Universidad CENFOTEC

Maestría profesional en Ciberseguridad

Documento final Proyecto de Investigación Aplicada 2

Titulo

Modelo de recuperación autónomo para entornos virtuales ante incidentes de seguridad.

Autor

Ing. David Solano Mora

Tutor

MSeg. Mario Andrés Zamora Madriz

Diciembre de 2025

TRIBUNAL EXAMINADOR

Este proyecto fue aprobado por el Tribunal Examinador de la carrera: Maestría Profesional en Ciberseguridad, requisito para optar por el título de grado de **Maestría,** para el estudiante: David Solano Mora.

Digitally signed by MIGUEL PEREZ MONTERO (FIRMA)
Date: 2026.03.03 07:58:51 -06'00'

M.Sc. Mario Zamora Madriz
Tutor

M.Sc. Miguel Pérez Montero
Lector 1

REBECA ESQUIVEL FLORES (FIRMA)
Firmado digitalmente por REBECA ESQUIVEL FLORES (FIRMA)
Fecha: 2026.03.03 14:42:43 -06'00'

M.Sc. Rebeca Esquivel Flores
Lector 2

San José, Costa Rica, 02 de marzo de 2026

Declaratoria

Yo, David Solano Mora, autor del trabajo titulado “Modelo de recuperación autónomo para entornos virtuales ante incidentes de seguridad”, presentado como requisito para optar por el grado de Maestría en Ciberseguridad en la Universidad Cenfotec, autorizo a la Universidad Cenfotec a poner a disposición de la comunidad académica este documento en su biblioteca y en los repositorios institucionales que considere pertinentes, con el fin de que pueda ser consultado, reproducido o utilizado con propósitos exclusivamente académicos, educativos o de investigación.

Esta autorización no implica la cesión de los derechos de autor sobre el contenido del trabajo. Los derechos intelectuales derivados de esta investigación continúan siendo propiedad del autor, por lo que cualquier uso distinto al estrictamente académico deberá contar con la autorización previa y expresa del autor.

Asimismo, se permite la reproducción parcial del documento siempre que se realice con fines académicos y se cite adecuadamente la fuente correspondiente.

Dedicatoria

Dedico este trabajo, en primer lugar, a Dios, por darme la fortaleza, la sabiduría y la oportunidad de culminar esta importante etapa de formación académica.

A mi esposa, por su amor, paciencia y apoyo constante durante todo este proceso. Su comprensión y motivación fueron fundamentales para seguir adelante en los momentos de mayor exigencia.

A mis hijos, quienes con su alegría, amor y energía han sido siempre una fuente de inspiración para esforzarme cada día y procurar ser un mejor ejemplo para ellos.

A mis padres y hermanos, por su apoyo, sus palabras de aliento y por motivarme siempre con su ejemplo de esfuerzo, trabajo y perseverancia.

Asimismo, expreso mi sincero agradecimiento a todas aquellas personas que, de una u otra forma, brindaron su apoyo durante el desarrollo de este trabajo. Ya fuera mediante una palabra de aliento, un consejo oportuno o su acompañamiento a lo largo de este proceso, su contribución fue valiosa para poder culminar satisfactoriamente esta etapa académica.

Modelo de recuperación autónomo para entornos virtuales ante incidentes de seguridad (diciembre 2025).

Ing. David Solano Mora[1], MSeg. Mario Andrés Zamora Madriz, M.[2]
[1] dsolanomo@ucenfotec.ac.cr
[2] mzamora@ucenfotec.ac.cr

RESUMEN. Hoy en día, las pequeñas y medianas empresas (PYMEs) son especialmente vulnerables a los ciberataques debido a sus limitaciones de presupuesto y personal especializado. Este trabajo final de graduación propone un modelo de recuperación autónomo ante incidentes de seguridad que prioriza la restauración de los sistemas después de un ciberataque y la continuidad operativa utilizando herramientas de código abierto. Esta arquitectura integra Wazuh para la detección de amenazas y respuesta activa (active response), *scripting* en Python para la orquestación de la respuesta y VirtualBox (VBoxManage) para la recuperación en entornos virtualizados. Se establecen criterios de activación, trazabilidad y un instrumento de medición aplicado en un laboratorio controlado, mediante el cual se evalúan métricas de desempeño como el tiempo medio de recuperación (MTTR) y la distribución de tiempos por fase del flujo automatizado. El aporte principal de este trabajo es un "mini-SOAR" de bajo costo, replicable y alineado con el contexto de las PYMEs que buscan reducir sus tiempos de recuperación y mejorar la ciber resiliencia sin depender de licencias propietarias. Como limitación, la validación del modelo se realiza en un entorno de laboratorio con un conjunto acotado de escenarios de ataque y una topología específica, por lo que se recomienda en trabajos futuros ampliar el número de casos de prueba y entornos evaluados para reforzar la evidencia empírica sobre su eficacia y viabilidad de adopción.

ABSTRACT. Today, small and medium-sized enterprises (SMEs) are especially vulnerable to cyberattacks due to limited budgets and a lack of specialized personnel. This final graduation project proposes an autonomous recovery model for security incidents that prioritizes system restoration after a cyberattack and business continuity using open-source tools. The architecture integrates Wazuh for threat detection and active response, Python scripting for response orchestration, and VirtualBox (VBoxManage) for recovery in virtualized environments. Activation criteria, traceability mechanisms, and a measurement instrument are defined and applied in a controlled laboratory, where performance metrics such as mean time to recovery (MTTR) and the distribution of times per phase of the automated workflow are evaluated. The main contribution of this work is a low-cost, replicable "mini-SOAR" aligned with the context of SMEs that seek to reduce recovery times and improve cyber resilience without relying on proprietary licenses. As a limitation, validation of the model is performed in a laboratory environment with a limited set of attack scenarios and a specific topology, so future work should expand the number of test cases and evaluated environments to strengthen the empirical evidence regarding its effectiveness and feasibility of adoption.

PALABRAS CLAVE. Ciber resiliencia, respuesta a incidentes, Wazuh, SOAR, VirtualBox, Python, snapshots, PYMEs, recuperación autónoma.

I. INTRODUCCIÓN.

Las pequeñas y medianas empresas (PYMEs) realizan sus operaciones bajo un presupuesto restringido y con falta de personal especializado, lo que causa un incremento en el riesgo cibernético. La historia demuestra que el *ransomware* continúa siendo un vector de alto impacto que afecta a la mayoría de los sectores y ponen bajo tensión los recursos utilizados por las organizaciones para mantener su continuidad operativa y capacidad de recuperación [1]. Según ENISA, la denegación del servicio y el ransomware se encuentran entre las principales amenazas y esto fuerza a las organizaciones a priorizar los mecanismos de recuperación para minimizar los tiempos de inactividad [2].

Tomando en cuenta este escenario, es importante que la resiliencia cibernética en las PYMEs integre una detección confiable, orquestación de respuestas y una recuperación automatizada. La guía NIST SP 800-61 Rev. 3 habla de preparar, contener, erradicar y recuperar de forma coordinada, con trazabilidad y mejora continua a lo largo del ciclo de vida de un incidente [3].

Este trabajo propone un modelo de recuperación autónomo ante incidentes de seguridad de bajo costo ya que se basa en tecnologías de código abierto y está orientado al contexto de las PYMEs. Su arquitectura integra Wazuh para la detección, Python para la orquestación de decisiones y VBoxManage (Virtual Box) para la restauración luego de un incidente utilizando *snapshots*. Se definen diferentes criterios de activación, trazabilidad y se utiliza un instrumento de medición de tiempos. La intención es reducir el tiempo objetivo de recuperación (RTO) y sostener la continuidad operativa con una solución que sea replicable y que no se requieran licencias. Se pretende montar el modelo para posteriormente, ejecutar escenarios controlados y analizar las métricas con lo que se podrá evaluar la eficacia y riesgos de falsos positivos, alineado con las mejores prácticas de respuesta a incidentes.

II. MATERIALES Y MÉTODOS.

Saber cómo funciona la recuperación autónoma de los sistemas ante incidentes, es una parte fundamental en la ciberseguridad de las PYMEs ya que esto ayuda a comprender bien la manera en que interactúan sus componentes (detección, orquestación y restauración) y adonde es que se concentran las vulnerabilidades. Esto permite identificar los puntos de fallo en la cadena de detección y poder diseñar las mejores estrategias que ayuden a reducir los tiempos objetivos de recuperación (RTO) para poder preservar la continuidad operativa tras un ciberataque.

Para este estudio se utilizó una metodología mixta que combinó un diseño experimental en un entorno controlado con análisis cualitativo y cuantitativo. La intención fue medir el comportamiento del modelo propuesto y recopilar información acerca de la eficacia en la recuperación autónoma en escenarios típicos de compromiso.

Por una parte, se realizó un análisis cualitativo para mapear la arquitectura y los flujos de decisión del modelo, lo que implicó:

- Revisar y ajustar reglas de Wazuh asociados a eventos de alta severidad
- Definir criterios de activación que deciden cuando se debe ejecutar cierta acción como la restauración automática de un *snapshot*
- Verificar la trazabilidad de los procesos

Por otro lado, se realizó un análisis cuantitativo para obtener métricas objetivas del desempeño del modelo. Se midieron tiempos de detección (desde que se produce el evento hasta que se produce la alerta), tiempos de respuesta (desde que se da la alerta hasta que inicia la acción automática) y tiempos de recuperación (desde que inicia la respuesta hasta que el sistema esta recuperado y en servicio). Adicional a esto, se registraron las tasas de éxito y los errores producidos en el proceso.

Además, se realizó una investigación exploratoria para seleccionar escenarios de prueba representativos. Para esto se debió revisar las buenas prácticas y la documentación técnica de las herramientas utilizadas con el fin de alinear los procedimientos de detección, respuesta y recuperación al contexto de las PYMEs y las limitaciones económicas y técnicas que tienen, así como a las limitaciones que conlleva un entorno virtual.

En cuanto a los materiales del estudio, para este laboratorio se utilizó una máquina *host* (física) con VirtualBox donde, por una parte, se inicializó una máquina con el sistema operativo Ubuntu con una instancia de servidor de Wazuh y un *dashboard*. Por otra parte, dos máquinas objetivo (Debian 12 y Windows 10) donde se instaló el agente de Wazuh y son monitoreadas en tiempo real por el servidor. La orquestación se realizó mediante un *script* desarrollado en Python 3, el cual, a partir de los parámetros que recibe en su ejecución, decide y ejecuta a través de VBoxManage la secuencia de aislamiento, restauración y arranque de la máquina comprometida.

Es importante que exista una detección temprana y acciones automáticas confiables, por esto, con la implementación de reglas de alerta y active response (Wazuh), es posible reconocer patrones típicos de un *ransomware* como cambio masivo en archivos y activar de forma oportuna la orquestación que aísla el activo, restaura el *snapshot* seguro y recupera la operatividad del servicio.

En el modelo desarrollado, la detección del incidente que dispara el flujo de contención y recuperación automática se basa en las capacidades de correlación y monitoreo de Wazuh. En particular, se emplea el módulo de File Integrity Monitoring (FIM), conocido como syscheck, el cual permite supervisar cambios en el sistema de archivos (creación, modificación o eliminación de archivos) y, en implementaciones más amplias, incluso el registro de Windows. Sobre esta base, se definieron reglas personalizadas en el motor de correlación de Wazuh para identificar patrones de comportamiento compatibles con un ataque de *ransomware*, tales como la modificación masiva de archivos en directorios críticos en un intervalo corto de tiempo, siguiendo el enfoque sugerido por la propia documentación y artículos especializados sobre detección de *ransomware* mediante FIM. [4]

Estas reglas personalizadas se configuraron con un nivel de criticidad elevado (rule.level ≥ 15), de acuerdo con la clasificación de niveles de alerta de Wazuh, donde los niveles más altos representan eventos de alta importancia o posible compromiso activo [5]. De esta forma, únicamente aquellos eventos que cumplen condiciones estrictas de volumen de cambios, tipo de archivo y contexto (por ejemplo, actividad concentrada en directorios de datos de usuario) son considerados suficientes para activar el flujo de respuesta automatizada.

```
</group>

<group name="fim">
 <rule id="100002" level="7" overwrite="yes">
   <if_sid>554</if_sid>
   <description>FIM - creacion de archivo con nivel elevado</description>
   <group>fim_burst_base,syscheck,</group>
 </rule>

 <rule id="100003" level="7" overwrite="yes">
   <if_sid>553</if_sid>
   <description>FIM - eliminacion de archivo con nivel elevado</description>
   <group>fim_burst_base,syscheck,</group>
 </rule>

<rule id="100004" level="7" overwrite="yes">
   <if_sid>550</if_sid>
   <description>FIM - modificacion de archivo con nivel elevado</description>
   <group>fim_burst_base,syscheck,</group>
 </rule>


 <!-- Dispara si hay >=100 eventos FIM en 60 s para el MISMO agente/ubicación -->
 <rule id="100005" level="15" frequency="100" timeframe="60" ignore="60">
   <if_matched_group>fim_burst_base,</if_matched_group>
   <same_agent />                          <!-- agrupa por agente/fuente -->
   <description>ALERTA! - Posible ransomware: 100+ cambios FIM en 60s</description>
   <group>ransomware_suspected,fim_burst,</group>
 </rule>

</group>
```

Figura 1. Reglas personalizadas de Wazuh.

Una vez que una de estas reglas se dispara y genera una alerta cuyo rule.level supera el umbral definido, entra en funcionamiento el mecanismo de Active Response de Wazuh. Este componente permite ejecutar acciones automáticas en respuesta a eventos de seguridad, tales como bloqueos en *firewall*, *scripts* de contención o integraciones con orquestadores externos. En el contexto de este proyecto, se configuró un Active Response que, ante la activación de una alerta de alto nivel asociada a comportamiento tipo *ransomware*, invoca el *script* denominado para el laboratorio como tfg_orc_new.py en el host, pasando

como parámetros la información relevante del evento (por ejemplo, nivel de regla, nombre de la máquina virtual afectada y posibles indicadores de compromiso).

```
<!-- Comando para notificar al orquestador TFG en el host Windows -->
<command>
  <name>tfg-orchestrator-webhook</name>
  <executable>tfg_orchestrator_webhook.sh</executable>
  <timeout_allowed>yes</timeout_allowed>
</command>


 <!-- Active Response que dispara el orquestador -->
 <!-- ante la detección de actividad tipo ransomware -->

<active-response>
  <command>tfg-orchestrator-webhook</command>
  <location>server</location>
  <level>15</level>
  <rules_id>100005</rules_id>
  <timeout>0</timeout>
</active-response>
```

Figura 2. Fragmento de configuración de Active Response en Wazuh.

En conjunto, la combinación de reglas de FIM orientadas a comportamiento de *ransomware* y el uso de Active Response permite que el modelo no dependa de una simple detección por firma, sino de un criterio de comportamiento. Únicamente cuando la actividad observada sobre el sistema de archivos alcanza un umbral que, según la literatura técnica, es compatible con un escenario de cifrado masivo, se considera que el incidente amerita la activación del flujo completo de contención, adquisición de evidencia digital y recuperación automática. [4]

Para que el proceso de recuperación automática se materialice, cuando el mecanismo de Active Response se dispara ante una alerta, el Wazuh Manager ejecuta el bloque <*command*> configurado localmente, el cual invoca a *curl* (u otro cliente HTTP) para emitir una petición HTTP POST hacia el *endpoint* expuesto por el orquestador. En dicha petición se envía un cuerpo JSON (Notación de Objeto de JavaScript, por sus siglas en inglés) que incluye información relevante de la alerta, como el rule.level, el identificador de la máquina virtual (vm_name) y otros metadatos necesarios para la toma de decisiones. La invocación del orquestador se realiza, por tanto, de forma indirecta mediante una petición HTTP POST al *endpoint* /wazuh/webhook, expuesto por el servicio Flask definido en el script tfg_orc_new.py, previamente inicializado en el *host* Windows. Esta petición es generada desde el *script* de Active Response mediante la instrucción curl, que actúa como puente entre Wazuh Manager y el módulo de orquestación de recuperación. Una vez recibido el JSON en /wazuh/webhook, el orquestador analiza el nivel de criticidad asociado a la alerta y decide si corresponde iniciar el flujo de aislamiento de la máquina afectada, preservación de evidencia forense y restauración automática de la máquina virtual mediante las operaciones de línea de comandos proporcionadas por VBoxManage.

```
# ---------------------------
# Webhook Flask para Wazuh
# ---------------------------
@app.route( rule: "/wazuh/webhook", methods=["POST"])
def wazuh_webhook():
    """
    Espera JSON: rule.level (int), opcional vm_name, y opcional ioc {ips:[], domains:[]}
    """
    data = request.get_json(force=True, silent=True) or {}
    rule_level = int(data.get("rule", {}).get("level", 0))
    vm_name = data.get("vm_name") or CONFIG["VM_NAME"]
    ioc = data.get("ioc", {})
    ioc_ips = ioc.get("ips", []) or []
    ioc_domains = ioc.get("domains", []) or []
```

Figura 3. Webhook receptor de alertas de Wazuh.

La máquina víctima del laboratorio se mantuvo intencionalmente sin servicios de negocio instalados, limitándose a una configuración mínima del sistema operativo y al agente de Wazuh. Esta decisión

metodológica responde a que el objetivo principal del experimento no es evaluar la continuidad de aplicaciones específicas, sino aislar y medir el comportamiento del flujo de orquestación y restauración automática del modelo. Al eliminar la variabilidad asociada a servicios adicionales (bases de datos, aplicaciones web, etc.), se reduce el ruido experimental y se obtiene una medición más limpia de los tiempos y de la estabilidad del proceso de recuperación (volcado de memoria, clonación de disco, restauración de *snapshot* y arranque de la máquina virtual). Como consecuencia, la validez externa de los resultados queda acotada a este entorno controlado, por lo que se reconoce como línea de trabajo futuro la reproducción del modelo sobre máquinas con cargas y servicios representativos de un entorno productivo real.

III. RESULTADOS.

El laboratorio se ejecutó en un entorno de virtualización basado en VirtualBox, con un servidor Wazuh sobre Ubuntu y una máquina víctima Debian 12 sobre la cual se simuló un incidente de seguridad tipo *ransomware* el cual fue detectado por Wazuh mediante sus reglas como cambios masivos en archivos, esto generó la alerta correspondiente y al cumplirse el umbral configurado, desencadenó la restauración automática de un *snapshot* seguro mediante un *script* en Python y el uso de VBoxManage. El flujo completo incluyó, además de la recuperación del servicio, la generación de evidencia técnica mediante: volcado de memoria, cálculo de *hash* SHA-256 de la imagen de memoria, recolección de *logs* del sistema invitado, clonación del disco virtual comprometido y cálculo de *hash* del clon.

Como resultado de cada ejecución del flujo automatizado, el modelo genera un conjunto de artefactos técnicos que documentan tanto la recuperación como la preservación de evidencia. En el directorio de trabajo del laboratorio se almacenan el volcado de memoria de la máquina víctima (.elf) y su *hash* SHA-256 correspondiente (.elf.sha256), un clon del disco virtual comprometido (.vdi) junto con sus archivos de verificación (.vdi.sha256), y un archivo comprimido con los registros del sistema invitado (.tgz). A esto se suma el reporte principal en formato Markdown (.md), donde se consolida la línea de tiempo del incidente, las métricas de duración por fase y un resumen de los artefactos generados. Este conjunto de archivos, que puede visualizarse fácilmente en un explorador de archivos o en capturas de pantalla, permite evidenciar de forma tangible cómo se materializa la adquisición de memoria, la clonación del disco, el cálculo de *hashes* y la documentación estructurada de cada incidente, facilitando tanto el análisis forense posterior como la trazabilidad de las pruebas realizadas.

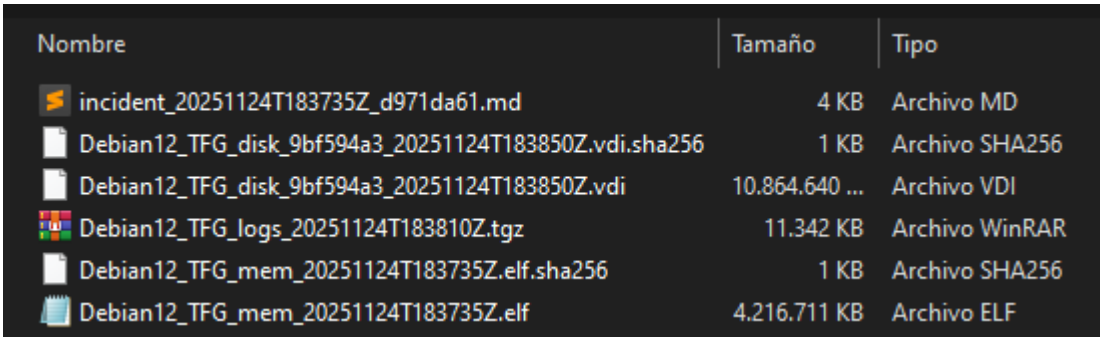

Figura 4. Conjunto de artefactos generados como resultado de la ejecución del modelo.

Con el fin de caracterizar el desempeño temporal del modelo, se realizaron veinte ejecuciones consecutivas del flujo de recuperación ante el mismo tipo de incidente. Para cada ejecución se registraron diez métricas temporales expresadas en el formato minutos:segundos,milisegundos:

Métrica	Descripción
time_to_restore	Tiempo total desde el disparo de la respuesta activa hasta que la máquina restaurada se encuentra disponible.
workflow_total	Duración completa del flujo automatizado, incluyendo recolección de evidencias.
memory_dump	Tiempo de generación del volcado de memoria.
memory_sha256	Tiempo de cálculo del *hash* SHA-256 del volcado de memoria.
guest_logs_collection	Tiempo de empaquetado y extracción de los registros del sistema invitado.
shutdown_and_poweroff	Tiempo de apagado y liberación de la máquina comprometida.
snapshot_restore	Tiempo de restauración técnica del *snapshot* desde VirtualBox.

vm_boot_and_guestcontrol_ready	Tiempo de arranque de la máquina restaurada hasta que el *guest* control está disponible.
disk_clone_total	Tiempo empleado en clonar el disco virtual de la máquina comprometida.
disk_clone_hash_total	Tiempo requerido para calcular el *hash* del clon de disco.

Tabla 1. Descripción de las métricas evaluadas.
Fuente: Elaboración propia.

A partir de las veinte ejecuciones se obtuvieron los promedios globales que se muestran a continuación, los cuales sintetizan el comportamiento del modelo de recuperación autónoma:

Métricas	Promedios de tiempo
time_to_restore	3:58,107
workflow_total	4:17,292
memory_dump	0:34,066
memory_sha256	0:04,366
guest_logs_collection	0:09,040
shutdown_and_poweroff	0:30,052
snapshot_restore	0:01,444
vm_boot_and_guestcontrol_ready	0:19,093
disk_clone_total	1:45,026
disk_clone_hash_total	0:52,631

Tabla 2. Promedios de las métricas evaluadas.
Fuente: Elaboración propia.

Estos resultados muestran que el modelo logra, de manera consistente, restaurar la máquina virtual afectada y dejarla nuevamente operativa en un tiempo cercano a cuatro minutos, mientras que el flujo completo que incluye la obtención de evidencia forense automatizada se sitúa ligeramente por encima de los cuatro minutos con diecisiete segundos.

En todas las ejecuciones del laboratorio el flujo se completó sin errores, es decir, se obtuvo una tasa de éxito del 100 % en la restauración del *snapshot* y en la ejecución de las tareas automatizadas de recolección de evidencias. No se registraron fallas de comunicación entre Wazuh, el *script* de orquestación en Python y VirtualBox, ni errores en la generación de los archivos de salida definidos en el modelo (volcado de memoria, *hash*, empaquetado de *logs*, clon de disco y reporte en formato Markdown).

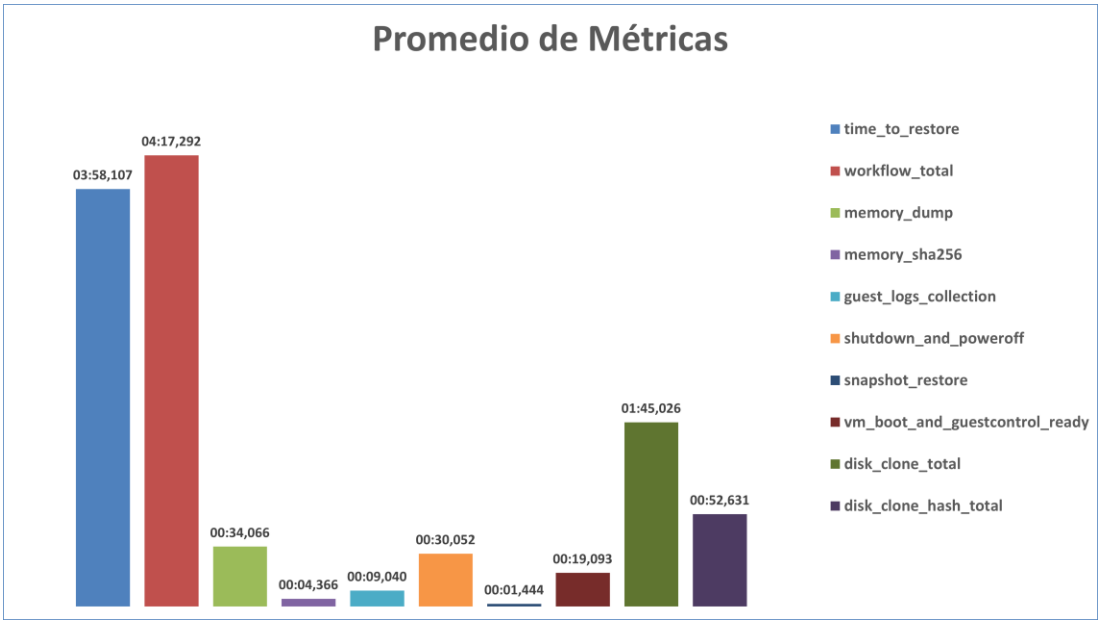

Figura 5. Tiempo promedio de las diferentes métricas evaluadas.
Fuente: Elaboración propia.

Distribución del tiempo por fases

Además del análisis de los promedios absolutos, se calculó el porcentaje del tiempo total del flujo que representa cada una de las fases internas registradas. En la figura 1 se muestra que la mayor proporción del tiempo se concentra en las operaciones asociadas al disco: la métrica disk_clone_total aporta aproximadamente un 41,07 % del tiempo total del flujo, mientras que disk_clone_hash_total representa alrededor de un 20,58 %. En conjunto, ambas fases consumen algo más de un 60% del tiempo de ejecución, lo que confirma que la clonación del disco virtual comprometido y el cálculo de su *hash* constituyen el principal componente temporal del modelo cuando se incorpora la obtención de evidencia orientada a análisis forense.

El resto de las fases presenta un peso relativo sensiblemente menor. La métrica memory_dump contribuye con cerca de un 13,32 % del tiempo total, seguida de shutdown_and_poweroff con aproximadamente 11,75 %, lo que indica que tanto la captura de memoria como el apagado controlado de la máquina virtual tienen un impacto relevante, aunque claramente inferior al de las tareas sobre disco. Por su parte, vm_boot_and_guestcontrol_ready representa cerca de un 7,47 %, mientras que guest_logs_collection y memory_sha256 aportan alrededor de un 3,54 % y un 1,71 %, respectivamente. Finalmente, la fase snapshot_restore apenas supone un 0,56 % del tiempo total, lo que confirma que la operación técnica de restauración del *snapshot* en VirtualBox es muy rápida en comparación con el resto de las actividades del flujo.

En conjunto, esta distribución porcentual evidencia que el modelo dedica la mayor parte de su tiempo a garantizar la preservación íntegra del estado del sistema comprometido, principalmente mediante la clonación y el aseguramiento criptográfico del disco, mientras que las fases de memoria, apagado, arranque y restauración del *snapshot*, introducen una carga temporal significativamente menor en el flujo global de recuperación.

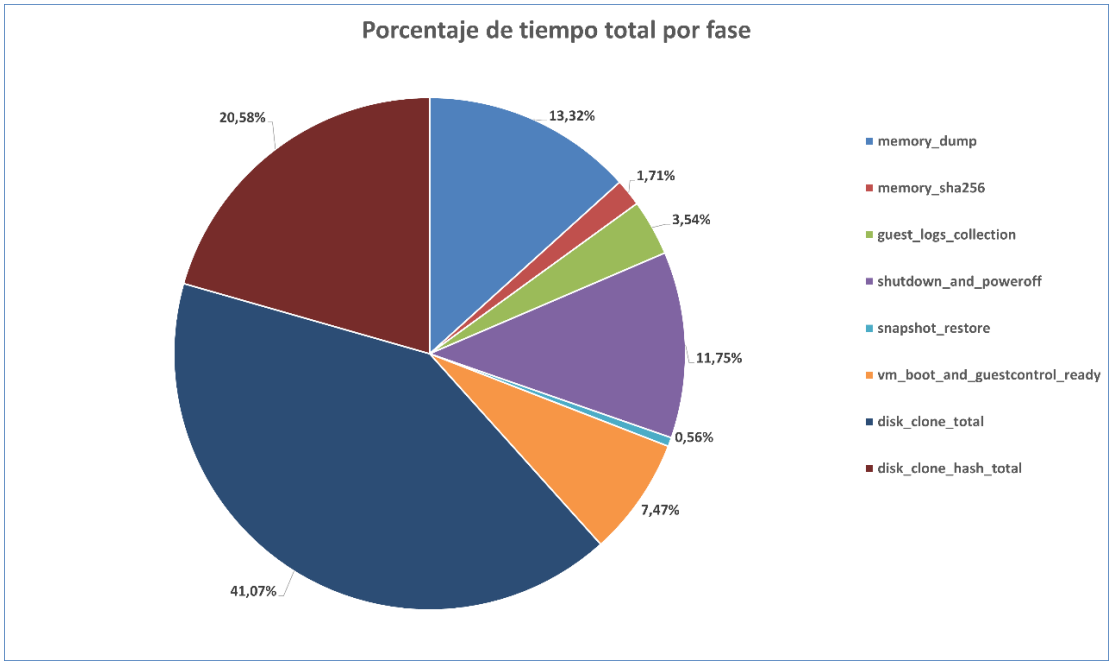

Figura 6. Porcentaje del tiempo total del flujo que corresponde a cada fase.
Fuente: Elaboración propia.

Estabilidad del MTTR y variación entre ejecuciones

El análisis secuencial de las veinte ejecuciones muestra que tanto time_to_restore como workflow_total presentan una tendencia decreciente conforme avanza la experimentación. En la primera ejecución, el tiempo de restauración se sitúa alrededor de 4:20,842, mientras que en la ejecución número veinte

desciende hasta aproximadamente 3:24,727. Esto representa una reducción cercana a 56 segundos en el MTTR a lo largo de las pruebas.

Un comportamiento similar se observa en la métrica workflow_total, que pasa de valores cercanos a 4:41,231 en las primeras ejecuciones a registros del orden de 3:45,621 en las últimas. Aunque ambas métricas presentan pequeñas fluctuaciones intermedias, la tendencia general es descendente y converge hacia valores más estables al final de la serie.

Esta disminución progresiva puede asociarse a varios factores propios del entorno de virtualización y del sistema operativo anfitrión. En primer lugar, el "calentamiento" del entorno provoca que bibliotecas, procesos y bloques de disco accedidos con frecuencia queden en caché, reduciendo los tiempos de lectura y escritura en ejecuciones posteriores del mismo flujo. En segundo lugar, la estabilización de servicios del sistema invitado y del servidor Wazuh, que ya se encuentran en estado operativo y con procesos en memoria, disminuye la latencia en la interacción entre componentes. Finalmente, la propia infraestructura física tiende a mantener recursos asignados de forma más eficiente después de las primeras ejecuciones, lo que contribuye a acortar los tiempos de arranque y apagado de las máquinas virtuales.

A pesar de esta variación decreciente, la dispersión de los datos es moderada y no se observan saltos bruscos ni fallos de ejecución. Las curvas correspondientes a time_to_restore y workflow_total mantienen una forma similar a lo largo de las veinte pruebas, lo que indica que el flujo de recuperación es estable y reproducible bajo las condiciones de laboratorio definidas en el trabajo final de graduación.

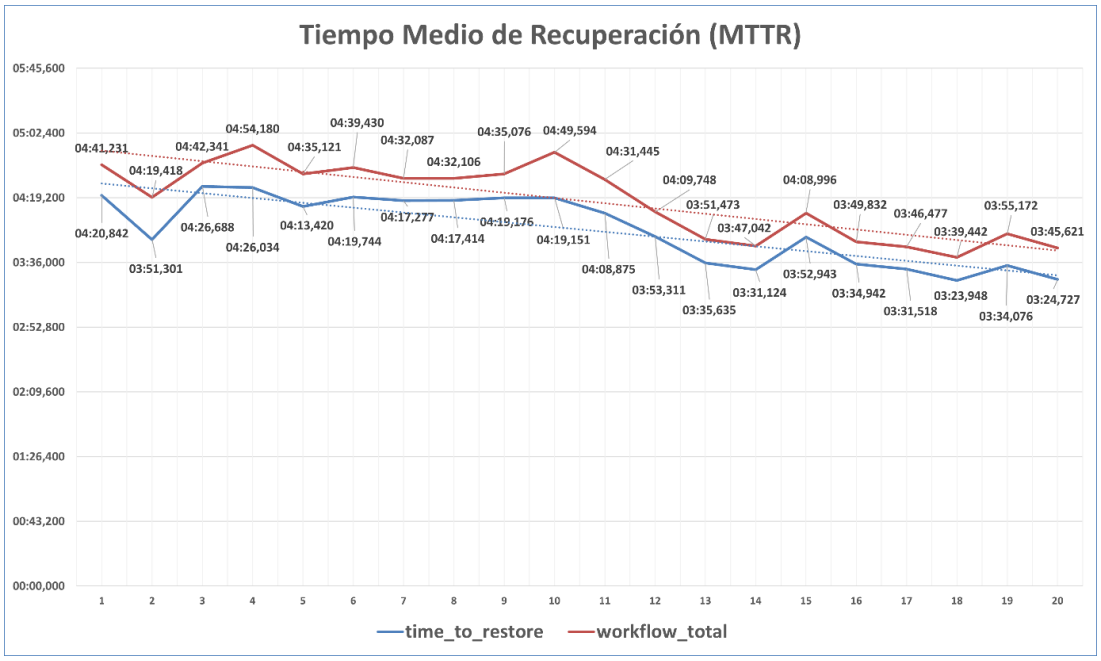

Figura 7. Estabilidad del MTTR y la variación marginal entre ejecuciones (time_to_restore, workflow_total).
Fuente: Elaboración propia.

Comportamiento de las fases de evidencia digital

Las fases asociadas a la generación de evidencia digital muestran un comportamiento estable entre ejecuciones. El memory_dump se mantiene cercano a los 34 segundos, sin variaciones significativas, lo que sugiere que el proceso de volcado de memoria es predecible para el tamaño y configuración de la máquina víctima utilizada. El memory_sha256 oscila alrededor de los 4 segundos, lo cual es coherente con el tamaño del archivo de memoria generado y con la potencia de cómputo del *host*.

La guest_logs_collection presenta un promedio cercano a los 9 segundos; la variación entre ejecuciones está asociada principalmente al volumen de registros generados por el sistema operativo invitado y por las propias herramientas de seguridad durante el flujo de recuperación. No obstante, el comportamiento global se mantiene dentro de márgenes acotados y sin valores atípicos extremos.

El shutdown_and_poweroff se mantiene en torno a los 30 segundos, lo que incluye tanto el apagado controlado de la máquina comprometida como las acciones previas de aislamiento definidas en el *script*. La métrica snapshot_restore, por su parte, se mantiene próxima a 1 segundo y medio, lo que confirma que la operación de revertir un *snapshot* en VirtualBox es altamente eficiente en este escenario de laboratorio.

Las métricas disk_clone_total y disk_clone_hash_total son las que introducen una mayor carga al flujo cuando se considera la obtención de evidencia para análisis posterior. Sin embargo, debido a la necesidad de disponer de un clon del disco exactamente en el estado comprometido previo a la restauración, de modo que cualquier análisis posterior se base en una imagen íntegra y representativa del incidente, es que el modelo asume explícitamente un compromiso entre rapidez y profundidad en la preservación de evidencia. Es decir, el MTTR aumenta por la inclusión de la clonación y el cálculo de *hash* del disco, pero a cambio se obtiene una copia completa y verificable del sistema afectado, adecuada para análisis detallados y para sustentar procesos de investigación o auditoría posteriores.

En conjunto, los resultados evidencian que el modelo de recuperación autónoma diseñado en este trabajo logra un equilibrio adecuado entre velocidad de restauración y generación de evidencias técnicas, manteniendo un MTTR inferior a cuatro minutos y un flujo completo cercano a los cuatro minutos con diecisiete segundos.

IV. DISCUSIÓN.

Los resultados obtenidos en el laboratorio corroboran la viabilidad técnica del modelo de recuperación autónoma propuesto en el trabajo final de graduación y su pertinencia para el contexto de las PYMEs. La arquitectura basada en Wazuh, *scripting* en Python y VirtualBox demuestra que es posible automatizar la respuesta ante un incidente simulado de compromiso, aislar el activo afectado y restaurar un *snapshot* seguro de manera consistente, sin intervención humana directa durante la ejecución del flujo.

Un primer aspecto relevante es que, aun incluyendo tareas de recolección de evidencias como volcado de memoria, extracción de registros y clonación de disco, el modelo mantiene un MTTR promedio de 3:58,107. Este valor se obtiene en un escenario donde la restauración se integra a la orquestación automática y no se limita a una simple operación de revertir *snapshot*, sino que incorpora pasos adicionales que aumentan la trazabilidad del incidente. Desde la perspectiva de las PYMEs, donde el personal técnico es limitado y los procesos suelen ser manuales, contar con un flujo automatizado que garantice tiempos de recuperación predecibles resulta especialmente valioso.

En segundo lugar, la comparación entre time_to_restore y workflow_total muestra que ambas métricas son muy próximas entre sí, lo que indica que prácticamente todo el tiempo de ejecución relevante para la recuperación está comprendido en el camino crítico que lleva desde la detección del incidente hasta la restauración efectiva de la máquina virtual. Dentro de ese intervalo se incluyen tanto las operaciones de recolección de evidencias (volcado de memoria, extracción de registros, clonación de disco y cálculo de *hashes*) como las acciones de apagado, restauración del *snapshot* y puesta en marcha de la instancia segura. En consecuencia, las tareas de evidencia no se ejecutan de forma marginal o diferida, sino que constituyen una parte sustantiva del tiempo de indisponibilidad del servicio y reflejan el compromiso explícito del modelo entre mantener una ventana de recuperación acotada y garantizar, al mismo tiempo, la preservación exhaustiva del estado comprometido del sistema.

La tendencia descendente del MTTR a lo largo de las veinte ejecuciones también aporta información relevante sobre la estabilidad del modelo. El hecho de que el tiempo de recuperación se reduzca progresivamente y se estabilice en valores inferiores a los de las primeras pruebas refleja un proceso de "ajuste" del entorno de virtualización y de los servicios implicados. Desde la perspectiva de operación continua de una PYME, esto sugiere que, una vez desplegado el modelo y superada la fase inicial de pruebas y calibración, el comportamiento temporal tendería a ser incluso más eficiente que el observado en las primeras ejecuciones de laboratorio.

Otra observación importante se relaciona con la integración de herramientas de código abierto. El modelo aprovecha las capacidades de Wazuh para la detección de eventos y respuesta activa, y utiliza Python para articular las decisiones de orquestación, delegando en VirtualBox la recuperación por *snapshot*. Esta combinación confirma que es posible construir un "mini-SOAR" de bajo costo, alineado con las limitaciones económicas y técnicas de las PYMEs, sin depender de licencias propietarias.

Tomando en cuenta estos resultados, cobra especial relevancia la comparación entre la restauración automatizada y la restauración manual del entorno comprometido. A partir de las mediciones realizadas en el laboratorio, se observó que el flujo orquestado completa la secuencia de aislamiento, obtención de evidencia, restauración del *snapshot* limpio y arranque de la máquina virtual en un intervalo cercano a los 4–5 minutos, sin intervención humana directa una vez generada la alerta de Wazuh. En contraste, la ejecución manual de los mismos pasos por parte de un administrador, incluyendo la identificación del incidente, el acceso a la consola de virtualización, la selección del *snapshot* adecuado, la restauración, el arranque de la máquina y las verificaciones básicas posteriores, se situó típicamente en un rango de 20 a 35 minutos, dependiendo de la experiencia de la persona y de su carga de trabajo en ese momento.

Esta diferencia no solo es temporal, sino también operacional. En un escenario manual, cada paso depende de la disponibilidad del administrador, de que siga correctamente los procedimientos establecidos y de que no existan errores al seleccionar *snapshots* o al manipular evidencias. En cambio, el flujo automatizado reduce la variabilidad asociada al factor humano, estandariza la secuencia de acciones y garantiza que tareas críticas, pero poco "visibles", como por ejemplo el volcado de memoria, el clonado forense de disco o la recolección sistemática de *logs*, se ejecuten siempre de la misma forma, incluso durante incidentes que ocurran fuera del horario laboral o cuando el personal técnico se encuentra atendiendo otras actividades. Desde la perspectiva de una PYME, esta automatización se traduce en una reducción del tiempo de indisponibilidad potencial, una menor dependencia de especialistas altamente calificados y una disminución del riesgo de omisiones o errores de procedimientos en momentos de alta presión.

Adicionalmente, aunque el laboratorio se centró en un escenario de ataque tipo ransomware, la arquitectura del modelo no está acoplada de manera rígida a este único vector de amenaza. Al basarse en reglas XML de Wazuh, mecanismos de Active Response y un orquestador externo desarrollado en Python, el mismo patrón puede adaptarse para otros tipos de incidentes mediante la incorporación de módulos adicionales y nuevas reglas de correlación. En la práctica, bastaría con definir conjuntos específicos de reglas en el ruleset local (por ejemplo, para detectar escaneos de red anómalos, intentos de elevación de privilegios, actividad sospechosa de cuentas o modificaciones no autorizadas en servicios críticos) y asociarlos a comandos de Active Response que invoquen al orquestador con parámetros distintos. El script de orquestación, por su parte, podría extenderse con rutinas especializadas (como por ejemplo deshabilitar cuentas, aislar sólo determinadas interfaces de red, generar volcados selectivos de memoria o preservar únicamente ciertos volúmenes de disco), manteniendo la misma lógica de flujo pero ajustando las acciones de contención y recuperación al tipo de incidente detectado. De este modo, el modelo se perfila como una base adaptable para la automatización de respuesta y recuperación ante una familia más amplia de eventos de seguridad, y no únicamente frente a escenarios de ransomware.

No obstante, los resultados deben interpretarse considerando las limitaciones del entorno experimental. En primer lugar, las pruebas se realizaron sobre un conjunto específico de máquinas virtuales y un tipo de incidente controlado, por lo que no se cubre toda la diversidad de escenarios que podrían presentarse en un entorno productivo real. En segundo lugar, aunque el modelo contempla diferentes métricas (detección, respuesta y recuperación), el énfasis del laboratorio se centra en la fase de restauración y en el flujo de obtención de evidencias, por lo que no se profundiza en métricas de falsa alarma, fatiga de alertas o escalamiento de incidentes complejos.

Adicionalmente, el diseño original del modelo contempla una tarea de escaneo post-restauración mediante una herramienta antivirus en la máquina recuperada. Sin embargo, esta fase se encuentra aún incompleta en el laboratorio actual. Esta fase está orientada a reforzar la confianza en el estado del sistema restaurado y a detectar posibles residuos de *malware* en el entorno post-recuperación. Su incorporación plena en futuras iteraciones permitirá fortalecer la validación de la integridad del servicio antes de devolverlo definitivamente a producción.

A pesar de estas limitaciones, el conjunto de resultados obtenidos ofrece evidencia suficiente para sostener que el modelo propuesto es técnicamente factible, replicable en otros laboratorios y potencialmente adaptable a entornos productivos de PYMEs con infraestructura de virtualización básica. La integración sistemática de detección, orquestación y recuperación automatizada aporta un enfoque coherente con las mejores prácticas de respuesta a incidentes descritas en la literatura utilizada para el desarrollo de este trabajo y se alinea con la necesidad de reducir el tiempo de inactividad y preservar la continuidad del negocio.

V. CONCLUSIONES. El análisis de las veinte ejecuciones del laboratorio confirma que el modelo de recuperación autónoma para entornos virtuales ante incidentes de seguridad cumple con su propósito principal: reducir el tiempo de recuperación y automatizar el proceso de restauración de sistemas comprometidos utilizando herramientas de código abierto.

En primer lugar, se verificó que el modelo logra restaurar la máquina víctima y dejarla disponible nuevamente con un MTTR promedio cercano a los cuatro minutos, manteniendo un flujo completo (incluida la recolección de evidencias) de aproximadamente cuatro minutos y diecisiete segundos. Estos resultados se obtuvieron de manera consistente en veinte ejecuciones consecutivas, sin registrar fallos en la ejecución del flujo, lo que evidencia la estabilidad operativa del modelo en el entorno de laboratorio.

En segundo lugar, se constató que las métricas time_to_restore y workflow_total presentan valores muy próximos entre sí, lo que indica que prácticamente todo el trabajo relevante para la recuperación se ejecuta dentro del camino crítico del modelo. Dentro de ese intervalo se incluyen tanto las operaciones de generación de evidencia digital (volcado de memoria, extracción de registros, clonación de disco y cálculo de *hashes*) como las acciones de apagado, restauración del *snapshot* y puesta en marcha de la máquina virtual restaurada. De esta forma, las tareas de evidencia no se encuentran al margen del proceso, sino que constituyen una parte sustantiva del tiempo de indisponibilidad del servicio. La distribución porcentual del tiempo entre las distintas fases confirma, además, que la clonación y el *hash* del disco virtual representan el componente temporal más significativo cuando se incorpora una capa de trazabilidad orientada al análisis forense, lo que refleja el compromiso explícito del modelo entre rapidez de recuperación y profundidad en la preservación del estado comprometido del sistema.

En tercer lugar, la observación de la tendencia decreciente del MTTR y del tiempo total del flujo a lo largo de las ejecuciones muestra que el modelo se beneficia del efecto de estabilización del entorno de virtualización. Una vez que los servicios del *host*, las máquinas virtuales y las herramientas de seguridad se encuentran en régimen, los tiempos de recuperación tienden a ser menores y más homogéneos que en las primeras ejecuciones, lo cual es una señal positiva para una futura adopción en contextos operativos.

Finalmente, se confirma que la arquitectura basada en Wazuh, Python y VirtualBox constituye una alternativa viable y de bajo costo para PYMEs que requieren mejorar su ciber resiliencia sin invertir en plataformas comerciales de alto costo. El modelo integra detección, orquestación y recuperación de forma coherente con las recomendaciones metodológicas revisadas en el Trabajo Final de Graduación y establece una base sólida para fases posteriores de investigación, en las que se podrán incorporar nuevos escenarios de ataque, métricas adicionales (como tasas de falsos positivos) y la automatización completa del escaneo post-restauración.

Con el objetivo de facilitar la replicabilidad del modelo propuesto y promover la investigación académica en el área de la ciber resiliencia en entornos virtualizados, el código correspondiente al script de orquestación utilizado en este trabajo ha sido publicado en un repositorio abierto. Este recurso permite a otros investigadores y profesionales analizar, reproducir y extender el enfoque presentado en esta investigación. El repositorio puede consultarse en: https://zenodo.org/records/18820905

VI. REFERENCIAS

[1] Verizon, «2024 Data Breach Investigations Report (DBIR),» Verizon, 2024. [En línea]. Available: https://www.verizon.com/business/resources/reports/2024-dbir-data-breach-investigations-report.pdf. [Último acceso: 15 octubre 2025].

[2] ENISA, «ENISA THREAT LANDSCAPE 2024,» ENISA, setiembre 2024. [En línea]. Available: https://www.enisa.europa.eu/sites/default/files/2024-11/ENISA%20Threat%20Landscape%202024_0.pdf.

[3] NIST, «Incident Response Recommendations and Considerations for Cybersecurity Risk Management: A CSF 2.0 Community Profile,» NIST, abril 2025. [En línea]. Available: https://csrc.nist.gov/pubs/sp/800/61/r3/final.

[4] J. Linares, «Preventing and detecting ransomware with Wazuh,» 12 diciembre 2019. [En línea]. Available: https://wazuh.com/blog/preventing-and-detecting-ransomware-with-wazuh/. [Último acceso: 29 noviembre 2025].

[5] Haircutfish, «Medium,» 7 junio 2024. [En línea]. Available: https://medium.com/%40haircutfish/tryhackme-room-custom-alert-rules-in-wazuh-5bce71e3bd01.

[6] R. Gibadullin y V. Nikonorov, «Development of the System for Automated Incident Management Based on Open-Source Software,» de *International Russian Automation Conference (RusAutoCon)*, Sochi, Russian, 2021.

[7] H. J. Hadi, «Cost-Effective Resilience: A Comprehensive Survey and Tutorial on Assessing Open-Source Cybersecurity Tools for Multi-Tiered Defense,» *IEEE Explore,* p. 24, 2024.

[8] A. Haydar y M. Ramparison, «Modeling Wazuh rules with Weighted Timed Automata,» *Procedia Computer Science,* pp. 75-82, 2024.

[9] F. Imán, «Infosec,» 12 marzo 2019. [En línea]. Available: https://www.infosecinstitute.com/resources/incident-response-resources/security-orchestration-automation-and-response-soar.

[10] M. R. Islam y R. Rafique, «Wazuh SIEM for Cyber Security and Threat Mitigation in Apparel Industries,» *International Journal of Engineering Materials and Manufacture (IJEMM),* pp. 136-144, 2024.

[11] MICITT, «Guia de Acción Ante Incidentes de Ransomware,» Octubre 2023. [En línea]. Available: https://www.micitt.go.cr/sites/default/files/2023-12/Playbook%20Ransomware.pdf.

[12] Oracle, «VBoxManage - Introduction,» [En línea]. Available: https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/vboxmanage-intro.html.

[13] Oracle, «VBoxManage snapshot,» 2024. [En línea]. Available: https://docs.oracle.com/en/virtualization/virtualbox/6.0/user/vboxmanage-snapshot.html.

[14] Wazuh, «Active Response,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/capabilities/active-response/index.html.

[15] Wazuh, «Rules Sintax,» [En línea]. Available: https://documentation.wazuh.com/current/user-manual/ruleset/ruleset-xml-syntax/rules.html.