Denegación de Servicio (DoS)

Capítulo destinado a acciones de respuesta ante ataques de denegación de servicio.

- Fail2ban para HTTP
- Análisis de registros posterior a un ataque DDoS

Fail2ban para HTTP

Anti - DoS con fail2ban (WebSites [ports:80/443])

Implementacion de anti-DoS con fail2ban en Sistema Operativo Debian 9

Instalar los paquetes “fail2ban” y “iptables-persistent”

```
~$ sudo apt-get install iptables-persistent fail2ban
```

Modificar la configuracion por defecto de fail2ban y agregar las siguientes lineas al archivo mencionado

```
~$ vim /etc/fail2ban/jail.conf
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
[http-get-dos]
enabled = true
port = http,https
filter = http-get-dos
# Cambiar a la ruta de los logs de Apache/Nginx/etc..
logpath = /var/log/*apache2*/*access.log
maxretry = 300
findtime = 300
#ban por 5 minutos
bantime = 600
action = iptables[name=HTTP, port=http, protocol=tcp]
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Crear un archivo en la ruta correspondiente con el siguiente contenido

```
~$ vim /etc/fail2ban/filter.d/http-get-dos.conf
```

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
# Fail2Ban archivo de configuracion
[Definition]
# Opcion: failregex
# Se debe configurar el maxretry y findtime en jail.conf muy cuidadosamente para evitar los falsos positivos.
failregex = ^<HOST> -.*"(GET|POST).*
# Optcion: ignoreregex (El IP de este campo no sera bloqueado)
ignoreregex =
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
```

Reiniciar el servicio de fail2ban

```
~$ sudo systemctl restart fail2ban.service
```

Para revisar las IP bloqueados

```
~$ sudo iptables -nvL
```

Remover IP “banneada”

```
//Desplegara las reglas que creamos en este caso seria http-get-dos
~$ sudo fail2ban-client status
//Elimina la IP "banneada"
~$ sudo fail2ban-client set http-get-dos unbanip <IP_ADDRESS>
```

Análisis de registros posterior a un ataque DDoS

El sitio web FalconFeeds.io contiene información sobre múltiples actores de amenaza. Es posible buscar información para recolectar datos sobre actores maliciosos.

Degradación de los servicios.

Verificar la página de Uptime Kuma para concluir, en base a los registros, el tiempo de respuesta de los sitios web afectados. En caso de caída del sitio, verificar el tiempo que se mantuvo fuera de línea.

Prevención interna.

- Configuración y mejora de los mecanismos de respuesta ante la detección de actividad de red inusual.
- Verificación de la configuración del firewall disponible ante la respuesta ante ataques de tipo SYN DDoS (ataque de denegación de servicio que explota el proceso de establecimiento de conexiones TCP).

Análisis con Goaccess.

Se ejecuta el comando "goaccess access.log" en la terminal para generar un informe con las solicitudes realizadas al servidor.

Con base a los resultados obtenidos con la herramienta

- Se revisan las métricas presentadas por GoAccess, incluyendo cantidad de solicitudes, IPs más activas y horas con mayor tráfico.
- Se utiliza cat access.log para visualizar el contenido del archivo de registros en los horarios que se haya tenido mas actividad, separando primero horas luego minutos y por ultimo si es necesario por segundos.
- Se aplica grep para filtrar líneas relevantes que contienen información sobre las solicitudes más frecuentes en determinadas horas.
- Se usa cut para extraer las direcciones IP de las solicitudes en los períodos de mayor actividad.

Obtención de las IPs en las horas de mayor tráfico

- Se ejecuta un pipeline con cat, grep, y cut para procesar los datos y obtener las IPs más activas durante los picos de tráfico.
- Se genera un listado de direcciones IP correspondiente a las horas con mayor número de solicitudes.

Análisis y posible uso de datos

- Se analizan las IPs extraídas para identificar posibles patrones, ataques o actividad inusual en el servidor.
- Se pueden utilizar estos datos para aplicar bloqueos, realizar auditorías o mejorar la seguridad del sistema.

Análisis con Wazuh Manager.

Realizar un laboratorio de prueba para simular la carga de registros en un servidor de prueba, descartando los registros de información y seleccionando alertas de nivel medio y alto.

- Utilizar un script de simulación de carga de registros en el mismo directorio donde se almacenan los registros a analizar.

```
nano simulador.sh
```

```
#!/bin/bash
#Archivo de entrada al script que contiene los registros del objetivo del ataque
input_file="access.log"
#Archivo de salida monitorizado por Wazuh para la generación de alertas
output_file="archivoaccess.log"
# Verifica si el archivo de salida existe y, si es así, lo elimina
# para evitar duplicar los registros o mezclarlos con anteriores
if [ -f "$output_file" ]; then
    rm "$output_file"
fi
# Lee el archivo línea por línea
while IFS= read -r line
do
    # Simula la recepción de logs añadiendo un retraso de 90 ms
    # para no sobrepasar el humbral de deteccion de eventos que tiene wazuh manager
    sleep 0.09
    # Escribe la línea en el archivo de salida
    echo "$line" >> "$output_file"
```

```
    echo "Log añadido: $line"
done < "$input_file"
# Registro de la cantidad de lineas copiadas mientras se ejecuta el script
echo "El archivo ha sido copiado línea por línea a $output_file"
```

- Añadir permisos de ejecución del archivo.

```
sudo chmod +x simulador.sh
```

- Añadir al archivo "ossec.conf" la configuración para el análisis del archivo de salida del script y que las alertas se generen en base a este comportamiento.

```
<ossec_config>
  <localfile>
    <log_format>syslog</log_format>
    <location>/home/wazuh/monitoreo/archivoaccess.log</location>
  </localfile>
</ossec_config>
```

- Reiniciar el servicio de Wazuh manager o Wazuh agent que se esté utilizando para aplicar los cambios realizados.

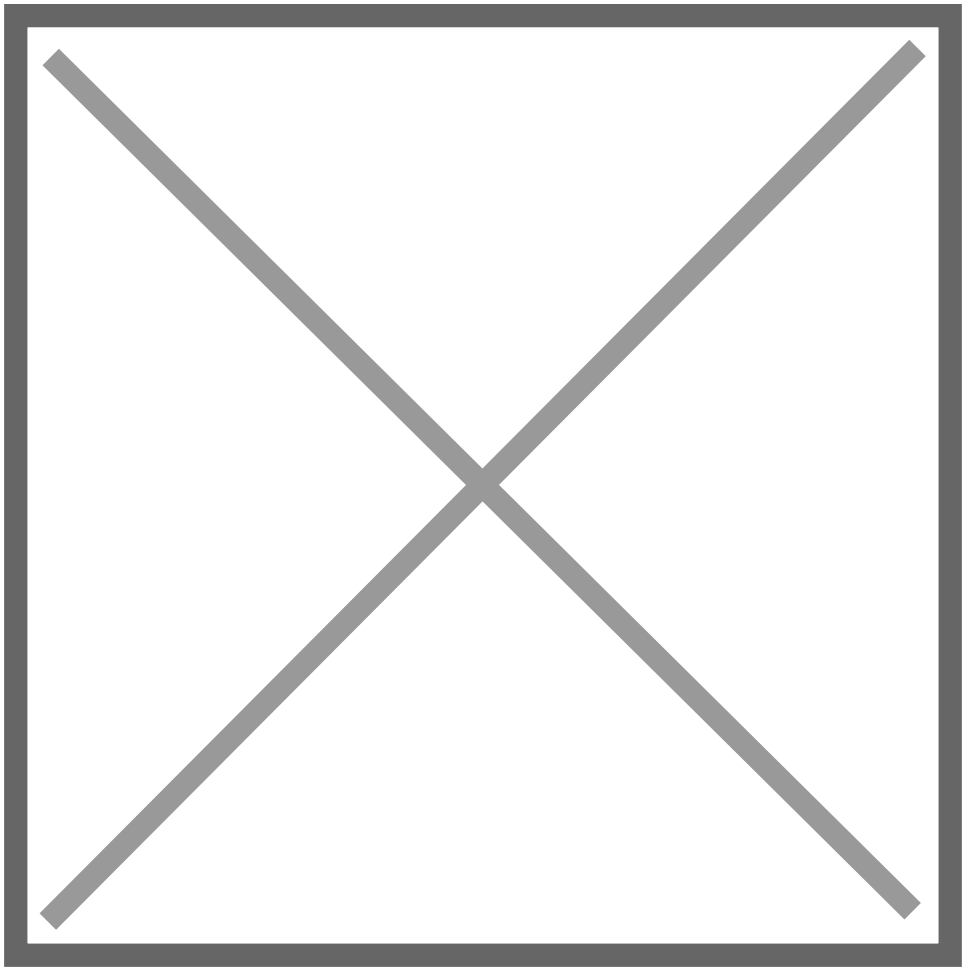
```
sudo systemctl restart wazuh-manager
```

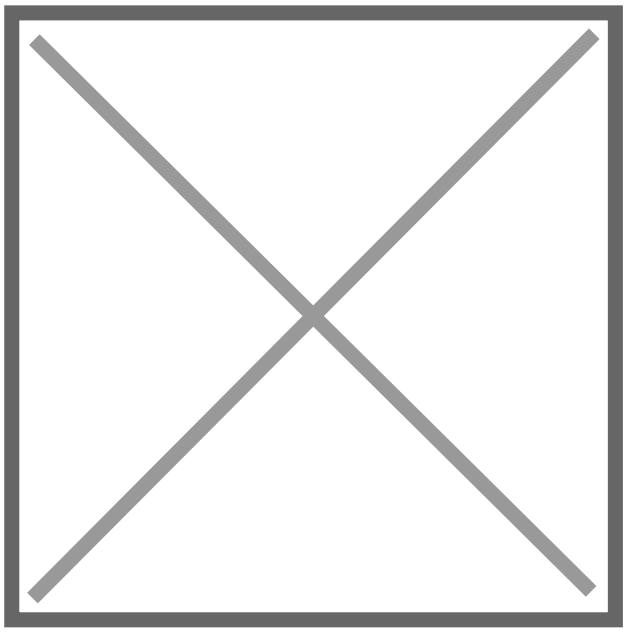
```
sudo systemctl restart wazuh-agent
```

- Ejecutar el script de simulación para observar las alertas.

```
./simulador.sh
```

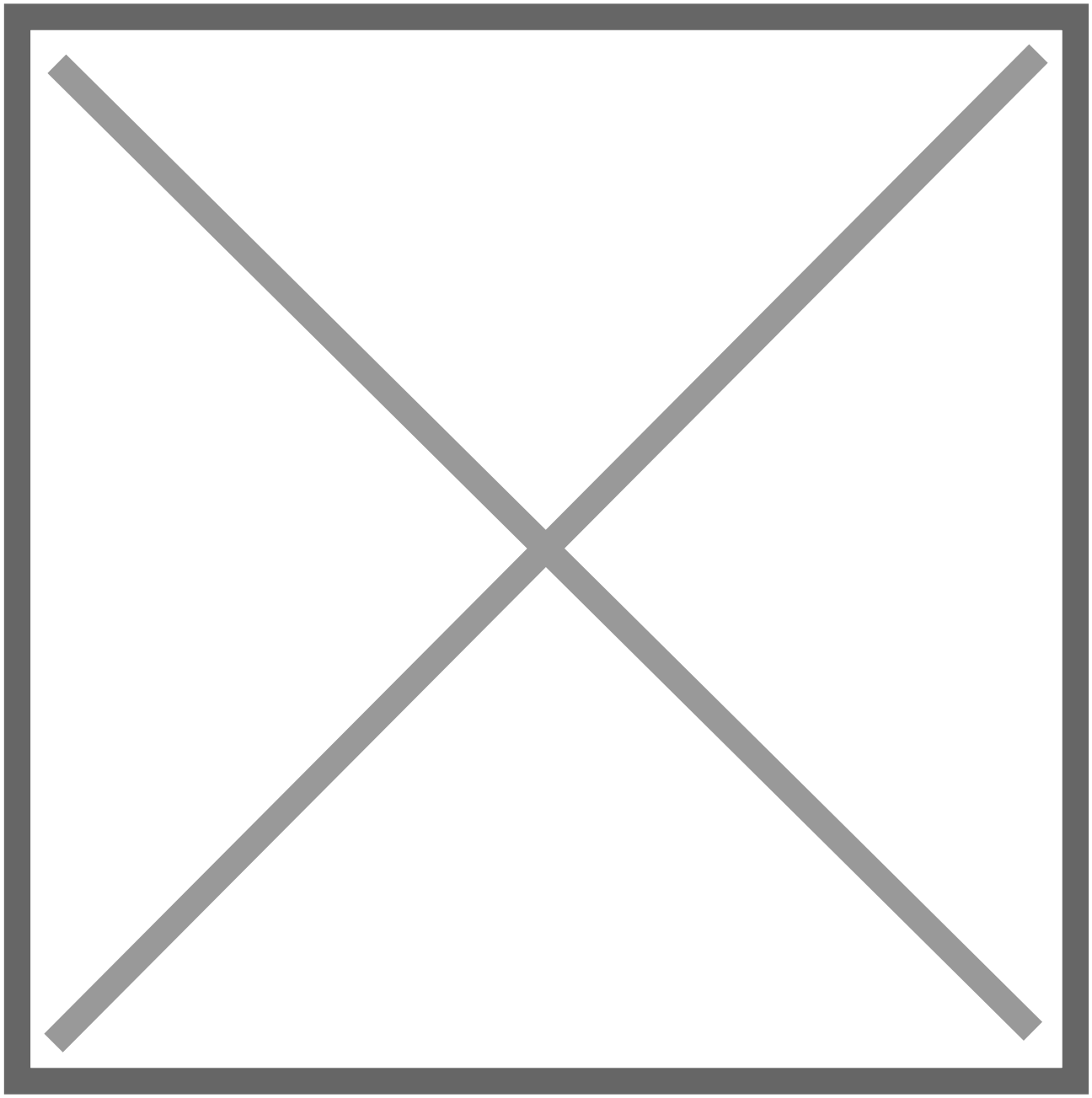
- Analizar las alertas en base a los resultados





Análisis de resultados con la plataforma MISP.

Con la lista de direcciones IP únicas, agruparlas de 1000 en 1000 para poder copiar la lista y observar qué direcciones IP tienen correlación con otro evento dentro de la región.



Distribución de la información.

Extraer de los archivos de registro las direcciones IP involucradas y crear un evento con nivel alto de peligrosidad dentro de la plataforma MISP, indicando todos los indicadores de compromiso para proporcionar la información a las instituciones.

Identificación de los países implicados.

Descargamos la base de datos de IP's y países disponible en github con el nombre de "GeoLite2-Country.mmdb"

- https://github.com/P3TERX/GeoLite.mmdb

Iniciamos y activamos el entorno virtual para iniciar el programa "geoip2" dentro del mismo directorio de la base de datos descargada.

```
python3 -m venv myenv
source myenv/bin/activate
```

Instalamos dentro del entorno virtual la herramienta.

```
pip install geoip2
```

Utilizamos un script en python para el uso de esta base de datos con un archivo de direcciones IP identificadas.

```
import csv
import geoip2.database
from collections import Counter

# Archivos de entrada y salida
archivo_entrada = "entrada.csv"
archivo_salida_temp = "salida.csv"
archivo_salida_final = "conteo_paises.csv"

# Cargar la base de datos GeoIP
ruta_db = "GeoLite2-Country.mmdb"
lector = geoip2.database.Reader(ruta_db)

def obtener_pais(ip):
    try:
        respuesta = lector.country(ip)
        return respuesta.country.name
    except geoip2.errors.AddressNotFoundError:
        return "IP no encontrada"
    except Exception as e:
```

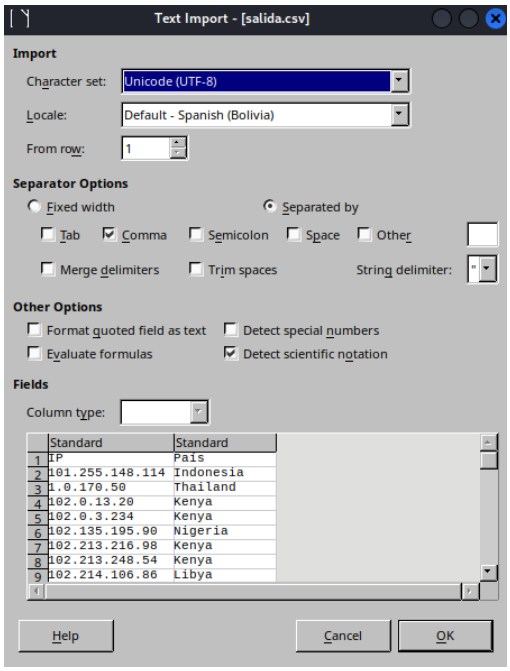
```
        return f"Error: {e}"

# Leer IPs desde entrada.csv y procesarlas
resultados = []
conteo_paises = Counter()

total_ips = 0
with open(archivo_entrada, "r", encoding="utf-8") as archivo:
    lector_csv = csv.reader(archivo)
    next(lector_csv, None)  # Saltar la cabecera si existe

    for fila in lector_csv:
        if not fila:  # Evitar líneas vacías
            continue
        ip = fila[0]
        pais = obtener_pais(ip)
        resultados.append((ip, pais))
        conteo_paises[pais] += 1
        total_ips += 1

# Guardar resultados en salida.csv
with open(archivo_salida_temp, "w", newline="", encoding="utf-8") as archivo:
    escritor = csv.writer(archivo)
    escritor.writerow(["IP", "País"])
    escritor.writerows(resultados)

# Guardar conteo de países en conteo_paises.csv
with open(archivo_salida_final, "w", newline="", encoding="utf-8") as archivo:
    escritor = csv.writer(archivo)
    escritor.writerow(["IP", "País", "Conteo"])
    for ip, pais in resultados:
        escritor.writerow([ip, pais, conteo_paises[pais]])

# Cerrar la base de datos
lector.close()

print(f"Proceso completado. Se procesaron {total_ips} IPs.")
```

```
print("Revisa los archivos 'salida.csv' y 'conteo_paises.csv'.")
```

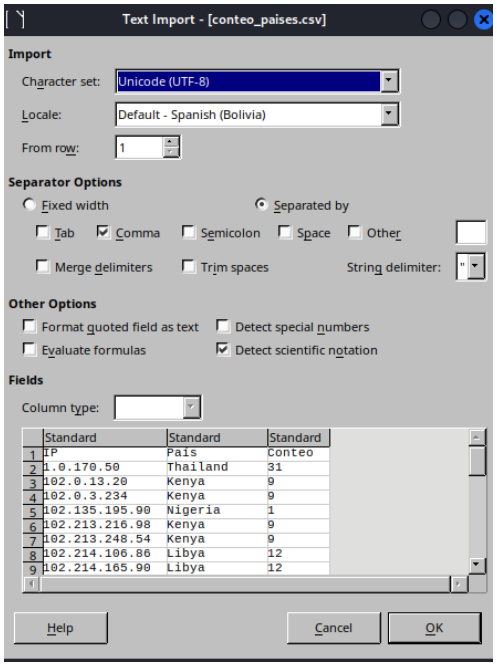
Ejecutamos el script con el archivo de entrada:

```
python geo.py
```

La salida del archivo "salida.csv" tiene las direcciones IP acompañadas del país de origen:



Mientras que la salida del archivo "conteo_paises.csv" tiene las direcciones IP, el país de origen y la cantidad de solicitudes que realizó cada país, se toman en cuenta solamente las direcciones IP las cuales realizaron más de 1000 solicitudes al servidor.



Con esta información se procede a comunicar a los países origen mediante una investigación en la plataforma RTIR.