72 (ARM v8) 64-bit SoC @ 1.5GHz
- 4GB LPDDR4-3200 SDRAM
- Broadcom VideoCore VI GPU
- 2.4 GHz and 5.0 GHz IEEE 802.11ac wireless, Bluetooth 5.0, BLE
- Gigabit Ethernet
- 2 USB 3.0 ports; 2 USB 2.0 ports.

- Management Server

All IDS sensors will collect alert logs locally and send them to a central management server where administrators can search, visualize, and analyze alerts in real time. This can be achieved by using a local server or a cloud-based server such as Amazon or any alternative. For testing purposes, we used a local server running Ubuntu-20.04.1-server operating system with following hardware configurations, 2.6 GHz Intel Core i7 6500U, 8 GB of memory with 200 GB disk space for log storage. In real world scenario the best choice is to opting a cloud hosted server. However, these are the advantages and dis advantages in both options.

- Other hardware requirements.
 - USB type-C Power supply
 - Storage 16 GB MicroSD card or higher SSD or HDD can be used for better reliability.
 - Ethernet Cables
 - Wi-Fi adaptor (optional)
 - USB Keyboard and Mouse (optional)
 - Micro-HDMI cable (optional)
 - Attacker computer running Kali Linux

2.2.2 Software requirements.

- Snort IDS

Snort IDPS is an open-source software maintained under GPL license, developed by Martin Roesch in 1998 who also the founder of the company Sourcefire, now operating under Cisco Systems [16]. Snort is a user-friendly software that have the capability to provide real-time network protection. It is capable of monitoring and analyzing network packets in a one single host or the entire network. Snort achieve real time traffic analysis and logging on network by using many different techniques. Ranging from simple packet sniffing and logging to TCP session tear down to mitigate attacks. It also has some function to block packets or devices from accessing the network The latest version of snort IDS is Snort 3.0.

Snort IDS supports windows and Unix based operating systems, even on low power systems such as Raspberry pi. There is plethora of add-ons for Snort IDS to improve its features. Some commercial products of Cisco are run on upgraded version on snort, as for example isr4000 and Asr1000 series routers which leverages the Linux based IOS XE operating system

Snort signatures are consisting of attack patters collected from previous attacks and use by snort to identify future attacks. These patterns are stored in a memory of snort application. However, this method only suitable for detecting existing attack patterns.

Unlike signatures, rules do not need previous attack pattern to detect future attacks, it relies on deep packet analyzation of packet content. Rules can be set by the user or available to download in three different tiers. 1st tire is community rules, it is available for free under GPLv2 license. 2nd tire is the registered rules, this ruleset is also freely available in order to download this rules user has to register on Snot official website. 3rd tier is the subscriber ruleset is available for paid users, Snort provide these rules without delay it includes rules to counter latest zero-day threats.

The major drawback of Snort is its single-threaded design, utilize only one CPU core even in multicore environment. Nevertheless, “if the bandwidth being passed by the network interface associated with a Snort instance is greater than it can handle, more instances of Snort can be launched, and the traffic can be load balanced across the instances” [17].

Following figure shows the major components of Snort IDS

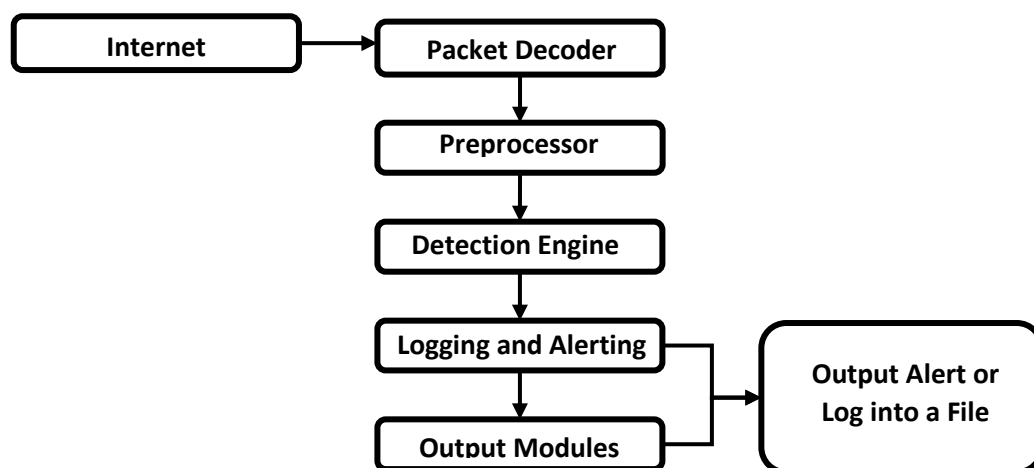


Figure 7: Components of Snort

- Raspberry OS

Raspberry OS previously known as Raspbian is a Debian based Linux operating system specifically designed for Raspberry pi devices. Initially Raspberry OS was started developing as a separate project by Mike Thompson and Peter Green. Alternatives for Raspberry OS are Ubuntu ARM OS and Arch Linux. Since there are no compatibility issues between Raspberry OS and Snort, we installed Raspberry OS.

- Barnyard2 Module

Barnyard2 is an open-source interpreter for Snort IDS unified2 binary output file format. It assists Snort by parsing binary data into various data formats allows Snort to perform more efficient manner. This also have the ability to send processed data to a database. When snort receives large number of packets this will cause packet drops, Barnyard2 module can help reduce the work load from Snort process. In this project Barnyard2 module is utilized to parse and write alert logs into MySQL Database.

- Pulledpork

In order to automate the Snort rule update process, we utilized Pulledpork module. This module is written in Perl language, it can identify the installed snort version and automatically download and install latest rule set. In this project Pulledpork process was scheduled using a Cronjob.

- Elastic stack

Elastic stack previously known as ELK stack, is a collection of open-source software products help users to centralize logging in any format and analyze, search, visualize in real-time. Elastic stack contains Elasticsearch, Logstash, Kibana and FileBeats. Elastic stack is developed, managed, and maintained by the company named Elastic. Below figure shows the elastic stack architecture according to this project.

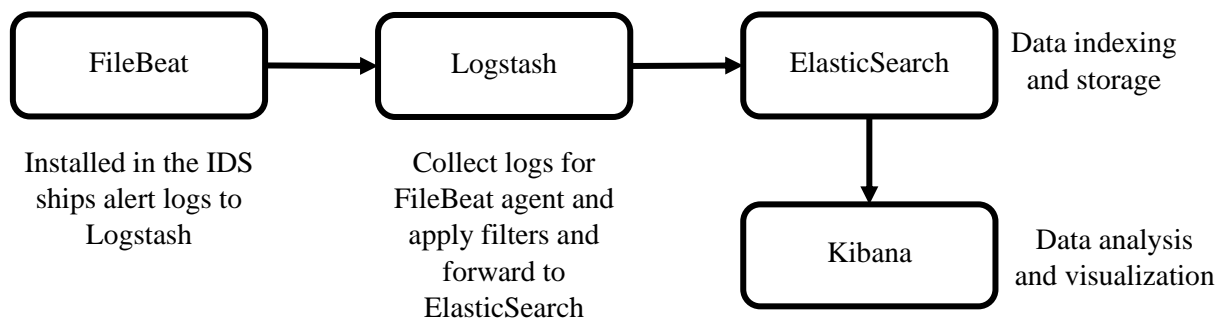


Figure 8: Elastic stack architecture

Elasticsearch is based on Lucene search engine, it is a NoSQL database built with RESTful API. Effortless deployment, high reliability and easy to use nature it its key characteristics [18]. It also offers advanced queries to perform detail analysis and stores all the data centrally. Elasticsearch also allows you to store, search and analyze big volume of data. It is mostly used as the underlying engine to powers applications that completed search requirements.

Logstash is a server-side data processing pipeline tool that collects data logs from the sources, transform and send them into Elasticsearch [19].

It consists of three components:

- **Input:** passing logs to process them into machine understandable format
- **Filters:** It is a set of conditions to perform a particular action or event
- **Output:** Decision maker for processed event or log

FileBeat is a light weight data shipper installed on hosts for shipping different kinds of data into the Logstash for filtering and analysis [20]. Filebeat acting as a logging agent installed on the host generating log files in this case it is the IDS. Filebeat cannot replace Logstash but can and should in most cases, be used in sequentially.

Kibana is playing data visualization task in elastic stack. It used for visualize ElasticSearch data and allow users to have a better understanding of log data [21]. Kibana dashboard offers various interactive diagrams, geospatial data, and graphs to visualize complex quires. It can be used for search, view, and interact with log data stored in Elasticsearch directories. Kibana helps users to execute advanced data analysis and visualize data in a variety of tables, charts, and maps.

2.3 Design Phase

After all requirements gathered, design phase process begins. At this stage a network diagram was created as a guideline. This diagram gives a better understanding of the network flow.

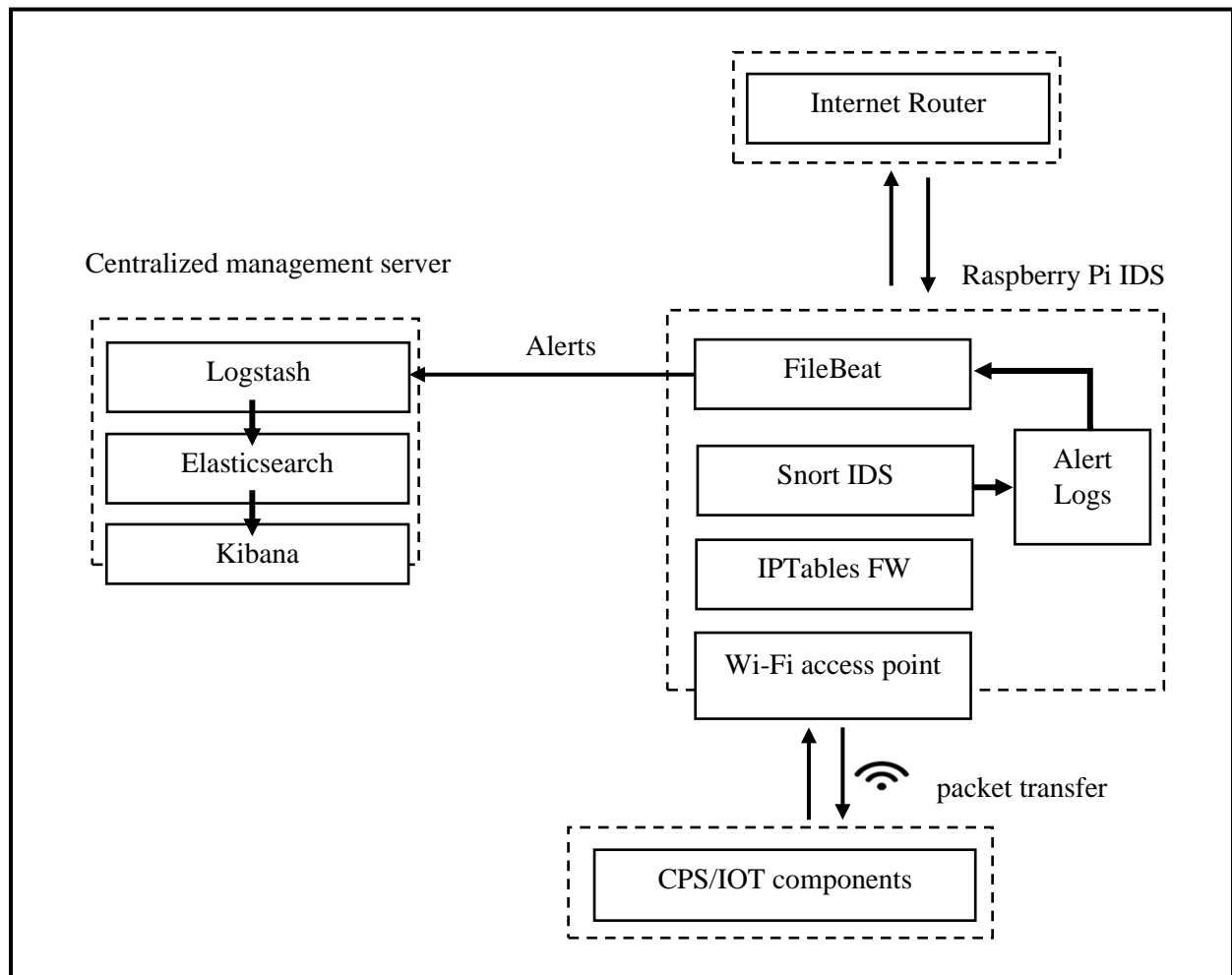


Figure 9: Network Diagram

As shown in above diagram IOT components are allow to directly connect to Raspberry PI via a wireless access point and reach to the internet, Snort IDS will perform real-time monitoring in the network. If there are any malicious packets present IDS will act according to the rules and alerts will be generated and forwarded to log management server. As an additional security layer IPTables firewall was configured. If there are any alert logs created, FileBeats will securely ship the logs to Logstash, it will apply filters and send processed data to Elasticsearch. Those logs can be access via the Kibana web dashboard.

2.4 Implementation

In implementation phase in where all the gathered knowledge and components will be utilized. Adhering to mentioned goals and objectives is crucial. All the procedures and steps mentioned previous chapters will be implemented. However, if any changes need to be made, it will be done in this phase.

2.4.1 Operating system installation

The initial task was to install OS in raspberry pi. The choice of OS was Raspberry OS. For testing purposes, we utilized a 32 GB MicroSD card. Hard disk drive can be utilized but first required to update Raspberry Pi 4 firmware.

Flashing the MicroSD card is a straightforward task, a software tool called Etcher was utilized. Raspberry OS was directly downloaded from the official website.

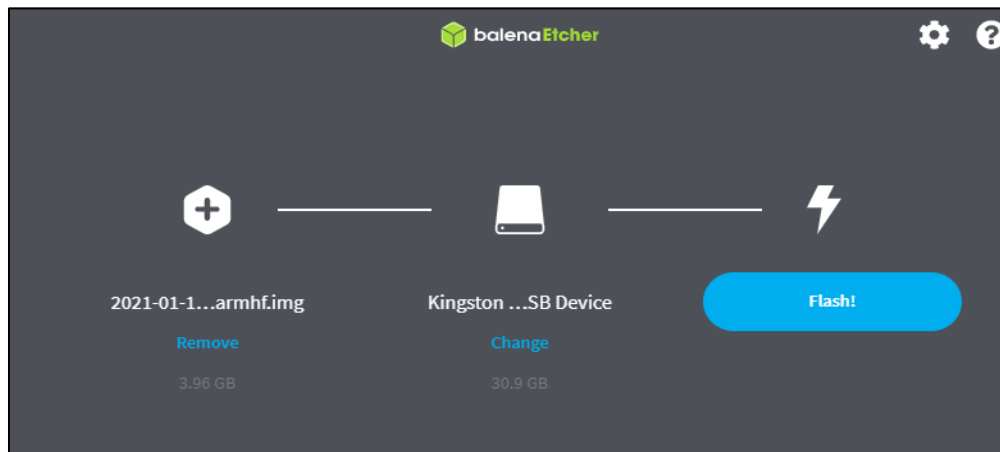


Figure 10: Flashing the OS

After flashing the SD card create an empty file named “ssh” inside the boot partition in order to enable login via SSH. On first boot the system see this file and enable SSH, deletes the file and then automatically resizes the filesystem to adapt to the size of the SD-card.

Windows command prompt or Putty can be used for SSH login. The default username is “pi” and the default password is “raspberrypi”. We recommend to change this immediately for security concerns. Below command can be use to update and upgrade OS to latest version.

```
sudo apt-get update && sudo apt-get dist-upgrade -y
```

2.4.2 Wireless access point configurations.

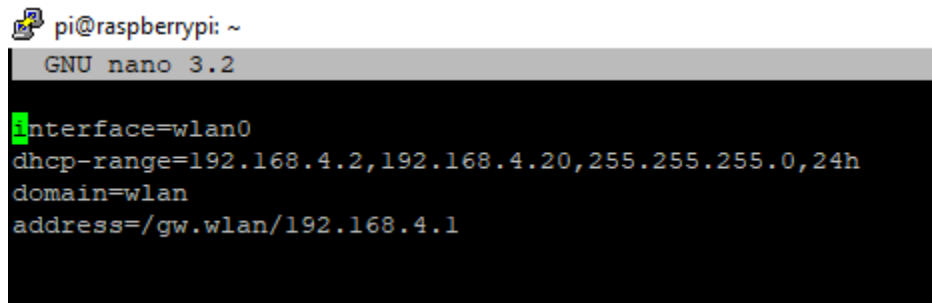
As shown in network diagram, host devices are allowed to connect to Raspberry Pi IDS to reach the internet. In order to achieve this Raspberry Pi required to connect to the internet via ethernet. Following configurations were done afterwards.

Hostapd and Dnsmasq was use in order to create an access point. Hostapd is a software that allows wireless interface card to act as a wireless access point. Dnsmasq is a lightweight software that provide DNS caching, DNS forwarding and DHCP services.

Afterwards, following firewall rule was added in order to masquerade clients connected to Raspberry Pi. A full detailed instruction is found in (Appendix B)

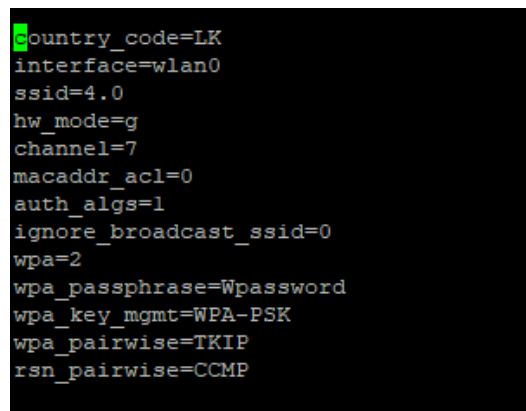
```
sudo iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Below figures 11, 12 shows DNS configurations and the Hostapd configurations.



```
pi@raspberrypi: ~  
GNU nano 3.2  
interface=wlan0  
dhcp-range=192.168.4.2,192.168.4.20,255.255.255.0,24h  
domain=wlan  
address=/gw.wlan/192.168.4.1
```

Figure 11: DNS configurations



```
country_code=LK  
interface=wlan0  
ssid=4.0  
hw_mode=g  
channel=7  
macaddr_acl=0  
auth_algs=1  
ignore_broadcast_ssid=0  
wpa=2  
wpa_passphrase=Wpassword  
wpa_key_mgmt=WPA-PSK  
wpa_pairwise=TKIP  
rsn_pairwise=CCMP
```

Figure 12: Hostapd configurations

2.4.3 Snort Installation and configuration

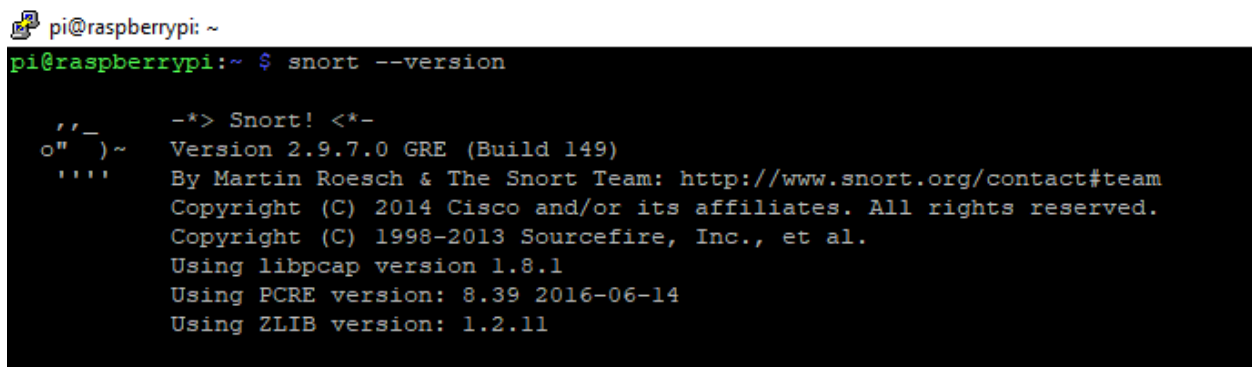
Initially there are three plugins needs to install as prerequisite plugins for Snort IDS to operate properly. Those plugins are pcap (Packet Capture), pcre (Perl Compatible Regular Expressions) and libdnet (a low-level portable interface for networking routines)

After installing these plugins Snort DAQ ((Data Acquisition Library) was source code was downloaded, compiled and installed, which was a necessary tool for packet monitoring.

Finally, all the pre requirements are installed, then Snort source code can be downloaded. The latest source codes are available in Snort official website. In order to compile and install use below command

```
./configure && make && sudo make install
```

By default, Snort will be installed in directory /etc/ so the directory. A full detailed instruction is found in (Appendix C)

A terminal window on a Raspberry Pi showing the command 'snort --version' and its output. The output displays the Snort version (2.9.7.0 GRE, Build 149), credits to Martin Roesch and The Snort Team, copyright information (2014 Cisco and/or its affiliates, 1998-2013 Sourcefire, Inc.), and the versions of dependencies: libpcap 1.8.1, PCRE 8.39 (2016-06-14), and ZLIB 1.2.11.

```
pi@raspberrypi: ~  
pi@raspberrypi:~ $ snort --version  
  
  _ _ _ _ _  
  o" )~  
  ' ' ' ' '  -*> Snort! <*-  
Version 2.9.7.0 GRE (Build 149)  
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team  
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.  
Copyright (C) 1998-2013 Sourcefire, Inc., et al.  
Using libpcap version 1.8.1  
Using PCRE version: 8.39 2016-06-14  
Using ZLIB version: 1.2.11
```

Figure 13: Snort Version

- Custom Snort rules

Snort registered rule set were used in this project. However, in addition to those rules user can write their own rules in Snort. By default, these rules were stored in “/etc/snort/rules/local.rules” file. The rules are specified based on protocol layer, packet type, priority, source and destination. User can refer to “/etc/snort/classification.config” file to get a better understanding of priority types.

As for demonstration purposes we created two custom rules figure 14, first rule is to drop SSH connections from the host 192.168.4.13. Second rule is to prevent attacks from the inside, by dropping outbound SSH traffic. Third rule is just there for testing, it alerts to ICMP ping requests.

```
# $Id: local.rules,v 1.11 2004/07/23 20:15:44 bmc Exp $
# -----
# LOCAL RULES
# -----
# This file intentionally does not come with signatures.  Put your local
# additions here.

drop tcp 192.168.4.13 any -> 192.168.4.1 22 (msg:"SSH Deny"; sid: 4000006; GID: 10006)
drop tcp 192.168.4.1 any -> any 22 (msg:"SSH Outbound deny"; GID: 10008; sid: 4000008; classtype:attempted-user; priority: 1)

alert icmp any any -> $HOME_NET any (msg:"ICMP test"; sid:10000001; rev:001;)
```

Figure 14: Custom rules

```
Rules Engine: SF_SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_FTPTELNET Version 1.2 <Build 13>
Preprocessor Object: SF_SDF Version 1.1 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_POP Version 1.0 <Build 1>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Commencing packet processing (pid=5262)
Decoding Ethernet

10/03-07:23:52.620413  [**] [1:10000001:1] ICMP test [**] [Priority: 0] {IPV6-ICMP} fe80::16
91:82ff:febe:5e4e -> ff02::1
10/03-07:23:56.045143  [Drop] [**] [10008:40000008:0] SSH Outbound deny [**] [Classification
: Attempted User Privilege Gain] [Priority: 1] {TCP} 192.168.4.1:46932 -> 192.168.4.9:22
10/03-07:24:07.514620  [Drop] [**] [10006:4000006:0] SSH Deny [**] [Priority: 0] {TCP} 192.1
68.4.13:64773 -> 192.168.4.1:22
```

Figure 15: Generated alerts

```
C:\Users\Admin>ssh pi@192.168.4.1
pi@192.168.4.1's password:
Connection reset by 192.168.4.1 port 22

C:\Users\Admin>
```

Figure 16: SSH attempt denied

2.4.4 Barnyard2 installation and configuration

After verifying Snort configurations, we proceed to install Barnyard2 module. This module is used to convert Snort unified2 binary files into human readable format. Same as before this module is also installed from the source code compilation. As for the configurations Barnyard2 was configured to read unified2 log files and write them into a MySQL database. Figure 17 shows Barnyard2 version, Figure 18 and 19 shows stored data in database.

```
pi@raspberrypi:~ $ /usr/local/bin/barnyard2 -v

      _ _ _ _ _
     /  _  _  \
    |o"  _ )~|
    + ' ' ' +

-*> Barnyard2 <*-
Version 2.1.14 (Build 337)
By Ian Firms (SecurixLive): http://www.securixlive.com/
(C) Copyright 2008-2013 Ian Firms <firmsy@securixlive.com>
```

Figure 17: Barnyard2 installation

```
MariaDB [snort]> select * from event;
+-----+-----+-----+-----+
| sid | cid | signature | timestamp |
+-----+-----+-----+-----+
| 1 | 1 | 507 | 2021-06-11 06:57:55 |
| 1 | 2 | 508 | 2021-06-11 06:57:55 |
| 1 | 3 | 509 | 2021-06-11 06:57:55 |
| 1 | 4 | 507 | 2021-06-11 06:57:56 |
| 1 | 5 | 508 | 2021-06-11 06:57:56 |
| 1 | 6 | 509 | 2021-06-11 06:57:56 |
| 1 | 7 | 507 | 2021-06-11 06:57:57 |
| 1 | 8 | 508 | 2021-06-11 06:57:57 |
| 1 | 9 | 509 | 2021-06-11 06:57:57 |
| 1 | 10 | 507 | 2021-06-11 06:57:58 |
| 1 | 11 | 508 | 2021-06-11 06:57:58 |
```

Figure 18: Stored alerts in database

```
MariaDB [snort]> use snort
Database changed
MariaDB [snort]> select * from signature limit 30;
+-----+-----+-----+-----+-----+-----+
| sig_id | sig_name | sig_class_id | sig_priority | sig_rev | sig_sid | sig_gid |
+-----+-----+-----+-----+-----+-----+
| 1 | dnp3: DNP3 Application-Layer Fragment uses a reserved function code. | 0 | 0 | 0 | 6 | 145 |
| 2 | dnp3: DNP3 Link-Layer Frame uses a reserved address. | 0 | 0 | 0 | 5 | 145 |
| 3 | dnp3: DNP3 Reassembly Buffer was cleared without reassembling a complete message. | 0 | 0 | 0 | 4 | 145 |
| 4 | dnp3: DNP3 Transport-Layer Segment was dropped during reassembly. | 0 | 0 | 0 | 3 | 145 |
| 5 | dnp3: DNP3 Link-Layer Frame was dropped. | 0 | 0 | 0 | 2 | 145 |
| 6 | dnp3: DNP3 Link-Layer Frame contains bad CRC. | 0 | 0 | 0 | 1 | 145 |
| 7 | modbus: Reserved Modbus function code in use. | 0 | 0 | 0 | 3 | 144 |
```

Figure 19: Stored signatures

2.4.5 Pulledpork module installation and configuration

In order to automate ruleset update process, we utilized Pulledpork module. It is written in Perl language. As done in previous installations this also can be downloaded as the source code.

Before configuration, user needs to register in Snort official website to obtain a register key known as “oinkcode”. Afterwards place the key inside Pulledpork configuration file located inside `/etc/snort/pulledpork.conf`. Below figure 20 shows the Pulledpork configuration file.

```
# note that the url, rule file, and oinkcode itself are separated by a pipe |
# i.e. url|tarball|123456789,
rule_url=https://www.snort.org/reg-rules/|snortrules-snapshot-29170.tar.gz|cb2e[REDACTED]b846f
# NEW Community ruleset:
rule_url=https://snort.org/downloads/community/|community-rules.tar.gz|Community
# NEW For IP Block lists! Note the format is urltofile|IPBLOCKLIST|<oinkcode>
# This format MUST be followed to let pulledpork know that this is a blocklist
rule_url=https://snort.org/downloads/ip-block-list|IPBLOCKLIST|open
# THE FOLLOWING URL is for emergingthreats downloads, note the tarball name change!
```

Figure 20: Pulledpork configuration

For the first-time run Pulledpork script manually. Then after this script can be added to a crontab. For optimum security ruleset must update at least once a week.

```
Writing /var/log/sid_changes.log...
Done
Rule Stats...
New:-----300
Deleted:---0
Enabled Rules:----10168
Dropped Rules:----2
Disabled Rules:---32516
Total Rules:-----42686
IP Blocklist Stats...
Total IPs:-----1414

Done
Please review /var/log/sid_changes.log for additional details
Fly Piggy Fly!
ni@racphorumi: /etc/snort $
```

Figure 21: Pulledpork installing rules

Rule validation can be done by using below command

```
sudo snort -T -c /etc/snort/snort.conf -i wlan0:eth0
```


2.4.6. Configuring the Iptables firewall

Since this device operating in the middle of the network set of firewall policy rules were created as an extra layer of protection for the IDS. These rules were created in a manner, normal operations will not be interrupt. Additionally, these rules will ease the load from Snort IDS process.

These rules are able to provide protection against attacks such as DDoS attacks, SSH attacks, brute force attacks and more.

```
pi@raspberrypi:~$ sudo iptables -L
Chain INPUT (policy ACCEPT)
target     prot opt source                destination           ctstate RELATED,ESTABLISHED
ACCEPT     all  -- anywhere              anywhere              ctstate RELATED,ESTABLISHED
ACCEPT     all  -- anywhere              anywhere              ctstate RELATED,ESTABLISHED
DROP       all  -- anywhere              anywhere              state INVALID
DROP       icmp -- anywhere              anywhere              icmp address-mask-request
DROP       icmp -- anywhere              anywhere              icmp timestamp-request
ACCEPT     tcp  -- anywhere              anywhere              tcp flags:RST/RST limit: avg 2/sec burst 2
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:ssh ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp spt:ssh ctstate ESTABLISHED
ACCEPT     tcp  -- 192.168.4.0/24         anywhere              tcp dpt:rsync ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              multiport dports http,https ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- 192.168.4.0/24         anywhere              tcp dpt:mysql ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:smtp ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:imap2 ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:imaps ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:pop3 ctstate NEW,ESTABLISHED
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:pop3s ctstate NEW,ESTABLISHED
LOG         all  -- 10.0.0.0/8             anywhere              limit: avg 5/min burst 7 LOG level warning p
DROP       all  -- 10.0.0.0/8             anywhere
DROP       all  -- anywhere              anywhere              MAC 00:0F:EA:91:04:08
ACCEPT     tcp  -- anywhere              anywhere              tcp dpt:ssh MAC 00:0F:EA:91:04:07
REJECT     all  -- 1.2.3.4                anywhere              TTL match TTL < 40 reject-with icmp-port-unr
tcp  -- anywhere              anywhere              tcp dpt:ssh ctstate NEW recent: SET name: DE
tcp  -- anywhere              anywhere              tcp flags:FIN,SYN,RST,ACK,SYN
limit: avg 1/sec burst 1
LOG         icmp -- anywhere              anywhere              limit: avg 1/sec burst 1 LOG level warning p
DROP       icmp -- anywhere              anywhere
DROP       all  -f anywhere              anywhere
DROP       tcp  -- anywhere              anywhere              tcp flags:FIN,SYN,RST,PSH,ACK,URG/FIN,SYN,RS
DROP       tcp  -- anywhere              anywhere              tcp flags:FIN,SYN,RST,PSH,ACK,URG/NONE
```

Figure 22: IPTables Firewall policy rules

2.4.7 Filebeat installation and configuration

In order to securely ship alert log files from IDS to log management server, Beats data shipper was installed in IDS. Once Filebeat is downloaded and installed the logs which needs to shipped to Logstash has to be specified in “/etc/filebeat/filebeat.yml” file, along with the log server IP and port number. However, this communication is not secured, to remedy this problem we encrypted this communication using as SSL certificate as shown in figure 23.

```
# ----- Logstash Output -----

filebeat.inputs:
- type: log
  paths:
    - /var/log/snort/snort.csv
  tags: ["snort"]

output.logstash:
  hosts: ["192.168.4.6:5044"]
  ssl:certificate_authorities: ["/etc/pki/filebeat/logstash.crt"]
  ssl.certificate: "/etc/client.crt"
  ssl.key: "/etc/client.key"
```

Figure 23: Filebeat configuration

2.4.8 Logstash install and configuration

Logstash collect data for Filebeat agents and filter them according to configurations, those filtered data will be forwarded to Elasticsearch. As shown in the figure 23 the host and port are defined. This IP and port number is the same host and port as mentioned in the filebeat output section additionally user needs to mention the SSL certificates that been used. User has to define the filters according to the data that been sent to Logstash. Output section contains the Elasticsearch address.

```
input {
  beats {
    port => "5044"
    type => "Snort"
    ssl => true
    ssl_certificate_authorities => ["/etc/ca.crt"]
    ssl_certificate => "/etc/server.crt"
    ssl_key => "/etc/server.key"
    ssl_verify_mode => "force_peer"
  }
}
```

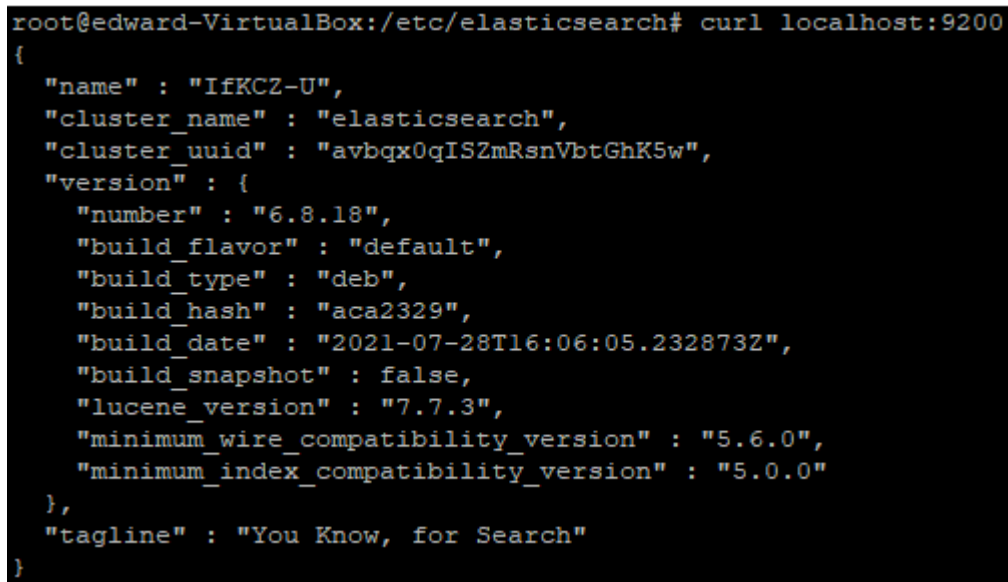
Figure 24: Input section Logstash configuration

```
filter {
  csv {
    separator => ","
    skip_header => "true"
    columns => ["msg", "timestamp", "proto", "src", "dst"]
  }
}
```

Figure 25: Filter and Output section

2.4.9 Elasticsearch installation and configuration

Elasticsearch has the ability to store, search and analyze huge volumes of data rapidly. This can be used with Logstash to collect, aggregate and parse the data to then file the data to the GUI Kibana. In order to install, Elasticsearch can be downloaded from elastic official website. There are no any configurations to be done initially to use Elasticsearch alongside with elastic stack.



```
root@edward-VirtualBox:/etc/elasticsearch# curl localhost:9200
{
  "name" : "IfKCZ-U",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "avbqx0qISZmRsnVbtGhK5w",
  "version" : {
    "number" : "6.8.18",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "aca2329",
    "build_date" : "2021-07-28T16:06:05.232873Z",
    "build_snapshot" : false,
    "lucene_version" : "7.7.3",
    "minimum_wire_compatibility_version" : "5.6.0",
    "minimum_index_compatibility_version" : "5.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Figure 26: The curl request and Elasticsearch response

2.4.10 Kibana installation and configuration

Kibana module is working in harmony with Elasticsearch to display alert logs. Kibana can be used to search, view and interact with the data stored by the Elasticsearch. User have the full ability to create and customize data charts and graphs to display in a presentable manner. Having a web-based interface makes it easy to access understand large volumes of data. Same as before Kibana can be downloaded via the Elastic official website. Afterwards, entering “<IP address>:5061” (5061 being Kibana’s default port) in the browser URL field Kibana should load up.

Below figure 27 shows Elasticsearch stored data without any filters or graphs. This can be filtered to only display useful information such as IP addresses, event message, timestamps shown on figure 28.

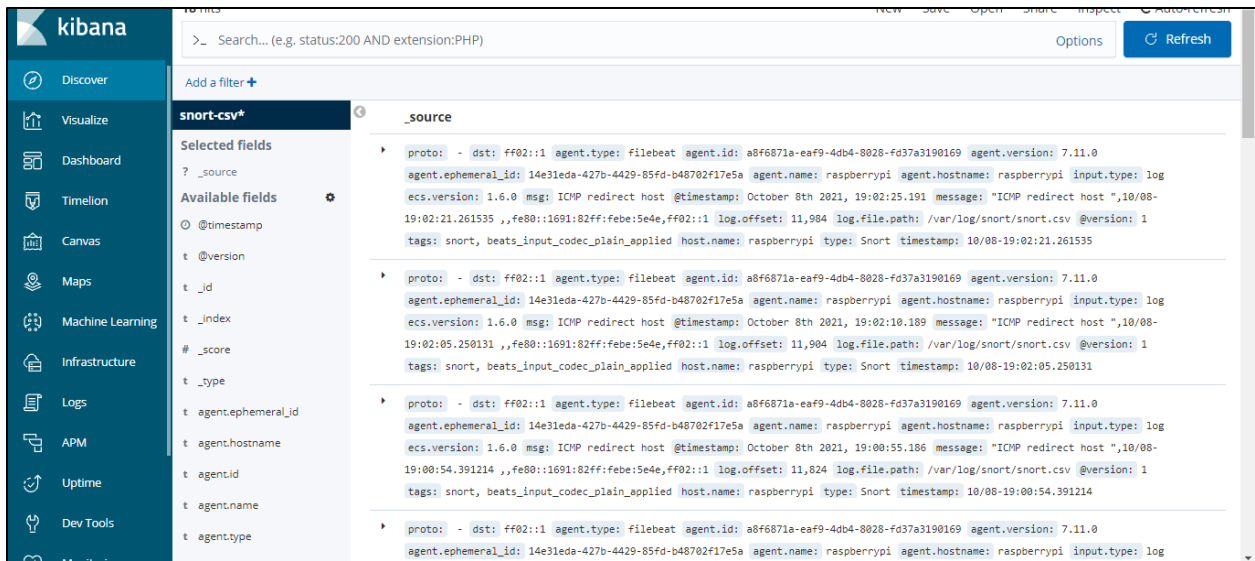


Figure 27: Unorganized Data in Kibana

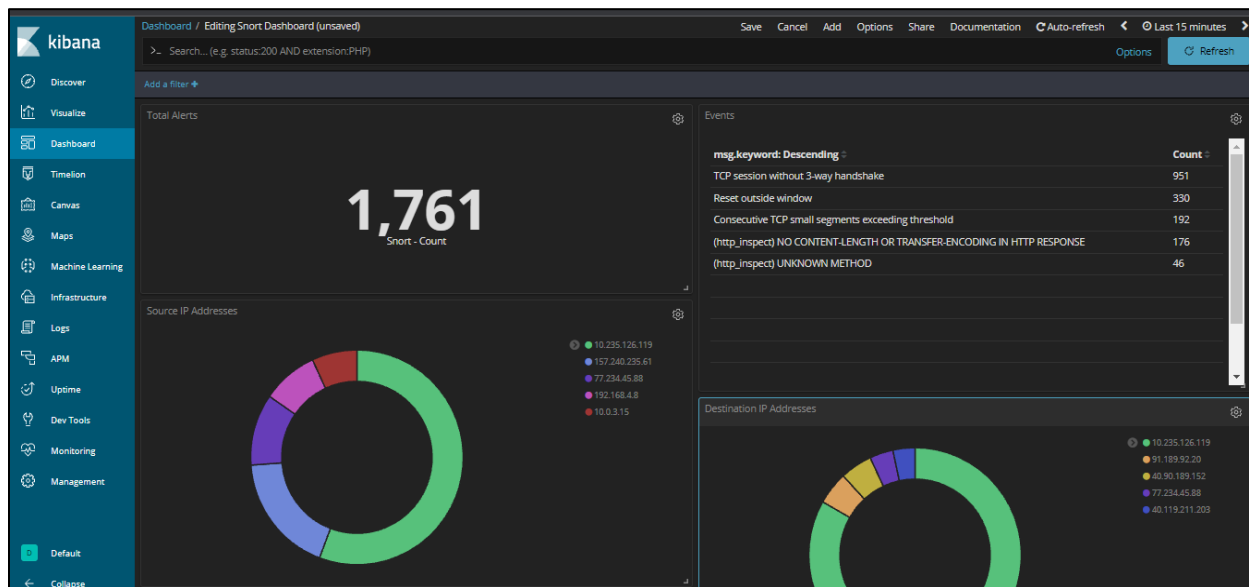


Figure 28: Visualize data using graphs and charts

2.5 Testing

The main goal of testing phase is to assess the Raspberry Pi4's performance while running Snort IDS. After validating installation and configurations, testing phase begins. There are two methods of testing will be used. The first method is to test using intrusion detection evaluation dataset. The second method is simulating real world attack scenarios. For all testing purposes we used Kali Linux virtual machine and some of its pre-install tools to simulate attacks. All these test scenarios were conducted using 1200 (Community ruleset), 3000, 6000 and 10 000 lines of rules, because previous research shows lines of rules have a direct correlation with RAM usage. Iptables Firewall rules were flushed and disabled.

2.5.1 Intrusion detection evaluation dataset

This method evaluates IDS's resource consumption as well as attack detection capabilities by using packet trace file dataset, using a Kali Linux virtual machine with TCPReplay 4.3.3 to replay network traffic at different speeds. While executing these attacks, IDS is running in-line mode between the interfaces eth0 and wlan0 with Barnyard2 and Filebeat processors running in background.

An intrusion detection dataset is a dataset that created to evaluate the performance of an IDS. These datasets are formulated by researchers under a specific set of requirements and conditions containing various types of normal internet traffic patterns as well as attack patterns such as DoS, DDoS, port scanning, brute force attacks and ransomware attacks.

Choosing a correct dataset for specific need is a challenge. There are two ways to choose a dataset. One method is to Creating a dataset from scratch satisfies the purpose of the experiment. This however is not an easy task. The researcher should have the time and capabilities to run, capture, and label different types of attacks. The other method is to use an existing dataset from internet sources. There are many types of datasets can be found in the internet varying from simple trace files contains 1000 lines to massive datasets such as Canadian Institute of Cybersecurity Intrusion Detection Dataset (CICIDS2017) containing 30 000 000 lines [22].

Since the CICIDS2017 dataset is a huge dataset with 2,830,744 rows, it is unrealistic to our requirements. We customized and created our own dataset by using many trace files from internet sources. We created this dataset in such manner it has the diversity of CICIDS2017 dataset. Our dataset contains 200 000 rows and illustrate below types of attacks table 4.

Table 4: Containing attack packets

Attack Type	Count
DDOS attacks	5000
Bot attacks	8000
Port Scans	700
Ransomware attacks	740
Brute Force attacks	1200

2.5.2 Simulation of attacks

This method is mainly focusing on evaluating attack detection capabilities by performing set of real-world attack simulations against host machine connected to the Raspberry Pi access point. using a Kali Linux virtual machine with its pre-installed tools and various other attack scripts. While executing these attacks, IDS is running in-line mode between the interfaces eth0 and wlan0.

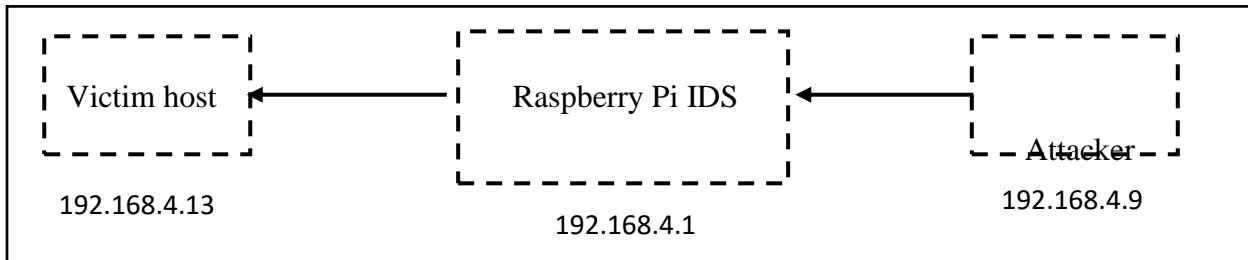


Figure 29: Testing network diagram

2.6 Commercialization Aspects

Advanced and efficient anti-virus solutions with built-in host-based intrusion detection systems are available in current market for a reasonable price. An argument can be made anti-virus solutions can protect for cyber threats. These solutions are targeted to protect only one host machine such as a computer and unable protect the vast amount of interconnected IOT devices from cyber threats. Only network-based IDS solutions are capable of protecting the entire network such solutions can be unaffordable for smaller organizations.

After successful testing phase we planned to promote this solution as a cost-effective network security and monitoring solution for small to medium businesses as well as home owners. Specially targeted towards businesses who are migrating towards Industry 4.0.

3. RESULTS & DISCUSSION

3.1 Results

These test results can be used for validate the feasibility of the proposed IDS solution. Below figure 30 shows the Snort IDS idling at 3.3 % of CPU usage and 709MB of Memory used with 10 000 lines of rules loaded.

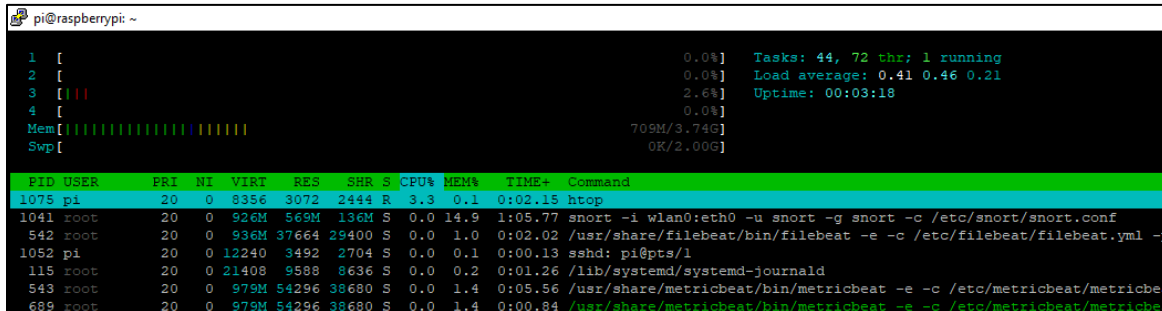


Figure 30: Snort idle resource consumption

Next the customized PCAP dataset were replayed using TCP replay at 5 different speeds starting from 5Mbps to 30 Mbps. This data speed was selected after benchmarking the Raspberry Pi 4's inbuilt wireless adaptor.

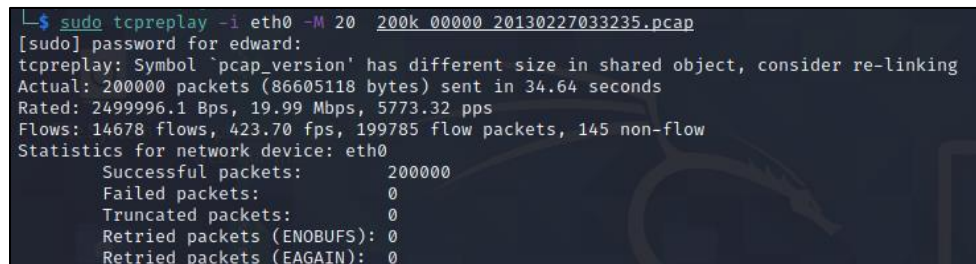


Figure 31: TCPReplay replaying at 20Mbps

3.1.1 CPU usage

Figure32 shows the relationship between the data rate which packets were transmitted from and the Raspberry Pi 4 CPU consumption. The main cause for this behavior is CPU interrupts, when a new packet arrives at IDS it triggers an CPU interrupt to inform about net packet arrival, these interrupts can overwhelm the CPU and could lead to packet drops. Moreover, tests reveal the number of rules have no considerable impact on CPU usage.

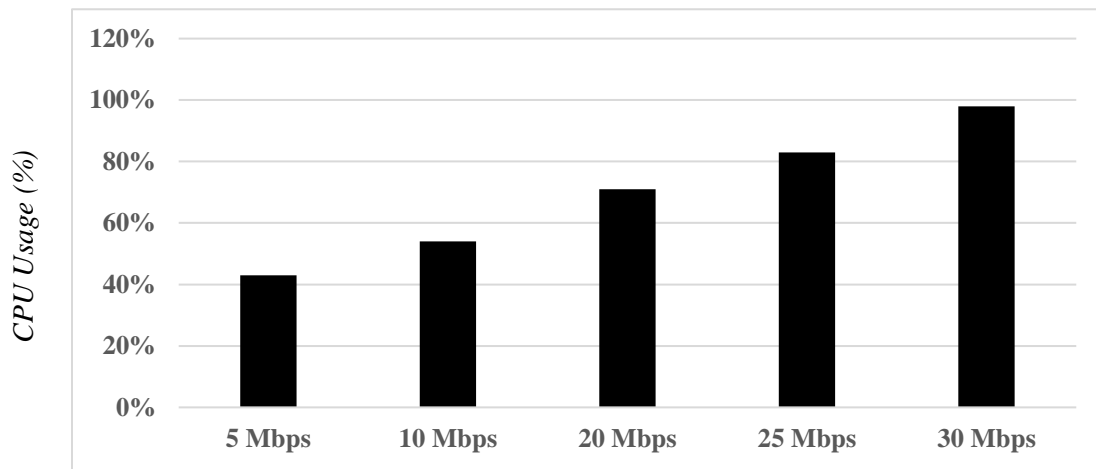


Figure 32: CPU Usage of Raspberry Pi IDS

Highest recorded CPU usage stated at 98%, this results a considerable amount of packet drops. below line graphs Figure 33 shows CPU usage and packet loss rate have a direct relationship.

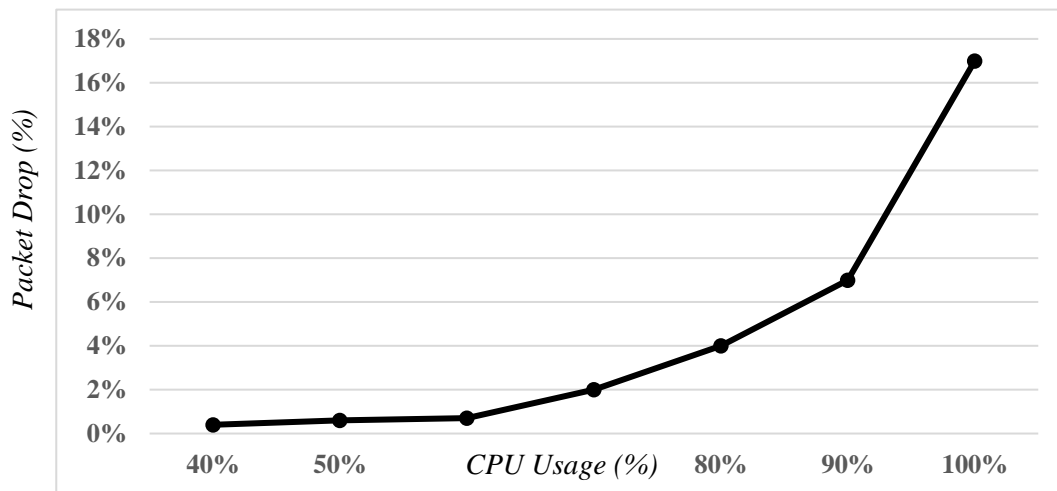


Figure 33: Packet loss and CPU usage

```

Packet I/O Totals:
  Received:      182792
  Analyzed:      182791 ( 99.999%)
  Dropped:       37727 ( 17.108%)
  Filtered:      0 ( 0.000%)
  Outstanding:   1 ( 0.001%)
  Injected:      0
  
```

Figure 34: Highest recorded packet drop

3.1.2 RAM usage

Figure 35 shows the correlation between the RAM usage and Snort rulesets. Even though rules have a negligible impact on CPU usage, it can directly impact RAM usage. The main cause is that Snort Detection engine loaded all the defined rules to its memory to carry out IDS functionalities, owing to this fact a higher number of rules demands a higher consumption of memory. A clear difference can be seen between the four rulesets.

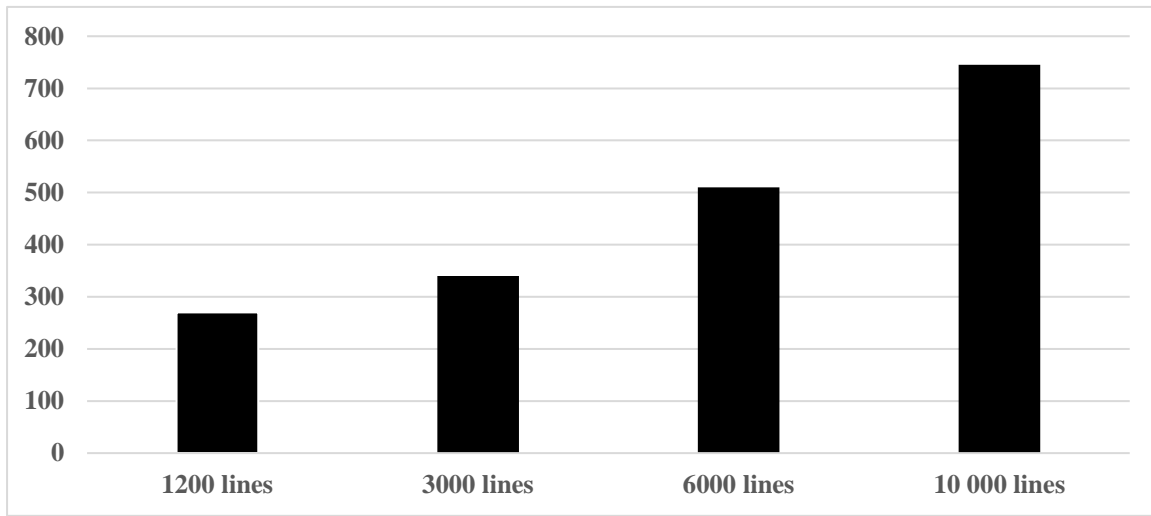


Figure 35: RAM usage

3.1.3 Generated alerts

Figure 36 shows the generated alerts while using the three different rulesets.

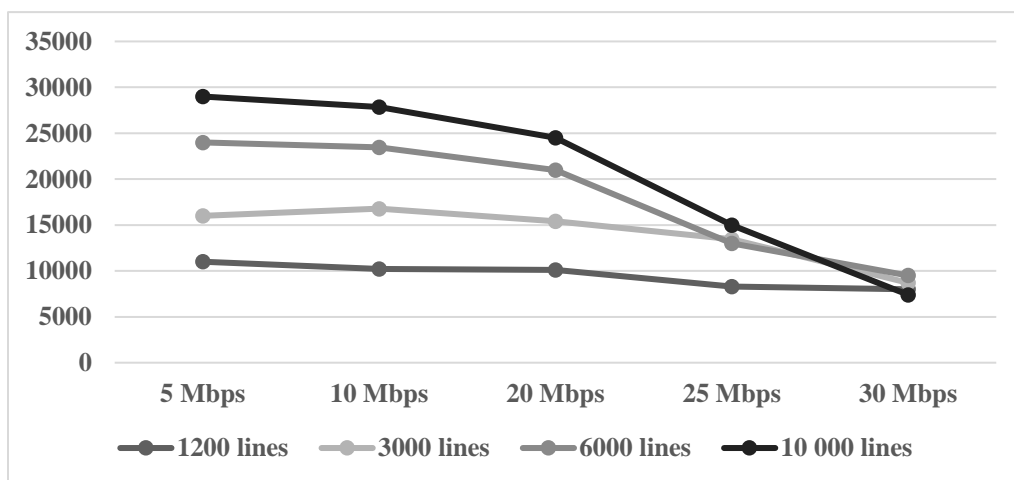


Figure 36: Generated alert

While running this test we observed the number of generated alerts were dropped when the data rate increases. The main reason for this is high CPU usage as shown in figure 32. As discussed previously when the CPU is overwhelmed with interrupts it loses the ability to process every packet, this will lead to packet drops and less rules will get triggered.

3.1.4 Real-world attack simulation

- Port scan

Initially a Nmap port scan was conducted using Kali Linux just to trigger a response from the IDS. As expected, Snort identified the attack and generated alerts. Below command was used and figure shows the generated alerts.

```
nmap -p- -sV -sT -A 192.168.4.1
```

```
06/09-06:03:22.000000 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] (UDP) 0.0.0.0:68 -> 255.255.255.255:67
06/09-06:03:24.935419 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] (UDP) 0.0.0.0:68 -> 255.255.255.255:67
06/09-06:03:25.672369 [**] [1:527:8] BAD-TRAFFIC same SRC/DST [**] [Classification: Potentially Bad Traffic] [Priority: 2] (UDP) 0.0.0.0:68 -> 255.255.255.255:67
06/09-06:04:19.518056 [**] [1:249:8] DDOS mstream client to handler [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:51278 -> 192.168.4.1:15104
06/09-06:04:19.979248 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] (UDP) 192.168.4.12:53394 -> 192.168.4.1:1900
06/09-06:04:21.006204 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] (UDP) 192.168.4.12:53394 -> 192.168.4.1:1900
06/09-06:04:21.995984 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] (UDP) 192.168.4.12:53394 -> 192.168.4.1:1900
06/09-06:04:23.000728 [**] [1:1917:6] SCAN UPnP service discover attempt [**] [Classification: Detection of a Network Scan] [Priority: 3] (UDP) 192.168.4.12:53394 -> 192.168.4.1:1900
06/09-06:04:25.348019 [**] [1:1421:11] SNMP AgentX/tcp request [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 192.168.4.9:60072 -> 192.168.4.1:161
06/09-06:04:27.248976 [**] [1:1418:11] SNMP request top [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 192.168.4.9:57938 -> 192.168.4.1:161
06/09-06:04:27.875817 [**] [1:1420:11] SNMP trap tcp [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 192.168.4.9:46494 -> 192.168.4.1:162
06/09-06:04:34.302503 [**] [1:257:9] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 192.168.4.9:58724 -> 192.168.4.1:53
06/09-06:04:38.373952 [**] [1:257:9] DNS named version attempt [**] [Classification: Attempted Information Leak] [Priority: 2] (TCP) 192.168.4.9:58744 -> 192.168.4.1:53
```

Figure 37: Port scan alerts

- Simulating DDOS attacks

Low Orbit Ion Cannon (LOIC) was utilized to simulate the DDOS attacks against Apache web server running on Ubuntu OS that were inside the same network as shown in figure 29.



Figure 38: LOIC DDOS simulation

Snort was able to recognize this attack, LOIC was created more than 500 000 requests towards the web server. Snort was able to detect average of 95% of packets.

Figure 39: Snort DDoS alerts.

Another DOS attack was launched using a python script from internet sources [23]. This is a Slowloris type DOS attack. Slowloris is an application layer attack that transmits partial packets to a web server. In this case Snort community ruleset failed to detect this attack. However, with registered rule set the attack was detected without any issue.

Figure 40: Slowloris attack response

In order to simulate brute force attack, Kali Linux pre-installed tool named Medusa was used, targeted towards the a IDS's open SSH port.

```

(edward@sam) - [usr/share/wordlists]
$ medusa -u root -P /usr/share/wordlists/fasttrack.txt -h 192.168.4.1 -M ssh
Medusa v2.2 [http://www.fooofus.net] (C) JoMo-Kun / Fooofus Networks <jmk@fooofus.net>

ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Spring2017 (1 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Spring2016 (2 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Spring2015 (3 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Spring2014 (4 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Spring2013 (5 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: spring2017 (6 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: spring2016 (7 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: spring2015 (8 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: spring2014 (9 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: spring2013 (10 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Summer2017 (11 of 221 complete)
ACCOUNT CHECK: [ssh] Host: 192.168.4.1 (1 of 1, 0 complete) User: root (1 of 1, 0 complete) Password: Summer2016 (12 of 221 complete)

```

Figure 41: SSH brute force attack using Medusa

In this case Snort did not successful detect all the SSH brute force attacks. Snort was only able to detect 104 out of 221 attempts. The detection average was 48%.

```

kcz10/07-12:48:28.832555 [**] [1:4:1] Potential SSH Brute Force Attack [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:58054 -> 168.4.1:22
kcz10/07-12:48:28.832555 [**] [1:4:1] Potential SSH Brute Force Attack [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:58054 -> 168.4.1:22
10/07-12:48:58.402667 [**] [1:4:1] Potential SSH Brute Force Attack [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:58060 -> 192.4.1:22
10/07-12:49:17.410910 [**] [1:10000001:1] ICMP test [**] [Priority: 0] (IPv6-ICMP) fe80::1691:82ff:febe:5e4e -> ff02::1
10/07-12:49:30.593694 [**] [1:4:1] Potential SSH Brute Force Attack [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:58066 -> 192.4.1:22
10/07-12:49:39.044552 [**] [1:10000001:1] ICMP test [**] [Priority: 0] (IPv6-ICMP) fe80::1691:82ff:febe:5e4e -> ff02::1
10/07-12:50:01.971562 [**] [1:4:1] Potential SSH Brute Force Attack [**] [Classification: Attempted Denial of Service] [Priority: 2] (TCP) 192.168.4.9:58072 -> 192.4.1:22
10/07-12:50:06.404346 [**] [129:15:2] Reset outside window [**] [Classification: Potentially Bad Traffic] [Priority: 2] (TCP) 138.201.57.222:443 -> 10.235.126.119:6

```

Figure 42: SSH brute force attack alerts

- Man in the middle attack

To simulate MITM attack, Ettercap tool was utilized. Ettercap was successfully poisoned the ARP cache and all the packets were routed through the attacker host machine. However, Snort was failed to detect this attack so the below custom rule was added to the snort ruleset.

```

alert icmp $EXTERNAL_NET any -> $HOME_NET any (msg : "ICMP redirect host"; icode : 1; itype : 5; sid : 472; )

```

When an MITM attack occurs all the traffic were routed through the attacker machine and thus a longer route is used to packets to reach the gateway router. By using this theory above rule was able to detect MITM attack.

```

10/08-00:04:26.013064 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:30.685540 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:47.154731 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:47.154888 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:48.161326 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:48.161455 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:49.176745 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:49.176860 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:50.192225 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:04:50.192346 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]
10/08-00:05:05.101313 [**] [1:10000001:1] ICMP redirect host [**] [Priority: 2]

```

Figure 43: Detection of MITM attacks

3.2 Research Findings

These test results revealing intrusion detection system based on Raspberry Pi 4 can handle 7 to 10 number of hosts with low to moderate traffic without any compromises. Well suited for home or small Industry 4.0 environments

While testing, we observed data rate and the CPU usage have a direct relationship as shown in figure 32, when packets transmitted with higher bandwidth the CPU usage reaches maximum, which also lead to packet drops and impact alert generate task. When it comes to RAM usage even with 10 000 lines of rules, IDS uses less than 1GB of RAM leaves plenty of headspace for other processors.

When simulating attacks community rule set performed poorly. It failed to detect any of network attacks other than the port scan. After adding 3000 lines of rules, it was able to detect LOIC DDOS simulation attack. 6000 lines of rules were able to detect Slowloris attack as well as brute force attack. However, even with 10 000 lines of rules IDS failed to detect MITM attack. This finding shows the important of custom rules. The results are presented in Table.

Table 5: Simulated attack findings

Attack Type	Community	Registered		
	1200 lines	3000 lines	6000 lines	10 000 lines
Port SCAN	✓	✓	✓	✓
LOIC DDOS simulation		✓	✓	✓
Slowloris attack			✓	✓
Brute force attack			✓	✓
MITM attack	A custom rule was created			

One of the main drawbacks of Raspberry Pi4 is its weak Wi-Fi bandwidth and strength, while benchmarking it only reaches 29.7 Mbps. This can be remedy by adding a capable external wireless adaptor or and external antenna. When it comes to heat generation with a 7mm cooling fan and heatsink maximum operating temperature recorded was 45 degrees Celsius.

3.3 Discussion

Proposed solution is a compelling solution for small to medium industry 4.0 organizations with little to no network monitoring functions, with expansion of the network option is available to add more IDS sensors without significant budget requirements. This solution can be easily tailored in for many different occasions & requirements, for instance this can act as a portable secure network gateway due to its small size and could even run using a power bank or can be use in passive mode with a network switch with minimal configurations.

A question can be made is this IDS solution is enough to protect from cyber-attacks? The simple answer is NO. IDS alone cannot provide full protection against cyber threats it is only a one aspect. However, with inclusion of security policies, standards, proper security configurations alongside with intrusion detection will provide a comprehensive protection against cyber threats.

Hereafter, we planned to deploy this IDS solution in industry 4.0 environment for real-world testing along with cloud-based log management server. Moreover, perform experiments with IDS software solutions such as Suricata and evaluate the performance on hardware solutions such as Asus Tinker board. Along with cloud-based log management server.

4. CONCLUSION

Due to growing number of interconnected devices having a proper network protection is crucial regardless of type or size of the organization. Even though security should be prioritized, organizations neglect security due to budget constraints. The proposed IDS solution provides network security and monitoring by utilizing Snort IDS in Raspberry Pi for small to medium Industry 4.0 environments. Rather than spending money on unnecessary and excessive cooperate level network protection, proposed solution can be utilize for a reasonable price.

In order to test these claims we conducted tests using two different methods involving custom made IDS evaluation dataset that we built to test out Raspberry Pi performance consumption and using real-world attack scenarios to test out attack detection capabilities. Additionally, we experimented with 4 different rulesets.

In first testing method IDS evaluation dataset was transmitted around the network in five different data speeds using tool known as TCPReplay. While testing we monitored statistics such as CPU usage, RAM usage, packet drop percentage, generated alert count and device temperature. While testing we observed a direct relationship between the data speeds and the CPU usage higher data speeds is the main culprit for CPU usage spikes eventually leads to packet drops and effecting alert generating ability. Number of rules loaded into the IDS impacted the RAM usage spike. However, highest RAM usage recorded was 750MB. The highest recorded temperature was at 45 degrees of Celsius.

In second testing method we evaluate by simulating real-world attacks. Initially started with port scan and then conducted DDOS attack and Brute force attack. Port scan was simulated using Nmap tool that can be found in Kali Linux. The results showed IDS identified the port scan without an issue. To simulate DDOS attack Low Orbit Ian Cannon (LOIC) attack simulator tool and Slowloris attack was utilized. IDS was able to recognize this attack with 6000 lines of rules. LOIC was created more than 500 000 requests towards the web server. Results shows if the correct rules are set IDS will be able to detect the attack with high accuracy, 95% of packets were detected by the IDS. Medusa, another pre-installed tool in Kali Linux was used to simulate the Brute force attack targeted towards openSSH running on IDS itself. In this case Snort did not successful detect all the SSH brute force attacks packets. Snort was only able to detect 104 out of 221 attempts. The

detection average was 48%. Snort completely failed to detect MITM attack due to lack of rules a custom rule was added to mitigate this.

Test results validate our claims, that the Raspberry Pi 4 is capable of running Snort IDS and act as a Wi-Fi access point without any major compromises. Even though security should be prioritized, organizations neglect security due to budget constraints. The proposed IDS solution is ideal cost-effective solution for small to medium Industry 4.0 environments.

Our recommendation is to utilize 8000 to 12 000 lines of rules for optimal performance and security since the management is an important task in Snort IDS. Adding more rules blindly does not translate to more security adding correct rules is more important than adding lots of rules. Adding higher number of rules is not recommended it could lead to high CPU and RAM usage eventually cause packet drops.

To improve wireless connectivity speed our recommendation is to invest in an external wireless adaptor that support 150 Mbps, or else the option is available to connect to a switch and enable port mirroring. Having a proper cooling system is equally important if the device got hot, it results thermal throttling.

When it comes to limitation of this project, almost all of our tests were conducted in a virtual environment. To conduct real-world testing, it must be installed in a real-world industry 4.0 environment. An ideal scenario would be a small manufacturing environment with more than 10 hosts including IoT components. In such scenario it might perform even better without the constant high bandwidth. Only a long-term testing process will be able to assess its true capabilities.

REFERENCES

- [1] K. Zhou, Taigang Liu, and Lifeng Zhou, "Industry 4.0: Towards future industrial opportunities and challenges," in 2015 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD), Zhangjiajie, China, Aug. 2015, pp. 2147–2152, doi: 10.1109/FSKD.2015.7382284.
- [2] H. Xu, W. Yu, D. Griffith, and N. Golmie, "A Survey on Industrial Internet of Things: A Cyber-Physical Systems Perspective," IEEE Access, vol. 6, pp. 78238–78259, 2018, doi: 10.1109/ACCESS.2018.2884906.
- [3] S. R. Chhetri, N. Rashid, S. Faezi, and M. A. Al Faruque, "Security trends and advances in manufacturing systems in the era of industry 4.0," in 2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD), Irvine, CA, Nov. 2017, pp. 1039–1046, doi: 10.1109/ICCAD.2017.8203896.
- [4] Robert Mitchell and Ing-Ray Chen. 2014. A survey of intrusion detection techniques for cyber-physical systems. ACM Comput. Surv. 46, 4, Article 55 (April 2014), 29 pages (accessed Mar. 07, 2021).
- [5] Bace, R. and Mell, P. (2001), Intrusion Detection Systems, Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD
- [6] Hwang,K., Cai,M., Chen,Y and Qin,M. , "Hybrid Intrusion Detection with Weighted Signature Generation over Anomalous Internet Episodes", IEEE Transactions on Dependable Computing, Volume: 4 Issue: 1, pp. 41- 55, 2007.
- [7] Michael P. Brennan "Using Snort For a Distributed Intrusion Detection System", version 1.3 in SANS Institute 2020, January 29, 2002
- [8] F. Hugelshofer, P. Smith, D. Hutchison, and N. J. Race, "OpenLIDS:a lightweight intrusion detection system for wireless mesh networks,"in Proceedings of the 15th annual international conference on Mobile computing and networking. ACM, 2009, pp. 309–320.
- [9] A. K. Kyaw, Yuzhu Chen and J. Joseph, "Pi-IDS: evaluation of open-source intrusion detection systems on Raspberry Pi 2," 2015 Second International Conference on Information Security and Cyber Forensics (InfoSec), 2015, pp. 165-170.
- [10] A. Sforzin, F. G. Mármol, M. Conti and J. Bohli, "RPiDS: Raspberry Pi IDS — A Fruitful Intrusion Detection System for IoT," 2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress.
- [11] eSecurityPlanet. 2021. Best Intrusion Detection & Prevention Systems [2021]: Compare 5+ Solutions | eSP. [online] Available at: <<https://www.esecurityplanet.com/products/intrusion-detection-and-prevention-systems/>> .
- [12] Kamble, S., Gunasekaran, A. and Gawankar, S., 2020. Sustainable Industry 4.0 Framework: A Systematic Literature Review Identifying The Current Trends And Future Perspectives. [24] Lins, Theo & Rabelo, Ricardo & Correia, Luiz & Sá Silva, Jorge. (2018). Industry 4.0 Retrofitting. 8-15. 10.1109/SBESC.2018.00011.
- [13]Lins, Theo & Rabelo, Ricardo & Correia, Luiz & Sá Silva, Jorge. (2018). Industry 4.0 Retrofitting. 8 -15. 10.1109/SBESC.2018.00011. [6]Moktadir, M., Ali, S., Kusi-Sarpong, S. and Shaikh, M., 2019. Assessing Challenges For Implementing Industry 4.0: Implications For Process Safety And Environmental Protection.

- [14]Moktadir, M., Ali, S., Kusi-Sarpong, S. and Shaikh, M., 2019. Assessing Challenges For Implementing Industry 4.0: Implications For Process Safety And Environmental Protection.
- [15] Raspberrypi.com. 2021. [online] Available at: <<https://www.raspberrypi.com/products/raspberry-pi-4-model-b/specifications/>> .
- [16] R. Gaddam and M. Nandhini, "An analysis of various snort-based techniques to detect and prevent intrusions in networks proposal with code refactoring snort tool in Kali Linux environment," 2017 International Conference on Inventive Communication and Computational Technologies (ICICCT), 2017, pp. 10-15, doi: 10.1109/ICICCT.2017.7975177.
- [17] N. Houghton. (2010, June) Single threaded data processing pipelines and the Intel architecture. [Online]. Available: <http://vrt-blog.snort.org/2010/06/single-threaded-data-processing.html>
- [18] Elastic.co. 2021. Elasticsearch Guide [7.15] | Elastic. [online] Available at: <<https://www.elastic.co/guide/en/elasticsearch/reference/current/index.html>>.
- [19] Elastic.co. 2021. Logstash Introduction | Logstash Reference [7.15] | Elastic. [online] Available at: <<https://www.elastic.co/guide/en/logstash/current/introduction.html>>.
- [20] Elastic.co. 2021. Filebeat overview | Filebeat Reference [7.15] | Elastic. [online] Available at: <<https://www.elastic.co/guide/en/beats/filebeat/current/filebeat-overview.html>>.
- [21] Elastic.co. 2021. Kibana—your window into Elastic | Kibana Guide [7.15] | Elastic. [online] Available at: <<https://www.elastic.co/guide/en/kibana/current/introduction.html>>.
- [22] Unb.ca. 2021. IDS 2017 | Datasets | Research | Canadian Institute for Cybersecurity | UNB. [online] Available at: <<https://www.unb.ca/cic/datasets/ids-2017.html>>.
- [23] GitHub. 2021. GitHub - gkbrk/slowloris: Low bandwidth DoS tool. Slowloris rewrite in Python.. [online] Available at: <<https://github.com/gkbrk/slowloris>>.

APPENDICES

Appendix A: Turnitin similarity score

The screenshot displays a Turnitin Match Overview report. The document title is "CYBERSECURITY AUTOMATION FOR AN INDUSTRY 4.0 GARMENT MANUFACTURING SYSTEM". The submission is dated "2021-11" and is identified as a "Final Project Thesis" by "A.D.H Jinadasa". The document is for a "B.Sc. (Hons) Degree in Information Technology Specialization in Cyber Security". The overall similarity score is 19%. The report lists 11 matches with their respective sources and similarity percentages.

Match Number	Source	Similarity Percentage
1	Submitted to Sri Lanka ... Student Paper	10%
2	www.guru99.com Internet Source	2%
3	ants.inf.um.es Internet Source	1%
4	www.iert.org Internet Source	1%
5	www.theseus.fi Internet Source	<1%
6	medium.com Internet Source	<1%
7	Submitted to Melbourn... Student Paper	<1%
8	ecommons.usask.ca Internet Source	<1%
9	kanazawa-u.repo.nii.ac... Internet Source	<1%
10	dspace.lboro.ac.uk Internet Source	<1%
11	eprints.nottingham.ac... Internet Source	<1%