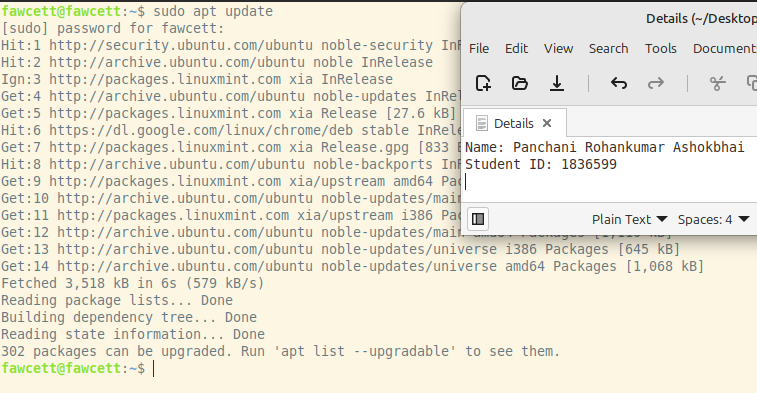
**Cyber Security Task**

**Name: Panchani Rohankumar Ashokbhai**

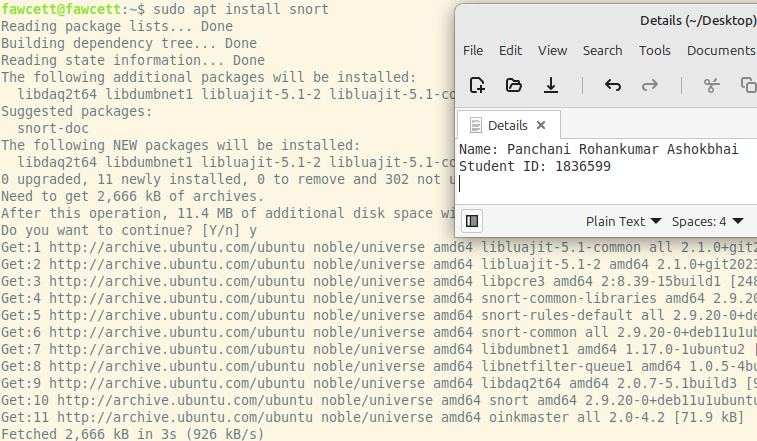
**Student ID: 1836599**

**Objective 1**

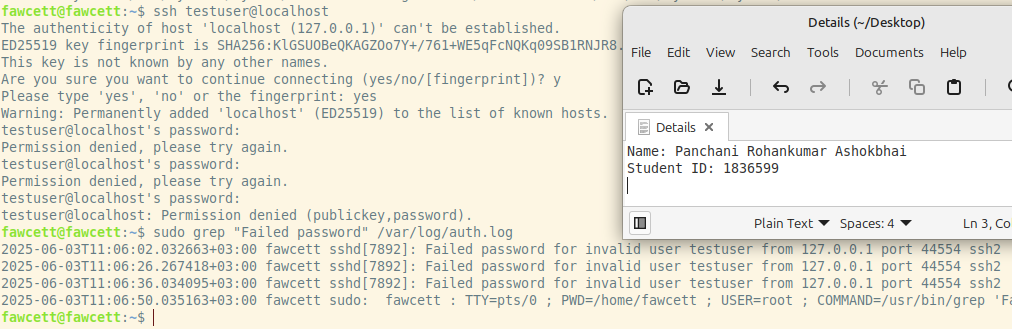
The first step in this task is to complete Objective 1, which involves configuring Snort to alert on more than three failed SSH login attempts within 10 minutes. First, we are going to update the system and install Snort as shown in the screenshot below:



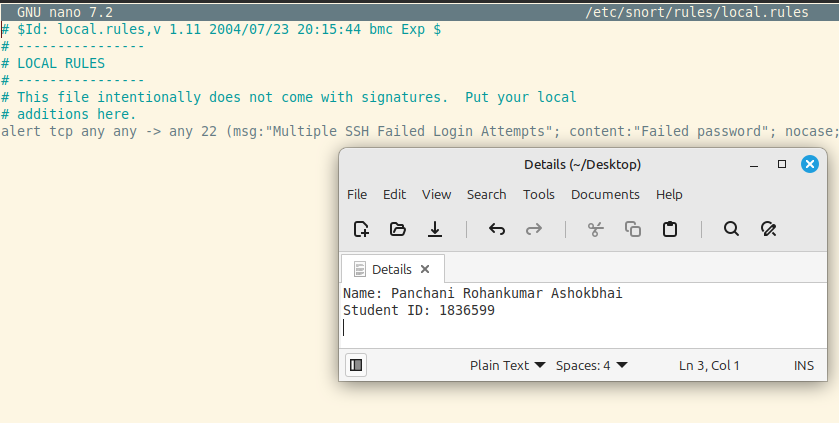
The second step is to install Snort since it does not come pre-installed in the Linux Mint system, as shown below:



In the next step, we identify the failed SSH login pattern:

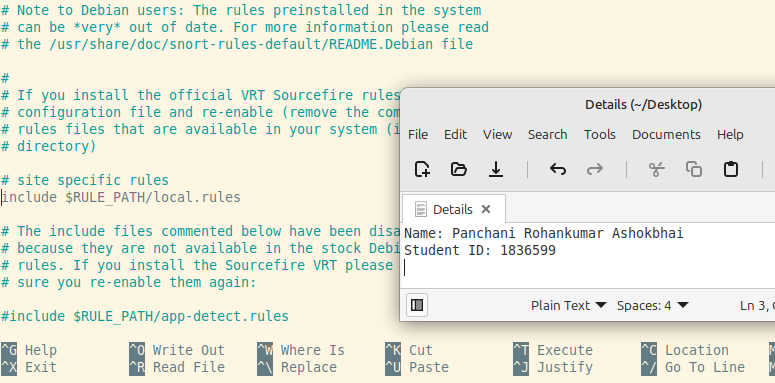


Next, we create a rule with a threshold by editing or creating a local rule:

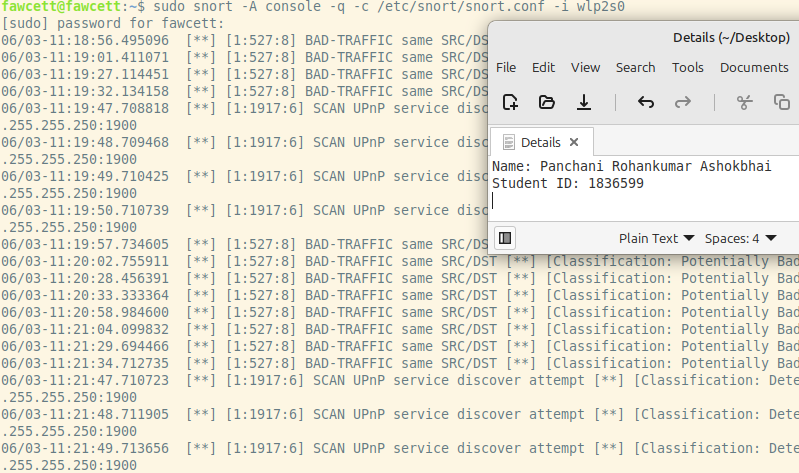


The above rule alerts if 3 matching packets (containing "Failed password") come from the same source IP within 600 seconds (10 minutes).

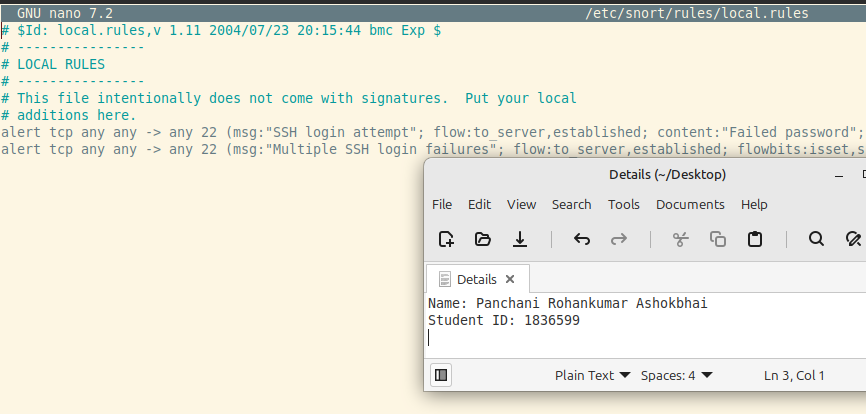
Make sure Snort.conf includes your rule file by ensuring the line for local rules is uncommented:



Snort running successfully



Add new rules:

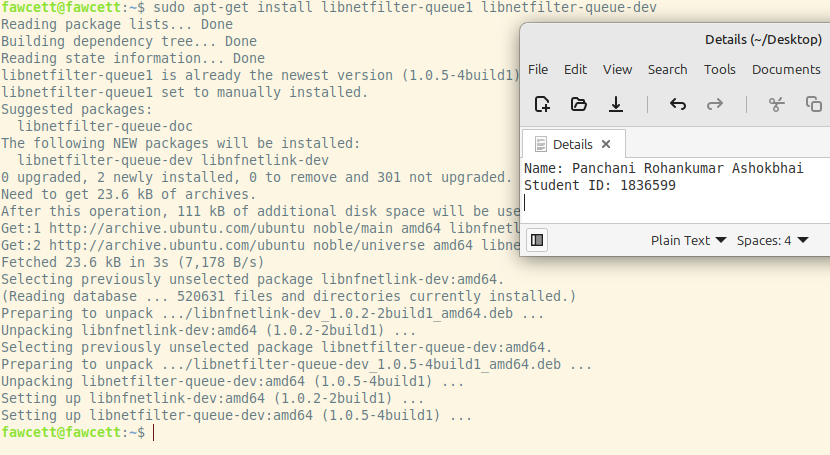


**Objective 2**

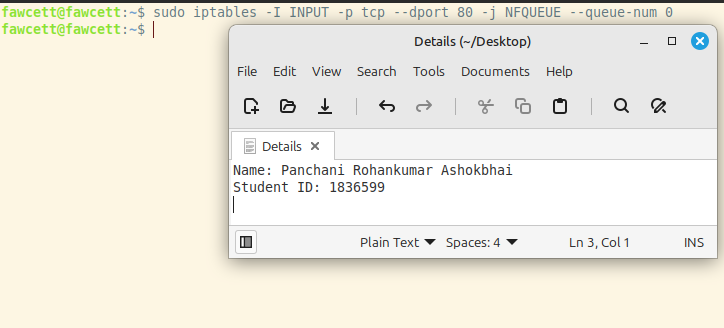
Modify Snort to drop HTTP requests that attempt to access example.com

To prevent users or attackers from accessing the domain example.com, we will configure Snort to inspect HTTP requests and drop any packets that contain a request to example.com.

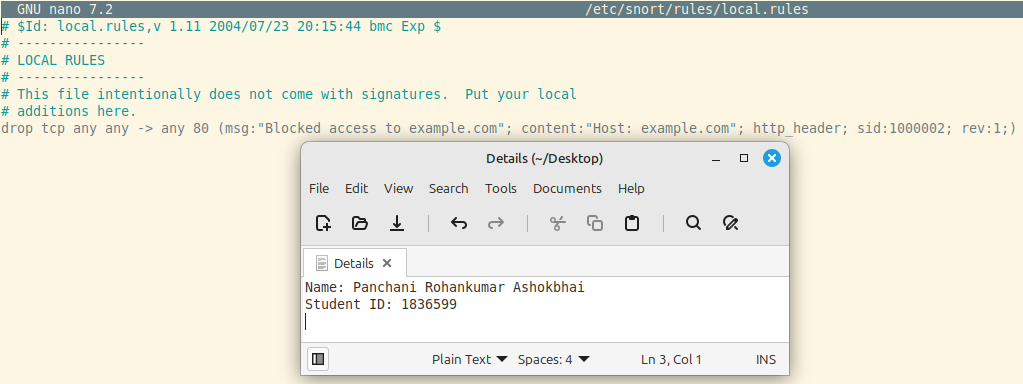
Ensure Snort is Running in Inline Mode. Snort must run in inline (IPS) mode to drop traffic. Install NFQUEUE support (if not already done):



Configure IPtables to Redirect HTTP Traffic to Snort. This allows Snort to inspect packets before they reach their destination.

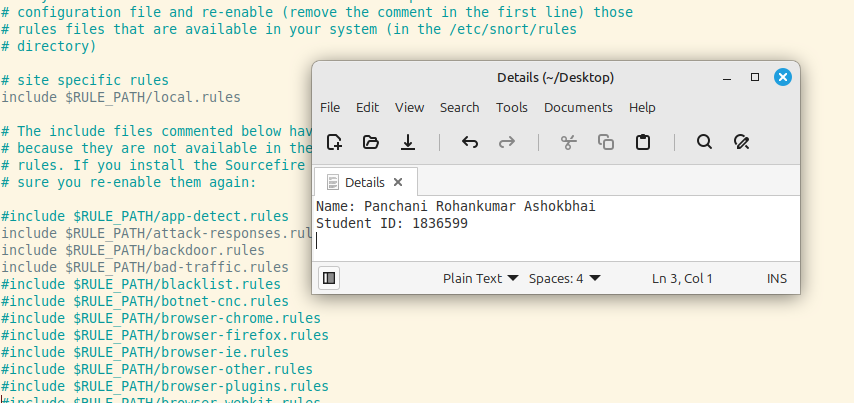


Add a Custom Drop Rule for [example.com](http://example.com) by edit your local.rules file:



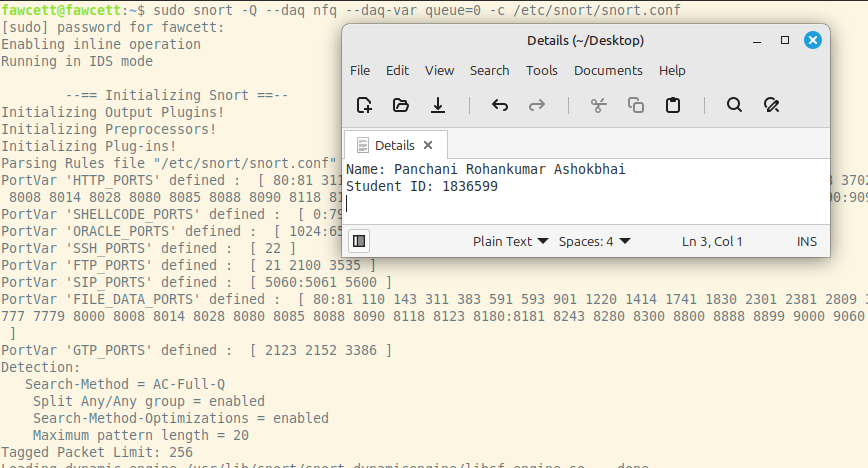
Confirm local.rules is Included in Snort Config

Open the config file:



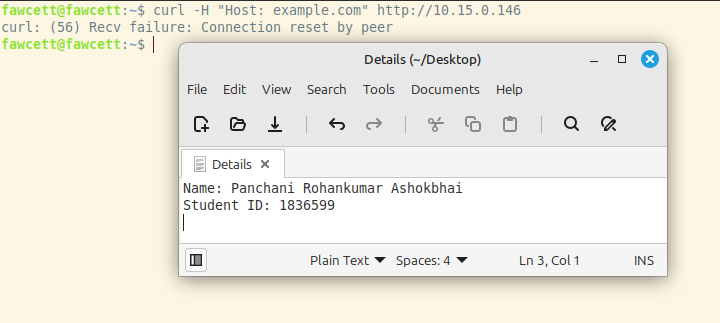
Start Snort in Inline Mode with NFQUEUE

Run Snort like this:



Test the Drop Rule

Open a web browser or use curl:If Snort is dropping it, you'll see:

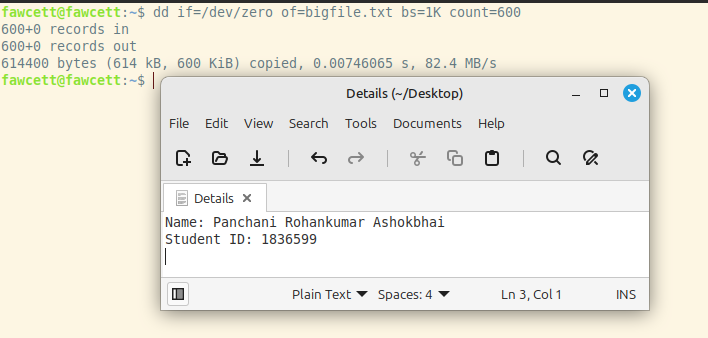


That means your Snort rule to drop example.com is working perfectly!

**Objective 3**

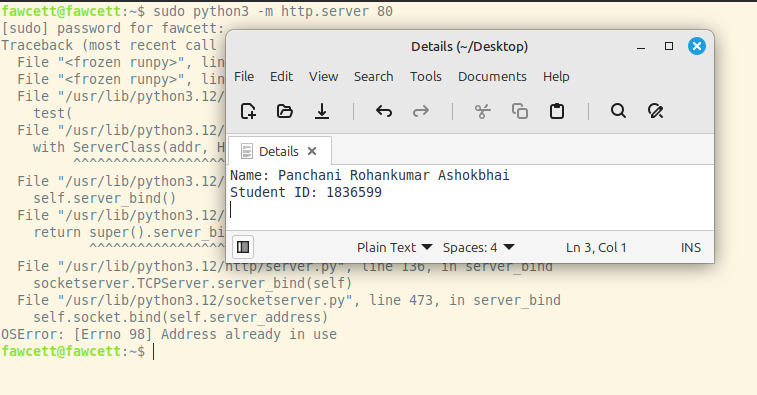
To achieve this objective, we need to configure Snort to inspect HTTP POST requests and raise an alert when the payload size exceeds 500 KB.

Before writing the Snort rule, we need a sample payload that mimics a large file upload. You can use the dd command in Linux to generate a file of any size. Run the following command in the terminal to create a 600KB file filled with zeroes:



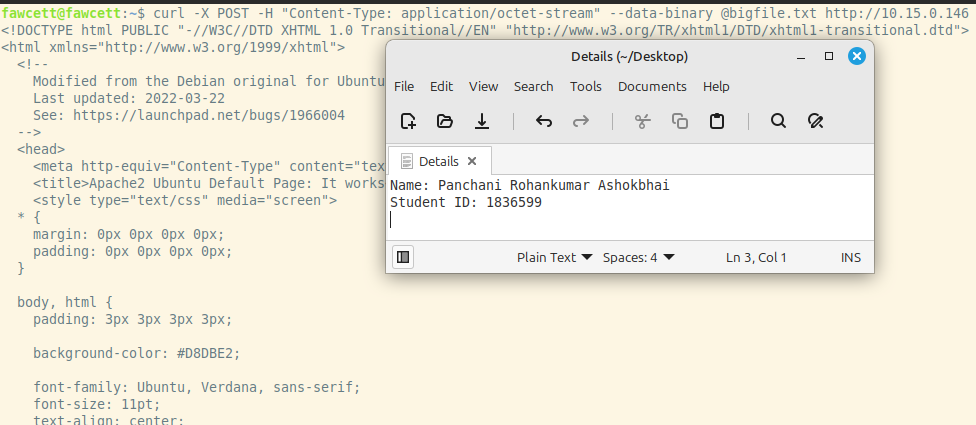
This command tells dd to take zeroes from /dev/zero, write in blocks of 1KB, and repeat 600 times, resulting in a file that is 600KB. This will act as the test payload for our HTTP POST request.

To simulate a real-world upload, we need a web server running on your machine. If you don't already have one, Python makes it easy to set up a basic HTTP server. Use this command to start an HTTP server on port 80:



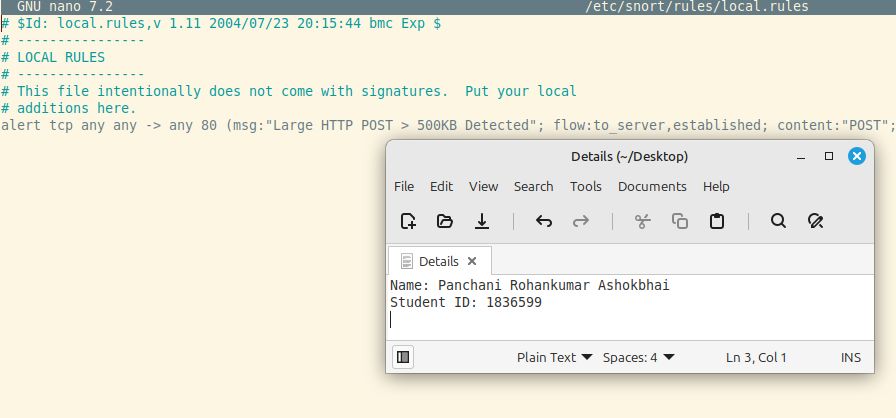
This command starts a web server in your current directory. It must be run with sudo because ports below 1024 require root privileges. The server will listen for incoming connections, including POST requests.

Now, simulate a client uploading a file using the curl command. This command sends an HTTP POST request with the bigfile.txt file as the binary payload:



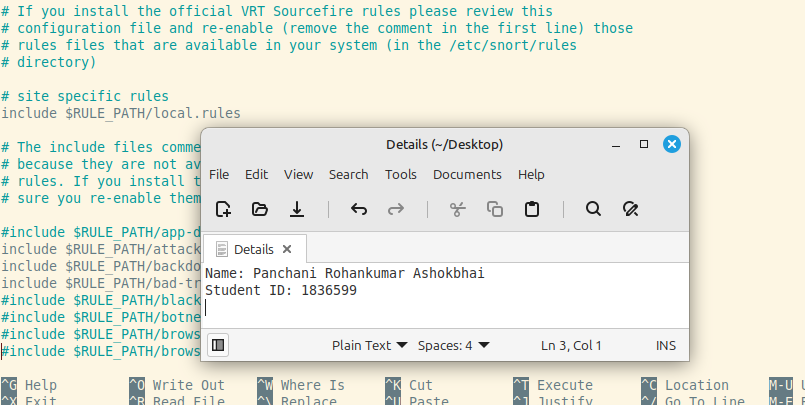
This will trigger an HTTP POST request with a payload size of 600KB, which we want Snort to detect.

Next, I open my Snort rule file (typically located at /etc/snort/rules/local.rules) using a text editor with root permissions:



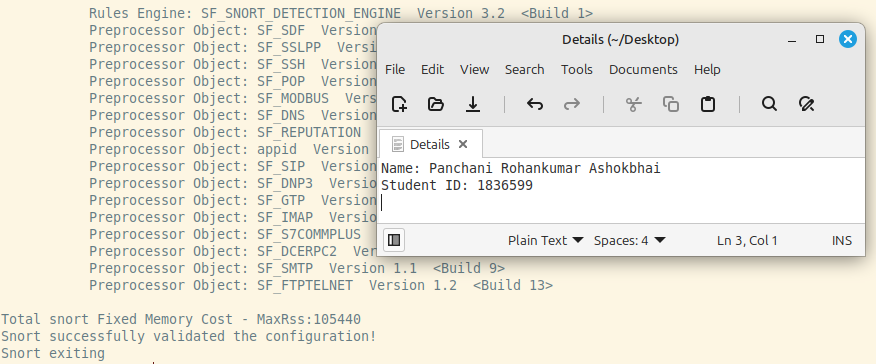
This rule approximates detection of large file uploads via POST by checking packet payload size.

Next, ensure your snort.conf file includes the local.rules file so that Snort loads your custom rule. Open the configuration file:



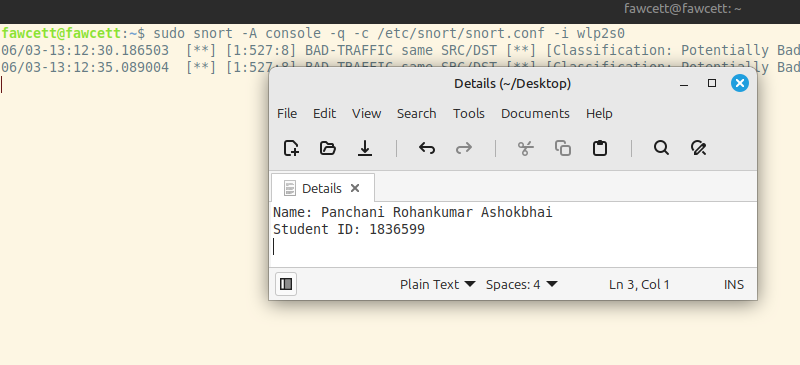
This tells Snort to load and evaluate all rules in the local.rules file when it starts.

Before launching Snort, check for syntax or configuration errors with the following test command:

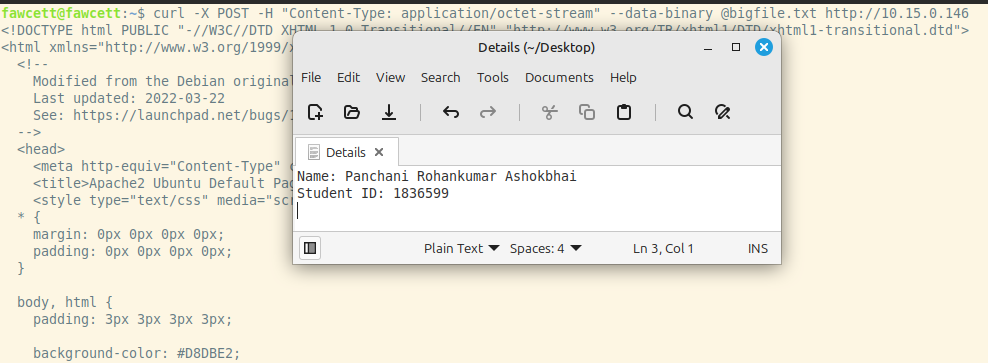


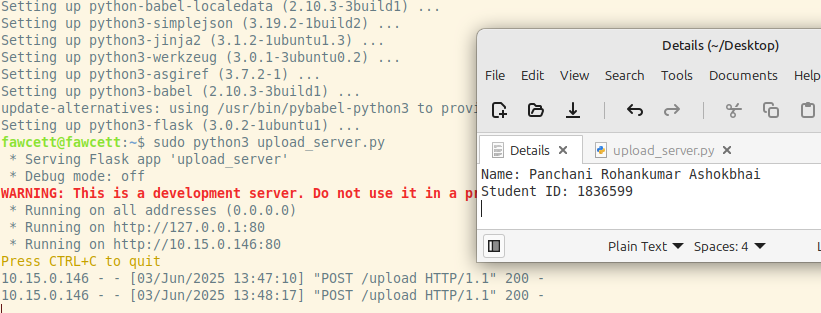
Test passes without errors, we ready to proceed. If you see errors, recheck your rule syntax and file paths.

Now, start Snort on your active network interface (wlp2s0 in my case) in alert mode:

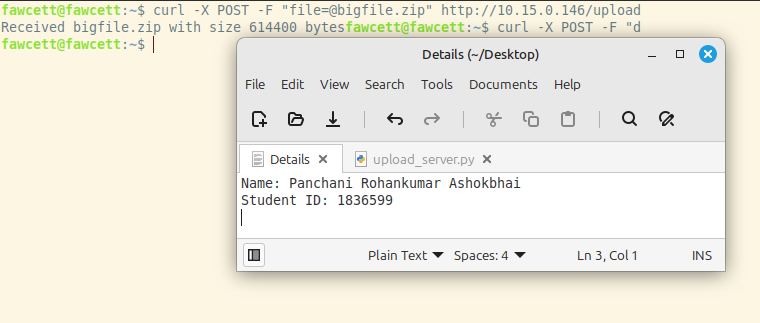


Go back to the other terminal and rerun the curl POST request:

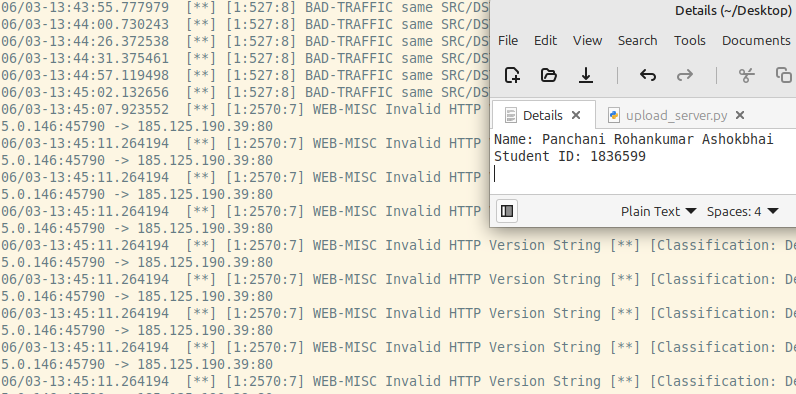




Send file upload

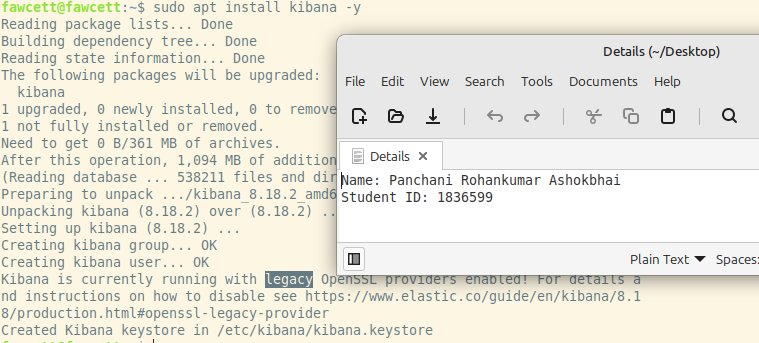


Snort output successful

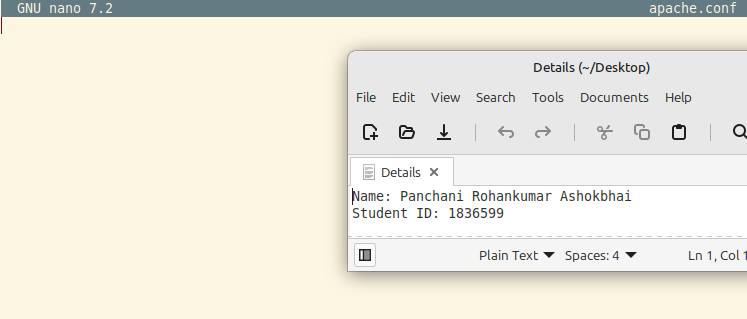


**Objective 7**

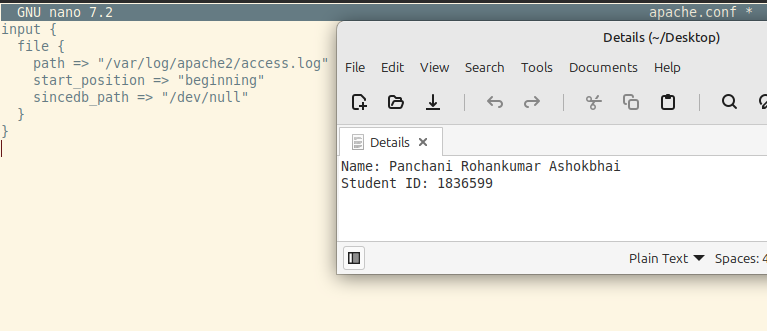
Install the ELK Stack (Elasticsearch, Logstash, and Kibana)



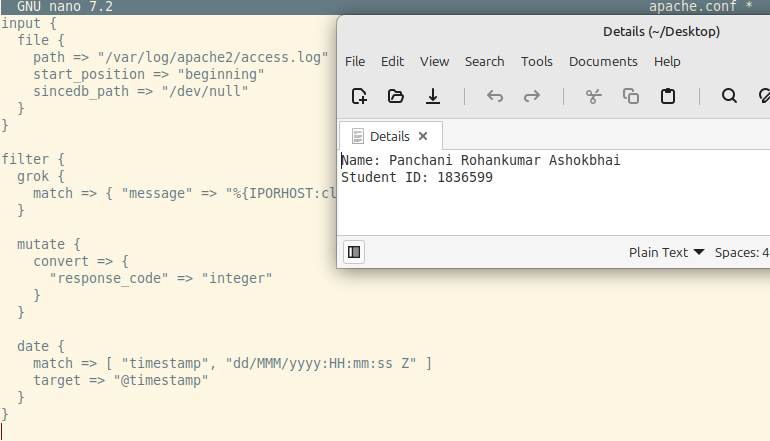
Logstash reads pipeline configuration from files, usually found in /etc/logstash/conf.d/ (Linux). These files define how input data is received, processed (filtered), and sent to the output (Elasticsearch).



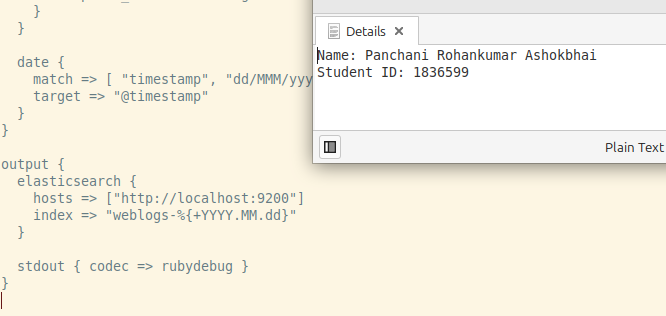
The input block tells Logstash where to read the logs from. You might be reading logs from a file (e.g., access.log), from syslog, or from Filebeat. We'll assume a local Apache access log file



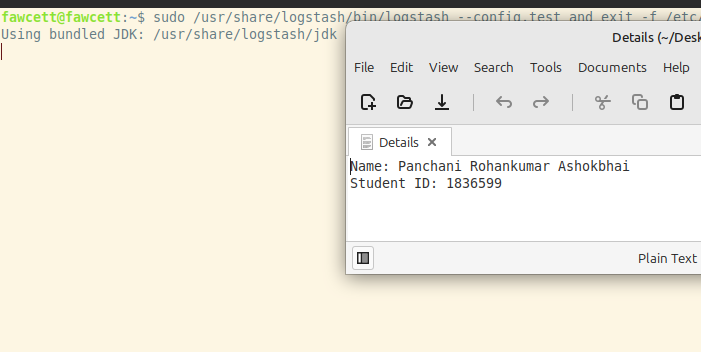
The filter block lets you use grok, which parses unstructured logs using patterns, to extract structured fields like client IP, method, and response code.



The output block tells Logstash where to send the parsed logs. In our case, that’s Elasticsearch, so Kibana can visualize it.



Before restarting Logstash, we validate the configuration to make sure there are no syntax errors.



**Done**