|  |  |
| --- | --- |
| **Name** : Muhammad Zain-Ul-Abideen | **Domain** : Cyber Security |
| **CID** : DEP6384 | **Task Instructor** : Noor-Us-Sama Tariq |
| **Topic** : Firewalls And Intrusion Detection | **Tools** : Snort IDS, Fail2Ban & Ubuntu Firewall |

**Task 4**

**Configuring Firewalls and Intrusion Detection Systems**

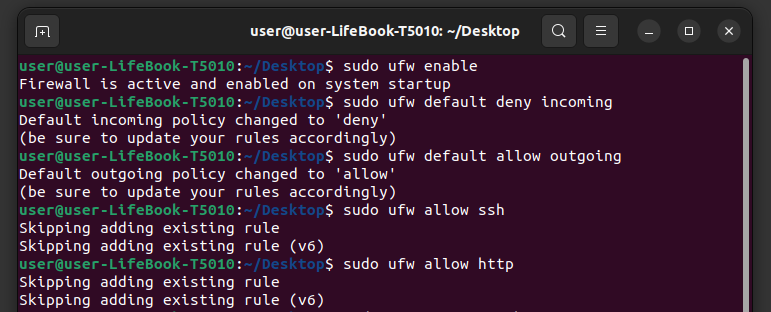
**1. Objective:** Protect the network by setting up firewalls and IDS**.**

**2. Description:** Implement firewalls and intrusion detection systems to monitor and control incoming and outgoing network traffic. Detect and prevent unauthorized access and attacks.

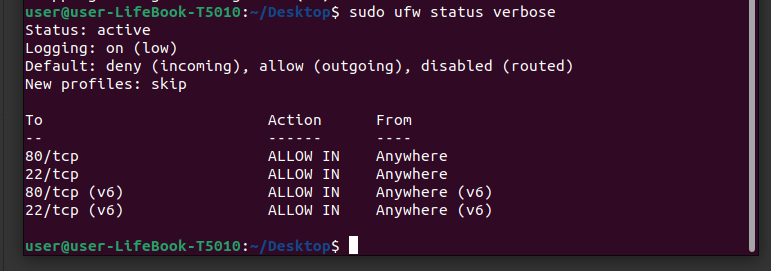
**Key Points:**

* Selecting appropriate firewall and IDS solutions.
* Configuring firewall rules and policies.
* Setting up IDS to monitor network traffic.
* Analyzing IDS alerts and responding to threats.
* Regularly updating and maintaining the configurations

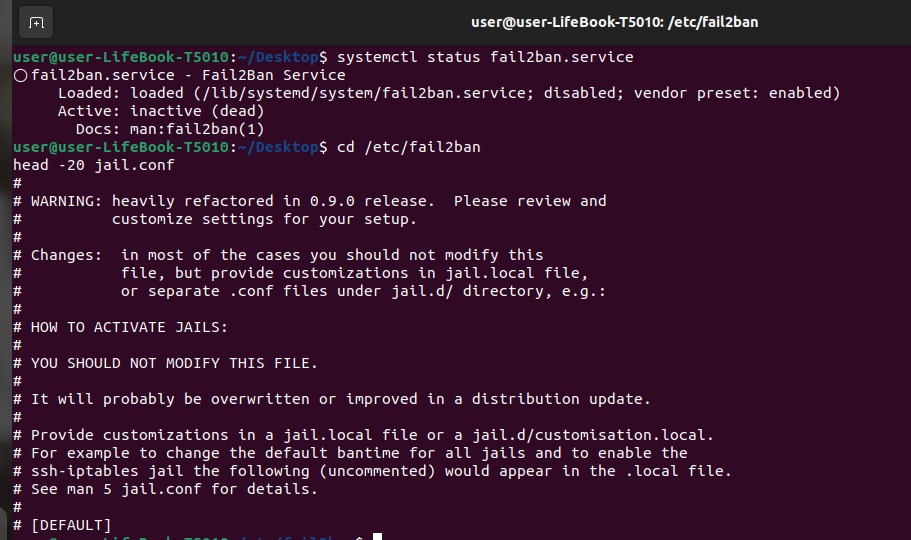
**Answer:** We are going to protect our system’s network from threats and malicious traffic by configuring, setting up and utilizing **Firewalls** and **Intrusion Detection Softwares** and I’ll be implementing this on my **Ubuntu 22.04 LTS (Jammy Version)** instead of using **VMWare.** So the **Firewall** we will select for the recommended task is **Ubuntu Firewall**, **Fail2Ban** & the **IDS System** which we are going to use is **Snort 3.0.** First thing is first we are going to download and **Enable** the **Ubuntu Firewall** on the system using the **Command Terminal.** We can also make our **Custom Rules** to ensure the traffic only we want can enter and access the **System.**



After we have set up **Rules & Policies** for the **Ubuntu Firewall** And **Enabled** it we can check the **Status** of the **Firewall** to ensure that it is functioning and operating smoothly.

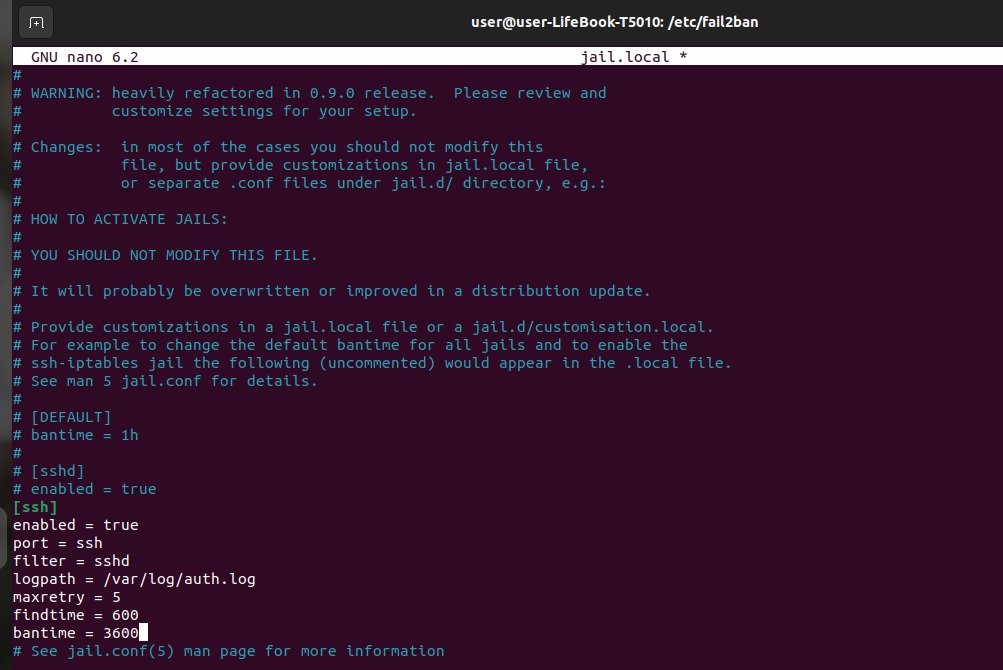


The **Ubuntu Firewall** seems to be running functional and our **Rules** which we have custom made for the **Firewall** also seem to be in proper action. Now we implement **FAIL2BAN :**

****

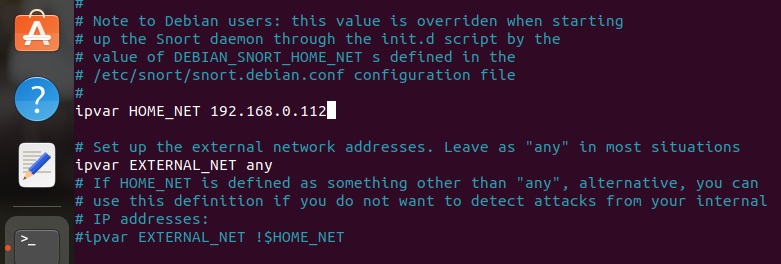
Furthermore we do implement **Fail2ban** to **Monitor** log files and automatically block **IP Addresses** that repeatedly **Fail** to log in via **SSH**. This will help to mitigate **Brute Force** attempts by blocking the source IP addresses of the attack. First we configure **Fail2Ban** as done above using the **Commands.** Then we open the **Configuration File** to add our own **Preferences** using following **Commands.**

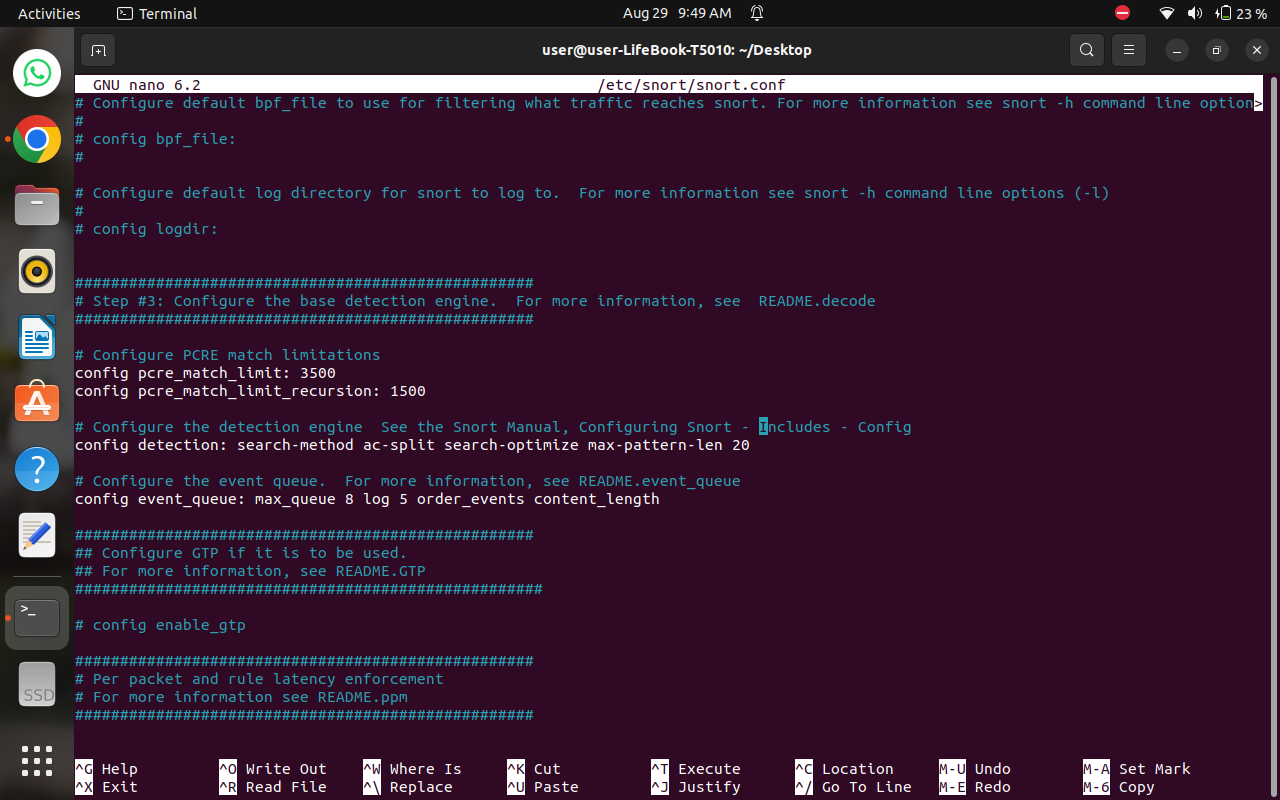




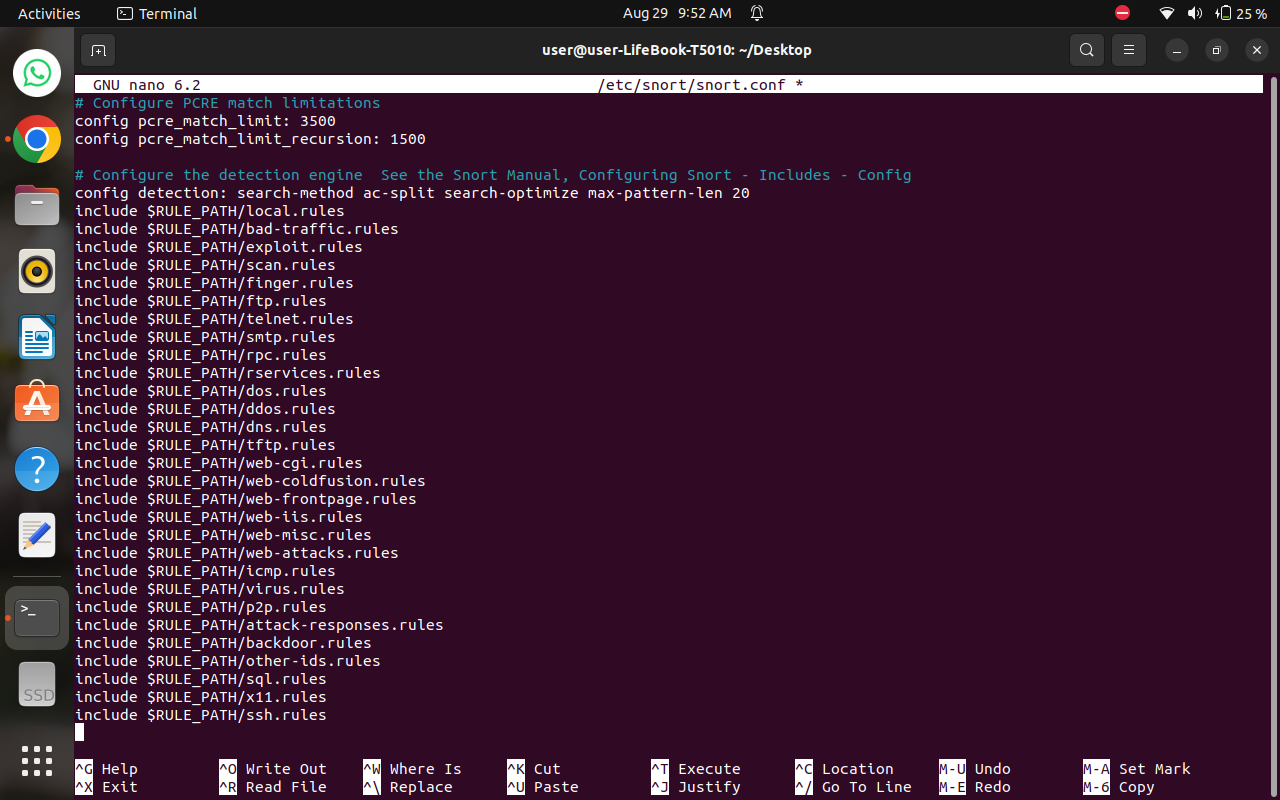


Then we add our own **Rules** & **Preferences** in the **Jail.Local File** before saving it and restarting the **Fail2Ban**. We use **Fail2ban** here to block **IPs** attempting to exploit the **Vulnerabilities**, limiting the attacker's ability to continue probing the server. Automatically block **IPs** that go through suspicious behavior, such as repeated connection attempts or high number of requests, to mitigate the attack. It will strengthen and also defends by blocking **IP addresses** attempting to use compromised credentials. Now we will install and configure **Snort 3.0** in collaboration with **Ubuntu Firewall.** Now here in the **Configuration Menu** we will set our **HOME\_NET Address** to the IP Address where we want **Snort 3.0** to work on. For instance here we can add our own **Network’s Address** which is **192.168.0.112**

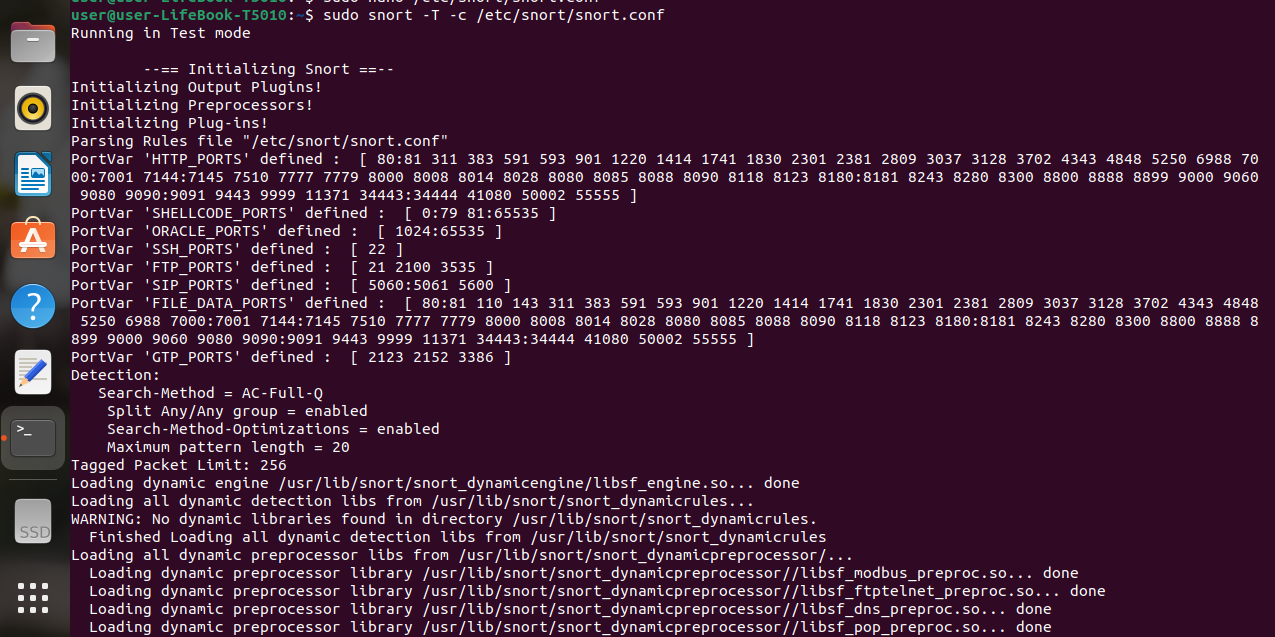




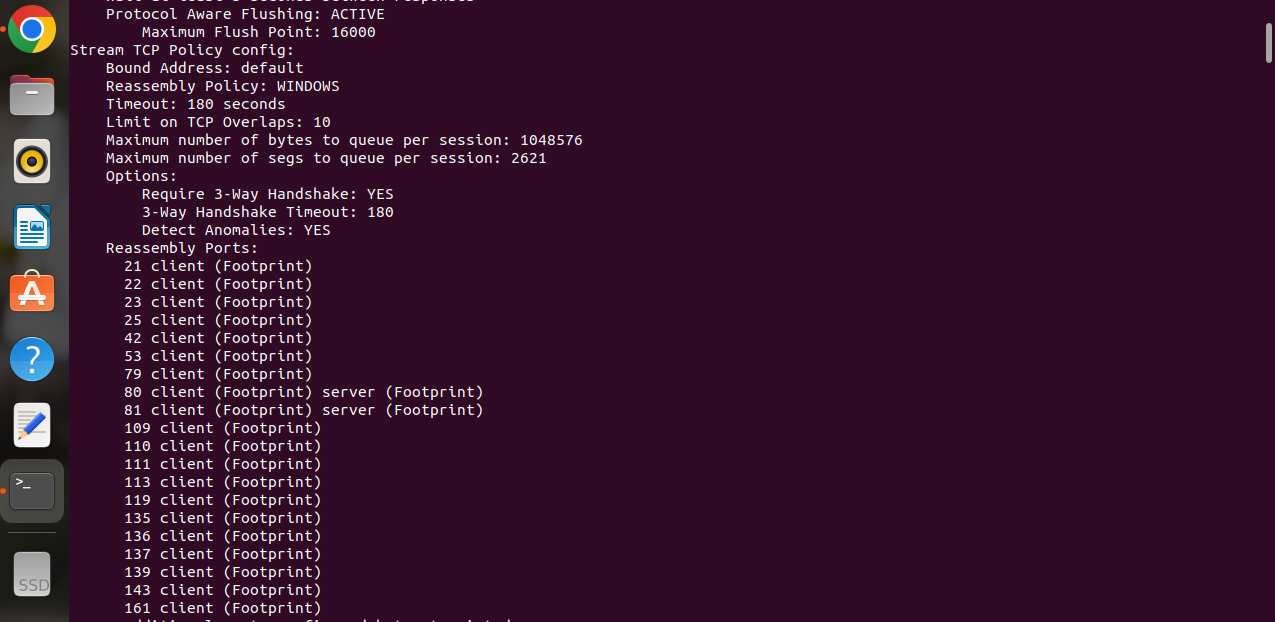
Now save this and in the **Configuration Menu** Select Ctrl+W & Search Include which will take us to **Detection Engine** where we can add our own **Rules** which will be detected using **Snort 3.0**



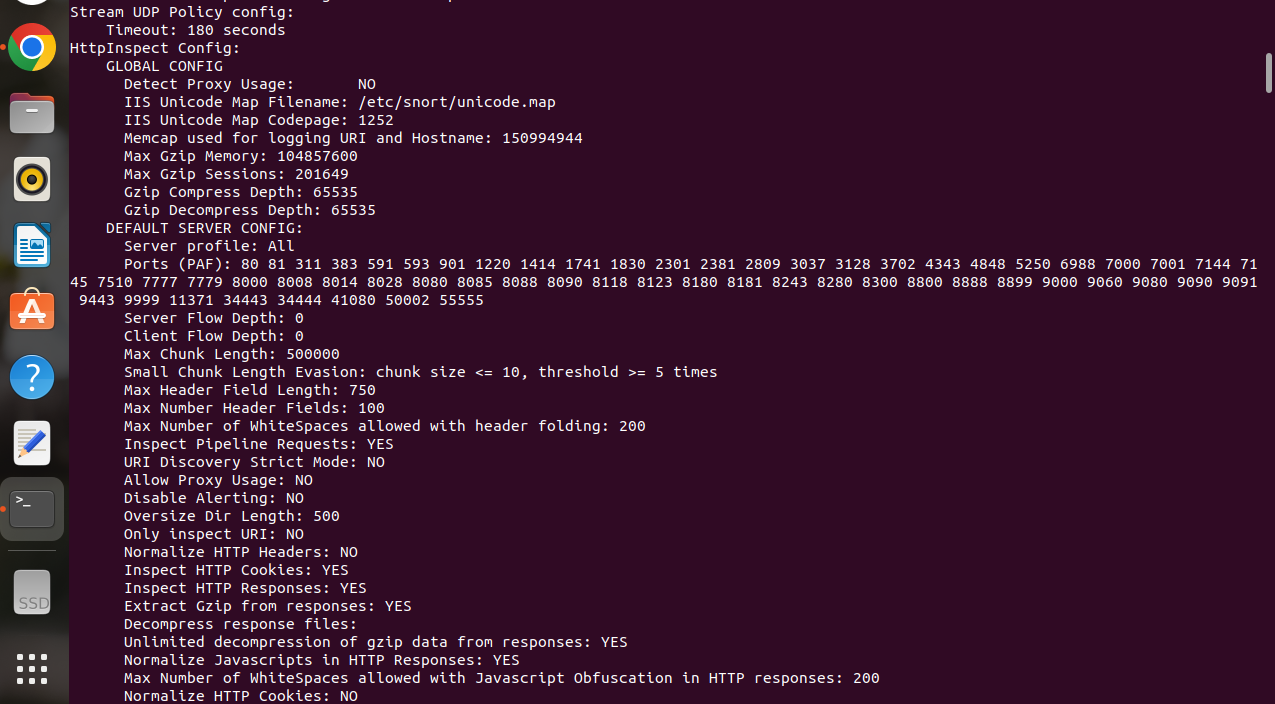
After we are done adding our **Rules** we can save the **Configuration File** and run **Snort** as a service which will keep a lookout in regards for this rules for **Intrusion Detection**. Using the **“sudo snort -T -c /etc/snort/snort.conf”** we can run the **Snort 3.0**



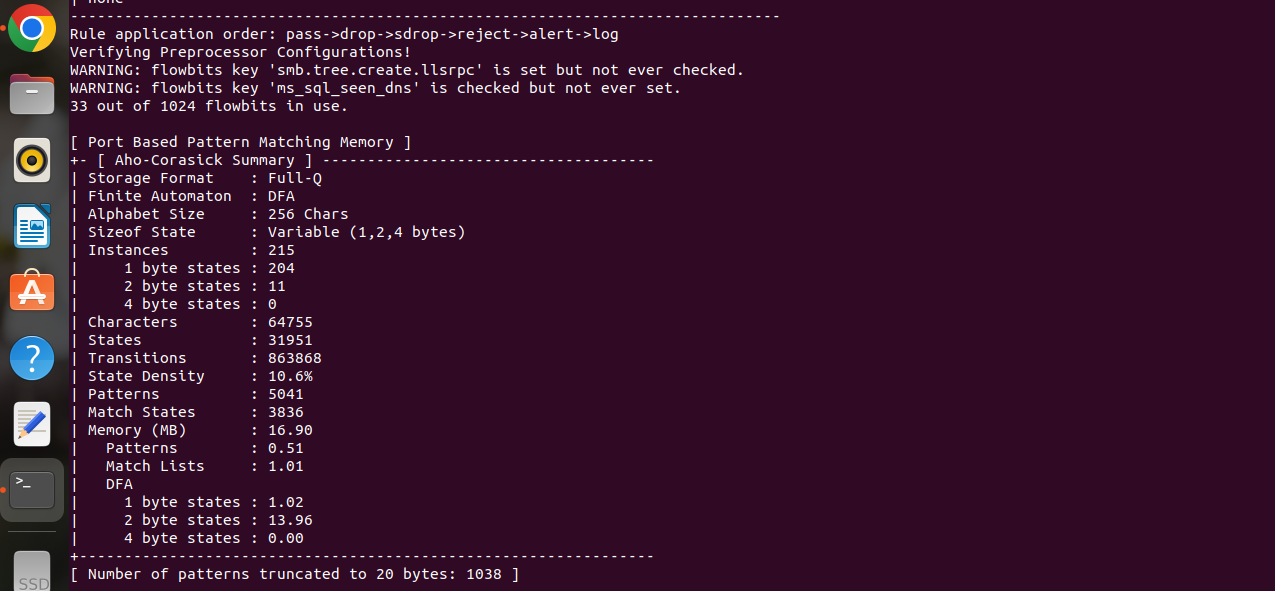
**Snort** is starting up and displaying its **Configuration Settings**, including the **Log Directory, Warnings** about disabled normalizations, and details about its **Fragment Reassembly** and **Session Tracking Preprocessors**. These are standard outputs during Snort's initialization, showing how it’s configured to process and analyze network traffic.



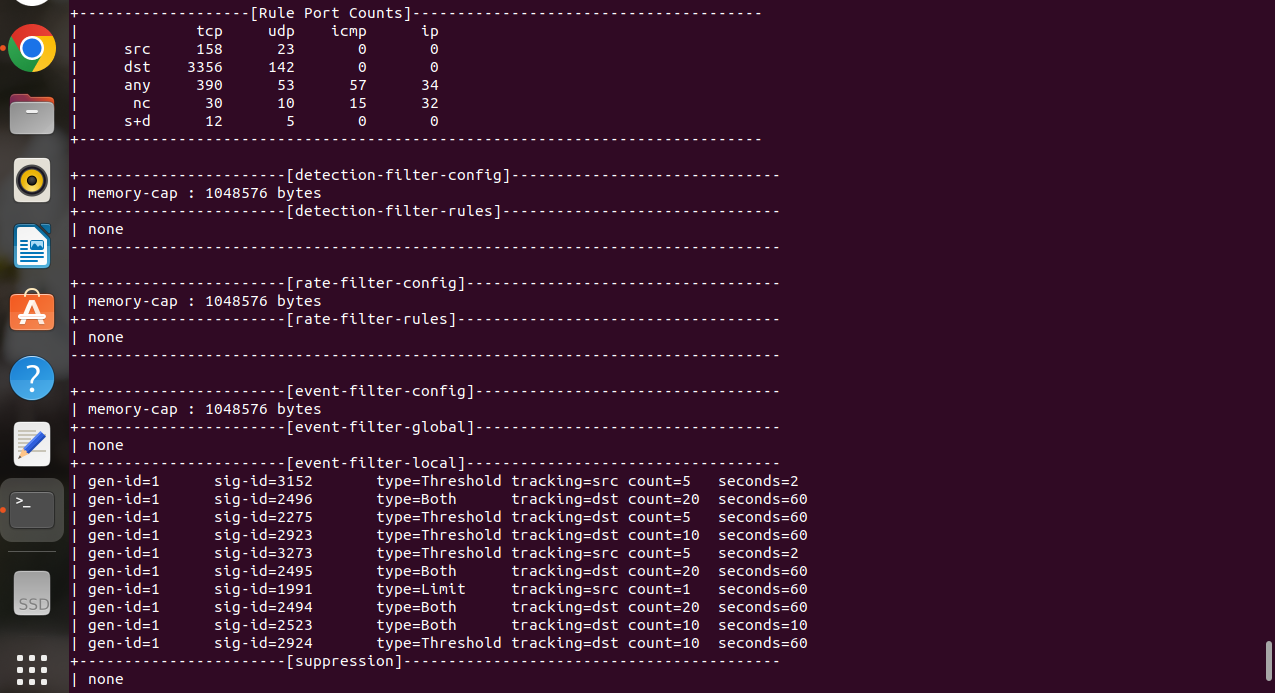
**Snort 3.0** is detailing the configuration settings for **TCP stream reassembly**. It includes information about the reassembly policy set to "**WINDOWS**", timeout settings, and the ports being monitored for client and server traffic referred to as "**Footprint**". The configuration also highlights the requirement for a **3-way TCP handshake** and the **Detection of Anomalies** during the reassembly process. This is essential for Snort's ability to **Detect Complex Attacks** that span multiple packets.



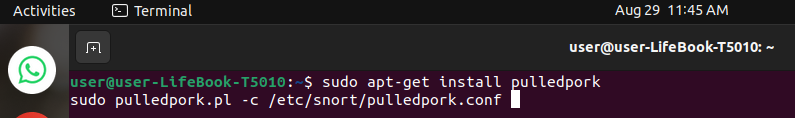
This section of the **Snort** output displays the configuration settings for the **HttpInspect preprocessor**. This module is responsible for normalizing and analyzing **HTTP traffic** to detect potential threats. The output includes **Global Configuration Settings** such as the memory cap for **Logging URI and Hostname**, **GZip Compression Settings**, and **Maximum Session Limits**. It also details the default server profile, including the ports monitored, limits on flow depth, chunk length, header fields, and other HTTP-specific parameters like **Inspecting cookies, responses**, and **JavaScript obfuscation**. These settings help **Snort** in detecting anomalies and potential attacks in **HTTP traffic**.



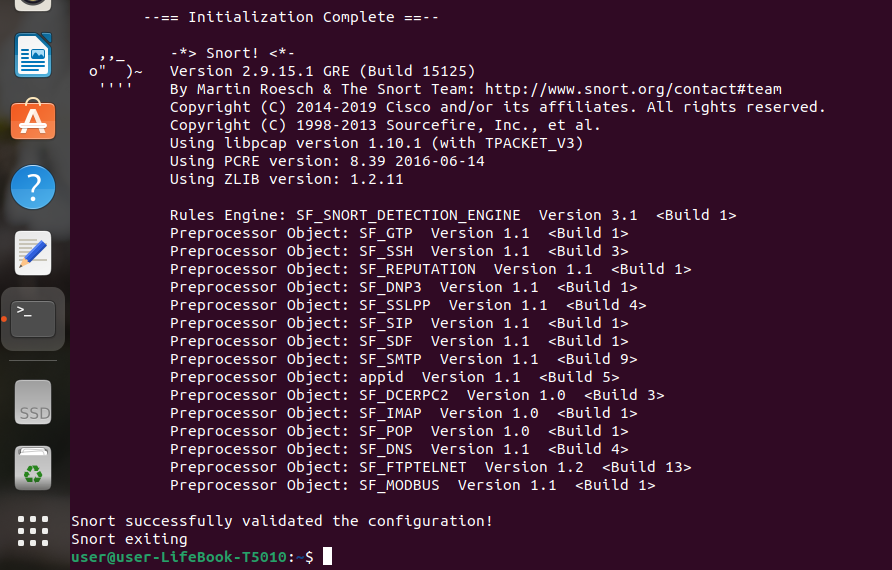
**Snort 3.0** will display information about the **Rule Application Order**, **Flowbit Usage**, and the performance of the port-based pattern matching memory. Specifically, it highlights that some flowbit keys are configured but not used, and provides details about the storage format and performance metrics of the **Aho-Corasick** pattern matching algorithm. The summary table shows the number of patterns, match states, and memory usage, both in total and broken down by state size. The information can be useful for **Network Administrators** to understand the configuration and performance of their firewall as by analyzing this information, administrators can identify potential bottlenecks, optimize performance, and ensure the firewall is configured correctly to meet their security and performance requirements.



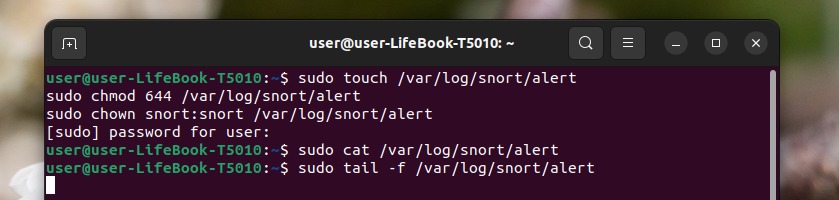
This output information about the **Rule Port Counts,** **Detection Filter**, **Rate Filter**, and **Event Filter Configurations**. Specifically, it shows the number of **TCP** and **UDP packets** for different source and **Destination IP Addresses**, the memory capacity of the **Detection Filter**, and the various event filter rules with their types, tracking parameters, and time intervals. The suppression section is empty, indicating that no suppression rules are currently active. After all this is done **Snort 3.0** displays the version information, copyright details, and the loaded preprocessor objects. The output indicates that **Snort** has successfully validated the configuration and is ready to start **Monitoring Network Traffic** and the **Intrusion Detection Service** offered by **Snort 3.0** will come into action further onwards as observed in the screenshot given on the next page. We can use the following command to keep the Snort 3.0 updated on a regular basis so it keeps up with **Detection** & **Monitoring Network Traffic.** It also keeps in regards of the already handled **Configuration File**



The below screenshot indicates **Snort 3.0** is **Functionally & Fully Configured** & is ready and enabled.



Now **Snort 3.0** will actively running as a **Background Service** and will alert us in any circumstances of an **Intrusion** if detected or any kind of **Anomaly**. We can also **Monitor Logs** & **Alerts** using the following commands if directory of logs isn’t made we can make it using first 3 commands first. After generating some traffic, check the alert log again if everything is set up correctly and **Snort** detects any suspicious activity, it should log the alerts in the specified file.



All of these are vital for security as **Firewalls and Intrusion Detection Systems (IDS) are crucial components of network security.** They act as a barrier, controlling incoming and outgoing network traffic, and detecting suspicious activity. By blocking unauthorized access and preventing data breaches, firewalls and IDS help safeguard networks and protect sensitive information.