



Centro Profesional
Universidad Europea Madrid
LAUREATE INTERNATIONAL UNIVERSITIES

UNIVERSIDAD EUROPEA



ESCUELA DE ARQUITECTURA, INGENIERÍA Y DISEÑO

CICLO FORMATIVO DE GRADO SUPERIOR ADMINISTRACIÓN
DE SISTEMAS INFORMÁTICOS EN RED

PROYECTO FIN DE CICLO

Laboratorio Forense

Pablo Casado Sánchez
Sergio Morales Durán
Óscar García Fernández

CURSO 2024-25

TÍTULO: TFG Laboratório Forense

AUTORES: Óscar García Fernández, Pablo Casado Sánchez y Sergio Morales Durán

TUTOR DEL PROYECTO: David Álvarez Boyero

FECHA DE LECTURA: 17 de Junio de 2025

CALIFICACIÓN:

Fdo:

Tutor/a del Proyecto

RESUMEN:

Este proyecto tiene como objetivo crear un laboratorio de análisis forense digital y ciberseguridad, orientado a la formación y la investigación. El entorno se basa en un sistema operativo principal que aloja directamente las herramientas clave para la monitorización, y se complementa con máquinas virtuales dedicadas al análisis forense y dinámico de malware.

En el sistema anfitrión (Ubuntu) se instala Wazuh, que se encarga de monitorizar el sistema, analizar logs y generar alertas de seguridad en tiempo real. Para el análisis forense, se utiliza una máquina virtual con Windows 10 en la que se ha instalado Autopsy. Esta herramienta permite realizar análisis forense sobre imágenes de disco que contienen archivos y programas maliciosos, facilitando la recuperación de datos y la detección de amenazas.

Cuando se detectan archivos sospechosos, estos se transfieren automáticamente a una máquina virtual con Cuckoo Sandbox, que los ejecuta en un entorno aislado para observar su comportamiento. Para ello, Cuckoo utiliza una segunda máquina virtual con Windows 7 como entorno de detonación, donde analiza acciones como conexiones externas, modificaciones del sistema o intentos de evasión.

Toda la actividad del laboratorio es registrada y visualizada mediante el visor de eventos de Wazuh, lo que permite mantener un control centralizado y detectar anomalías de forma eficiente.

Además, se ha automatizado el flujo entre Autopsy y Cuckoo, optimizando el proceso y reduciendo la intervención manual. En conjunto, el laboratorio ofrece un entorno seguro y controlado para el estudio práctico del análisis forense y el comportamiento del malware.

ABSTRACT:

This project aims to create a digital forensics and cybersecurity laboratory focused on education and research. The environment is based on a main operating system that directly hosts key monitoring tools and is complemented by virtual machines dedicated to forensic and dynamic malware analysis.

Wazuh is installed on the host system (Ubuntu), where it monitors the system, analyzes logs, and generates real-time security alerts. For forensic analysis, a virtual machine running Windows 10 is used, with Autopsy installed. This tool allows forensic analysis of disk images containing malicious files and programs, facilitating data recovery and threat detection.

When suspicious files are detected, they are automatically transferred to a virtual machine with Cuckoo Sandbox, which executes them in an isolated environment to observe their behavior. To do this, Cuckoo uses a second virtual machine running Windows 7 as the detonation environment, where it analyzes actions such as external connections, system modifications, or evasion attempts.

All laboratory activity is logged and visualized through Wazuh's event viewer, enabling centralized control and efficient anomaly detection.

Additionally, the workflow between Autopsy and Cuckoo has been automated, optimizing the process and reducing manual intervention. Overall, the laboratory provides a secure and controlled environment for the practical study of forensic analysis and malware behavior.

AGRADECIMIENTOS

Nos gustaría comenzar agradeciendo a nuestras familias, que nos han acompañado en todo momento con paciencia, cariño y apoyo incondicional. Gracias por estar siempre ahí, incluso en los momentos más intensos del proyecto.

También queremos dar las gracias a todos los profesores que nos han acompañado durante estos dos años. David, Raúl, Palmira, Juanjo, José Antonio, Alberto y Maite, gracias por enseñarnos con pasión y compartir vuestros conocimientos. También destacar la ayuda para resolver nuestras dudas y guiarnos para que este proyecto tome su forma final de la manera adecuada.

Finalmente, queremos dar las gracias a esos amigos que siempre estuvieron ahí, ya fuera para echarnos una mano, animarnos o simplemente hacernos más llevadero el camino. Sin ellos, todo esto habría sido mucho más difícil.



Esta obra se distribuye bajo una licencia Creative Commons.

Se permite la copia, distribución, uso y comunicación de la obra si se respetan las siguientes condiciones:

- Se debe reconocer explícitamente la autoría de la obra incluyendo esta nota y su enlace.
- La copia será literal y completa
- No se podrá hacer uso de los derechos permitidos con fines comerciales, salvo permiso expreso de los autores.

El texto precedente no es la licencia completa sino una nota orientativa de la licencia original completa(jurídicamente válida) que puede encontrarse en:

<http://creativecommons.org/licenses/by-nc-nd/3.0/deed.es>

1. INTRODUCCIÓN.....	1
1.1. OBJETIVOS	1
1.2. MOTIVACIÓN	2
1.3. ANTECEDENTES	2
2. DESARROLLO DE LA PRÁCTICA.....	4
2.1. MATERIAL	4
<i>Autopsy: Herramienta de Análisis Forense Digital</i>	4
Historia y evolución de Autopsy	5
Características principales	5
Comunidad y soporte.....	5
Funcionalidades clave en el análisis forense	6
Aplicaciones y casos de uso	6
Ventajas y limitaciones.....	6
<i>Cuckoo Sandbox: Herramienta de Análisis Automatizado de Malware</i>	6
Características Principales de Cuckoo Sandbox.....	7
Funcionamiento de Cuckoo Sandbox en el Análisis de Malware	8
Aplicaciones de Cuckoo Sandbox en el Análisis Forense de Malware.....	8
Ventajas y Limitaciones de Cuckoo Sandbox	9
<i>Wazuh: Plataforma de Seguridad y Monitorización Integral</i>	9
Características Principales de Wazuh SIEM	9
Funcionamiento de Wazuh SIEM	10
Aplicaciones de Wazuh SIEM.....	11
Ventajas y Limitaciones de Wazuh SIEM.....	11
<i>FTK Imager</i>	12
Características Principales de FTK Imager	12
Usos Principales de FTK Imager.....	13
Ventajas y Limitaciones de FTK Imager.....	13
2.2. PLANIFICACIÓN.....	13
Objetivo general	14
Pasos planificados	14
Recursos y materiales necesarios	14
2.3. DESCRIPCIÓN DEL TRABAJO REALIZADO.....	14
2.4. RESULTADOS Y VALIDACIÓN	31
Análisis estático con Autopsy	31
Análisis dinámico con Cuckoo Sandbox	32
Monitorización con Wazuh	33
3. CONCLUSIONES.....	35
3.1 APORTACIONES	35
3.2 TRABAJO FUTURO	36
4. BIBLIOGRAFÍA Y WEBGRAFÍA.....	38
5. ANEXO	I
5.1. SIEM EN UBUNTU 22.04.2:	I
Instalación.....	I
5.2. CONFIGURACIÓN CUCKOO.....	VII
Creamos el usuario cuckoo:.....	VII
Agregamos variables de entorno en Ubuntu, utilizaremos pyenv ya que Cuckoo Sandbox requiere de una herramienta específica de Python.	VIII
Instalamos pyenv con el usuario cuckoo con el siguiente comando, aunque vamos a necesitar instalar curl y git para poder ejecutarlo:	IX
Instalamos python2.7.18:.....	IX
Configuración win7 cuckoo	XI
Configuración de cuckoo:.....	XV
Accedemos de nuevo a Ubuntu y comenzamos a configurar cuckoo.	XVI
5.3. SCRIPT AUTOMATIZACIÓN CUCKOO	XVIII



1. INTRODUCCIÓN

En el presente Proyecto de Fin de Grado (PFG) se ha diseñado y desarrollado un laboratorio de análisis forense digital y ciberseguridad, con el objetivo de simular un entorno realista para la detección, análisis y monitorización de amenazas informáticas. Desde el inicio, el propósito del proyecto ha sido claro: construir una infraestructura funcional y segura que permita identificar archivos sospechosos, analizarlos en entornos controlados y asegurar la integridad del sistema utilizado.

A lo largo del proyecto se ha desplegado un entorno basado en un sistema operativo principal sobre el cual se instala Wazuh, encargado de la monitorización continua del sistema, la detección de eventos relevantes y la generación de alertas de seguridad.

Para el análisis forense estático, se ha configurado una máquina virtual en la que se ha instalado Autopsy. Esta herramienta permite examinar imágenes de disco que simulan incidentes de seguridad, facilitando la recuperación de datos y la identificación de posibles amenazas.

Los archivos considerados sospechosos tras el análisis con Autopsy se transfieren automáticamente a una máquina virtual que ejecuta Cuckoo Sandbox, una plataforma de análisis dinámico de malware. Esta, a su vez, utiliza una segunda máquina virtual con Windows 7 como entorno de detonación, donde se observan y registran los comportamientos del malware en un entorno aislado.

Este documento presenta el contexto del proyecto, sus motivaciones, los objetivos perseguidos y los fundamentos técnicos que lo sustentan, sirviendo como guía para entender tanto el diseño como la implementación del laboratorio propuesto.

1.1. Objetivos

Los objetivos principales de este Proyecto de Fin de Grado son los siguientes:

- Crear una imagen de sistema comprometido que simule un entorno realista, con herramientas, configuraciones y vectores de ataque típicos, con el fin de ser utilizada como objeto de análisis forense.
- Instalar y configurar Autopsy en una máquina virtual con Windows 10 para llevar a cabo análisis forenses sobre la imagen comprometida, facilitando la recuperación de datos y la identificación de archivos o comportamientos sospechosos.
- Automatizar el envío de archivos sospechosos detectados mediante Autopsy hacia una máquina virtual que ejecuta Cuckoo Sandbox, con el objetivo de realizar análisis dinámico en un entorno seguro y aislado.
- Analizar en profundidad el comportamiento del malware utilizando una segunda máquina virtual con Windows 7 como entorno de detonación de Cuckoo, permitiendo observar acciones como modificaciones del sistema, conexiones remotas o técnicas de evasión.
- Monitorizar y centralizar eventos y alertas mediante Wazuh, instalado en el sistema principal, actuando como sistema SIEM. Esta herramienta permite tener una visión global del estado del laboratorio, detectar anomalías y asegurar la integridad del entorno.



- Documentar detalladamente todo el proceso de diseño, implementación y uso del laboratorio, con el objetivo de que pueda servir como referencia o guía práctica para estudiantes, docentes o profesionales interesados en montar un entorno similar.

Cada uno de estos objetivos está orientado a garantizar un enfoque completo, combinando teoría y práctica, y fomentando el aprendizaje autónomo y la replicabilidad del entorno propuesto.

1.2. Motivación

La realización de este proyecto surge del interés creciente por el análisis forense digital y la ciberseguridad, áreas cada vez más relevantes tanto en el ámbito académico como en el profesional. Esta motivación se ha visto reforzada por la experiencia adquirida durante las prácticas en empresa, donde se trabajó con herramientas similares a las empleadas en este laboratorio. Se pudo comprobar de primera mano la importancia de contar con soluciones adecuadas para el análisis de incidentes, el tratamiento de archivos sospechosos y la monitorización de sistemas en tiempo real, todo ello con el objetivo de mantener la seguridad y el correcto funcionamiento de la infraestructura de la organización.

Además, antes incluso de conocer en profundidad estas herramientas, surgió el interés personal por el análisis forense a raíz de una experiencia cotidiana: la rotura de un dispositivo móvil que contenía información importante. En consecuencia, se planteó la posibilidad de recuperar los datos mediante técnicas de análisis forense, lo que despertó el interés por estas tecnologías y sus aplicaciones prácticas.

Este proyecto pretende, no solo reproducir un entorno técnico basado en experiencias del mundo profesional, sino también reflejar cómo las inquietudes personales pueden ser el origen de un proceso de aprendizaje profundo y estructurado en el ámbito de la ciberseguridad.

1.3. Antecedentes

El análisis forense digital y la monitorización de sistemas son disciplinas esenciales en el ámbito de la ciberseguridad, tanto en contextos académicos como profesionales. A medida que las amenazas informáticas evolucionan, también lo hacen las herramientas y metodologías empleadas para su detección, análisis y prevención. Este proyecto se sitúa en ese contexto, tomando como referencia diversas soluciones ampliamente utilizadas para construir un entorno de laboratorio funcional y realista.

En el campo del análisis forense, herramientas como Autopsy, de código abierto, han sido adoptadas por comunidades educativas y profesionales debido a su accesibilidad y capacidad para analizar sistemas de archivos, recuperar información eliminada y generar informes detallados. Sin embargo, existen soluciones comerciales como Magnet AXIOM que ofrecen funcionalidades avanzadas, como análisis de artefactos en la nube, acceso simplificado a evidencias en dispositivos móviles y motores de correlación más sofisticados. Estas características hacen que herramientas como AXIOM sean más adecuadas para investigaciones complejas, aunque requieren licencias de pago.



Para el análisis dinámico de malware, Cuckoo Sandbox representa una alternativa robusta y extensible dentro del software libre. Su arquitectura modular permite personalizar análisis y generar informes detallados de comportamiento. No obstante, soluciones comerciales como Recorded Future van más allá, integrando inteligencia de amenazas en tiempo real, puntuaciones de riesgo automatizadas y capacidades predictivas a partir de datos externos, lo que facilita la toma de decisiones estratégicas en entornos corporativos.

En cuanto a la monitorización, la pila ELK (Elasticsearch, Logstash y Kibana) es una opción muy popular gracias a su flexibilidad y potencia, especialmente en entornos de código abierto. No obstante, su versión comercial, Elastic Security, ofrece características mejoradas como detección basada en machine learning, respuesta automatizada a incidentes y escalabilidad optimizada para grandes infraestructuras. Del mismo modo, soluciones como Microsoft Sentinel combinan capacidades SIEM y SOAR en la nube, integrándose con otros servicios de Microsoft y permitiendo una gestión centralizada de eventos, detección de amenazas y automatización de respuestas.

La elección de herramientas de código abierto para este proyecto responde a criterios de accesibilidad, adaptabilidad y posibilidad de modificación, lo cual resulta fundamental en un entorno académico. No obstante, el conocimiento de las herramientas comerciales permite situar este laboratorio dentro del ecosistema profesional y entender las diferencias clave entre ambos enfoques.

2. DESARROLLO DE LA PRÁCTICA

La presente práctica ha consistido en el despliegue, configuración y análisis de diversas herramientas de seguridad y monitorización utilizadas en el ámbito profesional, con el objetivo de familiarizarse con su funcionamiento y evaluar su utilidad en contextos reales. A lo largo del trabajo, se ha seguido una metodología sistemática que incluye la instalación de entornos virtualizados, el uso de sistemas operativos específicos, y la configuración de herramientas orientadas a la recolección, análisis y validación de evidencias digitales y eventos de seguridad.

La elección de cada herramienta se ha basado en criterios de funcionalidad, relevancia en el sector y compatibilidad con los objetivos de la práctica. La validación se ha llevado a cabo mediante pruebas controladas, observando el comportamiento de las soluciones implementadas frente a distintos escenarios simulados.

A continuación, se detallan los materiales, entornos y herramientas utilizados.

2.1. Material

En los años 80, existían muy pocas herramientas específicas para la investigación forense digital. Por ello, los investigadores solían realizar análisis en vivo directamente sobre los dispositivos, accediendo al sistema operativo y utilizando herramientas de administración ya existentes para obtener posibles evidencias. Este enfoque conllevaba un alto riesgo de alterar los datos almacenados en el disco, ya fuera de forma accidental o intencionada, lo que podía dar lugar a acusaciones de manipulación de pruebas.

A partir de los años 90 comenzaron a desarrollarse herramientas diseñadas específicamente para evitar estas alteraciones y preservar la integridad de la evidencia digital. Este avance marcó el inicio de la forensia digital moderna, con soluciones orientadas a la adquisición, análisis y documentación de datos sin comprometer su validez legal. Desde entonces, el campo ha evolucionado considerablemente, integrando herramientas más complejas y especializadas que permiten afrontar amenazas actuales como el malware avanzado o los ciberataques dirigidos.

En el presente Proyecto de Fin de Grado se han empleado cuatro herramientas principales, cada una con un rol específico dentro del laboratorio, que se describen en detalle a continuación:

Autopsy: Herramienta de Análisis Forense Digital

Autopsy es una plataforma de software de código abierto diseñada para el análisis forense digital, ampliamente utilizada en investigaciones de incidentes de seguridad informática. Desarrollada inicialmente por Basis Technology, proporciona una interfaz gráfica intuitiva que facilita a los analistas la exploración de discos duros, sistemas de archivos y dispositivos de almacenamiento, permitiendo la recuperación y el análisis de datos relevantes para investigaciones forenses.



Historia y evolución de Autopsy

El desarrollo de Autopsy comenzó a principios de los años 2000 como un proyecto universitario con el objetivo de ofrecer una herramienta accesible y potente para el análisis forense digital. Desde entonces, ha evolucionado gracias al apoyo de la comunidad de código abierto y de Basis Technology, que ha ido integrando nuevas funcionalidades y adaptando la plataforma a las crecientes necesidades de la investigación forense. Hoy en día, Autopsy es una herramienta consolidada y reconocida internacionalmente, utilizada tanto por organismos gubernamentales como por profesionales del sector privado.

Características principales

- Interfaz gráfica de usuario (GUI): Su entorno visual simplifica el análisis forense, permitiendo a usuarios sin experiencia en línea de comandos explorar imágenes de disco y sistemas de archivos con facilidad.
- Compatibilidad multiplataforma: Autopsy funciona en Windows, Linux y MacOS, lo que lo hace versátil y adaptable a distintos entornos de trabajo.
- Soporte para múltiples sistemas de archivos: Es capaz de analizar sistemas como FAT, NTFS, ext3/ext4, entre otros, lo que permite recuperar archivos eliminados, reconstruir estructuras de directorios y extraer metadatos.
- Recuperación de archivos eliminados: Utiliza técnicas avanzadas para identificar y recuperar archivos borrados, incluso parcialmente sobrescritos.
- Análisis de actividad del usuario: Permite examinar accesos a archivos, modificaciones, historial de navegación y otras acciones del usuario.
- Investigación de comunicaciones digitales: Incluye módulos para el análisis de correos electrónicos, redes sociales y otros medios de comunicación, útiles en casos de acoso, fraude o delitos cibernéticos.
- Generación de informes forenses: Ofrece informes detallados sobre las evidencias encontradas, esenciales para documentar hallazgos en procesos judiciales.
- Extensibilidad mediante plugins: Autopsy permite integrar módulos adicionales que amplían sus capacidades, incluyendo análisis de imágenes, bases de datos o tráfico de red.

Comunidad y soporte

Al ser un proyecto de código abierto, Autopsy cuenta con una comunidad activa que contribuye al desarrollo, mantenimiento y mejora continua de la herramienta. Esto garantiza actualizaciones frecuentes, soporte a través de foros y documentación accesible para usuarios de todos los niveles. Además, su naturaleza abierta permite la personalización mediante plugins, facilitando la adaptación a necesidades específicas de cada investigación.



Funcionalidades clave en el análisis forense

- Adquisición y análisis de imágenes forenses: Soporta la creación y el análisis de imágenes físicas y lógicas, garantizando la integridad de los datos originales durante la investigación.
- Análisis de contenido: Permite visualizar y recuperar diversos tipos de archivos (documentos, imágenes, correos), así como detectar datos ocultos o eliminados.
- Estudio de artefactos del sistema: Extrae información sobre software utilizado, navegación web, y registros de acceso, revelando el uso del sistema por parte del usuario.
- Investigación de archivos de registro (logs): Analiza en profundidad los logs del sistema, permitiendo reconstruir la secuencia de eventos relacionados con un incidente.
- Correlación de datos: Integra y relaciona información obtenida desde distintos artefactos para construir líneas de tiempo precisas y coherentes.

Aplicaciones y casos de uso

Autopsy se utiliza tanto en el ámbito judicial como empresarial:

- Investigaciones de delitos cibernéticos: Fraude, robo de identidad, acoso en línea o acceso no autorizado.
- Recuperación de pruebas en procesos legales: Análisis de evidencias digitales en casos civiles, penales o corporativos.
- Análisis post-incidente en entornos corporativos: Detección de intrusiones, malware o comportamiento sospechoso en redes empresariales.

Ventajas y limitaciones

- Ventajas:
 - Software gratuito y de código abierto.
 - Interfaz amigable, apta para usuarios no expertos.
 - Ampliable mediante módulos personalizados.
- Limitaciones:
 - Requiere una curva de aprendizaje para dominar sus funcionalidades avanzadas.
 - El análisis de grandes volúmenes de datos puede ser intensivo en recursos y tiempo.

Cuckoo Sandbox: Herramienta de Análisis Automatizado de Malware

Cuckoo Sandbox es una herramienta de análisis automatizado de malware que se utiliza para detectar, analizar y comprender el comportamiento de software malicioso en un entorno controlado. Al ejecutar muestras de malware en máquinas virtuales aisladas, Cuckoo Sandbox permite observar cómo interactúa el malware con el sistema, qué archivos o procesos crea, qué cambios realiza en el sistema y cómo intenta comunicarse con otras redes. Su naturaleza de código abierto y su capacidad de integración con múltiples entornos la convierten en una de las soluciones más populares en la detección y análisis forense de malware.



Características Principales de Cuckoo Sandbox

- **Análisis automático de malware:** Cuckoo Sandbox permite realizar un análisis automatizado de muestras de malware sin la intervención manual de un analista. Este análisis incluye la ejecución de los programas maliciosos en un entorno virtualizado (generalmente una máquina virtual) donde el comportamiento del malware se monitorea de forma detallada y se recopilan datos clave, como los cambios en el registro, archivos creados, conexiones de red y demás interacciones con el sistema.
- **Aislamiento de entornos de análisis:** La característica más destacada de Cuckoo Sandbox es su capacidad para aislar el malware en entornos controlados, lo que evita que el código malicioso se propague fuera de la máquina virtual. Este aislamiento es fundamental para evitar daños en sistemas operativos reales y garantiza que las muestras de malware no interfieran con otros sistemas o redes.
- **Recopilación de información detallada:** Durante el análisis, Cuckoo Sandbox recolecta una gran cantidad de datos relacionados con el comportamiento del malware, que incluyen:
 - Cadenas de red: conexiones a servidores remotos, solicitudes HTTP, o intentos de exfiltración de datos.
 - Cambios en el sistema de archivos: archivos modificados o creados, así como registros de ejecución.
 - Análisis de registros de procesos: identificación de nuevos procesos iniciados por el malware, cambios en procesos existentes y sus interacciones.
 - Análisis de registros del sistema: inspección de registros del sistema operativo, como eventos del sistema y alertas.
- **Soporte multiplataforma:** Cuckoo Sandbox es compatible con múltiples sistemas operativos, lo que permite analizar muestras de malware para Windows, Linux y Android, entre otros. Esto lo convierte en una herramienta versátil que puede ser utilizada para analizar diferentes tipos de malware en diversas plataformas.
- **Interfaz de usuario web:** Cuckoo Sandbox ofrece una interfaz de usuario basada en web que facilita el control y visualización de los resultados del análisis. Los analistas pueden subir muestras de malware, gestionar las máquinas virtuales y revisar los informes generados desde cualquier lugar utilizando un navegador web. Esto permite una experiencia de uso cómoda y accesible.
- **Informes detallados:** Una vez que el malware ha sido ejecutado en el entorno aislado, Cuckoo Sandbox genera informes detallados que incluyen información sobre las acciones realizadas por el malware. Los informes son esenciales para comprender las tácticas, técnicas y procedimientos (TTPs) utilizados por el malware y son útiles tanto para la detección como para la prevención de futuros ataques.
- **Integración con otras herramientas:** Cuckoo Sandbox no solo es útil de manera independiente, sino que también se puede integrar con otras herramientas de seguridad y análisis, como Yara para la detección de patrones, Suricata para la captura de tráfico de red, o incluso sistemas de gestión de incidentes de seguridad. Esta capacidad de integración mejora la eficiencia y precisión en el análisis de malware.
- **Código abierto y extensibilidad:** Al ser una herramienta de código abierto, Cuckoo Sandbox permite a los usuarios modificarla y adaptarla a sus necesidades específicas. Los usuarios pueden crear nuevos módulos para analizar nuevos tipos de malware, modificar



los módulos existentes o agregar funcionalidades adicionales que se ajusten a los requisitos de su entorno de trabajo.

Funcionamiento de Cuckoo Sandbox en el Análisis de Malware

- Preparación del entorno de análisis: El primer paso es configurar las máquinas virtuales que serán utilizadas para ejecutar las muestras de malware. Cuckoo permite la creación de entornos virtualizados con diferentes configuraciones, lo que facilita la simulación de distintos sistemas operativos. Esto es fundamental para realizar un análisis completo y obtener una visión detallada del comportamiento del malware en diversos entornos.
- Ejecución del malware: Una vez que la máquina virtual está configurada, el malware se ejecuta de manera controlada en ella. Cuckoo Sandbox realiza un seguimiento constante de las acciones realizadas por el malware, como la creación de nuevos archivos, la modificación de registros del sistema y las conexiones de red.
- Monitoreo del comportamiento: Durante la ejecución, Cuckoo Sandbox registra todos los cambios que realiza el malware, incluyendo modificaciones en el sistema de archivos, conexiones a servidores externos, intentos de explotación de vulnerabilidades y alteraciones en la memoria. También se monitorizan los archivos de log y otros indicadores de actividad sospechosa.
- Generación de reportes: Una vez finalizado el análisis, Cuckoo Sandbox genera un reporte que incluye toda la información sobre el comportamiento del malware, los archivos afectados y las redes con las que el malware ha intentado comunicarse. Estos informes son detallados y pueden incluir capturas de pantalla, registros de red, información sobre las claves de registro modificadas, entre otros datos útiles.

Aplicaciones de Cuckoo Sandbox en el Análisis Forense de Malware

- Análisis de amenazas avanzadas: Cuckoo es especialmente útil para el análisis de amenazas avanzadas y ataques dirigidos (APT, por sus siglas en inglés). Los atacantes suelen emplear técnicas sofisticadas para evitar la detección, y Cuckoo permite estudiar cómo el malware interactúa con el sistema y las tácticas utilizadas por los atacantes.
- Investigación de incidentes de seguridad: En el contexto de una investigación post-incidente, Cuckoo Sandbox se utiliza para analizar archivos maliciosos encontrados en los sistemas comprometidos. Esto ayuda a los analistas a comprender el alcance del ataque, identificar las vulnerabilidades explotadas y elaborar un plan de respuesta adecuado.
- Desarrollo de firmas para detección de malware: A través de la ejecución y análisis de múltiples muestras de malware, Cuckoo Sandbox ayuda a los equipos de seguridad a desarrollar firmas o patrones para la detección automática de malware en futuras incidencias. Estas firmas pueden ser utilizadas para integrar sistemas de detección de intrusiones (IDS) o para mejorar las soluciones antivirus.
- Entrenamiento y educación: Además de su uso en investigaciones reales, Cuckoo Sandbox también se emplea en entornos educativos para enseñar sobre malware y su análisis. Al ser una herramienta de código abierto, los estudiantes y profesionales pueden practicar y experimentar con diferentes tipos de malware de manera segura.



Ventajas y Limitaciones de Cuckoo Sandbox

- Ventajas:
 - Automatización: permite análisis sin intervención manual continua, ahorrando tiempo
 - Código abierto: personalizable y adaptable a necesidades específicas
 - Soporte multiplataforma: análisis de malware en diferentes sistemas operativos.
- Limitaciones:
 - Dependencia de máquinas virtuales: puede no detectar malware diseñado para evitar entornos virtualizados
 - Requiere recursos: el análisis intensivo consume hardware y puede ralentizar el proceso.

Wazuh: Plataforma de Seguridad y Monitorización Integral

Wazuh es una plataforma de seguridad informática de código abierto que ofrece una solución integral para la monitorización, detección de amenazas, cumplimiento normativo, y análisis forense. Originalmente derivado de OSSEC, Wazuh ha evolucionado para convertirse en una de las soluciones más completas dentro del mundo de los SIEM (Security Information and Event Management) y XDR (Extended Detection and Response). Su arquitectura permite una integración eficaz con diversas plataformas y servicios, lo que lo convierte en una herramienta esencial en la gestión de la seguridad y en la protección de infraestructuras de TI.

Características Principales de Wazuh SIEM

- Monitorización de seguridad en tiempo real: Wazuh SIEM permite la monitorización continua de eventos de seguridad en tiempo real. Mediante la recolección y análisis de logs de diferentes dispositivos y sistemas, Wazuh puede detectar y responder de manera proactiva ante posibles amenazas. Los agentes de Wazuh se despliegan en los sistemas que se desea monitorear, y estos recogen eventos que son enviados al servidor Wazuh, donde se procesan y analizan.
- Detección de intrusiones y análisis de comportamiento: Wazuh utiliza técnicas avanzadas de detección de intrusiones (IDS), lo que permite identificar actividades maliciosas como escaneos de puertos, intentos de acceso no autorizado y malware conocido. A través de un sistema de reglas predefinidas, Wazuh puede correlacionar los eventos para identificar patrones sospechosos y generar alertas. Además, el análisis de comportamiento ayuda a detectar anomalías que podrían indicar una intrusión o una brecha de seguridad.
- Recopilación y correlación de logs: Una de las funcionalidades más destacadas de Wazuh es su capacidad para recoger logs de sistemas, aplicaciones y dispositivos de red. Estos logs son esenciales para la evaluación de la seguridad, el análisis de incidentes y la elaboración de auditorías. Wazuh utiliza esta información para generar alertas detalladas sobre actividades sospechosas y correlacionar eventos en tiempo real, lo que mejora la visibilidad de las amenazas.
- Cumplimiento normativo (Regulatory Compliance): Wazuh ayuda a las organizaciones a cumplir con normativas y estándares de seguridad como PCI DSS, GDPR, HIPAA, ISO 27001, entre otros. La plataforma incluye reglas y auditorías específicas para estos estándares, lo que facilita la evaluación y la generación de informes de cumplimiento.



Esto es especialmente útil para las empresas que deben auditar sus infraestructuras y demostrar que cumplen con los requisitos legales y regulatorios.

- **Protección de endpoints (XDR):** Además de su capacidad como SIEM, Wazuh se extiende a la protección de endpoints a través de su integración con XDR (Extended Detection and Response). Wazuh proporciona un monitoreo detallado de los endpoints, incluyendo estaciones de trabajo, servidores y dispositivos móviles. Esto permite una detección avanzada de amenazas en todos los niveles de la infraestructura, desde la red hasta los endpoints, proporcionando una respuesta rápida ante incidentes de seguridad.
- **Integración con otras herramientas de seguridad:** Wazuh se integra de manera fluida con otras herramientas de seguridad, como Elastic Stack (Elasticsearch, Logstash, Kibana) para la visualización y análisis avanzado de datos, y Kubernetes para su implementación en entornos de contenedores. La capacidad de Wazuh para trabajar con herramientas de terceros aumenta su eficacia y versatilidad, permitiendo una solución unificada de seguridad.
- **Análisis de vulnerabilidades:** Wazuh también realiza análisis de vulnerabilidades, identificando debilidades de seguridad en sistemas y aplicaciones antes de que sean explotadas por actores maliciosos. La integración con bases de datos de vulnerabilidades como CVEs y la comparación de las configuraciones de los sistemas con mejores prácticas ayudan a garantizar que los entornos estén protegidos contra amenazas conocidas.
- **Alertas y notificaciones configurables:** La capacidad de Wazuh para generar alertas personalizadas es clave para su efectividad. Las alertas pueden ser configuradas en función de la gravedad del incidente, y se pueden enviar por diferentes canales, incluyendo correo electrónico, Slack, o sistemas de gestión de incidentes. Esto permite a los equipos de seguridad reaccionar rápidamente ante incidentes y reducir el tiempo de respuesta ante amenazas.
- **Interfaz de usuario y visualización de datos:** Wazuh ofrece una interfaz web intuitiva que facilita la gestión y visualización de datos. A través de la interfaz de Kibana (parte del Elastic Stack), los usuarios pueden acceder a los informes de seguridad, visualizar las alertas y realizar análisis de tendencias y patrones a lo largo del tiempo. Esta interfaz permite que los administradores de seguridad puedan tomar decisiones informadas basadas en datos visuales y en tiempo real.

Funcionamiento de Wazuh SIEM

- **Arquitectura cliente-servidor distribuida:** Wazuh opera sobre una arquitectura cliente-servidor, en la que los agentes se instalan en los dispositivos que necesitan ser monitorizados, como servidores, estaciones de trabajo o dispositivos de red. Estos agentes recogen información de los logs del sistema, las aplicaciones y las redes. Posteriormente, los datos recolectados se envían al servidor Wazuh, donde son procesados y analizados. Los servidores Wazuh pueden estar distribuidos, lo que permite la escalabilidad en grandes infraestructuras de red.
- **Funcionalidad de los agentes:** Los agentes de Wazuh no solo recogen logs, sino que también están diseñados para realizar análisis de integridad en los archivos del sistema, realizar verificaciones de configuraciones de seguridad y proporcionar visibilidad de las amenazas en tiempo real. Cuando el servidor Wazuh detecta un evento de seguridad o



anomalía, genera una alerta que se puede visualizar en la interfaz de Kibana o enviarse a través de las notificaciones configuradas.

Aplicaciones de Wazuh SIEM

- **Detección de incidentes de seguridad:** Wazuh es especialmente útil para la detección de incidentes de seguridad, como intrusiones, accesos no autorizados y malware. Gracias a su integración con Elastic Stack, permite correlacionar los eventos y proporciona información detallada sobre las amenazas detectadas.
- **Análisis forense de seguridad:** En investigaciones post-incidente, Wazuh puede ser utilizado para analizar eventos previos a un ataque. El análisis de logs históricos y la correlación de eventos facilita la identificación de patrones maliciosos y la reconstrucción de incidentes de seguridad.
- **Cumplimiento de normativas de seguridad:** Como parte del cumplimiento normativo, Wazuh facilita la auditoría de los sistemas mediante la generación de informes detallados sobre el cumplimiento de estándares de seguridad. Esto es esencial para empresas que operan en sectores regulados como la banca, la salud o la tecnología.
- **Protección de infraestructuras de TI:** Wazuh se utiliza para proteger infraestructuras críticas, incluidas redes corporativas, servidores y aplicaciones. Sus capacidades de análisis de vulnerabilidades y su integración con herramientas de detección de amenazas avanzadas lo convierten en una solución integral de seguridad.

Ventajas y Limitaciones de Wazuh SIEM

- Ventajas:
 - Código abierto y sin costo: Wazuh es completamente gratuito y está basado en código abierto, lo que permite su personalización según las necesidades específicas de cada organización.
 - Escalabilidad: Wazuh se adapta a organizaciones de diferentes tamaños, desde pequeñas empresas hasta grandes corporaciones, gracias a su arquitectura distribuida.
 - Integración con Elastic Stack: La integración con Elastic Stack proporciona potentes capacidades de búsqueda y visualización de datos, lo que mejora la detección y análisis de eventos.
 - Cumplimiento normativo: Facilita la auditoría y el cumplimiento de regulaciones de seguridad mediante reglas y procedimientos específicos.
- Limitaciones:
 - Curva de aprendizaje: Aunque la interfaz es bastante amigable, la implementación y configuración inicial de Wazuh puede ser compleja para usuarios sin experiencia en administración de sistemas o seguridad.
 - Requiere recursos: El análisis de logs y eventos en tiempo real puede requerir recursos significativos, especialmente en entornos con grandes volúmenes de datos.



FTK Imager

FTK Imager es una herramienta forense digital desarrollada por AccessData, especializada en la adquisición y visualización de evidencia digital. Su propósito principal es crear imágenes forenses exactas de discos duros, particiones y otros dispositivos de almacenamiento, manteniendo la integridad de los datos originales. Aunque no es una herramienta de análisis forense en profundidad, FTK Imager es ampliamente utilizada en las etapas iniciales de una investigación digital por su facilidad de uso, precisión y compatibilidad con múltiples formatos y dispositivos.

Características Principales de FTK Imager

- **Creación de Imágenes Forenses:** Una de las principales funciones de FTK Imager es la creación de imágenes forenses. Este proceso implica la duplicación exacta de los datos de un dispositivo de almacenamiento, como discos duros, unidades USB o particiones, preservando la integridad de la información. FTK Imager permite la creación de imágenes en formatos estándar, como E01 y AFF, que son ampliamente aceptados en el ámbito forense.
- **Verificación de la Integridad de la Evidencia:** FTK Imager ofrece la capacidad de verificar la integridad de las imágenes creadas mediante el uso de sumas de verificación (hashing). Al calcular valores hash como MD5 o SHA1, la herramienta asegura que los datos adquiridos no hayan sido modificados en el proceso de adquisición, garantizando que la evidencia sea admisible en un tribunal.
- **Exploración y Visualización de Archivos:** Además de adquirir imágenes, FTK Imager permite la exploración de los archivos contenidos en una imagen forense, sin necesidad de restaurar completamente la imagen en el sistema. Los usuarios pueden ver la estructura de archivos, los metadatos y el contenido de los archivos, facilitando el análisis de los datos sin comprometer la evidencia original.
- **Soporte para Diversos Dispositivos:** FTK Imager es compatible con una amplia gama de dispositivos de almacenamiento, incluidos discos duros internos, discos ópticos, unidades USB, imágenes de disco de red, y dispositivos de almacenamiento en la nube. Su versatilidad permite a los investigadores forenses trabajar con una variedad de medios y realizar análisis de diferentes fuentes de evidencia.
- **Extracción de Archivos y Datos Específicos:** FTK Imager no solo permite la creación de imágenes completas, sino también la extracción selectiva de archivos y datos específicos de una imagen forense. Esto es útil cuando se desea obtener solo los archivos relevantes para la investigación sin tener que procesar toda la imagen, lo que optimiza el tiempo y los recursos del análisis.
- **Soporte de Sistemas de Archivos Comunes:** La herramienta es compatible con los sistemas de archivos más utilizados, como FAT32, NTFS, exFAT, HFS+ y EXT3/4, lo que facilita la adquisición y el análisis de dispositivos que utilizan diferentes tipos de sistemas de almacenamiento.
- **Interfaz de Usuario Intuitiva:** FTK Imager cuenta con una interfaz gráfica de usuario (GUI) intuitiva que simplifica el proceso de adquisición y análisis de evidencia. La interfaz permite una fácil navegación a través de las diversas opciones de configuración, visualización de archivos y gestión de imágenes forenses.



Usos Principales de FTK Imager

- **Adquisición Forense de Evidencia:** FTK Imager es fundamental para cualquier investigador digital cuando se trata de adquirir evidencia digital de manera forense. Su capacidad para crear imágenes forenses exactas garantiza que la información obtenida sea válida y no se vea alterada en el proceso, un requisito esencial para que la evidencia sea admisible en juicios legales.
- **Análisis Preliminar de Evidencia:** Gracias a su capacidad para explorar imágenes forenses sin necesidad de restaurarlas completamente, FTK Imager es útil para realizar análisis preliminares de la evidencia. Los investigadores pueden obtener una visión general rápida de los archivos y datos presentes en el dispositivo, lo que les permite identificar posibles áreas de interés sin comprometer la integridad de los datos originales.
- **Gestión de Evidencia Digital:** FTK Imager también se utiliza para gestionar evidencia digital a lo largo del ciclo de vida de la investigación, desde la adquisición hasta la generación de informes. La herramienta permite organizar y catalogar las imágenes forenses de manera eficiente, lo que facilita el proceso de auditoría y revisión en investigaciones complejas.
- **Pruebas de Integridad en Auditorías Forenses:** La capacidad de FTK Imager para verificar la integridad de las imágenes mediante el uso de sumas de verificación es crucial para el mantenimiento de la validez de la evidencia en auditorías forenses. Los valores hash calculados aseguran que los datos no se hayan alterado, proporcionando una capa adicional de seguridad en la cadena de custodia de la evidencia.

Ventajas y Limitaciones de FTK Imager

- **Ventajas:**
 - **Gratuita y Fácil de Usar:** FTK Imager es una herramienta gratuita, lo que la hace accesible para investigadores forenses en una variedad de entornos. Su interfaz fácil de usar reduce la curva de aprendizaje y permite una implementación rápida.
 - **Alta Precisión en la Adquisición de Datos:** Al crear imágenes bit a bit, FTK Imager garantiza una copia exacta del dispositivo original, lo que es esencial para evitar la alteración de la evidencia durante el proceso de adquisición.
 - **Compatibilidad con Diversos Sistemas de Archivos y Dispositivos:** FTK Imager es compatible con diversos sistemas de archivos y dispositivos, lo que la convierte en una herramienta versátil y adaptable a diferentes escenarios de investigación.
- **Limitaciones:**
 - **No Realiza un Análisis Exhaustivo:** Aunque FTK Imager es excelente para la adquisición y exploración de evidencia, no es una herramienta completa de análisis forense. Para un análisis más profundo de los datos, generalmente se requiere el uso de otras herramientas, como FTK o EnCase.
 - **Requiere Conocimientos Técnicos:** Aunque la herramienta es accesible, los investigadores deben tener un conocimiento técnico básico de las prácticas forenses y de los sistemas de archivos para utilizar FTK Imager de manera eficaz.

2.2. Planificación



Objetivo general

Definir y organizar el proceso para integrar varias herramientas forenses y de seguridad, con el fin de automatizar el análisis y monitorización de una imagen forense creada con FTK Imager, utilizando Autopsy, Cuckoo y Wazuh.

Pasos planificados

- Selección de herramientas: Se eligieron FTK Imager para la creación de imágenes forenses, Autopsy para el análisis inicial de la evidencia digital, Cuckoo Sandbox para el análisis dinámico automatizado, y Wazuh para la monitorización y generación de alertas.
- Preparación del entorno
 - Instalación y configuración de FTK Imager para la adquisición de imágenes bit a bit.
 - Configuración de Autopsy para cargar y examinar las imágenes adquiridas.
 - Instalación y configuración de Cuckoo Sandbox en un entorno aislado, preparado para ejecutar muestras automáticamente.
 - Instalación y ajuste de Wazuh para monitorizar logs y eventos relacionados con la ejecución de Cuckoo.
- Diseño del flujo de trabajo: Se planificó un proceso en el que:
 - La imagen adquirida con FTK Imager se analiza en Autopsy para extraer muestras o archivos de interés.
 - Un script automatiza el envío de estas muestras a Cuckoo para su análisis dinámico.
 - Wazuh monitoriza en tiempo real la actividad y resultados de Cuckoo, alertando sobre comportamientos sospechosos o eventos relevantes.
- Desarrollo del script de automatización: Creación de un script que facilite el envío automático de muestras desde Autopsy a Cuckoo y gestione la recepción y registro de los resultados.
- Validación y pruebas
 - Realizar pruebas con imágenes forenses de prueba para asegurar que el flujo se ejecuta correctamente.
 - Verificar que Wazuh detecta y reporta adecuadamente las alertas generadas.
 - Ajustar configuraciones y corregir errores detectados en las pruebas.

Recursos y materiales necesarios

- Equipo con suficiente capacidad para ejecutar máquinas virtuales y herramientas forenses.
- Imagen forense de prueba para análisis.
- Documentación y manuales de FTK Imager, Autopsy, Cuckoo y Wazuh.
- Entorno virtualizado para aislamiento y seguridad.

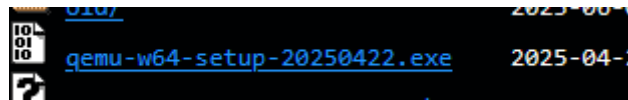
2.3. Descripción del trabajo realizado

Creación de imagen desde una máquina virtual (archivo .vdi o .vmdk)

Paso 1:

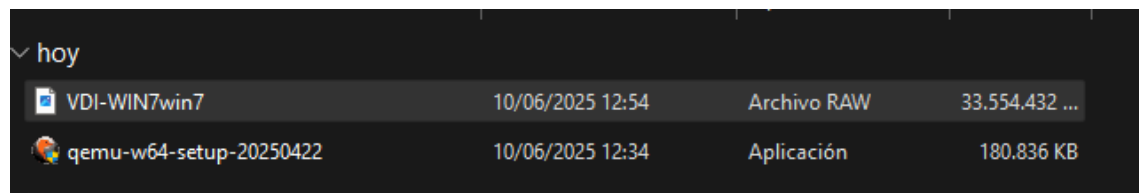
Primero debemos tener instalado qemu y FTK.

Es muy sencillo simplemente tenemos que aceptar los pasos y se instalará sin ningún problema.



Cuando los tengamos instalados, ejecutamos el siguiente comando con las rutas donde se encuentra el .vdi de Windows7 en Virtualbox y la ruta donde queremos que la imagen raw se guarde.

```
C:\Program Files\qemu>qemu-img.exe convert -f vdi -O raw "E:\Maquinas Windows\Windows 7\Windows 7.vdi" "C:\Users\alien\Downloads\VDI-WIN7win7.raw"  
C:\Program Files\qemu>
```

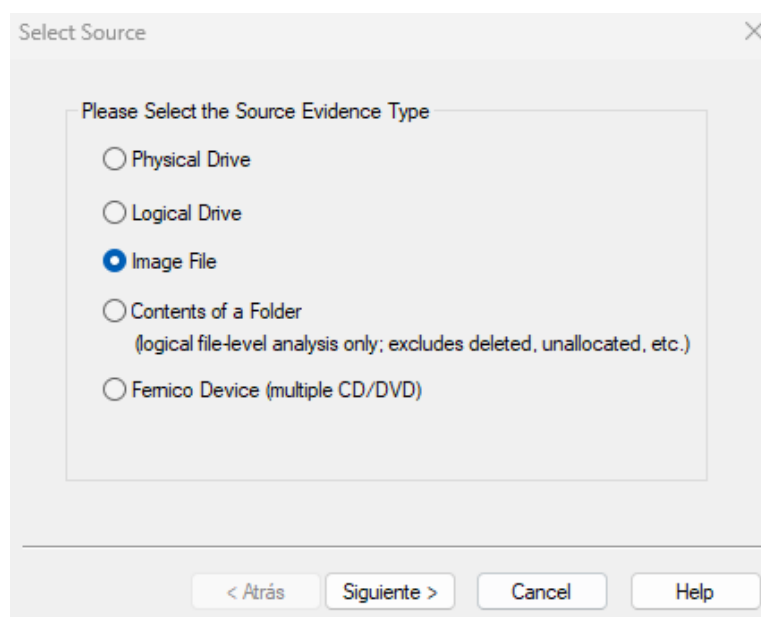


Comando:

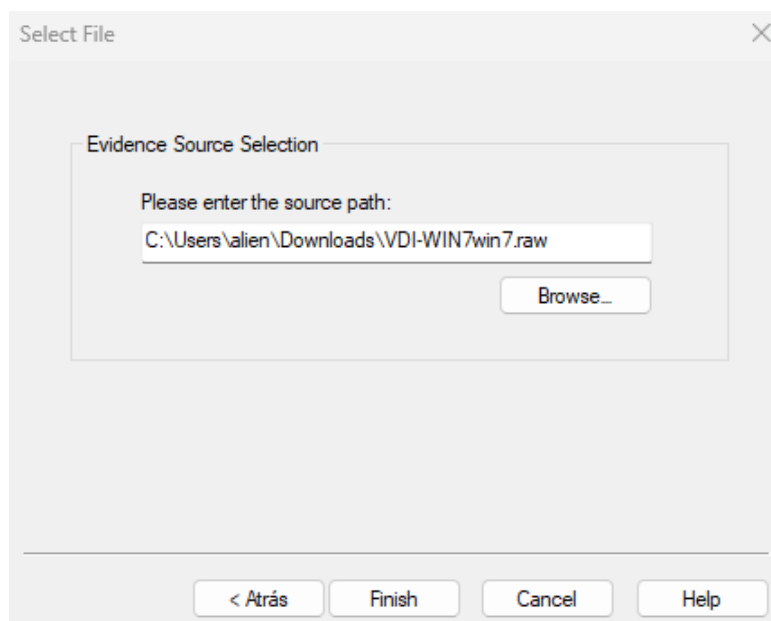
qemu-img.exe convert -f vdi -O raw "E:\Maquinas Windows\Windows 7\Windows 7.vdi" "C:\Users\alien\Downloads\VDI-WIN7win7.raw"

Paso 2: Convertir .raw a .E01 con FTK Imager

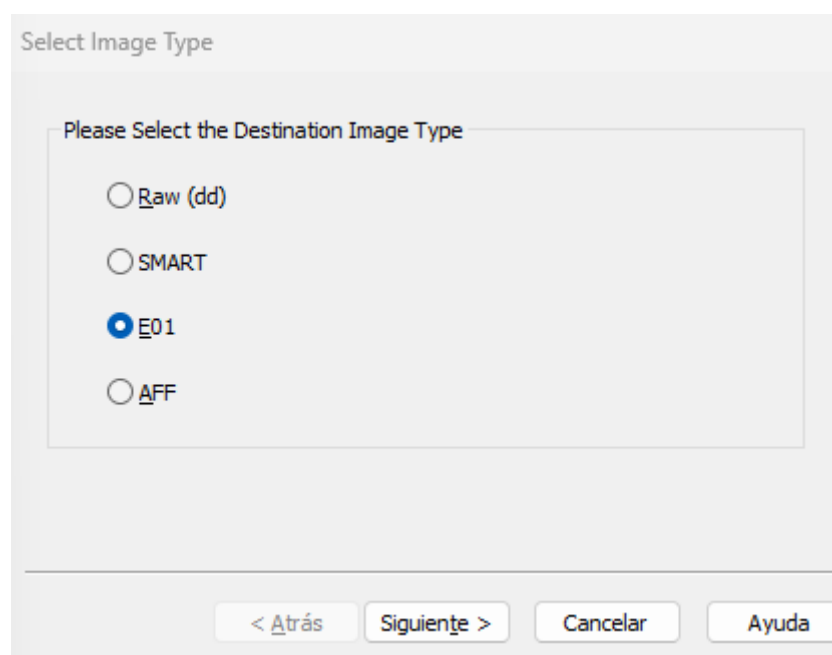
Abrimos FTK, y seleccionamos “Create Disk image”



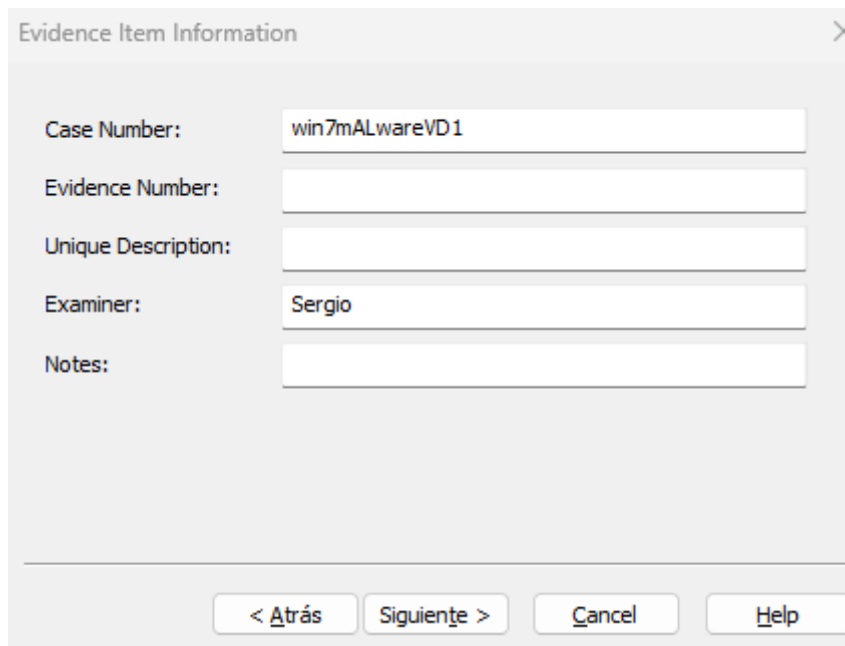
Seleccionamos el archivo raw, creado anteriormente con qemu.



Ahora elegimos E01, para realizar el análisis forense con autopsy en el futuro.



Rellenamos los campos necesarios.



Evidence Item Information

Case Number: win7mALwareVD1

Evidence Number:

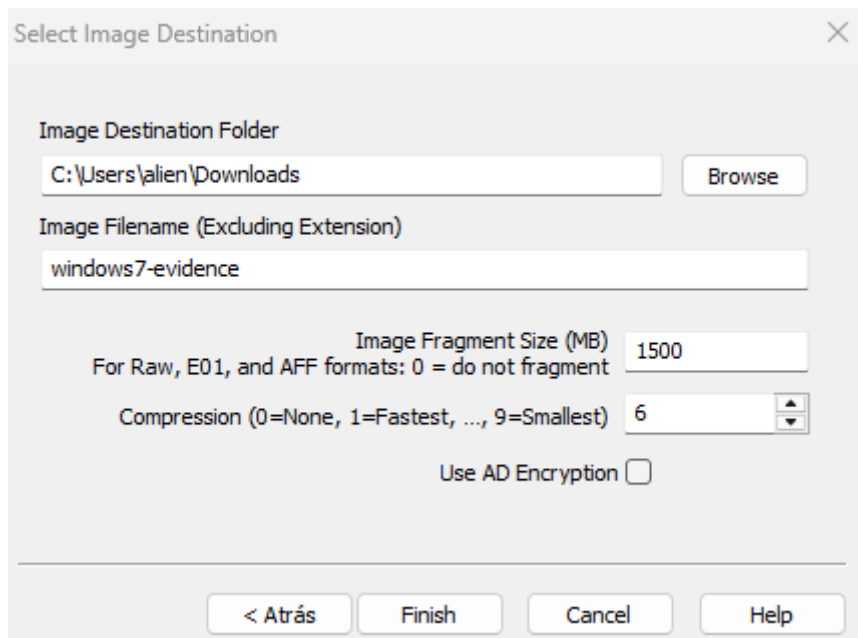
Unique Description:

Examiner: Sergio

Notes:

< Atrás Siguiete > Cancel Help

Seleccionamos las rutas adecuadas.



Select Image Destination

Image Destination Folder
C:\Users\alien\Downloads Browse

Image Filename (Excluding Extension)
windows7-evidence

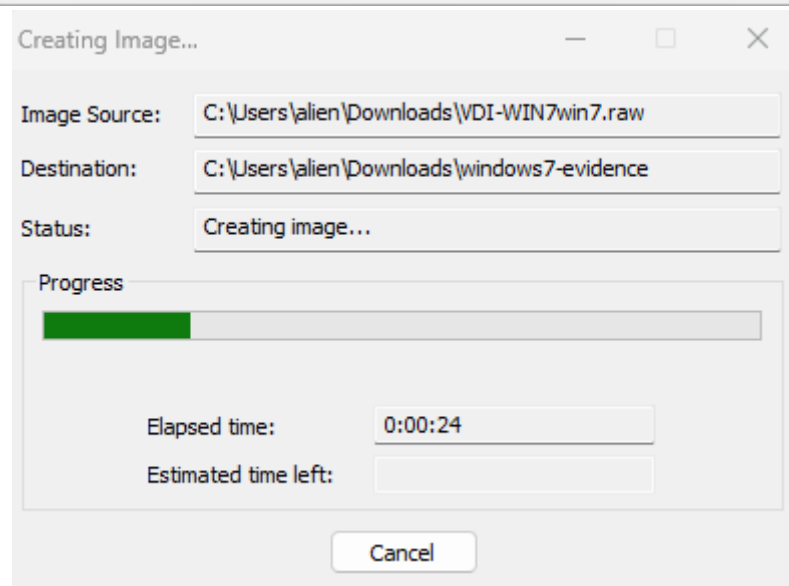
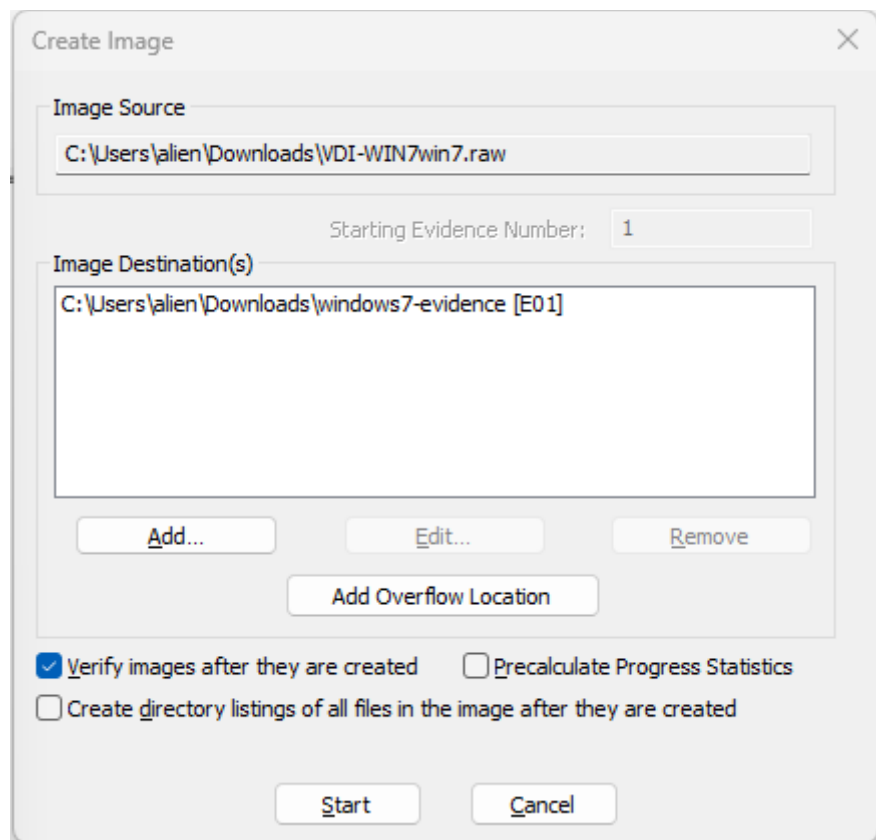
Image Fragment Size (MB)
For Raw, E01, and AFF formats: 0 = do not fragment 1500

Compression (0=None, 1=Fastest, ..., 9=Smallest) 6

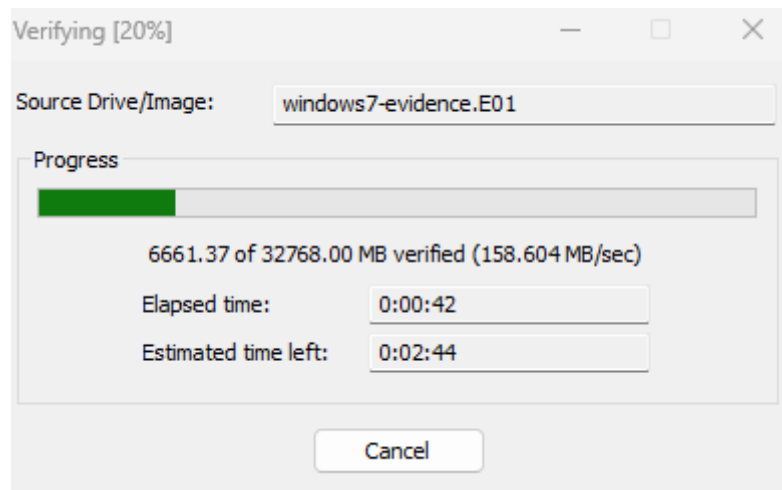
Use AD Encryption ☐

< Atrás Finish Cancel Help

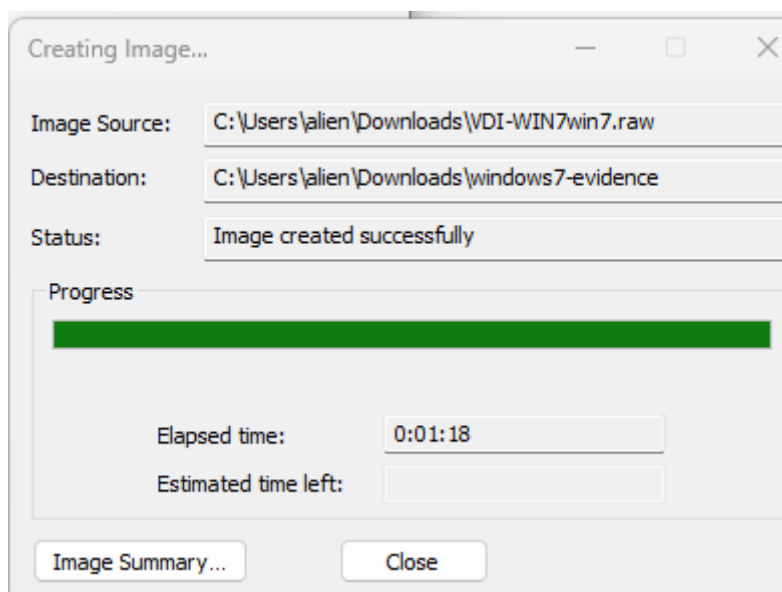
Cuando todo este listo, podemos hacer click en start.



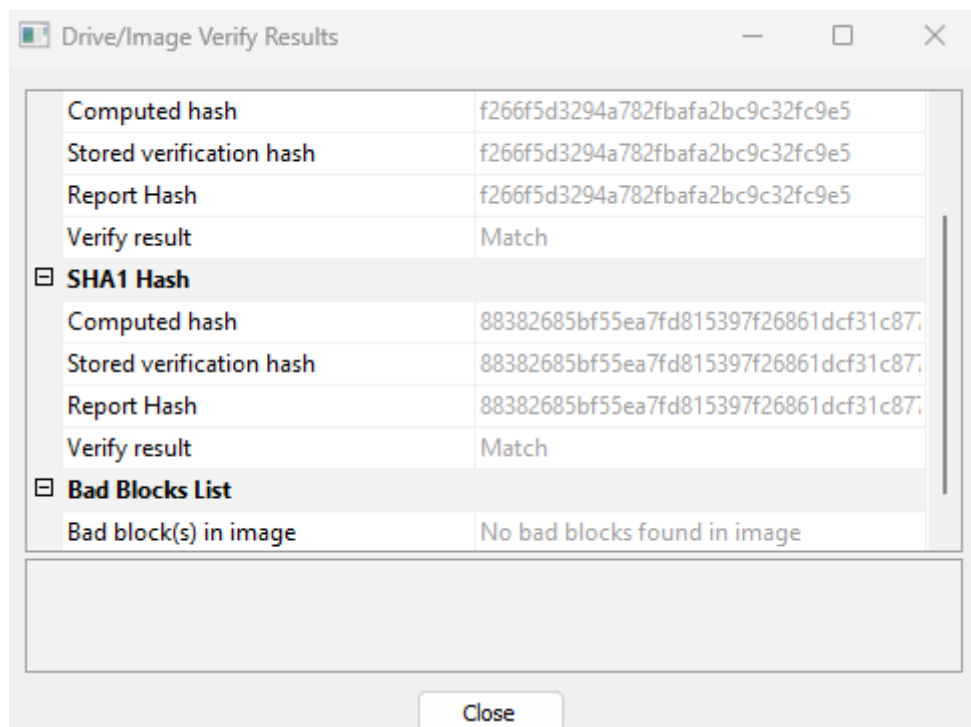
Esperamos a que se procese.



Vemos que a sido exitosa la creación de la imagen.



Aquí podemos ver los resultados, verificamos que no haya anomalías internas en los bloques al generar las imágenes.

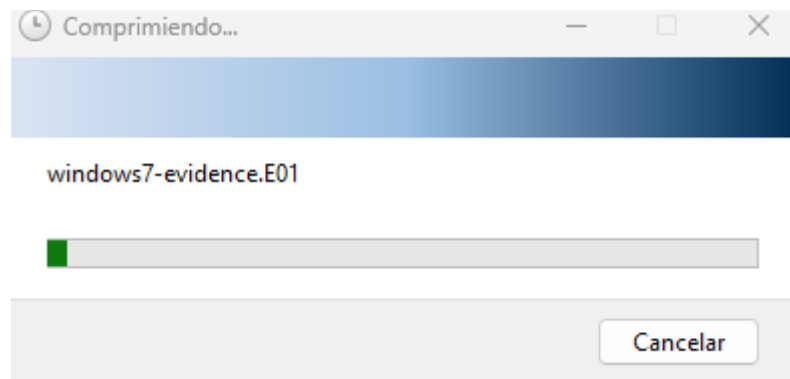


Estos archivos pueden analizarse directamente con herramientas como Autopsy o volver a cargarse en FTK Imager para explorarlos.

Es importante conservar todos los archivos juntos en una misma carpeta y no modificar sus nombres si se desea mantener la integridad de la evidencia.

Nombre	Fecha de modificación	Tipo	Tamaño
▼ hoy			
windows7-evidence.E01	10/06/2025 13:06	Documento de tex...	2 KB
windows7-evidence.E04	10/06/2025 13:04	Archivo E04	1.384.360 KB
windows7-evidence.E03	10/06/2025 13:03	Archivo E03	1.535.877 KB
windows7-evidence.E02	10/06/2025 13:03	Archivo E02	1.535.849 KB
windows7-evidence.E01	10/06/2025 13:03	Archivo E01	1.535.920 KB

Ahora procedo a comprimirla para poder pasarla a la máquina que contiene Autopsy.



Una vez que tenemos las evidencias en la máquina de autopsy, abrimos un nuevo caso





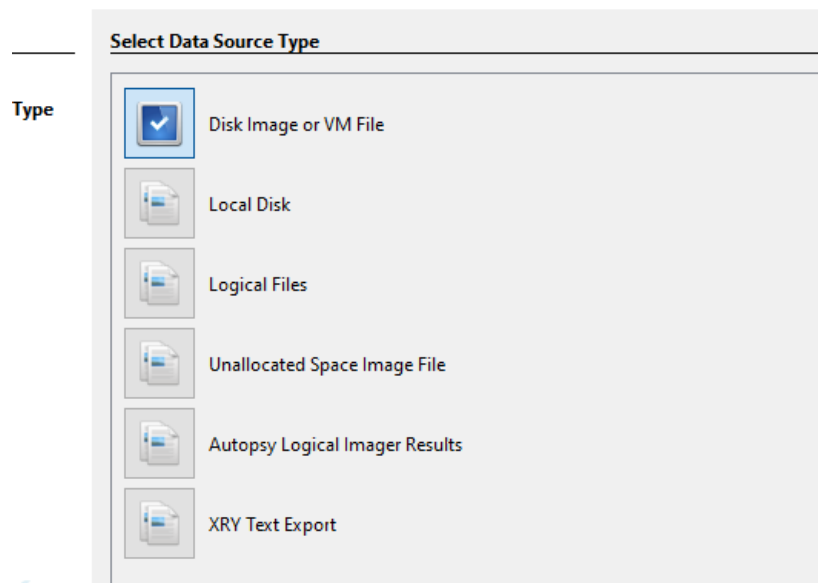
Vamos a establecer el nombre del caso “Caso_Windows_7” y la carpeta base es una carpeta compartida en el host principal que contiene el SIEM.

The screenshot shows the 'New Case Information' dialog box with the 'Case Information' tab selected. The 'Steps' list on the left shows '1. Case Information' and '2. Optional Information'. The 'Case Name' field contains 'Caso_Windows_7'. The 'Base Directory' field contains 'Z:\' and has a 'Browse' button next to it. The 'Case Type' section has two radio buttons: 'Single-User' (selected) and 'Multi-User'. Below this, a text box indicates 'Case data will be stored in the following directory:' followed by 'Z:\Caso_Windows_7'. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

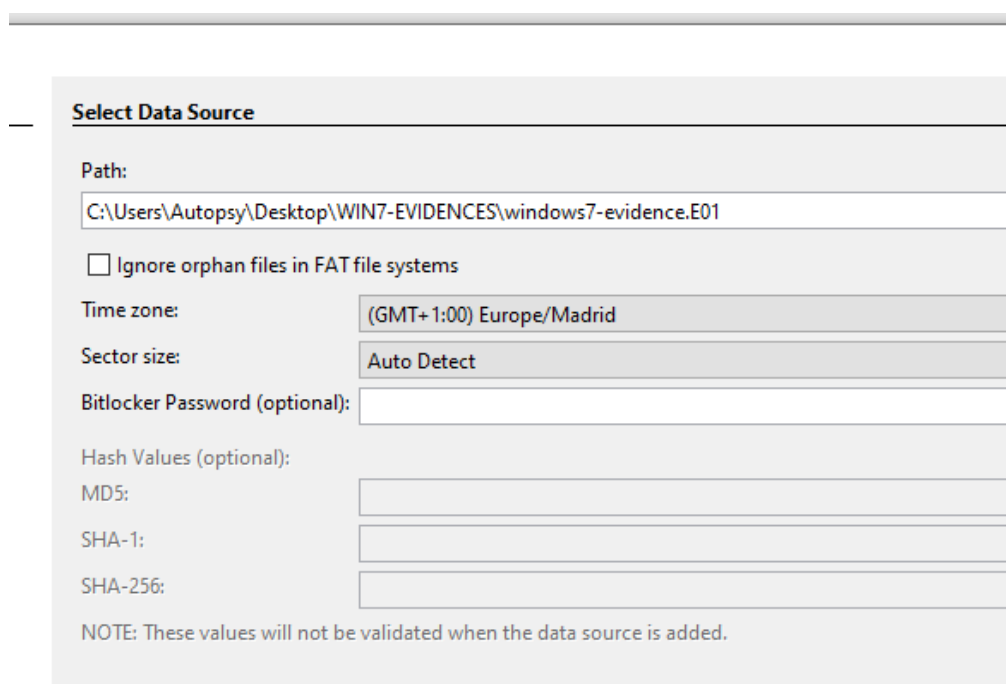
Ahora debemos rellenar los campos, esto saldrá como información en el reporte.

The screenshot shows the 'New Case Information' dialog box with the 'Optional Information' tab selected. The 'Steps' list on the left shows '1. Case Information' and '2. Optional Information'. The 'Case' section has a 'Number' field containing '001'. The 'Examiner' section has fields for 'Name' (containing 'Laboratorio'), 'Phone' (containing '666666666'), 'Email' (containing 'a@a.es'), and 'Notes' (empty). The 'Organization' section has a label 'Organization analysis is being done for:' followed by a dropdown menu set to 'Not Specified' and a 'Manage Organizations' button. At the bottom, there are buttons for '< Back', 'Next >', 'Finish', 'Cancel', and 'Help'.

Seleccionamos la opción “Disk Image or VM File”.



Seleccionamos la evidencia creada anteriormente con FTK.



Seleccionamos los módulos que nos interesen.

El módulo de detección de malware requiere una licencia de la que no disponemos.



Configure Ingest

Run ingest modules on:

All Files, Directories, and Unallocated Space

☒ Recent Activity
☒ Hash Lookup
☒ File Type Identification
☒ Extension Mismatch Detector
☒ Embedded File Extractor
☐ Picture Analyzer
☐ Keyword Search
☐ Email Parser
☐ Encryption Detection
☒ Interesting Files Identifier
☒ Central Repository
☒ PhotoRec Carver
☒ Virtual Machine Extractor
☒ Data Source Integrity
☐ Android Analyzer (aLEAPP)
☐ **Cyber Triage Malware Scanner**
☐ DJI Drone Analyzer
☐ Plaso
☒ YARA Analyzer
☐ iOS Analyzer (iLEAPP)
☒ GPX Parser
☐ Android Analyzer

Select All

Deselect All

History

Ingest Settings

This module requires a paid license.
See the Global Options panel for details

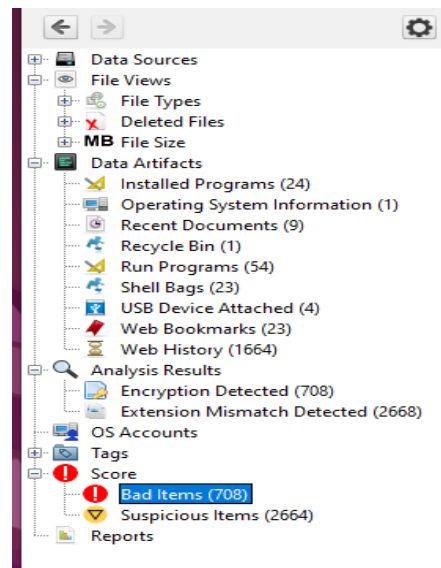
<

Identifies executable files with malware.

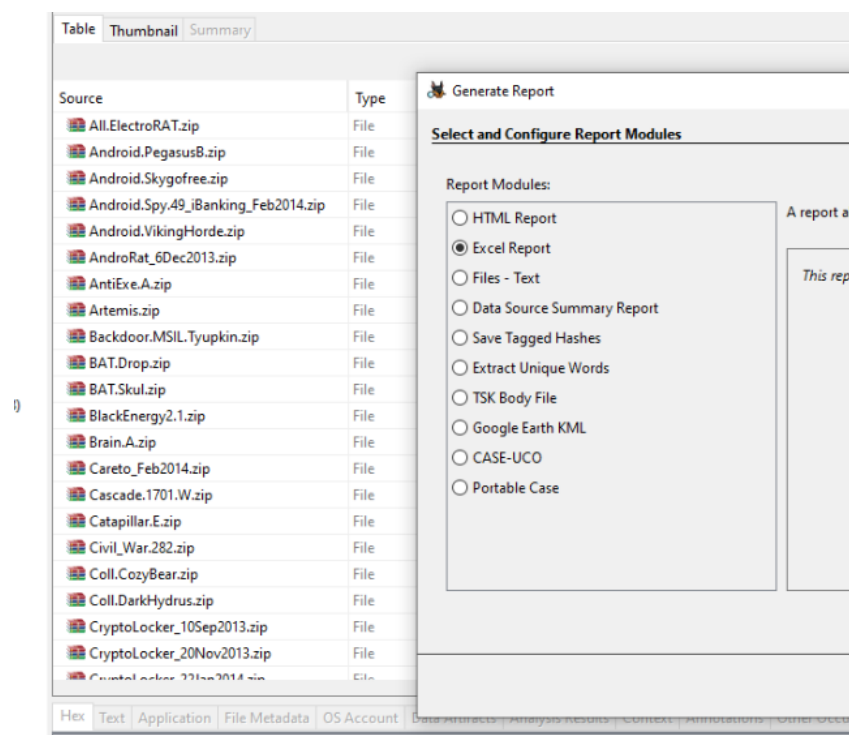
Global Set

Ahora cuando tengamos las opciones podemos empezar a analizar la imagen.

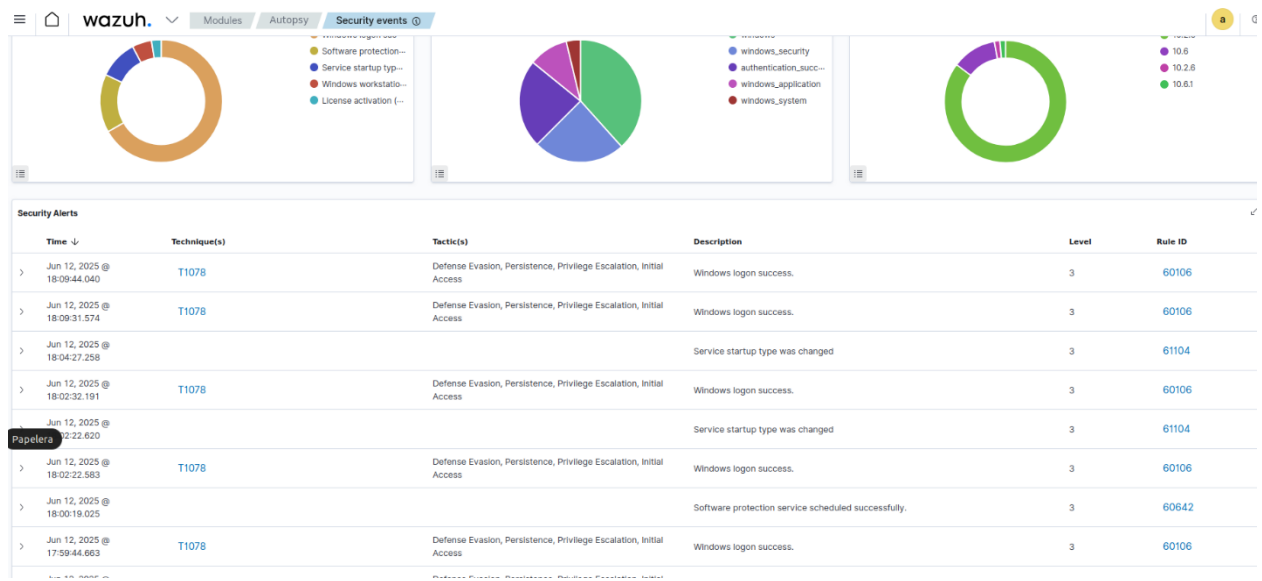
Cuando el escaneo finaliza podemos ver los “Bad Items y Suspicious Items” esto se refiere al malware añadido a la maquina virtual.



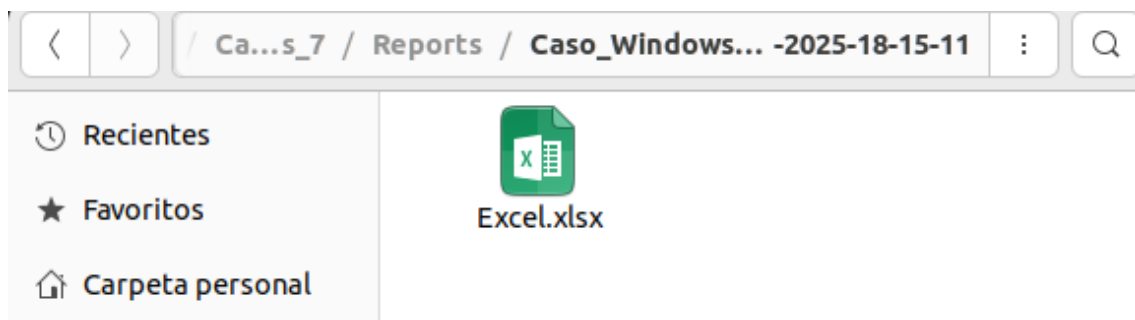
Ahora seleccionamos la opción “Generate Report” y seleccionamos formato Excel para el informe del apartado “Bad Items”.



Ahora nuestro SIEM Wazuh genera tráfico por lo que funciona correctamente.



Este es el reporte generado por autopsy en formato .xlsx.



Así es como se vería el excel, la parte de malware o archivos que pueden comprometer nuestro sistema.

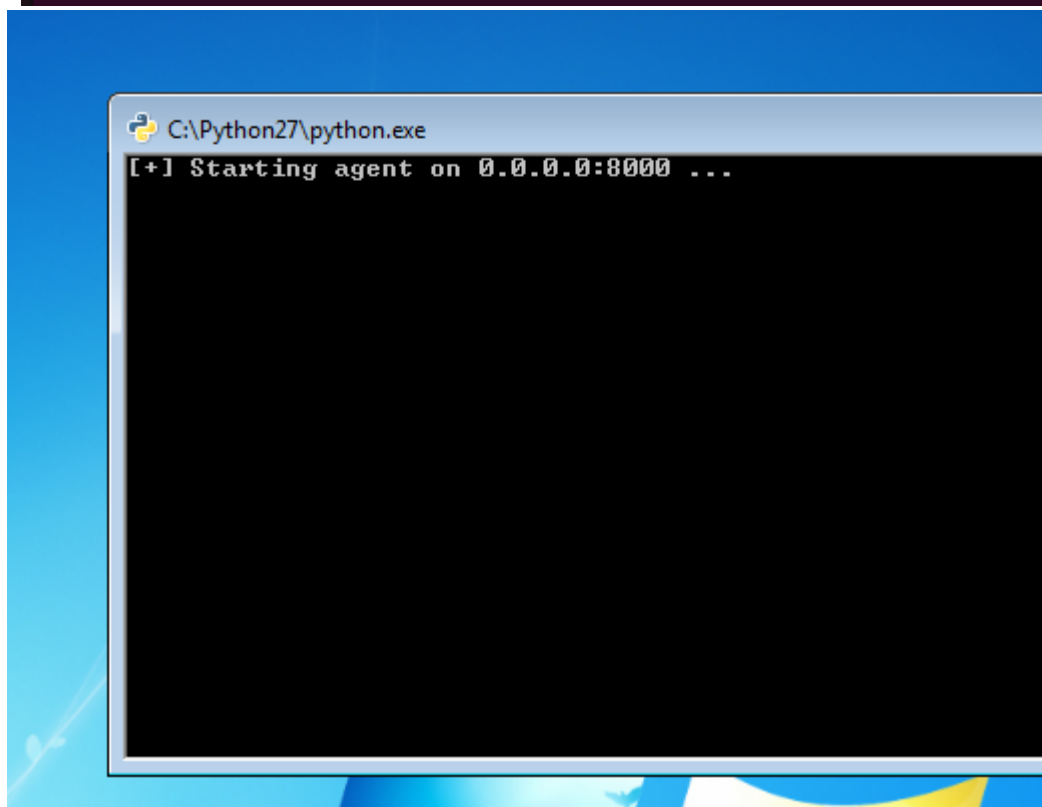
1	File
2	APT34.md5
3	APT34.md5
4	APT34.pass
5	APT34.pass
6	APT34.sha256
7	APT34.sha256
8	Alina.md5
9	Alina.md5
10	<u>Alina.pass</u>
11	<u>Alina.pass</u>
12	Alina.sha256
13	Alina.sha256
14	All.ElectroRAT.md5
15	All.ElectroRAT.md5
16	<u>All.ElectroRAT.pass</u>
17	<u>All.ElectroRAT.pass</u>
18	<u>All.ElectroRAT.shasum</u>
19	<u>All.ElectroRAT.shasum</u>
20	<u>AndroRat_6Dec2013.md5</u>
21	<u>AndroRat_6Dec2013.md5</u>
22	<u>AndroRat_6Dec2013.pass</u>
23	<u>AndroRat_6Dec2013.pass</u>
24	<u>AndroRat_6Dec2013.sha256</u>
25	<u>AndroRat_6Dec2013.sha256</u>
26	<u>Android.CEREBRUS.2.zip.001</u>
27	<u>Android.CEREBRUS.2.zip.001</u>
28	Android.CEREBRUS.md5
29	Android.CEREBRUS.md5
30	<u>Android.CEREBRUS.pass</u>
31	<u>Android.CEREBRUS.pass</u>
32	<u>Android.CEREBRUS.shasum</u>

Una vez terminamos con el autopsy, vamos a dejar en la carpeta de evidencias solamente 2 malwares detectados para su analisis en cuckoo. Si estuviéramos en un entrono real deberiamos ejecutar todo el malware para ver como actuan cada uno



Activamos las VM, en enviromnet de cuckoo y al agente en la maquina win7. Activmaos tambien la API de Cuckoo.

```
cuckoo@pablo-VirtualBox:~$ pyenv activate cuckoo-env  
(cuckoo-env) cuckoo@pablo-VirtualBox:~$
```



78

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~$ cuckoo api --host=0.0.0.0 --port=8090  
2025-06-12 19:06:44,135 [werkzeug] INFO: * Running on http://0.0.0.0:8090/ (Press CTRL+C to quit)
```

Ejecutamos el script autoreport.py en python.



```
laboratorio@laboratorio:~/Escritorio$ python3 autoreport.py
```

```
laboratorio@laboratorio:~/Escritorio$ python3 autoreport.py
[*] Iniciando análisis de archivos de Autopsy...
[~] Recuperando análisis pendiente: /home/laboratorio/evidencia_autopsy/Win32.nj
RAT/njRAT 0.3.6/bin/Debug/njRAT 0.3.6.vshost.exe
[✗] Error procesando pendiente: Error al consultar estado de la tarea

[+] Analizando: njRAT 0.3.6.vshost.exe
[→] Tarea creada en Cuckoo (ID 1)
port=8090
2025-06-12 19:17:35,768 [werkzeug] INFO: * Running on http://0.0.0.0:8090/ (Pre
s CTRL+C to quit)
2025-06-12 19:17:49,283 [werkzeug] INFO: 192.168.56.1 - - [12/Jun/2025 19:17:49]
"GET /tasks/view/4 HTTP/1.1" 404 -
```

En la maquina Windows 7, se pone en escucha en el puerto 8000, a la espera de la petición del script.

```
[+] Starting agent on 0.0.0.0:8000 ---
192.168.56.101 - - [12/Jun/2025 19:16:21] "POST /RPC2 HTTP/1.1" 200 -
```

El estado actual es “pending” es decir, está analizando el directorio y el malware.

```
[→] Tarea creada en Cuckoo (ID 2)
[~] Estado actual: pending
[~] Estado actual: pending
[~] Estado actual: pending
[~] Estado actual: running
[~] Estado actual: running
```

```
2025-06-12 19:44:48,509 [cuckoo.core.scheduler] INFO: Task #1: reports generati
on completed
2025-06-12 19:44:48,514 [cuckoo.core.scheduler] INFO: Task #1: analysis procedu
re completed
2025-06-12 19:46:48,244 [cuckoo.core.scheduler] INFO: Task #2: reports generati
on completed
2025-06-12 19:46:48,248 [cuckoo.core.scheduler] INFO: Task #2: analysis procedu
re completed
2025-06-12 19:46:55,409 [cuckoo.core.scheduler] INFO: Starting analysis of FILE
```

Vemos que empieza el análisis.



```
2025-06-12 19:46:55,435 [cuckoo.core.scheduler] INFO: Task #3: acquired machine
physical1 (label=win7-pc)
2025-06-12 19:46:55,470 [cuckoo.core.guest] INFO: Starting analysis #3 on guest
(id=physical1, ip=169.254.200.0)

[~] Estado actual: reported
[✓] Análisis completo. Resumen en: /home/laboratorio/resumenes_cuckoo_txt/stub.exe_20
250612_195315_resumen.txt

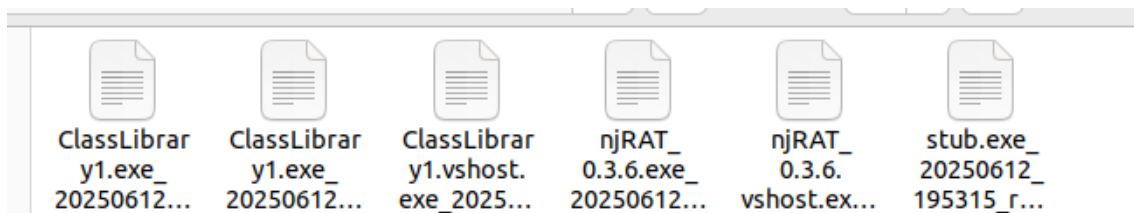
[+] Analizando: ClassLibrary1.vshost.exe
[DEBUG] HTTP 200
[DEBUG] Respuesta de Cuckoo: {
  "task_id": 5
}
```

El estado a cambiado a “reported”, es decir que el análisis a concluido de manera exitosa.

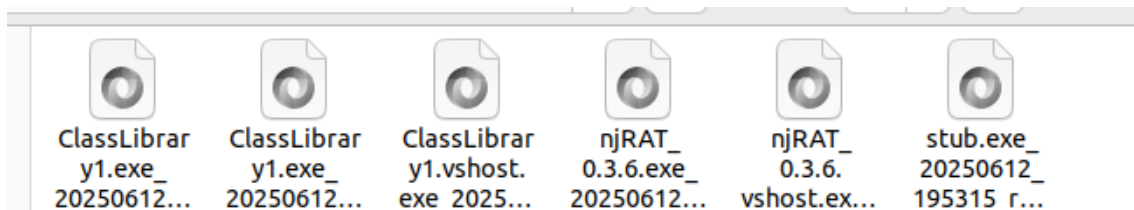
```
[~] Estado actual: running
[~] Estado actual: reported
[✓] Análisis completo. Resumen en: /home/laboratorio/resumenes_cuckoo_txt/ClassLibrar
y1.exe_20250612_200247_resumen.txt

[✓] Análisis finalizado.
```

Los reportes generados en formato .txt

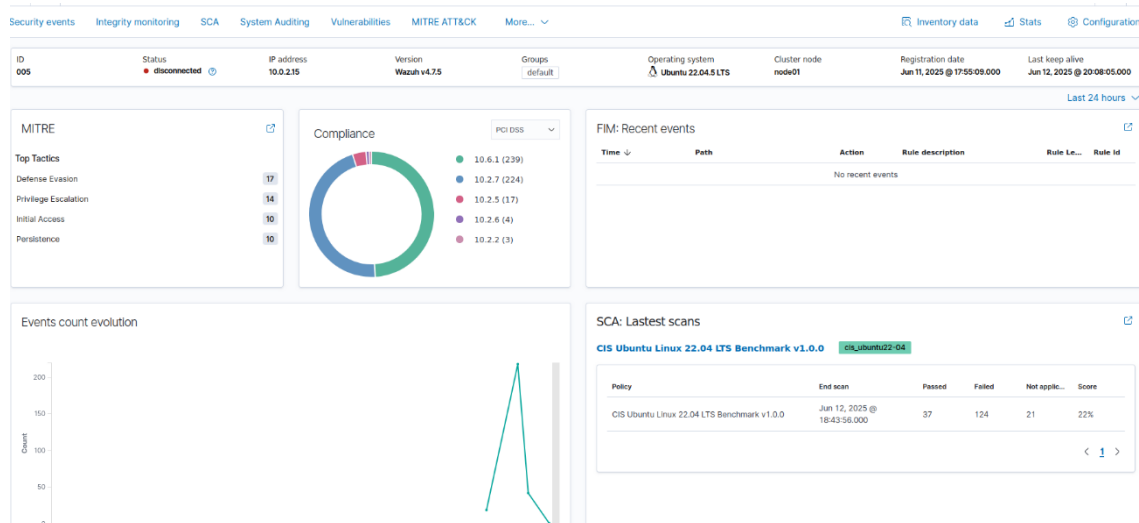


Los reportes generados en formato .json



En el SIEM Wazuh, vemos que el agente de la máquina a generádo logs.

El SIEM y los agentes funcionan de manera correcta en cada situación.



2.4. Resultados y validación

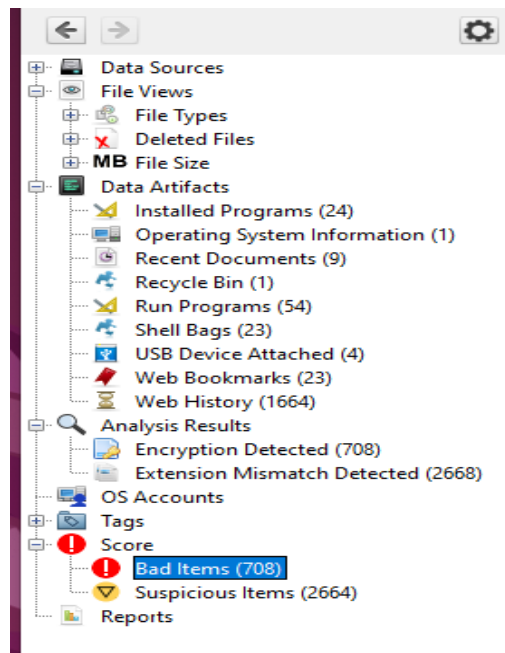
Una vez desplegado y configurado el laboratorio, se procedió a realizar análisis sobre muestras sospechosas utilizando herramientas de análisis forense tanto estático como dinámico, así como a verificar el correcto funcionamiento del sistema de monitorización. A continuación, se describen los principales resultados obtenidos.

Análisis estático con Autopsy

Tras la adquisición y carga de las imágenes de disco, se realizó el análisis forense con Autopsy. Esta herramienta permitió identificar diversos archivos potencialmente sospechosos, junto con sus metadatos asociados (fechas de creación, modificación, ubicación en el sistema, extensiones y asociaciones de usuario).

Como resultado, se generó un archivo en formato Excel que recoge de forma estructurada todos los artefactos extraídos durante el análisis. Este documento facilita la consulta ordenada de la información y constituye una base útil para investigaciones posteriores.

[Excel.xlsx](#)

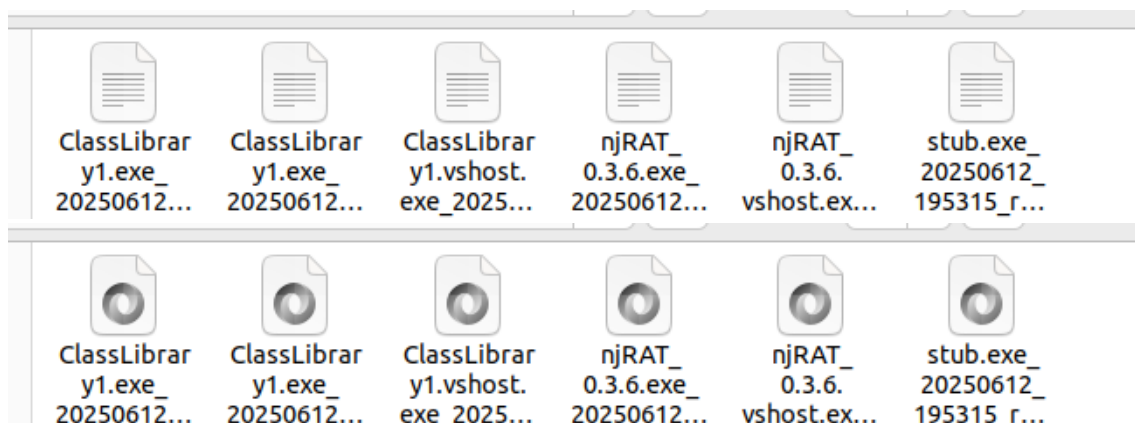


Análisis dinámico con Cuckoo Sandbox

Los archivos marcados como sospechosos fueron analizados dinámicamente con Cuckoo Sandbox. En esta fase se procesaron dos muestras distintas, cuyo comportamiento fue monitorizado y registrado en múltiples archivos de salida, entre ellos `.json` y `.txt`.

Estos archivos contienen datos sobre la actividad de cada muestra durante su ejecución, incluyendo creación de procesos, modificaciones en el sistema de archivos, consultas al registro e intentos de conexión a red.

En ninguno de los dos casos se detectaron firmas maliciosas conocidas, lo que podría deberse a la naturaleza de las muestras, a su posible ofuscación o a limitaciones en la base de datos de firmas disponible. Aun así, los registros obtenidos documentan el comportamiento observado y permiten evaluar el correcto funcionamiento del entorno de análisis.





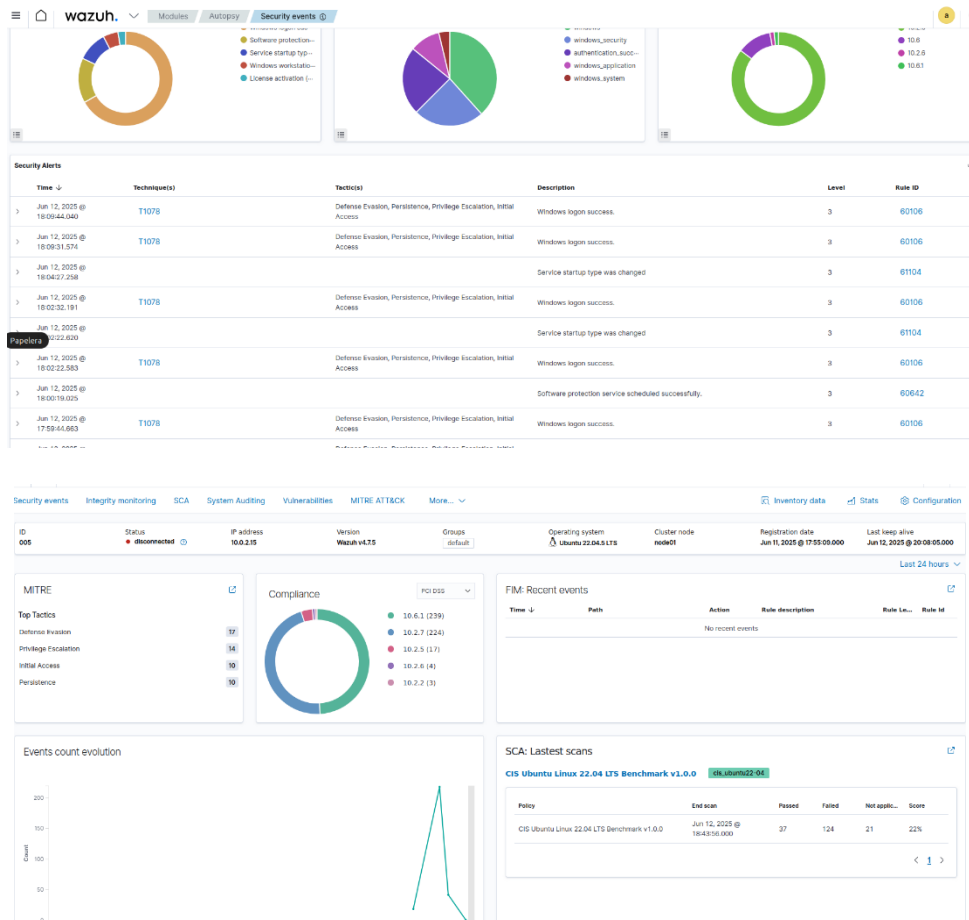
```
1 == RESUMEN DEL ANÁLISIS ==
2
3 Archivo: stub.exe
4 Puntuación (0-10): 0.0
5
6 Procesos detectados:
7 - No hay procesos
8
9 No se detectaron firmas maliciosas.
```

```
"info": {
  "added": 1749757615.031537,
  "started": 1749757615.379232,
  "duration": 180,
  "ended": 1749757796.083059,
  "owner": "",
  "score": 0.0,
  "id": 3,
  "category": "file",
  "gl": {
    "head": "13cbe0d9e457be3673304533043e992ead1ea9b2",
    "fetch_head": "13cbe0d9e457be3673304533043e992ead1ea9b2"
  },
  "monitor": "2deb9ccd75d5a7a3fe05b2625b03a8639ddee36b",
  "package": "",
  "route": "none",
  "custom": "",
  "machine": {
    "status": "stopped",
    "name": "physical",
    "label": "win7-pc",
    "manager": "Physical",
    "started_on": "2025-06-12 19:46:55",
    "shutdown_on": "2025-06-12 19:49:56"
  },
  "platform": "",
  "version": "2.0.7",
  "options": {}
},
"signatures": [],
"target": {
  "category": "file",
  "file": {
    "yara": [],
    "sha1": "7ed10507bad12b1119cbb8fb8413a6a68b3e9cd1",
    "name": "njRAT 0.3.0.exe",
    "type": "PE32 executable (GUI) Intel 80386 Mono/.Net assembly, for MS Windows",
    "sha256": "6dba01f4f5eb1d2d7e31ccdddbd73a2094a66d5bfe5b1d3ee14ff7b0b08004",
    "urls": [
      "http://xnjq8x.com"
    ],
    "crc32": "76815776",
    "path": "/home/cuckoo/.cuckoo/storage/binaries/6dba01f4f5eb1d2d7e31ccdddbd73a2094a66d5bfe5b1d3ee14ff7b0b08004",
    "sdeeph": null,
    "size": 1017344,
    "sha512": "710f09a349917cb5e51519924afdfc12f540a6c106a31574d821b952c1c7b1076bcff63792b4bffc42726494be4b520f65f1179ae5416daa14ac03bea56ebfe7",
    "md5": "2c8a8bd87d5ce8bcb4d39b5a9d24e9ca"
  }
},
}
```

Monitorización con Wazuh

Durante las distintas fases del proceso, el sistema de monitorización basado en **Wazuh** registró correctamente la actividad de los agentes desplegados. En el visor del SIEM se confirmó que el agente correspondiente a la máquina analizada generó los logs esperados, reflejando eventos del sistema relevantes para la seguridad.

Estos resultados validan que tanto el SIEM como los agentes funcionan de forma adecuada en distintos escenarios, lo que confirma la eficacia de la infraestructura de monitorización implementada.



3. CONCLUSIONES

El desarrollo de este proyecto nos ha permitido un aprendizaje enriquecedor tanto desde el punto de vista técnico como metodológico. La configuración de un entorno de laboratorio forense y de ciberseguridad funcional nos ha planteado varios desafíos y nos ha permitido una visión más amplia de como simular situaciones reales de análisis e investigación en entornos controlados.

A lo largo del proceso, se ha trabajado con diversas herramientas especializadas, cada una con requisitos de instalación y configuración muy distintos, lo que nos ha exigido un conocimiento detallado de diversos aspectos como la virtualización, la gestión de redes internas, la automatización mediante scripts y la integración de distintos sistemas entre sí. Estos entornos exigen una configuración precisa y bien planificada, donde cualquier error puede afectar directamente a la integridad de los datos o a la validez del análisis.

La configuración de herramientas como Cuckoo Sandbox, Wazuh y Autopsy ha permitido comprender cómo se implementan sistemas reales de monitorización, detección de amenazas y análisis de malware en entornos profesionales. También se ha aprendido a resolver problemas técnicos de instalación, compatibilidad entre versiones y dependencias, habilidades que resultan esenciales en cualquier entorno real de ciberseguridad.

En resumen, este proyecto no solo ha servido para aplicar conocimientos previos, sino también para desarrollar nuevas habilidades orientadas a la resolución de problemas reales, la investigación autónoma y el pensamiento crítico, preparando así una base sólida para afrontar la configuración entornos técnicos complejos en el ámbito profesional.

3.1 Aportaciones

Este proyecto ha dado lugar a una serie de aportaciones relevantes, tanto desde una perspectiva formativa como técnica. Una de las principales contribuciones es la integración de múltiples herramientas de ciberseguridad en un entorno unificado y funcional. La combinación de Autopsy, Cuckoo Sandbox y Wazuh en un mismo laboratorio, así como el uso de herramientas de creación de imágenes forenses como FTK Imager, permite cubrir distintas fases del análisis forense y de malware, desde la adquisición de la evidencia hasta su análisis estático, dinámico y la monitorización en tiempo real.

Otra contribución significativa es la automatización del flujo de trabajo entre herramientas. Se ha desarrollado un script que permite enviar automáticamente los archivos sospechosos identificados por Autopsy a Cuckoo Sandbox para su análisis dinámico. Esta automatización no solo reduce la intervención manual, sino que mejora la eficiencia y la trazabilidad del proceso forense, haciendo uso de prácticas utilizadas en entornos profesionales.

Por último, se ha elaborado una guía detallada de instalación y configuración de todas las herramientas utilizadas. Este recurso sirve como manual de referencia para reproducir el laboratorio desde cero, lo que facilita tanto su uso con fines educativos como su posible adaptación en otros proyectos o entornos profesionales.

Estas aportaciones suponen una base sólida y replicable para futuros trabajos en el ámbito de la seguridad informática, contribuyendo a la formación práctica y al desarrollo de soluciones funcionales en ciberseguridad.

3.2 Trabajo Futuro

Aunque el laboratorio desarrollado cumple con los objetivos propuestos, existen varias mejoras y ampliaciones que podrían abordarse en el futuro.

Una de las principales mejoras consistiría en la implementación de módulos personalizados en Autopsy, lo que permite ampliar sus capacidades de análisis y adaptarlas a necesidades específicas. Esta personalización facilitaría un estudio más exhaustivo de ciertos tipos de evidencia digital, optimizando el proceso forense según el contexto del caso.

En el caso de Cuckoo Sandbox, sería recomendable instalar y configurar adecuadamente los módulos comunitarios disponibles, como las firmas de comportamiento (signatures) y las reglas YARA, que enriquecen significativamente los informes generados y mejoran la precisión del análisis. Asimismo, la instalación de librerías adicionales, necesarias para ciertas funcionalidades avanzadas como la extracción de certificados digitales, permitiría ampliar las capacidades del sistema.

Una de las áreas con mayor potencial es la automatización de respuestas ante incidentes mediante el sistema de active-response. Esta función permite ejecutar acciones correctivas automáticamente ante ciertos eventos, como bloquear IPs maliciosas tras múltiples intentos fallidos de autenticación, detener procesos sospechosos o incluso aislar temporalmente un nodo comprometido. Esta capacidad reduce el tiempo de reacción humana y mitiga daños en tiempo real.

El control de integridad de archivos (File Integrity Monitoring) es una funcionalidad esencial para detectar cambios no autorizados en archivos críticos del sistema, configuraciones o scripts de producción. De cara al futuro, esta función se puede integrar con herramientas forenses o con bases de datos hash centralizadas para verificar la legitimidad de los cambios, fortaleciendo así la detección de intrusiones silenciosas.

Otra línea de evolución es la exportación de eventos a sistemas externos para análisis con algoritmos de machine learning. Mediante la integración con Elasticsearch, bases de datos o plataformas como Splunk, se pueden identificar patrones anómalos no visibles mediante reglas estáticas. Asimismo, es posible conectar Wazuh con plataformas de SIEM más complejas para reforzar las capacidades de análisis, cumplimiento y auditoría.

A medida que las empresas se enfrentan a normativas como GDPR, ISO 27001 o PCI-DSS, Wazuh puede configurarse para auditar sistemas y generar alertas en tiempo real sobre desviaciones o incumplimientos. Además, su capacidad de generar informes detallados permite llevar un control exhaustivo del estado de cumplimiento.

Otra mejora importante sería disponer de un equipo con mayores recursos de hardware. Las herramientas utilizadas en este proyecto, especialmente aquellas que implican virtualización intensiva como Cuckoo o monitorización a tiempo real, como Wazuh, pueden llegar a consumir una cantidad considerable de memoria y procesamiento. Contar con una infraestructura más potente permitiría realizar análisis más complejos en menos tiempo y con menor riesgo de bloqueos o ralentizaciones.

Además, en futuros desarrollos se podría contemplar la integración de herramientas adicionales que amplíen las capacidades del laboratorio. Algunas soluciones comerciales, como Magnet AXIOM, ofrecen funcionalidades avanzadas de análisis forense, recuperación de datos, detección de evidencia en la nube y automatización de flujos de trabajo. La incorporación de este tipo de herramientas, aunque suponga un mayor coste, permitiría agilizar procesos complejos y mejorar la calidad de los informes generados, especialmente en entornos profesionales.

También podría implementarse tecnologías como Docker para facilitar el despliegue del laboratorio, mejorar la portabilidad del entorno y simplificar tareas de mantenimiento o escalado.

4. BIBLIOGRAFÍA Y WEBGRAFÍA

- Sleuthkit. (s. f.-c). *autopsy/Running_Linux_OSX.md at develop · sleuthkit/autopsy*. GitHub. https://github.com/sleuthkit/autopsy/blob/develop/Running_Linux_OSX.md
- *Installation — Cuckoo Sandbox v2.0.7 book.* (s. f.). <https://cuckoo.readthedocs.io/en/latest/installation/>
- Caballero +Quezada, A. E. (2007). *Autopsy* (versión 1.1, 7 de noviembre de 2007) [PDF]. ReYDeS. Recuperado de https://www.reydes.com/archivos/autopsy_reydes.pdf
- Leandro. (2020, 20 abril). Autopsy, una herramienta de análisis digital forense. *tusclases.com*. <https://www.tusclases.com.ar/blog/autopsy-herramienta-analisis-digital-forense>
- Sala, J. (2025, 20 enero). *Autopsy, la herramienta forense por excelencia*. Infordisa / Security Operations Center. <https://www.infordisa.com/soc/autopsy-herramienta-forense/>
- *¿Cómo puede utilizar Cuckoo Sandbox para analizar malware y proteger su red?* (2023, 21 noviembre). [www.linkedin.com. https://es.linkedin.com/advice/0/how-can-you-use-cuckoo-sandbox-analyze-malware-2ewvf?lang=es&lang=es#:~:text=Una%20de%20las%20herramientas%20m%C3%A1s,aislados%20y%20observar%20su%20comportamiento](https://es.linkedin.com/advice/0/how-can-you-use-cuckoo-sandbox-analyze-malware-2ewvf?lang=es&lang=es#:~:text=Una%20de%20las%20herramientas%20m%C3%A1s,aislados%20y%20observar%20su%20comportamiento)
- Cilleruelo, C. (2024, 18 abril). ¿Qué es Cuckoo Sandbox? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. <https://keepcoding.io/blog/que-es-cuckoo-sandbox/>
- Echeverri, D. (2014, 18 noviembre). Cuckoo Sandbox y detección de malware - The Hacker Way. *The Hacker Way*. <https://thehackerway.es/2014/11/18/cuckoo-sandbox-y-deteccion-de-maleware/>
- Cilleruelo, C. (2024b, mayo 16). ¿Cómo usar Cuckoo Sandbox? | KeepCoding Bootcamps. *KeepCoding Bootcamps*. <https://keepcoding.io/blog/como-usar-cuckoo-sandbox/>
- NextTReT. (s.f.). *Wazuh: la solución de ciberseguridad de alta calidad* [página web]. Recuperado el 11 de junio de 2025, de <https://nexttret.net/ciberseguridad/wazuh/>
- Julia.Abril. (2024, 28 mayo). *Wazuh - Una plataforma de Código Abierto que unifica SIEM y XDR - Laboratori de recerca i innovació de la FIB*. Laboratori de Recerca I Innovació de la FIB. <https://inlab.fib.upc.edu/es/articulos/wazuh-una-plataforma-de-codigo-abierto-que-unifica-siem-y-xdr/2024/>
- Gálvez Soriano, P. (2024, 1 de julio). *Despliegue e implantación de un SIEM con Wazuh* [Trabajo de fin de grado, Universitat Politècnica de Catalunya]. UPCommons. <http://hdl.handle.net/2117/418178>
- *Configurar y usar Wazuh como SIEM.* (2025, 25 abril). 4Geeks. <https://4geeks.com/es/interactive-coding-tutorial/configurar-y-usar-wazuh-como-siem>

- Cilleruelo, C. (2024c, mayo 29). ¿Qué es FTK Imager? [2025] | KeepCoding Bootcamps. *KeepCoding Bootcamps*. <https://keepcoding.io/blog/que-es-ftk-imager-y-para-que-sirve/>
- Cilleruelo, C. (2024d, junio 28). ¿Qué es el análisis de evidencia con FTK Imager? [2025]. *KeepCoding Bootcamps*. <https://keepcoding.io/blog/analisis-de-evidencia-con-ftk-imager/>
- OpenAI. (2025). ChatGPT (GPT-4). <https://chat.openai.com/>



5. ANEXO

5.1. SIEM en Ubuntu 22.04.2:

Instalación

Debemos ejecutar los siguientes comandos para la instalación del SIEM

```
curl -sO https://packages.wazuh.com/4.7/wazuh-install.sh
```

```
bash wazuh-install.sh -a
```

```
alien@alien-SIEM:~$ sudo bash wazuh-install.sh -a
27/04/2025 09:16:05 INFO: Starting Wazuh installation assistant. Wazuh version: 4.7.5
27/04/2025 09:16:05 INFO: Verbose logging redirected to /var/log/wazuh-install.log
27/04/2025 09:16:14 INFO: --- Dependencies ---
27/04/2025 09:16:14 INFO: Installing gawk.
27/04/2025 09:16:17 INFO: Wazuh web interface port will be 443.
27/04/2025 09:16:20 INFO: --- Dependencies ---
27/04/2025 09:16:20 INFO: Installing apt-transport-https.
27/04/2025 09:16:24 INFO: Wazuh repository added.
27/04/2025 09:16:24 INFO: --- Configuration files ---
27/04/2025 09:16:24 INFO: Generating configuration files.
27/04/2025 09:16:25 INFO: Created wazuh-install-files.tar. It contains the Wazuh cluster key, cert
necessary for installation.
27/04/2025 09:16:25 INFO: --- Wazuh indexer ---
27/04/2025 09:16:25 INFO: Starting Wazuh indexer installation.
```

Al finalizar nos dará unas credenciales por defecto.

```
27/04/2025 09:20:24 INFO: You can access the web in
User: admin
Password: jKv2nFydFgNtV*o60btu0GIUV4LHyfan
27/04/2025 09:20:24 INFO: --- Dependencies ---
```

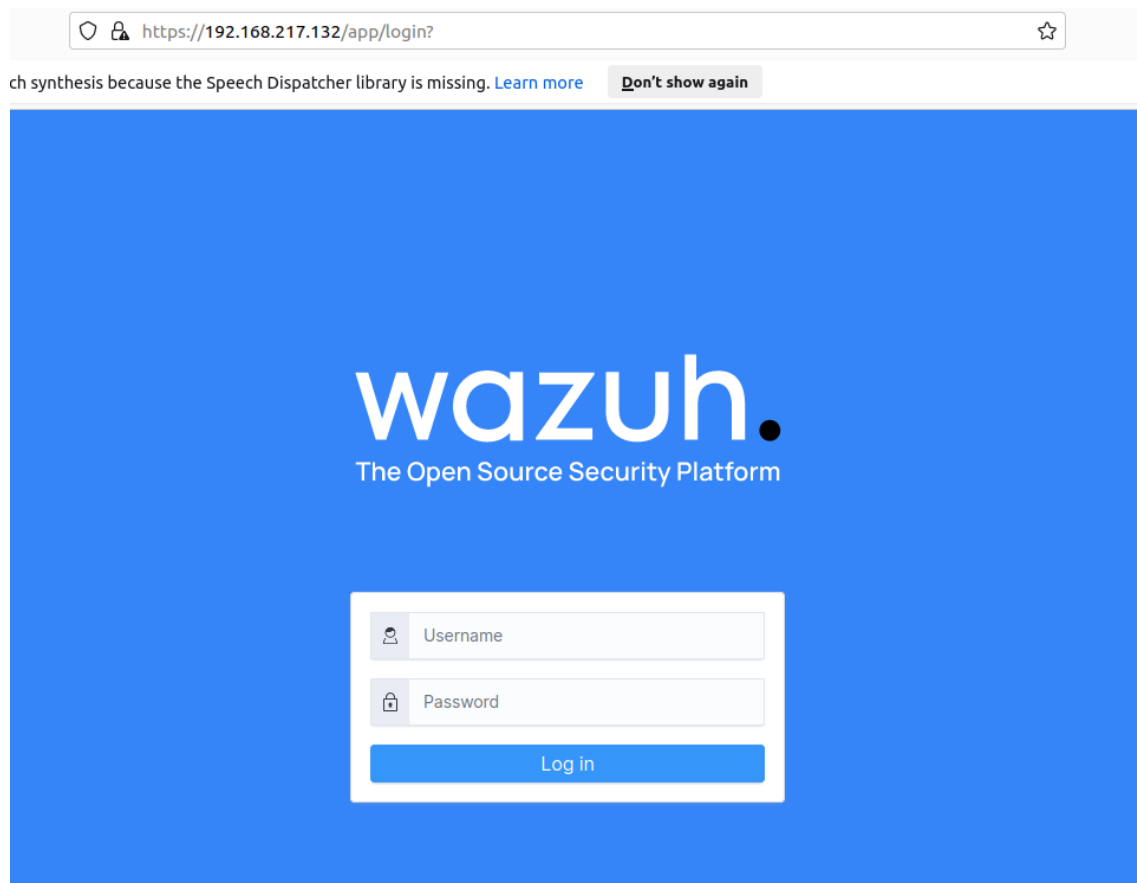
admin

DTwRaFseU7Est+Ri1*4V+HLJJgjkDZ6Q

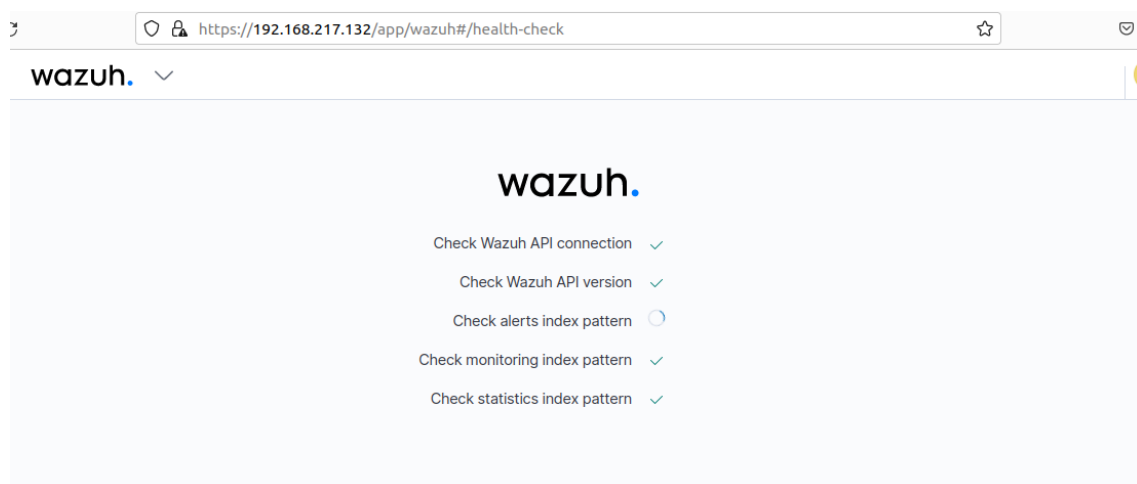
Una vez termine, abre tu navegador y entra en:

<https://Dirección IP:443>

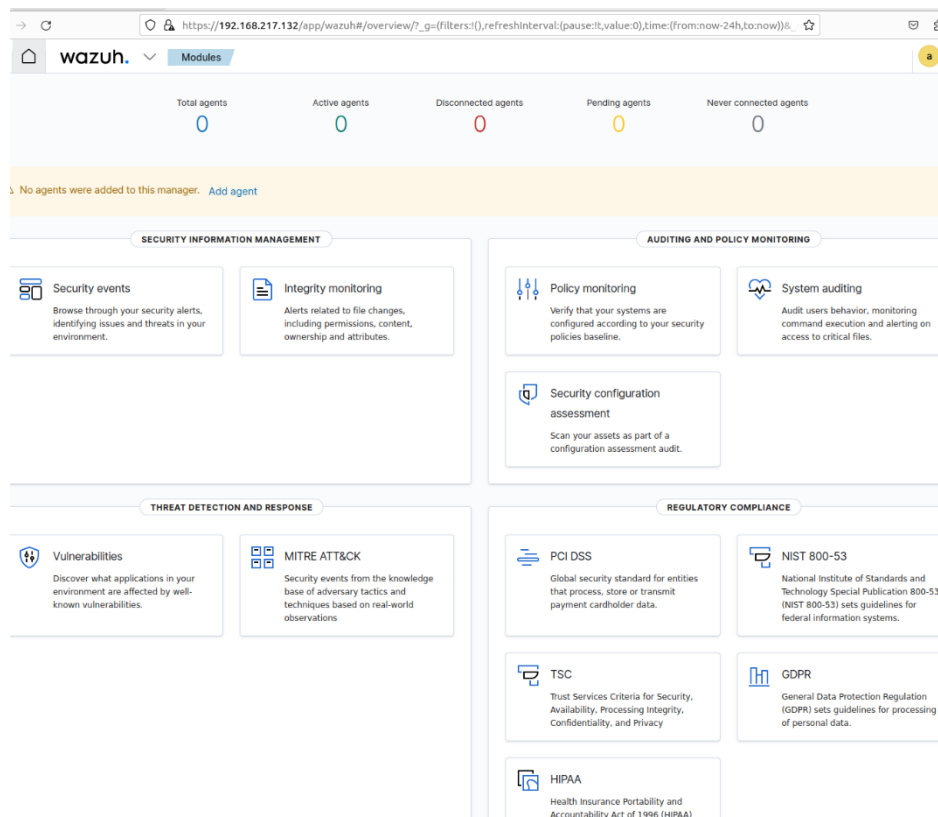
- Aquí podemos ver la interfaz web de Wazuh



- Vamos a insertar nuestras credenciales y esperar a que todo se inicie de manera correcta.



- Vista general de nuestro nuevo SIEM Wazuh.



Antes de nada nuestro SIEM consume una gran cantidad de recursos, por lo que vamos a bajar el uso de RAM a 2GB para no colapsar el sistema.

```
GNU nano 6.2 /etc/wazuh-indexer/jvm.options
## -Xmx4g
##
## See https://opensearch.org/docs/opensearch/install/important-settings/
## for more information
##
#####

# Xms represents the initial size of total heap space
# Xmx represents the maximum size of total heap space

-Xms2048m
-Xmx2048m

#####
## Expert settings
#####
```

El siguiente paso será configurar los agentes es decir, las maquinas virtuales que van a pasar su información por medio del SIEM.

Para instalar agentes Wazuh en otras máquinas para que envíen logs a nuestro SIEM debemos ejecutar los siguientes comandos en nuestro cliente.



Instalación en cuckoo

1. Nos descargamos el agente de Wazuh mediante el comando curl

```
cuckoo@pablo-VirtualBox:~$ curl -O https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent_4.7.5-1_amd64.deb
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           %             %             Dload  Upload  Total   Spent    Left   Speed
100 9159k  100 9159k    0     0  19.7M      0  --:--:-- --:--:-- --:--:--  19.7M
```

```
curl -O https://packages.wazuh.com/4.x/apt/pool/main/w/wazuh-agent/wazuh-agent\_4.7.5-1\_amd64.deb
```

2. Configuramos la dirección a la que se conectará el agente y el nombre de este nuevo agente que se creará

```
cuckoo@pablo-VirtualBox:~$ sudo WAZUH_MANAGER='192.168.1.139' WAZUH_AGENT_NAME='Cuckoo' dpkg -i wazuh-agent_4.7.5-1_amd64.deb
Seleccionando el paquete wazuh-agent previamente no seleccionado.
(Leyendo la base de datos ... 217824 ficheros o directorios instalados actualmente.)
Preparando para desempaquetar wazuh-agent_4.7.5-1_amd64.deb ...
Desempaquetando wazuh-agent (4.7.5-1) ...
Configurando wazuh-agent (4.7.5-1) ...
```

3. Autorizamos el agente para que saque automáticamente la key

```
2025/06/11 17:55:24 agent-auth: ERROR: Unable to add agent (from manager)
cuckoo@pablo-VirtualBox:~$ sudo /var/ossec/bin/agent-auth -m 192.168.1.139 -A Cuckoo
2025/06/11 17:55:09 agent-auth: INFO: Started (pid: 6594).
2025/06/11 17:55:09 agent-auth: INFO: Requesting a key from server: 192.168.1.139
2025/06/11 17:55:09 agent-auth: INFO: No authentication password provided
2025/06/11 17:55:09 agent-auth: INFO: Using agent name as: Cuckoo
2025/06/11 17:55:09 agent-auth: INFO: Waiting for server reply
2025/06/11 17:55:09 agent-auth: INFO: Valid key received
```

4. Activamos el agente

```
cuckoo@pablo-VirtualBox:~$ sudo systemctl enable --now wazuh-agent
Created symlink /etc/systemd/system/multi-user.target.wants/wazuh-agent.service
→ /lib/systemd/system/wazuh-agent.service.
```

5. Comprobamos que está activo en nuestro dashboard


Agents (1)

id=000 and

status=active

Deploy new agent

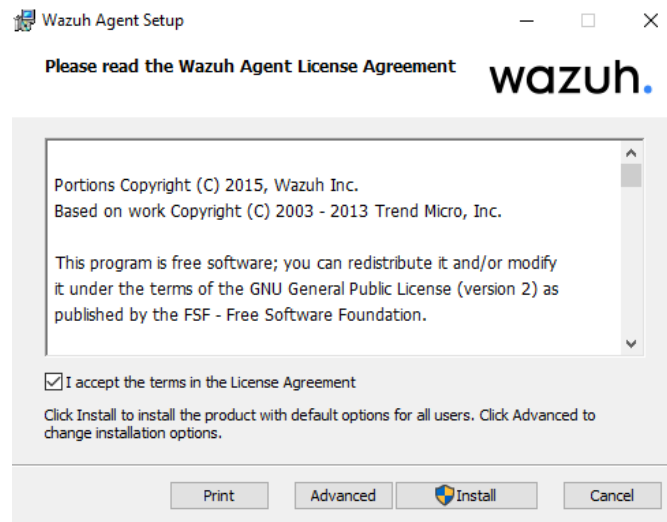
Refresh

ID ↑	Name	IP address	Group(s)	Operating system	Cluster node	Version	Status
005	Cuckoo	10.0.2.15	default	 Ubuntu 22.04.5 LTS	node01	v4.7.5	<div><div></div>active</div>

Rows per page: 10

Instalación en Autopsy

Nos descargamos el agente. En nuestro caso no estamos usando la última versión de Wazuh por temas de compatibilidad, pero lo que debemos bajarnos el agente de la versión que corresponde, en nuestro caso la 4.7.5



Una vez instalado, debemos crear el agente y conectarlo a nuestro host. Para ello desde nuestro host vamos a acceder al “agent manager”, donde creamos el agente y generamos la key para acceder.

```
laboratorio@laboratorio:~$ sudo /var/ossec/bin/manage_agents
[sudo] contraseña para laboratorio:

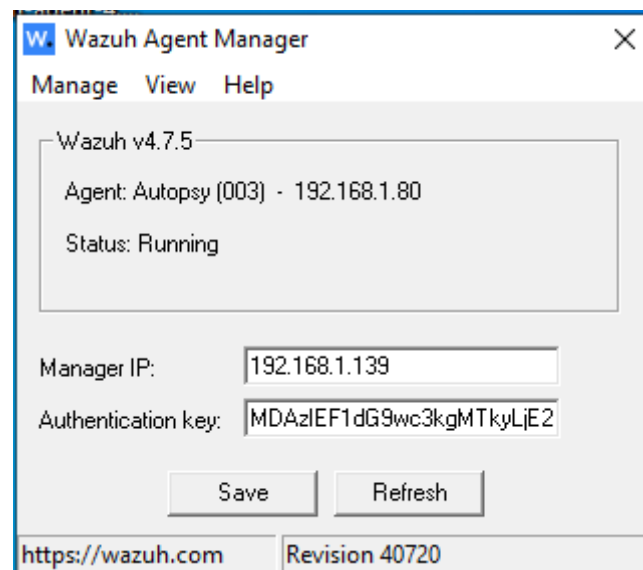
*****
* Wazuh v4.7.5 Agent manager.                *
* The following options are available:        *
*****
(A)dd an agent (A).
(E)xtract key for an agent (E).
(L)ist already added agents (L).
(R)emove an agent (R).
(Q)uit.
Choose your action: A,E,L,R or Q: e

Available agents:
  ID: 003, Name: Autopsy, IP: 192.168.1.80
Provide the ID of the agent to extract the key (or '\q' to quit): 003

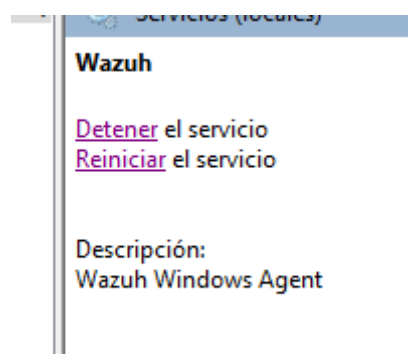
Agent key information for '003' is:
MDAzIEF1dG9wc3kgMTkyLjE2ODc4xLjgwIDBiMWZlZDRhbnZBknjY1ZDRiNjgwODEzODk0MTBhODA4Mzky
ZGM4YWZiNmY0Yzg1ZjE2ODc5NTNjNTg3YzQ3ZDY=

** Press ENTER to return to the main menu.
```

Una vez generado insertamos la ip de nuestra maquina host y la key correspondiente y se nos vinculará con wazuh



Activamos el servicio del agente de wazuh.



Comprobamos que el agente está activo.

```
laboratorio@laboratorio:~$ sudo /var/ossec/bin/agent_control -l

Wazuh agent_control. List of available agents:
  ID: 000, Name: laboratorio (server), IP: 127.0.0.1, Active/Local
  ID: 003, Name: Autopsy, IP: 192.168.1.80, Active

List of agentless devices:
```



Y ya podemos visualizar desde el dashboard de wazuh la actividad de la máquina.

SCA: Lastest scans [🔗](#)

CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0 cis_win10_enterprise

Policy	End scan	Passed	Failed	Not applic...	Score
CIS Microsoft Windows 10 Enterprise Benchmark v1.12.0	Jun 11, 2025 @ 17:21:21.000	115	275	4	29%

5.2. Configuración Cuckoo

Creamos el usuario cuckoo:

```
pablo@pablo-VirtualBox:~$ sudo adduser cuckoo
info: Añadiendo el usuario 'cuckoo' ...
info: Selecting UID/GID from range 1000 to 59999 ...
info: Añadiendo el nuevo grupo 'cuckoo' (1001) ...
info: Adding new user 'cuckoo' (1001) with group 'cuckoo (1001)' ...
info: Creando el directorio personal '/home/cuckoo' ...
info: Copiando los ficheros desde '/etc/skel' ...
Nueva contraseña:
CONTRASEÑA INCORRECTA: La contraseña tiene menos de 8 caracteres
Vuelva a escribir la nueva contraseña:
Las contraseñas no coinciden.
Nueva contraseña:
CONTRASEÑA INCORRECTA: La contraseña no supera la verificación de diccionario -
Es demasiado simple/sistemática.
Vuelva a escribir la nueva contraseña:
passwd: contraseña actualizada correctamente
Cambiando la información de usuario para cuckoo
Introduzca el nuevo valor, o pulse INTRO para usar el valor predeterminado
    Nombre completo []:
    Número de habitación []:
    Teléfono del trabajo []:
    Teléfono de casa []:
    Otro []:
¿Es correcta la información? [S/n] S
```

Lo agregamos a los grupos, grupo super usuario “sudo”, tcpdump y virtual box users:



```
pablo@pablo-VirtualBox:~$ sudo usermod -a -G sudo cuckoo
pablo@pablo-VirtualBox:~$ sudo usermod -a -G tcpdump cuckoo
pablo@pablo-VirtualBox:~$ sudo usermod -a -G vboxusers cuckoo
pablo@pablo-VirtualBox:~$
```

Vemos los grupos.

```
pablo@pablo-VirtualBox:~$ groups cuckoo
cuckoo : cuckoo sudo users tcpdump vboxusers
```

Instalamos una utilidad de Python que convierte la instrucción python3 en Python para más comodidad:

```
pablo@pablo-VirtualBox:~$ sudo apt-get install python-is-python3
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
Leyendo la información de estado... Hecho
Solving dependencies... Hecho
Se instalarán los siguientes paquetes NUEVOS:
python-is-python3
```

```
pablo@pablo-VirtualBox:~$ sudo apt-get install python3-pip python3-dev libffi-dev libssl-dev
```

```
pablo@pablo-VirtualBox:~$ sudo apt install libjpeg-dev zlib1g-dev swig
zlib1g-dev ya está en su versión más reciente (1:1.3.dfsg+really1.3.1-1ubuntu1).
fijado zlib1g-dev como instalado manualmente.
```

Accedemos al usuario creado anteriormente, en este caso cuckoo.

```
cuckoo@pablo-VirtualBox:/home/pablo$ sudo su cuckoo
[sudo] contraseña para cuckoo:
cuckoo@pablo-VirtualBox:/home/pablo$ cd /home/cuckoo
cuckoo@pablo-VirtualBox:~$
```

Agregamos variables de entorno en Ubuntu, utilizaremos pyenv ya que Cuckoo Sandbox requiere de una herramienta específica de Python.

```
export PYENV_ROOT="$HOME/.pyenv"
command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"
eval "$(pyenv init -)"
eval "$(pyenv virtualenv-init -)"
```

export PYENV_ROOT="\$HOME/.pyenv": Define la variable de entorno PYENV_ROOT, que le dice a tu sistema dónde está instalado pyenv. Guarda ~/.pyenv como directorio raíz de pyenv



(ahí es donde se almacenarán las versiones de Python instaladas con pyenv).

`command -v pyenv >/dev/null || export PATH="$PYENV_ROOT/bin:$PATH"`: Comprueba si el comando pyenv existe, en caso de no existir añade “\$PYENV_ROOT/bin” al PATH para que el sistema donde encontrarlo. Resumidamente, asegura que puedas usar el comando pyenv desde cualquier terminal.

`eval "$(pyenv init -)"`: inicializa pyenv en tu Shell. Permite que pyenv gestione qué versión de Python se usa cuando escribes python o pip

`eval "$(pyenv virtualenv-init -)"`: Habilita el uso de pyenv-virtualenv. Puedes usar pyenv virtualenv <version> <env-name> para crear entornos virtuales personalizados.

Instalamos pyenv con el usuario cuckoo con el siguiente comando, aunque vamos a necesitar instalar curl y git para poder ejecutarlo:

```
cuckoo@pablo-VirtualBox:~$ curl https://pyenv.run | bash
No se ha encontrado la orden «curl», pero se puede instalar con:
sudo snap install curl # version 8.13.0, or
sudo apt install curl # version 8.12.1-3ubuntu1
Consulte «snap info curl» para ver más versiones.
```

```
cuckoo@pablo-VirtualBox:~$ sudo apt install curl
[sudo] contraseña para cuckoo:
```

```
cuckoo@pablo-VirtualBox:~$ sudo apt-get install git
Leyendo lista de paquetes... Hecho
Creando árbol de dependencias... Hecho
```

```
cuckoo@pablo-VirtualBox:~$ curl https://pyenv.run | bash
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
100 270 100 270    0    0  485      0  --:--:-- --:--:-- --:--:--  487
Clonando en '/home/cuckoo/.pyenv'...
remote: Enumerating objects: 1380, done.
remote: Counting objects: 100% (1380/1380), done.
```

Con esto ya instalamos pyenv.

Instalamos python2.7.18:

```
cuckoo@pablo-VirtualBox:~$ pyenv install 2.7.18
Downloading readline-8.0.tar.gz...
-> https://ftpmirror.gnu.org/readline/readline-8.0.tar.gz
```



Nos aparecerán algunos errores sobre dependencias que faltan (bzip2 lib y SQLite3 lib), para ello ejecutamos los siguientes comandos:

```
cuckoo@pablo-VirtualBox:~$ sudo apt-get install libreadline-dev
```

```
cuckoo@pablo-VirtualBox:~$ sudo apt-get install libsqlite3-dev
```

```
cuckoo@pablo-VirtualBox:~$ sudo apt-get install libbz2-dev
```

Utilizamos la versión de python que hemos instalado y a continuación instalamos cuckoo:

```
cuckoo@pablo-VirtualBox:~$ python --version
Python 3.13.3
cuckoo@pablo-VirtualBox:~$ pyenv global 2.7.18
cuckoo@pablo-VirtualBox:~$ python --version
Python 2.7.18
```

Creamos la variable de entorno, en este caso se llamará cuckoo-env:

```
cuckoo@pablo-VirtualBox:~$ pyenv virtualenv 2.7.18 cuckoo-env
```

A continuación, activamos la variable de entorno:

```
cuckoo@pablo-VirtualBox:~$ pyenv activate cuckoo-env
(cuckoo-env) cuckoo@pablo-VirtualBox:~$ env
```

Instalamos cuckoo:

```
cuckoo@pablo-VirtualBox:~$ pip install cuckoo
```


Puede presentarnos un error, en ese caso tenemos que hacer lo siguiente:

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~$ sudo apt-get install execstack
[sudo] contraseña para cuckoo:
Leyendo lista de paquetes... Hecho
```

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~$ cd /home/cuckoo/.pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components$ sudo cp aplib-64.so aplib-64.so.bak
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components$ execstack -c aplib-64.so
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components$ execstack -q aplib-64.so
- aplib-64.so
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components$ cuckoo init
```

Iniciamos cuckoo:

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~/pyenv/versions/2.7.18/envs/cuckoo-env/lib/python2.7/site-packages/roach/native/components$ cuckoo init
```



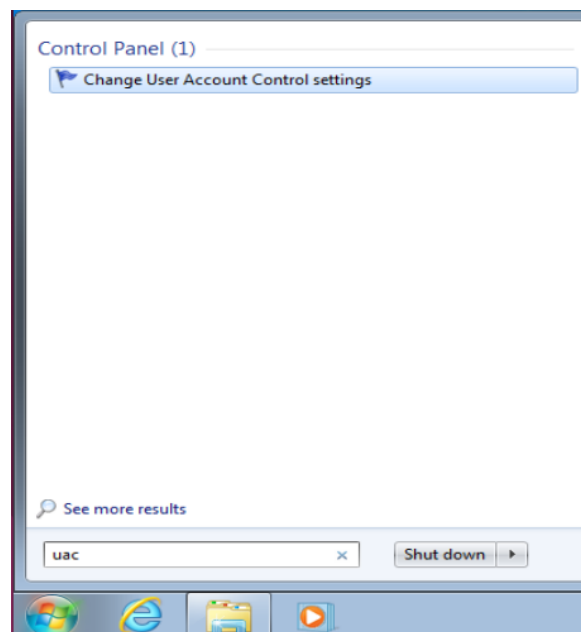
```
Cuckoo Sandbox 2.0.7
www.cuckoosandbox.org
Copyright (c) 2010-2018

=====
Welcome to Cuckoo Sandbox, this appears to be your first run!
We will now set you up with our default configuration.
You will be able to see and modify the Cuckoo configuration,
Yara rules, Cuckoo Signatures, and much more to your likings
by exploring the /home/cuckoo/.cuckoo directory.

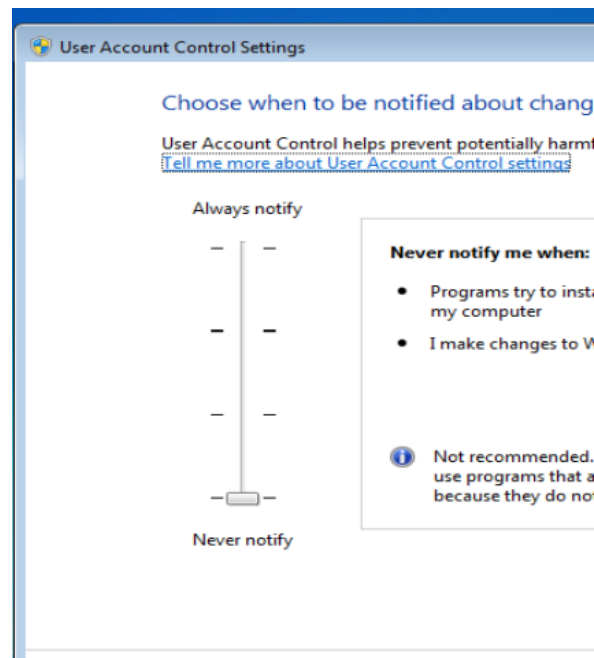
Among other configurable items of most interest is the
```

Configuración win7 cuckoo

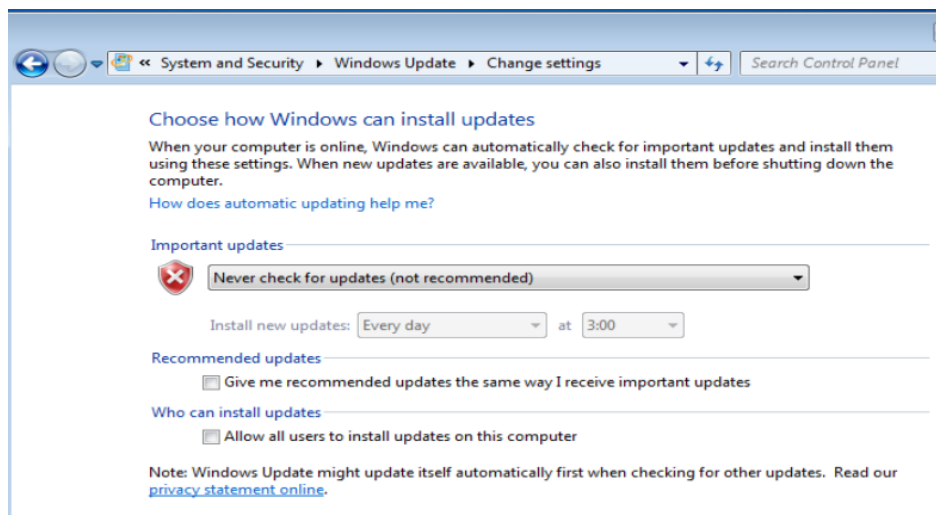
Desactivamos el control de cuantes de usuarios



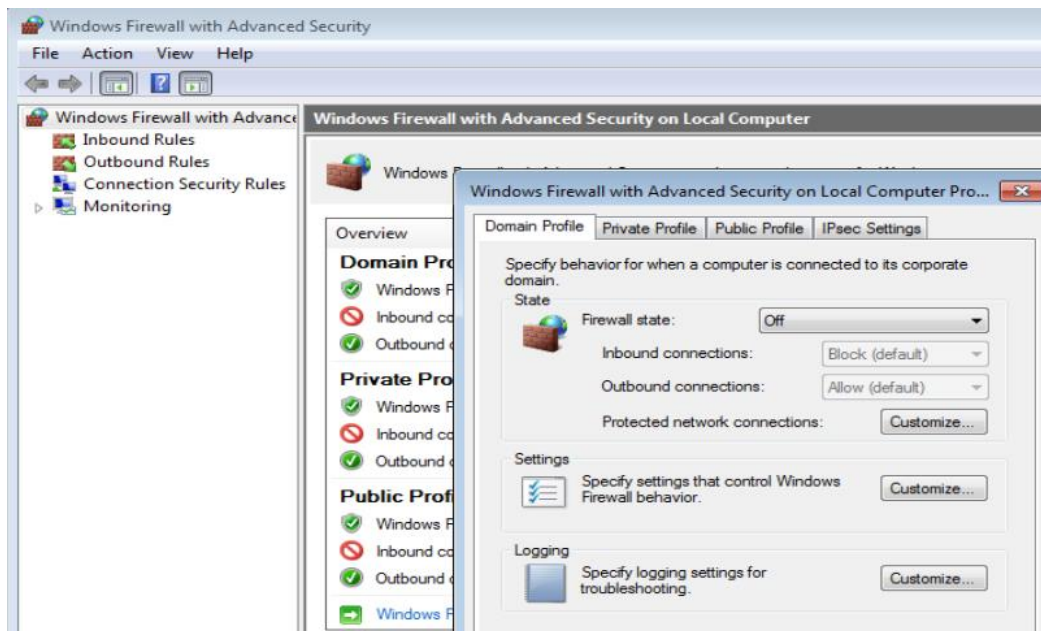
Desactivamos las notificaciones.



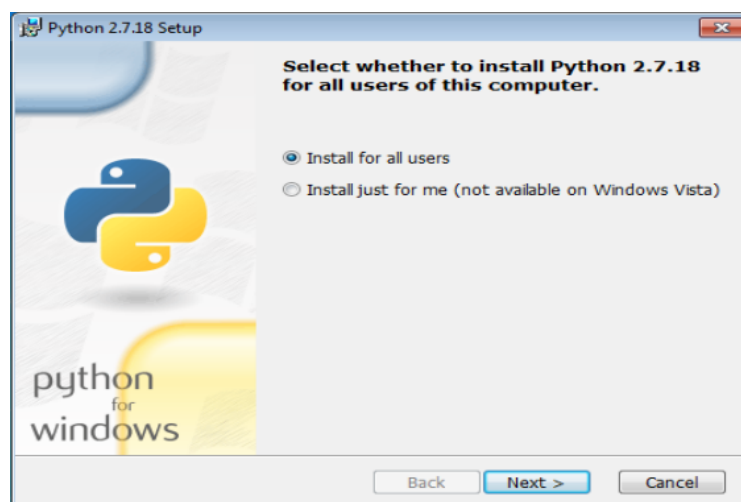
Desactivamos las actualizaciones de windows (Never check for updates) y desactivamos “allow all users to install updates on this computer”.



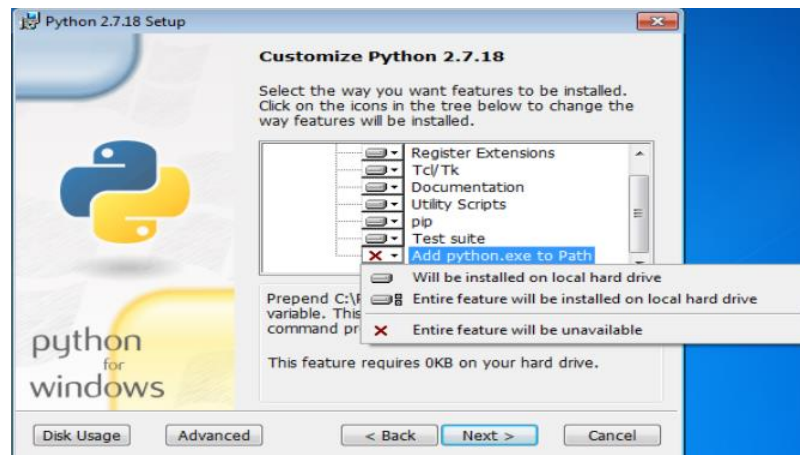
Desactivamos el firewall en los tres perfiles (Domain, Private y Public Profile):



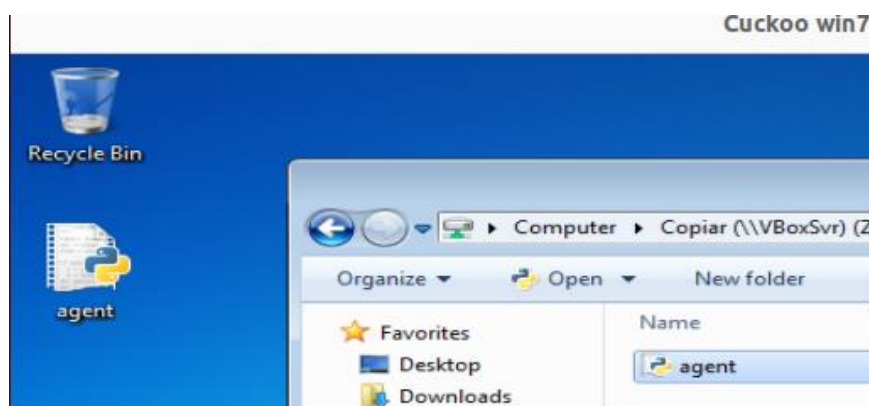
Descargamos python 2.7.18 para todos los usuarios:



Activamos “Add python.exe to path” (will be installed on local hard drive):



Una vez que se haya instalado el python, nos compartimos desde nuestra maquina cuckoo el agent.py (usaremos las carpetas compartidas de virtualbox).



Abrimos CMD e instalamos Pillow:

```
pip install pillow
```



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

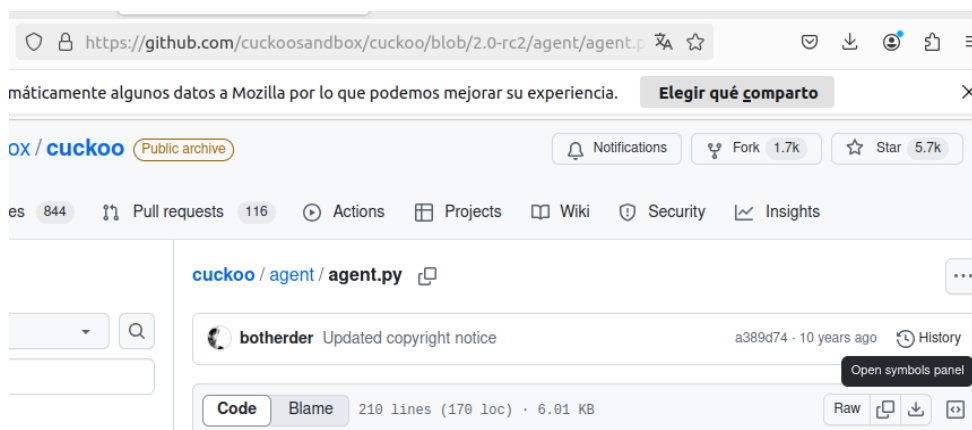
C:\Users\Win7>pip install pillow
DEPRECATION: Python 2.7 will reach the end of its life on January 1st, 2020. Please upgrade your Python as Python 2.7 won't be maintained after that date. A future version of pip will drop support for Python 2.7. More details about Python 2 support in pip, can be found at https://pip.pypa.io/en/latest/development/#python-2-support
Collecting pillow
  Downloading https://files.pythonhosted.org/packages/5f/cd/70086da48312e86d4be6dcf04d338e61f73032d6e1759aac/Pillow-6.2.2-cp27-cp27m-win32.whl (1.6MB)
    1.6MB 2.2MB/s
Installing collected packages: pillow
Successfully installed pillow-6.2.2
WARNING: You are using pip version 19.2.3, however version 20.3.4 is available. You should consider upgrading via the 'python -m pip install --upgrade pip' command.
C:\Users\Win7>_
```

Configuración de cuckoo:

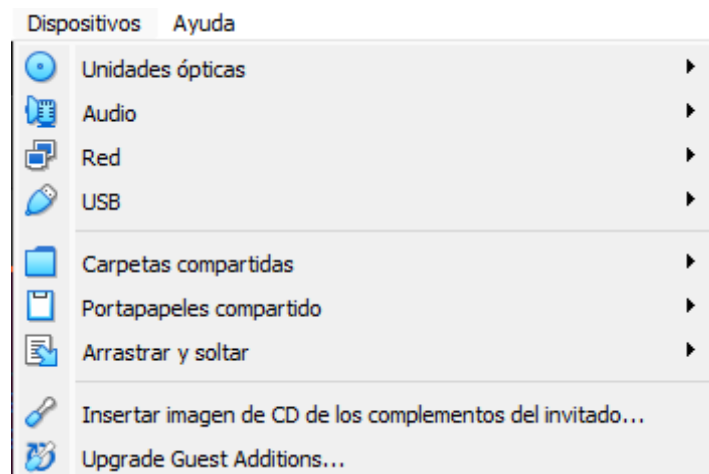
Accedemos a la carpeta agent:

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~$ cd .cuckoo/
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.cuckoo$ ls
agent analyzer conf cuckoo.db distributed elasticsearch
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.cuckoo$ cd agent/
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.cuckoo/agent$ ls
agent.py agent.sh
(cuckoo-env) cuckoo@pablo-VirtualBox:~/.cuckoo/agent$
```

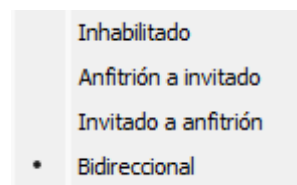
El archivo agent.py corresponde si vamos a utilizar una máquina con Ubuntu y vamos a virtualizar una máquina Windows, en este caso este agente no nos funciona, deberemos usar el siguiente cogido de github ya que si virtualizamos Ubuntu o Windows en la misma o diferente máquina no funciona.



Una vez instalado el archivo accedemos al windows7 y copiamos y pegamos el archivo anterior descargado, para ello, cambiamos la función de arrastrar y soltar y portapapeles compartido en “Dispositivos” en virtualbox y seleccionamos la opción “bidireccional”:



En ambos seleccionamos la opción “bidireccional”:



Una vez hecho esto copiamos y pegamos el archivo:IMAGEN

Accedemos de nuevo a Ubuntu y comenzamos a configurar cuckoo.

Accedemos a la carpeta conf y accedemos a cuckoo.conf:

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~/cuckoo/agent$ cd ..
(cuckoo-env) cuckoo@pablo-VirtualBox:~/cuckoo$ cd conf
(cuckoo-env) cuckoo@pablo-VirtualBox:~/cuckoo/conf$
```

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~/cuckoo/conf$ nano cuckoo.conf
```

Quitamos las “version check”, las “ignore vulnerabilities”, “machinery” y la ip de la máquina del ubuntu:

```
GNU nano 6.2 cuckoo.conf *
[cuckoo]
# Enable or disable startup version check. When enabled, Cuckoo will connect
# to a remote location to verify whether the running version is the latest
# one available.
version_check = no

# Cuckoo will stop at startup if the version check reports vulnerabilities in
# one of Cuckoo's dependencies. This setting ignores the vulnerabilities
# and starts anyway
ignore_vulnerabilities = yes
```

En “machinery” cambiamos “virtualbox” a “physical” porque la virtualización va a ser por separado y por ello cuenta como máquina física:



```
# Specify the name of the machinery module to use, this module will
# define the interaction between Cuckoo and your virtualization software
# of choice.
machinery = physical
```

Cambiamos la ip que tenemos predeterminada por la de nuestra máquina Ubuntu:

```
[resultserver]
# The Result Server is used to receive
# produced by the analyzer.
# Specify the IP address of the host. T
# to contact the host through such addr
# NOTE: if you set resultserver IP to 0
# `resultserver_ip` for all your virtua
ip = 192.168.56.101
```

EXTRA:

Si el tamaño de nuestro malware pesa mucho cambiamos el volumen máximo en el siguiente apartado:

```
# Maximum size of uploaded files from VM (screenshots, dropped files, log).
# The value is expressed in bytes, by default 128 MB.
upload_max_size = 134217728
```

Para lo siguiente deberemos acceder a nuestro Windows 7 y con ipconfig saber la ip de dicha máquina, el nombre de usuario con whoami y el nombre de nuestro adaptador (Accedemos a redes en panel de control, aunque si está conectado por cable no debería importar)

Una vez sabido lo anterior, accedemos a physical.conf en Ubuntu:

```
(cuckoo-env) cuckoo@pablo-VirtualBox:~/cuckoo/conf$ nano physical.conf
```

Cambiamos el nombre de usuario y la contraseña de dicho usuario:

```
# Credentials to access the machine
user = win7
password = 12345
```

(si no tenemos una contraseña puesta deberemos ponerla ya que al no tener contraseña marcaría un error)

En el siguiente paso ponemos el nombre de nuestro usuario de la máquina física que aparece en la línea de comandos:



```
[physical1]
# Specify the label name of the current
# physical machine configuration.
label = win7-pc
```

```
C:\Users\Win7>whoami
win7-pc\win7
```

Cambiamos la dirección ip y ponemos la de nuestra máquina windows:

```
# Specify the IP address
# is valid and that the
# will fail.
ip = 192.168.56.1
```

A continuación editamos el archivo auxiliary.conf

```
GNU nano 6.2
[sniffer]
# Enable or disable the use of
enabled = no

# Specify the path to your local
# path is correct.
tcpdump = no
```

5.3. Script automatización cuckoo

```
#!/usr/bin/env python3
```

```
"""
```

Script para automatizar el envío de archivos sospechosos desde Autopsy a Cuckoo usando carpetas compartidas.

Corre desde el sistema operativo principal.

```
"""
```

```
import os
import shutil
import requests
import time
```



```
import json
import re
import traceback
from datetime import datetime

# === CONFIGURACIÓN ===

AUTOPSY_SHARED_DIR = "/mnt/autopsy_shared/"
CUCKOO_SHARED_SAMPLE_DIR = "/mnt/cuckoo_samples/"
CUCKOO_API_BASE = "http://192.168.56.101:8090"
CUCKOO_API_TOKEN = "QQAPvaP7jP5HnnfP7CopQQ"

PROCESSED_LOG = "processed_files.txt"
REPORTS_DIR = "./cuckoo_reports/"
SUMMARIES_DIR = "./cuckoo_summaries/"
PENDIENTES_FILE = "tareas_pendientes.json"

SUSPICIOUS_PATTERN = re.compile(r'\\.(exe|docx?|js|bat|vbs)$', re.IGNORECASE)

os.makedirs(REPORTS_DIR, exist_ok=True)
os.makedirs(SUMMARIES_DIR, exist_ok=True)

HEADERS = {
    "Authorization": f"Bearer {CUCKOO_API_TOKEN}"
}

# === FUNCIONES PARA TAREAS PENDIENTES ===

def cargar_pendientes():
    if os.path.exists(PENDIENTES_FILE):
        with open(PENDIENTES_FILE, "r") as f:
            return json.load(f)
    return {}

def guardar_pendientes(pendientes):
    with open(PENDIENTES_FILE, "w") as f:
        json.dump(pendientes, f, indent=4)

def agregar_pendiente(archivo_path, task_id):
    pendientes = cargar_pendientes()
    pendientes[archivo_path] = task_id
    guardar_pendientes(pendientes)

def eliminar_pendiente(archivo_path):
    pendientes = cargar_pendientes()
    if archivo_path in pendientes:
```



```
del pendientes[archivo_path]
guardar_pendientes(pendientes)

def listar_pendientes():
    return cargar_pendientes()

# === FUNCIONES DE PROCESO ===

def validar_configuracion():
    if not os.path.exists(AUTOPSY_SHARED_DIR):
        raise ValueError(f"No se encuentra la carpeta compartida de Autopsy:
{AUTOPSY_SHARED_DIR}")
    if not os.path.exists(CUCKOO_SHARED_SAMPLE_DIR):
        raise ValueError(f"No se encuentra la carpeta compartida con Cuckoo:
{CUCKOO_SHARED_SAMPLE_DIR}")

def cargar_procesados():
    if os.path.exists(PROCESSED_LOG):
        with open(PROCESSED_LOG, "r") as f:
            return set(f.read().splitlines())
    return set()

def guardar_procesados(processed):
    with open(PROCESSED_LOG, "w") as f:
        for p in processed:
            f.write(p + "\n")

def buscar_archivos_sospechosos():
    archivos = []
    for root, _, files in os.walk(AUTOPSY_SHARED_DIR):
        for f in files:
            if SUSPICIOUS_PATTERN.search(f):
                full_path = os.path.join(root, f)
                archivos.append(full_path)
    return archivos

def copiar_a_cuckoo(local_path):
    nombre_archivo = os.path.basename(local_path)
    destino = os.path.join(CUCKOO_SHARED_SAMPLE_DIR, nombre_archivo)
    shutil.copy2(local_path, destino)
    return destino

def crear_tarea_en_cuckoo(filepath):
    with open(filepath, "rb") as f:
        files = {"file": (os.path.basename(filepath), f)}
```



```
response = requests.post(f"{CUCKOO_API_BASE}/tasks/create/file", files=files,
headers=HEADERS)
if response.status_code == 200:
    return response.json().get("task_id")
else:
    raise Exception(f"Error al crear tarea: {response.status_code} {response.text}")

def esperar_resultado(task_id):
    estado = "pending"
    while estado not in ["reported", "completed", "failed"]:
        time.sleep(10)
        r = requests.get(f"{CUCKOO_API_BASE}/tasks/view/{task_id}",
headers=HEADERS)
        if r.status_code == 200:
            estado = r.json().get("task", {}).get("status", "")
            print(f"[~] Estado actual: {estado}")
        else:
            raise Exception("Error al consultar estado de la tarea")
    return estado == "reported"

def limpiar_nombre_archivo(nombre):
    return re.sub(r'^a-zA-Z0-9_\-\.\|', '_', nombre)

def descargar_reporte(task_id, archivo_nombre):
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    archivo_nombre = limpiar_nombre_archivo(archivo_nombre)
    r = requests.get(f"{CUCKOO_API_BASE}/tasks/report/{task_id}", headers=HEADERS)
    if r.status_code == 200:
        json_path = os.path.join(REPORTS_DIR,
f"{archivo_nombre}_{timestamp}_report.json")
        with open(json_path, "w") as f:
            f.write(r.text)
        return json.loads(r.text), json_path
    else:
        raise Exception("Error al descargar el reporte")

def generar_resumen(json_data, archivo_nombre):
    timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
    archivo_nombre = limpiar_nombre_archivo(archivo_nombre)
    score = json_data.get("info", {}).get("score", "N/A")
    nombre_archivo = json_data.get("target", {}).get("file", {}).get("name", "desconocido")
    procesos = json_data.get("behavior", {}).get("processes", [])
    firmas = json_data.get("signatures", [])

    nombres_procesos = [p.get("process_name", "N/A") for p in procesos]
    descripciones_firmas = [s.get("description", "") for s in firmas]
```



```
resumen = f"" == RESUMEN DEL ANÁLISIS ==

Archivo: {nombre_archivo}
Score (0-10): {score}

Procesos detectados:
- "" + "\n- ".join(nombres_procesos if nombres_procesos else ["No hay procesos"])

if descripciones_firmas:
    resumen += "\n\nComportamientos sospechosos detectados:\n- " + "\n- ".join(descripciones_firmas)
else:
    resumen += "\n\nNo se detectaron firmas maliciosas."

resumen_path = os.path.join(SUMMARIES_DIR,
f"{archivo_nombre}_{timestamp}_summary.txt")
with open(resumen_path, "w") as f:
    f.write(resumen)
return resumen_path

# == EJECUCIÓN PRINCIPAL ==

def main():
    print("[*] Iniciando análisis de archivos de Autopsy...")
    validar_configuracion()
    procesados = cargar_procesados()
    sospechosos = buscar_archivos_sospechosos()

    pendientes = listar_pendientes()
    for archivo_path, task_id in pendientes.items():
        print(f"[?] Recuperando análisis pendiente: {archivo_path}")
        try:
            if esperar_resultado(task_id):
                json_data, _ = descargar_reporte(task_id,
os.path.basename(archivo_path))
                resumen_path = generar_resumen(json_data,
os.path.basename(archivo_path))
                print(f"[✓] Análisis completado (pendiente). Resumen en:
{resumen_path}")
                procesados.add(archivo_path)
                eliminar_pendiente(archivo_path)
            except Exception as e:
                print(f"[X] Error procesando pendiente: {e}")

    nuevos = [f for f in sospechosos if f not in procesados]
```



```
if not nuevos:
    print("[✓] No hay archivos nuevos para analizar.")
    return

for archivo in nuevos:
    nombre = os.path.basename(archivo)
    try:
        print(f"\n[+] Analizando: {nombre}")
        local_copy = copiar_a_cuckoo(archivo)
        task_id = crear_tarea_en_cuckoo(local_copy)
        agregar_pendiente(archivo, task_id)
        print(f"[→] Tarea creada en Cuckoo (ID {task_id})")

        if esperar_resultado(task_id):
            json_data, _ = descargar_reporte(task_id, nombre)
            resumen_path = generar_resumen(json_data, nombre)
            print(f"[✓] Análisis completo. Resumen en: {resumen_path}")
        else:
            print("[!] El análisis no se completó correctamente.")

        procesados.add(archivo)

    except Exception as e:
        print(f"[X] Error procesando {nombre}: {e}")
        print(traceback.format_exc())

guardar_procesados(procesados)
print("\n[✓] Análisis finalizado.")

if __name__ == "__main__":
    main()
```