



## Implementación de un sistema IDS/SIEM

**Pablo Jiménez Márquez**

**Ciclo Superior de Administración de Sistemas Informáticos en Redes**

**IES Medina Ahazara**

**13/06/2023**

# Índice

1. Introducción.....	6
1.1 Justificación.....	6
1.2 Objetivos.....	6
1.3 Estructura.....	6
1.4 Planificación.....	7
1.5 Presupuesto.....	8
2. HoneyPot.....	9
2.1 ¿Qué es un HoneyPot?.....	9
2.2 Configuración.....	9
2.3 Instalación servicio web.....	11
2.4 ¿Qué es DVWA?.....	12
2.5 Instalación DVWA.....	12
2.6 Configuración DVWA.....	13
3. Suricata.....	16
3.1 ¿Qué es suricata?.....	16
3.2 ¿Como funciona suricata?.....	16
3.3 Instalación de Suricata.....	17
3.4 Configuración de Suricata.....	19
3.5 Estructura de las reglas.....	21
3.6 Desarrollo de un bot de alertas.....	24
3.6.1 Creación del bot.....	24
3.6.2 Desarrollo del programa en python.....	25
4. Splunk.....	29
4.1 ¿Qué es Splunk?.....	29
4.2 Ventajas de Splunk.....	29
4.3 Instalación de Splunk.....	29
4.4 Configuración de Splunk.....	34
4.4.1 Instalación de Stamus Network.....	34
4.4.2 Configuración de puertos.....	35
4.4.3 Splunk Forwarder.....	39
5. Kali Linux.....	44
5.1 ¿Qué es Kali Linux?.....	44
5.2 Instalación de Kali Linux.....	44
6. Conclusiones.....	49
7. Glosario.....	50
8. Webgrafía.....	51

## Índice de figuras

Ilustración 1: Estructura del proyecto.....	7
Ilustración 2: Cronograma.....	8
Ilustración 3: Interfaz de red.....	9
Ilustración 4: Actualizar repositorios.....	10
Ilustración 5: Instalación apache2.....	11
Ilustración 6: Comprobación servicio.....	11
Ilustración 7: Descarga DVWA.....	12
Ilustración 8: config.inc.php.....	13
Ilustración 9: php.ini.....	14
Ilustración 10: DVWA setup.....	15
Ilustración 11: DVWA Inicio.....	15
Ilustración 12: Suricata repositorio.....	17
Ilustración 13: Suricata instalación.....	18
Ilustración 14: comprobación instalación.....	18
Ilustración 15: Directorio Suricata.....	18
Ilustración 16: <i>Directorio</i> logs.....	19
Ilustración 17: Suricata Logs.....	19
Ilustración 18: Elegir interfaz de red.....	20
Ilustración 19: Redes Suricata.....	20
Ilustración 20: Directorio de reglas.....	21
Ilustración 21: Reglas my.rules.....	21
Ilustración 22: Indicar reglas.....	21
Ilustración 23: regla para PING.....	22
Ilustración 24: Estado Suricata.....	23
Ilustración 25: Alerta en fast.log.....	23
Ilustración 26: Ayuda BotFather.....	24
Ilustración 27: Comando /newbot.....	24
Ilustración 28: Bot creado.....	25
Ilustración 29: Suricata imagen.....	25
Ilustración 30: Línea 1 a 7 del <i>programa</i> .....	26
Ilustración 31: Línea 15 a 33 del programa.....	26
Ilustración 32: Línea 36 a 58.....	27
Ilustración 33: Versión python.....	28
Ilustración 34: Alerta en el bot.....	28
Ilustración 35: Web de Splunk.....	30
Ilustración 36: Splunk Enterprise.....	30
Ilustración 37: Descarga distribuciones Linux.....	31
Ilustración 38: Enlace Wget de Splunk.....	31
Ilustración 39: Descarga del instalador de Splunk.....	31
Ilustración 40: Creación de usuario.....	33
Ilustración 41: Inicio de Splunk.....	33
Ilustración 42: Inicio de sesión de Splunk.....	34
Ilustración 43: Página principal de Splunk.....	34
Ilustración 44: Instalación de Stamus Network.....	35
Ilustración 45: Forward and receiveing.....	35
Ilustración 46: Configurar recibir datos.....	36
Ilustración 47: Puerto 9997.....	36
Ilustración 48: Data inputs.....	37

Ilustración 49: Data input UDP.....	37
Ilustración 50: configuración TCP/UDP.....	38
Ilustración 51: Inputs Settings.....	38
Ilustración 52: Confirmación de configuración.....	39
Ilustración 53: Splunk Universal Forwarder.....	40
Ilustración 54: Página web Universal Forwarder.....	40
Ilustración 55: Descarga del forwarder.....	41
Ilustración 56: Instalación Universal Forwarder.....	41
Ilustración 57: Añadir servidor para enviar datos.....	42
Ilustración 58: Suricata IDS Dashboard.....	42
Ilustración 59: Descarga Kali Linux.....	44
Ilustración 60: Escaneo con nmap.....	45
Ilustración 61: inyeccion SQL.....	46
Ilustración 62: subida de reverse shell.....	46
Ilustración 63: Detección de ejecución de comandos.....	47
Ilustración 64: Contenido reverse shell.....	47
Ilustración 65: Resultados en Splunk.....	48

## FICHA DEL PROYECTO FINAL

<b>Título del trabajo:</b>	Implementación de un sistema IDS/SIEM
<b>Nombre del autor:</b>	Pablo Jiménez Márquez
<b>Fecha de entrega (mm/aaaa):</b>	13/06/2023
<b>Área del Trabajo Final:</b>	Ciberseguridad
<b>Ciclo Grado Superior:</b>	Administración de Sistemas Informáticos en Red
<b>Resumen del Trabajo (máximo 250 palabras):</b>	
<p>A continuación, se procede a diseñar la arquitectura del sistema IDS/SIEM. Esto implica la selección de los dispositivos y herramientas adecuados que permitirán la detección de intrusiones y la recopilación de registros de eventos. Además, se definen las políticas de seguridad y se establecen los umbrales de alerta para notificar y responder a posibles incidentes de seguridad.</p> <p>Una vez diseñada la arquitectura, se procede a la implementación del sistema. Esto incluye la instalación y configuración de los sensores IDS en los diferentes puntos de la red, así como la integración con los sistemas existentes para la recopilación de registros de eventos. Se establecen las reglas y firmas necesarias para la detección de actividades sospechosas o maliciosas.</p> <p>Además del monitoreo en tiempo real, se realizan análisis periódicos de los registros almacenados para identificar patrones de actividad y posibles amenazas persistentes. Se generan informes de seguridad que proporcionan información detallada sobre los incidentes, las tendencias de seguridad y las recomendaciones de mejora.</p> <p>Finalmente, se establece un plan de mantenimiento y actualización continua del sistema IDS/SIEM para garantizar su efectividad a largo plazo. Se realizan revisiones periódicas de la arquitectura y se actualizan las reglas y firmas de detección según las nuevas amenazas identificadas.</p>	

# 1. Introducción

## 1.1 Justificación

La ciberseguridad es de vital importancia en una organización debido al creciente riesgo y la sofisticación de las amenazas cibernéticas en la actualidad. A medida que las organizaciones dependen cada vez más de la tecnología y la información digital, la protección de los activos y la preservación de la confidencialidad, integridad y disponibilidad de los datos son cruciales ya que un ciberataque puede provocar robo de datos provocando grandes daños a la empresa.

La implementación de una infraestructura con un HoneyPot es de vital importancia para el y ya que atraerá diversos tipos de ataques para evitar posibles ataques a otros servidores que se quieran proteger y prevenirlos de estos ataques.

La implementación de un sistema IDS(Sistema de Detección de Intrusiones) y SIEM(Gestión de Eventos e Información de Seguridad) se ha desarrollado con el propósito de analizar el tráfico malicioso de una red, para poder identificar a tiempo un potencial ataque y para estudiar los métodos que un ciberatacante puede emplear para vulnerar los diferentes servicios en la infraestructura de una organización.

## 1.2 Objetivos

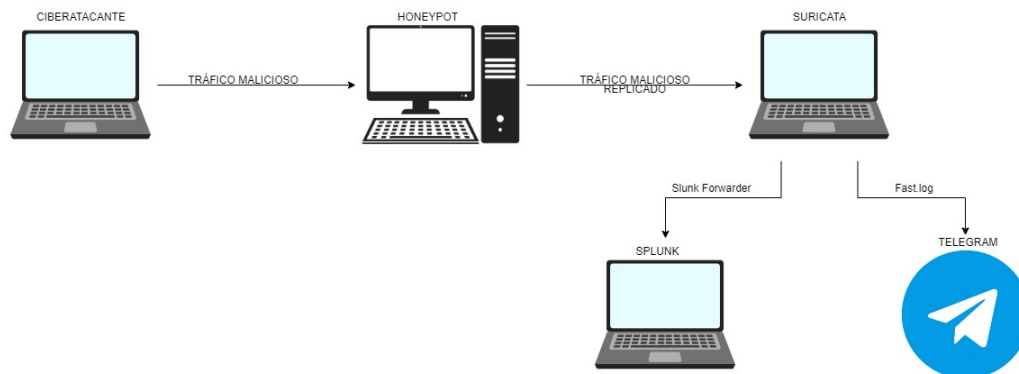
Se numerarán los objetivos a cumplir en este proyecto:

- Configuración de un **honeypot** de baja interacción con varios servicios.
- Instalación y configuración de **Suricata** como **IDS**.
- Instalación y configuración de **Splunk** como **SIEM**.
- Desarrollo de un bot en Telegram para la visualización en tiempo real de las alertas.
- Comprometer el **honeypot** para el estudio de diferentes tipos de **ciberataques**.
- Identificar los ataques y prevenirlos.
- Detener un ataque a tiempo y minimizar los daños.

## 1.3 Estructura

El laboratorio que crearemos en VirtualBox simulando la red de una organización en la que desplegaremos un servicio web visible a internet con la intención de que reciba ataques para analizarlos. El laboratorio tendrá las siguientes máquinas virtuales:

- Ciberatacante: Utilizaremos **Kali Linux** para realizar distintos tipos de ataques hacia el **HoneyPot**.
- HoneyPot: Máquina virtual con **Ubuntu Server** con un servicio web a la que dirigiremos los ataques para analizarlos.
- Suricata: En este equipo instalaremos **suricata** para analizar en tráfico replicado del HoneyPot y encontrar posibles amenazas.
- Splunk: Analizaremos todos los ataques detectados por suricata con este programa para **monitorear** y **analizar** grandes cantidades de datos.
- Telegram: Desarrollaremos un programa en **python** para enviar las alertas del registro **fast.log** a un **bot** de Telegram en tiempo real.



*Ilustración 1: Estructura del proyecto*

## 1.4 Planificación

Las tareas que se desarrollarán en el proyecto son las siguientes:

### 1. Instalación de un servidor web con apache2:

Crearemos una página web con diversas vulnerabilidades que posteriormente serán explotadas por el ciberatacante. Utilizaremos **DVWA**(Damn Vulnerable Web App), es una aplicación virtual con vulnerabilidades usadas por hackers para entender como funcionan los ataques y aprender a identificar fallos.

### 2. Instalación de IDS Suricata:

En una máquina virtual Debian instalaremos y configuraremos Suricata para la detección de ataques con las reglas necesarias para detectar los ataques. En esta máquina virtual se encontrará el programa en python con el que se enviarán las alertas a Telegram.

### 3. Instalación de Splunk:

En una máquina virtual Debian instalaremos y configuraremos el SIEM Splunk para que reciba las alertas de Suricata y analizarlas.

### 4. Entorno del ciberatacante:

Descargaremos una máquina Kali Linux y le implementaremos algunas herramientas para el pentesting en el Honeypot.

### Cronograma:

Semana / Actividad	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Seman 7	Seman 8	Seman 9	Seman 10
Actividad 1										
Actividad 2										
Actividad 3										
Actividad 4										

*Ilustración 2: Cronograma*

## 1.5 Presupuesto

La realización del proyecto se realizará en un laboratorio con VirtualBox en el que instalaremos y configuraremos las máquinas necesarias para el servidor, suricata y splunk.

Para el funcionamiento de este laboratorio solo será necesario un equipo con el hardware para desplegar las máquinas virtuales.

Para crear la infraestructura que se desarrollará en el proyecto necesitaremos equipos con ciertas características para que puedan desempeñar el trabajo correctamente:

- Para el HoneyPot necesitaremos un buen equipo ya que montaremos un servidor y desplegaremos servicios que consumirán recursos, ya que contaremos con buenos componentes el coste de este equipo es de 1000€. El precio también depende del tipo de servicio que queremos desplegar.
- Para el equipo que contendrá el software de Splunk es necesario que disponga de bastante espacio de almacenamiento ya que se estará guardando constantemente información recibida por Suricata, el coste de este equipo es de 800€. También se



contempla la instalación de un NAS para el guardado de datos y de copias de seguridad pero esto también conlleva un incremento del precio.

- Por último el equipo con Suricata no debe de ser un equipo demasiado potente a no ser que se quieran implementar en el más sistemas de seguridad. El coste es de unos 400€. La creación de un bot en Telegram es totalmente gratis y no son necesarias licencias.
- El presupuesto total es de 2.200€ aproximadamente.

## 2. HoneyPot

### 2.1 ¿Qué es un HoneyPot?

Es un sistema informático que se “sacrifica” para atraer ciberataques, como un señuelo. Simula ser un objetivo para los hackers y utiliza sus intentos de intrusión para obtener información sobre los cibercriminales y la forma en que operan o para distraerlos de otros objetivos.

En este caso instalaremos una aplicación web(**DVWA**) con varias vulnerabilidades para ser explotadas, este honeypot será de tipo interactivo ya que nos permitirá entrar al servicio y lanzar peticiones.

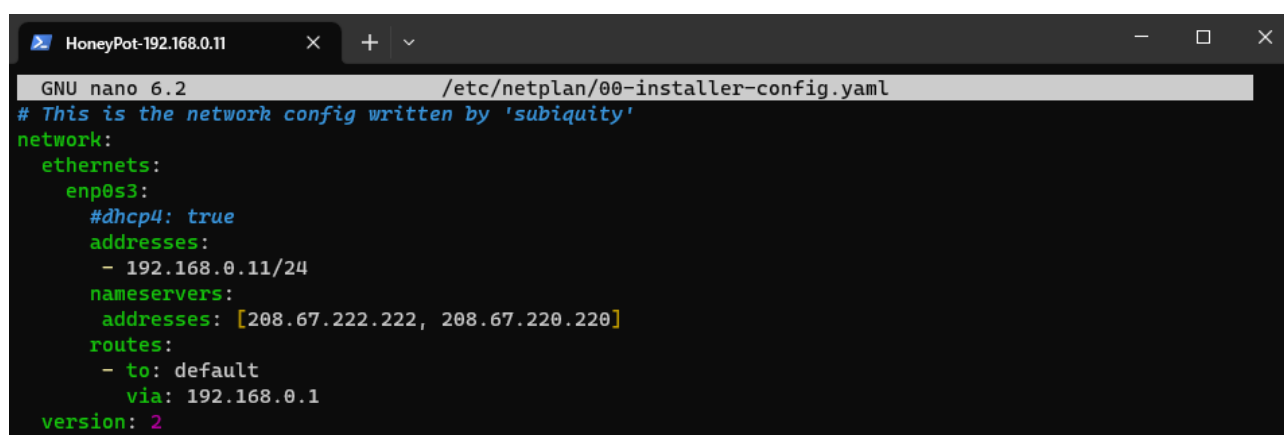
### 2.2 Configuración

Para esta máquina virtual hemos escogido un Ubuntu Server con las siguientes características:

- 2 Núcleos en la CPU.
- 1GB de RAM.
- 10GB de disco duro.
- Interfaz de red en adaptador puente.

Comenzaremos con la configuración del adaptador de red dentro de la máquina virtual:

```
sudo nano /etc/netplan/00-installer-config.yaml
```



```
GNU nano 6.2 /etc/netplan/00-installer-config.yaml
# This is the network config written by 'subiquity'
network:
  ethernets:
    enp0s3:
      #dhcp4: true
      addresses:
        - 192.168.0.11/24
      nameservers:
        addresses: [208.67.222.222, 208.67.220.220]
      routes:
        - to: default
          via: 192.168.0.1
  version: 2
```

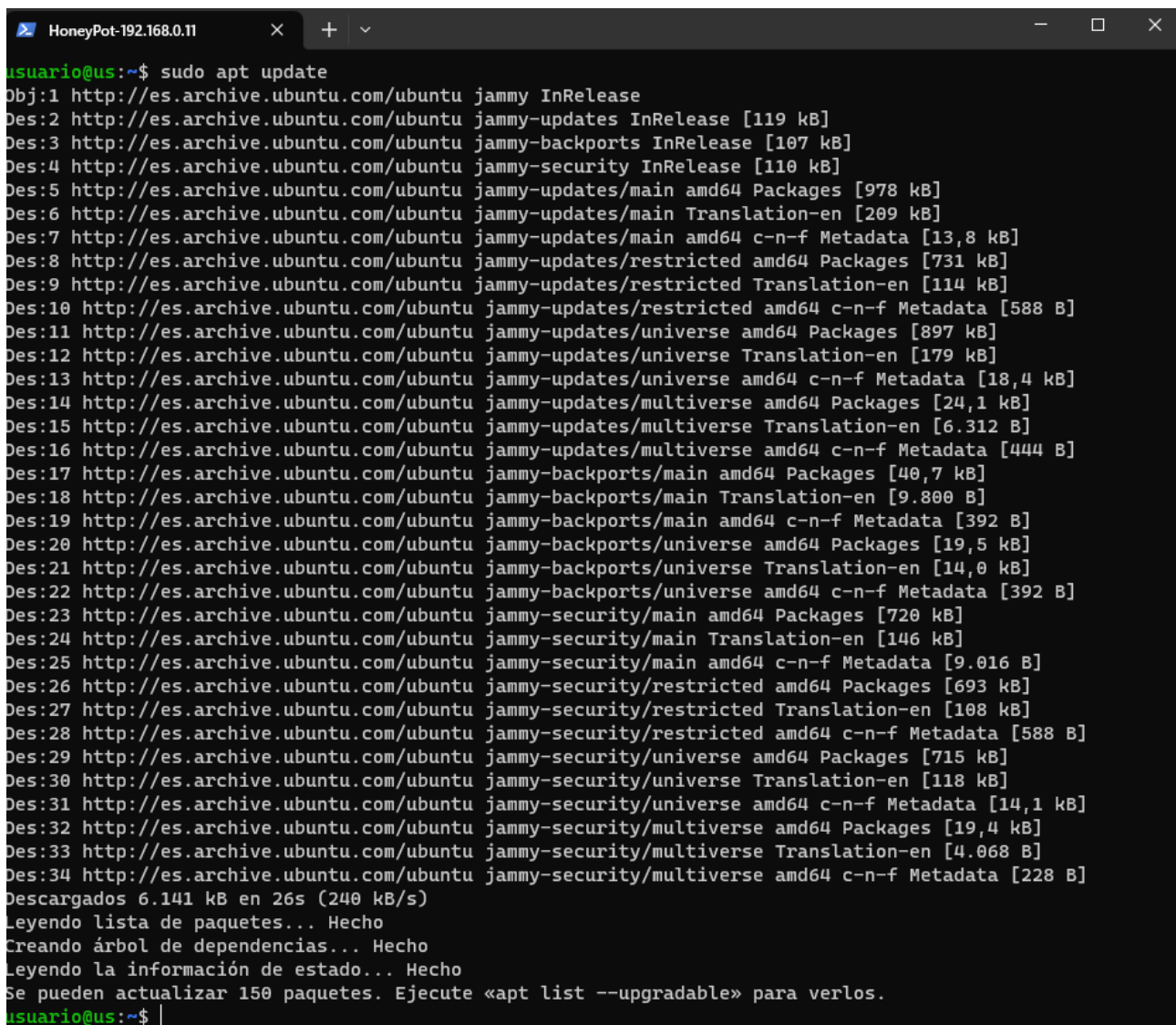
Ilustración 3: Interfaz de red

La IP de este equipo es la 192.168.0.11 y la **puerta de enlace** es la 192.168.0.1

Actualizaremos los repositorios y posteriormente actualizamos el equipo con upgrade

```
sudo apt update
```

```
sudo apt upgrade
```



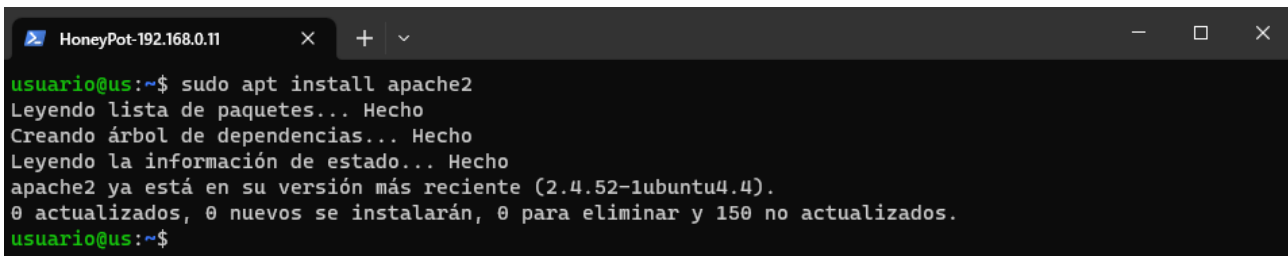
```
HoneyPot-192.168.0.11
usuario@us:~$ sudo apt update
Obj:1 http://es.archive.ubuntu.com/ubuntu jammy InRelease
Des:2 http://es.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Des:3 http://es.archive.ubuntu.com/ubuntu jammy-backports InRelease [107 kB]
Des:4 http://es.archive.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Des:5 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [978 kB]
Des:6 http://es.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [209 kB]
Des:7 http://es.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [13,8 kB]
Des:8 http://es.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [731 kB]
Des:9 http://es.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [114 kB]
Des:10 http://es.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 c-n-f Metadata [588 B]
Des:11 http://es.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [897 kB]
Des:12 http://es.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [179 kB]
Des:13 http://es.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 c-n-f Metadata [18,4 kB]
Des:14 http://es.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 Packages [24,1 kB]
Des:15 http://es.archive.ubuntu.com/ubuntu jammy-updates/multiverse Translation-en [6.312 B]
Des:16 http://es.archive.ubuntu.com/ubuntu jammy-updates/multiverse amd64 c-n-f Metadata [444 B]
Des:17 http://es.archive.ubuntu.com/ubuntu jammy-backports/main amd64 Packages [40,7 kB]
Des:18 http://es.archive.ubuntu.com/ubuntu jammy-backports/main Translation-en [9.800 B]
Des:19 http://es.archive.ubuntu.com/ubuntu jammy-backports/main amd64 c-n-f Metadata [392 B]
Des:20 http://es.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [19,5 kB]
Des:21 http://es.archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [14,0 kB]
Des:22 http://es.archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [392 B]
Des:23 http://es.archive.ubuntu.com/ubuntu jammy-security/main amd64 Packages [720 kB]
Des:24 http://es.archive.ubuntu.com/ubuntu jammy-security/main Translation-en [146 kB]
Des:25 http://es.archive.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [9.016 B]
Des:26 http://es.archive.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [693 kB]
Des:27 http://es.archive.ubuntu.com/ubuntu jammy-security/restricted Translation-en [108 kB]
Des:28 http://es.archive.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [588 B]
Des:29 http://es.archive.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [715 kB]
Des:30 http://es.archive.ubuntu.com/ubuntu jammy-security/universe Translation-en [118 kB]
Des:31 http://es.archive.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [14,1 kB]
Des:32 http://es.archive.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [19,4 kB]
Des:33 http://es.archive.ubuntu.com/ubuntu jammy-security/multiverse Translation-en [4.068 B]
Des:34 http://es.archive.ubuntu.com/ubuntu jammy-security/multiverse amd64 c-n-f Metadata [228 B]
Descargados 6.141 kB en 26s (240 kB/s)
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se pueden actualizar 150 paquetes. Ejecute «apt list --upgradable» para verlos.
usuario@us:~$ |
```

*Ilustración 4: Actualizar repositorios*

## 2.3 Instalación servicio web.

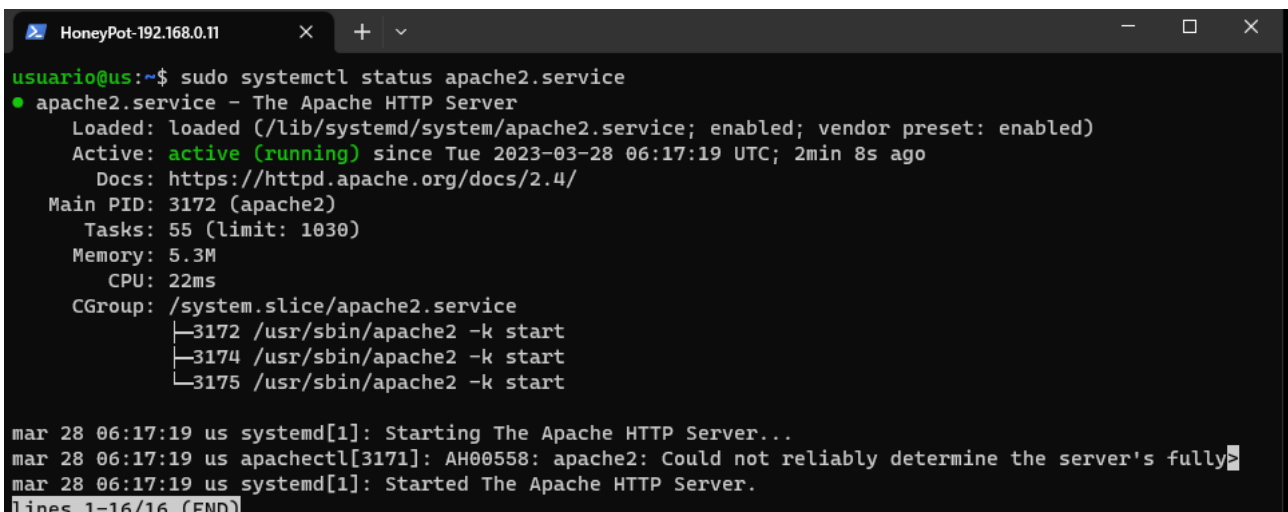
Después de hacer las configuraciones anteriores ya podremos instalar apache2, este será el servicio con el que accederemos desde el atacante.

```
sudo apt install apache2
```



```
HoneyPot-192.168.0.11
usuario@us:~$ sudo apt install apache2
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
apache2 ya está en su versión más reciente (2.4.52-1ubuntu4.4).
0 actualizados, 0 nuevos se instalarán, 0 para eliminar y 150 no actualizados.
usuario@us:~$
```

*Ilustración 5: Instalación apache2*



```
HoneyPot-192.168.0.11
usuario@us:~$ sudo systemctl status apache2.service
● apache2.service - The Apache HTTP Server
   Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2023-03-28 06:17:19 UTC; 2min 8s ago
     Docs: https://httpd.apache.org/docs/2.4/
   Main PID: 3172 (apache2)
    Tasks: 55 (limit: 1030)
   Memory: 5.3M
      CPU: 22ms
   CGroup: /system.slice/apache2.service
           └─3172 /usr/sbin/apache2 -k start
             └─3174 /usr/sbin/apache2 -k start
               └─3175 /usr/sbin/apache2 -k start

mar 28 06:17:19 us systemd[1]: Starting The Apache HTTP Server...
mar 28 06:17:19 us apachectl[3171]: AH00558: apache2: Could not reliably determine the server's fully
mar 28 06:17:19 us systemd[1]: Started The Apache HTTP Server.
lines 1-16/16 (END)
```

*Ilustración 6: Comprobación servicio*

Comenzaremos con la instalación de la aplicación web donde se encontraran las vulnerabilidades.

## 2.4 ¿Qué es DVWA?

Damn Vulnerable Web App (DVWA) es una aplicación virtual vulnerable a los ciberataques web más conocidos y utilizados por hackers. El propósito de esta aplicación es entender cómo funcionan estos ataques y, además, aprender a identificar fallos de seguridad en

otros sitios web. No obstante, DVWA es un entorno legal y seguro, en donde puedes practicar estas técnicas todo lo que desees.

Damn Vulnerable Web App (DVWA) está basada en PHP y MySQL y cuenta con páginas que son vulnerables a ataques como, entre otros:

- Fuerza bruta de una vulnerable web.
- Inyección de comandos.
- Subida de ficheros de la vulnerable web.
- Inyección SQL de la vulnerable web.

## 2.5 Instalación DVWA

Para la instalación de la aplicación web tendremos que descargarla de github con:

```
git clone https://github.com/ethicalhack3r/DVWA
```

El directorio debemos de descargarlo en **/var/www/html/**

A terminal window with a dark background. The prompt is 'root@HoneyPot: /var/www/ht'. The user has entered 'ls' and the output shows 'admin', 'admin.index', 'config', 'index.html'. The 'config' file is highlighted in green. The prompt is now 'root@HoneyPot: /var/www/html#'.

```
root@HoneyPot: /var/www/ht  X  +  -  X
root@HoneyPot: /var/www/html# ls
admin  admin.index  config  index.html
root@HoneyPot: /var/www/html#
```

*Ilustración 7: Descarga DVWA*

Daremos todos los permisos al directorio

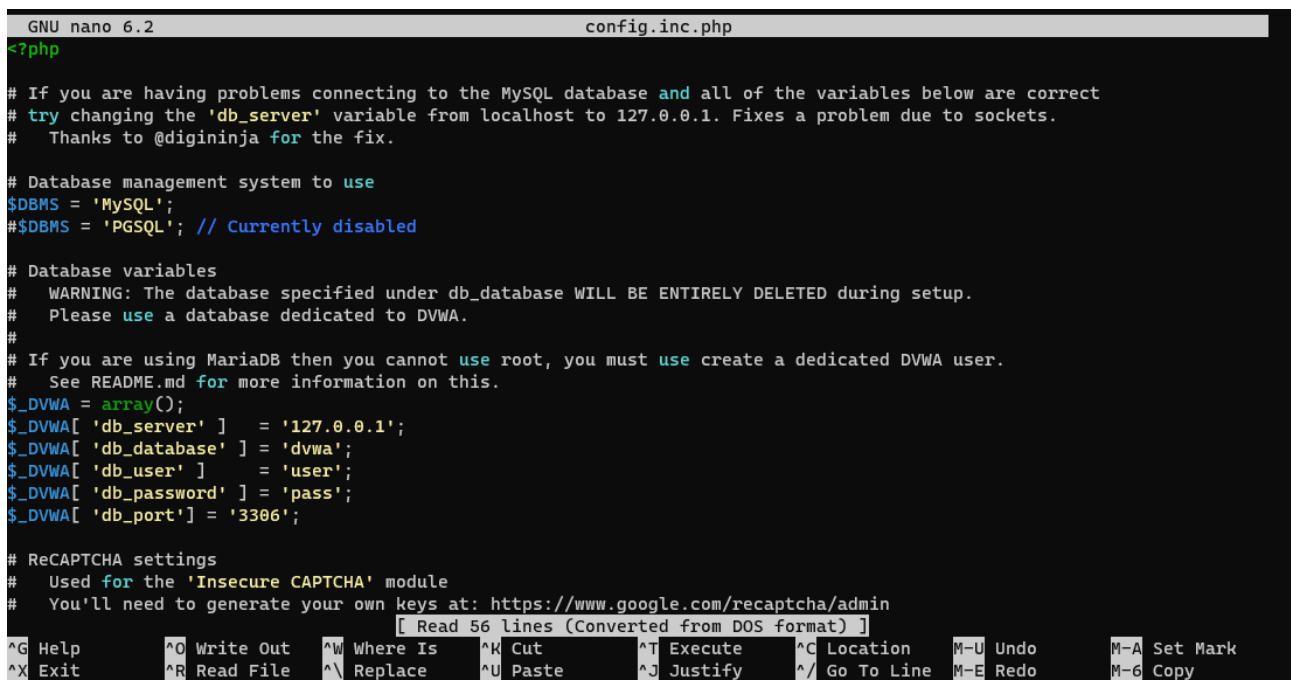
```
sudo chmod -R 777 DVWA
```

También deberemos de descargar los siguientes paquetes:

```
apt-get -y install mysql-server php php-mysqli php-gd
```

## 2.6 Configuración DVWA

En `/var/www/html/config/config.inc.php` encontraremos la configuración de la base de datos de la aplicación.



```
GNU nano 6.2 config.inc.php
<?php

# If you are having problems connecting to the MySQL database and all of the variables below are correct
# try changing the 'db_server' variable from localhost to 127.0.0.1. Fixes a problem due to sockets.
# Thanks to @digininja for the fix.

# Database management system to use
$DBMS = 'MySQL';
#$DBMS = 'PGSQL'; // Currently disabled

# Database variables
# WARNING: The database specified under db_database WILL BE ENTIRELY DELETED during setup.
# Please use a database dedicated to DVWA.
#
# If you are using MariaDB then you cannot use root, you must use create a dedicated DVWA user.
# See README.md for more information on this.
$_DVWA = array();
$_DVWA[ 'db_server' ] = '127.0.0.1';
$_DVWA[ 'db_database' ] = 'dvwa';
$_DVWA[ 'db_user' ] = 'user';
$_DVWA[ 'db_password' ] = 'pass';
$_DVWA[ 'db_port' ] = '3306';

# ReCAPTCHA settings
# Used for the 'Insecure CAPTCHA' module
# You'll need to generate your own keys at: https://www.google.com/recaptcha/admin
[ Read 56 lines (Converted from DOS format) ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo      M-A Set Mark
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^_ Go To Line  M-E Redo      M-6 Copy
```

Ilustración 8: `config.inc.php`

En mi caso el nombre de la base de datos es **dvwa**, el usuario es **user** y la contraseña **pass**.

Ahora crearemos la BD en la consola de mysql con las siguientes sentencias:

```
create database dvwa;
```

Creamos el usuario y establecemos contraseña:

```
create user 'user'@'127.0.0.1' identified by 'password';
```

Damos privilegios al usuario.

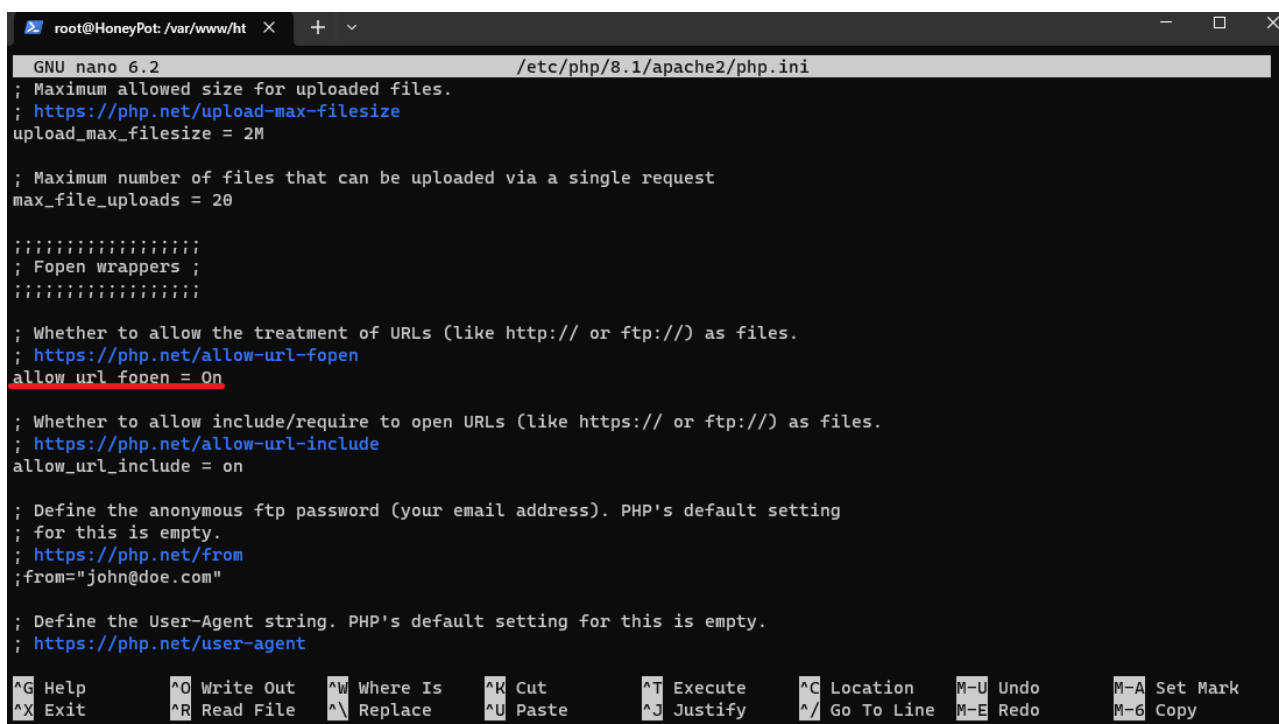
```
grant all privileges on dwwa.* to 'user'@'127.0.0.1';
```

Reiniciamos los servicios de apache2 y mysql para que se apliquen los cambios.

```
sudo systemctl restart apache2
```

```
sudo systemctl restart mysql
```

Por último deberemos de cambiar un parámetro en la configuración de apache2 y lo reiniciamos.



```
root@HoneyPot: /var/www/ht x + v
GNU nano 6.2 /etc/php/8.1/apache2/php.ini
; Maximum allowed size for uploaded files.
; https://php.net/upload-max-filesize
upload_max_filesize = 2M

; Maximum number of files that can be uploaded via a single request
max_file_uploads = 20

; Fopen wrappers
; Whether to allow the treatment of URLs (like http:// or ftp://) as files.
; https://php.net/allow-url-fopen
allow_url_fopen = On

; Whether to allow include/require to open URLs (like https:// or ftp://) as files.
; https://php.net/allow-url-include
allow_url_include = on

; Define the anonymous ftp password (your email address). PHP's default setting
; for this is empty.
; https://php.net/from
;from="john@doe.com"

; Define the User-Agent string. PHP's default setting for this is empty.
; https://php.net/user-agent

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

*Ilustración 9: php.ini*

Ahora seleccionamos Create / Reset Database y si todo sale bien podremos logearnos con el usuario de la BD.

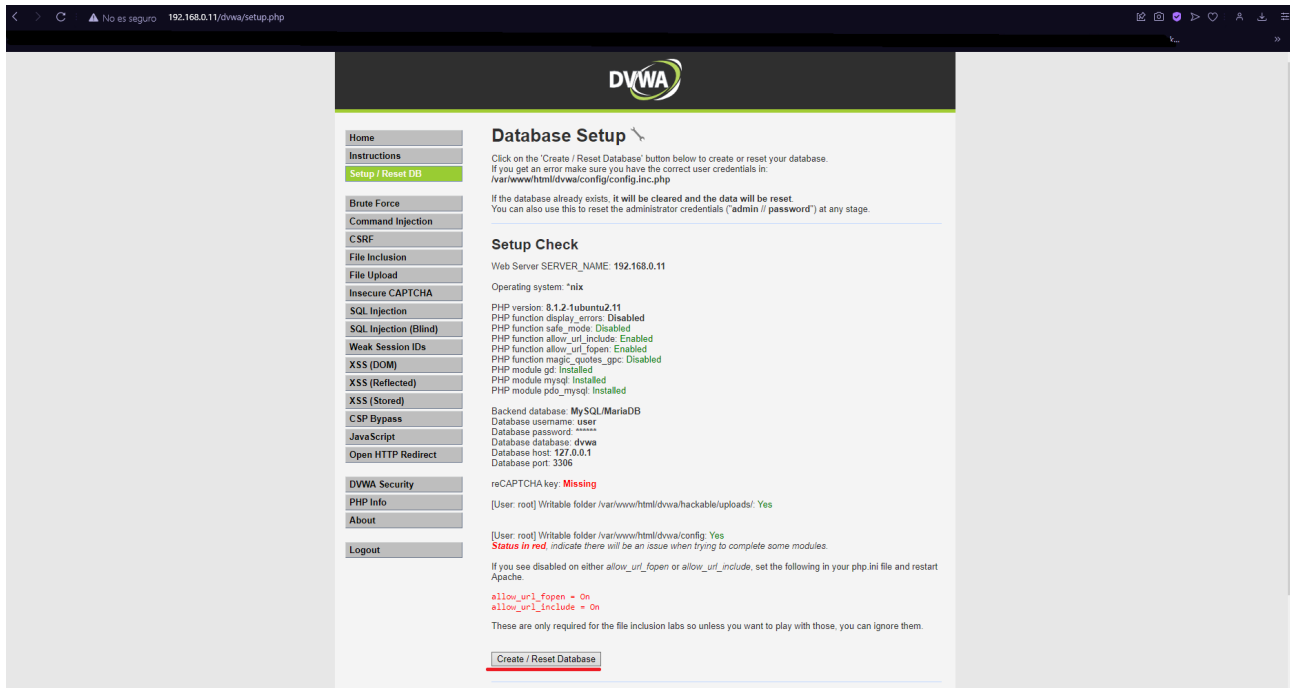


Ilustración 10: DVWA setup

Ya está configurada la aplicación web que será atacada con Kali Linux.

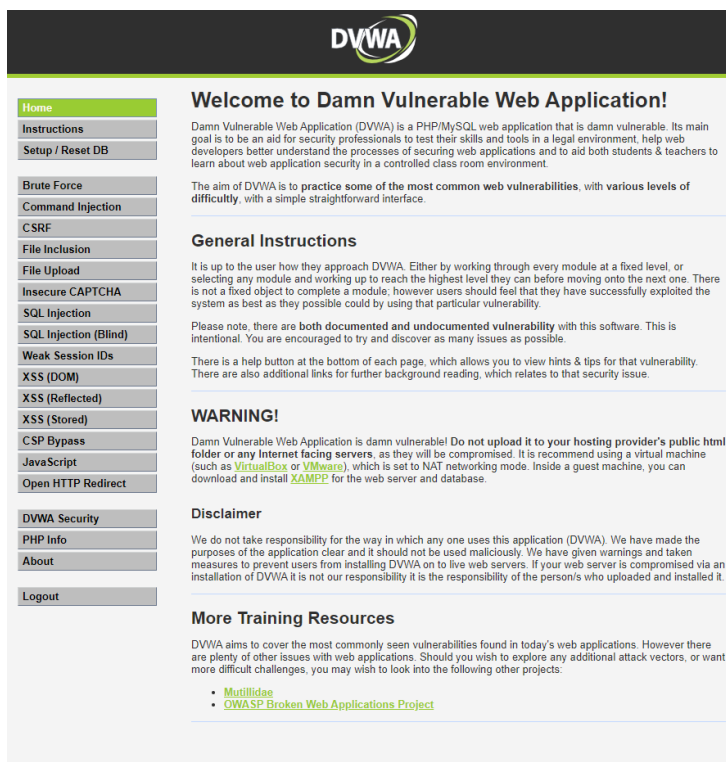


Ilustración 11: DVWA Inicio

## 3. Suricata

### 3.1 ¿Qué es suricata?

Suricata es una herramienta de código abierto para ciberseguridad, que sirve para monitorizar el tráfico de red de un sistema y es capaz de detectar comportamientos maliciosos. Suricata fue desarrollada y actualmente es mantenida por OSIF (Open Information Security Foundation), una organización sin ánimos de lucro. El programa puede ser descargado en la página de su repositorio de GitHub.

#### Modos de operación de Suricata:

- **Modo pasivo (IDS):** Se utiliza para detectar y almacenar información sobre el tráfico de red sea marcado como malicioso.
- **Modo activo (IPS):** Se utiliza para para detectar y almacenar información sobre el tráfico de manera aislada, para prevenir intrusiones en línea que pongan en riesgo al sistema.

En este caso lo utilizaremos como IDS o pasivo ya que Suricata solo detectará e informará de las amenazas y no denegará peticiones ni hará nada para impedir los ataques.

### 3.2 ¿Como funciona suricata?

Los análisis de tráfico de red de Suricata se basan en una serie de reglas predeterminadas que están diseñadas especialmente para identificar **malware** y actividades malignas en las comunicaciones de un sistema con internet. De esta forma, es posible, por ejemplo, detectar si un malware se está conectando a una **botnet** o si está descargando otro programa de algún **servidor**.

#### Suricata VS Snort:

Ya hemos visto qué es **Suricata** en **ciberseguridad** y también he mencionado una herramienta llamada **Snort**. Ambos programas son de código abierto y deben estar en el arsenal de cualquier miembro del **Blue Team**. Tanto **Suricata** como **Snort** funcionan con reglas aportadas por la comunidad y, por ese motivo, se complementan.

También es necesario tener en cuenta que **Snort** tiene una versión limitada y una versión de pago. Por esa razón, si se utiliza la versión gratuita, quizás se quiera complementar con otros **softwares**, como **Suricata**.



### 3.3 Instalación de Suricata

En una máquina virtual con Debian en VirtualBox comenzaremos a descargar Suricata, esta VM tendrá las siguientes especificaciones:

- 2GB de RAM.
- 3 Núcleos de CPU.
- 30GB de disco duro.
- Interfaz de red en adaptador puente.

Primero deberemos actualizar los repositorios y añadir el repositorio de Suricata.

```
sudo apt update
```

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

```
root@suricata:/home/usuario# add-apt-repository ppa:oisf/suricata-stable
Repository: 'deb https://ppa.launchpadcontent.net/oisf/suricata-stable/ubuntu/ jammy main'
Description:
Suricata IDS/IPS/NSM stable packages
https://suricata.io/
https://oisf.net/

Suricata IDS/IPS/NSM - Suricata is a high performance Intrusion Detection and Prevention System and Network Security Monitoring engine.

Open Source and owned by a community run non-profit foundation, the Open Information Security Foundation (OISF). Suricata is developed by the OISF, its supporting vendors and the community.

This Engine supports:

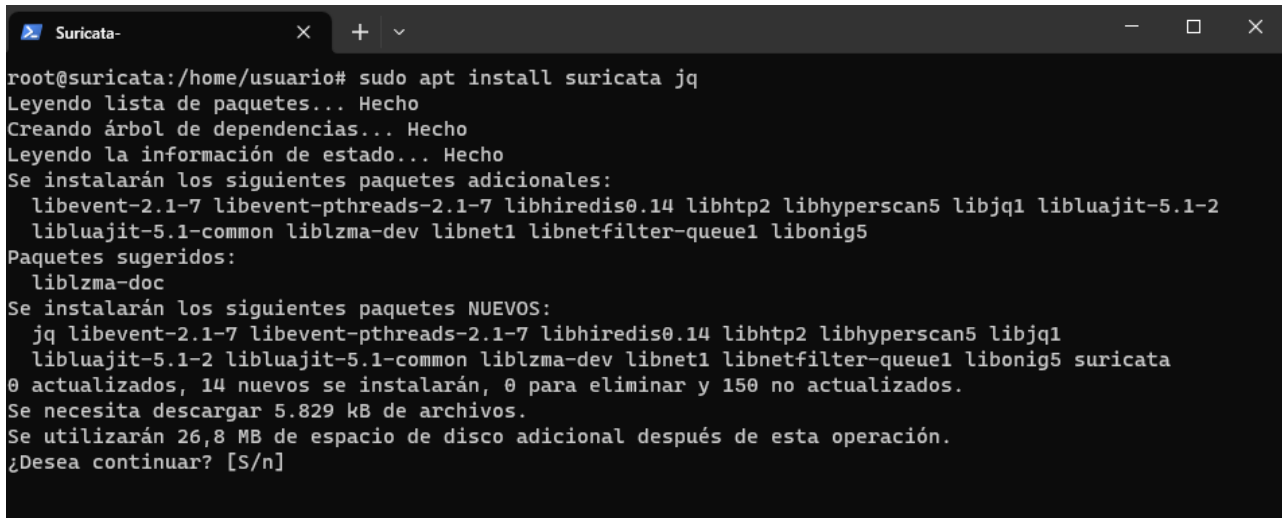
- Multi-Threading - provides for extremely fast and flexible operation on multicore systems.
- Multi Tenancy - Per vlan/Per interface
- Uses Rust for most protocol detection/parsing
- TLS/SSL certificate matching/logging
- JA3 TLS client fingerprinting
- JA3S TLS server fingerprinting
- IEEE 802.1ad (QinQ) and IEEE 802.1Q (VLAN) support
- VXLAN support
- All JSON output/logging capability
- IDS runmode
- IPS runmode
- IDPS runmode
- NSM runmode
- eBPF/XDP
- Automatic Protocol Detection and logging - IPv4/6, TCP, UDP, ICMP, HTTP, SMTP, TLS, SSH, FTP, SMB, DNS, NFS, TFTP, KRBS, DHCP, IKEv2, SNMP, SIP, RDP
- SCADA automatic protocol detection - ENIP/DNP3/MODBUS
- File Extraction HTTP/SMTP/FTP/NFS/SMB - over 4000 file types recognized and extracted from live traffic.
- File MD5/SHA1/SHA256 matching
- Gzip Decompression
- Fast IP Matching
- Datasets matching
- Rustlang enabled protocol detection
- Lua scripting

and many more great features -
https://suricata.io/features/all-features/
More info: https://launchpad.net/~oisf/+archive/ubuntu/suricata-stable
Adding repository.
Press [ENTER] to continue or Ctrl-C to cancel.
```

*Ilustración 12: Suricata repositorio*

Cuando lo añadamos presionamos enter y ejecutamos el siguiente comando para que comience la instalación.

```
sudo apt install suricata jq
```

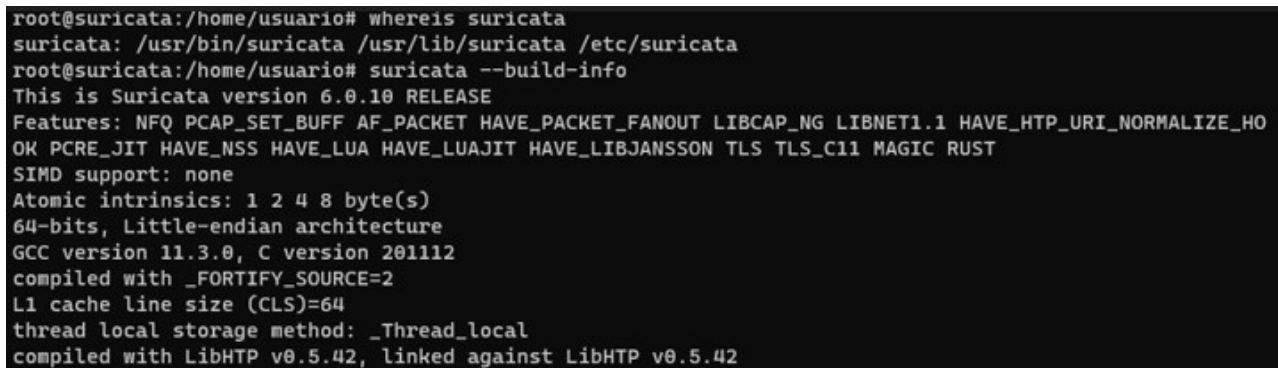


```
root@suricata:/home/usuario# sudo apt install suricata jq
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Se instalarán los siguientes paquetes adicionales:
 libevent-2.1-7 libevent-pthreads-2.1-7 libhiredis0.14 libhttp2 libhyperscan5 libjq1 liblua5.1-2
 liblua5.1-common liblzma-dev libnet1 libnetfilter-queue1 libonig5
Paquetes sugeridos:
 liblzma-doc
Se instalarán los siguientes paquetes NUEVOS:
 jq libevent-2.1-7 libevent-pthreads-2.1-7 libhiredis0.14 libhttp2 libhyperscan5 libjq1
 liblua5.1-2 liblua5.1-common liblzma-dev libnet1 libnetfilter-queue1 libonig5 suricata
0 actualizados, 14 nuevos se instalarán, 0 para eliminar y 150 no actualizados.
Se necesita descargar 5.829 kB de archivos.
Se utilizarán 26,8 MB de espacio de disco adicional después de esta operación.
¿Desea continuar? [S/n]
```

*Ilustración 13: Suricata instalación*

Cuando se haya instalado buscamos en que directorio se ha instalado con el comando:

```
whereis suricata
```

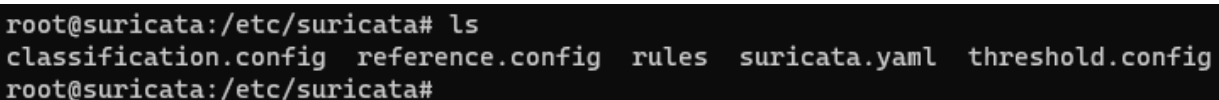


```
root@suricata:/home/usuario# whereis suricata
suricata: /usr/bin/suricata /usr/lib/suricata /etc/suricata
root@suricata:/home/usuario# suricata --build-info
This is Suricata version 6.0.10 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG LIBNET1.1 HAVE_HTTP_URI_NORMALIZE_HO
OK PCRE_JIT HAVE_NSS HAVE_LUA HAVE_LUAJIT HAVE_LIBJANSSON TLS TLS_C11 MAGIC RUST
SIMD support: none
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 11.3.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTTP v0.5.42, linked against LibHTTP v0.5.42
```

*Ilustración 14: comprobación instalación*

Podemos ver que se ha instalado en el directorio /etc/suricata donde se encuentran los archivos de configuración.

El archivo que editaremos es **suricata.yaml**:



```
root@suricata:/etc/suricata# ls
classification.config reference.config rules suricata.yaml threshold.config
root@suricata:/etc/suricata#
```

*Ilustración 15: Directorio Suricata*

En este archivo nos indicará donde se encuentran los registros donde se mostrarán las alertas. Dentro de este directorio encontraremos:

- **fast.log:** El archivo fast.log contiene información detallada sobre los eventos capturados por Suricata en tiempo real. Estos eventos pueden incluir alertas de intrusiones, intentos de escaneo de puertos, tráfico sospechoso, intentos de acceso no autorizado y otros comportamientos anómalos en la red.
- **eve.json:** El archivo eve.json contiene eventos detallados en formato JSON. Estos eventos pueden incluir alertas de intrusiones, intentos de escaneo de puertos, tráfico sospechoso, intentos de acceso no autorizado y otros comportamientos anómalos en la red. Este archivo lo usaremos más adelante para enviar los datos a nuestro SIEM.

```
# The default logging directory. Any log or output file will be
# placed here if it's not specified with a full path name. This can be
# overridden with the -l command line parameter.
default-log-dir: /var/log/suricata/
```

Ilustración 16: Directorio logs

```
root@suricata:/etc/suricata# cd /var/log/suricata/
root@suricata:/var/log/suricata# ls
eve.json          fast.log.1.gz      stats.log.4.gz
eve.json.1-2023052320.backup fast.log.3.gz      suricata.log
eve.json.1.gz     fast.log.4.gz      suricata.log.1-2023052320.backup
eve.json.3.gz     stats.log           suricata.log.1.gz
eve.json.4.gz     stats.log.1-2023052320.backup suricata.log.3.gz
fast.log          stats.log.1.gz     suricata.log.4.gz
fast.log.1-2023052320.backup stats.log.3.gz
root@suricata:/var/log/suricata#
```

Ilustración 17: Suricata Logs

### 3.4 Configuración de Suricata

Nos dirigiremos al directorio en que se encuentra el archivo de configuración **suricata.yaml** y buscaremos la línea **af-packet** donde indicaremos por cual interfaz de red queremos que escuche, en mi caso es el **enp0s3**.

```
GNU nano 6.2 /etc/suricata/suricata.yaml
##
## Step 3: Configure common capture settings
##
## See "Advanced Capture Options" below for more options, including Netmap
## and PF_RING.
##

# Linux high speed capture support
af-packet:
- interface: enp0s3
  # Number of receive buffers. "auto" uses the number of cores
```

Ilustración 18: Elegir interfaz de red

Si queremos indicarle la IP de un servidor o de la red que queremos que analice debemos indicarlo en las siguientes líneas:

```
GNU nano 5.4 suricata.yaml
%YAML 1.1
---
# Suricata configuration file. In addition to the comments describing all
# options in this file, full documentation can be found at:
# https://suricata.readthedocs.io/en/latest/configuration/suricata-yaml.html
##
## Step 1: Inform Suricata about your network
##

vars:
# more specific is better for alert accuracy and performance
address-groups:
  HOME_NET: "[192.168.0.0/16,10.0.0.0/8,172.16.0.0/12]"
  #HOME_NET: "[192.168.0.0/16]"
  #HOME_NET: "[10.0.0.0/8]"
  #HOME_NET: "[172.16.0.0/12]"
  #HOME_NET: "any"

  EXTERNAL_NET: "!$HOME_NET"
  #EXTERNAL_NET: "any"

  HTTP_SERVERS: "$HOME_NET"
  SMTP_SERVERS: "$HOME_NET"
  SQL_SERVERS: "$HOME_NET"
  DNS_SERVERS: "$HOME_NET"
  TELNET_SERVERS: "$HOME_NET"
  AIM_SERVERS: "$EXTERNAL_NET"
  DC_SERVERS: "$HOME_NET"
  DNP3_SERVER: "$HOME_NET"

[ 1885 líneas leídas ]
^G Ayuda      ^O Guardar    ^W Buscar     ^R Cortar     ^T Ejecutar   ^C Ubicación  M-U Deshacer  M-A Poner marca
^X Salir      ^R Leer fich. ^\ Reemplazar ^U Pegar      ^J Justificar ^_ Ir a línea  M-E Rehacer   M-6 Copiar
```

Ilustración 19: Redes Suricata

En mi caso lo dejaré como viene por defecto.

Ahora nos dirigiremos al directorio rules en **/etc/suricata/rules** donde crearemos nuestros archivos con extensión **.rule**, aquí creare un archivo con nombre **my.rules** donde crearemos las siguientes reglas personalizadas.

```

usuario@suricata:/etc/suricata/rules$ sudo touch my.rules
usuario@suricata:/etc/suricata/rules$ |

```

*Ilustración 20: Directorio de reglas*

En este archivo he creado reglas personalizadas de alertas que quiero que me muestre en los logs.

```

GNU nano 5.4 my.rules
#ICMP
alert icmp any any -> 192.168.0.11 any (msg:"ICMP encontrado hacia el servidor web";priority:1;sid:2;rev:1;)
#Fuerza bruta directorios
alert http any any -> 192.168.0.11 any (msg:"Gobuster - Fuerza bruta de directorios detectada"; flow:to_server,established; content:"GET"; http_method; content:"/"; http_uri; pcre:"/[A-Za-z0-9_.*-]+\/$/";
#Comandos
alert http any any -> any any (msg:"Posible inyeccion de comandos"; flow:established,from_server; file_data; content:"root|3a|3a|0|3a|0|3a|"; nocase; content:"|3a|/root|3a|/bin"; nocase; reference:url,doc.e
#Tráfico SQL
alert http any any -> 192.168.0.11 any (msg: "Possible SQL Injection attack (Contains singlequote)"; flow:established,to_server; content:"'"; nocase; http_uri; sid:4;)
alert http any any -> 192.168.0.11 any (msg: "Possible SQL Injection attack (Contains UNION)"; flow:established,to_server; content:"union"; nocase; http_uri; sid:5;)
alert http any any -> 192.168.0.11 any (msg: "Possible SQL Injection attack (Contains SELECT)"; flow:established,to_server; content:"select"; nocase; http_uri; sid:6;)
#Reglas NMAP
alert icmp any any -> 192.168.0.11 any (msg: "NMAP ping Scan"; priority:1; dsize:0;sid:10; rev: 1;)
alert tcp any any -> 192.168.0.11 any (msg:"NMAP XMAS Tree Scan"; flags:FPU; priority:1; sid:11; rev:1; )
alert tcp any any -> 192.168.0.11 any (msg:"NMAP FIN Scan"; flags:F; priority:1; sid:12; rev:1;)
alert tcp any any -> 192.168.0.11 any (msg:"NMAP NULL Scan"; flags:0;priority:1; sid:13; rev:1; )
alert udp any any -> 192.168.0.11 any ( msg:"Nmap UDP Scan";priority:1; sid:13; rev:1; )
#PHP
alert http any any -> any any (msg:"Posible subida de archivo PHP malicioso"; flow:established,to_server; content:"POST"; nocase; http_method; content:"<?php"; nocase; http_client_body; fast_pattern:only; ref
alert http any any -> any any (msg:"Bytes de PHP detectado"; flow:established,to_server; content:"<?php"; nocase; threshold:type limit, track by_src, seconds 60, count 1; classtype:bad-unknown; sid:15; rev:1;)

```

*Ilustración 21: Reglas my.rules*

Ahora deberemos de volver a la configuración de Suricata y indicar el nombre del archivo de las reglas en el siguiente apartado:

```

GNU nano 5.4 suricata.yaml
#
ports: [0-1,2-3]

# When auto-config is enabled the hashmode specifies the algorithm for
# determining to which stream a given packet is to be delivered.
# This can be any valid Napatech NTPL hashmode command.
#
# The most common hashmode commands are: hash2tuple, hash2tuplesorted,
# hash5tuple, hash5tuplesorted and roundrobin.
#
# See Napatech NTPL documentation other hashmodes and details on their use.
#
# This parameter has no effect if auto-config is disabled.
#
hashmode: hash5tuplesorted

##
## Configure Suricata to load Suricata-Update managed rules.
##

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

rule-files:
- suricata.rules
- my.rules
- hunting.rules
- http2.rules
- emerging.rules

```

*Ilustración 22: Indicar reglas*

En mi caso he añadido algunas reglas que he encontrado en github para mayor seguridad.

## 3.5 Estructura de las reglas.

El formato de regla en **suricata** es igual al de **snort** y se pueden usar las mismas reglas en uno como en otro.

Cada regla contiene los siguientes apartados:

### Encabezado:

- **Acción:** Especifica la acción que se tomará cuando se cumpla la condición de la regla. Puede ser "alert" para generar una alerta, "drop" para bloquear el tráfico o "reject" para rechazar la conexión. En mi caso solo quiero que mande una alerta así que solo pondremos **alert**.
- **Protocolo:** Indica el protocolo de red al que se aplica la regla, como TCP, UDP o ICMP.
- **Fuente y destino:** Especifica las direcciones IP o rangos de direcciones IP de origen y destino involucrados en la regla. Si tenemos diferentes servidores deberemos de especificarlo en la regla o en la configuración de Suricata para que solo analice el tráfico del servidor que queramos.
- **Puerto de origen y destino:** Indica los puertos de origen y destino involucrados en la regla. Puede ser un puerto único o un rango de puertos.

### Opciones:

- **Contenido:** Permite buscar un patrón específico en los datos de la capa de aplicación, como una cadena de texto o una expresión regular.
- **Clasificación:** Proporciona una categoría o etiqueta para clasificar la regla, como "malware", "ataque de red" o "explotación de vulnerabilidad".
- **Sid:** Es el identificador único de la regla y se utiliza para referirse a ella de manera única. Cada regla tiene un SID único.
- **Sid:** Es el identificador único de la regla y se utiliza para referirse a ella de manera única. Cada regla tiene un SID único.
- **Referencia:** Proporciona enlaces o referencias a fuentes externas que respaldan o explican la regla.

### Ejemplo de una regla para alertas de PING.

```
GNU nano 5.4 rules/my.rules
#ICMP
alert icmp any any -> 192.168.0.11 any (msg:"ICMP encontrado hacia el servidor web";priority:1;sid:2;rev:1;)
```

*Ilustración 23: regla para PING*

Cuando hayamos añadido las reglas que queramos reiniciamos el servicio para que se apliquen los cambios.

```
systemctl restart suricata
```

Comprobamos que esté funcionando correctamente

```
systemctl status suricata
```

```
root@suricata:/etc/suricata# systemctl status suricata
● suricata.service - Suricata IDS/IDP daemon
   Loaded: loaded (/lib/systemd/system/suricata.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2023-05-24 11:58:17 CEST; 1h 47min ago
     Docs: man:suricata(8)
           man:suricatasc(8)
           https://suricata-ids.org/docs/
   Process: 412 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited, status=0/SUCCESS)
   Main PID: 435 (Suricata-Main)
    Tasks: 9 (limit: 2337)
   Memory: 114.6M
      CPU: 4min 15.933s
   CGroup: /system.slice/suricata.service
           └─435 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

may 24 11:58:17 suricata systemd[1]: Starting Suricata IDS/IDP daemon...
may 24 11:58:17 suricata suricata[412]: 24/5/2023 -- 11:58:17 - <Notice> - This is Suricata version 6.0.1 RELEASE running in SYSTEM mode
may 24 11:58:17 suricata systemd[1]: Started Suricata IDS/IDP daemon.
root@suricata:/etc/suricata#
```

*Ilustración 24: Estado Suricata*

Por último comprobaremos una alerta iniciando una inyección SQL a un formulario de la aplicación web de DVWA con la Kali Linux. Escribimos el siguiente comando para ver en tiempo real las alertas en **fast.log**

```
root@suricata:/var/log/suricata# tail -f fast.log
05/24/2023-18:17:14.356856  [**] [1:4:0] Possible SQL Injection attack (Contains singlequote) [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.0.199:55472 -> 192.168.0.11:80
05/24/2023-18:17:14.356856  [**] [1:5:0] Possible SQL Injection attack (Contains UNION) [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.0.199:55472 -> 192.168.0.11:80
05/24/2023-18:17:14.356856  [**] [1:6:0] Possible SQL Injection attack (Contains SELECT) [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.0.199:55472 -> 192.168.0.11:80
```

*Ilustración 25: Alerta en fast.log*

Como podemos observar la IP 192.168.199 ha intentado hacer **inyección SQL** por el puerto 80 ya que en el paquete se ha encontrado una sentencia con las palabras claves SELECT y UNION.



## 3.6 Desarrollo de un bot de alertas

### 3.6.1 Creación del bot

Primero necesitamos crear un bot en Telegram al que llegarán los mensajes que enviaremos desde un programa en python desde nuestra máquina Debian con Suricata.

En Telegram buscaremos **BotFather** con el que crearemos el nuestro, escribimos **/start** y después para crearlo **/newbot**.

El bot nos mostrará una ayuda con los comandos disponibles.

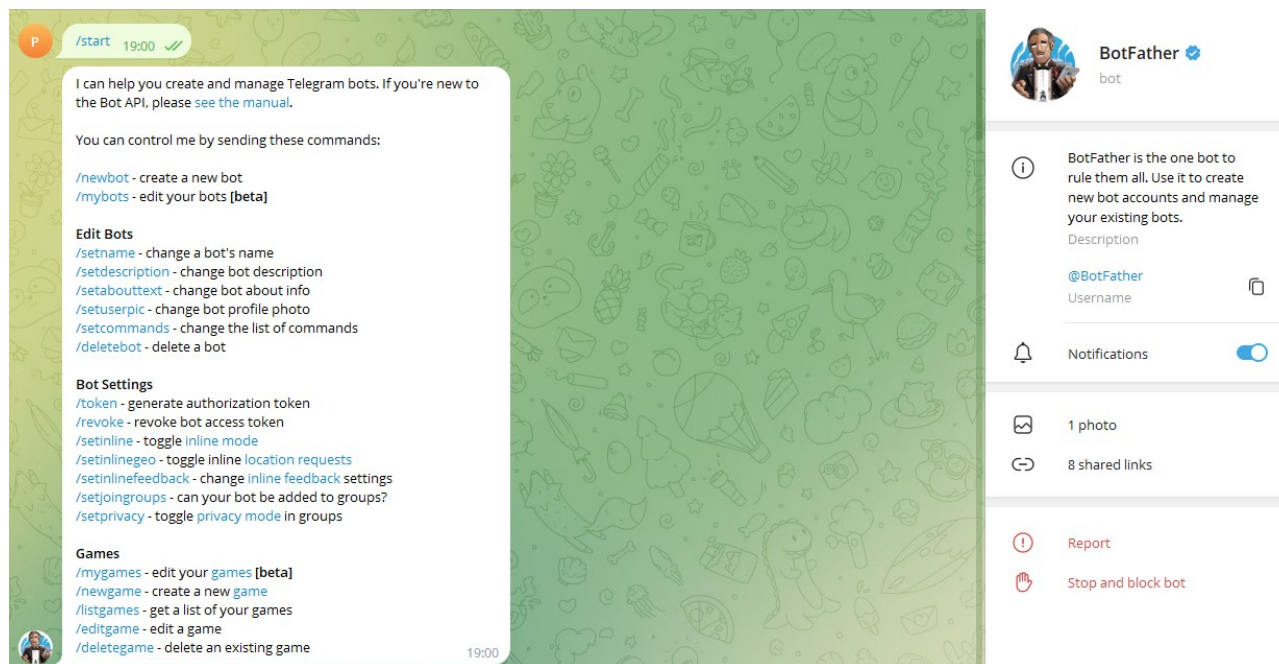


Ilustración 26: Ayuda BotFather

Después nos pedirá el nombre que queremos asignarle.

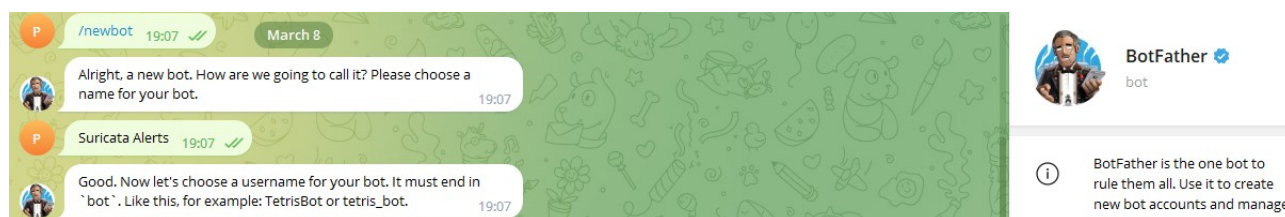


Ilustración 27: Comando /newbot



A continuación nos mostrará la API token del bot, esto lo necesitaremos más tarde para crear el programa en python que envíe los datos.



*Ilustración 28: Bot creado*

Por último yo le he añadido una imagen de perfil.



*Ilustración 29: Suricata imagen*

### 3.6.2 Desarrollo del programa en python

Para el desarrollo del programa usaremos **Visual Studio Code** ya que es gratuito y de código abierto, es extensible y personalizable, tiene una interfaz de usuario intuitiva y tienes soporte para muchos lenguajes de programación.

El programa deberá de enviar un mensaje por el chat del bot que hemos creado con el tipo de alerta que ha recibido, la fecha de la alerta, la hora a la que se ha producido, la IP origen, IP destino, puerto origen y puerto destino.

Las librerías que usaremos para el desarrollo son las siguientes:

- requests
- json
- time
- datetime
- re

En las primeras líneas importaremos las librerías previamente instaladas. A partir de la línea 7 crearemos algunas variables para sacar la fecha y hora actual. Además necesitamos una variable para indicar donde se encuentra el archivo que queremos que el programa lea.

```
1  import requests
2  import json
3  import time
4  import datetime
5  import re
6
7  archivo1 = "/var/log/suricata/fast.log"
8  a = "\n"
9  fecha_act = datetime.datetime.now()
10 hora_act = datetime.datetime.now().time()
11 h_format = "Hora: " + hora_act.strftime("%H:%M:%S") + "\n"
12 f_format = "Fecha: " + fecha_act.strftime("%d/%m/%Y") + "\n"
13 ultima = ""
14
```

*Ilustración 30: Línea 1 a 7 del programa*

Desde la línea 16 hasta la 33 crearemos una función en la que crearemos algunas variables para buscar cada parámetro en la línea del log. Utilizaremos la función “re” para buscar los datos que queramos encontrar en la línea del log. En la variable “url” pondremos la API token del bot de telegram que nos dio anteriormente.

La última parte de la función es un payload que enviará los datos contenidos en la variable text.

```
15
16 def enviar_datos():
17     url = [REDACTED]
18     ipo_detect = re.search(r'\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}', nueva)
19     ip_origen = "IP origen: " + ipo_detect.group() + a
20     ipd_detect = re.search(r'->\s*(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3})', nueva)
21     ip_d = ipd_detect.group() + a
22     ip_destino = "IP destino: " + ip_d.replace("->", "")
23     puertos = re.findall(r":[0-9]{1,5}", nueva)
24     puerto_o = "Puerto origen" + puertos[4] + a
25     puerto_d = "Puerto destino" + puertos[5] + a
26     payload = json.dumps({
27         "text": titulo+ f_format + h_format + ip_origen + puerto_o + ip_destino + puerto_d,
28         "chat_id": "2080269808"
29     })
30     headers = {
31         'Content-Type': 'application/json'
32     }
33     response = requests.request("GET", url, headers=headers, data=payload)
```

*Ilustración 31: Línea 15 a 33 del programa*

Crearemos un bucle para que el programa se esté ejecutando continuamente, le diremos que abra el archivo que le hemos indicado con la variable “archivo1”, también haremos que solo lea la última línea que se ha generado en el log para que la guarde en una variable para que no envíe sin parar los datos de la última línea. Si hay una nueva alerta lo comprobará con la que está guardada en la variable “ultima” y si es diferente continuará el programa.

Para finalizar buscaremos en cada línea una palabra clave que la identifique y si la encuentra llamará a la función “enviar\_datos” para que busque los datos y los envíe al bot.

```
36 while True:
37     with open(archivo1,"r") as archivo:
38         lineas = archivo.readlines()
39         nueva = lineas[-1]
40         linea_m = nueva.split(" ")
41         if nueva != ultima:
42
43             if "SQL" in nueva:
44                 titulo = "Se ha detectado posible inyección SQL" + a
45                 enviar_datos()
46             if "PHP" in nueva:
47                 titulo = "Se ha detectado subida archivo PHP en el servidor" + a
48                 enviar_datos()
49             if "HTTP" in nueva:
50                 titulo = "Se ha detectado trafico HTTP en el servidor" + a
51                 enviar_datos()
52             if "ICMP" in nueva:
53                 titulo = "Se ha detectado PING hacia el servidor" + a
54                 enviar_datos()
55
56         ultima = nueva
57         time.sleep(7)
58
```

*Ilustración 32: Línea 36 a 58*

Ahora solo nos queda pasar el bot a la máquina virtual con Suricata y ejecutarlo, pero antes deberemos de instalar python en este dispositivo.

Instalaremos python en la máquina virtual donde tenemos Suricata ya que el programa tiene que estar en este dispositivo para que pueda leer los logs.

Para instalarlo ejecutaremos los siguientes comandos:

```
wget https://www.python.org/ftp/python/3.9.7/Python-3.9.7.tgz
```

```
tar xzf Python-3.9.7.tgz
```

```
cd Python-3.9.7
```

```
./configure --enable-optimizations
```

```
make altinstall
```

Las librerías necesarias deberemos de instalarlas con el siguiente comando:

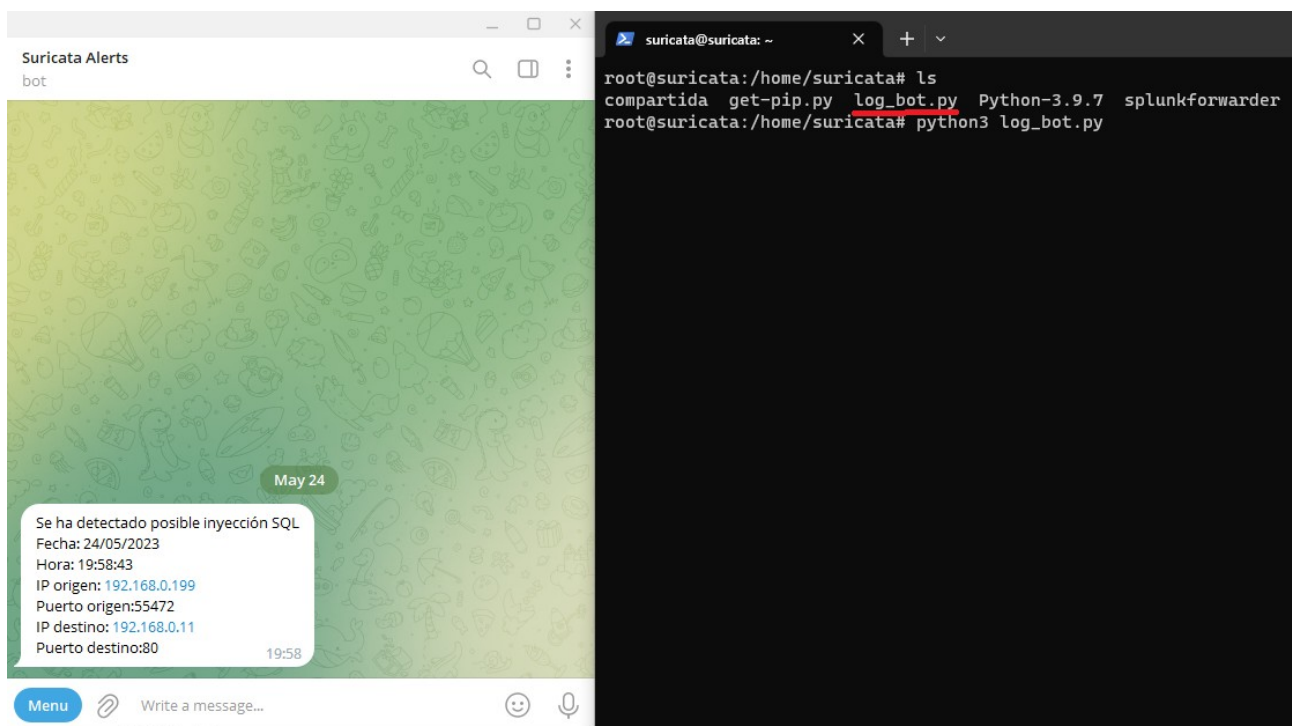
```
python3.9 -m pip install <nombre librería>
```

Comprobamos que está instalado:

```
root@suricata:/home/suricata# python3 --version
Python 3.9.2
root@suricata:/home/suricata#
```

*Ilustración 33: Versión python*

Ejecutamos el bot y comprobamos que nos ha llegado la alerta al chat de Telegram.



*Ilustración 34: Alerta en el bot*

## 4. Splunk

### 4.1 ¿Qué es Splunk?

Splunk es la solución SIEM (Security Information & Event Management) que permite monitorizar y analizar todo el **big data** de la empresa a través de una interfaz web. Toda esa información se interpreta y se plasma en un repositorio mediante **gráficos, alertas y paneles** con información clara para la toma de decisiones.

Esta aplicación busca, monitorea, analiza y visualiza grandes datos generados por máquinas, tales como sitios web, aplicaciones, servidores, redes, sensores y dispositivos móviles. Muchas organizaciones utilizan esta herramienta para profundizar en los negocios y conocer mejor a sus clientes, para reducir el riesgo de **ciberataques**, mejorar el rendimiento de sus servicios y reducir costes.

### 4.2 Ventajas de Splunk

- Funcionalidad y alcance: Splunk es conocido por su amplia funcionalidad y versatilidad en el análisis de datos en tiempo real, no solo en el ámbito de la seguridad. Puede analizar los registros de aplicaciones como Suricata.
- Arquitectura y escalabilidad: Splunk es conocido por su arquitectura distribuida y escalable, lo que permite procesar y analizar grandes volúmenes de datos, ya que en una empresa se pueden estar generando terabytes de datos que son difíciles de analizar sin un SIEM.
- Facilidad de uso: Splunk se ha ganado una reputación por su interfaz de búsqueda intuitiva y su facilidad de uso en comparación con algunos sistemas SIEM tradicionales.

### 4.3 Instalación de Splunk

Para instalar Splunk dispondremos de una máquina virtual con las siguientes características:

- 4GB de RAM.
- 3 núcleos en la CPU.
- 50GB de disco duro.

Necesitaremos más almacenamiento que en los otros equipos ya que se estarán almacenando y analizando los datos enviados por Suricata. Utilizaremos una máquina con Ubuntu Desktop.

Para descargar el instalador deberemos de hacernos una cuenta en <https://www.splunk.com> y dirigirnos a “Products” y seleccionaremos Splunk Enterprise.

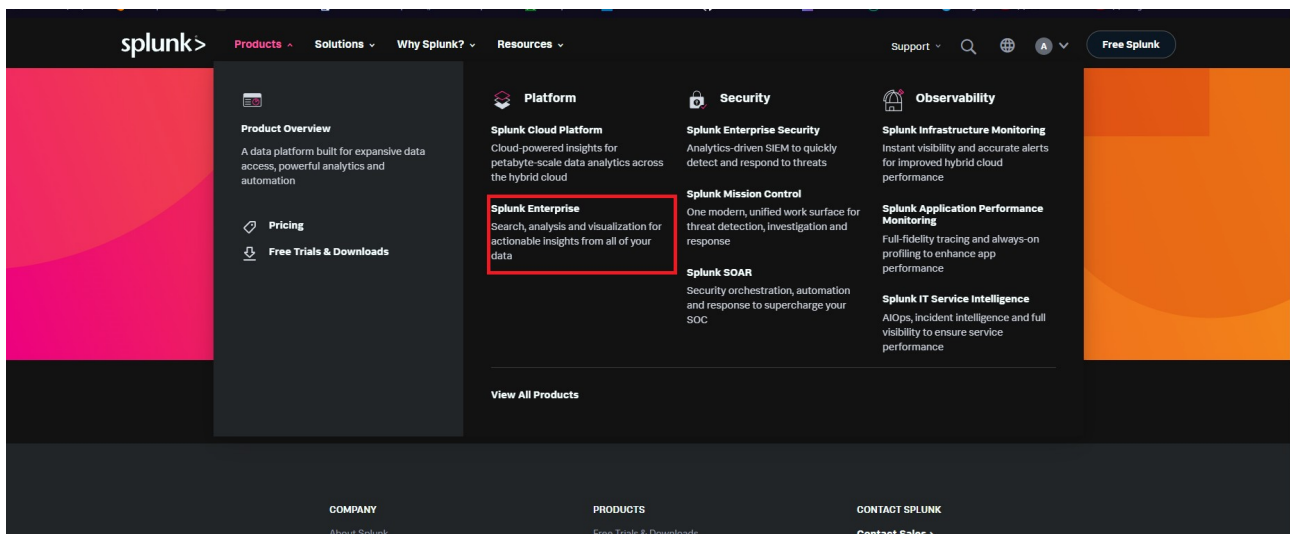


Ilustración 35: Web de Splunk

Haremos click en **Free Trial**.

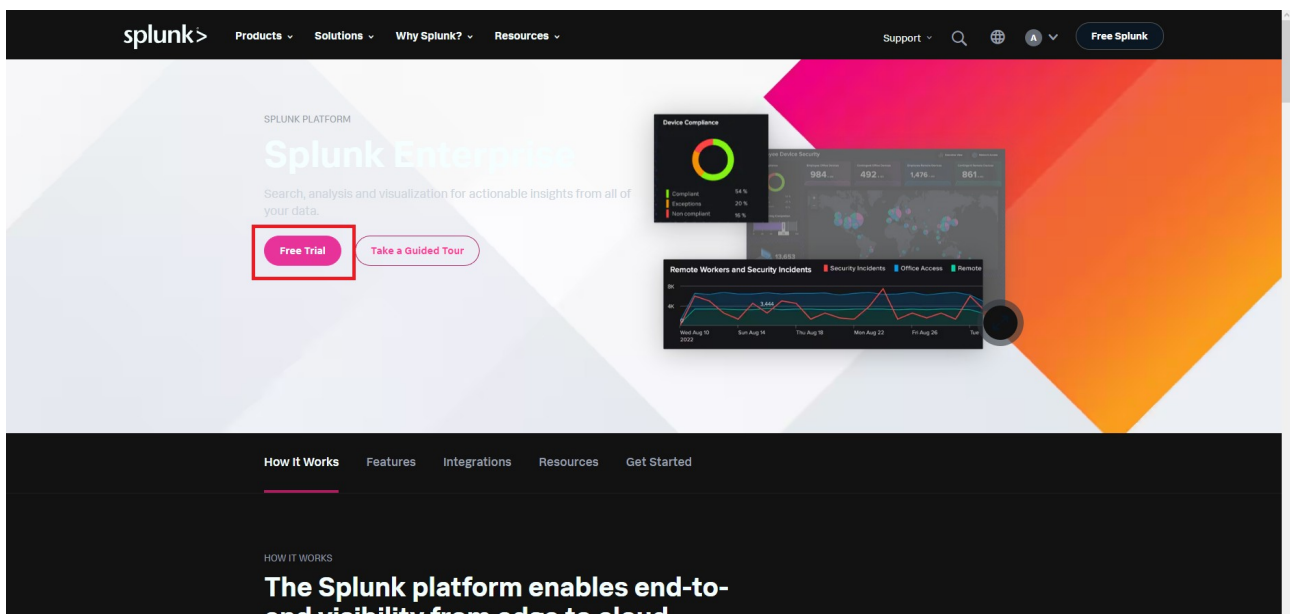


Ilustración 36: Splunk Enterprise

Nos descargaremos el archivo **.deb** para distribuciones Linux.

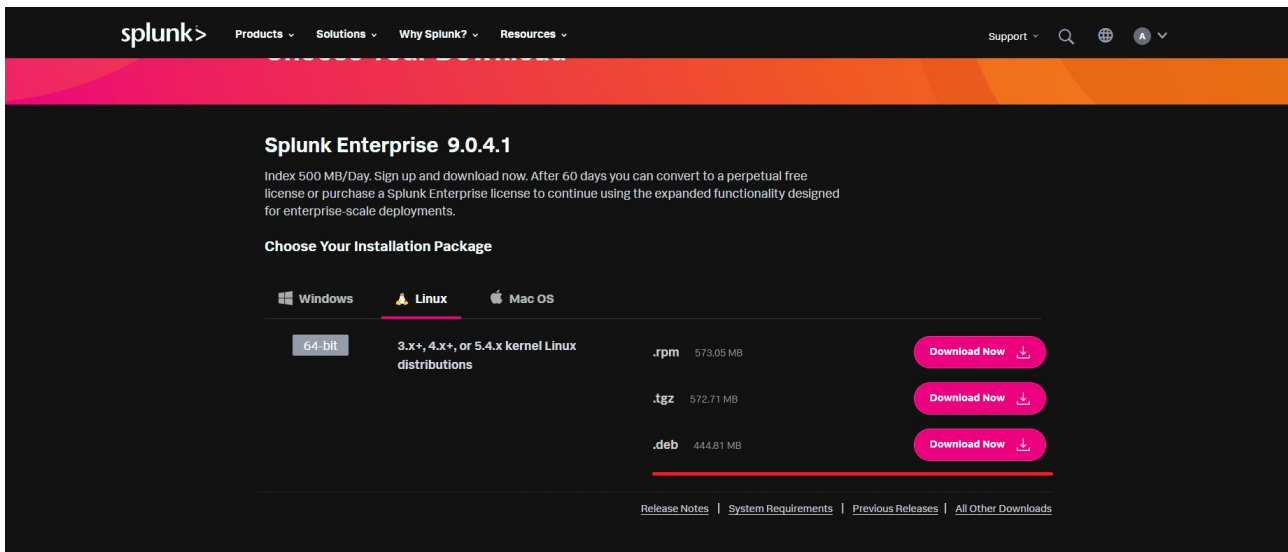


Ilustración 37: Descarga distribuciones Linux

En la siguiente página lo descargaremos con el comando **wget**.

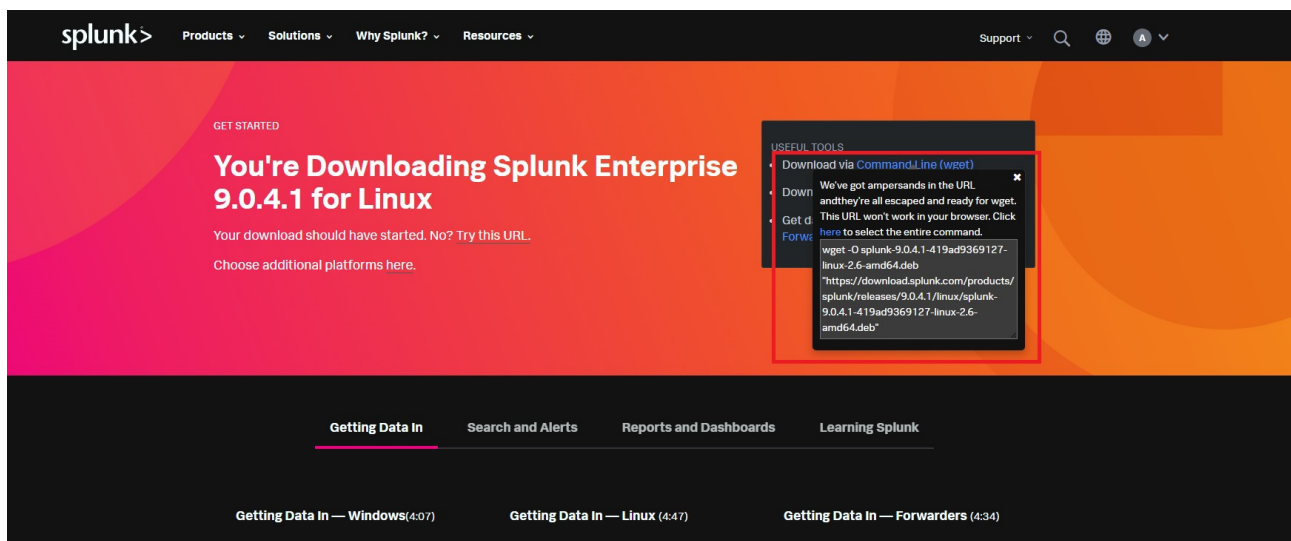


Ilustración 38: Enlace Wget de Splunk

Una vez copiado el enlace nos dirigiremos a nuestro **Ubuntu Desktop** para descargar el archivo de instalación. Lo instalaremos en el directorio **/opt**.



Ilustración 39: Descarga del instalador de Splunk



Con el comando dpkg instalaremos el archivo que previamente hemos descargado.

```
dpkg -i "nombre del archivo"
```

```
root@splunk: /opt
root@splunk:/opt# dpkg -i splunk-9.0.4.1-419ad9369127-linux-2.6-amd64.deb
(Leyendo la base de datos ... 199536 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar splunk-9.0.4.1-419ad9369127-linux-2.6-amd64.deb ...
Desempaquetando splunk (9.0.4.1) sobre (9.0.4.1) ...
Configurando splunk (9.0.4.1) ...
complete
root@splunk:/opt#
```

Nos dirigimos al directorio **/opt/splunk/bin/** como root y ejecutamos el siguiente comando para aceptar la licencia.

```
./splunk start--accept-license
```

```
root@splunk: /opt/splunk/bin
C&I Services rates are inclusive of any expenses. In the event the parties
agree that expenses are reimbursable under a Statement of Work, we will
mutually agree on any travel policy and any required documentation for
reimbursement.

11. Prepaid C&I Services
Unless otherwise expressly stated in a Statement of Work, all prepaid C&I
Services must be redeemed within twelve (12) months from the date of
purchase/invoice. At the end of the twelve (12) month term, any remaining
pre-paid unused C&I Services will expire; no refunds will be provided for any
remaining pre-paid unused C&I Services. Unless otherwise specifically stated
in a Statement of Work, Education is invoiced and payable in advance.

Configuration and Implementation Services Definitions Exhibit

"C&I Services" means the services outlined in the Statement of Work.

"C&I Services Materials" means the materials and other deliverables that are
provided to you as part of the C&I Services, and any materials, technology,
know-how and other innovations of any kind that we or our Personnel may create
or reduce to practice in the course of performing the C&I Services, including
without limitation all improvements or modifications to our proprietary
technology, and all Intellectual Property Rights therein.

"Customer Materials" means the data, information, and materials you provide to
us in connection with your use of the C&I Services.

"Fees" means the fees that are applicable to the C&I Services, as identified
in the Statement of Work.

"Intellectual Property Rights" means all worldwide intellectual property
rights, including copyrights and other rights in works of authorship; rights
in trademarks, trade names, and other designations of source or origin; rights
in trade secrets and confidential information; and patents and patent
applications.

"Personnel" means any employee, consultant, contractor, or subcontractor of
Splunk.

"Splunk Preexisting IP" means, with respect to any C&I Services Materials, all
associated Splunk technology and all Intellectual Property Rights created or
acquired: (a) prior to the date of the Statement of Work that includes such
C&I Services Materials, or (b) after the date of such Statement of Work but
independently of the C&I Services provided under such Statement of Work.

"Statement of Work" means the statements of work and/or any and all applicable
Orders, that describe the specific services to be performed by Splunk,
including any materials and deliverables to be delivered by Splunk.
Do you agree with this license? [y/n]:
```

Después nos pedirá que creamos un usuario y una contraseña para poder entrar a la aplicación de Splunk



```

Please enter an administrator username: root
Password must contain at least:
  * 8 total printable ASCII character(s).
Please enter a new password:
Please confirm new password:
Copying '/opt/splunk/etc/openldap/ldap.conf.default' to '/opt/splunk/etc/openldap/ldap.conf'.
Generating RSA private key, 2048 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
writing RSA key

Generating RSA private key, 2048 bit long modulus
.....+++++
.....+++++
e is 65537 (0x10001)
writing RSA key

Moving '/opt/splunk/share/splunk/search_mrsparkle/modules.new' to '/opt/splunk/share/splunk/search_mrsparkle/modules'.

```

*Ilustración 40: Creación de usuario*

A continuación iniciaremos el servicio desde el directorio /opt/splunk/bin/ con el siguiente comando.

```

root@splunk:/opt/splunk/bin# ./splunk start
Splunk> The IT Search Engine.

Checking prerequisites...
  Checking http port [8000]: open
  Checking mgmt port [8089]: open
  Checking appserver port [127.0.0.1:8065]: open
  Checking kvstore port [8191]: open
  Checking configuration... Done.
    Creating: /opt/splunk/var/run/splunk/appserver/i18n
    Creating: /opt/splunk/var/run/splunk/appserver/modules/static/css
    Creating: /opt/splunk/var/run/splunk/upload
    Creating: /opt/splunk/var/run/splunk/search_telemetry
    Creating: /opt/splunk/var/spool/splunk
    Creating: /opt/splunk/var/spool/dirmoncache
    Creating: /opt/splunk/var/lib/splunk/authdb
    Creating: /opt/splunk/var/lib/splunk/hashdb
New certs have been generated in '/opt/splunk/etc/auth'.
  Checking critical directories... Done
  Checking indexes...
    Validated: _audit _configtracker _internal _introspection _metrics _metrics_rollup _telemetry _thefishbucket history main summary
    Done
  Checking filesystem compatibility... Done
  Checking conf files for problems... Done
  Checking default conf files for edits...
  Validating installed files against hashes from '/opt/splunk/splunk-9.0.4.1-419ad9369127-linux-2.6-x86_64-manifest'
  All installed files intact.
  Done
All preliminary checks passed.

Starting splunk server daemon (splunkd)...
Generating a RSA private key
.....+++++
.....+++++
writing new private key to 'privKeySecure.pem'
-----
Signature ok
subject=CN=splunk/O=SplunkUser
Getting CA Private Key
writing RSA key
PYTHONHTTPSVERIFY is set to 0 in splunk-launch.conf disabling certificate validation for the httplib and urllib libraries shipped with the embedded Python interpreter; must be set to "1" for increased se
curity
Done

Waiting for web server at http://127.0.0.1:8000 to be available..... Done

```

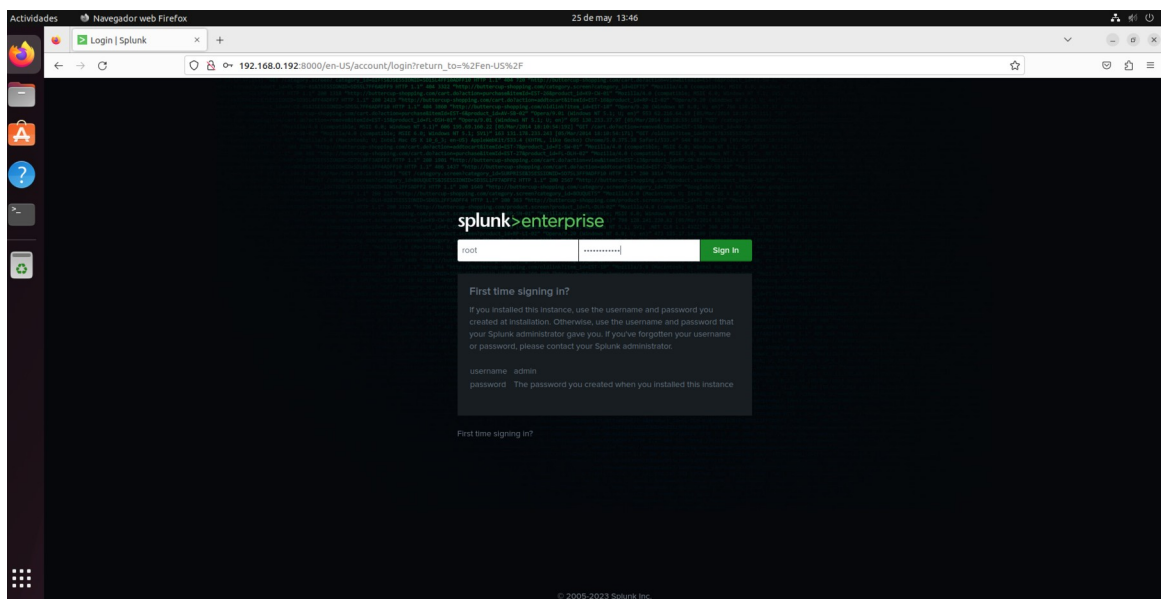
*Ilustración 41: Inicio de Splunk*

## 4.4 Configuración de Splunk

### 4.4.1 Instalación de Stamus Network

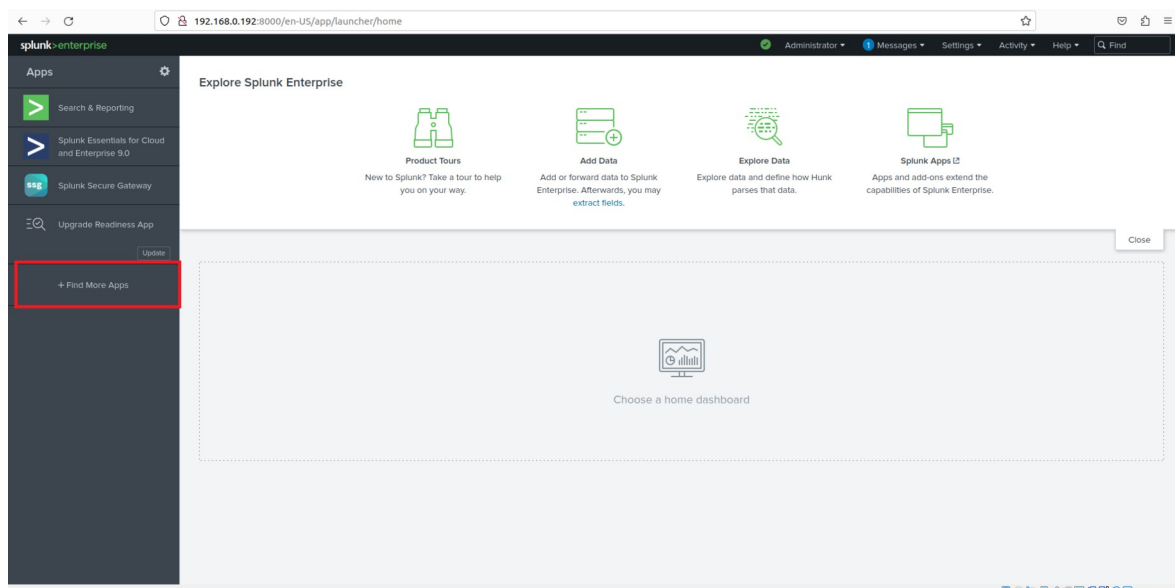
Necesitaremos instalar una extensión para poder analizar los registros de Suricata. Desde nuestra maquina virtual con Suricata deberemos de enviarle el registro eve.json para que la aplicación de Stamus Network App pueda digerir y clasificar los datos. Configuraremos los puertos por los que Splunk recibirá los datos y cómo los enviaremos a través de Splunk Forwarder.

Primero comenzaremos a instalar la extensión:



*Ilustración 42: Inicio de sesión de Splunk*

Una vez dentro tendremos que instalar una extensión para poder analizar los registros de Suricata. Instalaremos la extensión de Stamus Network App.



*Ilustración 43: Página principal de Splunk*

Escribimos en la barra de búsqueda la aplicación y la instalamos.

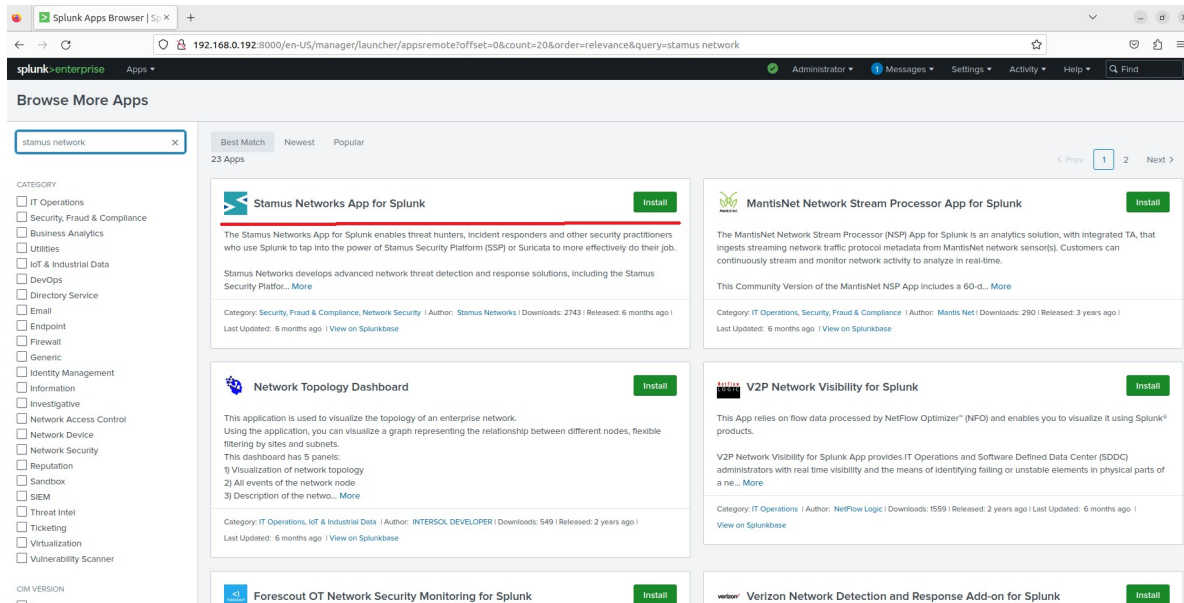


Ilustración 44: Instalación de Stamus Network

## 4.4.2 Configuración de puertos.

Una vez instalada tendremos que configurar los puertos para recibir los datos, nos dirigimos a **settings** y seleccionamos forwarding and receiving. Abriremos los puertos 9997 y 9998 que después configuraremos en splunk forwarden en la máquina virtual de Suricata.

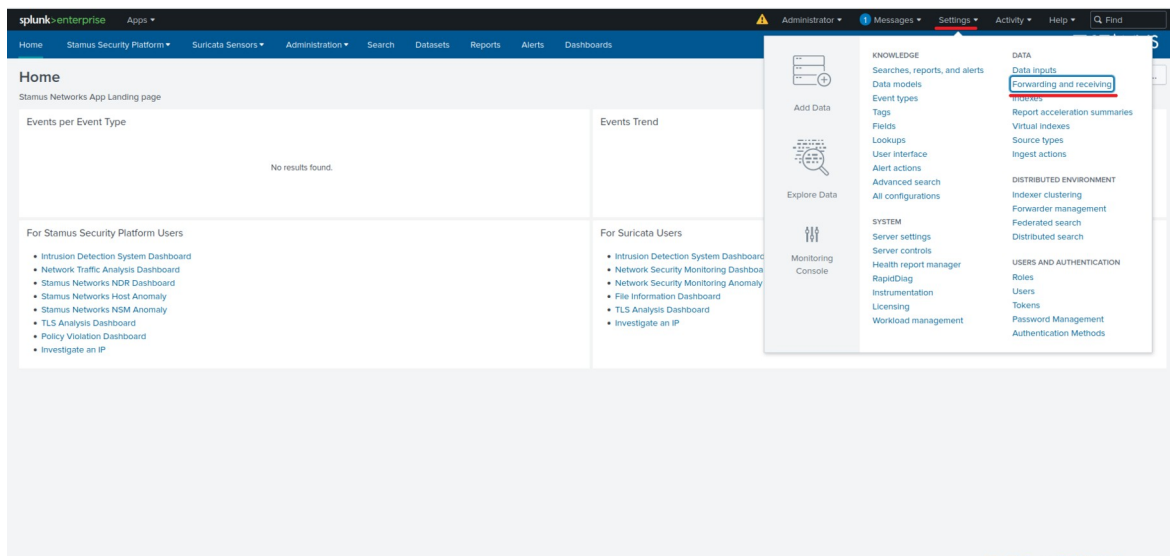
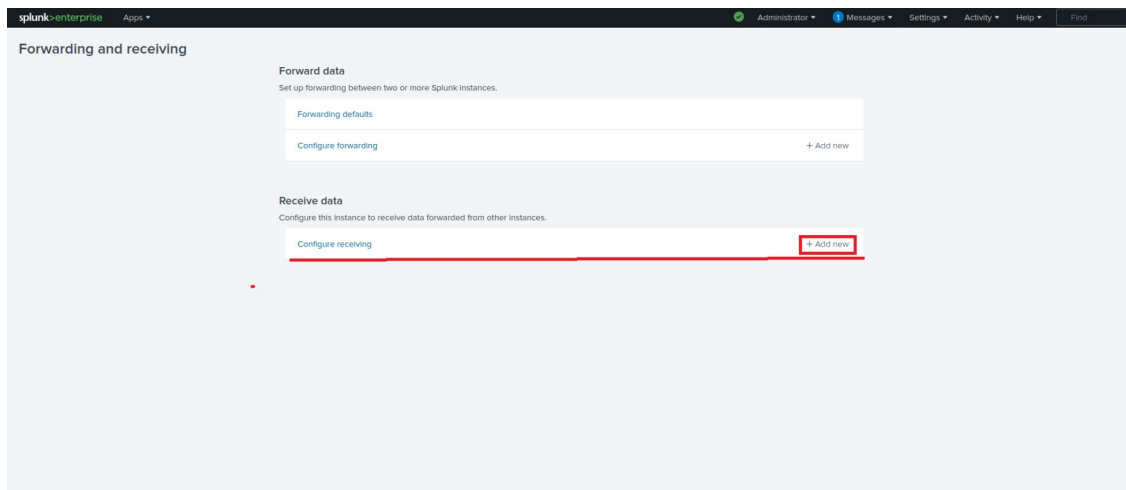


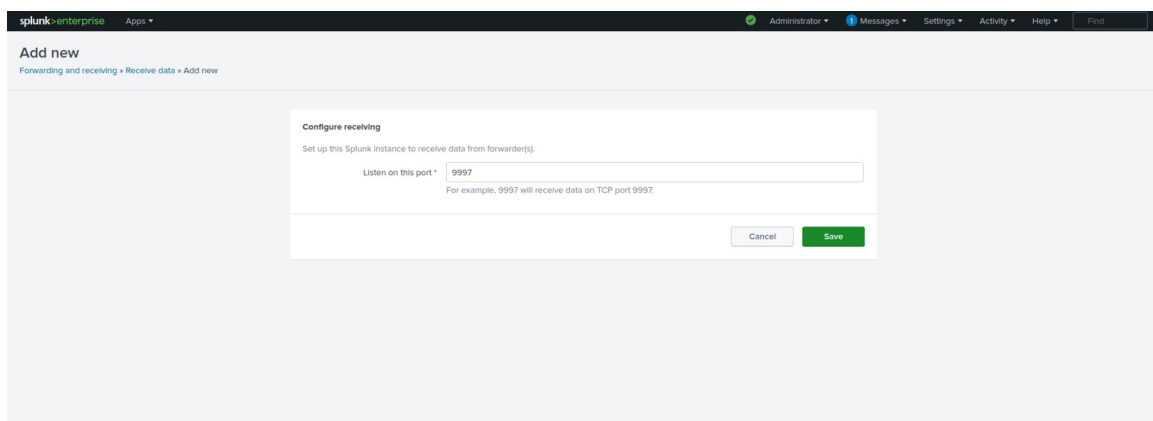
Ilustración 45: Forward and receiveing

Seleccionamos Configure receiving para recibir los datos.



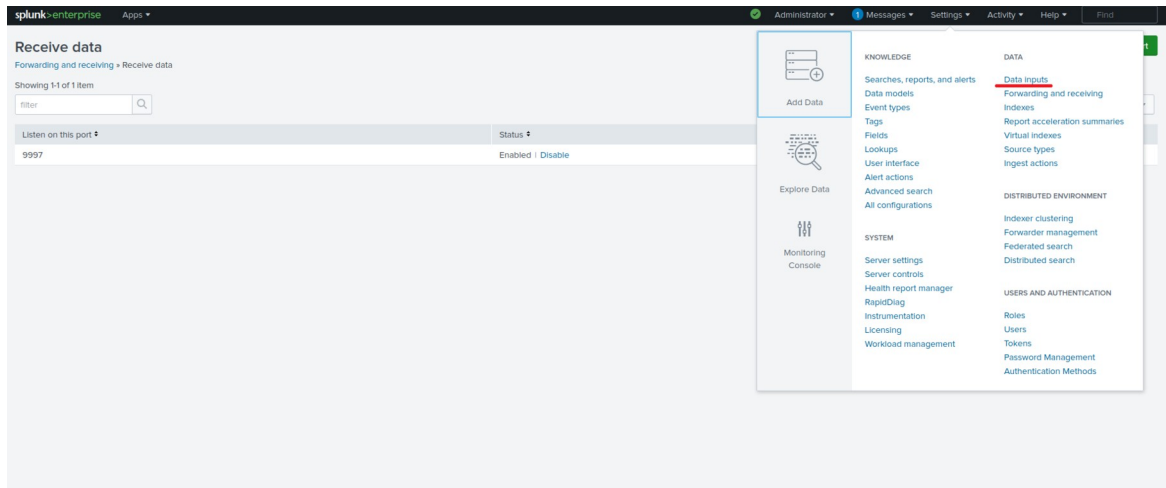
*Ilustración 46: Configurar recibir datos*

Elegimos el puerto por el que queremos que Splunk reciba los datos, en mi caso elegiré el 9997.



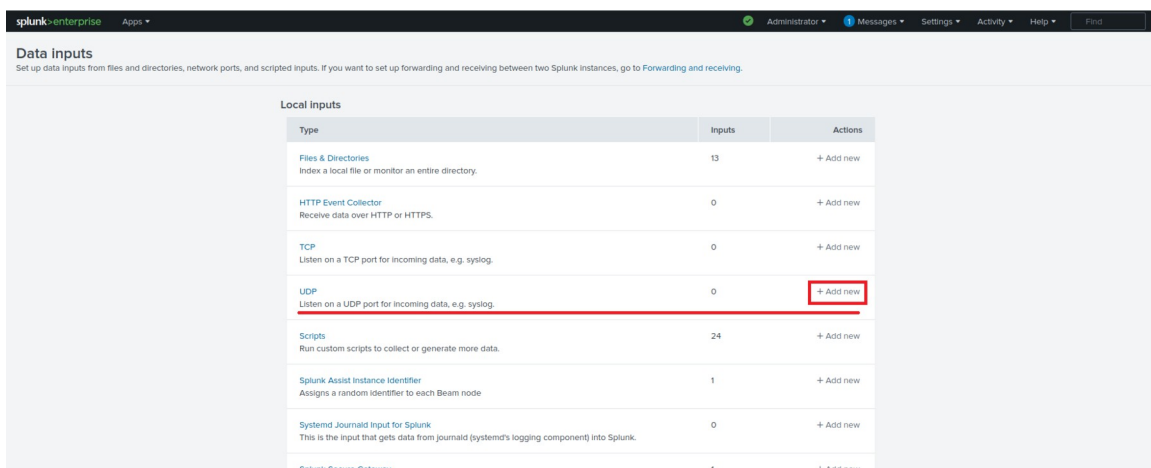
*Ilustración 47: Puerto 9997*

Ahora deberemos de configurar los **index** para que la aplicación de Stamus Network pueda organizar los datos que recibe en gráficos. Nos dirigimos a **Data inputs**.



*Ilustración 48: Data inputs*

Elegimos que reciba los datos por **UDP**.



*Ilustración 49: Data input UDP*

Elegimos el puerto que elegimos anteriormente y seleccionamos **next**. Las demás opciones son opcionales y no es necesario rellenarlas.

The screenshot shows the 'Add Data' configuration page in Splunk. The 'Select Source' step is active, showing a list of data sources on the left. The 'TCP / UDP' source is selected. On the right, the 'Input Settings' are configured: 'Port' is set to 9997, 'Source name override' is optional, and 'Only accept connection from' is optional. A 'Next >' button is visible at the top right.

Ilustración 50: configuración TCP/UDP

Ahora elegiremos como Splunk clasifica la información que recibe para que la aplicación de Stamus Network pueda analizarla.

En **source type** deberemos de seleccionar suricata para que Splunk clasifique todos los datos que recibe como sourcetype = suricata.

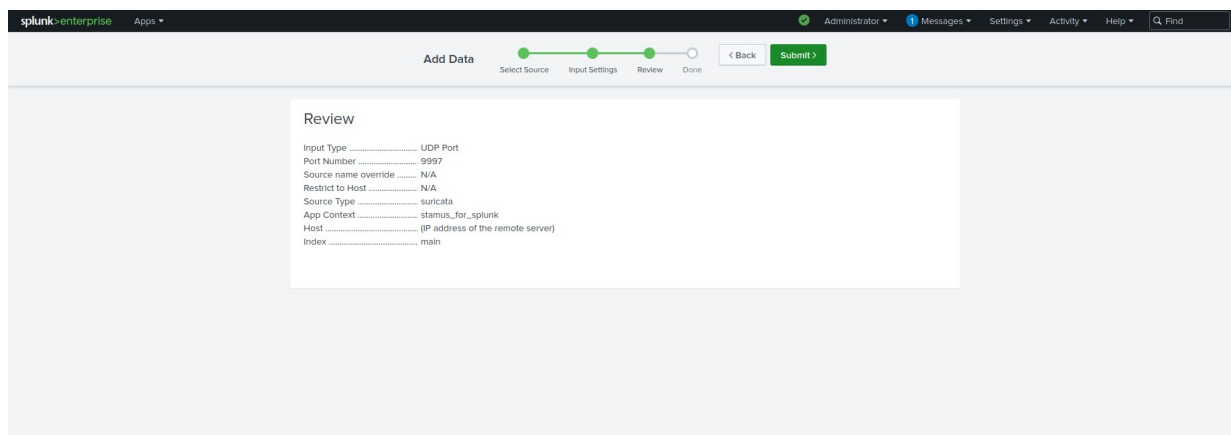
En App context deberemos de seleccionar la aplicación con la que analizaremos la información recibida que tenga como tipo de fuente suricata.

Como **index** elegiremos **main** ya que la aplicación utiliza este nombre para buscar los datos.

The screenshot shows the 'Input Settings' configuration page in Splunk. The 'Input Settings' step is active. The 'Source type' is set to 'suricata'. The 'App Context' is set to 'Stamus Networks App for Splunk (stamus\_for\_spl...)'. The 'Method' is set to 'IP'. The 'Index' is set to 'main'. A 'Review >' button is visible at the top right.

Ilustración 51: Inputs Settings

Por último seleccionamos **submit** para que se apliquen los cambios.



*Ilustración 52: Confirmación de configuración*

Deberemos de autorizar el recibimiento de datos en el firewall.

```
ufw allow 9997
```

```
root@splunk:/opt/splunk/bin# ufw allow 9997
Reglas actualizadas
Reglas actualizadas (v6)
root@splunk:/opt/splunk/bin#
```

### 4.4.3 Splunk Forwarder

Para enviar los datos a Splunk deberemos de instalar en la máquina de **Suricata**. Este componente sirve para **recopilar** y **monitorear** los datos del registro que elijamos para enviarlo por el puerto que queramos. **Splunk Forwarder** garantiza la seguridad de los datos enviados a través de técnicas de cifrados y autenticación para proteger la integridad y confidencialidad de los datos.

Se puede configurar para que se envíen a otros forwarders para un reenvío en forma de **cascada**, para nuestro caso lo enviaremos directamente a **Splunk**.

Nos dirigimos a la página web de Splunk para descargar el instalador de Splunk Forwarder.

Descargaremos el Universal Forwarder.

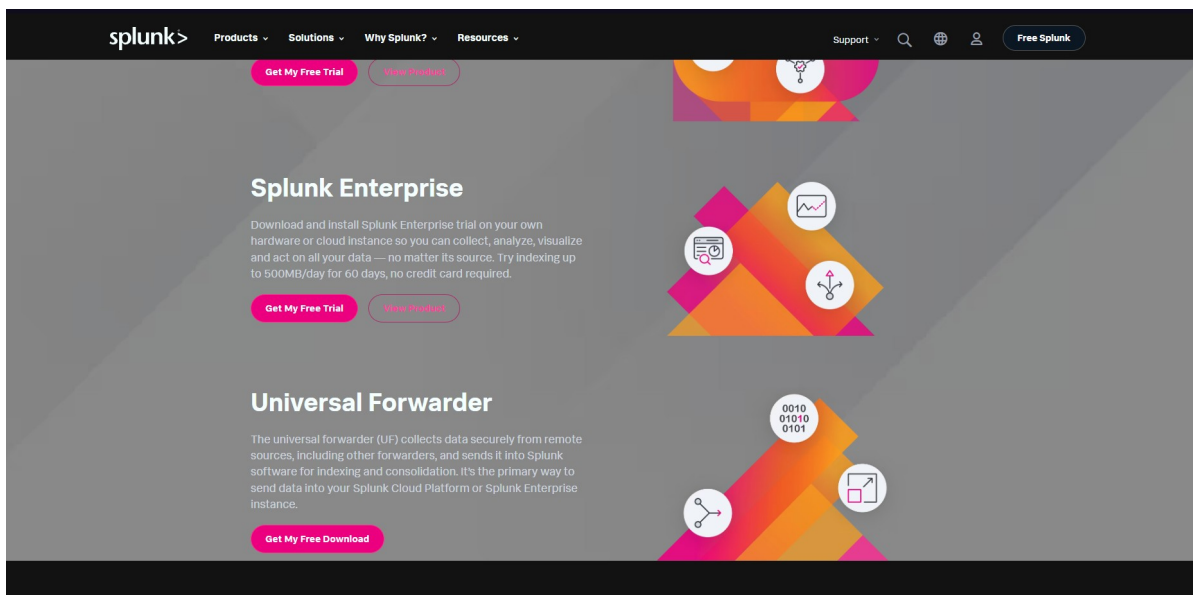


Ilustración 53: Splunk Universal Forwarder

Elegimos distribuciones Linux y descargamos el .deb.

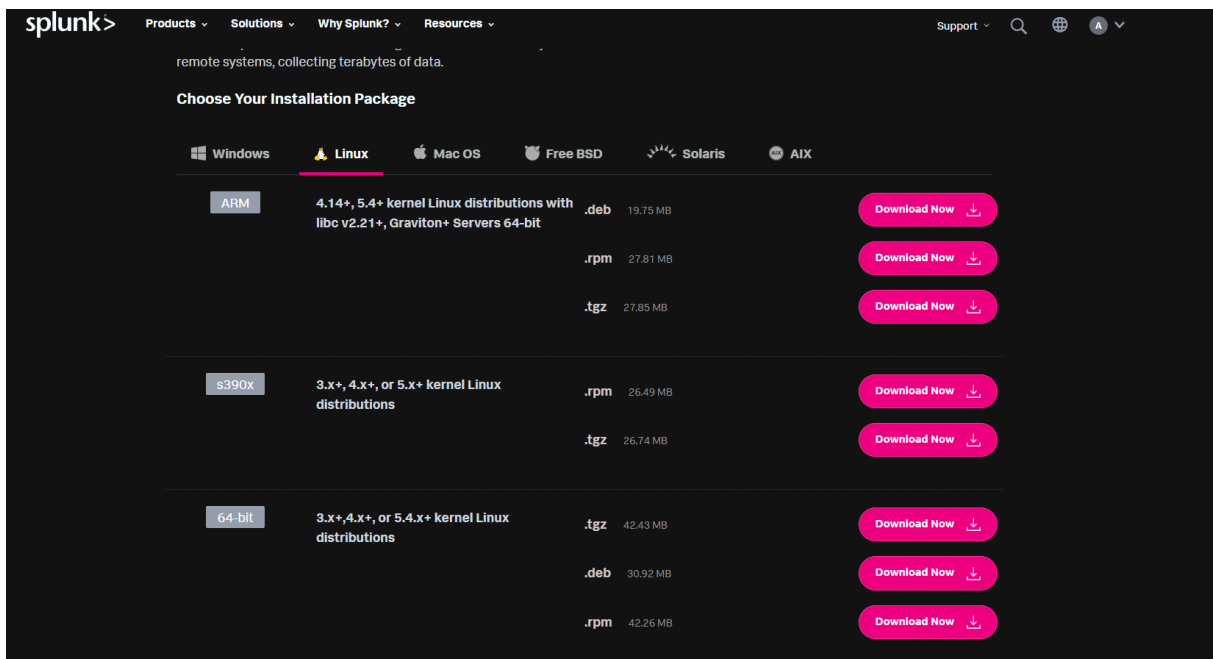


Ilustración 54: Página web Universal Forwarder



Con wget descargamos el forwarder y lo instalamos con dpkg.

```
wget -O "enlace"
```

```
ufw allow 9997
```

```
root@suricata:/opt# wget -O splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb "https://download.splunk.com/products/universalforwarder/releases/9.0.4/linux/splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb"
--2023-05-27 13:46:06-- https://download.splunk.com/products/universalforwarder/releases/9.0.4/linux/splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb
Resolviendo download.splunk.com (download.splunk.com)... 13.225.244.115, 13.225.244.76, 13.225.244.34, ...
Conectando con download.splunk.com (download.splunk.com)[13.225.244.115]:443... conectado.
Petición HTTP enviada, esperando respuesta... 200 OK
Longitud: 32420798 (31M) [binary/octet-stream]
Guardando como: 'splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb'

splunkforwarder-9.0.4-de405f4 100%[=====] 30,92M 16,0MB/s en 1,9s

2023-05-27 13:46:08 (16,0 MB/s) - 'splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb' guardado [32420798/32420798]

root@suricata:/opt# dpkg -i splunk
dpkg: warning: package 'splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb'
root@suricata:/opt# dpkg -i splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb
Seleccionando el paquete splunkforwarder previamente no seleccionado.
(Leyendo la base de datos ... 202463 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar splunkforwarder-9.0.4-de405f4a7979-linux-2.6-amd64.deb ...
Desempaquetando splunkforwarder (9.0.4) ...
```

*Ilustración 55: Descarga del forwarder*

Ahora nos dirigimos hacia el directorio donde hemos descargado el forwarder y ejecutamos el comando para instalarlo y aceptar la licencia. También nos pedirá que creamos un usuario.

```
root@suricata:/opt/splunkforwarder/bin# ./splunk start --accept-license
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunk /opt/splunkforwarder"

This appears to be your first time running this version of Splunk.

Splunk software must create an administrator account during startup. Otherwise, you cannot log in.
Create credentials for the administrator account.
Characters do not appear on the screen when you type in credentials.

Please enter an administrator username: root
Password must contain at least:
    * 8 total printable ASCII character(s).
Please enter a new password:
Please confirm new password:
Creating unit file...
Important: splunk will start under systemd as user: splunk
The unit file has been created.

Splunk> Needle. Haystack. Found.

Checking prerequisites...
    Checking mgmt port [8089]: open
    Creating: /opt/splunkforwarder/var/lib/splunk
    Creating: /opt/splunkforwarder/var/run/splunk
    Creating: /opt/splunkforwarder/var/run/splunk/appserver/i18n
    Creating: /opt/splunkforwarder/var/run/splunk/appserver/modules/static/css
    Creating: /opt/splunkforwarder/var/run/splunk/upload
    Creating: /opt/splunkforwarder/var/run/splunk/search_telemetry
```

*Ilustración 56: Instalación Universal Forwarder*

Ahora configuraremos el Universal Forwarder para que envíe los datos a Splunk. Deberemos de indicar la IP y por el puerto que configuramos anteriormente por el que recibirá los datos.

```
./splunk add forward-server 192.168.0.192:9997
```

```
root@suricata:/opt/splunkforwarder/bin# ./splunk add forward-server 192.168.0.192:9997
Warning: Attempting to revert the SPLUNK_HOME ownership
Warning: Executing "chown -R splunk /opt/splunkforwarder"
WARNING: Server Certificate Hostname Validation is disabled. Please see server.conf/[sslConfig]/cliVerifyServerName for details.
Added forwarding to: 192.168.0.192:9997.
root@suricata:/opt/splunkforwarder/bin#
```

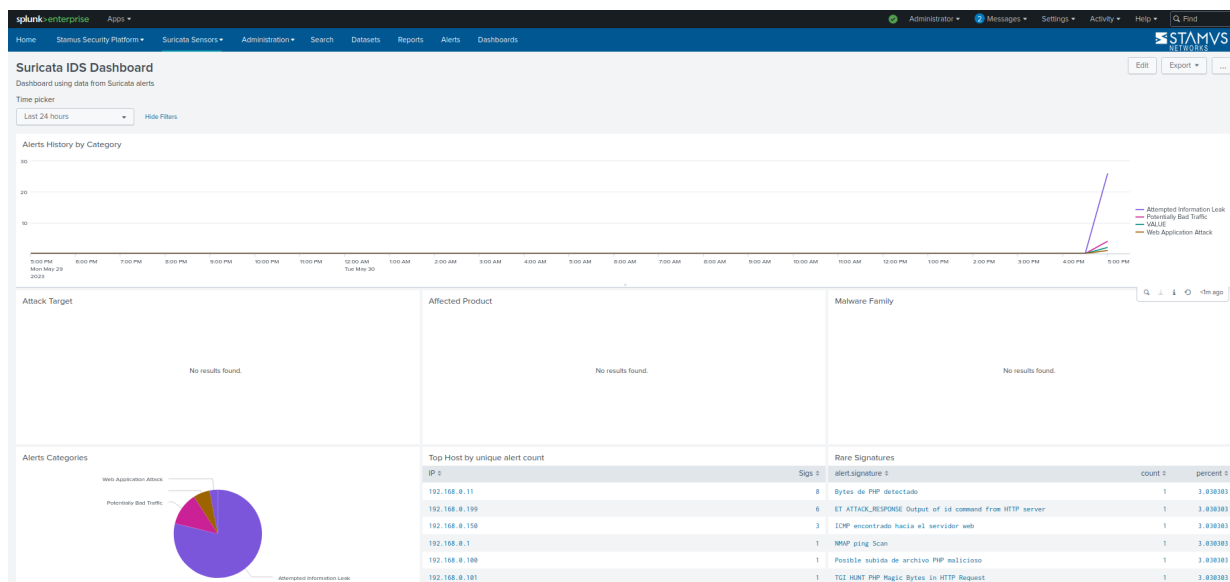
*Ilustración 57: Añadir servidor para enviar datos*

Por último indicaremos el registro que queremos que Splunk Universal Forwarder monitoree y envíe a nuestro SIEM.

```
./splunk add monitor -auth usuario:contraseña /var/log/suricata/eve.json
```

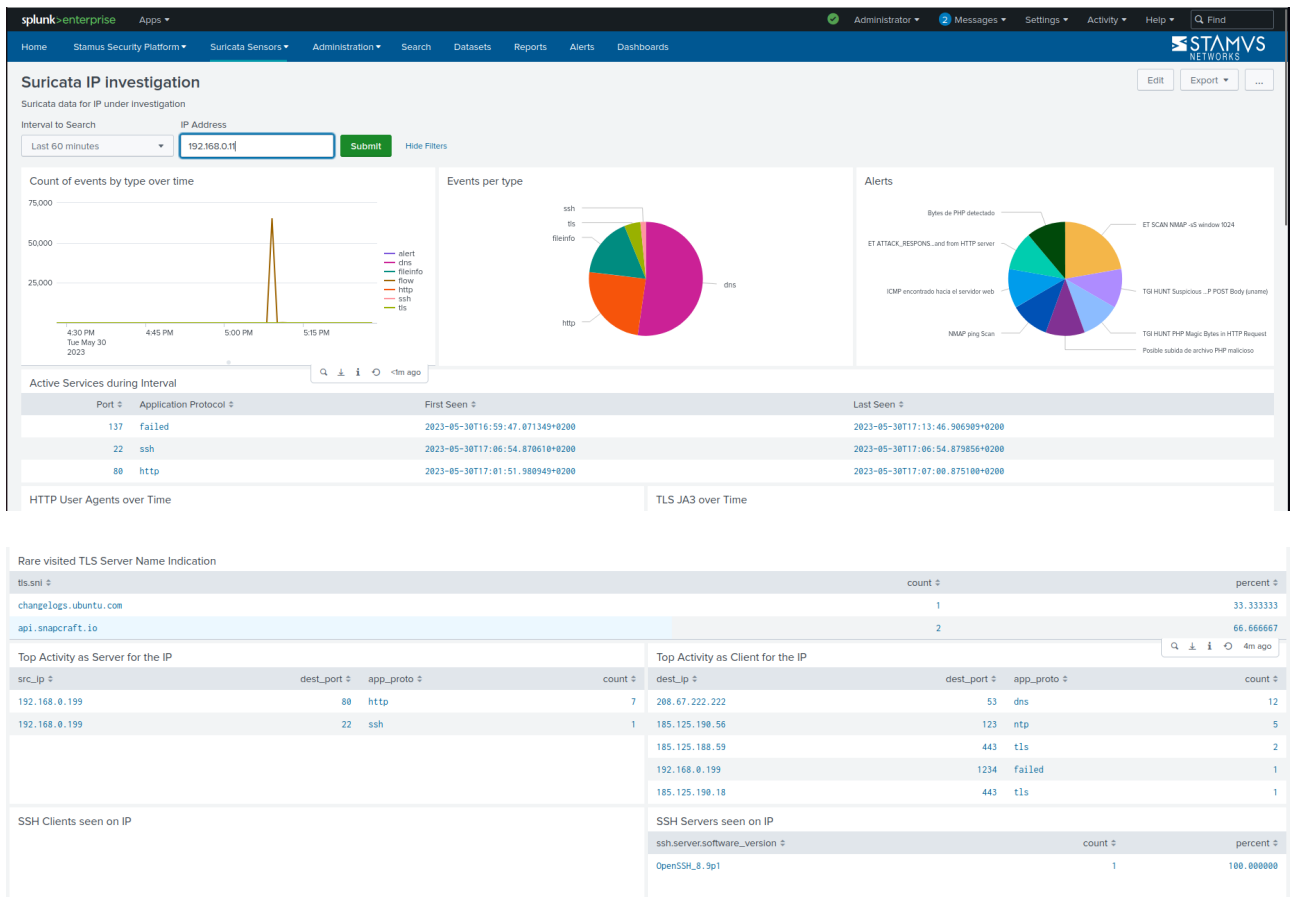
Este es el registro que deberemos de enviar al SIEM para que Stamus Network App clasifique la información en tiempo real y podamos analizar los ataques que recibe el HoneyPot.

Nos dirigimos a la aplicación que habíamos descargado en Splunk y comprobamos que están llegando datos.



*Ilustración 58: Suricata IDS Dashboard*

En la sección de **IP investigation** podemos analizar el tráfico que le llega al servidor, se puede observar que ha recibido diferentes tipos de ataques como **escaneos con nmap**, pings hacia el servidor, posible **inyección SQL**, intentos de conexión por **ssh** y subidas de archivos con extensión **.php** desde nuestra Kali Linux con IP 192.168.0.199.



## 5. Kali Linux

### 5.1 ¿Qué es Kali Linux?

Kali Linux es una distribución de Linux basada en Debian, específicamente diseñada para temas de seguridad muy variados, como análisis de redes, ataques inalámbricos, análisis forenses. Es una buena forma de comenzar en la ciberseguridad ya que contiene herramientas para llevar a cabo todas estas pruebas de seguridad y análisis.

Para atacar nuestro HoneyPot utilizaremos varios de los tipos de ataques más comunes que podremos encontrar hacia un servidor web con diferentes tipos de servicios.

- Iniciar una reverse shell mediante la subida de un archivo con extensión php.
- Inyección SQL para sacar información de los usuarios en la base de datos.
- Escaneos con nmap para descubrir los servicios abiertos con vulnerabilidades.
- Inyección de comandos.

Primero comenzaremos con la instalación de Kali.

### 5.2 Instalación de Kali Linux

Utilizaremos una máquina virtual con las siguientes características:

- 5 núcleos de CPU.
- 4GB de RAM.
- 80 GB de disco duro.

Nos dirigimos a la página web oficial de Kali Linux y descargamos la máquina virtual para Virtual Box.

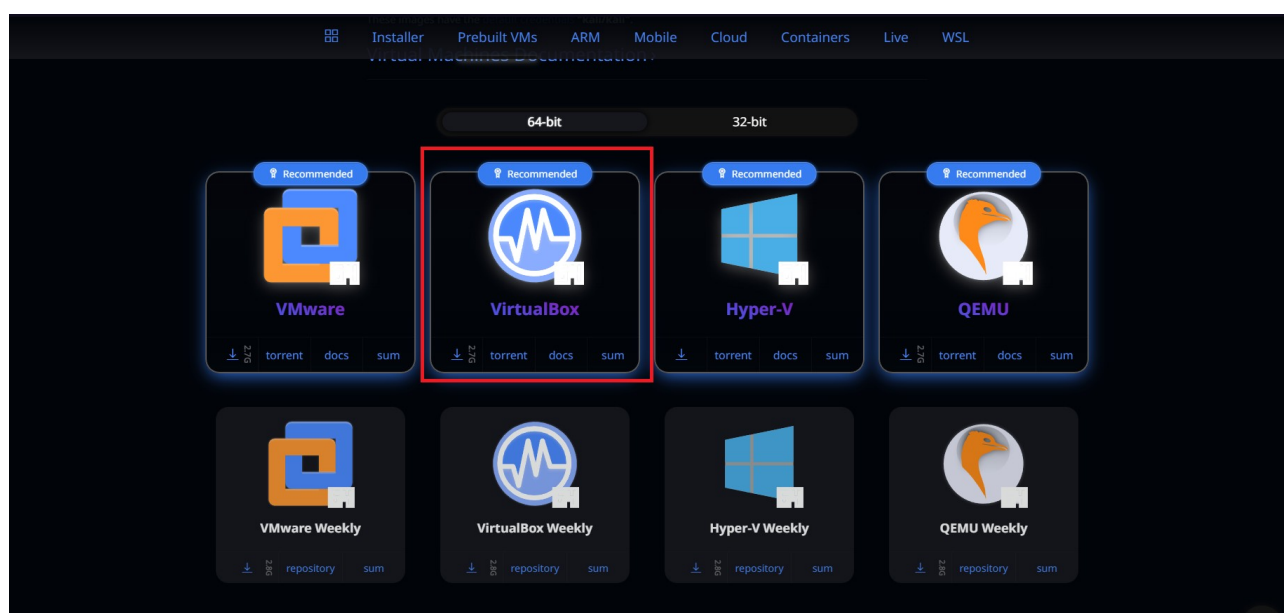


Ilustración 59: Descarga Kali Linux

Una vez descargada e importada a Virtual Box iniciamos la máquina y iniciamos sesión con el usuario Kali y contraseña Kali, es recomendable cambiar el hostname y la contraseña ya que algunos SIEM envían alertas cuando un equipo con hostname Kali se recibe IP en la red que está monitoreando.

Esta máquina virtual ya contiene las herramientas que utilizaremos para atacar el servidor web.

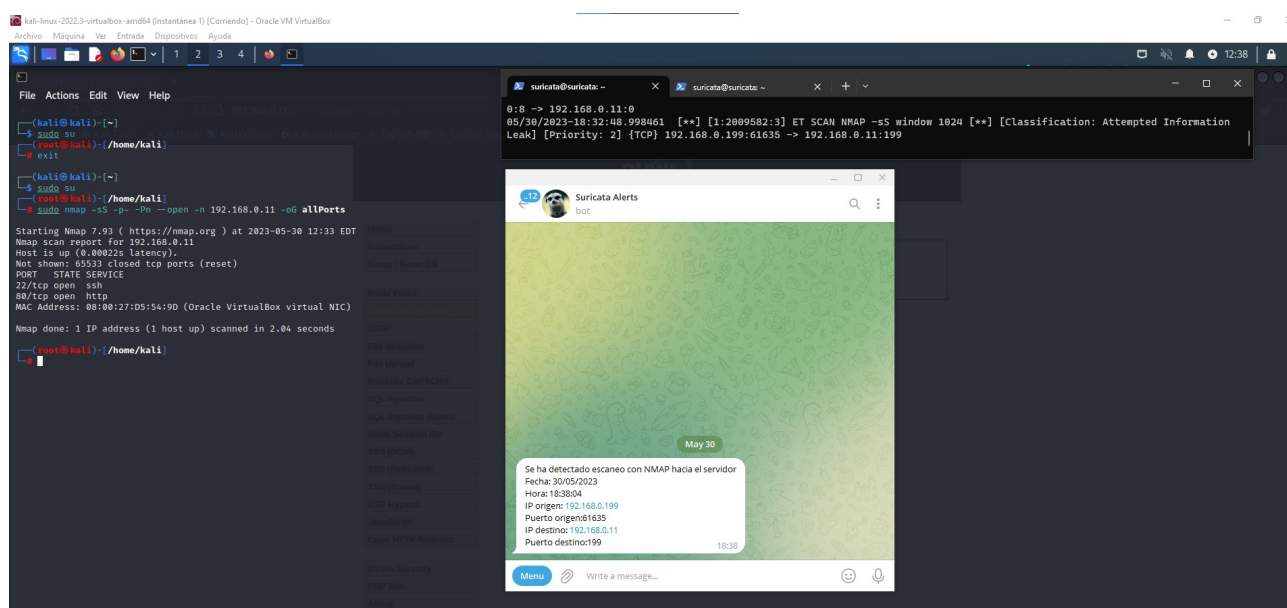
Abriremos una terminal y iniciamos sesión como root ya que herramientas como nmap nos pedirá permisos para poder utilizarlos.

Comenzaremos con un escaneo con nmap al servidor web, utilizaremos el siguiente comando:

```
sudo nmap -sS -p- -Pn --open -n 192.168.0.11 -oG allPorts
```

Este escaneo analizará todos los puertos que estén abiertos de la siguiente IP y lo guardará en un archivo .txt con los datos obtenidos.

Como podemos ver el puerto 80 y el 22 están abiertos y Suricata ha detectado el escaneo con nmap y ha enviado una alerta al bot de Telegram y a Splunk.



*Ilustración 60: Escaneo con nmap*

Al ver que el puerto 80 está abierto pondremos la IP en el navegador para entrar en la web del HoneyPot. Nos dirigimos a la sección de **SQL injection** donde tendremos un formulario en el que podemos ingresar datos, si probamos la siguiente sentencia:

```
%'or '0'='0' union select null, database() #
```

Comprobamos que la web nos muestra unos resultados que han sido mostrados por la consulta que hemos hecho ya que no se está revisando el tipo de consulta que se ha ejecutado y Suricata detecta que dentro de la sentencia están las palabras UNION y SELECT.

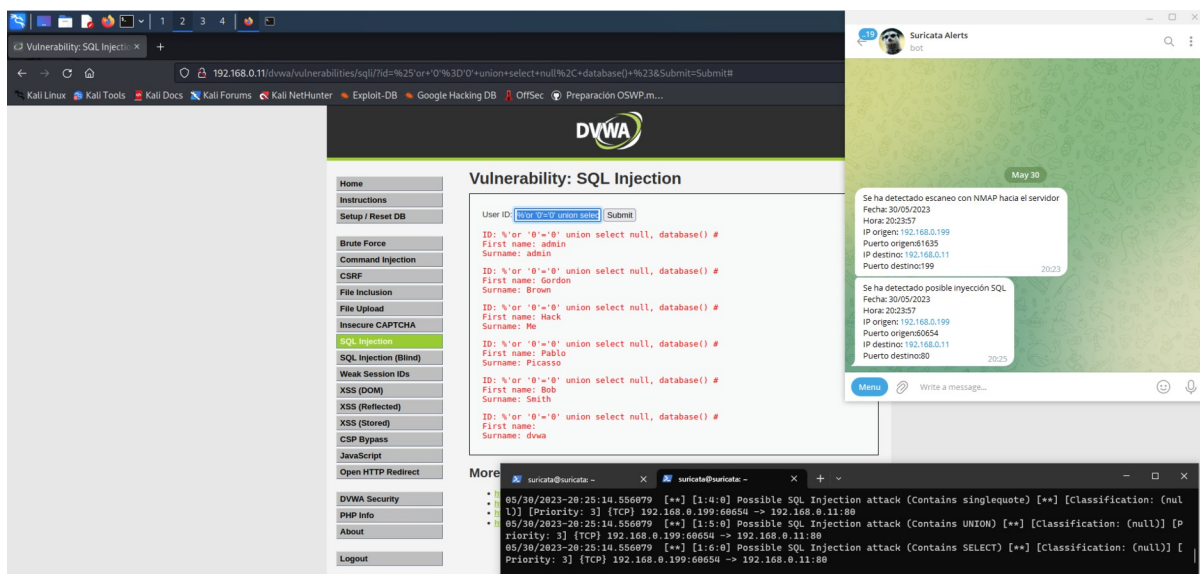


Ilustración 61: inyeccion SQL

Ahora iniciaremos una reverse shell subiendo un archivo .php que al ejecutarlo en la web deja abierto un puerto para conectarnos al servidor.

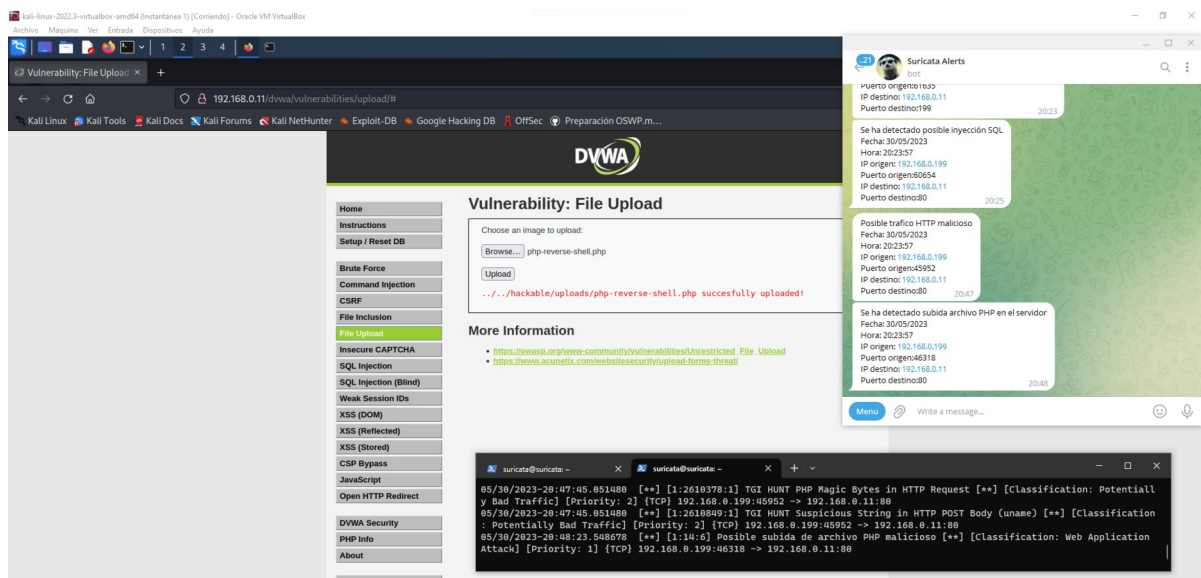


Ilustración 62: subida de reverse shell



Una vez subida nos dirigimos al directorio donde se ha subido y mediante un comando en la Kali estaremos escuchando por el puerto 1234 para entrar en el servidor.

```
nc -lvp 1234
```

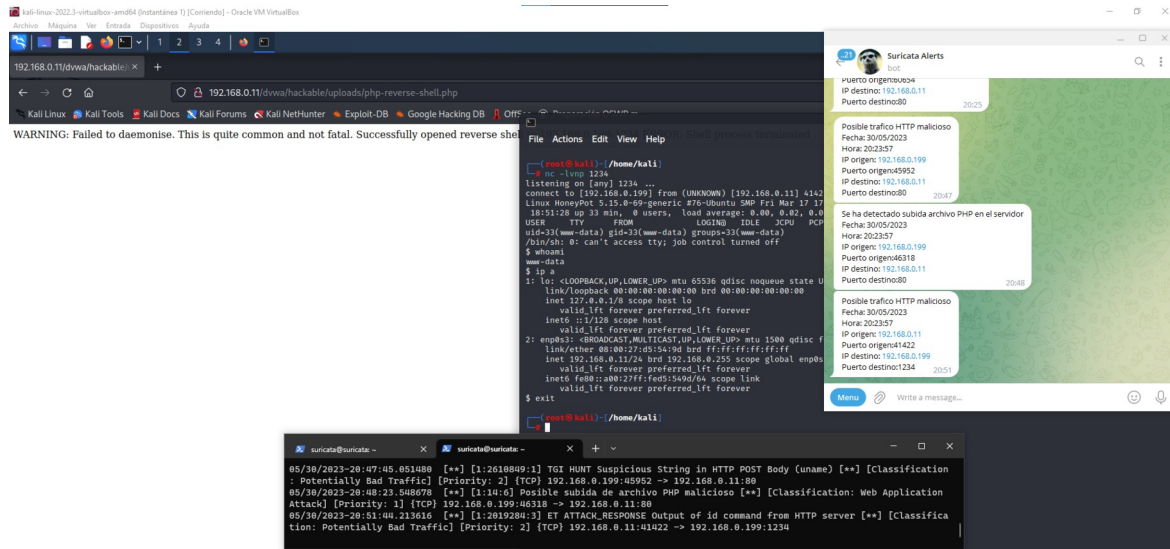


Ilustración 63: Detección de ejecución de comandos

Como podemos ver hemos conseguido entrar al servidor por el puerto 1234 y al salir Suricata ha detectado que se ha ejecutado el comando exit.

El contenido del archivo que activa la reverse shell es el siguiente:

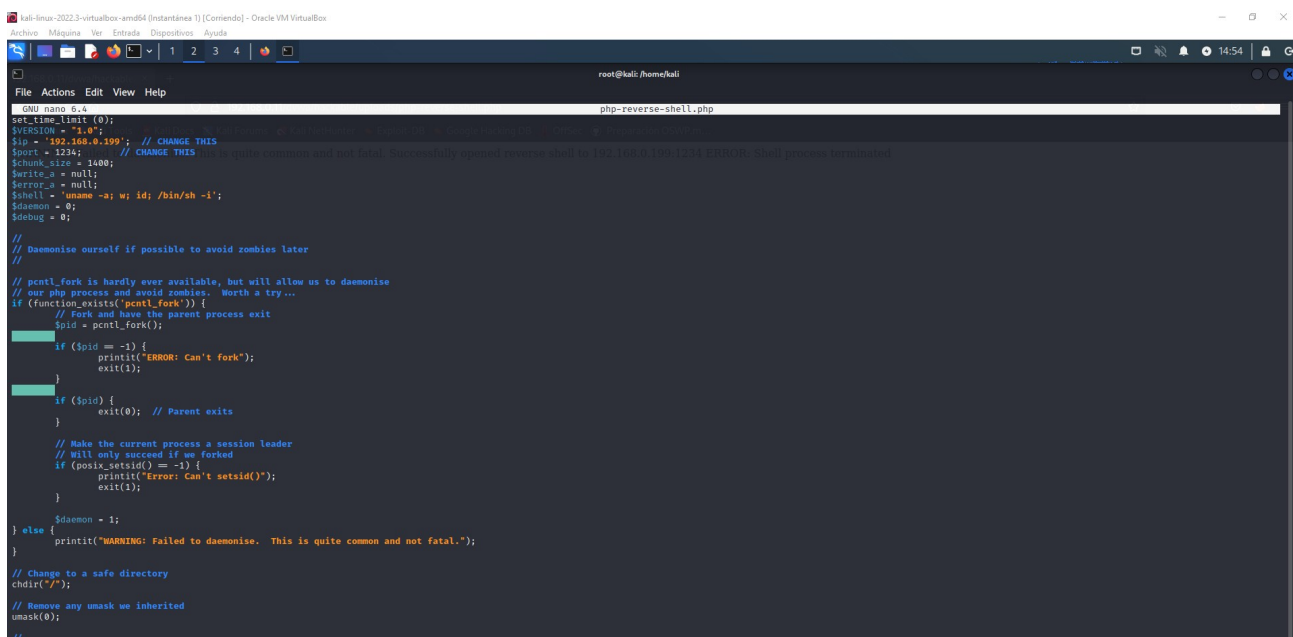


Ilustración 64: Contenido reverse shell

Si filtramos por IP en Splunk nos aparecerán todas las alertas enviadas por suricata por si necesitamos analizarlas y estudiarlas para tomar las medidas necesarias.

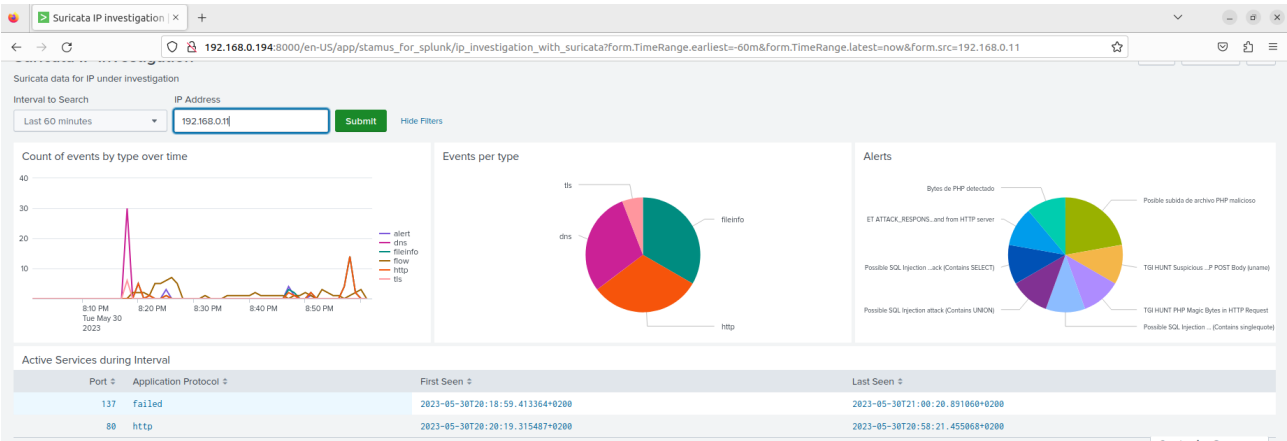


Ilustración 65: Resultados en Splunk



## 6. Conclusiones

Este proyecto he logrado alcanzar los objetivos planteados al inicio, el objetivo principal era el de crear un sistema capaz de detectar posibles amenazas en una red logrado gracias a el software de Suricata y Splunk.

A lo largo del desarrollo del proyecto he adquirido conocimientos en el campo de la ciberseguridad y gestión de sistemas para detectar algunos tipos de ciberataques, con soluciones efectivas para monitorear estos problemas que encontramos a la hora de lanzar un servidor a internet.

Las investigaciones realizadas y las soluciones implementadas son aplicables en entornos empresariales para contribuir en la seguridad de la red para prevenir costes provocados por ciberatacantes que comprometan la información de la empresa.

En el desarrollo de las diferentes tareas se han encontrado diversas dificultades. Sin embargo se han logrado superar. Esto me ha ayudado a aprender de ellos para fortalecer las habilidades de resolución de problemas y desarrollar una mentalidad analítica.

Como futuras mejoras considero que sería interesante aislar la máquina virtual que contiene el software de suricata para impedir que sea visible a la hora de realizar un escaneo con nmap, además de añadir más reglas de alertas para cubrir más ámbitos de la ciberseguridad.

El realizar un buen mantenimiento es importante revisar las reglas y cambiarlas en caso de que queden obsoletas, se deberá revisar que los datos de los registros se generan correctamente. El espacio del disco duro en Splunk no puede agotarse y puede dejar de almacenar los datos de los registros.

En resumen este proyecto me ha ayudado a adquirir conocimientos técnicos, desarrollar habilidades prácticas y contribuir al campo de la ciberseguridad.

## 7.Glosario

- **IDS:** Dispositivo de supervisión pasivo que detecta amenazas potenciales y genera alertas.
- **SIEM:** Dispositivo que se encarga de la administración de eventos e información de seguridad.
- **HoneyPot:** Es un sistema informático que se “sacrifica” para atraer ciberataques, como un señuelo.
- **Botnet:** Grupo de ordenadores o dispositivos que están bajo el control de un atacante, y que se usan para perpetrar actividades malintencionadas contra una víctima.
- **Port Forwarding:** Asignación o re-envío de puertos para transmitir información a través de una red. Esta técnica utiliza el protocolo TCP/IP, y se encarga de transmitir paquetes de información entre servidores externos a los servidores internos de una red particular.
- **SQL injection:** Tipo de vulnerabilidad en la que un atacante usa un trozo de código SQL (lenguaje de consulta estructurado) para manipular una base de datos y acceder a información potencialmente valiosa.
- **Reverse Shell:** proceso en el que la máquina de la víctima se conecta a la del atacante para recibir comandos.

## 8. Webgrafía

<https://keepcoding.io/blog>

<https://chat.openai.com>

<https://www.computing.es>

<https://www.splunk.com>

<https://openwebinars.net/blog/kali-linux-que-es-y-caracteristicas-principales/>

<https://tardesdesiem.com/splunk-y-suricata-siem-e-ids-pongamos-orden/>