PROJECT: SHADOW SENTRY

Security Operations Center

Table of Contents

Introduction

This study aims to construct an Elastic Cloud server to monitor, detect and analyse malicious activities on a separate client server acting as a honeypot. Two droplets on DigitalOcean were used as a platform to achieve this.

In order to test the functionality of the Security Information and Event Management (SIEM) system, a penetration testing script with three attack vectors was written and executed from an external address against the honeypot server.

To that end, alerts were configured to highlight these attacks when they occur.

Methods

Elastic Cloud Installation

This section details how to install Elasticsearch Logstash Kibana (ELK) v8.13.2 to monitor system event logs and collect them using Beats on the desired servers/nodes [1]. The user is assumed to be a non-root user with superuser permissions.

1. Installing Elasticsearch

As the Elasticsearch components are not available in Ubuntu's default package repositories, using APT to install will produce the following error:

```
root@ubuntu-elk:~# sudo apt install elasticsearch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
E: Unable to locate package elasticsearch
```

Hence, add Elastic's package source list to the '*source.list.d*' directory by importing the public GPG key for the packet manager to trust in case of package spoofing.

curl -fsSL https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/keyrings/elastic.gpg

Install the https package:

sudo apt-get install apt-transport-https

Then inform APT to use the key we downloaded.

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

Download and install Elasticsearch:

sudo apt update && sudo apt install elasticsearch

```
root@ubuntu-elk:~# apt install elasticsearch
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  elasticsearch
0 upgraded, 1 newly installed, 0 to remove and 73 not upgraded.
Need to get 576 MB of archives.
After this operation, 1136 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 elasticsearch amd64 8.13.2 [576 MB]
Fetched 576 MB in 51s (11.2 MB/s)
Selecting previously unselected package elasticsearch.
(Reading database ... 114303 files and directories currently installed.)
Preparing to unpack .../elasticsearch_8.13.2_amd64.deb ...
Creating elasticsearch group... OK
Creating elasticsearch user... OK
Unpacking elasticsearch (8.13.2) ...
Setting up elasticsearch (8.13.2) ...
--------------------------- Security autoconfiguration information ------------------------------

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

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

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

You can complete the following actions at any time:

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

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

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

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

Take note of the generated password for the elastic built-in superuser and save it on a local file. This will be required in subsequent steps.

Edit the main configuration file elasticsearch.yml to match the following:

sudo nano /etc/elasticsearch/elasticsearch.yml

```
# ---------------------------------- Node ------------------------------------
#
# Use a descriptive name for the node:
#
node.name: ubuntu-elk
#
# Add custom attributes to the node:
#
#node.attr.rack: r1
#
# ---------------------------------- 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
#
```

```
# ---------------------------------- 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.
#
# --------------------------------- Discovery ----------------------------------
#
# Pass an initial list of hosts to perform discovery when this node is started:
# The default list of hosts is ["127.0.0.1", "[::1]"]
#
discovery.seed_hosts: ["<ip address>"]
#
# Bootstrap the cluster using an initial set of master-eligible nodes:
#
#cluster.initial_master_nodes: ["node-1", "node-2"]
#
# For more information, consult the discovery and cluster formation module documentation.
#
```

Change the IP address to your own.

```
# Enable security features
xpack.security.enabled: true

xpack.security.enrollment.enabled: true

# Enable encryption for HTTP API client connections, such as Kibana, Logstash, and Agents
xpack.security.http.ssl:
  enabled: true
  keystore.path: certs/http.p12

# Enable encryption and mutual authentication between cluster nodes
xpack.security.transport.ssl:
  enabled: true
  verification_mode: certificate
  keystore.path: certs/transport.p12
  truststore.path: certs/transport.p12
# Create a new cluster with the current node only
# Additional nodes can still join the cluster later
cluster.initial_master_nodes: ["ubuntu-elk"]

# Allow HTTP API connections from anywhere
# Connections are encrypted and require user authentication
http.host: 0.0.0.0
```

By default, Elasticsearch listens for traffic on port 9200. Enable and start the Elasticsearch service.

sudo systemctl start elasticsearch && sudo systemctl enable elasticsearch

Ensure the service is installed properly and able to run smoothly. A similar output should be displayed:

```
root@ubuntu-elk:~# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
     Loaded: loaded (/lib/systemd/system/elasticsearch.service; enabled; preset: enabled)
     Active: active (running) since Wed 2024-04-17 08:14:56 UTC; 10h ago
       Docs: https://www.elastic.co
   Main PID: 767 (java)
      Tasks: 120 (limit: 9476)
     Memory: 4.8G
        CPU: 1h 8min 30.880s
     CGroup: /system.slice/elasticsearch.service
             ├─ 767 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=/usr/share/
             ├─1256 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=10
             └─1317 /usr/share/elasticsearch/modules/x-pack-ml/platform/linux-x86_64/bin/controller

Apr 17 08:13:48 ubuntu-elk systemd[1]: Starting elasticsearch.service - Elasticsearch...
Apr 17 08:14:03 ubuntu-elk systemd-entrypoint[767]: Apr 17, 2024 8:14:03 AM sun.util.locale.provider.LocaleProviderAdapter <clinit>
Apr 17 08:14:03 ubuntu-elk systemd-entrypoint[767]: WARNING: COMPAT locale provider will be removed in a future release
Apr 17 08:14:56 ubuntu-elk systemd[1]: Started elasticsearch.service - Elasticsearch.
```

Elasticsearch has been successfully installed.

Check if Elasticsearch is running as it should with:

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

```
root@ubuntu-elk:~# curl --cacert /etc/elasticsearch/certs/http_ca.crt -u elastic https://localhost:9200
Enter host password for user 'elastic':
{
  "name" : "ubuntu-elk",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "EkSSdDzcRjCTzyFs1-3wdA",
  "version" : {
    "number" : "8.13.2",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "16cc90cd2d08a3147ce02b07e50894bc060a4cbf",
    "build_date" : "2024-04-05T14:45:26.420424304Z",
    "build_snapshot" : false,
    "lucene_version" : "9.10.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

2. Installing Kibana

Now that the necessary components are in place with Elasticsearch installed, install Kibana next.

sudo apt install kibana

```
root@ubuntu-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 73 not upgraded.
Need to get 321 MB of archives.
After this operation, 938 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 kibana amd64 8.13.2 [321 MB]
Fetched 321 MB in 15s (22.1 MB/s)
Selecting previously unselected package kibana.
(Reading database ... 115644 files and directories currently installed.)
Preparing to unpack .../kibana_8.13.2_amd64.deb ...
Unpacking kibana (8.13.2) ...
Setting up kibana (8.13.2) ...
Creating kibana group... OK
Creating kibana user... OK
Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on
uction.html#openssl-legacy-provider
Created Kibana keystore in /etc/kibana/kibana.keystore
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

Proceed to edit the Kibana configuration file kibana.yml to allow connections from remote users and specify the port as 5601.

sudo nano /etc/kibana/kibana.yml

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

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

Setting the server.host IP to “0.0.0.0” – which serves to represent any IPv4 address – enables connections from remote users. Elsewhere ensure the other sections are as follows:

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

You may replace the server name to your own.

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

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

Create an enrolment token to sync with elasticsearch:

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

```
root@ubuntu-elk:~# sudo /usr/share/elasticsearch/bin/elasticsearch-create-enrollment-token -s kibana
eyJ2ZXIiOiI4LjEzLjIiLCJhZHIiOlsiMjA2LjE4OS44OS4zMzo5MjAwIl0sImZnciI6ImMxMGE3OWZmZjdlMWY0NWEwZGU3ZGY4ZG
JUeGVTMG80QnJxRW1oMlFDbmtaNzpQV3lHaDBlclNtNk50ZmMwdVJrVXlBIn0=
```

Next setup Kibana and enter the enrolment token when prompted.

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

```
root@ubuntu-elk:~# sudo /usr/share/kibana/bin/kibana-setup
Native global console methods have been overridden in production environment.
? Enter enrollment token: eyJ2ZXIiOiI4LjEzLjIiLCJhZHIiOlsiMjA2LjE4OS44OS4zMzo
OGM4ZjBmMmYwYjAiLCJrZXkiOiIyb09pNEk0QnhnVjQyU3RxWlVjRzphYk5KbkhqMlNydUJOSWpmY

✔ Kibana configured successfully.

To start Kibana run:
  bin/kibana
```

This will add a few settings to the config file, such as:

```
# This section was automatically generated during setup.
elasticsearch.hosts: ['https://                :9200']
elasticsearch.serviceAccountToken: AAEAAWVsYXN0aWMva2liYW
elasticsearch.ssl.certificateAuthorities: [/var/lib/kibar
```

Then generate the encryption keys for Kibana and copy them to the Kibana configuration file:

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

```
Settings:
xpack.encryptedSavedObjects.encryptionKey: d97e9c4be09cb5e1b641c3751581b69c
xpack.reporting.encryptionKey: f5d2bb0f6cf0281dd7e0f38662c2ef72
xpack.security.encryptionKey: 6f252066a0961ffab8d1b1262f3d805f
```

```
xpack.encryptedSavedObjects.encryptionKey: d97e9c4be09cb5e1b641c3751581b69c
xpack.reporting.encryptionKey: f5d2bb0f6cf0281dd7e0f38662c2ef72
xpack.security.encryptionKey: 6f252066a0961ffab8d1b1262f3d805f
```

Now start and enable the service as previously done for Elasticsearch.

sudo systemctl start kibana && sudo systemctl enable kibana

Check if Kibana is running:

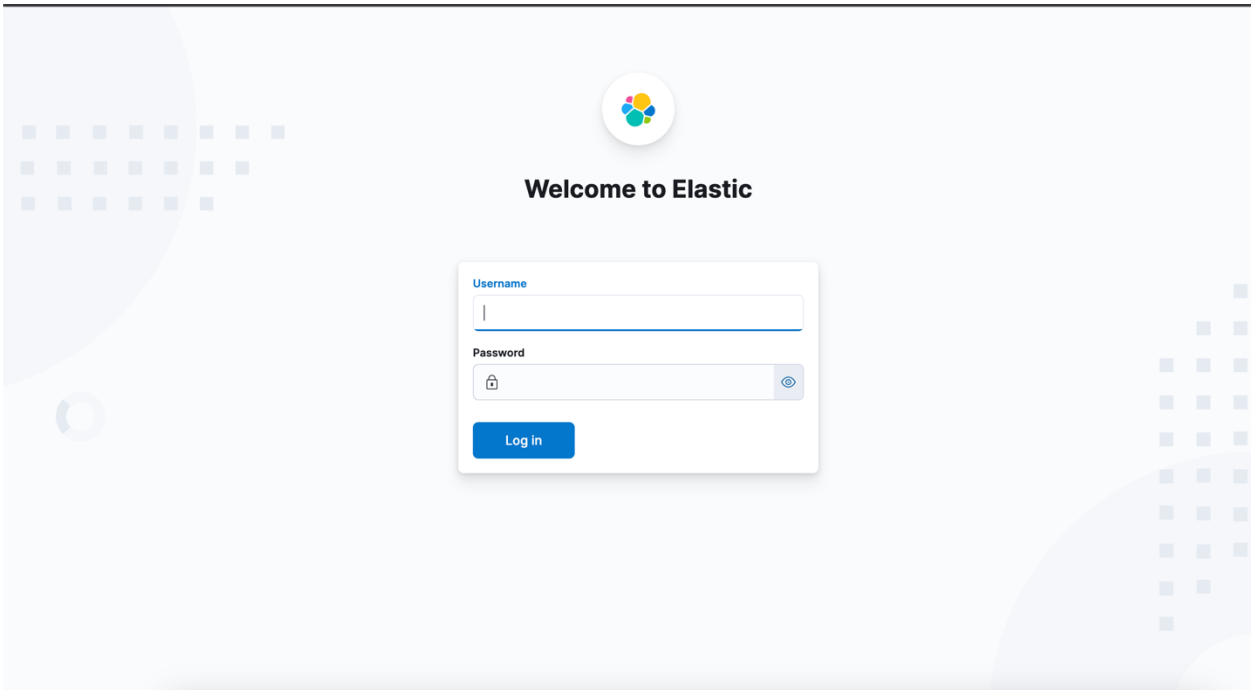
journalctl -u kibana -n 20

```
ubuntu-elk systemd[1]: Started kibana.service - Kibana.
ubuntu-elk kibana[2454]: Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how to d
ubuntu-elk kibana[2454]: {"log.level":"info","@timestamp":"2024-04-15T07:33:19.524Z","log.logger":"elastic-apm-node","ecs.version":"
ubuntu-elk kibana[2454]: Native global console methods have been overridden in production environment.
ubuntu-elk kibana[2454]: [2024-04-15T07:33:21.542+00:00][INFO ][root] Kibana is starting
ubuntu-elk kibana[2454]: [2024-04-15T07:33:21.686+00:00][INFO ][node] Kibana process configured with roles: [background_tasks, ui]
ubuntu-elk kibana[2454]: [2024-04-15T07:33:35.776+00:00][INFO ][plugins-service] The following plugins are disabled: "cloudChat,clou
ubuntu-elk kibana[2454]: [2024-04-15T07:33:35.887+00:00][INFO ][http.server.Preboot] http server running at http://0.0.0.0:5601
ubuntu-elk kibana[2454]: [2024-04-15T07:33:36.117+00:00][INFO ][plugins-system.preboot] Setting up [1] plugins: [interactiveSetup]
```

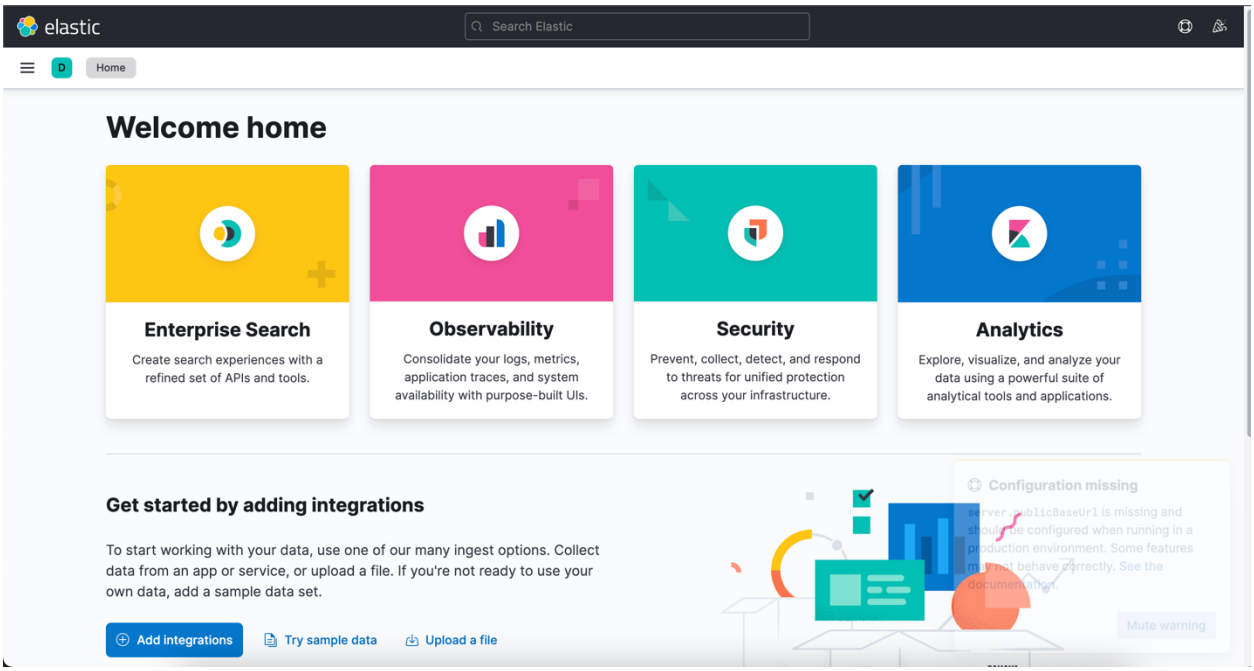
Test if the configuration is successful by accessing the Kibana dashboard:

http://<IP address>:5601

You will be prompted to enter the username and password. Use ‘elastic’ as the username and enter the elastic superuser password previously saved from the installation output of Elasticsearch.



If the credentials used are correct, you will enter the Kibana dashboard.



3. Installing Logstash

With the Kibana dashboard configured and accessible, install Logstash to collect the data from other sources, process them, and send them to Elasticsearch.

sudo apt install logstash

```
root@ubuntu-elk:~# apt install logstash
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  logstash
0 upgraded, 1 newly installed, 0 to remove and 73 not upgraded.
Need to get 404 MB of archives.
After this operation, 668 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 logstash amd64 1:8.13.2-1 [404 MB]
Fetched 404 MB in 1min 32s (4373 kB/s)
Selecting previously unselected package logstash.
(Reading database ... 205925 files and directories currently installed.)
Preparing to unpack .../logstash_1%3a8.13.2-1_amd64.deb ...
Unpacking logstash (1:8.13.2-1) ...
Setting up logstash (1:8.13.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.
```

To encrypt communications between client and server machines, we will need to create the necessary certificates and keys. First, create a temporary directory and lockdown access to it:

sudo mkdir certgen

sudo chmod 700 certgen

cd certgen

When the Elasticsearch 8.x deb package was installed, certificate authority (CA) keys and their respective certificates were generated automatically.

```
root@ubuntu-elk:~/certgen# ls /etc/elasticsearch/certs
http.p12  http_ca.crt  transport.p12
```

The CA cert used for HTTP encryption is identified as 'http_ca.crt', while the corresponding key is 'http.p12'. To retrieve the password of the p12 file from the Elastic keystore, use:

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

Creating the Logstash certificates and keys from the http.p12 file directly will produce an error. Hence in order to extract the HTTP CA key to place it in its own p12 file, dump the keys to a PEM file. When prompted for a password, simply enter the one obtained from the Elastic keystore.

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

Open the PEM file using any text editor. There will be two keys inside, but only '*http_ca'* is needed.

```
Bag Attributes
    friendlyName: http_ca
    localKeyID: 54 69 6D 65 20 31 37 31 33 31 36 34 34 35 36 31 30 30
Key Attributes: <No Attributes>
-----BEGIN PRIVATE KEY-----
```

```
-----END PRIVATE KEY-----
Bag Attributes
    friendlyName: http
    localKeyID: 54 69 6D 65 20 31 37 31 33 31 36 34 34 35 36 31 32 3
Key Attributes: <No Attributes>
```

Delete the lines until only the http_ca key is remaining.

```
-----END PRIVATE KEY-----
```

Package the key into its own p12 file. When prompted for a password, it is possible to reuse the same password for the 'http.p12' file.

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

Generate a key-certificate pair for the Logstash instance, replacing the file path and the IP address with your server's own.

/usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca /full/file/path/http_ca_key.p12 --ip <ip address> --name logstash --out /full/file/path/logstash.p12

Extract the key and certificates from logstash.p12 in PEM format:

openssl pkcs12 -in logstash.p12 -nocerts -nodes -out logstash.key

openssl pkcs12 -in logstash.p12 -nokeys -nodes -out logstash.crt

Because '*logstash.crt*' contains both the CA cert and Logstash cert, remove the CA cert using a text editor like before.

```
Bag Attributes
    friendlyName: logstash
    localKeyID: 54 69 6D 65 20 31 37 31 33 31 37 32 35 31 37 32 38 37
subject=CN = logstash
issuer=CN = Elasticsearch security auto-configuration HTTP CA
```

```
Bag Attributes
    friendlyName: ca
    2.16.840.1.113894.746875.1.1: <Unsupported tag 6>
subject=CN = Elasticsearch security auto-configuration HTTP CA
issuer=CN = Elasticsearch security auto-configuration HTTP CA
```

Copy the '*logstash.crt*', '*logstash.key*' and HTTP CA cert to the Logstash cert directory.

sudo cp logstash.crt /etc/logstash/certs/

sudo cp logstash.key /etc/logstash/certs/

cp /etc/elasticsearch/certs/http_ca.crt /etc/logstash/certs

Fix permissions for the Logstash directory to ensure the Logstash user can read its contents:

sudo chown -R root:logstash /etc/logstash

sudo chmod -R g+rX /etc/logstash

sudo chmod -R o-rwx /etc/logstash

sudo find /etc/logstash -type d -exec chmod g+s {} \;

Additionally you may also further adjust permissions of these directories if you encounter permission issues when running Logstash:

sudo chown logstash:logstash /etc/default/logstash

sudo chown -R logstash:logstash /usr/share/logstash

sudo chmod 777 /usr/share/logstash/data

Create a certificate pair for the client server this time, specifying the IP of the client instead:

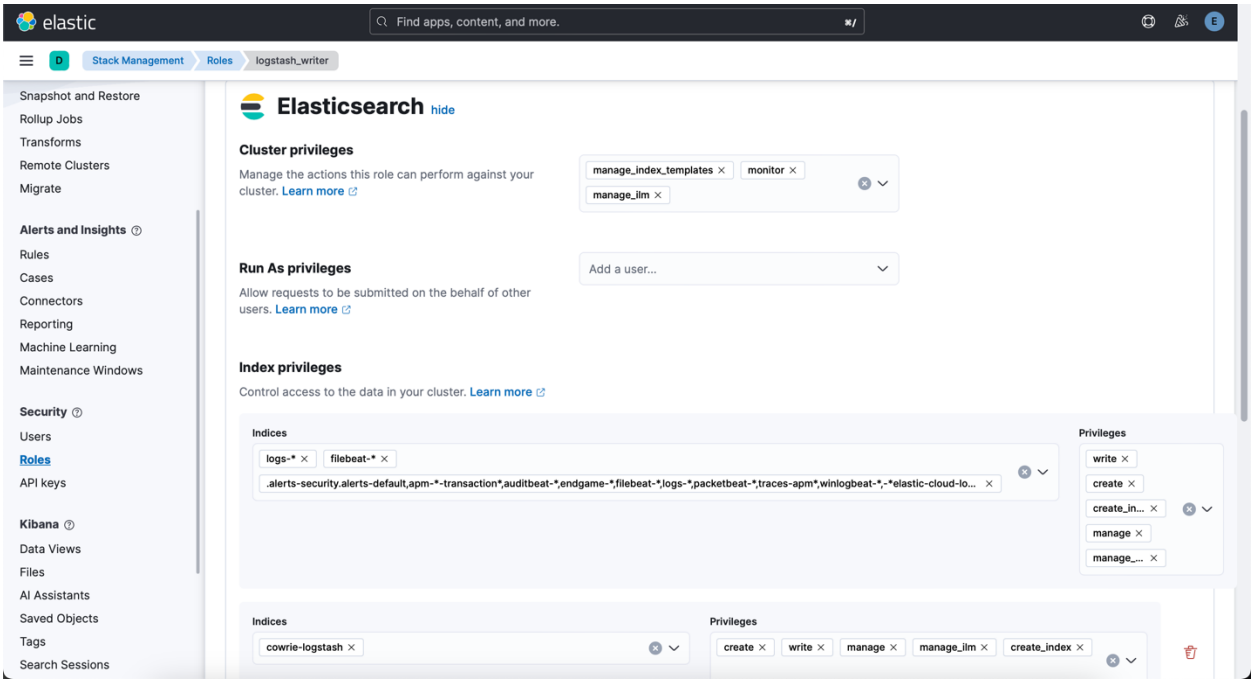
/usr/share/elasticsearch/bin/elasticsearch-certutil cert --ca /full/file/path/certgen/http_ca_key.p12 --ip <client ip> --name client --out /full/file/path/certgen/client.p12

To use basic authentication for communication between Logstash and Elasticsearch, access the Kibana webpage and navigate to '*Management -> Roles'*. Click on '*Create Role*'. Then fill in the fields as follows:

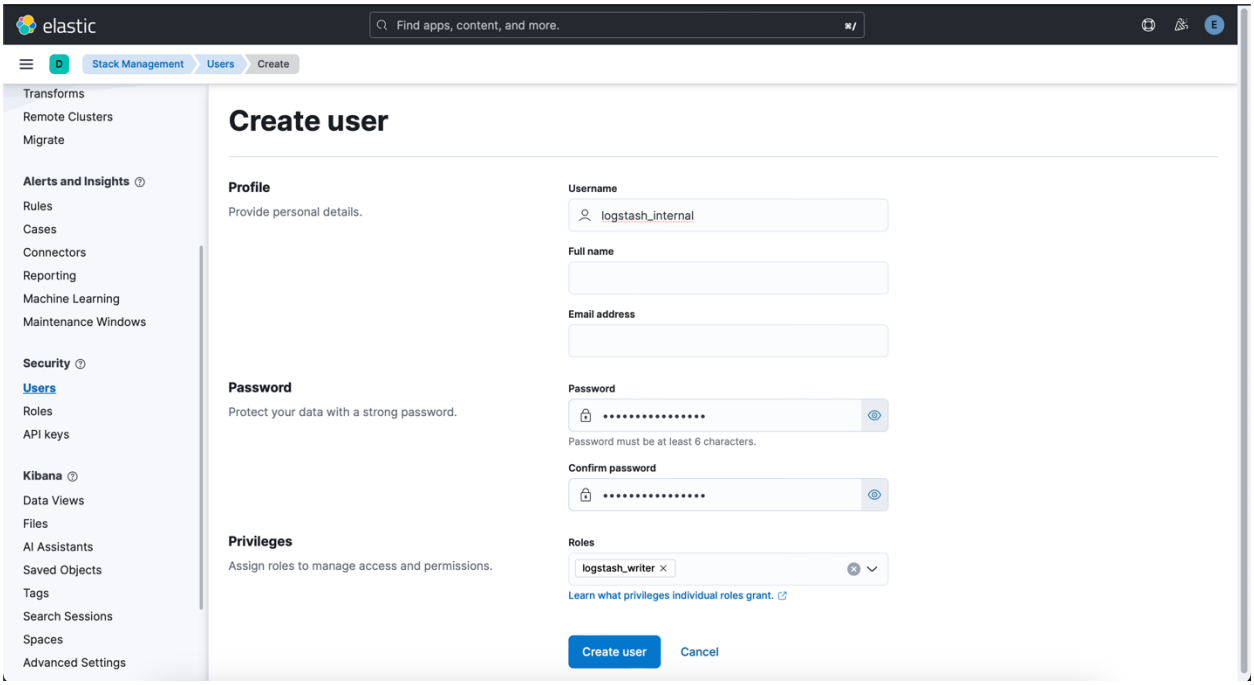
Role name: logstash_writer

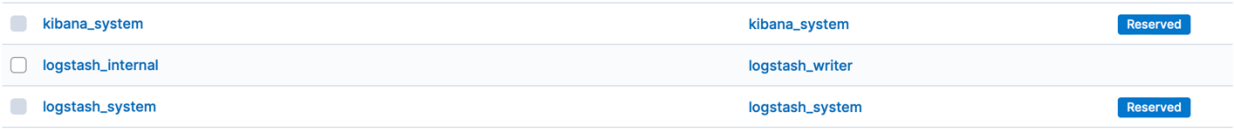
Cluster privileges: manage_index_templates, monitor, manage_ilm

Index privileges: write, create, create_index, manage, manage_ilm



Now under '*Management -> Users'*, create a new user named '*logstash_internal*' and assign it the '*logstash_writer*' role that was just created.





Ensure the following settings for the Logstash config file:

```
# ------------ Data path ------------------
#
# Which directory should be used by logstash and its plugins
# for any persistent needs. Defaults to LOGSTASH_HOME/data
#
path.data: /var/lib/logstash
```

```
# ------------ Debugging Settings --------------
#
# Options for log.level:
#   * fatal
#   * error
#   * warn
#   * info (default)
#   * debug
#   * trace
# log.level: info
#
# Options for log.format:
#   * plain (default)
#   * json
#
# log.format: plain
path.logs: /var/log/logstash
#
```

```
# X-Pack GeoIP Database Management
# https://www.elastic.co/guide/en/logstash/current/plugins-filters-geoip.
xpack.geoip.downloader.enabled: true
xpack.geoip.downloader.endpoint: "https://geoip.elastic.co/v1/database"
```

No changes were made to the '*pipeline.yml*' file, but you may choose to configure it to your needs:

```
# 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/*.conf"
```

Create a configuration file to forward the logs from the client machine to Elasticsearch.

sudo nano /etc/logstash/conf.d/beats.conf

Alternatively you may create a config file for each of the '*input*', '*filter*' and '*output*' settings.

Replace the host IP address with your own and use the password you set for '*logstash_internal*' earlier.

```
input {
  beats {
    port => 5044
    ssl => true
    ssl_certificate_authorities => ["/etc/logstash/certs/http_ca.crt"]
    ssl_certificate => "/etc/logstash/certs/logstash.crt"
    ssl_key => "/etc/logstash/certs/logstash.key"
    ssl_verify_mode => "force_peer"
  }
}
filter {
  if [event][module] == "apache" {
    if [fileset][name] == "access" {
      grok {
        match => { "message" => ["%{IPORHOST:[source][address]} - %{DATA:[user][name]} \[%{HTTPDATE:[apache][acc>
        "%{IPORHOST:[source][address]} - %{DATA:[user][name]} \\[%{HTTPDATE:[apache][access][time]}\\] \"-\" %{N>
        "\\[%{HTTPDATE:[apache][access][time]}\\] %{IPORHOST:[source][address]} %{DATA:[apache][access][ssl][pro>
        ] }
        remove_field => "message"
      }
      grok {
        match => { "[source][address]" => ["^(%{IP:[source][ip]}|%{HOSTNAME:[source][domain]})$"]}
      }
      mutate {
        rename => { "[event][created]" => "@timestamp" }
      }
      date {
        match => [ "[apache][access][time]", "dd/MMM/yyyy:H:m:s Z" ]
        remove_field => "[apache][access][time]"
      }
      useragent {
        source => "[user_agent][original]"
      }
      geoip {
        source => "[source][ip]"
        target => "[source][geo]"
      }
    }
    else if [fileset][name] == "error" {
      grok {
        match => {"message" => ["\[%{APACHE_TIME:[apache][error][timestamp]}\] \[%{LOGLEVEL:[log][level]}\] ( \[>
```

```
    else if [fileset][name] == "error" {
      grok {
        match => {"message" => ["\[%{APACHE_TIME:[apache][error][timestamp]}\] \[%{LOGLEVEL:[log][level]}\] ( \[>
            "\[%{APACHE_TIME:[apache][error][timestamp]}\] \[%{DATA:[apache][error][module]}:%{LOGLEVEL:[log][le>
        ]}
        pattern_definitions => {
          "APACHE_TIME" => "%{DAY} %{MONTH} %{MONTHDAY} %{TIME} %{YEAR}"
        }
        remove_field => "message"
      }
      date {
        match => [ "[apache][error][timestamp]", "EEE MMM dd H:m:s YYYY", "EEE MMM dd H:m:s.SSSSSS YYYY" ]
        remove_field => "[apache][error][timestamp]"
      }
      grok {
        match => { "[source][address]" => ["^(%{IP:[source][ip]}|%{HOSTNAME:[source][domain]})$"]}
      }
      geoip {
        source => "[source][ip]"
        target => "[source][geo]"
      }
    }
  }
  date {
    match => ["timestamp", "MMM dd HH:mm:ss"]
  }
```

A filter was added to parse the Apache module '*message*' outputs into separate categories [2].

```
output {
  if [@metadata][pipeline] {
        elasticsearch {
        ssl => true
        hosts => ["https://<ip address>:9200"]
        cacert => "/etc/logstash/certs/http_ca.crt"
        user => "logstash_internal"
        password => "<password>"
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        pipeline => "%{[@metadata][pipeline]}"
        }
  } else {
        elasticsearch {
        ssl => true
        hosts => ["https://<ip address>:9200"]
        cacert => "/etc/logstash/certs/http_ca.crt"
        user => "logstash_internal"
        password => "<password>"
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        }
  }
}
```

An index setting was included to create an index for Filebeat and direct its output there [3].

Once done, test the Logstash configuration using:

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

After a short period of time, the output result will display the following at the end:

```
[INFO ][logstash.runner          ] Using config.test_and_exit mode. Config Validation Result: OK. Exiting Logstash
```

If this result is not obtained, parse the output for any errors specified and correct them in the configuration files. Also ensure that the spacing and tabs used are correct for the file to be read accurately.

For greater details in the log outputs, test the config file(s) and reload the test automatically with:

sudo -u logstash /usr/share/logstash/bin/logstash -f /etc/logstash/conf.d/*.conf --config.reload.automatic

Start and enable Logstash, and ensure it is running:

sudo systemctl enable logstash && sudo systemctl start logstash

journalctl -u logstash -n 20

```
[WARN ][logstash.outputs.elasticsearch][main] Restored connection to ES instance {:url=>"https://>
[INFO ][logstash.outputs.elasticsearch][main] Elasticsearch version determined (8.13.2) {:es_vers>
[WARN ][logstash.outputs.elasticsearch][main] Detected a 6.x and above cluster: the `type` event >
[INFO ][logstash.outputs.elasticsearch][main] Not eligible for data streams because config contai>
[INFO ][logstash.outputs.elasticsearch][main] Data streams auto configuration (`data_stream => au>
[WARN ][logstash.filters.grok    ][main] ECS v8 support is a preview of the unreleased ECS v8, an>
[INFO ][logstash.javapipeline    ][main] Starting pipeline {:pipeline_id=>"main", "pipeline.worke>
[INFO ][logstash.javapipeline    ][main] Pipeline Java execution initialization time {"seconds"=>>
[INFO ][logstash.inputs.beats    ][main] Starting input listener {:address=>"0.0.0.0:5044"}
[INFO ][logstash.javapipeline    ][main] Pipeline started {"pipeline.id"=>"main"}
[INFO ][logstash.agent           ] Pipelines running {:count=>1, :running_pipelines=>[:main], :no>
[INFO ][org.logstash.beats.Server][main][a18876b0ad9c6d2767e127eb66c4a9205f977fa95d4d30cab5daaf07>
r/bundle/jruby/3.1.0/gems/manticore-0.9.1-java/lib/manticore/client.rb:534: warning: already init>
```

4. Installing Filebeat

To collect logs from a specific server, Beats should be installed in it. Thus access the intended target server to install Beats. For this tutorial, Filebeat is used to collect system logs.

First, repeat the steps to gain access to the Elasticsearch repositories in Step 1 before installing Beats with the following command:

sudo apt install filebeat

```
root@ubuntu-honey:~# sudo apt install filebeat
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  filebeat
0 upgraded, 1 newly installed, 0 to remove and 27 not upgraded.
Need to get 50.9 MB of archives.
After this operation, 188 MB of additional disk space will be used.
Get:1 https://artifacts.elastic.co/packages/8.x/apt stable/main amd64 filebeat amd64 8.13.2 [50.9 MB]
Fetched 50.9 MB in 9s (5686 kB/s)
Selecting previously unselected package filebeat.
(Reading database ... 124010 files and directories currently installed.)
Preparing to unpack .../filebeat_8.13.2_amd64.deb ...
Unpacking filebeat (8.13.2) ...
Setting up filebeat (8.13.2) ...
Scanning processes...
Scanning candidates...
Scanning linux images...

Running kernel seems to be up-to-date.
```

Copy the '*client.p12*' file generated in the previous step to the client server. For example:

sudo scp root@<ip address>:/path/to/file/client.p12 .

Extract the key and certificates from the file:

openssl pkcs12 -in client.p12 -nocerts -nodes -out client.key

openssl pkcs12 -in client.p12 -nokeys -nodes -out client.crt

The '*client.crt'* file contains both the CA and client certificates. Separate the certificates by making a copy of it and renaming it as '*http_ca.crt*'. Then delete the CA certificate in the '*client.crt*' file and vice versa.

```
Bag Attributes
    friendlyName: client
    localKeyID: 54 69 6D 65 20 31 37 31 33 31 37 34 33 37 39 32 31 36
subject=CN = client
issuer=CN = Elasticsearch security auto-configuration HTTP CA
```

```
Bag Attributes
    friendlyName: ca
    2.16.840.1.113894.746875.1.1: <Unsupported tag 6>
subject=CN = Elasticsearch security auto-configuration HTTP CA
issuer=CN = Elasticsearch security auto-configuration HTTP CA
```

Copy the key and certificates to Filebeat.

sudo cp client.key client.crt http_ca.crt /etc/filebeat/certs

Then edit the Filebeat configuration file as follows:

sudo nano /etc/filebeat/filebeat.yml

```
# ============================== Filebeat inputs ===============================

filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input-specific configurations.

# filestream is an input for collecting log messages from files.
- type: filestream

  # Unique ID among all inputs, an ID is required.
  id: my-filestream-id

  # Change to true to enable this input configuration.
  enabled: false

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

```
# ============================== Filebeat modules ==============================

filebeat.config.modules:
  # Glob pattern for configuration loading
  path: ${path.config}/modules.d/*.yml

  # Set to true to enable config reloading
  reload.enabled: false

  # Period on which files under path should be checked for changes
  #reload.period: 10s

# ======================= Elasticsearch template setting =======================

setup.template.settings:
  index.number_of_shards: 1
  #index.codec: best_compression
  #_source.enabled: false
```

```
# ================================== Kibana ====================================

# Starting with Beats version 6.0.0, the dashboards are loaded via the Kibana API.
# This requires a Kibana endpoint configuration.
setup.kibana:

  # Kibana Host
  # Scheme and port can be left out and will be set to the default (http and 5601)
  # In case you specify and additional path, the scheme is required: http://localhost:5601/
  # IPv6 addresses should always be defined as: https://[2001:db8::1]:5601
  host: "                :5601"
```

Take note to change the password to your own elastic superuser's in the 'output.elasticsearch' section:

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

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

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

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

# ------------------------------ Logstash Output -------------------------------
#output.logstash:
  # The Logstash hosts
 # hosts: ["206.189.89.33:5044"]

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

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

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

  #pipelining: 4
```

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

```
# ================================== Logging ===================================

# Sets log level. The default log level is info.
# Available log levels are: error, warning, info, debug
#logging.level: debug
```

Test the configuration with the following:

sudo filebeat test config

```
root@ubuntu-honey:~# filebeat test config
Config OK
```

sudo filebeat test output

```
root@ubuntu-honey:~# filebeat test output
elasticsearch: https://              :9200...
  parse url... OK
  connection...
    parse host... OK
    dns lookup... OK
    addresses:
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
  version: 8.13.2
```

Enable the following modules to monitor syslog, auth log, apache access logs and audit logs.

sudo filebeat modules enable system

sudo filebeat modules enable auditd

sudo filebeat modules enable apache

The modules help by telling Filebeat where to collect the logs from according to the default system paths. Ensure the respective filesets are set to '*true*' in each module's config file:

sudo nano /etc/filebeat/modules.d/<module>.yml

If the log files are located outside of the default path, uncomment '*var.paths:*' and specify the path. For example in the system.yml:

var.paths: ["filepath"]

```
- module: system
  # Syslog
  syslog:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    #var.paths:

  # Authorization logs
  auth:
    enabled: true

    # Set custom paths for the log files. If left empty,
    # Filebeat will choose the paths depending on your OS.
    #var.paths:
```

To view a full list of available modules and their statuses, run:

sudo filebeat modules list

Enable the module pipelines using:

sudo filebeat setup --pipelines --modules system -M

"system.syslog.enabled=true" -M "system.auth.enabled=true"

sudo filebeat setup --pipelines --modules apache -M

"apache.access.enabled=true" -M "apache.error.enabled=true"

sudo filebeat setup --pipelines --modules auditd -M "auditd.log.enabled=true"

The additional '*-M*' flags are specified here in case of a bug present in Elastic v8 packages. If the modules are functioning normally, then these are unnecessary.

Run the Filebeat setup to load the index template into Elasticsearch, and create the index patterns and load the dashboards into Kibana:

sudo filebeat setup

```
root@ubuntu-honey:~# filebeat setup
Overwriting lifecycle policy is disabled. Set `setup.ilm.overwrite: true` to overwrite.
Index setup finished.
Loading dashboards (Kibana must be running and reachable)
Loaded dashboards
Loaded Ingest pipelines
```

Once the setup has been completed, reconfigure the Filebeat settings to send its logs to Logstash instead.

```
# ---------------------------- Elasticsearch Output ----------------------------
#output.elasticsearch:
  # Array of hosts to connect to.
#  hosts: ["localhost:9200"]

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

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

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

```
# ------------------------------ Logstash Output -------------------------------
output.logstash:
  # The Logstash hosts
  hosts: ["<ip address>:5044"]

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

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

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

  pipelining: 4
```

Once again test the configurations and output:

```
root@ubuntu-honey:~# filebeat test config
Config OK
root@ubuntu-honey:~# filebeat test output
logstash:                :5044...
  connection...
    parse host... OK
    dns lookup... OK
    addresses:
    dial up... OK
  TLS...
    security: server's certificate chain verification is enabled
    handshake... OK
    TLS version: TLSv1.3
    dial up... OK
  talk to server... OK
```

With this, start and enable Filebeat on the server:

sudo systemctl enable filebeat && sudo systemctl start filebeat

To test if the configuration is successful, enter the following command while substituting the IP address again:

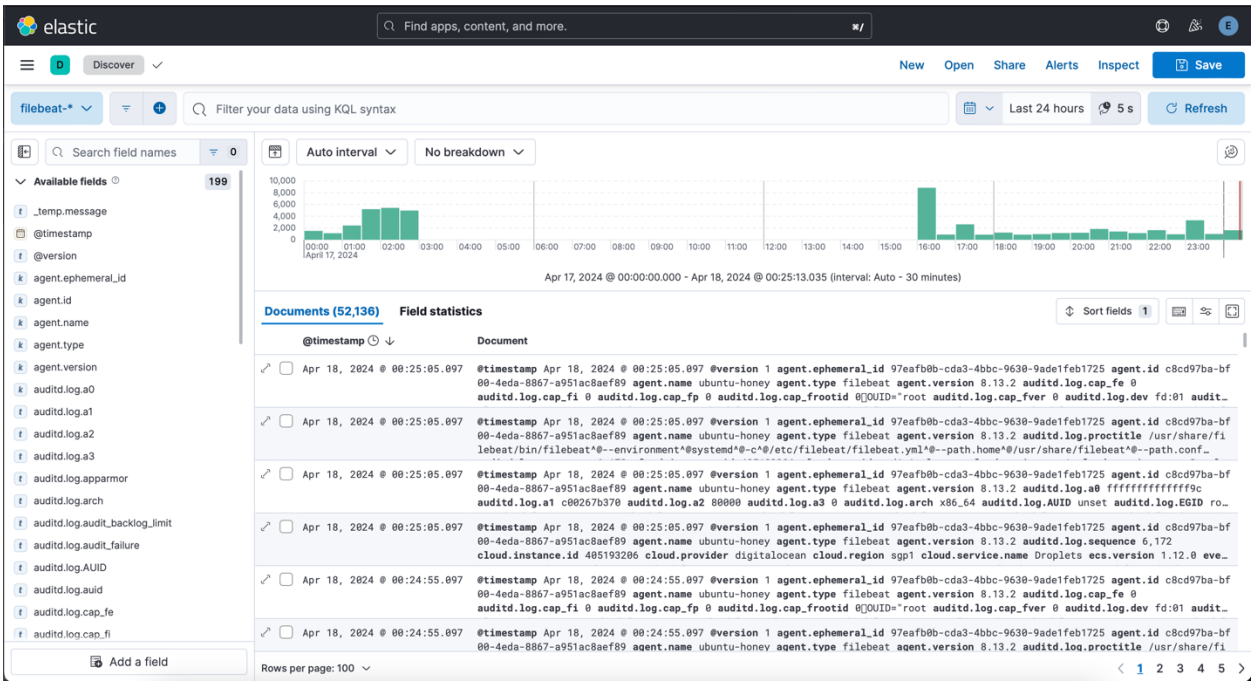
curl -XGET 'https://<ip address>:9200/filebeat-*/_search?pretty' -u elastic --cacert /etc/filebeat/certs/http_ca.crt

```
"}",
          "timezone" : "+00:00",
          "module" : "system",
          "dataset" : "system.syslog"
        },
        "timestamp" : "2024-04-16T09:29:56.095745+00:00"
      }
    },
    {
      "_index" : "filebeat-8.13.2-2024.04.16",
      "_id" : "Qxs-5o4BPliCotaqpti9",
      "_score" : 1.0,
      "_ignored" : [
        "message.keyword",
        "event.original.keyword"
      ],
      "_source" : {
        "process" : {
          "name" : "filebeat",
          "pid" : 758
        },
        "agent" : {
          "name" : "ubuntu-honey",
          "id" : "c8cd97ba-bf00-4eda-8867-a951ac8aef89",
          "type" : "filebeat",
          "ephemeral_id" : "f1aee0c4-7ea0-466b-ad23-c402f231def0",
          "version" : "8.13.2"
        },
        "log" : {
          "file" : {
            "path" : "/var/log/syslog"
          },
          "offset" : 2042122
        },
        "message" : [
          "2024-04-16T09:29:56.096139+00:00 ubuntu-honey filebeat[758]: {\"log.level\":\"info\",\"@timestamp\":
\"2024-04-16T09:29:56.096Z\",\"log.logger\":\"publisher_pipeline_output\",\"log.origin\":{\"function\":\"github.c
om/elastic/beats/v7/libbeat/publisher/pipeline.(*netClientWorker).run\",\"file.name\":\"pipeline/client_worker.go
\",\"file.line\":137},\"message\":\"Connecting to backoff(async(tcp://              :5044))\",\"service.name\":\"f
ilebeat\",\"ecs.version\":\"1.6.0\"}",
          "{\"log.level\":\"info\",\"@timestamp\":\"2024-04-16T09:29:56.096Z\",\"log.logger\":\"publisher_pipel
ine_output\",\"log.origin\":{\"function\":\"github.com/elastic/beats/v7/libbeat/publisher/pipeline.(*netClientWor
ker).run\",\"file.name\":\"pipeline/client_worker.go\",\"file.line\":137},\"message\":\"Connecting to backoff(asy
nc(tcp://              :5044))\",\"service.name\":\"filebeat\",\"ecs.version\":\"1.6.0\"}"
```


A similar looking output should be obtained. If the output displays 0 hits, Elasticsearch is unable to load the targeted logs. Revisit the previous steps to check for any errors, or ensure the Logstash configurations files are written correctly.

Finally, verify the logs in Kibana dashboard by returning to the Kibana web browser.

Then in the left-hand navigation bar, click on ‘Discover’. On that page, ensure that the index Filebeat-* is selected. The following result should be akin to this:



Filebeat has been successfully installed and setup to send its logs to Elasticsearch via Logstash. Proceed to adjust the settings according to needs.

Attack Surface Configuration

1. Cowrie Honeypot

Cowrie was installed on the honeypot server as a medium to high interaction honeypot that enables logging of brute force attacks and shell interaction by attackers on the SSH and Telnet services [4]. The installation was done according to the documentation provided [5].

To allow attackers to access the simulated honeypot server instead of the actual, the SSH service was configured to sit on port 22222. Attackers accessing the typical service ports of 22 and 23 were set to be routed to Cowrie's SSH and Telnet services run on ports 2222 and 2223 respectively.

To enable Filebeat to send Cowrie's logs to Logstash, the following settings were added to the '*filebeat.yml*' file to point it to where the logs were stored.

```
# ============================== Filebeat inputs ===============================

filebeat.inputs:

# Each - is an input. Most options can be set at the input level, so
# you can use different inputs for various configurations.
# Below are the input-specific configurations.

# filestream is an input for collecting log messages from files.
- type: filestream

  # Unique ID among all inputs, an ID is required.
  id: my-filestream-id

  # Change to true to enable this input configuration.
  enabled: false

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

- type: log
  id: cowrie-filestream
  enabled: true
  paths:
    - /home/medrare/cowrie/var/log/cowrie/cowrie.json*
  fields:
    event_type: cowrie
```

Adding the field '*event_type: cowrie'* is important to specify filter and output instructions for Cowrie's logs in the Logstash configuration settings.

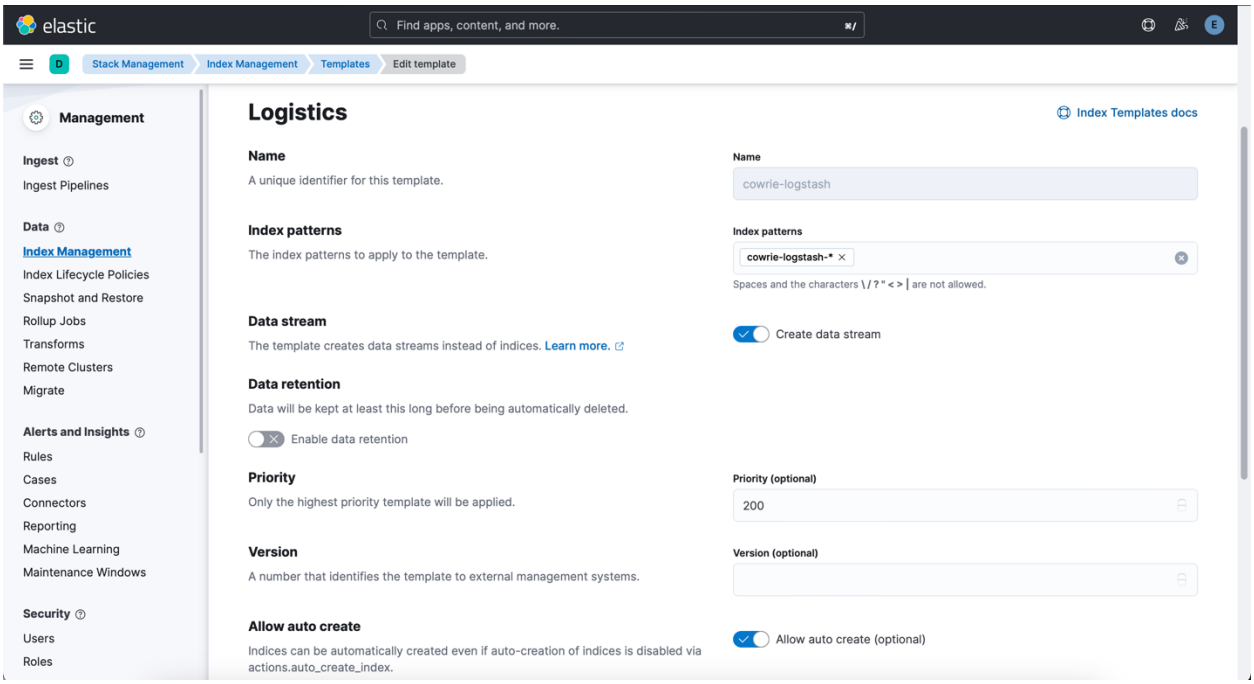
Accordingly, the following lines were added to the filter settings in the Logstash config file, directly below what was displayed in the Elastic Cloud installation steps [6]:

```
 if [fields][event_type] == "cowrie" {
     json {
         source => message
         target => honeypot
     }

     date {
         match => [ "timestamp", "ISO8601" ]
     }

     if [src_ip]  {

         mutate {
             add_field => { "src_host" => "%{src_ip}" }
         }

         dns {
             reverse => [ "src_host" ]
             nameserver => [ "8.8.8.8", "8.8.4.4" ]
             action => "replace"
             hit_cache_size => 4096
             hit_cache_ttl => 900
             failed_cache_size => 512
             failed_cache_ttl => 900
         }


         geoip {
             source => "src_ip"
             target => "geoip"
         }

     }

     mutate {
         # cut out useless tags/fields
         remove_tag => [ "beats_input_codec_plain_applied"]
         remove_field => [ "[log][file][path]", "[log][offset]" ]
     }
 }
}
```

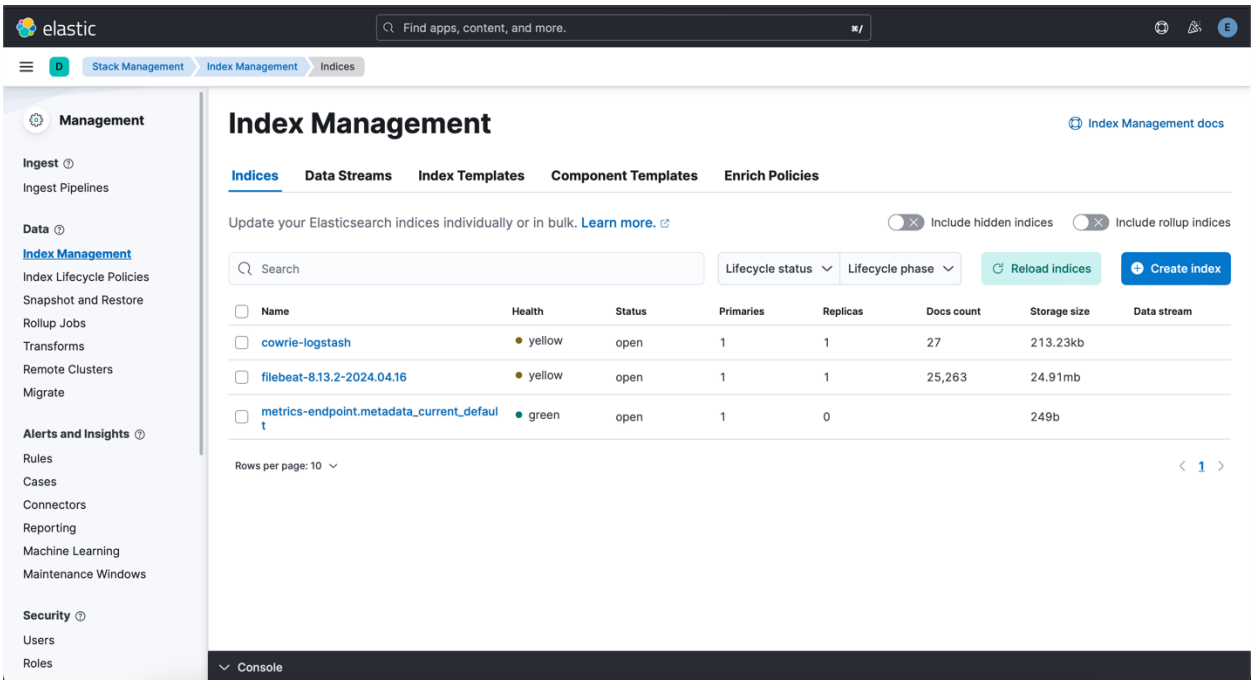
An output setting was also added as follows:

```
output {
  if [@metadata][pipeline] {
        elasticsearch {
        ssl => true
        hosts => ["https://              :9200"]
        cacert => "/etc/logstash/certs/http_ca.crt"
        user => "logstash_internal"
        password => "                    "
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        pipeline => "%{[@metadata][pipeline]}"
        }
  } else if [fields][event_type] == "cowrie" {
      elasticsearch {
          hosts => ["https://              :9200"]
          ssl => true
          cacert => "/etc/logstash/certs/http_ca.crt"
          user => "elastic"
          password => "                          "
          index => "cowrie-logstash-%{+YYYY.MM.dd}"
          ilm_enabled => auto
          ilm_rollover_alias => "cowrie-logstash"
      }
      file {
          path => "/tmp/cowrie-logstash.log"
          codec => json
      }
      stdout {
          codec => rubydebug
      }
  } else {
        elasticsearch {
        ssl => true
        hosts => ["https://              :9200"]
        cacert => "/etc/logstash/certs/http_ca.crt"
        user => "logstash_internal"
        password => "                    "
        manage_template => false
        index => "%{[@metadata][beat]}-%{[@metadata][version]}-%{+YYYY.MM.dd}"
        }
  }
}
```

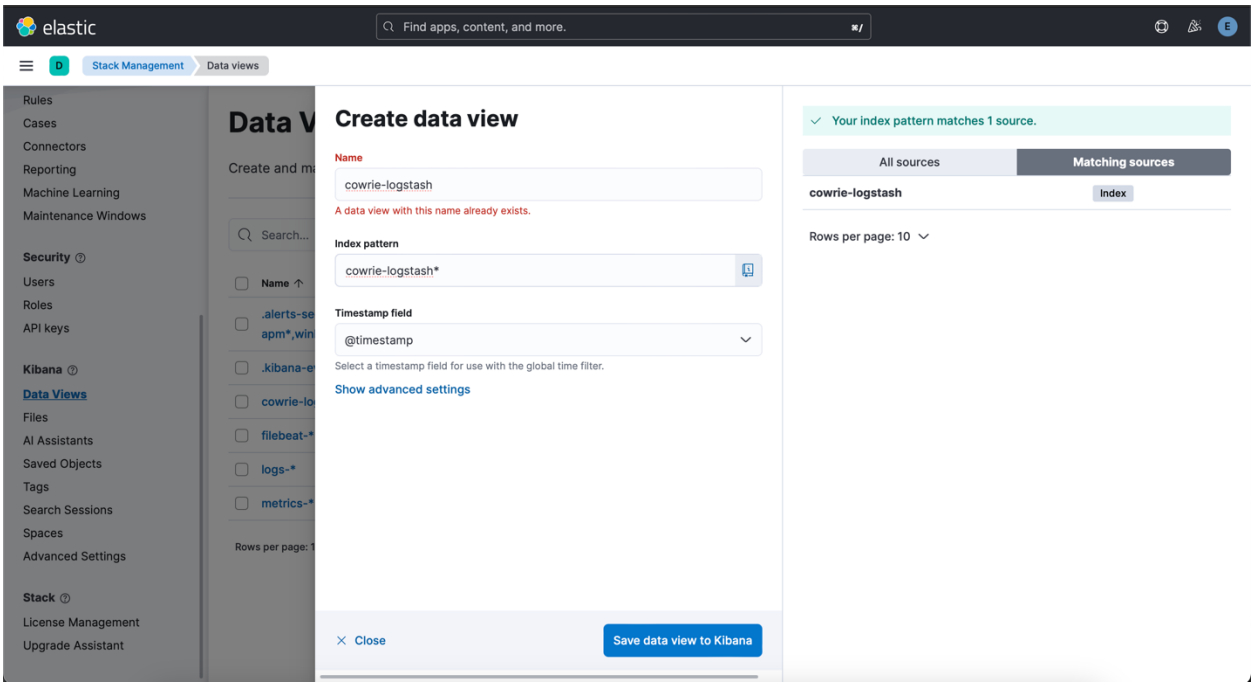
Additionally, an index template was configured in Kibana to allow the creation of an index for Cowrie logs.



With this an index was created in Kibana.



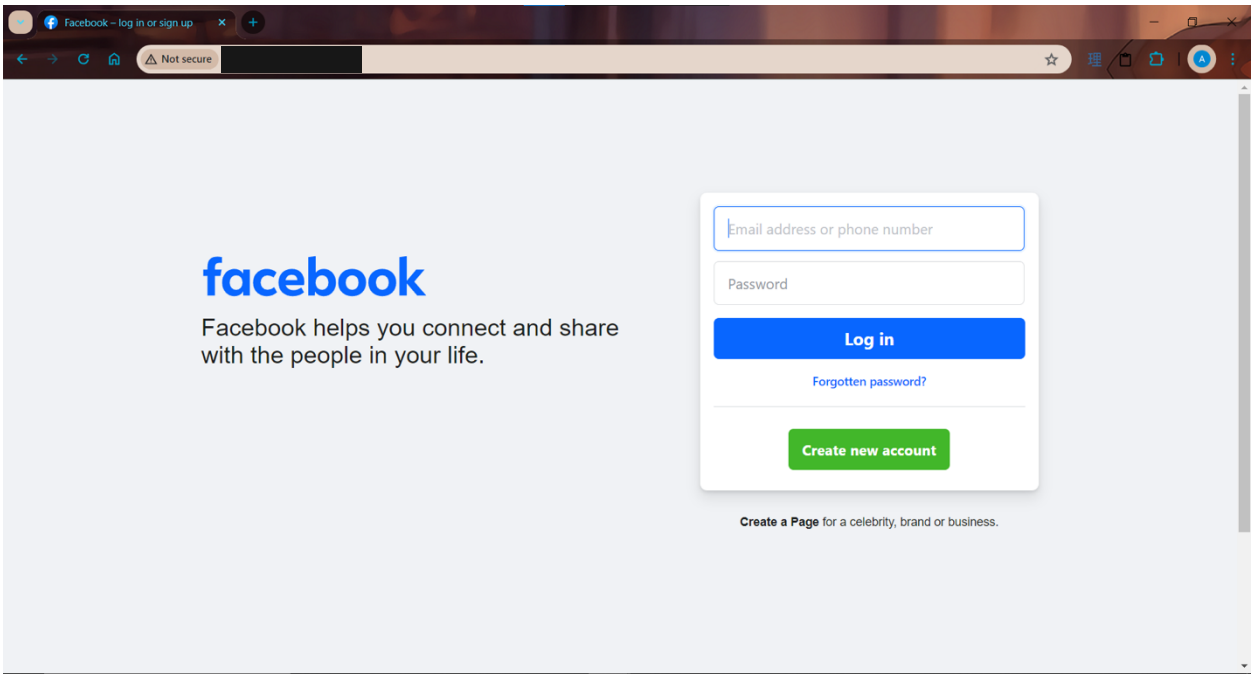
To visualise the logs as a category of its own, an index pattern was created.



2. Apache Webserver

To simulate a client server in production with its own web services and webpage, Apache2 was run on the honeypot server as well. This provides an attack surface on port 80. As an added realistic feature, the main homepage was set as a fake Facebook login page.

The Apache2 logs were stored in the '*access.log*' and '*error.log*' logs, and collected through the Apache module in Filebeat.



3. FTP Server

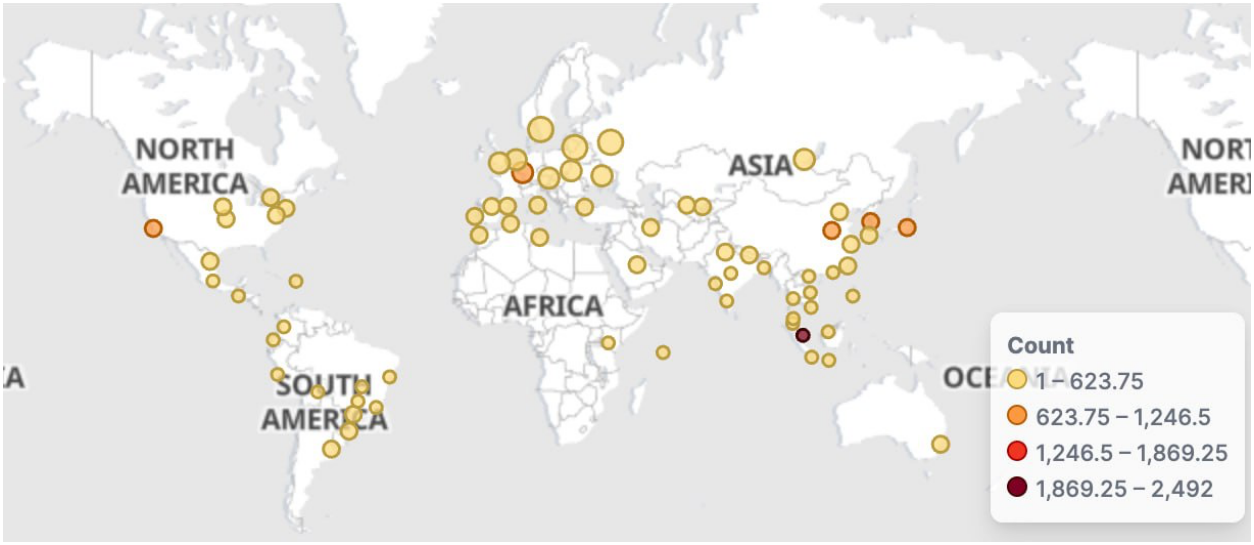
To study how backdoor exploits resulting in shell creation can be monitored, the vulnerable VSFTPD 2.3.4 service was intentionally installed on the honeypot server [7].

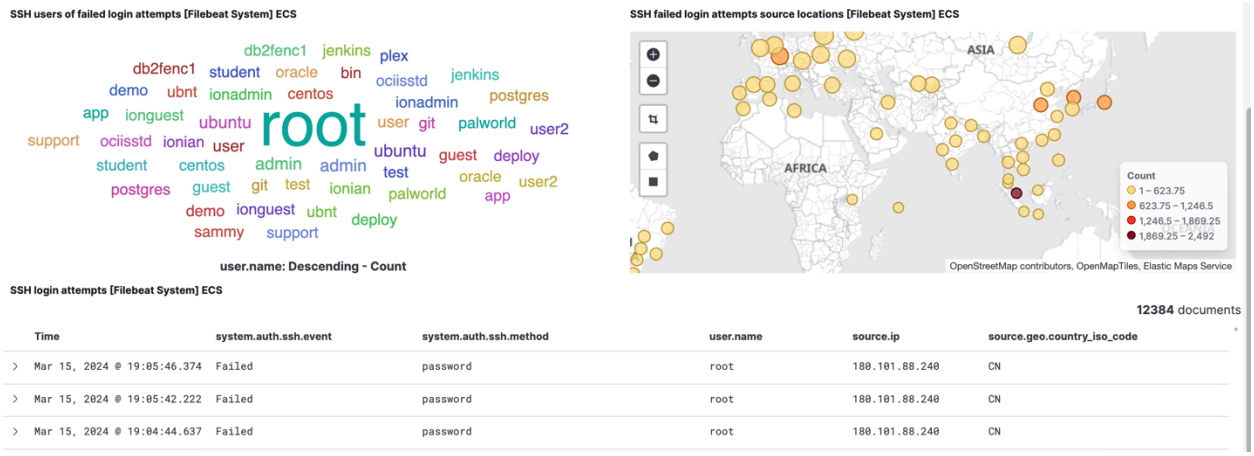
Since any command inputs during the attack is executed through the spawned shell instead of the FTP service itself, logs for this attack was recorded through the '*audit.log*' rather than the '*vsftpd.log*'. Hence the auditd module in Filebeat was used to collect and deliver the logs.

Server Hardening

1. Strong User Credentials

Remote access to the DigitalOcean droplet consoles were done through the SSH service. As such, it was necessary to keep the port open to external addresses, which made it openly visible to the public and vulnerable to attacks. Evidently, soon after the droplet was created, brute force attacks were recorded in the '*auth.log*'.





As shown in the Kibana visualisation above created from '*auth.log*' records, attacks came from across the world. Therefore, strong passwords were set for both the Elastic Cloud and honeypot servers to restrict unauthorised access.

2. Fail2ban

On top of configuring user credentials, Fail2ban was installed on both servers to track excessive failed login attempts from each IP address and place them on a timeout to interrupt their brute force attempt.

```
root@ubuntu-elk:~# service fail2ban status
● fail2ban.service - Fail2Ban Service
     Loaded: loaded (/lib/systemd/system/fail2ban.service; enabled; preset: enabled)
     Active: active (running) since Thu 2024-04-18 17:31:56 UTC; 1min 40s ago
       Docs: man:fail2ban(1)
   Main PID: 770 (fail2ban-server)
      Tasks: 5 (limit: 9476)
     Memory: 16.6M
        CPU: 323ms
     CGroup: /system.slice/fail2ban.service
             └─770 /usr/bin/python3 /usr/bin/fail2ban-server -xf start

Apr 18 17:31:56 ubuntu-elk systemd[1]: Started fail2ban.service - Fail2Ban Service.
Apr 18 17:31:57 ubuntu-elk fail2ban-server[770]: 2024-04-18 17:31:57,276 fail2ban.configreader   [770]: WARNING >
Apr 18 17:31:57 ubuntu-elk fail2ban-server[770]: Server ready
```

```
2024-04-18 17:31:57,674 fail2ban.jail           [770]: INFO    Jail 'sshd' started
2024-04-18 17:31:59,449 fail2ban.filter         [770]: INFO    [sshd] Found 85.51.24.68 - 2024-04-18 17:31:59
2024-04-18 17:32:01,670 fail2ban.filter         [770]: INFO    [sshd] Found 85.51.24.68 - 2024-04-18 17:32:01
2024-04-18 17:32:06,316 fail2ban.filter         [770]: INFO    [sshd] Found 119.15.87.26 - 2024-04-18 17:32:06
2024-04-18 17:32:08,133 fail2ban.filter         [770]: INFO    [sshd] Found 119.15.87.26 - 2024-04-18 17:32:08
2024-04-18 17:35:39,304 fail2ban.filter         [770]: INFO    [sshd] Found 65.76.238.3 - 2024-04-18 17:35:39
2024-04-18 17:35:40,918 fail2ban.filter         [770]: INFO    [sshd] Found 65.76.238.3 - 2024-04-18 17:35:40
```

3. Firewall configurations

The UFW firewall was enabled and configured on both servers to limit access to the services.

Aside from the SSH ports to allow console access, only the ports relevant for communication between the Elastic components were enabled. Specifically, incoming access to ports 5044 (Logstash) and 9200 (Elasticsearch) was restricted unless coming from the honeypot server. Only port 5601 (Kibana) allowed traffic from anywhere to enable access to the Kibana dashboard remotely.

```
root@ubuntu-elk:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
5601                       ALLOW       Anywhere
5044                       ALLOW       
9200                       ALLOW       
OpenSSH (v6)               ALLOW       Anywhere (v6)
5601 (v6)                  ALLOW       Anywhere (v6)
```

Meanwhile on the honeypot server, outgoing traffic via ports 5044, 5601 and 9200 were only allowed towards the Elastic Cloud server. Port 22222 was specifically enabled since the SSH service was configured to run on it. Although public access to Cowrie would not endanger the main server, access to ports 22, 23, 2222 and 2223 was disabled as a safety precaution in case of misconfigurations.

Simultaneously, incoming traffic from the IP address where the attack scripts are run is allowed to allow the attacks to go through.

```
root@ubuntu-honey:~# ufw status
Status: active

To                         Action      From
--                         ------      ----
OpenSSH                    ALLOW       Anywhere
                 5044      ALLOW       Anywhere
                 9200      ALLOW       Anywhere
                 5601      ALLOW       Anywhere
22222                      ALLOW       Anywhere
Anywhere                   ALLOW
OpenSSH (v6)               ALLOW       Anywhere (v6)
22222 (v6)                 ALLOW       Anywhere (v6)
```

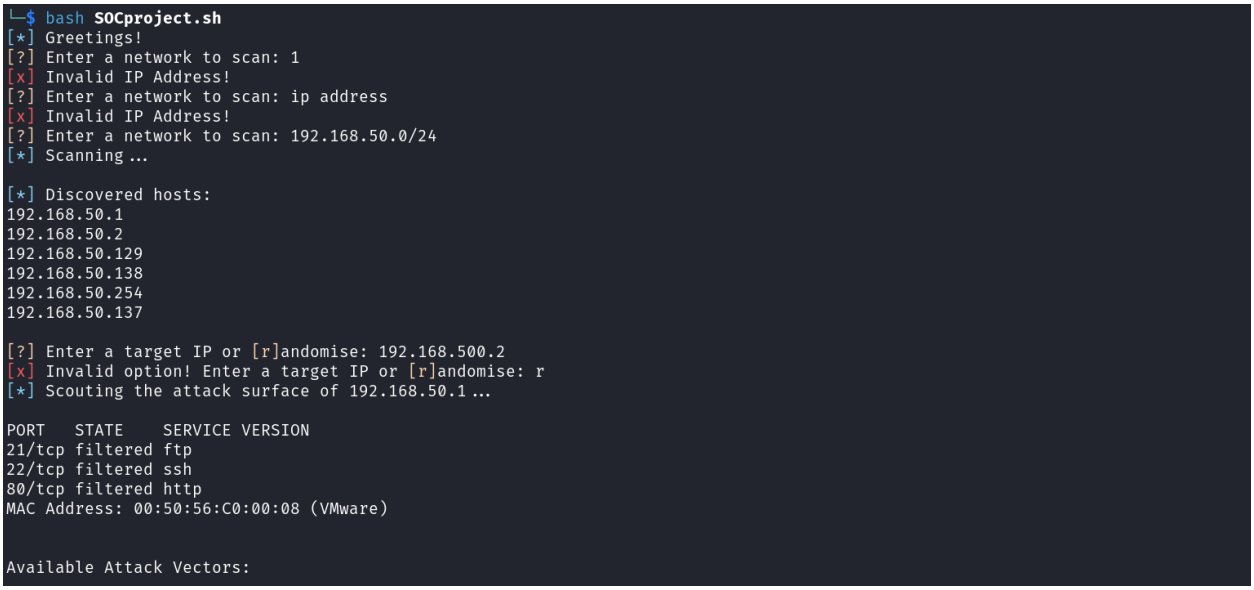
Attack Script

To test if the attack surface of the honeypot server was configured correctly and the Elastic Cloud was receiving the appropriate logs as it should while accurately detecting the attacks, a penetration testing script was written.

When run, the script first checks if the required SSHpass tool is installed and proceeds to install it if not. Otherwise, no output is shown for this process.

The user is then required to provide a valid network or IP address to scan. This step allows host discovery to identify which hosts are up. If an invalid address is provided, the user is informed and required to provide a new input.

Once a valid network or IP address is received, the scan is run and the discovered hosts is output to the user.



At this stage, the user is then requested to enter a target host to scan if the targeted services and ports are open. Again, the user input is validated. The user is also provided an option to randomly select a host to scan.

Upon a successful scan, the result is displayed followed by information of the available attack vectors the script provides and the expected results when completed.

Here the user may choose the attack to run, randomise an option, or return to scan a different host if they should desire.

```
Available Attack Vectors:
═══════════════════════

1 | Bruteforce
Target Service: SSH | Port 22
Description: Use a list of weak passwords to attempt login as root user via SSH by trial and error. If successful, gain persistence by creatin
g a netcat reverse connection cronjob.

2 | Distributed Denial-of-Service (DDoS)
Target Service: HTTP | Port 80
Description: Flood the target's webserver with traffic in the attempt to disrupt its service and force it offline.

3 | Backdoor Command Execution
Target Service: FTP | Port 21
Description: Exploits the backdoor found in VSFTPD v2.3.4 to spawn a shell and retrieve password hashes.

[?] Select an attack [1-3], [r]andomise, return [b]ack to choose a different IP address, or anything else to exit: b
[*] Returning to target selection ...
192.168.50.1
192.168.50.2
192.168.50.129
192.168.50.138
192.168.50.254
192.168.50.137

[?] Enter a target IP or [r]andomise: 192.168.50.138
[*] Scouting the attack surface of 192.168.50.138 ...
```

The option to quit the script is also given if they should choose not to run an attack.

```
[?] Select an attack [1-3], [r]andomise, return [b]ack to choose a different IP address, or anything else to exit: 4
[x] Exiting ...
[*] Goodbye!
```

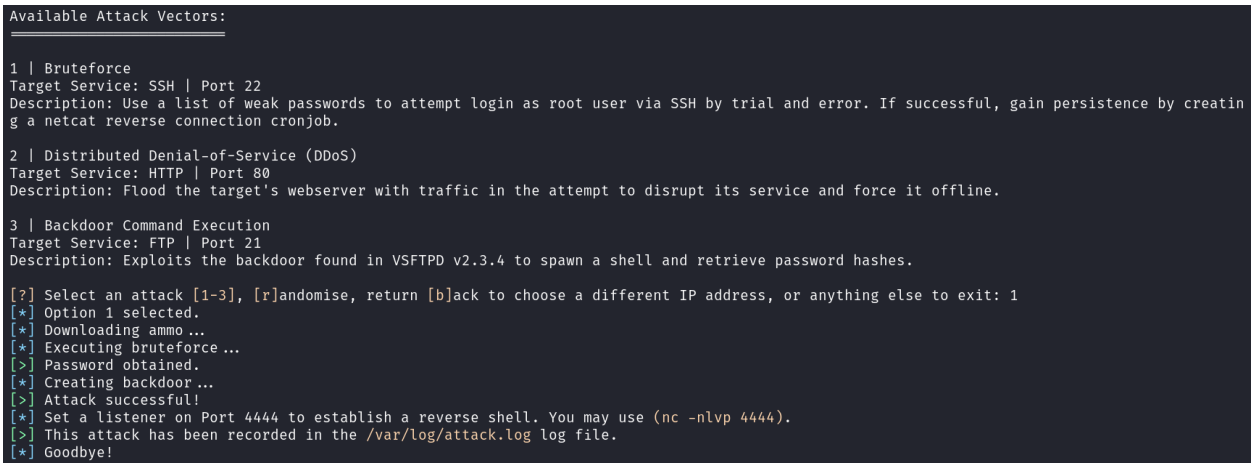
The following shows if the script was run on a single IP address from the beginning without invalid inputs:

```
└─$ bash SOCproject.sh
[*] Greetings!
[?] Enter a network to scan: [redacted]
[*] Scanning ...

[*] Discovered hosts:
[redacted]
[?] Enter a target IP or [r]andomise: r
[*] Scouting the attack surface of [redacted] ...

PORT   STATE SERVICE VERSION
21/tcp open  ftp     vsftpd 2.3.4
22/tcp open  ssh     OpenSSH 6.0p1 Debian 4+deb7u2 (protocol 2.0)
80/tcp open  http    Apache httpd 2.4.57 ((Ubuntu))
Service Info: OSs: Unix, Linux; CPE: cpe:/o:linux:linux_kernel
```

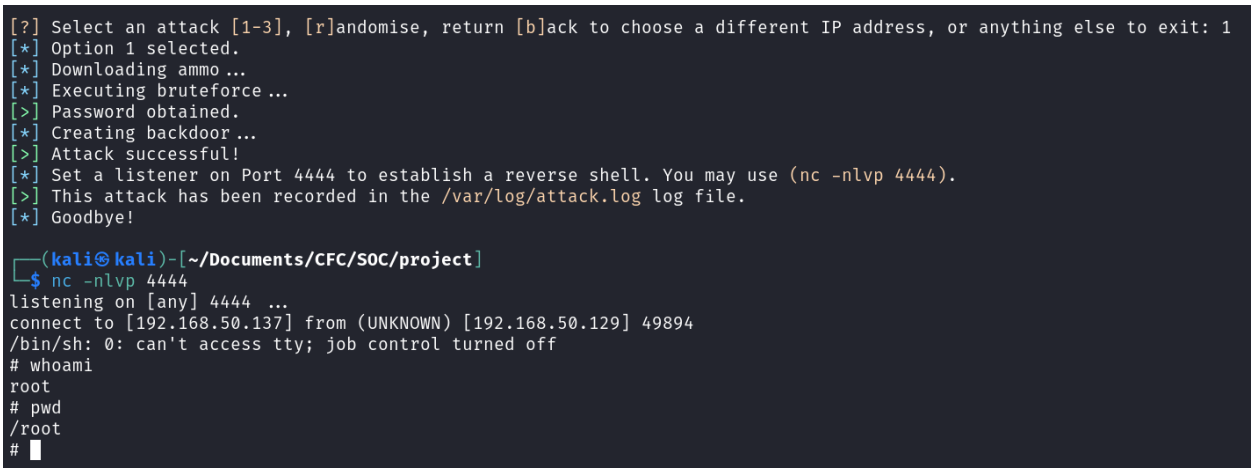
1. **Brute Force**

If option 1 was selected, a brute force attack using Hydra is executed on the SSH service to obtain valid login credentials. The script downloads a common password list to test for

possible logins as the root user [8]. Once a successful user-password match is found, the user is informed.



Then the SSH service is accessed and a backdoor is created by starting a cronjob that executes a netcat reverse shell connection with the '*mkfifo*' command [9], hence establishing persistence in the target server with the ability to create a shell by running a netcat listener anytime, even if the brute force attack was discovered and user credentials were changed subsequently. This cronjob was further hidden with layered obfuscation by using the carriage return method and writing it in the '*cron.d*' directory [10].



When the attack is finished, it is recorded in a log for future reference.

The following shows an example if the brute force procedure failed:

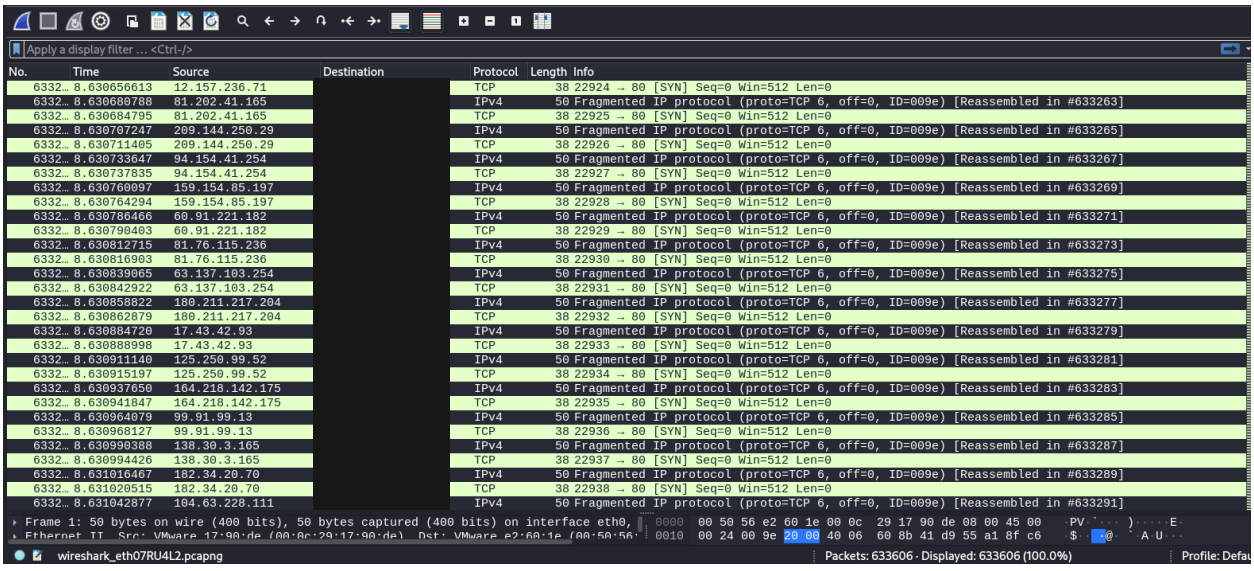
```
[?] Select an attack [1-3], [r]andomise, return [b]ack to choose a different IP address, or anything else to exit: r
[*] Randomising ...
[*] Option 1 selected.
[*] Downloading ammo ...
[*] Executing bruteforce ...
[x] No valid passwords were found. Process completed.
[*] Goodbye!
```

2. Distributed Denial-of-Service

When option 2 is selected, a Distributed Denial-of-Service (DDoS) attack using a TCP SYN flood is run against the target's webserver to deny access to it.

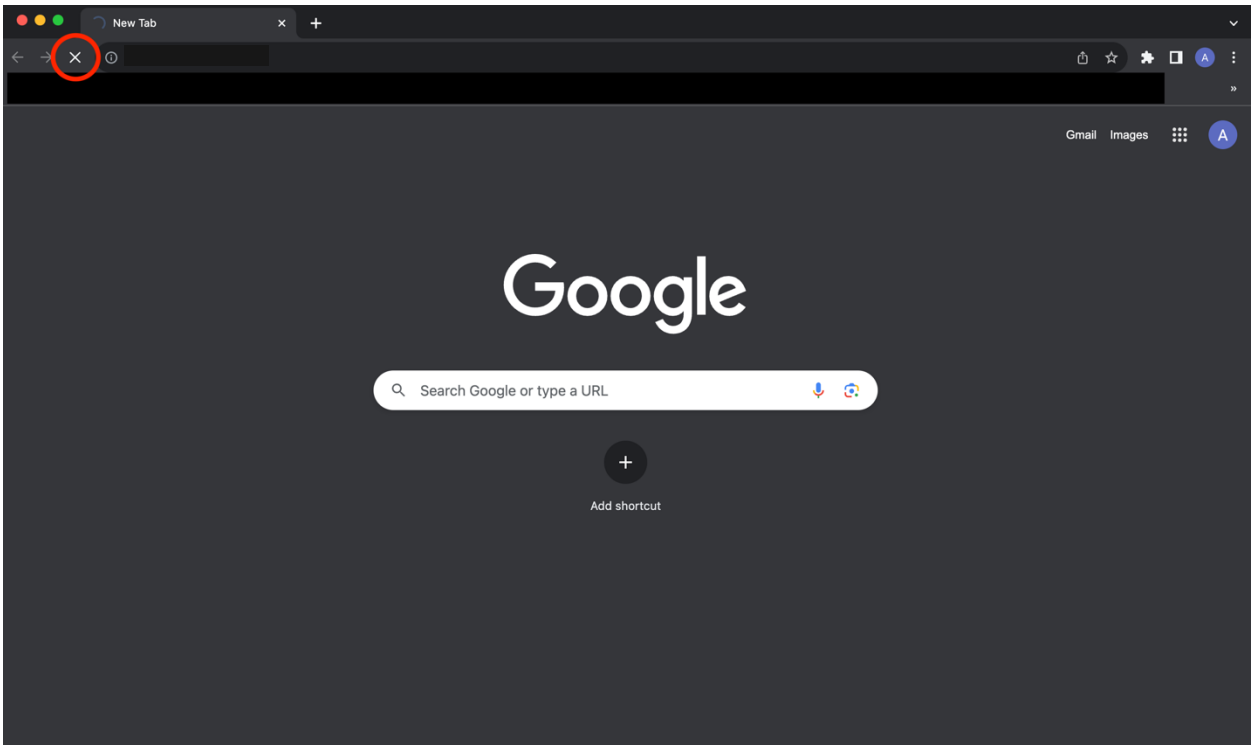
```
Available Attack Vectors:
═══════════════════════

1 | Bruteforce
Target Service: SSH | Port 22
Description: Use a list of weak passwords to attempt login as root user via SSH by trial and error. If successful, gain persistence by creatin
g a netcat reverse connection cronjob.

2 | Distributed Denial-of-Service (DDoS)
Target Service: HTTP | Port 80
Description: Flood the target's webserver with traffic in the attempt to disrupt its service and force it offline.

3 | Backdoor Command Execution
Target Service: FTP | Port 21
Description: Exploits the backdoor found in VSFTPD v2.3.4 to spawn a shell and retrieve password hashes.

[?] Select an attack [1-3], [r]andomise, return [b]ack to choose a different IP address, or anything else to exit: r
[*] Randomising ...
[*] Option 2 selected.
[*] Executing TCP SYN flood on target. Use 'Ctrl-C' to end the attack.
^C[x] Attack stopped.
[>] This attack has been recorded in the /var/log/attack.log log file.
[*] Goodbye!
```

This is done using the Hping3 tool with the '*fragmented*' and '*random-source*' flag options enabled to create a larger load and obscure the attacker's IP address by spoofing.



As seen from the traffic captured using Wireshark, the destination IP is shown to be receiving fragmented TCP SYN traffic from multiple sources in quick succession.

When accessing the webserver, the page is unable to load as it no longer has the capacity to handle the request; what should have been the fake Facebook page does not appear.



3. Backdoor Command Execution

Option 3 operates on the vulnerability of VSFTPD 2.3.4, spawning a shell when executed. This is done using Metasploit. After a shell is created, a hashdump post-exploitation module is run to harvest the credentials of the target server, namely by downloading the '*passwd*' and '*shadow*' files to obtain password hashes, thus gaining more options for subsequent attacks.

```
Available Attack Vectors:
═══════════════════════

1 | Bruteforce
Target Service: SSH | Port 22
Description: Use a list of weak passwords to attempt login as root user via SSH by trial and error. If successful, gain persistence by creatin
g a netcat reverse connection cronjob.

2 | Distributed Denial-of-Service (DDoS)
Target Service: HTTP | Port 80
Description: Flood the target's webserver with traffic in the attempt to disrupt its service and force it offline.

3 | Backdoor Command Execution
Target Service: FTP | Port 21
Description: Exploits the backdoor found in VSFTPD v2.3.4 to spawn a shell and retrieve password hashes.

[?] Select an attack [1-3], [r]andomise, return [b]ack to choose a different IP address, or anything else to exit: 3
[*] Option 3 selected.
[*] Exploiting the FTP backdoor ...
[>] Target credentials successfully retrieved! Files stored in '/home/kali/.msf4/loot/' by default unless configured otherwise.
[>] This attack has been recorded in the /var/log/attack.log log file.
[*] Goodbye!
```

The user is informed where the downloaded files are stored upon completion of the attack.

```
└─$ ls /home/kali/.msf4/loot/
20240410111933_default_192.168.50.138_linux.passwd_762025.txt      20240412163436_default_143.198.210.106_linux.passwd_330774.txt
20240410111933_default_192.168.50.138_linux.shadow_610799.txt      20240412163436_default_143.198.210.106_linux.shadow_832373.txt
20240410111934_default_192.168.50.138_linux.hashes_192800.txt      20240412163437_default_143.198.210.106_linux.hashes_188694.txt
20240410111934_default_192.168.50.138_linux.passwd.his_959154.txt  20240412163437_default_143.198.210.106_linux.passwd.his_048591.txt
```

By ‘cat’-ing the files, we can see the information stored in it.

```
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/run/ircd:/usr/sbin/nologin
_apt:x:42:65534::/nonexistent:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:998:998:systemd Network Management:/:/usr/sbin/nologin
systemd-timesync:x:997:997:systemd Time Synchronization:/:/usr/sbin/nologin
dhcpcd:x:100:65534:DHCP Client Daemon,,,:/usr/lib/dhcpcd:/bin/false
messagebus:x:101:106::/nonexistent:/usr/sbin/nologin
syslog:x:102:107::/nonexistent:/usr/sbin/nologin
systemd-resolve:x:996:996:systemd Resolver:/:/usr/sbin/nologin
uuidd:x:103:108::/run/uuidd:/usr/sbin/nologin
tss:x:104:109:TPM software stack,,,:/var/lib/tpm:/bin/false
tcpdump:x:105:110::/nonexistent:/usr/sbin/nologin
sshd:x:106:65534::/run/sshd:/usr/sbin/nologin
pollinate:x:107:1::/var/cache/pollinate:/bin/false
landscape:x:108:112::/var/lib/landscape:/usr/sbin/nologin
fwupd-refresh:x:109:113:fwupd-refresh user,,,:/run/systemd:/usr/sbin/nologin
polkitd:x:995:995:polkit:/nonexistent:/usr/sbin/nologin
lxd:x:999:100::/var/snap/lxd/common/lxd:/bin/false
secret:x:1000:1000:trueadmin,,,:/home/secret:/bin/bash
medrare:x:1001:1001:,,,:/home/medrare:/bin/bash
honeyd:x:1002:1002:,,,:/home/honeyd:/bin/bash
ftpuser:x:1003:1003:,,,:/home/ftpuser:/bin/bash
ftp:x:1004:1004::/var/ftp:/bin/sh
```

```
└─$ cat /home/kali/.msf4/loot/20240412163437_default_143.198.210.106_linux.hashes_188694.txt
root:$y$j9T$loa3HS71Wh9xSW6LlU.oz.$UllFf33hMS5cb8Am2FjJaQLRG2xGE10wBnJyYOuGwT7:0:0:root:/root:/bin/bash
polkitd:!*:995:995:polkit:/nonexistent:/usr/sbin/nologin
secret:$y$j9T$jMlnQiRign.ak0ys.5GkR.$6JBD26FQvyGjxcSja/TXiO5×7E8M/eFnaJvWKWU5YOB:1000:1000:trueadmin,,,:/home/secret:/bin/bash
honeyd:$y$j9T$FPEk1dZ6mCE/rZRIADmt4/$CFHe68GOBsptmeCtbz7CBudopy4vLjxfvL//Wc1Net9:1002:1002:,,,:/home/honeyd:/bin/bash
ftpuser:$y$j9T$INseikU5eqEHbxslpwb5s/$VrowOZM16AkPynD5GT5vWD7/UEPgFL5yb5BriYT2acC:1003:1003:,,,:/home/ftpuser:/bin/bash
```

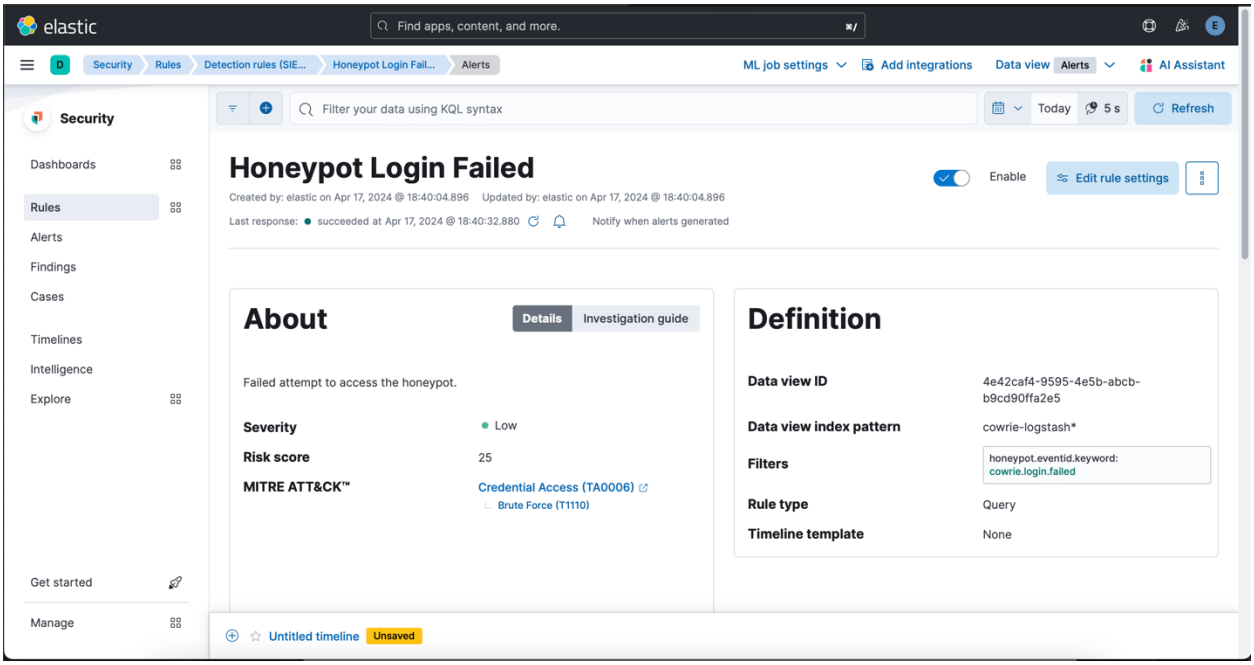
Again, the attacks are recorded in the log.

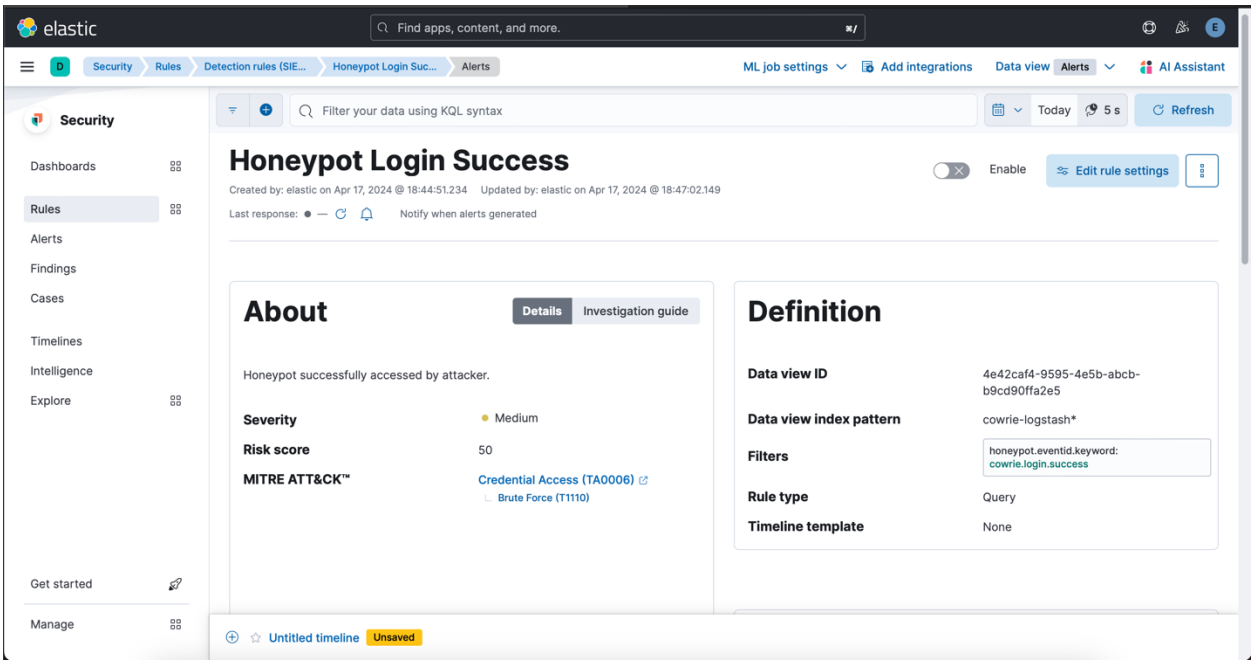
```
└─$ cat /var/log/attack.log
Thu Apr 11 02:04:03 AM +08 2024- [*] Bruteforce attack on 192.168.50.138 completed.
Thu Apr 11 02:04:23 AM +08 2024- [*] DDoS attack on 192.168.50.138 completed.
Thu Apr 11 02:04:26 AM +08 2024- [*] FTP backdoor exploit on 192.168.50.138 completed.
Fri Apr 12 04:34:37 PM +08 2024- [*] FTP backdoor exploit on [redacted] completed.
Fri Apr 12 04:59:42 PM +08 2024- [*] DDoS attack on [redacted] completed.
Fri Apr 12 05:54:45 PM +08 2024- [*] Bruteforce attack on [redacted] completed.
```

Results

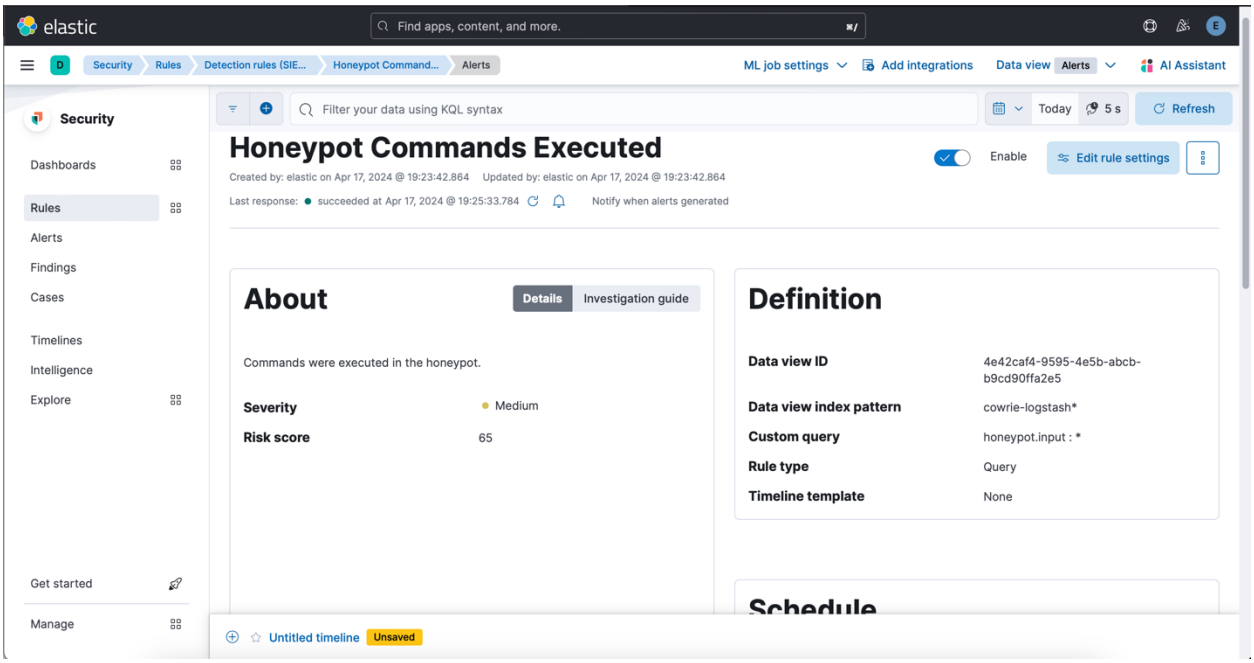
Detecting the Brute Force Attack

Security rules were configured to detect brute force attacks and command executions in the honeypot, providing an alert upon occurrence.



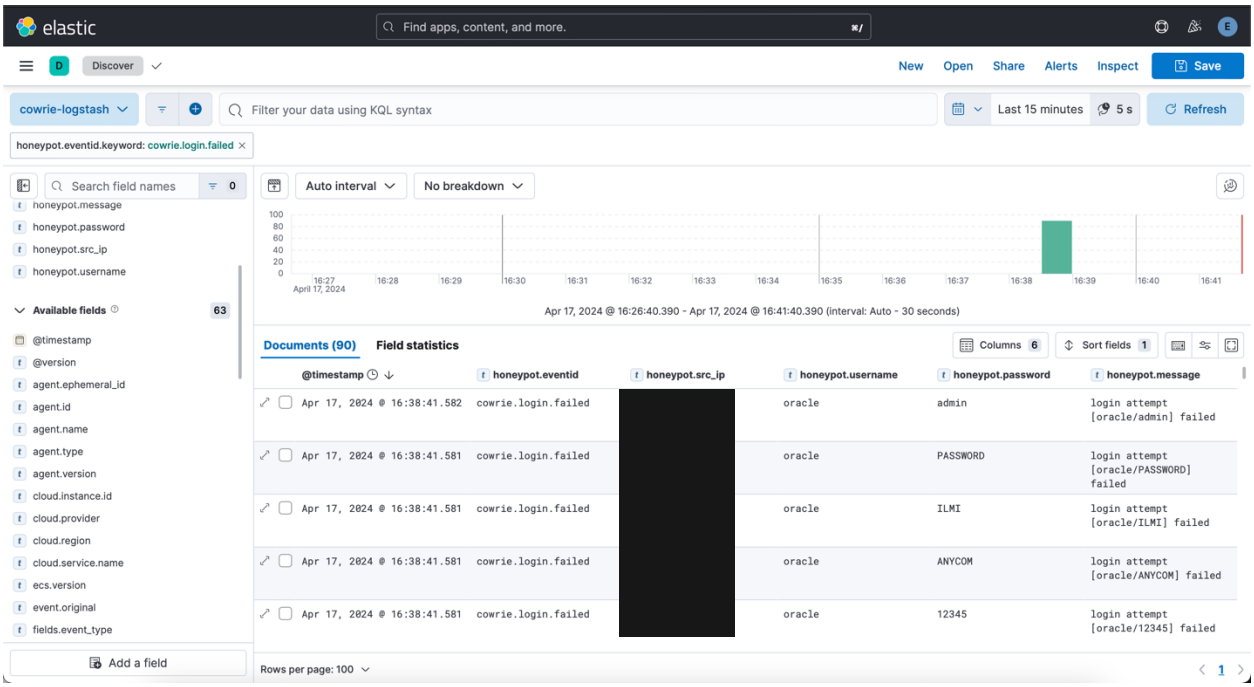


As any authorised user attempting to login to the server via SSH would not access the honeypot, any successful logins in Cowrie are recognised as legitimate attacks rather than false positives.

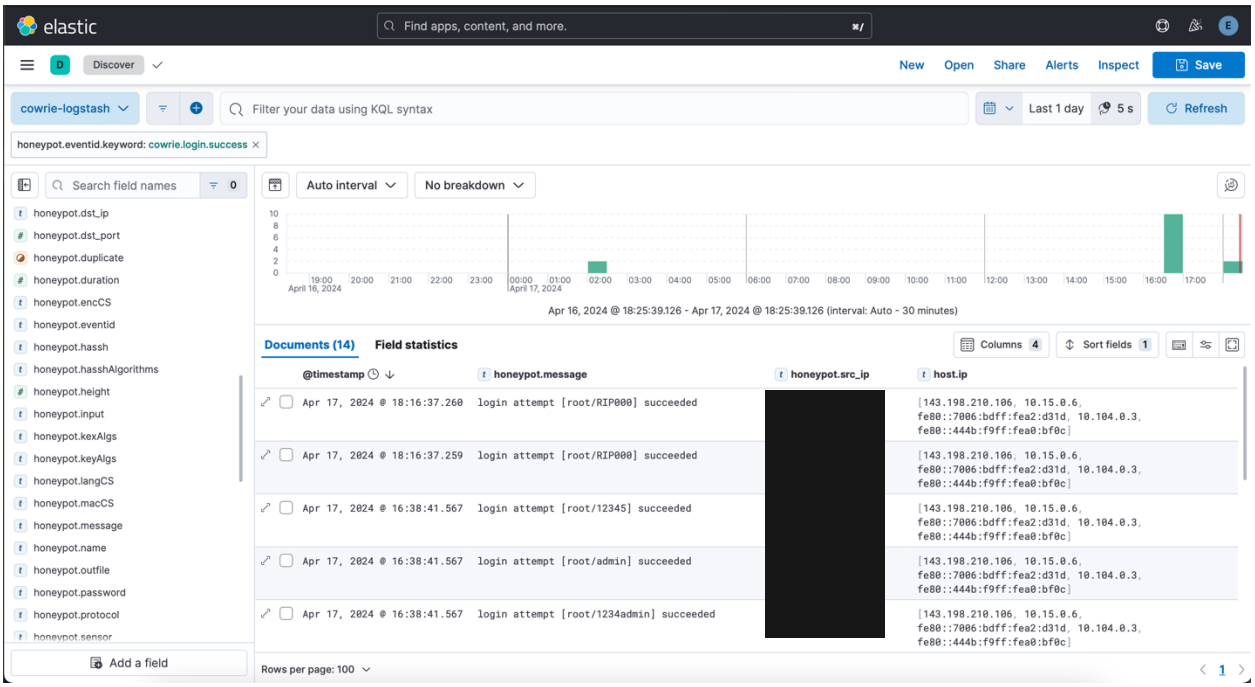


Because the default SSH port 22 was configured to route to the Cowrie Honeypot, any attacks performed on it are logged by Cowrie and thus flowed to the Cowrie index pattern as configured in the Cowrie installation setup.

By filtering for failed logins, ELK was able to capture the attempts.

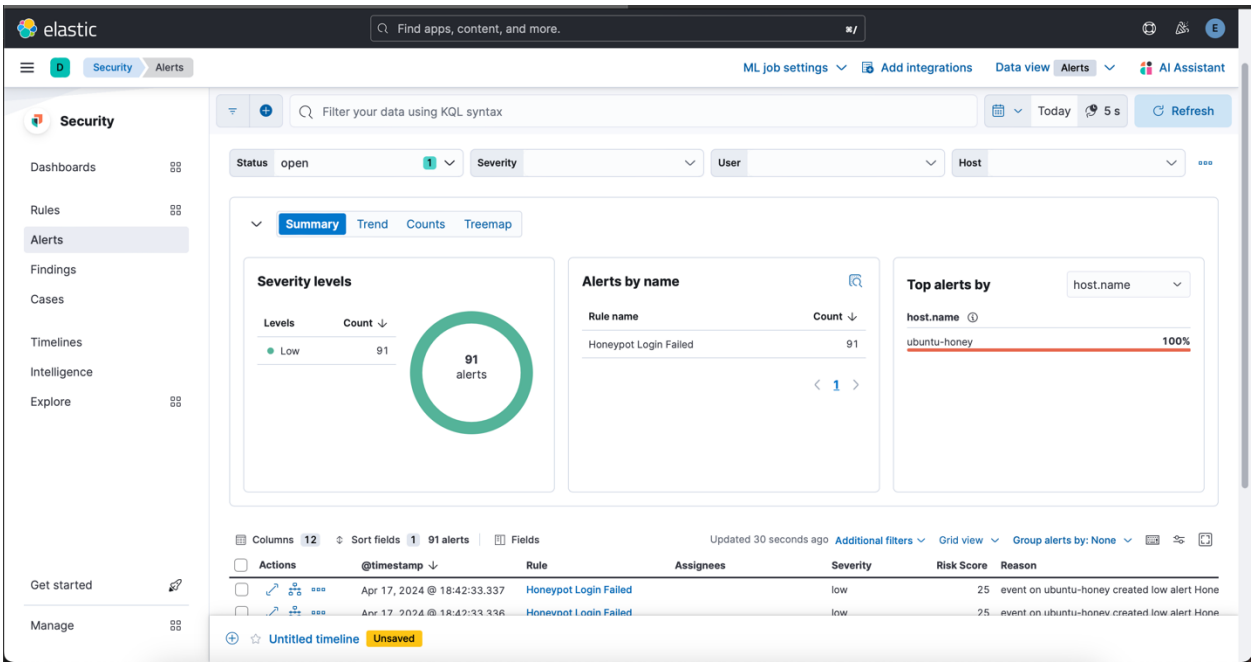


Crucial information such as the source IP address, attempted username and password pairings, and time of attack can be observed.

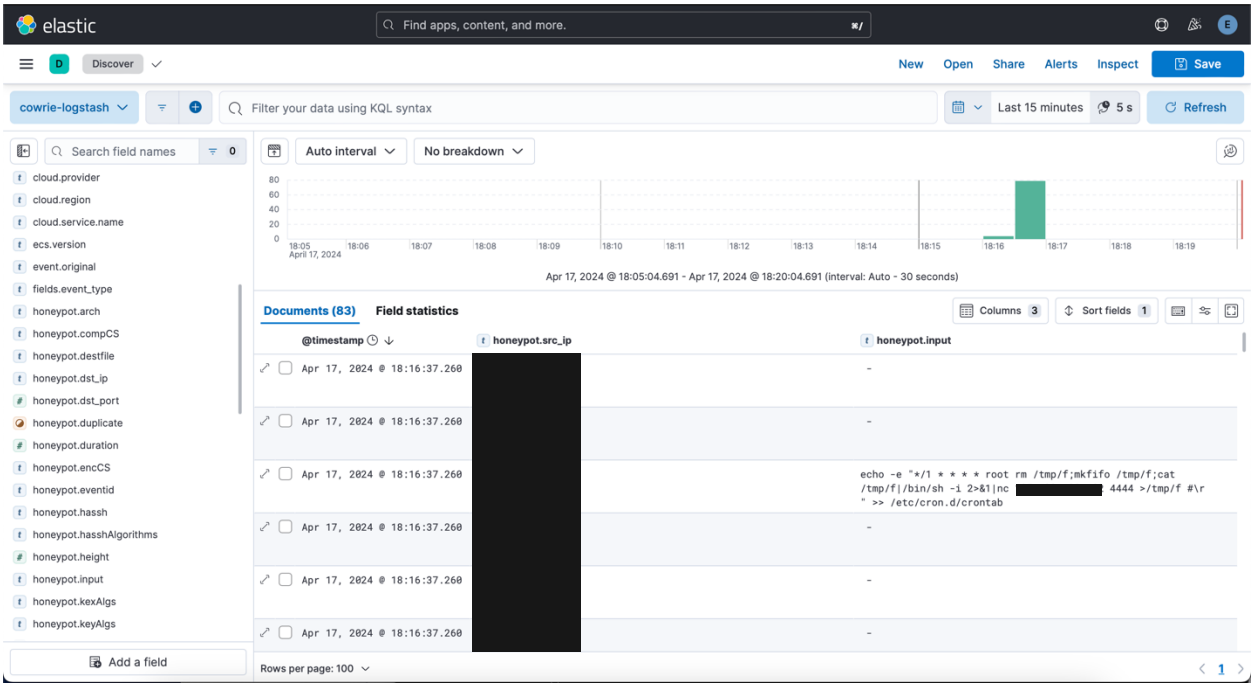


Similarly, if a successful login was made, the same information types can be seen, including which host was compromised if the Elastic Cloud was set to monitor several nodes.

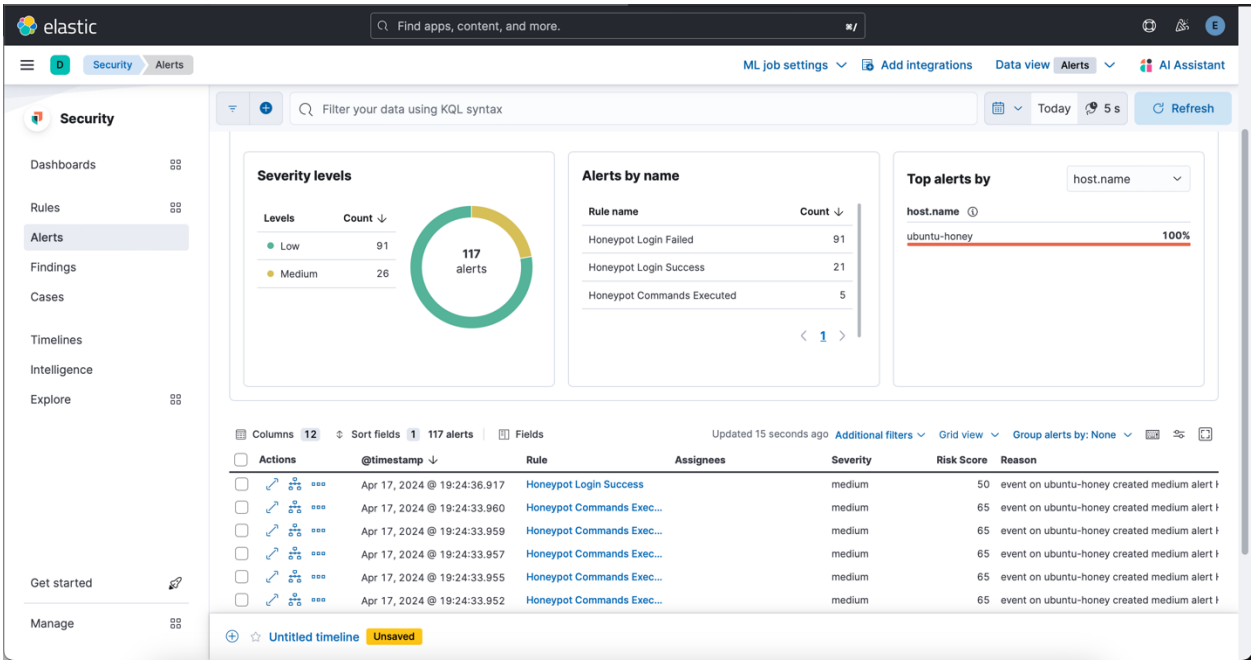
Additionally, alerts appeared according to prior configurations, marking successful detection of the attack.



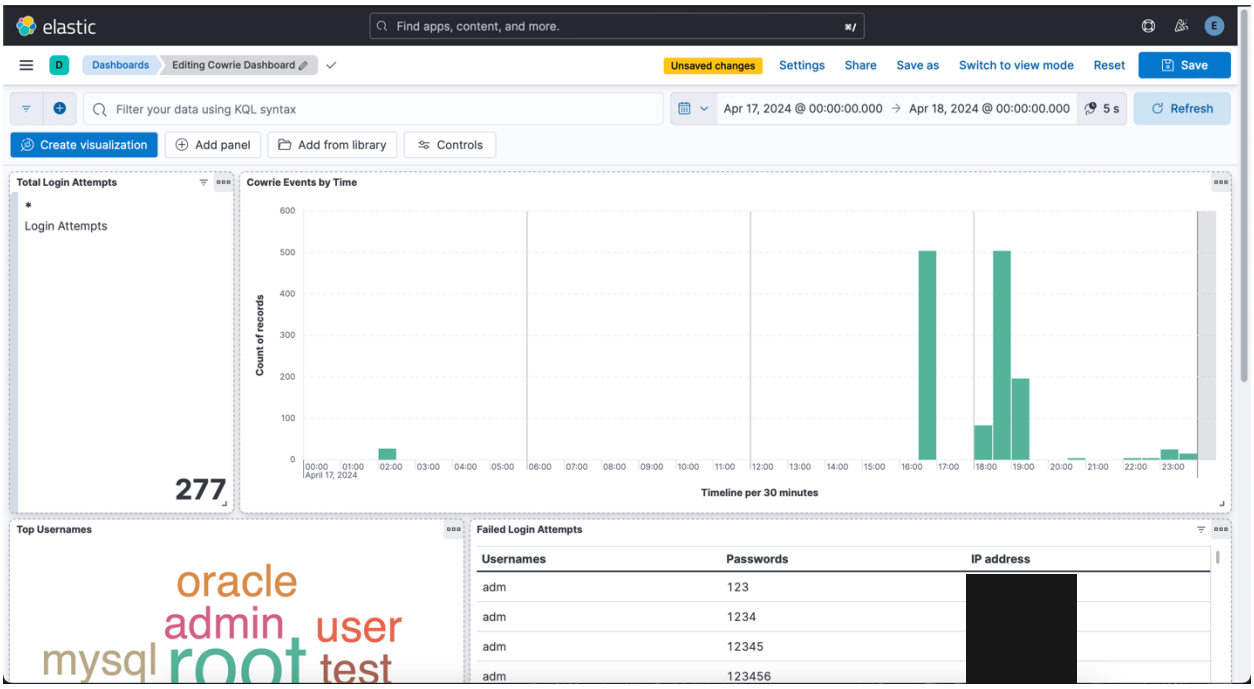
Moreover, the creation of the netcat reverse shell cronjob was detected by the system.

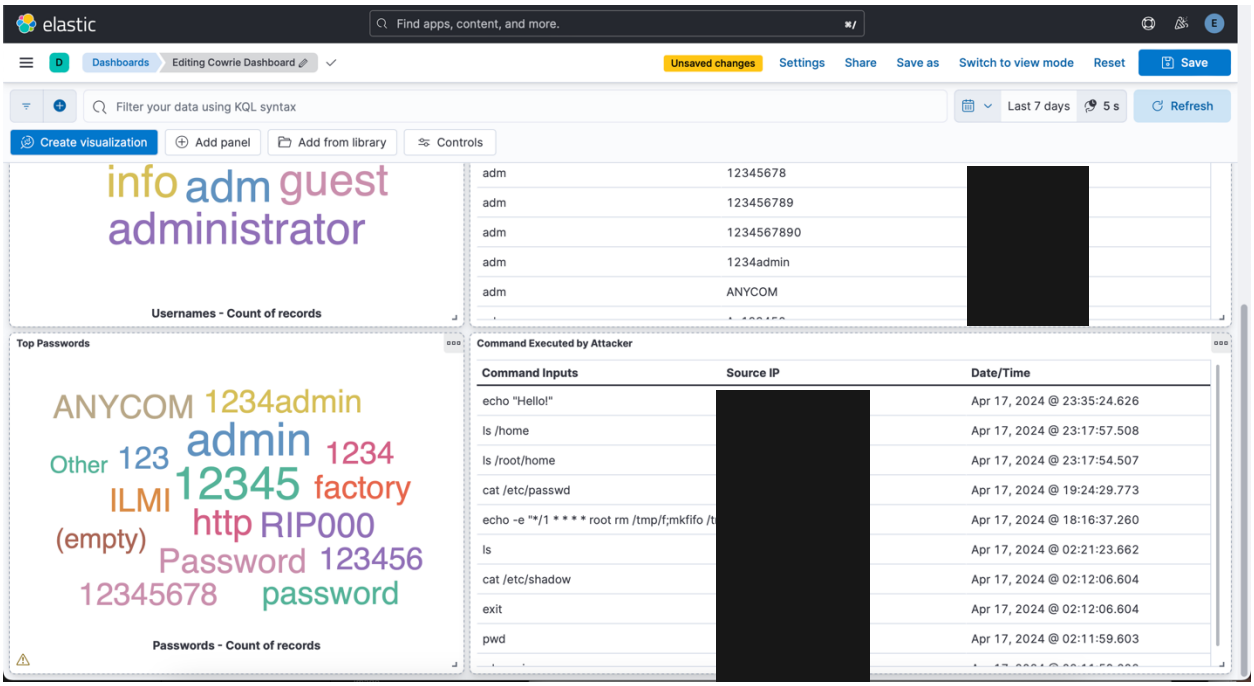


Similarly, a respective alert was given for detecting commands being executed in the honeypot.



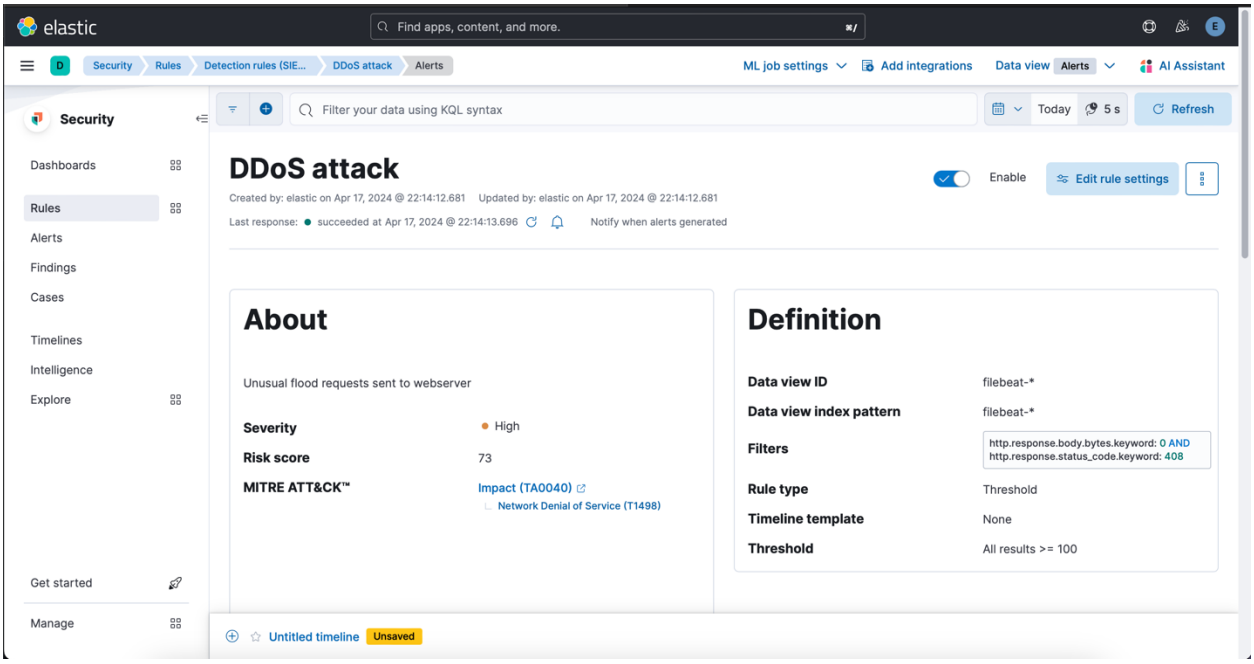
Through the created Cowrie dashboard visualisation, a quick view of the captured attack is provided.



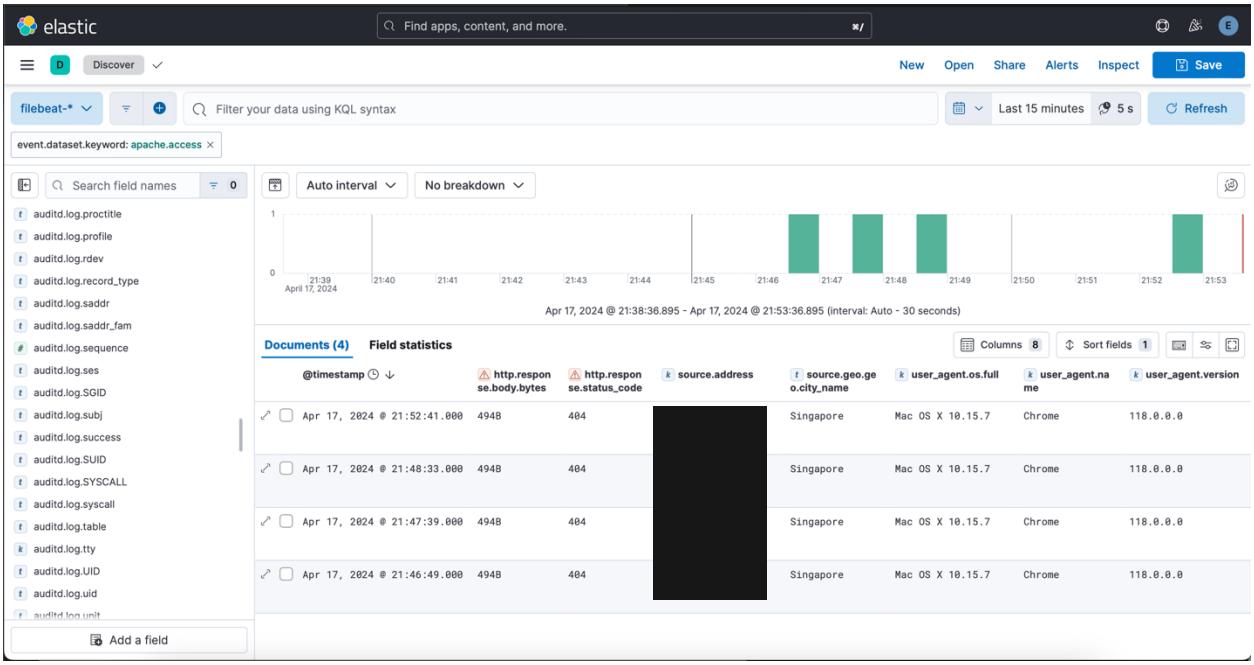


Detecting the DDoS Attack

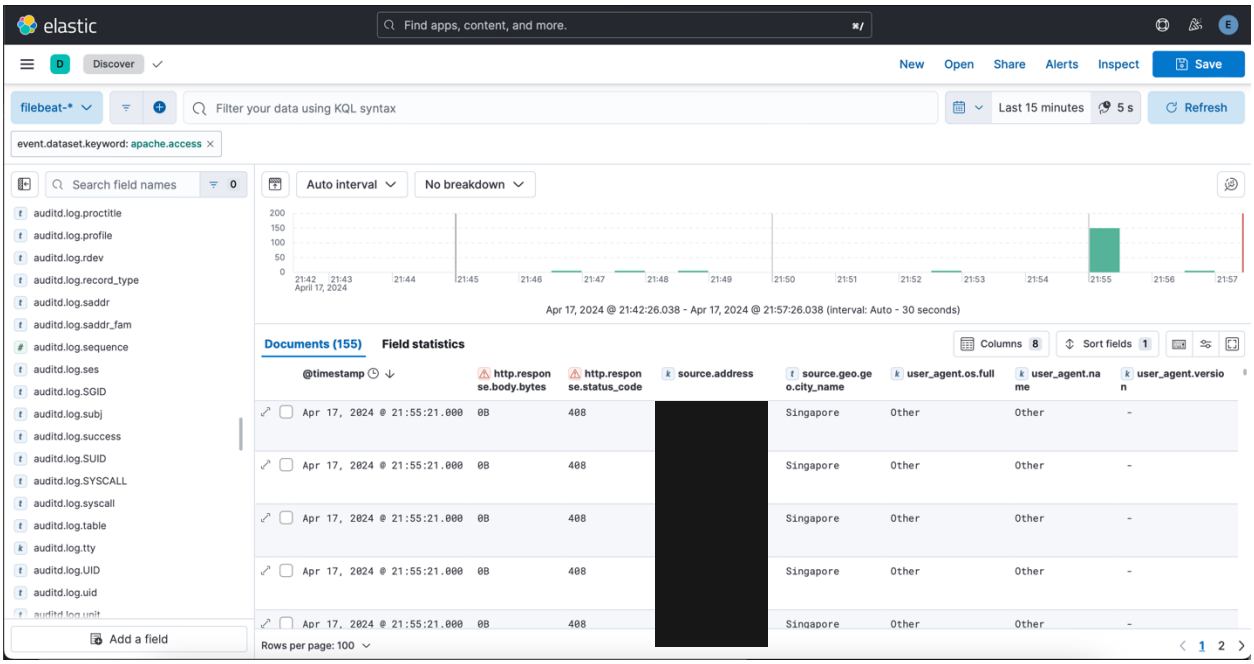
Likewise, a rule was created to detect DDoS attacks and provide an alert.



Logs for web events were logged in the access and errors logs of Apache2 collected through the Apache module of Filebeat. Consequently, they can be found under the Filebeat index pattern in Kibana.

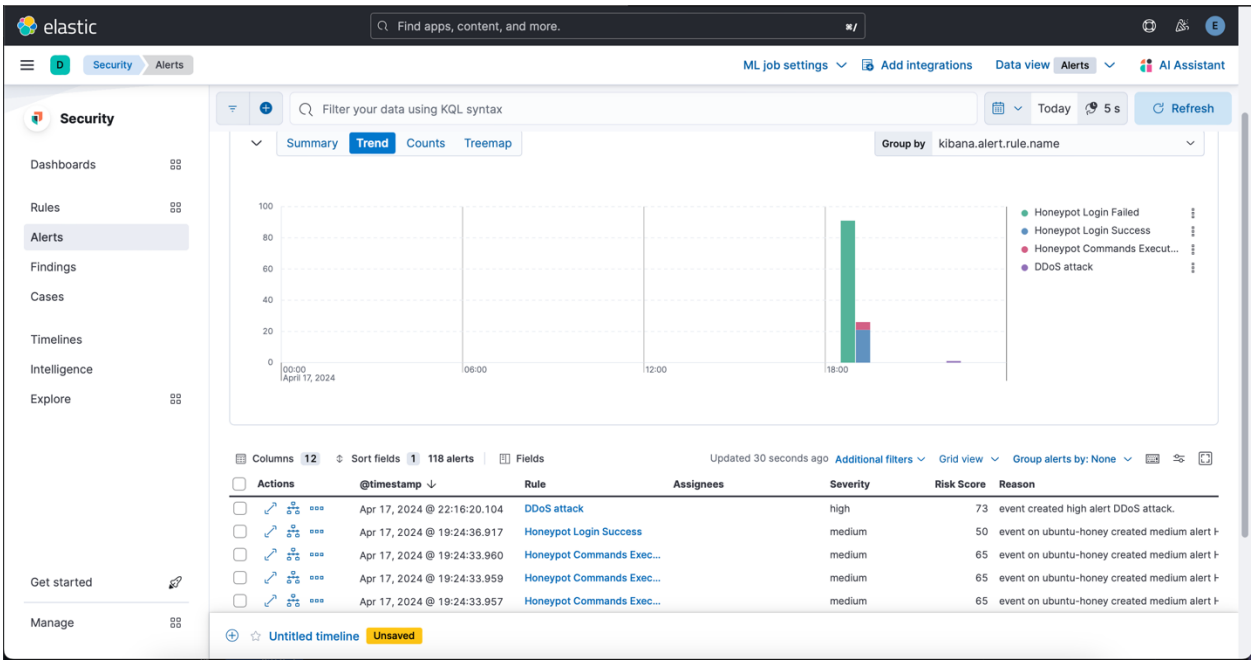


The above depicts logs when a user typically accesses the webpage, displaying the packet size of the http response, the response code transmitted, the source IP address that made the request, it's geolocation, the user OS, and the web browser type and versions, amongst other information available.



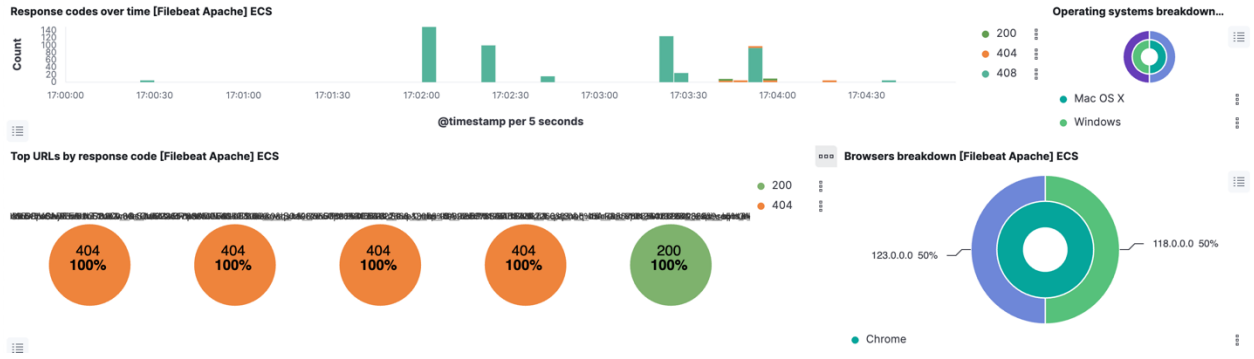
However, when the DDoS was executed with a TCP SYN flood, a burst of log counts was seen in the event timeline. The recorded HTTP response packet size was 0 bytes and a status code of 408 was given, indicating a request timeout. This was expected as the server attempted to close the idle connections that were still waiting for an ACK packet from the attacker which never arrived. Meanwhile the attacker's OS and browser information were hidden, although interestingly the source IP was still captured despite spoofing as shown from the Wireshark traffic.

Using the HTTP response size and status code as the rule for the alert, an alert was issued upon the attack.



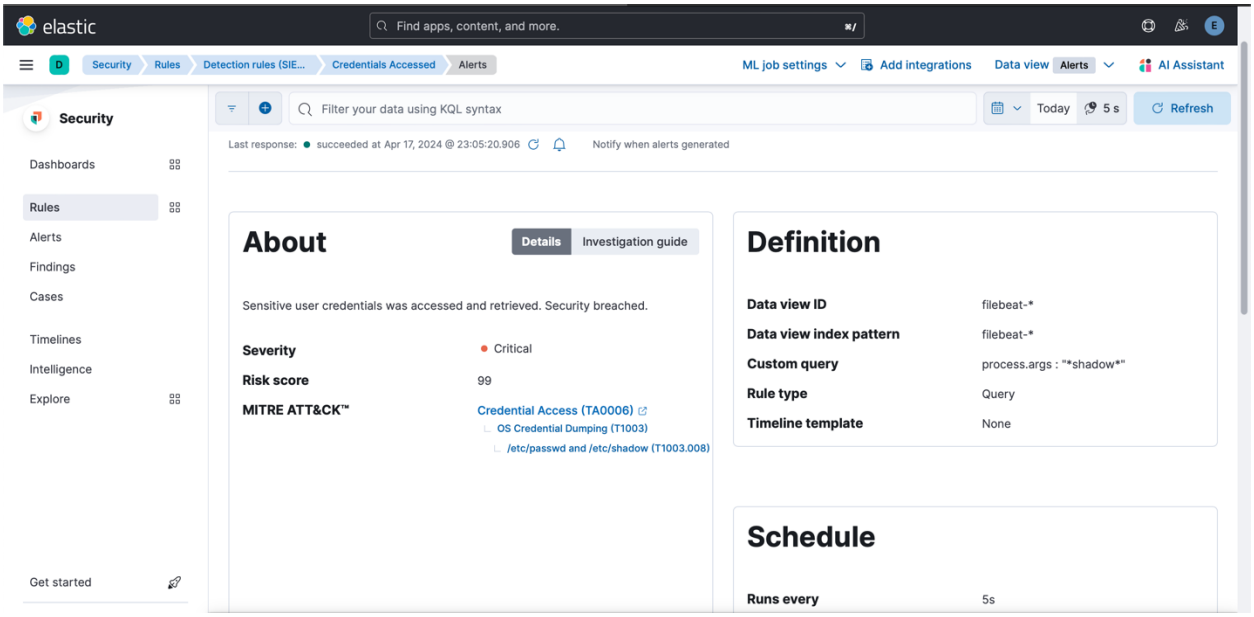
A single alert was issued upon reaching a excessive threshold count, thus accurately detecting a DDoS attack instead of a false positive.

Additionally the Filebeat dashboard was able to visualise the information received, displaying the status codes received and their counts, as well as the browsers and OS that accessed the webpage.

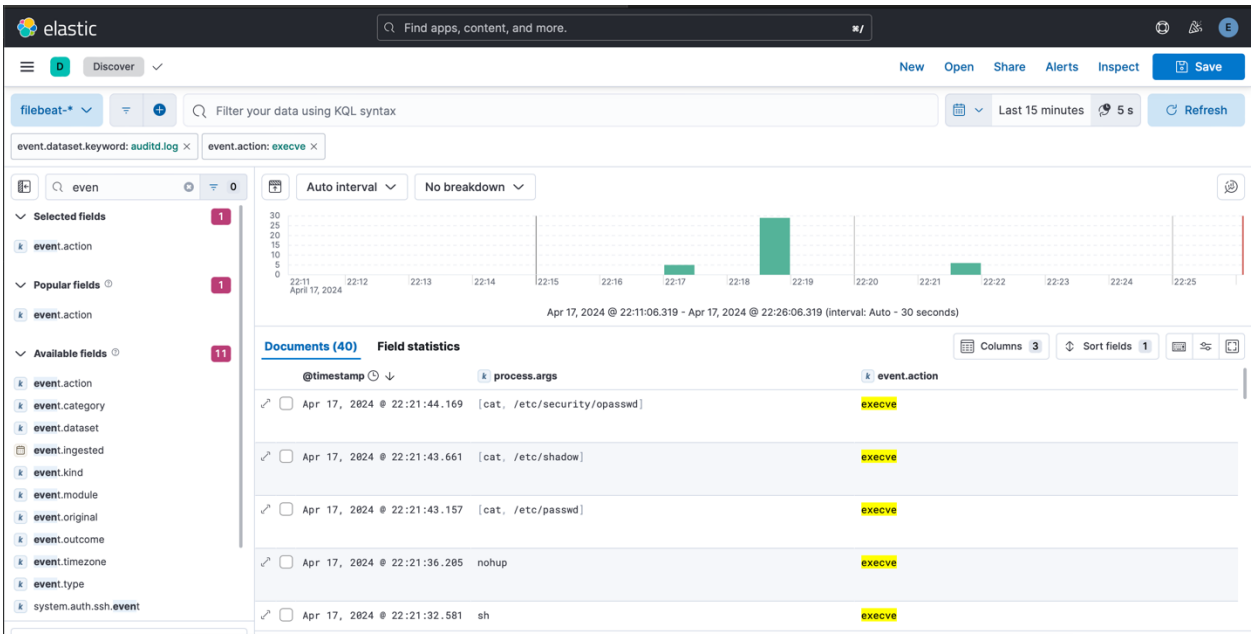


Detecting the Backdoor Command Execution

A rule detecting access to the '*shadow*' file was created to detect for credential dumping.

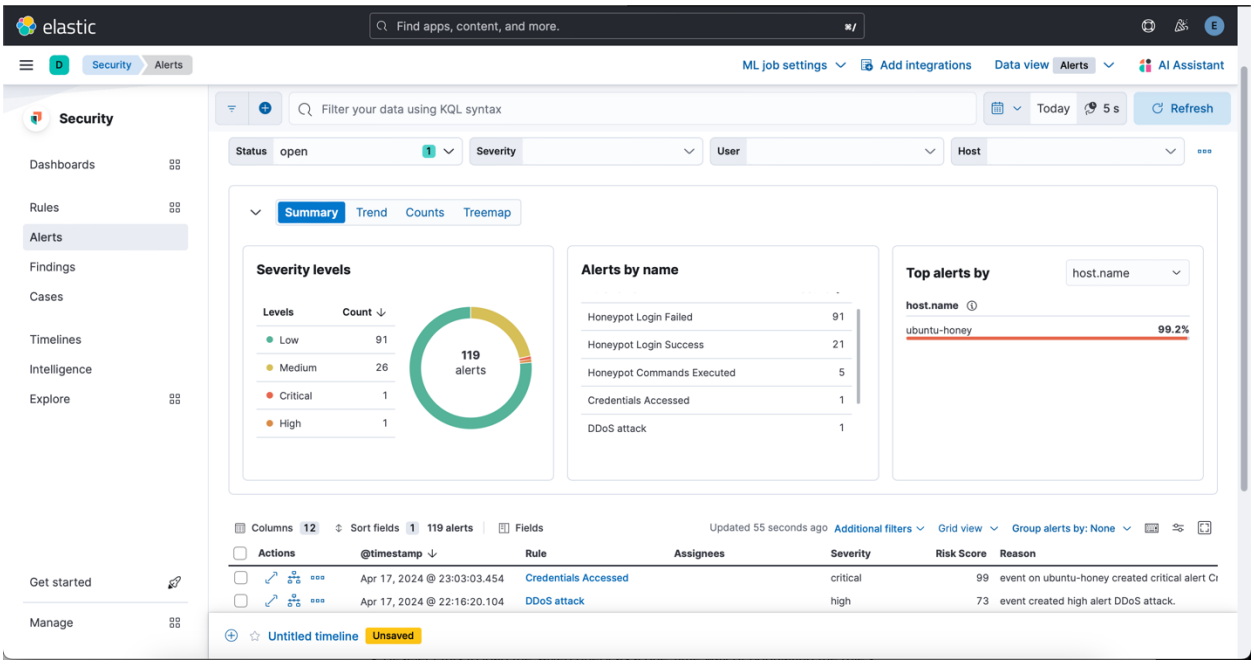


As the executed commands operated on a created shell, these events were logged in the audit.log and transported with the Auditd module of Filebeat. Filtering for the '*EXECVE*' event action under the Filebeat index, the attack sequence was successfully captured as shown below:

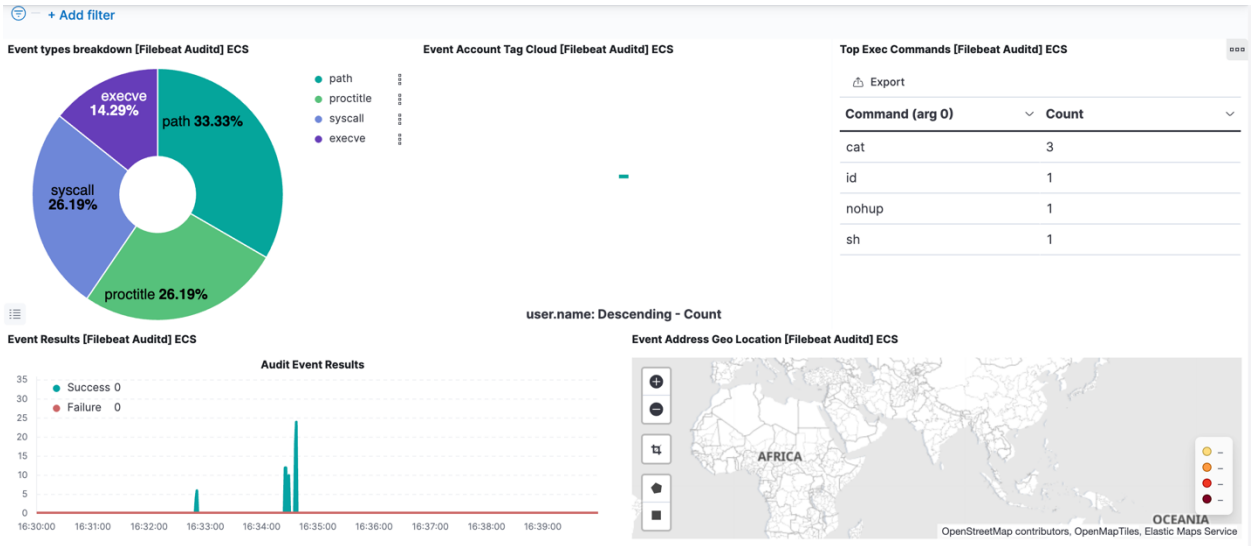


The command to create a shell with '*sh*' could be seen under the '*process.args*' field. About ten seconds later, the '*passwd*', '*shadow*' and '*opasswd*' files were accessed, depicting the exact process of the attack that was executed.

Accordingly, the alert for having the credentials accessed was triggered.



Through the Auditd dashboard a breakdown of the audit events could be viewed, including records of the process arguments entered into the system.



Discussion

The study sought to establish an Elastic Cloud server to monitor a separate client honeypot server and capture attacks executed on it for analysis. The Elastic server was successfully setup for all components to communicate with one another with security enabled. The attacks were also successfully conducted on the honeypot server as intended, with all of it captured and alerted for by the Elastic cloud.

In the first attack, one interesting finding was how the written cronjob to establish the netcat reverse shell connection was fully captured and displayed in the logs. This was in spite of efforts to hide it through obfuscation; running crontab to display the existing cronjobs and concatenate to output the contents of the file would not have revealed its existence. This shows that while measures can be taken to hide such attacks as best possible, it is not impossible to cover its tracks entirely upon thorough investigation and the existence of an effective monitoring system, highlighting the importance of one. Also to avoid being vulnerable to brute force attacks, it is recommended to always set strong user credentials and utilise services such as Fail2ban to disrupt and block attackers.

Elsewhere, the monitoring system was also effective in capturing the occurrence of a DDoS attack visually from the burst of events in the timeline, especially as compared to if one were to directly inspect the log files. Additionally, as mentioned in the results section, while not all information of the DDoS attacker could be obtained from the logs, the source IP address and location was nevertheless revealed. This is important to track down the attacker for subsequent action against them or to protect the server by banning the address. That said, the attack executed from the script was a simple and efficient attack, thus a more comprehensive level of DDoS attack with higher-level IP spoofing would have made it harder trace the attacker. Regardless, a DDoS attack would be detected visually from the traffic spike and from the rules written based on the HTTP response and status code, triggering an alert. Using prevention tools and rate limiting to restrict traffic volume over time periods can

help to mitigate against such attacks, especially against specific IPs or regions. In particular for services unlike webservers where only certain users require access, firewalls can help to control incoming traffic and block malicious HTTP traffic.

It is notable that during the third attack, retrieval of the credential files by the attacker were registered as a concatenate argument. If it was detected as a GET request or an SSH download command for example, then the attack would have been more obvious since any download of these files can be categorised as suspicious. But because the only form of detection was as a concatenate argument, it arises the possibility of false positives wherein an authorised user could have accessed it. Also because the attack pattern was observed over a chain of commands, it could be difficult to identify in an actual production server handling multiple processes or receiving heavy input. Therefore detection may not be as straightforward. Consequently, the best recommendation is to avoid allowing a shell creation to begin with as it is inherently dangerous. In this study, a vulnerable version of VSFTPD was intentionally installed to facilitate such a scenario. This also means that it is important to regularly update system packages and services to protect against known vulnerabilities and exploits, and also keeping up with the latest trends and discoveries as new vulnerabilities may spawn on a daily basis. It thus also demands strict protocols on what is installed on the server to reduce the attack surface, removing unnecessary applications and services, and controlling what is downloaded and run. The existence of a good antivirus system will also help detect for malicious programs executed.

Further recommendations in setting a server to receive attacks for analysis include installing more comprehensive or varied honeypots that would cover a larger attack surface. This allows the attacks to be recorded and isolated from the actual server body, thereby protecting it. In this study, Cowrie only provided an attack surface for the SSH and Telnet services, with the HTTP service and VSFTPD exploit both existing on the actual server. However, one issue encountered in this study was the installation of other honeypots as

most attempts failed due to outdated dependencies and were no longer supported. Going forward, it would thus be more ideal to develop a new honeypot tailored to one's needs. Nonetheless, attackers are increasingly more wary of honeypots these days, and there may be situations or for intentional enticement where the server is designed to receive the attacks itself, such as this study partially was. In such cases, it is recommended for the server to be isolated in a DMZ or lockdown region within the network, for example with the use of firewalls, to prevent the attacker from gaining access to other areas of the network.

Finally, although communication between the Elastic components was secured, this study did not establish security between Kibana and the web browser. In other words, the HTTPS encryption protocol via Transport Layer Security or Secure Sockets Layer (TLS/SSL) was not used to encrypt communications over the network. Establishing this layer of security caused unnecessary hassle to access the Kibana web page for the purposes of this study due to the employment of self-signed certificates which were unrecognised by the web browsers. However, in an actual production server with a fully qualified domain name (FQDN) and external authorities to sign the certificates, it is an essential step to implement to avoid listeners over the network and interception of the packets that could result in information leaks and subsequently security vulnerabilities.

In conclusion, this study demonstrates the importance and effectiveness of a SIEM system deployed to monitor high-value targets in an efficient manner visually, with information filtered according to situational requirements, and alerts for quick notifications and immediate response.

References

[1] D. Ogaro, “Setting up a secure Elasticsearch pipeline for logs analysis.” *Medium*, Mar. 24, 2022. https://medium.com/@ogaro/setting-up-a-secure-elasticsearch-pipeline-for-logs-analysis-c466e7c9f228 (accessed Apr. 19, 2024).

[2] jnimmo, “Logstash Ingress Filter - Apache (for Filebeat Apache module).” *Github*. https://gist.github.com/jnimmo/baa22f115e78646951a7f1e63d11a55b (accessed Apr. 19, 2024).

[3] Elastic, “Configure the Logstash output | Filebeat Reference [8.7] | Elastic.” *www.elastic.co*. https://www.elastic.co/guide/en/beats/filebeat/current/logstash-output.html (accessed Apr. 19, 2024).

[4] M. Oosterhof, “Cowrie.” *Github*. https://github.com/cowrie/cowrie (accessed Apr. 19, 2024

[5] M. Oosterhof, “Installing Cowrie in seven steps — cowrie 2.5.0 documentation.” *cowrie.readthedocs.io*. https://cowrie.readthedocs.io/en/latest/INSTALL.html#step-1-install-system-dependencies (accessed Apr. 19, 2024).

[6] M. Oosterhof, “cowrie/docs/elk/logstash-cowrie.conf.” *GitHub*. https://github.com/cowrie/cowrie/blob/master/docs/elk/logstash-cowrie.conf (accessed Apr. 19, 2024).

[7] C. Evans, “vsftpd-2.3.4-infected.” *GitHub*. https://github.com/nikdubois/vsftpd-2.3.4-infected/blob/vsftpd_original/INSTALL (accessed Apr. 19, 2024).

[8] barisbogdan, “danielmiessler/SecLists/Passwords/Default-Credentials/default-passwords.txt.” *GitHub*. https://github.com/danielmiessler/SecLists/blob/master/Passwords/Default-Credentials/default-passwords.txt (accessed Apr. 19, 2024).

[9] amine, “Create Bind and Reverse Shells using Netcat.” *patchthenet.com*. https://patchthenet.com/blog/create-bind-and-reverse-shells-using-netcat (accessed Apr. 19, 2024).

[10] Bret, “RedTeam Tip: Hiding Cronjobs.” *Cyber Gladius*. https://cybergladius.com/redteam-tip-hiding-cronjobs/ (accessed Apr. 19, 2024).