

SIEM Documentation

Joel Koszorus

Overview

The SIEM (Security Information and Event Manager) system comprises three individual software products, Elasticsearch, Logstash, and Kibana. These three programs come



together to form what is known as the ELK Stack, which is a widely used and open-source log analytic security solution. The ELK collection is

maintained by *Elastic*, the company whose founders created it. ELK is free to use and doesn't require any upfront or ongoing software licensing fees, however the building, growing, and maintaining of the stack requires some infrastructure and resources. The following document aims to assist and outline the steps for configuring the ELK Stack on a Ubuntu Server hosted by a Raspberry Pi for an on-premises production environment.

Getting Started

Before we begin it is important to have the necessary components so the system can operate properly. Below is a list detailing the required hardware and software.

1. Raspberry Pi 4 or 5, this will act as the server our SIEM will be installed on
 - a. 8gb RAM recommended at the very least, 16gb and 32gb machines are also common to run the ELK Stack
2. Raspberry Pi Imager software, which is recommended when imaging a Pi
 - a. or Ubuntu Server v22.04 (or later) iso file for manual imaging
3. Access to a PC, the OS can be preferential
4. Internet access, which is not required but is highly recommend
 - a. Installation can be done 'offline' however the following documentation only details the default 'online' installation
5. One or more endpoint devices (VM or physical computer), these will be used to ingest the data feeding into ELK

- a. The ELK Stack is compatible with MacOS, Linux, and Windows operating systems

Configurations

Raspberry Pi and Ubuntu Server

The first step in the process is imaging the Pi and installing Ubuntu Server 24.04 on the device. The simplest way to go about this is to use the Raspberry Pi Imager application, which can be downloaded from www.raspberrypi.com/software/. The imaging software works exceptionally well with Pi devices and it provides a smooth install. After writing the Ubuntu Server OS onto a microSD card, insert it into the device, connect the necessary peripherals (monitor, mouse, keyboard), and you're ready to power up the Pi.

After the device has successfully booted up, connect it to your local network, either via ethernet connection or wi-fi. Note that there is no GUI with Ubuntu Server and everything will be configured with commands on a command line interface (CLI). That being said, a good starting step would be to update the device by running the following commands:

- `sudo apt update && sudo apt upgrade -y`
- `sudo apt dist-upgrade -y`
- `sudo apt install zip unzip -y`
- `sudo apt install jq -y`
- `sudo reboot`

These commands will install dependencies, update it, and prepare the system for the rest of the configurations and installs we will be getting into. After the updates and installs are complete the last step is going to be setting a static IP address for the server, in my system I went with 192.168.0.100.

Elasticsearch

Now that the Pi device is set up and ready to go we can start with the elasticsearch install. First we need download and install the public signing key, we can do that with by entering the following command in the CLI:

```
→ wget -qO - https://artifacts.elastic.co/GPG-KEY-elasticsearch | sudo gpg --dearmor  
-o /usr/share/keyrings/elasticsearch-keyring.gpg
```

Next, we'll add the repository definition to our system:

```
→ echo "deb https://artifacts.elastic.co/packages/8.x/apt stable main" | sudo tee -a  
/etc/apt/sources.list.d/elasticsearch-8.x.list
```

From here all that's left to do is install Elasticsearch:

```
→ sudo apt install elasticsearch
```

After the installation is complete we can move further into the elasticsearch setup.

Elasticsearch configurations are done using a configuration file that allows you to configure general settings, as well as network settings (host and port). The config file we are looking for is elasticsearch.yml and it can be found where it was installed:

```
→ cd /etc/elasticsearch
```

From that directory we will edit the .yml file, and for that we use the nano command and specify the file:

```
→ nano elasticsearch.yml
```

From inside this config file it is important to specify the network.host and the http.port numbers. For network.host we can enter the IP address of the host server (our Pi) 192.168.0.100, and the default http port for elasticsearch is 9200 so we will enter that value in as well. The rest of the file can remain as is, save and exit elasticsearch.yml.

```
GNU nano 7.2 /etc/elasticsearch/elasticsearch.yml
# 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: 192.168.0.100
#
# 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: ["host1", "host2"]
```

Once back in the command line we can start the service:

- `systemctl start elasticsearch`
- `systemctl status elasticsearch`

If everything has been configured successfully we should see this:

```
root@pi:~# systemctl status elasticsearch
● elasticsearch.service - Elasticsearch
   Loaded: loaded (/usr/lib/systemd/system/elasticsearch.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-06-07 20:03:11 PDT; 21h ago
     Docs: https://www.elastic.co
   Main PID: 769 (java)
    Tasks: 116 (limit: 8686)
   Memory: 4.7G (peak: 4.7G)
      CPU: 1h 21min 2.084s
   CGroup: /system.slice/elasticsearch.service
           └─ 769 /usr/share/elasticsearch/jdk/bin/java -Xms4m -Xmx64m -XX:+UseSerialGC -Dcli.name=server -Dcli.script=/usr/share>
              1094 /usr/share/elasticsearch/jdk/bin/java -Des.networkaddress.cache.ttl=60 -Des.networkaddress.cache.negative.ttl=1>
              1130 /usr/share/elasticsearch/modules/x-pack-m1/platform/linux-aarch64/bin/controller

Jun 07 19:59:59 pi systemd[1]: Starting elasticsearch.service - Elasticsearch...
Jun 07 20:00:27 pi systemd-entrypoint[769]: Jun 07, 2024 8:00:27 PM sun.util.locale.provider.LocaleProviderAdapter <clinit>
Jun 07 20:00:27 pi systemd-entrypoint[769]: WARNING: COMPAT locale provider will be removed in a future release
Jun 07 20:03:11 pi systemd[1]: Started elasticsearch.service - Elasticsearch.
```

Elasticsearch is configured and we can leave it to run on its own while we configure its sister program, Kibana.

Kibana

The steps to install Kibana are very similar to how we installed and configured elasticsearch. Since we already installed the signing key in the first step for elasticsearch we can ignore this step for Kibana. We will simply run the following command to download and install Kibana:

- `sudo apt-get update && sudo apt-get install kibana`

After the install is complete we can begin configuring Kibana. We can do this the same way configured Elasticsearch:

- `cd /etc/kibana`
- `nano kibana.yml`

The specific settings we need to change are `server.port`, `server.host`, `elasticsearch.hosts`, `elasticsearch.username`, and `elasticsearch.password`. Change them to the following values:

1. `server.port: 5601`

2. `server.host: 192.168.0.100` (or whichever static IP you assigned to your Pi server)
3. `elasticsearch.hosts: ["http://192.168.0.100:9200"]`
4. `elasticsearch.username: "kibana_system"`
5. `elasticsearch.password: "password"`

```
GNU nano 7.2 /etc/kibana/kibana.yml
# For more configuration options see the configuration guide for Kibana in
# https://www.elastic.co/guide/index.html

# ===== System: Kibana Server =====
# 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
# 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: "192.168.0.100"

# Enables you to specify a path to mount Kibana at if you are running behind a proxy.
# Use the 'server.rewriteBasePath' setting to tell Kibana if it should remove the basePath
# from requests it receives, and to prevent a deprecation warning at startup.
# This setting cannot end in a slash.
server.basePath: ""
```

```
GNU nano 7.2 /etc/kibana/kibana.yml

# ===== System: Elasticsearch =====
# The URLs of the Elasticsearch instances to use for all your queries.
elasticsearch.hosts: ["https://192.168.0.100:9200"]

# 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: "kibana_system"
elasticsearch.password: "password"

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

# Time in milliseconds to wait for Elasticsearch to respond to pings. Defaults to the value of
```

The rest of the configuration can remain as it is with the default settings. Kibana should be all set up and ready to start now. Like we did with Elasticsearch, we are going to use the `systemctl` commands to get the service going:

- `systemctl start kibana`
- `systemctl status kibana`

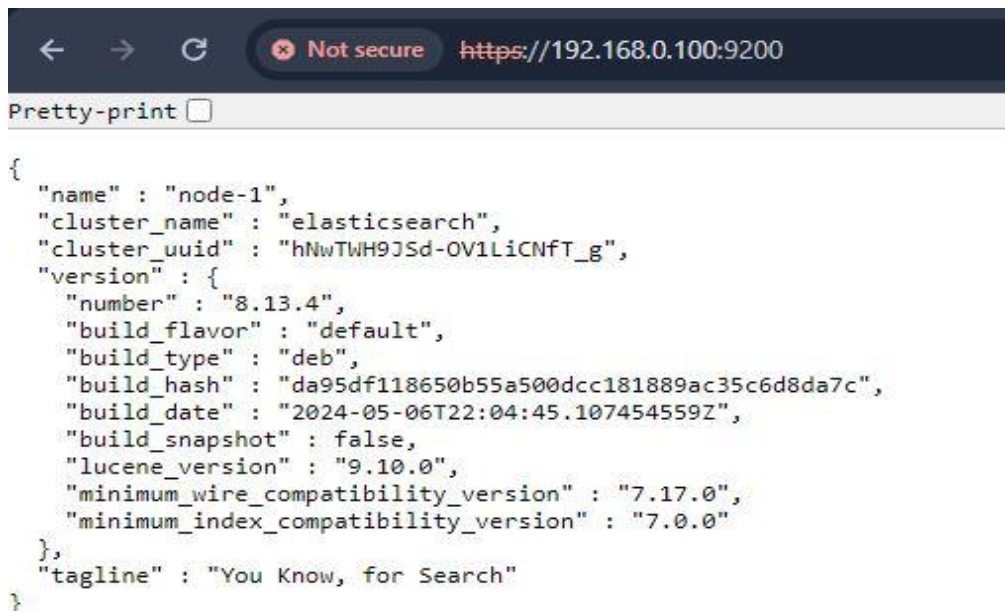
If everything has been configured successfully we should see this:

```
root@pi:~# systemctl status kibana
• kibana.service - Kibana
   Loaded: loaded (/usr/lib/systemd/system/kibana.service; enabled; preset: enabled)
   Active: active (running) since Fri 2024-06-07 19:59:59 PDT; 21h ago
     Docs: https://www.elastic.co
   Main PID: 770 (node)
    Tasks: 11 (limit: 8686)
   Memory: 686.5M (peak: 1.0G)
      CPU: 3h 47min 40.154s
   CGroup: /system.slice/kibana.service
           └─770 /usr/share/kibana/bin/../../node/bin/node /usr/share/kibana/bin/../../src/cli/dist

Jun 08 17:44:10 pi kibana[770]: [2024-06-08T17:44:10.429-07:00][INFO ][plugins.securitySolution.ManifestManager] Processed [0] Poli>
Jun 08 17:44:10 pi kibana[770]: [2024-06-08T17:44:10.475-07:00][INFO ][plugins.securitySolution.endpoint:user-artifact-packager:1.0>
Jun 08 17:45:12 pi kibana[770]: [2024-06-08T17:45:12.376-07:00][INFO ][plugins.securitySolution.endpoint:user-artifact-packager:1.0>
Jun 08 17:45:12 pi kibana[770]: [2024-06-08T17:45:12.446-07:00][INFO ][plugins.securitySolution.ManifestManager] Count of current s>
Jun 08 17:45:12 pi kibana[770]: [2024-06-08T17:45:12.943-07:00][INFO ][plugins.securitySolution.ManifestManager] Processed [0] Poli>
Jun 08 17:45:12 pi kibana[770]: [2024-06-08T17:45:12.975-07:00][INFO ][plugins.securitySolution.endpoint:user-artifact-packager:1.0>
Jun 08 17:46:15 pi kibana[770]: [2024-06-08T17:46:15.316-07:00][INFO ][plugins.securitySolution.endpoint:user-artifact-packager:1.0>
Jun 08 17:46:15 pi kibana[770]: [2024-06-08T17:46:15.370-07:00][INFO ][plugins.securitySolution.ManifestManager] Count of current s>
Jun 08 17:46:15 pi kibana[770]: [2024-06-08T17:46:15.915-07:00][INFO ][plugins.securitySolution.ManifestManager] Processed [0] Poli>
Jun 08 17:46:15 pi kibana[770]: [2024-06-08T17:46:15.966-07:00][INFO ][plugins.securitySolution.endpoint:user-artifact-packager:1.0>
```

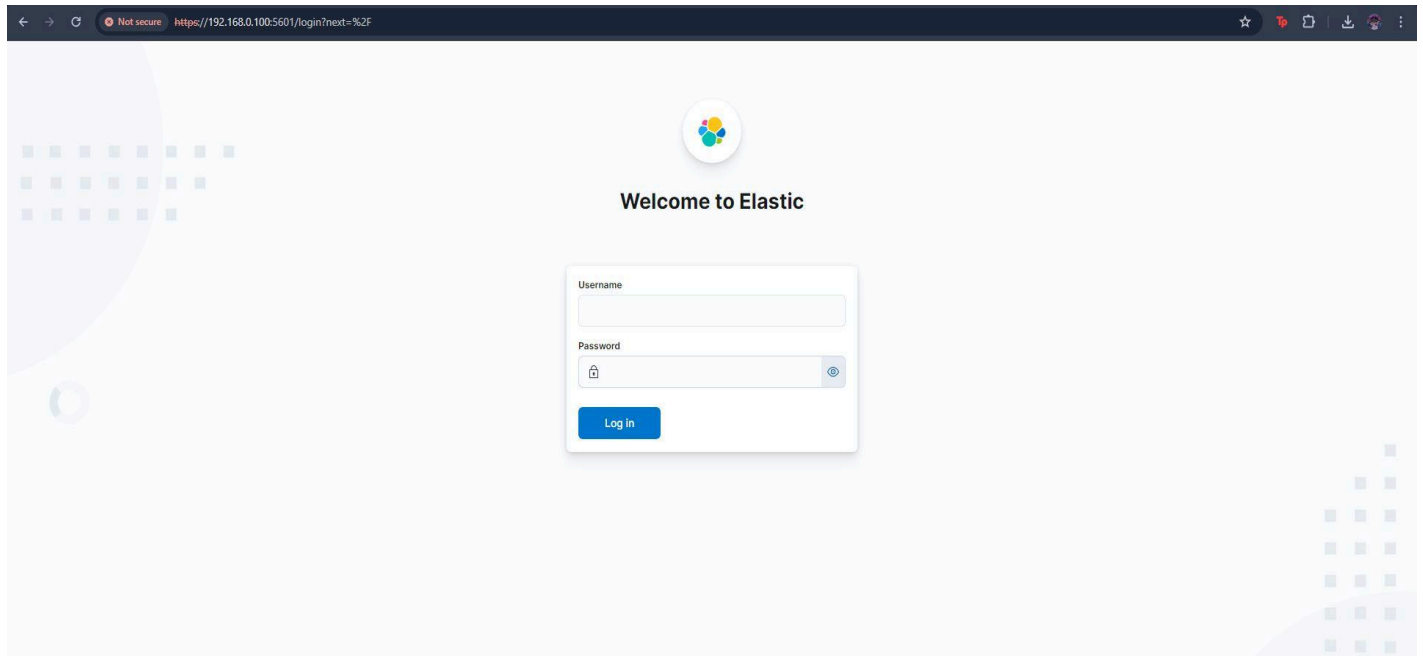
The configurations we made to Elasticsearch and Kibana enable them to communicate together on the same server. Now we can test each to see if we get any web response, we'll start with elasticsearch.

In a web browser enter the IP of the Pi server, in my case it was 192.168.0.100, and make sure to specify the port where we configured Elasticsearch on (:9200). Elasticsearch should return the following in the web browser:



```
{
  "name" : "node-1",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "hNwTWH9JSd-OV1LiCNfT_g",
  "version" : {
    "number" : "8.13.4",
    "build_flavor" : "default",
    "build_type" : "deb",
    "build_hash" : "da95df118650b55a500dcc181889ac35c6d8da7c",
    "build_date" : "2024-05-06T22:04:45.107454559Z",
    "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"
}
```

Moving onto Kibana, we will test it the same way. Start by entering the IP of the server into your web browser and append the Kibana port (:5601). It should look like “192.168.0.100:5601”. Hit enter and you should be redirected to the following page:



The default username for Elastic is “elastic” and the password will be the one you set when prompted from the Elasticsearch install. From here you can login and proceed to the dashboards.

Securing the ELK Stack

Security is essential even for the security system itself, and there are ways we can secure the connection between Elasticsearch, Kibana, and the users. A built-in feature with Elasticsearch is the “certutil” function, shorthand for “certificate utility”, allows us to generate certs. The process of securing Elasticsearch and Kibana starts with enabling TLS (Transport Layer Security) on the HTTP layer to ensure all communications to and from your server are encrypted.



When you run the elasticsearch-certutil tool in http mode, the tool asks several questions about how you want to generate certificates. While there are numerous options, the

following steps will result in certificates that will work for our environment. We are going to start by running certutil:

→ `/usr/share/elasticsearch/bin/elasticsearch-certutil http`

This command generates a .zip file that contains certificates and keys to use with Elasticsearch and Kibana. After generating your certificate, enter a password for your private key when prompted. Next, unzip the generated .zip file (`/usr/share/elasticsearch/elasticsearch-ssl-http.zip`), this compressed file contains one directory for both Elasticsearch and Kibana. In the `/elasticsearch` directory you will have the file `http.p12` and in the `/kibana` directory you will have the file `elasticsearch-ca.pem`. First, we will copy the `http.p12` file to the `/etc/elasticsearch/certs` directory:

→ `cp /usr/share/elasticsearch/elasticsearch/http.p12 /etc/elasticsearch/certs`

After that we are gonna do the same thing for the .pem file in the same `/kibana` directory:

→ `cp /usr/share/elasticsearch/kibana/elasticsearch-ca.pem /etc/kibana`

Now that we have moved the certificates to their proper locations we're going to take a look at those config files again and configure the security settings. Start with:

→ `cd /etc/elasticsearch`

→ `nano elasticsearch.yml`

Once we are in the file editor we are going to adjust some security settings, these can be identified `xpack.security` as their prefix. You want to specifically look at and configure as follows:

1. `xpack.security.http.ssl.enabled: true`
2. `xpack.security.http.ssl.keystore.path: /etc/elasticsearch/http.p12`

```
GNU nano 7.2 /etc/elasticsearch/elasticsearch.yml
# generated to configure Elasticsearch security features on 23-05-2024 22:32:06
#
# -----
# 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: /etc/elasticsearch/certs/http.p12
```


And to finish securing Elasticsearch we are going to add the password for your private key to the secure settings with the command:

```
→ /usr/share/elasticsearch/bin/elasticsearch-keystore add  
xpack.security.http.ssl.keystore.secure_password
```

We can now restart the Elasticsearch service using systemctl:

```
→ systemctl restart elasticsearch
```

Finally, we are going to encrypt HTTP client communications for Kibana. The way the stack functions, browsers send traffic to Kibana and Kibana sends traffic to Elasticsearch. These communication channels are configured to use TLS. We are encrypting traffic between Kibana and Elasticsearch, and then encrypting traffic between our browser and Kibana. First we are going to secure communicating between services, to do that we need to copy to the *elasticsearch-ca.pem* file from the */usr/share/elasticsearch/kibana* directory like we did with Elasticsearch:

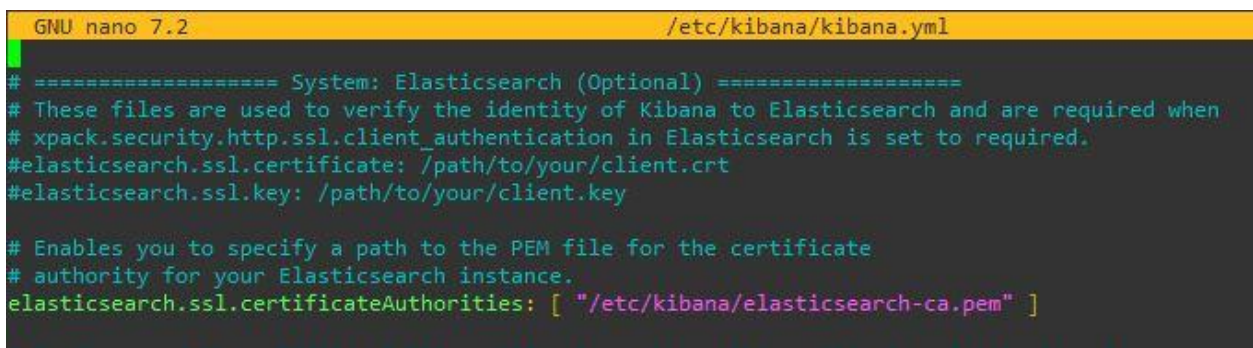
```
→ cp /usr/share/elasticsearch/kibana/elasticsearch-ca.pem /etc/kibana
```

Now that the *-ca* file is where it needs to be we can proceed to the *kibana.yml* file to make changes to the config. To do this enter the following commands:

```
→ cd /etc/kibana  
→ nano kibana.yml
```

From the text editor we need to change the following values:

1. `elasticsearch.ssl.certificateAuthorities: ["/etc/kibana/elasticsearch-ca.pem"]`
2. `elasticsearch.hosts: https://192.168.0.100:9200`



```
GNU nano 7.2 /etc/kibana/kibana.yml  
# ===== System: Elasticsearch (Optional) =====  
# These files are used to verify the identity of Kibana to Elasticsearch and are required when  
# xpack.security.http.ssl.client_authentication in Elasticsearch is set to required.  
#elasticsearch.ssl.certificate: /path/to/your/client.crt  
#elasticsearch.ssl.key: /path/to/your/client.key  
  
# Enables you to specify a path to the PEM file for the certificate  
# authority for your Elasticsearch instance.  
elasticsearch.ssl.certificateAuthorities: [ "/etc/kibana/elasticsearch-ca.pem" ]  
# To disable this setting, set it to an empty list: []
```

Save the *kibana.yml* file for now, we will need to make additional changes in a moment. At this point the communication between Kibana and Elasticsearch is encrypted, our next task is to encrypt the traffic between our browser and Kibana. The following steps are

going to show you how to create a Certificate Signing Request (CSR) for Kibana. A CSR contains information that a CA (certificate authority) uses to generate and sign a security certificate.

We are going to double back to the `/usr/share/elasticsearch/bin` directory to use `certutil` again to generate a server certificate and private key for Kibana. We can run the following command to generate a `csr-bundle.zip` file:

```
→ /usr/share/elasticsearch/bin/elasticsearch-certutil csr -name kibana-server
```

After executing this command a .zip file should be created, unzip that file and you should be left with the following files inside: `kibana-server.csr` and `kibana-server.key`. Now we need to copy these files to their proper location:

```
→ cp /usr/share/elasticsearch/kibana-server/kibana-server.csr /etc/kibana
```

```
→ cp /usr/share/elasticsearch/kibana-server/kibana-server.key /etc/kibana
```

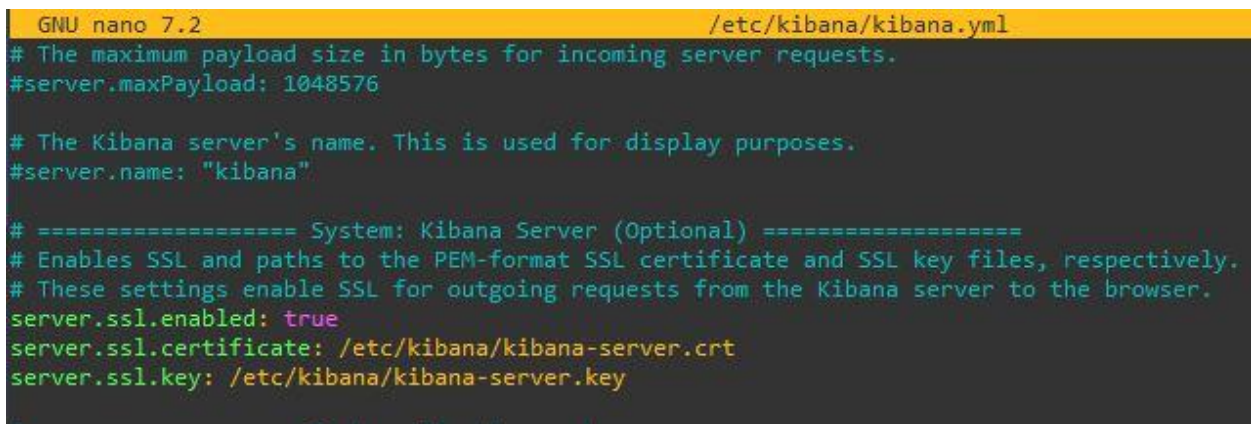
Now we navigate to our .yml file to finish the configuration:

```
→ cd /etc/kibana
```

```
→ nano kibana.yml
```

The settings we want to edit here are prefixed with `server.ssl`. We want the following values as shown:

1. `server.ssl.enabled: true`
2. `server.ssl.certificate: /etc/kibana/kibana-server.crt`
3. `server.ssl.key: /etc/kibana/kibana-server.key`



```
GNU nano 7.2 /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: true
server.ssl.certificate: /etc/kibana/kibana-server.crt
server.ssl.key: /etc/kibana/kibana-server.key
```

The security configurations are now complete for both Elasticsearch and Kibana. From this point on we will be accessing Kibana on our web browser via HTTPS. For example, `https://192.168.0.100:5601`.

Conclusion

Using the ELK Stack to power a SIEM on a Raspberry Pi provides a cost-effective and versatile solution for on-premises log analysis and security monitoring. It is a system capable of ingesting, processing, and visualizing security data from multiple endpoints. My documentation has detailed the essential steps for configuring Elasticsearch and Kibana on Ubuntu Server, hosted by a Raspberry Pi, and includes steps to secure the communications with TLS, ensuring that data integrity and confidentiality are maintained. The documentation also highlights proper hardware selection, it's important to have 8GB of RAM for efficient operation, which can be increased to meet the needs of different environments. It also aims to show the flexibility of the ELK Stack, which is compatible with multiple operating systems, making it a versatile choice for diverse infrastructures.

In summary, I hope that this information provides a firm foundation for setting up a SIEM system with the ELK Stack on a Raspberry Pi, and my aim was to equip you with the knowledge to build, configure, and secure your environment effectively yourself.

Developing this SIEM has been a privilege, it was a lot more challenging than expected and I learned a lot of valuable skills both technical and practical. After many hours, and lots of trial and error, I have built a solid understanding of how a SIEM operates, and specifically with the ELK Stack. It was an awesome experience that, no doubt, has had a positive impact on me as a new IT professional and furthermore as a future cybersecurity student.