Host-Based Intrusion Detection System Using the ELK Stack and Suricata

Cybersecurity Project

PREPARED BY

ATHUL KRISHNA A U

CONTENTS

INTRODUCTION **1**

METHODOLOGY **3**

INSTALLATION **4**

CONFIGURATION **6**

ELASTICSEARCH **6**
KIBANA **8**
LOGSTASH **13**
SURICATA **16**
FILEBEAT **18**

DASHBOARD SETUP **22**

CONCLUSION **24**

Introduction

In this project, I am going to make a **HIDS** by configuring Suricata, VirusTotal , AlienVault OTX and ELK Stack together to analyse real-time logs. So let's familiarize with it.

1) **What is HIDS?**

   It is a cybersecurity tool that monitors and analyses traffic to host for suspicious activities, policy violations, or potential attacks in real-time.

   If malicious activity is detected, the HIDS generates alerts for security analysts to investigate.

2) **What is ELK Stack?**

   The ELK Stack is a powerful open-source log management and data analysis platform used for collecting, storing, analysing and visualizing logs or data from multiple sources.

   ELK stands for:

   - **Elasticsearch:** A search and analytic engine that stores and indexes data for fast searching.
   - **Logstash:** A data processing pipeline that collects, filters, and sends data to Elasticsearch.
   - **Kibana:** A visualization tool that displays data stored in Elasticsearch using dashboards and graphs.

3) **What is VirusTotal API?**

   The VirusTotal API is a RESTful API provided by VirusTotal that allows tools and systems like Elastic Stack or Suricata-based HIDS setup to automatically query VirusTotal's database for threat intelligence information

4) **What is AlienVault OTX?**

   AlienVault OTX (Open Threat Exchange) is a threat intelligence sharing platform by AT&T Cybersecurity. It provides Indicators of Compromise (IOCs) — such as:

   - Malicious Ips
   - Domains
   - URLs
   - File hashes
   - Threat actor information.

   You can access this data through the OTX API using your API key.

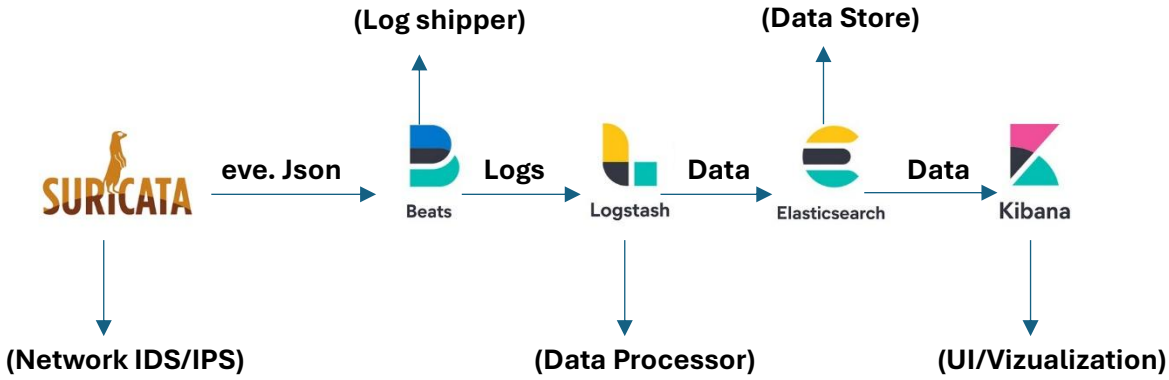
5) **What is Suricata?**

Suricata is an open-source Intrusion Detection system (IDS), Intrusion Prevention System (IPS) and Network Security Monitoring (NSM) tool developed by Open Information Security Foundation (OISF).

It monitors network traffic in real-time and analyses packets to detect malicious activities, attacks, or policy violations.

Suricata works by capturing packets from a network interface and inspect them using **rule-based signatures.** It then identifies attacks or anomalies and logs them for analysis.

Suricata generates the **eve.json** file through its **EVE (Extensible Event Format) logging module,** which is designed to output structured, machine-readable logs for use in analysis platforms like **ELK Stack**, **Splunk** or other SIEM tools.



- Suricata writes EVE JSON logs to disk.
- Filebeat monitors these logs, parses them using the Suricata module, and sends them to Logstash (or directly to Elasticsearch).
- Logstash (if used) processes and enriches the logs.
- Elasticsearch stores and indexes the processed logs.
- Kibana queries Elasticsearch to visualize and analyse the Suricata event data.

Methodology

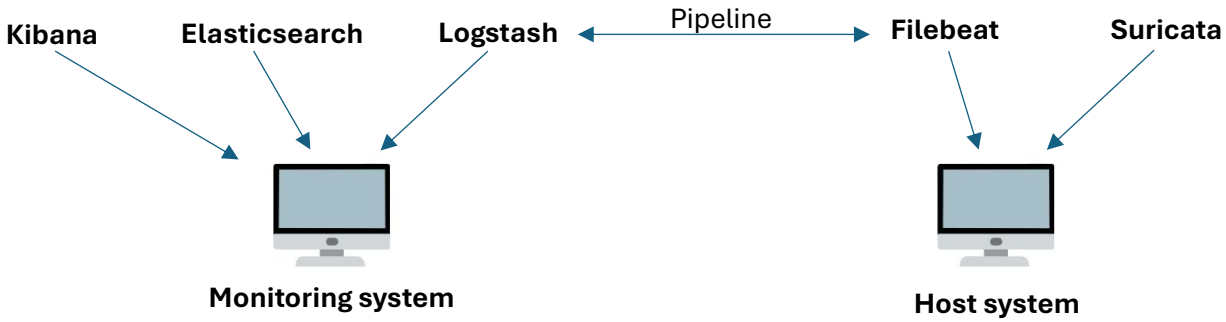
1) Tools required:

- Elasticsearch
- Logstash
- Kibana
- Filebeat
- Suricata
- VirusTotal
- AlienVault OTX

2) OS used:

Here I am using **Debian based OS** (**Kali Linux**). Any other OS such as Windows, MacOS, etc can also be used. Many other Linux-based operating systems, including Ubuntu, Mint, and Kali Linux, are based on Debian.

Both host and monitoring system are set on kali linux.



Installation

1) **Elasticsearch**

Download Elasticsearch package from **elastic.co**

https://www.elastic.co/downloads/elasticsearch

Here I am using kali linux so I am going with **deb x86_64** package.

Install Elasticsearch:

dpkg -i package_name.deb

for eg. **dpkg -i elasticsearch-9.1.5-amd64.deb**

Start the Elasticsearch service:

sudo systemctl enable elasticsearch

sudo systemctl start elasticsearch

To check the elasticsearch service status:

sudo systemctl status elasticsearch

2) **Logstash**

Download Logstash.

https://www.elastic.co/downloads/logstash

Install logstash:

dpkg -i package_name.deb

Start Logstash service:

sudo systemctl enable logstash

sudo systemctl start logstash

sudo systemctl status logstash

3) **Kibana**

Download Kibana package:

https://www.elastic.co/downloads/kibana

Install Kibana:

dpkg -i package_name.deb

Start Kibana service:

sudo systemctl enable kibana

sudo systemctl start kibana

sudo systemctl status kibana

Next download and install Filebeat and Suricata in host system.

4) Filebeat

Download Filebeat package:

https://www.elastic.co/downloads/beats/filebeat

Install Filebeat:

dpkg -i package_name.deb

Start Filebeat service:

sudo systemctl enable filebeat

sudo systemctl start filebeat

sudo systemctl status filebeat

5) Suricata

Add the OISF Repository:

sudo add-apt-repository ppa:oisf/suricata-stable

Update Package Lists:

sudo apt update

Install Suricata:

sudo apt install suricata

Start and Enable Suricata

sudo systemctl enable suricata

sudo systemctl start suricata

sudo systemctl status suricata

Next step is to configure these tools together by editing their yml/yaml, conf, rules file.

Note:

Don’t forget to restart the service after editing their file if the services are running.

For eg. if you edited the elasticsearch.yml file then restart the service by

sudo systemctl restart elasticsearch

Configuration

1) Configure Elasticsearch

Go to **/etc/elasticsearch** directory and edit the elasticsearch.yml

```
sudo nano elasticsearch.yml
```

Uncomment required lines

Network.host: “0.0.0.0” #using 0.0.0.0 exposes the service to all network connections.

http.port: 9200

transport.host: “0.0.0.0”

```
# ---------------------------------- 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
#
# ---------------------------------- Memory -----------------------------------
#
# Lock the memory on startup:
#
#bootstrap.memory_lock: true
#
# Make sure that the heap size is set to about half the memory available
# on the system and that the owner of the process is allowed to use this
# limit.
#
# Elasticsearch performs poorly when the system is swapping the memory.
#
# ---------------------------------- 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 ----------------------------------
```

```
# 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: ["RA-7"]

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

# Allow other nodes to join the cluster from anywhere
# Connections are encrypted and mutually authenticated
transport.host: "0.0.0.0"

#----------------------- END SECURITY AUTO CONFIGURATION -------------------------
```

Save and exit:

Ctrl+x

y, and enter

Enable and start the elasticsearch service, if already running then restart it.

```
sudo systemctl restart elasticsearch
```

If you want to restset the elastic password, then run this command

```
sudo /usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic
```

If you get a permission error then run these commands

```
sudo chown -R elasticsearch:elasticsearch /etc/elasticsearch
sudo chown -R 750 /etc/elasticsearch
```

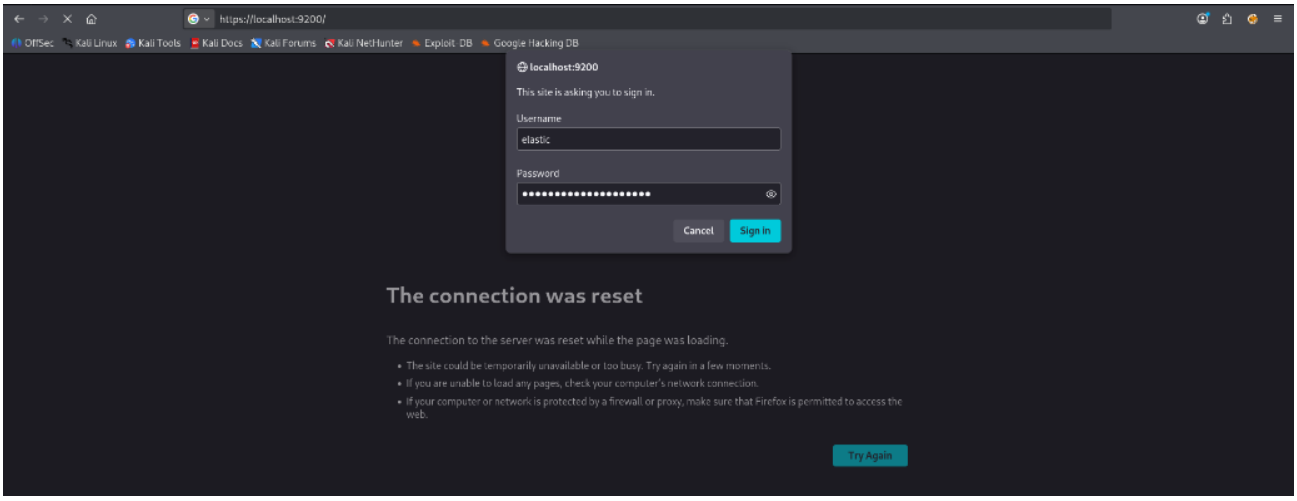
Then retry:

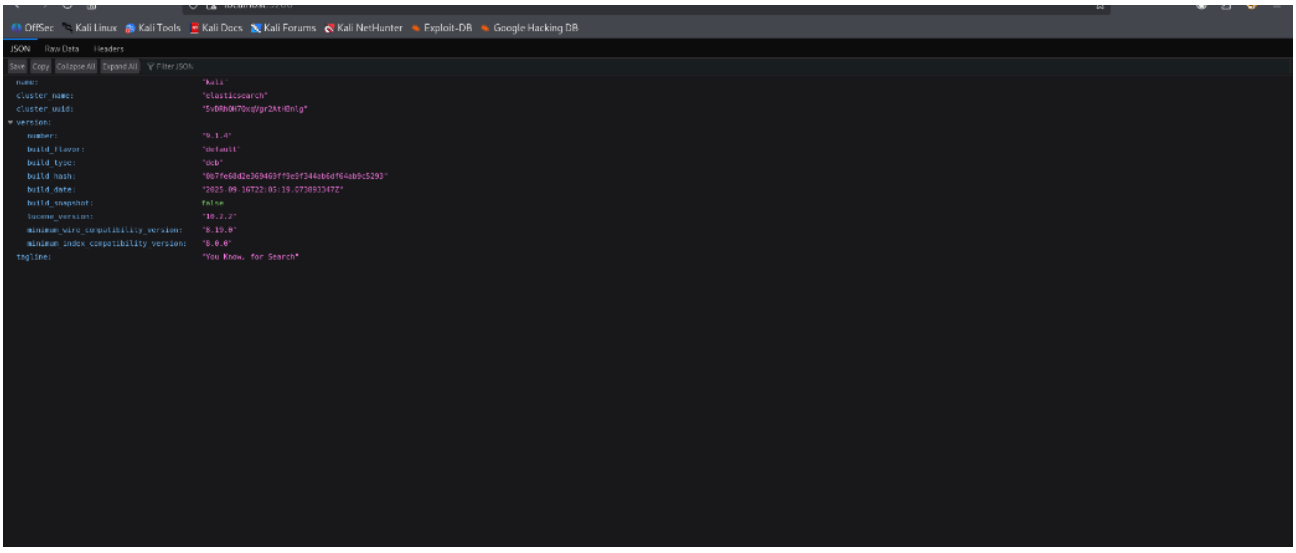
```
sudo /usr/share/elasticsearch/bin/elasticsearch-reset-password -u elastic
```

Note it somewhere, don't forget the password.

Check the elasticsearch is working or not:

Go to your browser and https://localhost:9200





Your Elasticsearch instance is running perfectely if you got a result like this.

2) Configure Kibana

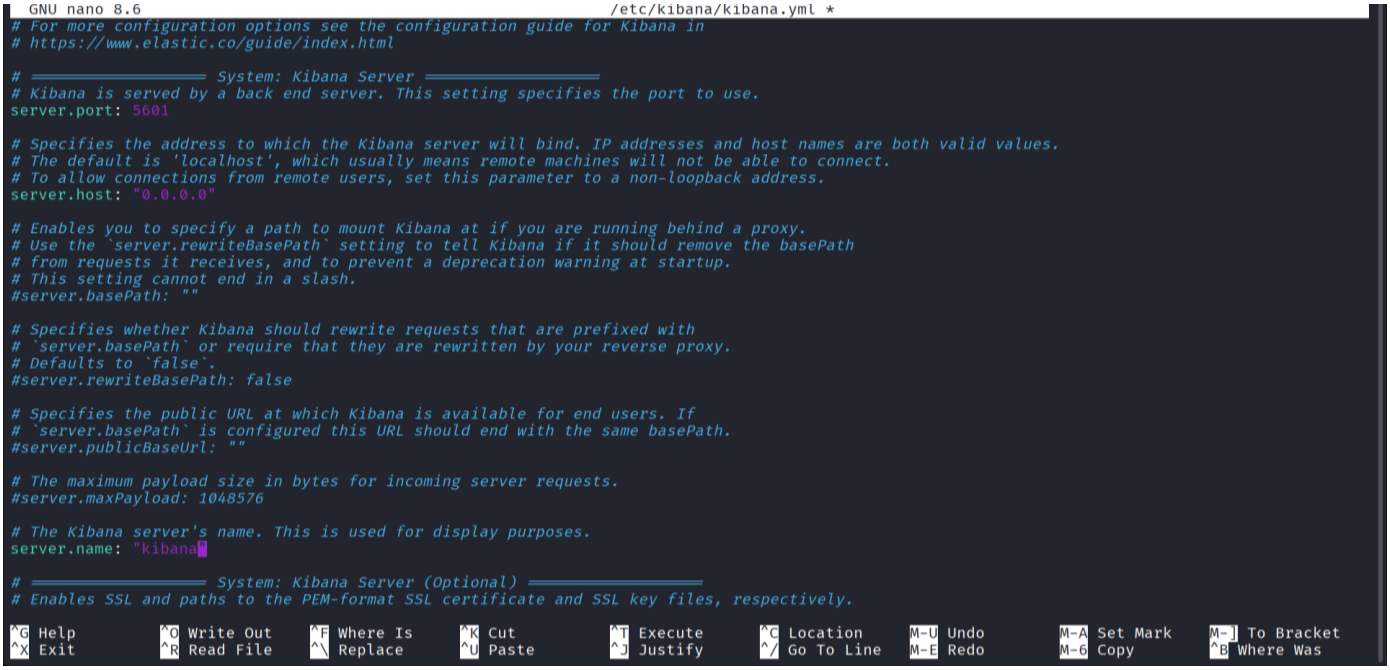
Go to **/etc/kibana** directory and run this command

sudo nano kibana.yml

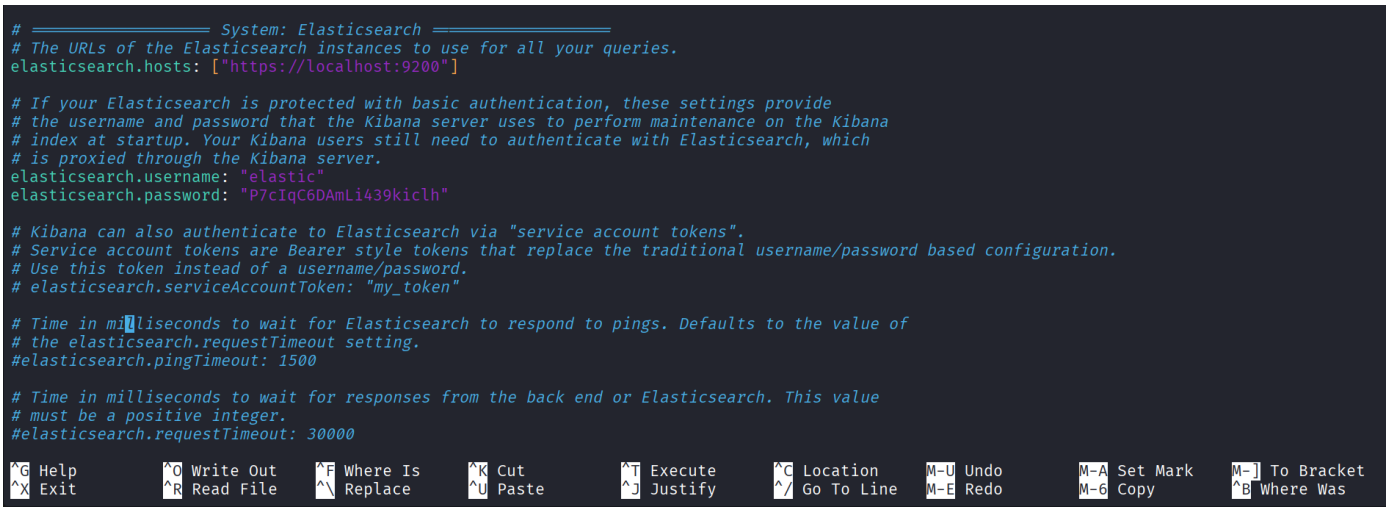
Uncomment these lines in **System: Kibana server** section

server.port: 5061

server.host: “0.0.0.0”



In **System: Elasticsearch** Section, **give elasticsearch.hosts,** elasticsearch’s username and password.



Or use **Elasticsearch.ServiceAccountToken** instead of username and password.

Generate **Elasticsearch.ServiceAccountToken** using this command

sudo /usr/share/elasticsearch/bin/elasticsearch-service-tokens create elastic/kibana kibana-token

```
 GNU nano 8.6                                                /etc/kibana/kibana.yml *

# The maximum payload size in bytes for incoming server requests.
#server.maxPayload: 1048576

# The Kibana server's name. This is used for display purposes.
# server.name: "kibana"

# ================== System: Kibana Server (Optional) ==================
# Enables SSL and paths to the PEM-format SSL certificate and SSL key files, respectively.
# These settings enable SSL for outgoing requests from the Kibana server to the browser.
#server.ssl.enabled: false
#server.ssl.certificate: /path/to/your/server.crt
#server.ssl.key: /path/to/your/server.key

# ================== System: Elasticsearch ==================
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["https://localhost:9200"]

# If your Elasticsearch is protected with basic authentication, these settings provide
# the username and password that the Kibana server uses to perform maintenance on the Kibana
# index at startup. Your Kibana users still need to authenticate with Elasticsearch, which
# is proxied through the Kibana server.
#elasticsearch.username: "elastic"
#elasticsearch.password: "P7cIqC6DAmLi439kiclh"

# Kibana can also authenticate to Elasticsearch via "service account tokens".
# Service account tokens are Bearer style tokens that replace the traditional username/password based configuration.
# Use this token instead of a username/password.
elasticsearch.serviceAccountToken: "AAEAAWVsYXN0aWMva2liYW5hL2tpYmFuYS10b2tlbjpSZmI1LVh6ZVNiYUhCRExnV2dHLTFn"

# Time in milliseconds to wait for Elasticsearch to respond to pings. Defaults to the value of
# the elasticsearch.requestTimeout setting.
#elasticsearch.pingTimeout: 1500

# Time in milliseconds to wait for responses from the back end or Elasticsearch. This value
Save modified buffer?
 Y Yes
 N No           ^C Cancel
```

Add Kibana's encryption keys

Run this command to geneate encryption keys

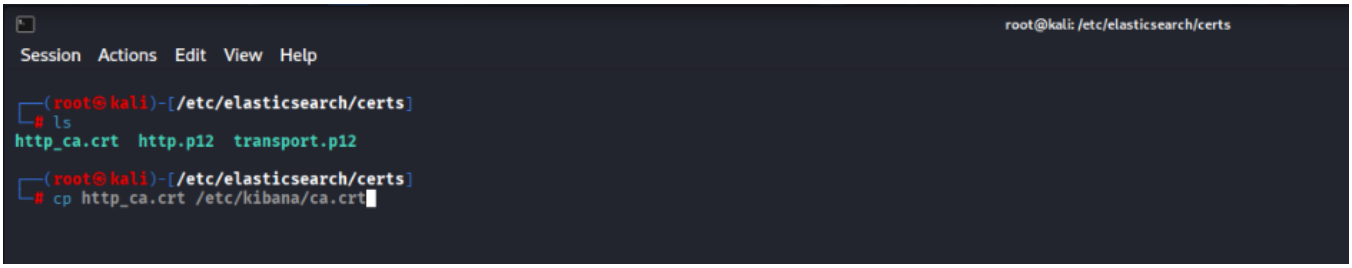
sudo /usr/share/kibana/bin/kibana-encryption-keys generate

```
└─$ sudo /usr/share/kibana/bin/kibana-encryption-keys generate
## Kibana Encryption Key Generation Utility

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

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

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

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


Already defined settings are ignored and can be regenerated using the --force flag.  Check the documentation links for instructions on how to rotate encryption keys.
Definitions should be set in the kibana.yml used configure Kibana.

Settings:
xpack.encryptedSavedObjects.encryptionKey: 5de5fc29c0bc3fa876c9b0147fc0ea65
xpack.reporting.encryptionKey: 1301548fcfffef5e0e93520feb97911e
xpack.security.encryptionKey: 4d93cf0e7c255e2dbb88845c937e78b0
```

```
 GNU nano 8.6                                                /etc/kibana/kibana.yml *
# The path where Kibana stores persistent data not saved in Elasticsearch. Defaults to data
#path.data: data

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

# Set the interval in milliseconds to sample system and process performance
# metrics. Minimum is 100ms. Defaults to 5000ms.
#ops.interval: 5000

# Specifies locale to be used for all localizable strings, dates and number formats.
# Supported languages are the following: English (default) "en", Chinese "zh-CN", Japanese "ja-JP", French "fr-FR".
#i18n.locale: "en"
xpack.encryptedSavedObjects.encryptionKey: 5de5fc29c0bc3fa876c9b0147fc0ea65
xpack.reporting.encryptionKey: 1301548fcfffef5e0e93520feb97911e
xpack.security.encryptionKey: 4d93cf0e7c255e2dbb88845c937e78b0

# ================== Frequently used (Optional)==================

# ================== Saved Objects: Migrations ==================
# Saved object migrations run at startup. If you run into migration-related issues, you might need to adjust these settings.
```

The elasticsearch uses **HTTPS,** So add **Elasticsearchs' CA certificate** in kibana.yml

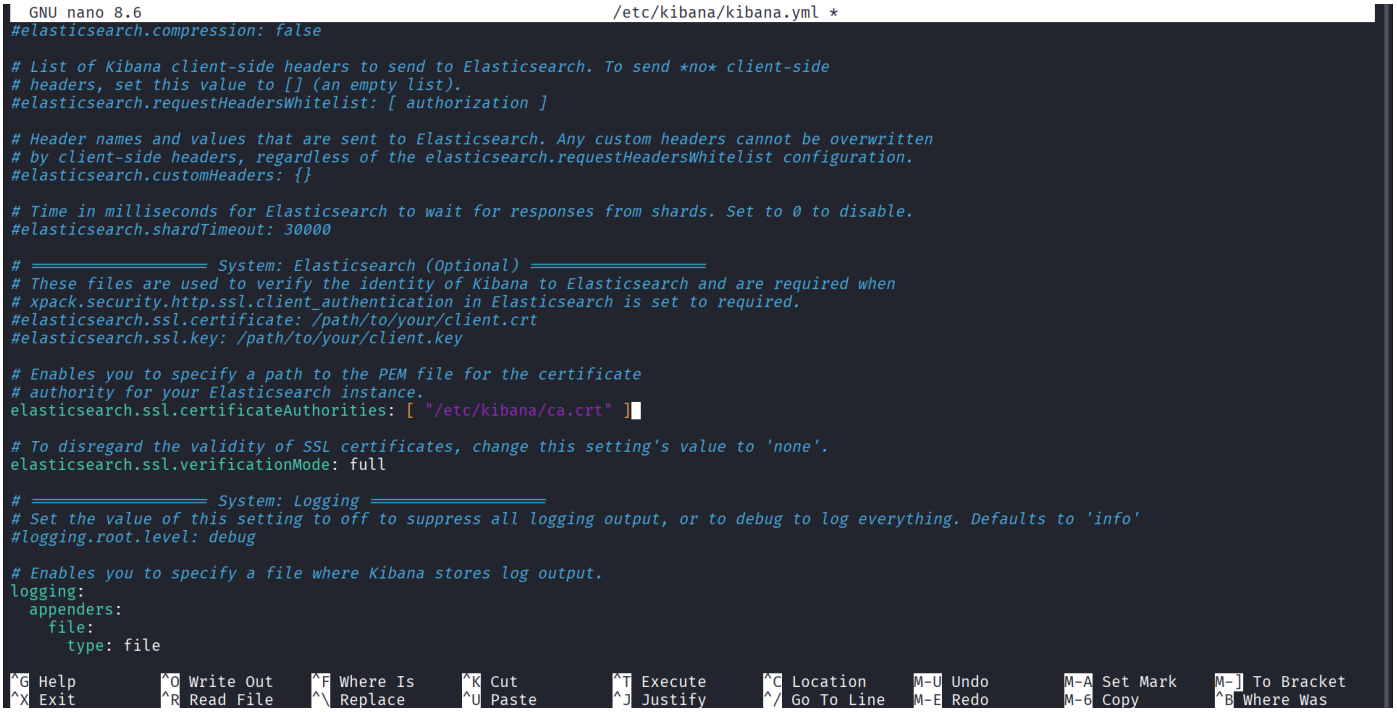
Go to **/etc/elasticsearch/certs** , From this folder copy the **http_ca.crt** file to kibana's folder or just specify it's path in kibana.yml. Here I'm copied and renamed it into **ca.crt**.



Add CA certificate to kibana.yml

elasticsearch.ssl.certificateAuthoroties: ["path/to/your/ca.crt"]

elasticsearch.ssl.verificationMode: full



Save and exit.

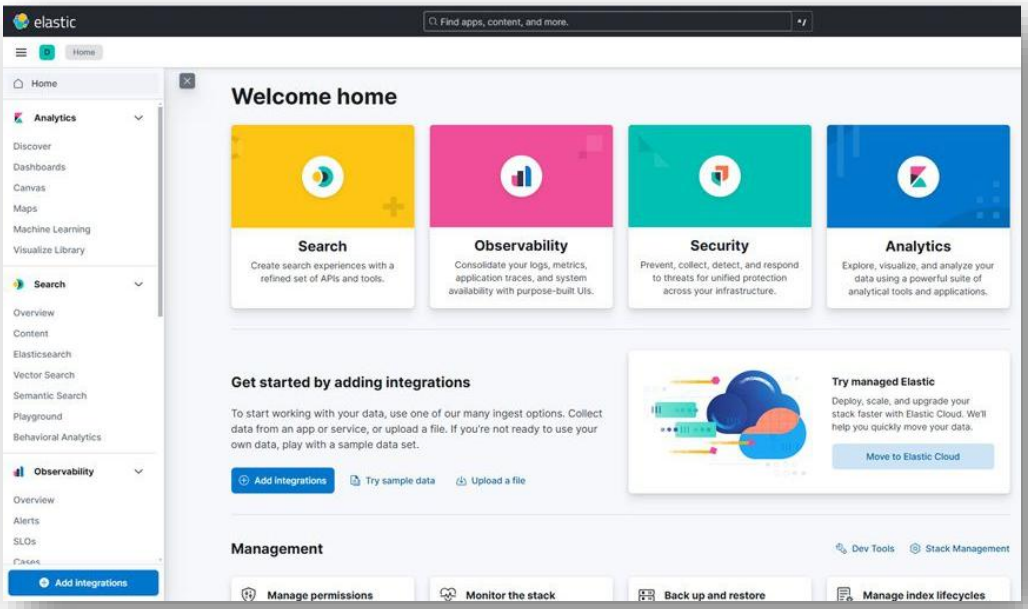
Why using CA certificate?

When Kibana connects to Elasticsearch over HTTPS, it must verify that the Elasticsearch server is legitimate, not an impostor or attacker pretending to be Elasticsearch.

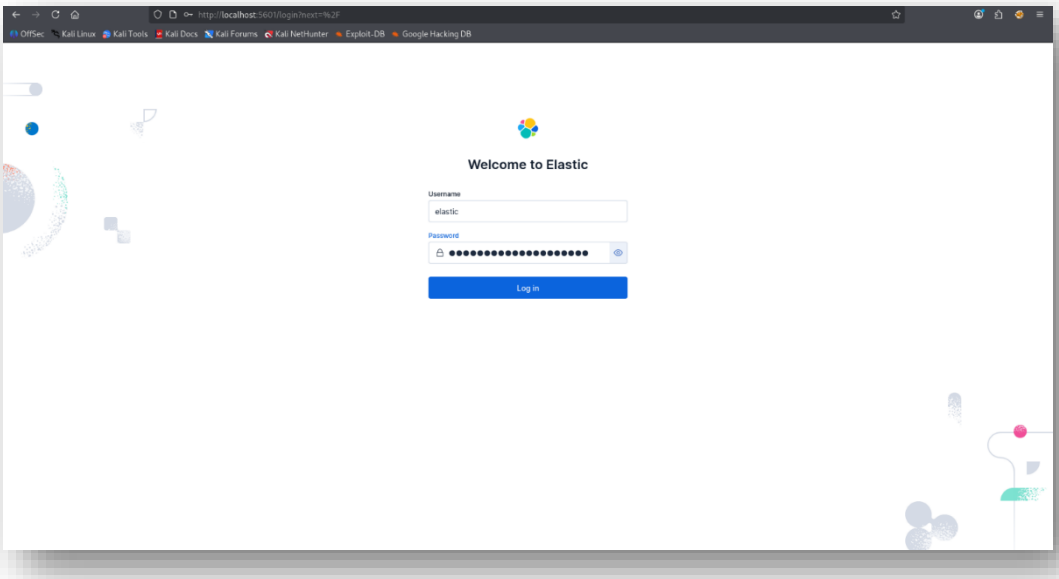
Now check the status:

Then open your browser and go to:

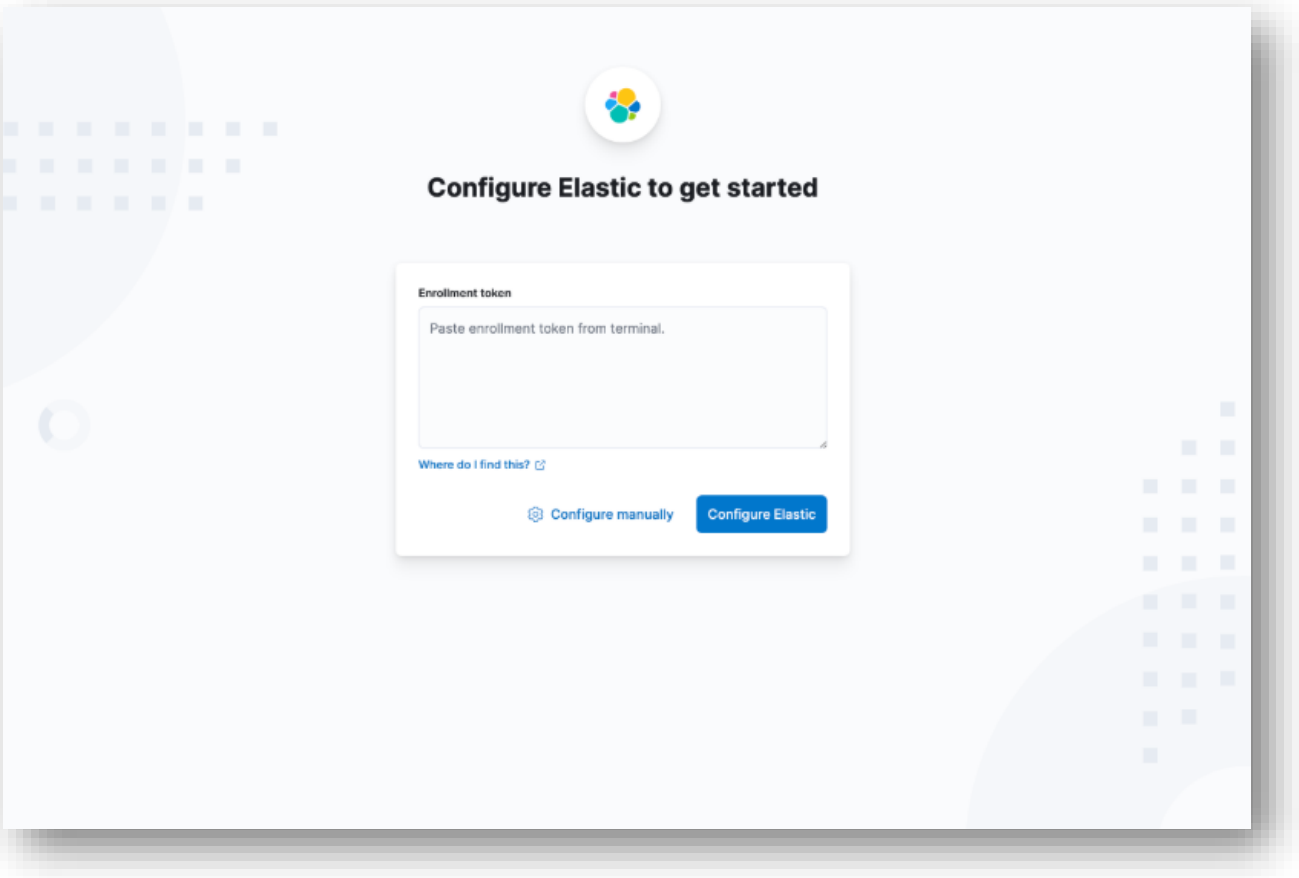
http://localhost:5061



Login with your elasticsearch's username and password.



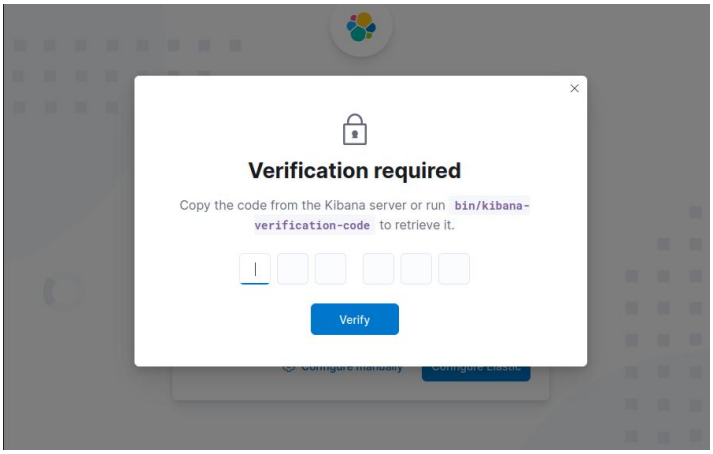
If Kibana asks for enrollment token:



Run this command to generate enrollment token:

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

after this kibana will ask for a verification code.



Get the verification code by this command:

sudo /usr/share/kibana/bin/kibana-verification-code

3) Configure Logstash

First make a pipeline configuration, This configuration file is the core instruction manual that tells Logstash which data to get (Input), how to clean it up (Filter), and where to send the final result (Output). Go to **/etc/logstash/conf.d** directory make one

sudo nano pipeline.conf

set the input and filter section like this

```
input {
  beats {
    port => 5044
  }
}

filter {
  # Suricata JSON processing
  if [message] and [message] =~ "^{.*}$" {
    json {
      source => "message"
      target => "suricata"
      skip_on_invalid_json => true
    }
    if [suricata][timestamp] {
      date {
        match => [ "[suricata][timestamp]", "ISO8601" ]
        target => "@timestamp"
      }
    }
  }

  if [suricata] {
    mutate {
      rename => { "[suricata][src_ip]" => "src_ip" }
      rename => { "[suricata][dest_ip]" => "dest_ip" }
      rename => { "[suricata][event_type]" => "event_type" }
      rename => { "[suricata][alert]" => "alert" }
    }
  }

  if [event_type] == "alert" or [suricata][event_type] == "alert" {
    mutate { add_tag => ["suricata-alert"] }
  }
```

Next add virustotal and alienvault integrations. Create account in this websites and get your api keys.

Virustotal: https://www.virustotal.com/gui/user/codex07/apikey

AlienVault OTX: https://otx.alienvault.com/api

Add this to filter section

```
  if [suricata] {
    mutate {
      rename => { "[suricata][src_ip]" => "src_ip" }
      rename => { "[suricata][dest_ip]" => "dest_ip" }
      rename => { "[suricata][event_type]" => "event_type" }
      rename => { "[suricata][alert]" => "alert" }
    }
  }

  if [event_type] == "alert" or [suricata][event_type] == "alert" {
    mutate { add_tag => ["suricata-alert"] }
  }

  if [src_ip] {
    http {
      url => "https://otx.alienvault.com/api/v1/indicators/IPv4/%{src_ip}/general"
      verb => "GET"
      headers => {
        "X-OTX-API-KEY" => "2c5[redacted]cd3"
        "Accept" => "application/json"
      }
      target_body => "otx_response"
      connect_timeout => 5
      request_timeout => 20
      socket_timeout => 5
    }

^G Help          ^O Write Out     ^F Where Is      ^K Cut           ^T Execute       ^C Location      M-U Undo         M-A Set Mark     M-] To
^X Exit          ^R Read File     ^\ Replace       ^U Paste         ^J Justify       ^/ Go To Line    M-E Redo         M-6 Copy         ^B Whe
```

Virustotal api

```
    if "_httprequestfailure" in [tags] or ![otx_response] {
      mutate { add_tag => ["otx-failed"] }
    } else if [otx_response][pulse_info] and [otx_response][pulse_info][count] > 0 {
      mutate { add_tag => ["otx-malicious-ip"] }
    }
  }

  if [dest_ip] {
    http {
      url => "https://www.virustotal.com/api/v3/ip_addresses/%{dest_ip}"
      verb => "GET"
      headers => {
        "x-apikey" => "1f2[redacted]773"
        "Accept" => "application/json"
      }
      target_body => "vt_response"
      connect_timeout => 5
      request_timeout => 20
      socket_timeout => 5
    }

    if "_httprequestfailure" in [tags] or ![vt_response] {
      mutate { add_tag => ["vt-failed"] }
    } else if [vt_response][data] and [vt_response][data][attributes][last_analysis_stats][malicious] > 0 {
      mutate { add_tag => ["vt-malicious-ip"] }
    }
  }

  # Auth.log parsing & filtering "Failed password"
  if ![suricata] {
```

Prasing auth.logto filter Failed password. You can modify this by adding your custom filters.

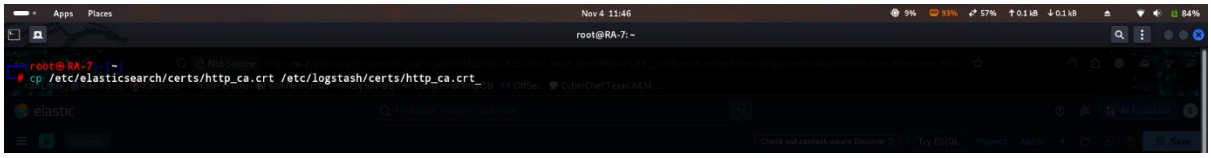
```
  }

  # Auth.log parsing & filtering "Failed password"
  if ![suricata] {
    if "Failed password" in [message] {
      grok {
        match => {
          "message" => "%{TIMESTAMP_ISO8601:iso_ts} %{HOSTNAME:host} sshd(?:\\[%{POSINT:pid}\\])?: Failed password for (?:invalid user )?%{USERNAME:user} from %{IP:src_ip} port %{INT:src_po
        }
        tag_on_failure => ["_grokparsefailure_auth_fail"]
      }
      mutate {
        add_tag => ["ssh-auth-fail"]
      }
      if [iso_ts] {
        date {
          match => [ "iso_ts", "YYYY-MM-dd'T'HH:mm:ss.SSSSSSZ" ]
          target => "@timestamp"
        }
      }
    } else {
      drop { }
    }
  }
}
```

Next set the output

Host => [“hostname/Ip_address:9200”]

Password => “your elasticsearch password”

ssl_certificate_authorities => get this from /etc/elasticsearch/certs/http_ca.crt

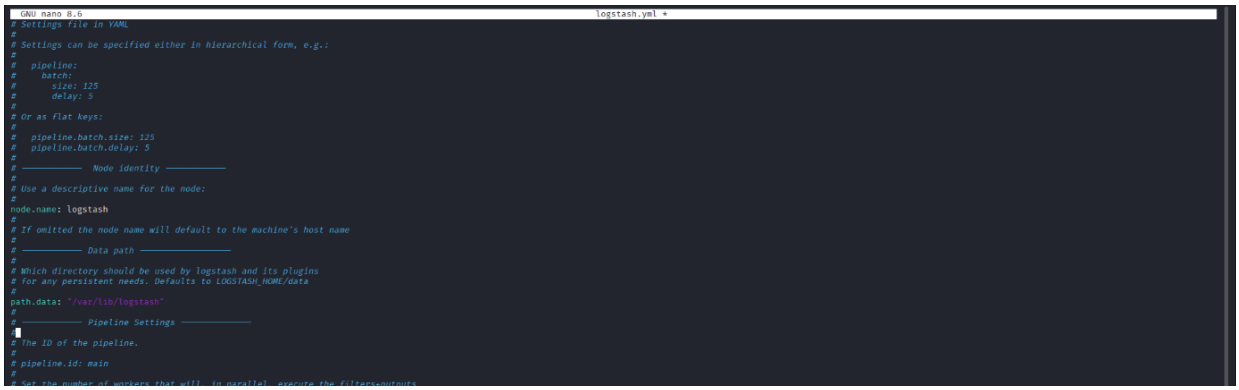



```
    } else {
      drop { }
    }
  }
}

output {
  elasticsearch {
    hosts => ["https://ra-7:9200"]
    user => "elastic"
    password => "p9SkODzUFlo6Wnz4vOvz"
    index => "suricata-%{+YYYY.MM.dd}"
    ssl_enabled => true
    ssl_certificate_authorities => ["/etc/logstash/certs/http_ca.crt"]
    ilm_enabled => false
  }

  stdout { codec => rubydebug }
}

^G Help        ^O Write Out   ^F Where Is    ^K Cut         ^T Execute     ^C Location    M-U Undo       M-A Set M
^X Exit        ^R Read File   ^\ Replace     ^U Paste       ^J Justify     ^/ Go To Line  M-E Redo       M-6 Copy
```

Now edit logstash.yml

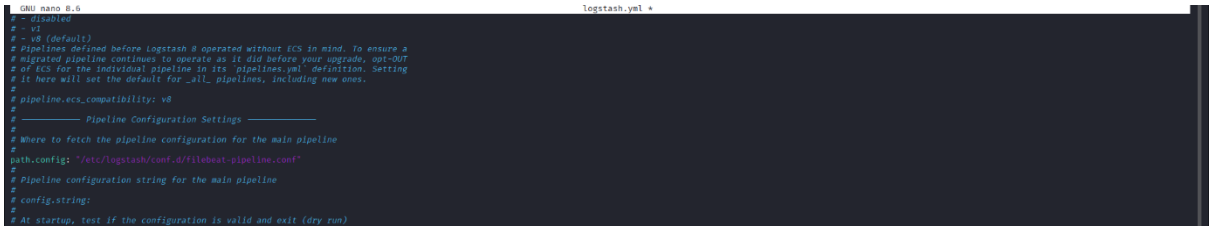
sudo nano /etc/logstash/logstash.yml

path.data: “/etc/lib/logstash”



Set the pipeline.conf path

Path.config: “/etc/logstash/conf.d/filebeat-pipeline.conf”

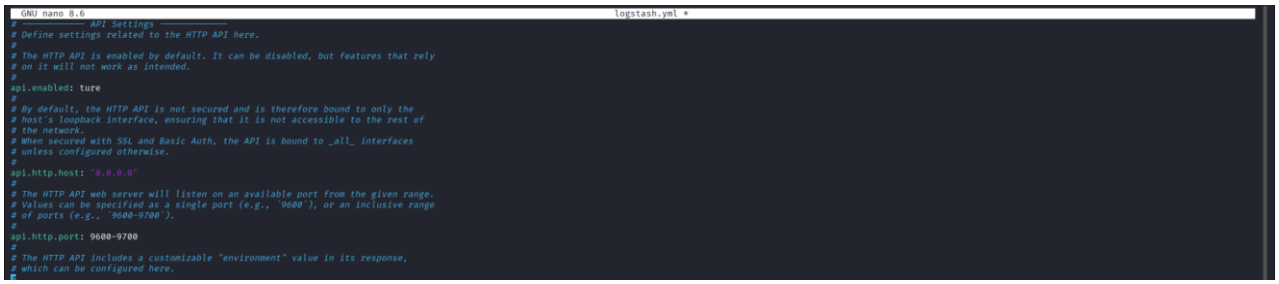


Edit the API Settings

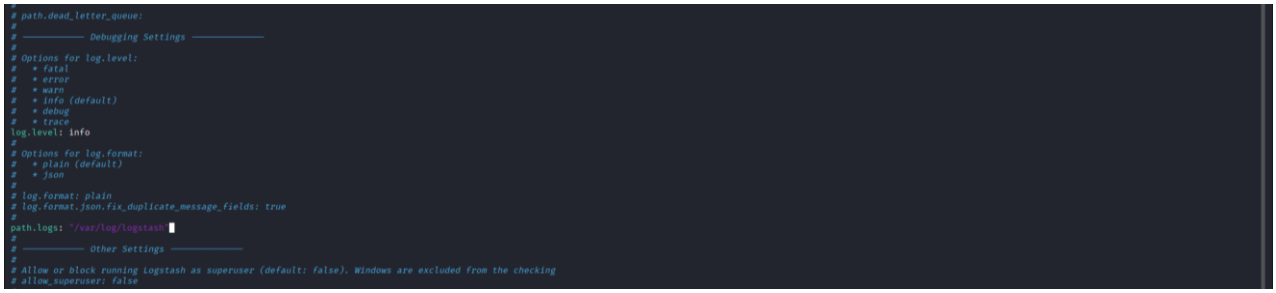
api.enable: true

api.http.host: “0.0.0.0”

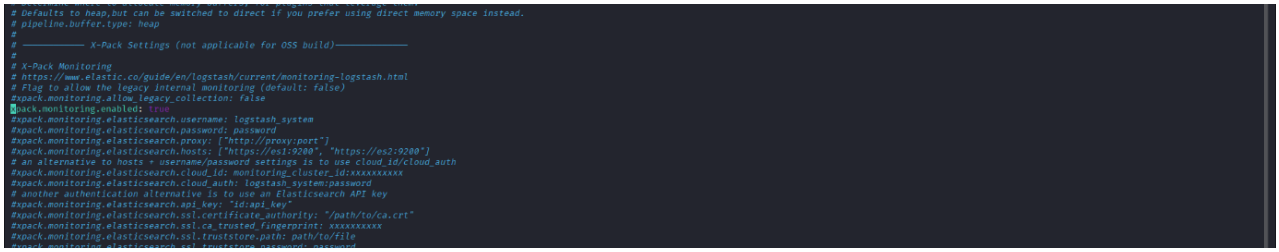
api.http.port: 9600-9700



path.logs: “/var/log/logstash”

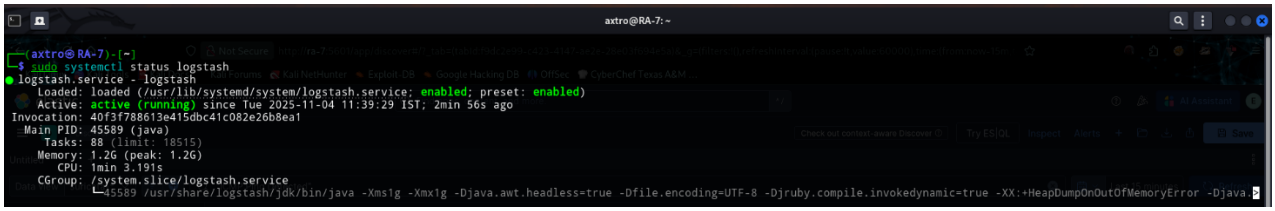


Enable xpack.monitoring



Save and exit

Check the status

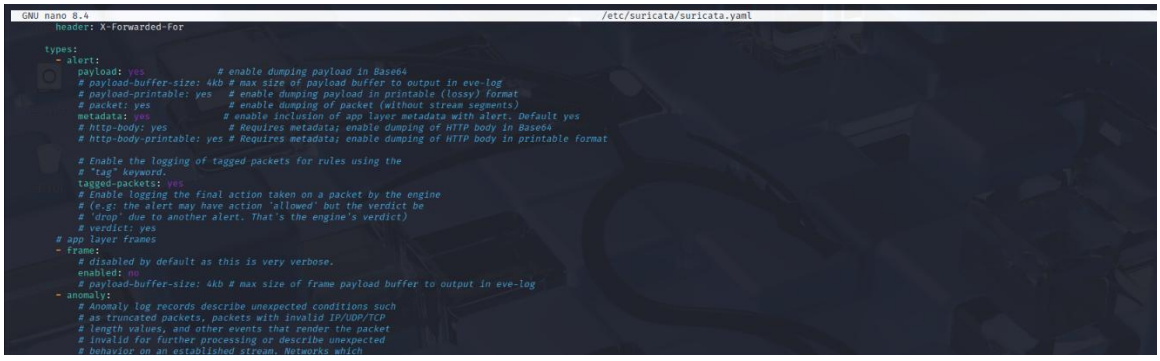


4) Configure Suricata

Go to **/etc/suricata** directory and edit **suricata.yaml** file

sudo nano suricata.yaml

set the output eve.json



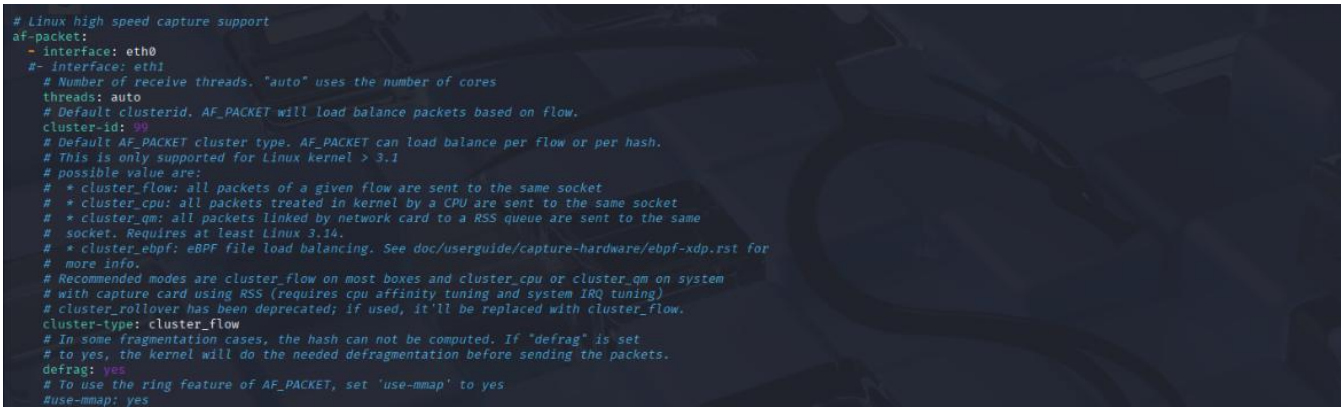
Now set your network interface to capture the traffic and log it in eve.json

here I'm using virtualbox, so mine is eth0

To know your interface:

ip link show OR ifconfig

then set it to yaml

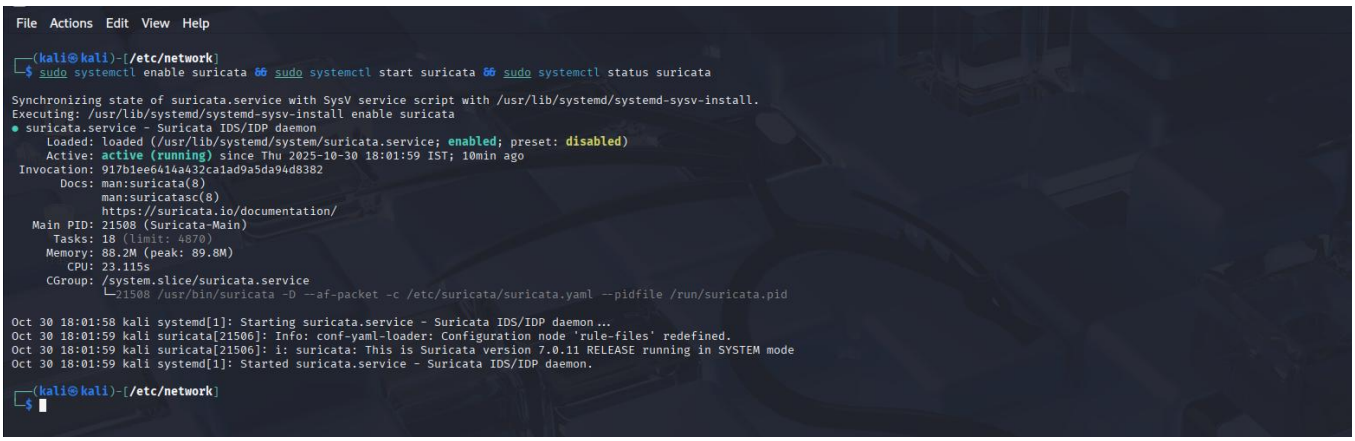


Next set the rules for suricata

In the rules directory you can see many pre-defined rules but they are all commented and not defined in suricata.yaml .You can create custom rules for example local.rules. Here I made two rule files (nmap.rules and suricata.rules).

First set the rules file in suricata.yaml. Scroll down to the end

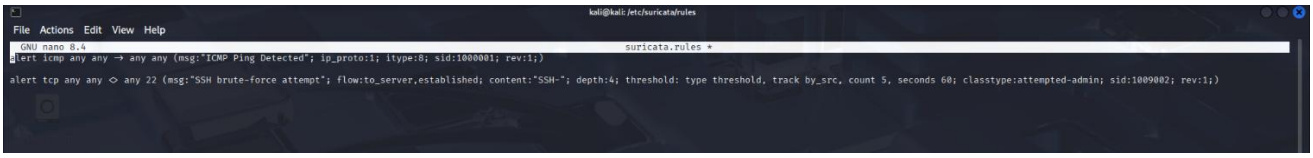
Save and exit, then check the status.



After this make the rule files

Go to /etc/suricata/rules directory.

sudo nano suricata.rules



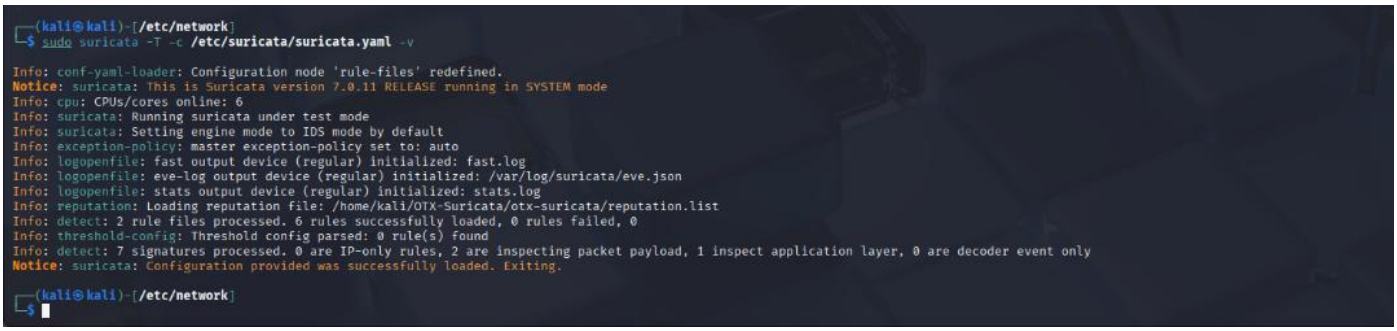
Here you can see two rules, The first one is to detect Ping attemps and the second one is to detect SSH brute-force attempt. Actually both this rules are for NIDS, that's why there is **any any → any any** which means from any source ip and source port to any destination ip and destination port. The second rule is to detect the login attempts to port 22 which is SSH service. Proper way to confirm this is to check the logs for **"Failed password"** from **auth.log**

If there are many logs with Failed password within a seconds, Then it's a Brute-Force attempt.

Like this create the nmap rules as well. Note that every rules sid must be unique, otherwise it won't work.

Run this command to check the rules are working properly or not:

sudo suricara -T -c /etc/suricata/suricata.yaml -v



Check the eve.json is generated or not in the folder /var/log/suricata

5) Configure Filebeat

Go to /etc/filebeat directory and edit the filebeat.yml file

sudo nano filebeat.yml

go to the filebeat.input setction and set the path to the logs which we want to analyse.

/var/log/auth.log

/var/log/suricata/eve.json

```
# ============================== 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: true

  # Paths that should be crawled and fetched. Glob based paths.
  paths:
    - /var/log/auth.log
    - /var/log/suricata/eve.log
    #- /var/log/syslog
    #- /var/log/boot.log
  json.keys_under_root: true
  json.add_error_key: true
    #- c:\programdata\elasticsearch\logs\*

  # Exclude lines. A list of regular expressions to match. It drops the lines that are
```

Next comment the output elasticsearch section and uncomment the output logstash section

hosts: [“your_IP/Host name:5044”] (Hostname of the system running Logstash)

It’s better to give the hostname instead to IP. IP address will change if you connect to different network.

```
  #ssl.certificate_authorities: "/etc/filebeat/certs/http_ca.crt"


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

  # Optional SSL. By default is off.
  # List of root certificates for HTTPS server verifications
  #ssl.certificate_authorities: ["/etc/pki/root/ca.pem"]

  # Certificate for SSL client authentication
  #ssl.certificate: "/etc/pki/client/cert.pem"

  # Client Certificate Key
  #ssl.key: "/etc/pki/client/cert.key"

# ================================ Processors =====================================
processors:
  - add_host_metadata:
      when.not.contains.tags: forwarded
  - add_cloud_metadata: ~
  - add_docker_metadata: ~
  - add_kubernetes_metadata: ~
  - add_fields:
      target: ''
      fields:
        event_type: 'alert'
```

Check the connection between filebeat and logstash. Note your logstash must be running otherwise it won’t connect

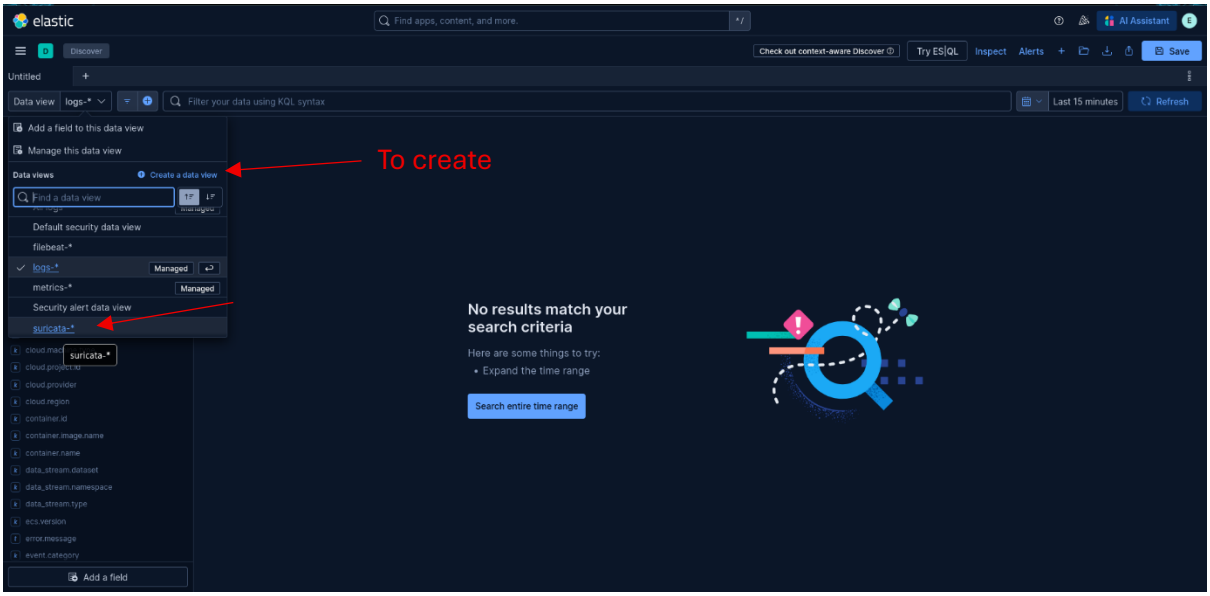
sudo filebeat test output

```
kali@kali: ~/Pictures
File  Actions  Edit  View  Help

┌──(kali㉿kali)-[~/Pictures]
└─$ sudo filebeat test output
logstash: ra-7:5044 ...
  connection ...
    parse host ... OK
    dns lookup ... OK
    addresses: 192.168.31.203
    dial up ... OK
  TLS ... WARN secure connection disabled
  talk to server ... OK

┌──(kali㉿kali)-[~/Pictures]
└─$
```

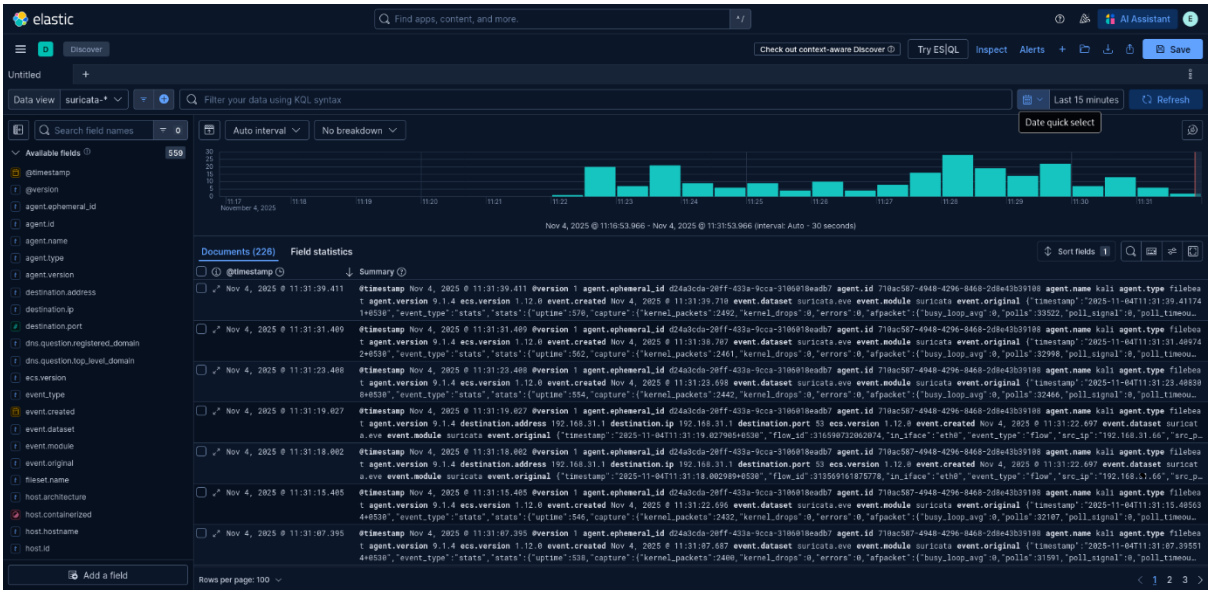
Check for logs in kibana

Go to Kibana => Discover, change the index to suricata-* (If not click create one) OR you can simply use filebeat-* by changing the intex in pipeline's output setcion

Index => "filebeat-%{+YYY.MM.DD}"

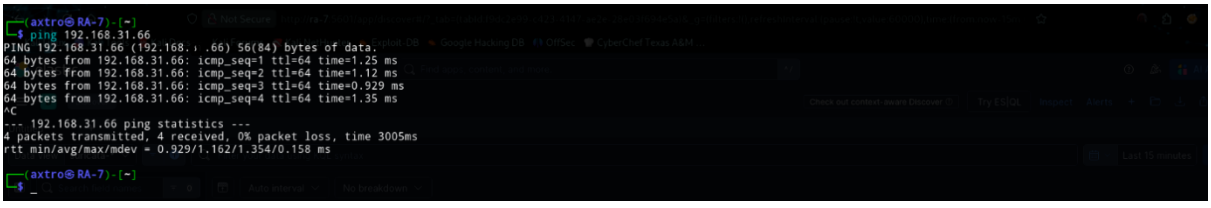


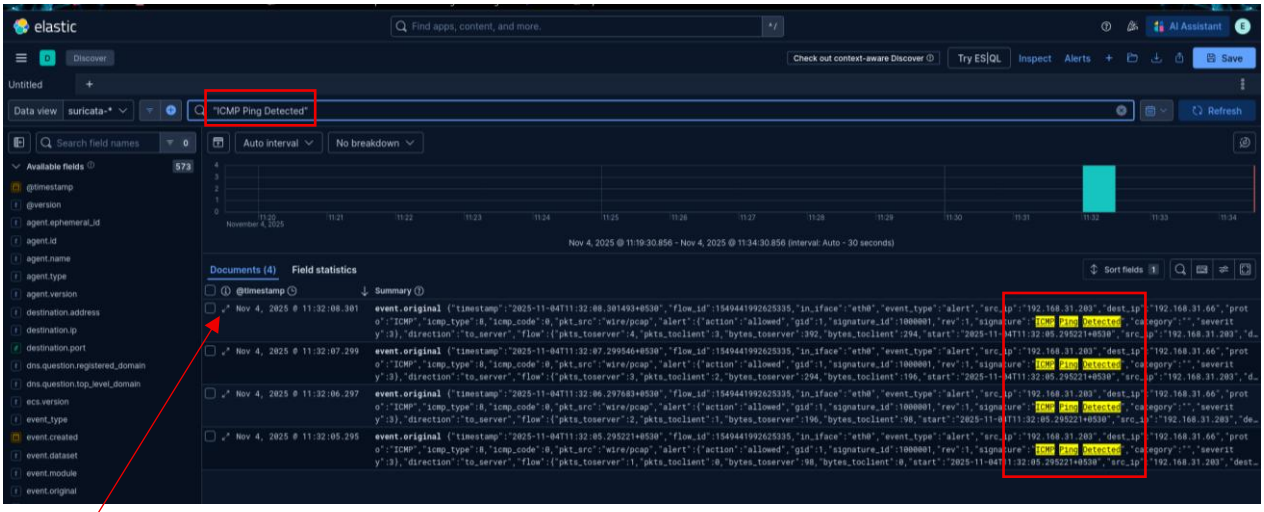
Then you can see the logs



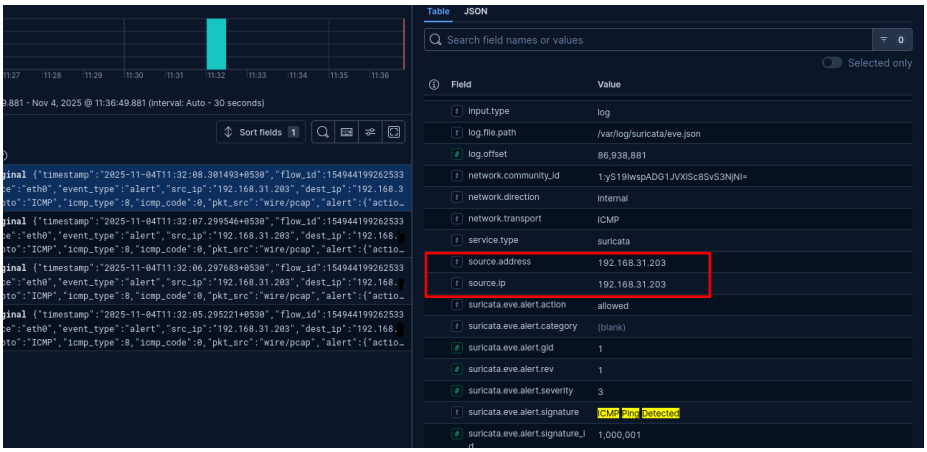
Now check the suricata rules are triggered

Let's ping to the host system and search for the message that we set in the rule to detect ICMP ping





Expand it



The rule works properly and can see the alert message. Expand the log to get more details like source IP, source port, etc. You can set alerts in the free Kibana version, but your options are limited to the features available with the basic license. With the free version, you can create the rules themselves like index threshold or query rules, but you are restricted to using only a limited set of actions for notifications, primarily logging to an index. Sending notifications through more advanced channels like Slack, email, or webhooks requires a paid commercial license.

Dashboard Setup

1) Create dashboard

- Kibana ⟶ Analytics ⟶ Dashboards
- Ckick **Create dashboard**
- Click **Save,** then give a name for your dashboard and save it.

2) Create visualization

Lets create a visualization for ICMP Ping

- Analytics ⟶ visualize library ⟶ Create visualization ⟶ Lens.
- Choose your data view (filebeat-* or suricata-*)
- Set the visualization type to **Area**
- Add filter (+ button on the top left)

  Field= suricata.eve.icmp_type
  Operator= is
  Field value= 8
- Drag **@timestamp** to the horizontal axis
- Click vertical axis and select **count** then set **suricata.eve.icmp_type** to the field.
- Breakdown

  Field= source.ip.keyword (if this not working you can use any field related to source ip. Eg. src_ip)

  No.of values= 10
- Click **Save** ⟶ Give name ⟶ Add to existing dashboard
- Search for your dashboard ⟶ Save ⟶ Go to dashboard

Visualization for Nmap scans

- Go to Lens
- Add filter : add filter based on the suricata rules we set earlier by their signature(msg).

  Field= alert.signature (if this not working try **suricata.eve.alert.signature.keyword**).

  Operator= is

  Field value= msg from the suricata nmap rules (Eg. Nmap Xmas scan)

  Click **OR** and do the same thing with other nmap signatures from the rules.

- Visualization type= **Pie**
- Slice by

  Field= **suricata.eve.alert.signature.keyword** (Same as you select as the field for filter).

  Add another field.

  Field= **source.ip.keyword**

  No.of volume= 10

  Ranked by count of records

Visualization for SSH bruteforce attempt

First make sure your pipeline can prase logs from auth.log. If not, add a grok filter. This is something you should handle carefully. Add a grok filter for auth.log without affecting your eve.json filter by separate them properly in the pipeline.

Afther this

- Go to **Lens**
- Choose your data view
- In the top filter bar, enter the KQL query: **message : "Failed password"**
- Drag @timestamp to the horizontal axis
- Select **Count** in vertical axis
- In breakdown, set **src_ip** and set top 10
- Save it and add to your dashboard.

You can create many visualizations like this.

Conclusion

The development of a Host-Based Intrusion Detection System (HIDS) using ELK Stack and Suricata demonstrates the effectiveness of open-source tools in strengthening system security and monitoring host level threats. By integrating VirusTotal and AlienVault OTX, the system gains access to real time threat intelligence, enhancing its ability to detect and correlate malicious activities efficiently.

While this project focuses on a host level monitoring, Suricata is inherently designed as a Network Intrusion Detection and Prevention system (NIDS/NIPS). When deployed on a network gateway or connected to a SPAN (Switched Port Analyzer) or mirror port, Suricata can capture and analyse traffic across the network, providing comprehensive visibility into network-level attacks.

This project demonstrated how modern cybersecurity frameworks can integrate host-level log analysis and external threat intelligence to provide a comprehensive defence against evolving cyber threats. Its modular and scalable design also supports future enhancements, including machine learning-driven anomaly detection and automated incident response, establishing a strong foundation for advanced cybersecurity implementations.