

# **Laboratorio de Ciberseguridad**

**pfSense + Wazuh + Suricata + MISP**

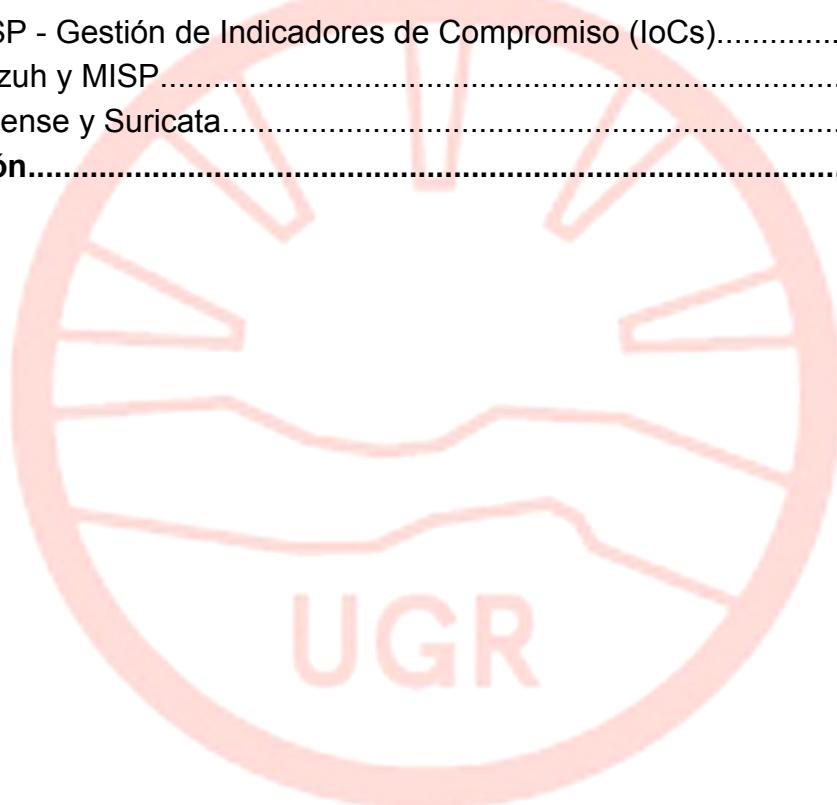


wazuh.



# Contenido

<b>Introducción.....</b>	<b>3</b>
¿Que es este proyecto?.....	3
<b>Desarrollo.....</b>	<b>4</b>
Infraestructura.....	4
Configuración de Herramientas.....	4
Wazuh.....	4
Configuración de reglas en el firewall pfSense.....	7
Configuramos eventos para intentos de sesión fallidos y mapeamos un....	7
evento a una técnica de MITRE ATT&CK.....	7
Suricata.....	9
MISP - Gestión de Indicadores de Compromiso (IoCs).....	12
Wazuh y MISP.....	13
PfSense y Suricata.....	16
<b>Conclusión.....</b>	<b>18</b>



# Introducción

## ¿Que es este proyecto?

En el marco de las actividades prácticas de ciberseguridad de la Universidad del Gran Rosario, desarrollamos un entorno de laboratorio orientado a la **detección, análisis y gestión de amenazas** mediante herramientas **open source** utilizadas en la industria. En este proyecto hacemos una propuesta accesible en términos de costos y escalable para la construcción de una infraestructura segura gracias a **pfSense, Wazuh, Suricata y MISP**.

Dentro del entorno virtual, se va poder distinguir distintos tipos de acciones que se tomaron para la mejora del laboratorio, las cuales son:

- Segmentación de red
- Protección de archivos
- Monitoreo de eventos
- Gestión y configuración de firewall
- Implementación de un agente SIEM
- Alertas automáticas por creación, eliminación o modificación de archivos

Estas decisiones son de vital importancia porque en su conjunto abarcan la mejora de la seguridad de nuestro sistema. Pero además, se demuestra que con materiales **open source** se puede realizar un trabajo eficiente.

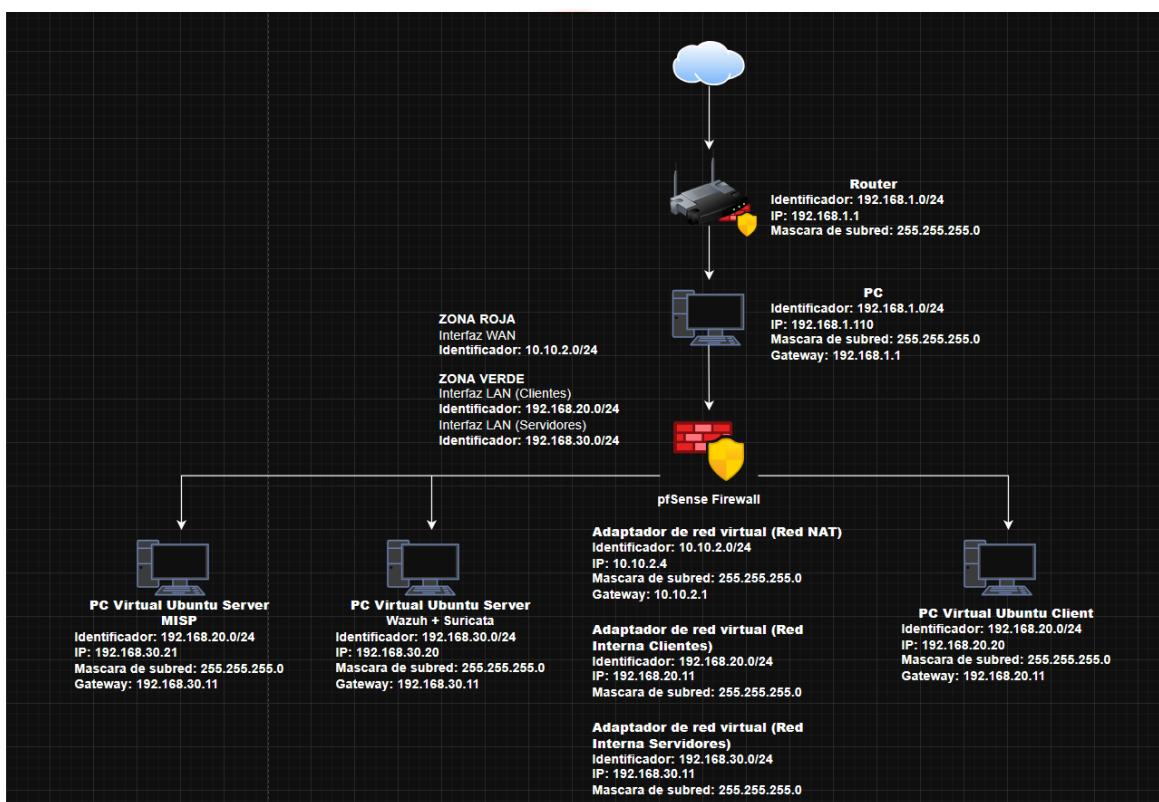
Dentro de la materia de Sistemas de Gestión de Seguridad de la Información, aprendimos que estas herramientas nos ayudan a poder cumplir con **marcos normativos** (como el GDPR, la Ley 25.326, o la ISO 27001).

# Desarrollo

## Infraestructura

La infraestructura del laboratorio virtual esta compuesta por:

- pfSense 2.7.2
- Ubuntu Cliente 24.04 - Cliente con agente Wazuh instalado
- Ubuntu Server 24.04 - Servidor con Wazuh + Suricata
- Ubuntu Server 24.04 - Servidor con MISP



## Configuración de Herramientas

### Wazuh

- Instalación de Wazuh con repositorio oficial en el Servidor Ubuntu 22.04.
- Creación de carpeta /home/Ubuntu/FIM/ en Cliente Ubuntu 22.04.
- Se instaló un agente de Wazuh en el Ubuntu Cliente
- Se configuró el archivo ossec.conf del agente instalado para que monitoree el directorio /home/Ubuntu/FIM/. El mismo se encuentra en la ruta /var/ossec/etc/

## ¿Qué es el archivo ossec.conf?

Este archivo es la configuración principal del agente de Wazuh, y define qué módulos están activos, cómo se comportan, a qué servidor conectarse, qué directorios monitorear, y cómo deben enviarse los logs y eventos.

En nuestro caso, vamos a enfocarnos en el módulo de *syscheck* (File Integrity Monitoring), el cual sirve para detectar cambios no autorizados en archivos críticos del sistema.

### Cambios realizados:

#### 1. *<frequency>60</frequency>*

Indica que Wazuh ejecutará un análisis completo del FIM cada 60 segundos (antes eran 12 horas = 43200 segundos por defecto).

Esto es útil en ambientes de prueba donde querés ver cambios rápidamente, pero no recomendado en producción porque puede generar mucha carga y eventos.

#### 2. *<directories check\_all="yes" report\_changes="yes" realtime="yes">/home/ubuntu/FIM</directories>*

Indica a Wazuh que debe monitorear el directorio /home/ubuntu/FIM con los siguientes parámetros:

##### ***check\_all="yes"***

Hace todas las verificaciones posibles sobre los archivos, como hashes (MD5/SHA1), permisos, propietario, tamaño, fecha de modificación, etc.

##### ***report\_changes="yes"***

Si un archivo cambia, Wazuh intenta generar un diff (diferencia) entre el contenido anterior y el nuevo (si es texto). Es útil para ver exactamente qué cambió.

##### ***realtime="yes"***

En lugar de esperar al análisis programado, Wazuh utiliza inotify (en Linux) para detectar cambios en tiempo real. Apenas un archivo cambia, se reporta el evento.

Luego vamos al dashboard de Wazuh y comprobamos que detecte el activo que acabamos de agregar.

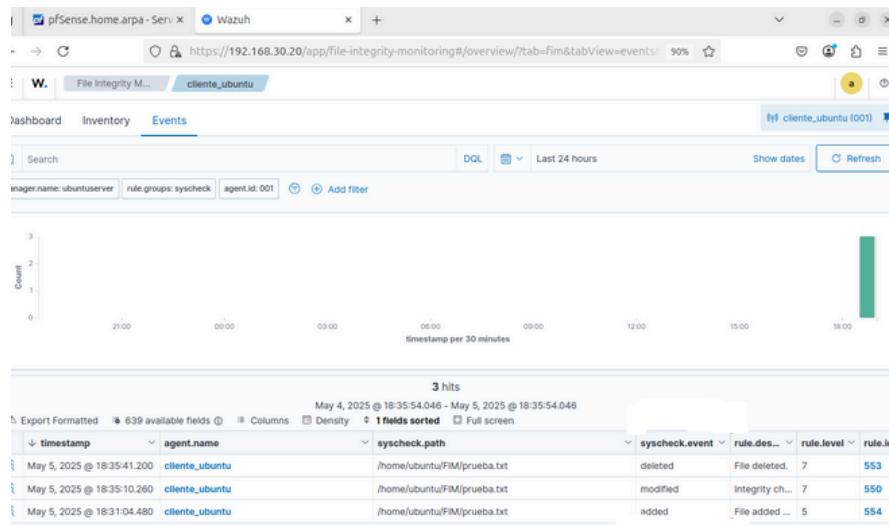
Con estos pasos realizados el File Integrity Monitoring (FIM) se encuentra activo para monitorear cambios en el directorio /home/Ubuntu/FIM/

### Pruebas realizadas en el directorio /home/Ubuntu/FIM/

- Se creó un archivo (prueba.txt) → alerta “added”.
- Se modificó el contenido → alerta “modified”.
- Se eliminó el archivo prueba.txt → alerta “deleted”.

```
ubuntu@ubuntu:~/Desktop$ cd ..
ubuntu@ubuntu:~$ ls
Desktop  Downloads  Music  Public  Templates
Documents  FIM      Pictures  snap    Videos
ubuntu@ubuntu:~$ cd FIM
ubuntu@ubuntu:~/FIM$ touch prueba.txt
ubuntu@ubuntu:~/FIM$
```

- Verificamos que las alertas lleguen con el resultado esperado



## Configuración de reglas en el firewall pfSense

Se configuraron tres reglas en el firewall pfSense ubicado entre el cliente y el servidor, con el objetivo de permitir únicamente el tráfico HTTP y bloquear todo el resto del tráfico:

The figure shows a screenshot of the pfSense Firewall Rules configuration page. The interface includes a sidebar with icons for Firewall, System, Interfaces, Services, VPN, Status, Diagnostics, and Help. The main area is titled 'Firewall / Rules / LAN'. It shows four rules listed under 'Rules (Drag to Change Order)'. The rules are:

States	Protocol	Source	Port	Destination	Port	Gateway	Queue	Schedule	Description	Actions
✓ 1/946 KIB	*	*	*	LAN Address	443 80	*	*		Anti-Lockout Rule	
✗ 0/54 KIB	IPv4 TCP	LAN subnets	*	192.168.30.20	443 (HTTPS)	*	none		Permitir Server Wazuh 192.168.30.20	
✗ 0/0 B	IPv4 TCP	LAN subnets	*	LAN2 subnets	80 (HTTP)	*	none		Permitir HTTP de LAN a LAN2	
✗ 0/4 Kib	IPv4	*	*	LAN2 subnets	*	*	none		Bloquear todo excepto HTTP	

At the bottom of the page, it says 'pfSense is developed and maintained by Netgate. © ESF 2004 - 2025 View license.'

## Configuramos eventos para intentos de sesión fallidos y mapeamos un evento a una técnica de MITRE ATT&CK

En el servidor Wazuh, buscamos la carpeta donde se encuentran las reglas, allí

nos encontramos que hay 2 tipos de ubicaciones de reglas:

- /var/ossec/ruleset/rules -> Ubicación de reglas predefinidas por wazuh
- /var/ossec/etc/rules -> Ubicación para reglas personalizadas (custom rules).

Podemos agregar reglas y quedarnos tranquilos que en futuras actualizaciones no se modificarán.

En el dashboard de Wazuh visualizamos en Explorer/Discover todos los eventos que se van generando, los mismos tienen un rule.id que los identifica, por lo que generamos un ataque de fuerza bruta con Hydra desde el cliente al servidor para generar eventos de inicio de sesión fallido al servidor, como consecuencia obtuvimos el rule.id=5760 y el rule.description="Failed password for"

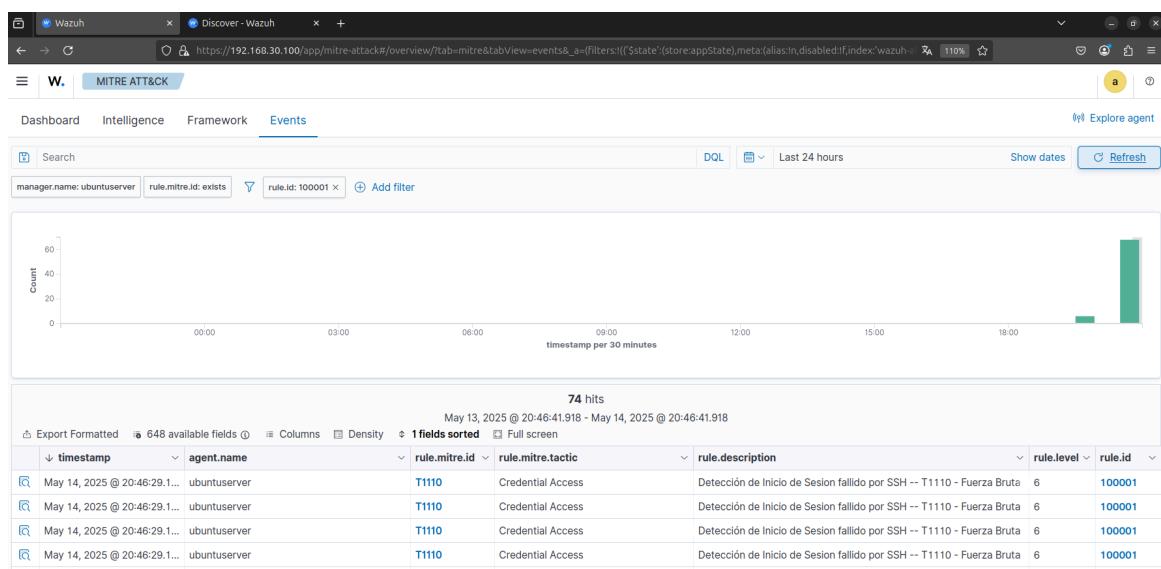
A partir de esta información, se creó una regla personalizada en /var/ossec/etc/rules que permite detectar este tipo de eventos de forma más precisa y asociarlos a la técnica T1110 – Brute Force del framework MITRE ATT&CK.

```
root@ubuntuserver: /var/ossec/etc/rules          x
GNU nano 6.2                                     local_rules.xml
<!!-- Local Rules -->
<!!-- Modify it at your will. -->
<!!-- Copyright (C) 2015, Wazuh Inc. -->

<group name="ssh,mitre,">
    <rule id="100000" level="0">
        <decoded_as>sshd</decoded_as>
        <description>SSH conexiones</description>
    </rule>

    <rule id="100001" level="6">
        <if_sid>5760</if_sid>
        <match>Failed password for</match>
        <description>Deteción de Inicio de Sesión fallido por SSH -- T1110 - Fuerza Bruta</description>
        <mitre>
            <id>T1110</id>
        </mitre>
        <group>default,</group>
    </rule>
</group>
```

Volvemos a ejecutar un ataque de fuerza bruta y los eventos pudieron visualizarse correctamente desde el dashboard de Wazuh, evidenciando la eficacia del sistema en la detección de intentos de acceso no autorizado.



## Suricata

Objetivo: Instalar y configurar una regla para detectar tráfico anómalo y vincular la regla a la técnica de MITRE ATT&CK T1046 de exploración de red.

Instalación:

- sudo apt-get install software-properties-common
- sudo add-apt-repository ppa:oisf/suricata-stable
- sudo apt update
- sudo apt install suricata -y
- sudo systemctl status suricata
- sudo suricata-update (nos genera un suricata.rules con reglas predefinidas)

Dentro del archivo suricata.rules agregamos la siguiente regla personalizada para detectar tráfico anómalo.

Alert tcp any any -> any any (msg:"[ATT&CK] T1046 - Escaneo de puertos detectado - Nmap Scan"; flags:S;threshold:type both, track\_by\_src, count 10, seconds 10; classtype:network-scan; sid:100010; rev:1;)

Luego agregamos suricata.rules a el archivo de configuración de suricata para que lo detecte -> /etc/suricata/

Verificamos la sintaxis de la regla establecida con:

➤ sudo suricata -T -c /etc/suricata/suricata.yaml -v

```

root@ubuntuserver:/var/lib/suricata/rules# suricata -T -c /etc/suricata/suricata.yaml -v
[notice]: suricata: This is suricata version 7.0.10 RELEASE running in SYSTEM mode
[info]: cpu: CPUs/cores online: 2
[info]: suricata: Running suricata under test mode
[info]: suricata: Setting engine mode to IDS mode by default
[info]: exception-policy: master exception-policy set to: auto
[info]: logopenfile: fast output device (regular) initialized: fast.log
[info]: logopenfile: eve-log output device (regular) initialized: eve.json
[info]: logopenfile: stats output device (regular) initialized: stats.log
[info]: detect: 1 rule files processed. 43478 rules successfully loaded, 0 rules failed, 0
[info]: threshold config parsed: 0 rule(s) found
[info]: detect: 43481 signatures processed. 1227 are IP-only rules, 4346 are inspecting packet payload, 37689 inspect application layer, 109 are decoder event only
[Notice]: suricata: Configuration provided was successfully loaded. Exiting.
root@ubuntuserver:/var/lib/suricata/rules#

```

Y después reiniciamos suricata para que tome efecto la regla establecida:

➤ sudo systemctl restart suricata

Luego tirar un tail sobre el fast.log, y al mismo tiempo desde una máquina cliente tirarle un escaneo con nmap a la IP del servidor

➤ sudo nmap -sS -p- --min-rate 5000 192.168.30.100 -Pn -vvv

```

ubuntuserver@ubuntuserver:~$ sudo tail -f /var/log/suricata/fast.log
[sudo] password for ubuntuserver:
05/15/2025-01:21:32.864850 [**] [1:100010:1] [ATT&CK] T1046 - Escaneo de puertos detectado - Nmap
Scan [**] [Classification: Detection of a Network Scan] [Priority: 3] {TCP} 192.168.20.100:51062 ->
192.168.30.100:53

```

Para realizar la integración de Suricata con Wazuh desde el servidor realizamos:

➤ sudo nano /var/ossec/etc/ossec.conf

Y agregamos lo siguiente:

```

root@ubuntuserver:/var/ossec/etc
GNU nano 6.2                               ossec.conf *
<log_format>journald</log_format>
<location>journald</location>
</localfile>

<localfile>
<log_format>syslog</log_format>
<location>/var/ossec/logs/active-responses.log</location>
</localfile>

<localfile>
<log_format>syslog</log_format>
<location>/var/log/dpkg.log</location>
</localfile>

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

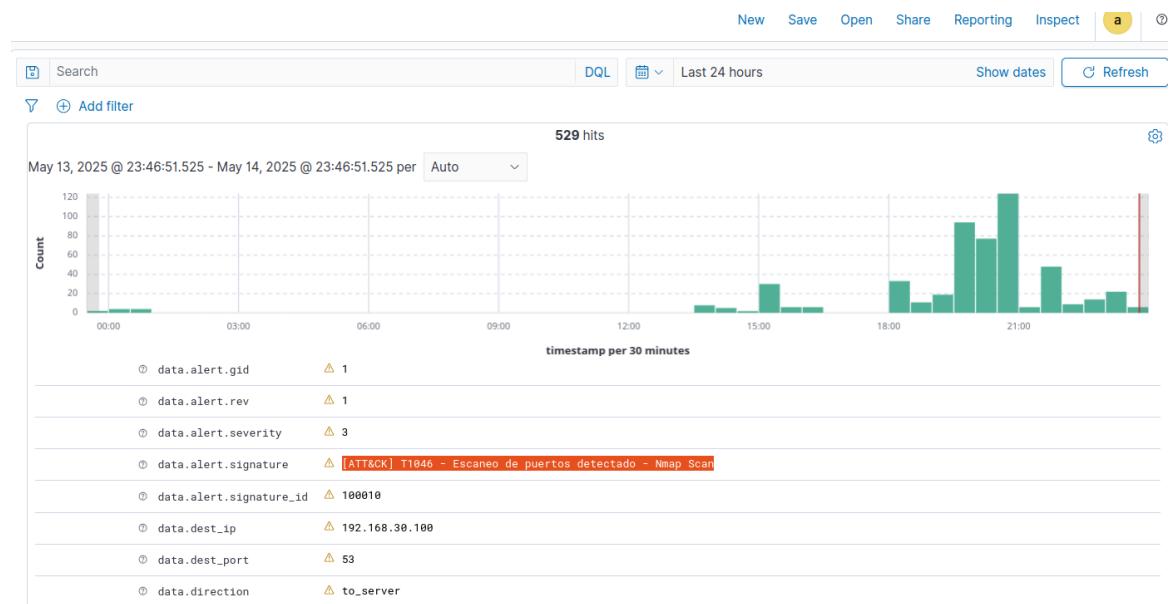
</ossec_config>

^G Help      ^O Write Out  ^W Where Is   ^K Cut      ^T Execute    ^C Location
^X Exit      ^R Read File  ^\ Replace    ^U Paste     ^J Justify    ^/ Go To Line

```

Desde el cliente realizamos un nmap para verlo en el dashboard de wazuh con el sid 100010

➤ nmap -sS -p- --min-rate 5000 192.168.30.100 -Pn -vvv



Luego, se agregaron al servidor las reglas custom para este evento relacionandolo con MITRE ATT&CK.

```
root@ubuntuserver:/var/ossec/etc/rules
GNU nano 6.2                                         local_rules.xml

<!-- Local Rules -->

<!-- Modify it at your will. -->
<!-- Copyright (C) 2015, Wazuh Inc. -->

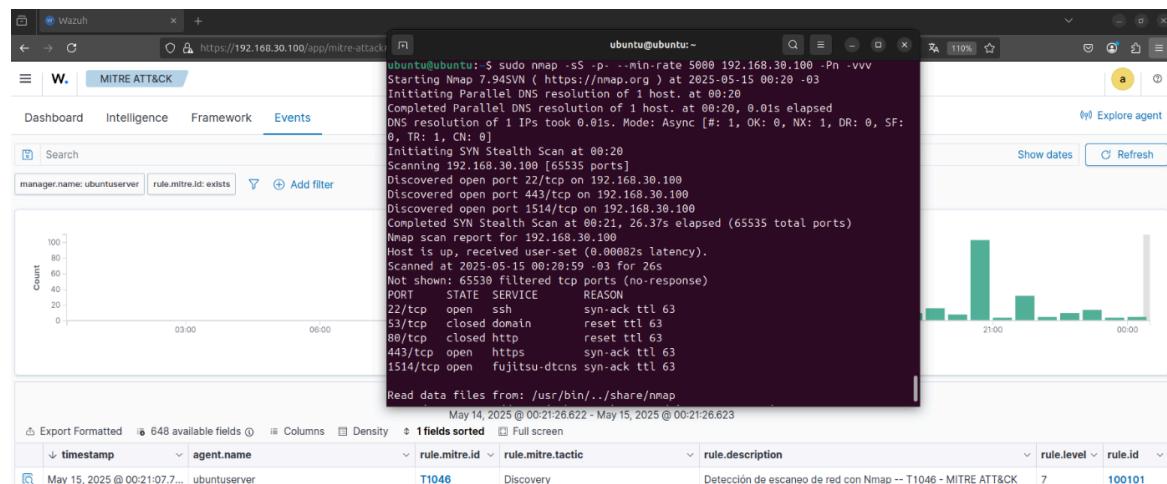
<group name="ssh,mitre,">
    <rule id="100000" level="0">
        <decoded_as>sshd</decoded_as>
        <description>SSH conexiones</description>
    </rule>

    <rule id="100001" level="6">
        <if_sid>5760</if_sid>
        <match>Failed password for</match>
        <description>Detección de Inicio de Sesión fallido por SSH -- T1110 - Fuerza Bruta</description>
        <mitre>
            <id>T1110</id>
        </mitre>
        <group>default,</group>
    </rule>
</group>

<group name="suricata,network_scan,mitre,">
    <rule id="100101" level="7">
        <if_sid>86601</if_sid>
        <match>[ATT&CK] T1046 - Escaneo de puertos detectado - Nmap Scan</match>
        <description>Detección de escaneo de red con Nmap -- T1046 - MITRE ATT&CK</description>
        <mitre>
            <id>T1046</id>
        </mitre>
        <group>default,</group>
    </rule>
</group>
```

A continuación, agregamos en el servidor de Wazuh las reglas custom para este evento en el archivo ossec.conf (/var/ossec/etc/ossec.conf) para relacionarlo con MITRE ATT&CK.

Ejecutamos un último nmap al servidor para probar la integración que acabamos de realizar.



Obtenemos el resultado esperado: una demostración de un evento para detectar tráfico anómalo y la vinculación de la misma a la técnica de MITRE ATT&CK T1046 de exploración de red.

## MISP - Gestión de Indicadores de Compromiso (IoCs)

- En el servidor Ubuntu 24.04 instalamos MISP siguiendo los siguientes pasos:

- Actualizar el sistema -> sudo apt update && sudo apt upgrade -y
- Instalar paquetes necesarios -> sudo apt install -y curl git gnupg
- Descargar el script oficial de instalación para Ubuntu 24.04 -> wget [https://raw.githubusercontent.com/MISP/MISP/2.5/INSTALL/INSTALL\\_L.ubuntu2404.sh](https://raw.githubusercontent.com/MISP/MISP/2.5/INSTALL/INSTALL_L.ubuntu2404.sh) -O INSTALL.sh
- Dar permisos de ejecución al script -> chmod +x INSTALL.sh
- Ejecutar el script como root (instala MISP) -> sudo ./INSTALL.sh

- Agregar hostname local para acceder a MISP como "https://misp.local" -> sudo nano /etc/hosts -> 127.0.0.1 localhost misp.local
- Acceder al portal MISP desde: https://misp.local

Una vez instalado, se procedió a importar un feed público de amenazas, en este caso el de CIRCL.lu, que contiene información útil sobre direcciones IP, dominios y otros IoCs.

Posteriormente, creamos manualmente un evento que incluyó un Indicador de Compromiso (IoC) simulado, representado por una dirección IP maliciosa ficticia. Esto permitió probar la funcionalidad básica de MISP y demostrar su utilidad para compartir y analizar amenazas de forma colaborativa.

The screenshot shows the MISP web interface with the following details:

- Event ID:** 129
- UUID:** b45344e8-030e-4c3e-a1b9-57213e1905a4
- Creator org:** ORGANIZATION
- Owner org:** ORGANIZATION
- Creator user:** admin@admin.test
- Protected Event (experimental):** Event is in unprotected mode.
- Tags:** (empty)
- Date:** 2025-05-15
- Threat Level:** High
- Analysis:** Initial
- Distribution:** Your organization only
- Warnings:** Confirmation warning: Your event has neither tags nor galaxy clusters attached - generally adding context to an event allows for quicker decision making and more accurate filtering. It is highly recommended that you label your events to the best of your ability.
- Published:** Yes (2025-05-15 23:50:27)
- #Attributes:** 1 (0 Objects)
- First recorded change:** 2025-05-15 23:43:50
- Last change:** 2025-05-15 23:43:50
- Modification map:** (empty)
- Sightings:** 0 (0) - restricted to own organization only.
- Navigation:** Events > Galaxy > Event graph > Event timeline > Correlation graph > Galaxy matrix > Event reports > Attributes > Discussion
- Galaxies:** (empty)
- Filtering:** Date: 2025-05-15, Context: Network activity, Type: ip-dst, Value: 195.100.87.202
- Related:** Related Events, Feed hits, iOTS, Distribution, Sightings, Activity, Actions

## Wazuh y MISP

- Integramos alertas con MISP para compartir IoCs y mapee una alerta a una técnica de MITRE ATT&CK (por ejemplo, T1078 - Credenciales Válidas).
- Vinculamos un evento a una técnica de MITRE ATT&CK

## Prerrequisitos

- Instalar un cliente Windows y agregarle un agente de Windows (esto lo hacemos ya que trackear eventos de red de linux requiere herramientas

- externas (como auditd, netstat, etc.), incluso de esta manera corremos riesgo de perdernos información).
- Al cliente Windows debemos instalar Sysmon desde su web oficial (<https://learn.microsoft.com/en-us/sysinternals/downloads/sysmon>)
  - Descargar la configuración de Sysmon (<https://github.com/SwiftOnSecurity/sysmon-config>)
  - Instalar la configuración descargada
    - sysmon.exe -accepteula -i sysmon-config/sysmonconfig-export.xml

## Integrar Sysmon con Wazuh

- En el archivo del agente Wazuh (ossec.conf, ubicado en C:\Program Files (x86)\ossec-agent) agregar lo siguiente:

```
<localfile>
  <location>Microsoft-Windows-Sysmon/Operational</location>
  <log_format>eventchannel</log_format>
</localfile>
```

- Reiniciar el servicio de Wazuh-Agent
  - net stop wazuh
  - net start wazuh
- Probamos funcionamiento de la integración, ejecutamos un ping a la ip maliciosa seteada en la etapa 2 (185.100.87.202) y buscamos un evento con el atributo *win.event.channel*:  
"Microsoft-Windows-Sysmon/Operational"

En **MISP** debemos obtener una API KEY, para esto dentro de la interfaz web debemos ir a Administration -> List Auth Keys -> Add authentication key y generar una nueva

En el servidor de **Wazuh**, debemos ir a la ruta /var/ossec/integrations/ y crear un archivo llamado **custom-misp**, en el mismo copiamos el script del siguiente repositorio

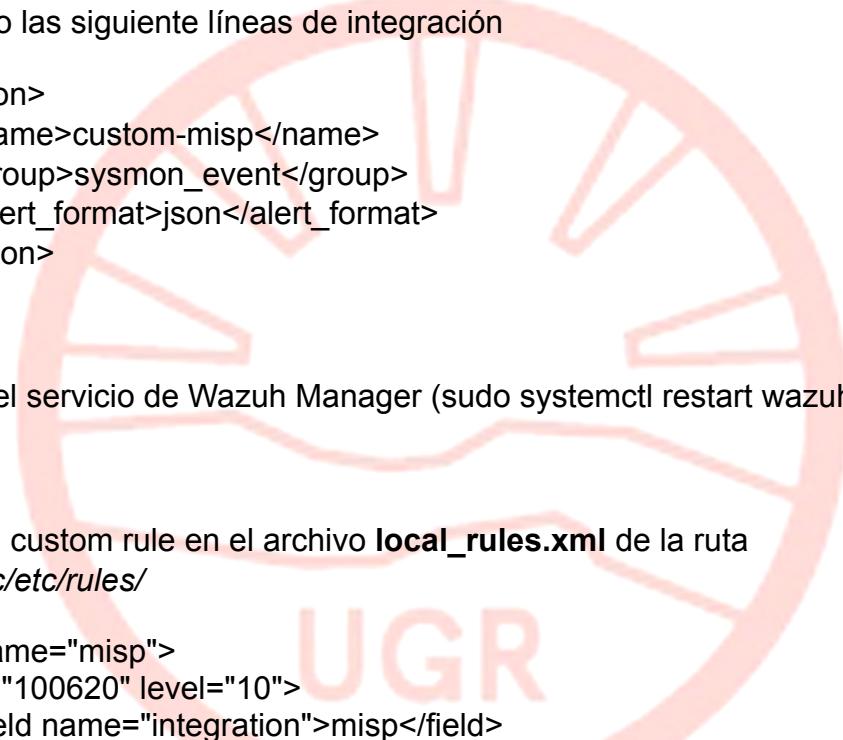
(<https://gist.github.com/OpenSecureCo/65b65be1b2cb170dcea9d6cf09710b38#file-custom-misp-py>) Este script nos ayudará a comunicar Wazuh con MISP para determinar si los eventos son un IoC o no.

IMPORTANTE: del archivo que acabamos de generar debemos realizar cambios en algunos valores

- misp\_base\_url : "<https://misp.local/attributes/restSearch/>"
- misp\_api\_auth\_key : "API\_KEY\_GENERADA"

Cambiar permisos del archivo:

- sudo chown root:wazuh /var/ossec/integrations/custom-misp
- sudo chmod 750 /var/ossec/integrations/custom-misp



```

root@wazuh-server:/var/ossec/integrations
GNU nano 8.3          custom-misp
    sock.send(string.encode())
    sock.close()

    False = False
    # Read configuration parameters
    alert_file = open(sys.argv[1])
    # Read the alert file
    alert = json.loads(alert_file.read())
    alert_file.close()
    # New Alert Output if MISP Alert or Error calling the API
    alert_output = {}
    # MISP Server Base URL
    misp_base_url = "https://**your_misp_instance**/attributes/restSearch/"
    # MISP Server API AUTH KEY
    misp_api_auth_key = "*Your API Key"
    # API - HTTP Headers
    misp_apicall_headers = {
        "Content-Type": "application/json",
        "Authorization": f"{misp_api_auth_key}",
    }

```

También debemos editar el archivo **ossec.conf** de la ruta `/var/ossec/etc` agregando las siguientes líneas de integración

```

<integration>
    <name>custom-misp</name>
    <group>sysmon_event</group>
    <alert_format>json</alert_format>
</integration>

```

Reiniciar el servicio de Wazuh Manager (`sudo systemctl restart wazuh-manager`)

Crear una custom rule en el archivo **local\_rules.xml** de la ruta `/var/ossec/etc/rules/`

```

<group name="misp">
    <rule id="100620" level="10">
        <field name="integration">misp</field>
        <match>misp</match>
        <description>MISP Events</description>
        <options>no_full_log</options>
    </rule>
    <rule id="100622" level="12">
        <if_sid>100620</if_sid>
        <field name="misp.category">.+</field>
        <description>MISP - IOC match Category: ${misp.category}, Attribute: ${misp.value}</description>
        <mitre>
            <id>T1190</id>
        </mitre>
        <group>misp_alert</group>
    </rule>
</group>

```

Desde el cliente Windows podemos tirar un ping a la ip con el indicador de compromiso o un dnslookup y de esta manera obtendremos un evento de MITRE

```
C:\Users\Administrator>ping tikonam.com  
Pinging tikonam.com [108.62.12.189] with 32 bytes of data:  
Reply from 173.208.124.9: Destination host unreachable.  
Reply from 173.208.124.9: Destination host unreachable.
```

```
t data.misp.category Network activity  
t data.misp.event_id 2233  
t data.misp.source.description Sysmon - Event 22: DNS Request by C:\Windows\system32\PING.EXE  
t data.misp.type domain  
t data.misp.value tikonam.com  
t decoder.name json  
t description MISP - IoC found in Threat Intel - Category: Network activity, Attribute: tikonam.com  
t id 1648743030.565533  
t input.type log  
t location misp  
t manager.name hydra  
t rule.description MISP - IoC found in Threat Intel - Category: Network activity, Attribute: tikonam.com  
# rule.firedtimes 2  
t rule.groups misp, misp_alert  
t rule.id 100622  
# rule.level 12
```

## PfSense y Suricata

Habilitamos Suricata desde el panel de PfSense:

Menú:

➤ System > Package Manager > Available Packages

Buscamos Suricata seleccionamos Install y luego en Confirm.

Creamos una regla personalizada de Suricata:

➤ Services > Suricata > Rules  
➤ Seleccionamos la interfaz y luego Custom Rules

Agregamos una regla por DNS sospechoso:

- alert udp any any -> any 53 (msg:"ALERTA DNS a dominio malware-checkup.com sospechoso"; content:"malware-checkup.com"; nocase; sid:1000011; rev:1;)

Para disparar la alerta desde un cliente:

- nslookup malware-checkup.com

La alerta nos aparece en el dashboard dentro de Suricata:

Date	Action	Pri	Proto	Class	Src	SPort	Dst	DPort	GID:SID	Description
05/25/2023 10:52:42	!	2	UDP	Potentially Bad Traffic	192.168.10.10	59046	192.168.10.1	53	1.2027758	ET DNS Query for .cc TLD
05/25/2023 10:52:40	!	2	UDP	Potentially Bad Traffic	192.168.10.10	59045	192.168.10.1	53	1.2027758	ET DNS Query for .cc TLD



# Conclusión

El laboratorio virtual desarrollado integra múltiples herramientas de ciberseguridad de código abierto para simular un entorno de monitoreo, detección y análisis de amenazas en tiempo real. La implementación de **Wazuh** como **SIEM** centralizado permite gestionar eventos desde distintos sistemas, con especial foco en el monitoreo de integridad de archivos (FIM), la correlación de eventos y la detección de ataques como fuerza bruta (técnica T1110 de MITRE ATT&CK). Por su parte, Suricata ofrece capacidades de inspección profunda de paquetes, detectando escaneos de red mediante reglas personalizadas.

La inclusión de MISP permite centralizar y compartir IoCs, lo cual se integra exitosamente con Wazuh para automatizar la detección de eventos maliciosos y vincularlos al marco MITRE ATT&CK.

La configuración del firewall pfSense permite controlar el tráfico y reforzar la segmentación de red, y su integración con Suricata refuerza las capacidades de detección en el perímetro.

Este laboratorio valida el funcionamiento conjunto de las herramientas y también demuestra un enfoque práctico y escalable para la defensa frente a amenazas con herramientas **open source**.

