Pruebas de concepto (PDC)

PDC 1: BLOQUEO DE DIRECCIÓN IP MALICIOSA CONOCIDA.

Se configurará servidor web Apache en Ubuntu y se intenta acceder a ellos desde un terminal Debian que simulara a un atacante con una IP con baja reputación en listas de IP maliciosas.

TERMINAL	***DESCRIPCIÓN***
Debian	Terminal del atacante que se conecta al servidor web de la víctima en el que usa la capacidad de la lista CDB de Wazuh para marcar su dirección IP como maliciosa.
Ubuntu	Terminal de la víctima que ejecuta un servidor web Apache 2.4.54. Se utiliza el módulo de respuesta activa de Wazuh para bloquear automáticamente las conexiones desde el terminal del atacante.

Configuración.

En Wazuh Manager.

Se debe agregar la dirección IP del extremo de Debian a una lista negra de usuarios peligrosos para la red y luego configurar las reglas y la respuesta activa.

En la guía de procedimientos de prueba de Wazuh manager se encuentran pasos de instalación del comando:

```
sudo apt update && sudo apt install -y wget
```

Descargue la base de datos de reputación de IP de Alienvault:

```
sudo wget https://raw.githubusercontent.com/firehol/blocklist-ipsets/master/alienvault_reputation.ipset -O
/var/ossec/etc/lists/alienvault_reputation.ipset
```

Agregamos la dirección IP del extremo del atacante (Debian) a la base de datos de reputación de IP. La dirección IP de Debian en el ejemplo es (192.168.24.73).

```
sudo echo 192.168.24.73 >> /var/ossec/etc/lists/alienvault_reputation.ipset
```

Descargamos el siguiente script para convertir del formato “.ipset” (formato utilizado para crear conjuntos de direcciones IP que se pueden usar como tablas para comparación) al formato “.cdb” (formato de lista que utiliza wazuh enlistar las direcciones maliciosas para su posterior bloqueo).

```
sudo wget https://wazuh.com/resources/iplist-to-cdblist.py -O /tmp/iplist-to-cdblist.py
```

Convertimos el archivo “alienvault_reputation.ipset” a un formato “.cdb” usando el script descargado previamente:

```
sudo /var/ossec/framework/python/bin/python3 /tmp/iplist-to-cdblist.py
/var/ossec/etc/lists/alienvault_reputation.ipset /var/ossec/etc/lists/blacklist-alienvault
```

Eliminamos “alienvault_reputation.ipset” y el script “iplist-to-cdblist.py”, para que no sean usados erronea o de forma maliciosa.

```
sudo rm -rf /var/ossec/etc/lists/alienvault_reputation.ipset
sudo rm -rf /tmp/iplist-to-cdblist.py
```

Asignamos los permisos y la propiedad correctos al archivo generado:

```
sudo chown wazuh:wazuh /var/ossec/etc/lists/blacklist-alienvault
```

La siguiente regla personalizada se utiliza para para activar una secuencia de comandos de respuesta activa de Wazuh cuando se detecte un evento en el cual se haya reconocido una dirección IP maliciosa realizando intercambio de información con puntos finales monitoreados por Wazuh manager, para esto ingresamos al archivo de reglas personalizadas de Wazuh.

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

En el archivo de conjunto de reglas personalizado del servidor Wazuh:

```
<group name="attack,">
  <rule id="100100" level="11">
    <if_group>web|attack|attacks</if_group>
    <list field="srcip" lookup="address_match_key">etc/lists/blacklist-alienvault</list>
```

```
    <description>IP address found in AlienVault reputation database.</description>
  </rule>
</group>
```

Editamos el archivo de configuración del servidor Wazuh.

```
sudo nano /var/ossec/etc/ossec.conf
```

Para que se pueda comparar con la lista de direcciones de alienvault agregamos la lista creada “etc/lists/blacklist-alienvault” a la sección “<ruleset>”.

```
<ossec_config>
  <ruleset>
    <!-- Default ruleset -->
    <decoder_dir>ruleset/decoders</decoder_dir>
    <rule_dir>ruleset/rules</rule_dir>
    <rule_exclude>0215-policy_rules.xml</rule_exclude>
    <list>etc/lists/audit-keys</list>
    <list>etc/lists/amazon/aws-eventnames</list>
    <list>etc/lists/security-eventchannel</list>
    <list>etc/lists/blacklist-alienvault</list>
    <!-- User-defined ruleset -->
    <decoder_dir>etc/decoders</decoder_dir>
    <rule_dir>etc/rules</rule_dir>
  </ruleset>
</ossec_config>
```

Agrega una respuesta activa al wazuh manager (con permisos de root):

```
sudo nano /var/ossec/etc/ossec.conf
```

Se debe quitar el símbolo de comentario al apartado <active-response> y escribir de acuerdo al ID de la regla creada en este caso 100100.

```
  <active-response>
      <command>firewall-drop</command>
      <location>local</location>
      <rules_id>100100</rules_id>
```

```
</active-response>
```

En el terminal víctima.

Se instala un servidor web con Apache para las pruebas, este tendrá un agente que constantemente está realizando el monitoreo de los registros de acceso del servidor Apache.

Configuramos el agente de Wazuh:

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/apache2/access.log</location>
</localfile>
```

Emulación del ataque.

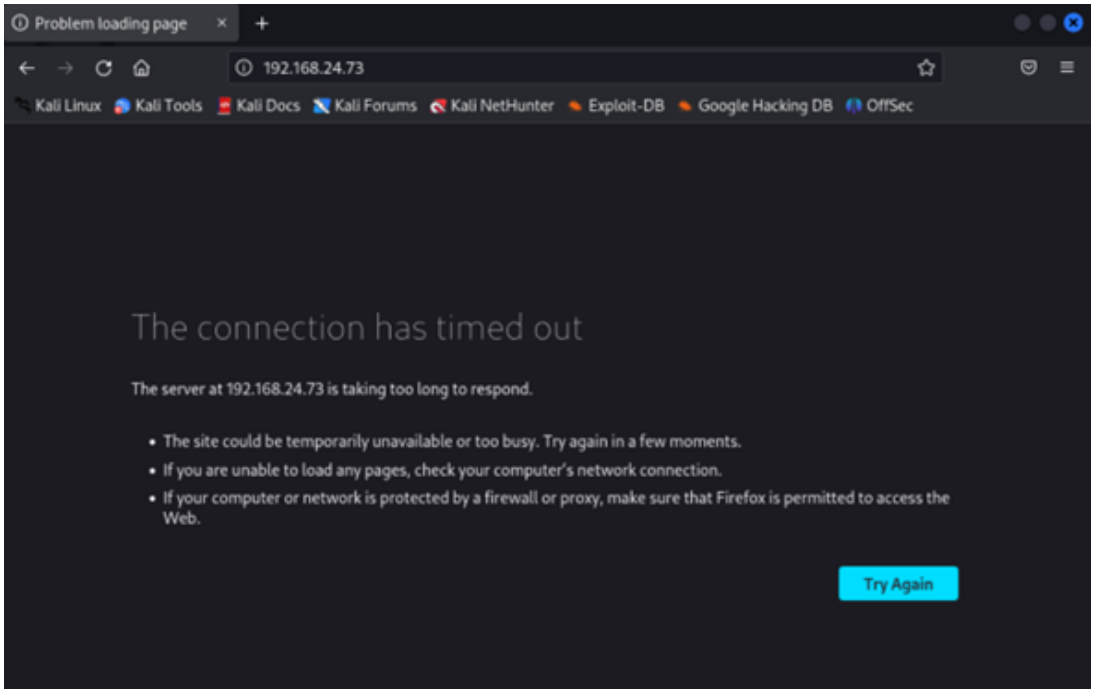
Para la emulación de ataque se debe acceder al servidor web desde el extremo Kali utilizando la dirección IP correspondiente:

Wazuh manager identificara la dirección

IP de conexión para ser analizada y al coincidir con la dirección IP de la base de datos de reputación enviara la alerta para activar la respuesta inmediata bloqueando la dirección IP con el número de ID de alerta 100100 que es el creado manualmente para el servidor.

Resultados.

Posteriormente Wazuh manager bloquea el intento de acceso y el servidor se hace inaccesible por parte del atacante.



PDC 2: MONITOREO DE INTEGRIDAD DE ARCHIVOS.

Wazuh tiene un módulo FIM (File Integrity Monitoring) el cual es un proceso de seguridad que se utiliza para monitorear la integridad de los archivos del sistema se revisan los cambios en el sistema de

archivos para detectar la creación, modificación y eliminación de archivos.

Configuración.

Para esta prueba utilizaremos Debian, el módulo FIM de Wazuh supervisa un directorio (/root) para detectar la creación, los cambios y la eliminación de archivos.

Se debe editar el archivo de configuración agregando dentro del bloque <syscheck> directorios para ser monitoreados (se puede configurar cualquier ruta en específico).

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<directories check_all="yes" report_changes="yes" realtime="yes">/root</directories>
```

Reiniciamos para aplicar los cambios.

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

Emulación del ataque.

En el ámbito de la seguridad informática existen directorios que no deben ser modificados ya que contienen configuraciones para el funcionamiento correcto de los programas ó en su defecto la copia de archivos maliciosos que contienen malware, por lo cual la prueba consiste en crear un archivo de texto en el directorio monitoreado.

```
cd /root/
touch archivo.txt
```

Agregue contenido al archivo de texto y guárdelo.

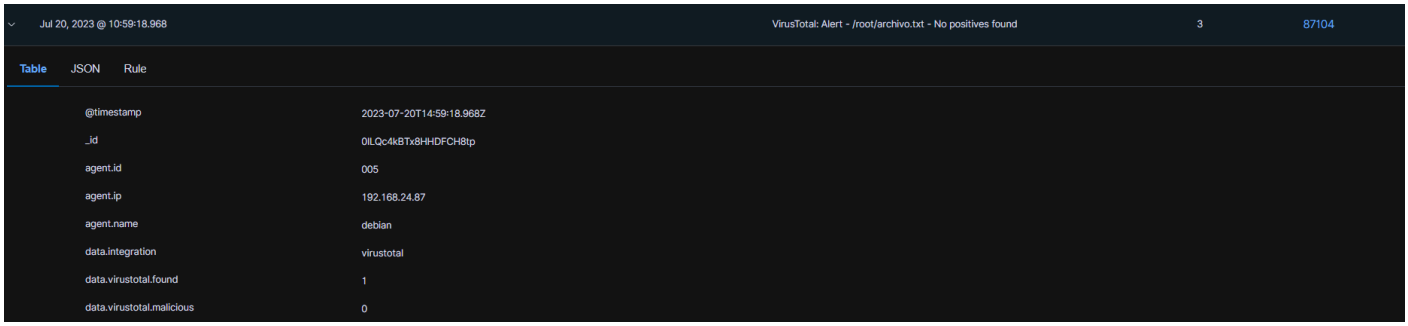
```
echo "contenido de archivo" >> archivo.txt
```

Elimine el archivo de texto del directorio supervisado.

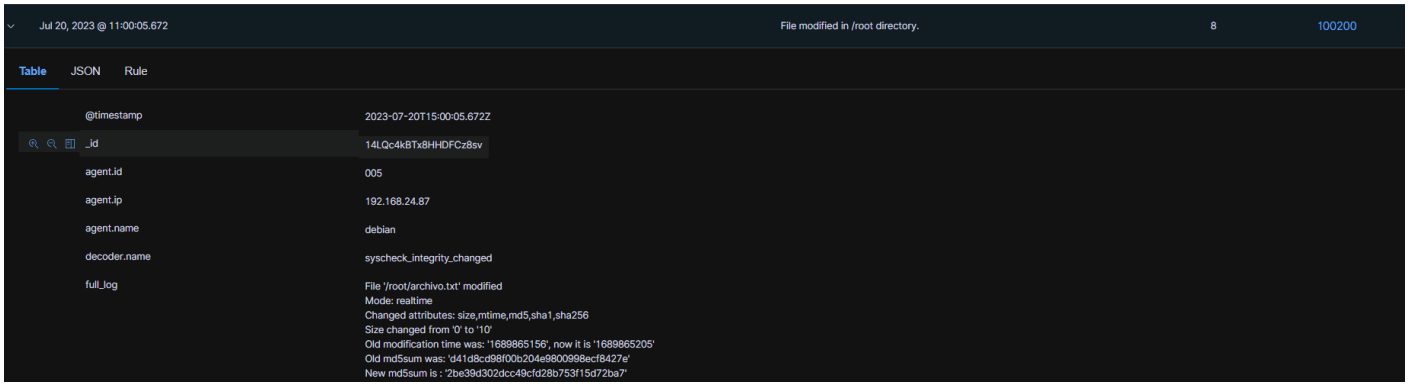
```
rm archivo.txt
```

Resultados.

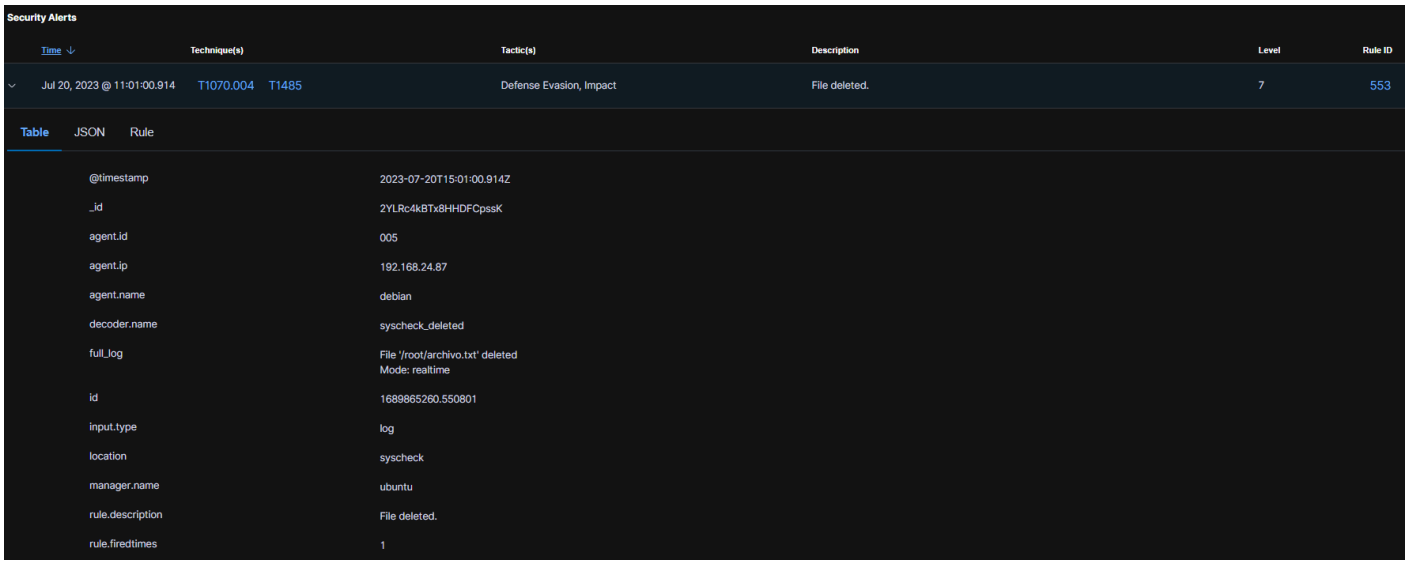
Con la creación del archivo se genera una alerta de nivel 3 ya que el archivo esta vacío.



La modificación del archivo genera una alerta de nivel 8 al ser un directorio importante para ser monitorizado.



Por ultimo se genera la alerta de eliminación del archivo de nivel 7.



PDC 3: ATAQUE DE FUERZA BRUTA.

TERMINAL	DESCRIPCIÓN

Debian	Terminal del atacante que realiza ataques de fuerza bruta. Con SSH instalado.
Ubuntu	Terminal víctima de ataques de fuerza bruta SSH. Es necesario tener un servidor SSH instalado y habilitado en este punto final.

Configuración:

Realice los siguientes pasos para configurar el terminal de Debian. Esto permite realizar intentos de falla de autenticación en la víctima Ubuntu.

En Debian instalar Hydra para utilizar un ataque de fuerza bruta mediante el protocolo ssh.

```
sudo apt update
sudo apt install -y hydra
```

Emulación del ataque

Crearemos un archivo de texto con 10 o más contraseñas aleatorias, esto simulara un diccionario de contraseñas de los cuales existen en la red que puede contener millones de contraseñas frecuentes para romper la seguridad mediante fuerza bruta.

```
sudo nano PASSWD_LIST.txt
```

```
123456
Password
Passwd
123
Root
Admin
Admin123
Root123
Server
Linux
```

Guardamos este archivo como nuestro diccionario limitado de contraseñas y configuramos el terminal victima con una de estas contraseñas inseguras.

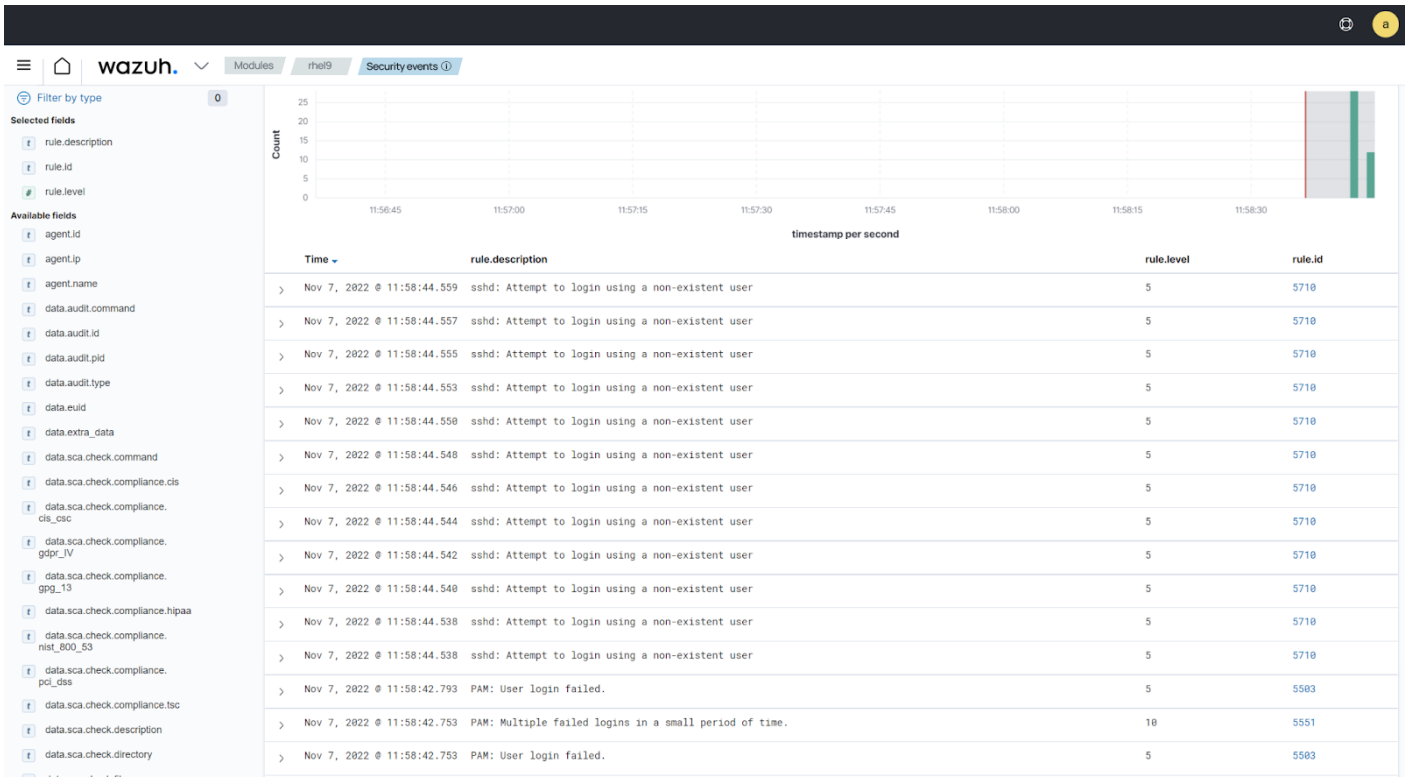
Por último ejecutamos "Hydra" desde el terminal atacante para realizar ataques de fuerza bruta contra el extremo de la víctima.

La sintaxis del comando es la siguiente tomando en cuenta **xubuntu** como nombre de usuario, PASSWD_LIST.txt como el diccionario de contraseñas, **192.168.24.72** la direccion IP de la víctima y por ultimo el protocolo de acceso que será **ssh**.

```
sudo hydra -l xubuntu -P PASSWD_LIST.txt 192.168.24.72 ssh
```

Resultados.

Visualización de las alertas en Wazuh manager:
Se pueden visualizar los datos de alerta en el panel de control de Wazuh, se debe acceder al módulo Eventos de seguridad para consultar las alertas.



PDC 4: DETECCIÓN DE PROCESOS NO AUTORIZADOS.

En este caso de uso, utiliza la capacidad de monitoreo de comandos de Wazuh para detectar cuándo se está ejecutando Netcat en un punto final (Netcat es una utilidad de red informática que se utiliza para escanear y escuchar puertos).

En el terminal víctima modificamos el modulo de monitoreo desde el agente de Wazuh, de esta forma podremos detectar un proceso de Netcat en ejecución.

Configuración.

Agregamos la siguiente configuración al archivo de configuración del agente de Wazuh. Nos permite obtener periódicamente una lista de procesos en ejecución:

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<ossec_config>
  <localfile>
    <log_format>full_command</log_format>
    <alias>process list</alias>
    <command>ps -e -o pid,uname,command</command>
    <frequency>30</frequency>
  </localfile>
</ossec_config>
```

Reiniciar el wazuh-agent para aplicar los cambios.

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

Posteriormente instalamos Netcat para realizar las pruebas.

```
sudo apt install ncat nmap -y
```

Servidor de Wazuh.

Se configuran los siguientes pasos para crear una regla que se active cada vez que se inicie el programa Netcat.
Agregando las siguientes reglas al archivo local de reglas en el servidor:

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

```
<group name="ossec,">
  <rule id="100050" level="0">
    <if_sid>530</if_sid>
    <match>^ossec: output: 'process list'</match>
    <description>Lista de procesos en ejecución </description>
    <group>process_monitor,</group>
  </rule>
```

```
  <rule id="100051" level="7" ignore="900">
    <if_sid>100050</if_sid>
    <match>nc -l</match>
    <description>netcat esta escuchando las conexiones entrantes.</description>
    <group>process_monitor,</group>
  </rule>
</group>
```

Reiniciamos el servidor para aplicar cambios.

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

Emulación de ataque.

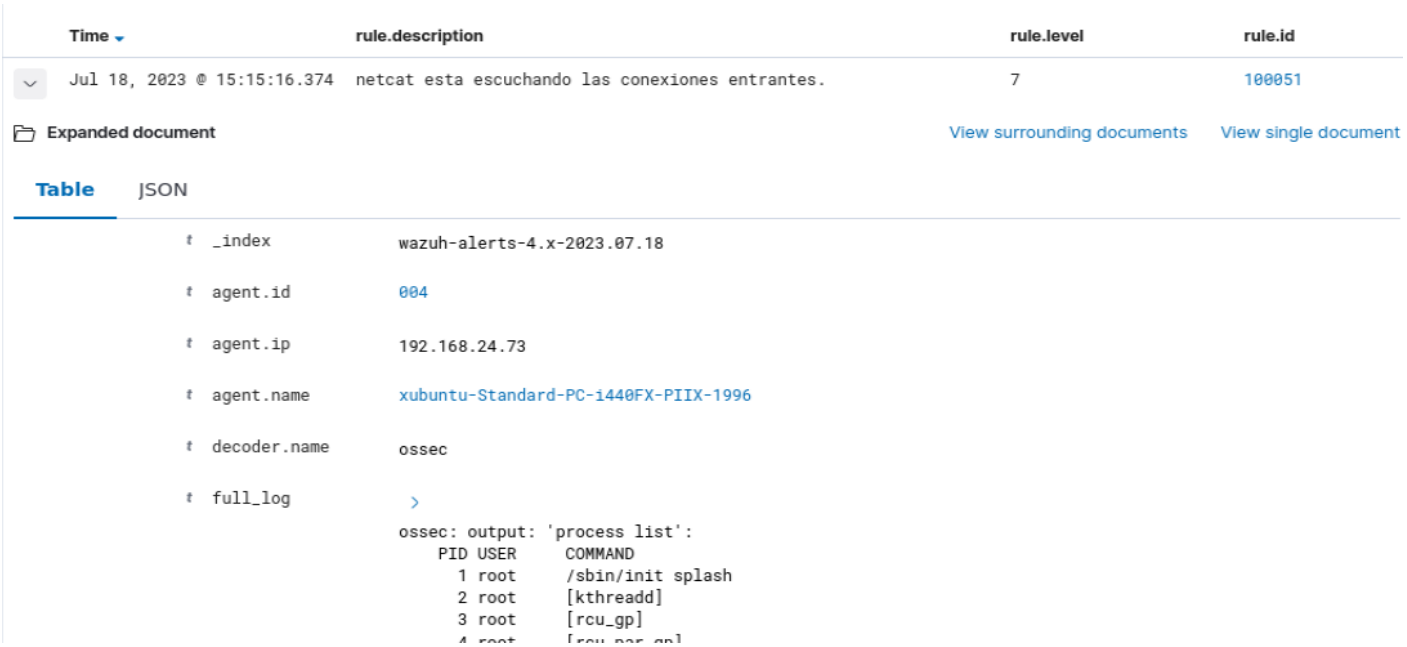
Ejecutamos en el terminal el siguiente comando durante aproximadamente 30 segundos.

```
nc -l 8000
```

Este comando utiliza para crear un servidor en el **puerto 8000**, **-l** indica que el comando debe escuchar en lugar de conectarse a un puerto. El servidor creado escuchará en el puerto 8000 y esperará conexiones entrantes. Si se recibe una conexión, el comando nc enviará los datos recibidos a la salida estándar.

Resultados.

El servidor de Wazuh manager nos mostrara el siguiente evento:



PDC 5:INTEGRACIÓN DE IDS DE RED.

La integración de Suricata con Wazuh mejora la detección y respuesta a las amenazas en los sistemas monitoreados.
Para esta prueba utilizaremos Debian donde se instalara Suricata, con esta integración se analiza el trafico de red generado en este terminal.

Configuración

Realizamos la instalación de Suricata en Debian.

```
sudo add-apt-repository ppa:oisf/suricata-stable
sudo apt-get update
sudo apt-get install suricata -y
```

Seguidamente descargamos y extraemos el conjunto de reglas de *Emerging Threats Suricata* el cual es un conjunto de reglas de detección de intrusiones que se utiliza para monitorear el tráfico de red en busca de cualquier actividad maliciosa, violaciones de políticas y amenazas que hará mas completo el monitoreo de este terminal.

```
cd /tmp/ && curl -LO https://rules.emergingthreats.net/open/suricata-6.0.8/emerging.rules.tar.gz
sudo tar -xvzf emerging.rules.tar.gz && sudo mv rules/*.rules /etc/suricata/rules/
sudo chmod 640 /etc/suricata/rules/*.rules
```

Una vez extraídos se debe modificar la configuración en el archivo “.yaml”

```
sudo nano /etc/suricata/suricata.yaml
```

```
HOME_NET: "192.168.24.87"
EXTERNAL_NET: "any"
default-rule-path: /etc/suricata/rules
rule-files:
- "*.rules"
# Global stats configuration
stats:
  enabled: no
# Linux high speed capture support
af-packet:
- interface: enp0s18
```

El nombre de la interface se obtiene con el comando:

```
ip address show
```

Reiniciar el servicio de Suricata.

```
sudo systemctl restart suricata
```

Para que el agente de Wazuh pueda leer el archivo de registros de Suricata se debe configurar el ossec.conf.

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<ossec_config>
  <localfile>
    <log_format>json</log_format>
    <location>/var/log/suricata/eve.json</location>
  </localfile>
</ossec_config>
```

Para aplicar cambios se reinicia el agente de Wazuh.

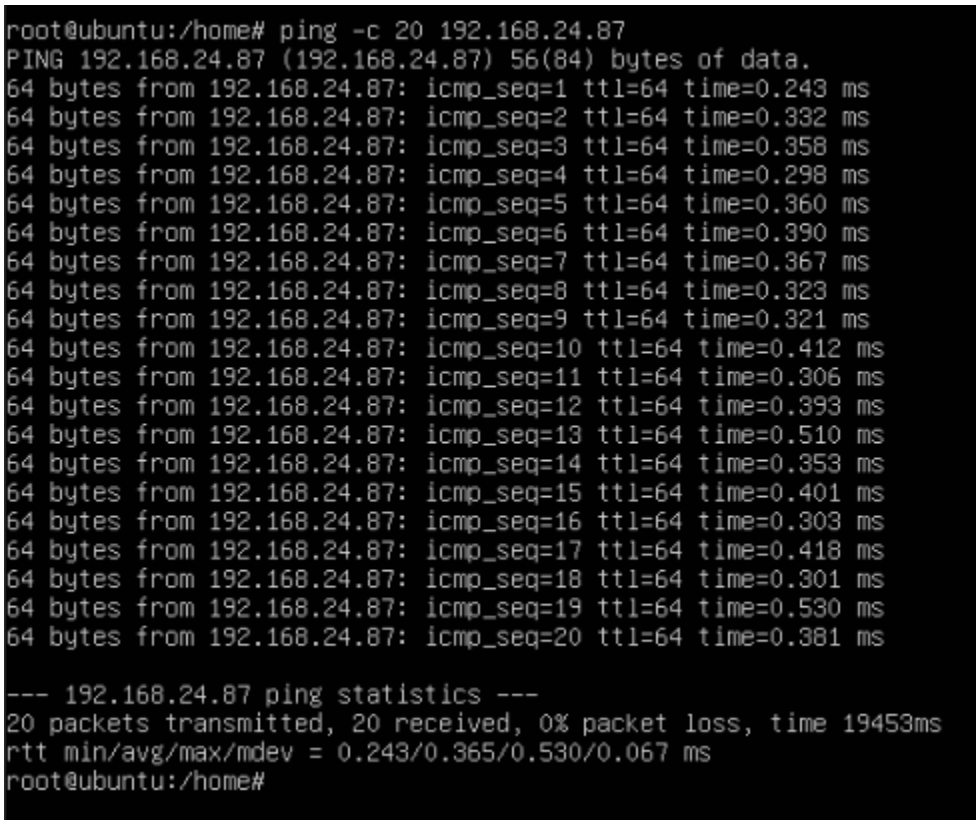
```
sudo systemctl restart wazuh-agent
```

Emulación del ataque.

Wazuh analiza automáticamente los datos de los logs que se encuentran en /var/log/suricata/eve.json y genera alertas relacionadas en el dashboard.

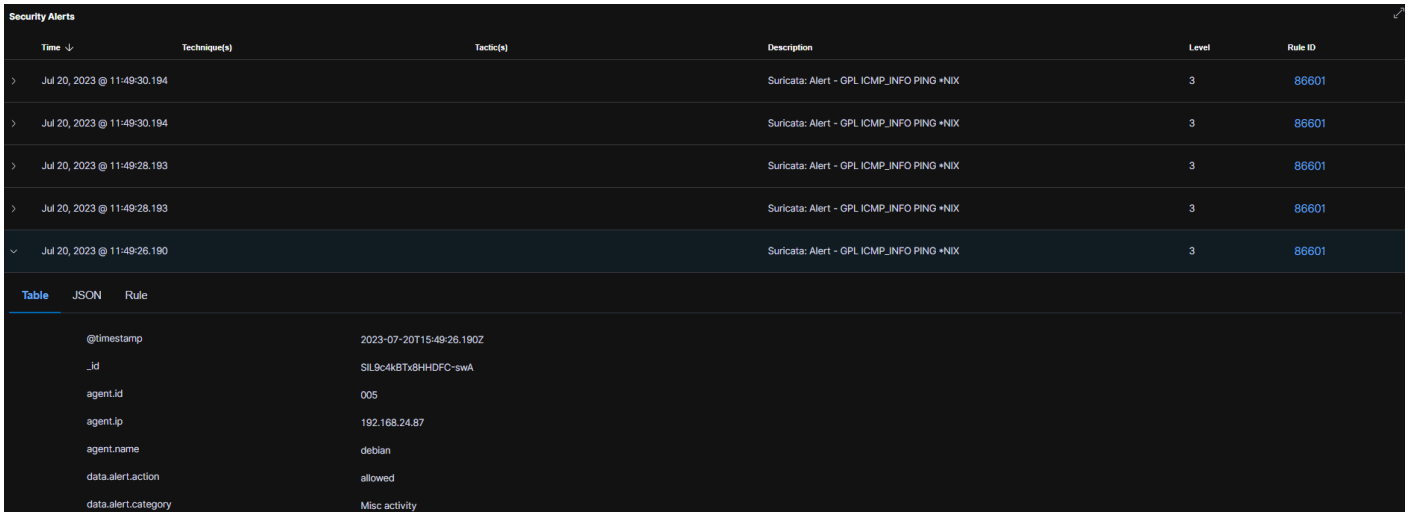
Hacemos una prueba de ping a la dirección IP del terminal Debian desde el servidor de Wazuh:

```
ping -c 20 192.168.24.87
```



Resultados.

Se genera la alerta de Suricata que se resume en el dashboard de Wazuh-Manager como ICMP proveniente de la dirección IP del atacante.



PDC 6: DETECCIÓN DE INYECCIÓN SQL.

TERMINAL	DESCRIPCIÓN
Debian	Terminal atacante con payload de inyección SQL
Ubuntu	Terminal víctima con servidor web Apache

Configuración.

En la víctima se debe realizar los siguientes pasos para instalar Apache y configurar el agente de Wazuh para monitorear los registros de Apache.

```
sudo apt update
sudo apt install apache2
```

Si el Firewall está habilitado, se debe modificar para permitir el acceso externo a los puertos web. Omita este paso si el Firewall está deshabilitado.

```
sudo ufw app list
sudo ufw allow 'Apache'
sudo ufw status
```

Verificar el estado del servicio Apache para ver que el servidor web se está ejecutando:

```
sudo systemctl status apache2
```

Utilizar en consola curl o en su defecto abrir http://<UBUNTU_IP> -> http://192.168.24.73 en un navegador para ver la página de inicio de Apache y verificar la ejecución:

```
curl http://<UBUNTU_IP>
```

En la parte del agente Wazuh agregue las siguientes líneas del archivo de configuración.

```
nano /var/ossec/etc/ossec.conf
```

El siguiente comando permite al agente de Wazuh monitorear los registros de acceso de su servidor Apache y enviar registros continuamente a Wazuh manager:

```
<ossec_config>
  <localfile>
    <log_format>apache</log_format>
    <location>/var/log/apache2/access.log</location>
  </localfile>
</ossec_config>
```

Reinicie Wazuh agent para aplicar los cambios de configuración:

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

Emulación del ataque.

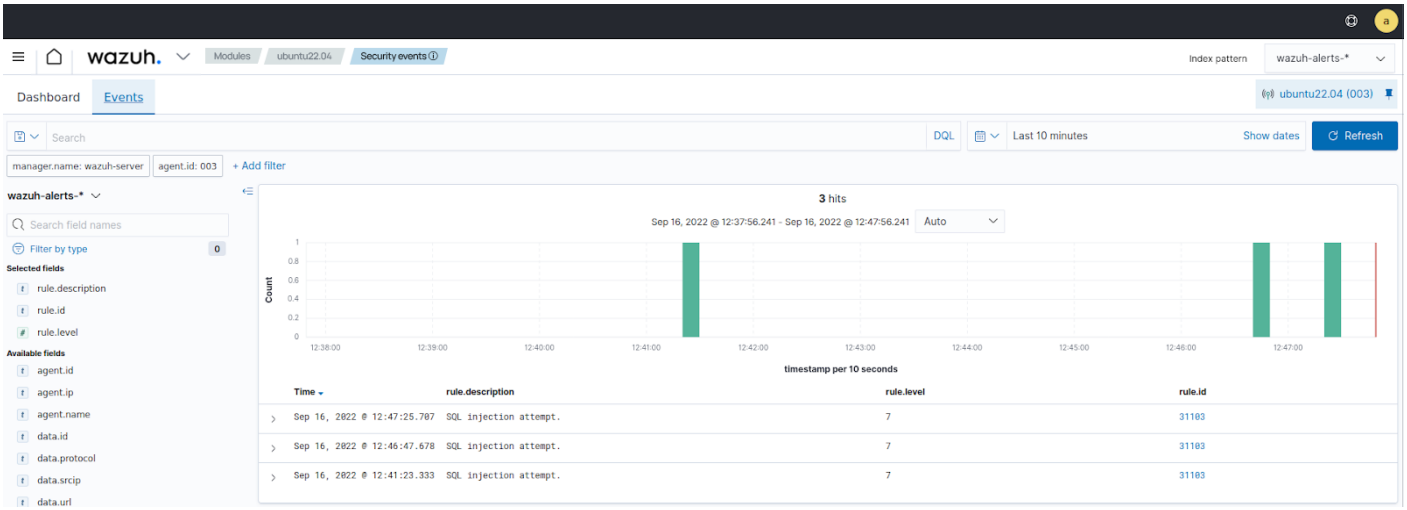
Utilice el siguiente comando con la dirección IP adecuada del servidor y ejecute el siguiente comando desde el extremo del atacante:

```
curl -XGET "http://IP_VICTIMA/users/?id=SELECT+*+FROM+users";
```

El payload contiene una consulta SQL donde el comando quiere realizar la selección de todas las filas y todas las columnas de la tabla llamada "users" que suele ser por defecto los usuarios que se usan para ingresar en un login cuando se configura por defecto.

Resultados.

El resultado generó una alerta de intento de inyección SQL.



PDC 7: DETECCIÓN DE SUSPICIOUS BINARIES.

Detectar binarios sospechosos es una capacidad de seguridad que permite identificar código ejecutable que puede ser malicioso o anómalo en un punto final. En este caso de uso, demostramos cómo el módulo rootcheck de Wazuh puede detectar un sistema binario troyano en un extremo victima.

El exploit se realiza reemplazando el contenido de un binario legítimo con un código malicioso para engañar al punto final para que lo ejecute como el binario legítimo, el módulo rootcheck de Wazuh también busca procesos, puertos y archivos ocultos.
Configuración.

Configuración.

De forma predeterminada, el módulo rootcheck de Wazuh está habilitado en el archivo de configuración del agente de Wazuh.
Verifique el apartado <rootcheck> en el archivo de configuración del terminal monitoreado y asegurarse que la configuración este dada de la siguiente manera

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<rootcheck>
  <disabled>no</disabled>
  <check_files>yes</check_files>
  <!-- Line for trojans detection -->
```

```
    <check_trojans>yes</check_trojans>
    <check_dev>yes</check_dev>
    <check_sys>yes</check_sys>
    <check_pids>yes</check_pids>
    <check_ports>yes</check_ports>
    <check_if>yes</check_if>
    <!-- Frequency that rootcheck is executed - every 12 hours -->
    <frequency>43200</frequency>
    <rootkit_files>/var/ossec/etc/shared/rootkit_files.txt</rootkit_files>
    <rootkit_trojans>/var/ossec/etc/shared/rootkit_trojans.txt</rootkit_trojans>
    <skip_nfs>yes</skip_nfs>
</rootcheck>
```

Emulación de ataque.

Creamos una copia del sistema binario original con:

```
sudo cp -p /usr/bin/w /usr/bin/w.copy
```

Ejecute el comando

```
sudo chmod +x /usr/bin/w.copy
```

Este da permisos de ejecución a la copia creada.

Reemplace el sistema binario original /usr/bin/w con el siguiente script de shell

```
sudo tee /usr/bin/w << EOF
!/bin/bash
echo "`date` this is evil" > /tmp/trojan_created_file
echo 'test for /usr/bin/w trojaned file' >> /tmp/trojan_created_file
Now running original binary
/usr/bin/w.copy
EOF
```

Este cambio en el archivo original es otro método de un ciberatacante y tiene las siguientes posibles consecuencias:

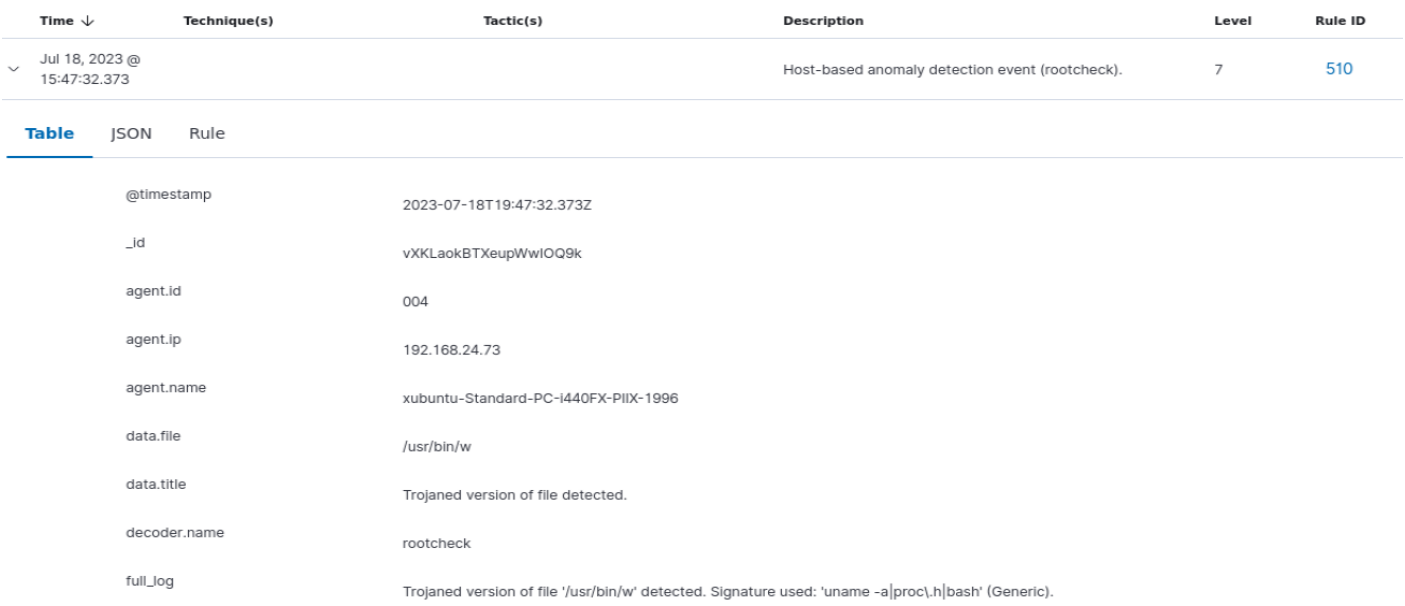
- Manipulación del sistema: El script sobrescribe el archivo legítimo /usr/bin/w, lo que podría interrumpir su funcionalidad prevista.
- Creación de datos: Crea un archivo llamado /tmp/trojan_created_file que contiene mensajes sospechosos.
- Comportamiento desconocido: No se conoce el propósito completo del script ni sus acciones potenciales, lo que lo convierte en un riesgo de seguridad.

El análisis de rootcheck se ejecuta cada 12 horas de forma predeterminada por lo que se debe forzar un escaneo reiniciando el agente de Wazuh.

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

Resultados.

Mientras tanto Wazuh manager nos da como resultado la siguiente respuesta:



Para restablecer nuestro las condiciones originales anteriores a la prueba de seguridad y evitar problemas de funcionamiento debemos realizar la operación inversa.

Abrir la terminal en el punto final donde reemplazó el binario original /usr/bin/w y elimine el script de shell que escribió en el binario original.

```
sudo rm /usr/bin/w
```

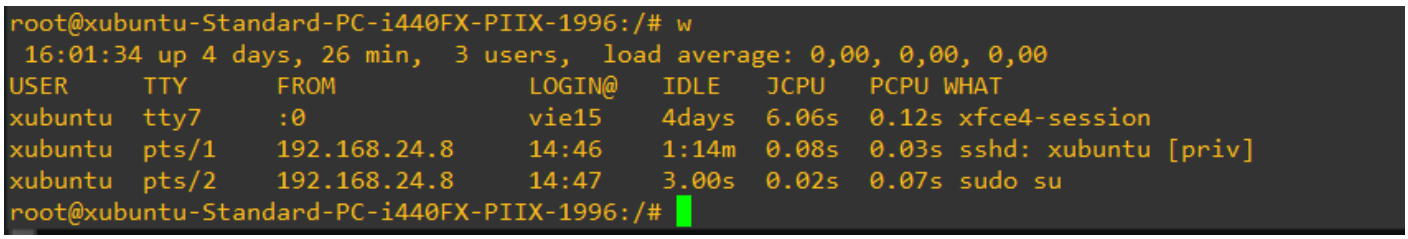
Se debe restaurar la copia del binario original.

```
sudo mv /usr/bin/w.copy /usr/bin/w
```

Por ultimo eliminar el archivo que creó el script de shell.

```
sudo rm /tmp/trojan_created_file
```

Verifique que el binario original funcione correctamente con el comando **w** los resultados serán los siguientes en caso de estar restablecido.



PDC 8: DETECCIÓN Y ELIMINACIÓN DE MALWARE MEDIANTE INTEGRACIÓN CON VIRUS TOTAL.

Wazuh puede utilizar un modulo de monitoreo para supervisar la integridad de los archivos en este caso buscando cambios realizados y la API de Virus-Total. Para la integración de Wazuh con Virus-Total. Todas estas configuraciones se realizan en el servidor primeramente y luego se realizará la configuración del agente.

Configuración.

Agregue las siguientes reglas al archivo local de reglas.

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

```
<group name="syscheck,pci_dss_11.5,nist_800_53_SI.7,">
  <!-- Rules for Linux systems -->
  <rule id="100200" level="7">
    <if_sid>550</if_sid>
    <field name="file">/root</field>
    <description>File modified in /root directory.</description>
  </rule>
  <rule id="100201" level="7">
```

```
    <if_sid>554</if_sid>
    <field name="file">/root</field>
    <description>File added to /root directory.</description>
  </rule>
</group>
```

Añadir la sección dentro del archivo de configuración Ossec.

```
sudo nano /var/ossec/etc/ossec.conf
```

Se realiza la configuración en este archivo para habilitar la integración de Virustotal. Reemplazando con la clave API de VirusTotal (Para obtener api_key se debe crear una cuenta en virustotal y seguir los pasos correspondientes para copiar la API gratuita que ofrece el sitio) permitiendo activar una consulta siempre que alguna de las reglas y se activan, la configuración se realiza dentro de <ossec_config> </ossec_config>.

```
<integration>
  <name>virustotal</name>
  <api_key>API_KEY</api_key>
  <rule_id>100200,100201</rule_id>
  <alert_format>json</alert_format>
</integration>
```

Agregamos los siguientes bloques al archivo del servidor Wazuh permitiendo una respuesta activa y activa el script cuando VirusTotal marca un archivo como malicioso en el archivo /var/ossec/etc/ossec.confremove-threat.sh

```
sudo nano /var/ossec/etc/ossec.conf
```

```
<ossec_config>
  <command>
    <name>remove-threat</name>
    <executable>remove-threat.sh</executable>
    <timeout_allowed>no</timeout_allowed>
  </command>
  <active-response>
    <disabled>no</disabled>
    <command>remove-threat</command>
    <location>local</location>
```

```
    <rules_id>87105</rules_id>
  </active-response>
</ossec_config>
```

Agregar reglas al servidor de Wazuh /var/ossec/etc/rules/local_rules.xml archivo para alertar sobre los resultados de la respuesta activa:

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

```
<group name="virustotal,">
  <rule id="100092" level="12">
    <if_sid>657</if_sid>
    <match>Successfully removed threat</match>
    <description>$(parameters.program) removed threat located at
$(parameters.alert.data.virustotal.source.file)</description>
  </rule>
  <rule id="100093" level="12">
    <if_sid>657</if_sid>
    <match>Error removing threat</match>
    <description>Error removing threat located at $(parameters.alert.data.virustotal.source.file)</description>
  </rule>
</group>
```

Reiniciamos el administrador de Wazuh para aplicar los cambios realizados.

```
systemctl restart wazuh-manager
```

Para esta prueba de concepto la maquina vulnerable es Debian linux en la cual realizaremos las siguientes configuraciones:

```
sudo nano /var/ossec/etc/ossec.conf
```

Dentro del archivo de configuración <disabled> se establece en no, permitiendo que Wazuh FIM controle los cambios de directorio.

```
<syscheck>
  <disabled>no</disabled>
```

Agregar una entrada dentro del apartado **<syscheck>** para configurar un directorio para ser monitoreado casi en tiempo real. En este caso se está monitoreando el directorio /root.

```
<directories realtime="yes">/root</directories>
```

Instalar "**jq**" la cual es una utilidad que procesa la entrada JSON del script de respuesta activo.

```
sudo apt update
sudo apt -y install jq
```

Creamos en el directorio **/var/ossec/active-response/bin/remove-threat.sh** el cual es una secuencia de comandos de respuesta activa para eliminar archivos maliciosos del punto final monitoreado.

```
#!/bin/bash
LOCAL=`dirname $0`;
cd $LOCAL
cd ../
PWD=`pwd`
read INPUT_JSON
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.data.virustotal.source.file)
COMMAND=$(echo $INPUT_JSON | jq -r .command)
LOG_FILE="${PWD}/../logs/active-responses.log"
if [ ${COMMAND} = "add" ]
then
 # Send control message to execd
 printf '{"version":1,"origin":{"name":"remove-threat","module":"active-response"},"command":"check_keys",
"parameters":{"keys":[]}}\n'
 read RESPONSE
 COMMAND2=$(echo $RESPONSE | jq -r .command)
 if [ ${COMMAND2} != "continue" ]
 then
  echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Remove threat active response aborted" >>
${LOG_FILE}
  exit 0;
 fi
fi
# Removing file
```

```
rm -f $FILENAME
if [ $? -eq 0 ]; then
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Successfully removed threat" >> ${LOG_FILE}
else
 echo "`date '+%Y/%m/%d %H:%M:%S'` $0: $INPUT_JSON Error removing threat" >> ${LOG_FILE}
fi
exit 0;
```

Cambiar la propiedad y permisos del archivo creado /var/ossec/active-response/bin/remove-threat.sh.

```
sudo chmod 750 /var/ossec/active-response/bin/remove-threat.sh
```

Cambiamos el propietario del archivo para que Wazuh pueda ejecutar el script en caso de alertas activas.

```
sudo chown root:wazuh /var/ossec/active-response/bin/remove-threat.sh
```

Para aplicar estos cambios reiniciamos el agente.

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

Emulación del ataque.

Para descargar un archivo de prueba EICAR en Debian Linux desde la consola. Abrir una terminal y acceda al directorio **/root** donde se guardará el archivo de prueba y ejecute el comando de descarga, en este caso:

```
sudo cd /root
sudo curl -LO https://secure.eicar.org/eicar.com && ls -lah eicar.com
```

Esperamos que se complete la descarga y verificamos el archivo eicar.com en el directorio, este archivo no contiene malware real sin embargo se reconoce como un malware de prueba para Virus Total.

Resultados.

Time ↓	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jul 19, 2023 @ 09:23:23.765			active-response/bin/remove-threat.sh removed threat located at /root/eicar.com	12	100092
Jul 19, 2023 @ 09:23:22.772	T1070.004 T1485	Defense Evasion, Impact	File deleted.	7	553
Jul 19, 2023 @ 09:23:22.718	T1203	Execution	VirusTotal: Alert - /root/eicar.com - 64 engines detected this file	12	87105
Jul 19, 2023 @ 09:23:20.727			File added to /root directory.	8	100201

Wazuh manager tiene los siguientes eventos de detección. El evento de respuesta activa nos indica que el archivo fue eliminado y puede verificarse en consola:

Jul 19, 2023 @ 09:23:23.765		active-response/bin/remove-threat.sh removed threat located at /root/eicar.com	12	100092

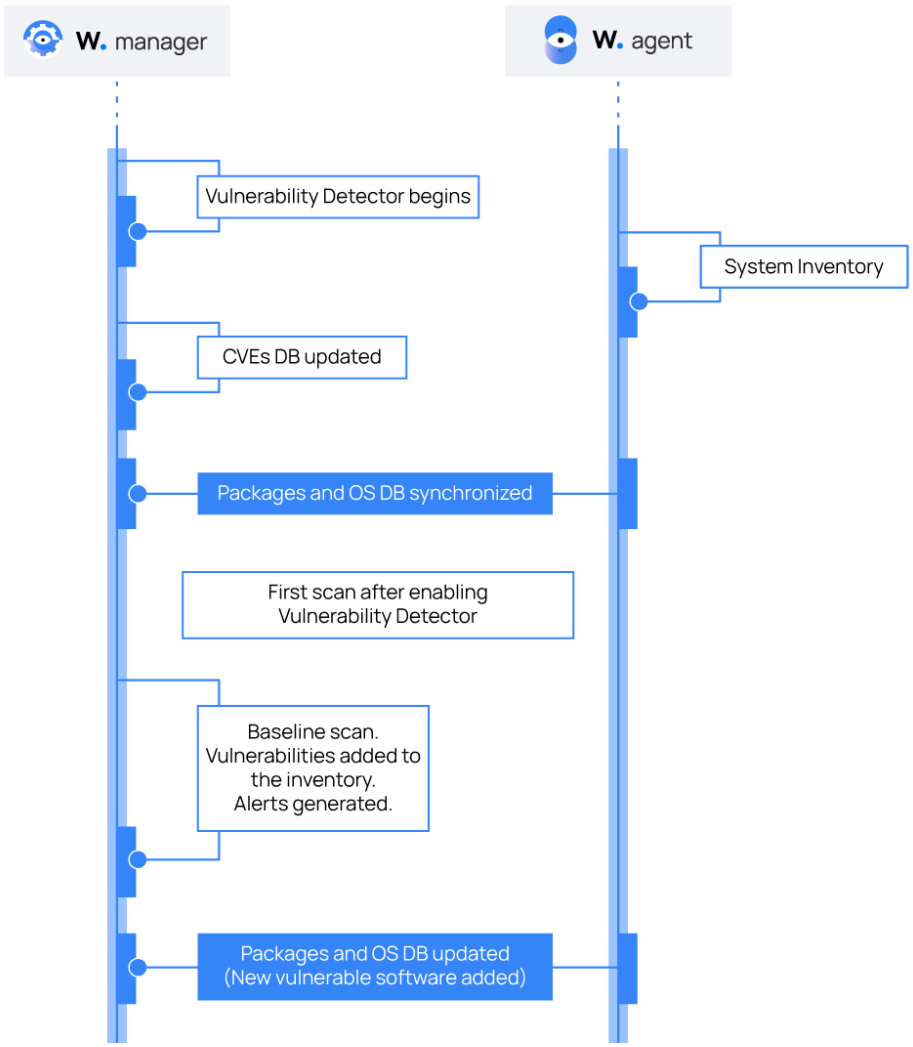
Table JSON Rule

@timestamp	2023-07-19T13:23:23.765Z
_id	yoJRbokBTx8HHDFC5sYS
agent.id	005
agent.ip	192.168.24.87
agent.name	debian
data.command	add
data.origin.module	wazuh-execd

PDC 9: DETECCIÓN DE VULNERABILIDADES.

Los agentes de Wazuh recopilan una lista de aplicaciones instaladas desde los terminales monitoreados y la envían periódicamente al servidor de Wazuh.

El servidor de Wazuh posee bases de datos SQLite que almacenan la lista enviada por los agentes, el servidor de Wazuh crea una base de datos de vulnerabilidad global a partir de repositorios **CVE** disponibles públicamente y utiliza esta base de datos para correlacionar esta información con los datos del listado de aplicaciones del agente.



La imagen muestra el flujo de trabajo de la detección de vulnerabilidades en Wazuh. El flujo de trabajo comienza con la recopilación de datos de los agentes de Wazuh. Estos datos pueden incluir registros, configuraciones de sistemas y resultados de escaneo de vulnerabilidades. Wazuh luego analiza estos datos en busca de posibles vulnerabilidades. Una vez que Wazuh ha identificado una vulnerabilidad, genera una alerta. Las alertas pueden ser enviadas a los usuarios por correo electrónico, mensaje de texto o notificación push.
La detección de vulnerabilidades es una parte importante de la gestión de seguridad de la información. Al identificar y remediar las vulnerabilidades, las organizaciones pueden proteger sus sistemas de ataques cibernéticos, los pasos que sigue el diagrama son:

- Recopilación de datos: Wazuh recopila datos de los agentes de Wazuh. Estos datos pueden incluir registros, configuraciones de sistemas y resultados de escaneo de vulnerabilidades.
- Análisis de datos: Wazuh analiza los datos recopilados en busca de posibles vulnerabilidades. Wazuh utiliza una variedad de métodos para analizar los datos, incluyendo reglas, firmas y aprendizaje automático.

- Generación de alertas: Una vez que Wazuh ha identificado una vulnerabilidad, genera una alerta. Las alertas pueden ser enviadas a los usuarios por correo electrónico, mensaje de texto o notificación push.
- Mitigación de vulnerabilidades: Los usuarios pueden utilizar la información proporcionada en las alertas de Wazuh para remediar las vulnerabilidades en sus sistemas.

Configuración.

Nos dirigimos al directorio /var/ossec/etc/ossec.conf para realizar las siguientes modificaciones, dentro del apartado "`<ossec_config>`"

```
<vulnerability-detector>
  <enabled>yes</enabled>
  <interval>5m</interval>
  <min_full_scan_interval>6h</min_full_scan_interval>
  <run_on_start>yes</run_on_start>
  <!--        Ubuntu OS vulnerabilities -->
  <provider name="canonical">
  <enabled>yes</enabled>
  <os>trusty</os>
  <os>xenial</os>
  <os>bionic</os>
  <os>focal</os>
  <os>jammy</os>
  <update_interval>1h</update_interval>
  </provider>
  <!--        Debian OS vulnerabilities -->
  <provider name="debian">
  <enabled>yes</enabled>
  <os>buster</os>
  <os>bullseye</os>
  <update_interval>1h</update_interval>
  </provider>
  <!--        RedHat OS vulnerabilities -->
  <provider name="redhat">
  <enabled>yes</enabled>
  <os>5</os>
  <os>6</os>
```

```
    <os>7</os>
    <os>8</os>
    <os allow="CentOS Linux-8">8</os>
    <os>9</os>
    <update_interval>1h</update_interval>
    </provider>
    <!--        Windows OS vulnerabilities -->
    <provider name="msu">
    <enabled>yes</enabled>
    <update_interval>1h</update_interval>
    </provider>
    <!--        Aggregate vulnerabilities -->
    <provider name="nvd">
    <enabled>yes</enabled>
    <update_from_year>2019</update_from_year>
    <update_interval>1h</update_interval>
    </provider>
  </vulnerability-detector>
```

Se tienen casos de adaptación para los casos de Kali Linux versión 2023 y 2024 donde se integran junto con la gestión de vulnerabilidades de Debian en base a la gestión de vulnerabilidades de sistemas no compatibles para el cual utilizamos la siguiente modificación al bloque de Debian de la siguiente manera.

```
    <!-- Debian OS vulnerabilities -->
    <provider name="debian">
      <enabled>yes</enabled>
      <os>buster</os>
      <os>bullseye</os>
      <os>bookworm</os>
      <os allow="Kali GNU/Linux-2023,Kali GNU/Linux-2024">buster</os>
      <update_interval>1h</update_interval>
    </provider>
```

Se utiliza el formato *OS_name-OS_majorcon* separados por comas y el atributo *allow* para incluir el sistema operativo. Para aplicar los cambios reiniciamos Wazuh manager.

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

En caso de revisión de la base de datos de Wazuh manager se deben seguir los siguientes pasos. En caso de no tener instalado SQLlite.

```
sudo apt install sqlite3
```

Seguidamente para abrir la base de datos de vulnerabilidades:

```
sqlite3 /var/ossec/queue/vulnerabilities/cve.db
```

Listamos las tablas en la base de datos.

```
sqlite> .tables
```

Para visualizar las tablas, Ej:

```
sqlite> select * from <table>;
```

El servidor de Wazuh crea automáticamente la base de datos de vulnerabilidad global con datos de los siguientes repositorios:

- https://canonical.com : se usa para extraer CVE para distribuciones Ubuntu Linux.
- https://www.redhat.com : se utiliza para extraer CVE para las distribuciones Red Hat y CentOS Linux.
- https://www.debian.org : se usa para extraer CVE para las distribuciones de Debian Linux.
- https://security.archlinux.org : se usa para extraer CVE para las distribuciones de Arch Linux.
- https://nvd.nist.gov : se usa para extraer CVE de la base de datos nacional de vulnerabilidad.
- https://feed.wazuh.com : se utiliza para obtener actualizaciones de seguridad de Microsoft (MSU) y fuentes ALAS.

Los feeds contienen información de parches y CVE para productos de Microsoft y Amazon Linux. Utilizan el Catálogo de actualizaciones de Microsoft y el Centro de seguridad de Amazon Linux como fuentes de información. Wazuh analiza y formatea los datos antes de cargarlos en el feed de Wazuh.
Una vez que el módulo de vulnerabilidades ha creado la base de datos de vulnerabilidad global que contiene los CVE, el proceso de detección busca paquetes vulnerables en las bases de datos de inventario.
Un paquete se etiqueta como vulnerable cuando su versión coincide con las del rango afectado de un CVE. Las alertas muestran los resultados y el módulo almacena los hallazgos en un inventario de vulnerabilidades por agente. Este inventario contiene el estado actual de cada agente e incluye vulnerabilidades que han sido detectadas y no resueltas. Los usuarios pueden consultar el inventario para buscar alertas e información sobre vulnerabilidades.

PDC 10: DETECCIÓN DE MALWARE INTEGRACIÓN CON YARA.

Esta integración se utiliza con Wazuh para escanear archivos agregados o modificados en el terminal en busca de malware en el directorio /var/ossec/active-response/bin/. El módulo de respuesta activa YARA escanea archivos nuevos o modificados cada vez que el módulo FIM de Wazuh activa una alerta.

Configuración.

Configuraremos el terminal final e instalaremos Yara y sus dependencias configurando los módulos FIM y de respuesta activa. Para instalar YARA seguimos los siguientes pasos:

```
sudo apt update
sudo apt install -y make gcc autoconf libtool libssl-dev pkg-config jq
sudo curl -LO https://github.com/VirusTotal/yara/archive/v4.2.3.tar.gz
sudo tar -xvzf v4.2.3.tar.gz -C /usr/local/bin/ && rm -f v4.2.3.tar.gz
cd /usr/local/bin/yara-4.2.3/
sudo ./bootstrap.sh && sudo ./configure && sudo make && sudo make install && sudo make check
```

Verificamos la instalación

```
yara
```

*El siguiente error aparece en distribuciones de Ubuntu.

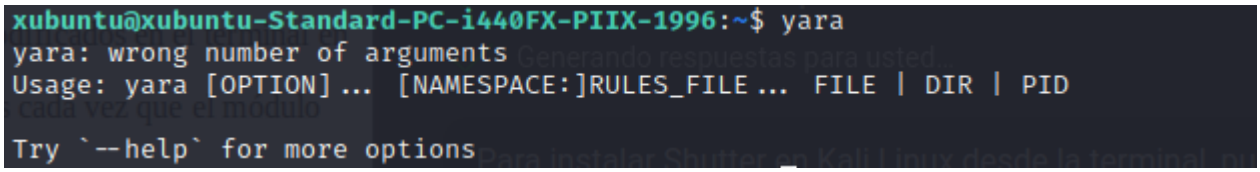
```
/usr/local/bin/yara: error while loading shared libraries: libyara.so.9: cannot open shared object file: No such file or directory.
```

La solución del error se realiza desde un usuario con privilegios de root, solo con permisos sudo este comando es rechazado.

```
sudo echo "/usr/local/lib" >> /etc/ld.so.conf
ldconfig
```

Una vez solucionado el error o si este no se encuentra la respuesta del terminal deberia ser de la siguiente manera:

```
yara
```



Proseguimos a descargar las reglas de detección de malware en Yara

```
sudo mkdir -p /tmp/yara/rules
sudo curl 'https://valhalla.nextron-systems.com/api/v1/get' \
-H 'Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8' \
-H 'Accept-Language: en-US,en;q=0.5' \
--compressed \
-H 'Referer: https://valhalla.nextron-systems.com/' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-H 'DNT: 1' -H 'Connection: keep-alive' -H 'Upgrade-Insecure-Requests: 1' \
--data
'demo=demo&apikey=1111111111111111111111111111111111111111111111111111111111111111&format=te
xt' \
-o /tmp/yara/rules/yara_rules.yar
```

Se debe crear un archivo yara.sh

```
#!/bin/bash
# Wazuh - Yara active response
# Copyright (C) 2015-2022, Wazuh Inc.
#
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.
        #------------------------ Gather parameters -------------------------#
# Extra arguments
read INPUT_JSON
YARA_PATH=$(echo $INPUT_JSON | jq -r .parameters.extra_args[1])
YARA_RULES=$(echo $INPUT_JSON | jq -r .parameters.extra_args[3])
FILENAME=$(echo $INPUT_JSON | jq -r .parameters.alert.syscheck.path)
```

```
# Set LOG_FILE path
LOG_FILE="logs/active-responses.log"
size=0
actual_size=$(stat -c %s ${FILENAME})
while [ ${size} -ne ${actual_size} ]; do
    sleep 1
    size=${actual_size}
    actual_size=$(stat -c %s ${FILENAME})
done
#------------------------ Analyze parameters -------------------------#
if [[ ! $YARA_PATH ]] || [[ ! $YARA_RULES ]]
then
    echo "wazuh-yara: ERROR - Yara active response error. Yara path and rules parameters are mandatory." >>
${LOG_FILE}
    exit 1
fi
#-------------------------- Main workflow --------------------------#
# Execute Yara scan on the specified filename
yara_output="$("${YARA_PATH}"/yara -w -r "$YARA_RULES" "$FILENAME")"

if [[ $yara_output != "" ]]
then
    # Iterate every detected rule and append it to the LOG_FILE
    while read -r line; do
        echo "wazuh-yara: INFO - Scan result: $line" >> ${LOG_FILE}
    done <<< "$yara_output"
fi
exit 0;
```

```
nano /var/ossec/active-response/bin/yara.sh
```

Por último cambiamos el propietario del archivo para que root:wazuh lo pueda manejar:

```
sudo chown root:wazuh /var/ossec/active-response/bin/yara.sh
sudo chmod 750 /var/ossec/active-response/bin/yara.sh
```

Agregar que dentro del bloque `<syscheck>` del agente Wazuh en el archivo de configuración este configurado para monitorear el directorio /tmp/yara/malware:

```
nano /var/ossec/etc/ossec.conf
```

```
<directories realtime="yes">/tmp/yara/malware</directories>
```

Reiniciar el agente de Wazuh

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

En el servidor de Wazuh tenemos que agregar eventos para extraer la información y los resultados de Yara.

```
nano /var/ossec/etc/rules/local_rules.xml
```

```
<group name="syscheck,">
  <rule id="100300" level="7">
    <if_sid>550</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File modified in /tmp/yara/malware/ directory.</description>
  </rule>
  <rule id="100301" level="7">
    <if_sid>554</if_sid>
    <field name="file">/tmp/yara/malware/</field>
    <description>File added to /tmp/yara/malware/ directory.</description>
  </rule>
</group>
```

La regla 100300 que especifica la modificación en un directorio es aplicable a un entorno delicado en el que los archivos deben mantener su integridad y la seguridad de estos sea de alta prioridad, en un despliegue con terminales remotos para usuario no es aconsejable ya que se consumen recursos al ejecutar el análisis con yara cada vez que un archivo cambia de tamaño al ser descargado o modificado dentro de un directorio monitorizado se recomienda utilizar el tipo de regla 100301 en caso de monitoreo a terminales de usuario.

```
<group name="yara,">
  <rule id="108000" level="0">
    <decoded_as>yara_decoder</decoded_as>
    <description>Yara grouping rule</description>
  </rule>
```

```
  <rule id="108001" level="12">
    <if_sid>108000</if_sid>
    <match>wazuh-yara: INFO - Scan result: </match>
    <description>File "$(yara_scanned_file)" is a positive match. Yara rule: $(yara_rule)</description>
  </rule>
</group>
```

En el archivo local_decoder agregamos la siguiente información:

```
nano /var/ossec/etc/decoders/local_decoder.xml
```

```
<decoder name="yara_decoder">
  <prematch>wazuh-yara:</prematch>
</decoder>
<decoder name="yara_decoder1">
  <parent>yara_decoder</parent>
  <regex>wazuh-yara: (\S+) - Scan result: (\S+) (\S+)</regex>
  <order>log_type, yara_rule, yara_scanned_file</order>
</decoder>
```

Emulación del ataque.

Crear el archivo /tmp/yara/malware/malware_downloader.sh en el punto final monitoreado para descargar muestras de malware.

```
touch /tmp/yara/malware/malware_downloader.sh
nano /tmp/yara/malware/malware_downloader.sh
```

```
#!/bin/bash
# Wazuh - Malware Downloader for test purposes
# Copyright (C) 2015-2022, Wazuh Inc.
# This program is free software; you can redistribute it
# and/or modify it under the terms of the GNU General Public
# License (version 2) as published by the FSF - Free Software
# Foundation.
function fetch_sample(){
  curl -s -XGET "$1" -o "$2"
}192.168.24.92
```

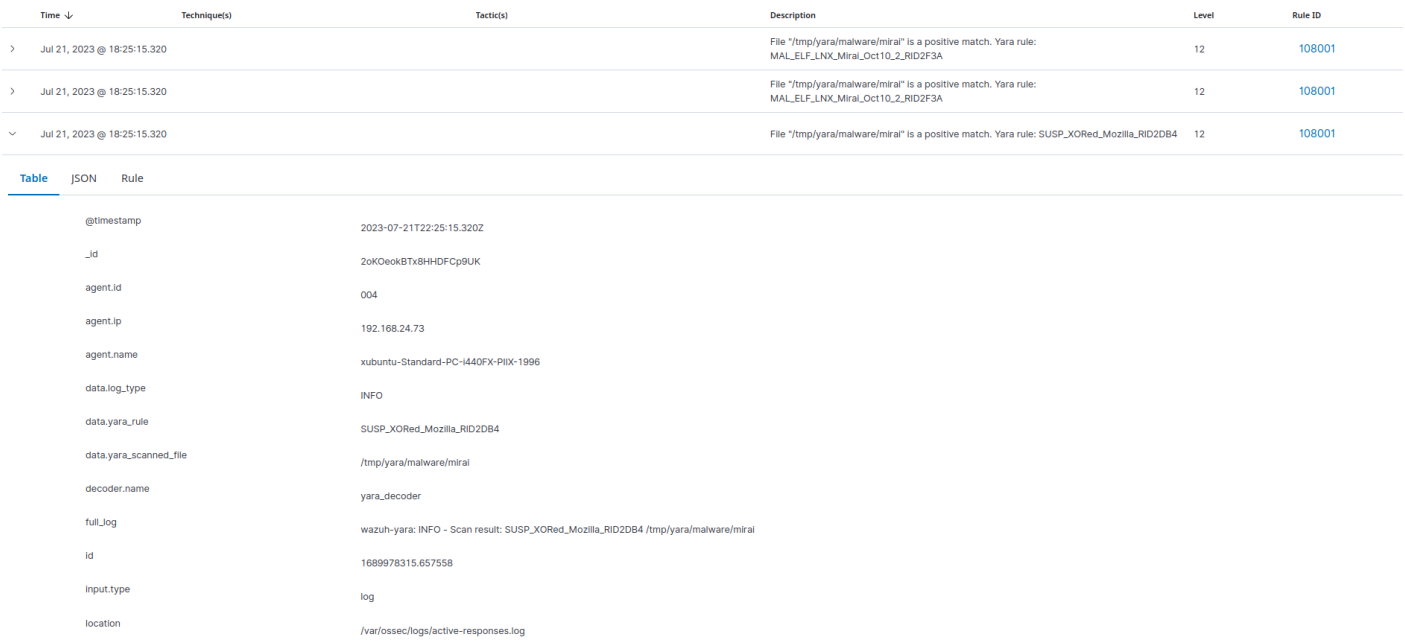
```
echo "WARNING: Downloading Malware samples, please use this script with  caution."
read -p "  Do you want to continue? (y/n)" -n 1 -r ANSWER
echo
if [[ $ANSWER =~ ^[Yy]$ ]]
then
    echo
    # Mirai
    echo "# Mirai: https://en.wikipedia.org/wiki/Mirai_(malware)"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/mirai" "/tmp/yara/malware/mirai" && echo "Done!" || echo "Error while downloading."
    echo
    # Xbash
    echo "# Xbash: https://unit42.paloaltonetworks.com/unit42-xbash-combines-botnet-ransomware-coinmining-worm-targets-linux-windows/"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/xbash" "/tmp/yara/malware/xbash" && echo "Done!" || echo "Error while downloading."
    echo
    # VPNFilter
    echo "# VPNFilter: https://news.sophos.com/en-us/2018/05/24/vpnfilter-botnet-a-sophoslabs-analysis/"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/vpn_filter" "/tmp/yara/malware/vpn_filter" && echo "Done!" || echo "Error while downloading."
    echo
    # Webshell
    echo "# WebShell: https://github.com/SecWiki/WebShell-2/blob/master/Php/Worse%20Linux%20Shell.php"
    echo "Downloading malware sample..."
    fetch_sample "https://wazuh-demo.s3-us-west-1.amazonaws.com/webshell" "/tmp/yara/malware/webshell" && echo "Done!" || echo "Error while downloading."
    echo
fi
```

Ejecutamos el script malware_downloader.sh para descargar muestras de malware al directorio /tmp/yara/malware.

```
sudo bash /tmp/yara/malware/malware_downloader.sh
```

Resultado.

Filtramos las alertas por rule.groups **yara**



	Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
›	Jul 21, 2023 @ 18:25:15.320			File "/tmp/yara/malware/mirai" is a positive match. Yara rule: MAL_ELF_LNX_Mirai_Oct10_2_RID2F3A	12	108001
›	Jul 21, 2023 @ 18:25:15.320			File "/tmp/yara/malware/mirai" is a positive match. Yara rule: MAL_ELF_LNX_Mirai_Oct10_2_RID2F3A	12	108001
⌄	Jul 21, 2023 @ 18:25:15.320			File "/tmp/yara/malware/mirai" is a positive match. Yara rule: SUSP_XORed_Mozilla_RID2DB4	12	108001

Table JSON Rule

@timestamp	2023-07-21T22:25:15.320Z
_id	2oKOeokBTx8HHDFCp9UK
agent.id	004
agent.ip	192.168.24.73
agent.name	xubuntu-Standard-PC-i440FX-PIIX-1996
data.log_type	INFO
data.yara_rule	SUSP_XORed_Mozilla_RID2DB4
data.yara_scanned_file	/tmp/yara/malware/mirai
decoder.name	yara_decoder
full_log	wazuh-yara: INFO - Scan result: SUSP_XORed_Mozilla_RID2DB4 /tmp/yara/malware/mirai
id	1689978315.657558
input.type	log
location	/var/ossec/logs/active-responses.log

PDC 11: DETECCIÓN DE PROCESOS OCULTOS.

Se realizara una prueba de concepto en Debian linux, descargando, compilando y cargando un rootkit. Luego, se configura el módulo rootcheck de Wazuh en el terminal para la detección de anomalías.

Configuración.

Para realizar el cometido primero realizamos los siguientes pasos desde el terminal:

```
sudo apt update
```

Instalamos los paquetes necesarios para el rootkit

```
sudo apt -y install gcc git
```

Configuramos el agente de Wazuh para ejecutar el escaneo de verificación de raíz cada 2 minutos. En el archivo ossec.conf

```
sudo nano /var/ossec/etc/ossec.conf
```

Seleccionamos *frequency* en 120 para realizar el análisis cada 2 minutos:

```
<frequency>120</frequency>
```

Reiniciamos el agente de Wazuh para aplicar los cambios.

```
systemctl restart wazuh-agent
```

Emulación del ataque.

Obtenemos un código de rootkit desde Diamorphine de GitHub.

```
git clone https://github.com/m0nad/Diamorphine
```

Navegamos hasta el directorio Diamorphine y compilamos el código fuente.

```
cd /var/ossec/etc/Diamorphine/
make
```

- Se produce un error en la terminal cuando el comando make no realiza ninguna acción, verificamos la instalación de build-essential en caso de no estar instalado de la siguiente manera: `sudo apt-get install build-essential`
- Se produce un error en la terminal cuando linux/syscalls.h" no se encuentra en el sistema, para solucionar el problema se debe realizar la siguiente instalación.

```
sudo apt-get install linux-headers-$(uname -r)
```

Luego cargamos el módulo del kernel del rootkit.

```
insmod diamorphine.ko
```

Se debe ejecutar la señal kill 63 con el PID de un proceso aleatorio que se ejecuta en el terminal. Esto muestra el rootkit Diamorphine. De forma predeterminada, Diamorphine se oculta para que no la detectemos ejecutando el comando lsmod.

```
lsmod | grep diamorphine
kill -63 509
lsmod | grep diamorphine
```

Se deben ejecutar los siguientes comandos para ver cómo funciona el proceso rsyslogd es primero visible y luego ya no es visible. Este rootkit le permite ocultar procesos seleccionados del dominio ps. Enviando la señal de kill 31 oculta/muestra cualquier proceso.

```
ps auxw | grep rsyslogd | grep -v grep
```

Si el proceso no ocasiona ninguna salida es que algunos paquetes no están instalados, se deben seguir las siguientes instrucciones:

```
sudo apt-get install linux-headers-$(uname -r)
sudo apt-get install rsyslog
systemctl restart rsyslog
systemctl status rsyslog
```

Una vez instalados los paquetes tenemos:

```
ps auxw | grep rsyslogd | grep -v grep
```

salida:

```
root      22534  0.0  0.1 221772  6036 ?        Ssl  16:34   0:00 /usr/sbin/rsyslogd -n -iNONE
```

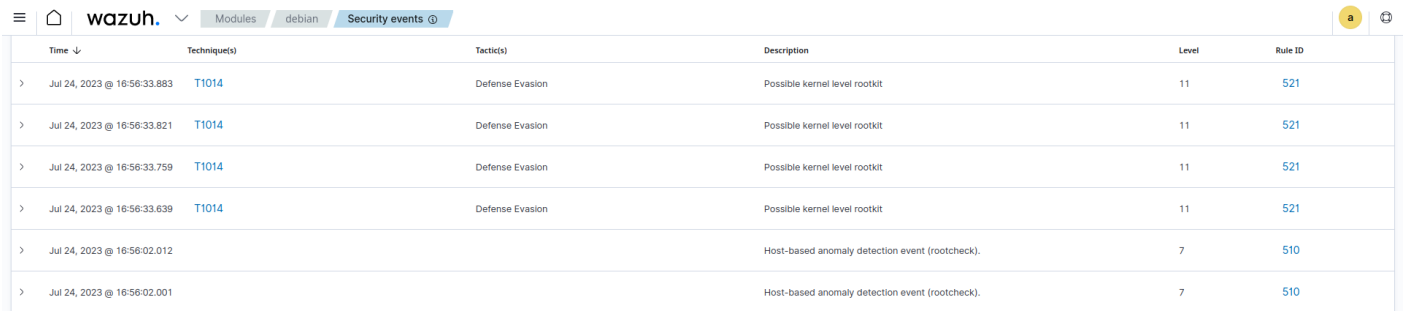
```
sudo kill -31 22534
```

```
ps auxw | grep rsyslogd | grep -v grep
```

El primer comando envía una señal kill al proceso 31 “rsyslogd”.
El segundo comando muestra una lista de procesos que contienen la cadena “rsyslogd” en su nombre.
La opción “-v grep” excluye cualquier proceso que contenga la cadena “grep” en su nombre.

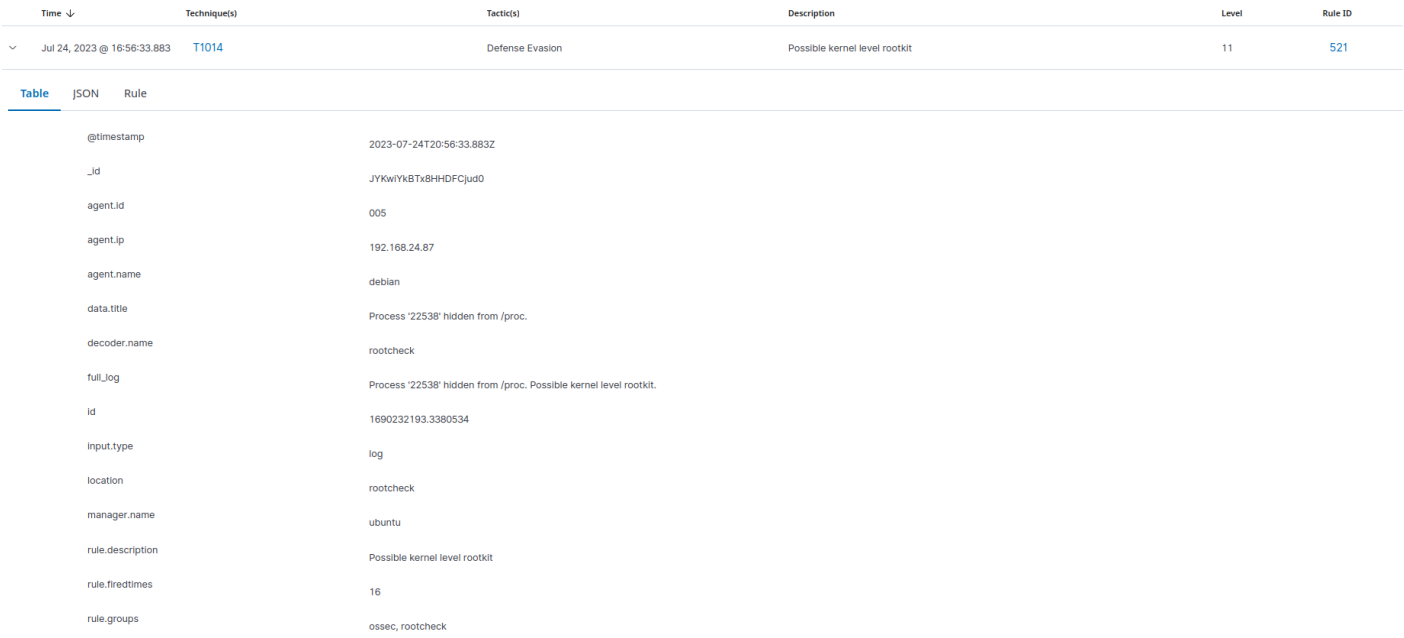
Resultados.

Se pueden visualizar los datos de alerta en el panel de control de Wazuh.



Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jul 24, 2023 @ 16:56:33.883	T1014	Defense Evasion	Possible kernel level rootkit	11	521
Jul 24, 2023 @ 16:56:33.821	T1014	Defense Evasion	Possible kernel level rootkit	11	521
Jul 24, 2023 @ 16:56:33.759	T1014	Defense Evasion	Possible kernel level rootkit	11	521
Jul 24, 2023 @ 16:56:33.639	T1014	Defense Evasion	Possible kernel level rootkit	11	521
Jul 24, 2023 @ 16:56:02.012			Host-based anomaly detection event (rootcheck).	7	510
Jul 24, 2023 @ 16:56:02.001			Host-based anomaly detection event (rootcheck).	7	510

Se realizo una detección de un posible rootkit.



Time	Technique(s)	Tactic(s)	Description	Level	Rule ID
Jul 24, 2023 @ 16:56:33.883	T1014	Defense Evasion	Possible kernel level rootkit	11	521

Table JSON Rule

@timestamp	2023-07-24T20:56:33.883Z
_id	JYKwiYkBTx8HHDFCJud0
agent.id	005
agent.ip	192.168.24.87
agent.name	debian
data.title	Process '22538' hidden from /proc.
decoder.name	rootcheck
full_log	Process '22538' hidden from /proc. Possible kernel level rootkit.
id	1690232193.3380534
input.type	log
location	rootcheck
manager.name	ubuntu
rule.description	Possible kernel level rootkit
rule.firedtimes	16
rule.groups	ossec, rootcheck

Por seguridad se deben eliminar los archivos y la instalación del sistema de este rootkit para esto realizamos.

```
sudo rm -rf /var/ossec/etc/Diamorphine
```

PDC 12: SUPERVISIÓN DE LA EJECUCIÓN DE COMANDOS MALICIOSOS.

Para este prueba configuramos ***Auditd*** en un terminal, con esto poder tener en cuenta todos los comandos ejecutados por un usuario determinado. Esto incluye comandos ejecutados por un usuario en modo o después de cambiar al usuario root. Configurar una regla de Wazuh personalizada para alertar de comandos sospechosos (sudo).

Configuración.

En el terminal final instalaremos Auditd posteriormente crearemos reglas para consultar todos los comandos ejecutados por un usuario con privilegios.

```
sudo apt -y install auditd
sudo systemctl start auditd
sudo systemctl enable auditd
```

Con credenciales root ejecutamos los comandos para agregar reglas de auditoría al archivo */etc/audit/audit.rules*

```
echo "-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b32 -S execve -k audit-wazuh-c" >>
/etc/audit/audit.rules
echo "-a exit,always -F auid=1000 -F egid!=994 -F auid!=-1 -F arch=b64 -S execve -k audit-wazuh-c" >>
/etc/audit/audit.rules
```

Cargamos las reglas para confirmar que están en el directorio correcto.

```
sudo auditctl -R /etc/audit/audit.rules
sudo auditctl -l
```

Debemos agregar la siguiente configuración al archivo del agente permitiendo que el agente lea el archivo de registros auditados en el directorio:

```
sudo nano /var/ossec/etc/ossec.conf
```

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

Para efectuar los cambios reiniciamos el agente de Wazuh.

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

En el servidor debemos ejecutar los pasos siguientes para crear una lista CDB de programas malintencionados y reglas para detectar la ejecución de los programas de la lista. Revisamos los pares clave-valor en el archivo de búsqueda /var/ossec/etc/lists/audit-keys
Esta lista CDB contiene claves y valores separados por dos puntos.

```
audit-wazuh-w:write
audit-wazuh-r:read
audit-wazuh-a:attribute
audit-wazuh-x:execute
audit-wazuh-c:command
```

Creamos una lista CDB y editamos el contenido /var/ossec/etc/lists/suspicious-programs

```
ncat:yellow
nc:red
tcpdump:orange
```

Agregamos la lista a la sección del archivo del servidor Wazuh en la sección **<ruleset>**

```
nano /var/ossec/etc/ossec.conf
```

```
<list>etc/lists/suspicious-programs</list>
```

Seguidamente creamos una regla de gravedad alta para activar cuando se ejecute un programa sospechoso. Para agregar esta nueva regla al archivo en el servidor Wazuh debemos añadir un grupo de reglas ubicado en el directorio de reglas.

```
sudo nano /var/ossec/etc/rules/local_rules.xml
```

```
<group name="audit">
  <rule id="100210" level="12">
     <if_sid>80792</if_sid>
  <list field="audit.command" lookup="match_key_value" check_value="red">etc/lists/suspicious-programs</list>
    <description>Audit: Highly Suspicious Command executed: $(audit.exe)</description>
     <group>audit_command,</group>
  </rule>
</group>
```

Para aplicar los cambios realizados reiniciamos el administrador de Wazuh.

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

Emulación del ataque.

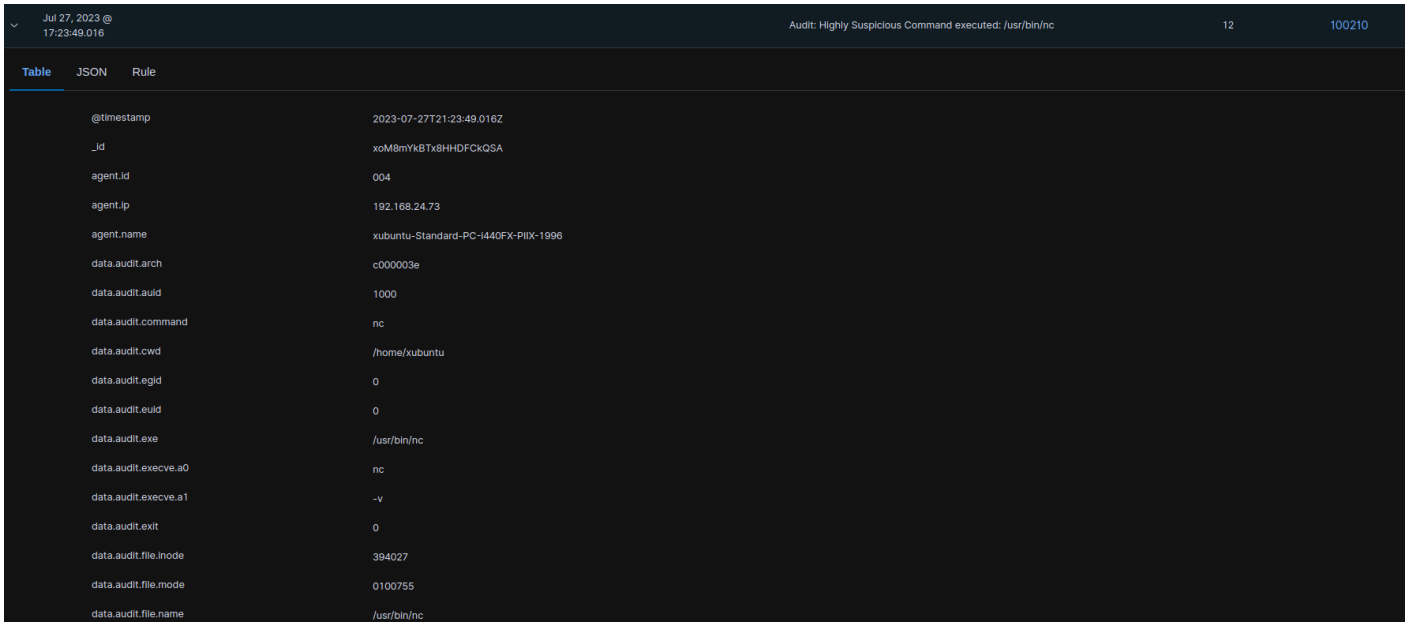
En el terminal de prueba utilizaremos un programa catalogado como “peligroso” intencionalmente con **netcat**.

```
sudo apt -y install netcat
nc -v
```

Resultados.

Con estos comandos instalamos Netcat es una herramienta de red que se utiliza para leer y escribir datos a través de conexiones de red utilizando TCP o UDP.

El segundo comando inicia sesión en modo verbose mostrando la información de forma mas detallada sobre cada paso al realizar la conexión.



PDC 13: DETECCIÓN DE UN ATAQUE DE SHELLSHOCK.

Wazuh es capaz de detectar un ataque Shellshock mediante el análisis de los registros del servidor web recopilados de un punto final monitoreado. En este caso de uso, configura un servidor web Apache en el punto final de víctima y simula un ataque de shellshock.

En este caso se necesitan de 2 terminales el primero el servidor web que sufrirá el ataque de un terminal atacante con sistema operativo Debian emulando el envío de solicitudes HTTP maliciosas al servidor web Apache.

Configuración.

En el terminal víctima se necesitan los paquetes actualizados del servidor Apache:

```
sudo apt update
sudo apt install apache2
```

Si el firewall está activo debe modificarse para permitir el acceso externo a los puertos web.

```
sudo ufw app list
sudo ufw allow 'Apache'
sudo ufw status
```

Comprobamos que el servidor está siendo ejecutado:

```
sudo systemctl status apache2
```

Agregamos las siguientes líneas al archivo de configuración del agente. Esto configura el agente Wazuh para monitorear los registros de acceso de su servidor Apache en el archivo de configuración.

```
nano /var/ossec/etc/ossec.conf
```

```
<localfile>
  <log_format>syslog</log_format>
  <location>/var/log/apache2/access.log</location>
</localfile>
```

Para aplicar los cambios reiniciamos el agente de Wazuh.

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

Emulación del ataque.

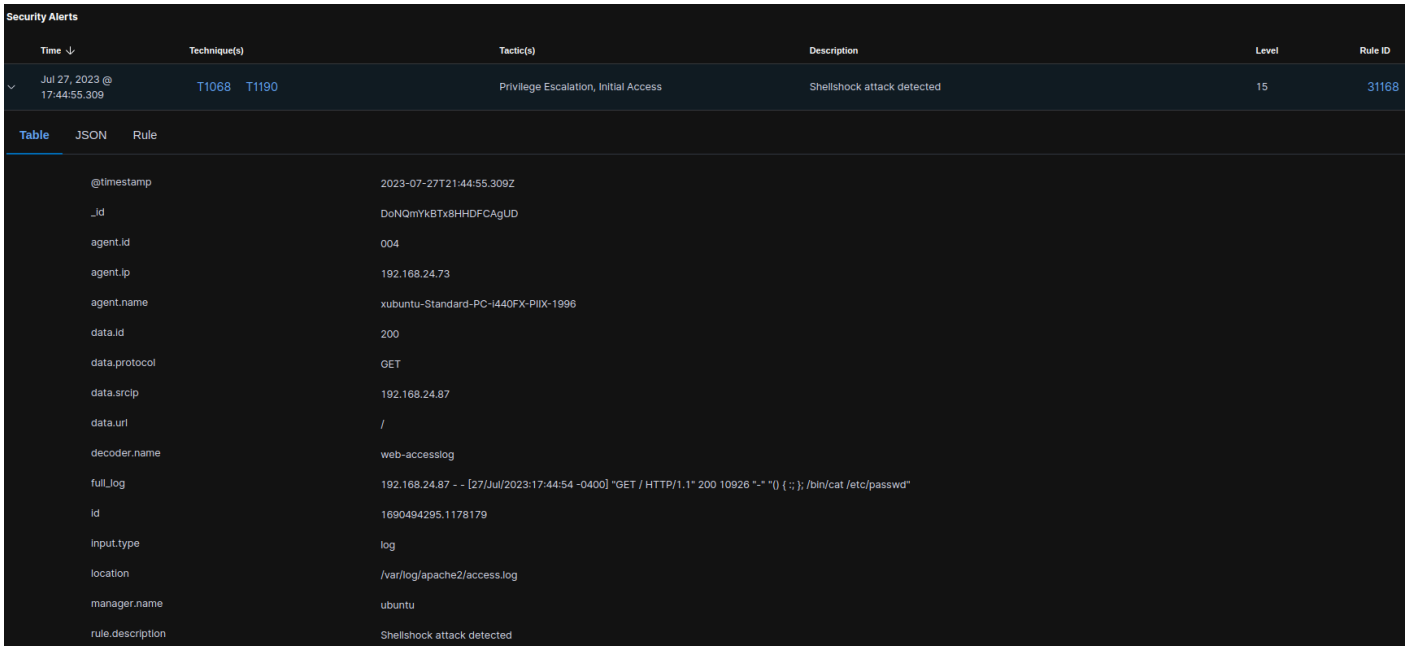
La dirección IP del terminal (192.168.24.73) nos sirve para ejecutar el siguiente comando desde el terminal Debian del atacante.

```
sudo curl -H "User-Agent: () { :; }; /bin/cat /etc/passwd" 192.168.24.73
```

El comando realiza un ataque de Shellshock es una vulnerabilidad de seguridad en el intérprete de línea de comandos Bash que se utiliza en muchos sistemas operativos Linux y Liberation Serif Unix. La vulnerabilidad permite a un atacante ejecutar código arbitrario en un sistema vulnerable enviando un comando especial al intérprete Bash.
El comando en particular que proporcionaste intenta ejecutar el comando /bin/cat /etc/passwd en el sistema vulnerable. El comando /bin/cat /etc/passwd muestra el archivo /etc/passwd, que contiene información sobre todos los usuarios del sistema.
Si el sistema vulnerable está infectado con Shellshock, el comando se ejecutará y el atacante podrá ver la información de todos los usuarios del sistema

Resultados.

Se detecta un ataque de shellshock con nivel de gravedad debido a que se realizaron algunos intentos.



Revision #20
Created 8 abril 2024 11:26:06 by Ricardo Chavez
Updated 6 mayo 2024 17:28:50 by Ricardo Chavez