Universidad Zaragoza

Trabajo Fin de Grado

Servicio avanzado de simulación de ciberataques y respuesta con MITRE Caldera

Advanced cyberattack simulation and response service with MITRE Caldera

Autor

Garikoitz Ramón Arellano Zubía

Directores

Ibai Marcos Cincunegui

Ricardo Julio Rodríguez Fernández

Grado en Ingeniería Informática

ESCUELA DE INGENIERÍA Y ARQUITECTURA
Julio 2025

AGRADECIMIENTOS

A mis padres por tener tanta paciencia conmigo y ayudarme a no darme por vencido.

A Ibai por ayudarme con el proyecto y las prácticas en la empresa.

A Ricardo por ayudarme con la memoria y la presentación.

A Amaya por animarme a seguir estudiando cuando estaba en la FP Superior.

RESUMEN

En la actualidad, la inteligencia artificial es cada vez más avanzada y usada por los cibercriminales para realizar ataques a los diferentes entornos. Es por ello que las auditorías de seguridad son una tarea esencial para conocer el estado de protección de los sistemas y aplicar las correspondientes correcciones. No obstante, las tareas de auditoría son complejas, costosas y lentas, lo que dificulta su realización. Este trabajo tiene como objetivo crear un sistema para facilitar la ejecución y seguimiento de auditorías de seguridad que combine tareas automáticas y manuales, reduciendo la carga del personal técnico. Para conseguir este objetivo, se ha optado por el desarrollo de un sistema de simulación de ataques en tiempo real basado en MITRE Caldera, que genere un informe tanto técnico como ejecutivo para proporcionar una visión del estado de seguridad del sistema. Además, se han integrado procesos de planificación, ejecución y respuesta, lo que permite detectar vulnerabilidades y establecer un seguimiento de las mismas.

Palabras clave: Cibercriminales, inteligencia artificial, simulación de ataques, MITRE Caldera

ABSTRACT

Currently, artificial intelligence is increasingly advanced and is used by cybercriminals to carry out attacks on different environments. Therefore, security audits are essential for determining the protection status of systems and applying the corresponding corrections. However, audit tasks are complex, costly, and time-consuming, making them difficult to perform. This work aims to create a system to facilitate the execution and monitoring of security audits that combines automated and manual tasks, reducing the burden on technical staff. To achieve this goal, we have developed a real-time attack simulation system based on MITRE Caldera, which generates both technical and executive reports to provide an overview of the system's security status. Additionally, planning, execution, and response processes have been integrated, allowing for the detection and monitoring of vulnerabilities.

Key words: Artificial intelligence, cybersecurity, attack simulation, MITRE Caldera

Lista de figuras

Lista de tablas

Índice

Capítulo 1

Introducción

1.1. Motivación

Actualmente, existe una gran variedad de herramientas que simplifican la realización de ataques a entornos. Con la llegada de la inteligencia artificial (IA), estas herramientas son cada vez más autónomas y potentes [4]. Por ejemplo, se ha registrado un aumento del 38 % en los ciberataques desde 2022. Este significativo aumento se relaciona directamente con el uso y la evolución de la IA para crear malware más complejo y usar nuevas técnicas de phishing más sofisticadas con las que pueden engañar al usuario [5].

Este nuevo entorno requiere que los equipos de Seguridad Operacional (SOC) se adapten, usando enfoques más eficientes y proactivos para defender sus infraestructuras. Una de las tareas de ciberseguridad, como son las auditorías, se ha convertido en una herramienta esencial para identificar y resolver vulnerabilidades y para garantizar el cumplimiento normativo [6].

Sin embargo, la realización de auditorías periódicas es un proceso laborioso que puede consumir una gran cantidad de recursos. Aquí es donde herramientas como MITRE Caldera [7] facilitan el trabajo del equipo SOC, ya que permiten la emulación automática de ataques predefinidos e incluso de ataques diseñados por el propio SOC, aumentando la eficiencia y optimizando el tiempo de los analistas.

1.2. Objetivos

Este trabajo tiene como objetivo principal fortalecer la resiliencia del SOC ante un panorama de amenazas en constante evolución mediante la implementación de un servicio avanzado de simulación de ciberataques. La finalidad de este servicio es facilitar la tarea de realizar auditorías a entornos, consumiendo la menor cantidad de recursos posible y optimizando los procesos del equipo SOC.

Para lograr el objetivo, se han definido una serie de hitos a cumplir. En primer lugar, investigar las distintas técnicas utilizadas por atacantes en entornos reales. El siguiente paso es elaborar un entorno virtual controlado para la simulación de ataques. A continuación, se debe realizar el estudio, análisis y despliegue de la herramienta de simulación, con su correspondiente documentación. Después, hay que realizar la simulación de ataques en el entorno desplegado en los pasos anteriores. La simulación de ataques permite asentar los conocimientos anteriormente estudiados. El siguiente paso es diseñar los procesos de simulación de ataques, de notificación y de respuesta. Tras el diseño de los procesos, es necesaria su implementación. Por ello, los dos siguientes pasos son la integración de la generación automática de informes técnicos y ejecutivos, y la integración de un sistema de avisos automáticos para notificar al equipo técnico.

1.3. Estructura de la memoria

Este documento se encuentra dividido en seis capítulos y un anexo. El Capítulo 2 expone el estado del arte de la simulación de adversarios, así como conceptos esenciales como la *Cyber Kill Chain* [2]. El Capítulo 3 se centra en MITRE, dividiéndose en dos secciones: una dedicada a la matriz MITRE ATT&CK y otra a la herramienta principal utilizada en este proyecto, MITRE Caldera. El Capítulo 4 describe las tecnologías utilizadas y la configuración del entorno del proyecto. En el Capítulo 5 se detalla la base sobre la que se ha construido este trabajo, incluyendo la fase de diseño del sistema global, los distintos flujos y la implementación. El Capítulo 6 presenta un caso de uso, en el que se programa una operación en Caldera y, tras su ejecución, el servicio desarrollado en este TFG se encarga de generar el informe y los tickets correspondientes. Finalmente, el Capítulo 7 recoge las conclusiones y posibles trabajos futuros del proyecto.

Al final del documento se incluye el Anexo A, en el cual se encuentran, en primer lugar, el diagrama de Gantt con las horas empleadas en el TFG y la distribución de horas dedicadas a cada parte del trabajo. Después se encuentran las tablas de categorización y de riesgos, diseñadas para guiar al equipo técnico, y un ejemplo de la traza del registro generado en el Capítulo 6.

Capítulo 2

Estado del arte

En este capítulo, se describe en qué consiste la técnica de simulación de adversarios utilizada en este trabajo. Después, para una mayor comprensión, se explica el ciclo de vida de un ciberataque utilizando como referencia la *Cyber Kill Chain*.

Como explicaron A. Applebaum y D. Miller en la conferencia de BlackHat de 2017 en su exposición acerca de MITRE Caldera [1], la emulación de adversarios es una técnica que permite comprobar de una forma más realista nuestro entorno mediante la incorporación de un "enemigo" más realista.

La simulación de adversarios es una técnica que lleva pocos años aplicándose en los entornos empresariales. No obstante, se ha demostrado que este tipo de técnicas son muy efectivas, puesto que permiten comprobar el estado del entorno frente a ataques que ya se han realizado en otros entornos [8]. La Figura 2.1 muestra un breve esquema que explica de forma gráfica los modelos de emulación de adversarios.

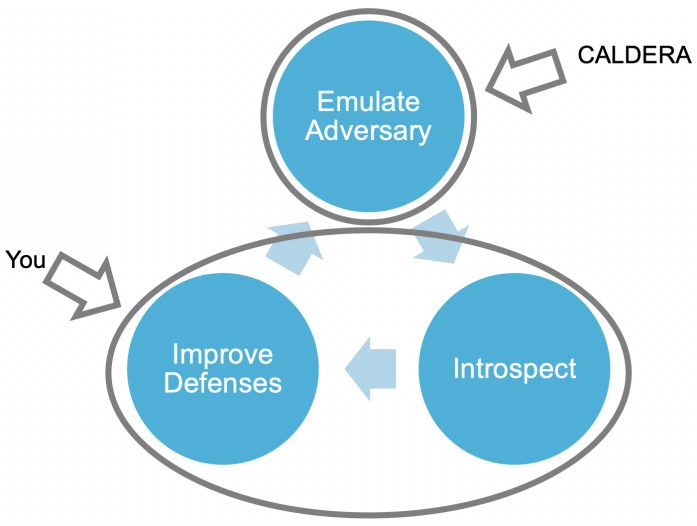

Figura 2.1: Modelo de emulación de adversarios. Fuente: [1]

Para comprender correctamente en qué consiste la simulación de ataques, primero hay que tener claro qué es un ciberataque y qué partes tiene.

Un ciberataque es cualquier intento deliberado de robar, exponer, alterar, deshabilitar o destruir datos, aplicaciones u otros activos mediante el acceso no autorizado a una red, sistema informático o dispositivo digital [9]. Un ciberataque tiene un ciclo de vida, también conocido como *Cyber Kill Chain*. Este ciclo describe las fases de un ciberataque dirigido, desglosando cada etapa para que los defensores puedan identificarlo y detenerlo [2]. *Cyber Kill Chain* describe estas siete etapas, que se pueden observar en la Figura 2.2.

La primera fase es el reconocimiento, que se basa en la recopilación de información por parte de los atacantes. Gracias a esta fase, el atacante obtiene información que puede utilizar para encontrar brechas de seguridad o puntos de acceso por los que entrar al sistema de la víctima. La segunda fase es la preparación, donde el atacante elige una o varias formas de intentar entrar al sistema de la víctima, utilizando la información obtenida de la fase anterior. Estas formas de intrusión en el sistema se denominan *vectores de ataque*. La tercera fase es la distribución, donde una vez que el atacante ya está dentro del sistema, puede distribuir los programas dañinos que tenga preparados. Estos programas pueden ser programados para ejecutarse en una fecha determinada o de forma inmediata o como respuesta a una acción específica, como puede ser una petición del atacante. La cuarta fase es la denominada explotación, donde una vez que el atacante ya ha distribuido sus programas dañinos, comienza con la explotación del sistema. La quinta fase se centra en la instalación de una puerta trasera en el sistema para tener acceso siempre que quiera. De esta manera, el atacante podrá acceder a los sistemas sin arriesgarse a ser detectado, comenzando un nuevo ataque. La sexta fase, comando y control, consiste en que el atacante tome el control del sistema, con el objetivo de instalar más programas dañinos o filtrar datos en un futuro. La séptima y última fase son acciones sobre los objetivos, donde el atacante comienza a filtrar o cifrar los datos de la víctima o incluso hacer que su red no esté disponible. Las acciones dependen de los objetivos del atacante.

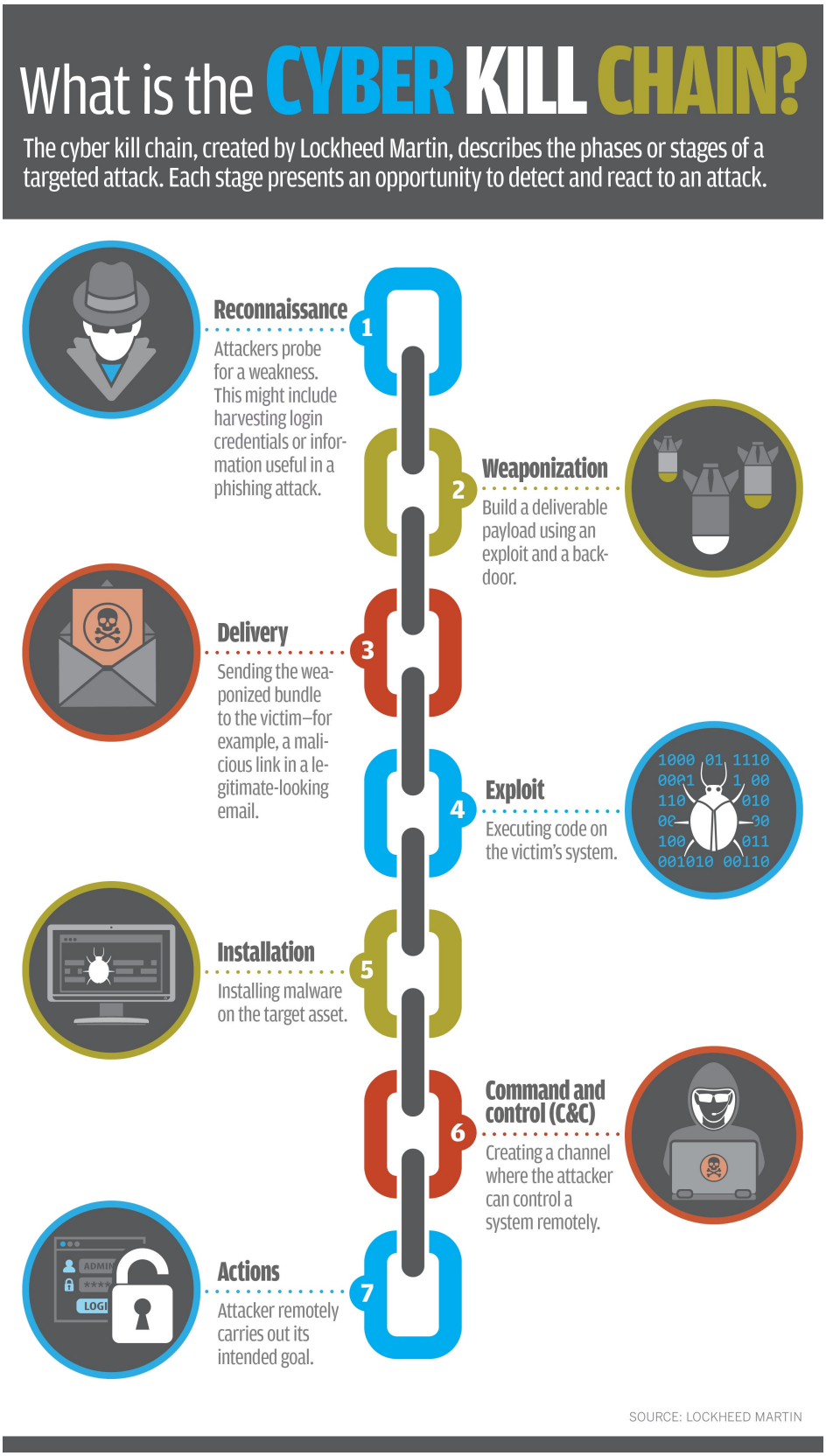

Figura 2.2: *Cyber Kill Chain* de Lockheed Martin. Fuente: [2]

Capítulo 3

Contexto

Este capítulo se centra en introducir la herramienta MITRE Caldera. Para ello, es necesario enmarcarla dentro del ecosistema MITRE, que es una organización estadounidense sin ánimo de lucro que colabora con agencias gubernamentales en sectores relacionados con la ciberseguridad, fundada en 1958. Primero, se ofrece una explicación y un análisis del marco MITRE ATT&CK, ya que es una herramienta clave para la simulación de ataques y constituye la base de MITRE Caldera. Por último, se explica qué es MITRE Caldera, cómo funciona y cuáles son sus componentes.

3.1. MITRE ATT&CK

MITRE Corporation comenzó a desarrollar en 2013 el marco MITRE ATT&CK y finalmente lo publicó en mayo de 2015. El objetivo de este marco es fortalecer los pasos que se toman después de que una organización se ha visto comprometida [10]. Este marco se encuentra en constante expansión, lo que lo convierte en una gran herramienta para averiguar vulnerabilidades en una arquitectura y determinar cuáles corregir primero basándose en el riesgo de cada una.

MITRE ATT&CK organiza el conocimiento sobre tácticas y técnicas empleadas por los atacantes en tres matrices principales. La primera es *Enterprise ATT&CK*, enfocada en tácticas y técnicas aplicables a sistemas operativos como Linux, Windows y macOS. La segunda es *PRE-ATT&CK*, que recoge las tácticas y técnicas utilizadas en las fases previas a una intrusión, es decir, durante la planificación y el reconocimiento de objetivos. Finalmente, la matriz *Mobile ATT&CK* se centra en aquellas tácticas y técnicas dirigidas específicamente a dispositivos móviles.

Cada matriz ATT&CK genera un informe dividido en columnas, donde cada una describe tácticas que pretende lograr el atacante. Por otro lado, están las técnicas, que son los métodos utilizados para conseguir las tácticas. Para este proyecto, la matriz de interés es la *Enterprise ATT&CK*, que se divide en 14 tácticas [11].

La primera táctica es *reconocimiento*, la cual se encarga de recopilar información que pueda ser útil para poner en marcha acciones contra la víctima. En segundo lugar está el *desarrollo de recursos*, cuyo objetivo es desarrollar, comprar o robar recursos que puedan ser útiles para ejecutar otras tácticas y cumplir con los objetivos del atacante. El siguiente es el *acceso inicial* para infiltrarse en las redes corporativas. Una vez infiltrado, la siguiente táctica es *ejecución* con el fin de ejecutar código dañino en la red corporativa. Sin embargo, es necesario mantener esa posición dentro de los sistemas y para ello se utilizan tácticas de *persistencia*. Posteriormente viene el *escalamiento de privilegios*, que consiste en intentar conseguir un mayor nivel de permisos en un sistema empresarial. La siguiente táctica es la *evasión de defensa* centrada en desinstalar las soluciones de seguridad instaladas o en la ofuscación y cifrado de ficheros ejecutables o scripts. La obtención o robo de nombres de usuario y contraseñas se encuentra englobada en la táctica *acceso a credenciales*. El *descubrimiento* tiene como objetivo conocer en profundidad el sistema y la red corporativa, lo cual permite posteriormente realizar un *movimiento lateral* para explorar toda la red corporativa y encontrar los objetivos. Una vez encontrados los objetivos, el atacante puede realizar una *recopilación* para obtener datos que sean de interés. Todas las tácticas utilizadas para intentar comunicarse y controlar los sistemas comprometidos en la infraestructura se encuentran dentro de *comando y control*. Finalmente, el atacante puede realizar una *exfiltración* para robar datos de los sistemas corporativos y/o un *impacto* para paralizar la operatividad de la empresa.

La Figura 3.1 muestra un ejemplo de una de las matrices de MITRE. En este caso, se muestra la matriz *Enterprise ATT&CK*, puesto que es la matriz que se va a utilizar para el desarrollo de este proyecto, al ser la especializada en sistemas Windows, Linux y macOS.

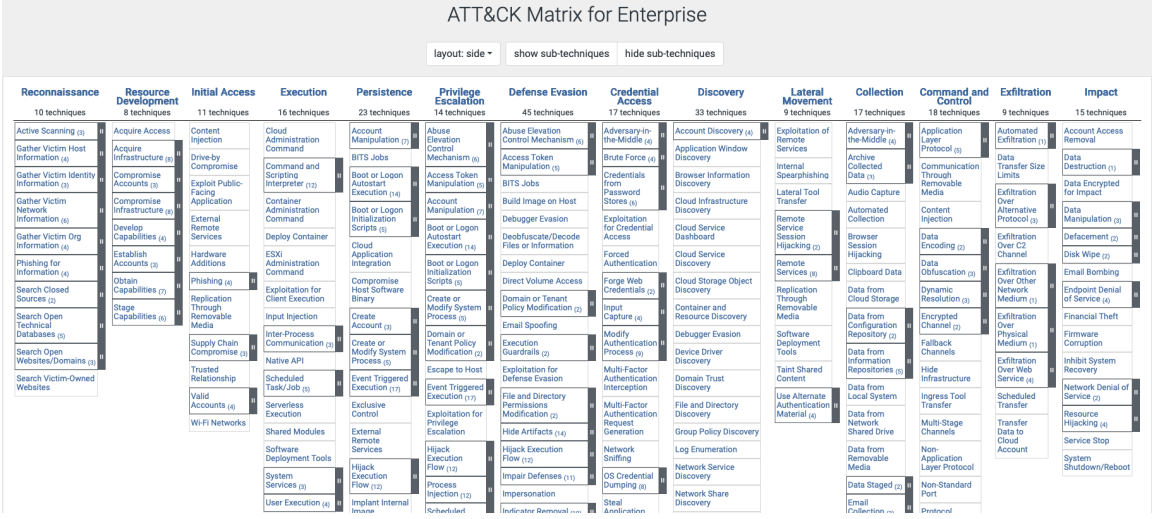

Figura 3.1: Matriz Enterprise MITRE ATT&CK. Fuente [3]

3.2. MITRE Caldera

El proyecto MITRE Caldera es una de las herramientas seleccionadas para cumplir parte de los objetivos de este trabajo debido a que es una plataforma de simulación de brechas y ataques de código abierto creada por MITRE Corporation, que permite a las organizaciones emular de manera automática las tácticas, técnicas y procedimientos (TTP) empleados por actores de amenazas persistentes avanzadas del mundo real. La plataforma está diseñada con un enfoque modular, lo que facilita su personalización para adaptarse a los requisitos específicos de una organización, según lo que se desee ejecutar o simular.

MITRE Caldera viene preconfigurado con dos usuarios, el primero es el usuario *red*, el cual proporciona la capacidad de poder ejecutar distintos tipos de TTPs extraídos directamente desde MITRE, de forma que se puede comprobar periódicamente si los sistemas de protección están realmente funcionando o tienen algún tipo de brecha. Por otro lado, existe el usuario *blue*, que ayuda al equipo de *Blue Team* (grupo encargado de la defensa de la infraestructura de una organización) a estudiar el comportamiento de los distintos tipos de TTPs para crear posteriormente alertas en el *SIEM* (*Security Information and Event Management*; en español, sistema de gestión de información y eventos de seguridad. Es una solución que centraliza, correlaciona y analiza los registros de eventos generados por los sistemas de una organización, con el objetivo de detectar amenazas y generar alertas en tiempo real) y su correspondiente respuesta desde el *EDR* (*Endpoint Detection and Response*; en español, detección y respuesta en punto final. Es una solución diseñada para monitorizar, detectar y responder a amenazas en dispositivos como ordenadores, servidores o portátiles, permitiendo ejecutar acciones de forma automática en tiempo real). En la Figura 3.2 se muestra la arquitectura de MITRE Caldera.

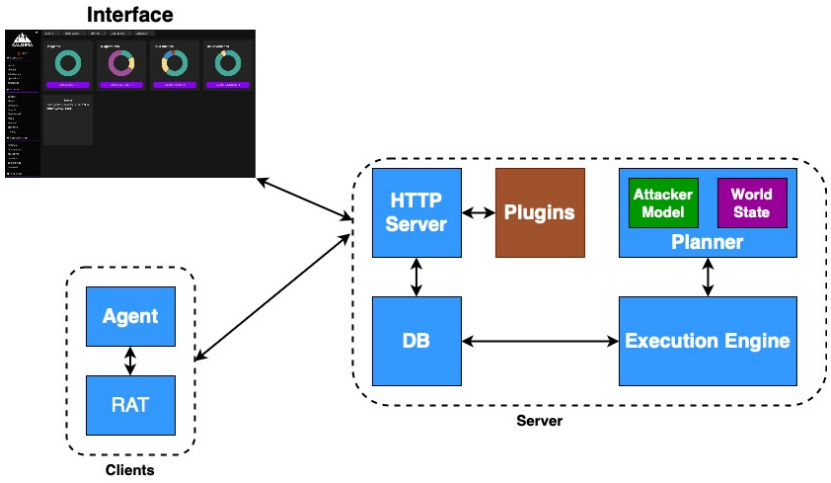

Figura 3.2: Arquitectura MITRE Caldera

3.2.1. Agentes

Un *Agente* es un programa escrito en un lenguaje de programación que ejecuta instrucciones de un adversario en los sistemas comprometidos. Normalmente, estos equipos se comunican mediante protocolos como HTTP. Además, el agente es el encargado de generar el *beaconing* (comportamiento de red en el que un dispositivo o sistema envía señales o mensajes de forma periódica a otros sistemas) al *C2* (*Command and Control*; en español, comando y control. En el contexto de MITRE Caldera es un mecanismo utilizado para coordinar los agentes desplegados desde un servidor central) de forma regular para preguntar si hay nuevas instrucciones que ejecutar. Los agentes instalados aparecerán en la interfaz gráfica del sistema en el apartado *Agents*. Los agentes pueden ser agrupados en grupos personalizados.

3.2.2. Habilidades

Una *habilidad* es un conjunto específico de instrucciones que un agente debe ejecutar en una máquina. En otras palabras, es una implementación específica de una técnica/táctica ATT&CK. Las habilidades pueden incluir comandos, la plataforma/ejecutor en la cual tienen que ejecutarse (por ejemplo, Windows / Powershell), payloads y referencias a módulos de parseo de Caldera.

3.2.3. Adversarios

Un adversario es un grupo de habilidades que representan TTPs disponibles. Los perfiles de adversarios se utilizan cuando se está ejecutando una operación para especificar qué habilidades tienen que ejecutarse.

3.2.4. Operaciones

Las operaciones ejecutan habilidades en grupos de agentes. Como se ha comentado anteriormente, la forma de seleccionar qué habilidades se quieren ejecutar es creando un perfil de adversario.

Para seleccionar el orden en el que se tienen que ejecutar las habilidades, MITRE Caldera dispone del elemento planificador, el cual especifica en qué orden se van a ejecutar las habilidades. Por defecto, Caldera viene con tres planificadores.

El primero es *Atomic*, el cual ejecuta las habilidades en el orden atómico en el que están definidas en el perfil de adversario. El segundo es *Batch*, que ejecuta todas las habilidades del perfil a la vez, y por último está *Buckets*, que ejecuta las habilidades del perfil agrupadas por la táctica ATT&CK.

Cuando una habilidad se está ejecutando en una operación, se genera un enlace (un enlace es la tarea que el agente debe realizar, por ejemplo, "ejecutar comando `whoami`") por cada agente, siempre y cuando se cumplan tres requisitos. Primero, que todos los *enlaces asociados a hechos* (pieza de información generada durante la ejecución de una operación, representa un dato aprendido o adquirido por el sistema mediante una acción de un agente) y los *hechos requeridos* (condiciones que una habilidad debe cumplir para ejecutarse) hayan sido cumplidos, es decir, para generar un enlace la habilidad debe haber obtenido la información requerida por la siguiente o siguientes habilidades dentro del perfil de adversario. También hay que asegurarse de que el agente tiene un ejecutor compatible, es decir, que la habilidad debe ser compatible con el ejecutor del agente. Si la habilidad está configurada para ejecutarse en un sistema Windows, pero el agente solo tiene un ejecutor para Linux, no se generará el enlace. Por último, hay que tener en cuenta que un agente no ejecutará la misma habilidad más de una vez, a menos que la habilidad esté marcada como *repeteable*. Por defecto, esta configuración está deshabilitada para evitar que se realicen acciones redundantes, a menos que sea explícitamente necesario.

Los hechos son piezas de información o datos que el sistema recopila, almacena y utiliza para tomar decisiones durante una operación. Estos hechos representan detalles específicos sobre el entorno, como nombres de usuario, direcciones IP, nombres de archivos, claves de registro, o cualquier otro dato relevante que pueda ser útil para ejecutar habilidades o tomar decisiones tácticas. Además, los comandos de los enlaces pueden ser ofuscados para evitar ser detectados por los antivirus.

Capítulo 4

Entorno de trabajo

Este capítulo tiene como objetivo presentar el entorno de trabajo en el cual se ha llevado a cabo el desarrollo del sistema. Para ello, primero se presentan las principales herramientas utilizadas en el sistema. Después, se procede con la configuración del entorno, donde se explican los requisitos del sistema operativo en el que se van a ejecutar las herramientas, cómo se ha realizado el despliegue de Caldera y cómo se ha preparado el entorno de programación del servicio.

4.1. Tecnologías utilizadas

Esta sección se centra en la explicación de las principales tecnologías utilizadas en este proyecto.

4.1.1. Asyncio y AioHTTP

Tradicionalmente, las solicitudes a una API web que se realizan de forma secuencial suelen producir cuellos de botella debido a que en cada petición hay que esperar a que finalice la anterior, ralentizando toda la ejecución del programa. Para solucionar este problema existen dos alternativas. Por un lado, la programación paralela, utilizando diferentes hilos para realizar el trabajo, y por otro lado, la programación concurrente, la cual nos permite ejecutar varias tareas al mismo tiempo, compartiendo recursos en un solo hilo, optimizando las tareas que implican esperas largas.

En el caso de este proyecto, la programación concurrente encaja mejor, debido a que permite optimizar el tiempo ejecutando las llamadas a la API de forma asíncrona sin necesidad de tener que utilizar múltiples hilos y evitando problemas de sincronización de los datos con exclusión mutua. Además, el uso de la biblioteca *Asyncio* [12] resulta muy cómodo para programadores con experiencia previa con JavaScript, ya que utiliza una sintaxis muy similar para la creación de tareas asíncronas. *Asyncio* permite realizar tareas de forma concurrente, pero para realizar peticiones HTTP de forma asíncrona

hay que hacer uso de la librería AIOHTTP [13].

4.1.2. Confluence y Jira

Confluence [14] es una plataforma en la nube de colaboración y trabajo en equipo, desarrollada por Atlassian. Esta herramienta permite a los usuarios crear, organizar y compartir información en un espacio centralizado. Los usuarios pueden editar en tiempo real y comentar las páginas. Estas páginas se almacenan en espacios de trabajo para una mayor organización. Estos espacios pueden ser públicos y privados, es decir, el usuario es quien determina la visibilidad del espacio. Por otro lado, Confluence dispone de una gran variedad de integraciones con aplicaciones de terceros que permiten la creación de plantillas personalizadas o diagramas UML. Por otra parte, Jira [15] es una herramienta en la nube creada por Atlassian, orientada a la gestión ágil de proyectos, que permite la creación y seguimiento de tickets para una organización óptima del trabajo.

Tanto Confluence como Jira disponen de una API para poder interactuar con sus servicios y crear automatizaciones.

4.1.3. Docker

Docker es una plataforma de software con la que se puede crear, probar e implementar aplicaciones de forma rápida y sencilla. Esta herramienta empaqueta el software en lo denominado *contenedor*, que incluye bibliotecas, herramientas del sistema, código y todo lo necesario para que la aplicación pueda ejecutarse de manera independiente.

La ventaja que proporciona Docker es la abstracción del sistema operativo en el que se ejecuta la aplicación. Debido a la virtualización que realiza, permite al usuario ejecutar la misma aplicación en distintos sistemas operativos sin preocuparse de las dependencias de cada uno.

4.1.4. Proxmox VE

Proxmox VE [16] es una plataforma de código abierto de virtualización que permite a los usuarios crear y gestionar tanto máquinas virtuales como contenedores y almacenamiento definido por software.

Se ha elegido esta herramienta como entorno de virtualización porque es el utilizado en la empresa donde se desarrolla el sistema. No obstante, una de las ventajas que proporciona Proxmox es la facilidad de crear máquinas virtuales, gracias a su interfaz web, la cual es muy intuitiva y completa. Es por ello que, pese a poder utilizar cualquier

otra plataforma de virtualización, el uso de Proxmox ha ayudado en gran medida, agilizando la creación de máquinas virtuales.

4.1.5. Crontab

Crontab es una herramienta de los sistemas basados en Unix con la cual se pueden programar tareas de una forma muy sencilla. Esta herramienta facilita al usuario la ejecución programada de scripts o la ejecución de comandos. Su configuración está basada en un fichero, que normalmente se encuentra localizado en la ruta `/etc/crontab`. Para configurar la ejecución de una tarea, existe una serie de reglas para especificar cuándo se debe ejecutar la tarea [17].

4.2. Configuración del entorno

En la Figura 4.1 se muestra el esquema de la infraestructura final del entorno. La infraestructura está compuesta por la red corporativa de la empresa y las redes de los clientes (en la Figura 4.1 solo se muestra la red de un cliente por simplificación). La red corporativa dispone de un Proxmox (véase la Sección 4.1.4). Este Proxmox aloja dos máquinas virtuales, una para el servidor Caldera y otra para el servicio Python de generación de informes y notificaciones. Como se ha explicado en la Sección 3.2 el agente de MITRE Caldera funciona en Windows, Linux y macOS, es por ello que los equipos de los clientes pueden ser de cualquiera de estos tres tipos de sistemas operativos.

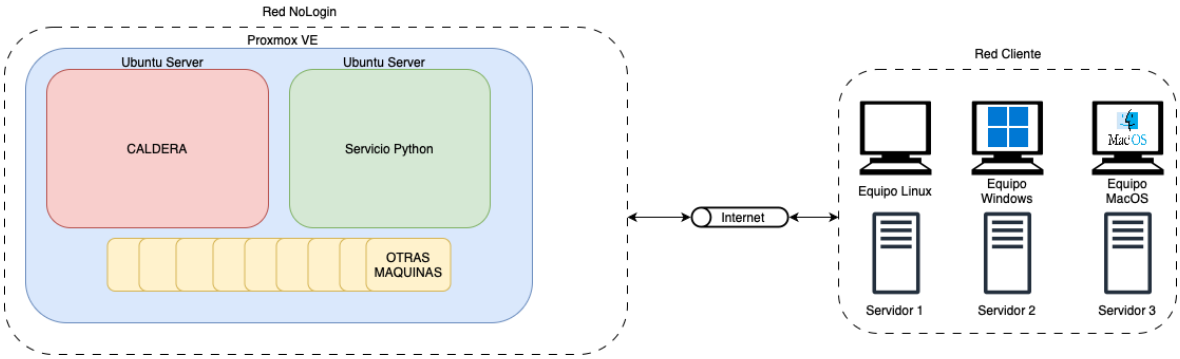

Figura 4.1: Infraestructura del sistema

4.2.1. Sistema Operativo

Para la elección del sistema operativo se tuvieron en cuenta tres requisitos, los cuales se especifican en la Tabla 4.1. El primer requisito es el tipo de sistema operativo que MITRE Caldera especifica que hay que utilizar. El segundo es la experiencia previa de

la empresa, es decir, con qué sistema operativo ha trabajado la empresa. Finalmente, el último está relacionado con el servicio Python desarrollado y explicado posteriormente.

Requisito	**Sistema Operativo**
MITRE Caldera	Linux y macOS
Experiencia previa de la empresa	Ubuntu Server
Servicio Python	Preferiblemente Linux

Tabla 4.1: Requisitos para la elección del sistema operativo

Debido a la experiencia y uso de la empresa, el cumplimiento de los tres requisitos anteriormente mencionados y la estabilidad y comunidad que tiene detrás, se optó por instalar la distribución *Ubuntu Server 24.04.2 LTS* con las características mencionadas en la Tabla 4.2.

Herramienta	**Sistema Operativo**	**Memoria RAM**	**Almacenamiento**	**Núcleos**
MITRE Caldera	Ubuntu Server 24.04.2 LTS	16 GB	64 GB	6
Servicio Python	Ubuntu Server 24.04.2 LTS	8 GB	32 GB	6

Tabla 4.2: Características del servidor principal

4.2.2. Despliegue de Caldera

Para realizar el despliegue de Caldera se hizo uso de su documentación oficial [7], en la cual se explica que existen dos modos de instalación de la herramienta. El primer modo es mediante *Docker Compose*, el cual automatiza y simplifica el despliegue de la herramienta, teniendo que solo ejecutar un único comando. El otro modo es mediante la instalación manual de las bibliotecas de Python, JavaScript y Go.

Tras probar en el laboratorio los dos tipos de instalación, se llegó a la conclusión de que se utilizaría el segundo modo de instalación (instalación manual), debido a que el primer modo tenía problemas activos con Docker [18] que provocaban dificultades a la hora de actualizar la herramienta.

Para seguir el segundo modo de instalación, primero es necesario instalar *Python 3.8* o superior con *pip3*, *NodeJS v16* o superior, *GoLang 1.17* o superior, así como todos los requisitos adicionales.

Una vez desplegado el servidor Caldera, hay que dirigirse al navegador web y entrar en `http://IP_Caldera:8888` y entrar con el usuario *red* o *blue*. La contraseña de ambos usuarios se crea en el momento en que se construye el servidor, por lo que hay que dirigirse a `caldera/conf/local.yml` donde se encuentran las contraseñas de ambos usuarios. Además, se ha creado un servicio en Linux para el despliegue automático de Caldera tras un reinicio del sistema, de esta forma el servicio estará disponible tras un reinicio sin necesidad de que un técnico tenga que intervenir. El código para generar el servicio está disponible en el repositorio GitHub asociado al proyecto (véase la Sección 5.4).

4.2.3. Entorno de programación del servicio

Para la programación del servicio Python se ha optado por el desarrollo de un contenedor *DevContainer* de Docker, disponible en el repositorio GitHub asociado al proyecto (véase la Sección 5.4), para evitar problemas de compatibilidad con el sistema operativo y con dependencias de bibliotecas.

Por último, para llevar un correcto control de versiones se ha utilizado la herramienta GitLab.

Capítulo 5

Análisis, diseño e implementación

En este capítulo se abordan cuestiones esenciales relacionadas con el análisis, diseño e implementación del proyecto. En primer lugar, se expone el análisis realizado para determinar aspectos clave del proyecto, como los requisitos y su alcance. A continuación, se describe el diseño del proyecto, centrándose en sus dos componentes principales. Finalmente, se presenta la implementación, explicando cómo se ha llevado a cabo.

5.1. Análisis

En esta fase de análisis se ha llevado a cabo la identificación de los requisitos que el sistema debe cumplir. Para ello, se ha realizado un análisis del entorno presente en la empresa, teniendo en cuenta los métodos y los recursos tecnológicos, humanos y temporales disponibles. Además, para llevar a cabo la priorización de requisitos, se ha utilizado la metodología MoSCoW [19], la cual ayuda a clasificar los requisitos según su prioridad.

La metodología MoSCoW se divide en cuatro tipos de prioridades según la criticidad del requisito. El primero es M (*Must*), que hace referencia a que el requisito debe estar obligatoriamente. El segundo es S (*Should*), que indica que es un requisito que debería estar implementado. La C (*Could*) es para aquellos requisitos que no son importantes y, por lo tanto, no es obligatorio que se cumplan, pero que sí podrían estar, añadiendo más valor. Finalmente, W (*Would*) sirve para aquellos requisitos que no son importantes ahora mismo y que se pueden implementar en un futuro.

Tras realizar el análisis anteriormente comentado, se han obtenido dos tablas. En cada una de ellas se describen los requisitos que se deben cumplir. Cada requisito está asociado a una prioridad siguiendo la metodología MoSCoW. En la Tabla 5.1 se pueden observar los requisitos funcionales del sistema, que describen lo que el sistema debe hacer, definiendo las funciones y procesos que debe realizar el sistema. En la Tabla 5.2 se encuentran los requisitos no funcionales del sistema, que describen cómo

debe comportarse, qué atributos de calidad debe cumplir y qué restricciones técnicas pueden existir.

Requisito	**Prioridad**
Simulación de ataques basada en TTPs	M
Personalización del perfil de adversario y del flujo de ataque	M
Agrupación de equipos por cliente	M
Generación de informes ejecutivos y técnicos de cada operación	M
Generación de tareas en ITSM	M
Notificación ante fallo del servicio de generación de informes	S
Planificación de operaciones a modo scheduler	S
Personalización de la ejecución de TTPs en clientes mediante una lista de autorización	C

Tabla 5.1: Requisitos funcionales

Requisito	**Prioridad**
Puesta en marcha automática del servicio de simulación de adversarios ante un reinicio del sistema	M
El sistema debe ser escalable y fácil de mantener	M
El sistema debe ser fácil de desplegar en cualquier entorno Linux	M
El sistema debe ser compatible con Jira y Confluence	M

Tabla 5.2: Requisitos no funcionales

5.2. Diseño

El diseño de este proyecto se divide en 2 partes: una parte centrada en el procedimiento que se debe llevar a cabo a la hora de realizar el ataque, y otra parte centrada en la ejecución que se debe llevar a cabo para realizar dicho ataque. Ambas se explican a continuación.

5.2.1. Diseño del flujo de ataque

Como se ha comentado anteriormente, un ataque cibernético se divide en siete pasos (véase la Sección 2). Sin embargo, cada ataque debe diseñarse específicamente en función del objetivo o grupo de objetivos que se quiera alcanzar. Por este motivo, esta sección se centra en definir unos criterios comunes que deben seguirse a la hora de planificar un ataque utilizando MITRE Caldera.

El flujo de ataque no es una secuencia predefinida válida para todos los clientes. Es el equipo técnico quien debe definir qué acciones desea ejecutar, basándose en los

objetivos particulares de cada operación. Para facilitar la elaboración de estos flujos, se han diseñado unas tablas de apoyo que permiten al equipo técnico tomar decisiones informadas sobre cómo proceder.

Específicamente, se han desarrollado dos tablas. Por un lado, se encuentra la tabla de *Categorización de Clientes* (véase la Figura A.2, en el Apéndice A.2), con la cual el equipo puede identificar los activos principales, el tipo de cliente, su nivel de riesgo y las amenazas más comunes. Por otro lado, está la tabla de *Categorización de Riesgos* (véase la Figura A.3, en el Apéndice A.3), que permite definir el nivel de riesgo, las tácticas ATT&CK más comunes, ejemplos en Caldera y la frecuencia con la que debería comprobarse el estado del entorno.

Además, se debe realizar una selección adecuada de los equipos en los que se ejecutarán las operaciones. En base a la experiencia transmitida por la empresa, se ha determinado que lo ideal es seleccionar tres objetivos. Algunos ejemplos podrían ser:

1. Usuario fuera de la infraestructura con acceso a la red pública. Por ejemplo, un usuario que está conectado a la red Wifi.

   Criterio de ataque:

   Fase 1. Descubrimiento de Red

   Fase 2. Acceso inicial

   Fase 3. Movimiento lateral

   Fase 4. Escalada de privilegios

   Fase 5. Exfiltración de datos

2. Usuario dentro de la infraestructura pero con limitación de permisos. Por ejemplo, un trabajador del departamento de Marketing.

   Criterio de ataque:

   Fase 1. Descubrimiento de Red

   Fase 2. Movimiento lateral

   Fase 3. Escalada de privilegios

   Fase 4. Exfiltración de datos

3. Usuario dentro de la infraestructura pero con permisos elevados. Un ejemplo sería el administrador de sistemas de la empresa.

   Criterio de ataque:

Fase 1. Descubrimiento de Red

Fase 2. Persistencia

Fase 3. Impacto

Fase 4. Exfiltración de datos

5.2.2. Diseño de la ejecución del servicio

El servicio Python se ha diseñado para reducir al máximo los recursos humanos necesarios para la ejecución completa del sistema. De esta manera, el servicio se encarga de recolectar las últimas operaciones ejecutadas utilizando la API de Caldera y de generar un informe técnico y ejecutivo por cada una de ellas, además de crear un ticket en Jira por cada acción que se ejecute con éxito. Sin embargo, este planteamiento puede generar ciertos problemas, como la sobrecarga de tickets por acciones triviales. Esto se debe a que, para ejecutar correctamente las operaciones, Caldera necesita realizar acciones orientadas a la obtención de información, como identificar el usuario actual, listar los procesos en ejecución u obtener el contenido del fichero `/etc/shadow`. Es evidente que algunas de estas tareas no deberían generar un ticket ni aparecer en el informe ejecutivo como una acción insegura.

Para resolver este problema, se han estudiado varias opciones. La primera consiste en eliminar aquellas acciones consideradas triviales, de modo que no se ejecuten y no generen ruido. No obstante, esta opción presenta varios inconvenientes. El primero es el riesgo de una ejecución incorrecta de las operaciones, ya que muchas de estas acciones se realizan con el objetivo de recopilar información necesaria para la ejecución de acciones posteriores dentro de la misma operación. Otro problema es que no todos los entornos son iguales, por lo que pueden existir casos en los que acciones aparentemente triviales, como obtener el contenido del fichero `/etc/shadow`, representen una vulnerabilidad real.

La segunda opción es utilizar una lista de autorización personalizable para cada cliente. Esta aproximación es más realista y escalable, ya que permite ejecutar la operación de forma normal y, posteriormente, al momento de generar el informe y los tickets, consultar la lista de autorización. Este enfoque permite personalizar para cada cliente qué acciones deben considerarse en el informe y cuáles no, evitando así la sobrecarga del equipo con tickets de falsos positivos. Debido a la forma de trabajo de la empresa, se ha optado por utilizar un repositorio de GitLab para gestionar estas listas blancas.

Este enfoque se basa en la idea de la mejora continua del sistema; es decir, al principio, durante las primeras ejecuciones del sistema, se generarán tickets con falsos

positivos, pero con el tiempo y tras sucesivas operaciones y refinamientos, solo se generarán tickets para aquellas acciones cuyo resultado represente un problema real de seguridad.

En la Figura 5.1 se muestra el diagrama completo del proceso de mejora continua con el servicio Python.

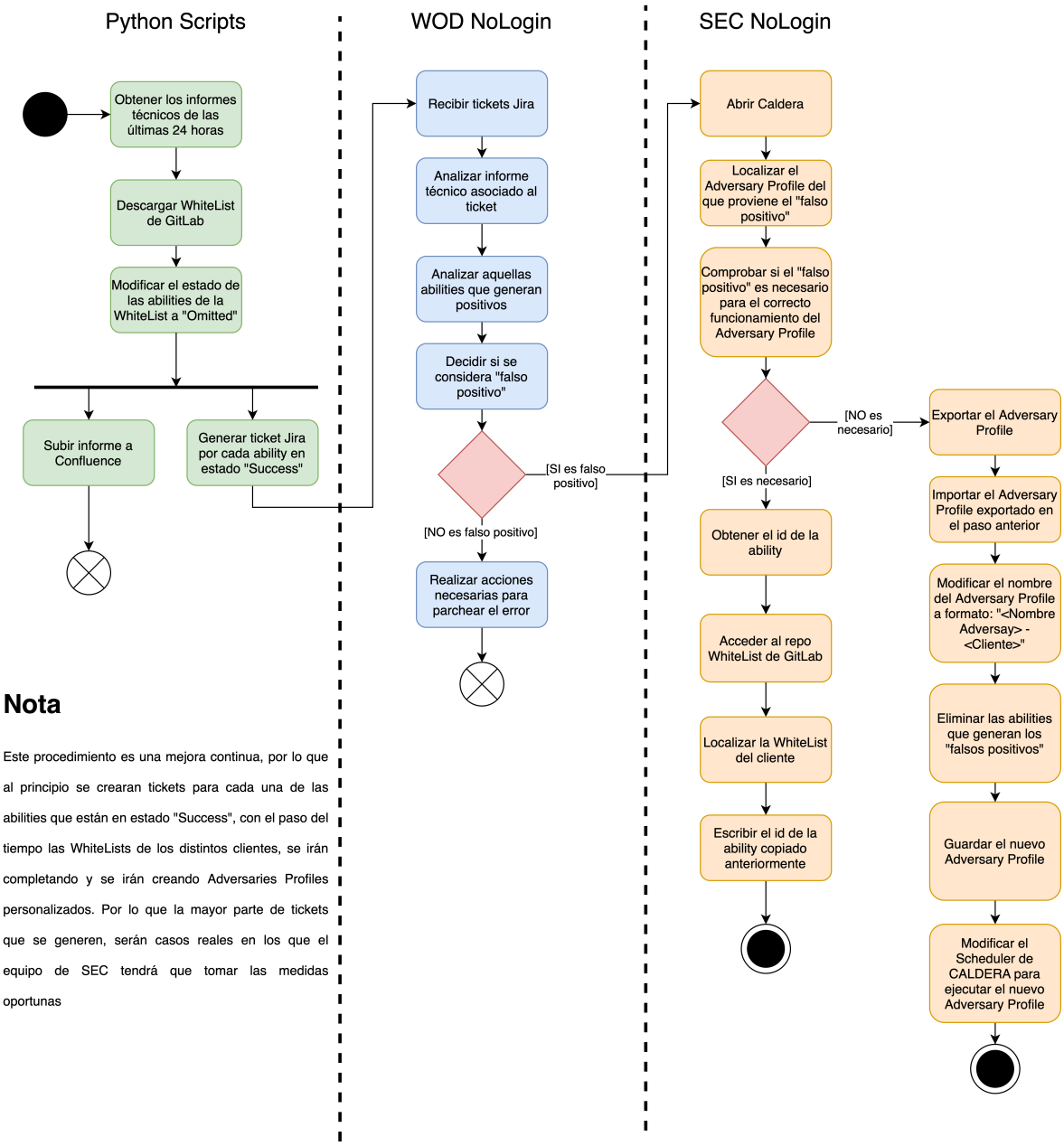

Figura 5.1: Diagrama procedimiento de mejora continua

5.3. Implementación

Para la implementación del sistema completo se han utilizado las tecnologías comentadas en la Sección 4.1. Para conectar todas estas tecnologías, se ha utilizado un

servicio Python. Pese a que a primera vista puede parecer que el núcleo del sistema es *MITRE Caldera*, el componente encargado de conectar todo y automatizar el trabajo del equipo técnico es dicho servicio Python.

Para relacionar todos los componentes y crear un flujo de ejecución completo, se han utilizado las API de los distintos softwares, como por ejemplo Caldera, Jira o Confluence. Para la comunicación del servicio con estas tecnologías, se ha utilizado la librería *AIOHTTP* para realizar las peticiones HTTP y *Asyncio* para implementar la concurrencia, mejorando así su eficiencia.

En la Figura 5.2 se muestra el diagrama de clases UML del servicio Python. Este servicio consta de nueve clases principales en las cuales cada una tiene una responsabilidad específica. La clase *Main* es la clase principal que se encarga de orquestar todo el servicio. También está la clase *Operation* que se encarga de obtener la información de las operaciones ejecutadas en MITRE Caldera. La clase *JiraReport* es la encargada de crear los tickets en el sistema Jira por cada habilidad en estado exitoso dentro de una operación. *CreatePage* es la clase utilizada para crear la página con el reporte en Confluence. La extracción de información y creación del reporte se realiza en la clase *CreateReport* con la ayuda de *GenerateHtml*, que se encarga de crear el código *html*. Además, el reporte cuenta con gráficas creadas por la clase *Statistics*.

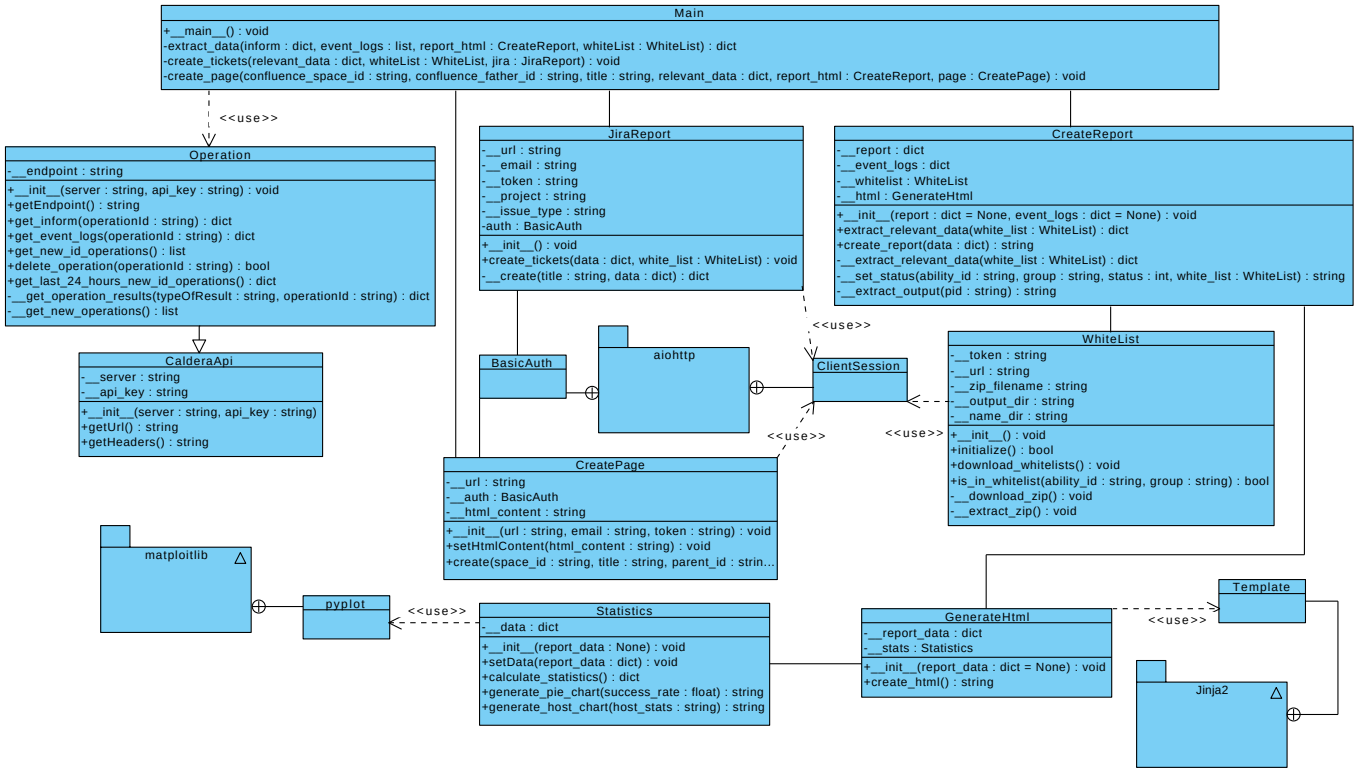

Figura 5.2: Diagrama de clases servicio Python

En la Figura 5.3 se muestra el diagrama UML del flujo principal del servicio Python, el cual comienza instanciando los objetos necesarios y obteniendo los ID de las nuevas operaciones disponibles en MITRE Caldera. A continuación, por cada ID, se extrae la información relevante de la operación. Para ello, primero se recupera el reporte y el registro de eventos generado por MITRE Caldera, con el fin de obtener los datos necesarios.

Una vez extraída la información, se comprueba el número de clientes a los que ha afectado la operación. En caso de ser uno, se establece el título correspondiente; en caso de ser más de uno, se agrupan los nombres de los clientes y se define un título conjunto. Finalmente, se crean los tickets en Jira, la página en Confluence y se elimina la operación del servidor MITRE Caldera.

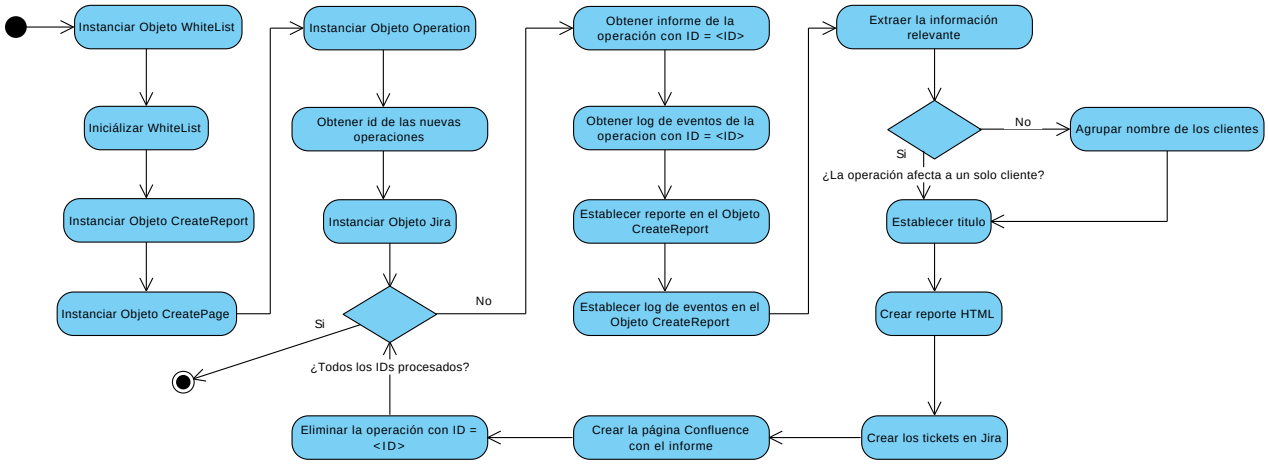

Figura 5.3: Diagrama del flujo principal del servicio Python

Por otro lado, para garantizar la persistencia del agente de Caldera en los distintos equipos de los clientes, se han desarrollado scripts específicos. En el caso de sistemas Windows, se ha implementado un script en PowerShell que simula que el ejecutable del agente es un servicio del sistema, además de crear una regla en el firewall interno que permite la comunicación con el servidor de Caldera. Para sistemas Linux y macOS, se ha optado por un script en Bash que ejecuta el agente como un servicio del sistema. Ambos scripts se encuentran disponibles en el repositorio GitHub asociado al proyecto.

Asimismo, aunque no forma parte directamente del sistema principal, éste está preparado para integrarse con servicios de monitorización. Esto permite que la empresa supervise el estado del servidor donde se ejecuta el servicio Python y reciba alertas en caso de fallos. Para ello, el servicio genera archivos de registro en cada ejecución, los cuales se almacenan en un volumen compartido de Docker. Esto garantiza la

persistencia de los registros incluso tras eliminar el contenedor, ya que el servicio Python se ejecuta en un contenedor efímero que se destruye una vez finalizado el proceso.

Como se ha mencionado anteriormente, el servicio Python está preparado para recolectar información y generar informes y tickets sobre operaciones que hayan sido ejecutadas de forma simultánea en distintos clientes. Por ejemplo, es posible ejecutar el perfil de adversario 1 en *cliente1* y *cliente2* al mismo tiempo. No obstante, por convención interna, cada operación debe realizarse sobre un único cliente. A pesar de ello, el servicio se ha diseñado para soportar este tipo de escenarios sin comprometer su funcionamiento. En estos casos, simplemente se genera un informe grupal en el que se incluyen todos los clientes afectados.

Por motivos de eficiencia, se ha implementado la eliminación automática de las operaciones en MITRE Caldera tras la generación de los informes y la creación de los tickets correspondientes. Esta medida se debe a que MITRE Caldera almacena las operaciones en memoria RAM y no en disco, lo que podría provocar un uso innecesario de recursos si no se eliminan las operaciones procesadas. Para implementar esta funcionalidad, se ha hecho uso de la API de Caldera.

En la Figura 5.4 se presenta un diagrama que ilustra el funcionamiento completo del sistema, desde la selección y programación de operaciones por parte del equipo técnico, hasta la ejecución en Caldera y la posterior generación de informes y tickets para su análisis y resolución.

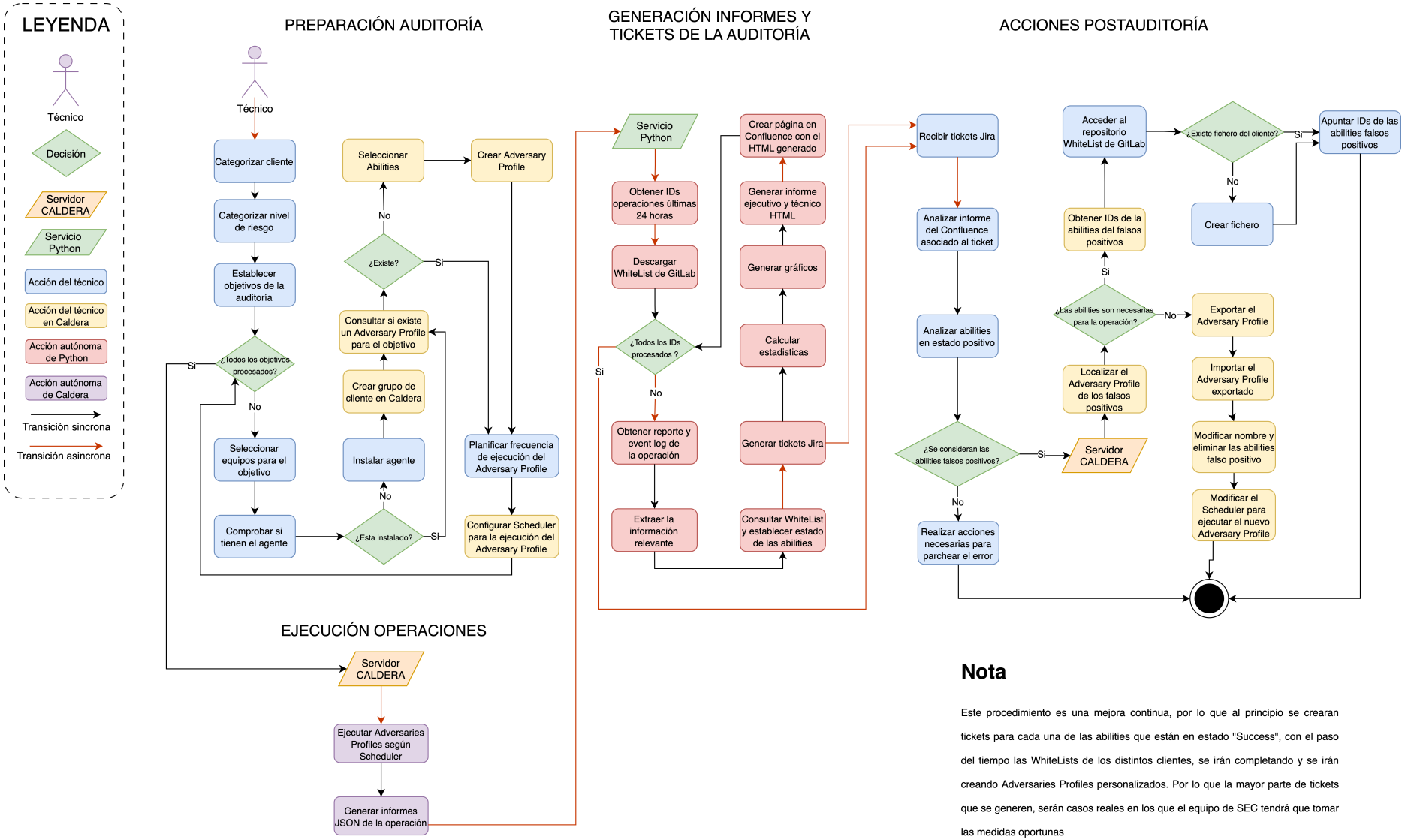

Figura 5.4: Diagrama completo del sistema

5.4. Código fuente y repositorio público

Todo el código desarrollado en este trabajo se encuentra disponible públicamente en el siguiente repositorio de GitHub:

`https://github.com/Garicore01/MITRE-Caldera-Atlassian-Reporting-Integration.git`

Este repositorio ha sido publicado bajo la licencia GNU General Public License v3.0 (GPLv3)[1]. Esta licencia permite la modificación o redistribución del código, conservando los mismos derechos para todos los usuarios.

[1]La licencia GPLv3 garantiza que los usuarios pueden ejecutar, estudiar, compartir y modificar el software, siempre que las versiones derivadas mantengan la misma licencia.

Capítulo 6

Caso de estudio

En este capítulo se expone una demostración del funcionamiento del sistema en la que se va a ejecutar una operación en un cliente que tiene dos equipos y una habilidad en su lista de autorización. El objetivo es mostrar el funcionamiento de la parte más autónoma del sistema: (a) la ejecución de operaciones programadas; y (b) su recolección y evaluación con la lista de autorización para la posterior creación de informes y tickets. Para ahorrar tiempo, la prueba comienza con un cliente en el que ya se han establecido los objetivos e instalado los agentes en dos equipos, uno Windows y otro Linux. En la Figura 6.1 se muestran los agentes disponibles en Caldera. Se puede observar que están los dos equipos Windows y Linux del *client2* y otros dos equipos más. El primero es un ejemplo de eliminación del agente desde Caldera y el segundo es un equipo de otro cliente, en este caso *red*.

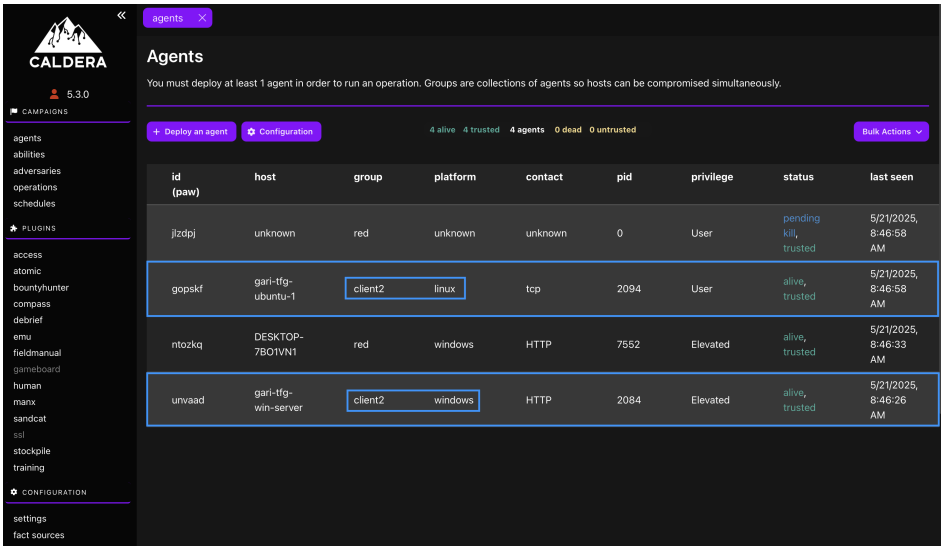

Figura 6.1: Agentes instalados en Caldera

6.1. Configuración en Caldera

Primero, hay que programar el programador de tareas para ejecutar el perfil de adversario seleccionado con la frecuencia acordada en los objetivos; en este caso, todos los lunes a las 2:00 a.m. Segundo, es importante seleccionar el cliente en el cual queremos que se ejecute la operación; en este caso, es *client2*. Por convención, se ha establecido en la empresa que cada operación debe hacer referencia a un único cliente. El siguiente paso es seleccionar el enlace asociado a hechos, en caso de que el perfil de adversario necesite información adicional para ejecutarse correctamente, como, por ejemplo, la dirección IP del controlador de dominio. El planificador por defecto es *atomic*, el cual ejecuta cada acción de la operación en orden secuencial. Por último, MITRE Caldera permite ofuscar comandos para intentar engañar y evadir los antivirus y es importante seleccionar la opción *Auto close operation*, ya que, si no se selecciona, MITRE Caldera no finalizará la operación automáticamente, incluso aunque esta no tenga nada más para ejecutar.

En la Figura 6.2 se muestra la configuración utilizada para programar la operación descrita anteriormente.

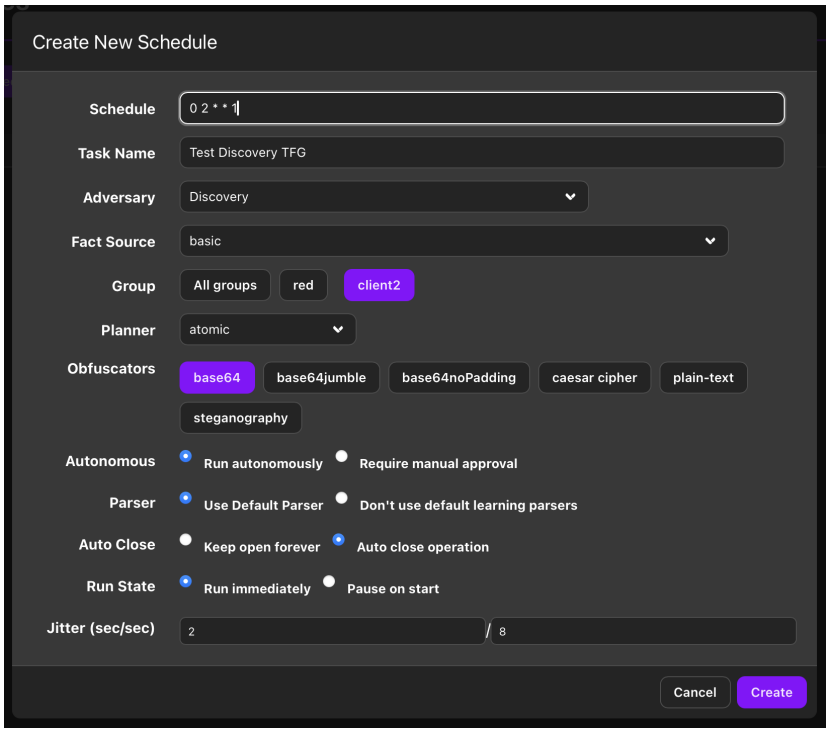

Figura 6.2: Ejemplo de programación en MITRE Caldera

6.2. Preparación del servicio Python

El servicio Python está programado con la herramienta *cron* para ejecutarse cada día a las 4:00 a.m. Esta hora ha sido seleccionada por la empresa, ya que presenta muy poca actividad.

A las 4:00 a.m., el servicio Python se ejecutará en un contenedor Docker, de forma que primero se creará el contenedor y, posteriormente, se lanzará un contenedor de un solo uso; es decir, que al finalizar su ejecución, el contenedor se eliminará. Tras la ejecución del servicio, este habrá generado un archivo de registro en un volumen compartido, por lo que es persistente al borrado del contenedor. En caso de que algo falle, se verá reflejado en el archivo de registro, de forma que herramientas como, por ejemplo, los SIEM puedan capturar ese error y notificarlo al equipo técnico.

6.3. Ejecución del sistema

Llegada la hora de ejecución programada en MITRE Caldera, se ejecutará la operación, dejando un registro que puede ser consultado desde la interfaz gráfica, como se muestra en la Figura 6.3.

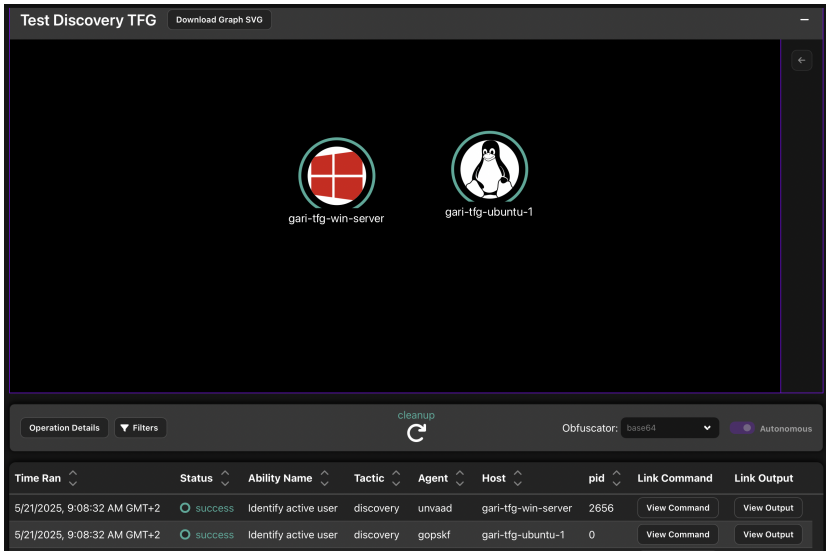

Figura 6.3: Operación Test Discovery TFG

Tras finalizar la ejecución de la operación, el sistema esperará a que se ejecute el servicio Python a las 4:00 a.m. Una vez llegada esa hora, el servicio recopila la información de las operaciones disponibles en MITRE Caldera y comienza con el procesamiento individual de cada una de ellas.

Primero, se extrae la información relevante, estableciendo un estado en función de si la acción ha sido exitosa o no, y si dicha acción aparece en la lista de autorización

del cliente, en cuyo caso se le asigna el estado omitido. Después, se crean los gráficos para la parte del informe ejecutivo y el contenido HTML con el informe completo. El siguiente paso es la creación de tickets en Jira por cada acción en estado exitoso. Una vez creados los tickets, se procede a la subida del informe a Confluence y, una vez finalizada esta acción, se elimina la operación del servidor MITRE Caldera.

A continuación se muestra un ejemplo del informe generado por el servicio Python. La Figura 6.4 muestra la sección ejecutiva del informe. Esta sección se denomina "ejecutiva" porque no incluye datos técnicos, sino únicamente información visual y fácil de interpretar para una persona no técnica, como puede ser un cliente. El color rojo representa que la operación ha logrado ejecutar todas las acciones que se habían configurado, lo cual, en principio, es un indicio de una mala configuración de seguridad informática (es decir, se han encontrado vulnerabilidades).

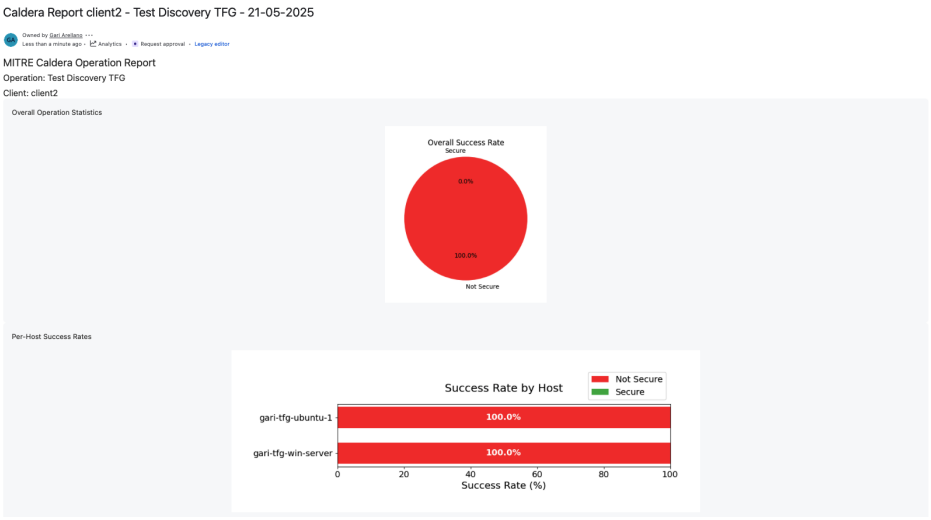

Figura 6.4: Ejemplo de informe ejecutivo

Por otro lado, la Figura 6.5 muestra el informe técnico, diseñado para ser leído por el equipo técnico encargado de la seguridad del cliente examinado. Este informe proporciona la información necesaria para que dicho equipo pueda analizar qué se ha ejecutado, con qué intención y cuál ha sido el resultado de cada acción.

Además, la Figura 6.6 muestra cómo es un ticket de Jira generado por el servicio Python. El ticket se compone de una breve descripción informando que se ha encontrado una vulnerabilidad y una tabla con los campos esenciales, como por ejemplo, el nombre del cliente, la máquina, el nombre de la acción ejecutada, su ID en Caldera, su descripción, el comando que se ha ejecutado en texto plano y la salida recibida.

Hosts Involved

Paw	Host	Platform	Privilege	IP Address
unvaad	gari-tfg-win-server	windows	Elevated	10.111.8.250
gopskf	gari-tfg-ubuntu-1	linux	User	Unknown

Executed Steps

Host	Step Name	Technique ID	Command	PlainText Command	Platform	Description	Output	Status
gari-tfg-win-server	Identify active user	T1033	powershell -Enc JABlAG4AdgA6AHUAcwBlAHIAbgBhAG0AZQA=	$env:username	windows	Find user running agent	GARI-TFG-WIN-SE$	Omitted
gari-tfg-win-server	Identify local users	T1087.001	powershell -Enc RwBlAHQALQBXAG0AaQBPAGIAagBlAGMAdAAgAC0AQwBsAGEAcwBzACAAVwBpAG4AMwAyAF8AVQBzAGUAcgBBAGMAYwBvAHUAbgB0AA==	Get-WmiObject -Class Win32_UserAccount	windows	Identify all local users	AccountType : 512Caption : GARI-TFG-WIN-SE\AdministradorDomain : GARI-TFG-WIN-SESID : S-1-5-21-3200826031-3971186199-900815489-500FullName : Name : AdministradorAccountType : 512Caption : GARI-TFG-WIN-SE\DefaultAccountDomain : GARI-TFG-WIN-SESID : S-1-5-21-3200826031-3971186199-900815489-503FullName : Name : DefaultAccountAccountType : 512Caption : GARI-TFG-WIN-SE\InvitadoDomain : GARI-TFG-WIN-SESID : S-1-5-21-3200826031-3971186199-900815489-501FullName : Name : Invitado	Success
gari-tfg-ubuntu-1	Identify active user	T1033	eval "$(echo d2hvYW1p | base64 --decode)"	whoami	linux	Find user running agent	caldera-ubuntu	Omitted
gari-tfg-ubuntu-1	Find local users	T1087.001	eval "$(echo Y3V0IC1kOiAtZjEgL2V0Yy9wYXNzd2QgfCBncmVwIC12ICdfJyBBlGdyZXAgLXYgJyMn | base64 --decode)"	cut -d: -f1 /etc/passwd | grep -v '_' | grep -v '#'	linux	Get a list of all local users	caldera-ubuntu	Success

Figura 6.5: Ejemplo de informe técnico

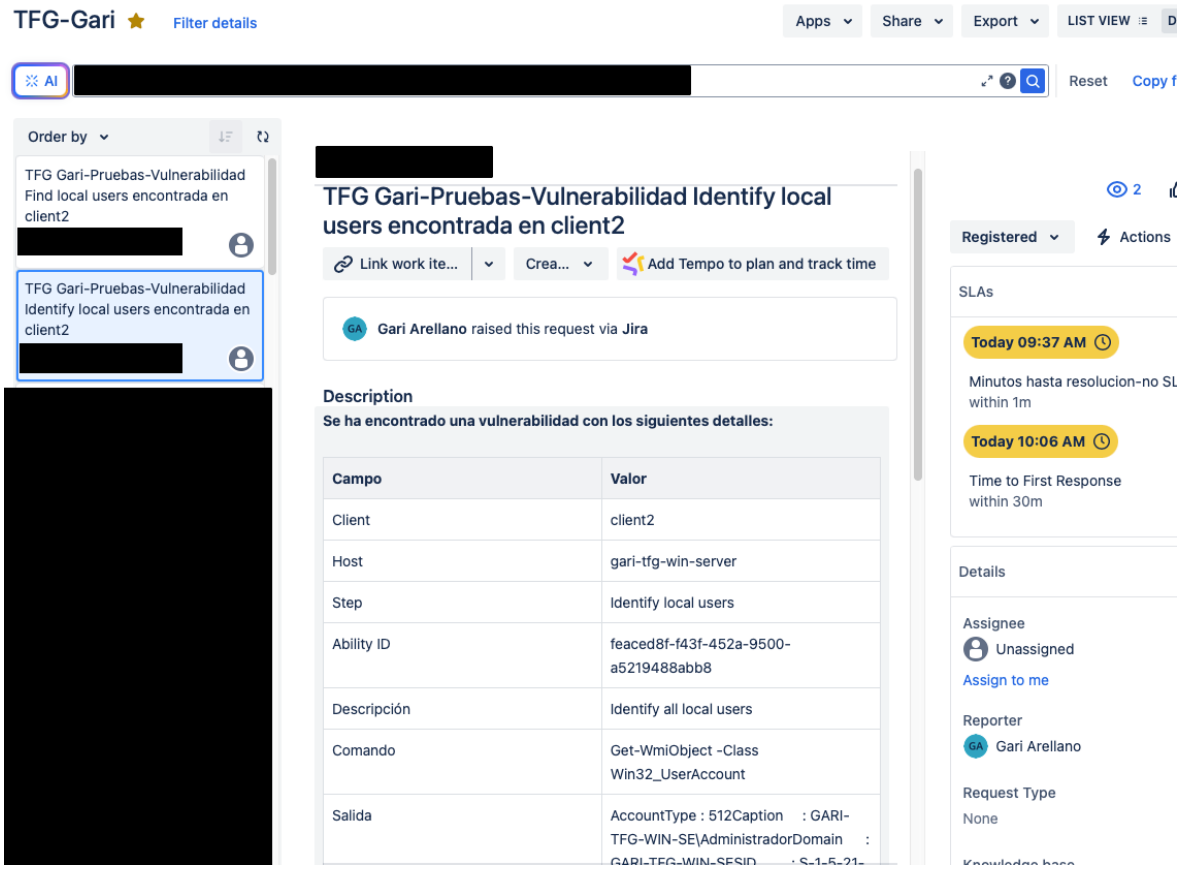

Figura 6.6: Ticket en Jira generado por el servicio Python

Por último, el equipo técnico es quien debe revisar los tickets y el informe, y decidir si alguna de las acciones es permisible. En caso de serlo, deberán copiar el ID del ticket y añadirlo a la lista de autorización del cliente. En caso contrario, deberán llevar a cabo las medidas pertinentes para solucionar la vulnerabilidad detectada.

De esta forma, un equipo técnico es capaz de realizar una auditoría a un entorno, consumiendo menos recursos y teniendo mayor capacidad para realizar otro tipo de

tareas.

Capítulo 7

Conclusiones

Este capítulo tiene como objetivo exponer las conclusiones generales obtenidas tras la realización del proyecto, así como explicar qué posibles mejoras se podrían implementar en el futuro.

7.1. Conclusiones generales

En un entorno en el que los ciberataques están impulsados por la IA las organizaciones deben adaptarse y adoptar enfoques más automatizados y eficientes para la realización de auditorías, reduciendo la cantidad de recursos y optimizando los procesos internos.

El trabajo realizado ha conseguido cumplir con el objetivo planteado al iniciar este proyecto, desarrollar un sistema que permita a un SOC automatizar las tareas de realización de auditorías a clientes. El sistema desarrollado se encarga de ejecutar operaciones y generar tanto los informes como los tickets asociados a cada una de ellas.

Desde una perspectiva tanto técnica como empresarial, la solución es escalable vertical y horizontalmente, lo que permite aumentar el número de clientes y operaciones sin afectar significativamente al rendimiento. Aunque existe una dependencia con las herramientas internas de la empresa como Jira o Confluence, la adaptación a otros sistemas *ITSM* (*Information Technology Service Management*; en español, sistema de gestión de servicios informáticos. Estos sistemas diseñados para gestionar los servicios informáticos de una organización) sería sencilla, bastando con modificar algunos componentes concretos del servicio Python.

Una de las ventajas que ofrece este sistema es su independencia del entorno del cliente, lo cual facilita considerablemente su despliegue y ejecución. En entornos que ya utilizan Jira y Confluence, se puede afirmar que el sistema no requiere cambios para ser implementado en otra organización, independientemente del entorno de virtualización.

7.2. Trabajo futuro

El sistema genera un informe con una parte ejecutiva y una parte técnica. Del mismo modo, sería muy interesante añadir IA para la generación de descripciones personalizadas basadas en el historial de operaciones ejecutadas en ese cliente. Esta descripción estaría destinada tanto a la sección ejecutiva como a la técnica del informe, de forma que explique qué se ha ejecutado en la operación, qué resultados se han obtenido y cuál es el estado actual del cliente en comparación con estados anteriormente registrados.

Bibliografía

[1] Andy Applebaum Doug Miller. CALDERA, Automating Adversary Emulation. https://www.blackhat.com/docs/eu-17/materials/eu-17-Miller-CALDERA-Automating-Adversary-Emulation.pdf.

[2] Michael J. Cloppert Eric M. Hutchins and Ph.D Rohan M. Amin. Intelligence-Driven Computer Network Defense Informed by Analysis of Adversary Campaigns and Intrusion Kill Chains. https://www.lockheedmartin.com/content/dam/lockheed-martin/rms/documents/cyber/LM-White-Paper-Intel-Driven-Defense.pdf.

[3] MITRE Corporation. MITRE ATT&CK Framework. https://attack.mitre.org.

[4] Krishnashree Achuthan, Sasangan Ramanathan, Sethuraman Srinivas, and Raghu Raman. Advancing cybersecurity and privacy with artificial intelligence: current trends and future research directions. *Frontiers in Big Data*, 7, 2024. https://www.frontiersin.org/articles/10.3389/fdata.2024.1497535/full.

[5] Alvaro Garrido. El impacto de la inteligencia artificial en la ciberseguridad. https://observatoriociber.org/el-impacto-de-la-inteligencia-artificial-en-la-ciberseguridad/?utm_source=chatgpt.com.

[6] Diego A. Arcentales Fernández and Xiomara Caycedo Casas. Auditoría informática: un enfoque efectivo. *Dominio de las Ciencias*, 2017. https://dialnet.unirioja.es/servlet/articulo?codigo=6102836.

[7] MITRE Caldera Documentation. https://caldera.readthedocs.io/en/latest/.

[8] TNECOM Canarias. Preparación para Incidentes Reales: La Importancia de las Simulaciones de Ataques. https://www.tnecomcanarias.es/blog/preparacin-para-incidentes-reales-la-importancia-de-las-simulaciones-de-ataques.

[9] IBM. ¿Qué es un ciberataque? https://www.ibm.com/es-es/topics/cyber-attack.

[10] Fortinet. ¿Qué es el marco MITRE ATT&CK ? https://www.fortinet.com/lat/resources/cyberglossary/mitre-attck.

[11] Ciber 4 All Team. MITRE ATTCK: ¿Qué tácticas y técnicas emplean los ciberdelincuentes?, 2023. https://www.tarlogic.com/es/blog/mitre-attck/.

[12] Documentación Asyncio. https://docs.python.org/es/3/library/asyncio.html.

[13] Documentación AIOHTTP. https://docs.aiohttp.org/en/stable/client_quickstart.html.

[14] Documentación Confluence. https://www.atlassian.com/es/software/confluence/resources/guides/get-started/overview#hosting-options.

[15] Documentación Jira. https://www.atlassian.com/es/software/jira/guides/getting-started/introduction#what-is-jira-software.

[16] Proxmox. https://isnum.com/glosario-ciberseguridad/proxmox/.

[17] Michael A. Schwarz. Linux Job Scheduling. *Linux Journal*, 2000. https://www.linuxjournal.com/article/4087.

[18] notmarshmllow. Installation Errors via Docker. https://github.com/mitre/caldera/issues/3137.

[19] Iker Landajuela. Técnica de priorización MOSCOW. https://soka.gitlab.io/blog/post/2019-06-05-tecnicas-metodologias-priorizacion/.

[20] BBVA. La IA en los dos lados de la ciberseguridad: aliada y amenaza en el mundo digital, 2024. https://www.bbva.com/es/innovacion/la-ia-en-los-dos-lados-de-la-ciberseguridad-aliada-y-amenaza-en-el-mundo-digital/.

[21] TECNOZERO. Cortex XDR Palo Alto. https://www.tecnozero.com/palo-alto-networks/cortex/#:~:text=Cortex%20XDR%20es,los%20usuarios%20y%20activos%20digitales.

[22] BlackMantiSecurity. Guía Básica de Adversary Emulation con Caldera Mitre. https://www.linkedin.com/pulse/gua-bsica-de-adversary-emulation-con-caldera-mitre-qthrf/.

[23] Arlethv. Arquitectura Docker, 2024. https://forum.huawei.com/enterprise/intl/es/thread/arquitectura-docker/807247511067901952?blogId=807247511067901952.

[24] Eva Dafne Alcala Lopez. Emulación de técnicas ofensivas con Mitre Caldera. Trabajo fin de grado, Universidad Politécnica de Valencia, 2022. https://riunet.upv.es/entities/publication/7db1a673-afc8-4768-a3a6-cdf6e229e668.

Anexos

Anexos A

A.1. Tiempo invertido y diagrama de Gantt

La Tabla A.1 muestra las horas invertidas en cada tarea del proyecto. Por otro lado, la Figura A.1 muestra el diagrama de Gantt con la división del trabajo. Primero se comienza con una etapa de investigación acerca de la simulación de ataques y la herramienta MITRE Caldera. Después comienza la etapa de diseño, centrada en diseñar el sistema. Seguidamente, comienza la etapa de implementación del sistema. Finalmente, la etapa de memoria está centrada en realizar la memoria del trabajo.

Tarea	**Horas**
Investigación y realización de pruebas	30
Diseño	20
Implementación	180
Memoria	85
Total	**315**

Tabla A.1: Distribución de horas por tarea

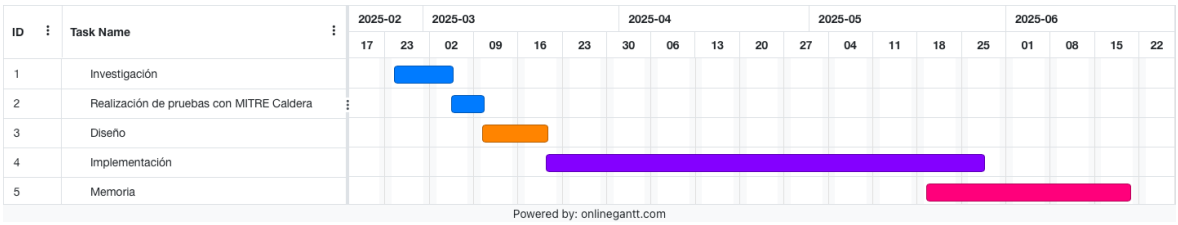

Figura A.1: Diagrama de Gantt

A.2. Categorización de clientes

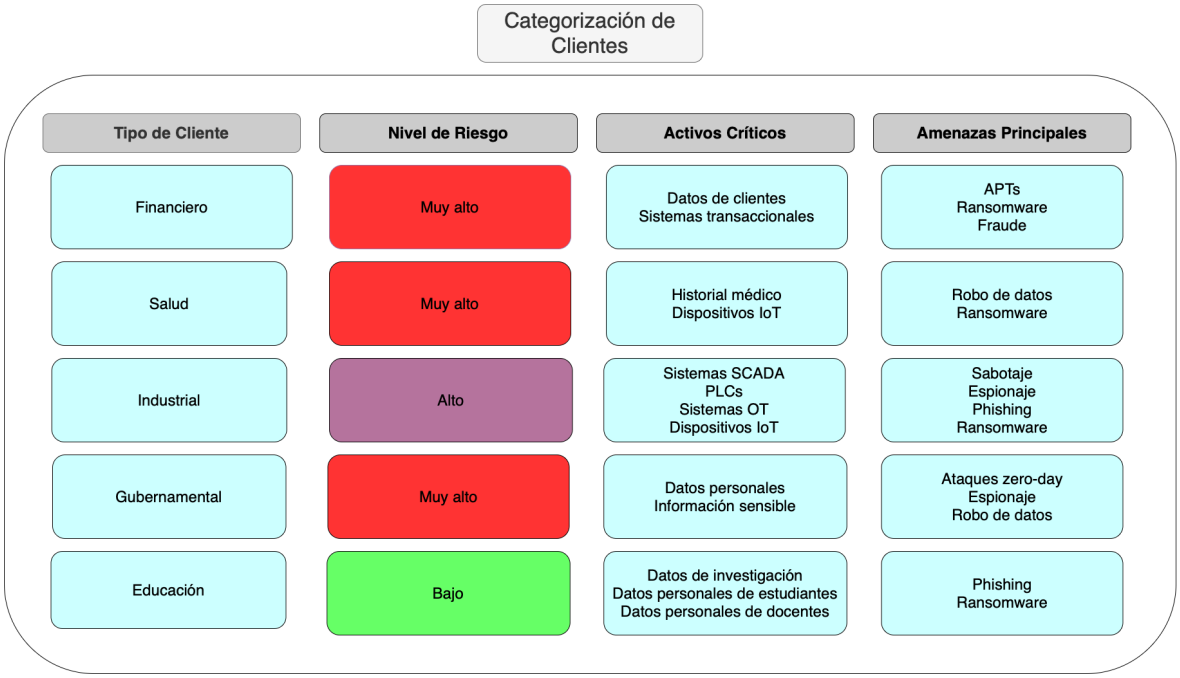

Figura A.2: Categorización de clientes

A.3. Categorización de riesgos

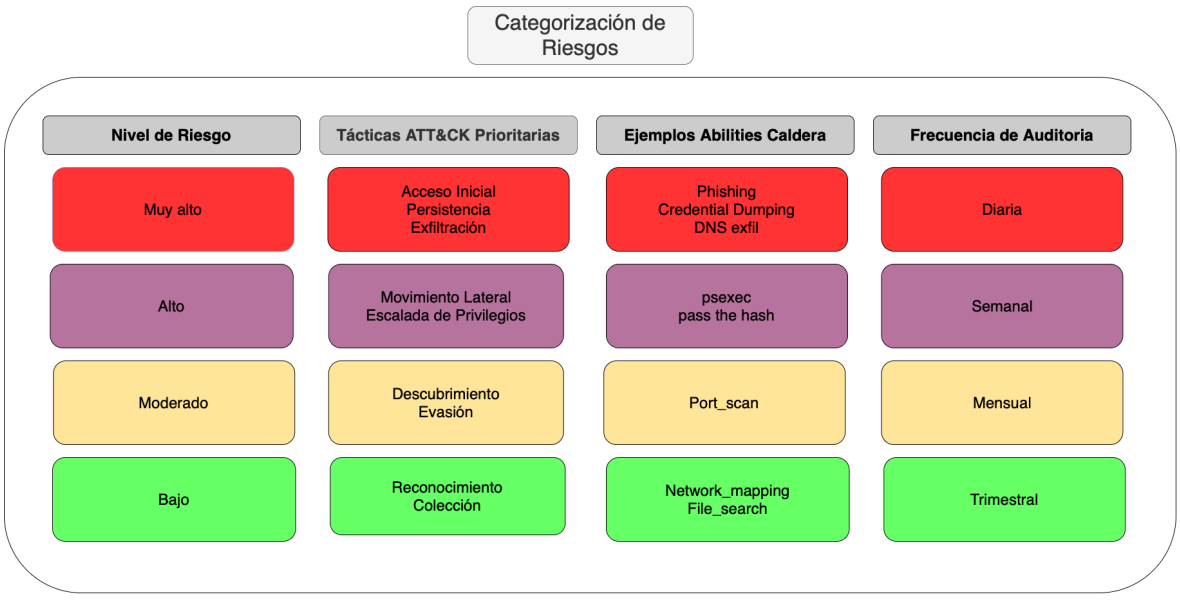

Figura A.3: Categorización de riesgos

A.4. Traza del registro creado por el Servicio Python

```
2025-05-21 07:36:38,682 - test_report - INFO - Starting test report execution
2025-05-21 07:36:38,683 - test_report - INFO - Initializing WhiteList class
2025-05-21 07:36:38,684 - test_report - INFO - WhiteList initialized successfully
2025-05-21 07:36:38,685 - test_report - INFO - Initializing report creation
2025-05-21 07:36:38,685 - test_report - INFO - Starting whitelist download process
2025-05-21 07:36:38,685 - test_report - INFO - Starting whitelist repository download
2025-05-21 07:36:39,270 - test_report - INFO - Whitelist repository downloaded successfully to whitelist_repo.zip
2025-05-21 07:36:39,271 - test_report - INFO - Extracting whitelist repository to repository
2025-05-21 07:36:39,275 - test_report - INFO - Whitelist repository extracted successfully
2025-05-21 07:36:39,275 - test_report - INFO - Whitelist download process completed successfully
2025-05-21 07:36:39,275 - test_report - INFO - Whitelists downloaded successfully
2025-05-21 07:36:39,275 - test_report - INFO - Initializing CreateReport class
2025-05-21 07:36:39,276 - test_report - INFO - Initializing WhiteList class
2025-05-21 07:36:39,278 - test_report - INFO - WhiteList initialized successfully
2025-05-21 07:36:39,278 - test_report - INFO - CreateReport initialized successfully
2025-05-21 07:36:39,279 - test_report - INFO - Initializing CreatePage class
2025-05-21 07:36:39,279 - test_report - INFO - CreatePage initialized successfully
2025-05-21 07:36:39,279 - test_report - INFO - Initializing Operation class
2025-05-21 07:36:39,279 - test_report - INFO - Operation endpoint set to: /operations
2025-05-21 07:36:39,327 - test_report - INFO - Successfully retrieved 1 operations
2025-05-21 07:36:39,328 - test_report - INFO - Initializing JiraReport class
2025-05-21 07:36:39,328 - test_report - INFO - JiraReport initialized successfully
2025-05-21 07:36:39,329 - test_report - INFO - Processing operation: Test Discovery TFG
(ID: b090d8e0-e437-4b1b-b3fc-b379b986cd58)
2025-05-21 07:36:39,329 - test_report - INFO - Requesting operation report for ID:
b090d8e0-e437-4b1b-b3fc-b379b986cd58
2025-05-21 07:36:39,406 - test_report - INFO - Successfully retrieved report for operation
b090d8e0-e437-4b1b-b3fc-b379b986cd58
2025-05-21 07:36:39,407 - test_report - INFO - Requesting event logs for operation ID:
b090d8e0-e437-4b1b-b3fc-b379b986cd58
2025-05-21 07:36:39,416 - test_report - INFO - Successfully retrieved event logs for operation
b090d8e0-e437-4b1b-b3fc-b379b986cd58
2025-05-21 07:36:39,416 - test_report - INFO - Setting new report data
2025-05-21 07:36:39,417 - test_report - INFO - Report data updated successfully
2025-05-21 07:36:39,417 - test_report - INFO - Setting new event logs
2025-05-21 07:36:39,417 - test_report - INFO - Event logs updated successfully
2025-05-21 07:36:39,417 - test_report - INFO - Extracting relevant data from report
2025-05-21 07:36:39,417 - test_report - INFO - Setting status of the step
2025-05-21 07:36:39,418 - test_report - INFO - Ability c0da588f-79f0-4263-8998-7496b1a40596 is in whitelist for
group client2
2025-05-21 07:36:39,418 - test_report - INFO - Setting status of the step
2025-05-21 07:36:39,419 - test_report - INFO - Setting status of the step
2025-05-21 07:36:39,419 - test_report - INFO - Ability c0da588f-79f0-4263-8998-7496b1a40596 is in whitelist for
group client2
2025-05-21 07:36:39,420 - test_report - INFO - Setting status of the step
2025-05-21 07:36:39,420 - test_report - INFO - Data extraction completed successfully
2025-05-21 07:36:39,420 - test_report - INFO - Operation group: client2
2025-05-21 07:36:39,420 - test_report - INFO - Starting tickets creation
2025-05-21 07:36:39,421 - test_report - INFO - Starting ticket creation for title: TFG Gari-Pruebas-Vulnerabilidad
Identify local
users encontrada en client2
2025-05-21 07:36:39,421 - test_report - INFO - Creating Jira ticket with title: TFG Gari-Pruebas-Vulnerabilidad
Identify local
users encontrada en client2
2025-05-21 07:36:45,756 - test_report - INFO - Jira ticket created successfully: WODNLGSEC-37863
2025-05-21 07:36:45,758 - test_report - INFO - Starting ticket creation for title: TFG Gari-Pruebas-Vulnerabilidad
Find local users
encontrada en client2
2025-05-21 07:36:45,759 - test_report - INFO - Creating Jira ticket with title: TFG Gari-Pruebas-Vulnerabilidad
Find local users
encontrada en client2
2025-05-21 07:36:47,210 - test_report - INFO - Jira ticket created successfully: WODNLGSEC-37864
2025-05-21 07:36:47,211 - test_report - INFO - Ticket creation process completed
2025-05-21 07:36:48,031 - test_report - INFO - Creating new page with title: Caldera Report client2 - Test Discovery
TFG - 21-05-2025
2025-05-21 07:36:48,658 - test_report - INFO - Page 'Caldera Report client2 - Test Discovery TFG - 21-05-2025'
created successfully
2025-05-21 07:36:49,346 - test_report - INFO - Confluence page created for operation Test Discovery TFG
2025-05-21 07:36:49,347 - test_report - INFO - Starting to delete operation Test Discovery TFG
(ID: b090d8e0-e437-4b1b-b3fc-b379b986cd58)
2025-05-21 07:36:49,356 - test_report - INFO - Successfully deleted operation b090d8e0-e437-4b1b-b3fc-b379b986cd58
2025-05-21 07:36:49,357 - test_report - INFO - Operation Test Discovery TFG (ID: b090d8e0-e437-4b1b-b3fc-b379b986cd
58) deleted
successfully in Caldera
2025-05-21 07:36:49,357 - test_report - INFO - Test report execution completed successfully
```