Experience Developing an Intrusion Detection System

Member Name and student ID(Printed)	Signature
Trevor Wachaga 13058317	[illegible]

Contents

Introduction

This report consists of the practical experience of creating a Network Intrusion Detection System. The system was created on Kali Linux and the following guide was used. "***Perform network intrusion detection by using Azure Network Watcher and open-source tools***" (https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-intrusion-detection-open-source-tools).

Furthermore, this experiment meant using unfamiliar technology so using the guide was a must. Whilst following the guide, research was also conducted so an understanding of what an intrusion detection system was and how an ELK (Elastic, Logstash and Kibana) stack helps achieve that.

This report details the steps taken to create an intrusion detection system, and goes through processes, challenges and evaluations of the experiment. A reflection was also done to conclude the experience.

Developing an Intrusion Detection System

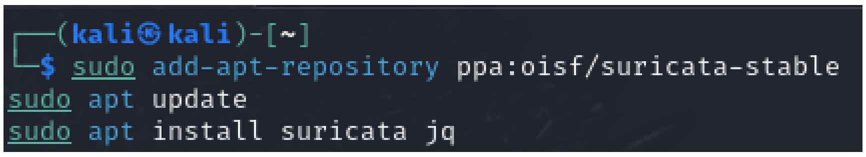
Below highlights the steps taken to successfully create the system, as well as any findings or shortcomings that were faced.

Suricata

This intrusion detection system platform is used for inspecting packets and creating alert logs whenever there is a threat towards the network. For this system that was created, there were two processes taken with Suricata: installing and configuring.

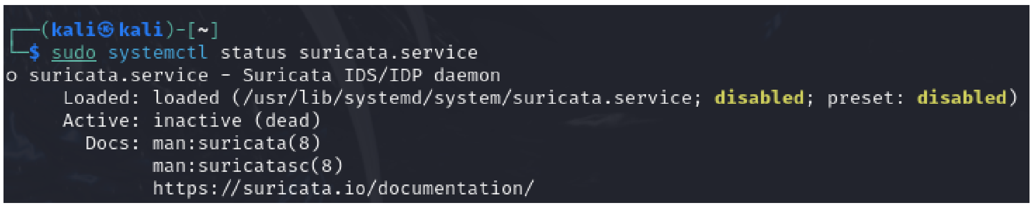
Installing Suricata

The command used to install Suricata was:



```
┌──(kali㉿kali)-[~]
└─$ sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt update
sudo apt install suricata jq
```

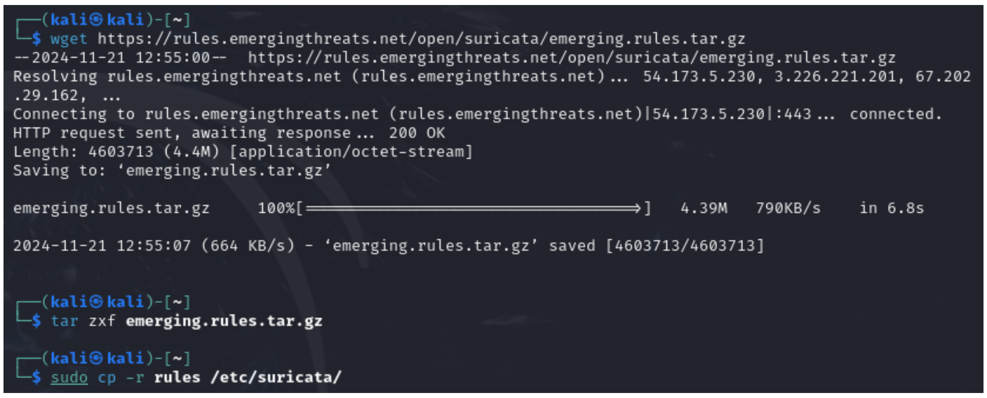
To confirm it had been installed, this command was used:



```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl status suricata.service
○ suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/usr/lib/systemd/system/suricata.service; disabled; preset: disabled)
     Active: inactive (dead)
       Docs: man:suricata(8)
             man:suricatasc(8)
             https://suricata.io/documentation/
```

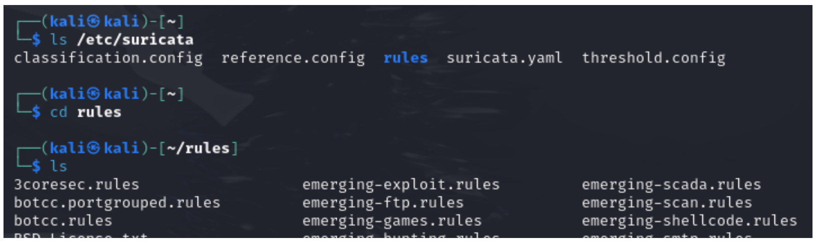
Now it was installed, it was time to install some threats that Suricata would notice.

The emerging threats file was installed, extracted and a rules directory was copied into '/etc/Suricata' using the following commands:



```
┌──(kali㉿kali)-[~]
└─$ wget https://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
--2024-11-21 12:55:00--  https://rules.emergingthreats.net/open/suricata/emerging.rules.tar.gz
Resolving rules.emergingthreats.net (rules.emergingthreats.net)... 54.173.5.230, 3.226.221.201, 67.202
.29.162, ...
Connecting to rules.emergingthreats.net (rules.emergingthreats.net)|54.173.5.230|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 4603713 (4.4M) [application/octet-stream]
Saving to: ‘emerging.rules.tar.gz’

emerging.rules.tar.gz     100%[===================================>]   4.39M   790KB/s    in 6.8s

2024-11-21 12:55:07 (664 KB/s) - ‘emerging.rules.tar.gz’ saved [4603713/4603713]


┌──(kali㉿kali)-[~]
└─$ tar zxf emerging.rules.tar.gz

┌──(kali㉿kali)-[~]
└─$ sudo cp -r rules /etc/suricata/
```

Below shows confirmation of those rules:



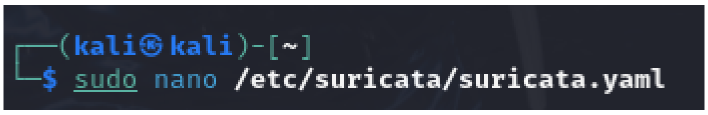
```
┌──(kali㉿kali)-[~]
└─$ ls /etc/suricata
classification.config  reference.config  rules  suricata.yaml  threshold.config

┌──(kali㉿kali)-[~]
└─$ cd rules

┌──(kali㉿kali)-[~/rules]
└─$ ls
3coresec.rules               emerging-exploit.rules       emerging-scada.rules
botcc.portgrouped.rules      emerging-ftp.rules           emerging-scan.rules
botcc.rules                  emerging-games.rules         emerging-shellcode.rules
```

Next was configuring an important Suricata file.

Configuring Suricata

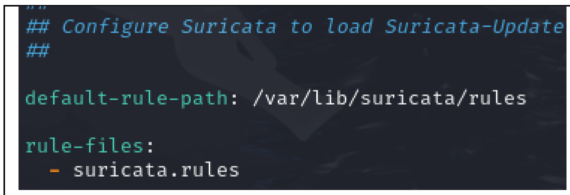
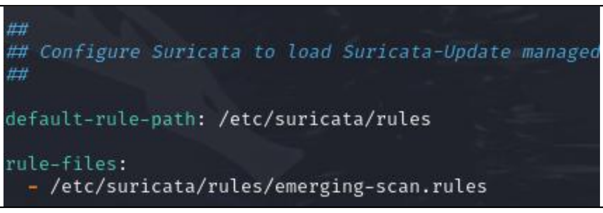
To set up Suricata, the ‘/etc/suricata/suricata.yaml’ file had to be configured, so it could use the right rules, analyse the right network, and output information to the correct directory.

The command below was how to edit the file that would be used for :



```
┌──(kali㉿kali)-[~]
└─$ sudo nano /etc/suricata/suricata.yaml
```

For the rules, the path for the emerging threats rules that had been installed were used. Below shows the changes made:

	
Before	After

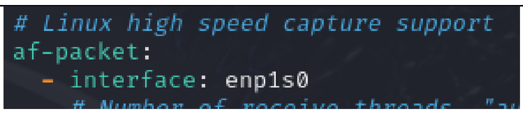
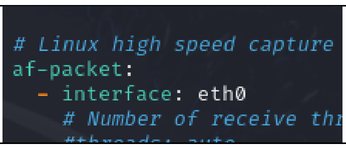
Before:

```
## Configure Suricata to load Suricata-Update
##

default-rule-path: /var/lib/suricata/rules

rule-files:
  - suricata.rules
```

After:

```
##
## Configure Suricata to load Suricata-Update managed
##

default-rule-path: /etc/suricata/rules

rule-files:
  - /etc/suricata/rules/emerging-scan.rules
```

For the right network to be analysed, below shows the interface was changed from ‘enp1s0’ to ‘eth0’ (the network to be used).

	
Before	After

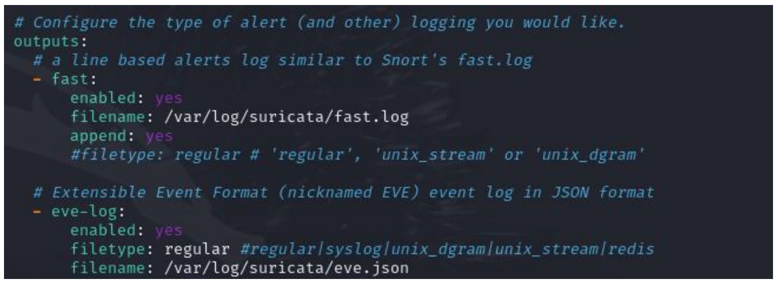
Before:

```
# Linux high speed capture support
af-packet:
  - interface: enp1s0
```

After:

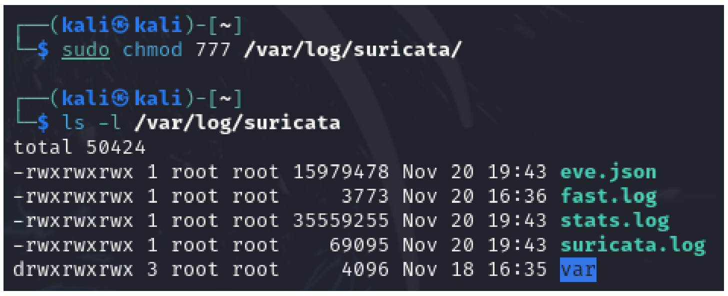
```
# Linux high speed capture
af-packet:
  - interface: eth0
    # Number of receive
```

For outputting information to the right directory and file, below shows the correct outputs where Suricata sends the alerts.



```
# Configure the type of alert (and other) logging you would like.
outputs:
  # a line based alerts log similar to Snort's fast.log
  - fast:
      enabled: yes
      filename: /var/log/suricata/fast.log
      append: yes
      #filetype: regular # 'regular', 'unix_stream' or 'unix_dgram'

  # Extensible Event Format (nicknamed EVE) event log in JSON format
  - eve-log:
      enabled: yes
      filetype: regular #regular|syslog|unix_dgram|unix_stream|redis
      filename: /var/log/suricata/eve.json
```

Furthermore, in order for the outputs to be able to be written to by Suricata, permissions needed to be changed so, the following command was run for the output directory:

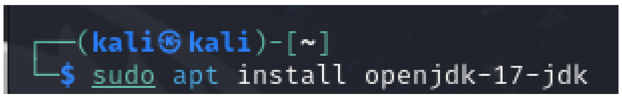


```
┌──(kali㉿kali)-[~]
└─$ sudo chmod 777 /var/log/suricata/

┌──(kali㉿kali)-[~]
└─$ ls -l /var/log/suricata
total 50424
-rwxrwxrwx 1 root root 15979478 Nov 20 19:43 eve.json
-rwxrwxrwx 1 root root     3773 Nov 20 16:36 fast.log
-rwxrwxrwx 1 root root 35559255 Nov 20 19:43 stats.log
-rwxrwxrwx 1 root root    69095 Nov 20 19:43 suricata.log
drwxrwxrwx 3 root root     4096 Nov 18 16:35 var
```

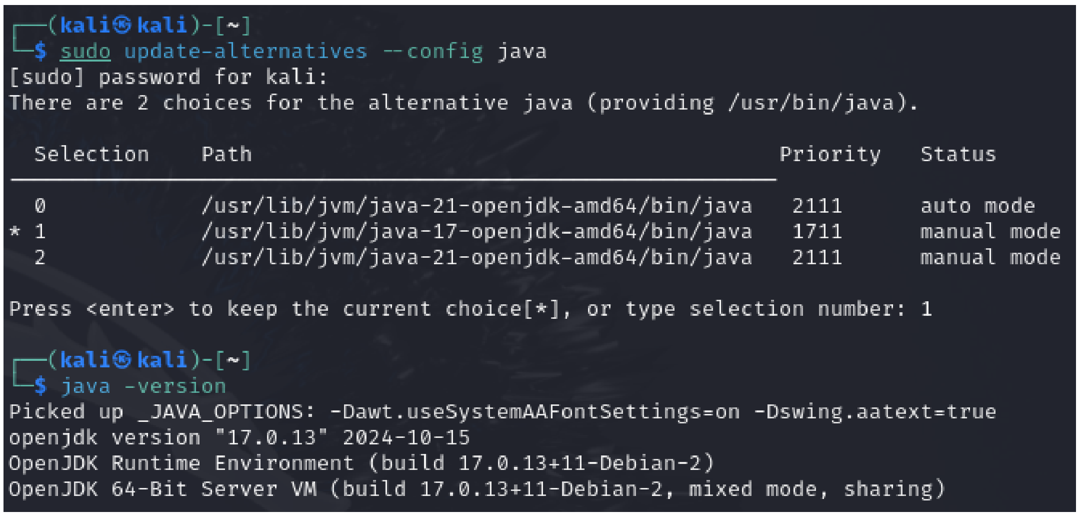
The installation and configuration was done, so doing the same for the other platforms was next.

Java

Java is necessary for the system to work as the other platforms require it. Furthermore, a specific Java version was needed, and this was JDK 17. Below shows the installation and confirmation of the correct version required.



```
┌──(kali㉿kali)-[~]
└─$ sudo apt install openjdk-17-jdk
```



```
┌──(kali㉿kali)-[~]
└─$ sudo update-alternatives --config java
[sudo] password for kali:
There are 2 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                         Priority   Status
────────────────────────────────────────────────────────────
  0            /usr/lib/jvm/java-21-openjdk-amd64/bin/java   2111      auto mode
* 1            /usr/lib/jvm/java-17-openjdk-amd64/bin/java   1711      manual mode
  2            /usr/lib/jvm/java-21-openjdk-amd64/bin/java   2111      manual mode

Press <enter> to keep the current choice[*], or type selection number: 1

┌──(kali㉿kali)-[~]
└─$ java -version
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
openjdk version "17.0.13" 2024-10-15
OpenJDK Runtime Environment (build 17.0.13+11-Debian-2)
OpenJDK 64-Bit Server VM (build 17.0.13+11-Debian-2, mixed mode, sharing)
```

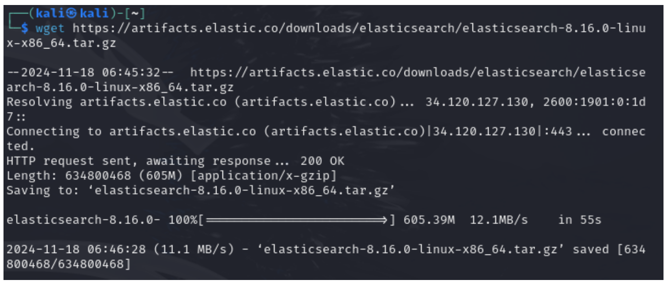
Elasticsearch

Elasticsearch deals with the logs from Suricata and can filter and query. Essentially, it is a database for large scale log data.

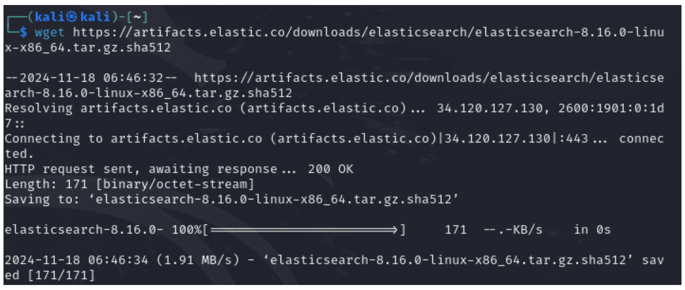
To get Elasticsearch running, installing and configuration had to be done.

Installing Elasticsearch

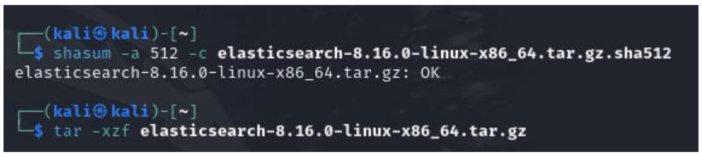
Elasticsearch was installed through an archive for Kali Linux, and below shows the commands used:



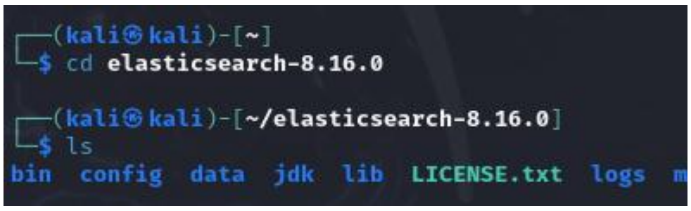
```
┌──(kali㉿kali)-[~]
└─$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.16.0-linu
x-x86_64.tar.gz

--2024-11-18 06:45:32--  https://artifacts.elastic.co/downloads/elasticsearch/elasticse
arch-8.16.0-linux-x86_64.tar.gz
Resolving artifacts.elastic.co (artifacts.elastic.co)... 34.120.127.130, 2600:1901:0:1d
7::
Connecting to artifacts.elastic.co (artifacts.elastic.co)|34.120.127.130|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 634800468 (605M) [application/x-gzip]
Saving to: ‘elasticsearch-8.16.0-linux-x86_64.tar.gz’

elasticsearch-8.16.0- 100%[===================>] 605.39M  12.1MB/s    in 55s

2024-11-18 06:46:28 (11.1 MB/s) - ‘elasticsearch-8.16.0-linux-x86_64.tar.gz’ saved [634
800468/634800468]
```



```
┌──(kali㉿kali)-[~]
└─$ wget https://artifacts.elastic.co/downloads/elasticsearch/elasticsearch-8.16.0-linu
x-x86_64.tar.gz.sha512

--2024-11-18 06:46:32--  https://artifacts.elastic.co/downloads/elasticsearch/elasticse
arch-8.16.0-linux-x86_64.tar.gz.sha512
Resolving artifacts.elastic.co (artifacts.elastic.co)... 34.120.127.130, 2600:1901:0:1d
7::
Connecting to artifacts.elastic.co (artifacts.elastic.co)|34.120.127.130|:443... connec
ted.
HTTP request sent, awaiting response... 200 OK
Length: 171 [binary/octet-stream]
Saving to: ‘elasticsearch-8.16.0-linux-x86_64.tar.gz.sha512’

elasticsearch-8.16.0- 100%[===================>]     171  --.-KB/s    in 0s

2024-11-18 06:46:34 (1.91 MB/s) - ‘elasticsearch-8.16.0-linux-x86_64.tar.gz.sha512’ sav
ed [171/171]
```



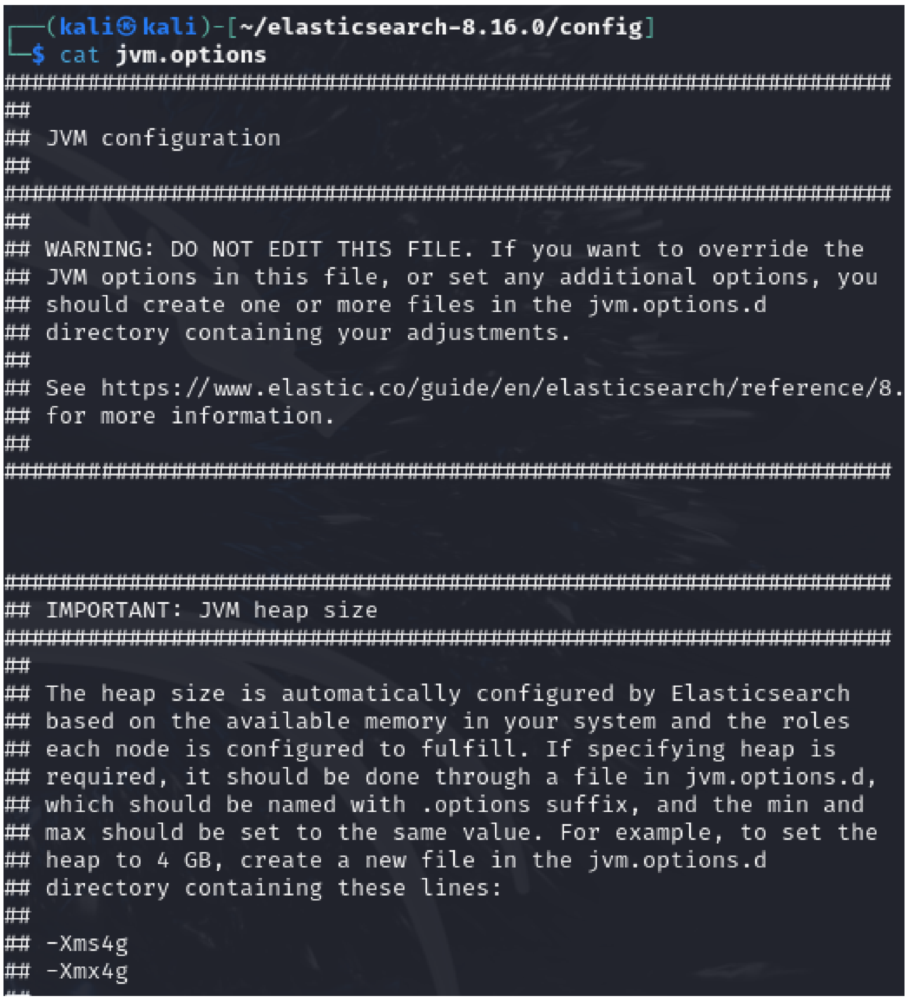
```
┌──(kali㉿kali)-[~]
└─$ shasum -a 512 -c elasticsearch-8.16.0-linux-x86_64.tar.gz.sha512
elasticsearch-8.16.0-linux-x86_64.tar.gz: OK

┌──(kali㉿kali)-[~]
└─$ tar -xzf elasticsearch-8.16.0-linux-x86_64.tar.gz
```



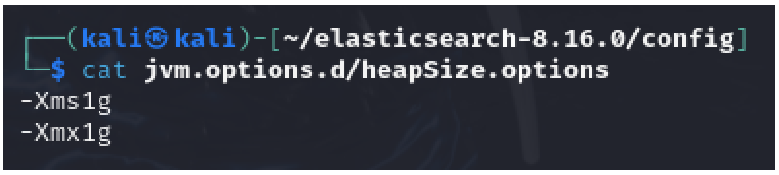
```
┌──(kali㉿kali)-[~]
└─$ cd elasticsearch-8.16.0

┌──(kali㉿kali)-[~/elasticsearch-8.16.0]
└─$ ls
bin  config  data  jdk  lib  LICENSE.txt  logs  m
```

Configuring Elasticsearch

Configuration was necessary as there were some files that needed modifying to address memory issues.



```
┌──(kali㉿kali)-[~/elasticsearch-8.16.0/config]
└─$ cat jvm.options
################################################################
##
## JVM configuration
##
################################################################
##
## WARNING: DO NOT EDIT THIS FILE. If you want to override the
## JVM options in this file, or set any additional options, you
## should create one or more files in the jvm.options.d
## directory containing your adjustments.
##
## See https://www.elastic.co/guide/en/elasticsearch/reference/8.
## for more information.
##
################################################################



################################################################
## IMPORTANT: JVM heap size
################################################################
##
## The heap size is automatically configured by Elasticsearch
## based on the available memory in your system and the roles
## each node is configured to fulfill. If specifying heap is
## required, it should be done through a file in jvm.options.d,
## which should be named with .options suffix, and the min and
## max should be set to the same value. For example, to set the
## heap to 4 GB, create a new file in the jvm.options.d
## directory containing these lines:
##
## -Xms4g
## -Xmx4g
```



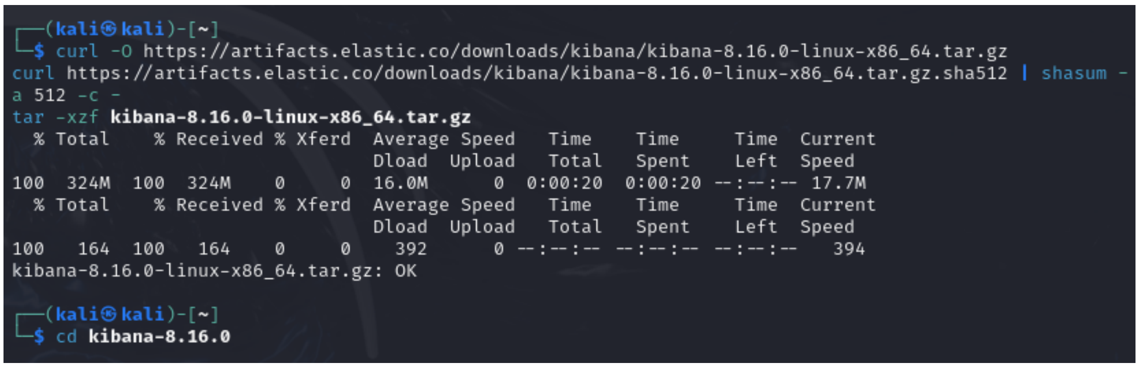
```
┌──(kali㉿kali)-[~/elasticsearch-8.16.0/config]
└─$ cat jvm.options.d/heapSize.options
-Xms1g
-Xmx1g
```

Kibana

Next was getting Kibana, which is used for visualization and a dashboard for Elasticsearch. It allows users to view data through graphs and charts, and monitor activity.

Installing Kibana

Like Elasticsearch, this was installed through the archive for Kali Linux and below shows the commands used to get it:



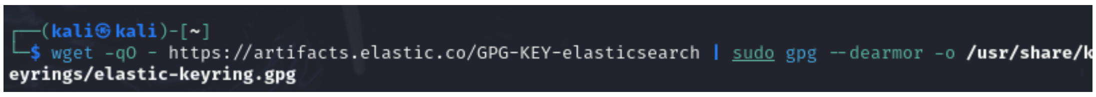
```
┌──(kali㉿kali)-[~]
└─$ curl -O https://artifacts.elastic.co/downloads/kibana/kibana-8.16.0-linux-x86_64.tar.gz
curl https://artifacts.elastic.co/downloads/kibana/kibana-8.16.0-linux-x86_64.tar.gz.sha512 | shasum -
a 512 -c -
tar -xzf kibana-8.16.0-linux-x86_64.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  324M  100  324M    0     0  16.0M      0  0:00:20  0:00:20 --:--:-- 17.7M
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   164  100   164    0     0    392      0 --:--:-- --:--:-- --:--:--   394
kibana-8.16.0-linux-x86_64.tar.gz: OK

┌──(kali㉿kali)-[~]
└─$ cd kibana-8.16.0
```

<u>Logstash</u>

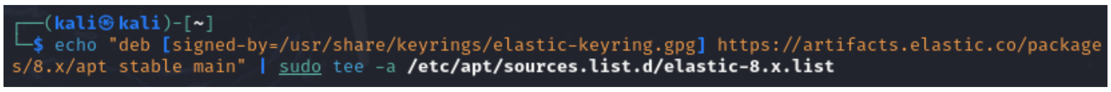
Logstash is a pipeline tools that collects and processes logs, and in this case, forwards them to Elasticsearch. Logstash collects these logs from Suricata and transforms them.

Installing Logstash

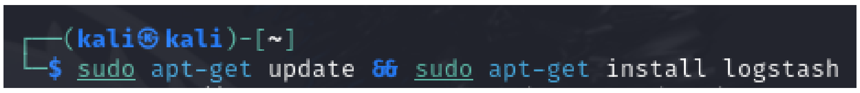
This program was installed through the APT repository and the commands used were:



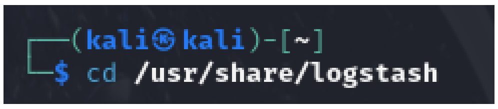
```
┌──(kali㉿kali)-[~]
└─$ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor -o /usr/share/k
eyrings/elastic-keyring.gpg
```



```
┌──(kali㉿kali)-[~]
└─$ echo "deb [signed-by=/usr/share/keyrings/elastic-keyring.gpg] https://artifacts.elastic.co/package
s/8.x/apt stable main" | sudo tee -a /etc/apt/sources.list.d/elastic-8.x.list
```



```
┌──(kali㉿kali)-[~]
└─$ sudo apt-get update && sudo apt-get install logstash
```



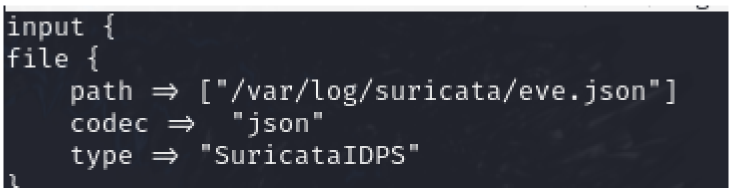
```
┌──(kali㉿kali)-[~]
└─$ cd /usr/share/logstash
```

Configuring Logstash

To get alerts from Suricata, configuration was necessary, and a file called ‘logstash.conf’ (found in Appendix A) was created in ‘/etc/logstash/conf.d’.

There were two sections of this that needed to be edited: input and output sections.

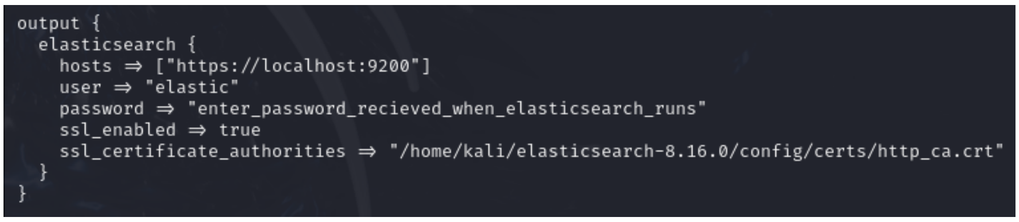
The modification for the input is shown below:



```
input {
file {
    path => ["/var/log/suricata/eve.json"]
    codec =>  "json"
    type => "SuricataIDPS"
}
```

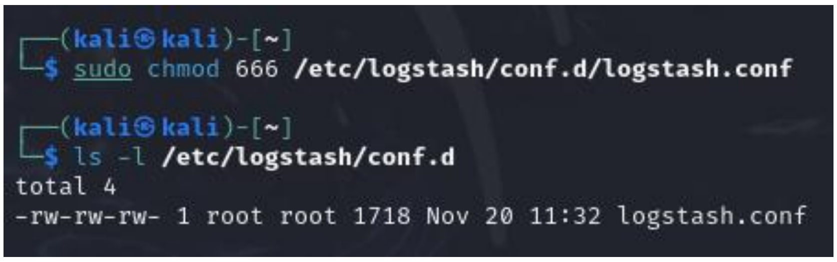
The path has to be from the ‘eve.json’ file that stores alerts/logs that Suricata has collected.

The modification for the output is shown below:



```
output {
  elasticsearch {
    hosts => ["https://localhost:9200"]
    user => "elastic"
    password => "enter_password_recieved_when_elasticsearch_runs"
    ssl_enabled => true
    ssl_certificate_authorities => "/home/kali/elasticsearch-8.16.0/config/certs/http_ca.crt"
  }
}
```

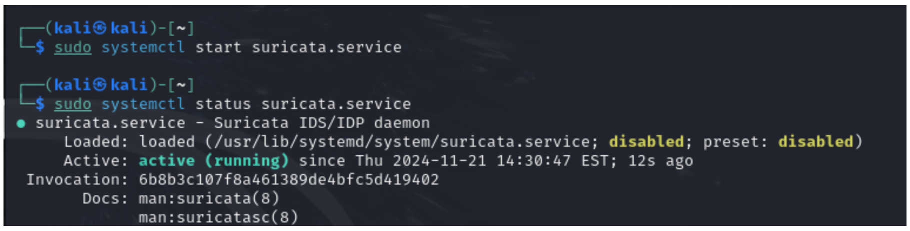
After saving the file, the permissions were changed so that they were able to be read by anyone, including Elasticsearch.



```
┌──(kali㉿kali)-[~]
└─$ sudo chmod 666 /etc/logstash/conf.d/logstash.conf

┌──(kali㉿kali)-[~]
└─$ ls -l /etc/logstash/conf.d
total 4
-rw-rw-rw- 1 root root 1718 Nov 20 11:32 logstash.conf
```

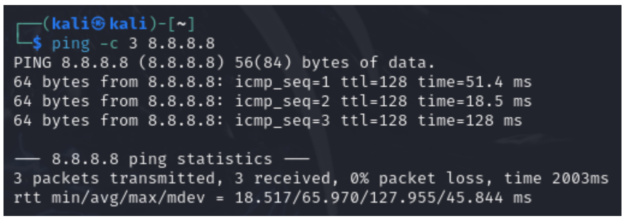
Running the System

Suricata was the first program that needed to run, and it was run as a service using the command below:



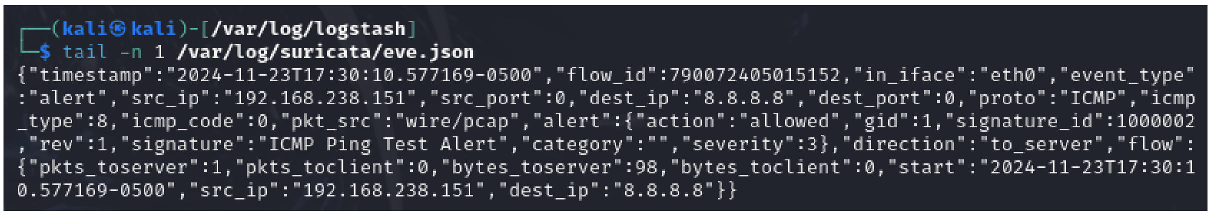
```
┌──(kali㉿kali)-[~]
└─$ sudo systemctl start suricata.service

┌──(kali㉿kali)-[~]
└─$ sudo systemctl status suricata.service
● suricata.service - Suricata IDS/IDP daemon
     Loaded: loaded (/usr/lib/systemd/system/suricata.service; disabled; preset: disabled)
     Active: active (running) since Thu 2024-11-21 14:30:47 EST; 12s ago
 Invocation: 6b8b3c107f8a461389de4bfc5d419402
       Docs: man:suricata(8)
             man:suricatasc(8)
```

To ensure Suricata was configured correctly, I did a scan with the command below:

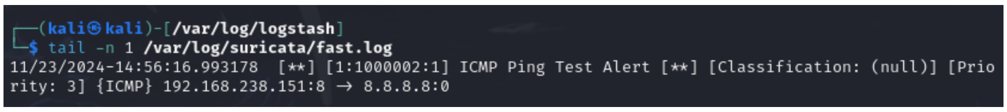


```
┌──(kali㉿kali)-[~]
└─$ ping -c 3 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=51.4 ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=18.5 ms
64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=128 ms

── 8.8.8.8 ping statistics ──
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 18.517/65.970/127.955/45.844 ms
```

This was the result, meaning Suricata was successfully configured and working as it should:

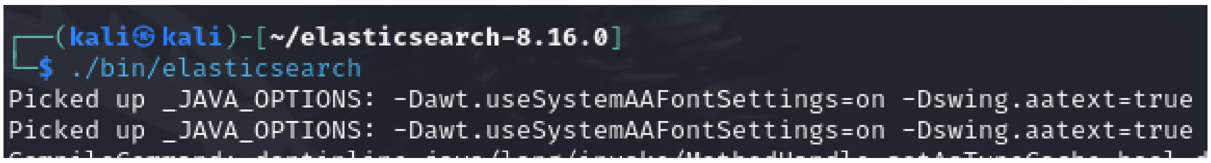


```
┌──(kali㉿kali)-[/var/log/logstash]
└─$ tail -n 1 /var/log/suricata/eve.json
{"timestamp":"2024-11-23T17:30:10.577169-0500","flow_id":790072405015152,"in_iface":"eth0","event_type"
:"alert","src_ip":"192.168.238.151","src_port":0,"dest_ip":"8.8.8.8","dest_port":0,"proto":"ICMP","icmp
_type":8,"icmp_code":0,"pkt_src":"wire/pcap","alert":{"action":"allowed","gid":1,"signature_id":1000002
,"rev":1,"signature":"ICMP Ping Test Alert","category":"","severity":3},"direction":"to_server","flow":
{"pkts_toserver":1,"pkts_toclient":0,"bytes_toserver":98,"bytes_toclient":0,"start":"2024-11-23T17:30:1
0.577169-0500","src_ip":"192.168.238.151","dest_ip":"8.8.8.8"}}
```



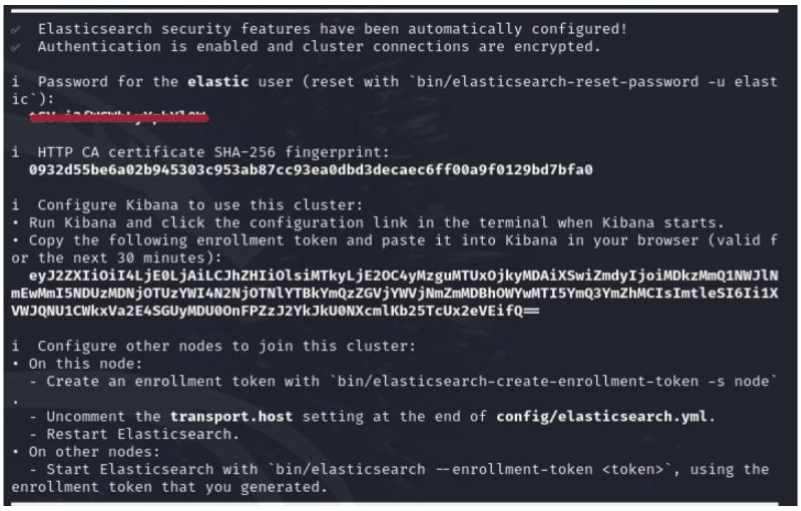
```
┌──(kali㉿kali)-[/var/log/logstash]
└─$ tail -n 1 /var/log/suricata/fast.log
11/23/2024-14:56:16.993178  [**] [1:1000002:1] ICMP Ping Test Alert [**] [Classification: (null)] [Prio
rity: 3] {ICMP} 192.168.238.151:8 → 8.8.8.8:0
```

Next was running Elasticsearch. The command below was how to start the program:



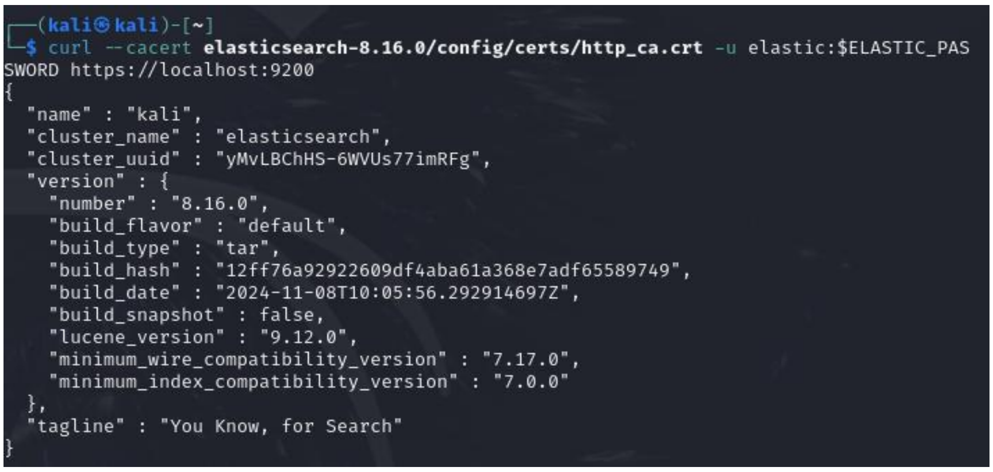
```
┌──(kali㉿kali)-[~/elasticsearch-8.16.0]
└─$ ./bin/elasticsearch
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
Picked up _JAVA_OPTIONS: -Dawt.useSystemAAFontSettings=on -Dswing.aatext=true
```

Furthermore, in its log, a password and token were displayed, which would be used for logging into Kibana.



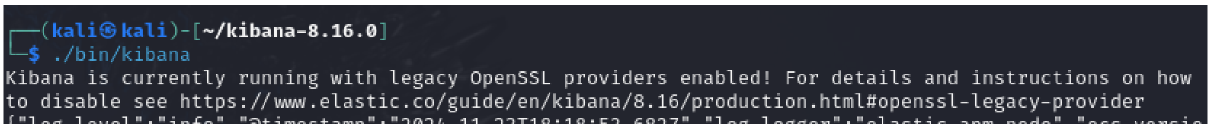
```
✓  Elasticsearch security features have been automatically configured!
✓  Authentication is enabled and cluster connections are encrypted.

i  Password for the elastic user (reset with `bin/elasticsearch-reset-password -u elast
ic`):
  [redacted]

i  HTTP CA certificate SHA-256 fingerprint:
  0932d55be6a02b945303c953ab87cc93ea0dbd3decaec6ff00a9f0129bd7bfa0

i  Configure Kibana to use this cluster:
• Run Kibana and click the configuration link in the terminal when Kibana starts.
• Copy the following enrollment token and paste it into Kibana in your browser (valid f
or the next 30 minutes):
  eyJ2ZXIiOiI4LjE0LjAiLCJhZHIiOlsiMTkyLjE2OC4yMzguMTUxOjkyMDAiXSwiZmdyIjoiMDkzMmQ1NWJlN
mEwMmI5NDUzMDNjOTUzYWI4N2NjOTNlYTBkYmQzZGVjYWVjNmZmMDBhOWYwMTI5YmQ3YmZhMCIsImtleSI6Ii1X
VWJQNU1CWkxVa2E4SGUyMDU0OnFPZzJ2YkJkU0NXcmlKb25TcUx2eVEifQ==

i  Configure other nodes to join this cluster:
• On this node:
  - Create an enrollment token with `bin/elasticsearch-create-enrollment-token -s node`
.
  - Uncomment the transport.host setting at the end of config/elasticsearch.yml.
  - Restart Elasticsearch.
• On other nodes:
  - Start Elasticsearch with `bin/elasticsearch --enrollment-token <token>`, using the
enrollment token that you generated.
```

To make sure Elasticsearch was configured and running properly, it was tested using the command below, and the result was a success:



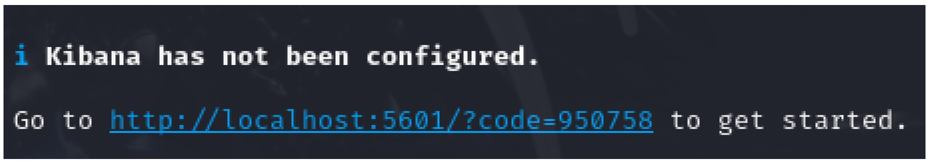
```
┌──(kali㉿kali)-[~]
└─$ curl --cacert elasticsearch-8.16.0/config/certs/http_ca.crt -u elastic:$ELASTIC_PAS
SWORD https://localhost:9200
{
  "name" : "kali",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "yMvLBChHS-6WVUs77imRFg",
  "version" : {
    "number" : "8.16.0",
    "build_flavor" : "default",
    "build_type" : "tar",
    "build_hash" : "12ff76a92922609df4aba61a368e7adf65589749",
    "build_date" : "2024-11-08T10:05:56.292914697Z",
    "build_snapshot" : false,
    "lucene_version" : "9.12.0",
    "minimum_wire_compatibility_version" : "7.17.0",
    "minimum_index_compatibility_version" : "7.0.0"
  },
  "tagline" : "You Know, for Search"
}
```

Since Elasticsearch was running, Kibana could now be ran.



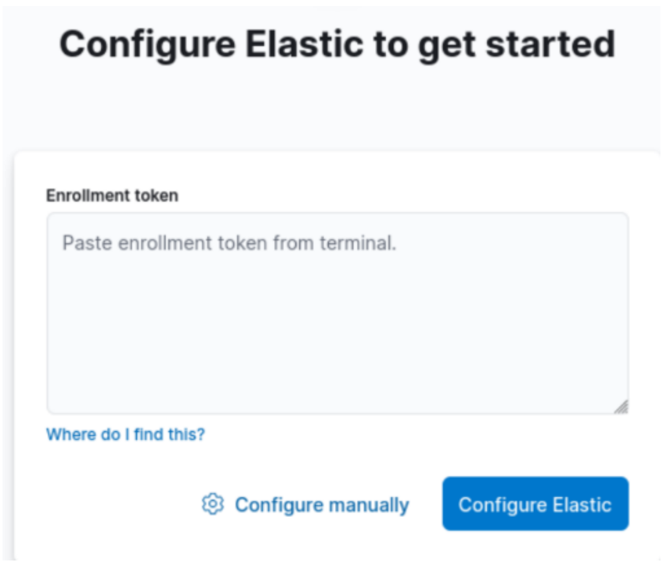
```
┌──(kali㉿kali)-[~/kibana-8.16.0]
└─$ ./bin/kibana
Kibana is currently running with legacy OpenSSL providers enabled! For details and instructions on how
to disable see https://www.elastic.co/guide/en/kibana/8.16/production.html#openssl-legacy-provider
```

When it ran, it displayed this information its log detailing that it needed to be configured (connecting to Elasticsearch).



```
i Kibana has not been configured.

Go to http://localhost:5601/?code=950758 to get started.
```

Clicking the link opened a page that asked for the enrolment token.

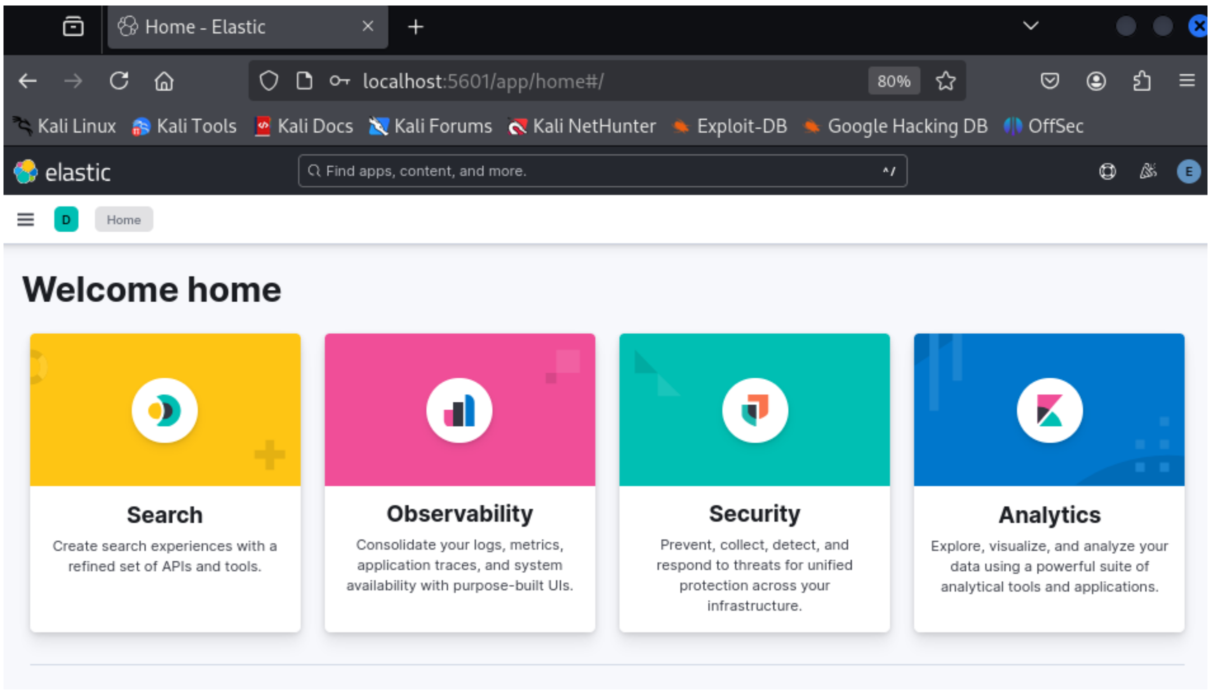

Configure Elastic to get started

Enrollment token

Paste enrollment token from terminal.

Where do I find this?

Once entered it does some configuration in the background to set up the Kibana dashboard, and it then redirects to Kibana's interface, where viewing logs and alerts can be done.



Welcome home

Search

Create search experiences with a refined set of APIs and tools.

Observability

Consolidate your logs, metrics, application traces, and system availability with purpose-built UIs.

Security

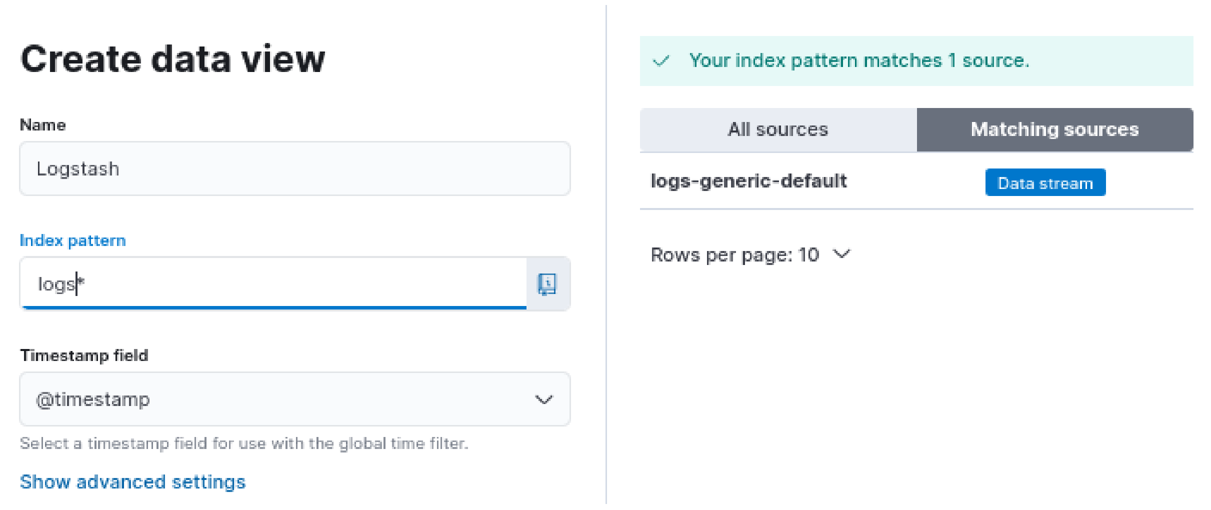
Prevent, collect, detect, and respond to threats for unified protection across your infrastructure.

Analytics

Explore, visualize, and analyze your data using a powerful suite of analytical tools and applications.

Viewing Alerts

In order to see alerts, a data view needed to be made because "Kibana requires a data view to access the Elasticsearch data that you want to explore" (Elastic, 2024). Furthermore, this will point towards the log data from Logstash, which will contain alerts.


Create data view

Name: Logstash

Index pattern: logs*

Timestamp field: @timestamp

Select a timestamp field for use with the global time filter.

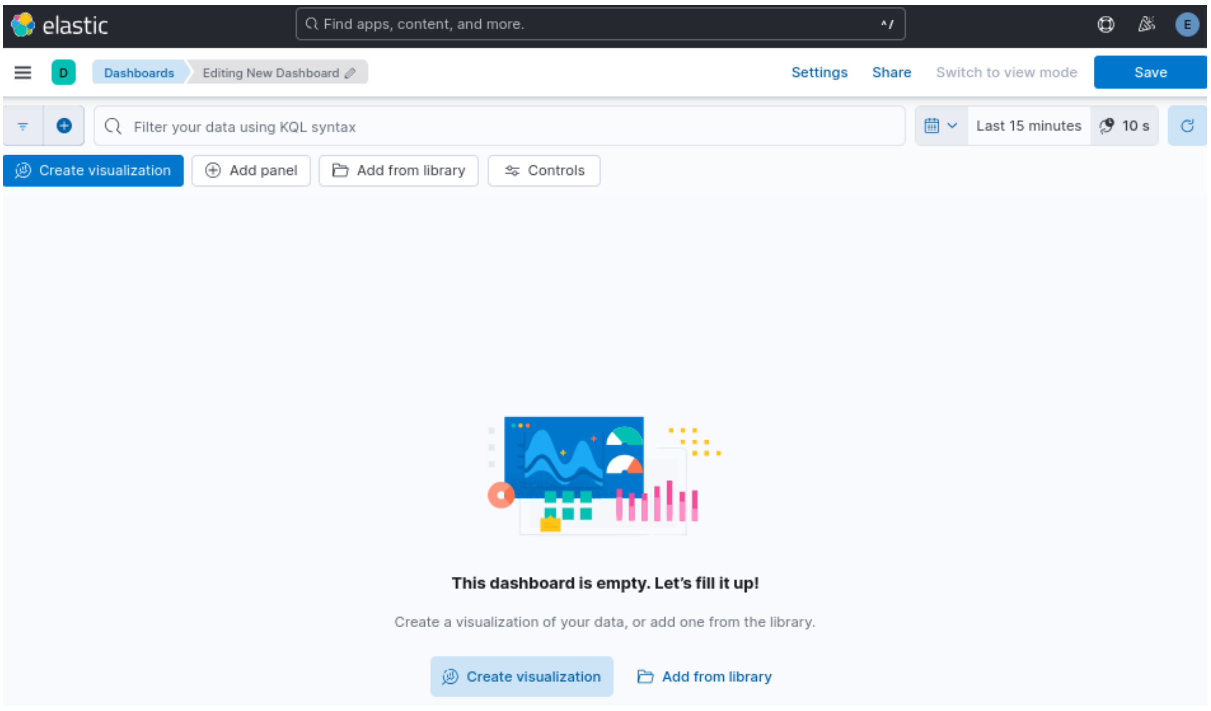
Show advanced settings

Your index pattern matches 1 source.

All sources	Matching sources
logs-generic-default	Data stream

Rows per page: 10

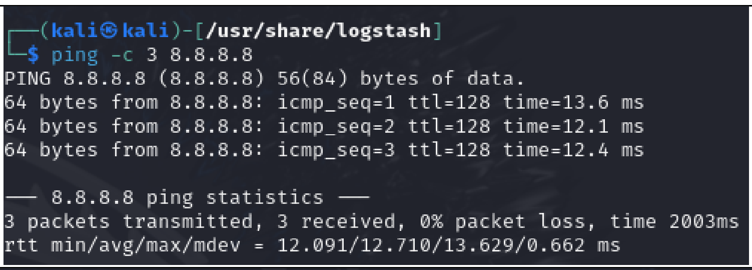
With the data view created, it meant that a dashboard could be created. It was possible to import pre-made visualisations, but the one from the article did not work. However, creating one was as easy as importing, as shown below.

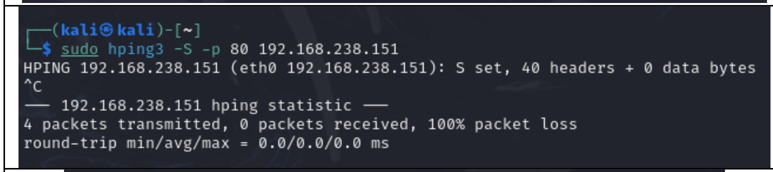


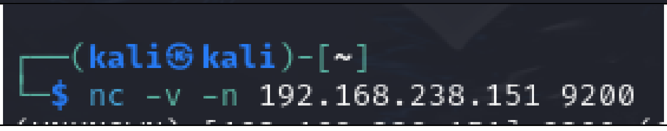
This dashboard is empty. Let's fill it up!

Create a visualization of your data, or add one from the library.

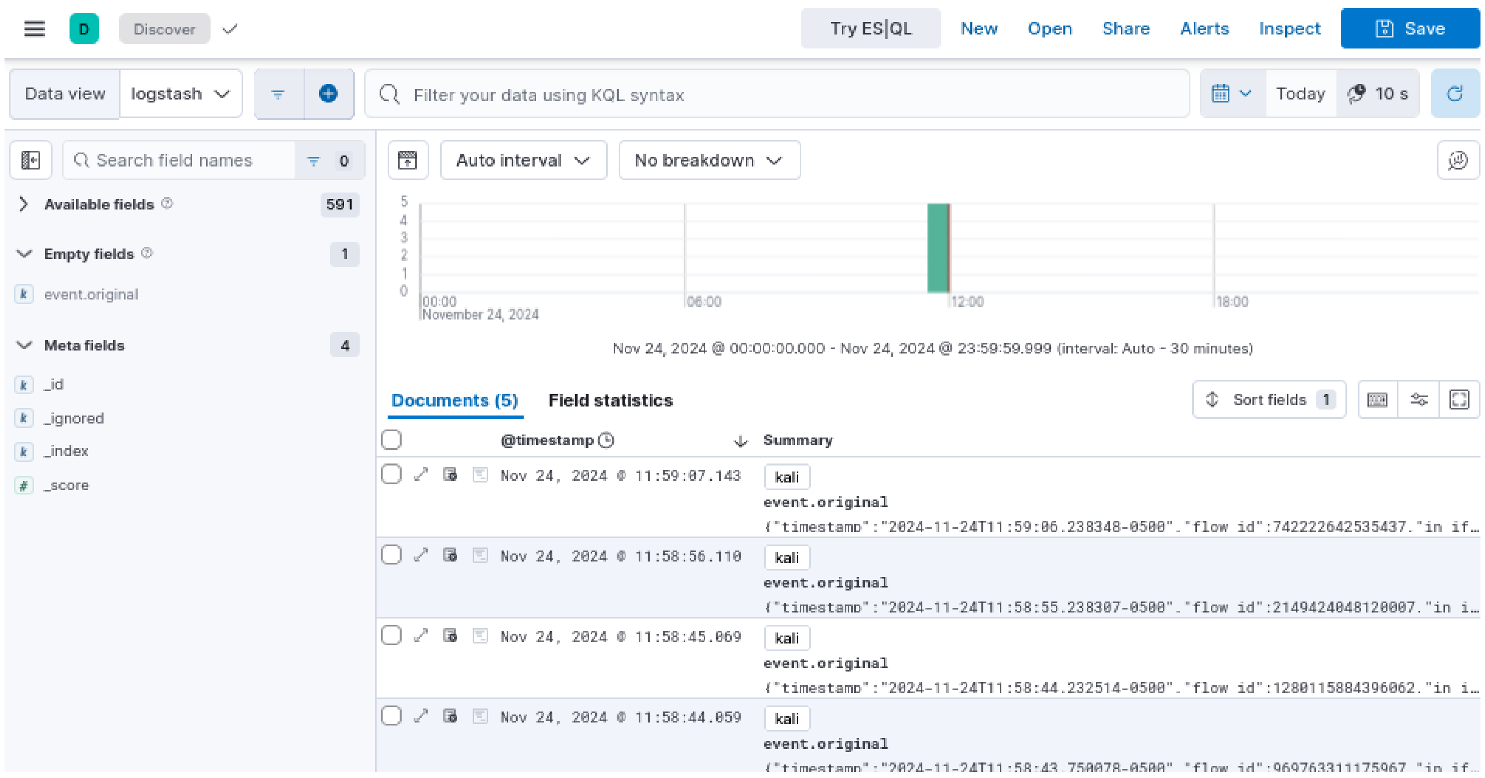
Below shows commands that were used to create alerts so that Kibana could see them. All these commands cause alerts to be made (Suricata captures them, sends them to Logstash, which sends them to Elasticsearch and then Kibana can display them):


`┌──(kali㉿kali)-[/usr/share/logstash]`
`└─$ ping -c 3 8.8.8.8`
`PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.`
`64 bytes from 8.8.8.8: icmp_seq=1 ttl=128 time=13.6 ms`
`64 bytes from 8.8.8.8: icmp_seq=2 ttl=128 time=12.1 ms`
`64 bytes from 8.8.8.8: icmp_seq=3 ttl=128 time=12.4 ms`
 	
`--- 8.8.8.8 ping statistics ---`
`3 packets transmitted, 3 received, 0% packet loss, time 2003ms`
`rtt min/avg/max/mdev = 12.091/12.710/13.629/0.662 ms`
	ICMP Ping Test


`┌──(kali㉿kali)-[~]`
`└─$ sudo hping3 -S -p 80 192.168.238.151`
`HPING 192.168.238.151 (eth0 192.168.238.151): S set, 40 headers + 0 data bytes`
`^C`
`--- 192.168.238.151 hping statistic ---`
`4 packets transmitted, 0 packets received, 100% packet loss`
`round-trip min/avg/max = 0.0/0.0/0.0 ms`
	Hping3 TCP SYN Scan


`┌──(kali㉿kali)-[~]`
`└─$ nc -v -n 192.168.238.151 9200`
	Netcat Banner Grabbing

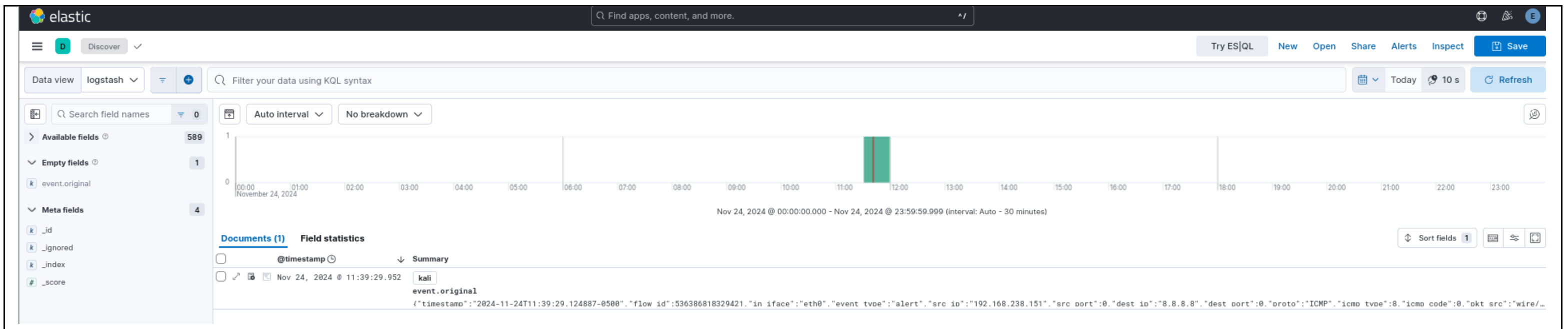
Furthermore, these alerts could be seen on Kibana's interface within the Discover section under Analytics, shown below:



Nov 24, 2024 @ 00:00:00.000 - Nov 24, 2024 @ 23:59:59.999 (interval: Auto - 30 minutes)

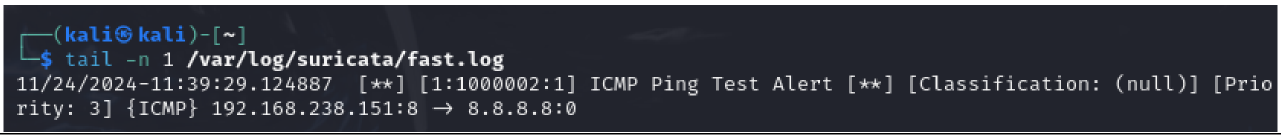
```
@timestamp                      Summary
Nov 24, 2024 @ 11:59:07.143     kali
                                event.original
                                {"timestamp":"2024-11-24T11:59:06.238348-0500","flow_id":742222642535437,"in_if...
Nov 24, 2024 @ 11:58:56.110     kali
                                event.original
                                {"timestamp":"2024-11-24T11:58:55.238307-0500","flow_id":2149424048120007,"in_i...
Nov 24, 2024 @ 11:58:45.069     kali
                                event.original
                                {"timestamp":"2024-11-24T11:58:44.232514-0500","flow_id":1280115884396062,"in_i...
Nov 24, 2024 @ 11:58:44.059     kali
                                event.original
                                {"timestamp":"2024-11-24T11:58:43.750078-0500","flow_id":969763311175967,"in_if...
```

To ensure that the alerts are the same ones from Suricata, the table below shows that this ICMP alert from Kibana has the same timestamp as the one from the logs in Suricata in the terminal, meaning that the stack connection works correctly.



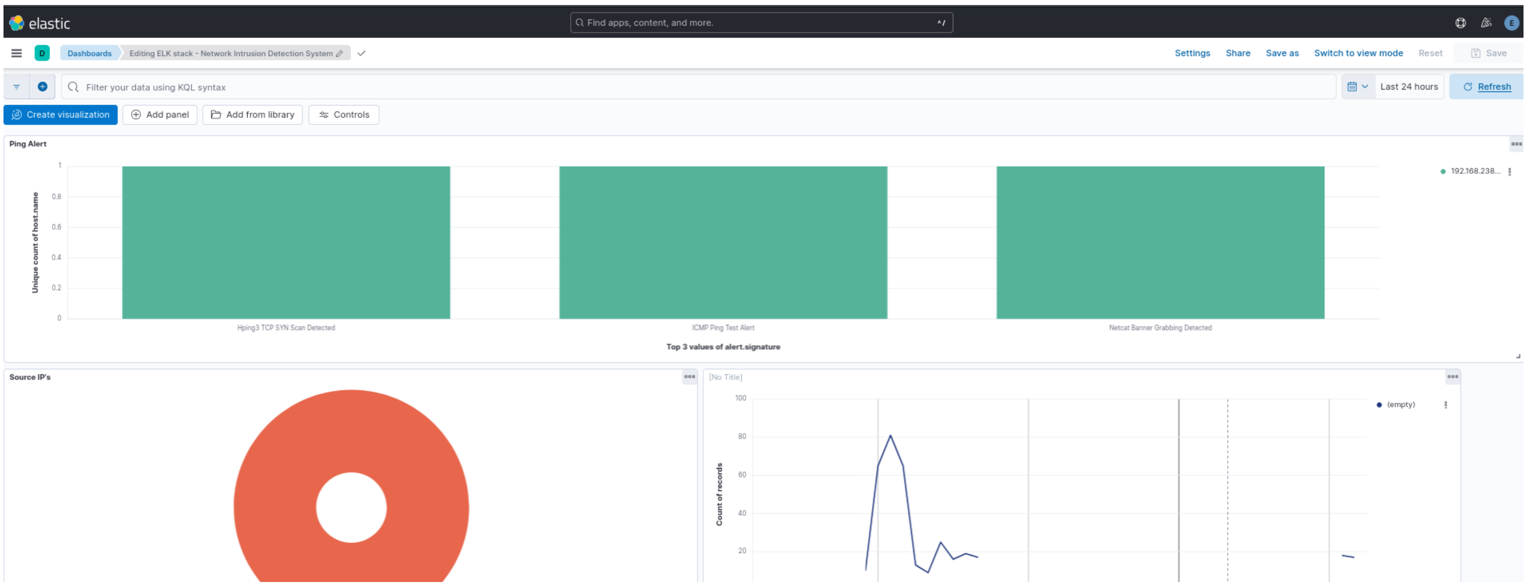
Nov 24, 2024 @ 00:00:00.000 - Nov 24, 2024 @ 23:59:59.999 (interval: Auto - 30 minutes)

```
@timestamp                      Summary
Nov 24, 2024 @ 11:39:29.952     kali
                                event.original
                                {"timestamp":"2024-11-24T11:39:29.124887-0500","flow_id":536386818329421,"in_iface":"eth0","event_type":"alert","src_ip":"192.168.238.151","src_port":0,"dest_ip":"8.8.8.8","dest_port":0,"proto":"ICMP","icmp_type":8,"icmp_code":0,"pkt_src":"wire/…
```

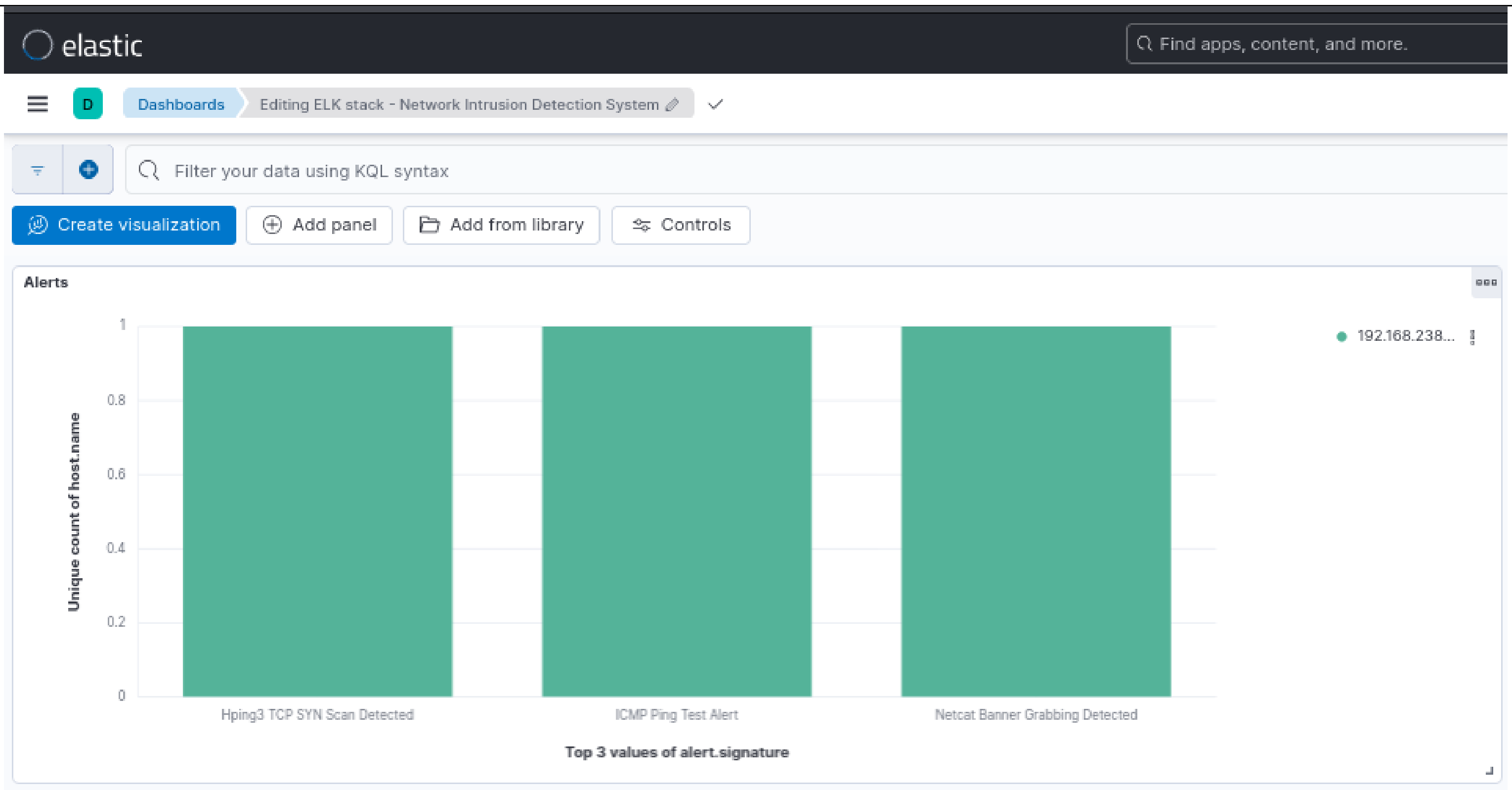


```
┌──(kali㉿kali)-[~]
└─$ tail -n 1 /var/log/suricata/fast.log
11/24/2024-11:39:29.124887  [**] [1:1000002:1] ICMP Ping Test Alert [**] [Classification: (null)] [Prio
rity: 3] {ICMP} 192.168.238.151:8 → 8.8.8.8:0
```

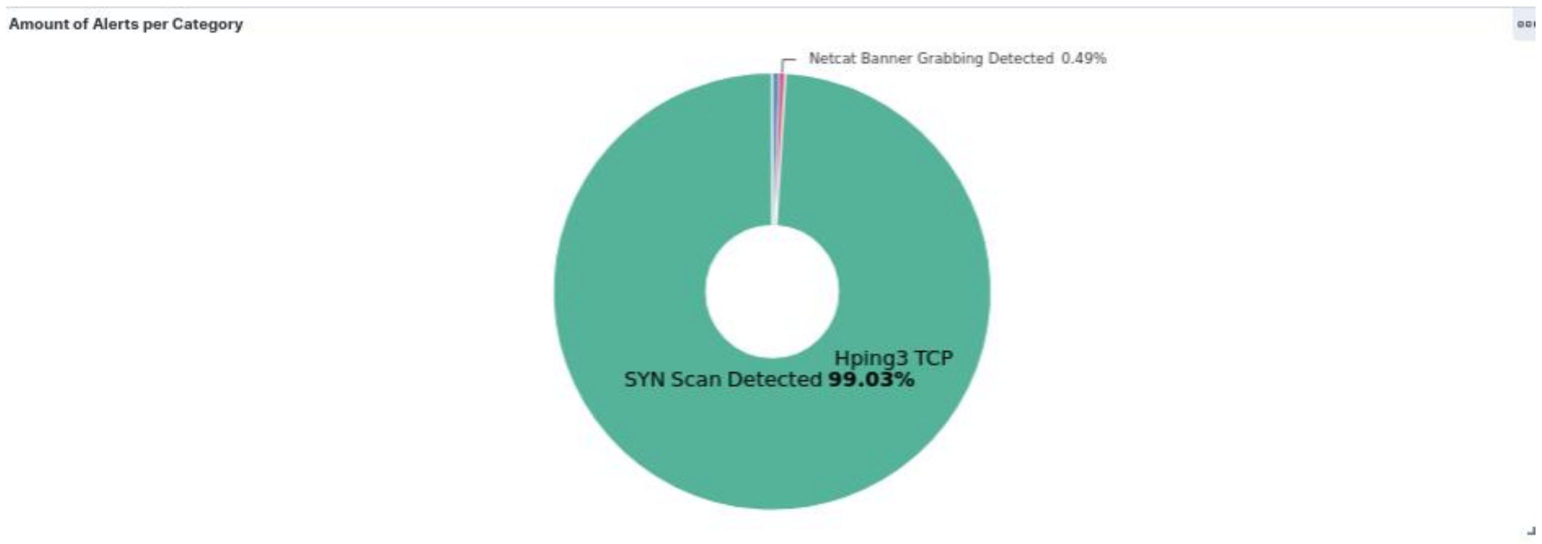
Now that the alerts were being sent, a dashboard could be created that would display them. Below shows the dashboard:



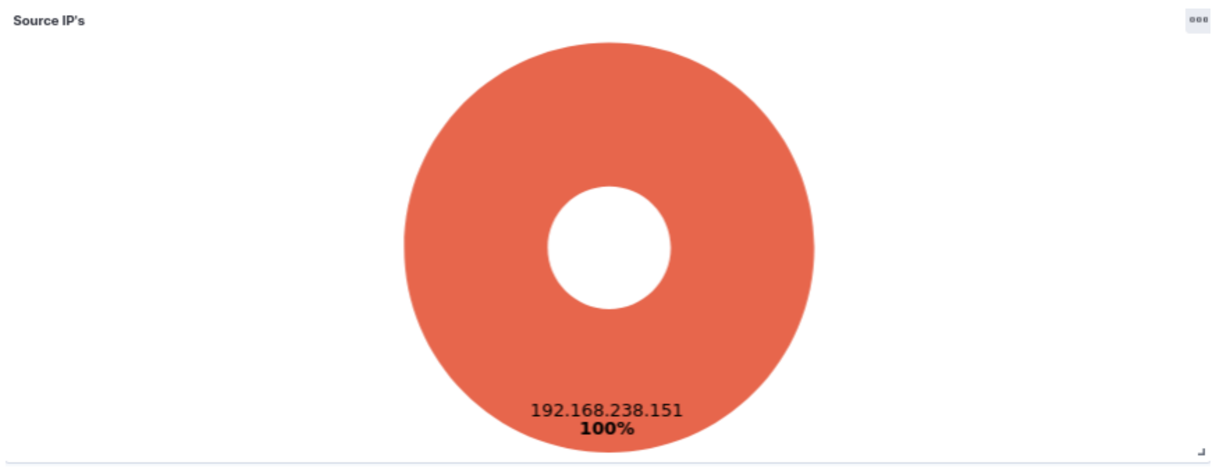
In depth view


elastic
Find apps, content, and more.
Dashboards
Editing ELK stack - Network Intrusion Detection System
Filter your data using KQL syntax
Create visualization
Add panel
Add from library
Controls
Alerts
Unique count of host.name
1
0.8
0.6
0.4
0.2
0
Hping3 TCP SYN Scan Detected
ICMP Ping Test Alert
Netcat Banner Grabbing Detected
Top 3 values of alert.signature
● 192.168.238...

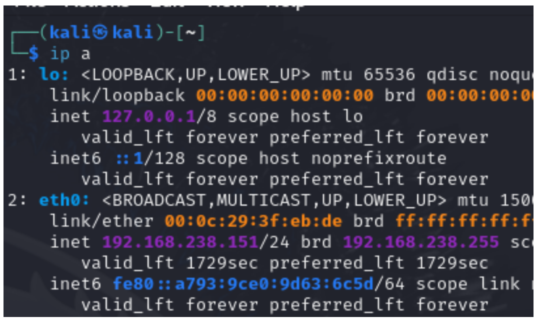
This bar chart shows the top 3 alerts of each hostname. As you can see, these are the alerts from the commands that were sent in the terminal. It clearly shows the stack connection works, and the network intrusion detection system is functioning.



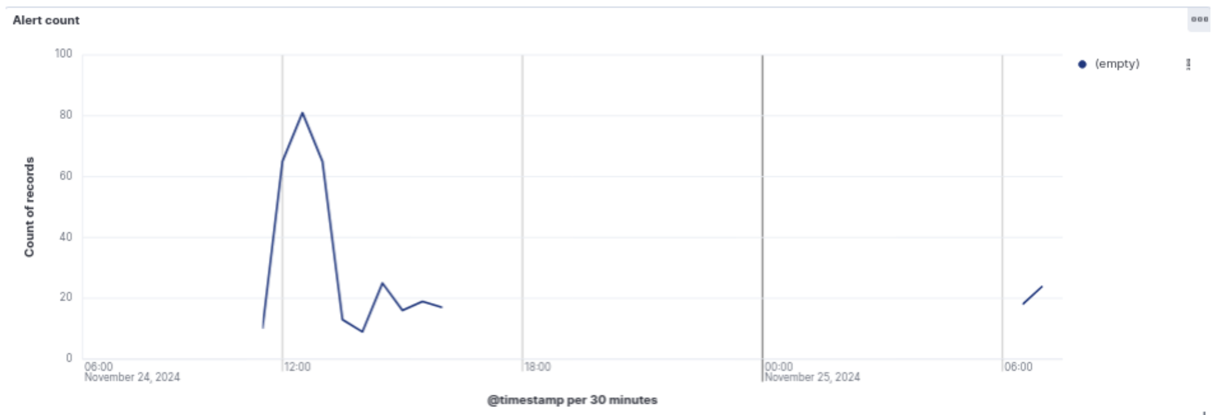
This pie chart shows the total percentage of alerts for each alert type.



This pie chart shows the source IP of the alerts, and the screenshot below confirms that it is showing the correct IP address:



```
┌──(kali㉿kali)-[~]
└─$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqu
    link/loopback 00:00:00:00:00:00 brd 00:00:00:0
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
       valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 150
    link/ether 00:0c:29:3f:eb:de brd ff:ff:ff:ff:f
    inet 192.168.238.151/24 brd 192.168.238.255 sc
       valid_lft 1729sec preferred_lft 1729sec
    inet6 fe80::a793:9ce0:9d63:6c5d/64 scope link
       valid_lft forever preferred_lft forever
```



This line graph shows the total alerts over a timespan per 30 minutes.

Experiment conclusion

As you can see from the process above, a successful Network Intrusion Detection System was created by following a guide and using unfamiliar technology. The report details findings and steps on the approach taken to make a functioning ELK stack that helps with detection systems. The next section covers an evaluation and a reflection.

Critical Evaluation

The experiment with the ELK stack for Network Intrusion Detection was a success. The system was able to detect rules that were set and produce alerts, for instance the network scans and the banner grabbing. These alerts were logged, and thanks to Kibana's interface, a clear visualisation was able to be seen. Of course, these alerts could have been easily seen on the terminal, but since Kibana provides a user-friendly interface, it allowed for better real-time monitoring and easier analysis. Furthermore, without Elasticsearch seeing these logs/alerts would not be possible as it provided all the data collected from Logstash. Despite the memory issues faced with Elasticsearch at the beginning of the experiment, the program was sufficient. Once fine-tuned, it was able to handle lots of data and it is obvious that the stack and the detection system are scalable. However, if Elasticsearch was to be used again, a better computer system with more memory space would be used to avoid any potential crashes and spending a lot of time on fine-tuning/configuration. In fact, a lot of system resources were used during this experiment, however since it was able to operate for a long period of time, it was acceptable that it used a lot of resources. It is important to note that rules were set from the emerging threats file. Furthermore, if this detection system was to be used for other people's systems, incorporating more rulesets would be needed to ensure better protection as not all threats were covered or detected by Suricata.

Altogether, despite some challenges, the stack was great to use. Furthermore, the main outcome, which was navigating unfamiliar technology and creating a network intrusion detection system, was achieved.

Reflection on Intrusion Detection System Development

The development of the intrusion detection system presented various difficulties and challenges, particularly due to the use of unfamiliar technology. Despite the obstacles, a successful intrusion detection system was developed.

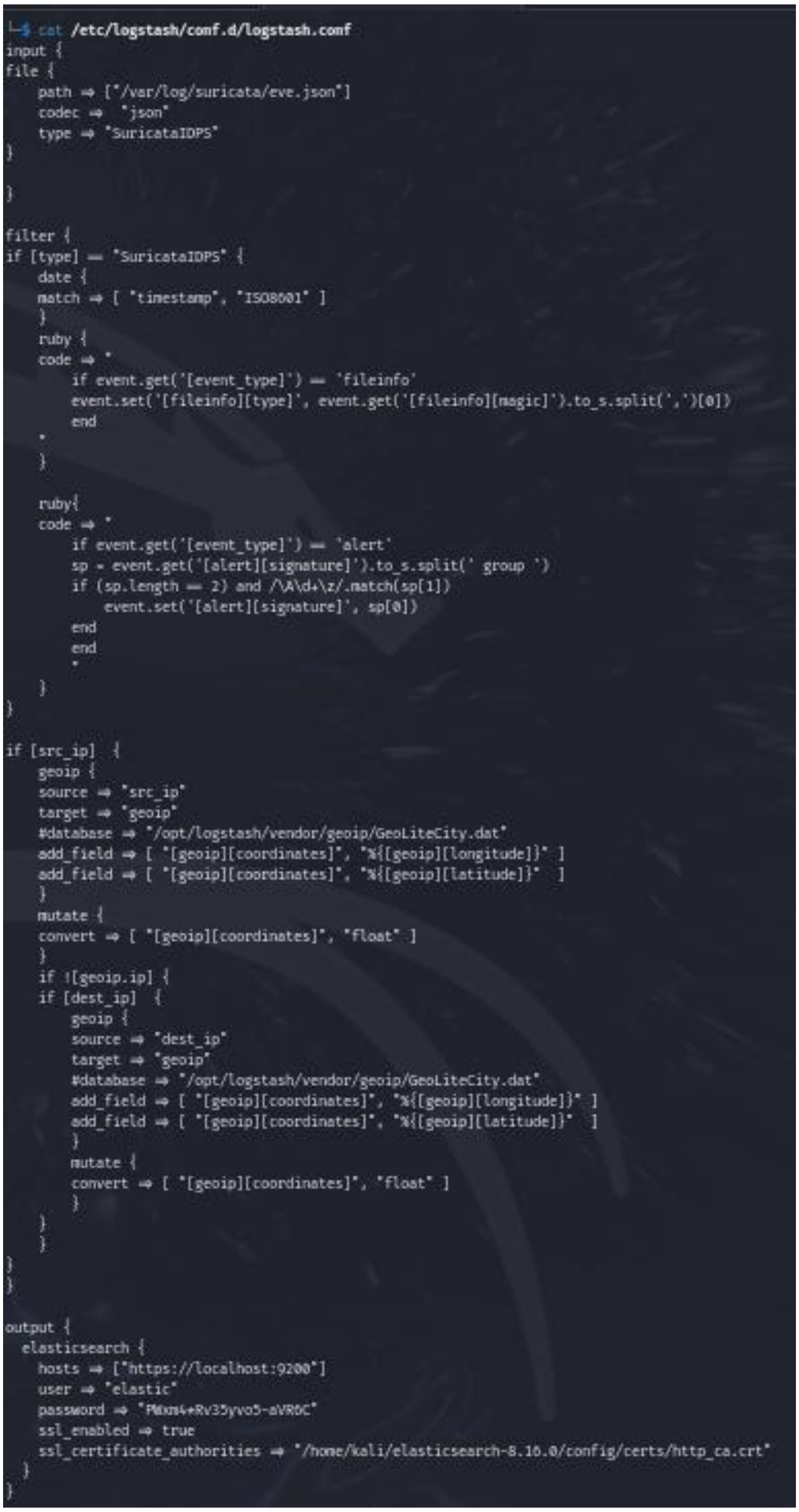
Initial stages focused on researching what an intrusion detection system was, as well as using an ELK stack for logging and examining the alerts. Understanding and outlining the requirements ensured that the intended outcome at the end of the development process was achieved.

Following the guide proved helpful, however it was limited due to some outdated links, so more research was required to find the correct versions of software to use. This was achieved and a combination of the guide and the other resources found allowed for an approach to be developed which led to the successful installation and configuration of all programs and files. This process enhanced organisational skills as it involved learning about the technology, gathering resources, and developing a structured method; these are key elements of effective project planning.

As stated, challenges occurred during the implementation stage as memory and compatibility issues arose. These challenges were addressed by finding articles and forums where similar problems had been talked about, and then identifying solutions based on their discussions. Fortunately, these issues were resolved, and the system was able to function. Furthermore, this developed problem-solving skills and the ability to troubleshoot complex technical challenges.

Appendix

Appendix A



```
└─$ cat /etc/logstash/conf.d/logstash.conf
input {
file {
    path => ["/var/log/suricata/eve.json"]
    codec =>  "json"
    type => "SuricataIDPS"
}

}

filter {
if [type] == "SuricataIDPS" {
    date {
    match => [ "timestamp", "ISO8601" ]
    }
    ruby {
    code => "
        if event.get('[event_type]') == 'fileinfo'
        event.set('[fileinfo][type]', event.get('[fileinfo][magic]').to_s.split(',')[0])
        end
    "
    }

    ruby{
    code => "
        if event.get('[event_type]') == 'alert'
        sp = event.get('[alert][signature]').to_s.split(' group ')
        if (sp.length == 2) and /\A\d+\z/.match(sp[1])
            event.set('[alert][signature]', sp[0])
        end
        end
        "
    }
}

if [src_ip]  {
    geoip {
    source => "src_ip"
    target => "geoip"
    #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
    add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
    add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
    }
    mutate {
    convert => [ "[geoip][coordinates]", "float" ]
    }
    if ![geoip.ip] {
    if [dest_ip]  {
        geoip {
        source => "dest_ip"
        target => "geoip"
        #database => "/opt/logstash/vendor/geoip/GeoLiteCity.dat"
        add_field => [ "[geoip][coordinates]", "%{[geoip][longitude]}" ]
        add_field => [ "[geoip][coordinates]", "%{[geoip][latitude]}"  ]
        }
        mutate {
        convert => [ "[geoip][coordinates]", "float" ]
        }
    }
    }
}
}

output {
  elasticsearch {
    hosts => ["https://localhost:9200"]
    user => "elastic"
    password => "PWxm4+Rv35yvo5-aVR6C"
    ssl_enabled => true
    ssl_certificate_authorities => "/home/kali/elasticsearch-8.16.0/config/certs/http_ca.crt"
  }
}
```

References

damendo. (2024, October 16). *Perform network intrusion detection with open source tools - Azure Network Watcher*. Learn.microsoft.com. https://learn.microsoft.com/en-us/azure/network-watcher/network-watcher-intrusion-detection-open-source-tools

Elastic. (2024). Kibana—your window into Elastic | Kibana Guide [8.16] | Elastic. *Elastic.co*. https://doi.org/8.16

Elasticsearch. (2024). *Install Elasticsearch from archive on Linux or MacOS | Elasticsearch Guide [8.11] | Elastic*. Www.elastic.co. https://www.elastic.co/guide/en/elasticsearch/reference/current/targz.html

Getting started with the Elastic Stack | Getting Started [7.4] | Elastic. (2018). *Elastic.co*. https://doi.org/7.4

Kibana—your window into Elastic | Kibana Guide [8.16] | Elastic. (2024). *Elastic.co*. https://doi.org/8.16